Wrocław, 17 października 2023

Przemysł 4.0

Laboratorium 2

Prosty system pomiarowy

prowadzący: *Dr inż. Radosław Idzikowski*

1 Wprowadzenie

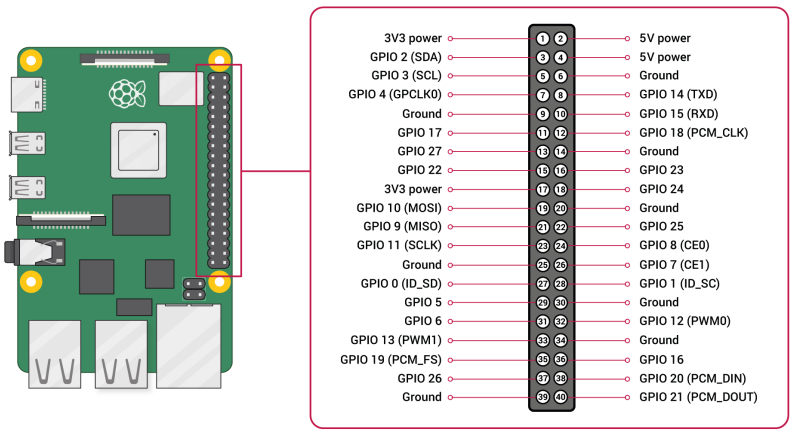
Celem laboratorium jest zapoznanie się budową prostego systemu pomiarowego z wykorzystaniem mikrokontrolera z rodziny Raspberry Pi w wersji 4b. W ramach zajęć będą do złożenia różne układy, które będą elementami składowymi prostego systemu pomiarowego. Praca będzie oceniana na bieżąco w trakcie zajęć wraz z postem podłączania i testowania układów. Po ukończeniu każdego układu należy zawołać prowadzącego w celu zaakceptowania etapu.

2 Zadania

W ramach zajęć należy w zespołach wykonać następujące zadania:

1. Zapoznanie się z generowaniem sygnału `PWM`.
2. Podłączenie i obsługa czujnika temperatury.
3. Rozbudowanie układu o wyświetlacz `LCD`.

Zadanie nr 1 nie jest na ocenę, ale jest obowiązkowe do zaliczenia zajęć. Za wykonanie zadania nr 2 jest ocena dostateczna (3.0). Za zrealizowania w pełni (z wykorzystaniem sygnału `PWM`) zadania nr 3 jest maksymalnie +2 do oceny. W celu uzyskania oceny bardzo dobrej (5.0) należy wykonać w pełni wszystkie 3 zadania. Dla przypomnienia na Rys. 1 przedstawiono układ pinów na Raspberry Pi. Ponadto przypominam o przesłaniu kodów programów (w szczególności z zadania nr 2 oraz 3) wraz ze zdjęciami działającego układu (po jednym na każdy układ).



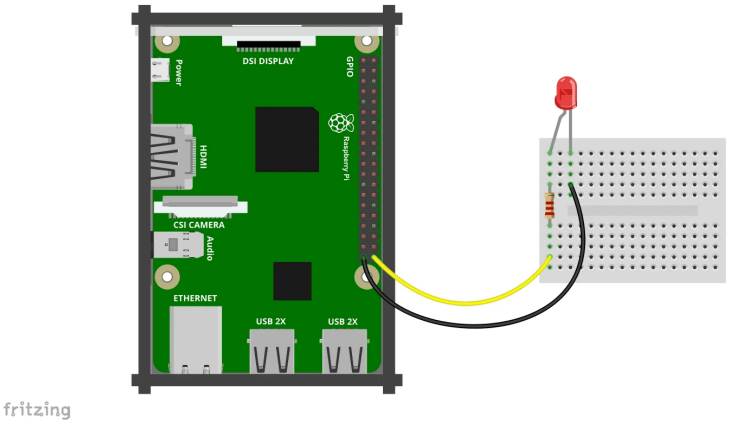
Rysunek 1: Numeracja pinów według `GPIO Board`

3 Opis zadań

3.1 Zadanie 1

Przed przystąpieniem to zadania warto przeczytać czym jest sygnał PWM? W skrócie jest to modulacja szerokością pulsu (*Pulse-Width Modulation*), tzn. jest to metoda sterowania sygnałem prądowym lub napięciowym, poprzez zmianę wypełnienia sygnału. Amplituda oraz częstotliwość sygnału pozostaje stała, ale zmienia się tak zwane wypełnienie. Najczęściej jest używany do zmiany jasności diod `LED`, sterowania prędkością silników lub pozycją serw modelarskich.

