Pametna garažna vrata

Uvod

Opis problema

Problem današnjih garažnih vrat je, da jih je običajni možno odpreti samo na dva načina. S priloženim daljincem ali s tipko, običajno nameščeno na notranji strani garažnih vrat. Če želimo torej garana vrata odpretni moramo bitjt ali v garaži ali pa moramo imeti pri sebi daljinec. To pa je v vsakdanjem življenju nepraktično, sploh v primeru, ko pri hiši živi veliko ljudi, vsi pa rabijo dostop do garaže.

V tej nalogi bom predstavil svojo idejo, pametna garažna vrata, kot sem si jih zamislil in poskušal realizirati.

Cilji

Za raziskovalno nalogo, sem si postavil naslednje cilje: - Garažna vrata bo možno upravljati preko telefona - Garažna vrata bo možno upravljati preko spletne strani - Raspberry Pi bo spremljal ali je avto v garaži ali ne in glede na to samodejno zapiral garažna vrata - Raspberry Pi bo spremljal temperaturo v garaži in jih samodejno zaprl v primeru prenizke ali previsoke temperature - Raspberry Pi bo samodejno zaprl garažna vrata, če ostanejo odprta po določeni uri - Raspberry Pi nas bo preko potisnih obvestil obveščal o spremembi stanja garažnih vrat - Raspberry Pi bo beležil kdo in kdaj je aktiviral garažna vrata - uporabnik bo imel možnost preklicati samodejno zapiranje vrat

Izbira komponent

Ker ne potrebujem veliko precesorske moči, hkrati pa želim, da je moj projekt kar se da kompakten kot krmilnik izberem Raspberry Pi Zero W. To je najmanjša verzija Raspberry Pi-ja, z že vgrajenim WiFi-jem in Bluetoothom. Slednja bosta pri projektu najverjetneje potrebna.

Za upravljanje garažnih vrat bom uporabil 1-kanalni rele. Le tega bom sprogramiral tako, da se bo obnašal kot tipka tj. zaprl se bo za kratek časovni interval prib. 0.5s, nato pa se znova odprl. Nameščen bo v bližini že obstoječe tipke, ki se uporablja za upravljanje garažnih vrat. Z le to bo vzporedno vezan.

Za spremljanje stanja garažnih vrat bom uporabil reed stikala. In sicer dve stikali ter in magnet. Stikali bosta nameščeni na ogrodje vrat, medtem ko bo magnet nameščen neposredno na garažna vrata.

Za spremljanje temperature v garaži uporabim 1-Wire digitalni element.... (nevem imena zle). Le ta bo nameščen nekje v garaži, po možnosti meter od tal, na najmanj prepišnem mestu v garaži.

Ultrazvočni senzor, s katerim bom preverjal ali je avto v garaži ali ne, bo nameščen ali na stropbu garaže, najverjetneje pa kar na motorju garažnih vrat.

Ker želim, da bo mogoče v garaži preveriti ternutno temperaturo ter čas, bom uporabil tudi 16x2 LCD zaslon.

Poleg že naštetih komponent bo uporabil še dve LED diodi in dve tipki. Le te bodo paroma uporabljene kot indikator stanja avta oziroma temperature v garaži. Če bo naprimer garaža odprta in bo vanjo pripeljal avto, se bo pognal program, ki bo po določenem času samodejno zaprl vrata. Istočasno, bo začela utripati ustrezna LED dioda, uporabnik pa bo imel s pritiskom tipke možnost da prekliče samodejno zapiranje garaže. Pri temperaturi je namen LED diode in tipke enak, le da spremljamo temperaturo v garaži.

Priprava Raspberry Pi-ja

Da bom lahko uporabljal Raspberry Pi, moram najprej naložiti usterzen opreacijski sistem na Raspberry Pi. Ker za svoj projekt ne potrebujem grafičnega vmesnika, na Raspberry Pi namestim Raspbian Lite. To storim tako, da iz uradne strani Raspberry Pi prenesem Raspbian Stretch Lite. Nato sledim navodilom za namestitev operacijskega sistema na microSd kartico, ki jo nato vstavim v Raspberry Pi.

