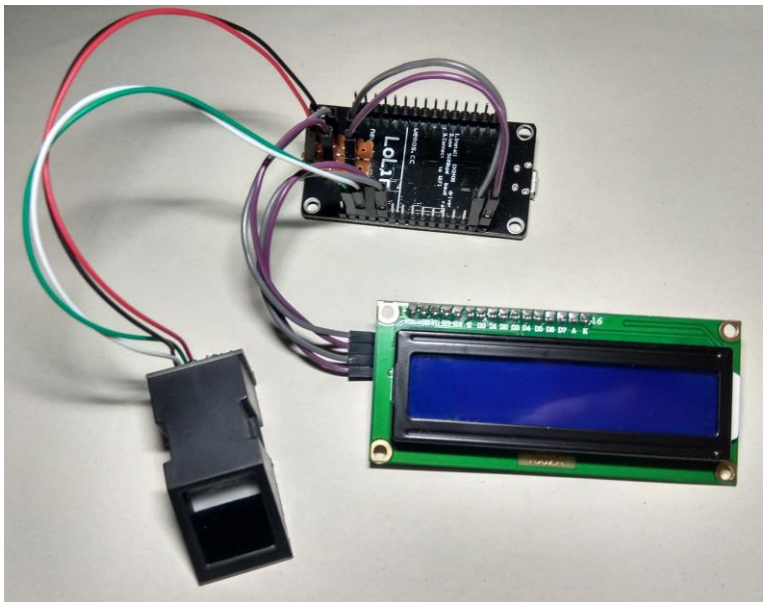
SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

DOCHÁZKOVÝ BIOMETRICKÝ SYSTÉM

Patrik Bartoš



Obor:	18-20-M/01 INFORMAČNÍ TECHNOLOGIE se zaměřením na počítačové sítě a programování
Třída:	IT4
Školní rok:	2019/2020

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

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

V Opavě 28. 12. 2019

podpis autora práce

ANOTACE

Cílem projektu je vytvoření docházkového systému s využitím čtečky otisků prstů. Základem projektu je vývojová deska NodeMCU. K desce je připojena čtečka otisků a LC displej. Deska je programována v jazyce Arduino. Další součástí projektu je databáze s otisky. K této databázi je možno přistupovat pomocí webové stránky. Komunikace mezi databází a deskou NodeMCU je uskutečněna protokolem MQTT. Jakožto prostředník MQTT komunikace byl zvolen broker Mosquitto. Hardwarová část má na starosti sejmutí otisku od uživatele a odeslat jej na MQTT broker. Data odeslaná z desky jsou odebírána aplikací napsanou v jazyce Python. Tato aplikace se po spuštění připojí na MQTT broker, naváže komunikaci s databází a následně do ní ukládá přijatá data.

Klíčová slova

NodeMCU; ESP8266; Arduino; Python; MQTT; Mosquitto; broker; otisk; čtečka; databáze

OBSAH

ÚVOD

Jako svůj projekt jsem se rozhodl vytvořit docházkový systém založený na otiscích prstů. Zvolil jsem tak, protože v dnešní době je zabezpečení přes otisky velmi běžné a chtěl jsem nějaký takový systém vytvořit sám. Tento systém by se mohl použít jak pro zabezpečení, tak i například jako databáze studentů ve škole nebo zaměstnanců v práci.

Hlavním cílem projektu je sejmout otisk prstu od uživatele a uložit jej do databáze pod příslušnými údaji. S tímto uloženým otiskem se poté může dále pracovat. Uživatel může k databázi přistupovat pomocí webových stránek, které mu dovolí uložená data smazat nebo upravit.

