SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

EVIDENČNÉ ČÍSLO: FCHPT-123283-76933

Využitie Raspberry Pi na riadenie zásobníka kvapaliny

BAKALÁRSKA PRÁCA

Bratislava 2017 Daniel Boroš

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

EVIDENČNÉ ČÍSLO: FCHPT-123283-76933

Využitie Raspberry Pi na riadenie zásobníka kvapaliny

BAKALÁRSKA PRÁCA

Študijný program: automatizácia, informatizácia a manažment v chémii a potravinárstve
Číslo študijného odboru: 2621
Názov študijného odboru: 5.2.14 automatizácia, 5.2.52 priemyselné inžinierstvo
Školiace pracovisko: Oddelenie informatizácie a riadenia procesov - ÚIAM FCHPT
Vedúci záverečnej práce/školiteľ: Ing. Richard Valo, PhD

Bratislava 2017 Daniel Boroš

Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie

ZADANIE BAKALÁRSKEJ PRÁCE

Autor práce:	Daniel Boroš
Študijný program:	automatizácia, informatizácia a manažment v chémii a potravinárstve
Študijné odbory:	5.2.14. automatizácia, 5.2.52. priemyselné inžinierstvo
Evidenčné číslo:	FCHPT-123283-76933
ID študenta:	76933
Vedúci práce:	Ing. Richard Valo, PhD.
Miesto vypracovania:	Oddelenie informatizácie a riadenia procesov, FCHPT
Názov práce:	**Využitie Raspberry Pi na riadenie zásobníka kvapaliny**
Jazyk, v ktorom sa práca vypracuje:	slovenský jazyk
Špecifikácia zadania:	Linux, ako voľne šíriteľný operačný systém, pracuje na vývojovom modely Open Source. Umožňuje najviac možností prispôsobenia sa užívateľovi. Jeho využiteľnosť je vhodné implementovať nielen na profesionálne a osobné

účely, ale aj na edukačné. Miniatúrny PC Raspberry vzbudil pozornosť širokej verejnosti, že napriek vskutku miniatúrnym rozmerom a nízkej cene je schopný plniť úlohy plnohodnotného počítača. Myšlienkou bakalárskej práce je podrobne preskúmať možnosti spojenia OS Linux s Raspberry PI.

Cieľom práce je navrhnutie cenovo dostupného nástroja na inteligentné riadenie zásobníka kvapaliny (logické riadenie výšky hladiny) vlastnej výroby.

Úlohy:

- Preskúmať možnosti využitia Raspberry PI v oblasti riadenia a regulácie.
- Opis HW a SW realizácie zásobníka kvapaliny.
- Realizácia dvojpolohového riadenia výšky hladiny.
- Realizácia vykresľovania závislosti výšky hladiny od času.

Rozsah práce: 30

Dátum zadania: 13. 02. 2017

Dátum odovzdania: 14. 05. 2017

Daniel Boroš
študent

prof. Ing. Miroslav Fikar, DrSc.
vedúci pracoviska

prof. Ing. Miroslav Fikar, DrSc.
garant študijného programu

Abstrakt

Cieľom bakalárskej práce bolo navrhnúť, zostrojiť a riadiť zásobník kvapaliny. Práca je rozdelená do štyroch kapitol. Prvá kapitola Návrh, konštrukcia a riadenie zásobníka kvapaliny predstavuje jadro práce a je rozdelená do štyroch celkov. Prvý celok je venovaný špecifikáciám Raspberry Pi. V ďalšom celku sa charakterizujú vlastnosti a zapojenie elektrotechnických súčiastok. V treťom celku je realizovaný návrh zásobníka kvapaliny a posledný celok sa venuje popisu fungovania riadiaceho programu. V druhej kapitole s názvom Prechodové charakteristiky sa nachádzajú výsledky riadenia zostrojeného zásobníka kvapaliny.
V predposlednej kapitole s názvom Diskusia sú uvedené problémy, s ktorými sme sa stretli pri realizácii projektu. Poslednou kapitolou je Záver.

Kľúčové slova: Raspberry Pi, Zásobník kvapaliny, riadenie

Obsah

Zoznam obrázkov

Zoznam skratiek a značiek

SI	Systeme International
V	volt, základná jednotka napätia v sústave SI
A	ampér, základná jednotka prúdu v sústave SI
l	liter, jednotka objemu
Hz	hertz, odvodená jednotka frekvencie v sústave SI
G	predpona giga,10^9
m	meter, základná jednotka dĺžky v sústave SI
c	predpona centi, 10^{-2}
Wi-fi	Wireless Fidelity, označenie bezkáblového pripojenia
LAN	Local Area Network
HDMI	High Definition Multimedia Interface
GPIO	General Purpose Input/Output

Úvod

S automatizáciou sa stretávame každý deň. Je súčasťou našich životov a pomaly, ale isto sa dostávame do bodu, keď si bez nej život nebudeme vedieť predstaviť. V tejto rýchlej dobe, keď mnohým záleží na každej minúte, nám automatizácia poskytuje možnosti ako každý deň ušetriť trošku drahocenného času a stráviť ho zmysluplnejšie.

S príchodom Raspberry Pi máme možnosť relatívne jednoducho a ekonomicky automatizovať zdĺhavé a nezáživné práce doma či v záhrade. Polievanie trávnika alebo záhrady, ovládanie svetiel v domácnosti, prípadne realizácia jednoduchého poplašného systému. S Raspberry Pi je to všetko na dosah ruky.

Cieľom mojej bakalárskej práce je navrhnúť, zostrojiť a riadiť zásobník kvapaliny pomocou Raspberry Pi. Zásobník kvapaliny ako riadený proces, som si vybral pretože na jeho príklade sa dajú vhodne demonštrovať výhody a nevýhody Raspberry Pi.

