TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

# ELEKTRONIZACE ÚLOH LABORATORNÍCH CVIČENÍ NA PLATFORMĚ NOVÉHO INTELIGENTNÍHO MĚŘICÍHO PRACOVIŠTĚ AP9

## Bakalářská práce

*Studijní program:* B2646 – Informační technologie
*Studijní obor:* 1802R007 – Informační technologie

*Autor práce:* **Jakub Hadač**
*Vedoucí práce:* Ing. Petr Pfeifer

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

# LAB PRACTICAL LESSONS ON THE PLATFORM OF THE NEW INTELLIGENT MEASUREMENT WORKPLACE AP9

## Bachelor thesis

*Study programme:* B2646 – Information Technology
*Study branch:* 1802R007 – Information Technology

*Author:* **Jakub Hadač**
*Supervisor:* Ing. Petr Pfeifer

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2014/2015

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Jakub Hadač** |
| Osobní číslo: | **M13000098** |
| Studijní program: | **B2646 Informační technologie** |
| Studijní obor: | **Informační technologie** |
| Název tématu: | **Elektronizace úloh laboratorních cvičení na platformě nového inteligentního měřicího pracoviště AP9** |
| Zadávající katedra: | **Ústav informačních technologií a elektroniky** |

Zásady pro vypracování:

1. Nastudujte nově vyvinutou platformu pro inteligentní měřicí pracoviště se systémem dálkové správy a zpracování dat.
2. Nastudujte stávající úlohy laboratorních cvičení včetně nového předmětu PMN a navrhněte metody implementace potřebných změn.
3. Vytvořte nové objekty, naprogramujte předlohy všech laboratorních cvičení a otestujte je na zkušebním systému AP9.

Rozsah grafických prací: **Dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:


[1] **Dokument Inteligentní učebna a měřicí pracoviště, TUL, ESF 2012/2013**

[2] **Dokumentace k osciloskopům GW Instek série GDS2000**

[3] **Architektury Klient/Server**

[4] **E. Naramore at al., Vytváříme webové aplikace v PHP5, MySQL a Apache, Computer Press**


Vedoucí bakalářské práce: **Ing. Petr Pfeifer**
Ústav informačních technologií a elektroniky

Datum zadání bakalářské práce: **12. září 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 12. září 2014

# Prohlášení

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

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

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

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15.5.2015

Podpis: 

# Poděkování

Na tomto místě bych rád poděkoval vedoucímu mé diplomové práce, Ing. Petru Pfeiferovi, MSc, MBA, Ph.D., za všechny rady, bohaté zkušenosti a poznatky, které mi pomohly k úspěšné realizaci této práce. Dále bych chtěl poděkovat své rodině za jejich nekončící podporu a možnost studovat na vysoké škole. Nakonec všem svým blízkým přátelům, kteří mně při studiu velice pomohli a motivovali.

# Abstrakt

Cílem této bakalářské práce je vytvořit webový formulář pro tvorbu protokolu z měření na webovém serveru Inteligentního měřicího systému. Tento protokol je vytvořen pomocí nově vytvořených a popsaných komponent, které nahrazují jednotlivé dílčí úkoly ve vypracovávaném měření. Na základě analýzy poskytnutých protokolů z měření byly vytvořeny komponenty v programovacích jazycích určených pro vývoj webových stránek. Pomocí těchto nově vytvořených komponent se realizovaly webové formuláře pro tvorbu protokolu z měření a výsledný technický dokument ve formátu PDF. Podařilo se vytvořit součást unikátního systému, která umožňuje vyvářet a realizovat protokol z laboratorního měření ve webovém prohlížeči atak urychlit a zjednodušit činnosti spojené s vytvářením protokolů z měření pro autory a vedoucí cvičeně, stejně jako pro studenty během samotného měření.



# Klíčová slova

Elektronizace; Laboratorní měření; Inteligentní měřicí systém; Komponenty;

# Abstract

This bachelor aims at and this document deals with development of components and forms for protocols to be created during measurements in laboratory classes, running on a web server and within the new intelligent remote control, measurement and data processing framework. The forms and protocols are created using the new developed components, describing each single tasks and expected outputs in performed measurements and laboratory classes. All the new and developed final components were created on the base of previously delivered forms of old protocols, and programmed in programming languages intended for creation of internet web pages. The new developed components do realize the web forms and the final protocols as a technical document in PDF format. All the effort clearly led to new parts of the unique system, allowing design and creation of laboratory classes, measurements and protocols, running easily in a web browser and speeding up the activities and tasks connected to the development and creation of all the documents even by authors of classes or lecturers, as well as tasks performed by students.



# Keywords

Electronization; Laboratory measurements; Intelligent measurement system; Components

# Obsah

# Seznam obrázků

# Seznam zkratek

- WWW (World Wide Web) –„celosvětová síť" – systém prohlížení, ukládání a odkazování dokumentů nacházejících se v Internetu
- HTML (HyperText Markup Language) – značkovací jazyk pro tvorbu webových stránek
- CSS (Cascading Style Sheet) – jazyk pro popis grafického zobrazení prvků na webové stránce
- HTTP (HyperText Transfer Protocol) – je internetový protokol k výměně hypertextových dokumentů
- PDF (Portable Document Format) – přenosný formát dokumentů od firmy Adobe
- PHP (Hypertext PreProcesor) – skriptovací programovací jazyk pro generování dynamických webových stránek
- JS (JavaScript) – skriptovací programovací jazyk
- API (Application Programming Interface) – rozhraní pro programování aplikací
- PDF (Portable Document Format) – přenosný formát dokumentů od firmy Adobe
- TCPDF – knihovna pro generování PDF souborů

# Úvod

Tuto bakalářskou práci jsem si zvolil, protože během mého studia na střední škole jsem vypracovával mnoho protokolů z měření a proto si velmi dobře vybavuji problematiku vytváření protokolu z laboratorního cvičení přímo během samotného měření.

Dříve se při realizaci technických protokolů z měření veškeré části dokumentu vytvářely ručně, od zápisu technickým písmem, až po rýsování grafu podle různých pomůcek (křivítka). Postupem času byly jednotlivé činnosti spojené s tvorbou technického protokolu ulehčeny různými počítačovými programy, procesory nebo aplikacemi. Rýsování tabulek a grafů nahradily tabulkové procesory, návrhy schémat zapojení elektrického či logického obvodu vznikají pomocí specializovaných návrhových programů, atd. Avšak v dnešní době nastává problém s množstvím programů, které jsou nutné pro vytvoření takového technického protokolu. Při chybném zápisu údajů do špatného programu je nutné tato data znovu přepisovat do správného programu. Občas narazíme i na problém propojení měřicího přístroje s naším tabulkovým procesorem, a proto je nutné využít dalšího programu, abychom získali potřebná data z měřicího přístroje, a ta pak dále mohli zpracovávat. Při tvorbě takového protokolu je nutné přepínat mezi velkou řadou programů, což může být dosti zdlouhavé a matoucí, a také to může odvádět pozornost od přesného provádění měření. Proto je někdy mnohem lepší v průběhu měření doplňovat protokol z měření ručně do připravené šablony, což nám zkrátí dobu potřebnou k vypracování. Ale bohužel nastává problém s formální úpravou výsledného dokumentu, jako jsou například schémata zapojení nakreslená od ruky, přeškrtané chyby v textu nebo text napsaný do nerovnoběžných řádků, apod.

Cílem této práce je vytvořit webové komponenty pro inteligentní měřicí systém v učebně AP-9 psané ve spojení programovacích jazyků Hypertext preprocessor (PHP), Javascript (JS) a systému pro tvorbu šablon Smarty určené pro moderní webové prohlížeče, které by umožňovaly vytvořit webový formulář pro realizaci a jednoduché vytvoření technického protokolu z měření. Komponenty mají za úkol nahradit potřebné schopnosti jednotlivých textových, tabulkových, grafických procesorů, ostatních programů a ruční práce tak, aby pomocí jejich kombinace bylo možné vytvořit kompletní technický protokol z měření ve webové aplikaci. Jednoduše by se dalo říci, že se jedná o vytvoření alternativy k jednotlivým procesorům, aplikacím a programům, tak aby jejich používání ulehčilo práci nejen těm, kteří měření vytvářejí, ale i těm, kteří jej zpracovávají.

# 1 Cíle a omezení práce

Cílem této bakalářské práce je analyzovat poskytnuté zadání k měření a šablony pro vypracování protokolu z laboratorního cvičení nebo měření. Na základě této analýzy navrhnout, vytvořit a otestovat komponenty pro webovou aplikaci, tak aby pomocí těchto komponent bylo možné realizovat plnohodnotný formulář pro vytvoření kompletního protokolu z měření v dané webové aplikaci. Komponenty musí nahradit veškeré dílčí úlohy, které jsou vypracovávané ručně nebo pomocí počítačového programu a zároveň by neměly prodlužovat dobu nutnou pro vypracování dokumentu.

Dalším cílem této práce je minimalizace počtu používaných aplikací, programů nebo procesorů při tvorbě technického protokolu z měření tak, aby se zkrátila doba strávená manipulací s těmito programy, a tím se urychlila tvorba protokolu z měření.

Protože cílem této bakalářské práce není řešení rozsáhlého komplexního problému v rámci celého systému, ale pouze realizace dílčí úlohy při vývoji Inteligentního měřicího systému pro učebnu AP-9, jsou zde určitá omezení.

Prvním omezením je samotný systém (Sieber, 2015), který byl vyvíjen souběžně s touto prací. Protože systém nebyl v době zadání bakalářské práce ještě kompletně hotový, bylo umožněno jeho vývoj ovlivňovat za přidávání vlastních rozšíření, která ulehčí vytváření komponent. Druhým omezením, které souvisí se systémem, jsou pevně dané programovací jazyky. Tedy nelze využít žádný jiný programovací jazyk, který není podporovaný systémem. V našem případě musíme požít jazyky PHP, JS a knihovnu Smarty. Posledním omezením je rozložení výpočetní zátěže mezi klienta a server, tak aby bylo možné řešit cvičení i na slabším počítači.

