TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

DIPLOMOVÁ PRÁCE

LIBEREC 2011 David Vápenka

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: N2612 Elektrotechnika a informatika

Studijní obor: 3906T001 Mechatronika

Vývoj a realizace monitorovacího systému vakuové napařovačky

Design and realization of a monitoring system for an evaporating machine

Diplomová práce

Autor: **David Vápenka**

Vedoucí práce: Ing. Jan Václavík

Konzultant: Ing. Pavel Oupický

V Liberci

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci (BP) se plně vztahuje zákon č. 121/2000 o právu autorském, zejména §60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé BP, a prohlašuji, že **souhlasím** s případným užitím mé BP (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít svou BP či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum: 20. května 2011

Podpis:

Poděkování

Rád bych využil této příležitosti a poděkoval svým rodičům za umožnění studia a za jejich vytrvalou podporu, bez níž bych se nedostal tak daleko a to jak ve studiu, tak v osobním životě. Dále bych chtěl poděkovat Ing. Janu Václavíkovi za cenné rady, poskytnutý čas a odborné vedení.

Abstrakt

Cílem této práce je návrh a praktická realizace monitorovacího systému vakuové napařovačky. Těžištěm praktické části je tvorba ovladače krystalového monitoru SQM242 pro vývojové prostředí Control Web. Následuje návrh aplikace řídící celou technologii napařování, využívající výše zmíněný ovladač a krystalový monitor. Důraz je kladen především na funkčnost a spolehlivost vytvořeného ovládacího programu a uživatelskou přívětivost.

Celá práce je kvůli přehlednosti členěna do tematických celků. V úvodní části práce je teoretický rozbor zabývající se problematikou tenkých vrstev a především technologickými postupy jejich nanášení. Následuje vlastní návrh a praktická realizace řídicího systému. Závěr práce se zabývá testováním v reálném provozu a vyhodnocením funkčnosti a spolehlivosti vytvořeného systému.

Klíčová slova: Control Web, vakuová napařovačka, ovladač

Abstract

The aim of this work is the design and realization of a monitoring system for an evaporating machine. The crux of the practical part is creation of crystal Card driver (SQM242) for the development environment Control Web. After that follows a design of control application making use of aforesaid crystal Card driver. Emphasis is placed on functionality and reliability of the control program and created user-friendliness.

The work is structured for greater clarity into thematic units. In the first part is a theoretical analysis of thin film evaporation technology. After that follows the design and implementation of control system. The conclusion deals with the testing in real traffic and evaluation of functionality and reliability of the system created.

Keywords: Control Web, evaporating machine, driver

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek zdrojového kódu

Seznam použitých symbolů

R	[Ω]	elektrický odpor
U	[V]	elektrické napětí
I	[A]	elektrický proud
P	[W]	výkon
L	[H]	indukčnost
C	[F]	kapacita
t	[s]	čas
T	[s]	perioda
f	[Hz]	frekvence
η	[%]	účinnost
p		převod transformátoru

Seznam použitých zkratek

T_{ON}	čas, po který je spínač sepnut
T_{OFF}	čas po který je spínač vypnut
PWM	pulzně šířková modulace
ESR	ekvivalentní sériový odpor kondenzátoru
DPS	deska plošných spojů
Obr.	obrázek
Tab.	tabulka
Eq.	rovnice
SMD	technologie povrchové montáže součástek
IO	integrovaný obvod
R_Z	odpor zátěže
GND	uzemnění
DC	stejnosměrné napětí (proud)
AC	střídavé napětí (proud)
RC	RC článek
LC	LC článek

Úvod

Nanášení tenkých vrstev patří mezi relativně nové a perspektivní technologie mající využití v celé řadě oborů. Hlavním úkolem tenké vrstvy je tvorba povrchové úpravy s určitými námi definovanými vlastnostmi. Příkladem může být index lomu či propustnost určitých vlnových délek světla u optických materiálů, tvrdost či otěruvzdornost u řezných nástrojů atd. O tenké vrstvě můžeme hovořit, jedná-li se o materiál s tloušťkou od několika desítek nanometrů po několik mikrometrů, který je vytvořený na základním materiálu tj. substrátu. Substrát tak získává povrchovou úpravu s vlastnostmi odvíjejícími se od chemického složení a tloušťky tenké vrstvy.

Existuje spousta technologií nanášení tenkých vrstev založených na různých fyzikálních principech. Jak již název práce napovídá, budu se zde zabývat technologií vakuového napařování, tedy především monitorováním a řízením této technologie.

Celý monitorovací a řídicí systém je koncipován následujícím způsobem. Základní informace o tloušťce a rychlosti napařování tenké vrstvy je získávána krystalovým monitorem *SQM242*. Tento monitor je pomocí ovladače (ve formě dynamicky linkované knihovny) zpřístupněn řídicí aplikaci vytvořené ve vývojovém prostředí *Control Web*. Ovládání a řízení celé technologie pak zprostředkovávají vstupně výstupní moduly *DataLab IO*, ovládané již zmíněnou aplikací.

Cílem této práce je tedy jak tvorba ovladače, který tak bude tvořit interface mezi řídicí aplikací a krystalovým monitorem *SQM242*, tak vývoj a realizace samotné aplikace, která má za úkol technologii napařování řídit. Hlavní Důraz bude kladen na co nejvyšší míru automatizace, funkčnost a především spolehlivost výsledného řídicího systému.

1 Problematika tenkých vrstev

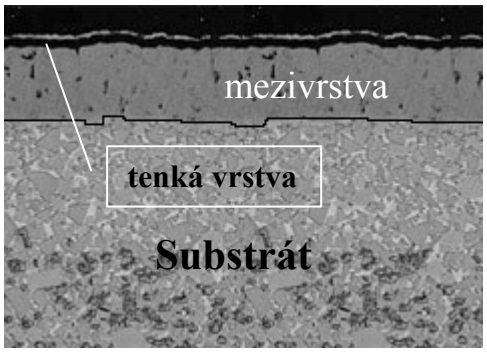
Vzhledem k tomu, že se tato práce nezabývá přímo tenkými vrstvami, nýbrž řízením technologie napařování, bude v této práci problematika tenkých vrstev popsána pouze okrajově. Tato kapitola se tedy této problematiky jen dotýká a je zaměřena především na metody depozice a způsoby měření jejich tloušťky.

1.1 Vlastnosti

Pojmem tenká vrstva se označuje materiál o tloušťce od několika nanometrů až po několik desítek mikrometrů, nanesený na základním materiálu, takzvaném substrátu. Velmi malá tloušťka tenké vrstvy má vliv na její fyzikální vlastnosti. Ty se tak liší od objemových vlastností materiálu, z kterého je zhotovena.

Rozlišení toho, zda se již jedná o tenkou vrstvu či nikoliv, závisí právě na tom, zda má tato vrstva odlišnou námi sledovanou vlastnost od objemových vlastností.

Na obrázku (Obr. 1) je vidět řez substrátem s nanesenou tenkou vrstvou. Je zde patrná struktura takovéhoto systému. Jednotlivé složky plní následující funkce.



Obr. 1: Řez systému s tenkou vrstvou

Substrát určuje tuhost, pevnost a geometrii tohoto systému. ***Mezivrstva*** má za úkol zvyšovat adhezi, působit jako bariéra rozvoje trhlin a kompenzovat dilatace a pnutí. Poslední součást systému ***Tenká vrstva*** plní funkce dle toho, za jakým účelem byla vytvořena (např. odolnost proti opotřebení, redukce tření, korozní odolnost, tepelná bariéra, antireflexní vrstva atd.).

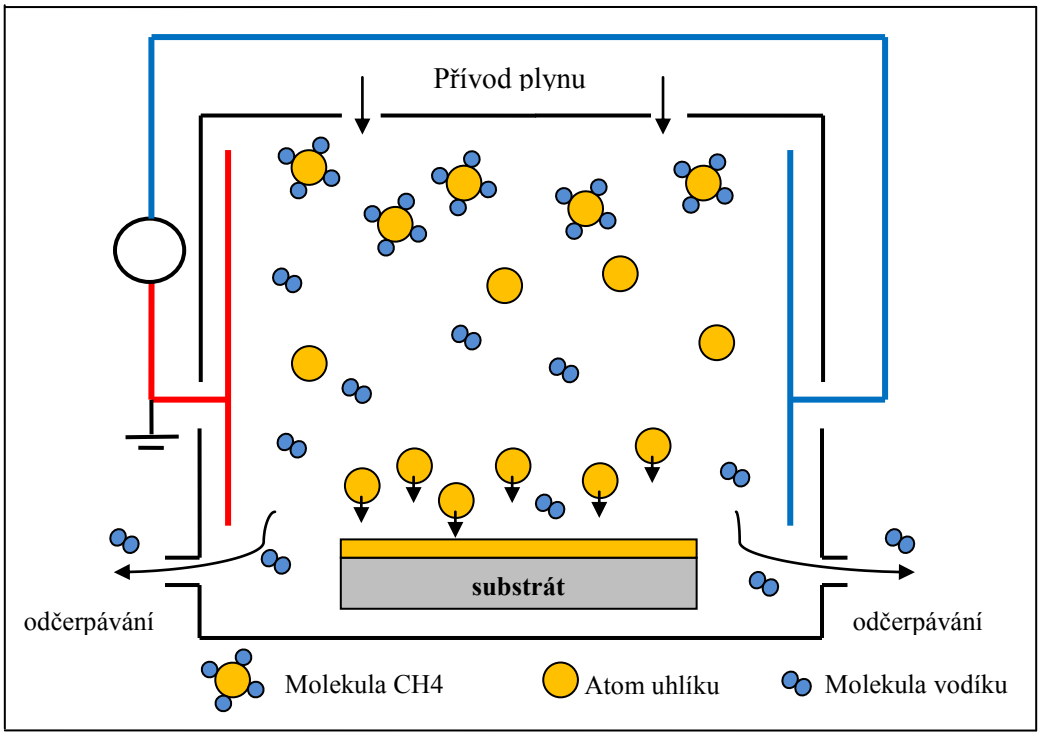
1.2 Metody nanášení tenkých vrstev

V průběhu let bylo vyvinuto spoustu rozličných metod nanášení tenkých vrstev založených na různých principech. Tyto metody lze rozdělit do několika základních skupin, z nichž uvedu tyto dvě. Chemická depozice z plynné fáze ***CVD*** (Chemical Vapour Deposition) a Fyzikální metody depozice ***PVD*** (Physical Vapour Deposition).

Hlavní rozdíl mezi CVD a PVD tkví ve způsobu přípravy vrstvy, z plynu u CVD a z takzvaného pevného terče u PVD. Jak chemické tak fyzikální metody mají své výhody a nevýhody, o nichž bude pojednáno níže [literatura].

1.2.1 Chemická metoda depozice z plynné fáze (CVD)

Jak již název napovídá, je na povrchu substrátu tenká vrstva vytvářena v důsledku chemických procesů probíhajících v objemu plazmatu. Reakční složky, takzvané prekurzory, vstupují do komory v plynné fázi, kde se rozkládají či reagují mezi sebou, za vzniku požadovaného materiálu ve formě tenké vrstvy nebo prášku na povrchu substrátu. Při této reakci se často uvolňují těkavé vedlejší produkty, které jsou z reakční komory odčerpávány vakuem či proudem plynu. Schematické znázornění této technologie je na obrázku (Obr. 2).



Obr. 2: Depozice diamantové vrstvy metodou CVD

Mezi výhody této technologie patří vysoká odolnost takto vytvořených vrstev vůči opotřebení, ekonomická výhodnost při tvorbě silných vrstev či dobrá využitelnost tam, kde je potřeba pokrýt nepřístupné dutiny a záhyby. Touto technologií lze nanést rozmanité vrstvy kovů, polovodičů a jiných chemických sloučenin (např. i diamantové vrstvy), přičemž tyto vrstvy vykazují vysokou čistotou.

Hlavní omezení této technologie spočívá ve vysoké teplotě depozičního procesu (950 – 1050 °C). CVD tak není možné použít tam, kde by vlivem teploty docházelo k degradaci substrátu. Mezi další nevýhody patří použití neekologických plynných směsí, energetická náročnost a dlouhý pracovní cyklus (8 – 10 hodin).

1.2.2 **Fyzikální metody depozice (PVD)**

Tyto metody jsou vesměs založeny na principu vypařování materiálu (vytvářejícího vrstvu) ve vakuu nebo rozprašování ve výboji udržovaném za nízkých tlaků. Obecně lze depoziční proces rozdělit do třech na sebe navazujících kroků.

- Převedení materiálu do plynné fáze
- Transport par ze zdroje k substrátu
- Vytváření vrstvy na povrchu substrátu

Způsob, jakým jsou jednotlivé kroky uskutečněny, se liší dle konkrétně použité PVD metody (budou popsány níže).

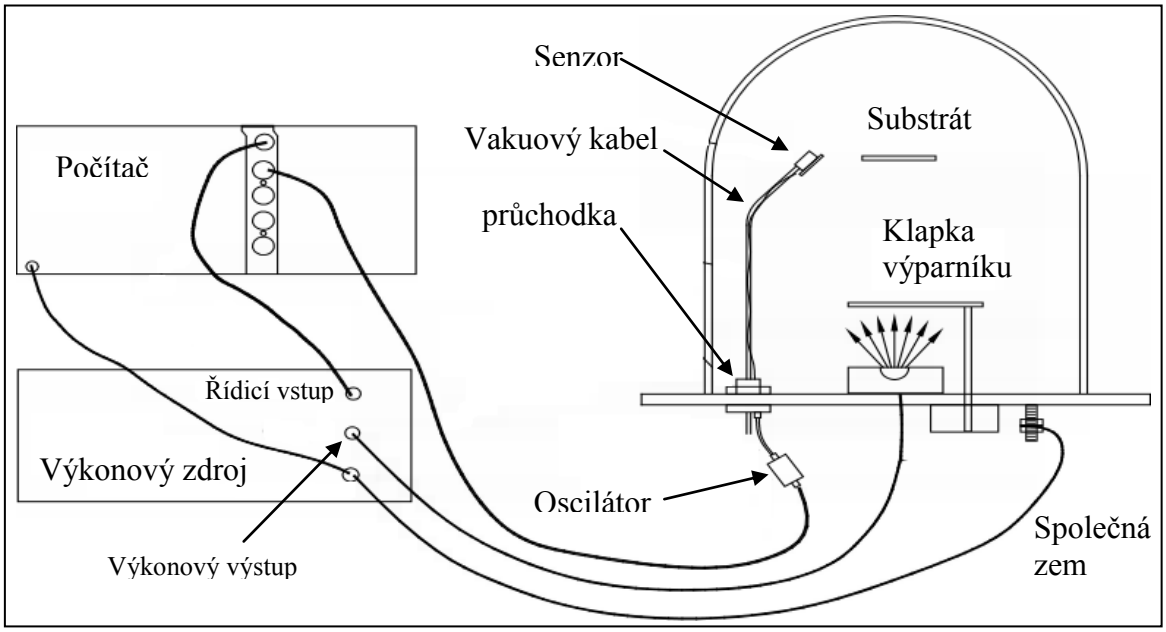
Zaměříme-li se na výhody těchto metod, je třeba v prvé řadě zmínit rozmezí teplot, při kterých se tento proces uskutečňuje. Teploty se pohybují v rozsahu 150 – 500°C, což je výrazně níže než při CVD procesu. Pomocí PVD tak lze nanášet tenké vrstvy i na substráty, na které by to pomocí CVD nebylo možné. Dalšími výhodami jsou: vysoká odolnost vrstev, nízký koeficient tření, možnost vytvořit velké množství různých kombinací vrstev a především možnost tvorby přesných tloušťek vrstev. Je také potřeba zmínit, že PVD je ekologicky nejšetrnější metodou depozice, jelikož zde nejsou používány nebezpečné materiály a při technologickém procesu nedochází k uvolňování toxických látek.