Ponownie podłączamy wszystko według schematu z poprzednich zajęć zawierającego układ z jedną diodą `LED`. Dla przypomnienia został pokazany na Rys.2. Pamiętamy o dobraniu odpowiedniego opornika do diody. Ponadto proszę nie robić pętli nieskończonych!



Rysunek 2: Układ z jedną diodą

W przypadku pracy z sygnałem `PWM` w pierwszej kolejności musimy standardowo ustawić tryb używanego pinu jako wyjście, następnie należy utworzyć instancję, do której przypiszemy ten sam pin oraz jego częstotliwość (dla diody `LED` odpowiednia częstotliwość będzie $50Hz$). W kolejnym kroku należy ustawić wypełnienie początkowe. Później można zmieniać wypełnienie w zakresie $[0, 100] \in \mathbb{C}$. Na poniższym listingu przedstawiono komplety kod do zadania.

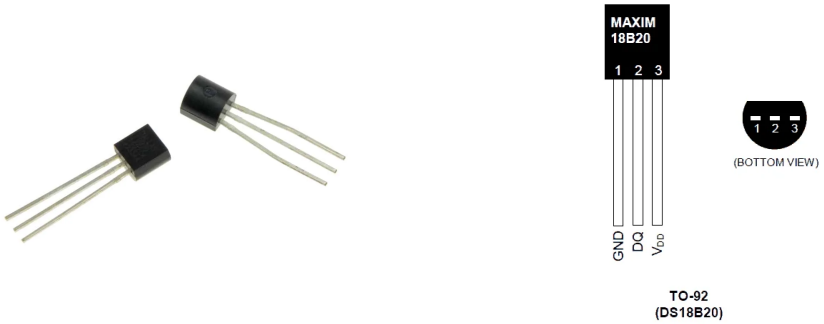
```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(40, GPIO.OUT)
led = GPIO.PWM(40, 50) #Nowa instancja PWM
fill = 0 #Wypełnienie sygnału PWM
led.start(fill) #Uruchomienie sygnału PWM

for in range(0,10):
    dioda.ChangeDutyCycle(wypelnienie) #Ustaw nową wartość wypełnienia
  time.sleep(0.5)
    fill +=10

LED.stop()
GPIO.cleanup()
```

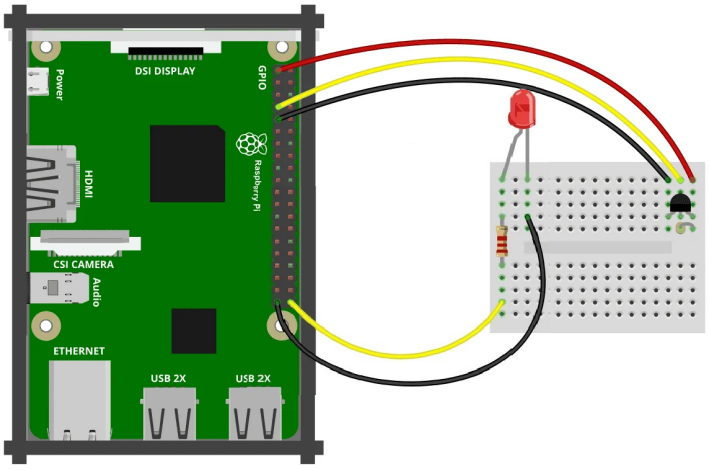
3.2 Zadanie 2

Zasadniczym celem zadania jest w pierwszej kolejności podłączenie czujnika temperatury, pokazanego na Rys. 3, następnie obsłużonego z poziomu programu w języku `PYTHON`. Układ z zadania należy rozszeżyć według schematu przedstawionego na Rys. 4.