Dále v této dokumentaci zmiňuji využité technologie a popisuji postupy při vytváření mého projektu. Na začátku popisuji hardware zapojení, použité součástky a proč jsem je zvolil. Také zde píši o problémech, s kterými jsem se v této části setkal. Poté se zmiňuji o programové části projektu. Určuji postup, který jsem při psaní kódu zvolil. Zdůvodňuji použité technologie, i proč jsem se rozhodl některé postupy změnit. V další kapitole obecně představuji využité technologie. V poslední kapitole podrobně píšu o postupu vytváření projektu. V závěru dokumentace projekt hodnotím, také se zde zmiňuji o částech projektu, které se mi nepovedlo dokončit a o jeho možných vylepšeních.

1 HARDWARE ZAPOJENÍ

Prvně jsem pro realizaci projektu využil vývojovou desku Arduino UNO, WiFi modul ESP8266-01 a čtečku otisků ZFM-20. Všechny tyto součástky se ovšem časem ukázaly jako pro můj projekt nepraktické. Nejvíce problémů nastalo u WiFi modulu ESP8266-01. Nakonec jsem usoudil, že by bylo lepší použít vývojovou deskou, která podporuje WiFi nativně.

Dále jsem tedy začal využívat vývojovou desku NodeMCU, která již WiFi modul ESP8266 obsahuje. Po připojení dosud používané čtečky ZFM-20 jsem ovšem narazil na další problém. Deska NodeMCU a tato čtečka měli mezi sebou problémy s komunikací. Ani po dlouhém hledání jsem řešení tohoto problému nenašel a rozhodl jsem se využít čtečku jinou. Konkrétně jsem vybral čtečku DY50, o které jsem nalezl mnohem více informací. S touto čtečkou již žádný problém nenastal.

K projektu jsem poté připojil 16x2 LC displej pro zobrazování informací při běhu programu. Ten jsem k desce připojil pomocí převodníku, který umožňuje komunikaci mezi vývojovou deskou a displejem přes sběrnici I2C.

V konečné podobě se tedy hardwarová část projektu se skládá z vývojové desky NodeMCU, která obsahuje WiFi modul ESP8266. K této desce je připojena čtečka otisků DY50 a 16x2 LC displej s I2C převodníkem.

2 SOFTWARE

Programování jsem rozdělil do několika částí. První byla napsat program pro vývojovou desku. Zahrnuje připojení k WiFi, obstarání funkcí čtečky a LC displeje a nakonec odesílání dat na MQTT broker. Pro vývoj tohoto programu jsem použil prostředí PlatformIO IDE a jazyk Arduino. Původně jsem vůbec nepoužíval protokol MQTT a k databázi jsem přistupoval přímo pomocí knihovny MySQL Connector pro Arduino. Toto řešení se ukázalo jako problematické, hlavně z hlediska bezpečnosti. Z tohoto důvodu jsem se rozhodl používat komunikaci přes MQTT.

Další část zahrnovala navrhnutí a vytvoření databáze a spuštění MQTT brokeru. Databázi jsem navrhnul v programu MySQL Workbench a nahrál na server vytvořený pomocí balíčku XAMPP. Databázi jsem spravoval pomocí nástroje phpMyAdmin. Jakožto MQTT broker jsem zvolil Eclipse Mosquitto.

Dále jsem programoval aplikaci, která odebírá data z MQTT brokeru a ukládá je do databáze. Pro vytvoření aplikace jsem zvolil jazyk Python a využíval jsem modulů Eclipse Paho a SQLAlchemy. Paho nám umožní vytvořit jednoduchý MQTT klient, který můžeme přihlásit k odběru dat přicházející na MQTT broker. Pomocí SQLAlchemy se potom můžeme připojit k vytvořené databázi a ukládat do ní přijatá data.

Nakonec jsem vytvořil jednoduchou webovou stránku, která vypisuje data z databáze do tabulky a dovoluje s nimi manipulovat. K tomu jsem využil jazyk PHP. Webové stránky, stejně jako databáze, běží na serveru vytvořeném pomocí balíčku XAMPP.

