

NAV - ESP32 Meranie srdečného tepu

Sebastián Krajňák - xkrajn05

Duben 2023

1 Popis projektu

Cieľom projektu bolo s poskytnutým vývojovým kitom Wemos D1 R32 na bázi SoC ESP32 od spoločnosti Espressif, oximetrom MAX30102, OLED displejom SSD1306 a rotačným enkodérom KX-040 (niekde uvádzaný aj ako KY-040) realizovať merač srdečného tepu.

Hodnoty tepu sú zobrazované na OLED displej pričom pri štarte merača je používateľ vyzvaný, pomocou menu (viď Obr.1), na výber užívateľského rozhrania, čo je realizované pomocou rotačného enkodéru.

2 Spôsob realizácie

Projekt bol realizovaný v jazyku C pomocou ESP-IDF frameworku vo vývojovom prostredí PlatformIO a voľne dostupných knižníc. Mikrokontrolér komunikuje s OLED displejom cez rozhranie SPI a oximeter cez rozhranie I2C, zapojenie jednotlivých súčiastok je uvedené v neskoršej podkapitole. Pri realizácii projektu boli použité časti kódu, príkladových zdrojových kódov jednotlivých knižníc. Používateľ má na výber 3 jednoduché rozhrania

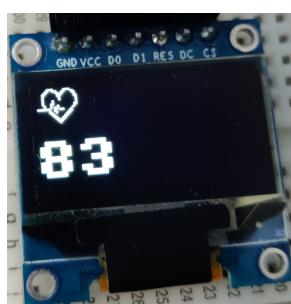
- textovú skratku (Obr. 2),
- malú ikonku (Obr. 3),
- len samostatné hodnoty BPM (Obr. 4).



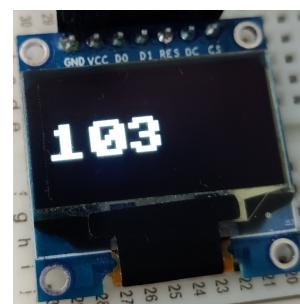
Obr. 1: Menu užívateľských rozhraní



Obr. 2: Rozhranie s textom



Obr. 3: Rozhranie s ikonou



Obr. 4: Rozhranie len s hodnotou BPM

2.1 Použité knižnice

Pri realizácii projektu boli použité voľne dostupné knižnice, konkrétnie

- esp-idf-ssd1306 : <https://github.com/nopnop2002/esp-idf-ssd1306>,
- max30102_esp-idf: https://github.com/Gustbel/max30102_esp-idf,
- esp-idf-RotaryEncoder: <https://github.com/nopnop2002/esp-idf-RotaryEncoder>.

Jedná sa o knižnice, ktoré neboli vytvorené samotnými výrobcami, preto majú niektoré nedostatky (napr. nepresné výpočty tepu a pod.). Bohužiaľ väčšina knižníc od výrobcov súčiastok sú vytvorené pre prácu na arduino frameworku a teda nefungujú na ESP-IDF frameworku (napr. populárna knižnica pre displeje od Adafruit).

2.2 Schéma zapojenia súčiastok

Do dosky Wemos D1 R32 boli jednotlivé súčiastky zapojené následovne:

SSD1306 OLED Displej	Wemos D1 R32
GND	GND
VCC	3V3
D0 (SCK)	GPIO18 (SCK)
D1 (MOSI)	GPIO22 (MOSI)
RES	GPIO17 (TX)
DC	GPIO16 (RX)
CS	GPIO05 (SS)

Tabuľka 1: Tabuľka zapojenia pinov OLED displeja

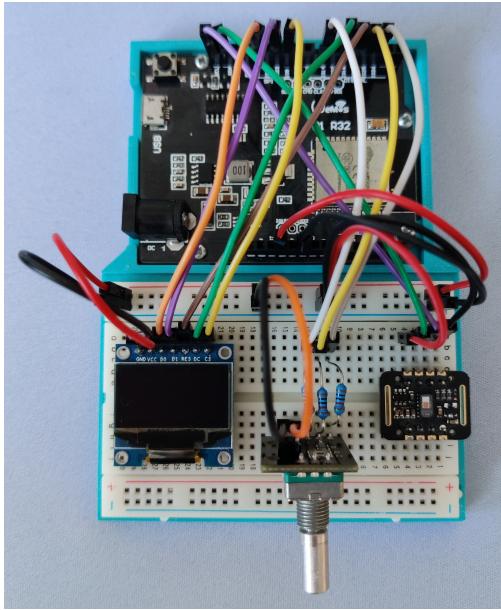
MAX30102 Oximeter	Wemos D1 R32
GND	GND
VIN	3V3
SDA	SDA
SCL	SCL

Tabuľka 2: Tabuľka zapojenia pinov oximetra

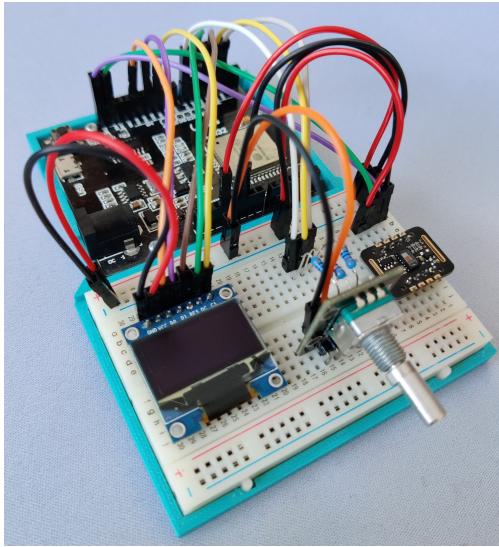
KX-040 Rotačný enkodér	Wemos D1 R32
GND	GND
+	3V3
SW	GPIO14 (cez 10K Ohm rezistor)
DT	GPIO25 (cez 10K Ohm rezistor)
CLK	GPIO26 (cez 10K Ohm rezistor)