Rysunek 3: Czujnik temperatury DS18B20

Używany czujnik pozwala wykonać pomiar temperatury z zakresu od -10^0C do $+85^0C$ stopni z dokładnością $\pm 0,5^0C$ oraz w zakresie od -55^0C do $+235^0C$ stopni z dokładnością $\pm 2^0C$. Czujnik wymaga podłączenia za pomocą interfejsu `1-wire`. Wyprowadzenie `GND` podłączamy do masy, linię danych `DQ` do pinu 7, a V_{DD} do zasilania $3,3V$, na koniec należy jeszcze zrobić mostek z użyciem rezystora $4,7k\Omega$ między zasilaniem a `DQ`.



Rysunek 4: Układ z jedną diodą i czujnikiem temperatury

Po poprawnym podłączeniu układu należy upewnić się, że interfejs `1-wire` jest włączony. Można użyć polecenia `lsmod`, jeśli na liście będą moduły `w1_gpio` oraz `w1_therm` to wszystko jest ok. Jeśli będzie brakować tylko `w1_therm` to znaczy, że system nie wykrył czujnika i należy wykonać jescze raz podłączenie najlepiej na wyłączonym urządzeniu. Jeśli na liście brakuje również `w1_gpio`, to należy włączyć interfejs `1-wire` w opcjach, komenda `sudo raspi-config` (w menu wybieramy `Interfacing Options`, następnie `1-wire`). Do obsługi programu niezbędny jest pakiet `w1thermsensor` dla języka `PYTHON`.

```
sudo pip3 install w1thermsensor
```

lub w przypadku braku `pip3`:

```
sudo apt-get install python3-w1thermsensor
```

```python
import w1thermsensor
sensor = w1thermsensor.W1ThermSensor()
temp = sensor.get_temperature()
print(temp)
```

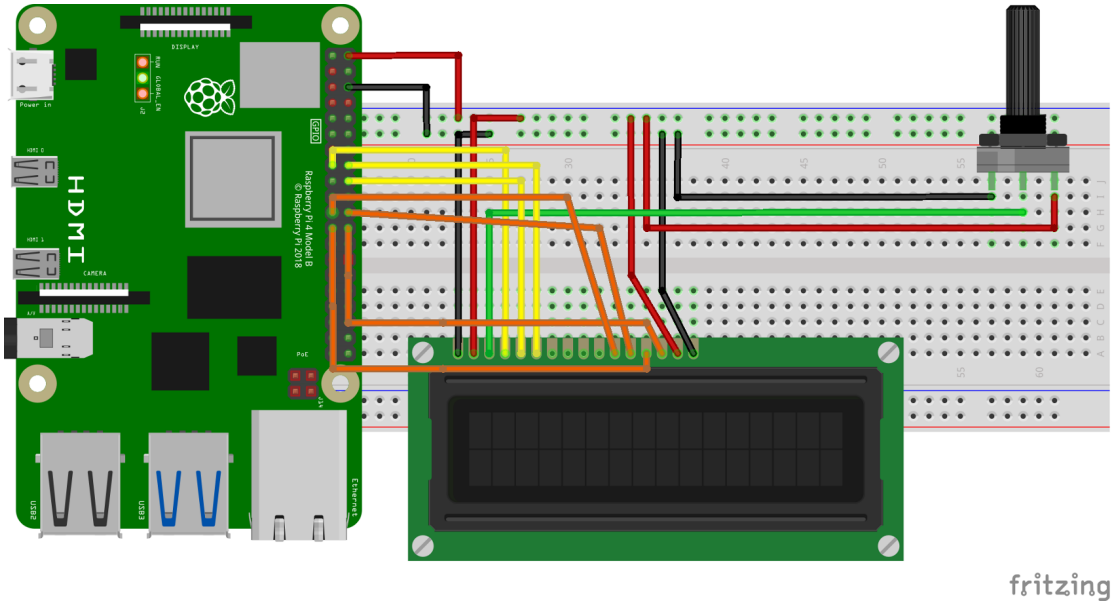
Na powyższym listnigu pokazano kod odczytu temperatury z czyjnika z wykorzystaniem pakietu `w1thermsensor`. Ostatnim krokiem zadania jest modyfikacja napisanego kodu w `PYTHON`. Temperaturę należy odczytywać z częstotliwością co $1s$ (debug–bieżąca temperatura powinna być wyświetlania w terminalu). Następnie w wersji podstawowej ma się zapalić dioda po przekroczeniu pewniej wartości progowej (po zmniejszeniu temperatury poniżej progu dioda powinna być wygaszona), ogrzać czujnik można przy przytrzymując go palcami. **Dla chętnych** zmiany temperatury można sygnalizować stopniem wypełnienia diody z wykorzystaniem sygnału `PWM`.

