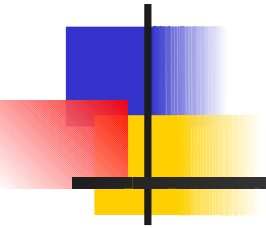


Vložené řídicí systémy
Univerzální řídicí deska M-Board

Ondřej Ježek
ZČU v Plzni, FAV, KKY

Obsah

- Programování embedded systémů
  - Struktura ES
  - Požadavky na ES
  - Programování
  - Operační systém FreeRTOS
- M-Board Univerzální řídicí deska
  - Úvod
  - Programování
- M-Board SDK
  - Popis
  - Příklady použití

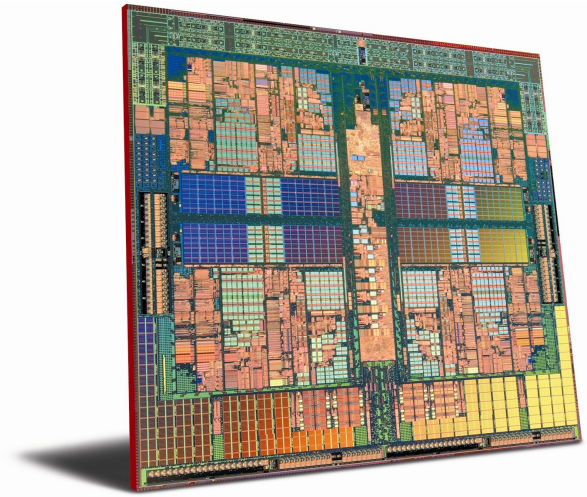
Obráběcí stroje

Požadavky ES

- Systémy reálného času
  - Nestačí jen získat správný výsledek, je důležité ho získat ve správném čase.
- Jednoúčelové systémy
- Omezené zdroje pro programování
  - Paměť
  - Energie
  - Výpočetní výkon

Struktura embedded systémů

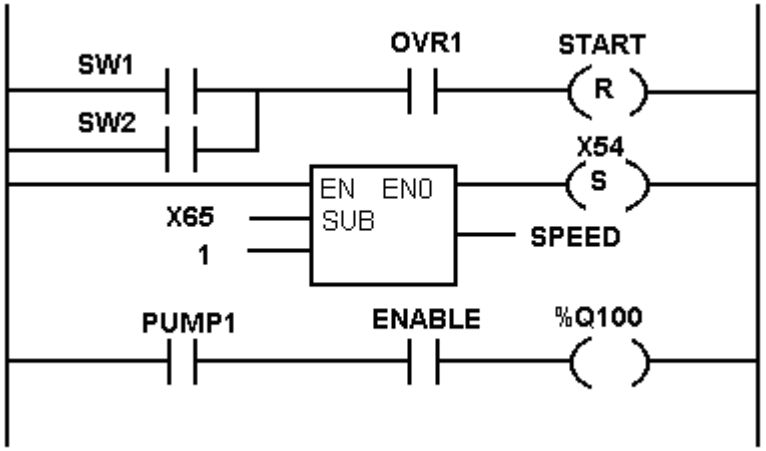
- Výpočetní jednotka
  - Slouží k provádění funkce zařízení
  - Mikropočítač, mikroprocesor nebo velká výpočetní jednotka, v závislosti na rozsahu systému.

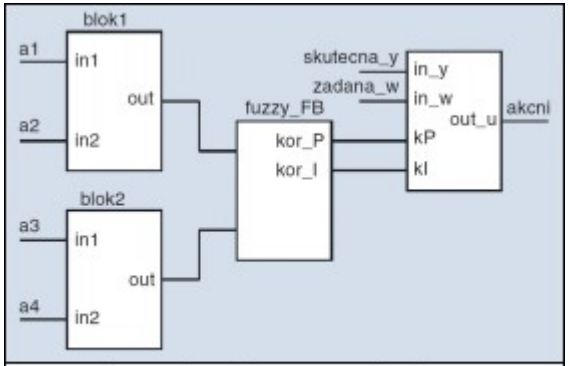


  - Periferie
    - Slouží k připojení do zařízení, ve kterém je jednotka vložena.
    - Relé, A/D nebo D/A převodníky, komunikace (RS232, Ethernet, Zigbee)