Co se týká konkrétních fyzikálních metod, zmíním zde tři nejpoužívanější. Reaktivní naprašování, vakuové napařování a reaktivní iontové plátování. Jelikož se tato práce zabývá řízením vakuové napařovačky, popíšu zde pouze vakuové napařování.

Vakuové napařování

Vakuové napařování patří mezi nejjednodušší technologie výroby tenkých vrstev. Částice nanášeného materiálu jsou uvolňovány v důsledku jeho zahřívání v uzavřeném prostoru. V tomto prostoru se ustaví rovnovážný tlak nazývaný tenze nasycených par. Je-li v tomto systému narušena rovnováha a v určitém místě je teplota nižší, dochází v tomto místě ke kondenzaci par. Tím jsou vytvořeny podmínky pro přenos materiálu z místa o vyšší teplotě (výparníku) do místa o teplotě nižší (substrát).

Ohřívání materiálu pro vypařování může být zajištěno odporovým ohřevem, iontovým svazkem, vysokofrekvenčním ohřevem atd. Celý proces probíhá ve vakuu (10^{-4} – 10^{-5} Pa) z důvodu zvětšení střední volné dráhy molekul nanášeného materiálu.



Obr. 3: Ilustrační obrázek vakuové napařovačky

Na obrázku (Obr. 3) je schematické znázornění vakuové napařovačky. Je zde vidět uzavřená vakuová komora, ve které se nachází výparník s nanášeným materiálem. Součástí výparníku je uzavíratelná klapka, která slouží k omezení jeho setrvačnosti[1]. Dále se v komoře nachází krystalový senzor tloušťky (někdy opatřen uzavíratelnou klapkou) a substrát. Vně vakuové komory se nachází oscilátor a výkonový zdroj, pomocí kterého se řídí ohřev výparníku (rychlost vypařování materiálu). Tento výkonový zdroj je řízen buď manuálně, nebo automatizovaně pomocí řídicího systému (PLC, PC atp.).

[1] vypneme li výkonový zdroj, bude se díky tepelné setrvačnosti obsah výparníku stále určitou dobu vypařovat. Uzavřením výparníku klapkou, dojde téměř k okamžitému zastavení vypařování.

Zpravidla bývá ještě vakuová napařovačka vybavena spoustou jiných zařízení, která nejsou na obrázku (Obr. 3) znázorněna (senzory vakua, vývěvy, chladící okruh, osvětlení, teplotní čidla atd.). Nicméně k pochopení principu vakuového napařování, není jejich popis nezbytný.

1.3 Snímače tloušťky

Při řízení jakékoli technologie patří snímače obecně k nejdůležitějším součástem měřícího řetězce, který vyhodnocuje aktuální stav systému či veličin na něj působících. Při řízení napařování tenkých vrstev je jedna z nejdůležitějších informací aktuální tloušťka napařované vrstvy (fyzikální vlastnosti vrstvy jsou závislé na její tloušťce). Na snímače tloušťky jsou tak kladeny vysoké nároky. Je tedy vhodné zde tuto problematiku blíže popsat.

Existuje celá řada metod pro měření tloušťky tenkých vrstev. V této kapitole budou však zmíněny pouze tyto dvě: dynamické vážení křemenným výbrusem a jedna z optických metod (konkrétně reflektivní snímání).

1.3.1 **Dynamické vážení křemenným výbrusem**

V prvé řadě je třeba zmínit, že tato metoda spadá do kategorie tzv. nepřímých metod. Krystalový senzor tak neměří tloušťku napařované vrstvy na substrátu, nýbrž na vlastním povrchu. Je-li však senzor vhodně umístěn nehraje tato skutečnost téměř žádnou roli. Co se týče polohy senzoru, je na obrázku (Obr. 3) znázorněno jedno z možných umístění.

Princip funkce krystalového senzoru je založen na piezoelektrickém efektu. Pokud je na krystal přiloženo napětí, začne kmitat s vlastní rezonanční frekvencí. Je-li teplota konstantní, zůstává frekvence kmitů stabilní. Napaříme-li na povrch krystalu tenkou vrstvu, zvýší se jeho hmotnost, což způsobí snížení rezonanční frekvence. Toto je popsáno následující rovnicí (Eq. 1)

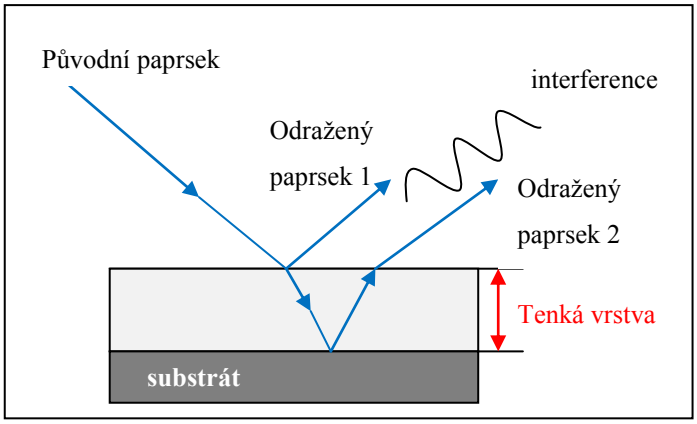
$$f = \frac{v_p}{2 \cdot t} = \frac{N}{t} \qquad \text{Rovnice 1}$$

Takovéto změny frekvence lze pak již poměrně snadno převést na tloušťku napařované vrstvy. Standardně jsou k měření frekvence, případně i k následnému převodu na tloušťku vrstvy, používány specializované měřící karty.

Jak bylo již výše zmíněno, je frekvence kmitů krystalu teplotně závislá. Je tedy nutné buď s touto závislostí počítat a v součinnosti s teplotním čidlem ji kompenzovat, nebo krystalový senzor chladit (zabraňovat změnám teploty). Je taky důležité uvést, že s tloušťkou usazenin napařených na krystalu, klesá jeho přesnost a začínají se projevovat nelinearity. V takovém případě je nutné krystal nahradit (cena se pohybuje okolo 10 $).

1.3.2 **Reflektivní snímání**

Tato metoda se stejně jako předchozí řadí k nepřímým metodám. Není tedy měřena tloušťka vrstvy napařené přímo na substrátu, nýbrž na referenčním sklíčku. Reflektivní snímání využívá princip interference na tenké vrstvě popsaný níže (Obr. 4).



Obr. 4: Interference na tenké vrstvě

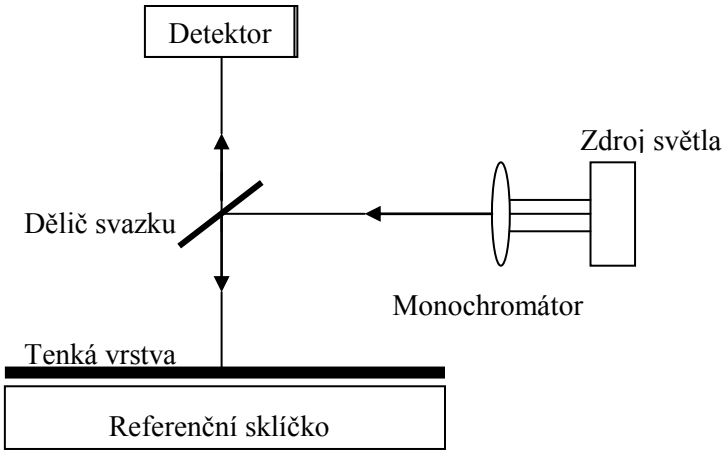
Koherentní vlnění dopadající na tenkou vrstvu, se od ní částečně odrazí a částečně prochází dál až k rozhraní tenká vrstva – referenční sklíčko. Zde dochází k dalšímu odrazu. Obě tyto odražené vlny spolu interferují, a jelikož jsou jejich trajektorie různě dlouhé, jsou vlny navzájem fázově posunuty. V závislosti na velikosti fázového posunutí, potažmo tloušťce vrstvy, dochází k zesílení či zeslabení jejich intenzity (destruktivní či konstruktivní interference)[2]. Tento jev je popsán interferenční rovnicí (Eq.2), kde $\varphi = \varphi_2 - \varphi_1$ označuje fázový rozdíl.

$$I = I_1 + I_2 + 2(I_1 \cdot I_2)^{1/2} \cdot \cos\varphi \qquad \text{Eq.2}$$

Interferují li takto navíc dvě vlny na vrstvě, jejíž tloušťka v závislosti na čase roste, je zřejmé, že jejich fázový rozdíl bude časově proměnný. Z časové závislosti intenzity interferujícího světla je tedy možné zjistit tloušťku tenké vrstvy.

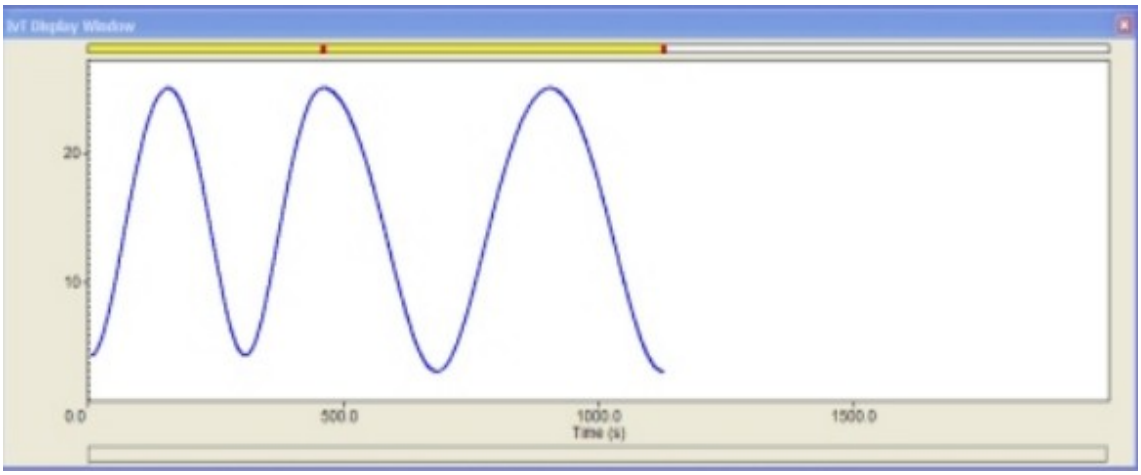
[2] Je li fázový rozdíl v rozmezí jedná se o interferenci konstruktivní (intenzita výsledné je vyšší než je součet intenzit jednotlivých vln)

Tímto se dostáváme k samotnému měření intenzity monochromatického světla. Obrázek (Obr. 5) znázorňuje princip jednoduchého reflektometru.



Obr. 5: Jednoduchý reflektometr

Světlo ze zdroje prochází přes monochromátor do děliče svazku. Zde se ho část odrazí na referenční sklíčko s tenkou vrstvou. Po dopadu se opět část světla odrazí zpět na dělič svazku, kterým odražené světlo projde a jeho intenzita je změřena detektorem a zaznamenána. Pro stanovení tloušťky vrstvy potřebujeme znát časovou závislost intenzity. Ta je zobrazena na následujícím obrázku (Obr. 6).



Obr. 6: Časová závislost intenzity

2 Krystalový monitor SQM242

Jednou ze stěžejních částí této práce je tvorba a zprovoznění ovladače měřící karty SQM242 a pod vývojovým prostředím Control web. Před započetím jakékoli práce, je tedy nutné se s touto měřící kartou podrobně seznámit. A to jak s její hardwarovou částí, tak se softwarovou podporou (interface atd.).

SQM242 je měřící karta připojitelná na PCI sběrnici. Spolu s odpovídajícími senzory a řídicím programem je schopná plnohodnotně řídit proces depozice tenkých vrstev.

2.1 Hardwarová část

2.1.1 Vstupy

Krystalový monitor je vybaven čtyřmi vstupními BNC konektory. Ty jsou určeny k připojení krystalových senzorů tloušťky. Výsledné propojení měřící karty a senzoru tvoří jakýsi řetězec složený z několika BNC kabelů, oscilátoru a speciálního vodou chlazeného pouzdra senzoru. Ostatně je to vidět na obrázku (**Chyba! Nenalezen zdroj odkazů.**). Parametry vstupů jsou uvedeny v tabulce (Tab. 1).

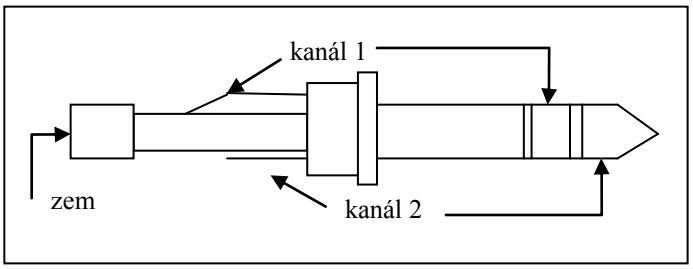
Tab. 1: parametry vstupů SQM242

Počet senzorů	4 (s aktivním oscilátorem)
Konektory	BNC
Rozsah dovolených frekvencí senzoru	1.0 – 10.0 MHz
Referenční frekvence accuracy	0.002 %
Stabilita referenční frekvence	+/- 2ppm (0 – 50 °C)
Frekvenční rozlišení	0.06 Hz (6 MHz)
Rozlišení tloušťky	0.027 A
Rozlišení rychlosti napařování	0.055 A/s

Ke vstupům karty je možné připojit jakýkoliv krystalový senzor, který frekvenčně vyhovuje podmínkám uvedeným v tabulce (Tab. 1), přičemž délka připojovacího BNC nesmí být vyšší než patnáct metrů.

