



**WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI**
POLITECHNIKI RZESZOWSKIEJ

Filip Walkowicz

System IoT wraz z przesyłaniem danych na chmurę Azure oraz
wyświetlaniem w czasie rzeczywistym przy pomocy Power BI

Opiekun pracy:

dr inż. Mariusz Borkowski prof. PRz

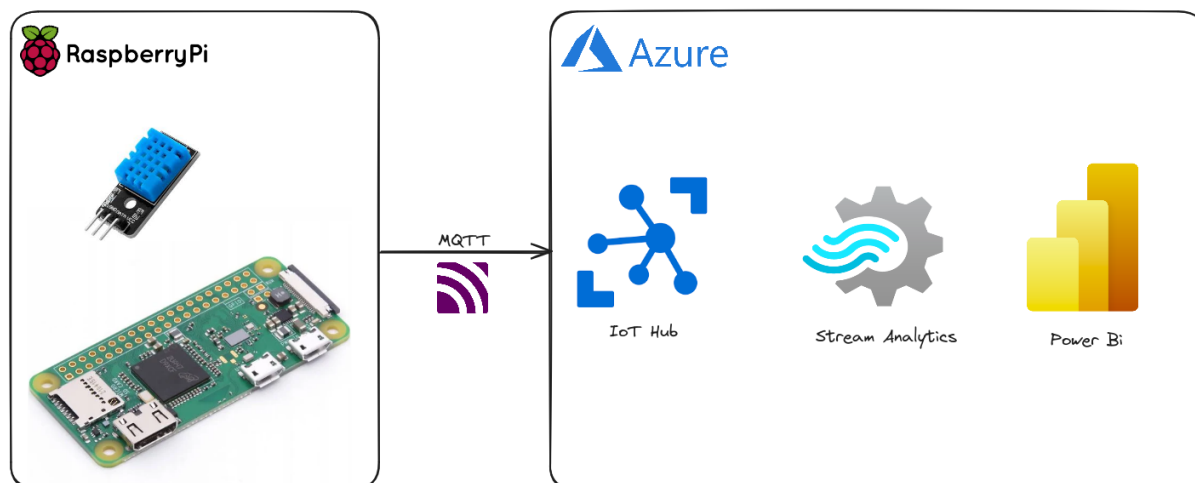
Rzeszów, 2024

Spis treści

1. Wstęp	3
2. Konfiguracja Raspberry Pi	4
2.1. Zbieranie danych z czujnika DHT11	5
2.2. Połączenie urządzenia z usługą IoT Hub	6
3. Konfiguracja usług na chmurze Azure	6
3.1. Konfiguracja IoT Hub	6
3.2. Konfiguracja Stream Analytics	9
3.3. Konfiguracja Power Bi	13
4. Podsumowanie i wnioski końcowe	15
Załączniki	16
Literatura	17

1. Wstęp

Celem projektu jest zbudowanie podstawowego obwodu z użyciem Raspberry Pi i czujnika DHT11, a następnie wykorzystanie go do gromadzenia danych oraz przesyłania ich i publikowania na serwerze Azure za pomocą protokołu MQTT. Po zebraniu danych przez chmurę zostaną one wyświetlone w czasie rzeczywistym w programie Power Bi.



Rysunek 1.1: Schemat projektu wraz z wykorzystanymi technologiami

Podczas projektu musimy wykonać podstawowe założenia projektowe które są niezbędne do poprawnego działania całego projektu:

- Poprawne zbieranie danych z czujnika
- Połączenie się z usługą Azure IoT Hub
- Zarejestrowanie czujnika oraz odbieranie danych
- Poprawne skonfigurowanie usługi Stream Analytics
- Ustawienie Query w pasujący dla nas sposób
- Przesyłanie danych do usługi Power Bi i wyświetlanie wyników

Kod został umieszczony na GitHub: <https://github.com/FWalkowicz/iot2cloud>

2. Konfiguracja Raspberry Pi

Podczas tego projektu korzystam z urządzenia Raspberry Pi zero W oraz czujnika temperatury i wilgotności powietrza DHT11, celem poprawnego działania naszego systemu musimy przygotować na początku nasz sprzęt. Zainstaluję aplikację od oficjalnego wydawcy ‘Imager’.

```
1 sudo apt install rpi-imager
```