# 2 Teoretická část

V této části práce bude rozebrána obecná koncepce vývoje komponent nebo webových stránek. Jako jsou například komunikace mezi klientem a serverem, jazyk pro strukturalizaci webových stránek apod. Dále také všeobecné stanovy pro vytváření protokolu z laboratorního měření nebo úlohy.

## 2.1 Webový server

Webový nebo také www server je program, nebo tzv. démon, který je spuštěn dlouhodobě a vyčkává na událost. Po obdržení dané události ji obslouží bez nutnosti zásahu obsluhy stroje, na kterém webový server běží. Server komunikuje s klienty pomocí internetového protokolu HTTP a pomocí tohoto protokolu odesílá klientům požadované webové stránky ve formátu HTML (Sintes, 2002).

Nejznámějším a nejpoužívanějším zástupcem je webový (www) server Apache od společnosti The Apache Software Foundation.

## 2.2 Jazyk HTML

HTML je zkratka jazyku HyperText Markup Language (hypertextový značkovací jazyk), který je v dnešní době stále nejoblíbenější jazyk pro vytváření jednoduchých www stránek. Název *hypertextový* znamená, že text může osahovat odkazy, díky kterým webové stránky fungují paralelně (internet není pouze jeden jediný dlouhý lineární text, ale miliardy stránek, které existují vedle sebe a jsou propojeny právě hypertextovými odkazy) (Jacobs, et al., 1999).

HTML používá definované značky (*tagy*) k vytváření a formátování dokumentů pro webové stránky. Mezi otevírací a ukončovací znaky je umístěn text, který definuje tzv. *html* prvek (*element*) spolu s jeho atributy. Atributy slouží k nastavení všech možných vlastností daného *html* prvku jako je výška, šířka, viditelnost apod (Raggett, 1997).

Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka. Proto se jazyk HTML měnil až do dnešní podoby označované jako HTML5. V této podobě přinesl podstatné změny, které zjednodušily formátování webové stránky a tvorbu mnohem přehlednější struktury. Další podstatnou změnou bylo přidání podpory pro přehrávání multimédií přímo ve webovém prohlížeči nebo podpora aplikací, které fungují bez připojení k internetu. Nakonec největší změnou bylo přidání mnoha nových *html* prvků a téměř všem stávajícím prvkům byly přidány nové atributy, které umožnily rozšířit jejich nastavení a tím odstranit některé činnosti. Například u *html* prvku *input* jde pomocí atributu *type* nastavit získání pouze číselné hodnoty, což znamená, že pokud je do tohoto prvku vyplněn alespoň jeden nečíselný znak, prvek jako svou hodnotu vrací prázdný řetězec. Proto není nutné vytvářet funkci, která by ošetřovala, zda daný *html* prvek obsahuje pouze číselné znaky (Hunt, 2010).

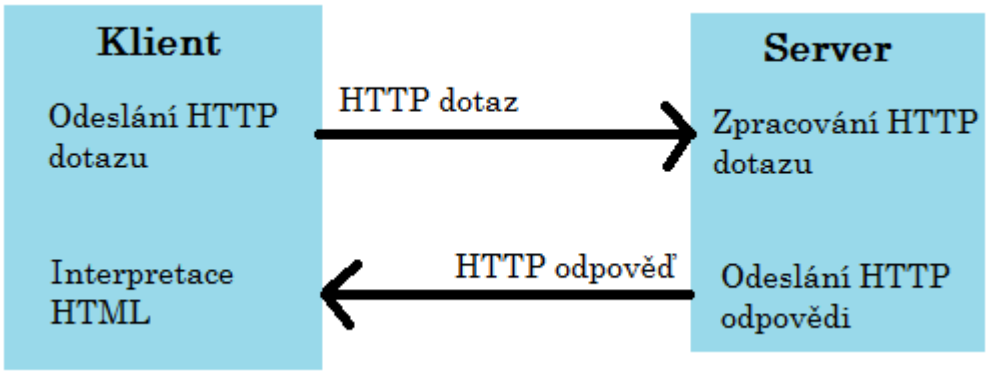
## 2.3 Klient-server architektura

Klient-server je síťová architektura, která odděluje klienta (nejčastěji aplikaci s grafickým uživatelským rozhraním) a server, kteří spolu nejčastěji komunikují pomocí protokolu HTTP přes počítačovou síť.

Architektura klient-server definuje vztah mezi dvěma počítačovými programy nebo aplikacemi, v nichž jeden program reprezentuje klienta, který posílá žádosti o určité služby na jiný program tedy server. Nejznámějším zástupcem tohoto modelu je webový prohlížeč, který reprezentuje program na uživatelském počítači tedy klienta, který přistupuje k libovolnému webovému serveru. Na základě požadavku od klienta webový server získá data, která následně odešle zpět do webového prohlížeče, kde jsou data zobrazena.

Server je v podstatě počítač a programové vybavení, které poskytuje služby buď jiným počítačům, nebo programům. Pasivně naslouchá na síti a reaguje na žádosti od klientů. Po přijetí požadavku jej vyhodnotí a vrací požadovaná data například z databáze uložené přímo na serveru nebo na externím úložišti.

Klient je koncové zařízení uživatele, na kterém běží program či aplikace zajišťující uživatelské rozhraní a aplikační logiku. Aktivně odesílá požadavky na službu poskytovanou serverem a následně čeká na odpověď ze serveru, který mu poskytne vyžádanou službu (Chung, 2000).




Obrázek 1: Komunikace klient-server

### 2.3.1 Protokol HTTP

HTTP (Hypertext transfer protokol) je bezstavový internetový protokol, pomocí kterého klient komunikuje s WWW serverem pro výměnu hypertextových dokumentů ve formátu HTML. V současné době je tento protokol používán i pro přenos dalších informací například díky MIME dokáže přenášet i soubory.

Protokol HTTP funguje na principu dotaz-odpověď. Uživatel pomocí programu odešle dotaz ve formě textu na server, který obsahuje označení požadovaného dokumentu, informace o schopnostech prohlížeče apod. Server na dotaz zareaguje tím, že dotaz vyhodnotí a odešle klientovi odpověď. Tato

odpověď obsahuje několik řádků, které klienta informují o výsledku dotazu, jestli se podařilo žádaný dokument najít nebo jakého typu dokument je. Dále v odpovědi následují samotná vyžádaná data (Marshall, 2010).

Velkým problémem tohoto internetového protokolu je nemožnost serveru rozpoznat, zdali více dotazů od jednoho klienta je na sobě nějak závislých nebo nikoliv, proto se HTTP protokolu také říká bezstavový protokol. Tedy, že protokol neumí uchovávat stav komunikace, a tím nedokáže vyhodnotit souvislost mezi jednotlivými dotazy. Tuto nepříjemnou vlastnost protokolu řeší rozšíření protokolu o HTTP cookies, které umožňují serveru si uchovávat informace o komunikace s daným klientem. Tímto vylepšením je tedy možné například v internetovém obchodě uchovávat informace o stavu nákupního košíku nebo identitě uživatele obchodu (Lampa, 2002).

Protokol umožňuje používat několik dotazovacích metod, které se mají provést nad uvedeným dokumentem. Nejpoužívanějšími metodami jsou metody GET a POST, které umožňují přenosu dat mezi klientem a serverem. Metoda GET umožňuje získat určený objekt či dokument s přidanými daty. Jedná se o nejpoužívanější metodu a to nejen protože se volá při každém zobrazení webové stránky. Druhá metoda POST odesílá uživatelská data na server například skrze webový formulář. Pro odesílání dat na server je možno použít i metodu GET, ale metoda POST je mnohem vhodnější pro velká data (větší než 512 bajtů) nebo pokud nechceme data zobrazovat v adrese URL (Jacobs, 2004).

## 2.4 Knihovna pro tvorbu šablon Smarty

Smarty je systém šablon pro skriptovací jazyk PHP, který umožňuje vkládat do HTML kódu speciální znaky, příkazy a tím vytvářet tzv. šablony. Díky takto upravenému kódu lze jednoduše oddělit aplikační logiku od prezentace dat. Aplikační logika představuje kód programu, který se

například stará o získávání dat z databáze, výpočty apod. Prezentační logika se stará pouze o výslednou podobu dat, tedy například zobrazení získaných dat z databáze do tabulky(Floroiu, 2002).

## 2.5 Protokol z laboratorního cvičení nebo měření

Protokol z měření je jedním z mnoha technických dokumentů, který slouží jako záznam experimentálního zkoumání daného jevu či zadaného úkolu. Nejsou pro něj specifikované žádné přesné náležitosti, a tak je v podstatě rozložení a zvolená forma protokolu na volbě osoby, která protokol vytváří.

Protokol by měl být jasně, stručně a přehledně napsaný, aby se v něm čtenář mohl snadno a rychle orientovat. Měl by obsahovat minimálně tyto části: hlavička, zadání práce, výsledky měření, diskuze a závěr (Klein, 2010) (Šidlof, 2012).

Hlavička je část dokumentu, která obsahuje základní údaje, jako je jméno řešitele úlohy, název úlohy, datum apod. Bývá uvedena na začátku dokumentu pro jednoduchost získání základních údajů.

Zadání práce je nutná část dokumentu, která specifikuje, co všechno se bude během měření provádět a co je účelem měření. Především slouží jako seznam jednotlivých bodů (kroků) měření.

Výsledky měření přehledně uvádějí naměřené hodnoty. Je vhodné tyto údaje zobrazovat ve formě tabulek, grafů nebo obrázků, ze kterých je mnohem jednodušší a rychlejší například pochopit schéma zapojení.

Diskuze porovnává naměřené hodnoty s teoretickou znalostí daného problému. Tato část dokumentu slouží především k obhajobě naměřených hodnot a jejich správnosti vůči teoretickým předpokladům.

Závěr je poslední částí dokumentu, ve kterém shrneme diskuzi a vyhodnotíme splnění jednotlivých úkolů uvedených v zadání práce.

# 3 Praktická část a popis řešení nových komponent

Na základě analýzy poskytnutých materiálů k laboratorním úlohám a novému předmětu PMN byly navrženy komponenty pro inteligentní měřicí systém, které reprezentují jednotlivé dílčí úkoly. Pomocí těchto komponent je možné sestavit jakékoliv poskytnuté laboratorní cvičení.