1. Návrh, konštrukcia a riadenie zásobníka kvapaliny

1.1 Raspberry Pi 3

1.1.1 Všeobecné špecifikácie

Raspberry Pi 3 použitý v tomto projekte, bol mikropočítač s rozmermi dĺžka 8,5 cm, šírka 5,6 cm a výška 1.7 cm. Počítač obsahoval procesor rady ARM s taktovacou frekvenciou 1,2 GHz a štyrmi jadrami. Bol vybavený operačnou pamäťou veľkosti 1 Gb. Zariadenie malo vo výbave Bluetooth, sieťovú kartu s podporou wi-fi a LAN portom na pripojenie ethernetového kábla a čítačku pamäťových kariet, ktorá slúžila ako zdroj pevného disku, ktorý bol vo forme micro-SD karty. Ovládanie zariadenia bolo riešené cez 4 USB vstupy, z toho jeden bol použitý na myšku a jeden na zapojenie klávesnice. HDMI výstup sprostredkúval vysielanie obrazu do monitoru, ktorý mohlo predstavovať akékoľvek zariadenie s HDMI vstupom. Medzi ďalšie vstupy patril 3.5mm Jack vstup na pripojenie mikrofónu alebo reproduktoru, mikro-USB vstup na napájanie zariadenia a 40 GPIO portov, ktoré slúžili ako rozhranie medzi Raspberry Pi a okolitým svetom.

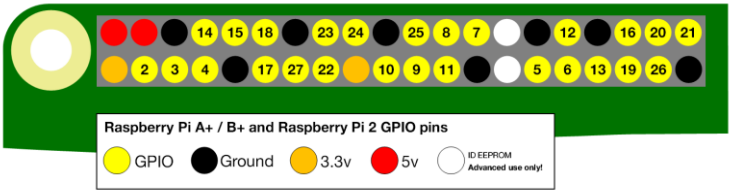
Obrázok 1: Pohľad z vrchu na Raspberry Pi 3

1.1.2 Operačný systém

Operačný systém Raspberry Pi sme si mohli vybrať akýkoľvek za predpokladu, že Raspberry Pi spĺňa základné požiadavky daného operačného systému. Na stránke výrobcu bolo uvedených niekoľko operačných systémov, ktoré boli vytvorené pre toto zariadenie. V tomto projekte fungovalo Raspberry Pi pod operačným systémom Raspbian, ktorý je oficiálnym systémom pre Raspberry Pi a je založený na linuxovom systéme Debian.
Raspbian mal po nainštalovaní mnoho aplikácii z oblasti vzdelávania, programovania a bežného užívania.

1.1.3 GPIO porty

GPIO alebo General purpose input/output porty predstavovali prepojenie Raspberry Pi s okolitým svetom. Dokázali sme ich prostredníctvom posielať a prijímať signály a poskytovať napájanie pre elektrotechnické súčiastky. Ovládať jednotlivé porty bolo možné cez programovacie jazyky ako Python, C a iné.



Obrázok 2: Rozdelenie GPIO portov na Raspberry Pi 3

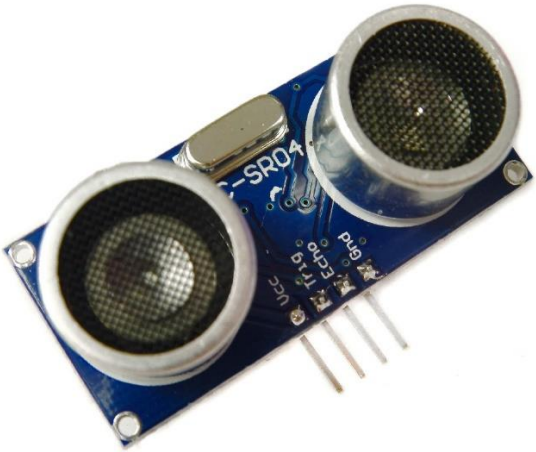
Raspberry Pi malo 26 portov s označením GPIO, ktoré slúžili na posielanie a prijímanie signálov. Všetkých 26 portov mohlo posielať alebo prijímať signál s maximálnym napätím 3.3 V.

Dva červené a dva oranžové porty slúžili na napájanie. Červené porty dokázali sprostredkovať prúd maximálnej veľkosti 1.5 A pri napätí 5 V. Oranžové porty dokázali zásobovať zariadenia prúdom maximálnej veľkosti 0.5 A pri napätí 3,3 V. Osem čiernych portov predstavovalo uzemnenie.

1.2. Elektrotechnické súčiastky

1.2.1 Ultrazvukový senzor HC-SR04

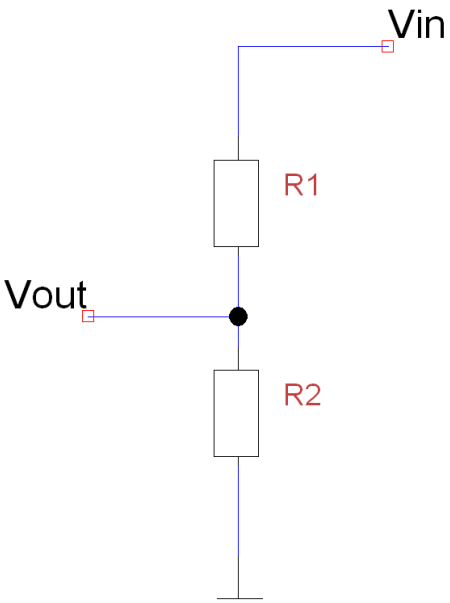
Ultrazvukový senzor HC-SR04 mal štyri porty: uzemnenie (Gnd), vstupný port (Trig), výstupný port(Echo) a 5V napájanie (Vcc). Dokázal merať vzdialenosti 2 až 400 cm. Frekvencia vyslaného zvukového signálu bola 40kHz. Senzor sme napájali prepojením červeného 5V portu na Raspberry a Vcc portu na senzore. Raspberry vysielalo zapínací signál z GPIO portu do Trig portu, čo spôsobilo vyslanie zvukovej vlny, ktorá sa odrazila od hladiny zásobníka kvapaliny a vrátila sa naspäť do senzoru. Senzor následne cez Echo port vyslal 5 V signál, ktorého trvanie sa rovnalo času potrebnému na to, aby sa zvuková vlny odrazila od hladiny zásobníka a vrátila do senzoru.



Obrázok 3: Ultrazvukový senzor HC-SR04

1.2.2 Zapojenie senzoru

Raspberry Pi malo maximálne napätie vstupného a výstupného signálu 3.3 V, zatiaľ čo napätie signálu z ultrazvukového senzoru bolo 5V. Keby sme posielali 5V signál do 3.3 V portu došlo by k poškodeniu zariadenia. Aby k tomu nedošlo použili sme obvod, ktorý rozdelil napätie. Napätie rozdeľujúci obvod sa skladal z dvoch rezistorov zapojených v sérii, ktoré znižovali napätie signálu na hodnoty, ktoré boli vhodné pre Raspberry Pi.



