IMP – Projekt, varianta č. 21
ESP32: Měření srdečního tepu [digitální senzor]

Dominik Vágner,
xvagne10

16. prosince 2022

Obsah

1 Úvod

Zadáním projektu bylo implementovat měřič srdečního tepu a míry okysličení krve pro mikrokontroler ESP32, pomocí digitálního senzoru z kterého měli tyto hodnoty být zobrazeny na OLED displeji.

2 Problematika

Problematiku si můžeme do rozdělit do dvou částí a to inicializace sběrnic a komponentů, následná práce s nimi a zpracování z nich získaných dat.

2.1 Inicializace sběrnic a komponentů

Každá z našich externích komponent komunikuje s mikrokontrolerem pomocí jiné sběrnice. Náš OLED displej používá sběrnici SPI, zatímco náš senzor tepu komunikuje na sběrnici I2C. Jako první věc v našem programu, před hlavní smyčkou, musíme inicializovat naše komponenty a jejich sběrnice. Při inicializaci musíme specifikovat jaké jsou čísla GPIO pinů které používáme, jejich mód, povolení pull-up rezistorů a jejich frekvenci. Prvotní inicializace senzoru také vyžaduje nastavení jeho konfiguraci, kde nastavujeme jeho hodnoty určující například délku pulzu, nebo proud jednotlivých LED.

2.2 Práce s komponenty a získanými daty

S displejem naše práce vypadal velmi přímočaře. Jediné jsme potřebovali udělat bylo zvolit rozlišení, font a při zachycení srdečního pulzu smazat aktuálně zobrazený obsah displeje a nechat vykreslit nový s aktuálními hodnotami.

U senzoru už je situace komplikovanější, protože z něj přímo hodnoty které potřebujeme zobrazovat nedostáváme. Data které můžeme číst jsou pouze sečtené hodnoty střídavého a stejnosměrného proudu. Tyto data jsou také v senzoru ukládané do fronty z které je musíme číst a potvrzovat jejich přečtení pro posunutí fronty [6]. Z přečtených dat musíme nejdříve nejdříve odstranit hodnoty stejnosměrného proudu a následně musíme data vyfiltrovat a z výsledků rozhodnout zda jsme detekovali nový puls. Pokud byl puls detekován tak musíme vypočítat počet pulsů neboli tepů za minutu a kyslíkovou saturaci.

Počet srdečních tepů za minutu počítáme tak že vydělíme délku minuty v milisekundách délkou jednotlivého pulsu, z těchto dat také děláme klouzavý průměr abychom dosáhli více srozumitelných výsledků.

Kvadratický průměr potřebný pro výpočet kyslíkové saturace můžeme vypočítat pomocí rovnice (1), jedná se o vydělení střídavých a stejnosměrných proudů červené a infračervené diody.

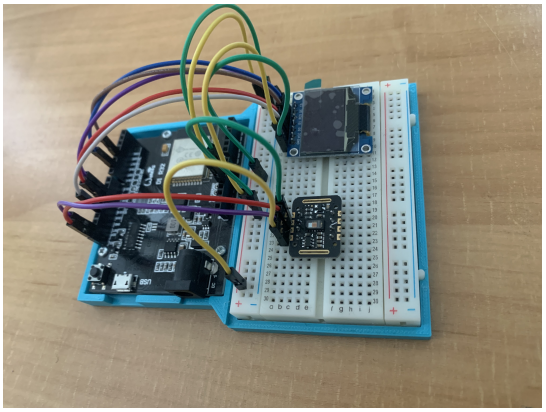
$$R = \frac{\frac{AC_{\text{red}}}{DC_{\text{red}}}}{\frac{AC_{\text{infrared}}}{DC_{\text{infrared}}}} \tag{1}$$

Po získaní daného průměru poté provedeme lineární aproximaci kyslíkové saturace. Po upravení hodnot z doporučených nastavení [5] výrobci čipu tak aby seděl na testovaný kus (v porovnání s komerčním měřičem kyslíkové saturace a tepu, viz sekce 5) se došlo na rovnici (2)

$$SpO2 = 117 - 18R \tag{2}$$

3 Implementace

Před počátkem implementace bylo třeba propojit mikrokontroler s komponenty umístěných na nepájivém poli.



Obrázek 1: Zapojení komponent

Práci s displejem za nás řeší knihovna, která nám poskytuje font a příkazy pro vykreslení textu a mazání obsahu displeje [1]. Pro senzor máme taky většinu ze zmiňovaných problémů jako interakci s frontou, zpracování přerušení nebo výpočet počtu tepů za minutu nebo kyslíkové saturace za nás řeší s pomocí knihovny [2]. U této knihovny bylo potřeba pro lepší výsledky udělat menší úpravy a nastavení jiných hodnot než byly uvedené v ukázkách.