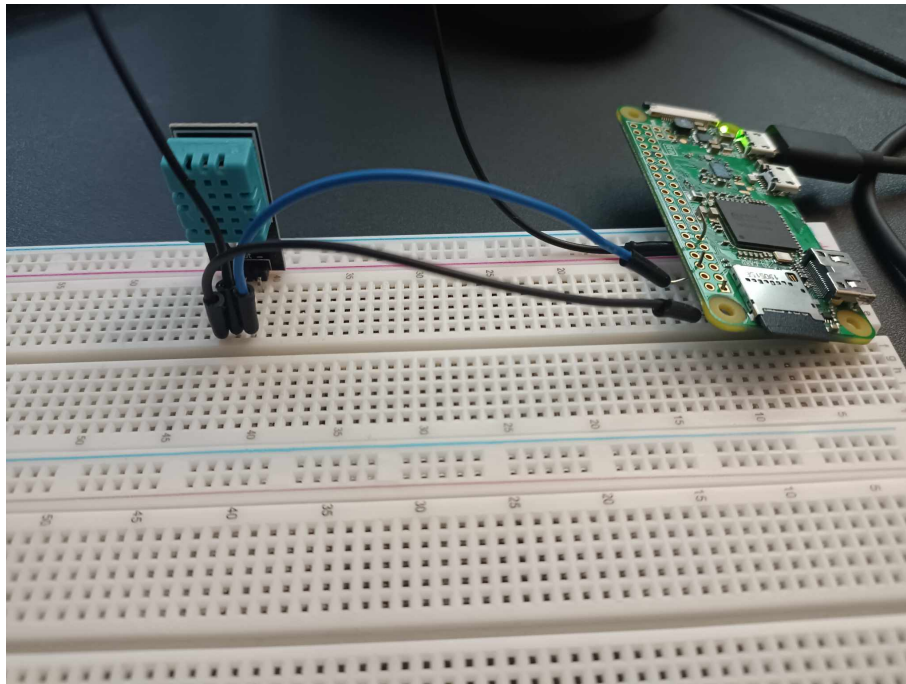
Listing 1: Instalacja oprogramowania Imager

Teraz możemy uruchomić program, wybrać odpowiedni obraz z listy, urządzenie oraz dostosować kastomizację naszego systemu pod nasze potrzeby. Dla naszych potrzeb będziemy właśnie musieli zastosować zmiany dla naszego obrazu. Dodamy do niego odpowiednią konfigurację użytkownika czyli dodamy mu nasze hasło, konfigurację sieci LAN gdzie podamy SSID naszej sieci oraz hasło i na koniec ustawimy odpowiednie ustawienia lokalne zgodnie z naszym miejscem zamieszkania.

The screenshot shows the 'Advanced options' window of the Raspberry Pi Imager. The window has a title bar with 'Advanced options' and a close button 'X'. Below the title bar, there is a dropdown menu for 'Image customization options' set to 'to always use'. The main content area is divided into two sections: 'Image customization options' and 'Persistent settings'. The 'Image customization options' section contains several checkboxes and input fields: 'Set hostname: raspberrypi.local' (checked), 'Enable SSH' (checked), 'Use password authentication' (selected with a radio button), 'Allow public-key authentication only' (unselected), 'Set authorized_keys for 'pi':' (empty field), 'Set username and password' (checked), 'Username: pi' (input field), 'Password: *****' (password field), 'Configure wireless LAN' (checked), 'SSID: TP-Link_4CD6' (input field), 'Hidden SSID' (unchecked), 'Password: *****' (password field), 'Show password' (unchecked), 'Wireless LAN country: PL' (dropdown menu), 'Set locale settings' (checked), 'Time zone: Europe/Warsaw' (dropdown menu), and 'Keyboard layout: us' (dropdown menu). The 'Persistent settings' section contains three checkboxes: 'Play sound when finished' (checked), 'Eject media when finished' (unchecked), and 'Enable telemetry' (checked). At the bottom of the window, there is a red 'SAVE' button.

Rysunek 2.2: Konfiguracja obrazu dla Raspberry Pi Zero

Podłączymy nasz czujnik teraz do naszego urządzenia:



Rysunek 2.3: Konfiguracja obrazu dla Raspberry Pi Zero

Następnie zainstalujemy odpowiednie biblioteki do komunikacji z chmurą Azure oraz do czujnika DHT, lecz najpierw musimy pobrać odpowiednie pliki z GitHub'a.

```
1 sudo pip3 install azure-iot-device
2 sudo pip3 install azure-iot-hub
3 sudo python setup.py install
```

Listing 2: Instalacja wymaganych paczek

2.1. Zbieranie danych z czujnika DHT11

```
1 import Adafruit_DHT
2
3 sensor = Adafruit_DHT.DHT11
4 pin = 17
5
6 while True:
7     humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
8     print(f"Temperature: {temperature}, humidity: {humidity}")
```

Listing 3: Kod prezentujący obsługę czujnika DHT11

2.2. Połączenie urządzenia z usługą IoT Hub

Microsoft udostępnia nam biblioteki dzięki którym w bardzo prosty sposób możemy połączyć się z naszym IoT Hub. Zaimportujemy wymagane biblioteki odpowiedzialne za nawiązanie połączenia oraz wysyłanie danych do chmury. Connection String uzupełnimy później gdy skonfigurujemy już naszą usługę IoT Hub oraz dodamy nowe urządzenie.

```
1 import Adafruit_DHT
2 from azure.iot.device import IoTHubDeviceClient, Message
3
4 sensor = Adafruit_DHT.DHT11
5 pin = 17
6
7 con_str = "XXXXXXXXXXXXXXXXXXXX"
8
9 client = IoTHubDeviceClient.create_from_connection_string(
10     con_str)
11
12 while True:
13     humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
14     message_data = {"temperature": temperature, "humidity":
15         humidity}
16     message = Message(message_data)
17     client.send_message(message)
```

Listing 4: Kod prezentujący połączenie z chmurą Azure oraz wysłanie pomiaru

3. Konfiguracja usług na chmurze Azure

Podczas tego punktu mojego projektu ważne było poprawne utworzenie oraz skonfigurowanie usług z których będę korzystał. Na samym początku tworzę IoT Hub którego zadaniem jest zbieranie danych z czujnika oraz rejestracja nowego urządzenia i stworzenie connection string'a, następnie te dane zostaną przepuszczone przez stream analytics job obrobione i wysłane do Power Bi albo do innych usług jakie zdefiniujemy na naszej chmurze.