Obrázok 4: Schéma zapojenia rozdeľovača napätia

Vin označovalo napätie prichádzajúce z echo portu a má hodnotu 5V. Vout označovalo napätie, ktoré sme dodávali do Raspberry Pi a malo mať hodnotu maximálne 3,3V. Platia nasledovné rovnice:

$$Vout = Vin * \frac{R2}{(R1 + R2)}$$

Úpravou tohto vzťahu sme dostali:

$$\frac{Vout}{Vin} = \frac{R2}{R1 + R2}$$

Jeden odpor sme si zadefinovali ako 1200 Ohm, pretože taký rezistor sme mali k dispozícii a na základe tejto hodnoty sme vypočítali hodnotu R2.

$$\frac{Vout}{Vin} = \frac{R2}{R1 + R2}$$

Dosadili sme hodnoty napätí a odpor R1

$$\frac{3.33V}{5V} = \frac{R2}{1200\ Ohm + R2}$$

Prenásobili sme výrazom (1200 Ohm + R2)

$$0{,}66 * (1200\ Ohm + R2) = R2$$

Roznásobili sme zátvorku

$$792\ Ohm + 0{,}66R2 = R2$$

Odčítali sme od výrazu 0,66R2

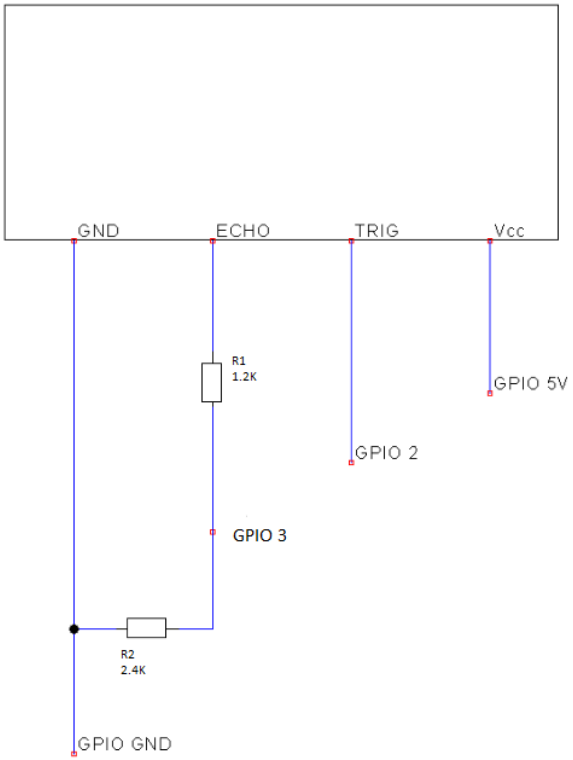
$$792\ Ohm = 0{,}34\ R2$$

Predelili sme výraz číslom 0,34

$$R2 = 2329\ Ohm$$

Odpor R2 použitý rozdeľovačom napätia mal hodnotu 2329 Ohm.

Obvod zapojenia senzora do Raspberry Pi vyzeral nasledovne:



Obrázok 5: Schéma zapojenia Ultrazvukového senzora HC-SR04

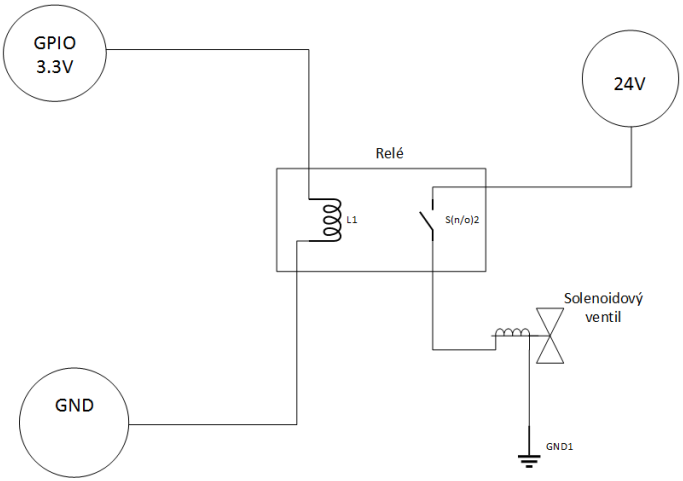
1.2.3 Ventil Irritrol Richdel 2400MT-M

Obrázok 6: Solenoidový ventil

Ventil značky Irritrol série Richdel model 2400MT-M (ďalej ventil) bol elektromagnetický ventil fungujúci na princípe elektromagnetického uzáveru prietoku kvapaliny. Skladal sa z jednoduchej cievky a jadra(ihli zabezpečujúcej prietok vody). Ventil fungoval pri napätí 24V a mal dva vodiče, kde jeden zabezpečoval jeho otváranie/zatváranie a druhý slúžil na uzemnenie.

1.2.4 Zapojenie solenoidového ventilu

Solenoidový ventil bol napájaný externe 24V, pretože Raspberry Pi nedokázalo privádzať prúd pri tomto napätí. Raspberry Pi ovládalo relé, ktoré v závislosti od toho, či je zapnuté alebo vypnuté, privádzalo prúd solenoidovému ventilu, čím ho otváralo alebo zatváralo.



Obrázok 7: Schéma zapojenia ventilu

1.3 Realizácia návrh zásobníka kvapaliny

1.3.1 Opis zásobníka

Obrázok 8: Zásobník kvapaliny pred poskladaním

Zásobník kvapaliny sme vytvorili zo 120 litrovej mierne kónickej nádoby. Na obrázku si okrem zásobníka môžete všimnúť aj sivú plastovú trubicu, ktorá po skrátení slúžila ako hniezdo pre ultrazvukový senzor.

Obrázok 9: Pohľad do vnútra zásobníka

V pohľade do vnútra zásobníka je vidieť potrubie, ktorým sme do zásobníka čerpali kvapalinu. Potrubie vedie z hornej časti zásobníka hadicou až k podstave zásobníka. Týmto umiestnením prívodného potrubia sme minimalizovali vlnenie povrchu hladiny a zlepšovali presnosť odčítania výšky hladiny senzorom. V jednej rovine s prívodným potrubím je aj vypúšťacie potrubie, ktoré si môžete lepšie prezrieť na obrázku číslo 10.

