VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV VÝROBNÍCH STROJŮ, SYSTÉMŮ A ROBOTIKY
INSTITUTE OF PRODUCTION MACHINES, SYSTEMS AND ROBOTICS

PLATFORMA PRO MĚŘENÍ, ZÁZNAM A VYHODNOCENÍ DAT NA RAPSBERRY PI
PLATFORM FOR MEASURING, RECORDING AND EVALUATING DATA ON RASPBERRY PI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Patrik Pokorný**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Rostislav Huzlík, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav výrobních strojů, systémů a robotiky
Student:	**Patrik Pokorný**
Studijní program:	Strojírenství
Studijní obor:	Základy strojního inženýrství
Vedoucí práce:	**Ing. Rostislav Huzlík, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Platforma pro měření, záznam a vyhodnocení dat na Rapsberry Pi

Stručná charakteristika problematiky úkolu:

Předmětem práce je navrhnout a realizovat platformu pro měření, záznam a zpracování dat na platformě Rapsberry Pi. V rámci práce se student nejprve seznání s možnostmi využití Rapsberry Pi a předloženými senzory. Dále student ve vhodném programovacím jazyky (např. Python) vytvoří program pro měření s využitím předložených senzorů, záznam dat z nich získaných a jejich vyhodnocení.

Cíle bakalářské práce:

Seznámení se s možnostmi platformy Rapsberry Pi.
Seznámení se s předloženými senzory.
Vytvoření programu pro měření, záznam a vyhodnocení dat na platformě Rapsberry Pi.
Otestování programu a zhodnocení možnosti využití Rapsberry Pi pro měření, záznam a vyhodnocení dat.
Závěr a doporučení pro praxi.

Seznam doporučené literatury:

PILGRIM, Mark, 2010. Ponořme se do Python(u) 3: Dive into Python 3. Praha: CZ.NIC. CZ.NIC. ISBN 978-80-904248-2-1.

BENTLEY, John P., 2005. Principles of measurement systems. 4th ed. New York: Pearson Prentice Hall. ISBN 0-13-043028-5.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

doc. Ing. Petr Blecha, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Tato bakalářská práce má za cíl vytvořit platformu pro měření, záznam a vyhodnocení dat na základu mikropočítače Raspberry Pi. K tomuto je využito dodatečného hardwaru v podobě základní desky a příslušenství od výrobce Seeed Studio. Měřené veličiny jsou v tomto případě teplota vzduchu, vlhkost vzduchu, intenzita světla a intenzita hluku. Pro měření těchto veličin byl napsán program v jazyce Python 3, který pro ukládání dat používá databázi MariaDB. Ještě před zápisem do databáze jsou data zpracována a přepočtena na odpovídající jednotky. Data z databáze je poté možno zobrazit a vykreslit do grafu pomocí aplikace napsané v témže jazyce s využitím frameworku Kivy.

ABSTRACT

Objective of this bachelor thesis is to create a platform for measurement, recording and evaluation of data based on Raspberry Pi microcomputer. For that purpose, additional hardware in form of a PCB and accessories from Seeed studio is used. Measured values in this scenario are air temperature and humidity, illuminance and acoustic intensity. For measuring these values, a Python 3 program was written, using MariaDB database for data storage. Prior to being saved into the database, data is processed and converted into appropriate units. Data from the database can then be plotted into a graph through an application written in the same language, using the Kivy framework.

KLÍČOVÁ SLOVA

Raspberry Pi, Python 3, Kivy framework, Seeed Studio Grove System, databáze MariaDB

KEYWORDS

Raspberry Pi, Python 3, Kivy framework, Seeed Studio Grove System, MariaDB database

BIBLIOGRAFICKÁ CITACE

POKORNÝ, Patrik. Platforma pro měření, záznam a vyhodnocení dat na Rapsberry Pi [online]. Brno, 2020 [cit. 2020-04-05]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/125068. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav výrobních strojů, systémů a robotiky. Vedoucí práce Rostislav Huzlík.

PODĚKOVÁNÍ

Rád bych tímto poděkoval mému vedoucímu Ing. Rostislavu Huzlíkovi, Ph.D. za konzultace a věcné připomínky při vedení této bakalářské práce. Dále děkuji svým kamarádům za pomoc při studiu a své rodině za podporu při studiu.

ČESTNÉ PROHLÁŠ ENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Rostislava Huzlíka, Ph.D. a s použitím literatury uvedené v seznamu.

V Brně dne 1.5. 2020 ……

Patrik Pokorný

OBSAH

1 ÚVOD 15

2 RASPBERRY PI 17
2.1 Historie 17
2.2 Použitý model 17
2.3 Operační systém 18
2.4 Porovnání s Raspberry Pi 4 18
2.5 Rozšíření a moduly 19
2.6 Další využití 19

3 SEEED STUDIO GROVE SYSTEM 21
3.1 Základní deska 21
3.1.1 Popis portů [19] 21
3.2 Použité snímače 22
3.2.1 Senzor teploty a vlhkosti vzduchu 22
3.2.2 Světelný senzor 22

4 POUŽITÝ SOFTWARE 24
4.1 GrovePi+ 24
4.2 Programovací jazyk Python 3 24
4.3 Databáze MariaDB 24
4.3.1 Konfigurace databáze 24
4.4 Správa databáze 25
4.5 Multiplexer 25
4.6 Kivy framework 25

5 PROGRAM PRO NAČTENÍ MĚŘENÝCH DAT A JEJICH ZÁPIS DO DATABÁZE 26
5.1 Vlákna programu 26
5.1.1 Vlákno getSoundLightValues 26
5.1.2 Vlákno getDHT 26
5.1.3 Vlákno writeSQL 26
5.1.4 Vlákno printAverage 27
5.2 Rychlost načítání dat 27
5.3 Importované balíky 27
5.3.1 databasesDB 28
5.3.2 emailDB 28
5.4 Převod dat 28
5.4.1 Převod jednotek zvuku 29
5.4.2 Převod jednotek světla 29

6 APLIKACE PRO ZOBRAZENÍ DAT V DATABÁZI A JEJICH VYKRESLENÍ DO GRAFU 31
6.1 Spuštění aplikace 31
6.2 Navigace 31
6.3 Omezení počtu zobrazených dat 32
6.4 Testování a výstup 32

7 ZHODNOCENÍ A DISKUZE 34

9 ZÁVĚR 35

10 SEZNAM POUŽITÝCH ZDROJŮ 36

11 SEZNAM ZKRATEK, SYMBOLŮ, OBRÁZKŮ A TABULEK 38

11.1 Seznam zkratek 38

11.2 Seznam tabulek 39

11.3 Seznam obrázků 39

12 SEZNAM PŘÍLOH 41

1 ÚVOD

Platforma Raspberry Pi vznikla jako podpůrný prostředek pro výuku programování a IT. Není proto divu, že vznikla celá řada nejrůznějších rozšíření a softwarů třetích stran, které posouvají možnosti využití této platformy a přispívají tak kreativitě jejích uživatelů. Za následek se dá považovat, že na internetu existuje nekonečné množství nejrůznějších nápadů na využití platformy společně s návody na jejich realizaci. Velkou výhodou takto velké komunity je, že kdykoliv neznáte odpověď na nějakou otázku související s tématem Raspberry Pi, je velmi vysoká pravděpodobnost, že už se na ni někdo ptal před vámi a že v online světě na ni existuje i odpověď. Tato práce přinese odpovědi na otázky související s tématem tvoření platformy pro měření, záznam a vyhodnocení dat na Raspberry Pi.

V úvodní části představí stručně historii, současnost a použitý model jak z hlediska hardwarové výbavy, tak z hlediska softwarové výbavy. Představí některá z možných rozšíření a možnosti využití platformy v různých odvětvích. Dále čtenáře blíže seznámí s rozšířením platformy nutným pro měření a představí senzory použity při měření i z hlediska fyzikální podstaty jejich fungování. Čtenář bude také seznámen s nutným softwarem, který musí být na platformě předinstalován a důvody, které vedly k jeho výběru. Dále pak pokračuje vysvětlením principu činnosti jednotlivých hlavních částí programu pro měření a záznam dat. Vyhodnocování dat poté probíhá pomocí aplikace, jejíž ovládání je rovněž popsáno a zmíněny jsou i její limity. Zmíněno je i testování a předveden je výstup jak programu pro měření a záznam, tak aplikace pro vyhodnocení dat.

2 RASPBERRY PI

Raspberry Pi je jednodeskový mikropočítač v dnešní době oblíbený nejen u nadšenců pro IT. Jeho cena ho činí vhodným kandidátem pro výuku informatiky na školách a poskytne taky slušný základ pro nejrůznější domácí projekty, zároveň je ale použitelný pro aplikaci v automatizaci. Dá se použít v chytré domácnosti, ale skýtá i prostor pro vývoj aplikací. Nezřídka se s ním setkáme v robotice.

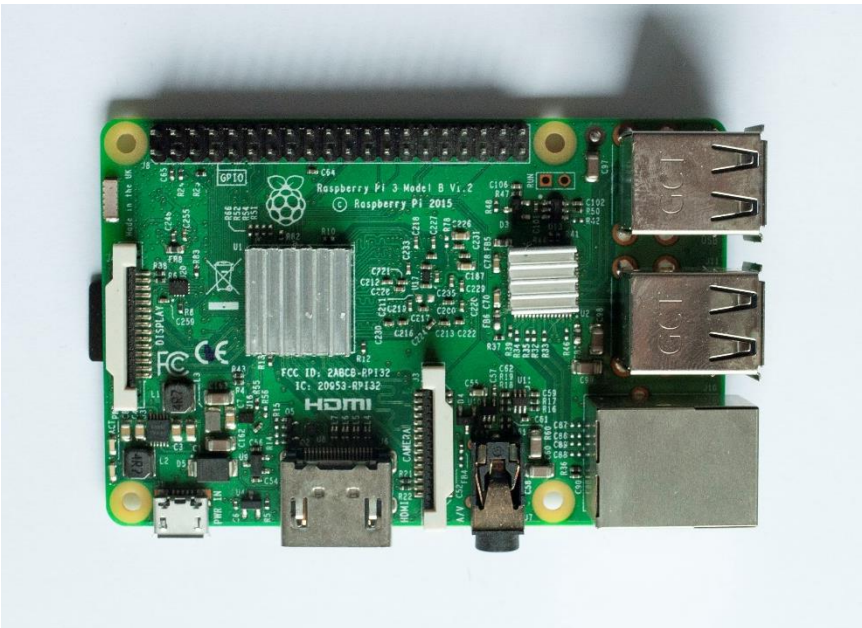
V nabídce je kromě platformy o velikosti platební karty použité v této bakalářské práci a popsané níže také Raspberry Pi Compute Module určený pro průmyslové aplikace a do modulárních systémů. Tento výpočetní modul se připojuje do socketu DDR2-SODIMM a nenajdeme na něm vychytávky jako WiFi, Bluetooth nebo HDMI port, zato nabízí rozšířené rozhraní GPIO. Pro nenáročné aplikace existuje také Raspberry Pi Zero, které má poloviční rozměry klasické verze a cenovku něco málo přes 10$. Proto je nutné zmínit, že pokud není v práci uvedena konkrétní platforma, myslí se pojmem Raspberry Pi mikropočítač ze stejné skupiny, ze které pochází mikropočítač použitý pro účely této práce.

2.1 Historie

První prototyp mikropočítače vznikl pod rukami anglického inženýra Ebena Uptona v roce 2006 v počítačové laboratoři na Cambridge University. S později prodávanými modely nemá příliš společného, další vývoj trval 6 let. Cílem bylo vytvořit levný počítač s cenovkou do 30$ a tak zpřístupnit programování každému zájemci. První zmínka o Raspberry Pi Foundation, která dnes stojí za vývojem platformy, se objevuje v roce 2009. O dva roky později představuje společnost první oficiální prototyp. Ten se sice stále významně lišil od prvního komerčně uvedeného výrobku. Neměl výstup HDMI ani USB porty, přesto bylo o rok později na anglický trh uvedeno první Raspberry Pi. Vyrábět tento studijní a testovací počítač se měl od začátku v továrně společnosti Sony v Anglii, vyjednat podmínky se ale podařilo až s první sérií dodanou z Číny. V současné době existují již 4 komerčně úspěšné generace, z nichž každá obsahuje více modelů, které se od sebe liší hardwarovou výbavou. Na konci roku 2019 bylo prodáno přes 30 milionů kusů.[4][5][6]

2.2 Použitý model

Pro účely této závěrečné práce budu využívat generaci Raspberry Pi 3 Model B+ (na obrázku) o rozměrech 85,6 mm x 56,5 mm x 17 mm, který je vybaven 64bitovým čtyřjádrovým ARM procesorem o taktu 1,4 GHz a sdílenou pamětí RAM o velikosti 1GB. O konektivitu se starají 4 porty USB 2.0, grafický výstup zajišťuje konektor HDMI a přístup k internetu zajišťuje standartní ethernetový konektor RJ45, nebo vestavěná WiFi. Jako interní paměť používá Raspberry Pi microSDHC kartu, jejíž kapacitu v základu může uživatel zvolit od 4GB do 32GB. Napájení je řešeno pomocí 5V microUSB portu, kdy zdroj by měl být schopen dodávat minimálně 2,5A v závislosti na dalších zařízeních připojených k periferiím. Silnou zbraní u mikropočítačů Raspberry Pi je 40x pin GPIO, pomocí kterého můžeme připojit množství modulů a rozšíření. Oproti běžným počítačům tato platforma využívá společný čip pro procesor, grafiku i operační paměť



Obr. 1) Raspberry Pi 3 Model B+ použit pro účely této práce

2.3 Operační systém

Raspberry Pi využívá jako operační systém některou z distribucí Linuxu upravenou pro jednodeskové počítače. Každá distribuce bývá optimalizována přímo pro konkrétní platformu, čímž se dosahuje nejvyššího výkonu a stability systému.

