Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Arduino: Automatizovaný skleník

Plzeň, 2014 Lukáš Vávra

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 9. května 2014

Lukáš Vávra

Abstract

This work describes the creation of greenhouse automatic control system using programmable module Arduino UNO. The system consists of three parts – control unit, control application and database server. System realization includes the assembly of control unit from the Arduino module, necessary *shields* and sensors and the software development for all three parts. Each part is developed in the programming language that is supported by the platform.

Abstrakt

V této práci je popsáno vytvoření systému pro automatické řízení skleníku využívající programovatelný modul Arduino UNO. Systém se skládá ze tří částí – řídící jednotky, ovládací aplikace a databázového serveru. Realizace systému zahrnuje sestavení řídící jednotky z modulu Arduino, potřebných *shieldů* a senzorů a vytvoření programového vybavení všech tří částí. Každá část je vyvíjena v programovacím jazyce, který je danou platformou podporován.

Obsah

1 Úvod

Cílem této práce je vytvořit finančně dostupný systém pro automatizované řízení pěstebního procesu v malých sklenících, který bude poskytovat možnosti komerčně prodávaných systémů. Problémem komerčních zařízení je poměrně vysoká pořizovací cena, čímž se pro drobné pěstitele stávají nedostupnými. Levná zařízení zase poskytují omezené možnosti ovládání, které mohou být pro náročnější plodiny nedostačující. Oba tyto problémy by mohlo vyřešit použití open-source prostředků a finančně dostupných programovatelných modulů.

V našem případě je pro řídící jednotku použit modul Arduino™ UNO[1]. K Arduinu[2] jsou připojeny rozšiřující moduly pro přímou komunikaci s uživatelem, připojení k síti a práci s microSD kartou, dále pak relé pro ovládání výstupních zařízení a snímače teploty a vlhkosti. Zbylou část systému tvoří především PC aplikace vytvořená v jazyce Java [1]. Tato aplikace slouží k výchozí konfiguraci jednotky a pro vizualizaci zaznamenaných dat. Přenos dat mezi Arduinem a aplikací v PC je realizován prostřednictvím souborů na microSD kartě nebo vzdáleným připojením přes Internet. Karta v řídící jednotce slouží zároveň jako úložiště zaznamenaných dat. V případě vzdáleného připojení do systému vstupuje spojovací uzel (dále jen *server*), kterým je kombinace PHP aplikace a MySQL databáze [2]. Server uchovává konfiguraci ovládání skleníku a zaznamenaná data v databázi, přičemž k nim uživateli i Arduinu umožňuje přístup prostřednictvím HTTP protokolu.

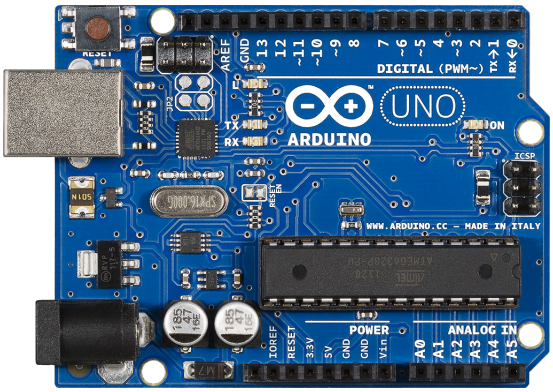
Následující kapitola obsahuje krátké shrnutí informací o Arduinu a popis principů pěstování plodin ve sklenících. Další kapitoly pojednávají o návrhu a realizaci jednotlivých částí systému a formátů dat pro přenos informací mezi těmito částmi. Před závěrečnou kapitolou jsou ještě diskutovány aspekty možného vylepšení systému.

[1] http://arduino.cc/en/Main/arduinoBoardUno
[2] http://cs.wikipedia.org/wiki/Arduino

2 Základní informace

Před vlastním návrhem systému je zapotřebí se seznámit s použitým programovatelným modulem Arduino a zjistit, na základě jakých informací je vhodné skleník ovládat.

2.1 Arduino



Obrázek 2.1: Arduino UNO R3

Arduino UNO je elektronická platforma sloužící k tvorbě prototypů [3, 4]. Z hlavní části je tvořena mikrokontrolérem Atmel ATMega328 [5], což je osmibitový mikroprocesor s vestavěnou 32 kB Flash pamětí (určenou pro uložení programu), 2 kB SRAM operační pamětí a 1 kB pamětí EEPROM. Zbylou část platformy tvoří převodník z USB na RS 232 a napájení. Mikrokontrolér v Arduinu je vybaven bootloaderem, který zabírá 0,5 kB programové paměti v případě modelu UNO. Bootloader slouží k pohodlnému přenosu programu z vývojového prostředí do mikrokontroléru prostřednictvím USB portu. Tím odpadá nutnost mít k dispozici programátor mikrokontrolérů. Arduino je také velmi variabilní nástroj díky své modularitě. K desce lze velmi snadno připojit rozšiřující moduly, tzv. Shieldy, a další samostatné periferie, pomocí kterých získává nesčetné množství využití [6, 7, 8]. Platforma byla původně vyvinuta jako jednoduchý nástroj pro studenty. Její popularita však časem vzrostla a dnes Arduino využívá mnoho jednotlivců i komunit po celém světě. Tomu odpovídá i poměrně velké množství webových stránek a diskuzních fór věnovaných právě Arduinu.

Programování Arduina probíhá v dodávaném prostředí[1], které je vytvořeno v jazyce Java. Samotné programování Arduina probíhá v jazyce C/C++ [9]. Ten je pro tyto potřeby rozšířen o funkce pro práci se vstupy a výstupy, textovými řetězci, sériovým portem, správou přerušení a časem ve formě uplynulých mikrosekund od spuštění mikrokontroléru. Na [10] existuje mnoho dalších volně dostupných knihoven a jiné praktické rady pro všemožné využití Arduina. Podrobný popis vývojového prostředí a procesu kompilace programu včetně jeho nahrání do mikročipu je k dispozici v [11]. Pro přeložení programu do jazyka mikrokontroléru je použit volně dostupný nástroj avr-gcc[2], pro upload zkompilovaného programu slouží avrdude[3].

2.2 Principy pěstování ve skleníku

Pěstování ve sklenících je stále velmi oblíbené. Důvodů je mnoho, počínaje finanční stránkou, kdy ceny potravin rostou neúměrně k mzdám, a konče nedůvěrou k prodávaným produktům. Skleníky proto bývají nedílnou součástí každé venkovské i příměstské zahrady.

Provoz skleníku však přináší svá specifika. V první řadě je potřeba plodiny zavlažovat, protože uvnitř skleníku nemohou získávat vláhu z deště. Zavlažování by mělo být účelné a úsporné, voda by se měla k rostlinám přivádět pouze v potřebném množství a nejlépe přímo ke kořenům. V letních měsících hrozí, že dojde k přehřívání vzduchu uvnitř skleníku. Dalším problémem je vyšší vzdušná vlhkost, která může vést k tvorbě plísní a jiných chorob. Těmto problémům lze předejít zvolením vhodného systému zavlažování a ventilace [12, 13].

Před návrhem systému pro automatické řízení skleníku je třeba si nejdříve uvědomit, jakým způsobem a na základě jakých informací je vhodné pěstování rostlin řídit. Výstupem by mělo být ovládání zavlažování a větrání. Na [14] je možné najít množství článků pojednávajících o podmínkách potřebných pro pěstování různých plodin, přičemž primárními ukazateli jsou teplota a vlhkost. Informace nutné pro řízení lze získat využitím snímačů fyzikálních veličin, ze kterých jsou nejdůležitější snímač teploty a snímač vlhkosti vzduchu. Také je možné využít snímač vlhkosti půdy pro přesnější kontrolu zavlažování.

[1] ke stažení na http://arduino.cc/en/main/software
[2] http://gcc.gnu.org/wiki/avr-gcc
[3] http://savannah.nongnu.org/projects/avrdude

3 Návrh systému

Systém bude tvořen dvěma základními prvky, kterými jsou řídící jednotka skleníku (dále již jen Arduino) a aplikace v PC. Volitelným prvkem bude webový server sloužící jako spojovací uzel pro vzdálenou správu skleníků. Prostřednictvím PC aplikace bude uživatel Arduino konfigurovat. Zároveň bude možné načítat záznamy ze snímačů a zobrazovat je na časové přímce. Přenos dat mezi PC aplikací a Arduinem bude realizován dvěma způsoby:

- soubory na microSD kartě, která bude v řídící jednotce zároveň sloužit jako úložiště dat,
- skrze webový server prostřednictvím internetového připojení.

3.1 Řídící jednotka skleníku

Základním prvkem řídící jednotky bude Arduino UNO, především proto, že se jedná o volně dostupnou open-source platformu určenou pro prototypování a poskytující možnost dalšího upravování a vylepšování všem, kdo o to mají zájem. Systém navrhovaný v této práci je určen pro využití široké veřejnosti, pořizovací náklady by tedy měli být minimální. Arduino je cenově příznivé, cena se pohybuje řádově ve stovkách korun[1]. Součástí Arduina musí být modul umožňující práci s SD kartou. Ta bude sloužit jako úložiště dat a budou se z ní do Arduina nahrávat konfigurační soubory. Dále je požadována přítomnost síťového modulu, který uživateli umožní připojit řídící jednotku k internetu, aby bylo možné ji spravovat vzdáleně. Oba tyto moduly jsou součástí Ethernet Shieldu [15], který na sobě nese jak kontrolér pro práci s Ethernetem W5100[2], tak i slot pro vložení microSD karty (na stránkách věnovaných tomuto shieldu[3] jsou dostupné podrobnější informace). Jednotka by měla poskytovat uživateli informace a možnosti základního ručního ovládání přímo na místě, tedy tam, kde je nainstalována - v blízkosti skleníku. K tomu dobře poslouží LCD Keypad Shield[4], který se skládá z LCD displeje a tlačítek.

[1] V době psaní práce je pořizovací cena modelu Arduino UNO R3 s mikroprocesorem Atmel ATMega 328 cca 600 Kč.

[2] http://www.wiznet.co.kr/

[3] http://arduino.cc/en/Main/ArduinoEthernetShield

[4] http://www.dfrobot.com/wiki/index.php?title=Arduino_LCD_KeyPad_Shield_(SKU:_DFR0009)

Displej obsahuje 2×16 znaků, každý znak je sedm zobrazovacích bodů vysoký a pět bodů široký. Tlačítek je celkem šest. Čtyři jsou popsána jako směrová (Left, Right, Up, Down), jedno jako tlačítko výběru volby (Select) a jedno pro znovuspuštění systému (Reset). Kromě tlačítka reset jsou všechna ostatní tlačítka připojena přes odporovou kaskádu k jednomu analogovému vstupu Arduina (A0).