Obrázok 10: Prívodné a vypúšťacie potrubie

Na prívodné potrubie sme z vonkajšej strany napojili solenoidový ventil, do ktorého sme záhradnou hadicou privádzali vodu. Vypúšťací ventil umiestnený v spodnej časti zásobníka sme ovládali manuálne. Z boku je na zásobník pripevnený meter, ktorý slúži na kalibráciu a sledovanie úrovne hladiny.

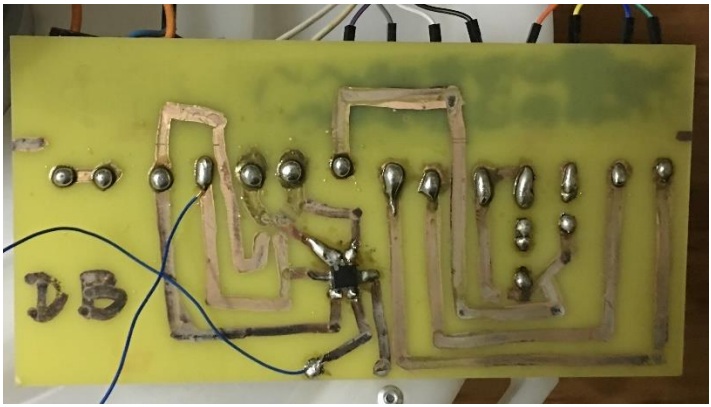
Obrázok 11: Hniezdo senzora

Senzor sme pripevnili do hornej časti trubice. Trubica má v dolnej časti vyvŕtané tri otvory, ktoré slúžia na prítok kvapaliny do trubice. Týmto usporiadaním sme dosiahli zmiernenie vlnenia hladiny a ochranu senzora pred vodou, pretože trubica vyčnieva nad maximálnu úroveň hladiny v zásobníku ako si môžete všimnúť na obrázku číslo 12.

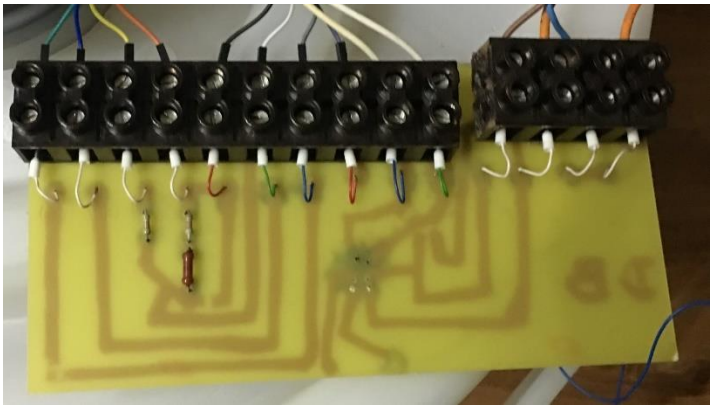
Obrázok 12: Zložený zásobník

Trubicu so senzorom sme pripevnili ku dnu, aby sme zabezpečili stálu polohu senzora a minimalizovali chybne namerané hodnoty. Pokrývku zásobníka sme prinitovali k zásobníku a vyrezali sme do nej otvor v oblasti prívodného potrubia, aby priliehala k zásobníku.

1.3.2 Realizácia zapojenia



Obrázok 13: Pohľad zdola na plošný spoj



Obrázok 14: Pohľad z vrchu na plošný spoj

Na zapojenie jednotlivých súčiastok sme si vytvorili plošný spoj. Na obrázku číslo 14 zelený, modrý, žltý a oranžový vodič smerujú z portov senzora Vcc, Trig, Echo a Gnd do plošného spoja.

Nasledovné štyri vodiče, v poradí čierny, biely, sivý a fialový slúžia na prepojenie Raspberry Pi so senzorom. Oranžový vodič(Gnd) vedúci zo senzora vstupuje do plošného spoja, kde sa napája na čierny vodič a vstupuje do uzemnenia na Raspberry Pi. Žltý vodič(Echo) po vstupe do plošného spoja prechádza deličom napätia, kde prúd s napätím 3.3 V sa odkloní a smeruje cez fialový vodič do Raspberry Pi na port číslo 3, ktorý je v programe definovaný ako ECHO. Zvyšok prúdu je odvádzaný čiernym vodičom na uzemňovací port na Raspberry Pi. Zelený a sivý vodič sú spolu prepojené a privádzajú prúd pri napätí 5 V z Raspberry Pi na senzor. Modrý a biely vodič sprostredkúvajú prepojenie medzi portom Raspberry Pi číslo 3 (biely), ktorý je v programe zadefinovaný ako trigger a Trig portom senzora. Posledné dva biele vodiče slúžia na ovládanie relé. Prvý je napojený na 4. port na Raspberry Pi a slúži na spínanie relé. Druhý predstavuje uzemnenie. Hnedým vodičom privádzame striedavý prúd určený na otváranie ventilu do relé. Oranžový vodič medzi hnedým a modrým slúži na prívod striedavého prúdu z relé do ventilu. Modrý a posledný oranžový vodič slúžia na uzemnenie.

1.4. Program

1.4.1 Použité knižnice a definovanie premenných

```
from gpiozero import LED
import RPi.GPIO as GPIO
import time
import numpy as np
import matplotlib.pyplot as plt
```

Pri programovaní zásobníka kvapaliny sme využili knižnice uvedené vyššie. Prvá z hore uvedených knižníc gpiozero a jej funkcia LED slúži na jednoduché zapínanie alebo vypínanie diódy. RPi.GPIO sa zaoberá ovládaním jednotlivých portov umiestnených na Raspberry Pi. Knižnica time ponúka mnoho funkcii, ktoré sa zaoberajú časom. Numpy je súbor funkcii na matematické operácie a matplotlib.pyplot zabezpečuje vykresľovanie grafov.

```
kal=73
```

Premenná kal slúži ako konštanta a informuje nás o výške zásobníka od tejto hodnoty odpočítavam aktuálne nameranú hodnotu vzdialenosti získanú senzorom. Získali sme ju pri testovacích meraniach na zásobníku kvapaliny. Cieľom bolo zvoliť takú hodnotu aby programom namerané hodnoty výšky hladiny boli približne súhlasné s hodnotami na metri pripevnenom k zásobníku.

```
limit=int(input("Vlož požadovanú hladinu: "))
nazov=input("vlož názov grafu: ")
```