3.1. Konfiguracja IoT Hub

Na samym początku musimy stworzyć naszą usługę IoT Hub. Wypełniamy wszystkie pola, w przypadku kiedy nie mamy jeszcze utworzonej Resource Group tworzymy nową jak widać na zdjęciu poniżej. Dajemy naszej usłudze nazwę oraz wybieramy tier który będzie odpowiedzialny za to ile razy możemy w ciągu dnia wysłać dane na chmurę. Gdy poprawnie uzupełniłem wszystkie dane kliknąłem opcję create.

Home > IoT Hub >

IoT hub

Microsoft

Basics Networking Management Add-ons Tags Review + create

Create an IoT hub to help you connect, monitor, and manage billions of your IoT assets. [Learn more](#)

Project details

Choose the subscription you'll use to manage deployments and costs. Use resource groups like folders to help you organize and manage resources.

Subscription * ⓘ Azure for Students

Resource group * ⓘ (New) walkowiczDHT
[Create new](#)

Instance details

IoT hub name * ⓘ walkowiczDHTiothub ✓

Region * ⓘ North Europe ✓

Tier * Basic ✓
[Compare tiers](#)

Daily message limit * ⓘ 400,000 (US\$10/month) ✓
[See all options](#)

Rysunek 3.4: Tworzenie usługi IoT Hub na chmurze Azure

Następnie musimy zarejestrować nasze urządzenie IoT żeby Hub wiedział że ono istnieje i może wysyłać dane. Klikamy w zakładkę Devices a następnie Add Device.

Microsoft Azure

Home > walkowiczdht11

walkowiczdht11 | Devices

Search

View, create, delete, and update devices in your IoT Hub. [Learn more](#)

+ Add Device Edit columns Refresh Assign tags Delete

enter device ID Types: All + Add filter

Device ID	Type	Status	Last status update	Authentication type	C2D messages queued	Tags
raspberrypiDHT11	IoT Device	Enabled	...	Shared Access Signature	0	

Device management

- Devices
- IoT Edge
- Configurations + Deployments
- Updates
- Queries

Hub settings

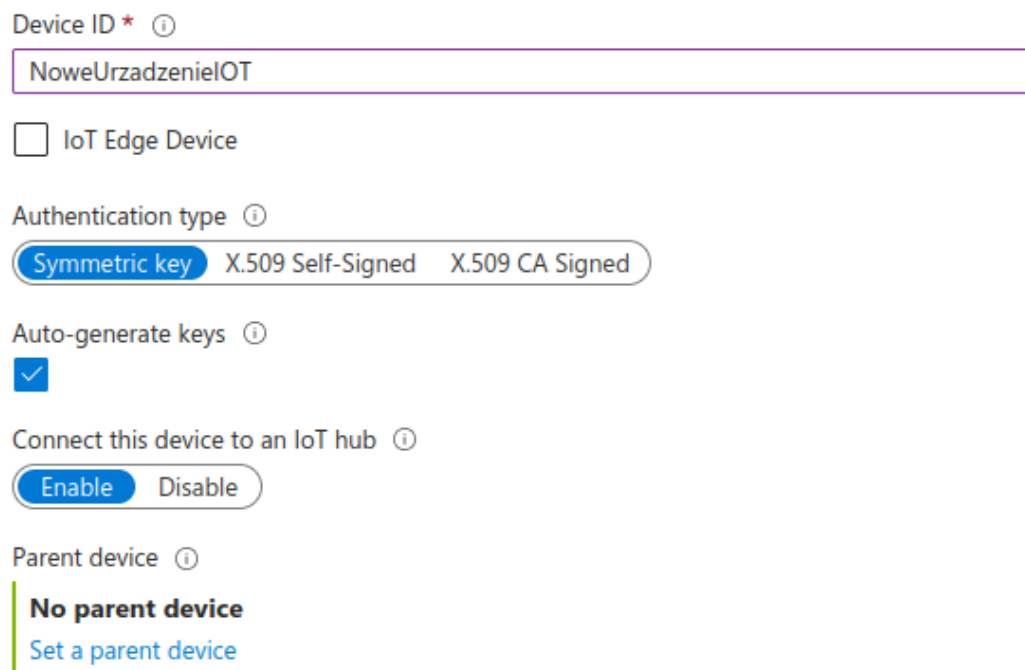
- Built-in endpoints
- Message routing
- File upload
- Fallover
- Pricing and scale
- Properties
- Locks