K tomu, aby jednotka mohla získávat data z okolí, je nutné k ní připojit zvolené snímače. V první řadě se nabízí snímač DHT11[5], který v sobě kombinuje snímač okolní teploty a snímač vlhkosti vzduchu. Také přijde vhod snímač půdní vlhkosti, který tvoří půdní sonda (tištěný spoj ve tvaru písmene U), a pomocný obvod pro měření průtoku proudu mezi oběma póly půdní sondy. Pro ovládání výstupních zařízení, sloužících k zavlažování a ventilaci, bude k Arduinu připojen modul obsahující dvě relé. Jejich spínací kontakty jsou dimenzovány na střídavé napětí 250V a proud 10A. Což by mělo být dostačující pro většinu elektromechanických zařízení používaných ve sklenících. Arduino UNO je vybaveno čtrnácti digitálními vstupy/výstupy[6] a šesti analogovými vstupy. Analogové vstupy jsou využity tři:

- **pin A0** pro tlačítka,
- **pin A1** pro snímač teploty a vlhkosti vzduchu DHT11,
- **pin A2** pro snímač půdní vlhkosti.

Z digitálních pinů jich je využito celkem třináct.

- **piny D1 a D2** slouží pro ovládání relé na výstupu,
- **piny D3, D5, D6, D7, D8 a D9** používá LCD display,
- **piny D4, D10, D11, D12 a D13** jsou použity pro Ethernet Shield, kde:
  - **piny D11 - D13** jsou společné pro Ethernet kontrolér a slot microSD karty, neboť komunikují po sběrnici SPI,
  - **pin D10** slouží pro řízení síťového modulu
  - **pin D4** slouží pro řízení slotu microSD karty.

[5] http://www.micro4you.com/files/sensor/DHT11.pdf
[6] Záleží na tom, jaký mód se pro jednotlivé piny nastaví.

Volný zůstal pouze digitální pin D0 a analogové vstupy A3 - A6. Jejich možná využití budou popsána v kapitole *Diskuze a budoucí práce*.

3.2 Uživatelské rozhraní v PC

Aby bylo řízení skleníku automatické, musí systém pracovat bez přítomnosti obsluhy na základě předem nastavené konfigurace. Tu bude uživatel vytvářet či upravovat v aplikaci ve svém PC. Aplikace prostřednictvím grafického uživatelského rozhraní poskytne možnosti konfigurace zavlažování a ventilace ve zvolených časech a po zvolenou dobu, případně podle informací ze senzorů. Aplikace by také měla podporovat většinu dostupných operačních systémů. Nelze totiž očekávat, že všichni uživatelé budou využívat stejný operační systém ve svém PC.

Programovacím jazykem pro tvorbu aplikace byla zvolena Java [1], která splňuje obě kritéria. Java je multiplatformní jazyk, tj. aplikaci vytvořenou v Javě lze bez opětovné kompilace spustit na jakékoliv platformě, pro kterou existuje JVM[7]. Program je překládán do tzv. „bajtkódu“, ve kterém je šířen a následně interpretován virtuálním strojem v okamžiku spuštění aplikace. S využitím knihovny Swing [16] nabízí širokou škálu nástrojů pro tvorbu GUI, které bude umožňovat přehlednou konfiguraci.

Aplikace musí poskytovat tyto základní funkce:

- Založení nového skleníku - vytvoření konfiguračních souborů na SD kartě.
- Editaci konfigurace Arduina.
- Prohlížení záznamů (čas, teplota, vlhkost atd.) na časové přímce a uložení aktuálního grafu ve formátu PNG na disk v PC.

Je nutné dodat, že poslední dva body musí být možno provádět jak lokálně (úpravou konfigurace na microSD kartě), tak i vzdáleně (přes server). Pro vzdálenou správu budou navíc na serveru a v PC aplikaci k dispozici funkce pro přihlášení, odhlášení a registraci nového uživatele. Uživatel bude mít možnost přidat, odebrat a prohlížet seznam připojených skleníků.

[7] Java Virtual Machine - Virtuální stroj Javy

3.3 Vzdálená správa přes server

V případě, že se uživatel rozhodne využívat vzdálenou správu, tedy konfiguraci Arduina a prohlížení záznamů prostřednictvím Internetu, je nutné zajistit komunikaci mezi PC aplikací a Arduinem. Pro přenos dat na microSD kartě jsou použity soubory obsahující textové řetězce. V rámci kompatibility proto bude vhodné zvolit takový přenosový protokol, který využívá stejný formát dat, v jakém jsou uloženy informace v souborech. Dalším kritériem pro volbu je pohodlné použití. Není žádoucí, aby se uživatel musel zabývat potížemi spojenými s připojením, jako je nastavení firewallu, proxy serveru atp.

Jednoduché a zároveň efektivní řešení poskytuje HTTP protokol [17]. HTTP protokol je jedním z nejvíce používaných protokolů na světě díky jeho využití v Internetu. Právě proto je v drtivé většině případů povolen přenos dat přes transportní protokol TCP [18] na portu 80, který je protokolem HTTP standardně využíván. HTTP je navíc primárně určen pro práci s ASCII, tedy s textovou podobou dat, což je pro naše účely vyhovující. Jedinou větší nevýhodou je absence zabezpečení a šifrování. Obecně se pro zabezpečené spojení používá protokol TLS [19] a spojení je pak označováno jako HTTPS. V této práci však HTTPS použít nelze, neboť implementace TLS, či staršího SSL, by byla v případě Arduina příliš komplexní. Společně se softwarem pro řízení by pak pravděpodobně přerostla paměťová omezení. Zde bude jako bezpečnostní prvek použita identifikace uživatele a řídící jednotky unikátním klíčem, který bude generovat aplikace na serveru. Přenos dat bude probíhat nezabezpečeně.

Pro tvorbu webové aplikace, která zprostředkuje přenos dat mezi uživatelem a Arduinem, bude použit skriptovací jazyk PHP. Skripty zpracovávají HTTP požadavky na straně serveru a vracejí odpovědi ve formě webové stránky. Pro obě základní části systému (PC aplikace a Arduino) budou vytvořeny separátní skripty, sloužící jako jejich síťová rozhraní na úrovni aplikační vrstvy modelu ISO/OSI [18]. Mezi těmito rozhraními bude probíhat komunikace prostřednictvím společného úložiště, kterým bude běžně dostupná databáze MySQL. V databázi budou data uchována i pro případ znovupoužití. PHP skripty rozloží příchozí data na jednotlivé složky (čas, typ, hodnoty), které budou uloženy odděleně, což umožní jejich snadnější zpracování. Tato kombinace je z části ovlivněna i finanční stránkou, neboť většina volně dostupných webových hostingů poskytuje právě tyto prostředky.

4 Program pro Arduino

Vytvořit jako první program pro Arduino je logická volba, neboť jde o část systému, která se potýká s největšími omezeními, a to především paměťovými. Právě proto je potřeba navrhnout formáty a způsob přenosu dat tak, aby byly pro Arduino paměťově nenáročné. Jak se ukázalo, 32 kB programové paměti není mnoho. V programu, vytvořeném pro řídící jednotku (`greenhouse_main`), jsou kvůli omezené velikosti paměti zavedena některá úsporná opatření spojená s redukcí velikosti zkompilovaného programu. I tak zabírá hlavní program téměř celou programovou paměť (aktuálně 30 182 B).

4.1 Paměťová omezení

V případě použití standardních knihoven dodávaných k vývojovému prostředí pro Arduino by velikost programu převýšila velikost paměti. Z tohoto důvodu muselo dojít k několika úpravám. Například vypuštění objektu *String*[1], který slouží pro práci s textovými řetězci. Namísto něj jsou implementovány vlastní funkce pro práci s textem, které jsou pro systém nezbytně zapotřebí. Jde o zápis dvou a tříznakového čísla s doplňováním nul, nalezení čísla v proudu dat[2], načtení řádku z proudu dat, či nalezení řídícího znaku v odpovědi serveru. Funkce pracují pouze s ukazateli a poli znaků [20] a jsou rozděleny podle oblastí použití na souborové, síťové a obecné. Operace s ukazateli a poli jsou záležitost operační paměti, takže nezvětšují velikost programu. V tabulce 4.1 je vidět, že použití objektu *String* přidá k velikosti programu 1508 B oproti použití pole znaků a ukazatele na něj, při kterém se velikost programu nezmění.

Společně s vynecháním objektu *String* přichází ještě jedno úsporné opatření. Tím je zápis všech textů použitých v Arduinu do paměti EEPROM. Pro práci s touto pamětí je k dispozici knihovna[3], která umožňuje zápis a čtení po jednotlivých bytech paměti. A proto, že jeden znak má velikost 1 B, je možné do EEPROM uložit 1024 znaků v případě ATMega328. Vzhledem k tomu, že texty jsou určené převážně pro výstup na LCD displej, který má řádky dlouhé 16 znaků, bude paměť rozdělena na 64 bloků po šestnácti znacích. Pro zápis

[1] http://arduino.cc/en/Reference/StringObject
[2] http://arduino.cc/en/Reference/Stream
[3] http://arduino.cc/en/Reference/EEPROM

`void setup() {}` `void loop() {}`	444 bytů

`char buffer[10];`
`char *p_buffer;`
`void setup() {}`
`void loop() {}`
	444 bytů

`String buffer[10];`
`void setup() {}`
`void loop() {}`
	1952 bytů

Tabulka 4.1: Velikosti zkompilovaných programů při použití String a char[]

textů do EEPROM je potřeba použít pomocný program `greenhouse_text`, který obsahuje seznam všech textů v Arduinu a zapisuje ho po znacích do paměti. Výrobce udává, že životnost paměti je 100 000 cyklů zápisu nebo mazání. V tomto případě však stačí zapsat textové řetězce do paměti pouze jednou, což znamená, že v nejhorším případě dojde k 1024 cyklům zápisu. Nehrozí tedy poškození paměti EEPROM a lze předpokládat, že data v této paměti zůstanou konzistentní po celou dobu provozu. Hlavní program si pro jejich načítání alokuje místo v operační paměti (dále jen buffer), kam poté kopíruje jednotlivé znaky z EEPROM a vytváří z nich použitelné texty pro další použití. Jelikož je textový řetězec v paměti reprezentován pouze jako pole znaků zakončené znakem s hodnotou nula ('\0'), je tento znak doplňován hlavním programem na konec načteného textu do bufferu. Program tento text neprodleně po načtení použije a následně ho v případě potřeby nahradí jiným. Tím je uspořena nejen programová paměť, ale i paměť operační, protože všechny operace s textem jsou prováděny v jednom vymezeném prostoru paměti.