Premennou limit zadávame požadovanú výšku hladiny v zásobníku. Premenná názov nám nastaví názov grafu pri ukladaní obrázku.

```
led=LED(4)
```

Príkazom LED(4) definujeme 4. pin na Raspberry Pi ako zdroj napätia pre diódu. V skutočnosti sme, ale týmto príkazom zabezpečili spínanie a vypínanie relé, ktoré následne zapína alebo vypína solenoidový ventil.

```
mod=input("ktorý mód chceš použiť?(S/U):")
```
Premennou mod si definujeme, ktorý pracovný režim chceme použiť.
S odkazuje na režim s širokým okolím, ktoré je určené ako 10 percent zo žiadanej hodnoty hladiny. U odkazuje na režim s úzkym okolím, ktoré má hodnotu plus mínus 1 cm. Oba režimy bližšie vysvetlíme v kapitole meranie a regulácia.

```
cas=[]
uroven=[]
caschyby=[]
urovenchyby=[]
```
Zoznamy cas, uroven, caschyby a urovenchyby sme využili neskôr pri vykresľovaní.

```
GPIO.setmode(GPIO.BCM)
TRIG = 2
ECHO = 3
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
```
GPIO.setmode(GPIO.BCM) nastavuje číslovanie pinov. TRIG a ECHO zaberajú piny 2 a 3 a pomocou GPIO.setup() ich nastavujeme na prijímanie(GPIO.IN) alebo odosielanie(GPIO.OUT) signálov.

1.4.2 Meranie, čistenie dát a regulácia

Hlavnú časť kódu sme realizovali cez štruktúru try-except-finally, ktorá sa používala na predchádzanie pádu programu kvôli chybe. Program pomocou try vyskúšal, či sa dá kód spustiť a ak narazil na chybu, tak sa spustili príkazy zapísané v bloku except. Danú štruktúru sme využili nasledovne. V try bloku prebehlo meranie senzora, kontrola dát a regulácia zásobníka. Po prerušení programu pomocou ctrl+c sa spustila chyba KeyboardInterrupt, ktorá prerušila program. Následne sa vykonali príkazy v časti except. Na záver sa spustili príkazy pod časťou finally, ktoré zabezpečovali zatvorenie ventilu a vyčistenie portov na Raspberry Pi.

```
try:
    for i in np.arange(0,7200,0.5):
        GPIO.output(TRIG, False)
        print("Prebieha meranie")
        time.sleep(0.5)
```

For cyklus sa opakoval 7200-krát, čo zodpovedalo približne jednej hodine. GPIO.output(TRIG, False) zabezpečoval, že senzor nevysielal žiadne zvukové vlny. Time.sleep(0.5) spôsobil pol sekundové čakanie programu. Nastavením for cyklu na periódu vzorkovania 0.5 a pol sekundovým čakaním programu sme nastavili periódu vzorkovania na pol sekundy. Program vypísal do terminálu „prebieha meranie“.

```
        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)
```

GPIO.output(TRIG, True) spôsobil vyslanie zvukových vĺn senzorom, následne program počkal 1 stotisícinu sekundy a vypol vysielanie signálu príkazom GPIO.output(TRIG, False).

```
        while GPIO.input(ECHO)==0:
          pulse_start = time.time()
        while GPIO.input(ECHO)==1:
           pulse_end = time.time()
```

Senzor nastavoval ECHO ako 1, na čas ekvivalentný vyslaniu a zaznamenaniu zvukovej vlny. Preto pomocou príkazu pulse_start = time.time() sme zaznamenali posledný čas, kedy bolo ECHO nastavené na 0 a následne príkazom pulse_end = time.time() sme zaznamenali posledný čas, kedy bolo ECHO nastavené na 1.

```
        pulse_duration = pulse_end - pulse_start
        draha = pulse_duration * 17150
        draha = round(draha, 2)
        hladina= round((kal-draha), 2)
        print("Úroveň hladiny je : %.2f cm"%(hladina))
```

Rozdielom časov pulse_end a pulse_start sme zistili dĺžku trvania pulzu vysielaného ECHOM. Tento čas sme vynásobili polovicou rýchlosti zvuku, 17150 v centimetroch za sekundu. Tým sme dostali dráhu medzi senzorom a hladinou kvapaliny, ktorú sme ešte zaokrúhlili na 2 desatinné miesta. Aby sme získali reálnu hodnotu hladiny kvapaliny v zásobníku, ktorú zaokrúhľujeme na dve desatinné miesta, tak bolo nutné vzdialenosť senzora od hladiny predstavovanú premennou draha odčítať od konštanty kal, ktorá predstavovala výšku zásobníka. Program vypísal aktuálnu výšku hladiny v zásobníku do terminálu.

```
    if i==0:
        pd=hladina
    if (hladina-pd)<-1.25 or (hladina-pd)>1.25:
        print("chybný udaj")
        caschyby.append(i)
        urovenchyby.append(hladina)
    else:
        pd=hladina
        cas.append(i)
        uroven.append(hladina)
```

Prvou podmienkou sme nastavili konštantu pd, ktorá predstavovala poslednú dobrú hodnotu. Konštanta pd sa aktualizovala novým údajom, ktorý prešiel čističom. Čistič dát kontroloval, či hodnota získaná senzorom nie je väčšia alebo menšia od poslednej dobrej hodnoty o viac ako 1.25 cm. Ak nová hodnota výšky hladiny prešla čističom, tak sme ju zaradili spolu s korešpondujúcim časom do zoznamov uroven a cas. Nová správna hodnota zároveň prepísala premennú pd a v ďalšom cykle s ňou program porovnával novú hodnotu. Ak nová úroveň výšky hladiny neprešla čističom, tak program vypísal do terminálu „chybný údaj“ a zaradil danú hodnotu aj s jej časom do zoznamov chýb urovenchyby a caschyby.

```
    if mod.upper()=="S":
        if pd<=(limit*0.9):
            led.on()
        elif pd>=(limit*1.1):
            led.off()
        else:
            print("Sme v okolí")
    elif mod.upper()=="U":
        if pd<=(limit-1):
            led.on()
        elif pd>=(limit+1):
            led.off()
        else:
            print("Sme v okolí")
    else:
        print("Vybral si nepodporovaný mód spusti program znova(R/P)")
```