Security settings

- Identity
- Channel server runtime

Rysunek 3.5: Menu zarejestrowanych urządzeń IoT

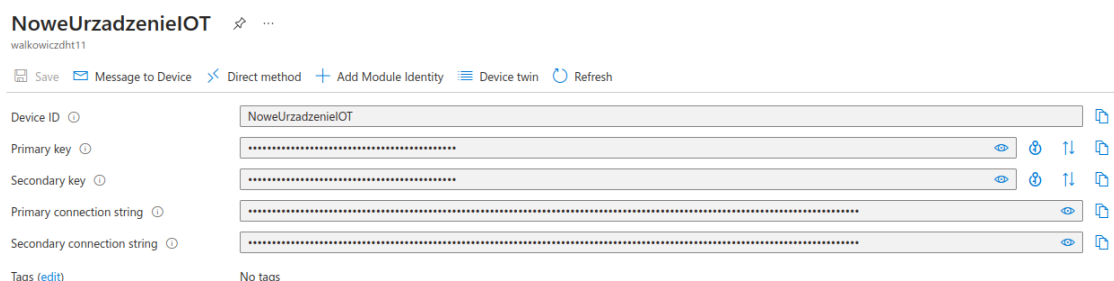
Przenieśmy się wtedy do nowej podstrony na której zostawiamy wszystkie domyślne opcje a dodajemy tylko wybraną przez nas nazwę urządzenia. Gdy poprawnie



The screenshot shows the 'Add new device' form in the Azure IoT Hub portal. The 'Device ID' field is filled with 'NoweUrządzenieIoT'. The 'IoT Edge Device' checkbox is unchecked. Under 'Authentication type', 'Symmetric key' is selected. 'Auto-generate keys' is checked. 'Connect this device to an IoT hub' has 'Enable' selected. Under 'Parent device', it says 'No parent device' with a link to 'Set a parent device'.

Rysunek 3.6: Tworzenie nowego urządzenia IoT

zarejestrowałem nowe urządzenie IoT, mogę z niego Devices je wybrać i zobaczyć potrzebne dane do połączenia z moją Raspberry Pi. Kopiuję stąd Primary connection string który zawiera wszystkie potrzebne informacje na komunikację z chmurą i wklejam go do naszego kodu.

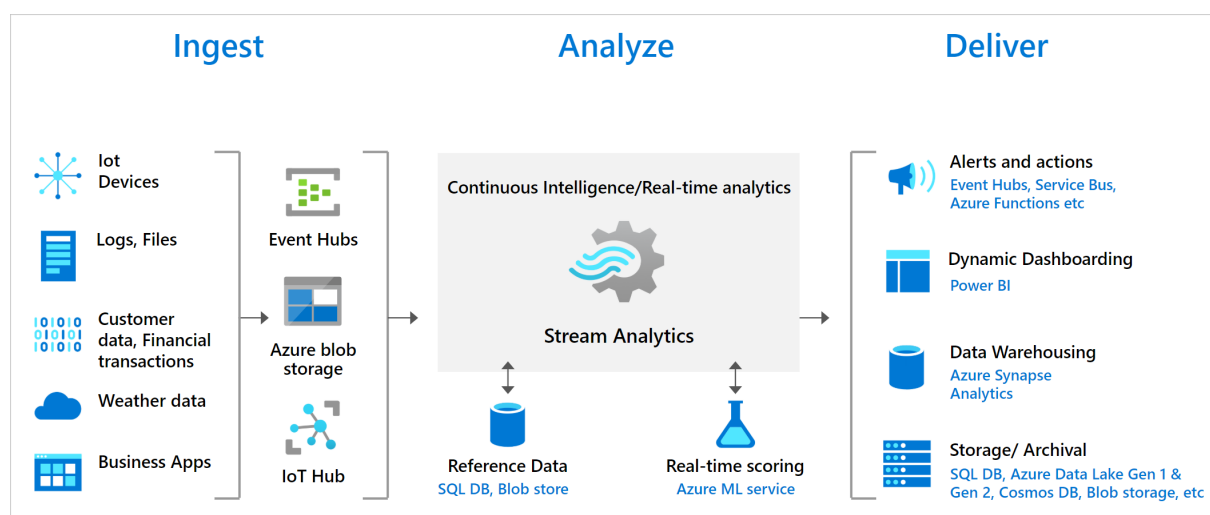


The screenshot shows the details page for the device 'NoweUrządzenieIoT'. It displays the 'Device ID', 'Primary key', 'Secondary key', 'Primary connection string', and 'Secondary connection string'. Each key and connection string is masked with asterisks and has a copy icon. There are also icons for refreshing and deleting. At the bottom, it says 'No tags'.

Rysunek 3.7: Dane dotyczące nowego urządzenia IoT

3.2. Konfiguracja Stream Analytics

Kolejnym krokiem będzie stworzenie oraz skonfigurowanie usługi Stream Analytics job która będzie odpowiedzialna za komunikację pomiędzy naszym IoT Hub a Power Bi. Azure Stream Analytics silnik przetwarzania strumieniowego, który analizuje i przetwarza duże ilości danych w czasie rzeczywistym oraz przekazywanie ich dalej tak jak u nas do Power Bi albo do innych zasobów na naszej chmurze np. Bazy danych, Modelu AI itd.



