Středoškolská technika 2015

Setkání a prezentace prací středoškolských studentů na ČVUT

DOMÁCÍ METEOSTANICE S WEBOVÝM ROZHRANÍM

Jakub Topič

Střední průmyslová škola elektrotechnická

V Úžlabině 320, Praha 10

Obsah

Úvod

Na zahraničním i tuzemském trhu je k nalezení široký výběr domácích meteorologických stanic, které poskytují měření teplot, atmosférického tlaku, relativní vlhkosti vzduchu a často i jednoduchou předpověď počasí na více dní dopředu. Ty levnější z nich jsou pouze budíky rozšířené o několik senzorů. Dražší domácí meteostanice lze připojit k PC za účelem dlouhodobého ukládání, sledování a vyhodnocování naměřených hodnot pomocí dodávaného programu. Tyto stanice často také disponují měřením rychlosti a směru větru nebo srážkových úhrnů.

Výrobci jako časté důvody pro koupi meteostanice udávají fakt, že se nemůžeme zcela spolehnout na předpověď počasí z veřejných médií, jelikož počasí může být v každé oblasti více či méně odlišné.

Po prozkoumání trhu s domácími meteorologickými stanicemi jsem nebyl zcela spokojen s nabídkou a rozhodl jsem se navrhnout, sestavit a naprogramovat vlastní amatérskou meteostanici, která bude měřit venkovní a pokojovou teplotu, atmosférický tlak, relativní vlhkost vzduchu a intenzitu a délku slunečního svitu.

Dalším cílem práce bylo vymyslet vhodné úložiště pro naměřená data. Mezi podmínky, které jsem si hned od začátku kladl, patřil návrh multiplatformní, intuitivní aplikace pro snadnou vizualizaci naměřených dat. Meteostanice by měla být v nejlepším případě bezúdržbová a rozšiřitelná tak, aby bylo možné v budoucnosti přidat nové funkce.

Hlavní motivací k vytvoření tohoto projektu bylo to, že mě lákalo vyzkoušet si propojit mé tři oblíbené počítačové technologie a činnosti do jednoho uceleného projektu. Jedná se o programování mikroprocesorů, práci s operačním systémem GNU/Linux a tvorbu webových aplikací.

Protokol jsem rozdělil na několik částí, ve kterých postupně popíši použité technické vybavení (hardware), použité technologie (software), řešení vlastní problematiky a dále backend a frontend webové aplikace, čili vše, co se týká meteostanice a získávání, ukládání a zpracování dat a popisu mnou vytvořeného webového rozhraní.

1 Technické vybavení

V této části popíšu veškeré technické vybavení a fyzické komponenty, které byly v projektu použity.

1.1 Arduino

Jedním z aspektů projektu bylo propojení senzorů s počítačem a realizace jejich komunikace. Jako rozhraní mezi senzory a počítačem jsem zvolil mikrokontrolérovou desku založenou na projektu Arduino, což je italská open-sourcová platforma využívající mikroprocesory ATmega od firmy Atmel. Nejedná se pouze o hardware (deska s mikroprocesorem), ale i o grafické vývojové prostředí a programovací jazyk Wiring.

Během vývoje meteostanice jsem pro testování kódu a funkcí používal desku Arduino Uno s mikroprocesorem Atmel ATmega328. Kvůli snížení celkových finančních nákladů tohoto projektu jsem pro vlastní provoz nasadil neoficiální klon Arduina se stejným procesorem, sériově vyráběný v Číně.

Použitý mikroprocesor pracuje na frekvenci 16 MHz a jeho vnitřní FLASH paměť pro kód a data programu je 32 kB. Poskytuje tedy dostatečné prostředky pro obsluhu senzorů, nemá ale dostatek paměti pro ukládání naměřených hodnot a nemá ani dostatečný výpočetní výkon pro běh jednoduchého webového serveru a zpracovávání dat. Proto jsem se rozhodl použít malý server s klasickým procesorem architektury AMD64.

Pro připojení senzorů slouží digitální vstupní a výstupní piny a analogové vstupní piny Arduina.

1.2 Použité senzory

1.2.1 Senzor pro měření pokojové teploty

Prvním použitým senzorem je jednoduchý polovodičový teplotní senzor TMP36 od firmy Analog Devices, který jsem použil pro měření pokojové teploty

Vlastnosti senzoru (dle dokumentace):

- Rozsah měření: -40 °C až 125 °C
- Přesnost měření: ±2 °C
- Rozlišení měření: 10 mV/°C
- Velmi nízká spotřeba

1.2.2 Senzor pro měření venkovní teploty a vlhkosti vzduchu

Pro okamžité měření relativní vlhkosti vzduchu jsem použil digitální senzor DHT-11, který umí navíc měřit i teplotu vzduchu. Pro měření venkovní teploty vzduchu se však kvůli svému omezenému rozsahu měření nehodí.

Vlastnosti senzoru (dle dokumentace):

- Rozsah měření vlhkosti: 20 % až 90 %
- Přesnost měření vlhkosti: ±5.0 %
- Rozsah měření teploty: 0 °C až 50 °C
- Přesnost měření teploty: ±2.0 %
- Časová prodleva: <5 s

1.2.3 Senzor pro měření atmosférického tlaku

Pro měření atmosférického tlaku jsem vybral digitální senzor BMP180 s rozhraním I^2C vyráběný firmou Bosch. Tento senzor disponuje i přesným teplotním čidlem.

Vlastnosti senzoru (dle dokumentace):

- Rozsah měření tlaku: 300 hPa až 1100 hPa (odpovídá atmosférickému tlaku v nadmořské výšce 9 000 metrů nad mořem až 500 metrů pod mořem).
- Velmi nízká chybovost: <0.02 hPa (odpovídá výškovému rozdílu 17 cm).
- Rozlišení měření tlaku: 0,01 hPa.
- Rozsah měření teploty: -40 °C až 85 °C.
- Rozlišení měření teploty: 0,1 °C.

Tento senzor měří teplotu a absolutní hodnotu atmosférického tlaku, který vyvolává tíha vzduchového sloupce sahajícího od výškové hladiny, ve které se provádí měření, až k horní hranici atmosféry.

Pozn. Atmosférický tlak se obvykle měří v hektopascalech (hPa). Absolutní hodnota tlaku není tak důležitá, jako změna této hodnoty a rychlost její změny, na čemž můžeme provádět jednoduchou předpověď počasí. Zvýšení tlaku signalizuje obvykle slunečné počasí, pokles většinou znamená oblačno či deštivé počasí. Jelikož je absolutní hodnota tlaku závislá kromě dalších faktorů (jako je teplota vzduchu, obsah vodní páry v atmosféře a zeměpisná šířka) na nadmořské výšce hladiny, ve které je prováděno měření, je pro relevantní využití této hodnoty potřeba znát nadmořskou výšku meteostanice, nebo lze atmosférický tlak přepočítat relativně k nadmořské výšce podle následujícího vzorce. Vzorec obsahuje empiricky stanovené konstanty, p_{abs} je absolutní hodnota atmosférického tlaku naměřená senzorem v hPa, p_{rel} je ekvivalentní hodnota atmosférického tlaku u hladiny moře v hPa a h je nadmořská výška v metrech.

$$p_{rel} = \frac{p_{abs}}{(1 - \frac{h}{44330})^{5,255}}$$

1.2.4 Senzor pro měření intenzity záření Slunce

Jako senzor slunečního svitu jsem použil jednoduchý fotorezistor, který je zapojen jako napěťový dělič, viz schéma zapojení. Elektrický odpor fotorezistoru se snižuje se zvyšující se intenzitou dopadajícího světla. Z výsledků měření, kterého jsem dosáhl, je patrné, že takto zapojený fotorezistor nelze pro měření intenzity záření použít. Jeho elektrický odpor je ovlivněn nejen dopadajícími slunečními paprsky, ale také okolní teplotou. Kromě toho není možné výstupní hodnotu porovnat s žádnou stupnicí nebo převést na běžně používanou jednotku W/m^2. Tento senzor je zde tedy pouze z experimentálních účelů a pro měření délky slunečního svitu, kterou nijak jeho negativní vlastnosti neovlivňují.

V praxi se v meteorologii na profesionální úrovni používá pro měření délky slunečního svitu měřicí přístroj zvaný heliograf, který díky skleněné kouli působící jako čočka (spojka) soustřeďuje sluneční paprsky do jednoho bodu, kde dochází k propálení měrné pásky umístěné pod čočkou. Po následné analýze měrné pásky se určí délka slunečního svitu.

K měření intenzity slunečního záření se v meteorologii používají především měřicí přístroje aktinometry a pyrheliometry. První zmíněný typ pracuje na principu teplem deformovaného bimetalového pásku, z druhé skupiny měřících přístrojů je nejpřesnější vodní pyrheliometr, který pracuje na principu měření teploty vody v tubusu s černým povrchem stěn. V případě mé práce by dle mého odhadu stačil pro měření intenzity slunečního záření prostý

fotovoltaický článek přeměňující elektromagnetickou energii světelných paprsků v energii elektrickou.

1.3 Server

Funkci serveru by v tomto projektu mohl zastávat jakýkoliv dnešní osobní počítač, proto se jeho hardwarem nebudu zabývat. Nejdůležitější byl však výběr platformy, čili operačního systému. Vzhledem k potřebám tohoto projektu a k dalšímu využití serveru pro domácí účely jsem zvolil serverovou linuxovou distribuci Ubuntu Server.

2 Použité technologie

2.1 Použité technologie na straně serveru

2.1.1 Linux, Ubuntu Server

Ubuntu je linuxová distribuce založená na distribuci Debian. Ubuntu je vyvíjeno pro osobní počítače, notebooky a servery (bez grafického prostředí) a jeho název je odvozen z jihoafrického slova „ubuntu“, které se obvykle překládá jako „lidskost“ nebo velmi volně „jsem tím, čím jsem, díky lidem okolo mne“. Ubuntu je vyvíjeno komunitou za podpory společnosti Canonical.

Verze Ubuntu vycházejí v 6 měsíčních cyklech a každé 2 roky vychází tzv. LTS (Long Term Support) verze, tedy verze s prodlouženou podporou, která trvá 5 let. V době nasazení serveru byla poslední LTS verze 12.04, vydaná v dubnu roku 2012. Právě tato verze nyní běží na mém serveru.

2.1.2 BASH

BASH je příkazový interpret v mnou používané distribuci Linuxu, který je založený na unixovém shellu Bourne shell. Jeho název je akronym pro Bourne again shell. BASH plní funkci rozhraní mezi operačním systémem a uživatelem. V interaktivním režimu čeká na zadání příkazu od uživatele. Příkaz může být přímo zabudován v shellu nebo lze skrze něj volat samostatné programy napsané v libovolném programovacím jazyce. BASH také umožňuje nastavení a přizpůsobení pracovního prostředí pomocí systémových proměnných.

