Teplotní čidla DS18B20 a NTC

1 Zadání

- Inicializujte 1-wire sběrnici a teplotní čidlo DS18B20 s pomocí dodané knihovny. Inicializujte sedmisegmentový LED displej. Periodicky spouštějte převod teploty a vypisujte ji na displeji.
- Nakonfigurujte ADC pro čtení z NTC teplotního čidla typu NTCC-10K. Na základně údajů v datasheetu tohoto čidla vytvořte MATLABový skript pro generování lookup tabulky, která bude indexována hodnotou z AD převodníku a bude obsahovat přepočtenou skutečnou teplotu.
- Do kódu doplňte funkci pro přepínání zobrazovaného údaje. Tlačítkem S2 se rozsvítí LED1 a zobrazí údaj z NTC, tlačítkem S1 se rozsvítí LED2 a zobrazí údaj z DS18B20.

2 Návod

2.1 Základní seznámení

- Vytvořte si pracovní kopii svého repozitáře z Githubu (Git Clone), příp. aktualizujte repozitář ze serveru (Git Pull).
- Založte nový projekt přes File / New / STM32 Project / Board Selector / NUCLEO-F030R8. Budeme využívat HAL knihovny, proto ponechte Targeted Project Type na STM32Cube. Potvrďte inicializaci všech periferií do výchozího nastavení.
- Driver pro sedmisegmentový LED displej využijte z předchozích cvičení, příp. doplňte zobrazení desetinné tečky. Stejně tak využijte LED diody (LED1 @ PA4, LED2 @ PB0, push-pull výstup) a tlačítka (S1 @ PC1, S2 @ PC0, vstup **s aktivním pull-upem**).

2.2 Teplotní čidlo DS18B20 a 1-wire sběrnice

- Komunikaci inicializujte voláním `OWInit()`. V nekonečné smyčce main()u volejte funkci `OWConvertAll()`, následovanou čekáním (`HAL_Delay()`) po dobu převodu (`CONVERT_T_DELAY` = 750ms) a čtením výsledku pomocí `OWReadTemperature()`.
- *Soubor 1wire.h obsahuje inline funkci _delay_us(), která je vyladěná pro výchozí konfiguraci použitých vývojových kitů. Pro univerzální řešení je vhodnější tuto funkci přepsat s využitím běžného časovače s granualitou 1µs. Vzhledem ke krátkým intervalům je třeba použít přímé přístupy k registrům a busy-wait čekání. U jiných jader (M3 a vyšší) lze elegantně využít počítadlo cyklů DWT_CYCCNT jednotky DWT.*
- Pro správnou funkci je třeba definovat pin DQ @ PA10, režim výstupní **open-drain**, output level high.
- Funkce `OWReadTemperature()` vrací 0 v případě chyby, nenulovou hodnotu v případě úspěchu. Dále předává odkazem změřenou hodnotu, která představuje teplotu v setinách stupně celsia, např. hodnota 1208 odpovídá 12,08°C a bude na LED displeji zobrazena jako „12.1“. Deklaraci funkce naleznete v souboru 1wire.h:

```
extern uint8_t OWReadTemperature(int16_t *temperature);
```

- Použití funkce tedy pro bude vypadat např. takto:

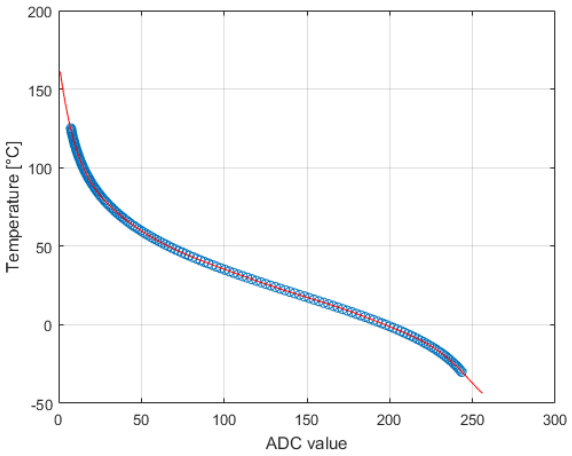
```
int16_t temp_18b20;
OWReadTemperature(&temp_18b20);
```

- Proveďte commit pracovní kopie.

