Wrocław, 11 października 2023

Przemysł 4.0

Laboratorium 1

Obsługa Raspberry Pi

prowadzący: *Dr inż. Radosław Idzikowski*

1 Wprowadzenie

Celem laboratorium jest zapoznanie się z podstawową obsługa mikrokontrolera z rodziny Raspberry Pi w wersji 4b. W ramach zajęć będą do złożenia różne podstawowe układy, aby zapoznać się z obsługą wyjść oraz w drugiej kolejności wejść urządzenia. Programy będą pisane w języku programowania `PYTHON` z użyciem połączenia zdalnego (*Remote Control*) z poziomu komputera klasy PC z wykorzystaniem MS Visual Studio Code. Czas przewidziany na zadanie to 1 termin wraz z ocenieniem pracy. 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 i odnotowania postępów.

2 Zadania

W ramach zajęć należy w zespołach wykonać następujące zadania:

1. Zapoznanie się z obsługą Raspberry Pi oraz wykonanie układu z jedną diodą sterowną przez program w języku `PYTHON` (migająca diodą z odświeżaniem co $2s$).
2. Rozbudowanie układu z zadania pierwszego o dwie kolejne diody (każda w innym kolorze) w celu zasymulowania czterofazowego układu sygnalizacji świetlnej (ale o przyspieszonym działaniu – zmiana fazy co $2s$). Ponownie układem steruje program w języku `PYTHON`.
3. Obsługa sygnałów wejściowych na przykładzie układu z pojedynczą diodą oraz przyciskiem odpowiedzialnym za zmianę jej stanu – również układem steruje program w języku `PYTHON`.

Za wykonanie zadania nr 1 jest ocena dostateczna, za każde kolejne zadanie jest +1 do ocenu. Na ocenę bardzo dobrą (5.0) należy wykonać wszystkie trzy zadania. Po zakończaniu zajęć należy kody programów (w szczególności zadań nr 2 i 3) zgrać i przesłać do prowadzącego, można dołączyć zdjęcia działającego układu (jedno na każde zadanie).

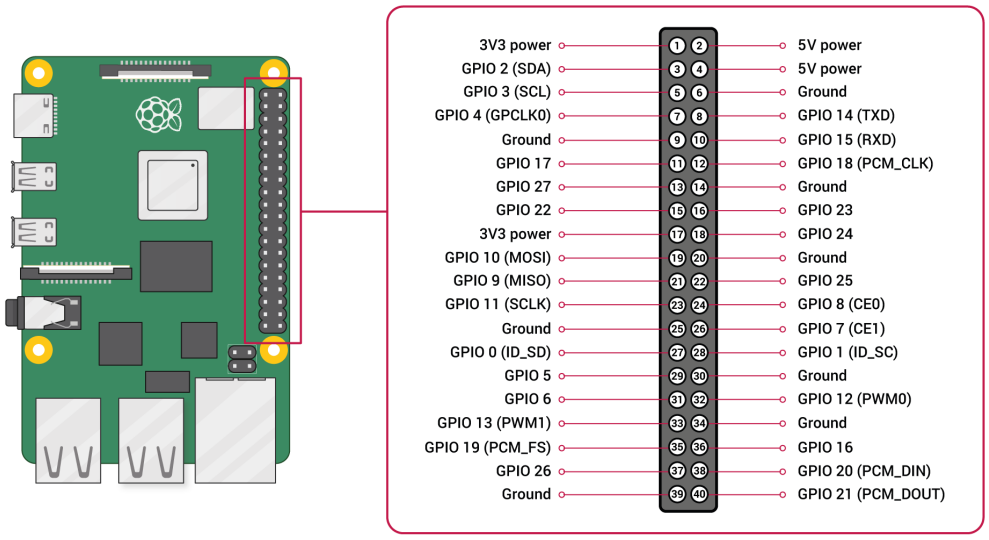
3 Opis zadań

3.1 Zadanie 1

Przed przystąpieniem do zadania warto zapoznać się z poniższymi materiałami:

- oficjalna dokumentacja Raspberry Pi,
- Opis czym jest Raspberry Pi,
- Kurs forbot dla Raspberry Pi,
- Sposób dobierania rezystorów do diody.