Jednotlivé komponenty jsou vždy tvořeny minimálně ze čtyř souborů, které oddělují aplikační a prezentační část. Dále mohou využívat externí knihovny. Komponenta je tvořena tzv. tělem, což je soubor v jazyce PHP, který je rozšířením abstraktní třídy Task (součást systému). V tomto souboru lze provádět doplňující operace pro správnou funkcionalitu komponenty. Soubor také obsahuje tři metody, které zobrazují grafické šablony určené pro vyučujícího, studenta nebo výsledný dokument.

Šablona **teacher.tpl** slouží k zobrazení formuláře pro nastavení parametrů komponenty vyučujícím. Šablona **student.tpl** zobrazuje výslednou podobu komponenty a získává zadaná data studentem. Oba soubory obsahují, povinou JS funkci **task.getData**, která uloží data do proměnné **$settings** a nebo **$data**, pokud funkce proběhne v pořádku, zavolá se funkce **task.save**, která uloží danou proměnnou do databáze systému. Poslední šablona **document.tpl** obsahuje výsledný HTML kód reprezentující výstup komponenty, který je doplněný o získaná data. Tento HTML kód je po ukončení cvičení převeden do souboru PDF pomocí knihovny TCPDF, která je součástí systému.

## 3.1 Komponenta TaskInput

Komponenta **TaskInput** je jedna z nejjednodušších komponent, která realizuje vstupní prvky formuláře, které slouží jako číselné nebo slovní odpovědi či delší souvislý text k zhodnocení výsledků. Dále může sloužit i jako formulářový prvek, na kterém jsme schopni ověřit teoretické znalosti uživatele.

Soubor **teacher.tpl** obsahuje tři formulářové prvky, které slouží k definici parametrů nastavujících výsledný vzhled komponenty. Prvním parametrem je **frontText**, který obsahuje textový řetězec zobrazený před vstupním formulářovým prvkem. Druhým povinným parametrem je **type**, do kterého se uloží pomocí *html* elementu *select* jeden ze tří typů - **number**, **text** nebo **textarea**. Posledním parametrem je **backText**, který není nutné vyplňovat. Pokut je *html* element *input* pro parametr **backText** vyplněn, bude obsahovat textový řetězec, který se zobrazí za vybraným typem vstupního formulářového prvku. Všechny tři parametry jsou uloženy pod proměnou **$settings**.

V souboru **student.tpl** na základě nastavujících parametrů předaných v proměnné **$settings** se vygeneruje výsledný *html* kód. Jelikož *html* element *input* v atributu *type* nemá definovanou hodnotu *textarea,* je nutné skript rozdělit podmínkou, jestli je v proměnné **$settings.type** uložen textový řetězec **textarea**, pokud ano dojde k vygenerování *html* elementu *textarea*, sloužícímu k zadávání dlouhých textových řetězců. Pokud není podmínka splněna, dojde k vygenerování *html* elementu *input* s atributem *type* nastaveným na hodnotu, která je uložena v proměnné **$settings.type**. Před a za vygenerovaný *html* element jsou vloženy textové řetězce uložené v proměnných **$settings.frontText** a **$settings.backText**. Údaje vyplněné do vygenerovaného vstupního *html* elementu jsou pomocí JS ověřeny a uloženy do proměnné **$data**.


Obrázek 2: Ukázka komponenty TaskInput

V souboru **document.tpl** se do html kódu vloží výsledný textový řetězec složený z textových řetězců uložených v proměnných **$settings.frontText**, **$data.value** a **$settings.backText**.

## 3.2 Komponenta TaskFormula

Komponenta **TaskFormula** realizuje jeden vstupní formulářový prvek, který slouží pro zadání matematického či fyzikálního vzorce nebo slouží k realizaci výsledku z minimalizace pravdivostní tabulky. Zadaný text je následně na straně serveru převeden do podoby, kterou nám umožňuje používat textový procesor pro reprezentaci vzorců. Této podoby dosáhneme za pomoci knihovny **mathpublisher** (Brachet, 2005), která z textového řetězce vytvoří obrázek obsahující výsledný vzorec, který svým vzhledem odpovídá výsledku z editoru rovnic.

Soubor **teacher.tpl** obsahuje pouze jeden vstupní formulářový prvek, který je povinný a je nutné jej správně vyplnit. Jediným parametrem, který se nastavuje je parametr **textFormula**. Tento parametr obsahuje řetězec zadaný pomocí *html* elementu *input* a je zobrazen před vstupním prvkem formuláře, který je určen pro zadání vzorce. Parametr po ověření je uložený pod proměnou **$settings**.

V souboru **student.tpl** se na základě nastavujících parametrů předaných v proměnné **$settings** vygeneruje výsledný *html* kód, který se vloží jako celek do webového formuláře. Protože se jedná o velice jednoduchou komponentu, dojde pouze k dosazení textového řetězce z proměnné **$settings.textFormula** před *html* element *input* s parametrem *type*, který je nastaven na *text*. Komponenta je doplněna o odkaz na webovou stránku knihovny **mathpublisher**, na které se student dozví jak správně psát vzorce tak, aby výsledek odpovídal jeho představě. Textový řetězec vyplněný do vygenerovaného vstupního *html* elementu *input* je pomocí JS funkce ověřen a uložen do proměnné **$data**.


Obrázek 3: Ukázka komponenty TaskFormula

Než jsou nastavující parametry uložené v proměnné **$settings** a data zadaná studentem uložená v proměnné **$data** odeslána do šablony, je nutné vygenerovat obrázek reprezentující zadaný vzorec studentem pomocí funkce **mathImage** z knihovny **mathpublisher**. Následně se obrázek převede na textový formát Base64 a je odstraněn ze serveru. Textový řetězec se jako proměnná **$base64** odešle do šablony.

V souboru **document.tpl** se na základě nastavujících parametrů předaných v proměnné **$settings** a datech předaných v proměnné **$base64** do *html* kódu vloží textový řetězec uložený v proměnné **$settings.textPicture**. Dále se do atributu *src* u *html* elementu *img* vloží výsledný textový řetězec ve formátu Base64, který vznikl za pomoci knihovny **mathpublisher**. Pomocí tohoto textového řetězce dojde k vygenerování obrázku reprezentujícímu vzorec zadaný studentem.

## 3.3 Komponenta TaskTable

Komponenta **TaskTabe** realizuje tabulku vstupních prvků formuláře, které slouží k zápisu číselných hodnot do tabulky. Z této tabulky pak může nebo nemusí být vykreslen graf změřených hodnot.

Soubor **teacher.tpl** obsahuje šest formulářových prvků, které slouží k definici parametrů nastavujících výsledný vzhled komponenty. Prvním parametrem je číselná hodnota **x**, která obsahuje číselnou hodnotu nastavující počet řádků výsledné tabulky. Druhým povinným parametrem je číselná hodnota **y**, ve kterém je uložena číselná hodnota nastavující počet sloupců výsledné tabulky. Dalším parametrem je **type**, do kterého se uloží pomocí *html*

elementu *select* jeden ze dvou typů - **table** nebo **graph**. Tento parametr určuje, zda výsledná tabulka v dokumentu bude zobrazena samotná nebo spolu se spojnicovým grafem vyhotoveným na základě tabulky. Dalším parametrem je textový řetězec **title**, ve kterém je uložen název tabulky a popřípadě i grafu. Posledními dvěma parametry jsou textové řetězce **xTh** a **yTh**, ve kterých jsou uloženy souvislé textové řetězce pro nadpisy řádků a sloupců tabulky. Parametr **yTh** není povinný, proto může zůstat nevyplněný. Všech šest parametrů je uloženo pod proměnou **$settings**.

Na základě nastavujících parametrů uložených v proměnné **$settings** je nutné si připravit parametry, které ulehčí vygenerování výsledného *html* kódu. Prvním parametrem je číselná hodnota jedna nebo nula, která nahrazuje logické hodnoty **true** nebo **false**. Tento parametr se nastaví na jedničku, pokud vyučující při vytváření komponenty vyplnil nepovinný parametr uložený v proměnné **$settings['yTh'],** v opačném případě se nastaví na nulu. Dále dojde k rozdělení nadpisů v řádcích a sloupcích uložených v proměnných **$settings['xTh']** a **$settings['yTh']** do polí, tak aby každý prvek pole obsahoval právě jeden nadpis k řádku nebo sloupce. Pole pro řádky je uloženo do proměnné **$rowNames** a pole pro sloupce do proměnné **$colNames**.

V souboru **student.tpl** na základě nastavujících parametrů předaných v proměnné **$settings** a z parametrů, které se vygenerovaly před zavoláním šablony, se vygeneruje výsledný *html* kód reprezentující tabulku. Pokud není pole obsahující jednotlivé nadpisy pro sloupce prázdné, dojde k vytvoření řádku o počtu sloupců rovném hodnotě uložené v proměnné **$settings.y** navýšené o jedničku, protože první sloupec obsahuje nadpisy řádků, zůstává prázdný, a první řádek se proto vyplňuje až od druhého políčka. Do každého políčka řádku se vloží jeden prvek z pole s nadpisy pro sloupce, který je uložen v proměnné **$colNames**. Dále se provede postupné generování jednotlivých řádků až po poslední řádek tabulky, jehož číselná hodnota je uložena

v proměnné **$settings.x**. Do prvního sloupce v řádku se vloží prvek z pole pro nadpisy řádků uloženého v proměnné **$rowNames**. Do zbývajících sloupců je vložen *html* element *input* s atributem *type* nastaveným na hodnotu *number*, který slouží k vyplňování číselné hodnoty. Údaje získané ze vstupních *html* elementů input jsou pomocí JS ověřeny a uloženy do proměnné **$data**.


Obrázek 4: Ukázka komponenty TaskTable

Před odesláním proměnných do šablony je nutné vygenerovat graf pomocí knihovny **phpgraphlib** (Brueggeman, 2014). Tato knihovna poskytuje funkce na nastavení názvu grafu, typu grafu, dosazení hodnot do osy x a y, nastavení barev apod. Graf je vygenerován na základě hodnot v proměnné **$data** a uložen na serveru jako obrázek. Poté je převeden na textový řetězec ve formátu Base64 a odstraněn ze serveru. Textový řetězec je předán do šablony pod proměnnou **$base64**.