2.1.2 **Výstupy**

SQM242 disponuje dvěma výstupy. Ty jsou určeny k řízení výkonového zdroje, který ohřívá materiál ve výparníku (řídí rychlost vypařování). Ač jsou výstupy dva, je na kartě patrný pouze jeden konektor. Při jeho bližším prozkoumání zjistíme, že jde o 1/4“ stereo phone jack. Tím je absence druhého konektoru vysvětlena. Uspořádání výstupů v konektoru je zřejmé z obrázku (Obr. 7).



Obr. 7: Uspořádání výstupů

Z technického pohledu je nejdůležitějším parametrem výstupů jejich napěťový rozsah. Ten činí 0 – +/-10V, přičemž horní i spodní mez je parametricky nastavitelná (skrz softwarové rozhraní). Takováto nastavitelnost mezních rozsahů je nezbytná pro zachování kompatibility s výkonovými zdroji (řídící vstupy výkonových zdrojů nejsou standardizovány). Stejně jako v případě vstupů jsou technické parametry výstupů uvedeny v tabulce (Tab. 2).

Tab. 2: Specifikace výstupů SQM242

Počet výstupů	2 neizolované
Konektor	1/4“ dual phone jack
Výstupní napětí	0 – +/-10V Dc
Impedance	1 KOhm
Rozlišení	15 bitů + znaménko

Oba výstupy jsou vybaveny PID regulátory, přičemž jednotlivé složky regulátorů jsou nastavitelné přes softwarové rozhraní.

2.2 Softwarové rozhraní

U tak složitého měřicího zařízení jako je SQM242 je nezbytně nutné aby výrobce poskytoval také softwarové rozhraní. V případě jeho absence by se uživateli výrazně znesnadnilo použití výrobku. Nejdůležitější je existence interface (ovladače),

skrz který se dá ke kartě přistupovat. Tzn. číst vstupy, zapisovat na výstupy, popřípadě nastavovat parametry. Dalším vhodným způsobem jak uživateli zpříjemnit používání zařízení je poskytnout spolu se zařízením podrobnou dokumentaci, popřípadě i okomentované zdrojové kódy od jednoduchých ilustračních programů, které zařízení využívají.

V případě výrobce karty SQM242, firmy Sigma Instruments, je tato softwarová podpora více než dostačující. Obsahuje podrobnou dokumentaci ke kartě, k jejímu interface (DLL knihovna funkcí obsluhujících kartu) a zdrojové kódy programů, které ilustrují její použití.

2.2.1 DLL knihovna (sqm242a.dll)

Tato DLL knihovna tvoří interface mezi kartou SQM242 a uživatelem. Obsahuje sedmnáct funkcí, pomocí kterých lze s kartou pracovat. Nejdůležitější z těchto funkcí jsou stručně popsány níže. Podrobnější popis je možno nalézt v originální dokumentaci (viz. příloha). Všechny funkce vrací hodnotu 0 proběhla li v pořádku. Nenulová hodnota značí nějakou chybu. Jedinou výjimku tvoří funkce *Sif142GetReadings*, která vrací počet hodnot v bufferu (max hodnota je 10).

Sif142Startup2 (long Mode, long CardStatus (0 to 7))

Připojí potřebné DLL knihovny a inicializuje kartu. Tato fce musí být zavolána jako první a to s parametrem *Mode* = 0. Parametr *CardStatus* je pole, které vrací informace o nainstalovaných kartách a kódy případných chyb.

Sif142Init (double Xfmax, double Xfmin, double Xinit, double Period)

Úkolem této funkce je inicializovat měření. Nastavit rozsahy, ve kterých se může pohybovat frekvence krystalu (dostane li se krystal mimo tento rozsah je ohlášena chyba). Volání funkce je nutné provést před počátkem měření.

- **Xfmax** – hodnota horní meze rozsahu, ve kterém se může pohybovat frekvence krystalu (maximální hodnota je 10 MHz).
- **Xfmin** – hodnota spodní meze rozsahu, ve kterém se může pohybovat frekvence krystalu (minimální hodnota je 1 MHz).
- **Xinit** – inicializační frekvence krystalu.
- **Period** – perioda měření. Hodnoty je možné zadat v rozmezí 0.2 – 2 sekundy.

Sif142Simulate (long Mode)

Tato funkce nastavuje režim karty. Jsou na výběr dva režimy: simulační (Mode = 1) a normální (Mode = 0).

Sif142StartMeas ()

Touto funkcí se spouští měření a nulují se senzory.

Sif142ZeroSensor (long SensorNum)

Zavolání této fce způsobí vynulování jednoho z 24 senzorů (při nainstalovaných šesti kartách). Číslo senzoru, který má výt vynulován, je dáno parametrem **SensorNum**.

Sif142Material (long Sensor, double Density, double Zfact, double Tooling)

Tato funkce nastavuje parametry napařovaného materiálu.

- **Sensor** – číslo senzoru, pro který je materiál nastavován.
- **Density** – hustota napařovaného materiálu.
- **Zfact** – akustická impedance napařovaného materiálu.
- **Tooling** – tento parametr má za úkol kompenzovat polohu senzoru.

Sif142FullScale (long Output; FullScaleVolts, MaxPwr, SlewRate: double)

Touto funkcí jsou nastavovány následující parametry výstupů.

- **Output** – číslo nastavovaného výstupu.
- **FullScaleVolts** – nastavuje rozsah výstupního napětí (maximální rozsah je -10 – 0 či 0 – 10)
- **MaxPwr** – maximální hodnota v procentech rozsahu nastaveného předchozím parametrem. Zadávané hodnoty jsou 0 – 1 (odpovídá 0 – 100%).

Sif142Auto (long Output)

Tato funkce přepíná výstupy z manuální kontroly na automatickou PID regulátorem řízenou kontrolu.

Sif142Loop2 (double Rate, double P, double I, double D, long Output)

Tato funkce nastavuje PID regulátoru příslušného výstupu.

- **Rate** – žádaná hodnota rychlosti napařování v angstrom/sekundu
- **P** – proporcionální složka regulátoru (rozmezí hodnot je 0 až 9999)
- **I** – integrační složka regulátoru (rozmezí hodnot je 0 až 99.9)

- **D** – derivační složka regulátoru (rozmezí hodnot je 0 až 99.9)
- **Output** – číslo výstupu, pro který je regulátor nastavován.

Sif142SetPower (long Output, double Power)

Přepne příslušný výstup z automatického řízení (řízeno PID regulátorem) na manuální kontrolu.

- **Output** – číslo příslušného výstupu
- **Power** – hodnota, na kterou bude výstup po zavolání této funkce nastaven. Rozmezí hodnot je 0 – 1 (odpovídá 0 – 100%).

Sif142MapSensors (long SensLoop(0 to 23))

Touto funkcí jsou propojeny námi zvolené senzory s výstupy.

- **SensLoop** – pole hodnot, které obsahuje informace o propojení. Index pole představuje číslo senzoru, hodnota 0 či 1 představuje číslo výstupu. Zadáme li hodnotu -1 je senzor nepřipojen.

Sif142GetReadings (SensorArray(0 to 23, 0 to 2), OutputArray (0 to13): double)

Zavoláním této funkce vyčteme hodnoty vstupů i výstupů.

- **SensorArray** – dvourozměrné pole, kde první index představuje číslo senzoru, a hodnoty druhého indexu představují: 0 – aktuální rychlost napařování v A/s, 1 – aktuální tloušťka vrstvy v A, 2 – frekvence krystalu v Hz.
- **OutputArray** – v tomto poli jsou uloženy aktuální hodnoty výkonu jednotlivých výstupů v procentech.

3 Vývojové prostředí Control Web

Z druhého bodu zadání diplomové práce vyplývá, že je potřeba zprovoznit výše popsanou kartu SQM242 pod vývojovým prostředím Control Web (dále jen CW). Před započetím jakékoli práce je tedy nutné se s tímto prostředím seznámit.

Stručně lze CW popsat jako programové prostředí pro rychlý vývoj a provozování aplikací v oblasti průmyslové automatizace, databází, komunikace, zpracování obrazu a strojového vidění. De facto jde o objektově orientovaný grafický generátor, který umožňuje monitorování, řízení a vizualizaci technologie. Je možné v něm vytvářet jak aplikace pracující v reálném čase, tak i datově řízené. Jednoduchou aplikaci lze přitom vytvořit pomocí **virtuálních přístrojů** (budou popsány v kapitole 3.2) a to i bez hlubších znalostí programování. U složitějších programů jsou už tyto znalosti potřeba.

Vzhledem k tomu, že dokumentace k CW (příloha B) je velice podrobná a navíc celá v českém jazyce, popíši toto prostředí pouze do hloubky potřebné k pochopení principu tvorby aplikací. Nejprve bude popsáno vývojové prostředí obecně, poté koncepce virtuálních přístrojů, časování aplikace a nakonec komunikace s technologiemi a ovladače.

3.1 Popis prostředí

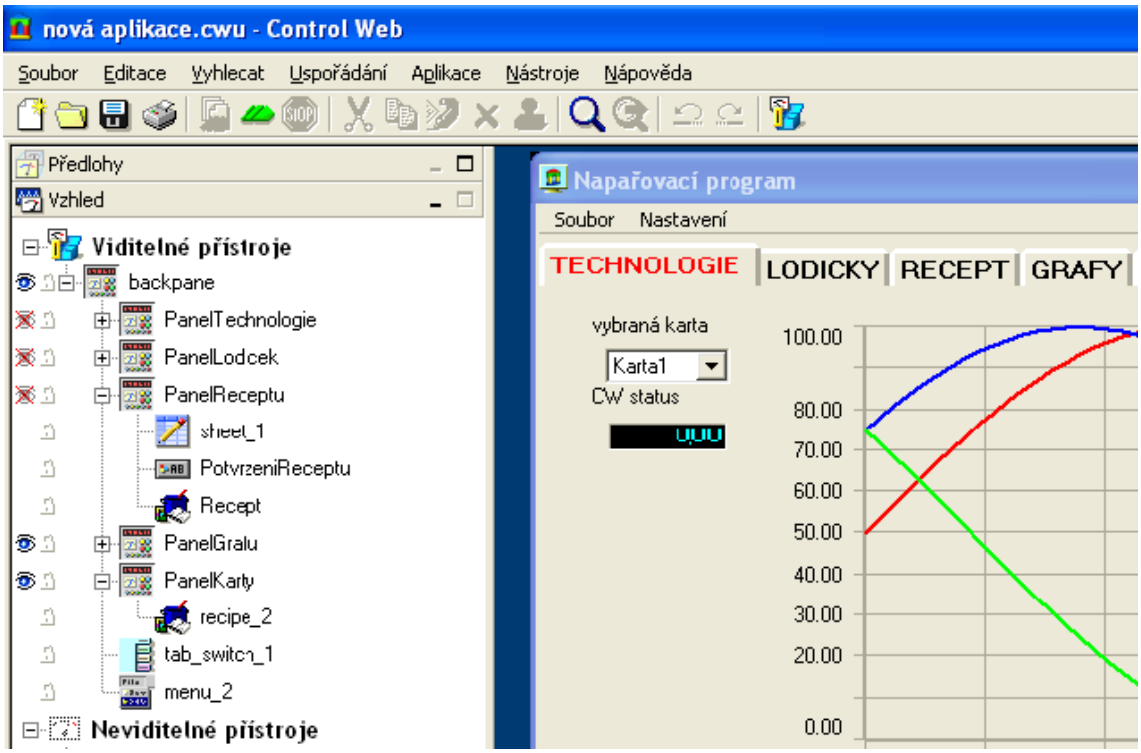
Uživatelské prostředí je rozděleno do tří logických bloků (Textový editor, Datové inspektory a Grafický editor), mezi kterými se uživatel pohybuje pomocí záložek (Obr. 8).

Obr. 8: Hlavní záložky vývojového prostředí

Je nutné zmínit, že jsou tyto tři bloky navzájem svázány, tj. uděláme li změnu v jednom z nich, promítne se tato změna do ostatních dvou. Přidáme li například v Grafickém editoru na hlavní panel aplikace virtuální přístroj, je v Textovém editoru vygenerován odpovídající programový kód a totéž platí i naopak.

3.1.1 Grafický editor

Tento editor (Obr. 9) slouží k snadné a přehledné tvorbě aplikací víceméně metodou táhni a pusť (drag and drop).



Obr. 9: Grafický editor

V jeho levé části je hierarchický strom s virtuálními přístroji, které v aplikaci využíváme. V pravé části je pak panel, na který již zmíněné přístroje z palety umisťujeme. Tento panel lze chápat jako hlavní okno aplikace, kterou vytváříme. V grafickém editoru však nevytváříme pouze grafický vzhled aplikace, ale určujeme i hierarchickou strukturu všech přístrojů.

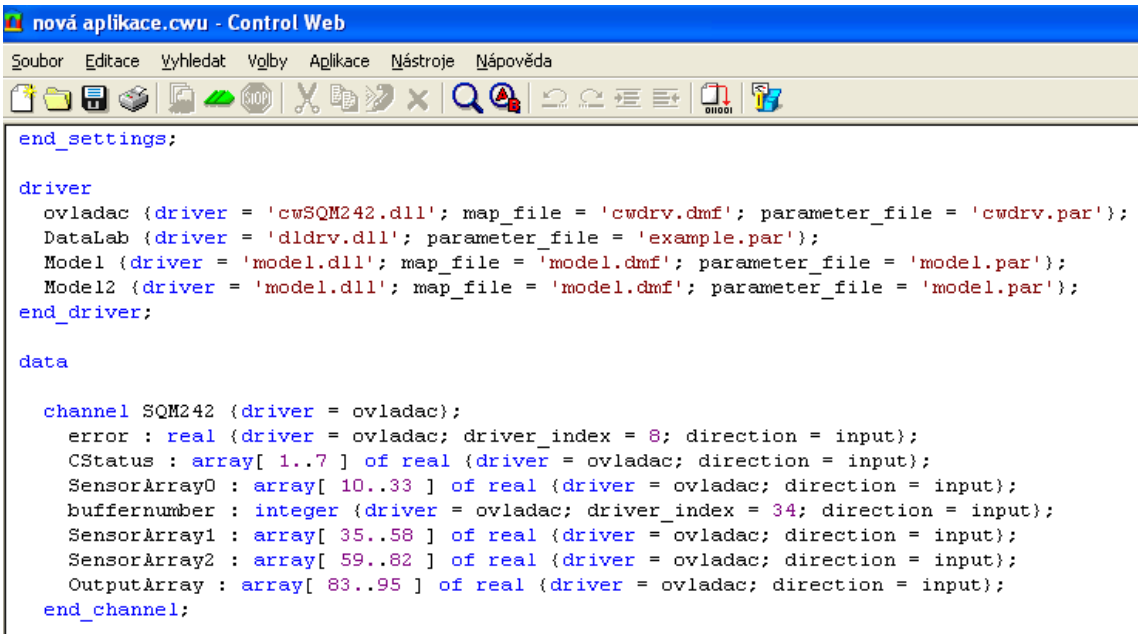
3.1.2 Datové inspektory

Pokud rozdělíme obecný řídicí systém na část řídicí a část datovou, pak lze zjednodušeně pokládat Grafický editor za editor, v němž vytváříme vzhled a řídicí část aplikace. Naproti tomu Datové inspektory lze chápat jako editor, ve kterém vytváříme datovou strukturu aplikace. Vytváříme zde proměnné, konstanty, kanály (popsány v kapitole 0) a další datové prvky. Datové inspektory plní ještě jednu velice důležitou funkci. Je to i jakýsi správce ovladačů (popsány v kapitole 3.3.1) tj. uživatel v nich určuje s jakými zařízeními a jakým způsobem bude aplikace s okolím komunikovat.