3 VYUŽITÉ TECHNOLOGIE

3.1 Hardware

3.1.1 NodeMCU ESP-12E

Vývojová deska NodeMCU ESP-12E tvoří základ projektu. Je založená na mikrokontroleru ESP8266 tvořený společností Espressif Systems. Tento mikrokontroler poskytuje jednoduché připojení k WiFi. Může fungovat jako klient i jako server. NodeMCU má vestavěný USB konektor, který se používá jako zdroj napájení a slouží k nahrání programu na desku. Obsahuje 13 GPIO pinů, z nichž jeden je analogový. Deska pracuje s napětím 3,3 V.

3.1.2 Čtečka otisků DY50

Otisky snímám čtečkou DY50 od společnosti Adafruit. Tato čtečka dokáže uložit 162 otisků do své vlastní paměti, to ovšem není pro tento projekt důležité, jelikož otisky ukládáme do vlastní externí databáze. Dokáže pracovat s napětím 3,3 – 6 V. Pro uložení otisku čtečka požaduje sejmout otisk dvakrát, abychom dostali přesnější obraz otisku. Pomocí knihovny, která je také od společnosti Adafruit, můžeme již uložené otisky z paměti získat v hexadecimální podobě.

3.1.3 LC displej 16x02

Tento displej slouží jako výstup informací popisující průběh programu. Dokáže zobrazit celkem 32 znaků (16 znaků na 2 řádcích). Pro připojení k desce jsem využil převodník I2C pro LC displeje, který sníží počet vodičů z původních 16 na pouhé 4.

3.2 Software

3.2.1 Jazyk Arduino

Programovací jazyk Arduino byl vytvořen přímo pro programování integrovaných obvodů. Je složen z funkcí napsaných v jazyce C a C++, kvůli čemu je programování v tomto jazyce téměř identické programování v jazyce C. V tomto jazyce byla naprogramována použitá deska NodeMCU.

3.2.2 PlatformIO IDE

PlatformIO je open source multiplatformní vývojové prostředí pro programování mikrokontrolerů. V tomto projektu jsem využil verzi pro editor VSCode. Mezi největší výhody tohoto prostředí oproti Arduino IDE patří například našeptávač při psaní kódu, jednoduchá přenositelnost projektu nebo manažér knihoven. Projekty založené v tomto prostředí obsahují konfigurační soubor, ve kterém můžeme upravovat nastavení aplikované pouze na tento projekt.

3.2.3 MariaDB

MariaDB je relační databáze, která je založena na MySQL. Pro vytvoření databáze jsem využil softwarový balíček XAMPP. Pro jednoduchou správu databáze jsem použil nástroj phpMyAdmin, který je v balíčku XAMPP obsažený.

3.2.4 Eclipse Mosquitto

Mosquitto je open source MQTT broker. Je nenáročný a tak vhodný i pro běh na jednoduchých zařízeních. V tomto projektu slouží jako centrální bod pro MQTT komunikaci.

3.2.5 Python 3

Open source interpretovaný programovací jazyk. Programování v tomto jazyce klade velký důraz na produktivitu programátora. V projektu byl použit pro naprogramování aplikace obstarávající MQTT komunikaci a ukládání do databáze.

3.2.6 Webové stránky

Pro vytvoření stránek jsem využil framework Bootstrap. Tento framework mi jejich tvorbu velmi usnadnil, jak z hlediska vzhledu, tak jejich funkčnosti. Na stránkách jsou vypsány data z databáze do HTML tabulky, které se zde dají upravovat nebo smazat. Toho bylo dosaženo pomocí skriptovacího jazyku PHP.

4 ZPŮSOBY ŘEŠENÍ A POSTUPY

4.1 Připojení k WiFi

Prvně byla potřeba připojit se k WiFi. Toto je uskutečněno pomocí knihovny ESP8266Wifi. Pomocí této knihovny se můžeme jednoduše připojit na WiFi využitím funkce begin(), do které jako parametry vložíme SSID a heslo od WiFi, na kterou se chceme připojit. Ověření, zda jsme se úspěšně připojili poté zajišťuje funkce status().