Tabuľka 3: Tabuľka zapojenia pinov rotačného enkodéru

Rotačný enkodér bol do dosky zapojený pomocou 10K Ohmových pull-up rezistorov z dôvodu obmedzenia/odstránenia bouncingu signálu. Samotná realizácia zapojenia celého zariadenia je zobrazená nižšie na obrázkoch Obr. 5 a Obr. 6.



Obr. 5: Zapojenie do kitu



Obr. 6: Zapojenie do kitu

3 Funkcionalita a vlastnosti riešenia

Riešenie projektu bolo realizované ako jeden ESP-IDF task vytváraný v hlavnom `app_main` tasku. Úlohou hlavného tasku je primárne konfigurácia a inicializácia ako jednotlivých zariadení, tak aj rozhraní I2C a SPI. Zároveň, ako bolo spomenuté, hlavný task vytvára task `oximeter_task`, ktorý zabezpečuje celkovú aplikáciu.

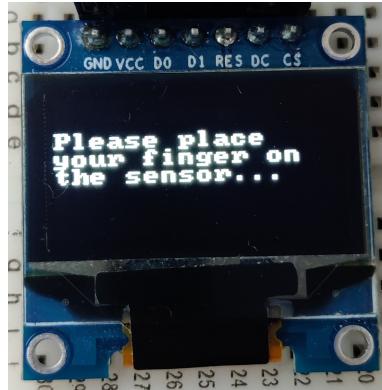
Funkcia `update_display()` je zodpovedná za zobrazovanie menu jednotlivých možností používateľského rozhrania. Aktuálne vybraná možnosť má inverzované farby celého riadku, pričom po stlačení tlačidla enkodéru je vybratá možnosť uložená do premennej `selected_option`. Funkcia je volaná stále ak dôjde k záchyteniu udalosti otočenia enkodéru, tj. ak sa premenná `event` rovná 0, stlačeniu tlačidla enkodéru napäk odpovedá hodnota 1.

Funkcia `scroll_options(int val)` je taktiež volaná stále keď dôjde k otočeniu enkodéru. Jej úlohou je zvyšovať alebo znížovať hodnoty premennej `selected_option`. V prípade, že je hodnota premennej `selected_option` mimo rozsahu ohraničeného počtom možností UI, je hodnota premennej obtočená na druhý koniec rozsahu tj. 0.

Funkcia `i2c_master_init()` má za úlohu inicializáciu I2C master komunikácie. Nastavuje SDA a SCL piny, povoľuje pull-up rezistory a nastavuje rýchlosť hodinového signálu. Je volaná len raz, pri spustení programu.

Funkcia `oximeter_task(void *arg)` predstavuje hlavnú úlohu celého programu. Na začiatku funkcie dochádza k inicializácii potrebných premenných ako napr. `bpmBuffer` pre ukladanie meraní tepu srdca, `cnt` na sledovanie počtu meraní tepu a `bpmAvg` a `bpmAvgSize`, ktoré sú použité pri výpočte priemerného tepu počas určitého počtu meraní. Zároveň pripraví počiatočné menu pomocou funkcie `update_display()`. Funkcia potom vstúpi do slučky `while`, ktorá nepretržite číta rotačný enkodér, kym používateľ nevyberie možnosť stlačením tlačidla enkodéra. Po výbere možnosti z menu sa displej vymaže a spustí sa meranie srdcového tepu. Celkové meranie tepu sa nachádza v nekonečnej `while` slučke, v ktorej prebieha k nastaveniu oximetra na mód merania srdečného tepu. Potom vstúpi do `for` slučky, kde načíta vzorky tepu a uloží ich do poľa `samples`. Pokiaľ nie je na senzore položený prst, teda `samples` obsahuje príliš malé hodnoty, tak sa výpočet tepu vynecháva, na displeji sa zobrazí výzva aby používateľ priložil prst na senzor (viď Obr. 7) a aktuálna iterácia hlavnej slučky sa preskočí na ďalšiu.

Ked sa na senzore nachádza prst a buffer meraní je plný, tak sa pomocou filtra kĺzavého priemeru vypočítá



Obr. 7: Výzva pri chýbajúcom prste na senzore

priemerný tep srdca a hodnota sa zobrazí na displeji, s predom vybraním UI.

Riešenie je plne funkčné avšak obsahuje zopár nedostatkov, zapríčinených dostupnými knižnicami. Hlavným problémom je nepresnosť merania tepla keďže hodnoty zvyknú v krátkom časovom intervale skákať v intervale od 60 do 200 BPM, tieto hodnoty sa po krátkom čase ale ustália. Ďalšou nevýhodou realizácie je knižnica displeja, displej je zobrazovaný správne avšak neexistujúca dokumentácia a poskytnuté funkcie značne stažovali prácu pri tvorbe menu a celého UI, aj napriek jeho jednoduchosti.