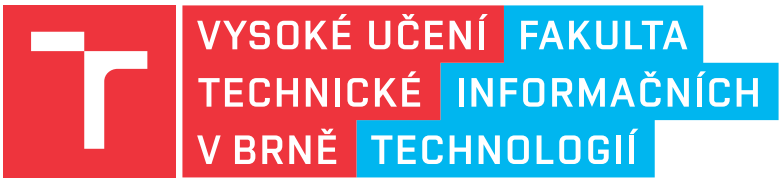


Mikroprocesorové a vestavěné systémy
(IMP) 2024/2025

ESP32: Měření srdečního tepu digitální senzor

11. prosince 2024

Marek Joukl `xjouk100`

Obsah

1 Úvod

Cílem projektu bylo vytvořit funkční měřič srdečního tepu za pomoci mikrokontroleru ESP32 na vývojové desce Wemos D1 R32, snímače srdečního tepu PulseSenzor a grafického OLED displeje SSD1306.

Projekt je implementován ve vývojovém prostředí Arduino IDE a jazyce C++ za použití knihoven Adafruit_SSD1306.h, Adafruit_GFX.h a Wire.h.

Program čte analogová data ze senzoru, filtruje signál a detekuje údery srdce, ze kterých vypočítá průměrný počet úderů za minutu (BPM) a vypíše na displej.

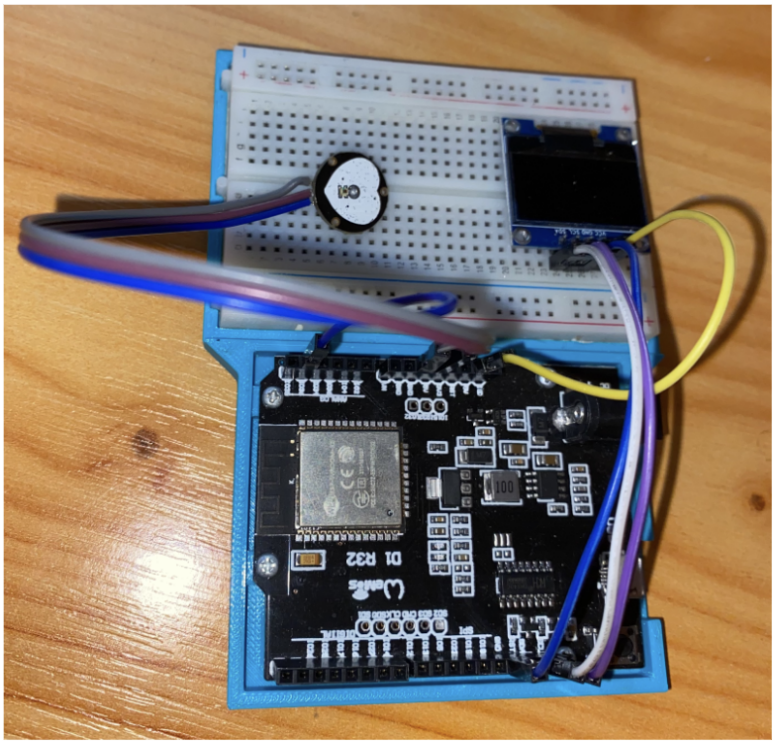
2 Zapojení

Pulse Senzor

- VCC → 5V (ESP32)
- GND → GND (ESP32)
- Signal → GPIO 36 (Analog input)

OLED Displej

- VCC → 5V (ESP32)
- GND → GND (ESP32)
- SCL → SCL (ESP32)
- SDA → SDA (ESP32)



Obrázek 1: Detail zapojení desky

3 Implementace

Snímač pulsu vysílá analogový signál, který kolísá v závislosti na srdečním tepu. ESP32 čte tento signál prostřednictvím svého ADC (analogově-digitálního převodníku). Pro vyhlazení signálu je použit dolnopropustný filtr (low-pass filter). Systém detekuje špičky (signály srdečního tepu) tak, že rozpozná, kdy signál překročí nastavený práh a následně opět klesne pod práh.