Další úsporu přináší nahrazení knihovny pro práci s SD kartou[4] (`SD.h`) paměťově úspornější knihovnou Fat16[5] (`Fat16.h`), která také poskytuje potřebné funkce a je mnohem menší. Jak je vidět v tabulce 4.2, rozdíl velikostí zkompilovaného programu, který pouze inicializuje přístup k SD kartě, je 4868 B. Tento rozdíl ještě vzroste při otevření souboru a práci s ním. Hlavní odlišností mezi těmito knihovnami jsou podporované souborové systémy. Jak

[4]http://arduino.cc/en/Reference/SD
[5]https://code.google.com/p/fat16lib/

`#include <SD.h>`
 	
`void setup() {`
`  SD.begin(4);`
`}`
`void loop() {}`
	7206 bytů

`#include <Fat16.h>`
 	
`SdCard card;`
`void setup() {`
`  card.init();`
`}`
`void loop() {}`
	2338 bytů

Tabulka 4.2: Velikosti zkompilovaných programů při použití knihovny SD a Fat16

již název knihovny napovídá, použitá knihovna podporuje pouze SD karty se souborovým systémem FAT16[6], kdežto standardní knihovna podporuje jak FAT16, tak i FAT32. Tento fakt s sebou nese omezení velikosti použité SD karty, protože FAT16 dokáže pracovat maximálně s 2 GB SD kartou. V době psaní práce jsou ještě v prodeji, ale již brzy je na trhu nahradí karty s větší kapacitou. Tento problém bude diskutován v kapitole *Diskuze a budoucí práce*. Knihovna Fat16 navíc umí pracovat pouze v kořenovém adresáři SD karty a soubory musí mít názvy ve formátu 8.3[7], tzn. jméno souboru smí mít délku nejvýše osm znaků a součástí může být tříznaková přípona. To však není překážka, neboť k přenosu dat stačí tři soubory, které mohou být v hlavním adresáři a název souborů bude také odpovídat požadavku.

4.2 Operace se soubory

V Arduinu slouží SD karta jako úložný prostor pro ukládání záznamů a uchovávání konfigurace řízení skleníku. Data jsou rozdělena do tří textových souborů:

[6] http://cs.wikipedia.org/wiki/File_Allocation_Table
[7] http://en.wikipedia.org/wiki/8.3_filename

- SETTINGS.TXT
- CONFIG.TXT
- RECORDS.TXT

První ze zmíněných souborů (SETTINGS.TXT) obsahuje informace, které jsou použity při každém spuštění Arduina. Jde o název jednotky, režim připojení k síti a v případě, že bude použito síťové spojení, MAC adresa, jméno uživatele, komunikační klíč a adresa serveru. Jednotlivé položky jsou uloženy po řádcích. Význam položek je popsán v části *Připojení k síti*. Obsah souboru s nastavením připojení využívajícího DHCP server může vypadat takto:

```
Sklenik 1
1
00-11-22-33-44-55
lukas
E77CFE02F7DDB58A30C80E5460E3177F
217.198.115.56
greenhouse.php5.cz
```

Druhý soubor (CONFIG.TXT) obsahuje data potřebná pro řízení výstupních zařízení. Jsou v něm uloženy časy spuštění jednotlivých výstupů (sepnutí relé), popř. i hodnoty pro automatické ovládání (na základě informací ze senzorů). Navíc obsahuje interval mezi záznamy dat a dobu odstavení automatického řízení v případě použití ručního ovládání. Podrobnější popis je v části *Ovládání výstupních zařízení*. Data v tomto souboru mohou vypadat následovně:

```
V 01 000 002
W 01 030 001
D 1 060
D 2 010
```

Do třetího souboru (RECORDS.TXT) jsou ukládány záznamy ze senzorů a stavů zařízení. Jednotlivé záznamy jsou uloženy po řádcích a jsou to souvislé řetězce s přesně definovanou délkou. Proto bylo možné vynechat mezery mezi jednotlivými složkami záznamu a tím uspořit 9 znaků na každém řádku. Velikost nejdelšího řádku je v případě záznamu dat ze senzorů (`E`) 23 znaků.

Přesný význam těchto dat je popsán v části *Záznamy*. Příklad záznamů z Arduina:

```
E064108201404092003700 0
E064609201404092003700 0
A06472220140409101
A06500320140409100
E065110201404091903700 0
E065610201404091903700 0
```

Po inicializaci knihovny Fat16 je možné otevřít soubory prostřednictvím funkce `open(jméno souboru, způsob otevření)`. Jméno souboru je načítáno z EEPROM do bufferu a způsoby otevření jsou různé kombinace předdefinovaných konstant. Těmi jsou:

O_READ - otevření pro čtení

O_WRITE - otevření pro zápis

O_RDWR - otevření pro čtení a zápis

O_APPEND - přemístí kurzor na konec souboru

O_CREAT - vytvoří soubor, pokud neexistuje

O_TRUNC - smaže obsah souboru

O_EXCL - vrátí chybovou hlášku, pokud soubor existuje

Pro čtení ze souboru slouží funkce `read()`, která vrací načtený znak nebo v případě konce souboru či chyby čtení vrací hodnotu -1. Zápis do souboru je zprostředkován funkcemi `print(data)` a `println(data)`. Tyto funkce umožňují zápis celého řetězce znaků najednou a v případě funkce `println()` i odřádkování.

4.3 Připojení k síti

Pro přístup k síti je Arduino vybaveno Ethernet Shieldem[8], který se k síti připojuje kabelem UTP[9] se standardním konektorem RJ-45. Ethernet Shield podporuje rychlost připojení 10/100 Mbps. Ve vývojovém prostředí Arduina je pro tento Shield připravena knihovna Ethernet[10], která slouží k vytvoření aplikace přistupující k lokální síti nebo k Internetu. Knihovna podporuje až čtyři souběžná připojení (příchozí, odchozí nebo kombinace obojího). K inicializaci knihovny a síťového nastavení slouží funkce `begin()`[11] ve stejnojmenné třídě `Ethernet`, která je součástí knihovny. Tato funkce je přetížená a nabízí pět režimů připojení s různým počtem argumentů:

- `Ethernet.begin(mac);`
- `Ethernet.begin(mac, ip);`
- `Ethernet.begin(mac, ip, dns);`
- `Ethernet.begin(mac, ip, dns, gateway);`
- `Ethernet.begin(mac, ip, dns, gateway, subnet);`

První případ používá jako parametr pouze fyzickou adresu Ethernet Shieldu. K připojení je pak použito nastavení získané z DHCP serveru. Tato možnost se objevila až od verze 1.0 vývojového prostředí Arduina. Jako jediná má také návratovou hodnotu, která vyjadřuje úspěch (`true`) či neúspěch (`false`) při získávání nastavení z DHCP. Další režimy slouží pro nastavení statické IP adresy, v případě využití jmenného serveru adresy DNS, a dále pak adresy brány a masky podsítě. Těmto funkcím odpovídají režimy síťového připojení Arduina, kterých je šest:

0 - Síťové spojení je vypnuto, nejsou zadány žádné další parametry

1 - Je použito spojení využívající DHCP server, musí být zadána MAC

2 - Je použita statická IP adresa (+ MAC adresa)

[8] http://arduino.cc/en/Main/ArduinoEthernetShield
[9] Unshielded Twisted Pair - Nestíněná kroucená dvojlinka
[10] http://arduino.cc/en/Reference/Ethernet
[11] http://arduino.cc/en/Reference/EthernetBegin

3 - Je použit jmenný server DNS (+ IP a MAC adresa)

4 - Je použita síťová brána (+ DNS, IP a MAC adresa)

5 - Je použita maska podsítě (+ adresa brány, DNS, IP a MAC adresa)

Číslo režimu je uloženo v souboru pro konfiguraci sítě (SETTINGS.TXT) a pokud je síť využívána, následují za ním v souboru údaje pro konfiguraci síťového připojení. Těmito údaji jsou jméno uživatele, komunikační klíč pro spojení se serverem, IP adresa serveru a URL[12] serveru.

Pro komunikaci přes síť jsou v knihovně k dispozici třídy `EthernetClient` a `EthernetServer`, které odpovídají modelu *klient/server* v případě socketové komunikace [22]. Třídě `EthernetServer` se v konstruktoru předá jako parametr číslo portu, na kterém potom server čeká na příchozí připojení. Zde popisované Arduino však pracuje pouze jako HTTP klient, tudíž tuto třídu nevyužívá a nebude dále popisována. V našem případě je použita pouze třída `EthernetClient`, která navazuje spojení se serverem a slouží k odesílání a přijímání dat. Spojení se navazuje prostřednictvím funkce `connect(adresa, port)`, která má dva způsoby použití. Adresu lze zadat jako objekt *String* s URL serveru nebo jako pole čtyř bajtů s IP adresou. V našem případě je pro spojení použita adresa IP nahraná ze souboru a číslo portu 80, tedy standardní port pro HTTP protokol. Návratová hodnota této funkce značí, zda bylo spojení navázáno. Pokud dojde k chybě, Arduino přejde do režimu „chyba spojení“. Tento režim je spolu s režimy „spojení vypnuto“ a „připojeno“ zobrazen v menu Arduina na LCD displeji. Uživatel má možnost pokusit se o opětovné navázání spojení stisknutím tlačítka na Arduinu. Pokud bylo spojení navázáno, odešle se kontrolní požadavek na server. Odesílaná data se předávají funkcím `print()` a `println()` jako parametr ve více formátech – textové řetězce jako objekt *String* nebo ukazatel na pole znaků (použité v našem případě) a čísla jako *byte, int, long*. Pro zápis čísel je možné použít ještě druhý parametr, který udává číselnou soustavu, např. `DEC` pro desítkovou, `HEX` pro šestnáctkovou atd. Funkce `println()` navíc za odeslanými daty posílá řídící znak konce řádku. Požadavky jsou odesílány jako hlavičky HTTP metodou GET, které následně zpracovává server. Metoda GET je oproti metodě POST mnohem jednodušší na implementaci, což je v případě Arduina žádoucí. Arduino zasílá tři typy požadavků: kontrolní, potvrzovací a požadavek s odesílanými daty. Tyto požadavky jsou uloženy v paměti EEPROM, a v případě potřeby jsou nahrány do bufferu a následně odeslány.

[12]Uniform Resource Locator - Jednotný lokátor zdrojů

Hlavička požadavku má tento tvar:

```
GET /uconn.php?action=<type>&key=<token>&name=<user> HTTP/1.1
Host: greenhouse.php5.cz
Connection: close
```

V případě kontrolního požadavku je na pozici `<type>` hodnota `check`. Při odesílání potvrzovacího požadavku je na této pozici hodnota `received` a hodnota `send` se vkládá při odesílání dat. V posledním zmíněném požadavku je navíc použita dvojice `da=<data>`, do které se vkládají odesílaná data. Místo značek `<token>` a `<user>` se dosazuje komunikační klíč a jméno uživatele z konfiguračního souboru. Verze požadavku je HTTP/1.1, jde o aktuální a nejpoužívanější verzi protokolu.

