PROTOKOL PRAČKA – ONDŘEJ NEDOJEDLÝ

ZÁDÁNÍ

Je zadáno vytvořit program pro **STM32F407VGT6U** s užitím **RTOS-RTX4**, jež bude fungovat jako jednoduchá „pračka“. Po programu je požadováno uživatelské menu, ve kterém si uživatel volí ze tří různých časových programů praní (60s, 90s, 120s) pomocí keypadu a LCD displaje. Po volbě pracího programu, je spuštěn „tásk“ jednoho pracího režimu (režimy: PRESOAK, WASH, RINSE, DRY); prací režim tvoří procentuální část celého časového pracího programu (respektive: 10 %, 50 %, 30 %, 10 %). Jeden prací režim navazuje na další. Vázání režimů je nutno provésti užitím „tásků“, jež budou volat „tásky“, které po skončení volají další následující „tásk“ pracího režimu; nota bene je nutno tasky mazat. Na prvním řádku LCD displeje během pracího programu je zobrazen čas do konce aktuálního pracího režimu. Druhý řádek zobrazuje čas do konce celého pracího programu. Po dokončení posledního pracího režimu, je prací program ukončen. Při dokončení všech procedur, je na LCD despleji zobrazena uživatelská nabídka.

TEORETICKÝ ROZBOR

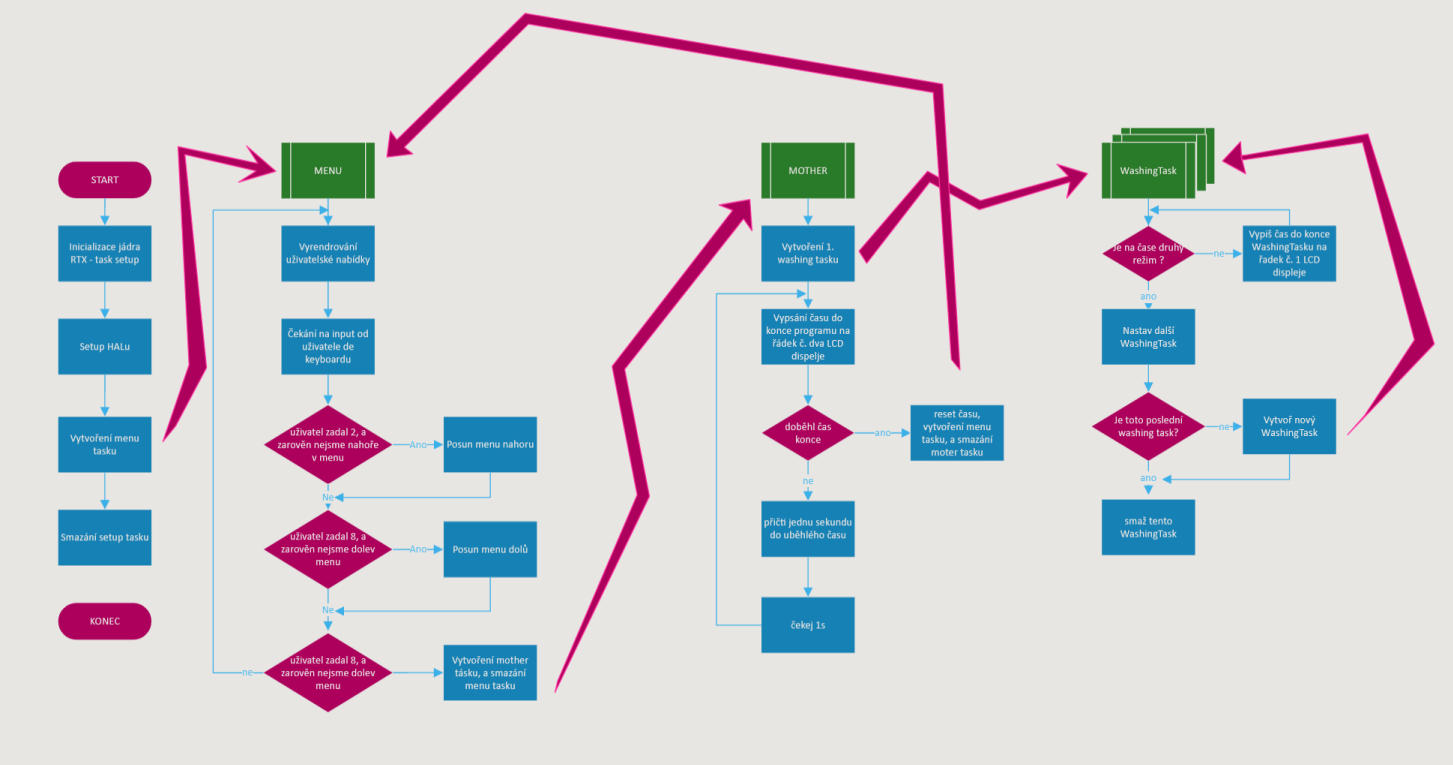
Popis HW

Jak avizováno výše, jedná se o přípravek STM32F407VTG6U od firmy STMicroelectronics. Přípravek je 32-bitový Arm® Cortex®-M4 architektury RISC, přípravek je zaobalen ve STM32F4-DISCOVERY kitu K přípravku **STM32F407VGT6U** jsme připojili LCD displej, konkrétně LCD 1602 verze 1.3, jak je z názvu zřejmé, jedná se o displej šířky 16 znaků a výšky 2 znaky. Další z periferií jest keypad, kde se jedná o čtyři řádky a tři slopce.

Popis SW

Real Time Operační Systém je druh operačních systému který nám umožňuje práci s kriticky náročnými požadavky, jelokož u GPOS není zaručeně dáno (kromě IRS), že se činnost provede do času *k*. RTOS definuje tasky(Někdy je "task" chápán jako synonymum ke slovu "thread"), které jsou zaměňovány dle R-R algoritmu.