Programování ES - jazyky

- Programovací jazyky
  - Assembler
  - **C/C++**, Ada, Java a další..
  - PLC - Bloková schémata, žebříčkové diagramy
  - Rex





Obr. 1. Příklad začlenění bloku fuzzy řadiče do programu v grafickém jazyce blokových schémat (FBD)

Programování ES – operační systémy

- Operační systémy
  - Linux
  - Embedded windows
- Operační systémy reálného času
  - FreeRTOS
  - Windows CE
  - VxWorks
  - uC/OS-II
  - uCLinux
  - A spoustu dalších...

Programování ES - design

- Jednoduchá smyčka
  - Jednoduché na programování
  - Problém dodržet časové požadavky
  - Problém s odezvou na události
- Řízení přerušeními
  - Při správném návrhu dobrá odezva na události
  - Komplexnější systém se stává složitým na návrh
  - Provázanost přerušení
  - Problém synchronizace

Programování ES - design

- Offline plánovač
  - Plná kontrola toho co systém bude dělat
  - Potřeba větších zásahů v případě změny
- Operační systém
  - Snadná rozšiřitelnost
  - Nedodržení časových požadavků
- Operační systém reálného času
  - Možnost plánovat podle časových požadavků
  - Snadná rozšiřitelnost
  - Systém sám spotřebovává nějaké prostředky

Programování ES

- Offline plánování
- Online plánování
  - Statické
    - Rate monotonic
    - Deadline monotonic
  - Dynamické
    - Earliest deadline first
    - Rate monotonic + dědění priorit

FreeRTOS

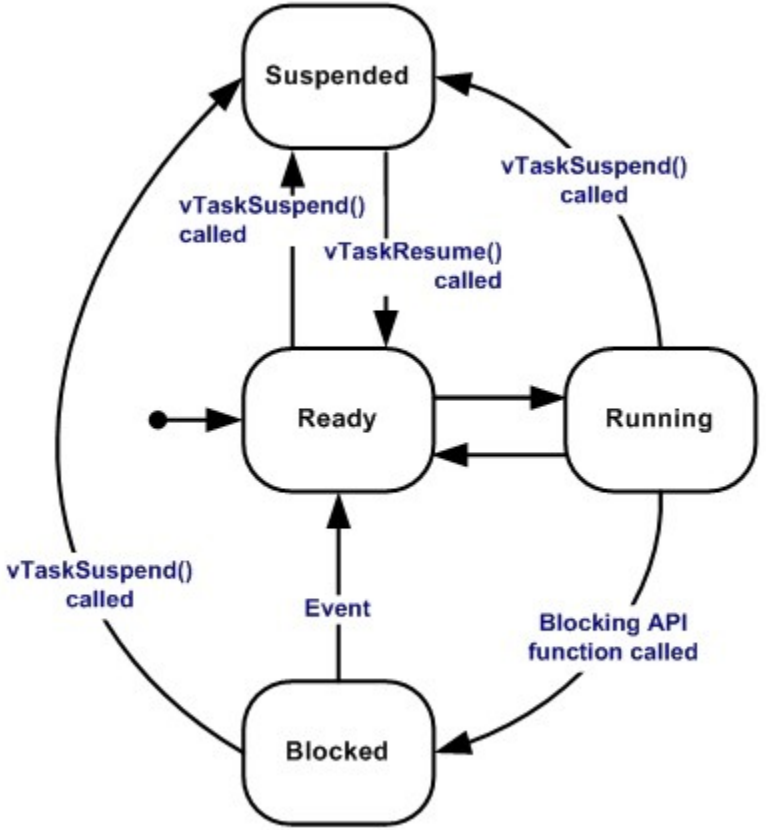
- Open source RTOS

- Reálně hojně využívaný v průmyslu
- Podporuje statické plánování (RM, DM)
- Oficiální port pro 31 architektur mikrokontrolerů
- GNU GPL licence s vyjímkou
  - Není nutné zveřejňovat vlastní kód, pouze změny samotného FreeRTOS