Ačkoliv se to nemusí zdát, je to velmi mocný programovací nástroj a lze pomocí něj vyřešit velké množství problémů a uplatní se v situacích, kde bychom zbytečně museli použít nějaký složitější programovací kompilovaný jazyk.

2.1.3 Cron

Cron je démon (čili služba na pozadí), který automatizovaně spouští v operačním systému v určitý čas či v časovém intervalu nějaký příkaz, program, skript atp.

2.1.4 Apache

Apache HTTP Server je open-sourcový webový server vyvíjený v jazyce C a C++ pro množství platforem (mj. GNU/Linux, FreeBSD, Solaris, OS X, Windows). Podle průzkumu serveru Netcraft z června roku 2013 byl Apache nasazen odhadem na 54.2 % serverů, čímž se stává nejpoužívanějším softwarovým webovým serverem na světě.

Apache podporuje rozsáhlé množství funkcí a programovacích skriptovacích jazyků, jako je PHP, Python nebo Perl. Má také podporu protokolů SSL a TLS pro šifrovanou komunikaci. Veškerá přenášená data dokáže komprimovat pomocí algoritmu gzip.

2.1.5 MySQL

MySQL je druhý nejpoužívanější databázový systém na světě. Je vydáván jak pod licencí GNU, tak v proprietární variantě, a podporuje množství platforem (mj. GNU/Linux, FreeBSD, Solaris, OS X, Windows).

MySQL používá relační databázový model a ovládá se pomocí jazyka SQL. SQL je jakožto nástupce jazyka SEQUEL strukturovaný dotazovací jazyk a obsahuje příkazy jak pro manipulaci s daty, tak pro definici dat a řízení přístupových práv.

2.1.6 PHP

PHP (PHP: Hypertext preprocessor) je skriptovací programovací jazyk určený především pro vývoj dynamických webových aplikací a internetových stránek. PHP lze použít i k programování konzolových aplikací, čehož jsem mj. využil i ve své práci. Hlavní využití PHP v mé práci je však k programování webové aplikace, sloužící pro vizualizaci naměřených dat.

Syntaxe jazyka PHP je inspirována jazykem C nebo Java, přesto se ve webových aplikacích používá v nekompilované podobě a webový server (Apache) volá tzv. interpret, který PHP skript zpracuje a odešle klientu pouze výsledek činnosti. Skripty jsou tedy prováděny zcela na straně serveru.

Výhodou PHP je nativní napojení na mnoho databázových systémů (mj. MySQL, Oracle DB, MSSQL, PostgreSQL). Ve svém projektu jsem ovšem zvolil databázový systém MySQL.

2.2 Použité technologie ve webové aplikaci

2.2.1 HTML5

Hypertext Markup Language je standardní značkovací jazyk používaný pro tvorbu webových stránek a webových aplikací, které jsou navzájem provázány hypertextovými odkazy. HTML je vyvíjeno konsorciem W3C, které je zároveň hlavní standardizační organizací pro WWW, a organizací WHATWG.

2.2.2 CSS3

CSS3 je označení pro kaskádové styly, jazyk pro popis stylu zobrazených elementů HTML nebo XHTML dokumentu. Bylo navrženo konsorciem W3C a jeho cílem je oddělení struktury a obsahu dokumentu od jeho stylu, tj. vzhledu. Díky tomuto oddělení se lépe zabraňuje přebytečným redundancím, protože stejné vlastnosti a styly může sdílet jak více různých elementů, tak i celé dokumenty.

2.2.3 JavaScript

JavaScript je dynamický objektově orientovaný programovací jazyk používaný zejména ve webových aplikacích. Jeho implementace je řešena na straně klientu.

2.2.4 WebApp manifest

Pro mobilní webové aplikace je důležité, aby fungovaly tak, jak by uživatel očekával, že bude fungovat nativní aplikace, tedy aplikace vyvinutá přesně pro danou mobilní platformu ve specifickém programovacím jazyce (například Java pro operační systém Google Android nebo objektově orientované C pro Apple iOS). Nativní aplikace se nejčastěji publikují přes repozitář dané platformy.

WebApp manifest přináší vývojáři schopnost určit způsob, jak bude webová aplikace spouštěna. Tento manifest je jednoduchý JSON soubor, který obsahuje název aplikace, cesty k jejím ikonám a způsob spouštění.

2.2.5 Google Charts API

Google Charts API je sada javascriptových knihoven vyvíjená společností Google, která slouží pro dynamickou vizualizaci dat ve webových aplikacích. Kromě rozsáhlé sady předpřipravených typů grafů obsahuje i sadu ovládacích prvků, jimiž lze grafy snadno ovládat a vybírat a třídit jejich data. Grafy jsou renderovány pomocí HTML5/SVG technologie, takže poskytují rozsáhlou podporu webových prohlížečů a lze je používat i na mobilních platformách.

2.2.6 OpenWeatherMap API

Kvůli nepříznivým podmínkám pro nasazení meteostanice, jsem nemohl nainstalovat anemometr, tudíž meteostanice postrádá funkci měření rychlosti a směru proudění větru. Uživateli by se též hodil stav oblačnosti, který nedokážu automatizovaně určit. Tato data získávám z OpenWeatherMap, což je online služba, která poskytuje bezplatné API k meteorologickým datům jako je aktuální počasí, předpověď a historická data. Zdrojem těchto dat jsou oficiální meteorologické stanice, stanice na letištích a meteorologické radary po celém světě.

Data aktuálního počasí z tohoto API jsou poskytována ve formátu JSON nebo XML feedu. Tento kód se zpracuje a následně z těchto dat velmi pohodlně vyberu pouze hodnoty, které potřebuji (tj. rychlost a směr větru a oblačnost). V kódu PHP skriptu jsem musel též ošetřit chybu v případě nedostupnosti serveru, který poskytuje API, jehož ukázku přikládám.

Zdrojový kód 3: Ukázka JSON feedu žádosti aktuálního počasí v Praze

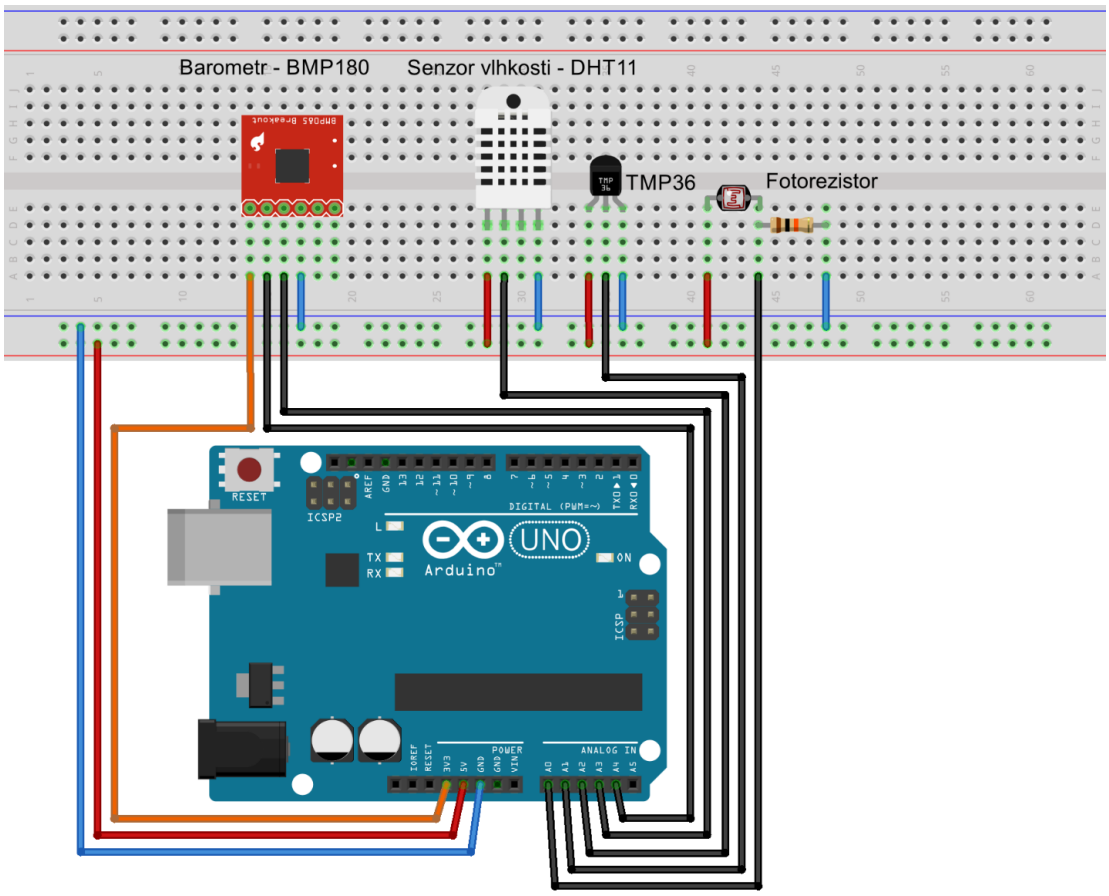
```
{"coord":{"lon":14.42,"lat":50.09},
"sys":{"message":0.0085,"country":"CZ","sunrise":1410763140,"sunset":141080
5620},
"weather":[{"id":804,"main":"Clouds","description":"zataženo","icon":"04d"}
],"base":"cmc stations",
"main":{"temp":6.639,"temp_min":6.639,"temp_max":6.639,"pressure":1009.11,"
sea_level":1041.87,"grnd_level":1009.11,"humidity":92},
"wind":{"speed":6.85,"deg":122.504},
"clouds":{"all":92},
"dt":1410794820,
"id":3067696,
"name":"Prague",
"cod":200}
```


3 Řešení problematiky

V této části práce popisuji postup získávání dat, jejich odeslání na server, následné zpracování těchto dat až po jejich interpretaci.

3.1 Připojení senzorů

Při práci s vybranými senzory jsem získal špatné zkušenosti s jejich nedostatečnou dokumentací od výrobce. Kvůli tomu trvalo naprogramování Arduina déle, než jsem původně očekával. Z důvodu neúplné dokumentace se totiž v některých případech programování bohužel blížilo až téměř k experimentálnímu řešení problému. Nakonec jsem ale vše vyřešil s velmi uspokojivými výsledky.