Ker do Raspberry Pi-ja že od samega začetka nimam dostopa preko tipkovnice, moram pred zagonom omogočiti še SSH ter vnesti podatke, ki jih Raspberry Pi potrebuje za povezavo na WiFi dostopno točko. Da

omogočim SSH, na boot particijo microSD katrice dodam datoteko ssh. Da pa se bo Raspberry Pi lahko povezal na WiFi dostopno točko, moram na boot particiji ustvariti datoteko wpa_supplicant.conf, v katero vnesem naslednje:

```
country=SI
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="imeDostopneTpočke"
    psk="geslo"
    key_mgmt=WPA-PSK
}
```

Nato microSD kartico vstavim v Raspberry Pi in priključim napajanje.

Nato se iz terminala na svojem računalniku preko SSH povežem na Raspberry Pi:

```
ssh pi@IP_RaspberryPi #uporabnik pi, privzeto geslo pa je raspberry
```

Ko je povezava vzpostavljena uporabim ukaz *passwd pi*, da spremenim geslo uporabnika pi.

Nato zaženem ukaz *sudo raspi-config*

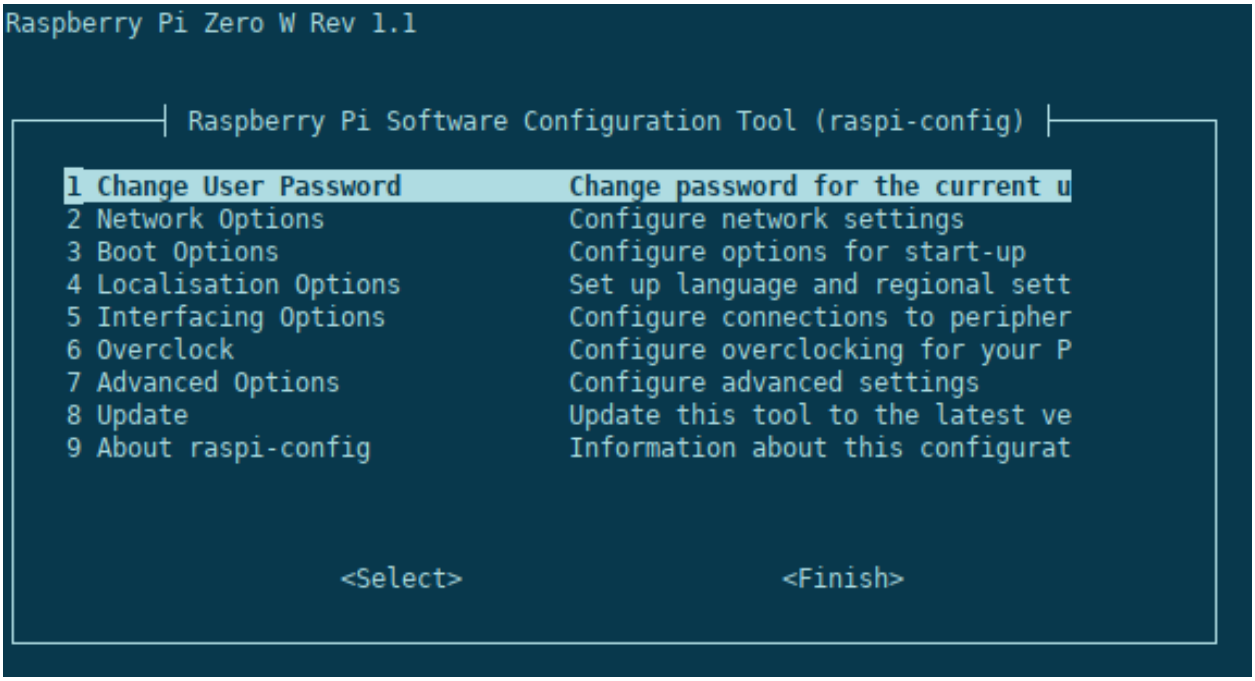

Figure 1: raspi-cofing

Pojavi se meni po katerem se premikam s pomočjo tipkovnice.

