Informatika – maturitní práce

Sledování vytíženosti prostoru pomocí ultrazvukových měřičů

Mikoláš Fromm

Vedoucí práce: Emil Miler

Školní rok: 2020/21

Obsah

Úvod

Jako svůj maturitní projekt jsem si zvolil po vlastním výběru **navržení obecného počítadla vytíženosti specifických objektů.** Jako reálný fyzický objekt, u kterého chci svým projektem vyzjistit onu vytíženost, je pak pánská toaleta, resp. pánské školní pisoáry v 1. patře před ředitelnou.

Zadání

Pro návrh jsem si zadal několik vstupních podmínek, které bude celý projekt splňovat a podle kterých budu projekt vytvářet. Projekt musí:

- anonymně počítat osoby u jednotlivých pisoárů
- porovnávat vytíženost s ostatními pisoáry
- zobrazovat aktuální data přes WiFi síť na vlastním web-severu
- správně vyhodnocovat vstupní data poskytnutá senzory
- fungovat na napětí max. 5V

Návrh

Pro realizaci projektu jsem si vybral 6 ultrazvukových měřičů vzdálenosti SR-HC04 a SR-HC05 pro svou vysokou přesnost a zároveň velký funkční rozsah, vývojový mikrokontrolér ESP32-DEVKIT1 pro správu všech senzorů, vyhodnocování výsledků měření a možnost data sdílet přes WiFi modul po bezdr. síti protokolu 802.11g a také mikropočítač OrangePi3 jakožto webserver a MQTT broker s grafickým zobrazením získaných dat.

SR-HCO4/HC05

Ultrazvukové měřice vzdálenosti jsou dostupné a spolehlivé měřiče, které fungují na jednoduchém principu rychlé iniciace ultrazvuku a následném čekání na zpětný odraz zvuku od měřené překážky. Jako výstup vrací zařízení čas, za který zvuk zdolal vzdálenost k překážce a zpět.

ESP32

Mikrokontrolér ESP32 jsem si vybral hlavně díky své možnosti data sdílet po WiFi síti. Původní plán byl využít mikrokontrolér Arduino UNO, který však nedisponuje WiFi modulem, a tedy se nehodí do konceptu mého projektu.

ESP se v mém projektu stará o většinu výpočtů a úloh. Iniciuje měření u jednotlivých senzorů, stejně jako přepočítává vzdálenost, vyhodnocuje obsazenost a posílá ji na webserver pomocí MQTT nodů. Jeho nejdůležitější logická úloha pak spočívá ve správném *bodovém* ohodnocení vytíženosti. Mikrokontrolér musí pomocí senzorů poznat, že u pisoáru stojí stále stejná osoba, případně že se osoby změnily. Stejně tak musí mikrokontrolér ohodnotit každou osobu pouze jednou, aby data byla relevantní.