Pro účely této práce byl na Raspberry Pi nainstalován operační systém Raspbian, který je i oficiálně podporován nadací Raspberry Pi Foundation. Tento operační systém je možné spustit buď s grafickým rozhraním podobnému např. Ubuntu, nebo bez GUI, kdy se musíme spokojit s rozhraním terminálu. Grafické rozhraní bylo z důvodu nepotřebnosti vypnuto.

Uživateli je ale umožněna i instalace jiných operačních systémů vhodných pro konkrétní aplikaci. Pro programování IoT zařízení se jeví jako vhodný kandidát například Windows IoT Core nebo Ubuntu Core. Pokud by Raspberry Pi bylo určeno k provozování multimediálního centra, nabízí se Open Source Media Center nebo OpenELEC. Možné využití nachází platforma také jako emulátor pro staré počítačové a konzolové hry. Pro tyto účely existují operační systémy jako Lakka nebo RetroPie. Pro aplikaci v chytré domácnosti existuje také OS Domoticz.[7][8]

Obecně lze říct, že na platformu Raspberry Pi s OS Raspbian lze instalovat veškerý software funkční na distribuci Debian. Je ale nutné myslet na omezené hardwarové možnosti Raspberry Pi.

2.4 Porovnání s Raspberry Pi 4

V době psaní této práce byla nově vydaná již celkově čtvrtá generace Raspberry Pi. Model 3B+ použitý v této práci lze chápat jako nejvyšší verzi třetí generace. Níže je uvedeno porovnání s nejvyšší verzí čtvrté generace.

Tab 1) Porovnání modelů Raspberry Pi

	Model 3B+	Model 4B
Procesor	1,4GHz (4 jádra) Cortex-A53	1,5GHz (4 jádra) Cortex-A73
Paměť	1GB LPDDR2	4GB LPDDR4
Grafika	VideoCore IV GPU	VideoCore VI GPU
Video výstup	1x HDMI	2x micro-HDMI
Ethernet	300Mbps	1000Mbps
USB	4x USB 2.0	2x USB 2.0 + 2x USB 3.0
GPIO	40x GPIO pin	
Napájení	microUSB	USB-C
Cílová cena	35$	55$

Parametry procesoru u novější verze zlepšuje kromě vyššího výchozího taktu také novější architektura. Podle vyjádřeni Raspberry Pi Foundation můžeme očekávat až trojnásobné navýšení výkonu proesoru. Oproti staršímu modelu je také k dostání s pamětí RAM až 4GB standartu LPDDR4, kdy model 3B+ se spoléhal na 1GB standartu LPDDR2. Rychlost sítě u nového modelu už není omezena společným řadičem pro USB 2.0 a můžeme tak hovořit o skutečném gigabit ethernetu.[9]

Grafická část je povýšena na 2x HDMI 2.0 výstupy, které by měly zvládnout video o rozlišení 4K při obnovovací frekvenci 30Hz při užití obou výstupů, nebo v případě využití jednoho monitoru 4K rozlišení s obnovovací frekvencí 60Hz. Předchozí model byl vybaven 1x výstupem HDMI 1.4 s maximálním rozlišením FullHD (1080p). Co se týče příslušenství, Raspberry 4 zůstává kompatibilní s předchozími modely.[10][11]

2.5 Rozšíření a moduly

K Raspberry Pi je možné přikoupit množství modulů či setů, kterými lze rozšířit využití platformy. Zmiňme například TV HAT, který rozšíří základní funkci o příjem DVB-T2 signálu pro sledování televize. Zajímavým doplňkem je PoE HAT, pomocí kterého je možné napájet mikropočítač přes ethernetový kabel. Pokud chceme používat grafické rozhraní a nechceme připojovat k modulu monitor či televizi, modul může být dovybaven o standartní či dotykový LCD displej, který se připojuje do DSI portu. Pro kreativní využití je možné přikoupit rozšíření s 8x8 maticovým led displejem. V nabídce je také oficiální kamera s 8MPx a rozlišením záznamu FullHD při 30fps.

2.6 Další využití

Pro aplikace využívající značný výpočetní výkon se dá vícero Raspberry Pi spojit do clusteru. Taková aplikace může být například vytváření panoramatických snímků složených z několika pořízených pomocí fotoaparátu. Extrémní případ je cluster vyvíjen Národní laboratoří Los Alamos složený z několika stovek desek Raspberry Pi s celkovým počtem okolo 3000 jader sloužící pro výzkum a vývoj chování superpočítačových clusterů.[12][13]

V době koronavirové krize vznikl v Americe také plicní ventilátor, který kombinuje lehce dostupné díly a Raspberry Pi. Vážný nedostatek plicních ventilátorů v Kolumbii dokonce vedl k tomu, že s testováním ventilátoru začaly tamní nemocnice. [14][15]

V technické praxi je například možné se setkat s průmyslovým počítačem Revolution Pi, což je počítač modulární koncepce založený na Raspberry Pi Compute modulu. Konkrétní případ využití průmyslového počitače je například ovládání kolaborativního robota určeného k paletizaci, nebo jako prvek pro zpracování dat v mobilní čističce vody.[16]

Na internetu existují i návody na sestavení datových úložišť, tyto projekty ale naráží na konstrukční limity Raspberry Pi. Platforma totiž není vybavena vysokorychlostní sběrnicí pro připojení pevného disku (např. SATA). Pevný disk tak musí být připojen pomocí USB, a i když bychom měli na mysli USB 3.0, rychlosti přes toto rozhraní nedosahují rychlostí SATA.[20] Pro domácí využití může být ale taková verze úložiště dostatečná.

3 SEEED STUDIO GROVE SYSTEM

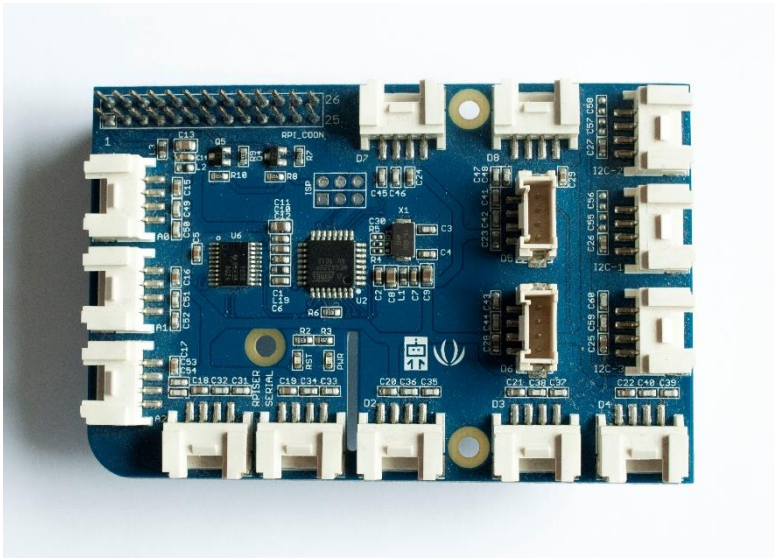
Jde o modulární rozšíření umožňující rychlé sestavení funkčního prototypu. Skládá se ze základní desky s mikroprocesorem, ke které můžeme pomocí standardizovaných konektorů připojit množství modulů, které lze zakoupit jak v setu, tak jednotlivě.[17]

V této práci byl použit set GrovePi+ kompatibilní s Raspberry Pi, který obsahuje snímače teploty a vlhkosti vzduchu, vzdálenosti, úhlu natočení, světla, zvuku, relé modul, tlačítko, LCD displej, LED diody a nezbytné příslušenství.

Nabídka výrobce obsahuje také sety pro jiné platformy, jako například Arduino, Linkit ONE, Intel Edison a jiné. Tyto sety se liší pouze základní deskou, která je přizpůsobena pro danou platformu. Nabízené vybavení je pak kompatibilní se všemi základními deskami nabízenými výrobcem.

3.1 Základní deska

Základní deska GrovePi+ (na obrázku) je nezbytnou součástí platformy pro měření. Základní deska GrovePi+ je nasazena na GPIO header Raspberry Pi. Senzory jsou poté připojovány k jednomu z patnácti 4pinových portů (7x digital port, 3x analog port, 3x I2C port a 2x SPI port) umístěných na základní desce. Srdce základní desky GrovePi+ tvoří mikrořadič ATMEGA328P obsahující mimo jiné 10bitový A/D převodník. Komunikace mezi základní deskou a mikropočítačem je prováděna pomocí I2C protokolu s výchozí přenosovou rychlostí 100kb/s a možností přetaktování až na 400kb/s.[18]



Obr. 2) Základní deska GrovePi+

3.1.1 Popis portů [19]

Analog – signál zpracován A/D převodníkem, převeden na I2C a poslán na Raspberry Pi. V případě potřeby můžou být analogové porty překonfigurovány na digitální

Digital – porty připojeny na digitální vstupy mikrořadiče a signál převeden na I2C, poté poslán na Raspberry Pi

I2C – tato sběrnice je přes převodník logických úrovní napojena přímo na Raspberry Pi

SPI – (Serial Peripheral Interface) – jeden port slouží k připojení některého z modulů (např. měřič CO_2) s možností přímého přístupu z Raspberry Pi, druhý slouží pro programování mikrořadiče ATMEGA.

3.2 Použité snímače

Po dohodě s vedoucím této bakalářské práce budou využity níže specifikované snímače.

3.2.1 Senzor teploty a vlhkosti vzduchu

Kombinovaný senzor je vybaven pro měření teploty termistorem s negativním teplotním koeficientem a pro měření vlhkosti kapacitním snímačem. Rozsah měření teploty je od 0 °C do 50 °C s přesností ±2 °C a rozsah měření vlhkosti od 20% do 90% RH s přesností ±5% RH.[21]

Termistor je prostý rezistor upraven tak, aby změna odporu byla se změnou teploty co nejvyšší. Negativní teplotní koeficient znamená, že s rostoucí teplotou odpor klesá (někdy se mu proto říká negastor). Jedná se o polovodičovou součástku vyrobenou metodou spékání kovů jako křemík, železo, mangan a jiné. Nevýhodou je nelineární závislost odporu na teplotě.[3]

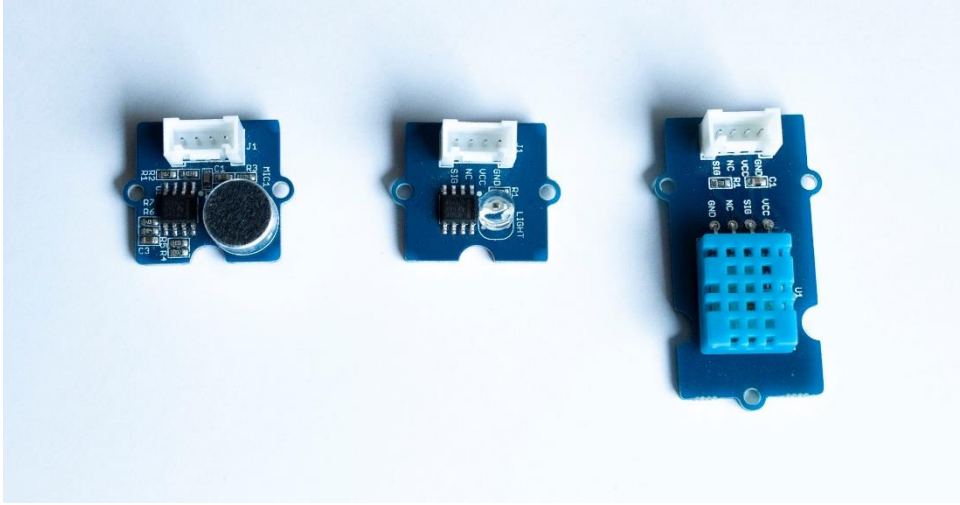
Kapacitní snímač pro měření vlhkosti je upravený kondenzátor, kde mezi elektrody je místo dielektrika vložen materiál se schopností absorbovat vodu (např. sůl nebo speciální polymery). Se vzrůstající vlhkostí se odpor mezi elektrodami snižuje, čímž dochází k nárustu proudu. Závislost proudu je přímo úměrná vlhkosti.[22]

Tento senzor je ke GrovePi+ připojen k digitálnímu portu. Jelikož je tento senzor reálně kombinací dvou snímačů, součástí senzoru je mikročip, který obstarává data z termistoru a kapacitního snímače a zpět posílá informaci v podobě 40 bitů (teplota a vlhkost je tvořena pakety po 16 bitech a 8 bitů tvoří kontrolní součet).[23]

3.2.2 Světelný senzor

Jde o fotorezistor, což je polovodičová součástka citlivá na světlo. Princip souvisí s předáváním energie mezi elektricky nabitými částicemi, konkrétně mezi fotonem ze zdroje světla a elektronem v elektrickém obvodu. Když foton dopadne na fotocitlivou vrstvu rezistoru, předá svoji energii elektronu v této vrstvě, který tak získá energii potřebnou k opuštění valenční sféry svého atomu. Ten se dále pohybuje obvodem jako volný elektron, čímž zvyšuje vodivost obvodu. Z toho vyplývá, že při větším nasvícení se odpor na fotorezistoru sníží.[2]

Použitý senzor se skládá z fotorezistoru LS06-S a zesilovače LM358. Senzor se připojuje na analogový vstup GrovePi+, kde je pak signál vzorkován pomocí zmíněného A/D převodníku.