Tukaj nastavim vse potrebno:

```
- hostname spremenim iz "raspberrypi" v "projectPi"
- časovno območje nastavim na Europe/Ljubljana
- omogočim SPI, I2C in 1-Wire (vse to bom potreboval za priklop senzorjev)
- razširim datotečni sistem
```

Preden namestim kakršnekoli nove pakete, je potrebno sistem posodobiti:

```
sudo apt update && sudo apt upgrade -y
```

Ker bom program za upravljanje garažnih vrat napisal v Pythonu, le tega namestim na Raspberry Pi:

```
sudo apt install python python-pip
```

Ker pa bom uporabil tudi LCD, moram namestiti še RPILCD knjižnico:

```
sudo pip install RPLCD
```

S tem je priprava Raspberry Pi-ja zaključena.

Priključitev komponent

Rele

Sam priklop releja je zelo enostaven. Modul z relejem ima tri priključke: VCC, GND in IN1. VCC priključek povežem na 5V pin. GND priključim na GND pin. Za IN1 izberem BMC pin 16.

Reed stikali

Ultrazvočni senzor razdalje

DS18B20 temperaturni senzor

Ker je bilo to prvič, da sem v projektu uporabil DS18B20 senzor temperature, sem si pomagal z vodičem na PiMyLifeUp.

LCD zaslon

Tipki in LED diodi

poglej doma kako so priključene komponente

Programske rešitve

Premikanje garažnih vrat

Da lahko preverim ali rele pravilno deluje, napišem enostaven program, ki zapre rele in ga čez 0.5 sekunde odpre.

```
#!/usr/bin/python

#nadomestil toggleGarage.sh

import RPi.GPIO as GPIO #import the GPIO library
import time

GPIO.setmode(GPIO.BOARD) #nastavi način številčenja GPIO pinov
GPIO.setup(12, GPIO.OUT) #nastavi pin 12 kot izhodnega
GPIO.output(12, 0) #zapre rele
time.sleep(.5) #počaka 0.5 sekunde
GPIO.output(12, 1) #odpre rele

GPIO.cleanup() #počisti GPIO nastavitve
```

Program sem nato pognal preko SSH:

```
./toggleGarage.py
```

Preverjanje stanja garaže

Da lahko preverjam ali so garažna vrata odprta, priprta oziroma zaprta, moram dobiti trenutno stanje reed stikal. Tudi za to napišem program v Pythonu. >preveri, če program pravilno deluje (pin FALSE??)

```
#!/usr/bin/python

import RPi.GPIO as GPIO #import the GPIO library
import time

GPIO.setmode(GOPIO.BCM) #nastavi način števičenja GPIO pinov
GPIO.setup(5, GPIO.IN, pull_up_down=GPIO.PUD_UP) #nastavi pin 5 kot vhodni pin in omogoči notranji pull
GPIO.setup(6, GPIO.IN, pull_up_down=GPIO.PUD_UP) #nastavi pin 6 kot vhodni pin in omogoči notranji pull

if GPIO.input(5) == False:
    print("Door is open") # če je pin 5
elif GPIO.input(g) == False:
    print("Door is closed")
else:
    print("Door is ajar")
```

Spremljanje temperature

Izpisovanje časa in temperature na LCD

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from datetime import datetime
import ConfigParser
from RPLCD.gpio import CharLCD
import RPi.GPIO as GPIO
import argparse
import os
import time
import glob
import sys

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def readConf(section, vars, val_dict):
    configParser = ConfigParser.RawConfigParser(allow_no_value=True)
    configParser.read(os.environ['HOME']+'/.garage/garage.conf')
    for value in vars:
        val_dict[value] = int(configParser.get(section, value))

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines
```

```
def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = int(temp_string) / 1000.0 # TEMP_STRING IS THE SENSOR OUTPUT, MAKE SURE IT'S AN INTEGE
        temp_c = round(temp_c, 1) # ROUND THE RESULT TO 1 PLACE AFTER THE DECIMAL
        return temp_c

def lcd_write(line1, line2):
    lcd.cursor_pos = (0, 0)
    lcd.write_string(line1)
    lcd.cursor_pos = (1, 0)
    lcd.write_string(line2)