2.3 Teplotní čidlo NTC a lookup tabulka

- Teplotní závislost odporu NTC teplotního čidla NTCC-10K není lineární. NTC je popsán nominálním odporem 10kΩ při teplotě 25°C a konstantou B=4050 K. Datasheet udává odpor čidla pro vybrané teploty. Pro použití v mikrokontroléru je nejvýhodnější mít v programové paměti uloženou tabulku, která bude indexována hodnotou z AD převodníku a bude přímo udávat skutečnou teplotu čidla. Takovou tabulku je nejsnadnější připravit v MATLABu (nebo Pythonu).
- Hodnoty odporu pro teploty od -30 do +125°C lze nalézt v datasheetu NTCC-10K. Tyto údaje jsou nachystané v souboru **`ntc.csv`** v eLearningu, připravené k načtení funkcí **`csvread()`** v MATLABu.
- Z hodnoty odporu *r* při teplotě *t* je třeba vypočítat údaj na AD převodníku. Jedná se o jednoduchou matematiku, čidlo NTC je zapojeno jako odporový dělič s rezistorem R5 (10k), napájení 3,3V je zároveň referencí ADC, takže se ze vztahu vykrátí. Rozsah ADC je maximálně 2^{12} = 4096, rozlišení je ale zbytečně vysoké (velikost tabulky, tolerance měření), proto budeme pracovat s rozsahem 2^{10} = 1024. Tímto výpočtem získáte vektor hodnot *ad*.
- Hodnoty si pomocí funkce **`plot()`** vykreslete do grafu (závislost *t* na *ad*) a následně je proložte polynomem dostatečně vysokého řádu:

```
p = polyfit(ad, t, 10);
```



- Následně vytvořte vektor ADC hodnot *ad2* (bude představovat indexy pole) a vypočítejte hodnoty *t2* zjištěného polynomu (budou představovat teplotu zaokrouhlenou na desetiny stupně pro tyto indexy). Výsledek dokreslíme do grafu:

```
ad2 = 0:1023;
t2 = round(polyval(p, ad2), 1);
hold on, plot(ad2, t2, 'r');
```

- Posledním krokem je uložení hodnot pole ve vhodném formátu, v případě konverze pro jazyk C tedy hodnot oddělených čárkou, formát v desetinách stupně:

```
dlmwrite('data.dlm', t2*10, ',');
```

- Implementace v C už je jednoduchá. Je třeba definovat statické konstantní pole typu **`int16_t`**, ve kterém budou hodnoty okopírované ze souboru **`data.dlm`**. Přečtená hodnota z ADC bude použita pro indexaci.
- ADC vstup NTC termistoru je ADC_IN1 @ PA1, rozlišení převodníku **10 bitů**, kontinuální konverze, přepis při přetečení. Pro jednoduchost není nutné používat přerušení ani DMA. V inicializaci je třeba spustit kalibraci a následně kontinuální převod:

```
HAL_ADCEx_Calibration_Start(&hadc);
HAL_ADC_Start(&hadc);
```

- Vyčítání AD hodnoty se provádí přes **`HAL_ADC_GetValue(&hadc)`**.
- Proveďte commit pracovní kopie, **včetně použitého `.m` souboru** pro generování lookup tabulky.

2.4 Přepínání údajů na displeji

- Doplňte dle zadání, řešte pomocí stavového automatu:
  - tlačítkem S2 se rozsvítí LED1 a zobrazí údaj z NTC
  - tlačítkem S1 se rozsvítí LED2 a zobrazí údaj z DS18B20
- Spuštění převodu čidla DS18B20 nesmí být voláno častěji než po 750 ms.
- Po delším provozu budou obě čidla vlivem ohřevu od okolí ukazovat teplotu cca o 3°C vyšší, než je teplota v místnosti.
- Proveďte commit pracovní kopie do Gitu, uložte repozitář pomocí Git Push.