3.1.3 Textový editor

Grafický editor a Datové inspektory tvoří pouze jakousi nadstavbu, která výrazně usnadňuje tvorbu aplikace. Cokoliv pomocí této nadstavby vytvoříme, ve skutečnosti vygeneruje programový kód v textovém editoru. Je však možné se od této nadstavby oprostit a celou aplikaci naprogramovat pouze pomocí textového editoru. Ve většině případů je však lepší Grafického editoru a Datových inspektorů využít, jelikož se tím vyvarujeme zbytečných chyb a tvorba aplikace je rychlejší a přehlednější.

Okno textového editoru vypadá následovně (Obr. 10).



Obr. 10: Textový editor

Podíváme li se na obrázek podrobněji, zjistíme, že CW syntaxí vzdáleně připomíná Object Pascal. Je taky patrné, že se jedná o poměrně standardní editor kódu, běžný i u jiných vývojových prostředí.

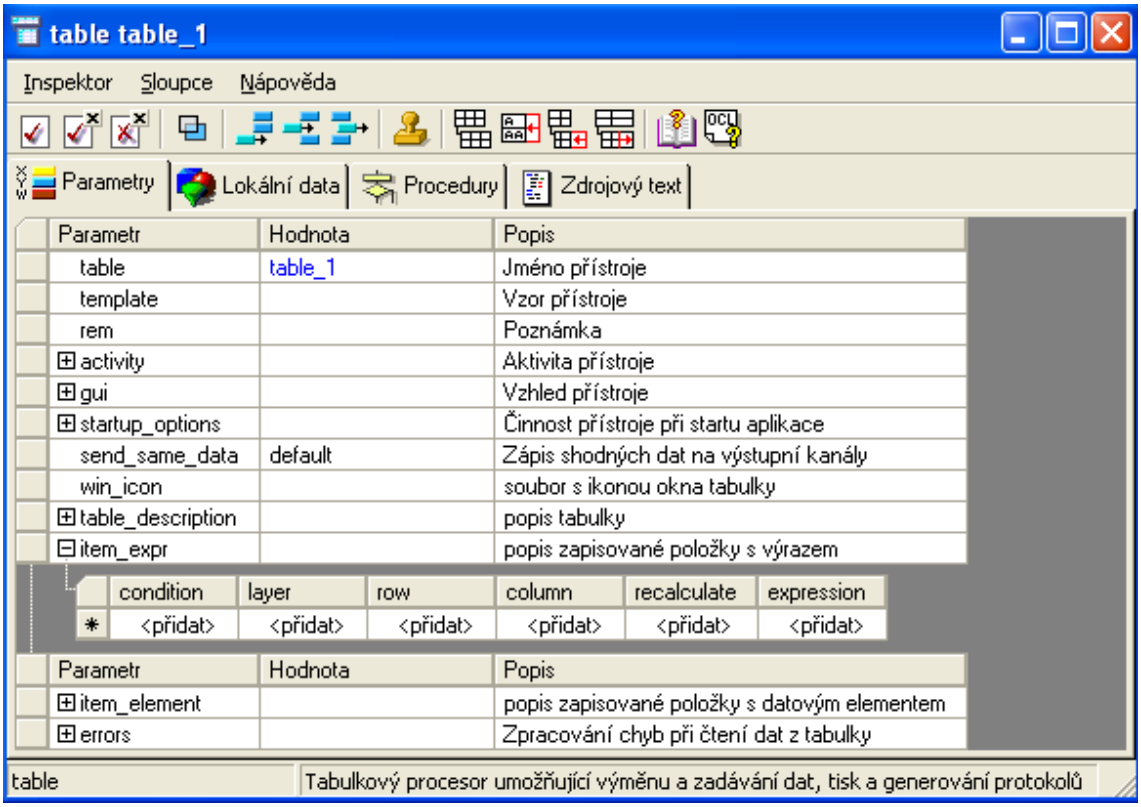
3.2 Virtuální přístroje

Virtuální přístroje lze chápat jako autonomní celky (komponenty), pomocí kterých je vyvíjená aplikace sestavována. Do množiny Virtuálních přístrojů patří zobrazovací a ovládací prvky, alarmy, archivy, historické trendy apod. Tato množina není navíc pevně dána a je ji možno libovolně rozšiřovat (každý virtuální přístroj je DLL knihovna detekovaná při startu systému).

Všechny přístroje obsahují vlastní vstupy, výstupy, lokální data, procedury a parametry. Prostřednictvím těchto vlastností je možné určit jejich vzhled a chování. V souvislosti s tím je potřeba zmínit (a blíže popsat) Inspektor a Paletu přístrojů.

3.2.1 Inspektor přístrojů

Tento inspektor je možné otevřít přes kontextovou nabídku jakéhokoli přístroje. V podstatě jde o jakýsi editor vlastností (vzhledu, chování atp.). Jak je z obrázku (Obr. 11) vidět, má inspektor více záložek. Záložka parametrů slouží k nastavení obecných vlastností jako například vzhledu, vstupních a výstupních elementů, viditelnosti, časování a vlastností okna aplikace (velikost, pozice). V dalších záložkách se nastavují lokální proměnné, barvy přístroje a především je důležitá záložka Procedury, ve které uživatel implementuje OCL procedury přístroje. Tedy jinými slovy řečeno, určuje, jak bude přístroj reagovat na předem definované podněty (kliknutí myší, aktivace jiným přístrojem, minimalizace okna atp.).



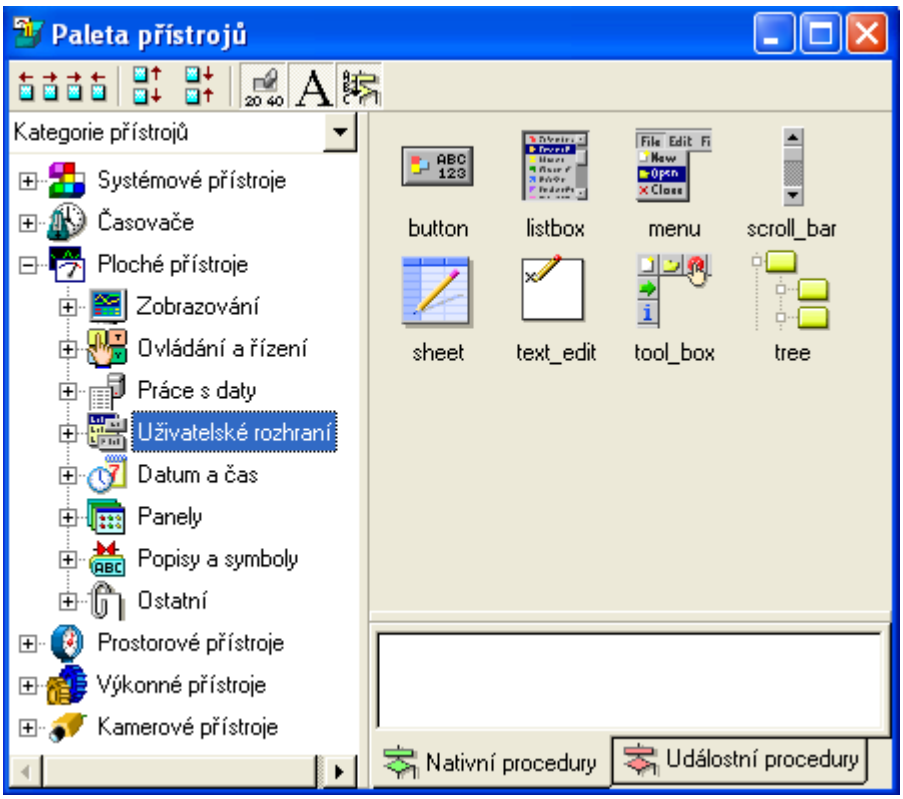
Obr. 11: Inspektor přístroje

Je potřeba dodat, že Inspektor přístroje není jedinou možností jak nastavovat či měnit vlastnosti, taktéž je to možné provést voláním tzv. nativních procedur z jiných

přístrojů (za běhu aplikace). Příkladem může změna textu titulku při stisku tlačítka (titulek i tlačítko jsou virtuální přístroje).

3.2.2 Paleta přístrojů

Přístroje jsou do programu přidávány pomocí tzv. palety přístrojů (Obr. 12). V té jsou přístroje setříděné do kategorií (zobrazování, ovládání a řízení, práce s daty, panely, popisky a symboly atp.). Umístění do programu probíhá jednoduše uchopením kurzorem myši a přetažením na námi zvolené místo v panelu aplikace.



Obr. 12: Paleta přístrojů

Z obrázku (Obr. 12) je také vidět, že opravdu vše v CW je ve skutečnosti virtuální přístroj. Od obyčejného panelu či tlačítka po komponenty pracující se soubory či komunikující po http.

3.3 Komunikace s technologií

Chceme li realizovat nějaký řídicí či měřící systém, neobejdeme se bez potřeby komunikovat s nějakým periferním zařízením. Příkladem může být i tak jednoduchá aplikace jakou je zobrazovač aktuální teploty. Zde potřebujeme komunikovat se zařízením, které teplotu fyzicky měří. Potřebujeme tedy jakýsi interface, který nám umožní data ze zařízení číst, či je do něj zapisovat. Zmíněný interface se (nejen) v CW nazývá ovladač.

3.3.1 **Ovladače**

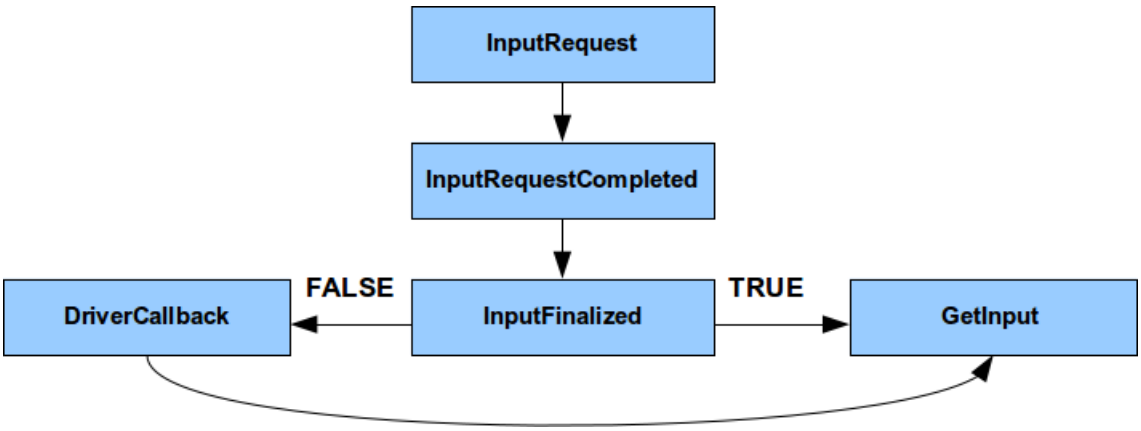
Ovladačem se v CW rozumí zpravidla DLL knihovna obstarávající přístup k danému zařízení. CW je navržen jako systém nezávislý na hardware. S patřičným ovladačem může pracovat s jakýmkoli zařízením (měřící karty, PLC, I/O moduly atd.).

Komunikace probíhá prostřednictvím tzv. kanálů. Kanál zpravidla reprezentuje fyzické vstupy a výstupy zařízení, se kterým komunikujeme. Jde o jakousi obdobu proměnné, která má určité specifické vlastnosti: rozlišení toho zda jde o vstup či výstup, index (číslo kanálu) atd. Vztah mezi ovladačem a kanálem je následující: ovladače jsou samostatné komponenty (malé autonomní programy – DLL knihovny), kanály jsou datové elementy, které jsou pomocí ovladače přenášeny mezi aplikací a daným zařízením. Ovladač tak funguje jako jakýsi prostředník, který povely aplikace transformuje do podoby, jíž rozumí dané zařízení.

Aby byl CW skutečně na hardware nezávislý, je třeba zajistit, aby měly všechny ovladače implementované jednotné rozhraní. Toto rozhraní sestává ze skupiny procedur a funkcí, které musí ovladač bezpodmínečně obsahovat, jinak nebude systémem CW akceptován. Rozhraní je popsáno v kapitole (4.1.3).

3.3.2 **Průběh komunikace**

CW ke komunikaci využívá některé funkce a procedury z rozhraní ovladače, které je podrobně popsáno v kapitole (4.1.3). Z toho důvodu není tato podkapitola zaměřena na popis jednotlivých metod, ale především na schéma takové komunikace (Obr. 13).