V tejto časti kódu prebiehalo regulovanie výšky hladiny v zásobníku. Program najprv zistil, ktorý režim bol použitý. Ak sme nastavili premennú mod ako „s“ alebo „S“ tak spustil režim so širokým okolím žiadanej veličiny, ktoré predstavovalo 10% zo žiadanej veličiny. Ak sme premennú mod nastavili ako „u“ alebo „U“ tak spustil režim s úzkym okolím, ktoré predstavovalo 1 cm. Program vždy pracoval s hodnotami senzora uloženými v premennej pd. Oba režimy fungovali na rovnakom princípe. Keď bola hladina v zásobníku pod hodnotou dolného okolia požadovanej hodnoty, tak sa ventil otvoril a hladina sa začala zvyšovať. Ventil sa zatvoril, až keď hladina kvapaliny v zásobníku prekročila hornú hranicu okolia. Keď sa dostaneme do okolia, tak nás program informuje správou v termináli „Sme v okolí“. Ak by sme zadali režim, ktorý nie je podporovaný, tak nám to program oznámi a odporučí reštartovanie programu.

1.4.3 Vykresľovanie

Táto časť kódu mala za úlohu po prerušení programu pomocou klávesnice(ctrl+c) vykresliť graf závislosti úrovne hladiny od času.

```
c=np.array(cas)
u=np.array(uroven)
chc=np.array(caschyby)
chu=np.array(urovenchyby)
plt.plot(c,u)
plt.plot(chc,chu,"rx")
if mod.upper()=="S":
    plt.plot(c,np.ones(len(c))*limit*0.9,"r")
    plt.plot(c,np.ones(len(c))*limit*1.1,"r")
    plt.plot(c,np.ones(len(c))*limit,"g")
elif mod.upper()=="U":
    plt.plot(c,np.ones(len(c))*(limit-1),"r")
    plt.plot(c,np.ones(len(c))*(limit+1),"r")
    plt.plot(c,np.ones(len(c))*limit,"g")
plt.legend(["namerané údaje","chyby senzora","spodné
okolie","horné okolie","žiadaná hodnota"],loc="best")
plt.xlabel("Čas/[s]")
plt.ylabel("Uroveň hladiny/[cm]")
plt.title("Závislosť úrovne hladiny od času")
plt.savefig("%s.png"%(nazov),dpi=768)
plt.show()
```

Aby sme mohli s nameranými dátami narábať, bolo nutné ich umiestniť cez knižnicu numpy do vektorov. Vytvorili sme si štyri vektory, ktoré predstavujú dobre namerané dáta(c=cas,u=uroven) a dáta chýb(chc=caschyby,chu=urovenchyby). Dobre namerané dáta sme vykreslili pomocou knižnice matplotlib pod skratkou plt prednastavenou modrou farbou. Chyby boli vykreslené ako červené krížiky. Následne si program overil, ktorý režim bol použitý a zakreslil adekvátne okolie červenou farbou a požadovanú úroveň hladiny zelenou farbou. Príkazom plt.legend sme do grafu doplnili legendu a argumentom loc=“best“ sme ju umiestnili na najvhodnejšie miesto.Príkazmi plt.x/ylabel() sme nastavili názvy osí a

príkaz plt.title() pridal názov grafu. Plt.savefig(„názov“,“dpi=) uložil obrázok pod zadaným názvom a v zadanej kvalite a plt.show ukázal graf v terminály.

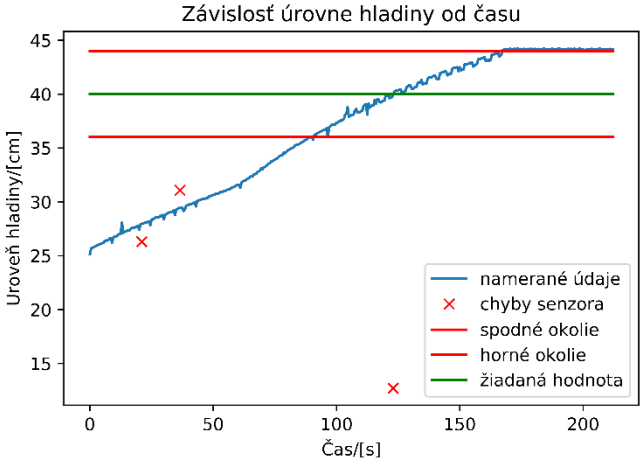
```
finally:
    led.off()
    GPIO.cleanup()
```

Na záver program zatvoril ventil a zresetoval porty použité v programe.

2. Prechodové charakteristiky

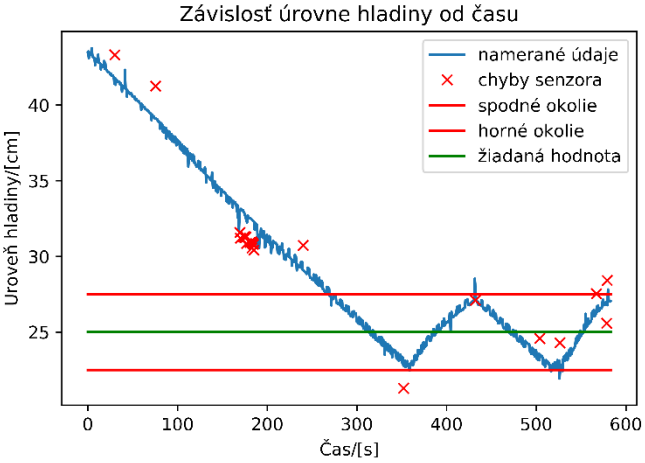
Po zostrojení zásobníka a vytvorení riadiaceho programu sme odsledovali ako zásobník reaguje na rôzne počiatočné podmienky. Nasledovné grafy závislosti úrovne hladiny od času, ukazujú výsledky našej práce.