4.2 LC Displej

Funkce pro připojený LC displej jsou obstarány knihovnou LiquidCrystal_I2C. Displej inicializujeme funkcemi init() a backlight(). Místo, na kterém chceme na displeji něco vypsat, zvolíme funkcí setCursor(), kde jako parametry vložíme řádek a pozici na řádku, kde chceme psát. Poté funkcí print() vypíšeme naši zprávu. Pro jednodušší programování jsem si vytvořil jednoduchou funkci printLC(), která toto vypisování urychluje.

```
//Funkce pro zjednodušení výpisu na displej
void printLC(String first, String second, int dl){
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(first);
  lcd.setCursor(0, 1);
  lcd.print(second);
  delay(dl);
}
```

Obrázek č. 1 Funkce pro výpis na displej

4.3 Sejmutí otisku

Základem pro práci se čtečkou otisků je knihovna Adafruit Fingerprint. Nejprve pomocí funkce verifyPassword() ověříme, zda komunikace se čtečkou probíhá správně

4.3.1 Sejmutí nového otisku

Pro sejmutí otisku použijeme funkci getImage(), která čeká, dokud čtečka nesejme platný obraz otisku. Poté funkcí image2Tz() převedeme obrázek na vzor, který obsahuje informace o důležitých rysech otisku. Obě tyto akce provedeme ještě jednou. Čtečka poté oba vzory porovná. Pokud jsou vzory dostatečně podobné, můžeme pomocí funkce createModel() vytvořit model otisku a uložit jej do paměti čtečky funkcí storeModel().

4.3.2 Získání otisku z paměti

Data uložená v paměti čtečky z ní dostaneme tak, že uložený model nahrajeme do bufferu funkcí loadModel(), a poté funkcí getModel() přeneseme data z bufferu na UART. Získáné data v hexadecimální podobě připisuji do String proměnné, abych k těmto datům poté mohl jednoduše připsat údaje o uživatelích, tedy jejich jméno a příjmení. Na tyto data se program uživatele zeptá až po úspěšném sejmutí otisku a jeho následném získání z paměti.

```
String downloadFingerpintTemplate(uint16_t id){
  //Nahrání do bufferu
  uint8_t p = finger.loadModel(id);
  //Kontrola operace
  switch (p){
    case FINGERPRINT_OK:
      break;
    default:
      return "Error";
  }

  //Přenesení na UART
  p = finger.getModel();
  //Kontrola operace
  switch (p) {
    case FINGERPRINT_OK:
      break;
   default:
      return "Error";
  }

  uint32_t starttime = millis();
  int i = 0;
  String otisk = "";
  //Uložení do String proměnné, jednotlivé byty jsou odděleny čárkou
  while ((i < 534) && ((millis() - starttime) < 20000)){
    if (SENSOR.available()){
      otisk += String(SENSOR.read(), HEX);
      otisk += ",";
      i++;
    }
  }
  return otisk;
}
```

Obrázek č. 2 Získání otisku z paměti

4.4 Komunikace s MQTT

Po získání dat byla potřeba je odeslat na MQTT broker. K tomu jsem využil knihovnu PubSubClient. Tato knihovna nám poskytne klienta, který může provádět jednoduchou MQTT komunikaci. Při zapnutí desky si funkcí setServer() nastavíme IP adresu a port našeho MQTT brokeru a s využitím funkce publish() poté můžeme odesílat data na určený topic.

4.5 Uložení do databáze

Aplikace obstarávající uložení do databáze byla vyvinuta v jazyce Python.