Obr. 3) Senzory použity v této práci. Z leva to jsou: senzor zvuku, senzor světla a kombinovaný senzor teploty a vlhkosti vzduchu

4 POUŽITÝ SOFTWARE

4.1 GrovePi+

Ke GrovePi+ je z githubu volně ke stažení repozitář obsahující balíky s firmwarem, vzorovými příklady a knihovnami použitými pro komunikaci se senzory.

4.2 Programovací jazyk Python 3

Celá aplikace je naprogramována v jazyku Python 3 pro jeho jednoduchost a možnost migrace mezi různými operačními systémy. Výhodou Pythonu 3 je také to, že je součástí standardní výbavy Raspbianu a výrobci GrovePi+ poskytují k výrobku knihovny pro tento programovací jazyk.

4.3 Databáze MariaDB

Data ze senzorů světla a zvuku je nutné také ukládat. Pro tento účel byla zvolena databáze MariaDB. Tato databáze je open source větví MySQL se silnou uživatelskou základnou a dobrou dokumentací. Během testování nebyl zaznamenán žádný problém s rychlostí zápisu do databáze a propojení databáze s vytvořeným programem se ukázalo jako velmi jednoduché a bezchybné.

4.3.1 Konfigurace databáze

Data jsou zapisována do tabulky **raspi_table**. Tato tabulka se skládá z celkem 6 sloupců, kdy z důvodů optimalizace využité paměti má každý sloupec vhodný datový typ.

První sloupec se jmenuje **id** v datovém typu unsigned int(). Jedná se o 4bajtové kladné číslo, takže možný rozsah je $0 - 2^{32}$-1, což by stačilo na více jak 13,5 let nepřetržitého zápisu při rychlosti 10 zápisů za sekundu. Tento sloupec zároveň slouží jako primární klíč tabulky a je tak automaticky generován (inkrementován) při každém zápisu do tabulky.

Druhý a třetí sloupec se nazývají **sound** a **light** a sdílejí 2bajtový datový typ unsigned smallint(). Jde tedy o kladné číslo v rozsahu $0 - 2^{16}$-1. Takto velká hodnota by nikdy nemohla být naměřena, ale menší datový typ už by byl z hlediska rozsahu nedostatečný. Tyto sloupce, jak už název napovídá, slouží k uložení úrovně hladiny zvuku a světla.

Čtvrtý a pátý sloupec slouží k uložení teploty a vlhkosti vzduchu. Sdílejí datový typ unsigned float, takže se jedná o typ s plovoucí desetinnou čárkou o velikosti 4 bajty. Unsigned v tomto případě nezvětší rozsah (ani to není potřeba, teplota a vlhkost je měřena na jedno desetinné místo), ale pouze zakáže záporná desetinná čísla.

Poslední šestý sloupec uchovává informaci o čase a datu zápisu do databáze. Jedná se o datový typ datetime() s přesností na mikrosekundy.

Prázdná tabulka zabírá velikost 12MB. Společně s 4 338 500 řádků (přibližně 5 dní záznamu) zabere necelých 214MB, což vychází přibližně 47 bajtů na jeden záznam.

Samotná tabulka určena jako záložní v případě výpadku spojení s primární databází není uložena na systémové SDHC kartě. Veškerá data jsou uložena na externím nosiči (USB klíčence). Je tak učiněno z toho důvodu, že tyto přenosné nosiče (paměťové karty, USB klíčenky a jiné flash paměti) jsou náchylnější na počet přepsání. Pokud by tedy byla vlivem velkého počtu zápisů do databáze paměť poškozena, nestalo by se tak na systémové paměti a přišli bychom „pouze‘‘ o tabulku. Samozřejmě i to by mohlo znamenat pro uživatele velký

problém. Řešení je tedy periodicky zálohovat data z primárního úložiště na sekundární. K tomu se dá využít například bezplatného softwaru rsync.

4.4 Správa databáze

Původně zamýšlený nástroj pro správu databáze byl PHPMyAdmin. Jedná se o bezplatný software pro správu databází a jejich tabulek. PHPMyAdmin je přístupný přes prohlížeč na portu 80 přes nezabezpečené připojení HTTP. To se nejevilo jako problém pro případ, kdy je správa databáze prováděna z lokální sítě. Pro vzdálený přístup je ale nutné jej z důvodu bezpečnosti dodatečně zabezpečit pomocí SSL certifikace a změnit tak připojení na HTTPS.[24]

Oproti tomu aplikace MySQL workbench není instalována na serveru ale na klientském počítači, čímž lze ušetřit místo na Raspberry Pi. Pro svoji funkčnost nepotřebuje instalovat PHP ani HTTP server, navíc je vybavena možností tunelovat připojení přes SSH a tak přistupovat k databázi i z vnější sítě pomocí šifrovaného připojení.

4.5 Multiplexer

Platformu pro měření může být potřeba umístit do obtížně přístupných prostor. Z tohoto důvodu je žádoucí, aby pro ovládání platformy a spuštění programu nebylo zapotřebí připojovat monitor ani klávesnici. Proto je celé ovládání zamýšleno přes terminál pomocí SSH.

Multiplexer kromě otevření více virtuálních terminálů umožňuje přerušit SSH spojení s platformou bez ukončení spuštěných instancí a v případě potřeby je možné kdykoliv se znovu připojit a zobrazit si například aktuální hodnoty senzorů nebo případné chybové hlášení.

Pro potřeby této práce byl na základě předchozí zkušenosti zvolen program TMUX.

4.6 Kivy framework

K samotnému programu pro zápis naměřených bylo vytvořeno ještě grafické rozhraní pro zobrazení záznamů v databázi a jejich vykreslení do grafu. K tomuto bylo využito frameworku Kivy určenému pro jazyk Python.

Framework je k dispozici zdarma a zjednodušuje vývoj programů pro operační systémy Windows, Linux, Android, OS X a jiné.[25]

5 PROGRAM PRO NAČTENÍ MĚŘENÝCH DAT A JEJICH ZÁPIS DO DATABÁZE

Po spuštění programu je uživatel vyzván k výběru databáze, do které se budou zapisovat naměřené hodnoty. Pokud vybere databázi ze seznamu, zobrazí se uživateli výzva k zadání počtu požadovaných vzorků. Minimální počet vzorků je nastaven na 100 kvůli velikosti paketů posílaných do databáze (menší počet vzorků by nebyl zapsán). Poté následuje hlavní část programu, kde se postupně spustí jednotlivá vlákna programu.

5.1 Vlákna programu

Program je koncipován jako vícevláknový, to znamená že jednotlivá vlákna fungují do jisté míry jako samostatné procesy a v případě výskytu chyby v jednom z nich neovlivní tato chyba chod zbytku programu.

5.1.1 Vlákno getSoundLightValues

Toto vlákno se stará o získávání hodnot ze senzoru hluku a světla. Běží buď v nekonečném cyklu, nebo dokud se počet naměřených hodnot nevyrovná hodnotě zadané na začátku uživatelem. Při každém průchodu pomocí knihovny **grovepi** zavolá senzor a naměřenou hodnotu uloží do příslušné proměnné. Proměnné jsou dále vloženy do dvou FIFO front a v dalším opakování cyklu jsou přepsány novou hodnotou.

5.1.2 Vlákno getDHT

V tomto vláknu jsou pouze načteny hodnoty teploty a vlhkosti vzduchu, které jsou uloženy do globálních proměnných. Dále program zkontroluje, zda jsou hodnoty v proměnných čísla a pokud ano, jsou dále vypsány na konzoli. Pokud došlo při načtení dat k chybě, data jsou načtena znovu. Jelikož se u teploty a vlhkosti vzduchu nepředpokládá výrazně rychlá změna, je vlákno uspáno na 30 sekund. Děje se tak i z důvodu toho, že načtení teploty a vlhkosti vzduchu trvá výrazně déle než načtení ostatních hodnot (rychlost načtení dat z DHT senzoru se pohybuje přibližně kolem 0,3 s).

V programátorské praxi se užívání globálních proměnných nedoporučuje, což platí dvojnásob u multivláknového programu.[1][26] V tomto případě je ale možné proměnnou považovat za konstantu. Jelikož ke změně proměnné dochází pouze v tomto vláknu a ostatní části programu hodnotu proměnné pouze načítají, neměl by tento způsob vyvolat žádné potíže. Je to také nejjednodušší způsob, jak sdílet proměnnou mezi více vlákny.

Jelikož vlákna pracují nezávisle na sobě, mohlo by dojít při pokusu načtení dat z více senzorů současně k chybě a pádu programu. Proto musí při volání senzoru dojít k synchronizaci všech vláken, k čemuž slouží příkazy **lockA.acquire()** a **lockA.release()**.

5.1.3 Vlákno writeSQL

Nejdůležitější částí programu je vlákno writeSQL. To je rozděleno na dvě hlavní části. První část **main_SQL()** je spuštěna jako první a ihned dojde k pokusu o navázání spojení s databází zvolenou uživatelem. Pokud je spojení úspěšně navázáno, instancuje se seznam **querryList**, do kterého jsou později přidávány jednotlivé řádky databáze. Ještě než se tak stane, jsou data vytažena z FIFO fronty přepočtena na známé jednotky a převedeny na řetězec. Každých 100 řádků je vyslán pokyn **executemany**, který vloží celý **querryList** do databáze. Po vložení dat dojde k odpojení od databáze, vyčištění proměnné **querryList** a celý proces se opakuje. Celý

cyklus je ukončen, pokud byly všechny hodnoty z FIFO fronty zapsány do databáze a během čtyř sekund nedošlo k načtení dalších dat.

V případě výpadku spojení během zápisu do databáze je vyvolána výjimka a část **main_SQL()** je opětovně volána, ovšem spojení je navázáno s databází instalovanou na měřicí platformě. Pokud je spojení úspěšně navázáno, pokračuje záznam dat tak jak je výše popsáno.

Pokud by se ale nepodařilo navázat spojení ani s lokální záložní databází, například pokud by tato služba nebyla spuštěna, program se přepne do druhé části **backup_File()**. Při inicializaci této druhé části je vytvořen textový soubor **backup.txt**, kam jsou data po řádcích zapisována. Pokud by se stalo, že tento soubor už existuje a jsou v něm uložena nějaká data, jsou nová data od starých oddělena řádkem s popiskem jednotlivých pozic.

5.1.4 Vlákno printAverage

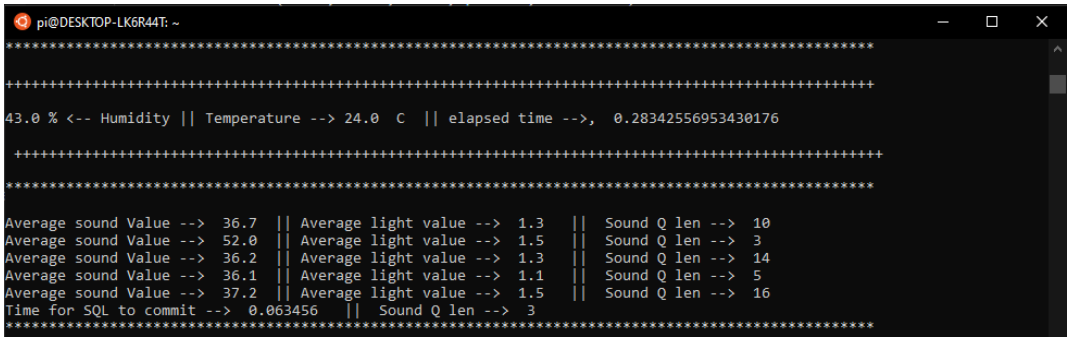
Program kromě zápisu do databáze taky vypisuje hodnoty do terminálu. Pro přehlednost není vypisována každá načtená hodnota, ale z hodnot načtených za jednu sekundu je vypočítán aritmetický průměr, který je vypsán do terminálu, ze kterého byl program spuštěn. Společně s informací o aktuální hodnotě senzorů je i vypsán počet prvků ve frontě obsahující hodnoty zvuku. Pokud by toto číslo bylo větší než 100, znamená to, že došlo k chybě a data nejsou ukládány do databáze. Celkový grafický výstup programu v terminálu je ukázán na obrázku 4).