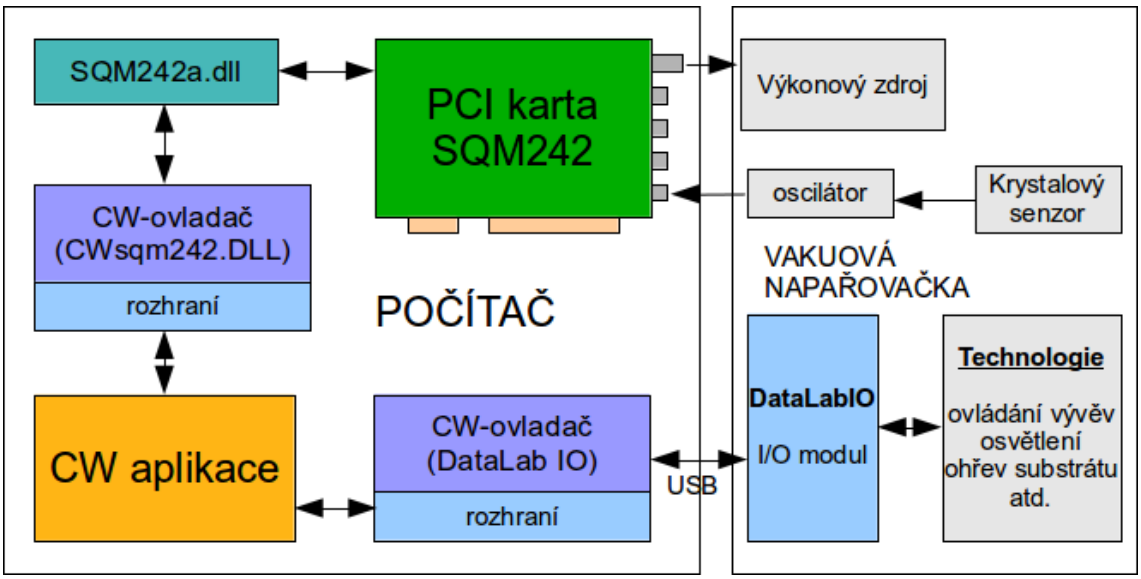
Obr. 13: Schematické znázornění komunikačního protokolu

Komunikace mezi CW a zařízením probíhá po následujícím komunikačním protokolu. Procedura ***InputRequest*** je jádrem CW volaná jednotlivě pro každý měřený kanál. Ovladač si tak sestaví jakýsi seznam kanálů, které má za úkol zařízení změřit. Po dokončení všech volání této procedury, zavolá jádro proceduru ***InputRequestCompleted***. Tím oznamuje, že již nemá další požadavky a že může začít ovladač měřit. Bezprostředně po této proceduře je volána funkce ***InputFinalized***, která vrátí hodnotu **true** je-li kanál doměřen, v opačném případě vrací hodnotu **false**. V případě PCI karty SQM242 vrací tato metoda vždy hodnotu **true**, jelikož PCI sběrnice má vysokou datovou propustnost a kanály jsou měřeny okamžitě. Důvod psát složitější kód je případě, že je komunikace pomalá (komunikace po RS232 atd.). Pokud vrátí funkce ***InputFinalized*** hodnotu **true** volá jádro funkci ***GetInput*** pro vyčtení hodnoty z kanálu. Naopak je-li vrácena hodnota **false** přichází na řadu procedura ***DriverCallback***, která je ovladačem volána v případě, že jsou doměřeny kanály (jeden nebo více), které jádru dosud nepotvrdil (při volání procedury ***InputFinalized*** vrátil ovladač hodnotu **false**). Tak dá ovladač aplikaci zprávu, že nyní může pro dané kanály zavolat metodu ***GetInput*** pro vyčtení potřebných hodnot.

Výše popsaný způsob komunikace platí pro čtení dat ze zařízení, pro zápis dat je situace téměř totožná jen s tím rozdílem, že se procedury a funkce jinak jmenují.

4 Tvorba ovladače krystalového monitoru SQM242

Ještě před započetím praktické práce bych si dovolil malou rekapitulaci již zjištěných skutečností a na jejich základě načrtl koncept, na kterém bude řídicí systém vakuové napařovačky postaven. Jako nejpřehlednější se jeví popis na blokovém schématu (Obr. 14).



Obr. 14: Blokové schéma koncepce systému

Nejdůležitější součástí systému je bezesporu měřící PCI karta SQM242. Ta obstarává hlavní monitorovací a regulační okruh vakuové napařovačky (snímá frekvenci krystalu a dle regulační odchylky řídí výkonový zdroj, potažmo rychlost náparu). Tato karta je přístupná skrze dynamicky linkovanou knihovnu SQM242a.dll. V té jsou obsaženy funkce, které práci s kartou zajišťují.

CW bohužel není schopen pracovat přímo s knihovnou SQM242a.dll. To je způsobeno snahou vývojářů tohoto systému ponechat ho nezávislým na hardware (všechny ovladače musí mít implementováno jednotné rozhraní). Ač existuje spousta již hotových ovladačů pro CW (PLC omron, simantic,měřící karty advantech atd.) krystalový monitor SQM242 mezi ně nepatří.

Cílem práce je tedy jak tvorba ovladače pro CW schopného komunikovat s kartou, tak tvorba řídicí CW aplikace, která bude s využitím tohoto ovladače a vstupně-výstupního modulu DataLabIO (k tomu již ovladač existuje) monitorovat a řídit proces depozice ve vakuové napařovačce.

Obecně lze CW ovladač rozdělit na dvě části. Na rozhraní, které musí být bezpodmínečně implementováno a přídavné funkce, pomocí kterých se vývojář snaží využít maximum potenciálu daného zařízení. CW pracuje s rozličným hardware a jednotné rozhraní přirozeně veškerý potenciál těchto zařízení pokrýt nemůže.

Jak již bylo v kapitole (CW ovladače) zmíněno jsou CW ovladače de facto DLL knihovny obstarávající CW přístup k danému zařízení. Při tvorbě ovladače SQM242 budu tedy postupovat následovně: nejprve bude vytvořena prázdná DLL knihovna. V té bude následně vytvořeno rozhraní, které je vyžadováno CW. Po dokončení rozhraní budou vytvořeny rozšiřující funkce, pomocí kterých bude uživatel schopen kartu SQM242 plnohodnotně ovládat.

Pro lepší pochopení následující části práce zde uvádím, že vytvořená DLL knihovna **cwSqm242.dll** sestává ze dvou zdrojových souborů. Ze souboru **cwSQM242.dpr**, který je popsán v úvodu následující kapitoly a ze souboru **drvpas.pas**, kterým se zabývají zbývající podkapitoly.

4.1 Tvorba DLL knihovny

Při tvorbě DLL knihovny je potřeba se nejprve rozhodnout, v jakém programovacím jazyce (popřípadě i prostředí) bude naprogramována. V mém případě přicházely v úvahu dvě varianty. Buďto programovat v jazyce C (či C++) nebo využít Object Pascal (konkrétně prostředí Delphi 7), se kterým již mám nějaké zkušenosti.

Po krátké úvaze jsem se rozhodl pro programování DLL knihovny v Delphi 7. Hlavním důvodem této volby byly již zmíněné zkušenosti s Object Pascalem, a především, dle mého názoru, přívětivější prostředí.

Prvním krokem při programování ovladače je tvorba prázdné DLL knihovny. Zdrojový kód vypadá následovně (zdrojový soubor **cwSQM242.dpr**):

Zdrojový kód 1: soubor sqm.dpr

```
library cwSQM242;

{$R 'drvform.dfm' :TForm(drvform.dfm)}
uses
      drvpas in 'drvpas.pas',
      drvform in 'drvform.pas' {DrvDialog};
{$DEBUGINFO ON}
exports
      Version,
      .
      .
      QueryProc;

{$R *.RES}
begin
end.
```

Zde jsou uvedeny názvy exportovaných procedur a funkcí

Knihovna je uvozena klíčovým slovem **library**, za kterým musí následovat jméno dané knihovny v našem případě cwSQM242. Pod klíčovým slovem **uses** jsou uvedeny zdrojové soubory, které knihovna využívá. Zde se jedná o soubor **drvpas.pas**, ve kterém je obsažen vlastní výkonný kód této knihovny (rozhraní a uživatelské funkce). Pod posledním klíčovým slovem – **exports** je uveden seznam jmen procedur a funkcí, které chceme z knihovny exportovat. Jinými slovy řečeno, jsou zde uvedeny názvy všech metod, u kterých chceme, aby byly nějakým vnějším programem využívány (volatelné, viditelné). V našem případě se seznam exportovaných metod zcela shoduje s rozhraním, které vyžaduje CW.

V následujících podkapitolách je popsána volací konvence, připojení knihovny sqm242a.dll, rozhraní a uživatelské funkce ovladače (vše je obsahem zdrojového souboru **drvpas.pas**)

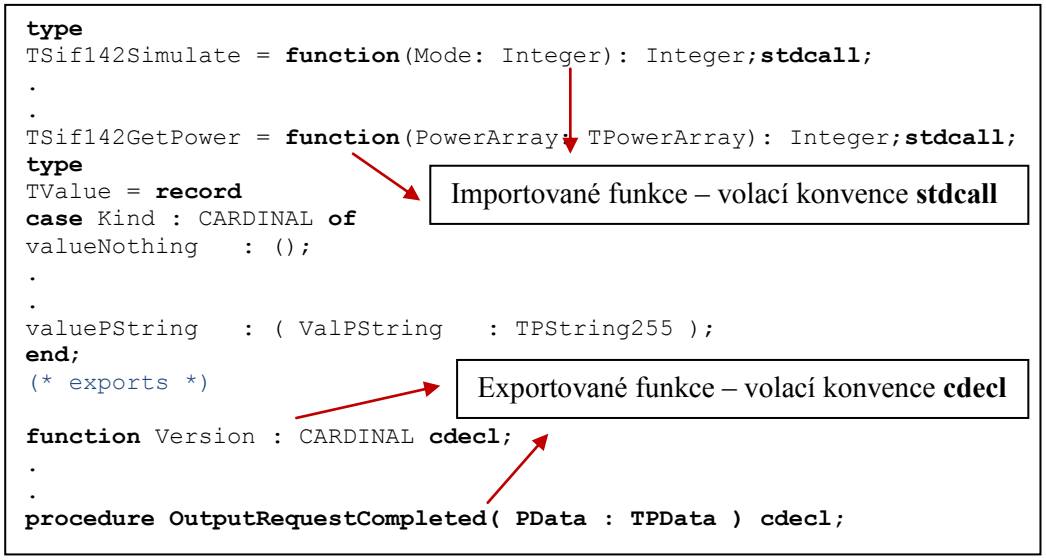
4.1.1 Volací konvence

Volací konvence je způsob, jakým jsou předávány parametry mezi procedurou a vykonávaným programem. Existuje několik různých způsobů, které se navzájem liší pořadím parametrů (buď zprava doleva, či naopak) a také tím, kdo uklidí zásobník (volající, nebo volaný kód). Tyto způsoby jsou navzájem nezaměnitelné. Bude li si procedura a program vyměňovat parametry s odlišnou volací konvencí, bude program ukončen chybovou hláškou.

V případě tvorby DLL knihovny je velice důležité vědět, jakou volací konvenci využívá program, který bude knihovnu využívat, popřípadě i volací konvence použitých DLL knihoven (DLL knihovna může využívat funkce z jiné DLL knihovny).

To je také náš případ jelikož CW požaduje volací konvenci **cdecl** a knihovna **sqm242a.dll**, která bude v naší knihovně využita, používá konvenci **stdcall**. Je potřeba zdůraznit, že volací konvence není spjata přímo s knihovnou, nýbrž s konkrétní procedurou či funkcí knihovny. Z toho plyne, že je možné v jedné knihovně využívat různé volací konvence pro různé metody. Příkladem může být následující ukázka zdrojového kódu:

Zdrojový kód 2:



Z této ukázky kódu je patrné, že rozdílná volací konvence linkované knihovny sqm242a.dll a CW nepředstavuje žádný problém. Také je vidět jakým způsobem se konkrétní metodě volací konvence v Delphi 7 nastavuje.

4.1.2 Připojení knihovny sqm242a.dll

Jelikož budeme v naší DLL knihovně využívat jinou DLL knihovnu (sqm242a.dll), je potřeba se zaměřit na možnosti připojování těchto knihoven.

V zásadě existují dva způsoby jak DLL knihovnu připojit, buďto staticky, nebo dynamicky. Staticky připojená knihovna je programem připojena na začátku jeho spuštění a odpojena je až při vypnutí programu. Naproti tomu u dynamického připojení knihovny je připojení i odpojení knihovny plně v rukou programátora. Je tedy možné knihovnu připojit až ve chvíli, kdy je skutečně potřeba. Výhodou statického připojení je tak jednodušší programování, a dynamického propojení šetření prostředků PC a plná kontrola nad tím, kdy bude knihovna připojena a odpojena.

Porovnáním všech pro a proti jsem se rozhodnul využít dynamické připojení knihovny. Na následující ukázce kódu je vidět, jak použití DLL knihovny vypadá (obsah souboru **drvpas.pas**).

Zdrojový kód 3:

```
const
  lib = 'sqm242a.dll';
type
  TSif142Simulate = function (Mode: Integer): Integer stdcall;
  .
  .
  TSif142GetPower = function (PowerArray: TPowerArray):Integer;stdcall;
var
  SqmSimulate: TSif142Simulate;
  .
  .
  SqmGetPower: TSif142GetPower;
  hDLLHandle : HModule;
  .
  .
function Simulace(Mode: String): Integer;
begin
  Simulace := SqmSimulate(StrToInt(Mode));
end;
.
.
function StartMereni():Integer cdecl;
begin
  StartMereni := SqmStartMeas();
end;
.
.
function MakeDriver : POINTER;
begin
  hDLLHandle := LoadLibrary(lib);
  INC( Handle );
    SqmSimulate := GetProcAddress(hDLLHandle, 'Sif142Simulate');
    .
    .
    SqmGetPower := GetProcAddress(hDLLHandle, 'Sif142GetPower');
  MakeDriver := @Handle;
end;
.
.
procedure Done( PData : TPData );
begin
  FreeLibrary(hDLLHandle);
end;
```

název připojované knihovny deklarován jako konstanta

vytvoření typu a posléze jejich deklarace

využití funkcí z připojené DLL knihovny

připojení knihovny a získání popisovače modulu (hDLLHandle)

získání ukazatele na exportované funkce

odpojení DLL knihovny

Při dynamickém připojení DLL knihovny je stěžejní použití následujících metod.

LoadLibrary(NazevKnihovny): zavoláním této funkce načteme DLL knihovnu a získáme popisovač modulů (je vrácen touto funkcí).

GetProcAddress(PopisovacModulu, NazevMetody): zavoláme li tuto funkci, získáme ukazatel na exportovanou funkci specifikovanou parametrem NazevMetody. To je důležité jelikož aplikace volají funkce DLL knihovny prostřednictvím ukazatele.

FreeLibrary(PopisovacModulu): tuto proceduru je nutné zavolat po dokončení práce s DLL knihovnou (dojde k jejímu odpojení).