4.5.1 Eclipse Paho MQTT client

Pro MQTT operace jsem využil knihovnu Eclipse Paho MQTT client. Tato knihovna nám dovolí připojit se na MQTT broker a odesílat nebo přijímat data. Aplikace se po zapnutí připojí na MQTT broker a začne odebírat topic, na který NodeMCU odesílá data. Po přijetí dat aplikace zkontroluje, zda zpráva obsahuje všechny informace (jméno, příjmení a data o otisku uživatele).

4.5.2 SQLAlchemy

Získané informace poté ukládáme do databáze pomocí knihovny SQLAlchemy. Nejprve vytvoříme Engine, který budeme používat pro navázání komunikace s databází. Poté vytvoříme Session, které přiřadíme vytvořený Engine. V této Session navážeme spojení s databází. Teď už můžeme data ukládat do databáze. Ze získaných dat vytvoříme model, který pomocí Sessionu ukládáme funkcemi add() a commit().

```
#Vytvoříme Engine, proměnná Server je IP adresa našeho serveru
engine = create_engine('mysql+pymysql://User:Password@' + Server + '/attendance')

#Navázání komunikace pomocí Session
Session = sessionmaker(bind = engine)
session = Session()
```

Obrázek č. 3 Navázání komunikace s databází

```
class Fingerprint(Base):
    #Název tabulky v databázi
    __tablename__ = 'fingerprints'

    #Jednotlivé sloupce v tabulce
    id = Column('id', Integer, primary_key=True)
    name = Column('jmeno', String(50), index=True)
    surname = Column('prijmeni', String(50), index=True)
    finger = Column('otisk', Text)

#Uložení do databáze, parametry jsou data získaná z MQTT brokeru
def insert_finger(name, surname, finger):
    fingerprint = Fingerprint(name = name, surname = surname, finger = finger)

    #Samotné uložení do databáze
    session.add(fingerprint)
    session.commit()
```

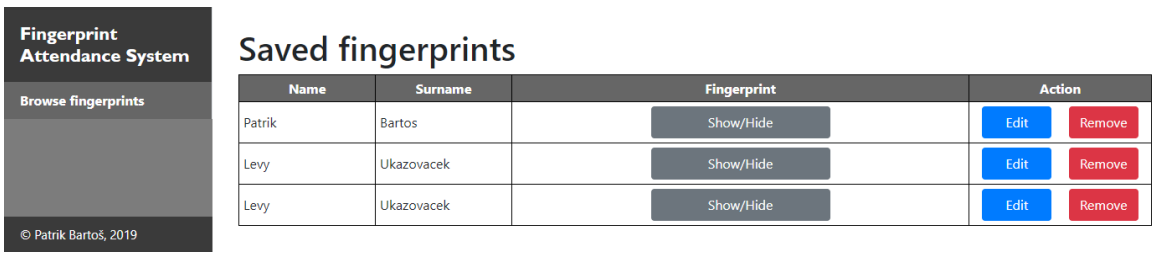
Obrázek č. 4 Uložení dat do databáze

4.6 Webové stránky

Při tvorbě stránek jsem využil framework Bootstrap. Po vytvoření HTML tabulky jsem ji potřeboval naplnit daty z databáze. K tomu jsem použil skriptovací jazyk PHP. Při načtení stránky se naváže komunikace s databází a pomocí MySQL příkazů se získají všechna data z tabulky v databázi. Úprava a mazání dat je zajištěna pomocí HTTP GET a POST požadavků.

4.6.1 Vzhled stránek

Vzhled stránek s tabulkou uložených otisků. Databáze obsahuje tři uložené otisky. Ve sloupci fingerprint můžeme zobrazit uložené informace o otisku. Ve sloupci action poté můžeme upravit jméno a příjmení uživatele nebo celý záznam smazat z databáze.



