VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Mikroprocesorové a vestavěné systémy 2020/2021

ESP8266: ovládání LED

20. prosince 2020 xmusko00

Obsah

1 Úvod

Tato práce vznikla jako součást projektu předmětu Mikroprocesorové a vestavěné systémy. Odkaz na demonstrační video je zde.

1.1 Motivace

Internet věcí je v dnešní době výrazným trendem. Tento pojem je definován jako "Síť fyzických zařízeních, vozidel, domácích spotřebičů a dalších zařízeních, která jsou vybavena elektronikou, softwarem, senzory/čidly a hlavně síťovou konektivitou. Ta umožňuje těmto zařízením se navzájem propojit a vyměňovat si data."[1]. Odhaduje se, že v roce 2020 se počet těchto zařízení pohyboval okolo 30 miliard, přičemž v předchozím roce se trh s IoT zařízeními vyšplhal k 80 miliardám dolarů.[2]

1.2 Zadání

Zadáním bylo navrhnout vestavěný systém umožňující ovládat prostřednictvím modulu NodeMCU tři k němu připojené LED diody.

- Zařízení (tj. modul NodeMCU) bude využívat WiFi v režimu AP, na které bude možné se připojit pomocí mobilního telefonu.
- Zajistěte, aby se systém choval za všech okolností přirozeně a byly rozumným způsobem ošetřeny vstupy.
- Kromě ovládání jednotlivých LED diod umožněte i spuštění nějaké sekvence (střídavé blikání, rotace a podobně). Řízení těchto sekvencí, časování a podobně musí být řešeno v procesoru (t.j. klient odešle přes webové rozhraní příkaz pro rozblikání světel a vlastní blikání dělá procesor).
- Při spouštění sekvencí musí procesor na závěr informovat uživatele o dokončení sekvence (není žádoucí, aby uživatel byl nucen manuálně provádět reload stránky, aby se dozvěděl, že je možné spustit další sekvenci).

2 Návrh řešení a použité technologie

Řešení se skládá ze dvou částí: `backendu` na ESP zařízení a `frontendu`. Tyto dvě části mezi sebou komunikují.

2.1 Backend

Veškerý backed je implemetován na ESP.

Acess Point
Dle zadání máme vytvořit na ESP Acces Point. Toho lze dosáhnout pomocí knihovny ESP8266WiFi vytvořením Softwarového `Acces Pointu`. Ten se využívá právě u zařízení, která původně nebyla vyrobena jako router. Zabezpečení je realizováno pomocí `WPA2 PSK`.

Server
Ovládání ESP je realizování pomocí webové stránky. Aby tedy zařízení mohlo poskytovat webový obsah, je třeba z něj vytvořit Webový server. K tomuto účelu byl použit AsyncWebServer. Z názvu vyplývá, že dokáže řešit asynchronní komunikaci, a tedy i více spojení naráz. Poskytuje taktéž jednoduché rozhraní pro definování obsluh požadavků.

SPIFFS

V některých případech lze stránku prezentovat jako `plain text` uložený ve stringu, který je poslán přímo. To se týká většinou minimálních řešení určených pro desktop. Pokud však chceme oddělit zdrojový kód ESP od frontendu, musíme použít jiný přístup.

U ESP se nám přímo nabízí SPIFFS, což je knihovna poskytující `souborový systém`. Je schopná nahrát do zařízení soubory uložené ve složce data, která je možné na dotaz poslat ze servru.

Komprese

Paměť zařízení přece jen není moc obsáhlá a rychlost přenosu také nedosahuje velkých rychlostí, což je problém při výše zmíněném uložení a přenosu frontendu. Standart HTTP ale podporuje kompresi [3]. Mezi nejčastější typy komprese patří `gzip`, která je využita i v tomto projektu. Tuto kompresi podporuje i použitý server.

Ovládání LED

LED světla jsou ovládána klasickým způsobem.

2.2 Frontend

Pro ovládání LED diod stačí pouze jedna webová stránka s potřebnou funkcionalitou.

Doprovodné soubory

V základu potřebujeme `html` a nějaké stylování. V běžných html dokumentach jsou v záhlaví uvedeny odkazy na css a javascript. Pokud se má ale mobilní zařízení připojit k webovému servru, ztrácí připojení k internetu a uvedené odkazy se nemají odkud stáhnout. Je tedy třeba soubory s javascriptem a css také přímo uložit na server, odkud se budou stahovat. V našem případě je na servru uložená minimální verze Bootstrapu.

AJAX

Abychom měli možnost asynchronně posílat požadavky na server bez nutnosti znovunačtení stránky, je použito rozhraní `XMLHttpRequest`.

3 Implementace

Celá aplikace funguje na bázi server-klient, kde klient zasílá `HTTP_GET` požadavky na server, který na ně odpovídá a provádí příslušné akce. Tyto požadavky mohou být: vypnutí/zapnutí příslušné LED diody, spuštění některé ze tří sekvencí a načtení nekterého z frontend souboru. V reakci na požadavek je následně posláno buď potvrzení, nebo příslušné soubory.

3.1 Backend

Backend byl implementaván v `C++` ve vývojovém prostředí Arduino IDE.