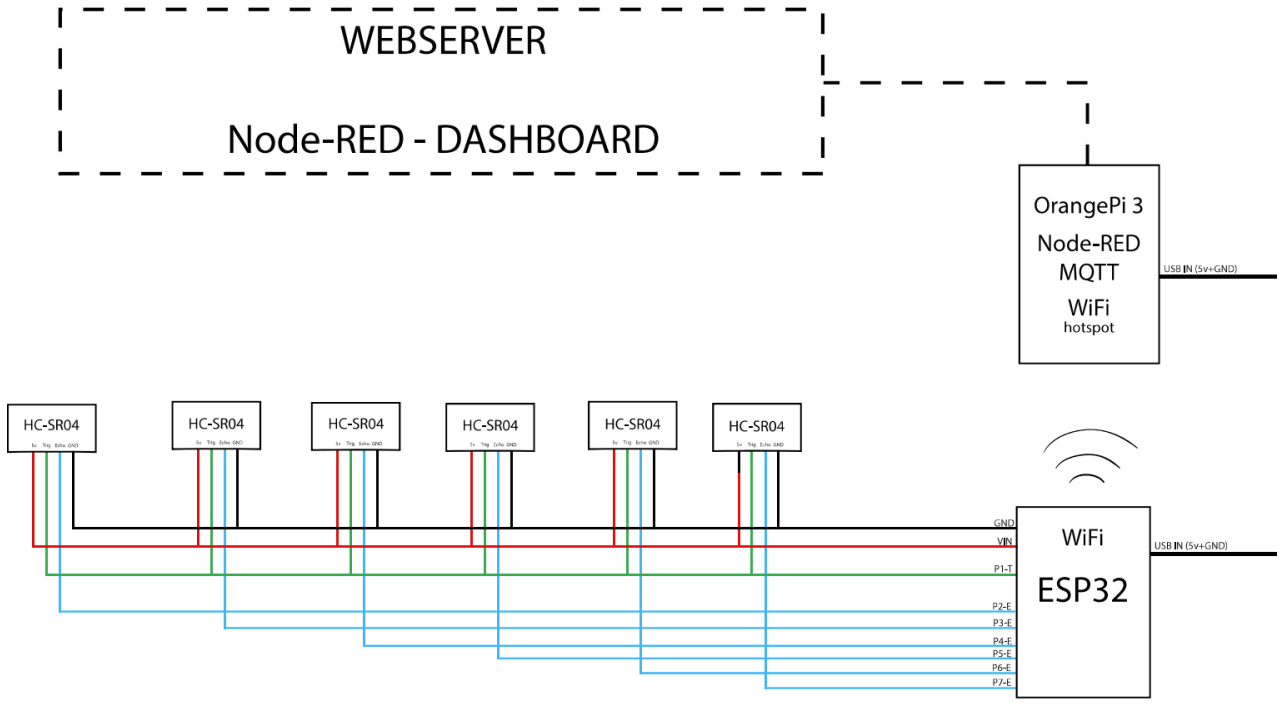
Má však jednu technickou nevýhodu, která se projeví zejména při zapojování více senzorů. Tento mikroprocesor pracuje s napětím 3,3V, zatímco senzory jsou nejpřesnější v 5V zapojení. To znamená, že do fyzického konceptu je třeba zapojit i „level-shifter“, tedy převodník napěťových úrovní, který dovoluje komunikaci mezi 5V a 3,3V zařízeními, aniž by došlo k poškození kteréhokoliv zapojeného zařízení. Schéma zapojení a koncept DPS (plošného spoje) bude k vidění níže.

OrangePi 3

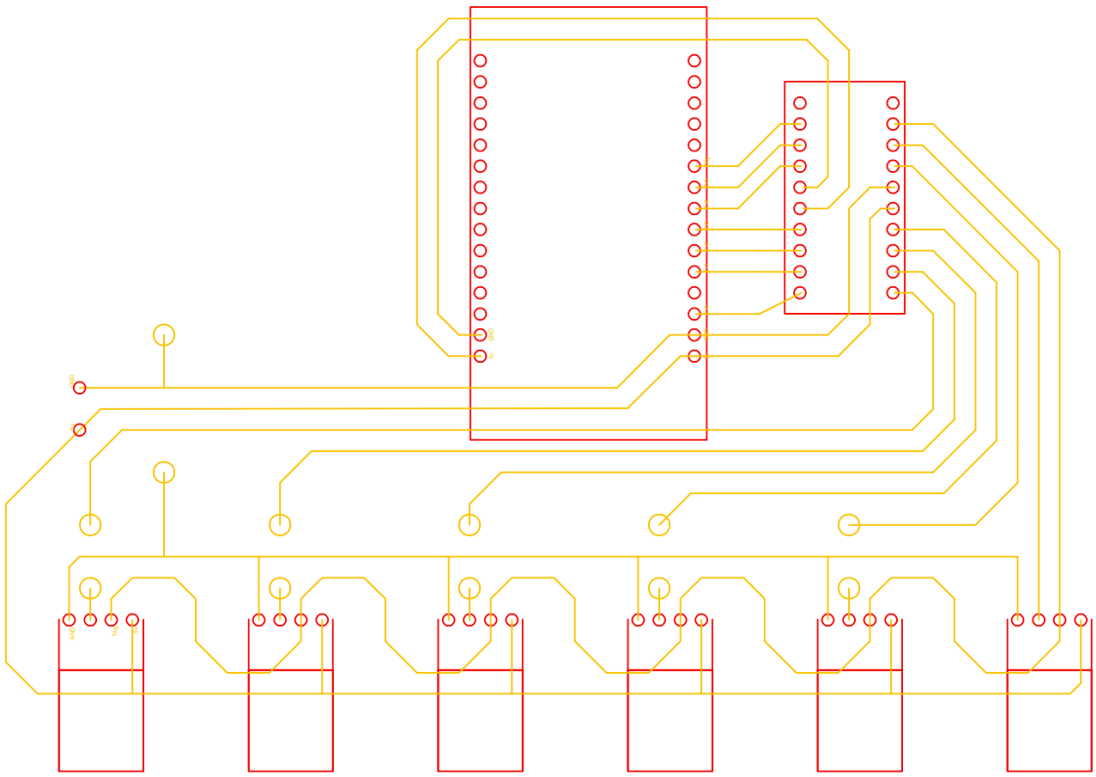
Mikropočítač OrangePi 3 jsem vybral hlavně díky svým dvěma vlastnostem: Má v sobě zabudovaný WiFi modul, který je schopný paralelně vytvářet Wifi-síť a zároveň z ní čerpat data, a je schopný plynule a paralelně pracovat jako webserver a MQTT broker. Ze všech alternativ, které u mikropočítače byly, jsem vybral OrangePi 3 pro jeho dostupnost.