Obrázek č. 5 Vzhled stránek

Závěr

Původním cílem bylo vytvoření celého funkčního docházkového systému, včetně ověřování otisků a zapisování časových údajů do databáze. Bohužel se nakonec ukázalo, že problematika ověřování otisků je složitější, než jsem prvně očekával a nepodařilo se mi problém vyřešit před termínem odevzdání práce. I přesto jsem se toho hodně naučil a vyřešil mnoho problémů. Obeznámil jsem se s novými technologiemi a naučil s nimi pracovat alespoň na jednoduché úrovni. Do budoucna bych chtěl vyřešit již zmíněné ověřování otisků. Dále bych chtěl vylepšit webovou část projektu, například přidat registraci a přihlašování uživatelů a oddělit tak administrátory a běžné uživatele.

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] *Guide to Fingerprint Sensor Module with Arduino* [online]. Dostupné z: https://randomnerdtutorials.com/fingerprint-sensor-module-with-arduino/

[2] *IoT Biometric Attendance System using NodeMCU* [online]. Dostupné z: https://how2electronics.com/iot-biometric-fingerprint-attendance-system-nodemcu/

[3] *ESP8266WiFi library* [online]. Dostupné z: https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi

[4] *Adafruit Optical Fingerprint Sensor* [online]. Dostupné z: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-optical-fingerprint-sensor.pdf

[5] *Adafruit Fingerprint Sensor Library* [online]. Dostupné z: https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library

[6] *Adafruit Fingerprint Arduino Library* [online]. Dostupné z: https://adafruit.github.io/Adafruit-Fingerprint-Sensor-Library/html/class_adafruit___fingerprint.html

[7] *Fingerprint Module Basics* [online]. Dostupné z: https://fingerprint-module-r305-python-and-mysql.readthedocs.io/en/latest/chapter02.html

[8] *Arduino Client for MQTT* [online]. Dostupné z: https://github.com/knolleary/pubsubclient

[9] *I2C LCD on NodeMCU* [online]. Dostupné z: https://www.instructables.com/id/I2C-LCD-on-NodeMCU-V2-With-Arduino-IDE/

[10] *LiquidCrystal_I2C* [online]. Dostupné z: https://github.com/johnrickman/LiquidCrystal_I2C

[11] *Beginners Guide To The Paho MQTT Python Client* [online]. Dostupné z: http://www.steves-internet-guide.com/into-mqtt-python-client/

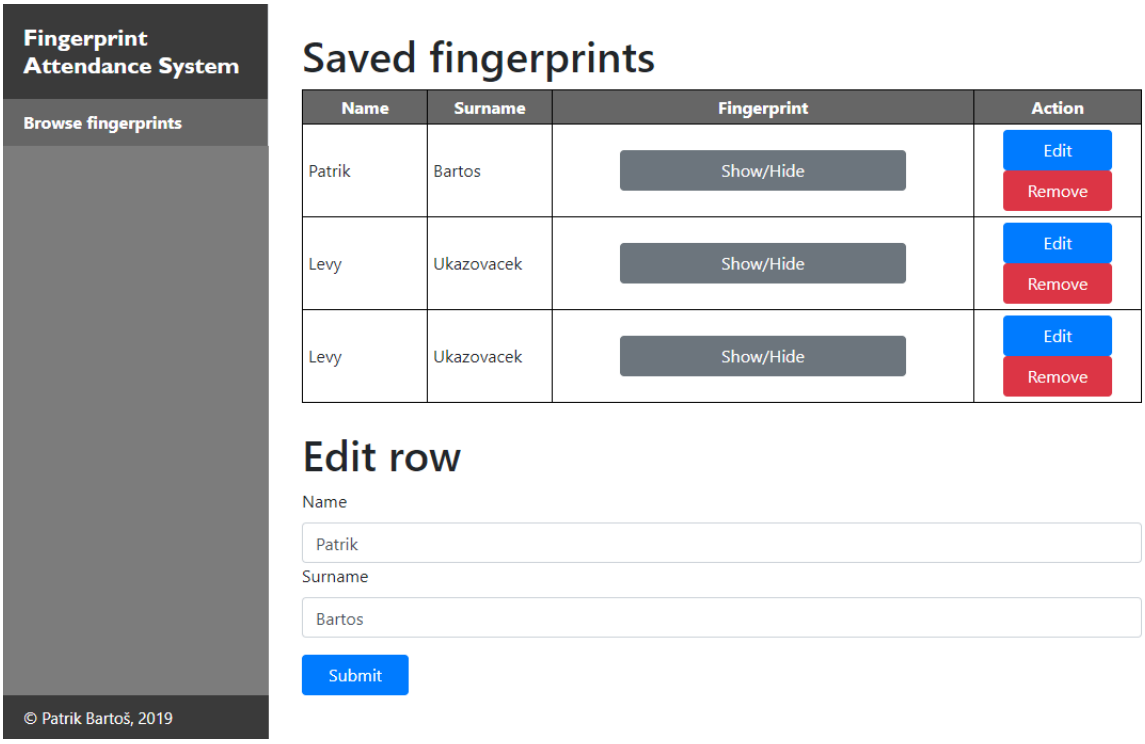
[12] *SQLAlchemy Basics Tutorial* [online]. Dostupné z: https://leportella.com/english/2019/01/10/sqlalchemy-basics-tutorial.html

