

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



IMP - Mikroprocesorové a vstavané systémy

Tachometr

[ukázka video](#)

Obsah

1	Úvod	2
2	Hardware a vývojové prostředí:	2
3	Manuál použití implementovaného tachometru:	2
4	Implementace	2
4.1	Snímání stisknutí tlačítka	2
4.2	Práce s NVS	2
4.3	Výpočet rychlosti a ujeté vzdálenosti	3
4.4	Displej informací	3
4.5	Strukturování kódu	3
5	Schéma Zapojení	4

1 Úvod

Cílem je vytvořit cyklopočítáč zobrazující aktuální rychlosť, průměrnou rychlosť a ujetou vzdáenosť na displeji. Systém bude ovládán jedním tlačítkem (uživatel) a jedno tlačítko bude simulovat oběh kola.

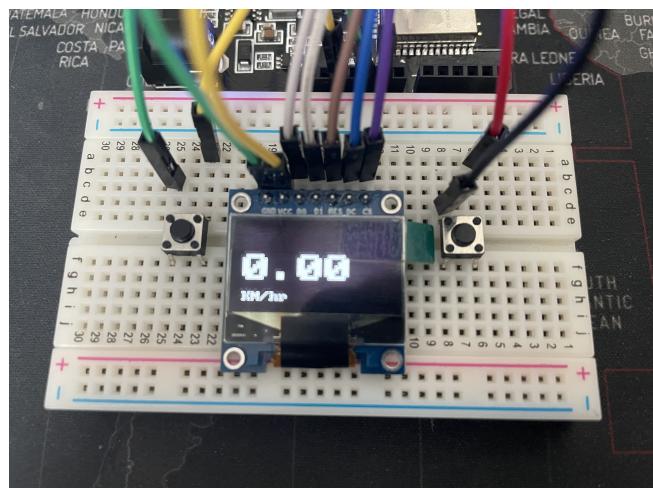
2 Hardware a vývojové prostředí:

Pro tento projekt jsem využil desku Wemos D1 R32 s ESP32 čipem, SPI displej SSD1306 pro vizuální zobrazení informací a 2 tlačítka připojená na breadboard.

V rámci vývoje jsem použil PlatformIO, které poskytuje efektivní nástroje a prostředí pro vytváření kódu pro ESP32 a správu periferií.

3 Manuál použití implementovaného tachometru:

Tachometr je ovládán pomocí dvou tlačítek, která jsou na Obr.1. Stisk pravého tlačítka slouží pro simulaci otočení kola. Jednoduchý stisk levého tlačítka slouží pro přepínání mezi displeji: Vzdáenosť, celková průměrná rychlosť a aktuální rychlosť. Zároveň po dlouhém stlačení levého tlačítka se vynuluje ujetá vzdáenosť.



Obr. 1: Breadboard zapojení

4 Implementace

4.1 Snímání stisknutí tlačítka

Pro zachytávání stisknutí tlačítek jsem využil přerušení GPIO a implementoval obslužné rutiny pro každé tlačítko. Při detekci stisknutí tlačítka je vytvořena fronta, do které je zaslán signál o stisku. Implementoval jsem debounce mechanismus pro minimalizaci rušení signálů. Obslužné rutiny jsou navázány na přerušení a posílají události do fronty pro další zpracování. Obě tlačítka mají vlastní frontu s přerušeními.

4.2 Práce s NVS

Pro ukládání a načítání dat (uložení ujeté vzdáenosť a času) jsem využil Non-volatile Storage (NVS). Implementoval jsem funkce pro ukládání a načítání hodnoty ujeté vzdáenosť a času stráveného

na kole. Při spuštění aplikace se nejprve zjišťuje, zda jsou data v NVS dostupná, a pokud ne, inicializuje se jejich hodnota. Do NVS se ukládá vzdálenost a čas běhu tachometru. Pro usnadnění zapisování a získání hodnot jsem implementoval funkce `get_km_traveled()`, `store_km_traveled()` a to samé i pro čas.

4.3 Výpočet rychlosti a ujeté vzdálenosti

Pro výpočet rychlosti jsem využil údajů o oběhu kola a času trvání oběhu. Vypočtená rychlosť je následně aktualizována a zobrazována na displeji. Ujetá vzdálenost se aktualizuje na základě oběhu kola a je ukládána do NVS pro zachování hodnoty i po vypnutí a zapnutí zařízení.

4.4 Displej informací

Implementoval jsem logiku pro zobrazování informací na displeji. Displej zobrazuje aktuální rychlosť, ujetou vzdálenost a průměrnou rychlosť. Uživatel může tlačítkem měnit zobrazované informace na displeji. Zvolil jsem tento přístup pro jasnou přehlednost, kdy cyklista může bez použití očí, stlačit naslepo levé tlačítko a následě se jen podívat na hodnotu na tachometru.

4.5 Strukturování kódu

Celý kód je strukturován do různých funkcí pro jednodušší správu a čitelnost. Každá část kódu má komentáře vysvětlující svou funkcionality a účel pro zlepšení čitelnosti.