Soubor **document.tpl** je upravenou kopií souboru **student.tpl**, který na základě parametrů předaných v proměnné **$settings**, přidaným parametrům uložených v proměnných **$base64**, **$colNames** a **$rowNames** a dále pak dat získaných od studentů uložených v proměnné **$data,** dojde k vygenerování výsledného *html* kódu reprezentujícímu tabulku. Tento výsledný kód se generuje podobně jako v souboru **student.tpl** až na několik rozdílů, a to zejména, že do sloupců, ve kterých byl vygenerován *html* element *input*, je vložena číselná hodnota z pole hodnot vyplněného studentem a uloženého v proměnné **$data**. Druhým rozdílem je to, že pokud je v proměnné **$settings.type** nastaven parametr **graph,** dojde k vygenerování obrázku

reprezentujícímu graf na základě textového řetězce ve formátu Base64 uloženého v proměnné **$base64**.

## 3.4 Komponenta TaskThruthTable

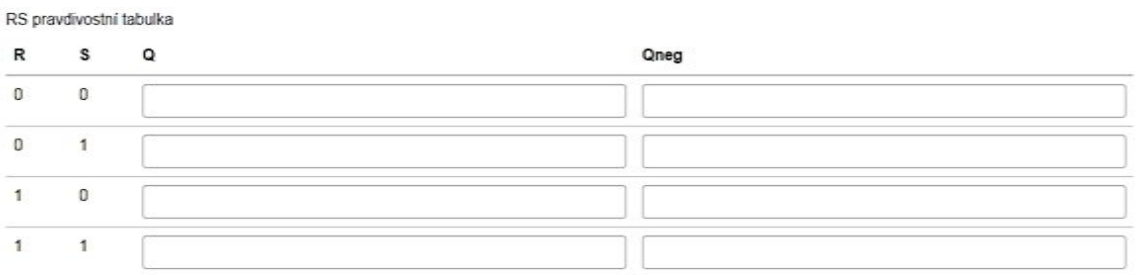
Komponenta **TaskThruthTable** realizuje tabulku vstupních prvků, které slouží k zápisu logických hodnot, stavů logického obvodu nebo jen samostatného logického hradla do pravdivostní tabulky.

Soubor **teacher.tpl** obsahuje čtyři povinné formulářové prvky, které slouží k definici parametrů nastavujících výsledný vzhled komponenty. Prvním parametrem je číselná hodnota **x**, který určuje kolik logických vstupů má logický obvod nebo hradlo. Druhým parametrem je číselná hodnota **y**, která určuje kolik logických výstupů má logický obvod nebo hradlo. Dalším parametrem je textový řetězec **title**, který obsahuje název pravdivostní tabulky. Posledním parametrem je textový řetězec **yTh**, který obsahuje v jednom souvislém řetězci nadpisy pro sloupce s logickými vstupy a výstupy. Všechny tři parametry jsou uloženy pod proměnou **$settings**.

Než jsou veškeré nastavující parametry z proměnné **$settings** odeslány do šablony, je nutné si vypočítat počet řádků ve výsledné pravdivostní tabulce podle vztahu **2^$settings['x']**. Dále pak převést čísla řádků z desítkové soustavy do soustavy binární. Tyto převedené hodnoty pak ještě rozdělit po jednotlivých číslicích do pole a doplnit chybějící nuly, tak aby součet všech jednotlivých číslic reprezentujících hodnotu řádku převedeného do binární soustavy byl roven počtu logických vstupů uložených v proměnné **$settings['x']**. Dále je nutné rozdělit nadpisy sloupců z jednoho souvislého řetězce uloženého v proměnné **$settings['yTh']** do pole, tak aby každý prvek pole obsahoval právě jeden nadpis ke sloupci. Tyto upravené a přidané parametry jsou nutné pro správné vygenerování výsledného *html* kódu. Proto je posíláme do šablony spolu s nastavujícími parametry, které jsou již uloženy v proměnné **$settings**.

Výsledný počet řádků pravdivostní tabulky se posílá uložený v proměnné **$pow**, pole s binárními hodnotami reprezentující jednotlivá čísla řádků uložená v proměnné **$value** a pole s jednotlivými nadpisy ke sloupcům uložené v proměnné **$colNames**.

V souboru **student.tpl** na základě nastavujících parametrů předaných v proměnné **$settings** a parametrů, které se vygenerovaly před zavoláním šablony, je vygenerován výsledný *html* kód reprezentující pravdivostní tabulku. Na základě počtu logických vstupů z proměnné **$settings.x** a počtu logických výstupů z proměnné **$settings.y** dostaneme celkový počet sloupců pravdivostní tabulky. Do prvního řádku, který není počítán do celkového počtu řádků pravdivostní tabulky je vložen postupně do každého sloupce jeden prvek z pole s nadpisy, který je uložen v proměnné **$colNames**. Dále se provede postupné generování jednotlivých řádků až po poslední řádek pravdivostní tabulky, jehož číselná hodnota je uložena v proměnné **$pow**. Do sloupců, které mají svoji hodnotu menší, než je číselná hodnota logických vstupů uložená v proměnné **$settings.x,** se vloží jednotlivé číslice uložené v proměnné **$value** reprezentující binární hodnotu daného řádku. Do zbylých sloupců se vygeneruje *html* element *input* s nastaveným atributem *type* na hodnotu *text*, protože pravdivostní tabulka může obsahovat například zakázaný stav, hodnotu *low* a *high* určující připojené napětí. Údaje získané ze vstupních *html* elementů input jsou pomocí JS ověřeny a uloženy do proměnné **$data**.


RS pravdivostní tabulka

| R | S | Q | Qneg |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

Obrázek 5: Ukázka komponenty TaskThruthTable

Soubor **document.tpl** je upravenou kopií souboru **student.tpl**, který na základě parametrů předaných v proměnné **$settings**, přidaným parametrům uložených v proměnných **$pow**, **$value** a **$colNames**. Dále pak dat získaných od studentů a uložených v proměnné **$data** dojde k vygenerování výsledného *html* kódu. Tento výsledný kód se generuje podobně jako v souboru **student.tpl** až na jediný rozdíl, a to že do sloupců, ve kterých byl generován *html* element *input*, je vložena číselná nebo textová hodnota z pole hodnot vyplněného studentem a uloženého v proměnné **$data**.

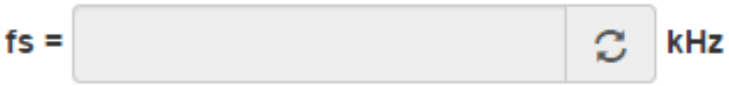
## 3.5 Komponenta TaskDeviceValue

Komponenta **TaskDeviceValue** realizuje vstupní formulářový prvek s tlačítkem, které slouží pro získání číselné nebo textové hodnoty z měřicího přístroje či z osciloskopu připojeného k počítači, na kterém je realizované měření.

Soubor **teacher.tpl** obsahuje tři povinné formulářové prvky, které slouží k nastavení parametrů komponenty. Prvním parametrem je textový řetězec **textBefore**. Druhým parametrem je textový řetězec **command**, do kterého se uloží příkaz, který je poslán do měřicího přístroje. Posledním parametrem je textový řetězec **textAfter**, který není povinný, a tedy nemusí být vyplněn. Všechny tři parametry jsou uloženy pod proměnou **$settings**.

V souboru **student.tpl** dojde k doplnění textového řetězce před *html* element *input* z proměnné **$settings.textBefore**. Pokud není prázdný textový řetězec uložený v proměnné **$settings.textAfter**, je vložen za *html* element *input*. Poslední součástí komponenty je tlačítko, které aktivuje JS funkci, která se také nachází v souboru. Tato funkce při aktivaci vyvolá dialogové okno, ve kterém si student vybere přístroj ze seznamu, na který se odešle příkaz uložený v proměnné **$settings.commnad**. Seznam přístrojů je získán pomocí funkce uložené v systémovém souboru **app/api/student/device.php**, která získá

seznam přístrojů pro daný počítač z databáze systému. Po vybrání přístroje je pomocí funkce **event.add** ze studentského API vložena do databáze systému událost s funkcí **callback**. Tato funkce se provede tehdy, když server obdrží příslušná data z měřicího přístroje. Funkce **callback** provede vložení získaných dat do *html* elementu *input* a zároveň získaná data uloží do proměnné **$data**.




Obrázek 6: Ukázka komponenty TaskDeviceValue

V souboru **document.tpl** se do *html* kódu vloží textový řetězec, který je složený z textových řetězců uložených v proměnných **$settings.textBefore**, **$data** a **$settings.textAfter**.

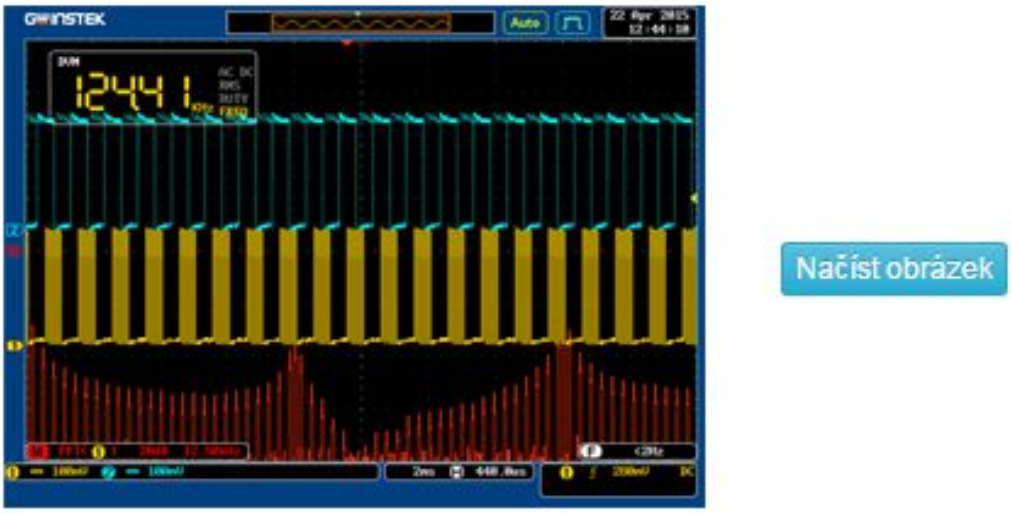
## 3.6 Komponenta TaskOscilloscope

Komponenta **TaskOscilloscope** realizuje tlačítko, které slouží pro získání snapshotu (kopie obrazovky) z osciloskopu uloženém v databázi systému a připojeného k počítači, na kterém je realizováno měření.