Poza tym należy pamiętać o:



Rysunek 1: Numeracja pinów według `GPIO Board`

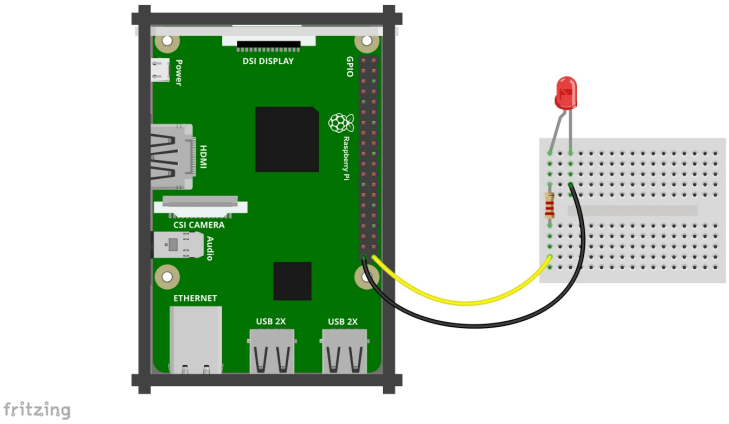
- dodaniu odpowiedniego opornika przed diodą, aby jej nie uszkodzić!
- kierunek podłączenia diody ma znaczenie!
- upewnić się co do wybranego sposobu numeracji pinów! Szczególnie przy kopiowaniu kodu proszę zwrócić uwagę na różnicę między `GPIO.setmode(GPIO.BOARD)`(przedstawiona na Rys. 1) a `GPIO.setmode(GPIO.BCM)`.
- na tych zajęciach zignorować piny odpowiedzialne za chłodzenie (zabezpieczają dostęp do napięcia $5V$, które mogłoby uszkodzić niektóre komponenty).

Po zapoznaniu się z dokumentacją i zapamiętaniu powyższych uwag, jesteśmy gotowi do połączenia się z Raspberry Pi. W tym celu należy przygotować sprzęt do pracy – w pierwszej kolejności należy uruchomić edytor MS Visual Code, następnie:

1. W dolnym lewym rogu kliknąć na ikonę “Open a Remote Window”.
2. Z rozwijanej listy wybrać opcję “Connect to Host...”
3. Wpisać nazwę Państwa urządzenia – “rp4b-p40-X”, gdzie X jest numerem na obudowie (od 1 do 8). Hasło: “raspberry”.
4. Kliknąć w opcję “Open Folder”, wpisać hasło ponownie.
5. Stworzyć folder na urządzeniu – odpowiadający Państwa grupie, gdzie przechowywane będą wszystkie programy, przykładowa nazwa: *wtorek_Imię1_Imię2*.
6. Stworzyć plik kodu w nowo utworzonym folderze – np. test.py. Następnie zedytować ten plik.
7. Za pomocą terminala uruchomić kod programu poleceniem: *python test.py*.

W celu łatwiejszego zapoznania się z programowaniem w języku `PYTHON` pod Raspberry Pi, poniżej zamieszczono gotowy kod wraz z komentarzami do całego zadania pierwszego. Przykładowy układ z wykorzystaniem pinu 40 (sygnał) oraz 39 (uziemienie) pokazano na Rys. 2. Przed uruchomieniem programu nalezy się upewnić, że jest zainstalowana biblioteka `RPi.GPIO`.

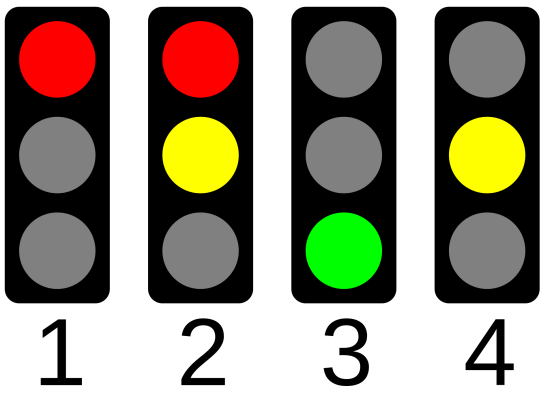
```python
import RPi.GPIO as GPIO # biblioteka niezbędna do kontrolowania stanu pinów
import time # biblioteka odpowiedzlna za reprezentację czasu

GPIO.setwarnings(False) # ignorowanie ostrzeżeń
GPIO.setmode(GPIO.BOARD) # WAŻNY KROK -- ustawiamy jaką numerację pinów
    wykorzystamy
GPIO.setup(40, GPIO.OUT) # Pin 40 ustawiony jako źrodło sygnału wyjściowego

for i in range(0, 10): # prosta pętla
    GPIO.output(40,GPIO.HIGH) # Na pinie 40 ustawiamy stan 1
    time.sleep(2) # 2s przerwy
    GPIO.output(40, GPIO.LOW) # Na pinie 40 ustawiamy stan 0
    time.sleep(2) # 2s przerwy

GPIO.cleanup() # na koniec programu -- sprzątanie
```



