Wrocław, 22 października 2024

Przemysł 4.0

Laboratorium 1

Obsługa wyjść i wejść w module

prowadzący: *Dr inż. Radosław Idzikowski*

1 Wprowadzenie

Celem laboratorium jest zapoznanie się z podstawową obsługą dedykowanego modułu bazującego na popularnym mikrokontrolerze z rodziny `Raspberry Pi CM4` (*Compute Module 4*). Podstawowym zadaniem będzie zapoznanie się z obsługą wyjść oraz następnie 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 wykonanie zadania to 1 termin wraz z ocenieniem pracy. Praca będzie oceniana na bieżąco w trakcie zajęć wraz z postępem programowania i testowania kolejnych przykładów. **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 napisanie pierwszego programu do sterowania diodami `LED` z użyciem języka `PYTHON`.
2. Obsługa prostych sygnałów wejściowych z wykorzystaniem przycisków.
3. `Reactometer` – program mierzący czas reakcji użytkownika na losowe zapalenie diod `LED`.

Za wykonanie zadania nr 1 jest ocena dostateczna, za każde kolejne zadanie jest +1 do oceny. Na ocenę bardzo dobrą (5.0) należy wykonać wszystkie trzy zadania. Po zakończeniu zajęć należy kody źródłowe napisanych programów (w szczególności zadań nr 2 i 3) zgrać i przesłać do prowadzącego. Można dołączyć zdjęcia potwierdzające poprawne działanie.

3 Opis zadań

3.1 Zadanie 1

Celem zadania jest sprawdzenie działania wszystkich diod `LED`. Należy napisać program, który będzie kolejno zapalał jedną diodę na 2s, następnie ją gasił i przechodził do następnej. Po zgaszeniu ostatniej diody, należy zapalić wszystkie 4 diody jednocześnie na 2s. Cały cykl należy powtórzyć 3 krotnie, ponieważ nie należy robić pętli **nieskończonych!**.

Po zapoznaniu się z dokumentacją i zapamiętaniu, jesteśmy gotowi do połączenia się z modułem. 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 adres przydzielonego urządzenia wraz z loginem– `pi@IP`, gdzie `IP` jest adresem urządzenia (pełna nazwa hosta jest widoczna po włączeniu wiersza poleceń na urządzeniu). 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*, polecenie `mkdir folder`.
6. Stworzyć plik kodu w nowo utworzonym folderze, osobny dla każdego zadania..
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 w Listingu 1, zamieszczono kod wraz z komentarzami do programu pozwalającego na sterowanie jedną diodą podłączoną na `PIN GPIO 19` . Przed uruchomieniem programu należy się upewnić, że jest zainstalowana biblioteka `RPi.GPIO`. W module kolejne diody `LED1`, `LED2`, `LED3` oraz `LED4` zostały wyprowadzone odpowiednio na `GPIO 19`, `GPIO 18`, `GPIO 13`, `GPIO 12`. Po zakończeniu wykonywania programu, zawsze trzeba pamiętać o wysyczeniu sygnałów na wejściach i wyjściach.

```
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.BCM) # WAŻNY KROK -- ustawiamy jaką numerację pinów
    wykorzystamy
GPIO.setup(19, GPIO.OUT) # Pin 40 ustawiony jako źrodło sygnału wyjściowego

for i in range(0, 4): # prosta pętla
    GPIO.output(19,GPIO.HIGH) # Na pinie 40 ustawiamy stan 1
    time.sleep(2) # 2s przerwy
    GPIO.output(19, GPIO.LOW) # Na pinie 40 ustawiamy stan 0
    time.sleep(2) # 2s przerwy

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

Listing 1: obsługa diody

3.2 Zadanie 2

W tym zadaniu sprawdzimy działanie przycisków przy wykorzystaniu diod `LED`. Należy zmodyfikować wcześniejszy kod tak, aby kliknięcie konkretnego przycisku zapalało i gasiło naprzemiennie odpowiadającą mu diodę. Tzn. pierwsze kliknięcie przycisku `SW1` ma zapalić diodę `LED1`, a drugie kliknięcie ma ją gasić. Wszystkie pary przycisków i diod mają działać niezależnie od siebie.

Obsługę przycisku przedstawiono na poniższym listingu. Podobnie jak wcześniej 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(17)` i wciśnięciu przycisku otrzymamy stan niski 0. W module kolejne przyciski `SW1`, `SW2`, `SW3` oraz `SW4` zostały wyprowadzone odpowiednio na `GPIO 17`, `GPIO 4`, `GPIO 3`, `GPIO 2`.

```
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.BCM) # WAŻNY KROK -- ustawiamy jaką numerację pinów
    wykorzystamy
GPIO.setup(17, 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(17) == 0:
        print("button pressed!")

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

3.3 Zadanie 3

Głównym zadaniem będzie implementacja systemu do pomiaru czasu reakcji. Do osiągnięcia celu ponownie będzie trzeba wykorzystać wszystkie diody oraz przyciski. W module są 4 diody `LED` ułożone pionowo i analogicznie poniżej nich przyciski. Program w pętli zapala losowo jedną diodę, następnie użytkownik ma nacisnąć przycisk odpowiadający zapalonej diodzie tak szybko, jak to możliwe. Program bezzwłocznie ją gasi i po chwili zapala losowo kolejną diodę. Ponadto program mierzy czas reakcji użytkownika od momentu zapalenia się diody do naciśnięcia odpowiedniego przycisku. Jeżeli użytkownik naciśnie niewłaściwy przycisk, program przerywa pracę sygnalizując błąd poprzez zapalenie wszystkich 4 diod, następnie w konsoli należy wypisać liczbę poprawnych kliknąć oraz średni i najlepszy czas reakcji z poprawnych prób.