3.1 Inicializace

Ve funkci `setup()`, která se spouští při zapnutí zařízení, se nastaví BAUD rate, inicializuje OLED displej a zobrazí úvodní hláška. Dále je program ovládán z hlavní smyčky `loop()`.

3.2 Zpracování signálu

V hlavní smyčce jsou data ze senzoru čtena pomocí funkce `analogRead()`, která jsou dále filtrována dolní propustí, což eliminuje náhodný šum a usnadňuje další zpracování signálu. Jednotlivé údery srdce jsou detekovány na základě překročení předem definované prahové hodnoty. Tato prahová hodnota je kalibrována na základě signálu v klidovém stavu, kdy senzor není v kontaktu s prstem (baseline), a hodnot signálu při úderech srdce.

Před samotným výpočtem BPM je ověřeno, že signál opět klesl pod prahovou hodnotu. Tímto způsobem se zajišťuje, že detekovaný úder srdce je skutečný a že nedošlo k falešné detekci způsobené šumem nebo náhodnými výkyvy v signálu.

Po potvrzení detekce platného úderu se spočítá časový interval mezi aktuálním úderem a posledním zachyceným úderem (tzv. beat interval). Tento interval je pak použit k výpočtu BPM pomocí vzorce

$$BPM = \frac{60000}{\text{beatInterval}},$$

kde `beatInterval` představuje časový rozdíl v milisekundách mezi dvěma po sobě jdoucími údery srdce. Pokud vypočtená hodnota BPM spadá do přijatelného rozmezí (40–200 BPM), je hodnota BPM považována za platnou a použije se jak pro okamžité zobrazení na OLED displeji, tak pro výpočet průměrného BPM (`avgBPM`). Průměrné BPM se počítá jako průběžný průměr všech detekovaných platných úderů:

$$\texttt{avgBPM} = \frac{\texttt{totalBeats}}{\texttt{beatCount}}.$$

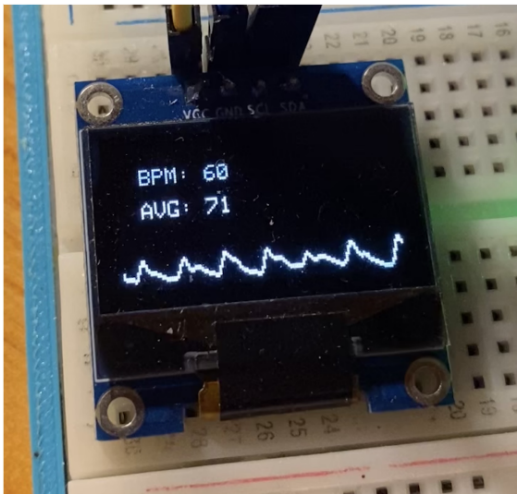
Tímto způsobem je průměrné BPM neustále aktualizováno na základě všech zaznamenaných úderů srdce.

Pro zvýšení přesnosti je také zavedena proměnná `debounceTime`, která zabraňuje falešné detekci úderů během jedné periody srdečního cyklu.

3.3 Vykreslování signálu

Funkce `plotHeartbeat()` zajišťuje vykreslování průběhu srdečního tepu na OLED displeji pro příjemnější uživatelský zážitek. Každý nový bod grafu se vykreslí posunutím o 1 pixel doprava. Pokud graf dosáhne pravého okraje displeje, začne vykreslování znovu od levého okraje a starý grafický obsah se vymaže pomocí funkce `fillRect()`, čímž se udržuje čistý a aktuální vzhled.

Analogová hodnota signálu je převedena pomocí funkce `map()` do rozsahu, který odpovídá spodní polovině displeje (mezi `PLOT_TOP` a `PLOT_BOTTOM`). Tento proces zajišťuje, že grafický průběh nepřekrývá horní část displeje, kde jsou zobrazeny textové informace. Pokud je hodnota mimo tento rozsah, je automaticky omezena tak, aby se nevykreslovala mimo vymezenou oblast.