2.1. Režim širokého okolia



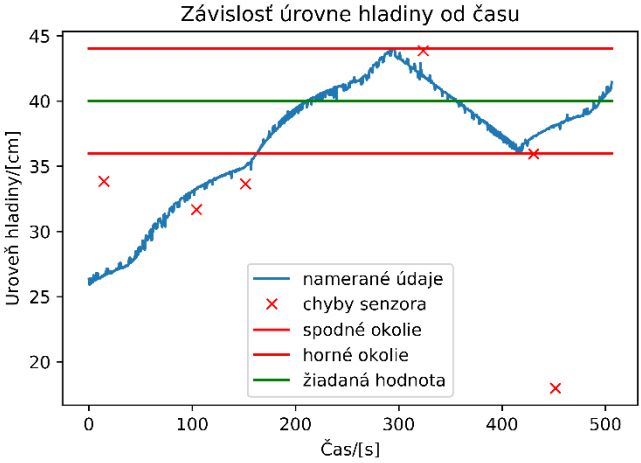
Obrázok 15: Široký režim bez vypúšťania, w=40cm

Obrázok číslo 15 predstavuje graf závislosti úrovne hladiny od času. Zásobník mal na začiatku úroveň hladiny približne 25 cm. Program mal požadovanú hodnotu úrovne hladiny nastavenú na 40 cm, preto sa otvoril ventil a hladina začala takmer lineárne stúpať. Proces by mal byť lineárny, ale vplyvom poklesu tlaku vo vodovodnom potrubí sa prietok ventilom mení. Pri tomto meraní sme zaznamenali iba tri chybne namerané údaje a dosiahli sme horné okolie požadovanej výšky hladiny.



Obrázok 16: Široký režim s vypúšťaním, w=25cm

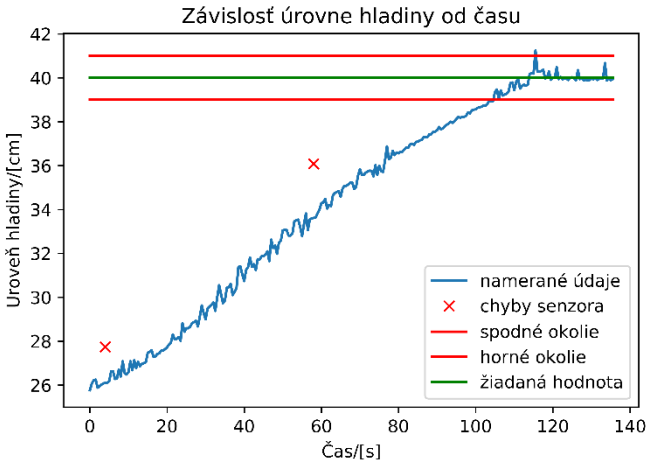
Obrázok číslo 16 ukazuje situáciu, kedy sme mali na začiatku vysokú úroveň hladiny okolo 44 cm, ale potrebovali sme úroveň hladiny v zásobníku udržiavať na nižšej hodnote, 25cm. Pri tomto meraní bol počet chýb omnoho vyšší. Napriek tomu sa programu podarilo systém uregulovať a výška hladiny sa držala v stanovených hraniciach.



Obrázok 17: Široký režim s vypúšťaním ,w=40cm

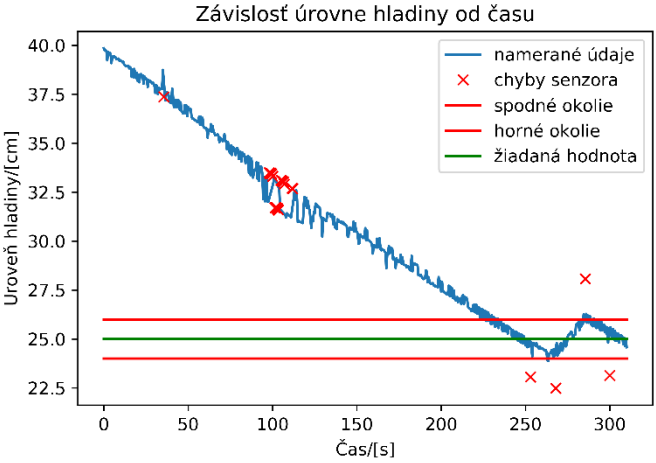
Obrázok 17 ukazuje situáciu, v ktorej sme potrebovali zvýšiť úroveň hladiny z 25 cm na 40 cm, ale súčasne dochádzalo aj k vypúšťaniu zásobníka. Namerané údaje neboli lineárne, čo bolo spôsobené poklesom a následným stúpnutím tlaku vody v prívodnom potrubí. Pri tomto meraní senzor nameral iba 6 chybných údajov a regulácia prebehla úspešne.

2.2. Režim úzkeho okolia



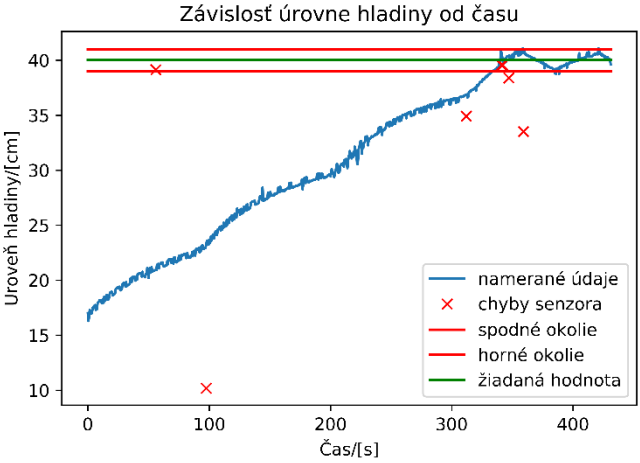
Obrázok 18: Úzky režim bez vypúšťania, w=40cm

Obrázok 18 ukazuje aplikáciu režimu úzkeho okolia s požadovanou hodnotou úrovne hladiny 40 cm bez vypúšťania. Týmto režimom sa reálna hodnota ustálila na úrovni 40 cm. Ustálenie výšky hladiny na požadovanej hodnote bolo spôsobené osciláciami v meraní senzora v okolí požadovanej hodnoty výšky hladiny. Výkyvy smerovali nahor, boli nižšie ako 1.25 cm, preto ich čistič dát spracoval ako správne údaje. Údaje prekonali hranicu horného okolia čo spôsobilo vypnutie prívody kvapaliny, čo v konečnom dôsledku viedlo k ustáleniu hladiny na žiadanej hodnote, namiesto na hornom okolí žiadanej hodnoty.