Obrázek 1 Zjednodušené schéma zapojení na nepájivém poli

3.1.1 Senzor TMP36

Jedná se o jednoduchý polovodičový senzor teploty, který má 3 piny: $+V_s$, V_{out} a GND. Ze schématu zapojení je patrné, že tento senzor pracuje jako dělič napětí. Na výstupním (prostředním) pinu se objevuje napětí 0–5 V, které přivádím na jeden z analogových vstupů Arduina. Metody programovacího jazyku Wiring mi umožňují číst hodnotu jednotlivých vstupních pinů a procesor převádí analogovou hodnotu napětí na digitální (10bitové číslo v intervalu 0–1024). Pomocí následujícího vzorce určím z číselné hodnoty analogového vstupu A1 okamžitou teplotu ve °C.

$$t = (\frac{5 \cdot A_1}{1024} - 0{,}5) \cdot 100$$

3.1.2 Senzor DHT11

Tento senzor pro měření teploty a relativní vlhkosti vzduchu je v komunitě kolem Arduina používán velmi často, proto lze na internetu nalézt velké množství hotových knihoven pro

načítání hodnot z tohoto senzoru. Nic mi nebránilo některou z těchto knihoven použít, ale většina z dostupných knihoven byla napsána tak, aby fungovala univerzálně i s dalšími podobnými senzory, a já jsem potřeboval načítat pouze hodnotu relativní vlhkosti. Z důvodu snížení velikosti programu jsem si napsal vlastní knihovnu.

3.1.3 Senzor BMP180

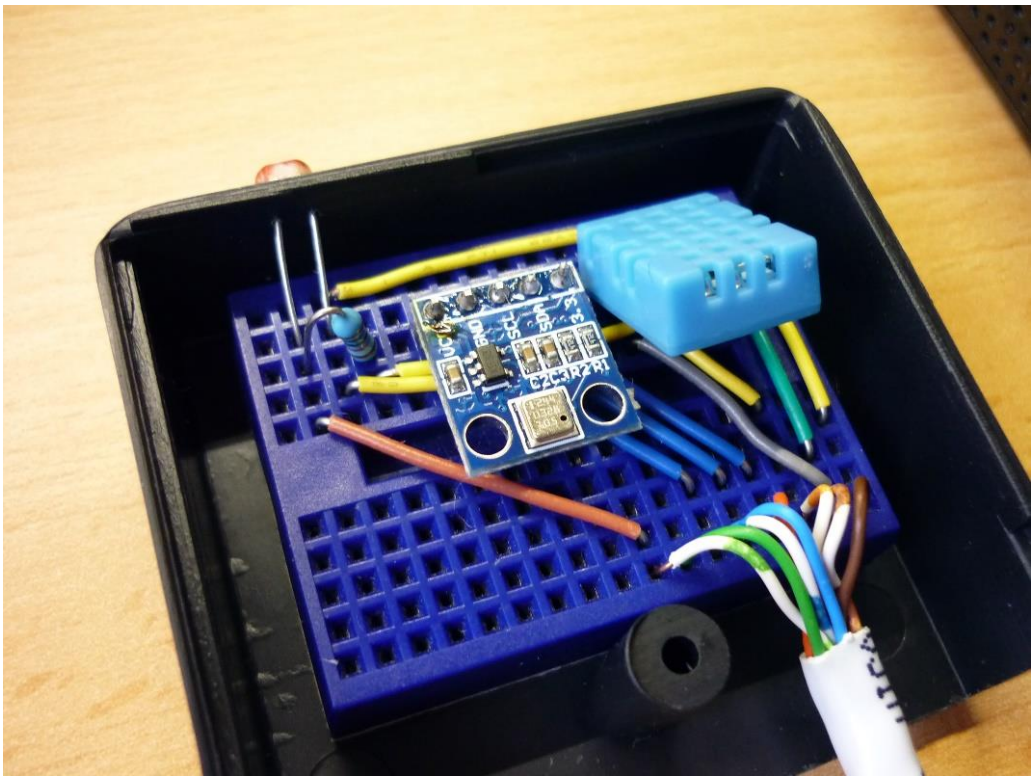
Největší problémy nastaly u implementace tohoto senzoru pro měření teploty a atmosférického tlaku. Dokumentace dodávaná od výrobce byla velmi chabá a jen stěží se dal pochopit princip komunikace se senzorem přes rozhraní I^2C. Svou knihovnu pro Arduino jsem tedy naprogramoval za použití rad z internetového fora Arduina, tudíž velmi experimentálně. V tuto chvíli však již existuje alespoň jedna použitelná knihovna pro Arduino, kterou poskytuje internetový obchod SparkFun electronics. Překvapivě jsem svou knihovnu navrhl správně a naměřená data byla přesná, ve své meteostanici ale aktuálně používám zmíněnou knihovnu, jejíž autor je Limor Fried.

3.1.4 Fotorezistor

Fotorezistor jsem zapojil jako napěťový dělič, aby se choval obdobně jako senzor TMP36. Hodnotu z analogového pinu již však nijak neupravuji.

3.1.5 Umístění senzorů

Veškeré venkovní senzory jsou uloženy v plastovém konstrukčním boxu a k Arduinu, umístěném v serveru, jsou připojeny přes osmižilovou kroucenou dvojlinku. K propojení senzorů používám malé nepájivé pole, aby bylo možné v budoucnu senzory vyměnit nebo přidat nové. Pokud se nepájivé pole neosvědčí a jeho kontakty budou časem například oxidovat, zaměním jej za plošný spoj. Na obrázcích 2 a 3 je vidět konstrukční box v raném stádiu vývoje meteostanice. Nyní je konstrukční box nabarven bílou barvou a je upevněn ve vzdálenosti asi 20 cm od stěny budovy.



Obrázek 2: Otevřený konstrukční box se senzory připojenými přes nepájivé pole

Obrázek 3: Konstrukční box se senzory připevněný k vnější straně lodžie (raná fáze vývoje)

3.2 Komunikace mezi Arduinem a PC

Mikrokontrolérová deska Arduino disponuje pro komunikaci s počítačem synchronním/ /asynchronním sériovým rozhraním USART (piny RX - receive a TX - transmit). Tato sériová linka je připojena přes převodník PL2303HX (dále jen převodník) k USB portu mého serveru. Tento převodník je použit pro emulaci chybějícího sériového portu a také poskytuje stejnosměrné napájení pro Arduino o napětí +5 V. Modulační rychlost, tedy počet přenesených symbolů za sekundu, je 9 600 Bd.

Pro navázání sériové komunikace s Arduinem z pohledu serveru nyní využívám fakt, že v Linuxovém souborovém systému jsou všechna bloková a znaková zařízení zastoupena speciálními soubory umístěnými v adresáři „dev“ v kořenovém adresáři systému. Po správné změně nastavení příkazové řádky a nastavení modulační rychlosti pomocí příkazu *stty* můžeme odesílat Arduinu řetězce textu pomocí klasického přesměrování výstupu příkazu *echo* do zařízení */dev/ttyUSB0*, což je připojený převodník pro sériovou komunikaci. Vše potřebné již zařídí operační systém a následnou přímou komunikaci s Arduinem zajišťuje převodník. Čtení dat, které Arduino odešle, je obdobné jako při výpisu obsahu textového souboru – tedy pomocí příkazu *cat*.

Druhou možností komunikace by bylo například použití *SimpleMessageSystem* v programu *screen*, což by poskytlo interaktivní rozhraní pro komunikaci s Arduinem. V tomto případě jsem se ale spokojil s první variantou, která se mi jevila jako nejelegantnější řešení.

Nyní bych chtěl věnovat krátkou pozornost zjednodušenému bashovému skriptu, který jsem vytvořil pro načtení dat z Arduina. Na začátku skriptu definuji, jaký interpret se má pro zpracování skriptu použít (v tomto případě BASH) a nastavím způsob komunikace s převodníkem. Následně si připravím aktuální počet minut v hodině a odešlu do Arduina textový řetězec. V ukázce níže se jedná o řetězec *start*.

V praxi však odesílám každou minutu jiná, proměnná data včetně času. Pokud je právě 30. nebo 0. minuta v hodině, načtu příkazem *cat* hodnoty přijaté z Arduina a výstup přesměruji do pomocného souboru *buffer*. Skript se spouští pomocí CRON úlohy v intervalu jedné minuty a veškerý jeho případný výstup je přesměrován do souboru určeného pro log a tak lze jakékoliv chyby snadno identifikovat.

Zdrojový kód 1: Ukázka skriptu pro načtení hodnot z Arduina a uložení do pomocného souboru

```
#!/bin/bash
stty -F /dev/ttyUSB0 9600 cs8 -cstopb
sleep 0.1

time=$(date +%H:%M)
cut=$(echo "$time" | cut -c4-5)

echo "start" > /dev/ttyUSB0

sleep 0.5

if [ "30" = "$cut" ] || [ "00" = "$cut" ]
then
   timeout 50.3s  cat /dev/ttyUSB0 >
/mnt/store2/www/jakubtopic.cz/sub/meteo/buffer
   cd /mnt/store2/www/jakubtopic.cz/sub/meteo/; php -q procedure.php
fi
```

3.3 Zápis hodnot do databáze

V této fázi zmíněný bashový skript spustí PHP skript *procedure.php*, který se stará o zápis hodnot do databáze. Poté se bashový skript ukončí.

V první části PHP skriptu se načte a zpracuje soubor *buffer*, ve kterém jsou z předchozího skriptu uložené aktuální hodnoty ze senzorů meteostanice. Ty jsou již částečně zpracované z Arduina, takže jediné, co je třeba udělat, je vypočítat rosný bod.

Při návrhu databáze jsem se musel rozhodnout, zda vypočítat rosný bod předem a jeho hodnotu uložit, což představuje nadbytečná redundantní data, nebo počítat rosný bod pokaždé, kdy jej bude potřeba, což ušetří místo v databázi, ale na druhou stranu prodlužuje vykonávání skriptů a tudíž je potřeba větší výpočetní výkon.

Po zvážení obou možností jsem zvolil variantu s výpočtem předem. Databáze meteostanice bude postupem času zabírat kapacitu pouze v řádech megabytů, tudíž se nemusím o její velikost příliš starat. Pokud bych ale při zpracování řádově tisíců záznamů musel u každého počítat rosný bod, mohlo by se to výrazně projevit na rychlosti zpracování skriptu.

Rosný bod (t_{dp}) lze vypočítat podle následujícího vzorce, kde t je venkovní teplota vzduchu ve °C a V je relativní vlhkost vzduchu v %. Ve vzorci se vyskytují empiricky stanovené konstanty.

$$t_{dp} = \frac{243{,}5(\frac{V}{100} \cdot e^{\frac{17{,}67 \cdot t}{243{,}5+t}})}{17{,}67 - ln(\frac{V}{100} \cdot e^{\frac{17{,}67 \cdot t}{243{,}5+t}})}$$

Vzorec pro výpočet rosného bodu

Následně se všechny hodnoty uloží do určené tabulky v databázi. Struktura tabulky je velmi jednoduchá – obsahuje pouze časovou značku (timestamp) a dále jednotlivá pole pro meteorologické prvky včetně rosného bodu. Na konci dne provádí tento PHP skript ještě jednu proceduru a to je zápis do tabulky *klima*. Vypočítá tedy maximum a minimum venkovní teploty za uplynulý den a dle následující tabulky určí, zda se jedná o tropický, letní, arktický, ledový nebo mrazový den. Hodnoty je třeba porovnávat ve správném pořadí, aby se například den neurčil jako letní, přestože více odpovídá tropickému. V případě, že jedna z těchto možností vyhovuje, uloží se do tabulky *klima* datum a identifikátor typu dne.