Kontrolní požadavek se zasílá cyklicky v předdefinovaném časovém intervalu. Server na tento požadavek vrací odpověď, kterou Arduino přijme a uloží do 16 kB velké mezipaměti kontroléru W5100. K datům v této paměti se přistupuje jako k proudu dat, ze kterého je možné číst po znacích. To znamená, že po přečtení znaku je tento znak z mezipaměti odstraněn a ukazatel se posune na další znak. Ve třídě `EthernetClient` je pro tuto operaci funkce `read()`. Společně s funkcí `available()`, která indikuje, zda jsou v mezipaměti další data, je tato funkce použita pro zpracování odpovědi od serveru. Server ve své odpovědi zasílá řídící značky, tzv. tagy, následované daty pro Arduino. Tyto tagy jsou v podobném formátu jako tagy jazyka HTML. To proto, aby byl provoz na portu 80 co nejpodobnější běžnému provozu, jako v případě procházení webu. V tabulce 4.3 je popis všech tagů a následných hodnot.

V Arduinu je vytvořena funkce `char get_tag(*EthernetClient)`, které se předá ukazatel na instanci klientské třídy. Nalezený tag předává funkce jako návratovou hodnotu. Hledání tagu v proudu probíhá tak, že se čtou znaky, dokud není nalezen znak začátku tagu, tedy `<`, potom funkce vrací znak, který po něm následuje. Ten určuje, o jaký tag se jedná a jakou operaci má s případnými následujícími daty Arduino vykonat. Pro čtení čísel z proudu dat v mezipaměti je implementována funkce `uint16_t http_int (*EthernetClient)`, která vyhledá první výskyt čísla v proudu dat, přečte ho a vrátí jako číslo `uint16_t`[13]. Tato funkce je využita pro příjem autor-

[13] Šestnáctibitové bezznaménkové celé číslo (0 - 65535)

Tag	Hodnota	Popis
<a>	0, 1	Autorizační tag – server ověřuje komunikační klíč a jméno uživatele
<t>	Počet sekund od půlnoci 1.1.1970	Synchronizační tag – server zasílá Arduinu aktuální čas
<s>		Synchronizační tag – server žádá Arduino o zaslání konfigurace
<r>	Nastavení ovládání Arduina	Synchronizační tag – server signalizuje Arduinu přijetí nové konfigurace z PC aplikace a zasílá ji po řádcích Arduinu ve stejném formátu, v jakém je uložena v souboru
<o>		Potvrzovací znak – server informuje Arduino, že požadavek byl zpracován
</>		Ukončovací znak – přenos dat byl dokončen

Tabulka 4.3: Seznam tagů zasílaných serverem do Arduina

izační odpovědi, času a konfigurace ovládání. Přijaté nastavení je okamžitě uloženo do souboru CONFIG.TXT. Po přijetí dat odešle Arduino potvrzení na server, aby aktualizoval synchronizační příznak. Pokud Arduino odesílá konfigurační data, ukládá do bufferu jednotlivé řádky, které pak zasílá jako samostatné datové požadavky.

4.4 Ovládání výstupních zařízení

K výstupním pinům Arduina jsou připojena relé, která uživateli slouží pro spínání vlastních zařízení. Spínání těchto relé je ovládáno na základě konfigurace, kterou si uživatel nastaví prostřednictvím aplikace v PC. Do Arduina je tato konfigurace předána jako soubor textových řetězců CONFIG.TXT nebo

ze sítě. První znak řetězce určuje jeho typ. Konfiguraci větrání značí písmeno `V`, zalévání písmeno `W` a pro nastavení prodlevy záznamu dat a ručního ovládání je vyhrazeno písmeno `D`. Ke spínání ventilace a zálivky dochází buď v předem nastaveném čase, nebo dle vstupních dat ze senzorů. Za písmeny `V` nebo `W` následuje číslo řetězce oddělené mezerou. Pokud je toto číslo trojciferné, znamená to, že je použito ovládání využívající senzory. Za číslem řetězce navazují hodnoty nastavení. Pokud je číslo řetězce jedno- nebo dvojciferné, jde o nastavení času sepnutí. Ten je zadán jako další číslo za číslem řetězce. Toto číslo je od 0 do 144 a reprezentuje čas po desetiminutových krocích. Následující číslo je doba, po kterou bude výstup spuštěn. Ta je dána v minutách v případě zalévání a v pětiminutových krocích v případě ventilace. Tato dělení na kroky jsou použita z důvodu možnosti použít pro udržování hodnot nastavení pole datových typů *byte*[14] místo typu *uint16_t* a tím snížit potřebný prostor v paměti na polovinu. Před použitím se hodnoty pouze vynásobí daným krokem. Pokud se jedná o nastavení automatického ovládání, jsou za číslem řetězce uvedeny maximální a minimální hranice hodnot senzorů. V případě nastavení prodlevy, tedy řetězce začínajícího písmenem `D`, následuje také číslo řetězce. To udává o jakou prodlevu je jedná (0 pro záznam, 1 pro ruční ovládání). Poslední hodnota řetězce je pak hodnota prodlevy.
Jak bylo řečeno, data jsou pro přímé použití uchovávána v paměti. Časová nastavení jsou uložena jako dvojrozměrné pole typu *byte* ve tvaru:

`[čas]` - *krok 10 min,* `[prodleva]` - *krok 1 min / 5 min*

Mezní hranice hodnot pro ovládání podle senzorů jsou uloženy v samostatných proměnných, které jsou také typu *byte*. Konfigurační data jsou v zadaných intervalech porovnávána s aktuálním časem, popř. se vstupními hodnotami senzorů. Před samotným porovnáváním se inicializují dvě proměnné, do kterých je ukládán dočasný stav na základě aktuálně porovnávaných dat. Hodnoty těchto proměnných jsou po skončení celého porovnávacího procesu odeslány na výstupní pin. Porovnávání probíhá podle priority od nejnižší po nejvyšší, aby byl výstup ovlivněn nastavením s nejvyšší prioritou. Automatické ovládání má vyšší prioritu než časové. Vyhodnotí se tedy nejprve časy sepnutí a pokud některý odpovídá aktuálnímu času, je do proměnné zapsána logická 1. Pak se vyhodnotí automatické nastavení a popřípadě se změní stav proměnných. Například pokud je teplota ve skleníku nižší, než jaká je nastavená v konfiguraci, změní se stav v proměnné se stavem ventilace zpět na logickou 0 i pokud čas spuštění odpovídá aktuálnímu času. Jako další příklad

[14] Osmibitové celé kladné číslo (0 - 255)

je situace, kdy není nastaven žádný čas zálivky, ale pouze minimální hranice vlhkosti půdy. Pak se jednoduše spustí zálivka, pokud vlhkost klesne pod nastavenou úroveň. Z toho plyne, že automatické nastavení slouží zároveň jako ochrana před těmito havarijními stavy:

- Sepnutí ventilace v případě nízké teploty
- Sepnutí zálivky v případě již zavlažené půdy
- Nesepnutí ventilace v případě vysoké teploty nebo vlhkosti vzduchu
- Nesepnutí zálivky v případě příliš suché půdy

Pokud by si uživatel chtěl sepnout některý z výstupů přímo ve skleníku, aniž by musel měnit data konfigurace, je mu k dispozici ruční ovládání výstupů pomocí tlačítek v menu Arduina. V momentě, kdy uživatel změní stav ručním ovládáním, vyřadí se na přednastavenou dobu vyhodnocování ovládacích dat. Ručním ovládáním je tedy možné přepínat relé i v případě havarijních stavů.

4.5 Záznamy

Arduino vytváří dva typy záznamů. Prvním typem jsou záznamy hodnot ze senzorů DHT11 a vlhkosti půdy. Tyto záznamy se vytvářejí průběžně v nastaveném intervalu. Do záznamu se zapisuje čas, teplota, vlhkost vzduchu a vlhkost půdy. Druhým typem je sledování změn stavů obou výstupních zařízení. Pokaždé, když dojde ke změně se vytvoří záznam obsahující čas změny, informaci o tom, zda byla změna vyvolána řízením nebo ručním režimem, zařízení které bylo ovlivněno a stav do kterého přešlo. Oba typy záznamů se ukládají do souboru RECORDS.TXT na microSD kartě a pokud je aktivní síťové spojení, odesílají se zároveň na server. Záznamy mají podobu textového řetězce, který začíná znakem typu záznamu. Tím je buď písmeno `E`, tedy záznam vlastností prostředí, nebo písmeno `A`, což je záznam změny stavu výstupního zařízení. Za tímto znakem následuje datum a čas záznamu ve tvaru `hhmmssrrrrmmdd`, např. `06410820140409` vyjadřuje čas 6:41:08 dne 9.4.2014. V případě záznamu ze senzorů jsou dalšími hodnotami teplota, vlhkost vzduchu a vlhkost půdy. Teplota je udána dvojciferně, neboť teplotní rozsah DHT11 je 0-55 °C. Obě vlhkosti jsou trojciferné a jsou vyjádřeny v procentech. Záznam změny výstupu má také tři hodnoty, všechny jsou však jednociferné. První udává, zda byla změna vyvolána v ručním režimu (`0`) nebo na základě řídících dat (`1`). Druhá hodnota definuje ovlivněné zařízení – ventilaci (`0`) nebo zavlažování (`1`). Poslední hodnotou je stav zařízení po změně, tj. vypnuto (`0`) nebo zapnuto (`1`).

5 Ovládací aplikace pro PC

Aplikace umožňuje uživateli přístup ke všem funkcím, které systém poskytuje. Jsou rozlišeny dva základní způsoby přístupu k datům v Arduinu. Lokální přístup k souborům na microSD kartě, kterou uživatel vyjme z Arduina a vloží do čtecího zařízení v PC nebo vzdálený přístup skrze internetové připojení prostřednictvím webového serveru. Oba způsoby poskytují stejné možnosti konfigurace ovládání a prohlížení záznamů. Nastavení připojení Arduina k síti lze pouze lokálním způsobem.

