Mateusz Borowiec – Sterowanie wentylacją w zależności od temperatury w oparciu o Raspberry Pi Zero W v1.1

Wstęp

Raspberry Pi to minikomputer stworzony w celu wspomagania nauki informatyki. Pozwala na tworzenie różnorodnych projektów wykorzystujących elektronikę cyfrową. Składa się z pojedynczego obwodu drukowanego oraz różnych gniazd, których rodzaje i ilość zależne są od danego modelu.

W przypadku użytego przeze mnie modelu, tzn. Raspberry Pi Zero W specyfikacja jest następująca:

- Procesor jednordzeniowy, taktowanie 1 GHz
- 512MB RAMu
- Porty
    - Mini HDMI
    - Micro USB OTG
- Zasilanie poprzez osobny port Micro USB (zasilacz 5V/2A)
- 40 pinów GPIO
- Wyjście wideo CSI
- Czytnik kart Micro SD
- Połączenia sieciowe
    - Wi-Fi (2,4 GHz, 802.11n)
    - Bluetooth 4.0

Minikomputer działa na systemie operacyjnym Raspbian, bazującym na Debianie.

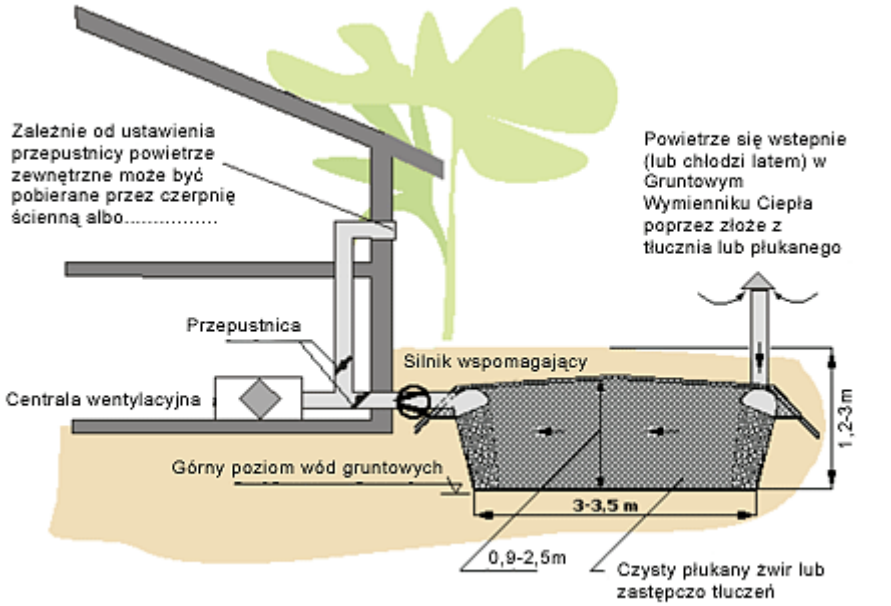
Opis projektu

Projekt zakłada wykorzystanie wraz z RPi, również dwóch termometrów DS18B20 i siłownika Siemens GSD 326.1A, oraz dodatkowych elementów elektronicznych, takich jak:

- Rezystor podciągający 4,7 kΩ
- Płytka rozszerzająca do pinów GPIO
- Dwukanałowy moduł przekaźników 2PH63091A
- Zasilacz Micro USB 5V/2.1A
- Przejściówkę Micro USB – USB 2.0
- Hub USB – do podłączenia klawiatury, myszki, oraz pamięci masowej, zawierającej napisane przeze mnie skrypty
- Przejściówkę Mini HDMI – VGA do przekazania obrazu na monitor
- Ekran dotykowy XPT2046 o przekątnej 3,5”
- Dodatkowe kable, piny i wtyki

Opis instalacji wentylacyjnej

W zależności od różnicy temperatur między termometrami siłownik ma za zadanie zmieniać źródło świeżego powietrza, wybierając pomiędzy powietrzem z zewnątrz, a gruntowym wymiennikiem ciepła. Instalacja wygląda następująco:

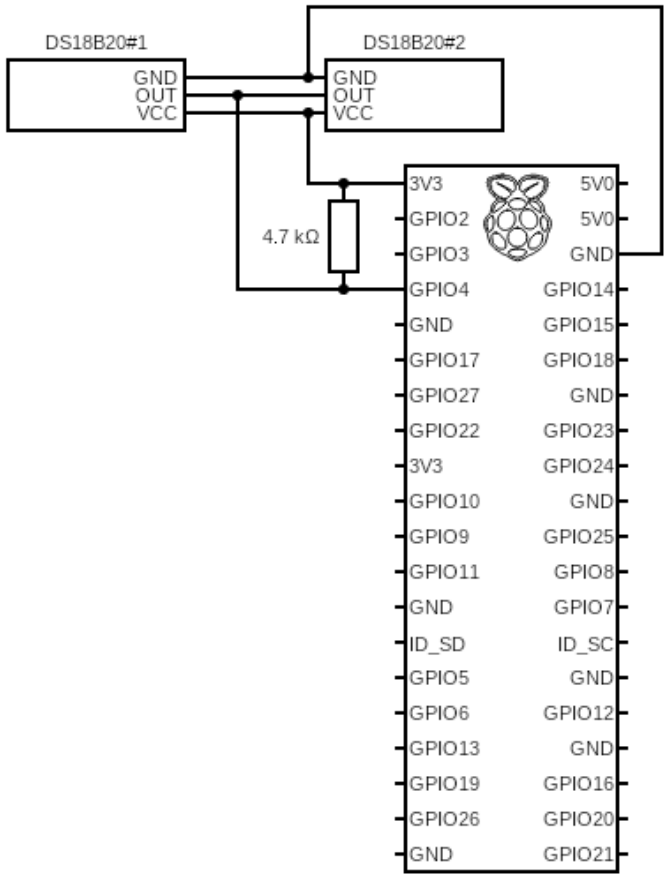


Jedyne co różni instalację którą projekt będzie sterował, od pokazanej na rysunku powyżej, to jedna przepustnica zamiast dwóch, będzie ona otwierała jeden kanał przepływu powietrza, równocześnie zamykając drugi.

Realizacja projektu

Początkowo do Raspberry podłączony został jeden termometr DS18B20, a następnie ekran dotykowy, jednak ze względu na jego wielkość rozwiązaniem ułatwiającym pracę zostało połączenie sieciowe poprzez VNC. Niestety znacząco spowolniło ono działanie mikrokomputera, w związku z czym zakupiłem odpowiednie przejściówki, umożliwiające podłączenie przez HDMI do monitora oraz pracę bezpośrednio na RPi. Następnie skonfigurowałem Domoticz oraz połączenie pomiędzy systemem, a czujnikiem z pomocą poradnika: https://forbot.pl/blog/kurs-raspberry-pi-projekty-domoticz-ds18b20-maile-id27526 .

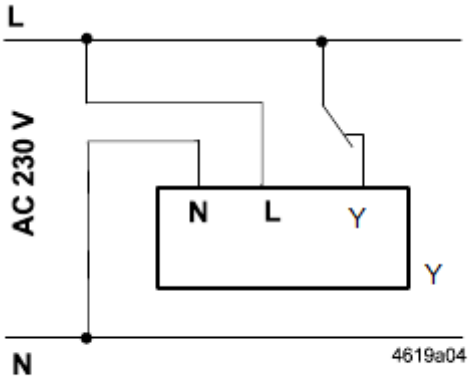
Fakt, że termometry można podłączać równolegle, postanowiłem wypróbować, czego rezultatem jest pokazane na schemacie połączenie z GPIO.



Następnym krokiem było podłączenie siłownika Siemens GSD 326.1A.

Wychodzą z niego dwa kable. Jeden z nich zawiera przewody zasilające oraz określający kierunek obrotu siłownika, z kolei drugi przesyła sygnały określające jego położenie.

Jako, że siłownik jest zasilany napięciem sieciowym, Raspberry steruje włącznikiem oraz kierunkiem obrotu poprzez moduł dwóch przekaźników, równocześnie odbierając sygnały określające położenie siłownika.
Wewnętrzny schemat zasilania prezentuje się następująco:

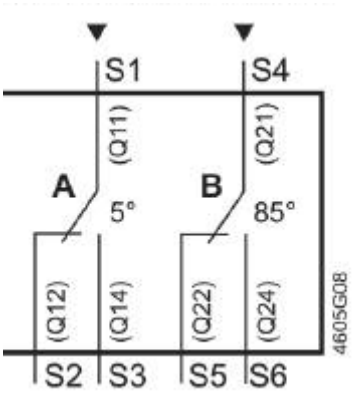


GSD32..1A

Przewody N i L są przewodami zasilającymi, z kolei przewód Y wskazuje kierunek obrotu:

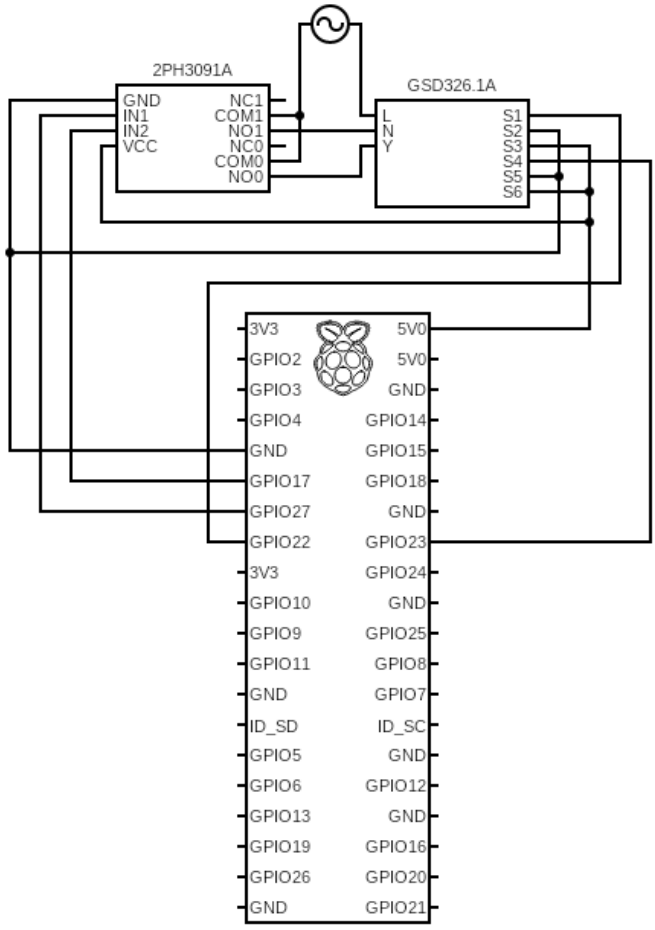
- rozłączony - siłownik obraca się zgodnie z ruchem wskazówek zegara
- podłączony - siłownik obraca się przeciwnie do ruchu wskazówek zegara

Siłownik posiada również drugi kabel, przesyłający informacje o jego położeniu. Zawiera on 5 żył, których role opisuje schemat poniżej:



Jak widać, są to dwa przekaźniki, które pokazują kąt rozwarcia siłownika.

Wyprowadzenia S2 i S5 zostały podłączone do uziemienia. S3 i S6 są podłączone do pinu 5V, z kolei S1 i S4 są podłączone do pinów GPIO. Schemat połączeń zamieszczono poniżej.



Sygnały przesyłane przez siłownik zależnie od kąta rozwarcia pokazuje poniższa tabela:

Kąt	S1	S4
5°	0	0
(5°, 85°)	1	0
85°	1	1

Dla potrzeb pokazu obsługi silnika i zmiany jego pozycji, napisałem skrypt w języku Python, wykorzystując bibliotekę RPi.GPIO. Listing programu widoczny jest poniżej.

```
import RPi.GPIO as GPIO

# setup variables
degrees_85_pin = 16  # 4'th GPIO pin in BCM mode
degrees_5_pin = 15  # Same as above
trigger_pin = 11  # Same as above
rotation_pin = 13  # Same as above

# setup pins
GPIO.setmode(GPIO.BOARD)
GPIO.setup(degrees_5_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(degrees_85_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(trigger_pin, GPIO.OUT)
GPIO.setup(rotation_pin, GPIO.OUT)

# set rotation
rotate_clockwise = True
GPIO.output(rotation_pin, rotate_clockwise)

# start engine
engine_off = False
GPIO.output(trigger_pin, engine_off)


def change_direction():
    if GPIO.input(degrees_85_pin) == GPIO.input(degrees_5_pin):
        if GPIO.input(degrees_85_pin) == GPIO.LOW:
            rotate_clockwise = False
        elif GPIO.input(degrees_85_pin) == GPIO.HIGH:
            rotate_clockwise = True
        GPIO.output(rotation_pin, rotate_clockwise)

# event listener
GPIO.add_event_detect(degrees_85_pin, GPIO.FALLING, callback=lambda x:
change_direction(), bouncetime=300)
GPIO.add_event_detect(degrees_5_pin, GPIO.RISING, callback=lambda x:
change_direction(), bouncetime=300)

try:
    while True: pass
except:
    GPIO.remove_event_detect(degrees_5_pin)
    GPIO.remove_event_detect(degrees_85_pin)
    GPIO.output(rotation_pin, False)
    GPIO.output(trigger_pin, False)
    GPIO.wait_for_edge(degrees_5_pin, GPIO.RISING)
    GPIO.output(rotation_pin, True)
    GPIO.output(trigger_pin, True)
    GPIO.cleanup((degrees_5_pin, degrees_85_pin, trigger_pin,
rotation_pin))
```