ID	**Typ dne**	**Podmínka**
1	Tropický den	$t_{max} > 30$ °C
2	Letní den	$t_{max} > 25$ °C
3	Mrazový den	$t_{min} < 0$ °C
4	Ledový den	$t_{max} < 0$ °C
5	Den s $t_{min} < -10$ °C	$t_{min} < -10$ °C
6	Arktický den	$t_{max} > -10$ °C

Tabulka 1: Tabulka pro určení dnů podle teploty vzduchu

4 Návrh webového rozhraní

4.1 Backend

4.1.1 Struktura webové aplikace

Hlavní částí mé webové aplikace je soubor *index.php*, který obsahuje většinu PHP a HTML kódu. V tomto souboru je tedy uložen veškerý obslužný kód, který generuje výstup ve formě HTML v závislosti na požadavku uživatele (tvaru webové adresy).

Často používané funkce, jako je například připojení k databázovému systému, jsem vyčlenil do souboru *function.php*. Další soubor je *style.css*, který obsahuje kaskádové styly, čili kód určující vzhled celé webové aplikace. V adresáři *fonts* lze potom nalézt open-sourcový font *Roboto*, který ve své aplikaci používám, a v adresáři *images* veškeré obrázky.

4.1.2 Vykreslování grafů

Hlavní účel webové aplikace je vizualizovat naměřená data, proto bylo nutné navrhnout co nejlepší způsob této vizualizace. Ze začátku vývoje webového rozhraní meteostanice jsem grafy vykresloval a generoval na straně serveru jako PNG obrázky (bitmapu) pomocí vestavěné grafické knihovny *php_gd2* v PHP. Jelikož tyto obrázky představovaly znatelný datový tok a jejich generování mírně prodlužovalo zpracování skriptu, zvolil jsem nakonec pro vizualizaci dat Google Charts API. Výhodou tohoto řešení je, že se data přenáší pouze v podobě textu a vykreslování probíhá na straně klientu pomocí javascriptu a SVG. Na rozdíl od statického obrázku je tato vizualizace dynamická.

4.1.3 Využití technologie WebApp manifestu

WebApp manifest je jednoduchý JSON soubor metadat, která popisují mobilní webovou aplikaci. Při „instalaci“ webové aplikace na mobilní zařízení, tedy při jejím přidání na plochu (launcher) přes možnost ve webovém prohlížeči, vybere systém z manifestem poskytnutých ikon takovou, která se nejlépe hodí pro dané zařízení z hlediska jejích rozměrů a hustoty pixelů displeje. Tato ikona se použije pro zástupce aplikace na ploše.

Dále máme v tomto manifestu možnost určit, která webová stránka se načte jako první při spuštění aplikace pomocí vlastnosti *start_url*. Vlastnost *display* nabývá 3 hodnot *browser, standalone* nebo *fullscreen*, přičemž určuje, zda se má webová aplikace otevřít klasicky v prohlížeči jako další karta, nebo se spustí jako nová aktivita webového prohlížeče bez jakýchkoliv jeho ovládacích prvků a chová se potom obdobně jako nativní aplikace. V přepínači spuštěných aplikací ji vidíme jako samostatnou položku a na první pohled nemá s webovým prohlížečem nic společného.

Hodnota *fullscreen* navíc skryje stavovou a navigační lištu, takže aplikace zaujímá skutečně celou plochu displeje zařízení. Já jsem ve své aplikaci zvolil hodnotu *standalone*. Poslední vlastností je *orientation*, která určuje orientaci aplikace – na výšku, na šířku nebo automaticky.

WebApp manifest v tuto chvíli podporuje pouze webový prohlížeč Google Chrome od verze 39 běžící v operačním systému Android od verze 4.0 a používá jej i mobilní operační systém Firefox OS. Lze očekávat, že se tento koncept stane brzy standardem W3C a bude následně implementován i v dalších webových prohlížečích a na dalších platformách, viz http://www.w3.org/TR/appmanifest/.

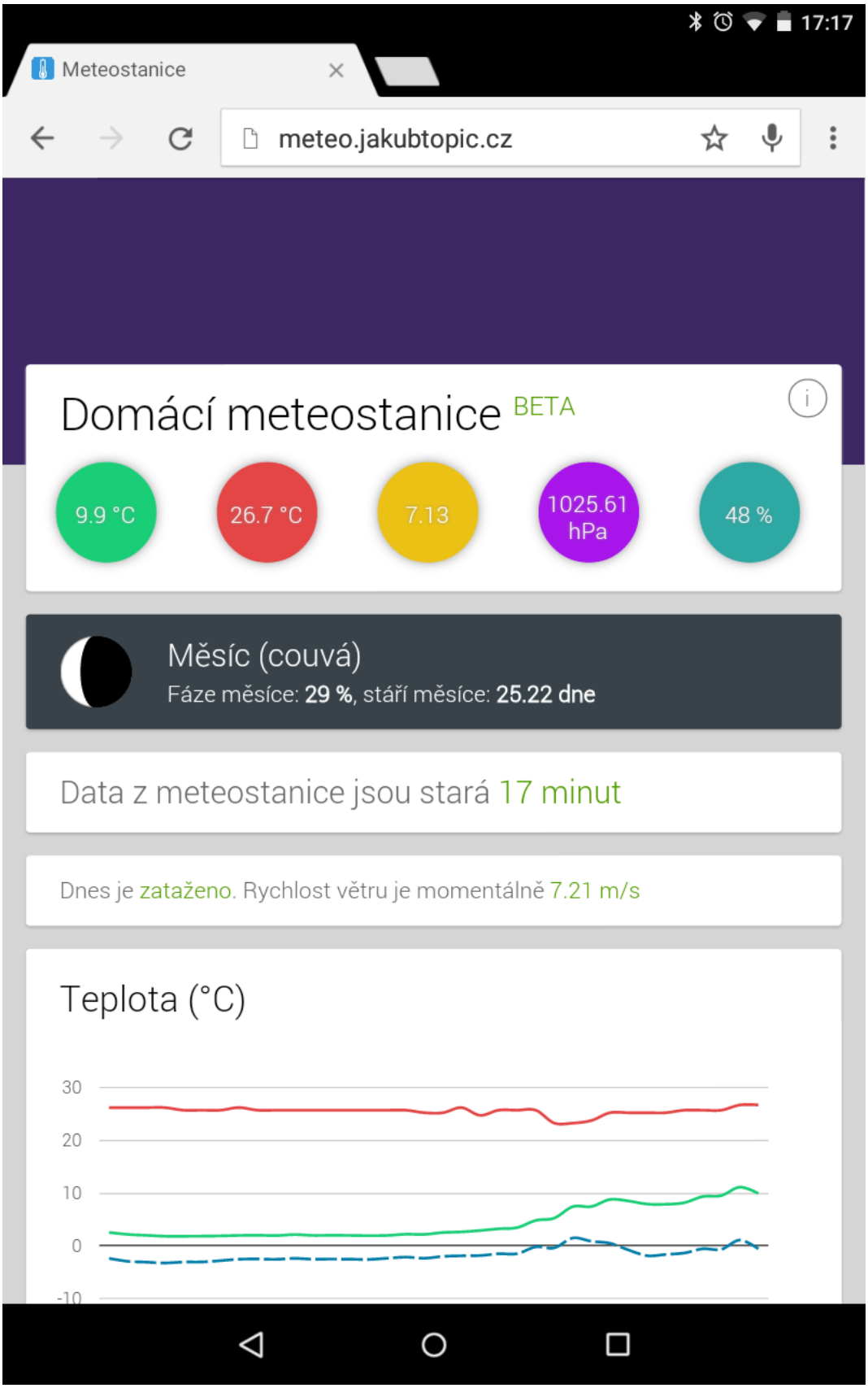
Ukázka zdrojového kódu 2: Provázání HTML dokumentu s manifestem

```
<link rel="manifest" href="/manifest.json">
```

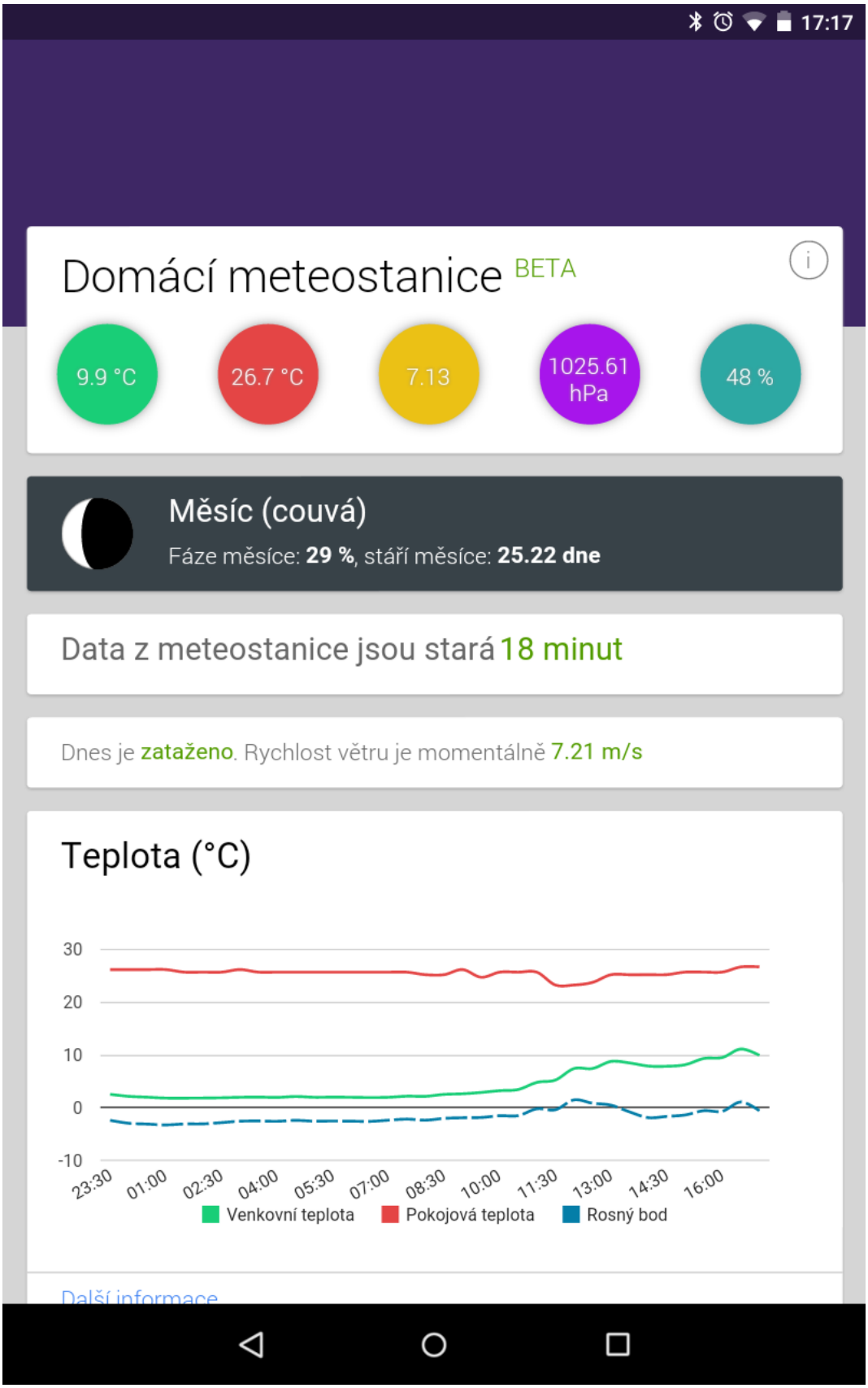
Ukázka zdrojového kódu 3: Příklad použití WebApp manifestu

```
{
  "short_name": "Meteostanice",
  "name": "Domácí meteostanice",
  "icons": [
    {
      "src": "images/launcher-icon-3x.png",
      "sizes": "144x144",
      "type": "image/png"
    },
    {
      "src": "images/icon-highres.png",
      "sizes": "192x192",
      "type": "image/png"
    }
  ],
  "start_url": "index.php",
  "display": "standalone",
  "orientation": "landscape"
}
```