- www.freertos.org

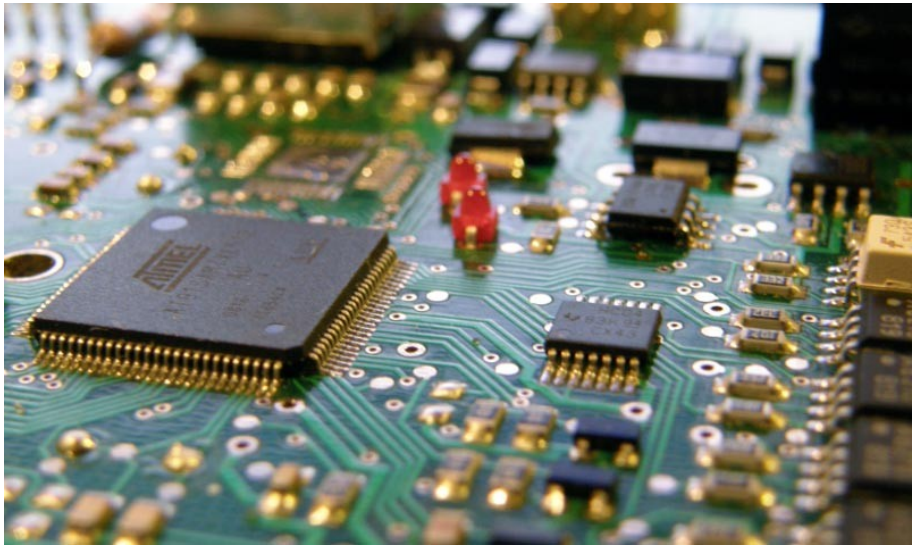
FreeRTOS

- Multitasking
  - Preemptivní
  - Kooperativní
- Komunikace úloh
  - Binární semafory
  - Čítací semafory
  - Mutex
  - Fronty



M-Board – Úvod

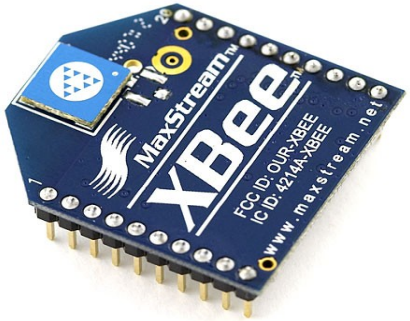
- Systém pro vložené řízení vytvořen na KKY a KAE
- Navržen pro
  - Výukové účely
  - Testování algoritmů na reálném systému
  - Aplikaci v reálných problémech řízení