Z tohoto důvodu jsou načtená data zapisována do dvou FIFO front. Jedna slouží pro komunikaci s vláknem writeSQL a druhá pro komunikaci s vláknem printAverage.

5.2 Rychlost načítání dat

Pro zjištění rychlosti načítání dat byl vytvořen jednoduchý skript, který vychází z vlákna getSoundLightValues. Ten obsahuje pouze import nezbytných knihoven a zapisuje načtená data do proměnných. Čas nutný na průběh jednoho cyklu je zaznamenán a společně s daty ze senzorů vypsán na terminál.

Tímto testem bylo zjištěno, že jedno načtení dat trvá okolo 14 ms. Po konzultaci s vedoucím práce je tato hodnota omezena na rychlost 10 vzorků za sekundu.



Obr. 4) Ukázka grafického výstupu programu pro načtení dat a jejich zápis do databáze

5.3 Importované balíky

GrovePi – jedná se o sadu funkcí sloužících ke komunikaci a načítání dat ze senzorů

Threading – umožňuje spouštět více vláken programu

Datetime – slouží k načtení aktuálního data, použito pro timestamping

Time – slouží k načtení aktuálního času, použito pro měření rychlosti vláken

Math – importuje pokročilé matematické funkce, použito pro kontrolu načtených dat z DHT senzoru

Queue – pro sdílení dat mezi vlákny je využito FIFO (first in – first out) zásobníku

Mysql.connector – zajišťuje komunikaci s MariaDB databází

databasesDB – obsahuje seznam databází, ke kterým je možné se přípojit

emailDB – slouží k zasílání emailových notifikací v případě vyvolání výjimky v důležité části programu

5.3.1 databasesDB

Jedná se o balík, který nastaví připojení s MariaDB serverem, který byl zvolen a vrátí connector, který je potom volán při každém vkládání zápisů do databáze.

Tento balík importuje mysql.connector a dále definuje třídu, která proměnným zadá odpovídající hodnotu podle zvolené databáze. Ta je zvolena funkcí, která jako argument přijímá číslo, které určuje pozici databáze v seznamu *databases*. Pro správné fungování je nutné, aby na nulté pozici v seznamu databází byla uvedena ta, která je nainstalována na platformě pro měření nebo jiná databáze, která je zamýšlena jako záložní.

5.3.2 emailDB

Důležité části programu běží v bloku Try/Except. V případě vyvolání výjimky je tato zachycena a společně s řetězcem popisujícím část programu, ve kterém byla výjimka zachycena jsou data poslána na libovolný email.

Kód využívá knihovnu *smtplib* a *ssl*. Jedná se o knihovny umožňující jednoduše poslat email pomocí kterékoliv služby nabízející SMPT protokol API. V tomto případě bylo využito emailového klienta Gmail.

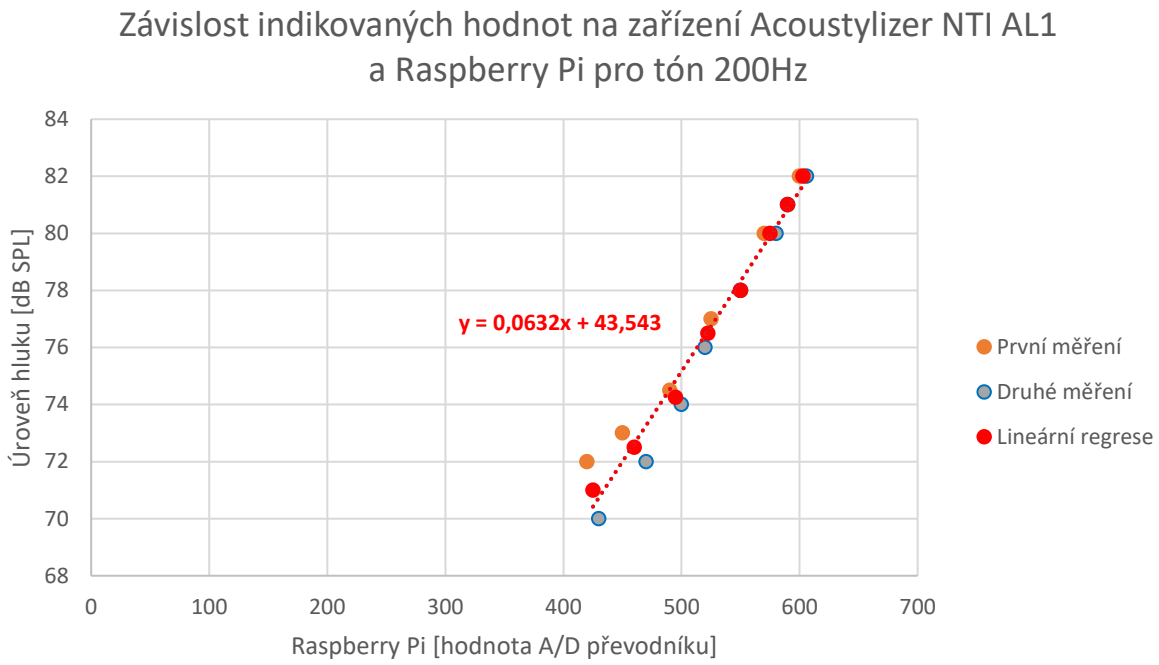
Volanou částí kódu je funkce, která přijímá dva argumenty. První argument slouží jako předmět emailu a slouží k identifikaci části kódu, ve kterém došlo k vyvolání výjimky. Druhý argument je text emailu, který obsahuje chybovou hlášku. První řádek emailu vždy tvoří časový údaj, kdy došlo k vyvolání výjimky. Emailem je též zaslána notifikace v případě úspěšného dokončení programu.

5.4 Převod dat

Jak už bylo zmíněno dříve, hodnoty úrovně světla a zvuku načtené z převodníku nemají žádnou vypovídající hodnotu. Proto je nutné je převést na běžně používané jednotky. V případě zvuku na jednotky akustického tlaku přepočteného na decibely (dB SPL) a v případě světla na luxy (lx).

5.4.1 Převod jednotek zvuku

Pro převod hodnot zvuku na běžné jednotky bylo využito experimentu, kdy z reproduktoru byl pouštěn nepřerušovaný tón o frekvenci 200 Hz, kdy byla různě měněna jeho intenzita a kdy úroveň hladiny zvuku byla současně měřena pomocí platformy Raspberry Pi a druhého zařízení (Acoustylizer NTI AL1), určeného k měření úrovně zvuku, které úroveň hlasitosti interpretuje v jednotkách akustického tlaku přepočteného na decibely. Data pro stejnou intenzitu z obou zařízení byla vynášena do tabulky a poté byla za pomocí programu Excel proložena spojnicí trendu, jejíž rovnice slouží v programu pro převod na výše zmíněné jednotky.



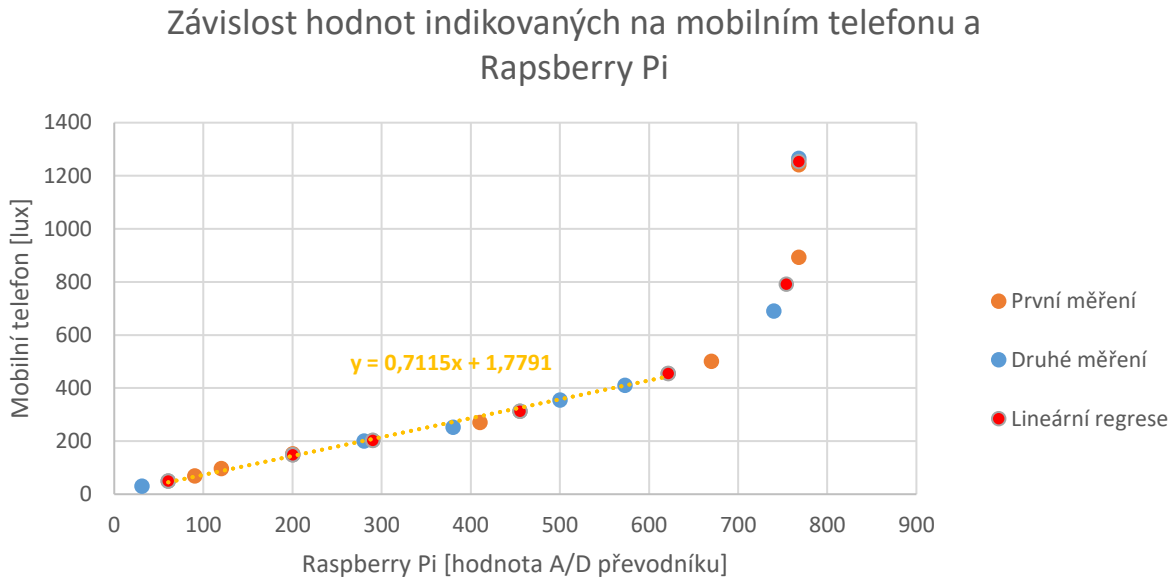
Obr. 5) Graf závislosti indikovaných hodnot na zařízení Acoustylizer NTI AL1 a Raspberry Pi pro tón 200Hz

Závislost mezi hodnotou zvuku indikovanou na Raspberry Pi a hodnotou zobrazovanou na zařízení pro měření úrovně zvuku je zjevně lineární.

5.4.2 Převod jednotek světla

Pro převod jednotek světla byl použit experiment založen na stejném principu jako experiment na převod jednotek zvuku s tím rozdílem, že na měření úrovně světla byl použit mobilní telefon s nainstalovanou vhodnou aplikací. Stejně jako u předchozího experimentu slouží v programu pro převod jednotek rovnice spojnice trendu.

Na rozdíl od experimentu měření zvuku, závislost není lineární. Po dosažení hodnoty blížící se 700 indikovanýh na Raspberry Pi při dalším zvyšování intenzity světla se nárust indikovaný na Raspberry Pi zpomaluje a zastaví se na maximální hodnotě 768. A/D převodník je ovšem 10bitový a měl by tudíž zobrazovat maximální hodnotu 1023. Toto mohlo být způsobeno například dosažením maximálního rozsahu fotodiody, nebo dosažením limitu operačního zesilovače.



Obr. 6) Graf závislosti hodnot indikovaných na mobilním telefonu a Rapsberry Pi

6 APLIKACE PRO ZOBRAZENÍ DAT V DATABÁZI A JEJICH VYKRESLENÍ DO GRAFU

Data z databáze je sice možné dostat pomocí programu pro správu databáze nebo příkazy v konzoli MariaDB, toto ale klade zvýšené nároky na znalosti uživatele. Pomocí zmíněného frameworku Kivy byla vytvořena aplikace pro základní zobrazení dat v databázi a jejich vykreslení do grafu.

Kivy framework je zdarma dostupný z internetových stránek vývojáře. Pro instalaci frameworku se předpokládá již existující instalace programovacího jazyku Python 3, protože samotná instalace frameworku probíhá jako instalace knihoven k tomuto jazyku. Nutno říct, že se bavíme o instalaci na zařízení, ze kterého jsou měřená data vyhodnocována a ne zaznamenávána. Framework Kivy tudíž nemusí být instalován na samotné platformě pro měření, kde by ostatně byl zbytečný pro deaktivované GUI.

6.1 Spuštění aplikace

Podobně jako aplikace pro načtení dat a jejich zápis do databáze musí být tato spuštěna přes terminál jako aplikace napsaná v jazyce Python 3. Samotný zdrojový kód se skládá ze dvou souborů.

První soubor s příponou .py je zdrojový kód napsaný v jazyce Python 3. Tento soubor je spouštěn. Importuje druhý soubor s příponou .kv a dále množství knihoven z Kivy frameworku, které obsahují různá tlačítka, layouty, grafy atd. Také obsahuje definice funkcí, metod a tříd používaných programem. Tyto slouží například k navázání spojení s databází, načtení hodnot z databáze a jejich vypsání do tabulky.

Druhý soubor s příponou .kv je tzv. Kivy file, který obsahuje pouze grafické části programu a definuje tak vzhled aplikace. Interaktivní prvky jsou namapovány na metody a funkce obsažené v prvním souboru.

6.2 Navigace

Po spuštění aplikace se zobrazí okno o velikosti 1200x500px. Velikost je možné standardním způsobem zvětšit či zmenšit. 20% levé části tvoří lišta ovládacích prvků – tlačítek. Zbytek tvoří tabulka pro zobrazení dat z databáze.