5.1 Lokální přístup

Pokud uživatel zakládá nový skleník nebo pokud nevyužívá vzálenou správu skleníku, přenáší data mezi Arduinem a PC aplikací v souborech na kartě. Java poskytuje pro práci se soubory mnoho tříd ve standardní knihově [23]. Pro manipulaci se soubory a složkami slouží třída `File` z knihovny `java.io`. V našem případě zastupuje kořenový adresář microSD karty i jednotlivé textové soubory na ní. Tato třída nabízí pro výpis obsahu adresáře metodu `list()`, která vrací názvy všech souborů a složek. Tímto způsobem aplikace zjišťuje, které z použitých souborů jsou na kartě k dispozici. Podle toho aktivuje příslušná tlačítka pro příslušné operace. Soubory se síťovým nastavením Arduina a s konfigurací ovládání jsou povinné a jsou k dispozici vždy. Pokud nejsou na kartě, znamená to, že jde o zakládání nového skleníku a je potřeba tyto soubory vytvořit. K tomu slouží metoda `createNewFile()`, která vytvoří soubor se jménem zadaným v konstruktoru třídy `File`. Pro vstup a výstup dat používá Java zobecnění v podobě datového proudu. Ten reprezentuje zdroj, popř. cíl dat, zpřístupňuje nebo přijímá libovolné datové bloky a skrývá detaily dění uvnitř datového objektu. Datové proudy se dělí podle směru dat na vstupní a výstupní. V Javě jsou reprezentovány základními třídami `InputStream` a `OutputStream` v případě práce s bajty nebo třídami `Reader` a `Writer` v případě práce s textem. Vstupní proudy poskytují pro čtení metodu `read()`, pro zápis do výstupního proudu slouží metoda `write()`. Základní třídy se obvykle nepoužívají samostatně, ale v kombinaci s dalšími vrstvami, které data zpřístupňují skrze efektivnější rozhraní. Pro čtení je to třída `BufferedReader`, která poskytuje metodu `readLine()` pro načtení celého řádku, a pro zápis celého řádku nabízí třída `PrintWriter` metodu `println()`. Všechny tyto třídy jsou také součástí knihovny `java.io`.

5.2 Vzdálený přístup

Pro práci se sítí obecně nabízí Java knihovnu `java.net`. Její součástí jsou i třídy pro přístup ke zdrojům na daném URL [24]. K reprezentaci samotného URL slouží stejnojmenná třída `URL`. V dialogu nastavení vzdálené správy zadá uživatel URL webového rozhraní pro aplikaci, která se třídě předá v konstruktoru. Třída `UrlConnection` umožňuje kontrolu nad komunikací *klient/server*. Instance této třídy se získává jako návratová hodnota metody `URL.openConnection()`. Pro komunikaci prostřednictvím HTTP protokolu existuje nadstavbová třída `HttpUrlConnection`, která zjednodušuje práci s HTTP hlavičkami a umožňuje snadné použití metody POST. Tato metoda požadavku je vhodnější pro odesílání dat. Data jsou v binání podobě odesílána bezprostředně za hlavičkou. Není tedy omezeno množství přenesených dat jako u metody GET, která předává data jako součást URL v hlavičce požadavku. Pro nastavení metody požadavku je k dispozici metoda `HttpUrlConnection.setRequestMethod()`, v našem případě je to "POST". Zděděnou metodou `setDoOutput(true)` se nastaví příznak odesílání těla požadavku (data po odeslání hlavičky požadavku) na server. Tato data jsou stejně jako v případě metody GET ve formátu `klíč=hodnota` a spojeny znakem `&`. Metoda `getOutputStream`, zděděná od třídy `URLConnection`, otevírá výstupní datový proud a umožňuje zápis po bajtech. Společně s ní se používá obalovací třída `DataOutputStream`, která převádí odesílané objekty na bajty. Celá konstrukce vypadá takto:

```
HttpURLConnection conn =
(HttpURLConnection) url.openConnection();
OutputStream stream = conn.getOutputStream();
DataOutputStream out = new DataOutputStream(stream);
out.writeBytes(param);
```

Tento kód je implementován ve třídě `HttpConnector`. Ta má pouze dvě metody, `send(String[])` a `String[] receive()`. Třídě se předá URL v konstruktoru a následně jsou volány metody pro odesílání a přijímání dat z HTTP serveru. Každý dotaz na server se chová jako samostatné připojení, které je po přijetí dat od serveru ukončeno. Tomu odpovídá i použití těchto metod. Metodě `send()` se jako argument předá pole *String*, což jsou jednotlivé dvojice `klíč=hodnota`. Ty jsou nejprve spojovacím znakem poskládány do jednoho řetězce (`param`). Následně je navázáno spojení se serverem a data odeslána využitím výše zmíněné konstrukce. Spojení zůstává otevřené do

té doby, než je zavolána metoda `receive()`, která vrací také pole *String*. V tomto případě jsou to však jednotlivé řádky těla odpovědi.

Čtení odpovědi probíhá stejně jako při práci se soubory prostřednictvím proudu dat. Metoda `HttpConnection.getInputStream` vrací vstupní proud pro čtení dat z aktivního spojení. Ke čtení jsou použity dvě třídy typu `Reader`, protože všechna přenášená data jsou v textové podobě. `InputStreamReader` převádí datový proud z binárního na textový. `BufferedReader` z něj pak čte text a vrací ho po řádcích metodou `readLine()`. Tato část kódu vypadá následovně:

```
InputStream stream = conn.getInputStream();
InputStreamReader input = new InputStreamReader(stream);
BufferedReader reader = new BufferedReader(input);
String line = reader.readLine();
```

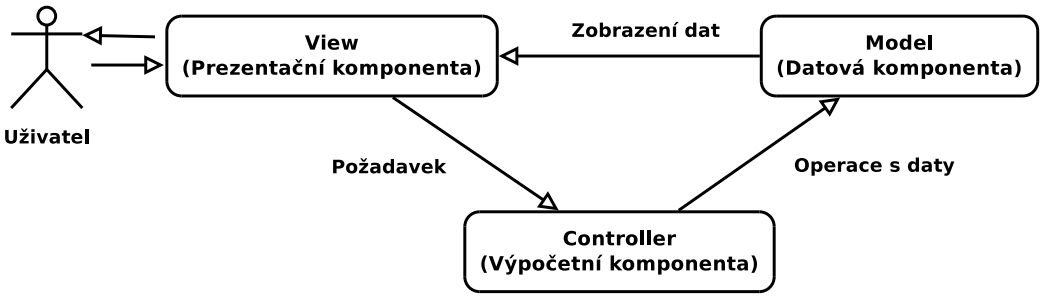
Třída `HttpConnector` je v aplikaci využívána další vlastní třídou `WebDB`, která obsahuje metody pro odesílání a přijímání dat a jednotlivých příkazů. Pro každou interakci se serverem existuje, stejně jako v Arduinu, vlastní příkaz, který je zasílán jako hodnota s klíčem `action`. V tabulce 5.1 je přehled těchto příkazů. Pro identifikaci připojeného uživatele slouží komunikační klíč, tzv. token. Ten je generován serverem při registraci a každém novém přihlášení. Při těchto dvou operacích se serveru zasílá uživatelské jméno (*user*) a heslo (*password*). Server ve své odpovědi zašle token, který je pak používán při všech ostatních operacích. Pokud jsou to operace s daty zvoleného skleníku, musí být v požadavku zasláno ID skleníku v databázi (*unitid*). Toto číslo získá aplikace při výpisu dostupných skleníků (`unitlist`) nebo při přidání nového skleníku (`addunit`). Pokud uživatel odesílá konfigurační data do Arduina, jsou tato data rozdělena na jednotlivé řádky (*data*), které jsou odesílány jako samostatné požadavky. Pro synchronizaci dat v Arduinu a serveru jsou v databázi definované binární proměnné, tzv. příznaky. Ty Arduinu oznamují, že uživatel odeslal novou konfiguraci, kterou má přijmout a uložit na SD kartu (`updatecnf`), nebo aby odeslal používaná data na server (`requestcnf`).

Příkaz	**Odesílaná data**	**Popis**
`login`	user, password	Přihlášení uživatele - server odešle vygenerovaný komunikační token
`register`	user, password	Registrace nového uživatele - server odešle vygenerovaný komunikační token
`check`	token, user	Ověření platnosti komunikačního tokenu
`getinfo`		Získání URL a IP adresy serveru
`addunit`	token, unitname	Přidání nového Arduina do databáze – server vrací ID skleníku
`removeunit`	token, unitid	Odebrání Arduina z databáze
`unitlist`	token	Výpis připojených jednotek – server vrací jména a ID všech skleníků
`requestcnf`	token	Nastavení příznaku zaslání konfiguračních dat všem skleníkům skleníky
`updatecnf`	token, unitid	Nastavení příznaku přijetí konfiguračních dat zvolenému skleníku
`getrec`	token, token	Stažení záznamů zvoleného skleníku z databáze
`getcnf`	token, unitid	Stažení konfiguračních dat zvoleného skleníku z databáze
`sendcnf`	token, unitid, data	Odeslání konfiguračních dat pro zvolený skleník do databáze
`clearrec`	token, unitid	Smazání všech záznamů zvoleného skleníku z databáze
`clearcnf`	token, unitid	Smazání všech konfiguračních dat zvoleného skleníku z databáze

Tabulka 5.1: Seznam příkazů odesílaných z PC aplikace na server

5.3 Nastavení Arduina a prohlížení záznamů

Pro operace s daty ovládání a záznamy a jejich uchování slouží dvě třídy typu *model*, které podle architektury *MVC*[1] [25, s. 235] zastupují výpočetní i datovou komponentu, viz obrázek 5.1. Konfigurační data jsou spravována třídou `ConfigModel`, která poskytuje metody `void parseData(String[])` a `String[] getData()`. Tyto metody slouží pro operace s daty v textové podobě. Pro práci s textovými řetězci slouží obdobně jako v Arduinu třída `String`. Pole těchto řetězců (`String[]`) reprezentuje data po řádcích. Díky tomu není tato třída závislá na zdroji vstupních dat, neboť v obou případech (soubory nebo připojení k síti) je s daty pracováno v tomto formátu. V případě, že jsou konfigurační data spravována prostřednictvím vzdálené správy, udržuje si třída ID skleníku získané od serveru. To je výhodné v případě, že se uživatel rozhodne upravovat konfigurační data pro více skleníků najednou. Toto číslo je odesíláno společně s konfiguračními daty při zápisu dat na server, aby identifikovalo příslušné Arduino. Uživatel zadává konfi-

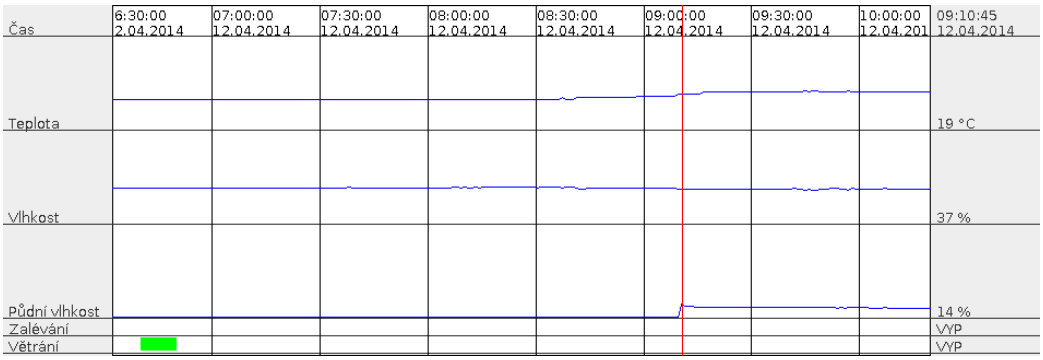


Obrázek 5.1: Diagram architektury *MVC*

gurační data prostřednictvím grafického rozhraní, které je implementováno ve třídě `ConfigFrame`. Ta tvoří prezentační komponentu architektury *MVC*. Okno pro konfiguraci ovládacích dat obsahuje seznamy časů spuštění ovládaných zařízení a textová pole pro zadání mezních hodnot ze senzorů. Pro volbu ovládání podle hodnot ze senzorů jsou k dispozici zaškrtávací pole. Při načtení konfiguračních dat ze zdroje do *modelu* se tato data zobrazí v okně konfigurace. Pokud je uživatel upraví nebo doplní (popř. redukuje), může je uložit zpět do zdroje. Při této akci se data přenesou z prezentační komponenty zpět do *modelu*. Ten data převede do již zmíněného textového formátu a zapíše je do zdroje, ze kterého byla nahrána.