Rysunek 3.8: Schemat działania Stream Analytics

Z widoku menu głównego azure, wpisuje w wyszukiwarke "Stream Analytics Job" i wybieram pierwszą wyszukaną opcję, przekierowuje mnie to do widoku tworzenia owej usługi, wybieram moją odpowiednią grupę oraz nadaję unikatową nazwę. Resztę opcji zostawiam domyślne.

New Stream Analytics job ...

BasicsStorageTagsReview + create

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Azure for Students

Resource group * ⓘ

dht11

[Create new](#)

Instance details

Name *

walkowiczDHTstream

Region * ⓘ

(Europe) North Europe

Hosting environment *

☒ Cloud

☐ Edge

Streaming unit details

Streaming units (SUs) represents the computing resources that are allocated to execute a Stream Analytics job. The higher the number of SUs, the more CPU and memory resources are allocated for your job. The number of SUs can be modified once you create the job. You will be charged for the job's Streaming Units only when the job runs. [Learn more about streaming units](#)

All new Stream Analytics jobs created through the portal use Standard V2 pricing. [Visit Stream Analytics's pricing page to learn more.](#)

[For more options, visit here](#)

Streaming units *

1

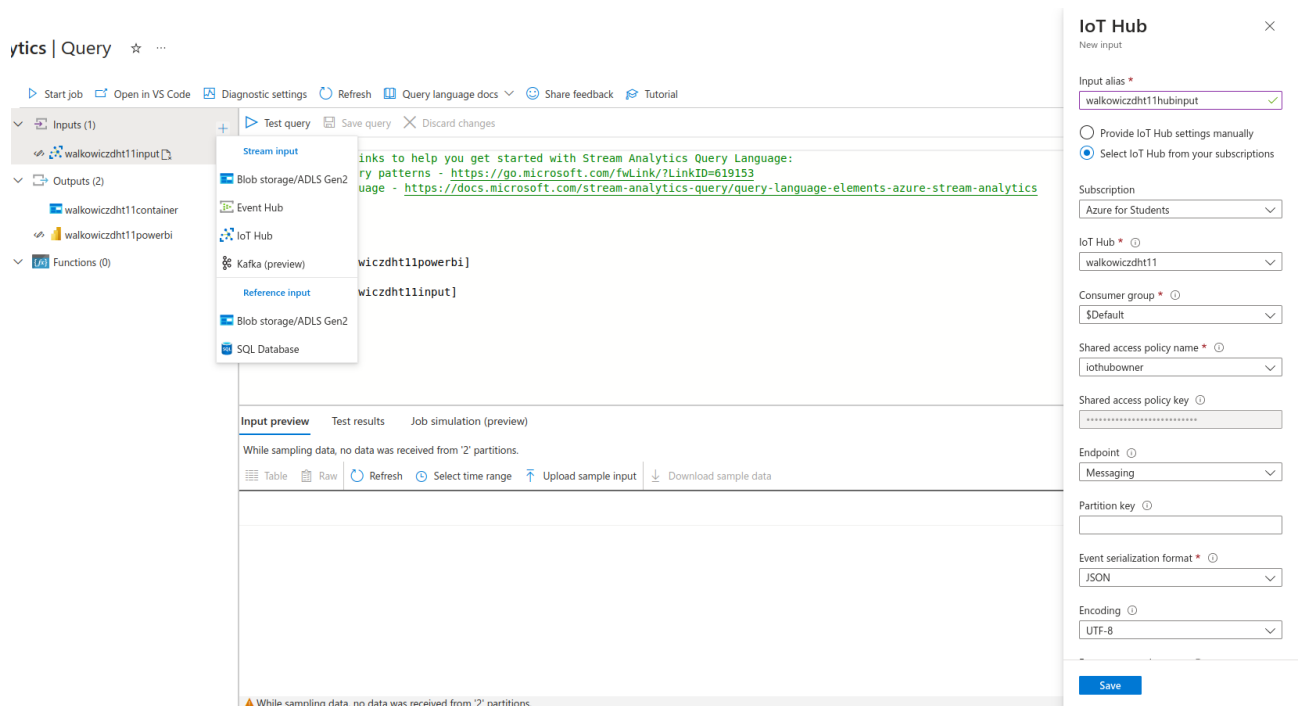
Previous

Next

Review + create

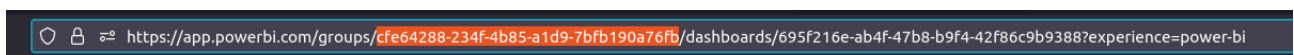
Rysunek 3.9: Tworzenie usługi Stream Analytics

Następnym krokiem będzie poprawne skonfigurowanie naszej usługi poprzez dodanie danych wejściowych, poprawnego zapytania oraz wyjścia gdzie te dane mają być przesłane. W tym celu przechodzę do zakładki Query gdzie szybko będę mógł dodać "Inputczyli IoT Hub oraz Outputczyli nasz Power Bi. W tym widoku także poprawnie skonfigurujemy "Query"które zawiera informacje o naszych pomiarach. W Sekcji input klikam "plus"i wybieram nasze źródło jako IoT Hub. Ustawiamy alias pod jakim chcemy mieć zapisany input, resztę rzeczy automatycznie zostanie wypełnione przez chmurę.