Stisknutím tlačítka **GO** ihned po spuštění je uživatel vyskakovacím oknem vyzván k výběru databáze, ze které budou data načteny. Opakovaným stiskem vypíše program posledních 100 záznamů z databáze.

Tlačítko **Show graph** otevře okno obsahující 4 grafy a 2 tlačítka. Tlačítko **Plot all** slouží k vykreslení grafu a tlačítko **Close** k uzavření okna s grafy. V případě, že uživatel nenačetl data do tabulky a pokusil se vykreslit prázdnou tabulku, je vyzván vyskakovacím oknem, aby data nejdříve načetl. Grafy se dynamicky přizpůsobují hodnotám v nich vykreslených. Osy grafu jsou popsány s tím, že popis osy x se rovněž dynamicky upravuje a nese informaci buď o počtu vykreslených vzorků nebo informaci o počtu vykreslených minut či hodin.

Tlačítko **Clear window** slouží pouze k vymazání načtených dat.

Pod tlačítkem **Settings** se ukrývá nastavení. Celkem obsahuje 3 karty. Po otevření okna s nastavením se první zobrazí karta **Basic settings**, kde může být změněn počet načtených dat

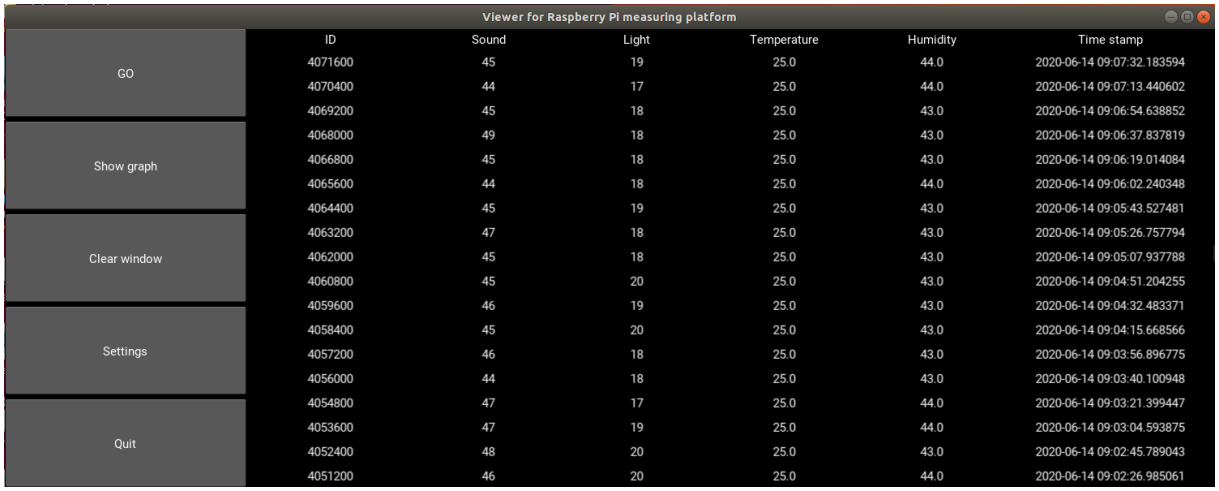
a způsob jejich řazení (sestupně či vzestupně). Změna v počtu vypsaných řádků musí být potvrzena tlačítkem **Apply.**

Karta **Every n-th record** v případě zaškrtnutí políčka přináší možnost vypsat do tabulky záznam pořízený každých x sekund (výchozí nastavená hodnota je 1 s). Tato funkce využívá informaci o rychlosti zápisu, že načítání dat je zpomaleno na 10 vzorků za sekundu. Tímto způsobem se dá s ohledem na limit vypsaných řádků (1000) zobrazit průběh hodnot načtených za první nebo poslední minutu, hodinu, den i více. Změna v textovém poli musí být opět potvrzena tlačítkem **Apply**.

Obě tlačítka **Apply** v nastavení mají vliv na potvrzení obou hodnot v textových polích. Pokud tedy uživatel změní hodnotu požadovaných vypsaných řádků a poté stiskne tlačítko **Apply** až u možnosti zobrazení každého ntého řádku, změna v obou polích se uloží.

Třetí a poslední karta se podobá vyskakovacímu oknu popsanému u tlačítka **GO** a slouží k změně načítané databáze.

Nastavení a jiná otevřená okna lze opustit buď tlačítkem escape na klávesnici, nebo kliknutím mimo okno.



Obr. 7) Ukázka grafického rozhraní s vypsanými naměřenými hodnotami

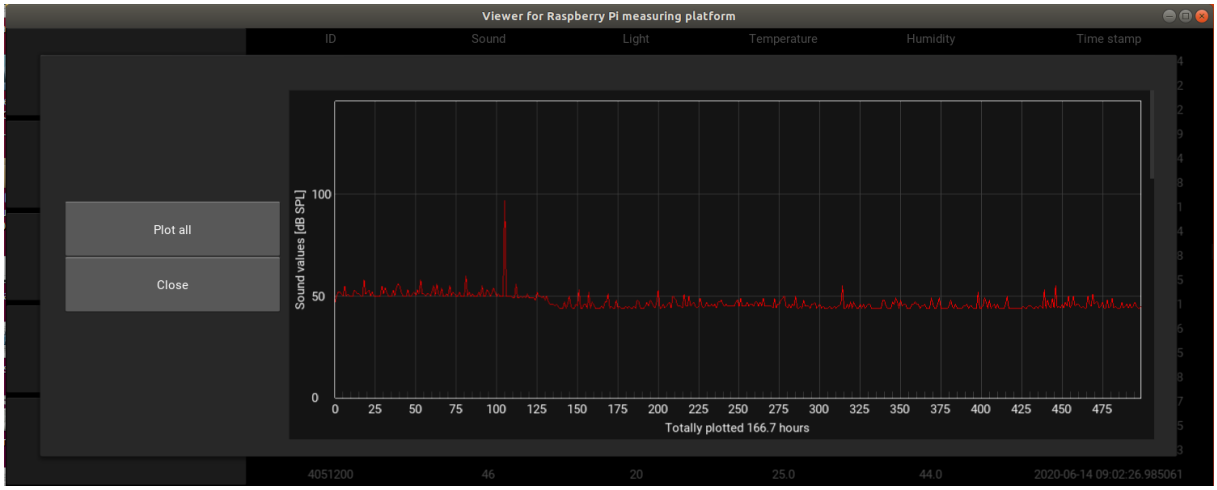
6.3 Omezení počtu zobrazených dat

Vypsání dat z databáze do tabulky v aplikaci probíhá iteračně. To má za následek, že vypsání 1000 záznamů v aplikaci trvá přibližně 15 sekund, během kterých nelze s aplikací nijak nakládat. Z tohoto důvodu je maximální počet vypsaných řádků v tabulce omezen na 1000.

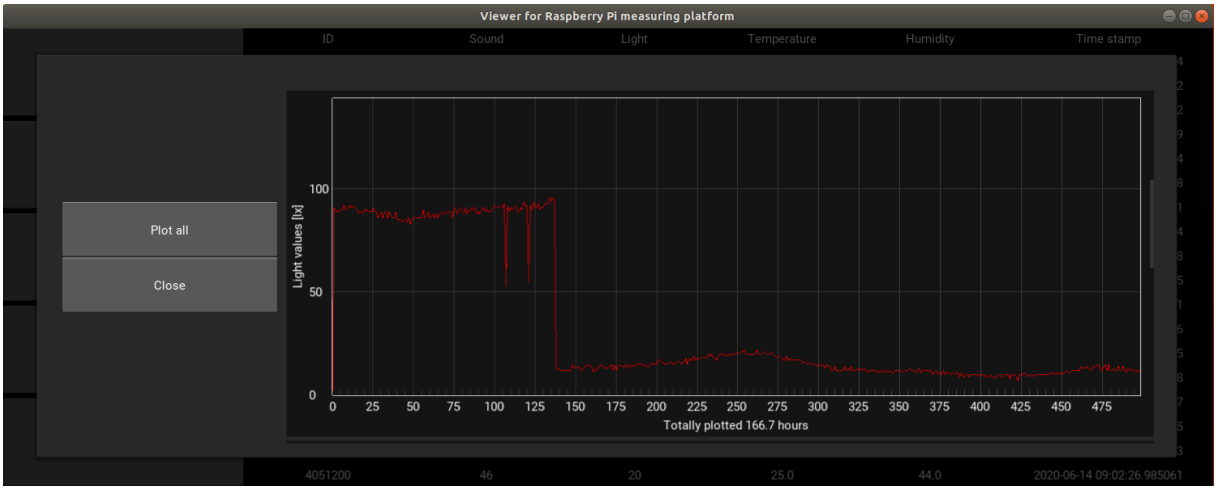
6.4 Testování a výstup

V rámci testování byla platforma pro měření spuštěna po dobu 5 dní, během kterých bylo do databáze uloženo 4 338 500 záznamů. Během této doby v programu pro měření nedošlo k vyvolání žádné výjimky.

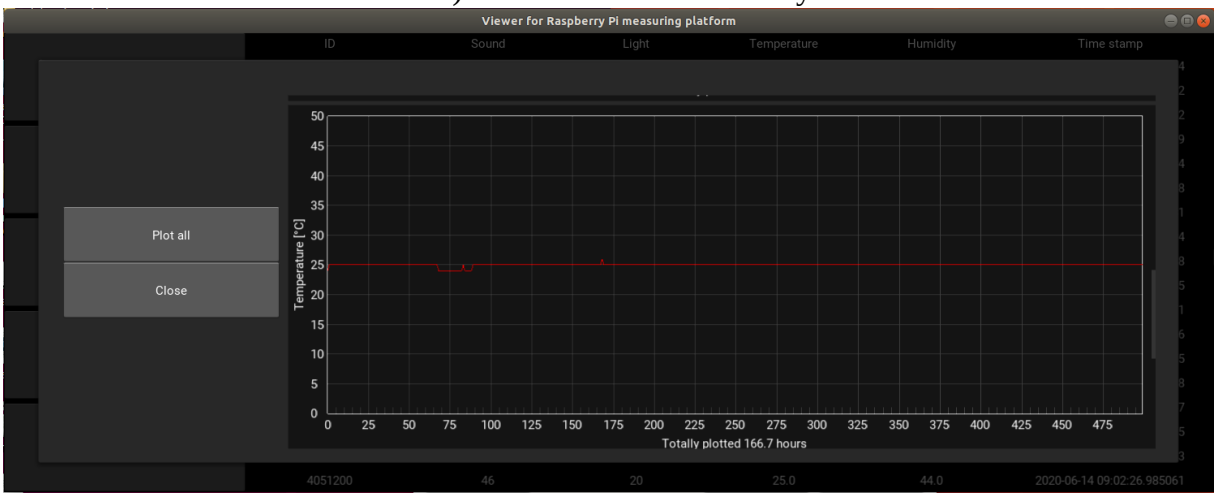
Grafický výstup z aplikace pro zobrazení dat v databázi v podobě vykreslených grafů vypadá následovně.



Obr. 8) Graf záznamu intenzity hluku



Obr. 9) Graf záznamu intenzity světla



Obr. 10) Graf záznamu teploty

7 ZHODNOCENÍ A DISKUZE

S ohledem na přesnost senzorů se Raspberry Pi pro měření, záznam a vyhodnocení dat osvědčilo. Díky jeho kompaktnosti a konektivitě si dokážu představit využití i v těžce přístupných prostorách. V takovém případě bych však doporučil dovybavit sestavu o PoE nástavbu, díky které bychom se zbavili nutnosti vést k platformě napájení v podobě zásuvky na 230V (k PoE je nutný i injektor, který napájí ethernetový kabel).

Pokud by platforma měla najít využití v praxi, bylo by nutné celou platformu pro měření osadit do vhodného šasi. V nabídce výrobce je pouze jednoduchá krabička složená ze dvou rovných desek a čtyř distančních sloupků. Pro použití v průmyslovém prostředí by bylo vhodnější vyrobit uzavřenou krabičku například pomocí 3D tisku. Stejným principem by bylo možné vyrobit i obal pro senzory. Rovněž kabeláž by bylo s největší pravděpodobností nutné pořídit delší. K setu jsou totiž dodávány kabely v délce pouhých 20 cm.

9 ZÁVĚR

V současné době se velký počet států potýká s nedostatkem IT expertů, což se pak projevuje na sice příznivém finančním ohodnocení lidí s touto profesí, ale zároveň vysokou cenou IT zakázek. Raspberry Pi je důležitý projekt v oblasti zpřístupnění IT široké veřejnosti a svou rolí tak jistě přispívá ke zvýšení zájmu mladých lidí o tento obor. Ne nadarmo má společnost stojící za jejím vznikem ve svém názvu *foundation*, tedy v českém překladu nadace.