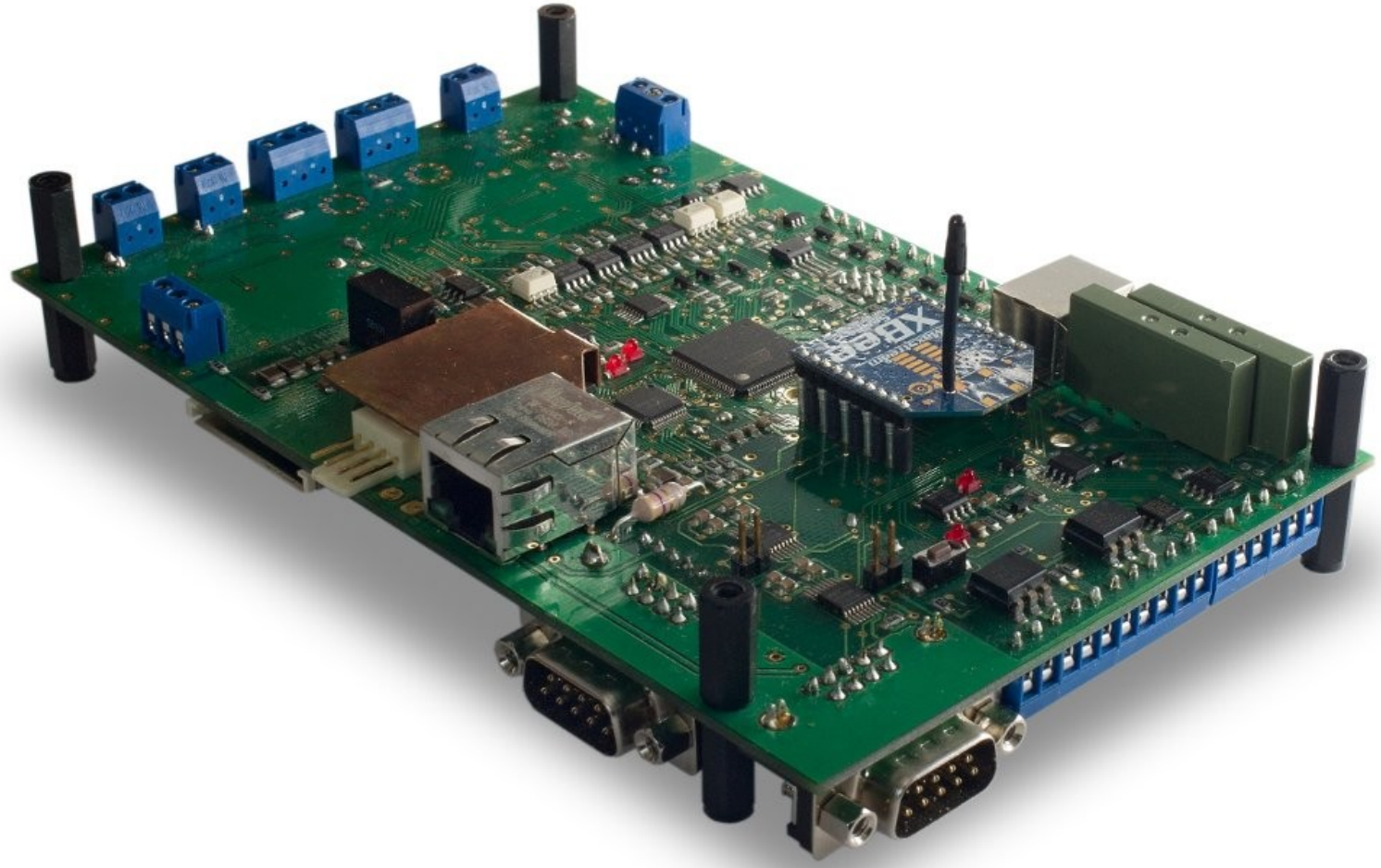
- Výuka v předmětu VŘS
- Stále ve vývoji

M-Board – Požadavky

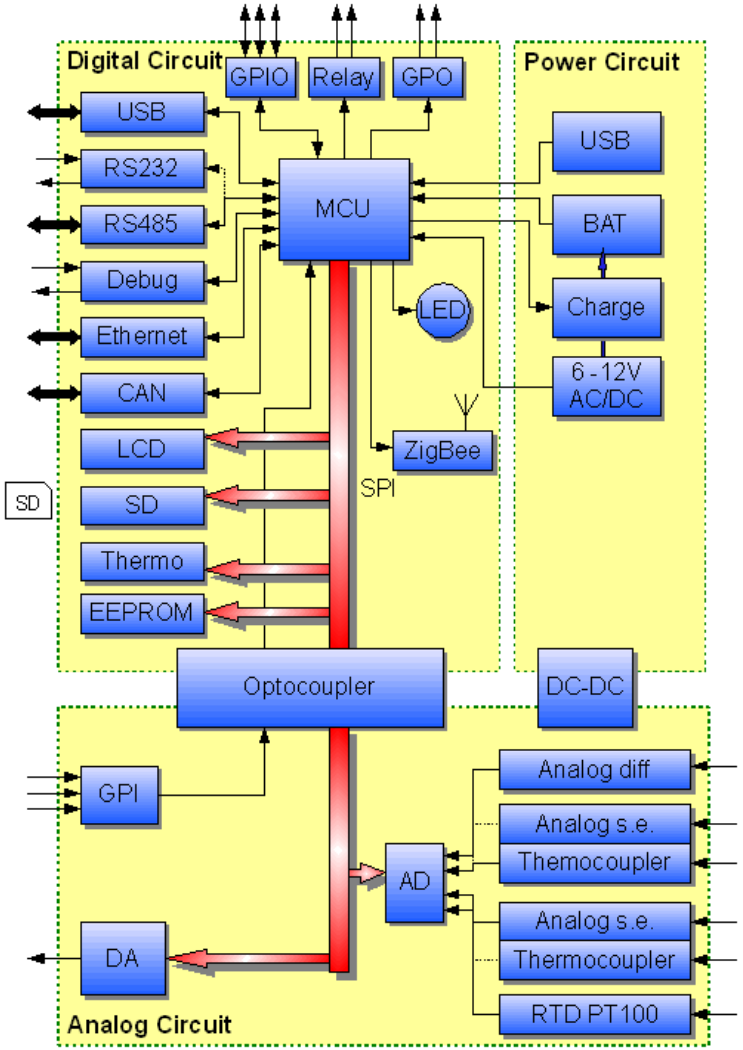
- Zařízení vzniklo z několika požadavků, které se v komerčních aplikacích nevyskytovaly
  - Bezdrátovou komunikaci
  - Řídicího systému Rex
  - Přímého připojení do procesu