Vývojový diagram



Vývojový diagram 1

ŘEŠENÍ

Zdrojový kód s komentáři

Vzhledem k velikosti kódu je kód umistěn v příloze "src.c".

Jako přímo komentovaný kód je zde vložen task jednoho pracího „tásku" (obr. 1). Ve zdrojovém kódu můžeme vidět, do kdy běží jeden prací režim, a to do doby než-li globální proměnná `g_timePassed`

```
__task void washingTask(void* argv) {
    size_t option = *(size_t*)argv;
    char buff[LCD_COLS+1] = {0};

    while(1) {
        if(g_washingCycleRelative[option] > g_timePassed) {
            size_t min = (g_washingCycleRelative[option]-g_timePassed)/60,
                   sec = (g_washingCycleRelative[option]-g_timePassed)%60;

            snprintf(buff, LCD_COLS, "%c %02zi:%02zi",g_washingCycles[option].name, min, sec);
            printLCD(buff, LINE1);
        } else {
            (*(size_t*)argv)++;
            if(option < 3)
                os_tsk_create_ex(washingTask, 0, argv);
            os_tsk_delete_self();
        }
    }
}
```

Obrázek 1

bude větší než relativní časová stopa daného režimu, která říká kdy od začátku programu se má program přerušit. Díky tomu že užíváme jeden cyklus pro počítání času, se nám dostává jednotnost času, jak pěkné.

ZÁVĚR

Hodnotí se program „pračky“ jakožto úspěšný, podařilo se split všechny body zadání. Jakožto největší problém a obtíž, byl v avizovaném postupném vytváření „tásků“, kde se nedařilo několik hodin vyřešit zdánlivě jednoduchý problém. Možné vylepšení kódu je kód per se; ve zdrojovém kódu lze nalézt několik možných optimalizací a celkového zjednodušení kódu. Ve vylepšení rozšíření lze uvést jako možnost:

- více pracích programů, ergo režimů;
- žádost o vložení pracího prášku;
- kontrola uzavření dveří;
- počítání spotřeby energie.