Hlavní funkce

- `app_main()`: Hlavní funkce obsahuje inicializaci, konfiguraci periferií a vládání aplikace. Zde se provádí inicializace NVS, konfigurace displeje SSD1306 SPI, vytvoření a spuštění tasků pro různé úkoly a nastavení tlačítka.

Funkce pro práci s NVS (Non-Volatile Storage)

- `store_km_traveled(float km_traveled)`: Ukládá hodnotu ujeté vzdálenosti do NVS.
- `get_km_traveled(float *km_traveled)`: Načítá hodnotu ujeté vzdálenosti z NVS.
- `store_time_spent(uint64_t time_spent_ms)`: Ukládá hodnotu stráveného času do NVS.
- `get_time_spent(uint64_t *time_spent_ms)`: Načítá hodnotu stráveného času z NVS.

Funkce pro obsluhu GPIO přerušení

- `gpio_interrupt_handler1(void *args)`: Obsluha přerušení pro tlačítko 1. Přidává přerušení tlačítka 1 do fronty pro další zpracování v `Wheel_Rotation_Task(void *params)`.
- `gpio_interrupt_handler2(void *args)`: Obsluha přerušení pro tlačítko 2. Přidává přerušení tlačítka 2 do fronty pro další zpracování v `Second_Button_Handle(void *params)`.

Tasky (úkoly)

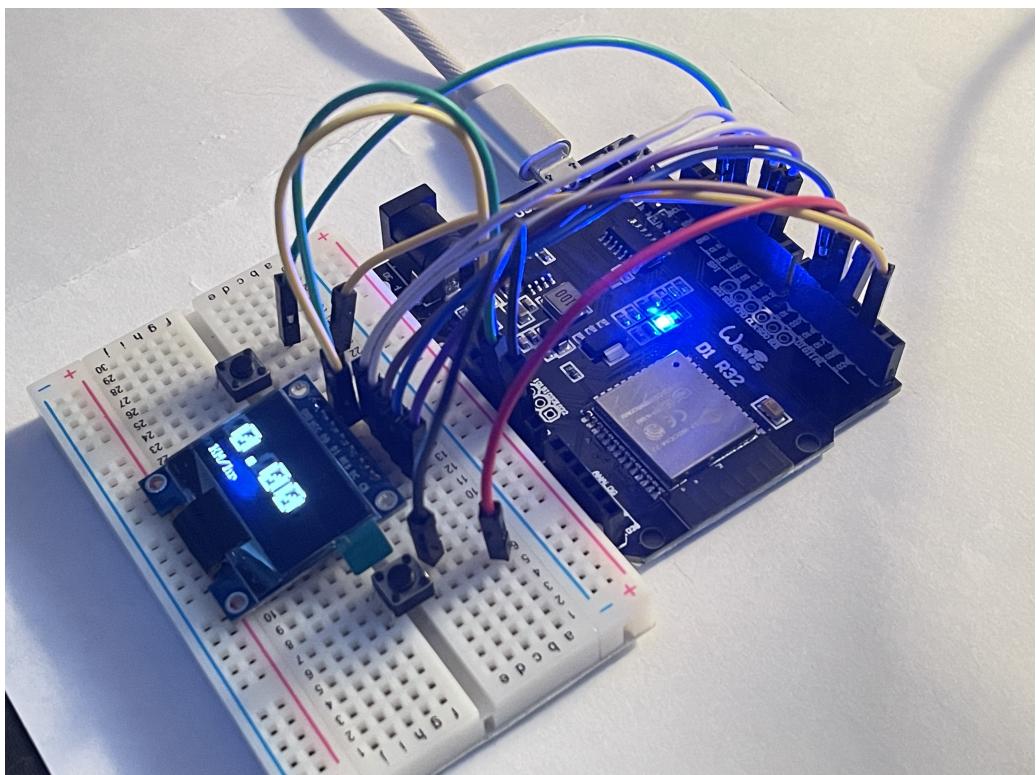
- `Wheel_Rotation_Task(void *params)`: Task pro výpočet rychlosti a ujeté vzdálenosti na základě rotace kola.
- `Second_Button_Handle(void *params)`: Task pro obsluhu tlačítka 2, který reaguje na krátké a dlouhé stisky. Při krátkém přepíná displeje a při dlouhém nuluje vzdálenost a čas.

Timer callback funkce

- `calculateSpeed(TimerHandle_t xTimer)`: Funkce volaná při expiraci časovače pro výpočet rychlosti. Slouží pro aktualizaci aktuální rychlosti mezi otočením celého kola.

5 Schéma Zapojení

Obě tlačítka jsou připojeny ke GND a GPIOP. Pravé tlačítko je k GPIOP 16 a levé k GPIOP 26. Displej je zapojen VCC - 3.3V; GND - GND; D0 (SCLK) - IO18, D1 (MOSI) - IO23, CS - IO5, DC (DATA/control) - IO27, RESET - IO17.



Obr. 2: schéma zapojení