Soubor **teacher.tpl** obsahuje jeden povinný vstupní parametr, který nastavuje velmi důležitou vlastnost komponenty. Jedná se o parametr **command**, do kterého se uloží příkaz, který se pošle do vybraného osciloskopu. Zvolený parametr **command** se uloží pod proměnou **$settings**.

Soubor **student.tpl** obsahuje *html* kód reprezentující *html* element *img*, který je pomocí atributu *style* obsahujícího nastavení kaskádových stylů (CSS) nastaven na **display: none**. Toto nastavení zaručuje, že pokud není získaný obrázek (snapshot), je tento element skrytý. Dále soubor obsahuje *html* element *button*, který aktivuje JS funkci pro získání obrázku z osciloskopu. Tato funkce funguje stejně jako u komponenty **TaskDeviceValue** kromě návratové funkce **callback**, která získaný textový řetězec ve formátu Base64 vloží do proměnné

**$data,** a zároveň do atributu *src* u *html* elementu *img*, u kterého nastaví hodnotu v atributu *style* na **display: inline,** a tak dojde k zobrazení obrázku studentovi.




Obrázek 7: Ukázka komponenty TaskOscilloscope

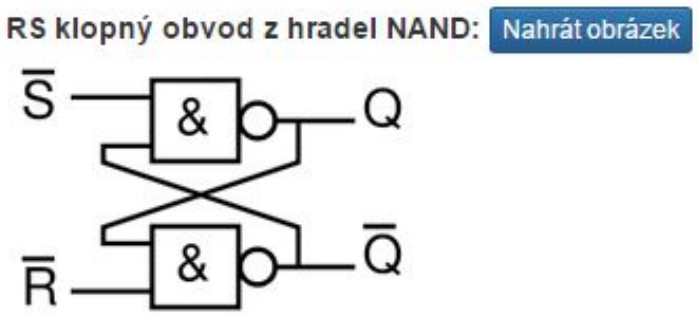
V souboru **document.tpl** dojde k vygenerování obrázku pomocí textového řetězce ve formátu Base64 z proměnné **$data** vloženého do atributu *src* u *hml* elementu *img*.

## 3.7 Komponenta TaskPicture

Komponenta **TaskPicture** realizuje vstupní prvek formuláře, který slouží pro vložení souboru, v našem případě se jedná o soubor typu obrázek. Tato komponenta slouží především pro vkládání obrázků ze speciálních, simulačních či jiných programů, které se nám nepodařilo nahradit jinou komponentou nebo které budou přidání v rámci rozvíjení cvičení z měření.

Soubor **teacher.tpl** obsahuje pouze jediný a povinný vstupní formulářový prvek, který slouží k definici parametru komponenty. Tímto jediným parametrem je **textPicture**, ve kterém je uložen textový řetězec. Parametr je po správném vyplnění uložen pod proměnou **$settings**.

V souboru **student.tpl** na základě nastavující parametrů předaných v proměnné **$settings** se vygeneruje výsledný *html* kód. Protože se jedná o velice jednoduchou komponentu dojde pouze k vložení textového řetězce z proměnné **$settings.textPicture** před *html* element *input* s atributem *type*, který je nastaven na *file*. Dalším atributem je *accept*, který specifikuje, jaký typ souboru určený příponou souboru server přímá, je nastaven na jakýkoliv obrázek pomocí hodnoty **image/***. Dále je zde *html* element *img*, který je stejný jako *html* element *img* v souboru **student.tpl** u **TaskOscilloscope**. Součástí souboru je i funkce napsaná v jazyce JS, která po každém vybrání souboru přes vstupní *html* element *input* zavolá funkci uloženou v souboru **app/api/student/file.php**. Tato funkce ověří, zda je vkládaný soubor skutečně typu *image* a pokud ano, tak provede jeho nahrání na server. Následně obrázek převede na textový formát Base64 a obrázek smaže ze serveru. Výsledný textový řetězec je vrácen do JS funkce. Funkce tento řetězec vloží do proměnné **$data** a do atributu *src* u *html* elementu *img*, tak dojde k vykreslení nahraného obrázku. Následně je změněn atribut *style* na **display: inline**, tímto nastavením se nahraný obrázek zobrazí studentovi hned pod vstupním prvkem formuláře.




Obrázek 8: Ukázka komponenty TaskPicture

V souboru **document.tpl** se do *html* kódu vloží textový řetězec uložený v proměnné **$settingstextPicture** a obrázek vytvořený pomocí textovému řetězci ve formátu Base64, který je uložen v proměnné **$data**.

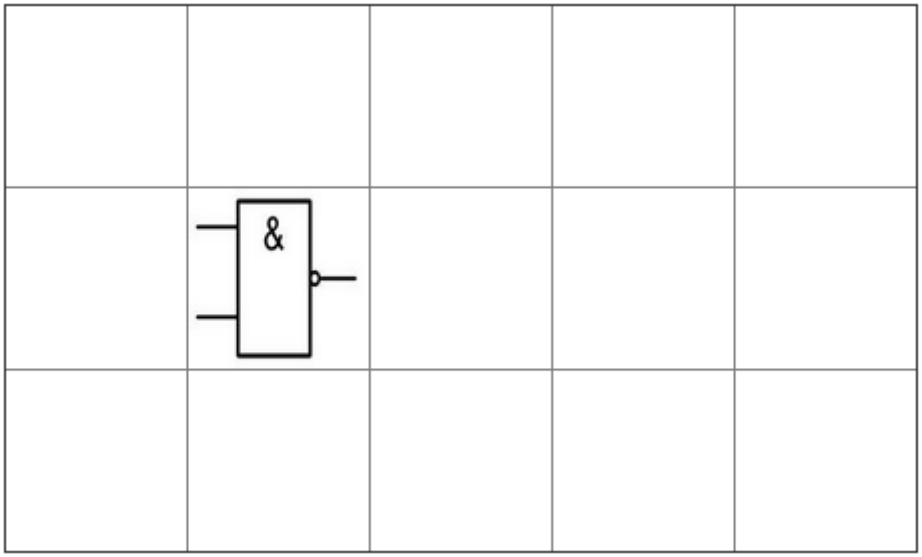
## 3.8 Komponenta TaskCreatePicture

Komponenta **TaskCreatePicture** realizuje jednoduchý návrhový prvek, který slouží k návrhu logického obvodu za pomoci seznamu obrázků. Výsledný obrázek je tvořen tabulkou uživatelem vybraných obrázků uložených v databázi serveru.

Soubor **teacher.tpl** obsahuje tři povinné formulářové prvky, které slouží k nastavení parametrů komponenty. Prvním parametrem je číselná hodnota **row**, která určuje počet řádků návrhové tabulky. Druhým parametrem je číselné hodnota **col**, která určuje počet sloupců návrhové tabulky. Posledním parametrem je pole **images**, které obsahuje textové řetězce ve formátu Base64. Vybrané obrázky jsou pomocí JS funkce nahrány na server, následně dekódovány do formátu Base64 a opětovně smazány. Nakonec jsou textové řetězce vráceny a uloženy do pole **images**. Ke každému řetězci se vygeneruje jedinečná kombinace znaků, která slouží k jednoznačné identifikaci obrázku. Pomocí těchto řetězců jsou reprezentovány jednotlivé vybrané obrázky, ze kterých si bude moci student vybrat při tvorbě návrhu logického obvodu. Všechny tři parametry jsou uloženy pod proměnou **$settings**.

V souboru **student.tpl** dojde na základě nastavujících parametrů předaných v proměnné **$settings** k vygenerování tabulky s počtem řádků uložených v proměnné **$settings.row** a počtem sloupců uložených v proměnné **$settings.col**. Do každého políčka tabulky je vložen *html* element *img*, který slouží k zobrazování prázdného nebo vybraného obrázku a *html* element *input* s parametrem *type* nastaveným na *hidden*, který je skrytý a slouží k ukládání textového řetězce pro *html* element *img*. Součástí souboru je i JS funkce, která po kliknutí na jakýkoliv obrázek z tabulky vyvolá dialogové okno. V tomto okně si student vybere z obrázků uložených v proměnné **$settings.images**, které vyučující nahrál do databáze systému při vytváření této komponenty. Po vybrání a potvrzení se textový řetězec ve formátu Base64 reprezentující

vybraný obrázek vloží do skrytého *html* elementu *input* a do atributu *src* u *html* elementu *img* do políčka tabulky, ve kterém leží obrázek, na který student kliknul. Tím dojde k nahrazení předchozího obrázku obrázkem vybraným v dialogovém okně. Při každé změně obsahu tabulky dojde k uložení komponenty, tedy dojde k uložení všech textových řetězců umístěných v *html* elementech *input* do proměnné **$data**.




