Wrocław, 6 listopada 2023

Przemysł 4.0

Laboratorium 4

IoT Hub

prowadzący: *Dr inż. Radosław Idzikowski*

1 Wprowadzenie

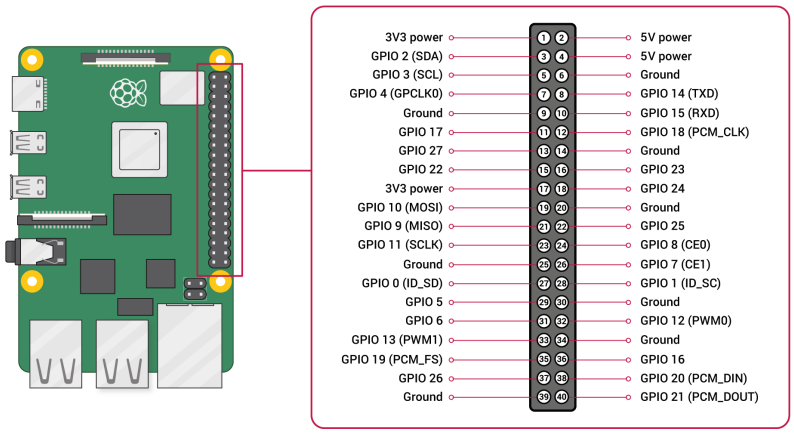
Celem laboratorium jest zapoznanie się z usługą `Azure IoT Hub`, obejmuje to budowę prostego układu pomiarowego oraz skonfigurowanie dedykowanego dashboardu. Praca będzie oceniana na bieżąco w trakcie zajęć wraz z postępem wykonywania zadań. Po ukończeniu każdego układu należy zawołać prowadzącego w celu zaakceptowania etapu.

2 Zadania

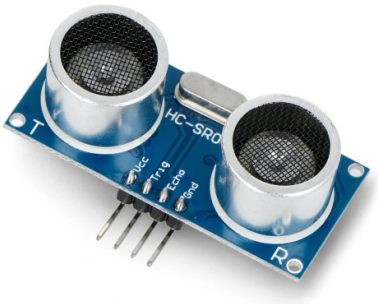
W ramach zajęć należy w zespołach wykonać następujące zadania:

1. Budowa prostego układu z czujnikiem odległości.
2. Konfiguracja usługi `Azure IoT Hub`.
3. Wizualizacja danych z wykorzystaniem technologii `Power BI`.

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. 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`



Rysunek 2: Ultradźwiękowy czujnik odległości `HC-SR04 2-200cm - justPi`

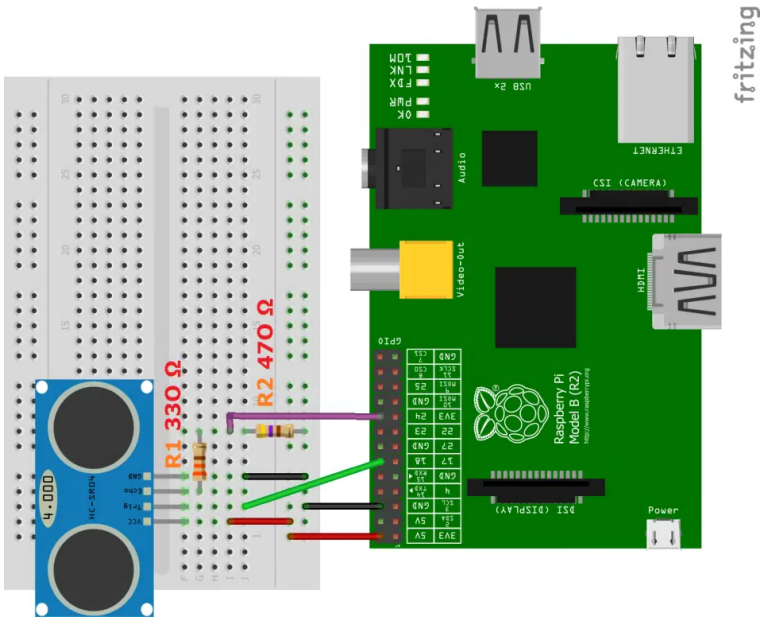
3 Opis zadań

3.1 Zadanie 1

W ramach zadania należy zbudować prosty układ pomiarowy wyposażony w ultradźwiękowy czujnik odległości (przedstawiony na Rysunku 2). Czujnik pozwala na pomiar odległości w zakresie $2 - 200cm$. Moduł dokonuje pomiaru odległości przy pomocy fali dźwiękowej o częstotliwości $40kHz$. Do mikrokontrolera wysyłany jest sygnał, w którym odległość zależna jest od czasu trwania stanu wysokiego i można ją obliczyć ze wzoru:

$$s_t = t_{hl} * v_s, \tag{1}$$

gdzie s_t testowana odległość, t_{hl} czas stanu wysokiego oraz v_s prędkość dźwięku. Czujnik należy podpiąć według schematu na Rys. 3, podłączając kolejno piny czujnika: `VCC` do napięcia $5V$, `GND` do uziemienia, `TRIG` do pinu 12 oraz `ECHO` **z użyciem rezystora** 330Ω do pinu 18 oraz do masy również **z użyciem rezystora**, ale w tym wypadku 470Ω.