[1] Model-View-Controller – Oddělení datové, prezentační a výpočetní komponenty

Na stejném principu fungují operace se záznamy z Arduina. *Model* pro tyto záznamy tvoří třída `RecordsModel`. Tento *model* je pouze jednosměrný, tzn. data se nemodifikují a nevracejí zpět do zdroje, ale jsou pouze zobrazována. Jediným případem manipulace s daty v příslušném zdroji je jejich smazání. K tomu slouží metoda `void clear()`. Jelikož i v tomto případě může uživatel zobrazovat záznamy z více skleníků, je nutné si v *modelu* uchovávat jejich ID. Třída `RecordsModel` stejně jako `ConfigModel` obsahuje metodu `void parseData(String[])`, která převádí data z textové podoby do objektů `Record`. Tyto objekty zastupují jednotlivé záznamy a umožňují k nim snadný přístup. Data jsou předávána prezentační komponentě `RecordsDiagram`, která je vykresluje v přehledném grafu, viz obr. 5.2. Tato komponenta je umístěna v okně pro zobrazení záznamů `RecordsDialog`. V tomto okně jsou zároveň umístěna tlačítka pro odstranění záznamů a pro uložení aktuálního grafu jako obrázku ve formátu PNG[2]. Při ukládání obrázku do souboru se nejprve vykreslí graf do objektu `BufferedImage`, což je třída reprezentující grafické obrázky z knihovny `java.awt`. Objekt je poté prostřednictvím statické metody `write()` třídy `ImageIO` zapsán jako soubor PNG do zvoleného umístění. Jako název tohoto souboru se implicitně nastaví datum a čas z levého okraje aktuálně zobrazeného grafu.



Obrázek 5.2: Zobrazení záznamů z Arduina v grafu

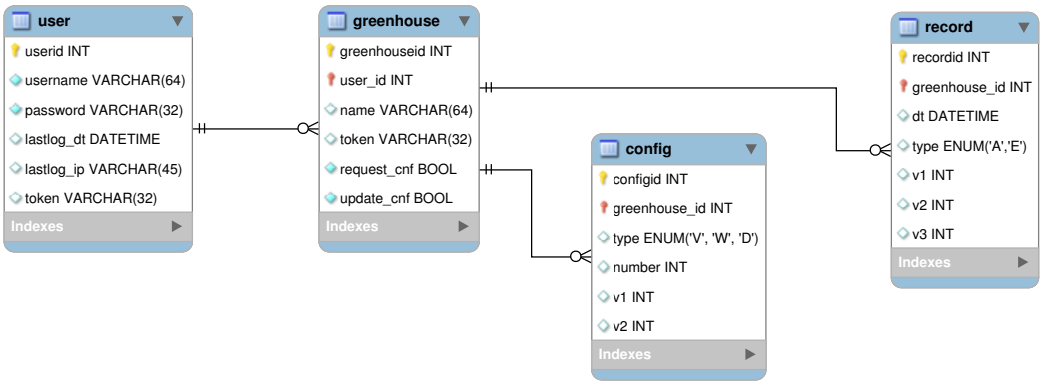
[2]Portable Network Graphics – Přenosná síťová grafika, formát pro bezztrátovou kompresi rastrové grafiky

6 Aplikace pro webový server

Software umístěný na serveru se skládá ze samostatných PHP skriptů. Ty slouží nejen pro komunikaci s Arduinem a aplikací v PC, ale i pro zobrazení základního uživatelského rozhraní v okně prohlížeče. Data jsou na serveru uchovávána v databázovém systému MySQL. K propojení mezi skripty a databází je použita knihovna PDO, která se objevila v PHP od verze 5.1 [26].

6.1 Databázový model

K uchovávání dat je použit relační databázový systém [27], který je na serveru k dispozici. Většina neplacených hostingů poskytuje uživateli přístup k databázi, která je vyhrazena pro jeho webový prostor na serveru. Nejčastěji bývá spravována databázovým strojem MySQL[1], který je vydáván v několika verzích. Na neplacených serverech bývá zpravidla bezplatná verze MySQL, která je šířena pod licencí GPL[2]. Komunikace s relační databází probíhá prostřednictvím dotazovacího jazyka SQL[3], který obsahuje příkazy pro manipulaci s daty, definici dat, řízení přístupových práv a řízení transakcí.



Obrázek 6.1: ER model databáze. Vytvořeno v *MySQL Workbench 6.1*

Na obrázku 6.1 je vidět E-R-A diagram databáze, který byl vytvořen při návrhu modelu. Tabulka informací o uživatelích `user` uchovává jména a hesla

[1] http://www.mysql.com/
[2] https://www.gnu.org/licenses/gpl.html
[3] Structured Query Language - Strukturovaný dotazovací jazyk

uživatelů, data posledních přihlášení a aktuální komunikační klíče pro spojení s PC aplikací. Základní informace o sklenících jednotlivých uživatelů jsou v tabulce `greenhouse`. Ta obsahuje ID majitele, název skleníku, komunikační klíč pro připojení Arduina a příznaky pro synchronizaci konfiguračních dat mezi serverem a Arduinem. Ke každému skleníku přísluší záznamy v tabulkách `config` a `record`. V první tabulce jsou aktuální data konfigurace Arduina. Do druhé tabulky jsou ukládány záznamy z Arduina.

6.2 Knihovna PDO

Pro spojení serverové aplikace s databází je použita knihovna PDO. *Datové objekty PHP*, neboli PDO, je rozšíření PHP5, které vytváří abstraktní rozhraní mezi PHP a mnoha typy SŘBD[4]. PDO bylo navrženo tak, aby programátor nemusel při změně databázového systému upravovat všechny skripty, které k databázi přistupují. Každý databázový systém definuje vlastní API[5] pro funkce, které jsou pro většinu SŘBD společné. Těmito funkcemi jsou např. vytvoření připraveného výrazu (Prepared Statement) nebo zpracování chyb (Error Handling). Fakt, že tato API nejsou sjednocená, přináší v případě změny SŘBD nutnost přepisovat mnoho řádků kódu a vede k novým chybám. Ty je následně potřeba opět odladit, což bývá časově náročné. Absence jednotné knihovny, jako je JDBC[6] v případě Javy, řadila jazyk PHP v žebříčku popularity za významné programovací jazyky. Nyní, když taková knihovna existuje, získává PHP lepší pozici a je oblíbenou platformou pro miliony programátorů [28].

6.3 Rozhraní pro Arduino a PC aplikaci

Server je navržen jako spojovací uzel mezi Arduinem a PC aplikací. Pro komunikaci s Arduinem je na serveru skript `uconn.php`. Ten přijímá od Arduina všechny tři typy požadavků (kontrolní, potvrzovací a odesílání dat) a odpovídá na ně prostřednictvím tagů popsaných v kapitole *Program pro Arduino*. Pokud skript přijme kontrolní požadavek, ověří jméno uživatele a token zaslaný Arduinem. V případě, že autorizace neproběhne úspěšně, odešle

[4]Systém pro řízení báze dat

[5]Application Programing Interface - Rozhraní pro programování aplikací

[6]Java Database Connectivity – Rozhraní pro jednotný přístup k databázím v Javě

se Arduinu negativní odpověď. Jakmile je token v databázi nalezen a uvedené jméno je totožné se zaslaným, je Arduinu odesláno kladné potvrzení autorizace a aktuální čas serveru. Následně skript zkontroluje v databázi příznaky pro synchronizaci. Jestliže jsou na serveru k dispozici konfigurační data pro Arduino, načtou se z databáze, převedou se do textového formátu a zašlou se jako součást odpovědi na kontrolní požadavek. Arduino data přijme, odešle potvrzovací požadavek a skript nastaví v databázi příznak `update_cnf` na logickou 0. Pokud je nastaven příznak `request_cnf`, je Arduinu odeslán příslušný informační tag, viz tabulka 4.3. Arduino po přijetí tohoto tagu začne odesílat konfigurační data. Po přijetí prvního požadavku s těmito daty nastaví skript synchronizační příznak na logickou 0. Požadavek s odesílanými daty může obsahovat také záznamy z Arduina. Skript tento záznam prostřednictvím funkce `explode()` rozloží na jednotlivé položky, které uloží do databáze.

Pro komunikaci s PC aplikací slouží skript `clientconn.php`. Ten zpracovává všechny příkazy uvedené v tabulce 5.1. Po přihlášení uživatele zašle aplikace příkaz `requestcnf` a skript v databázi nastaví příznak `request_cnf` na logickou 1 u všech skleníků, které patří uživateli. To zajistí, že budou v databázi aktuální data ze všech skleníků pro případ, že by je chtěl uživatel upravovat. Při odesílání konfiguračních dat z PC aplikace na server je použit příkaz `sendcnf`. Po odeslání všech dat je příkazem `updatecnf` nastaven příznak `update_cnf` na logickou 1. Tím je dosaženo toho, že Arduino začne přijímat konfigurační data až po jejich úplném odeslání na server. Při každém přihlášení (příkaz `login`) nebo při registraci (příkaz `register`) probíhá generování tokenu pro komunikaci aplikace se serverem. Tento token je pak odeslán aplikaci. Kód pro generování unikátního tokenu vypadá následovně:

```
$token = md5(uniqid(mt_rand()) . $_SERVER['REMOTE_ADDR']);
```

Token je vytvářen funkcí MD5, která vypočítává hash z náhodného čísla a IP adresy uživatele. Stejný kód je použit pro generování tokenu pro Arduino. Pokud uživatel založí nový skleník s připojením k serveru, odešle aplikace příkaz `addunit` a skript přidá do tabulky `greenhouse` novou položku vázanou na přihlášeného uživatele.