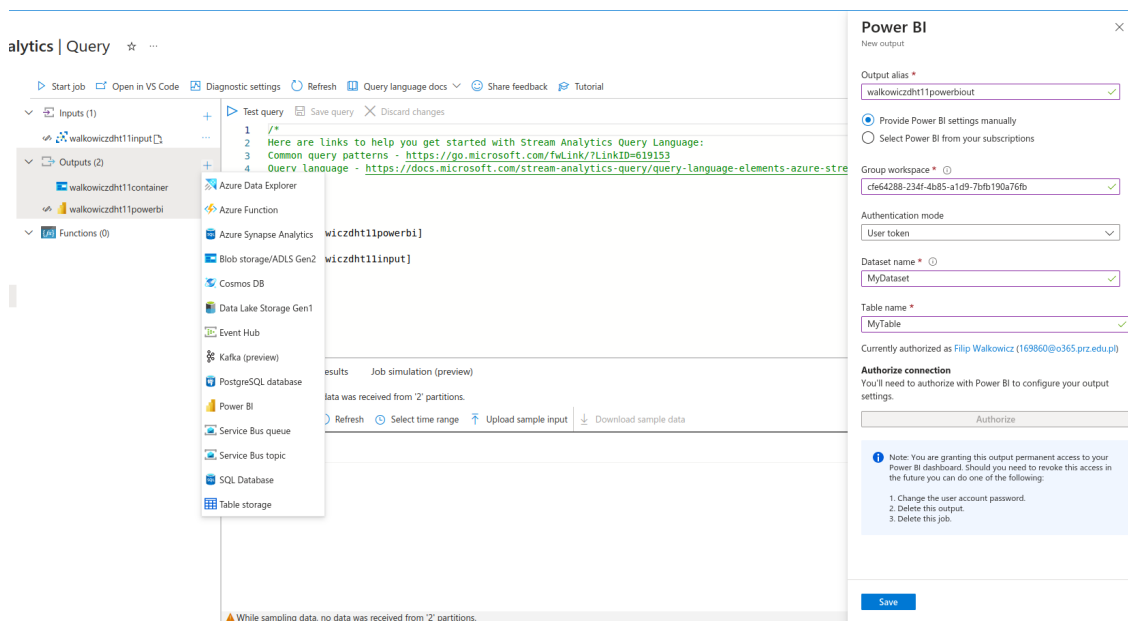
Rysunek 3.10: Dodawanie wejścia z IoT Hub

Następnie będziemy musieli przejść do usługi Power Bi gdzie będziemy musieli skopiować identyfikator grupy który będzie wymagany żeby dodać wyjście Power Bi. Znajdziemy go w url jak pokazano na zrzucie ekranu poniżej.



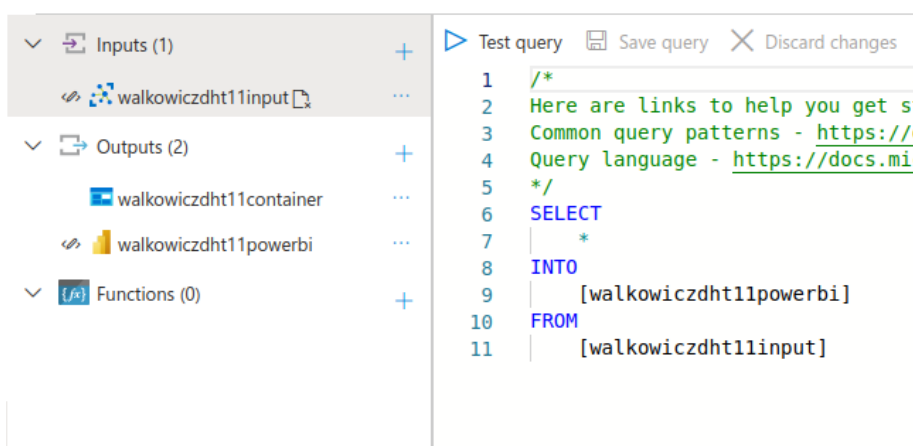
Rysunek 3.11: Identyfikator grupy Power Bi

Wracamy do naszej chmury i znowu w zakładce "Query"i teraz z pola "Outputs"klikamy plusik a następnie jako wyjście wybieramy Power Bi. Ustawiamy pasujący nam alias, wklejamy wcześniej skopiowany identyfikator grupy oraz ustawiamy nazwę zbiory danych oraz tabeli. Ostatnią rzeczą jaką będziemy musieli zrobić podczas dodawania to przeprowadzić autoryzację Power Bi poprzez zalogowanie się na konto na którym mamy tą usługę.Zbiór danych pod wybraną nazwą zostanie dodany i widoczny po poprawnym uruchomieniu usługi Stream Analytics.