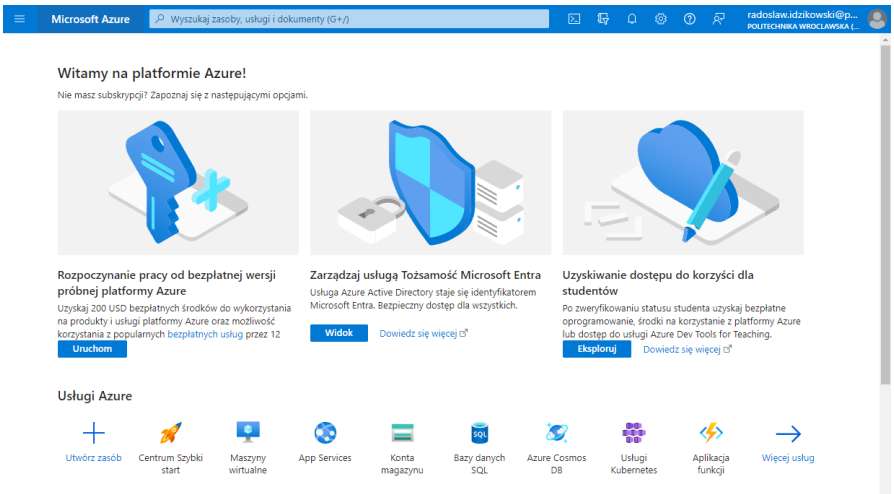
Rysunek 3: Układ z czujnikiem pomiaru odległości

```python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO_TRIGGER = 12
GPIO_ECHO = 18
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

def distance():
    GPIO.output(GPIO_TRIGGER, True) # ustawienie Trigger jako wysoki
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False) # zmiana Trigger na niski po 0.01ms
    StartTime = time.time() # zaincjowanie zegara
    StopTime = time.time() # zaincjowanie zegara
    while GPIO.input(GPIO_ECHO) == 0: #zapisanie czasu wysłania sygnału
        StartTime = time.time()
    while GPIO.input(GPIO_ECHO) == 1: #zapisanie czasu powrotu sygnału
        StopTime = time.time()
    TimeElapsed = StopTime - StartTime # wyliczenie różnicy czasu między wysł
    aniem a powrotem
    distance = (TimeElapsed * 34300) / 2 # wyliczenia dystansu
    return distance
```

Na powyższym listingu przedstawiono kod do obsługi czujnika ultradźwiękowego, który wyliczy zmierzony dystans. Czujnik wyślę sygnał dopiero po ustawieniu stanu wysokiego na wejściu `TRIGGER`. Podczas wysyłania sygnału i czekania na jego powrót ma wyjściu `ECHO` jest sygnał wysoki, więc wystarczy zmierzyć jego czas. Następnie przy użyciu wzoru (1) możemy wyliczyć dystans, odległość należy podzielić przez 2, ponieważ sygnał musi dojść do obiektu i wrócić. W ramach tego zadania wystarczy na bieżąco wyświetlać zmierzoną odległość w konsoli.