4.1.3 **Rozhraní ovladače**

Je dobré si uvědomit, že ač musí být všechny procedury a funkce rozhraní ovladače nadefinovány, nemusí obsahovat žádný kód. Z toho důvodu zde uvedu pouze ty, které nějaký výkonný kód skutečně obsahují. Navíc bude kladen důraz především na popis jejich funkce, přičemž ilustrační ukázka zdrojového kódu jedné z nich je vidět výše (Zdrojový kód 3), konkrétně jde o metodu Done (kompletní zdrojový kód je obsahem přílohy).

GetDriverInfo – Vrací jméno ovladače pro zobrazení v inspektoru ovladačů CW.

MakeDriver – Vytvoří novou instanci ovladače a vrací její handle do systému. Systém pak volá všechny ostatní procedury a funkce ovladače s parametrem driver, kterým je právě tento handle.

DisposeDriver – Procedura ruší instanci ovladače. Volá se při ukončení aplikace nebo při zastavení překladu z důvodu chyby.

Init – Volá se po zavedení ovladače do paměti a slouží k inicializaci ovladače. Procedura vrací hodnotu 1, když inicializace ovladače proběhla korektně. V případě, že se inicializace nezdařila, vrátí hodnotu 0.

Done - Volá se při ukončení činnosti ovladače.

InputRequest – Procedura slouží k předání požadavku na získání hodnoty kanálu ovladače. V rámci právě zpracovávaného časového kroku se volá postupně pro všechny požadované kanály. Zadávání požadavků se ukončí voláním procedury *InputRequestCompleted*. Další požadavek na stejný kanál může přijít až po dokončení komunikace a potvrzení procedurou *InputFinalized*.

InputRequestCompleted – Procedura ukončí zadávání požadavků na čtení kanálů ovladače (InputRequest). V tomto okamžiku může ovladač zahájit činnost vedoucí k získání všech požadovaných hodnot (zahájení měření, komunikace apod.).

InputFinalized – Volá se po zadání požadavků na čtení kanálů a zahájení operací čtení (InputRequestCompleted) postupně pro všechny požadované kanály. Vrací hodnotu 1, je-li operace ukončena. Nejsou-li hodnoty kanálů ještě k dispozici, vrátí hodnotu 0.

GetInput – Pomocí této procedury se přečte hodnota kanálu z ovladače a zapíše se do tabulky aktuálních hodnot kanálů, kde je k dispozici pro všechny virtuální přístroje.

OutputRequest – Procedura slouží k předání požadavku na zápis hodnoty kanálu do ovladače. V rámci právě zpracovávaného časového kroku se volá postupně pro všechny požadované kanály. Zadávání požadavků se ukončí voláním procedury *OutputRequestCompleted*. Ovladač musí být schopen přijímat požadavky i v okamžiku, kdy je zaneprázdněn komunikací se zařízením.

OutputRequestCompleted – Procedura ukončí zadávání požadavků na zápis kanálů do ovladače po volání (OutputRequest). V tomto okamžiku může ovladač zahájit činnost vedoucí k zápisu všech požadovaných hodnot (nastavení výstupů, zahájení komunikace apod.).

OutputFinalized – Volá se po zadání požadavků na zápis kanálů a zahájení operace zápis (OutputRequestCompleted) postupně pro všechny požadované kanály. Vrací hodnotu 1, je-li operace ukončena. Nejsou-li hodnoty kanálů ještě zapsány, vrátí hodnotu 0.

QueryProc – Procedura slouží k vyvolání obecné uživatelské funkce ovladače prostřednictvím nativní procedury: core.DriverQueryProc. Tato metoda i uživatelské funkce jsou blíže popsány v kapitolách 4.1.4 a 4.1.5.

4.1.4 Uživatelské funkce a procedury

Jak již bylo výše uvedeno, sestává ovladač karty kromě rozhraní také z uživatelských funkcí. Ty mají za úkol zpřístupnit uživateli veškeré funkcionality plynoucí ze specifičnosti daného zařízení. Zde konkrétně karty SQM242. O těchto funkcích lze souhrnně napsat, že de facto tvoří jakýsi bypass mezi funkcemi z knihovny sqm242a.dll a CW. Jednotlivé procedury a funkce tedy pouze volají odpovídající metody z knihovny sqm242a.dll.

Pro pochopení způsobu, jakým jsou tyto funkce naprogramovány, je postačující ukázka kódu jedné z nich, jelikož ostatní jsou naprogramovány obdobným způsobem. U zbylých metod bude tedy pouze v tabulce (Tab. 3) uvedeno jakou funkci z knihovny sqm242a.dll využívají (funkce z sqm242.a byly podrobně popsány v kapitole DLL knihovna (sqm242a.dll)2.2.1).

Zdrojový kód 4

```
function InicializaceKarty(Xfmax,Xfmin,Xinit,Period: String):Integer cdecl;
begin
  InicializaceKarty := SqmInit(StrToFloat(Xfmax),StrToFloat(Xfmin),
                               StrToFloat(Xinit),StrToFloat(Period));
end;
```

Na této ukázce kódu (Zdrojový kód 4) je vidět, že funkce v podstatě tvoří rozhraní mezi CW a metodami z DLL knihovny sqm242a.dll. Je li tedy zavolána funkce InicializaceKarty s potřebnými parametry, jsou tyto parametry předány odpovídající metodě z knihovny sqm242a.dll (v tomto případě SqmInit), která svou návratovou hodnotu předá zpět uživatelské funkci. Je rovněž patrné, že parametry uživatelské a knihovní funkce jsou různého typu. To je způsobeno tím, že bezpodmínečně všechny uživatelské funkce musí mít parametry typu String. Důvod této skutečnosti je vysvětlen v následující podkapitole.

Tab. 3: Seznam uživatelských funkcí

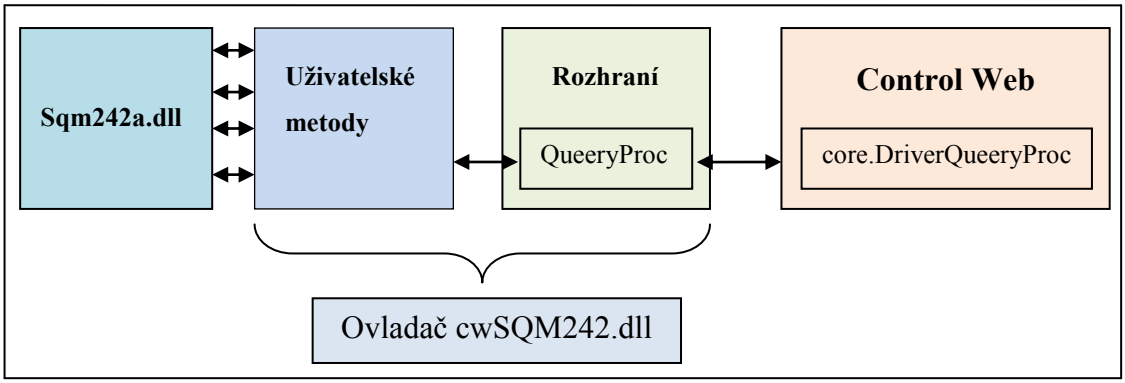
Uživatelské funkce	**Knihovní funkce**
InicializaceKarty	Sif142Init
Simulace	Sif142Simulate
NastaveniVystupu	Sif142FullScale
NastaveniMaterialu	Sif142Material
NastaveniRegulatoru	Sif142Loop2
PropojeniSenzorVystup	Sif142MapSensors
VynulovaniSenzoru	Sif142ZeroSensor
VynulovaniVystupu	Sif142Zero2
NastaveniPower	Sif142SetPower
Auto	Sif142Auto
StartMereni	Sif142StartMeas

4.1.5 **Metoda QueeryProc**

Metoda QueeryProc, o které bude v této kapitole blíže pojednáno je součástí rozhraní námi tvořeného ovladače. Rozhraní a jeho procedury a funkce byly sice již popsány v kapitole 4.1.3. Důvodem, proč je zde tato metoda popsána podrobněji, je skutečnost, že má přímý dopad na uživatelské funkce a výrazně ovlivňuje podobu jejich

výkonného kódu. Jde totiž o metodu, prostřednictvím které jsou všechny uživatelské funkce Control webu zpřístupněny.

Princip volání uživatelských metod je vysvětlen opět pomocí blokového schématu.



Obr. 15: Volání uživatelských metod

Procedura core.DriverQueeryProc má následující parametry:

- DriverName : string – určuje symbolické jméno ovladače, kterému je volána procedura QueeryProc.
- DrvParam1 : any – tento parametr je předáván do ovladače.
- &DrvParam2 : any – tento parametr slouží k předání hodnoty jak do ovladače, tak z ovladače.

Z blokového schématu je vidět, že je li zavolána procedura core.DriverQueeryProc, pak je dle parametru DriverName nalezen příslušný ovladač, ve kterém je následně zavolána procedura QueeryProc. Ta poté dle parametru DrvParam1 zavolá příslušnou uživatelskou funkci a předá jí parametr &DrvParam2. Uživatelská funkce vykoná svůj kód a uloží svou návratovou hodnotu zpět do parametru &DrvParam2. Ten může být následně vyčten v CW.

Tento mechanismus má však jeden nedostatek. Je patrné, že k předání hodnot funkcím máme k dispozici pouze jeden parametr (&DrvParam2). Uživatelské funkce našeho ovladače jsou však veskrze víceparametrové. Tento problém může mít několik níže uvedených způsobů řešení.

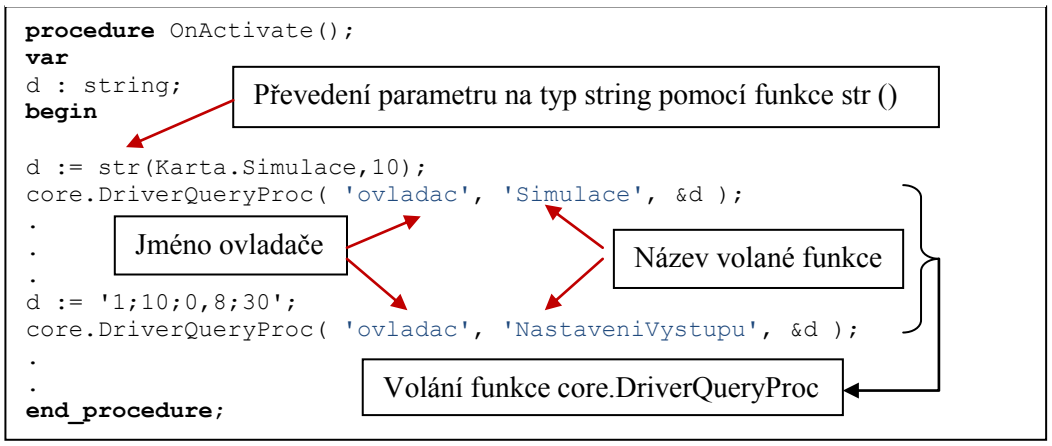
Prvním z možných řešení by mohlo být předání parametru &DrvParam2 funkci jako pole či záznam. V tomto případě zde však vyvstává další problém, jelikož parametrů je obecně neurčité množství a navíc jsou obecně různých typů. V tomto

případě by se tedy složitě programoval kód, který by byl jednoduše schopný oddělit od sebe neurčité množství položek neurčitých typů a poté je předat příslušné funkci. Všechny tyto problémy řeší druhá možnost.

Deklarujeme li parametr &DrvParam2 jako string a uložíme li do něj textový řetězec složený z jednotlivých položek oddělených například středníky, dostaneme de facto parametr jednotného typu (řetězec znaků), který je možné opět jednoduše rozdělit na jednotlivé parametry a ty následně potřebně přetypované předat příslušné funkci. Toto řešení bylo pro svou univerzálnost a jednoduchost také využito. Je zde tedy vysvětleno proč mají uživatelské a knihovní funkce sice stejný počet a význam parametrů nicméně jiný typ (uživatelské fce vždy string).

Následující ukázky kódu jsou zaměřeny na složení textového řetězce z jednotlivých parametrů, volání procedury core.DriverQueeryProc, výkonný kód fce QueeryProc který parametry opět oddělí a předá příslušné funkci.

Zdrojový kód 5:



Výše uvedená ukázka pochází z Control webu. Je zde deklarována proměnná d typu string, do které je ukládán řetězec parametrů oddělených středníky. Tento parametr je následně předán funkci core.DriverQueryProc. Tato funkce volá funkci QueryProc konkrétního ovladače určeného jejím prvním parametrem. Výkonný kód této funkce je zde:

Zdrojový kód 6:

```
procedure QueryProc( PData : TPData;Param1 : TValue;var Param2 : TValue );
var
  StringovyParametry,navratovaHodnota : String;
  i,j : integer;
begin
  j := 0;
  for i:=0 to 23 do begin
      poleParametru[i]:= '';
  end;
  if (Param1.Kind = valuePString) && (Param1.ValPString <> nil) then
  begin
       StringovyParametry := Param2.ValPString^;
  end;
  for i:=1 to length(StringovyParametry) do
  begin
      if (StringovyParametry[i]<>';')then
         poleParametru[j] := poleParametru[j] + StringovyParametry[i];
      if StringovyParametry[i] = ';' then inc(j);
  end;

  if ( Param1.Kind = valuePString )&&( Param1.ValPString <> nil ) then
    if StrComp( Param1.ValPString^, 'InicializaceKarty' ) = 0 then
      if ( Param2.Kind = valuePString ) &&( Param2.ValPString <> nil ) then
        begin
          navratovaHodnota := IntToStr(InicializaceKarty(poleParametru[0],
                                       poleParametru[1],poleParametru[2],
                                       poleParametru[3]));
          StrLCopy( Param2.ValPString^,PAnsiChar(navratovaHodnota) ,
                                      Length( navratovaHodnota ) );
        end;
end;
```

Tato část kódu rozdělí textový řetězec na jednotlivé parametry

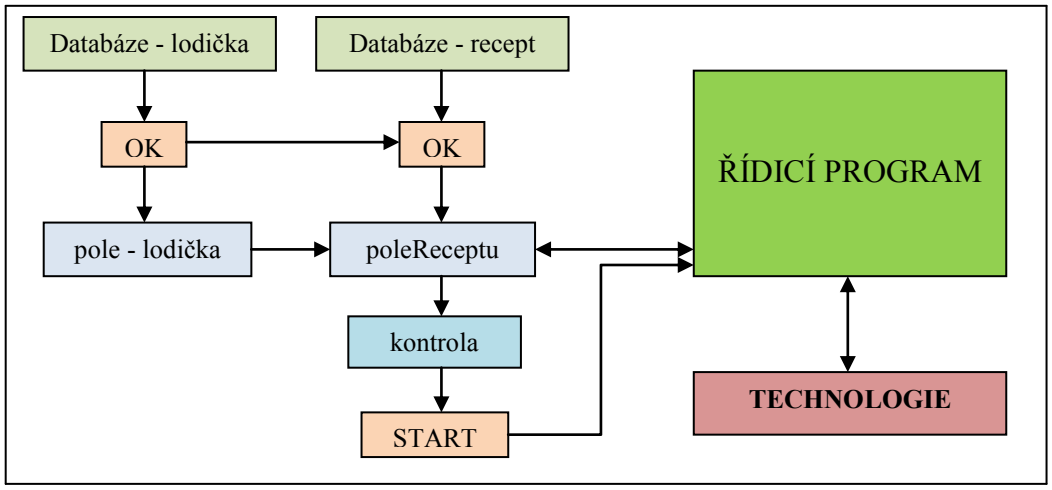
Tato část kódu již vykonává příslušné uživatelské funkce

5 Tvorba monitorovacího systému

Teprve po dokončení ovladače krystalového monitoru SQM242 je možné začít pracovat na tvorbě monitorovacího systému vakuové napařovačky ve vývojovém prostředí Control Web. Aplikace bude mít za úkol monitorovat a řídit proces depozice tenkých vrstev. Nejprve bude nastíněn koncept celého systému a poté se bude v podkapitolách zabývat podrobněji jednotlivými částmi.