Rysunek 3.12: Dodawanie wyjścia do Power Bi

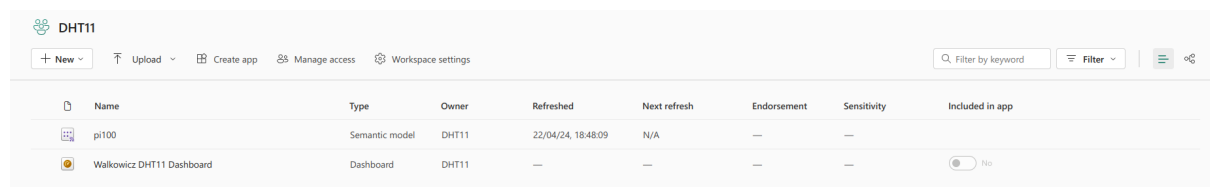
Teraz wystarczy tylko odpowiednio skonfigurować nasze zapytanie odpowiednie za przesyłanie danych. Ustawiamy odpowiednie aliasy z naszego wejścia oraz wyjścia. Pozostawiamy przesyłanie wszystkich danych przechodzących z wejścia. Po poprawnym Skonfigurowaniu zapytania należy uruchomić naszą usługę co może potrwać kilka minut.



Rysunek 3.13: Konfiguracja Query

3.3. Konfiguracja Power Bi

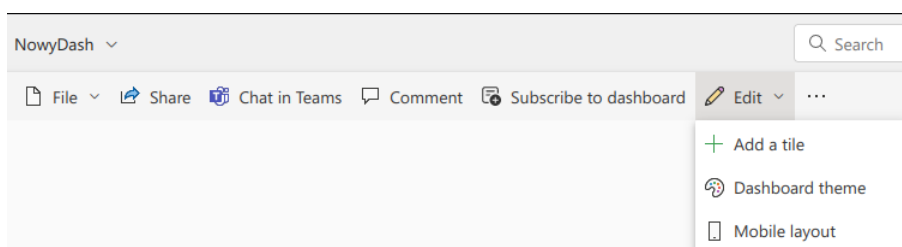
Teraz przejdziemy do naszej usługi Power Bi gdzie będziemy wyświetlać dane w czasie rzeczywistym za pomocą Dashboard'a. Przechodzimy do odpowiedniego Workspace'a którego identyfikator podaliśmy wcześniej podczas konfiguracji wyjścia. Jak widzimy już pojawił nam się w usłudze nowy dataset o naszej nazwie podanej wcześniej. Są to dane które dostarczyliśmy z naszego czujnika.



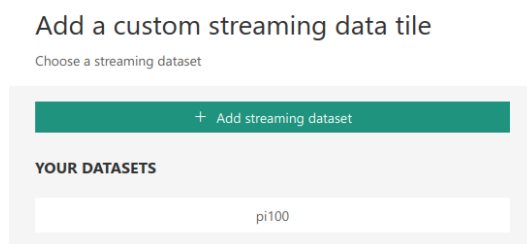
	Name	Type	Owner	Refreshed	Next refresh	Endorsement	Sensitivity	Included in app
	pi100	Semantic model	DHT11	22/04/24, 18:48:09	N/A	—	—	
	Walkowicz DHT11 Dashboard	Dashboard	DHT11	—	—	—	—	<input type="checkbox"/> No

Rysunek 3.14: Widok Workspace z danymi w Power Bi

Kolejną i ostatnią czynnością jaką wykonam jest dodanie do Dashboard'u nasze elementy które będą wyświetlały dane w czasie rzeczywistym. Z menu dodawania panelu wybieramy "real-time data" i przechodzimy dalej, będziemy mogli wybrać nasz dataset który zawiera dane z czujnika.