Obrázek 9: Ukázka komponenty TaskCreatePicture

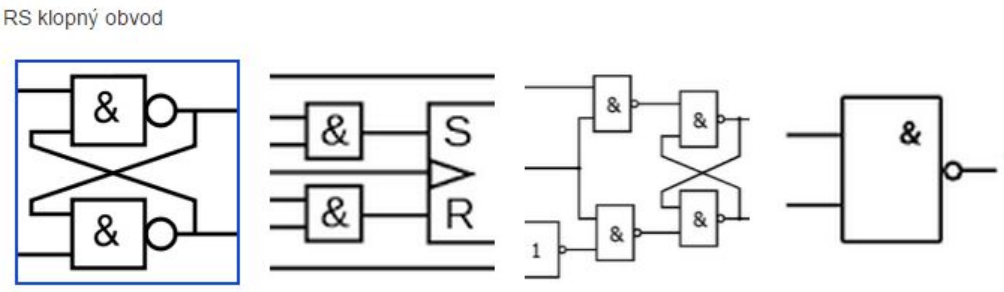
V souboru **document.tpl** se vygeneruje stejná tabulka jako v souboru **student.tpl**, která se liší pouze v jedné věci. A to, že do parametru *src* u *html* elementu *img*, který je umístěn v každém políčku tabulky, je vložen textový řetězec ve formátu Base64 z proměnné **$data.**

## 3.9 Komponenta TaskChoose

Komponenta **TaskChoose** realizuje výběr jednoho z několika obrázků, které jsou předem vybrány vyučujícím při nastavování komponenty a uloženy v databázi systému. Tato komponenta může dále sloužit i jako jednoduchý ukazatel, zda je student vyhotovující dané cvičení znalý problematiky související se cvičením.

Soubor **teacher.tpl** obsahuje dva povinné formulářové prvky, které slouží k nastavení komponenty. Prvním nastavovaným parametrem je textový řetězec **textBefore**. Druhým parametrem je pole **images**, které obsahuje textové řetězce ve formátu Base64 reprezentující vybrané obrázky. Součástí je i JS funkce, která již byla popsána v odstavci pro soubor **teacher.tpl** u komponenty **TaskCreatePicture**.

V souboru **student.tpl** je do *html* kódu vložen textový řetězec z proměnné **$settings.textBefore** informující studenta, který obrázek má vybrat. Dále je vygenerován *html* kód pro každý obrázek z proměnné **$settings.images**, který obsahuje *html* element *input* a *img*. *Html* element *input* má nastavený atribut *value* na speciální kombinaci znaků, která slouží k jednoznačné identifikaci obrázků. Dále je atribut *type* nastaven na *radio* a všechny tyto *html* elementy *input* mají nastavenou stejnou hodnotu v atributu *name*. Takto nastavené *html* elementy zaručují, že může být vybrán pokaždé pouze jeden. Do atribut *src* u *html* elementu *img* vložen textový řetězec ve formátu Base64 reprezentující obrázek z pole uloženého v proměnné **$settings.images**. Pomocí CSS jsou nastaveny veškeré html elementy kromě img aby nebyly viditelné. Po vybrání obrázku se kolem něj vytvoří modrý rámeček odlišující vybraný obrázek. Textový řetězec z atributu *value* u *html* elementu *input*, který patří k vybranému obrázku je uložen od proměnné **$data**.




Obrázek 10: Ukázka komponenty TaskChoose

V souboru **document.tpl** je do *html* kódu vložen textový řetězec z proměnné **$settings.textBefore** dále je do atributu *src* u *html* element *img* vložen textový řetězec ve formátu Base64 z proměnné **$settings.image**, který je identifikovaný speciální kombinací znaků uloženou v proměnné **$data**.

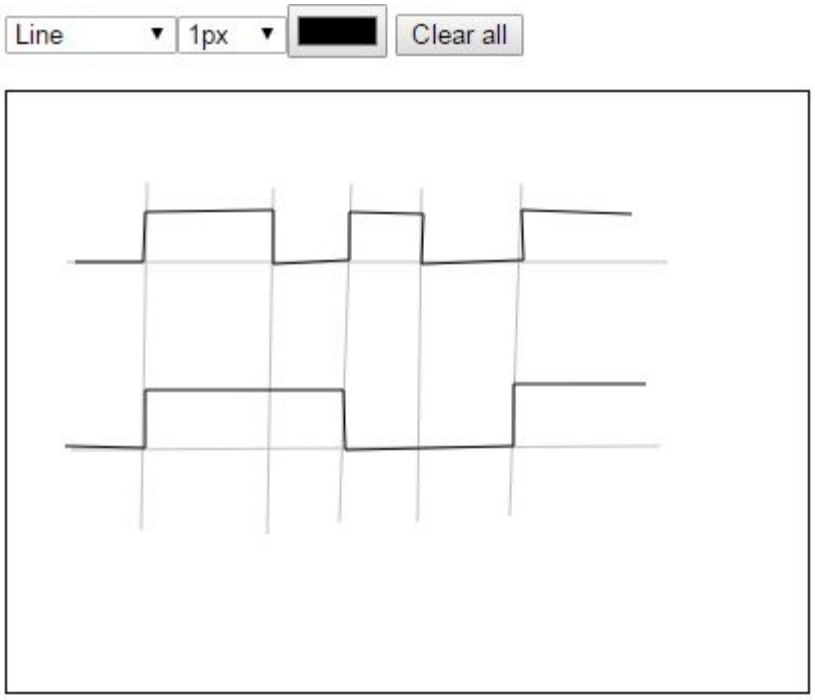
## 3.10 Komponenta TaskCanvas

Komponenta **TaskCanvas** realizuje jednoduchý grafický nástroj vytvořený za pomoci *html* elementu *canvas*, který slouží k jednoduchému grafickému zhotovení schématu logického obvodu, průběhu signálu, závislosti výsledného signálu v logickém obvodu apod.

Soubor **teacher.tpl** obsahuje pouze text informující vyučujícího o vlastnostech této komponenty. Ačkoliv se nejedná o jednoduchou komponentu, nejsou nutné žádné nastavující parametry, které by ovlivňovaly výsledný vzhled komponenty.

Soubor **student.tpl** obsahuje pět *html* elementů, které slouží k funkčnosti této komponenty. Prvním prvek je *html* element *select*, který nastavuje jeden ze čtyř kreslících nástrojů - tužku, přímku, obdélník a kružnici. Druhým *html* elementem je znovu *select*, který nastavuje šířku čáry vykreslovaného obrazce v *pixelech*. Dalším prvkem je *html* elementem input s atributem *type* nastaveným na *color*, který nastavuje barvu vykreslovaného obrazce. Předposledním prvkem je *html* element *button*, který slouží jako tlačítko pro vymazání nakresleného obrázku. Posledním prvkem je *html* element *canvas*, do kterého pomocí kódu v jazyce JS můžeme kreslit různé obrazce. Součástí souboru je také JS funkce, která nám pomůže nastavit a pracovat s *html* elementem *canvas*. V této funkci nejprve proběhne inicializační část, která má na starosti definici vnitřních proměnných. Dále jsou získány *html* elementy pomocí funkce **getElementById** a jsou jim přidány události, na které se vyvolá akce pomocí funkce **addEventListener**. Například při změně šířky čáry dojde k nastavení

jiné velikosti u proměnné **lineWidth** pro vykreslení do *html* elementu *canvas* nebo při stisknutí tlačítka pro smazání dojde k odstranění všech vykreslených obrazců. V neposlední řadě dojde k definování jednotlivých kreslících nástrojů, tak že každému nástroji nastavíme pomocí funkce **addEventListener** událost **mouseDown**, **mouseMove** a **mouseUp**. **MouseDown** je událost, která se aktivuje při stisknutí levého tlačítka myši v oblasti *html* elementu *canvas*. **MouseMove** je událost, která se aktivuje při pohybu myší přes *html* element *canvas* a událost **mouseUp** se aktivuje při uvolnění levého tlačítka myši v oblasti *html* elementu *canvas*. U každého kreslícího nástroje se funkcionalita na jednotlivé události liší v závislosti na tom, o jaký vykreslovaný obrazec se jedná. Při vykreslení jakéhokoliv obrazce dojde k uložení výsledného obrázku pomocí funkce **toDataURL**, která nám vrátí textový řetězec ve formátu Base64.




Obrázek 11: Ukázka volného kreslení v komponentě TaskCanvas

V souboru **document.tpl** na základě dat předaných v proměnné **$data** se do atributu *src* u *html* elementu *img* vloží textový řetězec ve formátu Base64, který vznikl během návrhu studenta v komponentě. Pomocí tohoto textového

řetězce dojde k vygenerování obrázku reprezentujícímu obrázek navržený studentem.

## 3.11 Komponenta TaskEndOfPage

Komponenta **TaskEndOfPage** realizuje pouze grafický prvek, který slouží jako oddělovač pro zalomení stránky ve výsledném PDF souboru. Především je to z důvodu zachování formálnosti dokumentu.

Soubor **teacher.tpl** obsahuje pouze text informující vyučujícího o vlastnostech této komponenty. Protože se jedná o velmi jednoduchou komponentu, není nutné zadávat žádné nastavující parametry, které by nějak ovlivnily výsledný vzhled.

Tato komponenta má nastavenou vlastnost **stundentTemplateDisplay** na logickou hodnotu **false**. To znamená, že soubor **student.tpl** se studentovi při vyplňování vygenerovaného formuláře pro cvičení nezobrazí a tím nijak neovlivňuje celistvost formuláře. Proto tento soubor nemusí obsahovat žádný *html* kód, který by se zobrazoval studentovi.