3.1 Funkce

- `sensor_task` – Nejdůležitější funkce programu která periodicky kontroluje zda se vyskytl srdeční puls a případně obnoví hodnoty zobrazené na displeji novými nově získanými ze senzoru.
- `display_init` – Inicializuje displej za pomocí knihovny použité k jeho ovládání.
- `sensor_i2c_init` – Inicializování I2C sběrnice pro senzor tepu.
- `sensor_init` – Inicializace a konfigurace senzoru tepu.

3.2 Rozběhnutí projektu

Projekt byl vytvářen s pomocí rozšířeni PlatformIO pro vývojové prostředí Visual Studio Code. Po vytvoření projektu pro platformu Espressif ESP32 s použitým frameworkem ESP-IDF. Musíme do projektu vložit zdrojové kódy z odevzdaného archivu a nechat projekt přeložit a nahrát do mikrokontroleru.

Poté po přiložení prstu na senzor tepu a míry oksyličení krve, by se na displeji měli automaticky začít vykreslovat hodnoty.

4 Rozšíření

Pro rozšíření projektu mě napadlo udělat z ESP mikrokontroleru taky Wi-Fi access point a na něm zprovoznit web server na kterém by se pravidelně obnovovali hodnoty, tedy stejně jako na displeji. Toto rozšíření mi přijde velmi praktické, chtěl jsem si ho vyzkoušet, protože by jej šlo využít i pro domácí prostředí kdybychom měli nějaký senzor a hodnoty z něho bychom chtěli mít přístupné odkudkoliv. Také by poté šlo vynechat zobrazování na displeji a zjednodušila by se tak komplexita zapojení a snížila se tak i cena.

4.1 Funkce potřebné pro rozšíření

- `wifi_init` – Inicializace a konfigurace ESP jako Wi-Fi přístupového bodu [4].
- `http_init` – Inicializuje HTTP server a přidání dvou koncových bodů [3].
- `index_handler` – Vytvoření obsluhy pro koncový bod `/`, který zobrazuje stejné hodnoty jako displej a pomocí asynchronních dotazů na koncový bod `/values` je obnovuje.
- `values_handler` – Vytvoření obsluhy pro koncový bod `/values`

5 Testování

Projekt byl testován oproti komerčnímu produktu pro měření tepu a krevní kyslíkové saturace. Hodnoty kyslíkové saturace se podařilo odladit až na téměř identické, ale u hodnot tepů za minutu se občas stane, že na chvíli vyskočí na moc vysoké hodnoty a pak se po chvilce vrátí zpět. Nepodařilo se úplně přesně zjistit příčinu tohoto jevu, ale nejpravděpodobněji se bude jednat o nedostatečné omezení okolního světla nebo nekonzistentní tlak na prstu na senzor. Také nejde vyloučit možnost chyby v knihovně.

Výsledek a ukázku testování si lze prohlédnout zde zde.

6 Závěr

Projekt považuji za zajímavý prvotní úvod do programování vestavěných systémů. Zároveň bylo zajímavé nakouknout do knihoven pro ovládání externích komponent.

Autoevaluace

Na projektu si myslím, že se mi podařilo implementovat vše s plnou funkčností a to s kvalitně napsaným kódem. Také jsem projekt rozšířil i další funkcionalitu jako HTTP server. Myslím že zdokumentování by možná mohlo být kvalitnější, ačkoliv nevidím v něm zase natolik velké nedostatky. Projekt bych si tedy ohodnotil plným počtem.

Literatura

[1] *ESP-IDF Driver for the SSD1306 OLED display.* [vid. 2022-06-12]. Dostupné z: `https://github.com/nopnop2002/esp-idf-ssd1306`.

[2] *ESP-IDF Library for MAX30102.* [vid. 2022-06-12]. Dostupné z: `https://github.com/JoshDumo/esp32-max30102`.

[3] *Guide for setting up a web server on ESP32.* [vid. 2022-06-12]. Dostupné z: `https://embeddedexplorer.com/esp32-web-server/`.

[4] *Guide for setting up an ESP32 as an access point.* [vid. 2022-06-12]. Dostupné z: `https://embeddedexplorer.com/esp32-wifi-access-point/`.

[5] *MAX30102 Application Node - Recommended Configuration and Operating Profiles.* [vid. 2022-06-12]. Dostupné z: `https://pdfserv.maximintegrated.com/en/an/AN6409.pdf`.

[6] *MAX30102 Datasheet.* 2018. [vid. 2022-06-12]. Dostupné z: `https://datasheets.maximintegrated.com/en/ds/MAX30102.pdf`.