7 Diskuze a budoucí práce

Vytvořený systém ovládá dva výstupy (zavlažování, ventilace) a získává data ze tří vstupů (tlačítka, DHT11, vlhkost půdy). Pokud by měl uživatel zájem připojit další výstupní zařízení, např. osvětlení nebo vytápění, bylo by to po hardwarové stránce možné, protože na Arduinu zůstal jeden nezapojený výstupní pin **D0**. Z hlediska softwaru je to také reálné, ale je potřeba doplnit a částečně upravit stávající programy ve dvou částech systému. V případě Arduina by musel být rozšířen algoritmus pro ovládání výstupů a v ovládací aplikaci pro PC přidán další panel v okně konfigurace. Stejný případ nastává, pokud se uživatel rozhodne připojit k Arduinu další senzory, např. pro měření vnější teploty nebo intenzity slunečního svitu. Do volných vstupních pinů **A3 - A6** je možné připojení až čtyř senzorů. V Arduinu by pak musela být rozšířena funkce pro záznam dat a v PC aplikaci rozšířen graf pro zobrazení záznamů. Pokud by uživatel chtěl podle těchto dat ovládat některý z výstupů, muselo by být rozšířeno konfigurování systému. Možnost řídit vybrané výstupní zařízení podle dalšího senzoru znamená přidání polí pro zadání mezních hodnot tohoto senzoru do panelu pro nastavení vybraného výstupního zařízení v okně konfigurace. V programu pro Arduino by to obnášelo upravení algoritmu pro porovnávání konfiguračních dat a hodnot získaných ze senzorů.

Při realizaci systému, konkrétně při tvorbě programu pro Arduino, se vyskytl problém s omezenou dostupností microSD karet s velikostí do 2 GB, což je maximální velikost, kterou podporuje souborový systém FAT16. Tyto karty již téměř vymizely z obchodů. Tento problém by způsobil nepoužitelnost vytvořeného systému. Řešením je použít větší SD kartu a vytvořit na ní 2 GB oddíl FAT16, přičemž zbylý prostor zůstane nevyužitý. K vytvoření tohoto oddílu slouží např. program `fdisk`, který je dostupný v GNU/Linux a MS Windows.

8 Závěr

V této práci byli vytvořeny tři aplikace, které dohromady tvoří systém pro automatizované řízení skleníku. První aplikací je software pro Arduino, které společně s *shieldy*, senzory a relé tvoří řídící jednotku umístěnou ve skleníku. Pro realizaci této aplikace bylo nutné jednotku nejdříve sestavit. Cena Arduina včetně ostatních komponent nepřesáhla částku 1500 Kč. Protože jsou to zároveň jediné finanční náklady, lze celý systém považovat za cenově dostupný, neboť komerčně prodávaná zařízení v této cenové relaci slouží pouze k ovládání samotného zavlažování[1]. Software pro Arduino umožňuje spínání ovládaných výstupních zařízení v nastaveném čase po zvolenou dobu nebo při dosažení nastavených mezních hodnot teploty a vlhkosti. Konfigurační data pro ovládání jsou do Arduina přenášena v souborech na SD kartě nebo přes síťové spojení. Arduino zaznamenává hodnoty ze senzorů a stavy ovládaných zařízení v čase. Konfiguraci ovládání provádí uživatel v PC prostřednictvím aplikace vytvořené v Javě. Ta umožňuje zobrazovat zaznamenaná data v grafu, export grafu jako obrázku PNG, nastavení síťového připojení Arduina a vzdálenou správu Arduina. Webový server, sloužící jako spojovací uzel, je vytvořen v jazyce PHP a k ukládání dat používá SQL databázi. Na serveru přijímají a poskytují data skripty představující webové rozhraní pro obě zmíněné aplikace. Systém tedy splňuje všechny požadavky, které jsou uvedeny v zadání práce.

[1] http://www.gardena.com/cz/water-management/water-controls/

Literatura

[1] HEROUT, Pavel. *Učebnice jazyka Java*. 5., rozš. vyd. České Budějovice: Kopp, 2010, 386 s. ISBN 978-80-7232-398-2.

[2] GILMORE, Jason W. *Velká kniha PHP a MySQL 5: kompendium znalostí pro začátečníky i profesionály*. Vyd. 1. [i.e. 2. vyd.]. Brno: Zoner Press, 2007, 864 s. ISBN 80-868-1553-6.

[3] BANZI, Massimo. *Getting started with Arduino*. 1st ed. Make: Books / O'Reilly, 2009. ISBN 978-059-6155-513.

[4] ARDUINO SA. *Arduino* [online]. 2005 [cit. 2014-04-24]. Dostupné z: `http://arduino.cc/`

[5] ATMEL CORPORATION. *Atmel 8-bit Microcontroller with 4/8/16/32KBytes In-System Programmable Flash*. San Jose, CA, 2012.

[6] MONK, Simon. *30 Arduino projects for the evil genius*. New York: McGraw-Hill, c2010, 191 s. ISBN 00-717-4133-X.

[7] DI JUSTO, Patrick a GERTZ, Emily. *Atmospheric monitoring with arduino: building simple devices to collect data about the environment*. First edition. Farnham: O'Reilly, c2013, 76 s. ISBN 14-493-3814-3.

[8] MARGOLIS, Michael. *Arduino cookbook*. 2nd ed. Sebastopol, Calif.: O'Reilly, c2012, 699 s. ISBN 14-493-1387-6.

[9] VIRIUS, Miroslav. *Jazyky C a C++: kompletní kapesní průvodce programátora*. 1. vyd. Praha: Grada, 2006, 518 s. ISBN 80-247-1494-9.

[10] ARDUINO SA. *The Arduino Playground* [online]. 2005 [cit. 2014-04-24]. Dostupné z: `http://playground.arduino.cc/`

[11] MCROBERTS, Michael. *Beginning Arduino*. New York: Apress, 2010, 433 s. technology in action. ISBN 978-1-4302-3240-7.

[12] VONDRÁK, Aleš. *Skleníky*. 1. vyd. Brno: ERA, 2007, 116 s. Stavíme. ISBN 978-80-7366-105-2.

[13] SVOJANOVSKÝ, Josef. *Skleníky, pařeniště, fóliovníky*. 1. vyd. Praha: Grada, 1998, 97 s. Česká zahrada. ISBN 80-716-9473-8.

[14] UNIVERSITY OF MASSACHUSETTS AMHERST. *Fact Sheets on Greenhouse Management and Engineering* [online]. 2014 [cit. 2014-04-24]. Dostupné z: `http://extension.umass.edu/floriculture/fact-sheets/greenhouse-management-engineering`

[15] MALÝ, Martin. Arduino: webový server i klient do ruky. In: *Root.cz* [online]. 27.7.2010 [cit. 2014-04-26]. Dostupné z: `http://www.root.cz/clanky/arduino-webovy-server-i-klient-do-ruky`.

[16] LOY, Marc a ECKSTEIN, Robert. *Java Swing*. 2nd ed. Sebastopol, CA: O'Reilly, c2003, 1252 s. ISBN 05-960-0408-7.

[17] KOSEK, Jiří. Protokol HTTP. In: *PHP - tvorba interaktivních internetových aplikací: podrobný průvodce*. Vyd. 1. Praha: Grada, 1999, s. 435-460. Průvodce (Grada). ISBN 80-7169-373-1.

[18] DOSTÁLEK, Libor. *Velký průvodce protokoly TCP/IP a systémem DNS*. 2. aktualiz. vyd. Praha: Computer Press, 2000, 426 s. ISBN 80-722-6323-4.

[19] RESCORLA, E. *SSL and TLS*. Vyd. 1. Boston: Addison-Wesley, 2001, 499 s. ISBN 02-016-1598-3.

[20] KERNIGHAN, Brian a RITCHIE, Dennis. Ukazatele a pole. In: *Programovací jazyk C*. Vyd. 1. Brno: Computer Press, 2006, s. 111-144. ISBN 80-251-0897-x.

[21] SIMON, Mark. *A Crash Course in PDO* [online]. Comparity Training Resources, 2010 [cit. 2014-04-20]. Dostupné z: `http://resources.comparity.net/pdf/pdo.pdf`

[22] SPURNÁ, Ivona. *Počítačové sítě: praktická příručka správce sítě*. Vyd. 1. Kralice na Hané: Computer Media, c2010, 180 s. ISBN 978-807-4020-360.

[23] ECKEL, Bruce. *Myslíme v jazyku Java: knihovna zkušeného programátora*. Praha: Grada, 2000, 472 s. ISBN 80-247-0027-1.

[24] FRIESEN, Jeff. *Beginning Java 7*. New York, NY: Apress, c2011, 898 s. ISBN 978-1-4302-3909-3.

[25] BOLLINGER, Gary a NATARAJAN, Bharathi. *JSP - Java Server Pages: podrobný průvodce začínajícího tvůrce*. 1. vyd. Praha: Grada, 2003, 418 s. Moderní programování. ISBN 80-247-0340-8.

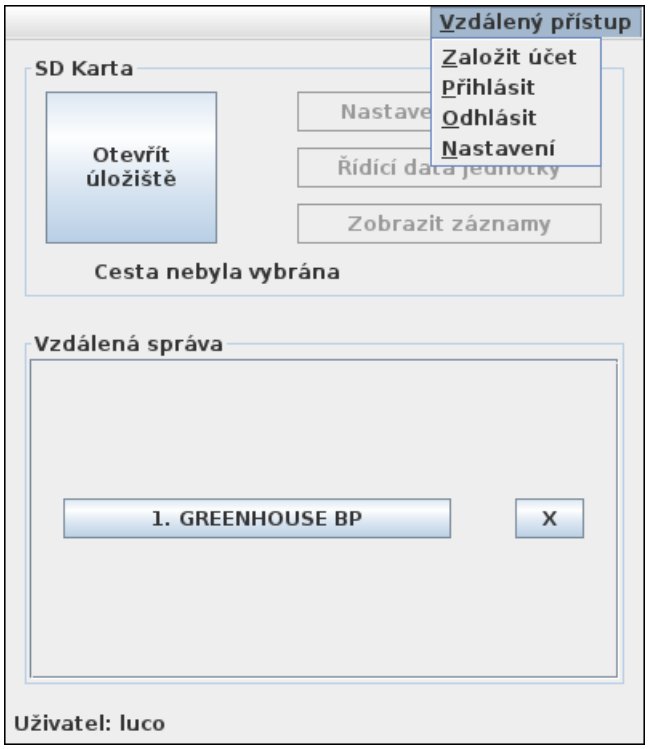
[26] THE PHP GROUP. *PHP Data Objects* [online]. c2014 [cit. 2014-04-25]. Dostupné z: `http://www.php.net/manual/en/book.pdo.php`

[27] LACKO, Ľuboslav. *SQL: kapesní přehled*. Vyd. 1. Brno: CP Books, 2005, 96 s. ISBN 80-251-0788-4.