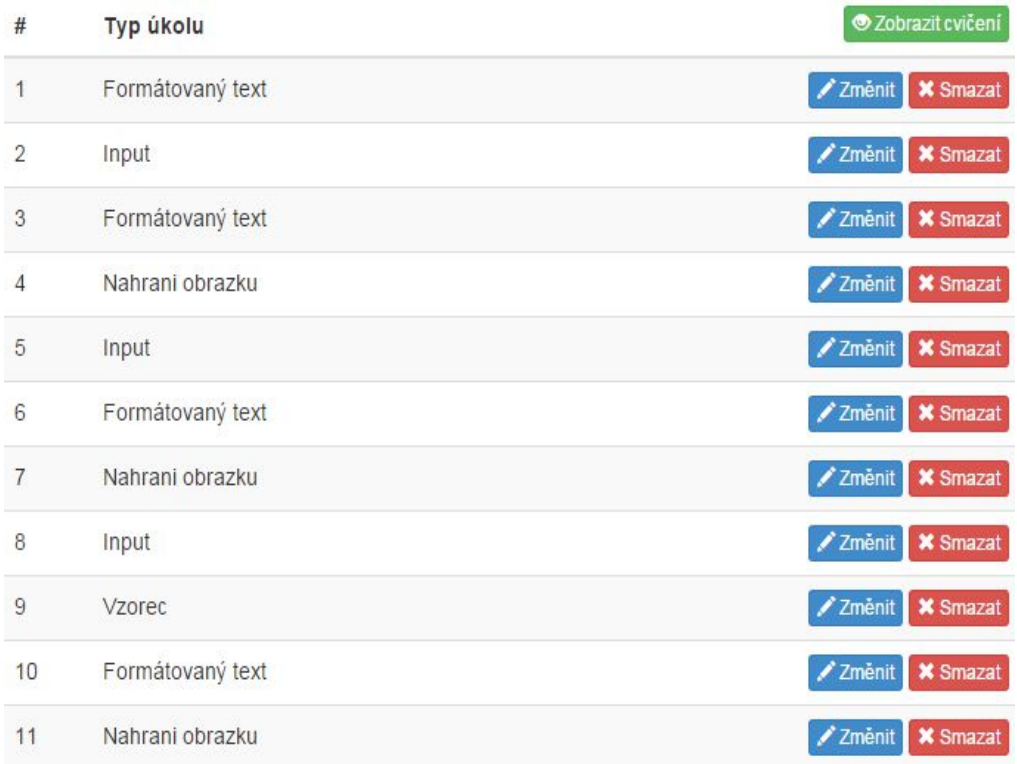
Pro zalomení stránky je využito vlastnosti knihovny TCPDF, která se stará o převod *html* kódu do podoby souboru PDF. V souboru **document.tpl** je vložen pouze jeden *html* element pro nový řádek s pomocným atributem **pagebreak** nastaveným na logickou hodnotu **true**, který při převodu *html* kódu do PDF souboru aktivuje vlastnost knihovny TCPDF a ta vytvoří zalomení stránky. Další komponenty následující za komponentou **TaskEndOfPage** tedy začínají na nové stránce dokumentu.

## 3.12 Příklad konečného řešení

Na základě poskytnutých protokolů z laboratorních cvičení a šablon připravených pro vyplňování naměřených hodnot, schémat zapojení logických obvodů a zapisování pravdivostních tabulek, bylo vytvořeno celkem třináct

cvičení, které nahrazují tyto dokumenty webovým formulářem tvořeným komponentami pro realizaci protokolu z měření.

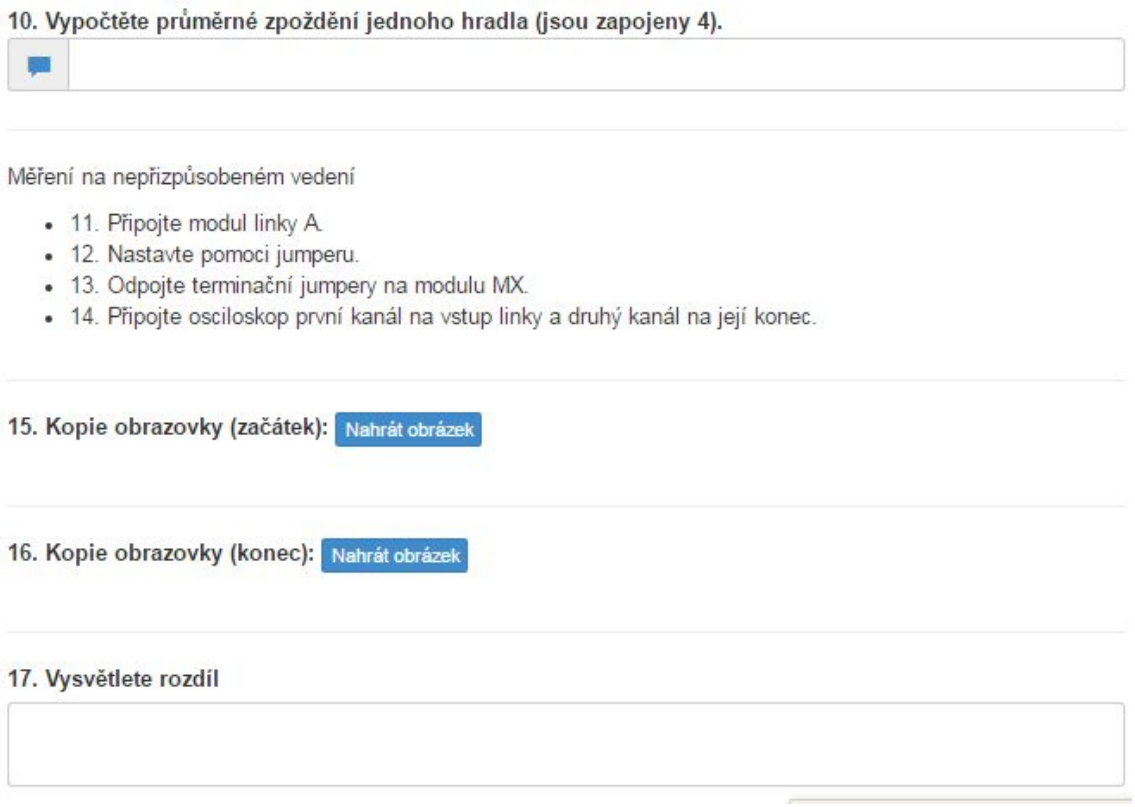
Jako ukázka bylo vybráno cvičení PMN_09, které v případě vyučujícího vypadá jako seznam po sobě jdoucích komponent, které mají nastavené parametry přesně podle předlohy uložené v dokumentech se šablonami pro elektronická cvičení. I když stejné komponenty mají identický název v seznamu, jejich reprezentace se liší právě na základě nastavujících parametrů, tedy je-li v seznamu dvakrát uvedená komponenta *Input* (TaskInput), tak jedna může reprezentovat zadávání číselné hodnoty a druhá zadávání delšího textu například pro zhodnocení výsledků. Je zde přidáno tlačítko Zobrazení cvičení, které umožňuje vyučujícímu zobrazit si cvičení v podobě, jakou vidí student.




Obrázek 12: Část seznamu komponent při vytváření cvičení

Studentovi nebo řešiteli se cvičení zobrazí jako jeden velký webový formulář, kde jsou jednotlivé úkoly reprezentované nastavenými komponentami, od sebe odděleny pomocí vykreslené čáry. Při každé změně,

která je provedena při vyplňování dat do vstupních *html* prvků komponent je zavolána funkce, která vyhodnotí změnu a je-li daný *html* prvek správně vyplněn, jeho hodnota je uložena do databáze systému, pokud není správně vyplněn, prvek se ohraničí červeným rámečkem a data nejsou uložena. Toto varování funguje i v případě jsou-li data smazána z prvku a nejsou nahrazena novými. Na konci formuláře jsou dvě tlačítka umožňující cvičení uložit nebo uzamknout.




Obrázek 13: Část webového formuláře zobrazeného studentovi

Výsledný dokument je vytvořen na základě uzamčení cvičení, které může proběhnout automaticky po vypršení platnosti daného cvičení, nebo když jej student uzamkne pomocí tlačítka *Uzamknout cvičení*, které se nachází ve webovém formuláři pro řešení cvičení, kdy se do šablon pro dokument vloží získaná data.

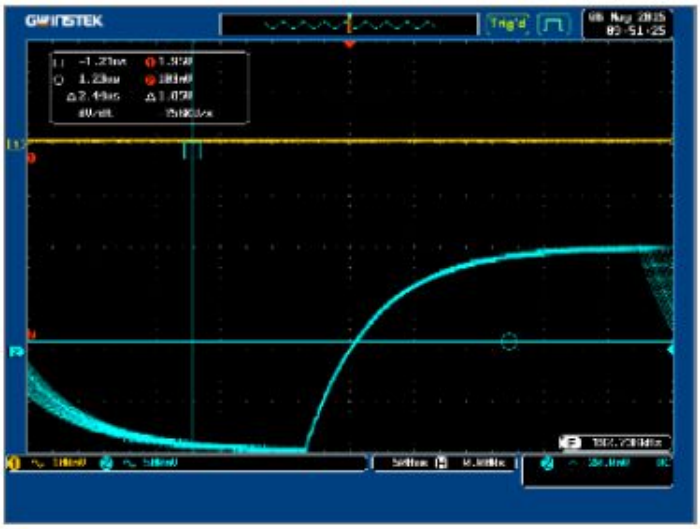
10. Vypočtěte průměrné zpoždění jednoho hradla (jsou zapojeny 4).

$$\frac{Zp_1+Zp_2+Zp_3+Zp_4}{4}=\frac{10+9+11+10}{4}=10$$

Měření na nepřizpůsobeném vedení

- 11. Připojte modul linky A.
- 12. Nastavte pomoci jumperu.
- 13. Odpojte terminační jumpery na modulu MX.
- 14. Připojte osciloskop první kanál na vstup linky a druhý kanál na její konec.

15. Kopie obrazovky (začátek):




Obrázek 14: Část vygenerovaného souboru PDF

# 4 Zhodnocení řešení

Jednotlivé navržené a realizované komponenty byly v průběhu vývoje dostupné na testovacím počítači, který realizoval klíčový server. V průběhu vývoje komponenty procházely testováním, nejdříve byly testovány samostatně především kvůli snadnějšímu nalezení a odladění chyb. Testováno bylo správné zobrazení komponenty vzhledem k nastavujícím parametrům, reprezentace získaných dat, jestli jsou na správném místě nebo ve správném tvaru apod. Nakonec se testoval celkový vzhled komponenty, jestli není výrazně kontrastní vůči ostatním komponentám. Dále byly komponenty testovány ve větším množství v testovacím cvičení, které obsahovalo vždy více zástupců od každé komponenty. Především se testovalo, zda jsou data vracena ke správné komponentě, jako například v případech, kdy by po kliknutí na tlačítko *Nahrát obrázek* u komponenty **TaskOscilloscope** nedošlo k tomu, že se vrácený obrázek zobrazuje jen v jedné a té samé, nebo naopak v jiné komponentě stejného druhu.