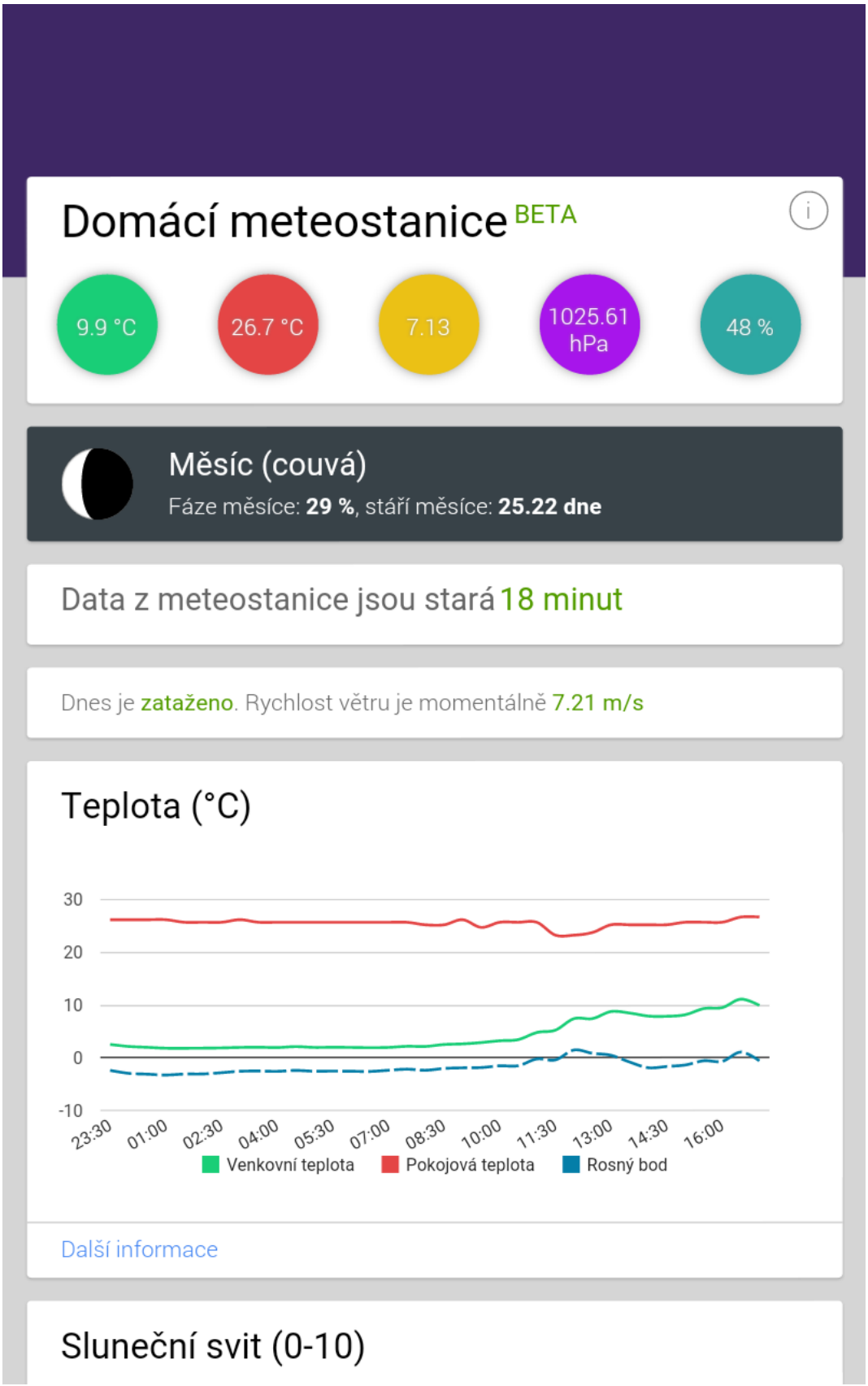
Následující screenshoty ukazují mnou vytvořenou webovou aplikaci meteostanice spuštěnou na tabletu s operačním systémem Google Android 5.1 Lollipop ve třech možných módech (*browser*, *standalone* a *fullscreen*). Na 4. screenshotu lze vidět, že se po nastavení vlastnosti *display* ve WebApp manifestu na *standalone* nebo *fullscreen* tváří webová aplikace jako nativní a má též svou vlastní kartu v přepínači spuštěných aplikací.



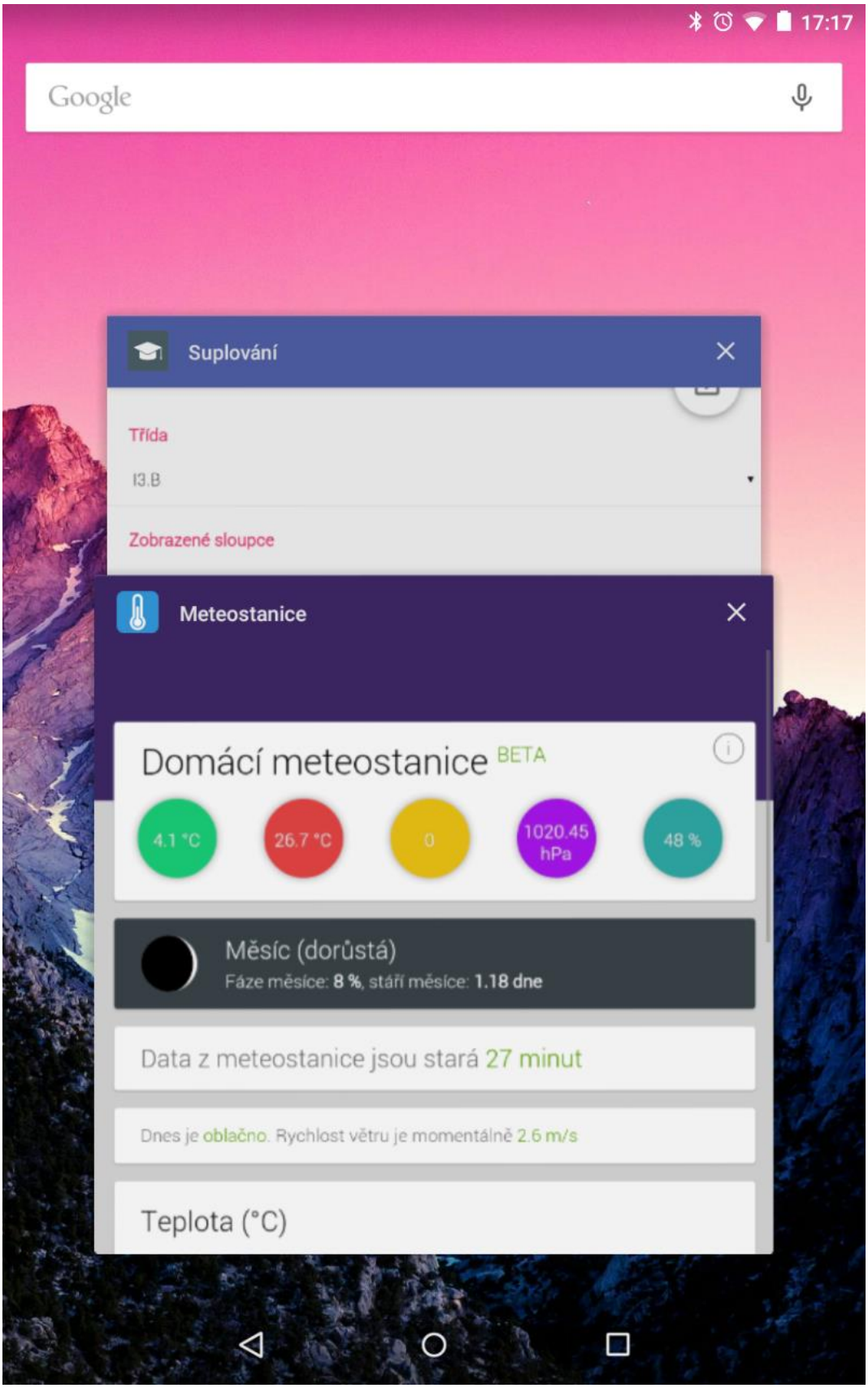
Obrázek 4: Spuštěná aplikace při použití vlastnosti *display*: *browser*