Výhodou této platformy je bezesporu její cena a dostupnost. Důležitým předpokladem pro úspěch této platformy je fakt, že operační systém je založen na jádru Debianu, což je jedna z nejstarších distribucí Linuxu na světě. Pro Raspbian tedy existuje obrovské množství softwaru, který značně ulehčuje práci uživatelům platformy Raspberry Pi. Za velkou výhodu považuji, že celá platforma pro měření mohla být realizována velmi minimalisticky. Skládá se pouze z Raspberry Pi, základní desky GrovePi+ a zmiňovaných senzorů. Raspberry Pi se pak dokáže postarat kompletně o měření, záznam i vyhodnocení dat. Pokud bychom k měření dat chtěli použít například Arduino, ukládání a vyhodnocování dat by bylo o dost složitější. Arduino totiž není použitelné jako plnohodnotný stolní počítač a ukládání dat do databáze by tak znamenalo tvrdší oříšek. Přitom ale například zmíněné ukládání dat do databáze je důležitým předpokladem pro využití platformy pro měření v reálném prostředí. Data z databáze jsou totiž při správné konfiguraci databáze přístupná odkudkoliv s přístupem k internetu.

Avšak co lze považovat za největší výhodu Raspberry Pi, to je zároveň jeho kámen úrazu. Tato platforma zřídka najde uplatnění v oblasti komerčního zájmu. A když už ano, jedná se spíš o verzi zmíněného Compute modulu. V průmyslovém prostředí je totiž těžko představitelné, že by se výrobce spoléhal na systém běžící na SD kartě. Rovněž je těžko představitelné připojování nezbytného hardwaru přes USB kabel. Výrobci proto nejčastěji sáhnou po embedded systémech vyvíjených na míru konečného produktu. Jak se nakonec ukázalo, i přesnost dat získaných pomocí přiložených senzorů je diskutabilní. Především světelný senzor nedokáže pokrýt ani celkový rozsah denního světla. Jeho využití tak vidím maximálně v podobě jednoduché světelné závory.

Tomu se ale nelze divit, set Seeed Studio GrovePi+ stejně jako Raspberry Pi jsou určeny především pro edukativní účely a zde podle mě plní svou funkci výborně. Nakonec i já jsem na začátku psaní této práce měl s programováním minimální zkušenost, ale i tak jsem byl schopen vytvořit funkční platformu pro měření, zápis a vyhodnocení dat pomocí Raspberry Pi.

10 SEZNAM POUŽITÝCH ZDROJŮ

[1] SUMMERFIELD, Mark. Python 3: výukový kurz. Brno: Computer Press, 2010, s. 178. ISBN 978-80-251-2737-7.

[2] OPAVA, Zdeněk. Elektřina kolem nás. 2. dopl. vyd. Praha: Albatros, 1985, s. 203. ISBN 13-724-85 14/66.

[3] PRŮCHA, Jan. *Termoregulační zařízení s Peltieriho články* [online]. Brno, 2011 [cit. 2020-06-19]. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/5138/final-thesis.pdf?sequence=8&isAllowed=y. Bakalářská práce. Vysoké učení technické v Brně.

[4] HEATH, Nick. How the Raspberry Pi was created: A visual history of the $35 board. In: TechRepublic [online]. Louisville, Kentucky, 1999, 19 December 2018 [cit. 2020-06-19]. Dostupné z: https://www.techrepublic.com/pictures/how-the-raspberry-pi-was-created-a-visual-history-of-the-35-board/

[5] FROMAGET, Patrick. The awesome story of Raspberry Pi. In: RaspberryTips [online]. c2020 [cit. 2020-06-19]. Dostupné z: https://raspberrytips.com/raspberry-pi-history/

[6] CHANTHADAVONG, Aimee. Raspberry Pi sales jump: Here's why the tiny computer's in demand in coronavirus crisis. In: *ZDNet* [online]. CBS Interactive, c2020, 14 April 2020 [cit. 2020-06-19]. Dostupné z: https://www.zdnet.com/article/raspberry-pi-sales-jump-heres-why-the-tiny-computers-in-demand-in-coronavirus-crisis/

[7] KLOSOWSKI, Thorin. The Best Operating Systems for Your Raspberry Pi Projects. In: *Lifehacker* [online]. New York: G/O Media, 2005, 5.5. 2016 [cit. 2020-06-19]. Dostupné z: https://lifehacker.com/the-best-operating-systems-for-your-raspberry-pi-projec-1774669829

[8] OKOI, Martins D. 20 Best Operating Systems You Can Run on Raspberry Pi in 2020. In: *Fossmint* [online]. c2019, 12 May 2020 [cit. 2020-06-19]. Dostupné z: https://www.fossmint.com/operating-systems-for-raspberry-pi/

[9] ORPHANIDES, K.G. Raspberry Pi 4 review: finally ready to replace your desktop PC. In: *Wired* [online]. San Francisco: Condé Nast Britain, 1993, 24 June 2019 [cit. 2020-06-19]. Dostupné z: https://www.wired.co.uk/article/raspberry-pi-4-review-price-release

[10] HEATH, Nick. Raspberry Pi 4 Model B review: This board really can replace your PC. In: *TechRepublic* [online]. Louisville, Kentucky, 1999, 23 June 2019 [cit. 2020-06-19]. Dostupné z: https://www.techrepublic.com/article/raspberry-pi-4-model-b-review-this-board-really-can-replace-your-pc/

[11] AUFRANC, Jean-Luc. Raspberry Pi 4 vs Pi 3 – What are the differences? In: *CNX Software* [online]. Hong Kong, 2010, 24 June 2019 [cit. 2020-06-19]. Dostupné z: https://www.cnx-software.com/2019/06/24/raspberry-pi-4-vs-pi-3-what-are-the-differences/

[12] WATSON, J.A. Raspberry Pi computing cluster: What I'm using it for, and what I've added to it. In: *ZDNet* [online]. San Francisco: CBS Interactive, c2020, 24 July 2017 [cit. 2020-06-19]. Dostupné z: https://www.zdnet.com/article/raspberry-pi-computing-cluster-what-im-using-it-for-and-what-ive-added-to-it/

[13] BATE, Alex. Raspberry Pi clusters come of age. In: *Raspberry Pi* [online]. Cambridge: Raspberry Pi Foundation, 2009, 28 November 2017 [cit. 2020-06-19]. Dostupné z: https://www.raspberrypi.org/blog/raspberry-pi-clusters-come-of-age/

[14] THOMAS, Zoe. Coronavirus: Raspberry Pi-powered ventilator to be tested in Colombia. In: *BBC News* [online]. London: BBC Books, c2020, 13 April 2020 [cit. 2020-06-19]. Dostupné z: https://www.bbc.com/news/technology-52251286

[15] BÍLEK, Petr. Nemocnice v Kolumbii testuje plicní ventilátor řízený počítačem Raspberry Pi. In: *OTechnice* [online]. Praha: Mopax, 2017, 15.4. 2020 [cit. 2020-06-19]. Dostupné z: https://otechnice.cz/nemocnice-v-kolumbii-testuje-plicni-ventilator-rizeny-pocitacem-raspberry-pi/

[16] *Revolution Pi* [online]. Denkendorf: Kunbus, 2008 [cit. 2020-06-19]. Dostupné z: https://revolution.kunbus.com/

[17] Grove System. *Seeed Studio* [online]. Shenzhen: Seeed Technology, 2008 [cit. 2020-06-19]. Dostupné z: https://wiki.seeedstudio.com/Grove_System/

[18] GrovePi. In: UpWiki [online]. 2017, 3 April 2019 [cit. 2020-06-19]. Dostupné z: https://wiki.up-community.org/index.php?title=GrovePi&oldid=593

[19] GrovePi Port description. *Dexter Industries* [online]. Washington D.C., 2009 [cit. 2020-06-19]. Dostupné z: https://www.dexterindustries.com/GrovePi/engineering/port-description/

[20] USB3 vs. SATA Disk Performance Comparison. *Flexense - Data Management Software* [online]. 2007 [cit. 2020-06-19]. Dostupné z: https://www.flexense.com/usb3_vs_sata_disk_performance_comparison.html

[21] AOSONG. Temperature and humidity module - DHT11 Product Manual. [online katalogový list] In: *Aosong* [online]. c2003 [cit. 2020-06-19]. Dostupné z: https://drive.google.com/file/d/0B4B30jzMyzG8VFJ3QlozMnp6Y0E/view

[22] How DHT11 DHT22 Sensors Work & Interface With Arduino. *LastMinuteEngineers.com* [online]. c2020 [cit. 2020-06-19]. Dostupné z: https://lastminuteengineers.com/dht11-dht22-arduino-tutorial/

[23] Grove - Temperature&Humidity Sensor (DHT11). In: *Seeed Studio* [online]. Shenzhen: Seeed Technology Co., c2008 [cit. 2020-06-19]. Dostupné z: https://wiki.seeedstudio.com/Grove-TemperatureAndHumidity_Sensor/

[24] SAIVE, Ravi. 4 Useful Tips to Secure PhpMyAdmin Login Interface. In: *TecMint* [online]. Mumbai, 2012, 4 October 2016 [cit. 2020-06-19]. Dostupné z: https://www.tecmint.com/secure-phpmyadmin-centos-ubuntu/

[25] *Kivy framework* [online]. Kivy organization, 2011 [cit. 2020-06-19]. Dostupné z: https://kivy.org/#home

[26] RASMUSSEN, Brian. Are global variables bad? [příspěvek v diskuzním fóru]. In: *Stack Overflow* [online]. January 27 2009 [cit. 2020-06-19]. Dostupné z: https://stackoverflow.com/questions/484635/are-global-variables-bad

11 SEZNAM ZKRATEK, SYMBOLŮ, OBRÁZKŮ A TABULEK

11.1 Seznam zkratek

A		Ampér
dB SPL		Akustický tlak
DSI	Display Serial Interface	Sběrnice pro přenos obrazu
fps	Frames per second	Snímky za vteřinu
GB		Gigabajty
GPIO	General-purpose input/output	Programovatelné piny na Raspberry Pi
GUI	Graphical Users Interface	Uživatelské rozhraní
HDMI	High-Definition Multimedia Interface	Periferie pro přenos obrazového a zvukového signálu
Hz		Herz
IoT	Internet of Things	Internet věcí
lx		Lux
Mbps	Megabits per second	Megabity za sekundu
mm		milimetry
OS	Operating system	Operační systém
PoE	Power over Ethernet	Napájení pomocí ethernetového kabelu
px	pixel	
RH	Relative humidity	Relativní vlhkost
SATA	Serial ATA	Vysokorychlostní sběrnice
SD	Secure Digital	Paměťové médium
USB	Universal Serial Bus	Periferie pro připojení dalších zařízení
V		Volt
WiFi	Wireless Fidelity	Bezdrátové připojení k síti

11.2 Seznam tabulek

TAB 1) POROVNÁNÍ MODELŮ RASPBERRY PI 19

11.3 Seznam obrázků

OBR. 1) RASPBERRY PI 3 MODEL B+ POUŽIT PRO ÚČELY TÉTO PRÁCE18

OBR. 2) ZÁKLADNÍ DESKA GROVEPI+ 21

OBR. 3) SENZORY POUŽITY V TÉTO PRÁCI. Z LEVA TO JSOU: SENZOR ZVUKU, SENZOR SVĚTLA A KOMBINOVANÝ SENZOR TEPLOTY A VLHKOSTI VZDUCHU 23