U původního návrhu řešení komponenty **TaskCanvas** byly objeveny problémy, které znemožnily tuto variantu využít. Komponenta měla fungovat na principu posouvání obrázků po *html* elementu *canvas* a dále jejich propojování pomocí čar vytvořených pohybem myši od jednoho obrázku k druhému. Ovšem v průběhu testování byl zjištěn závažný problém, který při velkém překrytí obrázků znemožnil pohybovaní danými obrázky. Tedy pokud jeden obrázek zakrýval jiný alespoň z 30 - 40 procent, došlo k zablokování obou obrázků. Protože obrázky byly reprezentovány jako jednotlivé objekty, které se vybíraly kliknutím na příslušný obrázek, mohlo docházet k chybě na straně prohlížeče a jeho implementaci jazyka JS. I přesto byla snaha všemi možnými úpravami kódu, konzultacemi nad kódem nebo vyzkoušením jiných webových prohlížečů tuto chybu odstranit, bohužel bez úspěchu. Proto bylo nutné

komponentu přepracovat do podoby, která je uvedená v kapitole 3.10, kde je komponenta blíže popsána.

Velkou výhodou při vytváření komponent byl samotný systém, který se vyvíjel souběžně s vývojem jednotlivých komponent. Proto bylo velmi jednoduché přidávat funkce do API šablon, a tím nevytvářet duplicitní kód v komponentách. Například funkce *message.error*, která byla přidána do API, slouží k informování vyučujícího při nastavování parametrů dané komponenty v dialogovém oknu o nesprávně zadaném parametru.

Podařilo se přenést většinu programů potřebných k vytvoření protokolu z měření na internet, bohužel složitější schémata logických nebo elektrických obvodů je nutné vytvářet ve specializovaných softwarech, které na tuto činnost jsou zaměřeny. Ovšem výsledné schéma je jednoduché vložit do dokumentu pomocí připravené komponenty **TaskPicture**, která umožňuje nahrát obrázek na server a vložit ho do dokumentu.

V poslední řadě je samotné vytváření cvičení z jednotlivých komponent, které je navrženo pro jednoduché a intuitivní ovládání. Protože samotné cvičení se v systému založí pomocí tlačítka *Přidat cvičení*. Po vyplnění základních údajů cvičení, které reprezentují název cvičení, předmět pod kterým je cvičení uložené, datum od kdy bude cvičení přístupné a doba po kterou bude přístupné, se přejde přímo k vytváření samotného cvičení. To je vytvářeno postupným přidáváním komponent s nastavenými parametry, které reprezentují jednotlivé dílčí úkoly z cvičení. Výsledné cvičení je sestaveno z vybraných komponent.

# Závěr

Cílem bakalářské práce bylo vytvořit a otestovat komponenty pro webový systém, které vzniknou na základě nastudování a analyzování poskytnutých materiálů jako jsou zadání a šablony z předmětů sloužící k výuce zaměřené na elektronická měření nebo metodiku návrhu.

Na začátku práce bylo nutné definovat, co vše se skrývá pod problémem souvisejícím s tvorbou komponent pro webový systém. Stanovit si teoretické znalosti ohledně webových serverů, architekturou klient server, komunikací klienta a serveru pomocí protokolu HTTP a jazyku HTML sloužícímu k předávání informací ze serveru. Dále bylo důležité definovat obecné vlastnosti technického protokolu z laboratorního měření nebo cvičení.

Hlavním cílem této práce bylo vytvořit jednotlivé komponenty, které v rámci webového prohlížeče umožnily nahradit výhody ručně nebo pomocí počítačových programů vytvářených protokolů z laboratorních cvičení či měření. Komponenty jsou navrženy za pomoci jazyka PHP, JS a systému pro tvorbu šablon Smarty. Tyto jazyky byly stanovené počáteční podmínkou, která nám na základě jazyků, ve kterých je vytvořený Inteligentní měřicí systém, stanovuje použitelné programovací jazyky. Za pomoci těchto jazyků a analýzy poskytnutých protokolů z laboratorního měření byly navrženy a poté realizovány jednotlivé komponenty, které reprezentují dílčí úkoly prováděné během tvorby protokolu z měřené. Navržených jedenáct komponent umožňuje vytvářet tabulky s grafy, pravdivostní tabulky, vkládání obrázků, vytváření schématu logického nebo elektrického zapojení, psaní vzorců a výsledných funkcí, získávání snapshotu obrazovky osciloskopu nebo hodnoty z měřicího přístroje, výběr jednoho obrázku z několika možných variant, vkládání *html* prvků pro zhodnocení výsledku nebo zápisu krátké odpovědi. Komponenty

byly úspěšně vytvořeny, otestovány a umožnily přesun vytváření dokumentu na webový systém.

Dílčím cílem práce bylo minimalizovat počet programů, které jsou nutné k tvorbě protokolu z laboratorního cvičení. Tento úkol se podařilo dobře vyřešit, protože k realizaci dokumentu je nutné použít jen webový prohlížeč, ve kterém se vyplní webový formulář. Data z formuláře jsou odeslány na server, kde proběhne vytvoření výsledného dokumentu a jeho odeslání vyučujícímu a studentovi. Jediným externím programem tedy je software sloužící k návrhu schémat logického nebo elektrického obvodu, ve kterém by bylo nutné vytvořit složité nebo rozsáhlé schéma zapojení. Proto dojde ke značné úspoře času potřebného k vyhotovení výsledného dokumentu studentem.

Posledním dílčím cílem práce bylo nastudování systému, do kterého komponenty měly být navrženy. Tento systém se vyvíjel souběžně spolu s vývojem komponent, a proto celkové nastudování na začátku práce nebylo možné. Ovšem tento stav měl velký přínos na vývoj komponent, protože bylo možné kontaktovat vývojáře a vložit některé funkce přímo do součásti systému a tak ulehčit realizaci komponent.

Systém a komponenty byly navrženy pro vytváření a realizaci protokolů z měření. Celý systém by mohl dále směřovat k obsažení dalších součástí spojených s výukou, například realizaci, vytváření a vyhodnocování testů. Komponenty se mohou vytvořit tak, aby obsahovaly otázky a odpovědi a následně vyhodnocovaly správnost zvolených odpovědí. Výsledný vygenerovaný dokument by obsahoval veškeré otázky, možnosti, zvolené nebo vyplněné odpovědi studentů a bodové ohodnocení každé otázky. Při malé úpravě systému by bylo možné provádět i celkové klasifikační vyhodnocení testu na základě součtu bodů ze všech jednotlivých otázek.

# Seznam použité literatury

**Brachet, Pascal. 2005.** PhpMathPublisher Documentation. *xm1math.net.* [Online] 2005. [Citace: 05. 12 2014.] http://www.xm1math.net/phpmathpublisher/.

**Brueggeman, Elliott. 2014.** PHPGraphLib Documentation. *ebrueggeman.com.* [Online] 24. 03 2014. [Citace: 05. 01 2015.] http://www.ebrueggeman.com/phpgraphlib/documentation.

**Floroiu, Cezar. 2002.** PHP Templating with Smarty - Zend Developer Zone. *devzone.zend.com.* [Online] devzone.zend.com, 12. 08 2002. [Citace: 21. 04 2015.] http://devzone.zend.com/139/php-templating-with-smarty/.

**Hunt, Lachlan. 2010.** HTML5 Reference. *www.w3c.org.* [Online] W3C, 10. 08 2010. [Citace: 19. 04 2015.] http://dev.w3.org/html5/html-author/.

**Chung, Lawrence. 2000.** Client-Server Architecture. *www.utdallas.edu.* [Online] The University of Texas, Dallas, 23. 03 2000. [Citace: 20. 04 2015.] https://www.utdallas.edu/~chung/SA/2client.pdf.

**Jacobs, Ian. 2004.** URIs, Addressability, and the use of HTTP GET and POST. *http://www.w3.org/.* [Online] W3C, 21. 03 2004. [Citace: 17. 04 2015.] http://www.w3.org/2001/tag/doc/whenToUseGet.html.

**Jacobs, Ian, Le Hors, Arnaud a Raggett, Dave. 1999.** HTML 4.01 Specification. *www.w3c.org.* [Online] W3C, 24. 12 1999. [Citace: 18. 04 2015.] http://www.w3.org/TR/html401/.

**Klein, Lukáš. 2010.** Tvorba protokolu. *fei1.vsb.cz.* [Online] 23. 02 2010. [Citace: 2. 02 2015.] http://fei1.vsb.cz/kat430/data/Tvorba_protokolu.pdf.

**Lampa, Petr. 2002.** Protokol HTTP. *www.fit.vutbr.cz.* [Online] FIT VUTBR, 26. 09 2002. [Citace: 17. 04 2015.] http://www.fit.vutbr.cz/~lampa/WWW/http.html.cs.

**Marshall, James. 2010.** HTTP Made Really Easy. *jmarshall.com.* [Online] 10. 12 2010. [Citace: 17. 04 2015.] http://www.jmarshall.com/easy/http/.

**Raggett, Dave. 1997.** HTML 3.2 Reference Specification. *http://www.w3.org/.* [Online] W3C, 14. 01 1997. [Citace: 18. 04 2015.] http://www.w3.org/TR/REC-html32-19970114.

**Sieber, Lukáš. 2015.** *Inteligentní měřicí pracoviště se systémem dálkové správy a zpracování dat.* Liberec : TU Liberec, 2015. Diplomová práce.

**Sintes, Tony. 2002.** App server, Web server: What's the difference? *javaworld.com.* [Online] The Computer Language Company Inc., 23. 08 2002. [Citace: 20. 04 2015.] http://www.javaworld.com/article/2077354/learn-java/app-server-web-server-what-s-the-difference.html.

**Šidlof, Petr. 2012.** Ukázkový protokol. *http://bacula.nti.tul.cz/.* [Online] 1. 10 2012. [Citace: 01. 03 2015.] http://bacula.nti.tul.cz/~petr.sidlof/vyuka/LA1/ukazkovy-protokol.pdf.

# Příloha na CD-ROM

Elektronická podoba dokumentu ve formátu PDF.

Zdrojové kódy komponent.

Zadání cvičení, poskytnuté šablony a protokoly ze cvičení.

Vytvořené předlohy cvičeni.