Obrázek 5: Spuštěná aplikace při použití vlastnosti *display*: *standalone*



Obrázek 6: Spuštěná aplikace při použití vlastnosti *display*: *fullscreen*



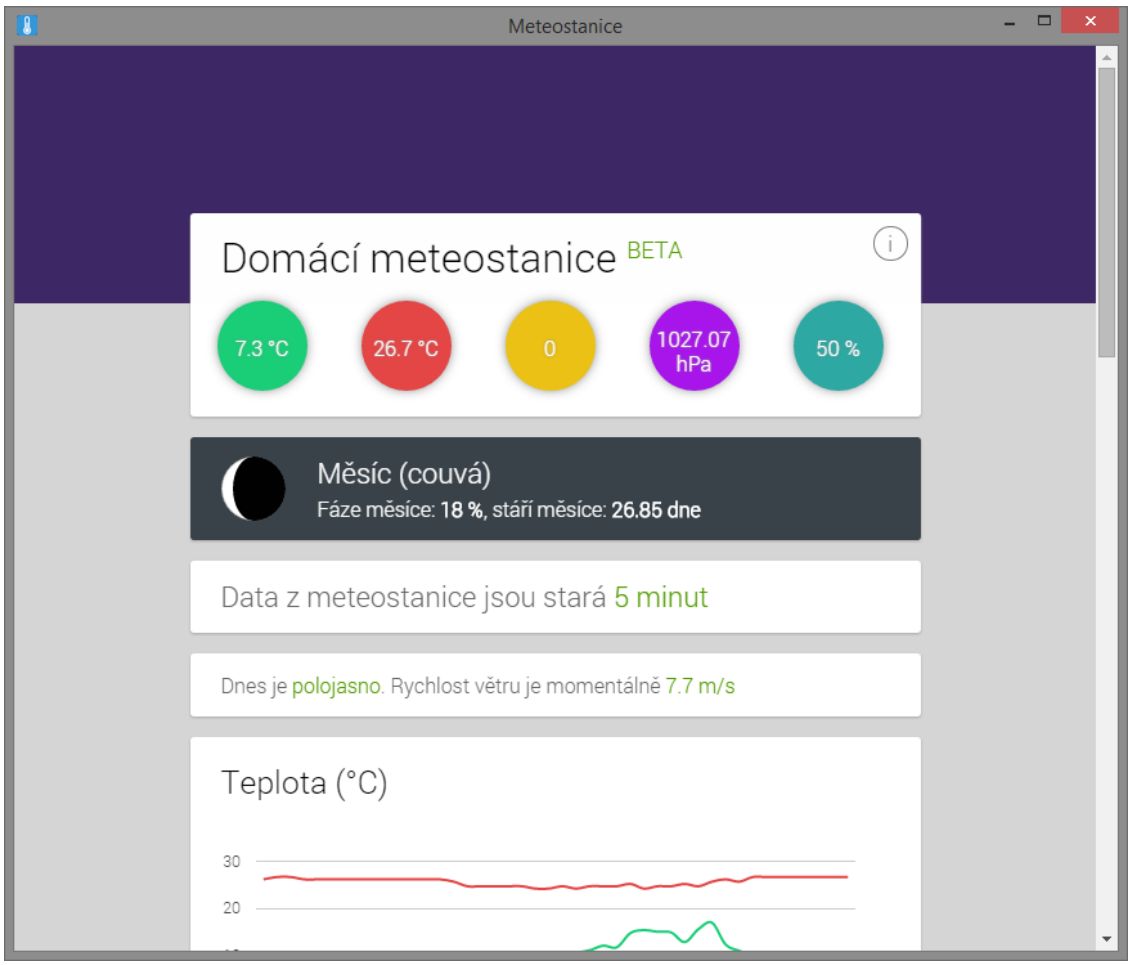
Obrázek 7: Zobrazení aplikace v přepínači spuštěných aplikací

4.2 Frontend - funkčnost webového rozhraní

Jelikož bylo cílem vytvořit aplikaci nejen pro různé platformy, ale i různé *form factors* (druhy zařízení jako jsou stolní počítače, notebooky, tablety, telefony apod.), bylo zapotřebí vymyslet jednotný návrh, který se přizpůsobí danému zařízení k co nejpohodlnějšímu ovládání. Takový návrh se označuje **responzivní design**. Základem je flexibilní struktura, které se dosahuje použitím procentuálních šířek místo pevně daných velikostí v pixelech. V kaskádových stylech CSS3 se také často používají tzv. *Media Queries*, která umožňují definovat různé styly v závislosti na šířce obrazovky a typu zařízení. Pomocí *Media Queries* lze nastylovat třeba i rozložení při tisku.

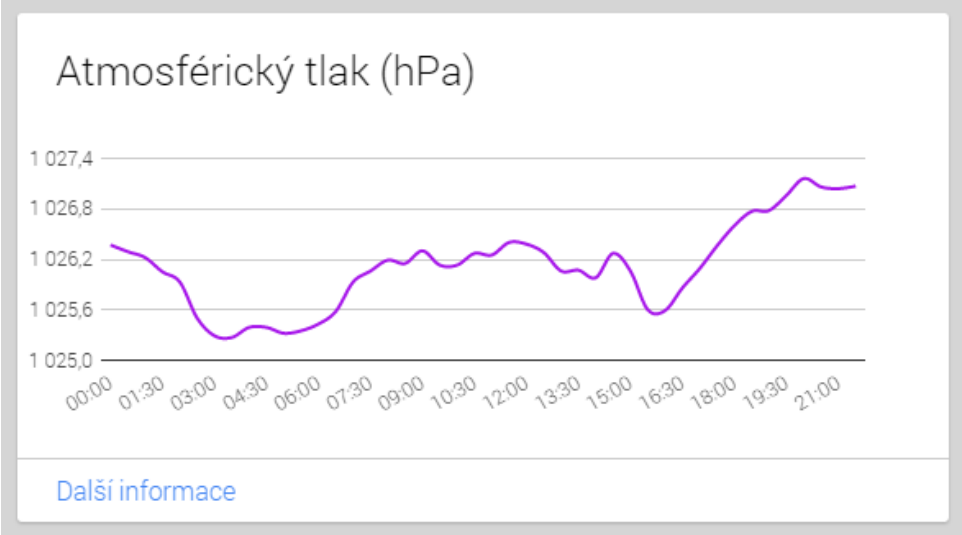
Webové rozhraní je dostupné na adrese: http://meteo.jakubtopic.cz.

4.2.1 Úvodní stránka webové aplikace



Obrázek 8: Úvodní stránka meteostanice v okně webového prohlížeče

Při návrhu webového rozhraní jsem se snažil vytvořit nějaký nekonvenční koncept, který bude vypadat na první pohled designově čistě a zároveň bude pro uživatele intuitivní. Nakonec jsem zvolil systém posloupnosti karet, tedy jakýchsi bloků, které slučují informace jednoho druhu. První karta na hlavní stránce zobrazuje čerstvá data ze senzorů, následuje karta informující o stáří těchto dat a pak karta s oblačností a rychlostí větru. Dále následují grafy hodnot jednotlivých senzorů z téhož dne. Každý tento graf je umístěn ve své kartě. Poslední dvě karty jsou odkazy na pokročilý výběr dat a doplňující informace.



Obrázek 9: Ukázka karty s grafem

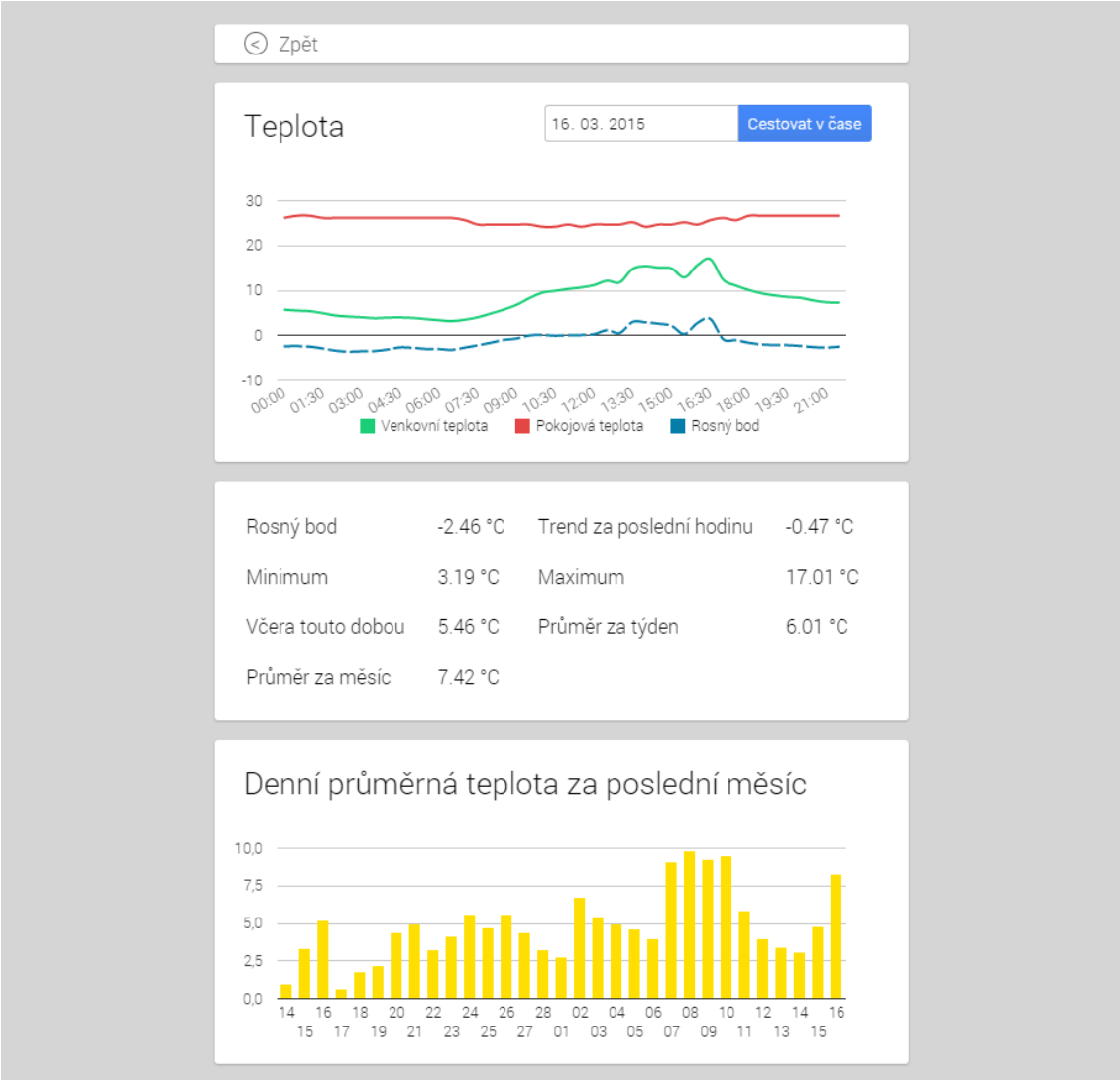
4.2.2 Stránka s detaily daného senzoru

Pokud uživatel poklepe na kartu s grafem hodnot některého z použitých senzorů, dostane se na stránku věnovanou pouze danému senzoru. Zde se zobrazí karta s tímtéž grafem a možností vybrat určitý den z minulosti a vykreslit na grafu data z vybraného dne.