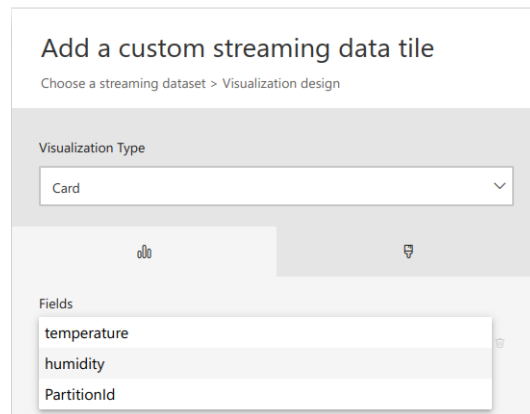


Rysunek 3.15: Dodawanie nowego wykresu do panelu Power Bi



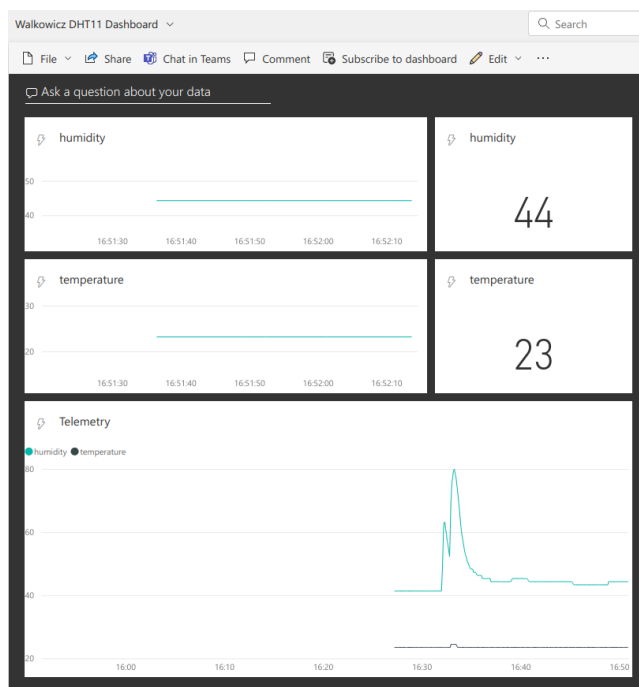
Rysunek 3.16: Dane z czujnika w usłudze Power BI

Teraz pozostało nam dodać interesujące nas panele wybierając odpowiednie źródła z naszej tabeli.



Rysunek 3.17: Dodawanie wykresów w Power BI

Końcowy wygląd naszego panelu zawiera trzy wykresy oraz dwa pola wyświetlające aktualną temperature oraz wilgotność powietrza. Wykresy prezentują następująco wilgotność, temperature w okresie ostatniej minuty oraz obie te wartości w ostatniej godzinie.



Rysunek 3.18: Końcowy wygląd dashboard'u w Power BI

4. Podsumowanie i wnioski końcowe

Projekt miał na celu zbudowanie podstawowego obwodu wykorzystującego Raspberry Pi i czujnik DHT11, oraz wykorzystanie go do zbierania danych, przesyłania ich i prezentowania w czasie rzeczywistym za pomocą chmury Azure i usługi Power BI. W ramach projektu zostały wykonane kluczowe założenia, takie jak poprawne zbieranie danych z czujnika, połączenie z usługą Azure IoT Hub, skonfigurowanie usługi Stream Analytics oraz wyświetlanie danych w Power BI.

Podczas konfiguracji Raspberry Pi, zainstalowałem niezbędne oprogramowanie, takie jak Imager, oraz biblioteki do komunikacji z chmurą Azure i obsługi czujnika DHT11. Następnie skonfigurowałem urządzenie do zbierania danych z czujnika oraz wysyłania ich do IoT Hub za pomocą protokołu MQTT.

W chmurze Azure skonfigurowałem usługę IoT Hub, zarejestrowałem urządzenie i skonfigurowałem połączenie. Następnie skonfigurowałem usługę Stream Analytics, dodając dane wejściowe z IoT Hub i wyjściowe do Power BI. Skonfigurowałem również zapytanie, które przekazywało dane z wejścia do wyjścia.

W usłudze Power BI dodałem dane w czasie rzeczywistym z naszego datasetu i stworzyliśmy panele do wyświetlania interesujących nas informacji.

Załączniki

```
1 # Importowanie bibliotek
2 import Adafruit_DHT
3 import time
4 from azure.iot.device import IoTHubDeviceClient, Message
5
6 # Definiowanie czujnika
7 sensor = Adafruit_DHT.DHT11
8 pin = 17
9
10 # Zmienne do polaczenia z chmura
11 CONNECTION_STRING = "XXXX"
12 MSG_SND = '{{"temperature": {temperature},"humidity": {humidity}
13     }}'
14
15 # Tworzenie klienta
16 def iotHub_client_init():
17     client = IoTHubDeviceClient.create_from_connection_string(
18         CONNECTION_STRING)
19     return client
20
21 # Zczytywanie danych z czujnika oraz wysyłanie
22 def iotHub_client_telemetry_sample_run():
23     try:
24         client = iotHub_client_init()
25         print("Sending data to IoT Hub, press Ctrl-C to exit")
26         while True:
27             humidity, temperature = Adafruit_DHT.read_retry(
28                 sensor, pin)
29             msg_txt_formatted = MSG_SND.format(temperature=
30                 temperature, humidity=humidity)
31             message = Message(msg_txt_formatted)
32             print("Sending message: {}".format(message))
33             client.send_message(message)
34             print("Message successfully sent")
35             time.sleep(3)
36     except KeyboardInterrupt:
37         print("IoTHubClient stopped")
38
39 # Start programu
40 if __name__ == '__main__':
41     print("Press Ctrl-C to exit")
42     iotHub_client_telemetry_sample_run()
```

Listing 5: Kod źródłowy

Literatura

- [1] <https://medium.com/linkit-intecs/azure-iot-with-raspberry-pi-send-temperature-and-humidity-sensor-data-to-azure-iot-hub-using-python-d84035f3911d>.
- [2] <https://learn.microsoft.com/en-us/azure/iot-hub/>
- [3] <https://learn.microsoft.com/en-us/azure/stream-analytics/power-bi-output>
- [4] <https://learn.microsoft.com/en-us/azure/stream-analytics/stream-analytics-real-time-fraud-detection>
- [5] <https://pypi.org/project/Adafruit-DHT/>