M-Board



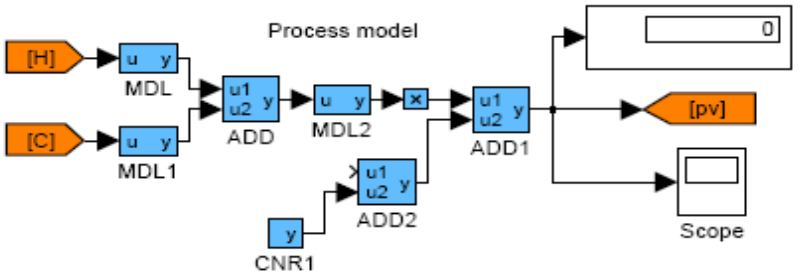
M-Board – Realizace



- Napájecí část
  - Napájení z více zdrojů
  - Možnost nabíjení baterie
- Digitální část
  - Komunikace
  - Digitální V/V
  - Paměti
- Analogová část
  - A/D převodníky
  - D/A převodník
  - Oddělené V/V

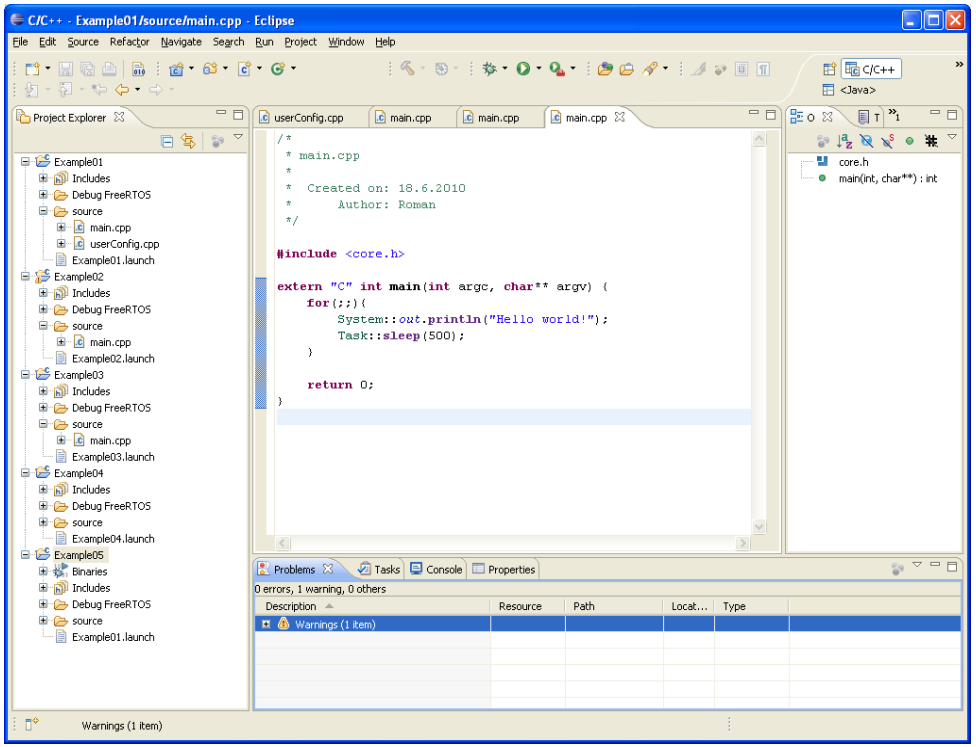
M-Board – Programování

- Programování pomocí SDK (Software Developement Kit)
  - Pro potřeby programování se vyvíjí SDK
  - Funkce pro obsluhu zařízení bez nutnosti znalosti hardware
- Programování automaticky generovaným kódem
  - Firmware lze vytvořit graficky z bloků přímo v řídicím systému Rex a potom zkompilovat.



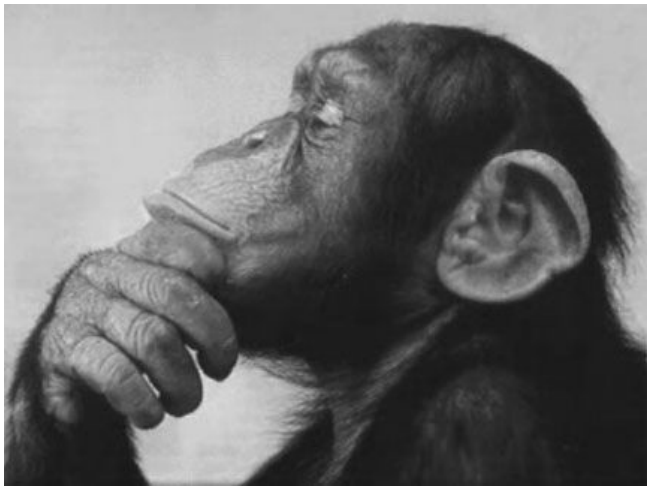
M-Board SDK

- Využívá GNU prostředky a Eclipse
- Kompletní uživatelské prostředí
- Programování v C++
- Podpora ladění
- Podpora multitaskingu
  - abstraktní vrstva
  - interně použit FreeRTOS
- Podpora periferií