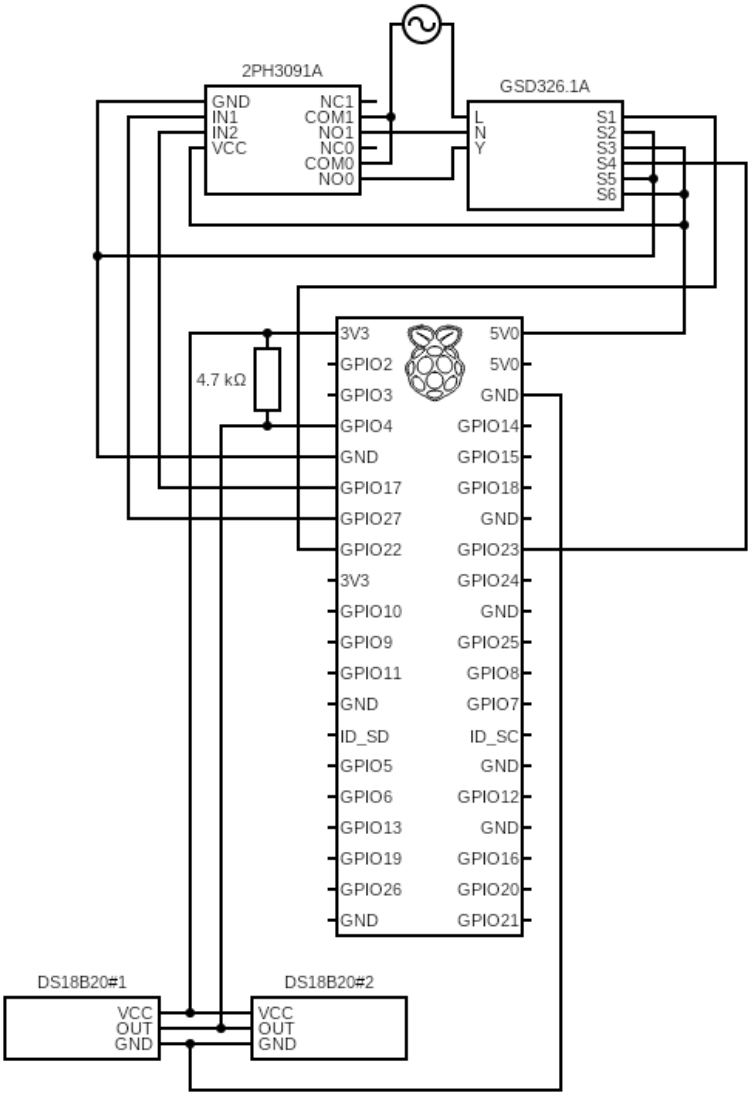
Na początku programu następuje wybranie trybu numeracji pinów dla biblioteki pomiędzy BCM a rzeczywistym rozmieszczeniem pinów na Raspberry.

Następnie następuje przypisanie roli danych pinów, oraz w przypadku wejść jest użyty programowy rezystor podciągający. Później następuje ustalenie kierunku obrotu, oraz uruchomienie silnika.

Następnie dodane czujniki zdarzeń nasłuchują zmian stanów wejść, i odpowiednio na nie reagują za pomocą funkcji change_direction(). Program wykonywany jest w nieskończoność, dopóki nie zostanie wywołane przerwanie za pomocą kombinacji klawiszy CTRL+C.

Wywołany wyjątek usuwa nasłuch z wejść, ustala kierunek obrotu na przeciwny do ruchu wskazówek zegara oraz włącza silnik. Po wykryciu maksymalnego skrętu w lewo, wszystkie wyjścia zostają wyłączone, oraz usunięte zostają wszelkie przypisania wykonane przez program.

Finalny schemat wszystkich podłączeń prezentuje się następująco:



Podsumowanie i wyniki

Na chwilę obecną udało się nawiązać połączenie pomiędzy dwoma termometrami a Raspberry Pi poprzez system Domoticz. Udało się również uruchomić silnik oraz sterować kierunkiem jego obrotu bazując na danych o jego pozycji poprzez skrypt napisany w języku Python.

W planach jest połączenie odczytu temperatury ze sterowaniem silnikiem. Temperatura będzie mierzona w dziennych odstępach. Zmiana pozycji przepustnicy będzie uruchamiana z opóźnieniem, gdy po kilku odczytach średnia różnica temperatur nadal będzie przesłanką do zmiany źródła powietrza w wentylacji. Projekt będzie kontynuowany.