5.1 Koncept systému

Před započetím jakékoli práce a tím spíše je li touto prací programování nějakého systému, je třeba si důkladně rozmyslet celkový koncept. Není li tato příprava promyšlena důkladně, může se stát, že řešení elementárních problémů za pochodu povede do slepé uličky.



Obr. 16: Koncept řídicího systému

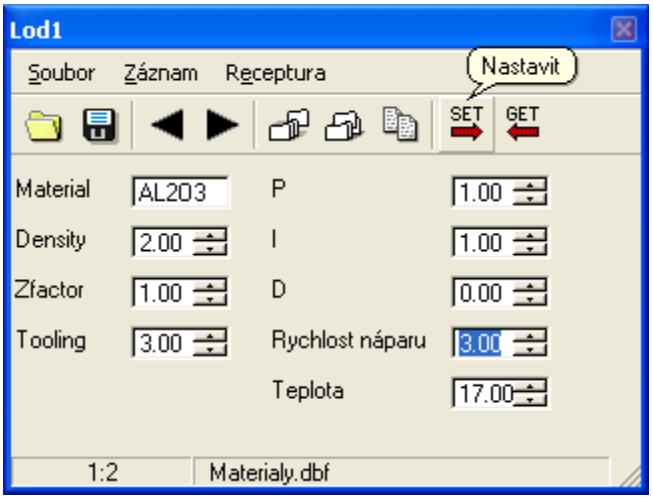
Na obrázku (Obr. 16) je vidět zjednodušené blokové schéma řídicího systému. To bude nejprve vysvětleno jako celek. Jednotlivé bloky budou podrobněji popsány v následujících podkapitolách.

Depoziční proces sestává zpravidla z napaření více než jedné vrstvy. Pořadí, potřebná tloušťka a technologické parametry jsou uloženy ve dvou databázových souborech, na obrázku (Obr. 16) označeny světle zelenou barvou. V těchto databázích může uživatel listovat jednotlivými záznamy, popřípadě je i upravovat, přidávat či mazat. Je li vybrán příslušný záznam a je li potvrzen, dojde k sestavení takzvaného receptu (na obrázku poleReceptu). Tento recept má charakter pole záznamů a obsahuje položky seřazené v pořadí jednotlivě napařovaných vrstev. Je li recept sestaven úspěšně

(potvrzeno kontrolou), je možné stisknout potvrzovací tlačítko START, po kterém se rozeběhne programový blok řídicí depoziční proces (na obrázku označen tmavě zelenou barvou).

5.1.1 Databázové soubory

Control web podporuje práci s různými databázemi přes rozhraní ODBC (Open DataBase Connectivity), SQL dotazy atd. Navíc přímo disponuje virtuálními přístroji pro práci s těmito databázemi. Jedním z nich je přístroj recipe (receptura). V podstatě jde o editor databázových souborů typu *.DBF. Jak je na obrázku (Obr. 17) patrné disponuje přístroj ovládací i zobrazovací částí. Práce s přístrojem je velmi jednoduchá a intuitivní a proto byl vybrán pro přístup k databázím lodiček a receptů.



Obr. 17: Virtuální přístroj recipe

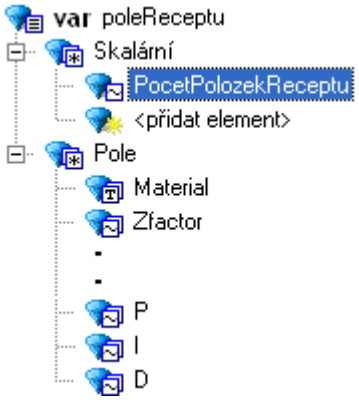
Databáze lodiček (výparníků) sestává ze záznamů, které obsahují parametry spjaté s materiálem (například i technologické parametry na materiálu závislé). Naproti tomu databáze receptu obsahuje položky týkající se samotného napařování, například tloušťku vrstvy, žádaný tlak uvnitř komory atd. Navíc obsahuje taky jeden důležitý parametr a to číslo lodičky, z které bude ta či ona vrstva napařována. To jakým způsobem je z těchto souborů sestaven výsledný recept, je vysvětleno v následující kapitole.

5.1.2 Sestavení receptu

K tomu abychom mohli automatizovaně napařit určitý počet vrstev, musíme nejprve sestavit jejich seznam seřazený v pořadí, v jakém je budeme napařovat. Vstupními daty takového seznamu jsou výše zmíněné databázové soubory. Je tedy

nutné z nich získat potřebné informace a ty seskupit do jediného datového bloku, takzvaného receptu.

Tvorba receptu probíhá následujícím způsobem: nejprve je třeba nadefinovat datový blok, ve kterém bude tento recept uchováván. V podstatě jde o záznam složený z polí a skalárních datových elementů.



Obr. 18: Struktura záznamu poleReceptu

Struktura záznamu poleReceptu je patrná z obrázku (Obr. 18). Obsahuje jeden skalární datový element (PocetPolozekReceptu), který nese údaj o aktuálním počtu položek v datové podsekci záznamu, která obsahuje soubor polí. Z programátorského hlediska je důležité zmínit, že přístup k tomuto záznamu a jeho položkám je přes tečkovou konvenci a jednotlivá pole v záznamu je možné indexovat. To umožňuje jednoduchou práci v programových cyklech. Pro upřesnění je zde (Zdrojový kód 7) uvedena jednoduchá ukázka zápisu hodnoty 5.4 do dvanácté položky pole P (pole proporcionálních konstant).

Zdrojový kód 7:

```
poleReceptu.P[12] := 5.4
```

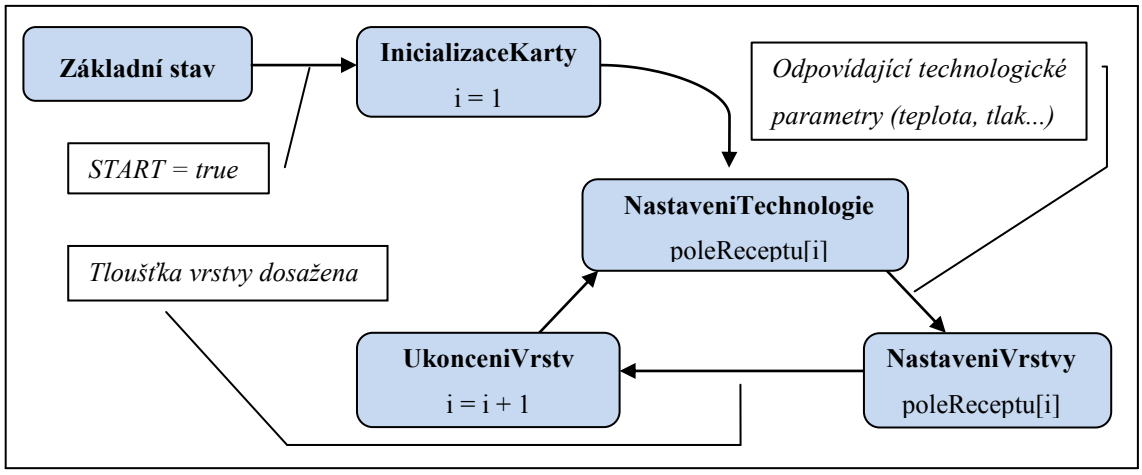
Máme li vytvořené potřebné datové sekce (kromě poleReceptu také poleLodicek), je možné sestavit program, který bude položky z databází číst a do poleRecepu zapisovat. Logika tohoto programu je znázorněna na blokovém schématu (Obr. 16) uvedeném výše. Po nastavení odpovídajícího záznamu v databázi Lodiček a stisknutí potvrzovacího tlačítka, je jednak obsah tohoto záznamu zapsán do datového elementu poleLodicek a jednak je uživateli zpřístupněno potvrzovací tlačítko u databáze receptu. Nastaví li nyní uživatel odpovídající záznam v databázi receptu a stiskne li již zpřístupněné tlačítko, uloží se do poleReceptu údaje obsahující hodnoty jak z databáze

receptu, tak z poleLodicek. Tím je sestaveno kompletní poleReceptu, dle kterého se bude depoziční proces řídit.

Je li sestaven recept a proběhne li úspěšně kontrola (zda některé hodnoty nevybočují z povoleného rozsahu atp.) je uživateli zpřístupněno tlačítko START, po jehož stisku je spuštěn program řídící samotnou depozici.

5.1.3 **Řídicí program**

Řídicí program je jakési jádro systému. Podle hodnot uložených v receptu (datový element poleReceptu), nastavuje prostřednictvím ovladače kartu SQM242 a parametry technologie. Funkce řídicího programu je vysvětlena na blokovém schématu (Obr. 19).



Obr. 19: Blokové schéma logiky řídicího programu

Řídicí program se skládá ze čtyř speciálních virtuálních přístrojů. Jedná se o přístroje **program**. V podstatě jde o klasické podprogramy, které je možné spouštět periodicky či podmínkou.

Před stiskem tlačítka START se systém nachází v *základním stavu*. Po stisku tlačítka je spuštěn podprogram *InicializaceKarty*. V něm jsou postupně prostřednictvím procedury *core.DriverQueeryProcc* (vysvětleno v kapitole 4.1.5) volány uživatelské funkce ovladače *InicializaceKarty*, *Simulace* a *nastaveniVystupu*. Bezprostředně poté je spuštěn podprogram *NastaveniTechnologie*. V něm jsou nastaveny technologické parametry uvedené v receptu (teplota, tlak atd.).

Je li těchto parametrů dosaženo a jedná li se o ustálené hodnoty, je spuštěn podprogram *nastaveniVrstvy*. Tím jsou uživatelskými funkcemi (*nastaveniMaterialu*,

nastaveniRegulatoru, *propojeniSenzorVystup*) nastaveny parametry depozice uvedené v receptu (tloušťka vrstvy, rychlost napařování atd.) a spuštěn depoziční proces.

Dosáhne li tloušťka napařované vrstvy hodnoty uvedené v receptu je spuštěn podprogram *ukonceniVrstvy*, který zastaví napařování, zapíše hodnotu skutečně napařené tloušťky vrstvy a zvýší index vrstvy o jednu a aktivuje znovu podprogram *nastaveniTechnologie*.

Tímto způsobem jsou podprogramy aktivovány do doby, než podprogram *ukonceniVrstvy* vyhodnotí, že bylo dosaženo konce receptu. Tím pádem jsou všechny vrstvy napařeny a proces depozice je ukončen.

5.2 Vzhled a prostorové rozložení aplikace

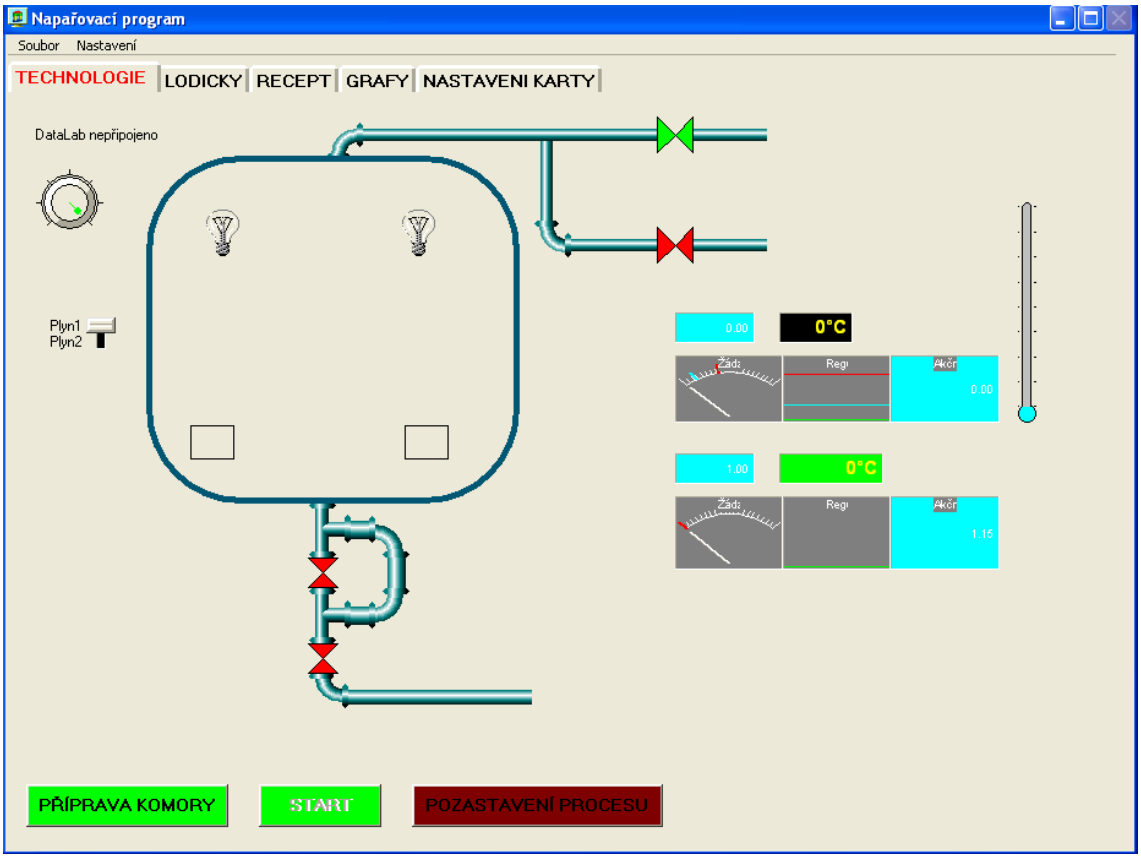
Popis vzhledu a rozložení prvků aplikace není sice v této práci bezpodmínečně nutný, nicméně dotváří celkový obraz o tomto systému.