[28] POPEL, Dennis. *Learning PHP data objects: a beginner's guide to PHP data objects, database connection abstraction library for PHP 5*. 1. publ. Birmingham, UK: Packt Pub. Ltd, 2007. ISBN 978-184-7192-660.

A Uživatelská příručka

Ovládací aplikace v PC slouží uživateli k vytváření a editování ovládacího nastavení Arduina a k prohlížení zaznamenaných informací ze senzorů a stavů výstupních zařízení. Hlavní okno ovládací aplikace je rozděleno na dvě sekce, viz obr. A.1. Jedna slouží pro práci s daty na SD kartě, druhá pro ovládání prostřednictvím vzdálené správy. V sekci SD Karta je tlačítko pro výběr kořenového adresáře SD karty. Po vybrání adresáře se aktivují příslušná tlačítka pro práci s jednotlivými soubory. Ta slouží pro nastavení Arduina, konfiguraci ovládacích dat a prohlížení uložených záznamů. Pro nastavení vzdálené správy je v horní liště menu Vzdálený přístup. Seznam dostupných skleníků je v sekci Vzdálená správa. Zde je možné odebrat Arduino z databáze tlačítkem (X).



Obrázek A.1: Hlavní okno aplikace

A.1 Založení skleníku a načtení dat z SD karty

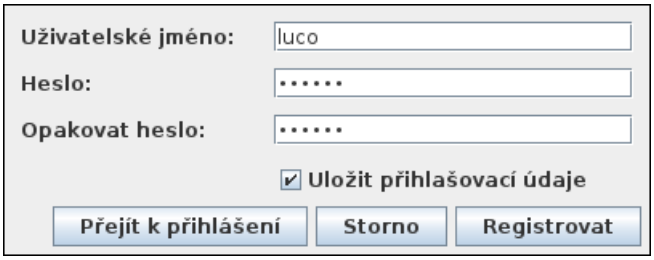
Před samotnou konfigurací je potřeba do čtecího zařízení v PC vložit SD kartu s oddílem FAT16. Stisknutím tlačítka Otevřít úložiště se otevře dialog pro výběr adresáře. V něm je nutné zvolit kořenový adresář SD karty. Následně se aktivují tlačítka Nastavení jednotky a Řídící data jednotky. Pokud je na kartě soubor s uloženými záznamy, aktivuje se zároveň tlačítko Zobrazit záznamy.

A.2 Vzdálená správa

Pokud uživatel využije možnosti ovládání a sledování skleníku přes Internet, musí nejprve v aplikaci nastavit URL webového rozhraní. V menu Vzdálený přístup je položka Nastavení, která otevře dialog pro zadání URL (obrázek A.2). Následně je zapotřebí vytvořit vlastní uživatelský účet. Zvolením Založit

Obrázek A.2: Dialog pro nastavení URL webového rozhraní

účet v menu Vzdálený přístup se zobrazí dialog pro vytvoření nového účtu (obrázek A.3). Pokud má uživatel účet již vytvořen, může se přihlásit volbou Přihlásit (obrázek A.4). Odhlášení uživatele a smazání případných přihlašovacích údajů je prováděno volbou Odhlásit.



Obrázek A.3: Dialog pro vytvoření nového účtu

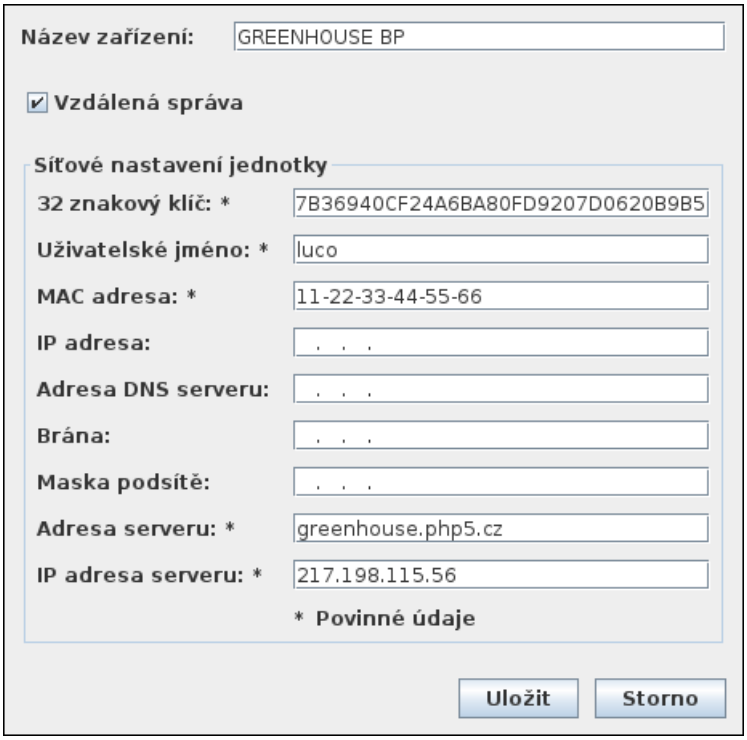
Obrázek A.4: Dialog pro přihlášení uživatele

A.3 Nastavení Arduina

Stisknutím tlačítka Nastavení jednotky se zobrazí dialog pro nastavení názvu skleníku, režimu připojení k síti a případně i všech parametrů síťového připojení, viz obrázek A.5. Zaškrtnutím políčka Vzdálená správa se v případě, že je uživatel přihlášen, aktivuje sekce Síťové nastavení jednotky. Pokud uživatel přihlášen není, zobrazí se nejprve dialog pro přihlášení. Aplikace automaticky předvyplní klíč, jméno uživatele, adresu serveru a IP serveru. Tyto informace získá aplikace od serveru, na kterém proběhne přidání Arduina do databáze. Pokud bude Arduino využívat DHCP server, stačí zadat pouze MAC adresu.

A.4 Nastavení ovládání

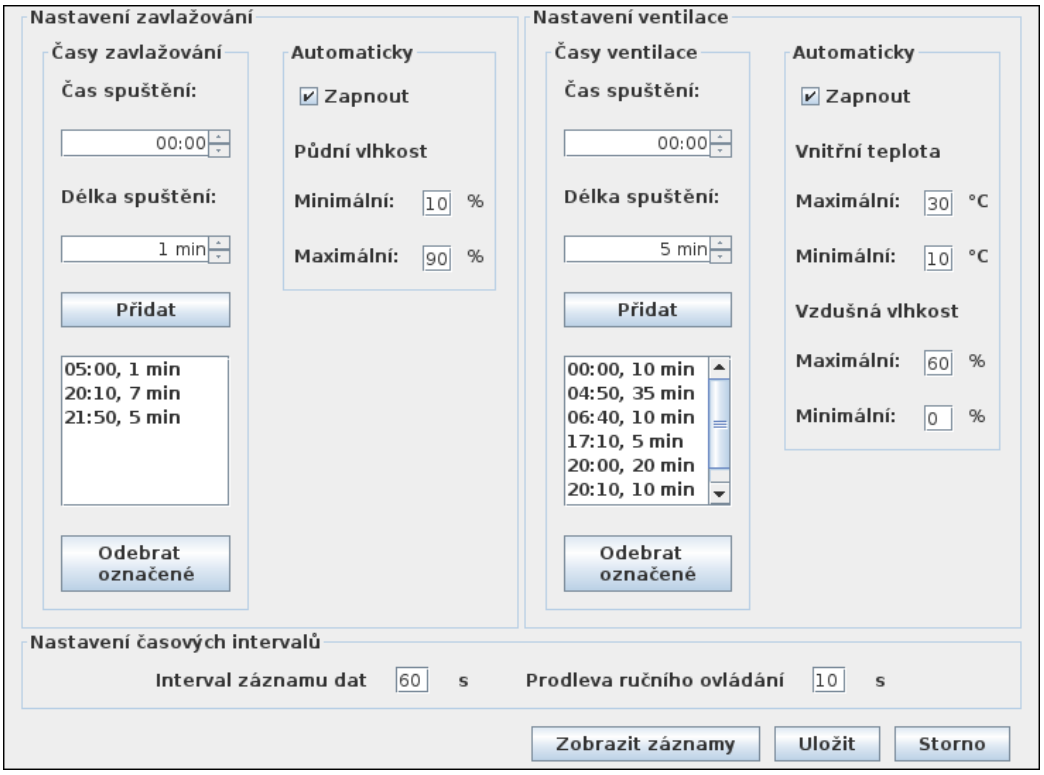
Stisknutím tlačítka Řídící data jednotky nebo tlačítka s názvem skleníku se zobrazí okno pro nastavení řídících dat Arduina (obrázek A.6). Na základě těchto dat Arduino ovládá výstupní zařízení a zapisuje záznamy. Uživatel má možnost nastavit až 32 různých časů spuštění jednotlivých zařízení po dobu, která může být pro každý čas spuštění různá. Zároveň má možnost zapnout ovládání na základě vstupních dat ze senzorů. V tom případě musí nastavit hraniční hodnoty jednotlivých veličin. Také jsou k dispozici pole pro nastavení prodlevy mezi jednotlivými záznamy hodnot ze senzorů a doby, po kterou bude vyřazen algoritmus ovládání podle konfigurace, pokud uživatel použije ruční režim.



Obrázek A.5: Dialog pro nastavení Arduina

A.5 Zobrazení záznamů

Stisknutím tlačítka Zobrazit záznamy v okně pro nastavení ovládání nebo přímo v hlavním okně aplikace se zobrazí okno se záznamy vyobrazenými v grafu (obrázek A.7). Pomocí kurzoru myši je možné prohlížet hodnoty ze senzorů ve zvoleném čase. Tlačítkem Obnovit se opětovně nahrají záznamy a aktualizuje se graf. Po stisknutí tlačítka Uložit jako obrázek se zobrazí dialog pro výběr umístění souboru s obrázkem. Záznamy se z úložiště (souboru nebo databáze) mažou tlačítkem Odstranit záznamy.



Obrázek A.6: Dialog pro konfiguraci ovládání

Obrázek A.7: Dialog pro zobrazení záznamů