M-Board SDK

- M-Board SDK je knihovna
  - Základní systém
  - V/V systém
  - Pomocná knihovna
- M-Board SDK je vývojové prostředí založené na Eclipse



M-Board SDK - Základní systém

- Úlohy
- Synchronizace
  - Kritická sekce
  - Semafor
  - Mutex
  - Atomické operace
- Jedná se o C++ nadstavbu nad FreeRTOS

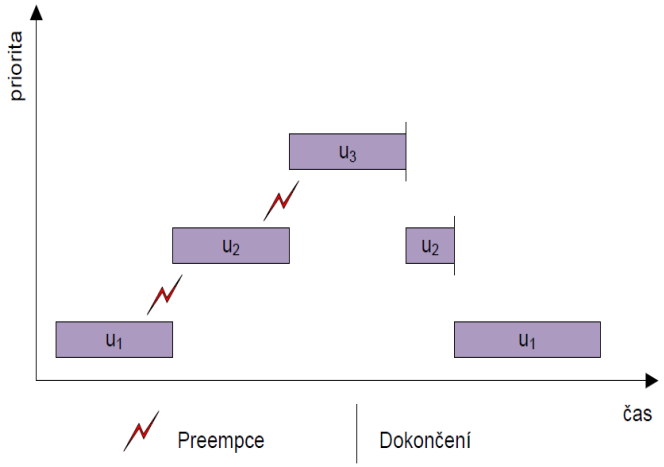
Úloha(Task)

- Základním stavebním kamenem systému M-Board třída `Task`
- Všechny úlohy se odvozují od třídy `Task`

```
class MyTask : public Task {
    public:
        // Konstrutor ulohy
        MyTask()
        : Task( // Volani konstruktoru zakladni ulohy
              1, // Priorita ulohy
           4096, // Velikost zasobniku
           true, // Uloha je vytvorena oddelene
   "MyTaskName" // Nazev ulohy
           ) {}; // Prazdne telo metody
    protected:
        // Kod ulohy
        virtual void execute() {
        ...
        };
};
```