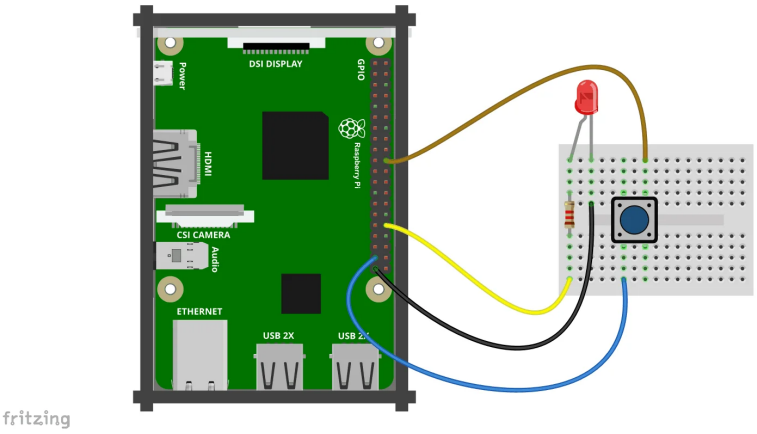
Rysunek 2: Układ z jedną diodą

3.2 Zadanie 2

W tym zadaniu należy rozbudować układ z zadania poprzedniego o dodatkowe diody w odpowiednim kolorze, pamiętając o wykorzystaniu odpowiednich rezystorów zależnie od koloru diody. Sygnalizacja świetlna zmienia się według schematu przedstawionego na Rys. 3. Trzeba pamiętać, aby użyć odpwiednich pinów, do uziemienia można użyć jednego pinu z wykorzystaniem szyny na płytce stykowej. Następnie należy zmodyfikować kod programu w `PYTHON`, aby sygnalizacja wchodziła we wszystkie 4 fazy z częstotliwością odświeżania $2s$. Dla ułatwienia można część funkcjonalności zamknąć w funkcjach.



Rysunek 3: Kolejne fazy sygnalizacji świetlnej



Rysunek 4: Przykład podłączenie układu z jedną diodą i przyciskiem

4 Zadanie 3

W tym wypadku również możemy bazować na układzie z zadania pierwszego. Dioda ponownie będzie sterowana przez program w języku `PYTHON`. Należy dodatkowo w naszym układzie dołożyć przycisk tak jak pokazano na Rys. 4. Sygnał do diody na pinie 12 oraz podłączanie do masy, przycisk również podłączamy do masy i do pinu 40.

Obsługę przycisku przedstawiono na poniższym listingu. Podobnie jak przy diodzie trzeba ustawić tryb pracy odpowiedniego pinu jako wejście. Ponadto jeśli zadeklarujemy wejście z podciągnięciem do dodatniej szyny zasilania (pull-up), to przy odczytaniu wartości poleceniem `GPIO.input(40)` i wciśnięciu przycisku otrzymamy stan niski 0.

```
import RPi.GPIO as GPIO # biblioteka niezbędna do kontrolowania stanu pinów
import time # biblioteka odpowiedzlna za reprezentację czasu

GPIO.setwarnings(False) # ignorowanie ostrzeżeń
GPIO.setmode(GPIO.BOARD) # WAŻNY KROK -- ustawiamy jaką numerację pinów
    wykorzystamy
GPIO.setup(40, GPIO.IN, pull_up_down=GPIO.PUD_UP) # deklaracja wejścia na pinie
    40 z podciągnięciem do dodatniej szyny zasilania

while True: # prosta pętla
    if GPIO.input(26) == 0:
        print("button pressed!")

GPIO.cleanup() # na koniec programu -- sprzątanie
```

Stosując opisane połączenie napotkamy problem z wielokrotnym wykryciem wciśnięcia przycisku. Jednym z rozwiązań jest zastosowanie pewnego opóźnienia bezpośrednio po wykryciu wciśnięcia. Bardziej skomplikowanym rozwiązaniem będzie zastosowanie rozwiązania sprzętowego.