Zpracování

V následujícím schématu je přesně popsané zapojení celého projektu:



OrangePi3 je zapojením velmi jednoduchý, avšak schéma zapojení ESP32 kontroléru a jednotlivých senzorů, zejména díky jiným napěťovým úrovním, je složitější a vyžadovalo individuální řešení pomocí DPS a to hlavně pro zapojení „převodníku“, což je obdélníková součástka vpravo na schématu vedle většího ESP32:



Zakončení DPS ve spodní části počítá s použitím konektorů, díky kterým bude možné lépe pracovat s dlouhými vodiči vedoucími k senzorům.

DPS bylo vytvořeno *ručním způsobem*, tedy fixou a následným leptáním. Tzn. že cesty se nejdříve na celou měděnou desku předkreslí, aby byla následně deska ponořena do roztoku **chloridu železitého**, který naleptá všechen povrch, který nebyl zakryt speciální fixou. Ve výsledku tak zůstane pouze deska tištěných spojů, které odpovídají konkrétním cestám v návrhu. Profesionální řešení ve formě zadání výroby DPS externí firmě jsem nezvolil zejména kvůli časové náročnosti, resp. i daleko vyšší finanční náročnosti, která je spojená s objednávkami malého počtu kusů.

Dalším důležitým prvkem byl návrh správného firmwaru pro ESP32. Po několika provedených testech, které jsou mimo jiné také obsažené v repositáři, jsem získal jasnou představu, jakého výsledku lze s touto sestavou docílit. Stanovil jsem si tedy několik základních prvků, které musí můj softwarový návrh splňovat:

1) Musí umět rozlišovat stav 0 a 1, tedy *volný* nebo *obsazený*.
2) Musí mít možnost nastavit rozsah, ve kterém bude zaznamenávat obsazenost.
3) Musí umět změřit reálný čas, jak dlouho stála překážka v rozsahu.
4) Musí umět tento čas porovnat se zadaným kritériem minimálního času.
5) Na základě splnění podmínek musí umět do statistiky přidat bodové ohodnocení.
6) Všechny data musí umět zobrazit a porovnat na webserveru.

Repositář je dostupný na tomto odkazu: https://github.com/MeCoolGJK/GJK_WC_counter

Instalace Node-RED, stejně jako MQTT brokeru *Mosquitto* a Node-RED – dashboard na Ubuntu-server běžící na OrangePi3 proběhla bez problémů podle základních dokumentací a návodů, a proto jí nebudu věnovat více prostoru, mimo jiné proto, že OrangePi a poddružený webserver není klíčovým prvkem projektu.