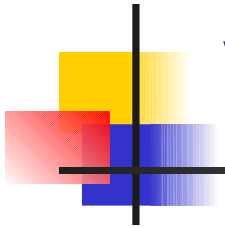
Úloha (`Task`)

- Prioritní preemptivní plánování
- Životní cyklus úlohy
  - Joined
  - Dettached
- Úlohy pracují se společným paměťovým prostorem
- Synchronizují se pomocí synchronizačních objektů



Synchronizační objekty

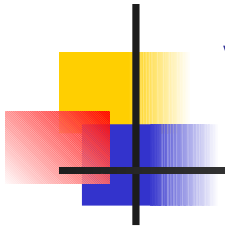
- Kritická sekce
  - Blokuje přerušení a nebo i úlohy s vyšší prioritou
- Semafor
  - Počítací semafor
- Mutex
  - Binární semafor pro zajištění vzájemného vyloučení
  - Řeší problém inverze priorit
- Atomická operace
  - Atomické operace inkrement dekrement
  - Využívají vlastností procesoru



V/V Systém

- Založen na principu datových proudů
- Každá třída proudu odvozena od základní třídy stream

Metoda	Popis
available()	Počet dat ke čtení bez blokování
flush()	Vypuštění mezipaměti výstupu
read()	Čtení sekvence bytů
write()	Zápis sekvence bytů
control()	Provedení V/V operace
copyFrom()	Kopírování z jiného streamu



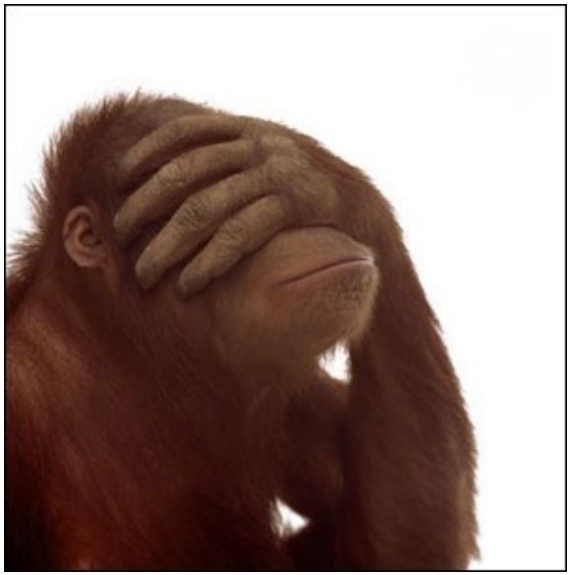
V/V systém – standardní V/V

- Pro základní komunikaci se používá standardní V/V
- Defaultně je přesměrován na sériovou linku

```
System::out.println("Blah blah");
System::out.print("This is ").print(4).print(" fun\n");

String s = System::in.readLine();
```

Pomocná knihovna

- Speciální funkce pro práci s nestandardními ovladači
- Třídy bufferů
- Třídy proudů
- Třídy pro práci s řetězci
  - Převody na číselnou hodnotu
  - Spojování řetězců



Příklad 1 - „Hello World!“

```
#include <core.h>

extern "C" int main(int argc, char** argv) {
    for(;;){
        System::out.println("Hello world!");
        Task::sleep(500);
    }
}
```

- Jednoduchý výpis na sériovou linku
  - Každých 500ms
  - Funkce main už je vlákno

Příklad 2 - „Multitasking“

```
class DaTask: public Task{
public:
    DaTask(int priority);
protected:
    virtual void execute();
};
```

```
extern "C" int main(){
    daTask = new DaTask(3);
    daTask->start();
    for(;;){
        Task::sleep(500);
        System::out.println("Multi");
    }
}
```