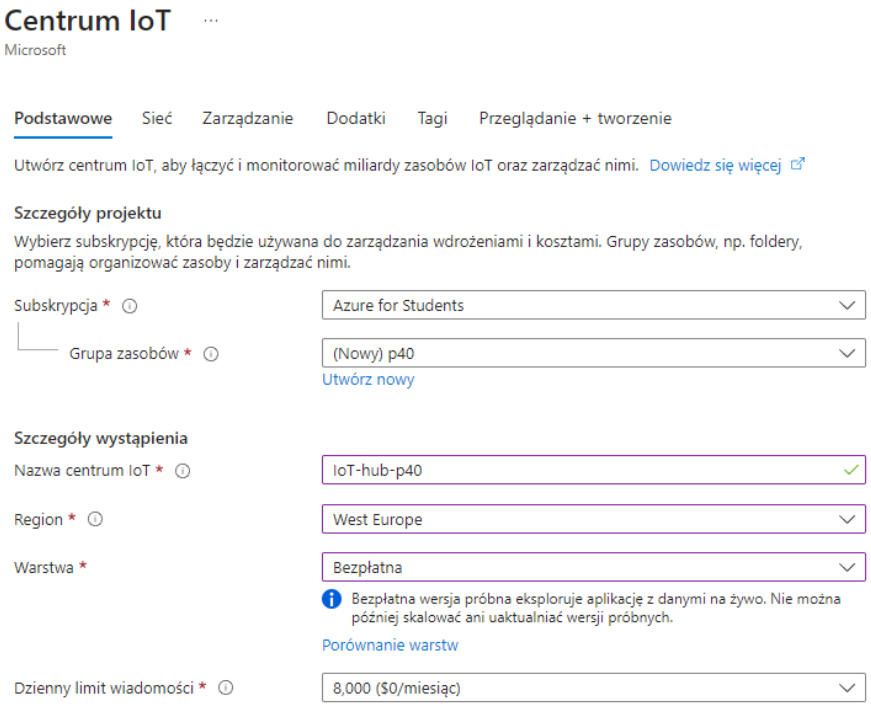
3.2 Zadanie 2

Celem zadania jest stworzenie interfejsu do obsługi Raspberry Pi w usłudze Azure. W pierwszej kolejności zaczniemy od konfiguracji odpowiednich komponentów w serwie **Azure**. Do portalu należy zalogować się z wykorzystaniem swojego konta **Microsoft**. Po zalogowaniu będzie widoczny widok pokazany na Rysunku 4. W ramach **Usługi Azure** należy utworzyć nowy zasób. Z kategorii **Internet rzeczy** wybieramy **IoT Hub**. Do utworzenia zasobu niezbędne jest wybranie subskrypcji (można wybrać wersję próbną lub dla studentów).



Rysunek 4: Interfejs usługi Azure po zalogowaniu

Do utworzenia zasobu potrzebne jest uzupełnienie odpowiednich pól według schematu na Rysunku 5. W szczegółach projektu (*projcet details*) należy wybrać aktywną subskrypcję oraz



Rysunek 5: Widok tworzenia nowego zasobu `IoT Hub`

wskazać grupę zasobów (początkowo nie istnieje ani jedna, więc należy utworzyć nową). Następnie w szczegółach wystąpienia (*Instance details*) należy określić nazwę naszego huba, region, warstwę (*Tier*, zalecany wybrać bezpłatny, ale można wybrać standardowy, ponieważ w próbnej subskrypcji powinien być darmowy depozyt w wysokości 100$, związany jest z tym limit przesyłanych wiadomości).

W zakładce `sieć` (*Networking*) należy określić ustawienia sieciowe. Podczas zajęć skorzystamy z dostępu publicznego. Ustawienie dostępu będzie można edytować nawet po utworzeniu huba. W zakładce `zarządzanie` (*Management*) należy określić model dostępu. Można zostawić ustawiania domyślnie (`Zasady dostępu współdzielonego i kontrola dostępu oparta na rolach`, oraz **brak** przypisania roli współautora). Liczba partycji wskazuje relację między komunikatami urządzenie-chmura, a liczbą czytników jednocześnie odczytujących te komunikaty. W przypadku większości centrów IoT wystarczą tylko cztery partycje, a w naszym wypadku powinny wystarczyć 2 (jest to maksymalna liczba w wersji dla studentów). Można włączyć tryb wersji zapoznawczej (ale nie jest to zalecane przy pierwszym korzystaniu) w caelu sprawdzania nowych funkcji, jednak może się to związać z brakiem dostępu części innych funkcjonalności.

Usługa `Azure` udostępnia również dodatkowo płatne dodatki: (1) `Device Update for IoT Hub` – do aktualizacji urządzania `IoT` z poziomu usługi oraz (2) `Defender for IoT` – do ochrony naszych urządzeń w sieci. W naszym wypadku żaden z wymienionych dodatków nie będzie potrzebny! Pola należy zostawić **odznaczone**!

Ostatnim krokiem podczas tworzenia `IoT Hub` jest przypisanie tagów i wartości w celu łatwiejszej ich identyfikacji przy rozliczaniu w przypadku większej liczby zasobów i urządzeń. Można też nie uzupełniać tagów. Na ostatniej zakładce widoczne będzie podsumowanie wybranych przez nas usług. Możemy potwierdzić utworzenie zasobu poprzez kliknięcie przycisku `Utwórz`.