Funkčnost projektu

Program je díky celkové koncepci schopný rozeznat příchozí osobu k pisoáru či jinému objektu. Jakmile se osoba, případně i jakákoliv jiná překážka, dostane do požadovaného vzdálenostního rozsahu, program si zaznamená čas této události (čas je relativní od spuštění programu), přepne senzor do stavu *obsazený* a pokračuje v dalším měření. Jakmile se osoba nebo jiný předmět vzdálí od senzoru natolik, že je mimo požadovaný rozsah, a zároveň byl před senzorem požadovanou dobu, program si opět zaznamená čas této události (relativní od spuštění programu), přepne senzor do stavu *volný* a následně vypočte rozdíl mezi zaznamenanými časy, který se rovná strávenému času před senzorem, a publikuje ho pomocí MQTT na server. Pokud by osoba či jiný objekt před senzorem byl v požadovaném vzdálenostním rozsahu, přesto v něm nesetrval požadovanou dobu, program tento případ ignoruje a neovlivní tak data. V opačném případě, že osoba či jiná překážka setrvá před senzorem jakkoliv delší čas než je požadován, systém i přesto přiřadí do celkové statistiky pouze jeden bod a to až po vzdálení osoby či jiného objektu. Takto běží program paralelně na všech 6 senzorech / pisoárech. O všechny početní úkony, jako je přepočet vzdálenosti z uraženého času, přičtení bodu při splnění podmínek a zaznamenání do celkové statistiky, se stará ESP32, které následně tyto data ve formě nodů pošle bezdrátově pomocí Wifi na MQTT server, který tyto nody zpracovává. Pro zjednodušení situace je pak MQTT server fyzicky i MQTT client, který data přijímá a WebServer, který data zobrazuje. Je pravděpodobné, že přenos dat je pak rychlejší, ale hlavně je celé schéma díky tomuto sjednocení velmi jednoduché a praktické pro aplikaci „v terénu“.

Úskalí projektu

Přestože jsem se snažil vybrat nejvhodnější prostředky pro můj projekt, nejsou všechny naprosto ideální. Jmenovitě jde o samotný senzor a kvantitu mikrokontrolerů:

Senzor má sice velmi vysokou přesnost a velmi široký vzdálenostní rozsah, ale není soběstačný – neexistuje u něj pasivní mód, a tedy mikrokontrolér ho musí v každém cyklu spouštět, aby zjistil, jestli není objekt v rozsahu. To při použití 6 senzorů a jednoho mikrokontroleru může tvořit problémy, resp. dlouhou pomyslnou frontu, jelikož bude v jistých situacích dlouhá odezva na změnu vzdáleností; *zatímco si osoba stoupne před první pisoár v čase t_0 a program bude v čase t_0 u začátku cyklu druhého pisoáru, bude program chronologicky a postupně iniciovat a vyhodnocovat všechny zbylé senzory / pisoáry, než opět dojde metodicky k prvnímu a vyhodnotí, že před ním někdo stojí.* Na obhajobu projektu je však potřeba zmínit, že se předpokládá, že u pisoáru člověk přetrvává, nežli že by okolo něj jen rychle proběhl, což nepřímo vyřazuje tento problém jako příčinu zkreslení dat. Jako řešení by se nabízelo senzory oddělit od jednoho mikrokontroleru a provozovat například 3 jednotky ESP32 mikrokontroleru celkem – tedy 2 senzory na jeden mikrokontroler. Takové řešení je však příliš drahé vzhledem k účelu.

Hodnocení / Evaluace

(Testovací) provoz ukázal, že zmíněné úskalí projektu se neprojevilo. To hlavně proto, že jsem použil odlišný princip sběru dat než při původním testování senzorů. Proto program nemá dlouhou odezvu na pohyb před senzory a jeho odpověď na aktuální situaci je rychlá.

Tvorba DPS se také povedla, až na několik malých chyb, které vznikly pravděpodobně příliš tenkou vrstvou naneseného fixu – dvě cesty se porušily a bylo je potřeba napravit tenkým drátkem. Společně s tím na části destičky bylo příliš mědi, a proto se nevyleptala všude. Tento problém jsem řešil frézováním desky, abych vytvořil oddělené cesty.

Při návrhu DPS jsem počítal s využitím konektorů pro jednodušší manipulaci, avšak ty se neosvědčily jako spolehlivé, a proto jsem je zpětně odstranil a vodiče pevně naletoval.

Pro senzory jsem zároveň vymodeloval i kryt, který jsem následně tiskl na 3D tiskárně.

Přestože jsem chod celého projektu nemohl ostře nasadit do provozu ve škole, provedl jsem několik simulací (pro doplnění se hodí říct, že simulace neobsahovaly nic společného s toaletami), ve kterých se projekt jevil jako velmi povedený a přesně funkční.

Se svou prací jsem velmi spokojen, jelikož obsahovala spoustu prvků při návrhu takového projektu – od obecného návrhu, přes hardwarový návrh, přes testování, přes fyzickou realizaci (letování, 3D tisk, tvorba DPS) až po finální softwarový návrh aplikovatelný do mého projektu. Pouze mě mrzí, že jsem nemohl společně se svým odevzdáním práce předložit i jasná data, která by konečně rozsekla, který ze 6 pisoárů je na školních záchodech nejméně vytížen.