Obrázek 2: Detail zapojení desky

3.4 Spuštění

Projekt je možné spustit v prostředí Arduino IDE. Je nutné mít nainstalované knihovny zmíněné v úvodu. Dále již stačí vybrat desku Wemos D1 R32 a správný port a nahrát program do zařízení tlačítkem Upload.

4 Testování

Správnost výstupu mého měřiče tepu byla ověřena srovnáním se senzorem zabudovaným v chytrých hodinkách. Testování probíhalo na dvou osobách, aby se zajistila spolehlivost zařízení u různých uživatelů a za různých fyzických podmínek. Hodnoty byly porovnávány jak v klidovém stavu, tak po fyzické námaze, kdy je tepová frekvence výrazně vyšší.

Toto opakované testování umožnilo důkladně zkontrolovat přesnost naměřených údajů a zajistit, že měřič poskytuje konzistentní výsledky. Výsledky z mého měřiče a hodinek byly zaznamenávány a porovnány, aby se zjistilo, zda naměřené hodnoty odpovídají. V následující tabulce jsou uvedeny příklady hodnot naměřených v různých podmínkách:

Osoba	Klidový (můj)	Klidový (chytré hodinky)	Po cvičení (můj)	Po cvičení (chytré hodinky)
1	62	65	112	117
2	71	73	132	136

Tabulka 1: Přehled naměřených hodnot ve vybraných situacích.

5 Videoukázka

Demonstrace funkčnosti projektu dostupná zde:
`https://drive.google.com/file/d/1XfVxuwtyYrqaY6fT22sxy4qXZxiEHp-n/view?usp=sharing`.
Senzor a mikrokontroler je nejprve připojen k napájení, dále se zobrazí úvodní hláška a poté je zobrazena hlavní obrazovka s daty. Pro správné měření je nutné po přiložení prstu nejprve chvíli setrvat (10-15s), aby se senzor správně zkalibroval a zobrazoval validní údaje. Na konci videa je pro srovnání vidět hodnoty naměřené chytrými hodinkami.

6 Závěr

6.1 Autoevaluace

Celkové hodnocení je určeno dle rovnice hodnotícího klíče:

$$\Sigma = (K1 + K2 \cdot F/5) \cdot (E + F + Q + P + D),$$

kde

- **Konstanta K1** = 0.25
- **Konstanta K2** = 0.75
- **Přístup k řešení E** = 2
  S řešením projektu jsem začal velmi brzy, což mi i přes problémy s nasazením a rozbitým senzorem (dráty vedoucí k senzoru se zlomily v části u senzoru, bylo nutné ve škole znovu zpájkovat) umožnilo projekt včas dokončit a odevzdat. Měl jsem zároveň dostatek času nejen splnit body dané v zadání, ale i zdokonalit vizuální stránku dodatečnou funkcí na vykreslování grafu, což senzor činí uživatelsky přívětivější.
- **Funkčnost řešení F** = 5
  Implementace splňuje všechny věci zmíněné v zadání. Měří tep, který zobrazuje na displeji zároveň s průměrným tepem.
- **Kvalita řešení Q** = 2
  Jak je popsáno i v části přístup, senzor je velice uživatelsky přívětivý a je jednoduché ho spustit. Stačí nahrát implementaci dle bodu 3.4. Zdrojový kód je přehledný a patřičně okomentovaný dle zásad správného programování.
- **Prezentace P** = 2
  Videoukázka je důkazem, že vše funguje jak má.
- **Dokumentace D** = 3
  Dokumentace obsahuje veškeré informace potřebné k pochopení, zapojení i vyzkoušení projektu. Je také typograficky kvalitně zpracovaná.

$$\Sigma = (0.25 + 0.75 \cdot 5/5) \cdot (2 + 5 + 2 + 2 + 3) = \mathbf{14}$$

7 Zdroje

1. https://lastminuteengineers.com/pulse-sensor-arduino-tutorial/
2. https://github.com/adafruit/Adafruit-GFX-Library
3. https://github.com/adafruit/Adafruit_SSD1306