Kolejnym etapem jest utworzenia urządzenia (*device*) w ramach stworzonego `IoT Hub`. Nazwa identyfikatora może być tożsama z nazwą hosta używanego Raspbery Pi. Zostawiamy typ uwierzytelniania jako klucz symetryczny oraz automatyczne wygenerowanie kluczy. Należy się upewnić, że opcją `Połącz to urządzenie z centrum IoT` jest **włączona**. Po utworzeniu urządzania można wejść w szczegóły urządzenia i sprawdzić wartości kluczy niezbędnych do połączenia.

Po skonfigurowaniu `IoT Hub` należy przejść do skonfigurowania Raspbery Pi. Aplikację klienta

można napisać z wykorzystaniem wielu języków programomania, jednak na potrzeby laboratorium skorzystamy standardowo z `Python` w wersji 3.X. Przed przystąpieniem do zadania należy zainstalować poniższe biblioteki:

```
sudo pip3 install azure-iot-device
sudo pip3 install azure-iot-hub
```

Na poniższym listingu przedstawiono kompletny kod dla klienta. Pomiar dystansu jest symulowany poprzez zmienią losową, a temperatura została wpisana na sztywno. Jako `CONNECTION_STRING` **konieczne** jest podanie wartości pola `Podstawowe parametry połączenia` w ustawieniach naszego urządzenia wewnątrz `IoT Hub`.

```
import random
import time
from azure.iot.device import IoTHubDeviceClient, Message

CONNECTION_STRING = "HostName=IoT-hub-p40.azure-devices.net;DeviceId=rp4b-p40-8;
    SharedAccessKey=vIPdvvWuLBWtmNqQkmOIwbZuqfuoTgs0/AIoTO3Y1+E="
MSG_TXT = '{{"distance": {distance},"temperature": {temperature}}}'
def init():
    client = IoTHubDeviceClient.create_from_connection_string(CONNECTION_STRING)
    return client

def run():
    client = init()
    for _ in range(3):
        param1 = random.random() * 15
        param2 = 21
        msg_txt_formatted = MSG_TXT.format(distance=param1, temperature=param2)
        message = Message(msg_txt_formatted)
        print( "Sending message: {}".format(message) )
        client.send_message(message)
        print ( "Message successfully sent" )
        time.sleep(1)

run()
```

Program możemy przetestować na szybko z wykorzystaniem `Cloud Shell`  . Przy pierwszym kliknięciu zostaniemy poproszeni o wybranie powłoki (zalecane `PowerShell` lub `Bash`). Następnie będziemy musieli utworzyć magazyn dla danych w ramach naszej subskrypcji. Kolejnym krokiem jest instalacja wewnątrz `Cloud Shell` poniższego pakietu:

```
az extension add --name azure-iot
```

Następnie możemy uruchomić monitor, aby podejrzeć podesłane komunikaty:

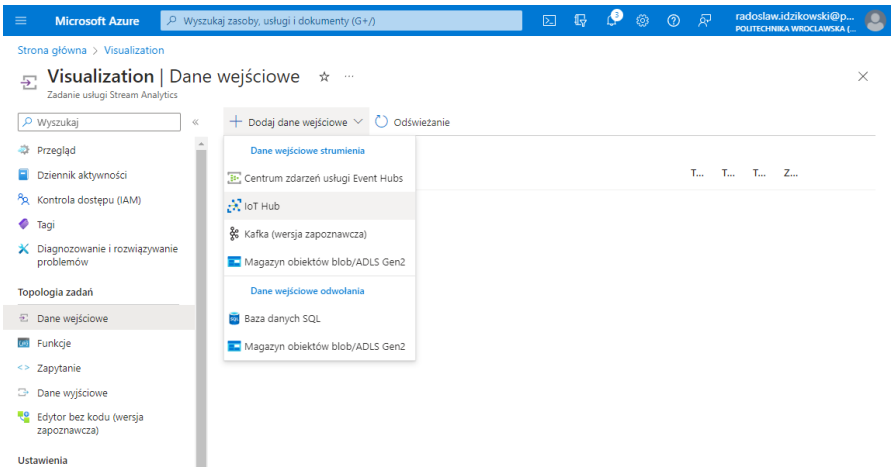
```
az iot hub monitor-events --hub-name IoT-hub-p40 --device-id rp4b-p40-8
```

Podając odpowiednie nazwy dla `--hub-name` oraz `--device-id` według wcześniejszych ustawień. Po włączaniu, monitor nasłuchuje przychodzących komunikatów. Następnie uruchamiamy program na Raspberry Pi, w wersji niezmodyfikowanej powinien przesłać 3 komunikaty z losowo wygenerowaną odległością. **Program należy zmodyfikować tak, aby przesyłał wartości zmierzone przez czujnik raz na sekundę!**

3.3 Zadanie 3

Na samym początku należy przejść na naszego `IoT Hub`, następnie `Ustawienia centrum` oraz `Wbudowane punkty końcowe`, ponieważ musimy ustawić naszą grupę konsumencką. Grupy konsumentów zapewniają niezależne widoki strumienia zdarzeń, które umożliwiają aplikacjom i usługom platformy `Azure` niezależne korzystanie z danych z tego samego punktu końcowego. Można zaproponować dowolną nazwę, będzie potrzebna w późniejszych krokach.