Těsně pod grafem je tabulka s informacemi, jako je aktuální (dnešní) minimum a maximum, trend za poslední hodinu, hodnota včera touto dobou, průměr za posledních 7 dní a průměrná hodnota z uplynulých 30 dní. V případě teploty je zde ještě vypsán aktuální rosný bod.

Další karta obsahuje sloupcový graf, který zobrazuje denní průměrné teploty za posledních 30 dní. Na ose X jsou tedy jednotlivé dny a na ose Y jsou průměrné hodnoty daných dnů.

V případě teploty následuje za tímto grafem karta s tabulkou počtů tropických, letních, mrazových, ledových a arktických dnů aktuálního roku.



Obrázek 10: Ukázka stránky s detaily daného senzoru

4.2.3 Pokročilý výběr dat a API

Jedna z posledních karet hlavní stránky odkazuje na „Pokročilý výběr dat a API“. Jedná se o stránku s informacemi a instrukcemi pro používání mnou poskytovaného API k exportu dat z databáze mé meteostanice.

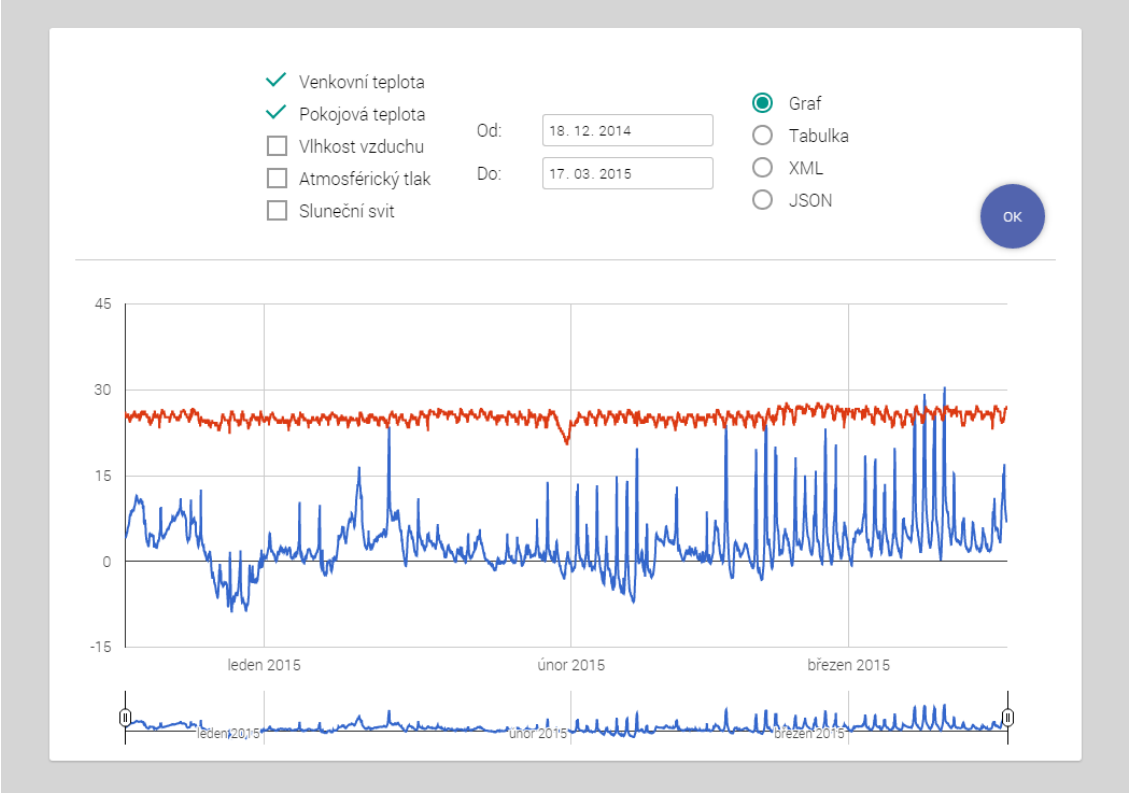
Data z databáze meteostanice lze exportovat ve formátu JSON nebo XML feedu, jehož obsah se ovlivňuje parametry v HTTP požadavku. Jednak lze vybírat určitá pole pomocí parametrů *venkovni*, *pokojova*, *vlhkost*, *tlak*, *svit*, kterým nastavíme hodnotu *on*, pokud je chceme nechat vypsat, a pak lze určit časový rozsah dat parametry *od*, *do*. Požadovaný formát data je YYYY-MM-DD.

Parametr *output* potom určuje formát výstupu. Na výběr je *xml* pro výstup ve formátu značkovacího jazyka XML, *json* pro JSON a *table* pro HTML tabulku.

Příklad HTTP požadavku, který vybere hodnoty venkovní a pokojové teploty z období 1. až 24. prosince 2014 a výstup vygeneruje ve formátu JSON, potom může vypadat takto:

http://meteo.jakubtopic.cz/api.php?api=on&Venkovni=on&Pokojova=on&od=2014-12-01&do=2014-12-24&output=json

Další funkcí této stránky je výběr a následná interpretace dat. Pro tento účel je zde jednoduchý formulář, kde lze zaškrtnout požadovaná pole (venkovní a pokojová teplota, vlhkost vzduchu, atmosférický tlak a sluneční svit), vybrat časový rozsah pomocí dvou vstupních polí a nakonec vybrat formát výstupu. Oproti API je zde možnost vybrat navíc vykreslení hodnot do grafu. Pro pohodlnější práci s grafem jsem pod něj umístil filtr rozsahu, ve kterém lze z již vykreslených hodnot vybrat pouze určité časové období – ne však kratší než jeden den.



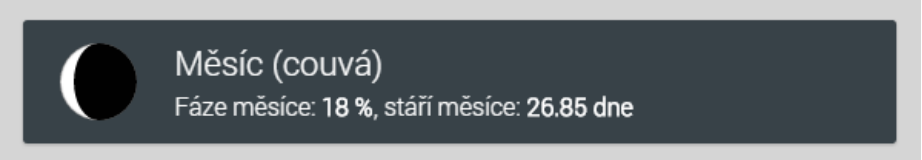
Obrázek 11: Pokročilý výběr dat

4.2.4 Doplňující informace

Poslední karta na hlavní stránce odkazuje na stránku s doplňujícími informacemi o meteostanici, kde je krátký neformální popis návrhu a funkcí meteostanice. Dále jsou zde karty s počtem dní, kdy byla meteostanice v provozu, s aktuální velikostí databáze, krátkým výpisem použitých technologií a mapou přibližného umístění. Co se týče mapy, jedná se o vloženou mapu z online služby Google Maps.

5 Rozšíření funkcí

5.1 Rozšiřující karty



Obrázek 12: Karta Měsíce

Poté, co jsem dokončil webové rozhraní meteostanice, jsem začal přemýšlet nad tím, čím ještě obohatit její funkce. Nejprve mě napadlo vypisovat v aplikaci nějaká astronomická data, která by nezávisela na pozorování, ale šlo je snadno určit výpočty. Jako první vznikla tato karta Měsíce, která zobrazuje jeho fázi, tedy jak velkou Sluncem ozářenou část Měsíce můžeme ze Země pozorovat, a jeho stáří, tedy dobu od posledního novu.

V budoucnu bych chtěl přidat další podobné karty s údaji například o Slunci (východ, západ, transity planet před Sluncem, rovnodennost, slunovrat apod.) nebo dalších astronomických objektech.

5.2 Výpis hodnot na displeji serveru

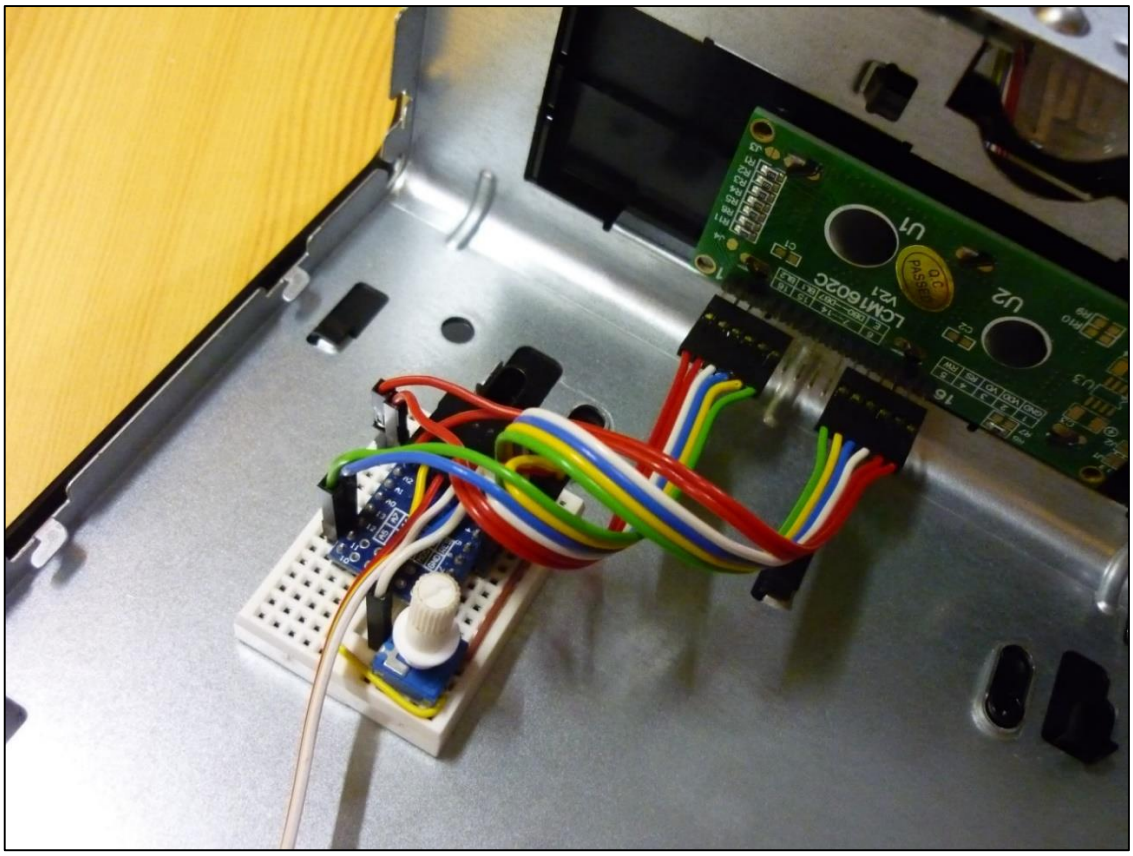
Ne vždy je po ruce chytrý telefon nebo zapnutý počítač a tak se mi zdálo rozumné neustále zobrazovat aktuální data z meteostanice na nějakém dalším zařízení. Zde jsem se inspiroval u síťových úložišť (tzv. NAS), z nichž velká část má zabudovaný malý LCD displej, na kterém zobrazují využití disků a dalších systémových prostředků.