def init():
    #variables setup
    global lcd,base_dir,device_folder,device_file
    #read from config
    GPIO_VARS = ['TRIG','ECHO','RELAY','OVERRIDE_CAR','OVERRIDE_TEMP','LED_MONITOR_CAR','LED_MONITOR_TE
    TEMP_VARS = ['MAX_TEMP','MIN_TEMP']
    TIMEOUTS_VARS = ['AJAR_TIMEOUT','CAR_STATUS_TIMEOUT','BEGIN_TEMP_WATCH','AJAR_CLOSE_ATTEMPTS']
    LCD_VARS = ['cols','rows','pin_rs','pin_e','d4','d5','d6','d7']
    global GPIO_VARS_DICT, TEMP_VARS_DICT, TIMEOUTS_VARS_DICT, LCD_VARS_DICT
    TEMP_VARS_DICT = dict()
    TIMEOUTS_VARS_DICT = dict()
    GPIO_VARS_DICT = dict()
    LCD_VARS_DICT = dict()
    readConf('gpio',GPIO_VARS,GPIO_VARS_DICT)
    readConf('temperature',TEMP_VARS,TEMP_VARS_DICT)
    readConf('timeouts',TIMEOUTS_VARS,TIMEOUTS_VARS_DICT)
    readConf('lcd',LCD_VARS,LCD_VARS_DICT)
    #GPIO setup
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(GPIO_VARS_DICT['OVERRIDE_TEMP'],GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
    GPIO.setup(GPIO_VARS_DICT['LED_MONITOR_TEMP'], GPIO.OUT)   # Set LedPin's mode is output
    #LCD setup
    lcd = CharLCD(cols=LCD_VARS_DICT['cols'], rows=LCD_VARS_DICT['rows'], pin_rs=LCD_VARS_DICT['pin_rs']
    #temp sensor setup
    os.system('modprobe w1-gpio')
    os.system('modprobe w1-therm')
    base_dir = '/sys/bus/w1/devices/'
    device_folder = glob.glob(base_dir + '28*')[0]
    device_file = device_folder + '/w1_slave'
    #pushover setup
    configParser = ConfigParser.RawConfigParser(allow_no_value=True)
    configParser.read(os.environ['HOME']+'/.garage/garage.conf')

def destroy():
    GPIO.output(GPIO_VARS_DICT['LED_MONITOR_TEMP'], GPIO.LOW)   # led off
    GPIO.cleanup()
```

```
if __name__=="__main__":
    pid = str(os.getpid())
    pidfile = "/tmp/LCD_temp.pid"

    if os.path.isfile(pidfile):
        print "%s already exists, exiting" % pidfile
        sys.exit()
    file(pidfile, 'w').write(pid)
    try:
        init()
        lcd.clear()
        # Do some actual work here
        while True:
            lcd_write(datetime.now().strftime("%d.%m.%y  %H:%M"), "Temp: " + str(read_temp())+unichr(223
            time.sleep(1)
    except:
        os.unlink(pidfile)
        #os.delete(pidfile)
        destroy()
```

Samodejno zapiranje zaradi spremembe temperature

```
def monitorTemp():
    time.sleep(TIMEOUTS_VARS_DICT['BEGIN_TEMP_WATCH'])
    GPIO.add_event_detect(GPIO_VARS_DICT['OVERRIDE_TEMP'],GPIO.RISING,bouncetime=300)
    count = 0
    while 1:
        blink(GPIO_VARS_DICT['LED_MONITOR_TEMP'])
        if GPIO.event_detected(GPIO_VARS_DICT['OVERRIDE_TEMP']):
            break
        elif checkDoor() == 'zaprta':
            break
        temp = read_temp()
        if temp < TEMP_VARS_DICT['MIN_TEMP']:
            toggleGarage()
            pushover.send_message("Temperatura v garaži prenizka! Zapiram garažo!", title="Garaža prehla
            while checkDoor() != "zaprta":
                time.sleep(1)
            pushover.send_message("Garaža zaprta zaradi prenizke temperature!", title="Garaža zarta!")
            time.sleep(2)
            break;
        elif temp > TEMP_VARS_DICT['MAX_TEMP']:
            toggleGarage()
            pushover.send_message("Temperatura v garaži previsoka! Zapiram garažo!", title="Garaža pret
            while checkDoor() != "zaprta":
                time.sleep(1)
            pushover.send_message("Garaža zaprta zaradi previsoke temperature!", title="Garaža zarta!")
            time.sleep(2)
            break;
        count += 1
```