Teraz należy dodać nowy zasób `Azure Stream Analytics on IoT Edge`. Należy dodać nazwę zadania np. *Visualization*, wybrać naszą subskrybuję, następnie wcześniej zdefiniowaną grupę zasobów, lokalizację zostawiamy bez zmian oraz środowisko jako chmura.



Rysunek 6: Widok zadania usługi `Stream Analytics`

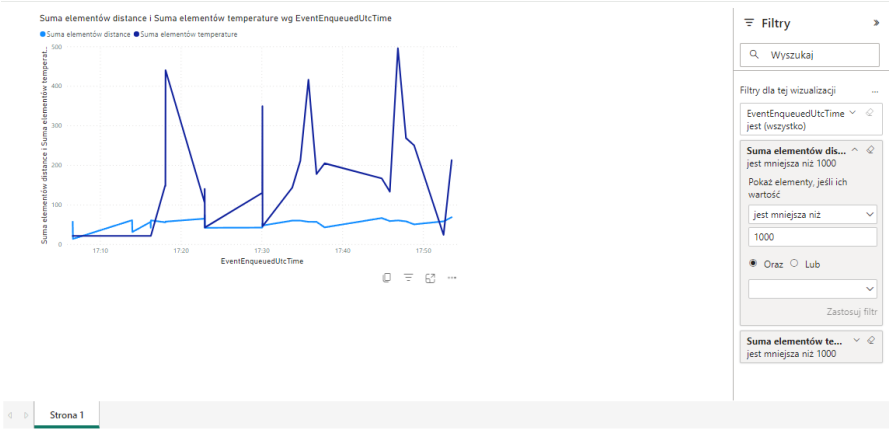
Po utworzeniu nowego zasobu `Stream Analytics`, przechodzimy do jego szczegółów, w sekcji `Topologia zadań` oraz `Dane wejściowe` dodajemy nowe dane wejściowe ze strumienia `IoT Hub` według schematu pokazanego na Rysunku 6. Większość ustawień zostawiamy domyślnych, należy wskazać odpowiednią grupą konsumentów z korku poprzedniego.

Kolejnym krokiem jest dodanie danych wyjściowych. Należy wybrać `Power BI`. Wprowadzamy nazwę np.: *PowerBIVisualizatio*, zostawiamy obszar roboczy grupy oraz typ uwierzytelniania. Należy wprowadzić nazwy dla zestawu danych oraz tabeli. Po uzupełnieniu trzeba autoryzować połączenie (potwierdzić dane logowania w usłudze `Power BI`). Na koniec wystarczy dać `zapisz`. **Następnie należy jeszcze uruchomić zadanie**.

Przechodzimy do usługi Power BI. W `obszarze roboczym` , powinien być widoczny utworzony wcześniej zbiór danych (*DataSet*) w ramach wyjścia z modułu `Stream Analytics`. Wybieramy utwórz raport. Z dostępnych wizualizacji wybieramy wykres liniowy, następnie z naszej tabeli wybieramy interesujące nas wcześniej zdefiniowane i mgazynowane dane: (1) `distance`, (2) `temperature` oraz (3) `EventEnqueuedUtCTime`. Stworzony wykres można wyeksportować na strone `HTML` (W `Plik`, opcja `Osadź raport, następnie Witryna internetowa lub portal`).

```
<!doctype html>
<html lang="pl">

<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="style.css">
  <title>Monitor</title>
</head>

<body>
<iframe title="test2" width="1140" height="541.25" src="https://app.powerbi.com/
    reportEmbed?reportId=be9475e6-74ad-43d5-8a83-5c665b67531b&autoAuth=true&ctid=
    d0da435b-a7e7-4d74-a4ae-f72cf8f3b2db" frameborder="0" allowFullScreen="true">
    </iframe>
</body>

</html>
```

Dla chętnych: Do istniejącego układu należy podpiąć jeszcze czujnik temperatury. W efekcie należy poprawnie wizualizować dwa mierzone parametry.



Rysunek 7: Widok wykresu z narzędzia `Power BI` na stronie `HTML`