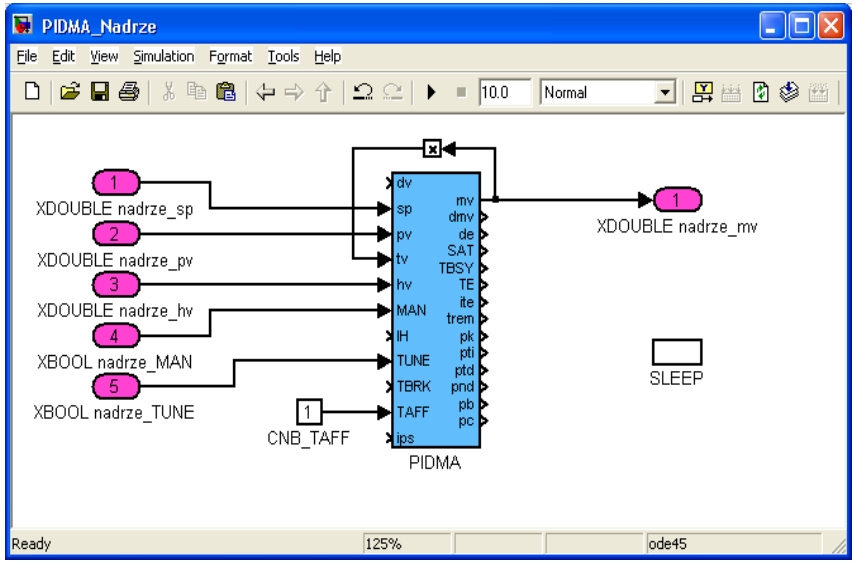
```
void DaTask::execute(){
    da = daOpen(DAC_0,spi);
    uint16_t voltage = 0;
    for(;;){
        voltage += 10;

        daWriteRaw(da,voltage);

        if(voltage > 0x1FFF){
            voltage = 0;
        }
        Task::sleep(10);
    }
}
```

Příklad 2 - „Multitasking“

- Definování vlákna DaTask
  - Vytváří pomalý pilový výstup na DA převodníku
- Spuštění vlákna z main
  - Main je také vlákno, demonstrováno periodickým výpisem Multitasking na sérivou linku.

Příklad 3 - „Mikrorex“

- Použití automaticky nastavitelného PID regulátoru




```
/* Control Algorithm functions */
extern int InitControlAlgorithm();
extern int MainControlAlgorithm();
extern int ExitControlAlgorithm();

/* Input variables */
extern XDOUBLE nadrze_sp;
extern XDOUBLE nadrze_pv;
extern XDOUBLE nadrze_hv;
extern XBOOL nadrze_MAN;
extern XBOOL nadrze_TUNE;

/* Output variables */
extern XDOUBLE nadrze_mv;
```

Příklad 3 - „Mikrorex“

```
extern "C" int main(int argc, char** argv) {

    spiDevice = new spi();
    dacDevice = new dac(spiDevice);
    adcDevice = new adc(spiDevice);

    while (loop) {
        clock_t lastWakeTime = System::getTickCount();

        pAD = adcDevice->adMeasureMultiple(XTRUE, AD_MASK);
        nadrze_pv = (XDOUBLE) ((*pAD)[CH_IN2]);

        MainControlAlgorithm();

        dacDevice->daWrite ((float) nadrze_mv); // Nastaveni vystupu

        System::sleepUntil(lastWakeTime,period);

     }
     ExitControlAlgorithm();

     return 0;
}
```

Příklad 3 - „Mikrorex“

- Mikrorex generuje C soubor, který se připojí k programu
- K programu je třeba připojit i knihovnu obsahující bloky
- Algoritmus je pak třeba ručně inicializovat a spustit

Současná verze

- Současná verze M-SDK 1.1
  - Abstraktní OS
  - Ovladač RS232
  - Ovladač SPI
  - Ovladač AD
  - Ovladač DA
  - Ovladač DIO (včetné relé)

M-Board informace

- Informace o desce jsou ke stažení na http://www.rexcontrols.cz/devzone
  - Návody na instalaci SDK
  - Návody na kompilaci příkladů
  - Návody na programování
- Operační systém FreeRTOS na http://www.freertos.org

Ukázka