OBR. 4) UKÁZKA GRAFICKÉHO VÝSTUPU PROGRAMU PRO NAČTENÍ DAT A JEJICH ZÁPIS DO DATABÁZE 27

OBR. 5) GRAF ZÁVISLOSTI INDIKOVANÝCH HODNOT NA ZAŘÍZENÍ ACOUSTYLIZER NTI AL1 A RASPBERRY PI PRO TÓN 200HZ 29

OBR. 6) GRAF ZÁVISLOSTI HODNOT INDIKOVANÝCH NA MOBILNÍM TELEFONU A RAPSBERRY PI.....30

OBR. 7) UKÁZKA GRAFICKÉHO ROZHRANÍ S VYPSANÝMI NAMĚŘENÝMI HODNOTAMI 32

OBR. 8) GRAF ZÁZNAMU INTENZITY HLUKU 33

OBR. 9) GRAF ZÁZNAMU INTENZITY SVĚTLA.....33

OBR. 10) GRAF ZÁZNAMU TEPLOTY.....33

12 SEZNAM PŘÍLOH

Příloha 1 – Program pro měření a zápis dat – pokorny_final_DB.py

Příloha 2 – Skript pro posílání emailových notifikací – emailDB.py

Příloha 3 – Skript se seznamem databází – databasesDB.py

Příloha 4 – Aplikace pro vyhodnocení dat – spouštěná část final_kivyDB.py

Příloha 5 – Aplikace pro vyhodnocení dat – importovaná část final_kivyDB.kv

PŘÍLOHY

Program pro měření a zápis dat – pokorny_final_DB.py

```
#!/usr/bin/env python3

import grovepi
import threading
import datetime
import time
import math
from queue import Queue
import mysql.connector
import databasesDB
import emailDB

sound_sensor = 1
light_sensor = 0
dht_sensor = 2
grovepi.pinMode(sound_sensor,"INPUT")
grovepi.pinMode(light_sensor,"INPUT")


class HowManyRows():
    def __init__(self):
        numberOfRows = input("\nHow many entries collect?\nInsert integer (min
                             . 100) otherwise is set to infinity\n\n")
        try:
            int(numberOfRows)
            if int(numberOfRows) < 100:
                raise ValueError
            self.numberOfRows = numberOfRows
            print("Number of rows set to ",self.numberOfRows)

        except ValueError:
            self.numberOfRows = True
            print("Number of rows set to INFINITY")

    def isSet(self):
        return self.numberOfRows

def connectMYSQL(mydb):
    try:
        cursor = mydb.cursor()

        if mydb.is_connected():
            print("DATABASE CONNECTED SUCCESFULLY")
        else:
```

```
            print("!!! DATABASE NOT CONNECTED !!!")

        return mydb, cursor

    except Exception as e:
        print("\nError while establishing MYSQL connection!\n\n",e)


def getSoundLightValues(condition):
    i=0

    try:
        while condition.isSet() == True or int(condition.isSet()) > i:
            i+=1
            start_time = time.time()

            lockA.acquire()
            sound_value = grovepi.analogRead(sound_sensor)
            light_value = grovepi.analogRead(light_sensor)
            lockA.release()

            sound_q.put(sound_value)
            light_q.put(light_value)
            averageSound_q.put(sound_value)
            averageLight_q.put(light_value)
            elapsed_time = time.time() - start_time
            try:
                time.sleep(0.1 - elapsed_time)
            except:
                pass

    except Exception as e:
        emailDB.sendMail("Exception in getSoundLightValues", e)

def getDHT():
    try:
        while True:
            start_time = time.time()

            global temperature, humidity

            lockA.acquire()
            [temperature,humidity] = grovepi.dht(dht_sensor,0)
            lockA.release()

            if math.isnan(temperature) == False and math.isnan(humidity) == Fa
                                            lse:
                pass
            else:
```

```
                continue
            elapsed_time = time.time() - start_time

            print(100*"*","\n")
            print(100*"+","\n")
            print(humidity, "% <-- Humidity || Temperature --
                  >",temperature," C", " || elapsed time -->, ",elapsed_time)
            print("\n",100*"+","\n")
            print(100*"*","\n")

            time.sleep(30)

    except Exception as e:
        emailDB.sendMail("Exception in getDHT", e)

def writeSQL(db,):
    time.sleep(0.5)
    try:
        def main_SQL(db):
            [mydb,cursor] = connectMYSQL(db)
            counterA = 0
            i = 0
            querryList = []
            try:
                while True:
                    if sound_q.empty() and i == 2:
                        flag = True
                        return flag
                    if sound_q.empty():
                        time.sleep(2)
                        i += 1
                        continue
                    i = 0
                    start_time = time.time()
                    row = [str(0.0632*sound_q.get()+43.543), str(0.7115*light_
                           q.get()+1.7791),str(temperature), str(humidity)]
                    querryList.append(row)
                    counterA += 1
                    if counterA >= 100:
                        cursor.executemany("INSERT INTO raspi_table (sound,lig
                                           ht,temperature,humidity) VALUES (%
                                           s,%s,%s,%s)", querryList)
                        mydb.commit()
                        querryList = []
                        elapsed_time = time.time() - start_time
                        print("Time for SQL to commit --
                              > ", round(elapsed_time, 6), "  ||  Sound Q len
                               --> ",len(sound_q.queue))
                        counterA = 0
```

```
            except Exception as e:
                try:
                    emailDB.sendMail("Exception in main_SQL querry",e)
                    print("!!! Exception in main_SQL\n...Connecting backup dat
                           abase...\n", e)
                    main_SQL(databasesDB.dbConnect(0))
                except Exception as e:
                    emailDB.sendMail("Triggering backup to file!", e)
                    print("...Couldn't connect backup database...\n", e)
                    raise Exception

        def backup_File():
            try:
                i = 0
                j = 0
                print("!!! WRITING INTO FILE !!!")
                backupFile = open("backupDB.txt","a+")
                backupFile.write("\nID|TIME|SOUND|LIGHT|TEMPERATURE|HUMIDITY\n
                                 ")

                while True:
                    if sound_q.empty() and j == 2:
                        break
                    if sound_q.empty():
                        time.sleep(1)
                        j += 1
                        continue
                    j = 0
                    i += 1
                    sound = sound_q.get()
                    light = light_q.get()
                    backupFile = open("backupDB.txt","a+")
                    backupFile.write("%d |%s|%d|%d|%d|%d\n" % (i,datetime.date
                                     time.now(),sound,light,temperature,humidi
                                     ty))
                    backupFile.close()
            except Exception as e:
                print("Writing into file unsuccsesfull!")
                emailDB.sendMail("!!! ERROR DURING BACKUP ENTRY !!!", e)

        main_SQL(db)


    except Exception as e:
        emailDB.sendMail("Exception in thread writeSQL...",e)
        print("!!! THREAD writeSQL exception\n",e)
        backup_File()
```

```
def printAverage():
    try:
        def main():
            start_time = time.time()
            averageSoundBuffer = []
            averageLightBuffer = []
            while True:
                current_time = time.time()
                averageSoundBuffer.append(averageSound_q.get())
                averageLightBuffer.append(averageLight_q.get())
                elapsed_time = current_time - start_time
                if elapsed_time > 1:
                    averageSound = round(sum(averageSoundBuffer)/len(averageSo
                                         undBuffer), 1)
                    averageLight = round(sum(averageLightBuffer)/len(averageLi
                                         ghtBuffer), 1)
                    print("Average sound Value --
                           > ",averageSound," || Average light value --
                           > ",averageLight, "  ||  Sound Q len --
                           > ",len(sound_q.queue))
                    break
        while True:
            main()

    except Exception as e:
        print("Exception in printAverage\n",e)



if __name__ == "__main__":
    try:
        lockA = threading.Lock()

        sound_q = Queue()
        light_q = Queue()
        averageSound_q = Queue()
        averageLight_q = Queue()

        while True:
            try:
                db =  databasesDB.dbConnect(int(input("\nSELECT DATABASE YOU W
                                                      ANT TO CONNECT\n0 -
                                                      - for RASPI local datab
                                                      ase\n1 -
                                                      - for MICROS remote dat
                                                      abase\n\n")))
                break
            except (ValueError, IndexError):
                print("\nInvalid input, try again... ")
```

```
            except Exception as e:
                print(e)

        condition = HowManyRows()

        t1 = threading.Thread(target=getSoundLightValues, args=(condition,))
        t2 = threading.Thread(target=getDHT, daemon=True)
        t3 = threading.Thread(target=writeSQL, args=(db,))
        t5 = threading.Thread(target=printAverage, daemon=True)

        t1.start()
        t2.start()
        t3.start()
        t5.start()

        t3.join()
        t1.join()
        print("SUCCESS")
        emailDB.sendMail("PROGRAM SUCCSESFULLY ENDED","(timestamp of program e
                         nd)")

    except Exception as e:
        emailDB.sendMail("Exception in MAIN METHOD!!!",e)
```

Skript pro posílání emailových notifikací – emailDB.py

```
import smtplib
import ssl
import datetime

port = 465
notice_mail = "exception.reporter.python@gmail.com"

def sendMail(subject,text):
    try:
        message = "Subject: %s\n\n%s\n%s" % (subject,datetime.datetime.now(),t
                                             ext)
        with smtplib.SMTP_SSL("smtp.gmail.com", port, context=ssl.create_defau
                              lt_context()) as server:
            server.login("exception.reporter.python@gmail.com", "Chytnichybu")
            server.sendmail("exception.reporter.python@gmail.com", notice_mail
                            , message)
            server.close()

    except Exception as e:
        print("Error during exception reporting!\n",e)
        pass
```

Skript se seznamem databází – databasesDB.py

```
import mysql.connector

class DBList:

    def __init__(self, host, user, passwd, database):
        self.host = host
        self.user = user
        self.passwd = passwd
        self.database = database

db0 = DBList("127.0.0.1","raspi","123456","raspi_database")
db1 = DBList("192.168.0.106","micros","123456","micros_database")
databases = [db0,db1]

def dbConnect(position):
    mydb = mysql.connector.connect(
    host = databases[position].host,
    user = databases[position].user,
    passwd = databases[position].passwd,
    database = databases[position].database
    )
    return mydb
```

Aplikace pro vyhodnocení dat – spouštěná část final_kivyDB.py

```
import kivy
import math
import mysql.connector
import kivydatabasesDB

from kivy.app import App
from kivy.lang import Builder
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.floatlayout import FloatLayout
from kivy.uix.gridlayout import GridLayout
from kivy.uix.anchorlayout import AnchorLayout
from kivy.uix.scrollview import ScrollView
from kivy.uix.label import Label
from kivy.uix.textinput import TextInput
from kivy.uix.button import Button
from kivy.uix.togglebutton import ToggleButton
from kivy.uix.behaviors import ToggleButtonBehavior, FocusBehavior
from kivy.uix.popup import Popup
from kivy.properties import ListProperty
from kivy.clock import Clock
from kivy.uix.recycleview.views import RecycleDataViewBehavior
from kivy.uix.recyclegridlayout import RecycleGridLayout
from kivy.core.window import Window
from kivy.uix.accordion import Accordion, AccordionItem
from kivy.uix.checkbox import CheckBox
from kivy.garden.graph import Graph, MeshLinePlot, MeshStemPlot, LinePlot, Smo
othLinePlot, ContourPlot
from kivy.uix.modalview import ModalView
from operator import itemgetter


kivy.require("1.11.1")
Builder.load_file('final_kivyDB.kv')
Window.size = (1200, 500)

class SoundGraph(Graph):
    def addPlot(self,*args):
        try:
            mainApp = App.get_running_app()
            result = mainApp.root.result
            self.plot = LinePlot(color=[1,0,0,1])
            self.plot.points = [(i, result.getRecords()[i][1]) for i in range(
                                0, result.getLen())]
            self.add_plot(self.plot)

        except:
            pass
```

```
class LightGraph(Graph):
    def addPlot(self, *args):
        try:
            mainApp = App.get_running_app()
            result = mainApp.root.result
            self.plot = LinePlot(color=[1,0,0,1])
            self.plot.points = [(i, result.getRecords()[i][2]) for i in range(
                                0, result.getLen())]
            self.add_plot(self.plot)

        except:
            pass

    def getSound(self, *args):
        pass

class TempGraph(Graph):
    def addPlot(self,*args):
        try:
            mainApp = App.get_running_app()
            result = mainApp.root.result
            self.plot = LinePlot(color=[1,0,0,1])
            self.plot.points = [(i, result.getRecords()[i][3]) for i in range(
                                0, result.getLen())]
            self.add_plot(self.plot)

        except:
            pass

class HumiGraph(Graph):
    def addPlot(self,*args):
        try:
            mainApp = App.get_running_app()
            result = mainApp.root.result
            self.plot = LinePlot(color=[1,0,0,1])
            self.plot.points = [(i, result.getRecords()[i][4]) for i in range(
                                0, result.getLen())]
            self.add_plot(self.plot)

        except:
            pass

class Records():
    def setRecords(self,record):
        self.records = record

    def getRecords(self, *args):
        return self.records
```

```
    def getLen(self, *args):
        try:
            self.records
            lenght = len(self.records)
            return lenght
        except AttributeError:
            pass

    def areEntries(self, *args):
        try:
            self.records
            return True
        except AttributeError:
            mainApp = App.get_running_app()
            mainApp.root.showPopup("You need to read entries first")
            pass



class DBApp_layout(BoxLayout):
    rows = ListProperty([])

    class SelectDBpopup(FloatLayout):
        pass

    class SettingsPopup(BoxLayout):
        pass

    class GraphWindow(BoxLayout):
        pass

    class WarningPopup(FloatLayout):
        pass

    class Connection():
        def __init__(self,db, cursor):
            self.db = db
            self.cursor = cursor

    class MYSQLquerry():
        def __init__(self,*args):
            self.asc_desc = "DESC"
            self.number = 100
            self.querryNumber = 0
            self.second = 10
            self.rowsProcessed = 0

        def asc(self,*args):
```

```
        self.asc_desc = "ASC"

    def desc(self, *args):
        self.asc_desc = "DESC"

    def changeNumber(self, cislo):
        try:
            if int(cislo) > 1000 or int(cislo) < 0:
                pass
            else:
                self.number = int(cislo)
        except ValueError:
            pass
    def changeSecond(self, second):
        try:
            if float(second) >= 1:
                self.second = round(float(second) * 10)
            elif float(second) < 1:
                mainApp = App.get_running_app()
                mainApp.root.showPopup("Input must be 1 second or bigger!\
                                       nIf you want to see every record,
                                       uncheck this option.")
                pass

        except ValueError:
            mainApp = App.get_running_app()
            mainApp.root.showPopup("Invalid input!")
            pass

    def isAscending(self, *args):
        if self.asc_desc == "ASC":
            return True
        else:
            return False

    def changeQuerry(self, *args):
        if self.querryNumber == 0:
            self.querryNumber = 1
        else:
            self.querryNumber = 0

    def querryText(self, *args):
        querry = "SELECT * FROM raspi_table ORDER BY id {} LIMIT 0,{}".for
                 mat(self.asc_desc,self.number)
        querry1 = "SELECT * FROM raspi_table WHERE id%{} = 0 ORDER BY id {
                   } LIMIT {}".format(self.second,self.asc_desc,self.numbe
                   r)
        if self.querryNumber == 0:
            return querry
```

```
            else:
                return querry1

    def getConnection(self,position):
        db = kivydatabasesDB.dbConnect(position)
        cursor = db.cursor()
        self.database = self.Connection(db,cursor)
        self.selectedDatabase = position


    def setQuerry(self, *args):
        try:
            self.querry
        except AttributeError:
            self.querry = self.MYSQLquerry()

    def initiateResult(self,*args):
        self.result = Records()
        self.selectedDatabase = None
        self.maxSound = 100
        self.maxLight = 500


    def test(self, *args):
        try:
            for radek in self.result:
                print(radek[1])
        except Exception as e:
            print(e)
            pass

    def connectDB(self, *args):
        while True:
            try:
                self.database.cursor.execute(self.querry.querryText())
                result = self.database.cursor.fetchall()
                self.result.setRecords(result)
                self.result.getRecords()
                self.database.db.commit()
            except AttributeError:
                self.showSelectDB()
                break
            for row in result:
                for col in row:
                    self.rows.append(col)
            self.maxSound = max(result, key=itemgetter(1))[1]
            self.maxLight = max(result, key=itemgetter(2))[2]
            break
```

```
    def clearRows(self, *args):
        self.rows.clear()
        for _ in range(6):
            self.rows.append("")
        self.rows.clear()
        self.result = Records()


    def showSelectDB(self,*args):
        show = self.SelectDBpopup()
        self.selectDBpopup = Popup(title="Database selection", content=show, s
                                   ize_hint=(None,None), size=(400,400))
        self.selectDBpopup.open()

    def showSettings(self, *args):
        show = self.SettingsPopup()
        self.settingsPopup = Popup(title="Settings", content=show, size_hint=(
                                    None,None), size=(800,400))
        self.settingsPopup.open()

    def showGraph(self, *args):
        show = self.GraphWindow()
        self.graphWindow = ModalView(size_hint=(.95, .9))
        self.graphWindow.add_widget(show)
        self.graphWindow.open()

    def showPopup(self,popupText):
        self.warningPopup = Popup(title="Warning", content=Label(text=popupTex
                                   t,font_size="15sp"), size_hint=(None,None),
                                   size=(400,200))
        self.warningPopup.open()

class DbApp(App):
    title = "Viewer for Raspberry Pi measuring platform"
    def build(self):
        self.layout = DBApp_layout()
        return self.layout

    def on_start(self, **kwargs):
        self.layout.setQuerry()
        self.layout.initiateResult()



if __name__ == "__main__":
    DbApp().run()
```