Aplikace obsahuje pouze jedno okno, v jehož horní části je lišta sloužící k přepínání jednotlivých záložek. Každá z těchto záložek v sobě sdružuje virtuální přístroje, které k sobě logicky i funkčně patří. Pořadí záložek z leva doprava je navíc koncipováno tak, že provází uživatele celým procesem napařování od jeho počátku (nastavení napařovací komory, sestavení receptu atp.) až po záložku obsahující zobrazovací prvky (grafy a panely zobrazující aktuální hodnoty parametrů).

Následující podkapitoly budou věnovány jednotlivým záložkám.

5.2.1 **Záložka technologie**

Tato záložka obsahuje interaktivní vizualizaci technologie (vakuové napařovačky). Jak je vidět na obrázku (Obr. 20), jsou zde zobrazeny jak všechny potřebné informace, tak ovládající prvky, kterými je možné nastavovat žádané hodnoty technologických parametrů (tlak, teplotu, osvětlení atd.).



Obr. 20: Záložka technologie

Zároveň s vizualizací jsou zde také umístěna tlačítka ovládající proces depozice (START, STOP atd.). Pro operátora obsluhujícího technologický proces je vizualizace jednoznačnou výhodou. Operátor se tak rychleji orientuje, čímž se zrychluje jeho reakce na případný problém. Navíc je možné snadněji predikovat následky jakéhokoli zásahu.

5.2.2 Záložka lodiček a receptu

Tyto dvě záložky mají téměř totožný vzhled. V obou jsou umístěny pouze virtuální přístroje recipe. Důvod proč nejsou tyto záložky sloučeny do jedné, spočívá v logice celé aplikace. Uživatel nemůže pracovat s receptem, pokud nevybral a nepotvrdil materiál ve výparnících (lodičkách). Navíc by sloučení mohlo vést k nepřehlednosti celého panelu.

Vzhled těchto záložek není v této kapitole nutné uvádět, jelikož sestává pouze z přístrojů recipe, který již na obrázku (Obr. 17) zobrazen byl.

5.2.3 Záložka grafů

Na této záložce jsou zobrazeny veškeré veličiny popisující průběh depozičního procesu. Konkrétně jde o rychlost napařování, tloušťku vrstvy, výkon na výparníku a

frekvenci krystalu. Tyto údaje nejsou zobrazeny pouze číselně, nýbrž i graficky a to ve virtuálním přístroji data_wievew. U toho je možné měnit zobrazení. Lze ho zobrazit jako graf či jako tabulku hodnot. Hodnoty je možné ukládat do souborů či přímo tisknout.

Co se týká obrázku ilustrujícího vzhled této záložky, bylo by zbytečné ho v této podkapitole uvádět, jelikož v následující části práce bude ještě mnohokrát zobrazen (například zde:).

6 Testování aplikace a výsledky

Před uvedením aplikace do ostrého provozu, je třeba ji řádně otestovat, zhodnotit její funkčnost a spolehlivost a opravit případné nedostatky. Je potřeba si uvědomit, že v případě nějaké nepodchycené chyby může obecně hrozit například poškození zařízení, či v horším případě újma na zdraví obsluhy. V našem případě sice zařízení ani obsluha ohrožena není, ovšem pokud by byl naprogramovaný systém uveden ihned do ostrého provozu, mohlo by dojít znehodnocení výrobků (nemalá ekonomická ztráta).

Testování systému proběhne ve třech fázích. Nejprve proběhne základní test funkčnosti monitorovací části. Ten bude spočívat v připojení odpovídajícího oscilátoru a senzoru ke kartě SQM242 a vyčítání hodnoty frekvence. Dopadne li tento test úspěšně, bude na řadě test systému jako takového. To znamená sestavení receptu a následné spuštění depozičního procesu (samozřejmě v simulačním režimu karty). Třetím a zároveň posledním testem bude zkouška řízení skutečné vakuové napařovačky (ovšem bez výrobků uvnitř komory).

6.1 Test komunikace s kartou SQM242

Jde o nejjednodušší možný test. Jeho úkolem je zjistit, zda karta SQM242 komunikuje s naprogramovaným řídicím systémem a zda komunikuje korektně. Bezchybná komunikace je stěžejní podmínkou pro pokračování testování systému. Jinými slovy řečeno, teprve bude li tento test úspěšný, je možné postoupit k další zkoušce.

Test se skládá ze dvou fází. Nejprve bude odzkoušena komunikace s kartou v simulačním režimu, poté s připojeným reálným oscilátorem a krystalovým senzorem. Předpokládané hodnoty, které by potvrdily bezchybnou komunikaci, jsou 5950000 Hz u simulačního režimu karty a u reálného senzoru hodnoty v rozmezí 1-10 MHz (vychází z datasheetu karty SQM242). Kvůli přehlednosti budou výsledky testu uvedeny v následující tabulce (Tab. 4).

Tab. 4: Výsledek testu komunikace

	Hodnoty potvrzující úspěšnost testu	Naměřené hodnoty
Simulace	5950000 Hz	5950000 Hz
Reálný senzor	1 – 10 MHz	5005230 Hz

Z tabulky je patrné, že obě fáze této zkoušky proběhly úspěšně. V simulačním režimu činila hodnota frekvence skutečně 5950000 Hz. S reálným senzorem byla frekvence 5005230, čili v daném rozmezí. Bylo tedy možné přistoupit k další fázi testování.

6.2 Test řídicího systému

Tato zkouška otestuje správnost celého konceptu řídicího systému. Bude opět probíhat ve dvou krocích. Nejprve bude systém testován s kartou v simulačním režimu. Poté, bude li tato fáze úspěšná, proběhne test přímo na reálném zařízení.

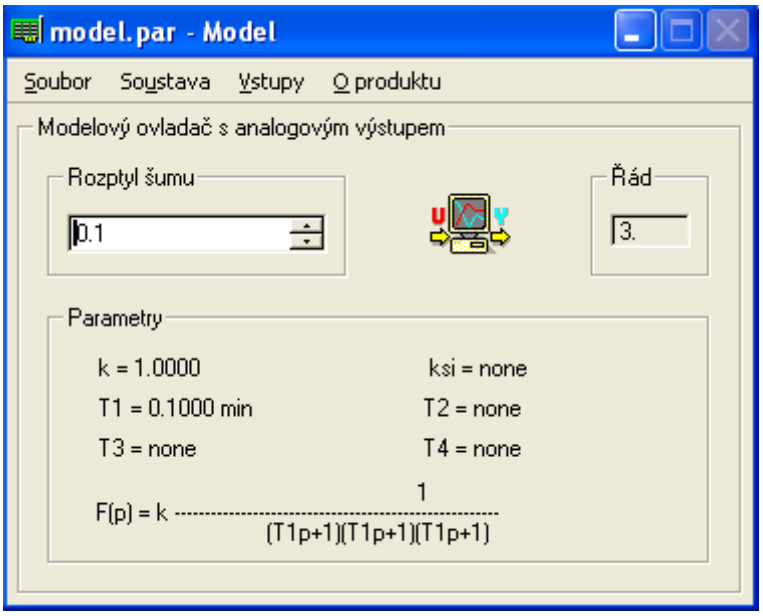
Pro obě fáze testu je zvolen stejný postup. Nejprve bude vytvořen recept. Poté bude spuštěn depoziční proces, který bude probíhat v závislosti na sestaveném receptu. Výsledkem obou těchto zkoušek by, v případě úspěchu, měl být bezchybný napařovací cyklus, který bude plnit požadavky jednotlivých položek receptu (správná teplota, tlak, tloušťka vrstev atd.).

6.2.1 **Test – simulační režim**

Jak již bylo mnohokrát zmíněno, podporuje karta SQM242 práci v takzvaném simulačním režimu. V podstatě jde o komplexní interaktivní simulaci, kdy je karta schopna na základě hodnot výstupního výkonu simulovat změny frekvence imaginárního senzoru, potažmo rychlost náparu, přičemž všechny vstupní parametry (materiálové konstanty, parametry PID regulátorů atd.) chod simulace ovlivňují stejně jako ve skutečnosti. Simulace je tak velmi komplexní.

Pouze schopnost karty pracovat v simulačním režimu však k tomuto testu nestačí. Aby mohlo dojít ke kompletně simulovanému testu systému, je třeba namodelovat i některé části technologie. Především ty, které pracují s veličinami obsaženými v sestaveném receptu (teplota, tlak atd.). Je tedy přinejmenším nutné vytvořit model okruhu ohřívajícího výrobky (substrát) a model okruhu regulujícího tlak v komoře.

Zde je další možnost jak využít CW. Jeho součástí je totiž modelový ovladač **Model.dll**, který dokáže simulovat soustavy až čtvrtého řádu. CW navíc obsahuje i virtuální přístroj **PID_controler**, kterým je možné tyto soustavy regulovat (jedna soustava tlak, druhá teplota). Nastavovací okno modelového ovladače vypadá následovně (Obr. 21).



Obr. 21: Nastavovací okno modelového ovladače

Jak je z obrázku (Obr. 21) patrné jsou oba okruhy namodelovány jako soustavy třetího řádu. Skutečné přechodové funkce ohřevu substrátu ani tlakového okruhu nejsou při testu systému důležité. Vytvořené modely totiž nemají v tomto případě za úkol věrně simulovat odpovídající reálné ekvivalenty. Slouží pouze k testování, zda je podmínka ustálenosti veličiny (tlak, teplota) provedena správně. Jinými slovy řečeno, je testováno, zda řídicí systém začíná napařování další vrstvy ve správném okamžiku. Pokud by totiž napařování další vrstvy začalo za ještě neustálené teploty či tlaku, došlo by s největší pravděpodobnosti ke znehodnocení výrobku.

Co se týká samotného testu, byl vytvořen pokusný recept, uvedený v tabulce (). V tom jsou uloženy žádané hodnoty námi sledovaných veličin (tlak, teplota, tloušťka vrstvy, rychlost napařování atd.). Výstupem z napařovacího procesu jsou časové závislosti těchto veličin. Aby bylo možné tento test prohlásit za úspěšný, musí nutně časové závislosti k žádaným hodnotám, obsaženým v receptu, konvergovat. Tato podmínka musí být splněna pro každou jednotlivou položku receptu.

Naměřené časové závislosti jsou zobrazeny v následujícím grafu ().

Podíváme li se na graf pozorně, je patrné, že se skládá z několika fází. Každá z těchto fází ve skutečnosti představuje napařování jedné z vrstev definovaných v receptu. Jejich počet je tedy shodný s počtem položek receptu. Srovnáme li tyto fáze

s odpovídajícími položkami receptu, zjistíme, zda dopadl test úspěšně či neúspěšně. Kvůli přehlednosti je srovnání uvedeno v následující tabulce (tab.).

Z výše uvedené tabulky vyplývá, že se hodnoty jednotlivých veličin v grafu s receptem v mezích tolerancí shodují. Není tedy pochyb o tom, že test dopadl úspěšně. Je tedy možné přejít k poslední fázi testování a to ke zkoušce řídicího systému na reálném napařovacím zařízení.

Závěr

Náplní této práce byl vývoj a realizace monitorovacího systému vakuové napařovačky. Z předchozí kapitoly (Zhodnocení výsledků) vyplývá, že požadavky kladené na vytvořený řídicí systém byly, až na drobné nedostatky, splněny (Je nutno dodat, že zmíněné nedostatky byly již odstraněny).

Při testování v reálném provozu byl systém schopen plně automaticky řídit napařování jednotlivých vrstev dle nastaveného receptu (uživatelsky definované pořadí vrstev s jednotlivými parametry). Co se týká spolehlivosti, nedošlo zatím při testování k žádnému nežádoucímu či neočekávatelnému chování systému.

Při vytváření vzhledu a rozmístění ovládacích prvků, bylo vycházeno z již existujících systémů, jejichž vzhled tvoří jakýsi standard. Dle mého názoru je tak ovládání systému intuitivní a uživatelsky přívětivé (byl využit potenciál vizualizačních nástrojů vývojového prostředí Control Web).

Co se týká návrhů na další možné rozšiřování či vylepšování systému, je zde především v měřící části ještě nevyužitý potenciál. Vylepšení by tak mohlo směřovat k implementaci optického senzoru tloušťky napařeného materiálu.

Seznam použité literatury

[1] *Datasheetcatalog : datasheet* [online]. 2009 [cit. 2009-05-17]. Dostupný z WWW: <http://www.datasheetcatalog.org/datasheet/infineon/1-tda46053.pdf>.

[2] *Datasheetcatalog : datasheet* [online]. 2009 [cit. 2009-05-17]. Dostupný z WWW: <http://www.datasheetcatalog.org/datasheet/stmicroelectronics/9387.pdf>.

[3] *Datasheetscatalog* : *datasheet* [online]. 2009 [cit. 2009-05-17]. Dostupný z WWW: <http://www.datasheetcatalog.org/datasheet/eic/SF33.pdf>.

[4] *Farnell : Datasheet* [online]. [cit. 2009-05-17]. Dostupný z WWW: <http://www.farnell.com/datasheets/41319.pdf>.

[5] KREJČIŘÍK, Alexandr. *Napájecí zdroje 1 : Základní zapojení analogových a spínaných napájecích zdrojů*. 2. vyd. Praha : BEN, 1997. 351 s. ISBN 80-86056-02-3.

[6] KREJČIŘÍK, Alexandr. *Napájecí zdroje 2 : Integrované obvody ve spínaných zdrojích*. 2. vyd. Praha : BEN, 1997. 351 s. ISBN 80-86056-03-1.

[7] PRESSMAN, Abraham I. . *Switching power supply design*. 2nd edition. New York : McGraw-Hill, 1997. 682 s. ISBN 0-07-052236-7.

[8] SCLOCCHI, Michele. *Switching power supply design : continuous mode flyback converter* [online]. [cit. 2009-05-17]. Dostupný z WWW: <www.national.com/appinfo/power/files/flyaback-cont2000-5000-new.pdf>.

[9] TŮMA, Jiří, et al. *Základy automatizace*. 1. vyd. Ostrava : Ediční středisko VŠB – TUO , 2007. 283 s. ISBN 978-80-248-1523-7.

Seznam příloh

B: CD

Obsah CD

- Elektronická podoba diplomové práce
- Literatura
    - Základy automatizace.pdf
    - Switching power supply design.pdf