[13] *Retrieve Data from MySQL Database* [online]. Dostupné z: https://www.tutsmake.com/php-retrieve-or-fetch-data-from-mysql-database-and-display/

[14] *Edit and Delete Record from Databse Using PHP* [online]. Dostupné z: https://www.allphptricks.com/insert-view-edit-and-delete-record-from-database-using-php-and-mysqli/

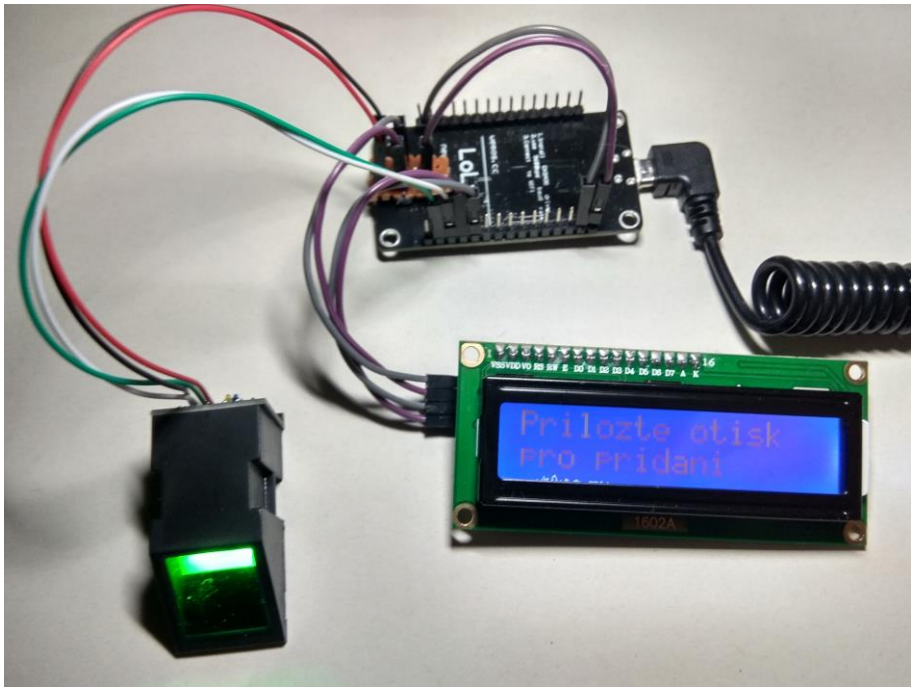
SEZNAM PŘÍLOH

Příloha č.1: Vzhled webových stránek



Stránky včetně formuláře pro editaci záznamu

Příloha č.2: Podoba zařízení



Finální podoba se zdrojem napájení