Obrázok 19: Úzky režim s vypúšťaním, w=25cm

Obrázok 19 opisuje rovnakú situáciu ako obrázok 16, ale aplikoval sa režim úzkeho okolia. Pri porovnaní so širokým režimom, úzky režim sa dostal bližšie k požadovanej hodnote, ale zapínanie a vypínanie ventilu prebieha s vyššou frekvenciou ako pri režime so širokým okolím. Senzor vykazoval chybné údaje najmä v okolí stej sekundy. Cieľ regulácie bol dosiahnutý napriek chybám senzora a úroveň hladiny sa udržala v stanovených hraniciach.



Obrázok 20: Úzky režim s vypúšťaním, w=40cm

Obrázok 20 ukazuje situáciu, v ktorej sme súčasne vypúšťali aj napúšťali zásobník za chodu úzkeho režimu. Na grafe sa prejavilo kolísanie tlaku v potrubí. Pri tomto meraní sa chyby merania senzora výraznejšie neprejavili a regulácia prebehla úspešne.

3. Diskusia

V našej práci sme sa zaoberali návrhom, realizáciou a riadením zásobníka kvapaliny pomocou dvojpolohového ventilu.

Najprv sme museli vybrať vhodný senzor na meranie výšky hladiny. Medzi možné merače výšky hladiny patria plavákový, kapacitný, odporový, ultrazvukový senzor a mnohé ďalšie. Rozhodli sme sa pre ultrazvukový senzor HC-SR04, z dôvodov nízkej ceny oproti iným alternatívam a veľkému rozsahu meraných hodnôt. Keďže senzor mal napájacie napätie 5V a Raspberry Pi síce dokáže napájať 5 V, ale nedokáže prijímať signáli vysielané pri danom napätí, tak bolo nutné implementovať delič napätia, ktorý spôsobil, že napätie prichádzajúceho signálu na Raspberry Pi bude rovné 3.3V. Pri tomto napätí prichádzajúceho signálu je Raspberry Pi schopné signál zaznamenať bez toho aby došlo k poruche zariadenia. Problému s napájaním senzora by sa dalo predchádzať aj použitím inej riadiacej jednotky, ktorá by dokázala spracovať signál pri danom napätí. Ako príklad môžeme uviesť systém Arduino, ktorý je programovateľný pomocou variácie programovacieho jazyka C a dokázal by napájať senzor 5 V napätím a aj prijímať signál pri danom napätí.

Pri výbere ventilu sme pôvodne chceli implementovať ventil so spojitým otváraním, ale tieto druhy ventilov neboli dostupné. Preto sme sa rozhodli prítok kvapaliny realizovať cez solenoidový ventil s napájaním 24 V, ktorý bol ľahko dostupný a cenovo prijateľný. Navrhli sme elektrický obvod, kde zapínanie a vypínanie ventilu závisí od spínania a vypínania relé, ktoré sme ovládali pomocou Raspberry Pi. Využili sme pri tom jednoduchú funkciu, ktorá pôvodne slúži na zapínanie a vypínanie diódy. Ventil bol napájaný z externého zdroja.

Pri konštrukcii zásobníka sme si ako základ zvolili mierne kónickú nádobu s objemom 120 l. Do spodnej časti nádoby sme vyvŕtali otvor, ktorý sme upravili na manuálne ovládateľné vypúšťacie potrubie. V hornej časti zásobníka sme vyvŕtali otvor, z ktorého sme vytvorili napúšťacie potrubie. Napúšťacie potrubie sme hadicou predĺžili až po podstavu zásobníka aby sme limitovali vznik vĺn na hladine. Senzor sme umiestnili do plastovej trubice zakotvenej v strede zásobníka, ktorá je vyššia ako zásobník, aby sme

ochránili senzor pred stykom s kvapalinou. Trubica zároveň slúži na ďalšie minimalizovanie vlnenia hladiny kvapaliny.

Návrh zapojenia sme realizovali na medenom plošnom, spoji do ktorého sme najprv ochrannou fixkou načrtli zapojenie jednotlivých elektrotechnických súčiastok a následne sme prebytočnú meď rozpustili roztokom chloridu železitého. Potom sme do vyleptaného plošného spoja osadili potrebné rezistory a relé. Výber vhodného relé pôsobil problémy, pretože sa muselo dať spínať pomocou 3.3V signálu a muselo zabezpečiť prechod prúdu o napätí 24 V.

Pri riadení procesu program navrhnutý cez Python fungoval bez problémov, ale senzor vykazoval miestami veľké oscilácie, až priam nereálne hodnoty výšky hladiny. Preto sme sa rozhodli implementovať čistič dát, ktorého hlavnou úlohou bolo odstrániť dáta, ktorých hodnoty boli nereálne. Sekundárnou úlohou čističa dát bolo zmierňovať oscilácie, čo sme dosiahli vhodným nastavením podmienok čističa.

4.Záver

Cieľom našej práce bolo navrhnúť, skonštruovať a riadiť zásobník kvapaliny.

Výsledkom našich meraní na zásobníku kvapaliny bolo zistenie, že aj napriek nevýhode, ktorú predstavoval dvojpolohový ventil, s ktorým bolo jemnejšie ladenie regulácie nemožné, je riadenie zásobníka realizovateľné s relatívne veľkou presnosťou, najmä pri použití režimu úzkeho okolia. Čím presnejšie by sme sa chceli priblížiť k žiadanej hodnote, tým viac by sa zvyšovala záťaž na ventil, vplyvom neustáleho zapínania a vypínania, najmä pri situácii kedy zo zásobníka súčasne vypúšťame aj do neho napúšťame kvapalinu. Preto sme implementovali aj režim širokého okolia, ktorý znižuje záťaž na ventil za cenu zníženej presnosti. Použitý čistič dát taktiež znižuje frekvenciu zapínania a vypínania ventilu vďaka tomu, že regulačný algoritmus používa len dáta, ktoré boli vyhodnotené ako reálne a správne.

Zoznam použitých zdrojov

1. Python, The Complete Manual, 2016, Imagine Publishing Ltd, ISBN 978-1785-462-689
2. The Python Book, 2015, Imagine Publishing Ltd, ISBN 9781785460609
3. Kurniawan A., Getting Started with Raspberry Pi 3, 2016