3.3 Zadanie 3

W ramach tego zadania nasz system zostanie uzupełniony o wyświetlacz `LCD` na którym będziemy wyświetlać bieżąco temperaturę (z odświeżeniem co $1s$). Zanim przejdziemy do podłączania, w pierwszej kolejności należy pobrać bibliotekę do obsługi wyświetlacza, przez użycie następującego polecenia:

```
$ wget -c https://github.com/dbrgn/RPLCD/archive/refs/heads/master.zip
```

Po pobraniu możemy rozpakować archiwum wpisując `unzip master.zip`. Katalog `RPLCD.gpio` po wypakowaniu powinien się znajdować w tym samym katalogu co nasz skrypt, żeby był widoczny z jego poziomu. Przykład połączenia pokazano na Rys. 5.



Rysunek 5: Schemat podłączenia samego wyświetlacza z użyciem potencjometru

Dla poprawienia czytelności w poniższej tabeli przedstawiono dokładny opis podłączania pinów wyświetlacza `LCD` według kolejności występowania. Poprawne podłączenie pinów jest bardzo istotne do prawidłowego działania. **UWAGA!** wyświetlacz działa z napięciem $5V$, więc te piny proszę podłączyć jako ostatnie.

LCD	VSS	VDD	V0	RS	RW	E	D0	D1	D2	D3	D4	D5	D6	D7	A	K
in	GND	$5V$	X	15	18	16	-	-	-	-	21	22	23	24	$5V$	GND

Pin `V0` wyświetlacza jest odpowiedzialny za kontrast wyświetlanego tekstu w zależności wartości sygnału z przedziału $[0, 1] \in \mathbb{R}$. W wersji podstawowej można użyć potencjometru z zestawu

i podłączyć go jak pokazano na Rys. 5, skrajna nóżka z symbolem Ω do uziemiania, przeciwna skrajna do napięcia $3,3V$ (nie ~~$5V$~~!!!), a środkowa do pinu `VO` wyświetlacza. Alternatywnie zamiast potencjometru można użyć odpowiedniego rezystora. **W wersji zaawansowanej na** 5.0 do sterowania kontrastem należy użyć sygnału `PWM`. Kolejnym krokiem jest uruchomienie poniższego kodu w jeżyku programowania `PYTHON`:

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

from RPLCD.gpio import CharLCD
#import the CharLCD library from RPLCD.gpio

GPIO.setwarnings(False)
#to ignore the warnings

lcd = CharLCD(pin_rs = 15, pin_rw=18, pin_e=16, pins_data= [21,22,23,24],
numbering_mode = GPIO.BOARD, cols=16, rows=2, dotsize=8)
#declare the LCD pins with GPIO pins of Raspberry Pi 4

lcd.clear()
#clear the screen of LCD

lcd.write_string("Industry 4.0")
#display the text on 16x2 LCD
```

Powyższy kod należy tak zmodyfikować, aby wyświetlał temperaturę na bieżąco z odświeżaniem co $1s$. Proszę zauważyć, że w kodzie nie ma instrukcji do sterowania sygnałem `PWM`. Jeszcze raz przypominam, żeby zachować ostrożność przy pracy z dwoma różnymi napięciami. Ponadto na koniec należy przesłać napisane kody programów wraz ze zdjęciami (jedno na układ).