Aplikace pro vyhodnocení dat – importovaná část final_kivyDB.kv

```
#:kivy 1.11.1

<DBApp_layout>:
    id: dbApp

    BoxLayout:
        orientation:"vertical"
        spacing: 5
        size_hint: .2, 1
        Button:
            text: "GO"
            on_press: dbApp.clearRows()
            on_release: dbApp.connectDB()

        Button:
            text:"Show graph"
            on_press: app.root.showGraph()

        Button:
            text:"Clear window"
            on_press: dbApp.clearRows()

        Button:
            text: "Settings"
            on_press: dbApp.setQuerry()
            on_press: dbApp.showSettings()

        Button:
            text:"Quit"
            on_release: app.stop()


    BoxLayout:
        orientation: "vertical"
        size_hint: .8,1
```

```
GridLayout:
    id: firstRecycleRow
    size_hint: 1, None
    size_hint_y: None
    height: 25
    cols: 6
    cols_minimum: {0: 50, 1: 25,2: 25, 3: 25, 4: 25,5: 100}

    Label:
        text: "ID"

    Label:
        text: "Sound"

    Label:
        text: "Light"

    Label:
        text: "Temperature"

    Label:
        text: "Humidity"

    Label:
        text: "Time stamp"

RecycleView:
    viewclass: 'Label'
    data: [{'text': str(x)} for x in root.rows]
    RecycleGridLayout:
        cols: 6
        cols_minimum: firstRecycleRow.cols_minimum
        default_size: None, dp(30)
        default_size_hint: 1, None
        size_hint_y: None
        height: self.minimum_height
        orientation: 'vertical'
```

```
<SelectDBpopup>:
    Label:
        text: "Please select database"
        font_size: "25sp"
        size_hint: 0.6, 0.2
        pos_hint: {"x":0.2, "top":1}

    Button:
        text: "Raspi local"
        font_size: "20sp"
        size_hint: 0.8, 0.2
        pos_hint: {"x":0.1, "y":0.3}
        on_press: app.root.getConnection(0)
        on_release: app.root.selectDBpopup.dismiss()

    Button:
        text: "Micros remote"
        font_size: "20sp"
        size_hint: 0.8, 0.2
        pos_hint: {"x":0.1, "y":0.1}
        on_press: app.root.getConnection(1)
        on_release: app.root.selectDBpopup.dismiss()

<SettingsPopup>:
    id: settingspopup
    orientation: "vertical"
    spacing: 5

    Accordion:
        AccordionItem:
            title: "Database selection"
            FloatLayout:
                Label:
                    text: "Select database" if app.root.selectedDatabase == None else "Change
                          database"
```

```
            font_size: "25sp"
            size_hint: 0.6, 0.2
            pos_hint: {"x":0.2, "top":1}

        ToggleButton:
            text: "Raspi local"
            group: "databaseSelectionButtons"
            font_size: "20sp"
            size_hint: 0.8, 0.2
            pos_hint: {"x":0.1, "y":0.3}
            on_press: app.root.getConnection(0)
            state: "down" if app.root.selectedDatabase == 0 else "normal"



        ToggleButton:
            text: "Micros remote"
            group: "databaseSelectionButtons"
            font_size: "20sp"
            size_hint: 0.8, 0.2
            pos_hint: {"x":0.1, "y":0.1}
            #on_press: app.root.getConnection(1)
            state: "down" if app.root.selectedDatabase == 1 else "normal"
AccordionItem:
    title: "Every n-th record"
    BoxLayout:
        orientation: "vertical"
        FloatLayout:
            Label:
                text: "Display every n-th record"
                size_hint: 0.3, 0.3
                pos_hint: {"x":0.1, "y":0.15}
            CheckBox:
                id: nthCheckbox
                size_hint: 0.3, 0.3
                pos_hint: {"x":0.5, "y":0.15}
                active: True if app.root.querry.querryNumber == 1 else False
                on_active: app.root.querry.changeQuerry()
```

```
        FloatLayout:
            opacity: 1 if nthCheckbox.active else 0
            Label:
                text: "Display record taken every"
                size_hint: 0.3, 0.3
                pos_hint: {"x":0.111, "y":0.8}
            TextInput:
                id: nthRecord
                text: "{}".format(app.root.querry.second/10)
                size_hint: 0.15,0.3
                pos_hint: {"x":0.58, "y":0.8}
                halign: "center"
                padding_y: 15
                font_size: 17
                multiline: False
            Label:
                text: "second"
                size_hint: 0.3,0.3
                pos_hint: {"x":0.65, "y":0.8}
            Button:
                size_hint: 0.3, 0.3
                pos_hint: {"x":0.35, "y":0.1}
                text:"Apply"
                on_press: app.root.querry.changeSecond(nthRecord.text)
                on_press: app.root.querry.changeNumber(numberOfRows.text)

    AccordionItem:
        title: "Basic settings"
        FloatLayout:
            Label:
                text: "Set order by ID"
                size_hint: 0.217, 0.3
                pos_hint: {"x":0.1, "y":0.52}

            ToggleButton:
                id: ascending
                text: "Descending" if ascending.state == "normal" else "Ascending"
```

```
            size_hint: 0.3, 0.15
            pos_hint: {"x":0.6, "y":0.6}
            state: "normal" if app.root.querry.isAscending() == False else "down"
            on_press: app.root.querry.asc() if ascending.state == "down" else
                      app.root.querry.desc()
        Label:
            size_hint: 0.3, 0.3
            pos_hint: {"x":0.15, "y":0.32}
            text: "Number of samples (max. 1000)"

        TextInput:
            id: numberOfRows
            text: "{}".format(app.root.querry.number)
            size_hint: 0.3,0.15
            pos_hint: {"x":0.6, "y":0.4}
            halign: "center"
            padding_y: 15
            font_size: 17
            multiline: False

        Button:
            text: "Apply"
            size_hint: 0.3, 0.15
            pos_hint: {"x":0.35, "y":0.1}
            on_press: app.root.querry.changeNumber(numberOfRows.text)
            on_press: app.root.querry.changeSecond(nthRecord.text)


<GraphWindow>:
    orientation: "horizontal"
    size_hint: 0.95, 0.9
    spacing: 10
    AnchorLayout:
        size_hint: .2, 1
        BoxLayout:
            size_hint: 1, .3
            orientation:"vertical"
```

```
        Button:
            text: "Plot all"
            on_press: app.root.result.areEntries()
            on_release: soundGraph.addPlot()
            on_release: lightGraph.addPlot()
            on_release: tempGraph.addPlot()
            on_release: humiGraph.addPlot()

        Button:
            text:"Close"
            on_press: app.root.graphWindow.dismiss()

BoxLayout:
    id: boxik
    orientation: "vertical"
    size_hint: .8, .95

    ScrollView:
        do_scroll_x: False
        do_scroll_y: True
        bar_width: 5
        bar_color: [44,101,128,0.4]
        scroll_type: ['bars']
        size_hint: (1,1)
        size: (boxik.width, boxik.height)

        GridLayout:
            size: (boxik.width, 4*boxik.height)
            cols: 1
            spacing: 5
            size_hint_y: None

            SoundGraph:
                id: soundGraph
                xlabel: 'Totally plotted {} samples'.format(app.root.result.getLen()) if not
                        app.root.querry.querryNumber == 1 else 'Totally plotted {}
                        minutes'.format(round(app.root.result.getLen()*app.root.querry.second/
```

```
                60,1)) if app.root.result.getLen()*app.root.querry.second < 200 else
                'Totally plotted {}
                hours'.format(round((app.root.result.getLen()*app.root.querry.second)/3
                600,1))
        ylabel: 'Sound values [dB SPL]'
        x_ticks_minor: 5
        x_ticks_major: 25
        y_ticks_major: 50 if app.root.maxSound > 50 else 10
        y_grid_label: True
        x_grid_label: True
        padding: 5
        x_grid: True
        y_grid: True
        xmin: 0
        xmax: app.root.querry.number -1
        ymin: 0
        ymax: app.root.maxSound * 1.5
        stop: True
        background_color: [0, 0, 0, .5]

    LightGraph:
        id: lightGraph
        xlabel: soundGraph.xlabel
        ylabel: 'Light values [lx]'
        x_ticks_minor: 5
        x_ticks_major: 25
        y_ticks_major: 50 if app.root.maxLight > 50 else 10
        y_grid_label: True
        x_grid_label: True
        padding: 5
        x_grid: True
        y_grid: True
        xmin: 0
        xmax: app.root.querry.number -1
        ymin: 0
        ymax: app.root.maxLight * 1.5 if app.root.maxLight > 10 else 20
        stop: True
```

```
    background_color: [0, 0, 0, .5]

TempGraph:
    id: tempGraph
    xlabel: soundGraph.xlabel
    ylabel: 'Temperature [°C]'
    x_ticks_minor: 5
    x_ticks_major: 25
    y_ticks_major: 5
    y_grid_label: True
    x_grid_label: True
    padding: 5
    x_grid: True
    y_grid: True
    xmin: 0
    xmax: app.root.querry.number -1
    ymin: 0
    ymax: 50
    stop: True
    background_color: [0, 0, 0, .5]

HumiGraph:
    id: humiGraph
    xlabel: soundGraph.xlabel
    ylabel: 'Humidity [% RH]'
    x_ticks_minor: 5
    x_ticks_major: 25
    y_ticks_major: 10
    y_grid_label: True
    x_grid_label: True
    padding: 5
    x_grid: True
    y_grid: True
    xmin: 0
    xmax: app.root.querry.number -1
    ymin: 0
    ymax: 100
```

stop: True

background_color: [0, 0, 0, .5]