V setup funkci probíhá probíhá inicializace:

- LED diod
- SPIFFS
- Acess Pointu
- Funkcí obsluhy dotazů

U Acess Pointu je nastaven identifikátor WiFi sítě a její heslo, které musí být alespoň o dělce 8 znaků.

V loop funkci probíhá realizace sekvence, pokud nějaká příjde, a přepínání LED diod.

Obsluha dotazů

Obsluha dotazů se vytváří pomocí funkce `on`. Oproti běžné odpovědi, je třeba přidat hlavičku informující o kompresi.

V případě požadavku na změnu LED diody, jsou v obslužné funkci zkontrolovány parametry obsahující identifikátor LED diody a její stav. Jelikož je tato funkce asynchroní a dle dokumentace se nedoporučuje přistupovat k pinům, je pouze změněn globální stav dané LED diody a její přepnutí se odehrává v loopu. Na konci je odeslána informace potvrzující změnu dané LED diady, a to pomocí asynchronního JSONu.

V jednu chvíli server zpracovává pouze jednu sekvenci. Funkce pro obsluhu sekvence tedy při přijetí požadavku přepne globální proměnnou `is_sequence_running` na `true`, aby server nepřijímal žádné další sekvence a uloží ukazatel na požadavek. Jak už bylo zmíněno, obslužná funkce nesmí sahat na piny, proto se poznačí číslo sekvence a její vykonání opět závisí na loopu. Po dokončení je poslána informace o dokončení sekvence.

Přepínání LED diod

Informace o diodách je obsažena v globální struktuře `leds`, která v sobě obsahuje čísla portů (4, 0, 2), aktuální stav diody, hodnotu pro zapnutí a hodnotu pro vypnutí.Aktuální stav je modifikován v obslužných funkcí. V loopu je pak volána funkce `run_led()`, která každou diodu vypne, nebo zapne podle stavu.

Spouštění sekvence

Pokud je v loop zjištěno na základě proměnné `is_sequence_running`, že má dojít ke spuštění sekvecnce, je zavolána funkce `runy_seq` s argumentem `selected_sequence`, což je globální proměnná obsahující id sekvence. V ní se pak provádějí sekvence zapináním/vypínáním diod a použitím `delay`.

3.2 Frontend

Hlavní úlohou frontendu je pohodlná kontrola. Ovládání je rozděleno na dvě části.

V horní se pomocí `toogle check boxu` dá přepínat stav dané led diody. Po přepnutí se pošle požadavek na server, ale samotná dioda ve frontendu se rozsvítí až po přijetí potvrzení, tedy zároveň s rozsvícením diody na zařízení.

Sekvenci lze vybrat z tabulky. Při prvním zpuštění je stav sekvence `Ready`, při odeslání požadavku se přepne na `Running` a po dokončení na `Done`.

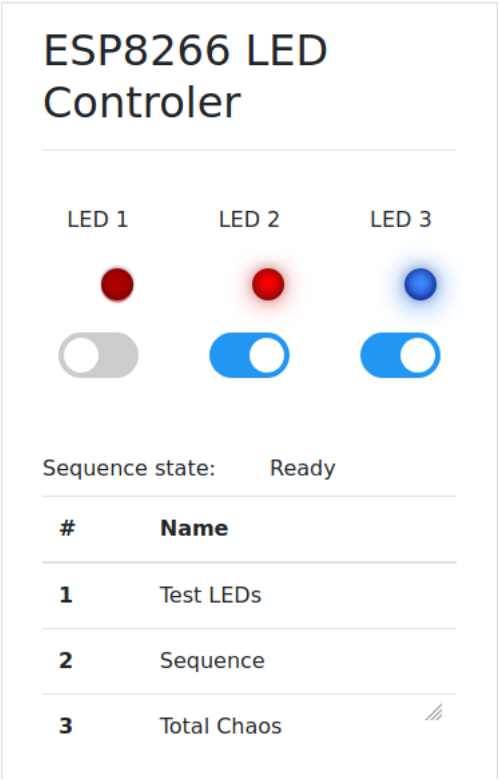
Led diody lze přepínat i během spuštěné sekvence. Po jejím dokončení bude jejich stav aktualizován

Požadavky

Požadavky jsou realizovány pomocí `XMLHttpRequest objektu` a jsou posílány metodou `GET` s případnými parametry. Jsou také nastaveny funkce odchytávající odpověď.

Design

Design je uzpůsoben tak, aby bylo možno zobrazit obsah i na mobilním zařízení. Z frameworku je vyu-



Obrázek 1: Front end

žita minimální verze Bootstrapu. Stylování LED diod
a tvar přepínačů byl převzat z codepen a w3schools.

4 Závěr

V rámci projektu byl vytvořen asynchronní web ser-
ver na zařízení ESP8266 umožňující ovládání LED diod pomocí jednoduchého rozhraní.

Zdroje

[1] [online]. Dostupné na: <https://www.iotport.cz/iot-novinky/ostatni-clanky-o-iot/co-to-je-iot>.

[2] [online]. Dostupné na: <https://cs.wikipedia.org/wiki/Internet_v%C4%9Bc%C3%AD#Stavebn%C3%AD_a_dom%C3%A1c%C3%AD_automatizace>.

[3] [online]. Dostupné na: <https://en.wikipedia.org/wiki/HTTP_compression>.