Podobnou funkci jsem přidal i svému serveru, a to tak, že jsem připojil podsvícený alfanumerický LCD displej ke svému Arduinu. Pro displej jsem vyřízl otvor v záslepce místo

optické mechaniky. Arduinu nyní odesílám každou minutu informace o systémových prostředcích (využití a teplota procesoru, kapacita disků, stav běžících služeb, využití operační paměti a oddílu pro odkládání dat virtuální paměti). O odesílání dat se stará bashový skript (viz kapitola 3.2), který jsem musel pro tuto novou funkci modifikovat. Obdobně jsem obohatil i program Arduina. Kromě zmíněných údajů se načtou i aktuální hodnoty senzorů. Všechna data se v průběhu jedné minuty postupně zobrazí na displeji a další minutu se proces opakuje s novými, aktuálními daty. Jelikož Arduinu odesílám také datum a čas, může určit, kdy má data ze senzorů nejen načíst a zobrazit na displeji, ale i odeslat serveru. Jak jsem již zmínil, děje se tak každou 0. a 30. minutu každé hodiny.

Do budoucna ještě přehodnotím výměnu displeje. Vybrané LCD má totiž špatné pozorovací úhly a je pouze znakové, takže na něm nemohu vykreslovat ani jednoduchou grafiku. Uvažuji nad nějakým grafickým OLED displejem, na který bych mohl vykreslovat grafy vývoje hodnot senzorů. Technologie OLED by také umožnila větší pozorovací úhly.

Následují fotky realizace zmíněného informačního LCD displeje. Na prvním obrázku lze vidět použitou mikrokontrolérovou desku Arduino, která představuje rozhraní mezi serverem a senzory meteorologických prvků. Na této fotce však není k Arduinu připojena osmižilová dvoulinka, přes kterou jsou zapojeny senzory.



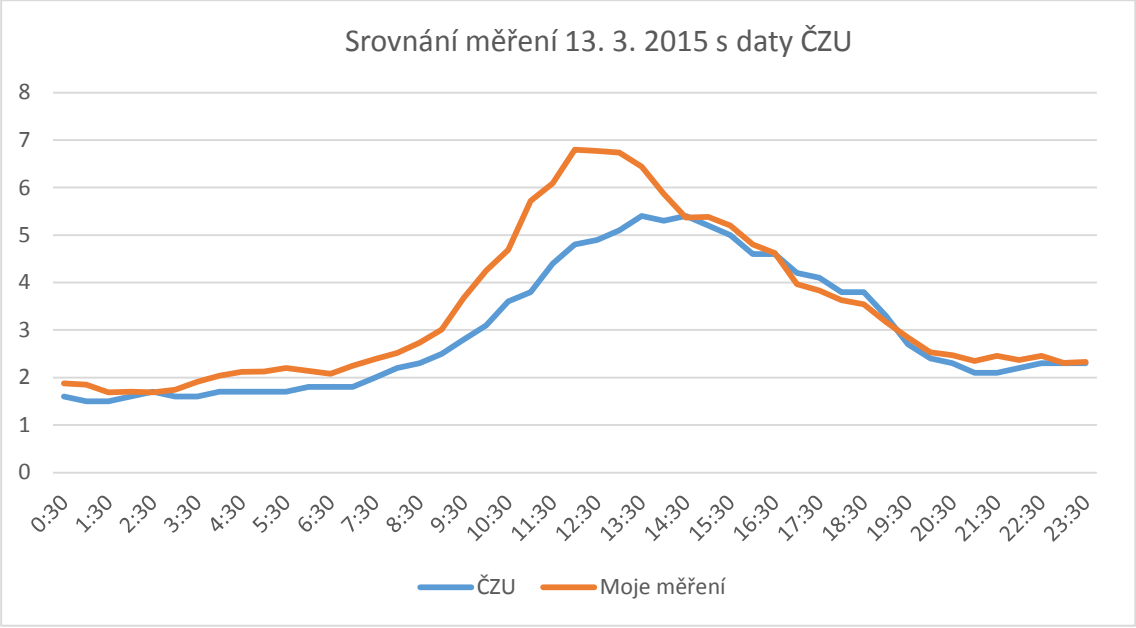
Obrázek 13: Arduino s připojeným LCD displejem již nainstalované ve skříni serveru (bez připojených senzorů meteorologických prvků)

Obrázek 14: Server s nainstalovaným LCD displejem s ukázkou zobrazených dat

6 Ověření funkcí

Vzhledem k nepříliš vhodnému umístění senzorů, které jsem však bohužel nemohl ovlivnit, nejsou naměřená data zcela relevantní. Profesionální meteorologické stanice měří například teplotu vzduchu ve výšce 2 metry nad travnatým povrchem v dostatečné vzdálenosti od všech budov. Já jsem musel senzory umístit z vnější strany prosklené lodžie (orientované na západ) našeho panelákového bytu, jiná možnost nebyla. Měření tedy provádím ve výšce zhruba 30 metrů nad zemí a ve slunečných dnech ho značně ovlivňuje teplý vzduch sálající od zdi panelového domu.

Následující graf znázorňuje měření venkovní teploty mé meteostanice a meteorologické stanice České zemědělské univerzity v Praze dne 13. 3. 2015. Z grafu je patrné, že odchylky měření jsou kromě poledních hodin do 1 °C, je však nutné vzít v úvahu, že toto měření bylo prováděno v oblačný den. Větší rozdíly v poledních hodinách jsou způsobeny špatným umístěním. V podmínkách, které by vyhovovaly normám, by bylo měření mnohem přesnější.



Závěr

V závěru své práce bych chtěl zhodnotit úspěchy a neúspěchy svého projektu, jak je vnímám. Jsem spokojen s konstrukcí meteostanice, řešením jejího propojení se serverem a s programováním webové aplikace. Meteostanice zaznamenává prakticky všechny meteorologické prvky, které jsem měl původně v plánu. Vzhledem ke špatnému umístění senzorů ale nejsou naměřená data zcela relevantní.

Stádium vývoje meteostanice rozhodně nepovažuji za konečné. V budoucnu bych chtěl rozšířit počet měřených meteorologických prvků přidáním dalších měřicích přístrojů, jako je například srážkoměr.

Také se nabízí možnost komercializace projektu, která by spočívala v návrhu *standalone* zařízení se síťovým rozhraním. Toto zařízení by stačilo pouze připojit k internetu a ke zdroji napájení, umístit na vhodné místo a naměřená data by pak samo odesílalo opět na centrální server, ovšem přes internet. K vizualizaci dat by opět sloužila webová aplikace. V tomto případě by se už vyplatilo kromě webové aplikace vytvořit nativní mobilní aplikaci. Každý majitel by se také mohl rozhodnout, zda publikovat naměřená data veřejně. Díky tomu bych mohl vytvořit veřejně přístupnou síť amatérských domácích meteorologických stanic, jejichž data bych mohl analyzovat.

Seznam použité literatury

[1] WELLING, Luke; THOMSON, Laura. *PHP a MySQL: Rozvoj webových aplikací*. Praha: SoftPress, 2005. 718 s. ISBN 80-86497-20-8.

[2] VRÁNA, Jakub. *1001 tipů a triků pro PHP*. Praha: Computer Press, 2010. 456 s. EAN 9788025129401.

[3] CASTRO, Elizabeth; HYSLOP, Bruce. *HTML5 a CSS3*. Praha: Computer Press, 2012. 439 s. ISBN 978-80-251-3733-8.

[4] *Arduino Language Reference* [online]. 2015 [cit. 2015-2-15]. Dostupné z WWW: <http://arduino.cc/en/Reference/HomePage>.

[5] *PHP Manual* [online]. 2015 [cit. 2015-2-15]. Dostupné z WWW: <http://php.net/manual/en/>.

[6] *Ubuntu Wiki* [online]. 2015 [cit. 2015-2-15]. Dostupné z WWW: <http://wiki.ubuntu.cz/>.

[7] *Official Ubuntu Documentation* [online]. 2015 [cit. 2015-2-15]. Dostupné z WWW: <https://help.ubuntu.com/>.

[8] *Dokumentace - Bosch BMP180* [online]. 2013 [cit. 2015-2-15]. Dostupné z WWW: <http://ae-bst.resource.bosch.com/media/downloads/pressure/bmp180/Flyer_BMP180_08_2013_web.pdf>.

[9] *BMP180 – Arduino library* [online]. 2013 [cit. 2015-2-15]. Dostupné z WWW: <https://github.com/sparkfun/BMP180_Breakout>.

[10] *Dokumentace – DHT11* [online]. 2010 [cit. 2015-2-15]. Dostupné z WWW: <http://www.micropik.com/PDF/dht11.pdf>.

[11] *Dokumentace – TMP36* [online]. 2015 [cit. 2015-2-15]. Dostupné z WWW: <http://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf>.

[12] *Arduino and Linux TTY* [online]. 2015 [cit. 2015-2-15]. Dostupné z WWW: <http://playground.arduino.cc/Interfacing/LinuxTTY>.

[13] *Google Visualization API Reference* [online]. 2015 [cit. 2015-2-15]. Dostupné z WWW: <https://developers.google.com/chart/interactive/docs/reference>.

[14] *Dokumentace – OpenWeatherMap API* [online]. 2015 [cit. 2015-2-15]. Dostupné z WWW: <http://openweathermap.org/api>.

[15] *W3C - Manifest for web application* [online]. 2015 [cit. 2015-2-15]. Dostupné z WWW: <http://www.w3.org/TR/appmanifest/>.

[16] *Rosný bod* [online]. 2015 [cit. 2015-2-15]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Rosn%C3%BD_bod/>.

Seznam obrázků

Seznam použitých zkratek

API	Application Programming Interface
BASH	Bourne-again shell
CRON	Command Run On
CSS	Cascading Style Sheets
GND	Ground - uzemnění
GNU	GNU’s Not Unix
GZIP	GNU Zip
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
JSON	JavaScript Object Notation
LCD	Liquid Crystal Display
NAS	Network Attached Storage
OLED	Organic Light-emitting Diode
PHP	PHP: Hypertext Preprocessor
RX	Receive - přijmout
SQL	Structured Query Language
SSL	Secure Sockets Layer
SVG	Scalable Vectore Graphics
TLS	Transport Layer Security
TX	Transmission - přenos
USART	Universal Synchronous / Asynchronous Receiver and Transmitter
USB	Universal Serial Bus
WWW	World Wide Web
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language