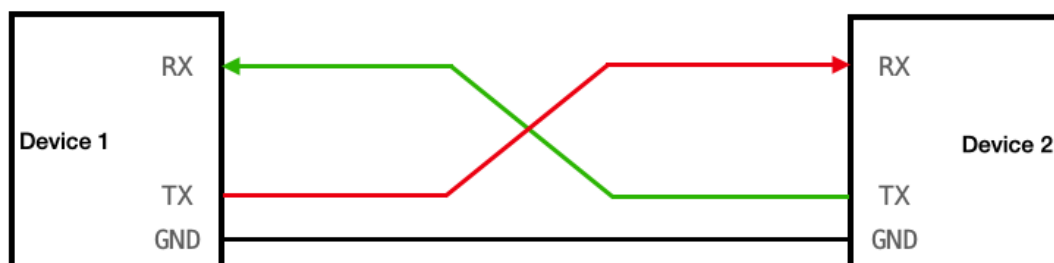


Communication UART

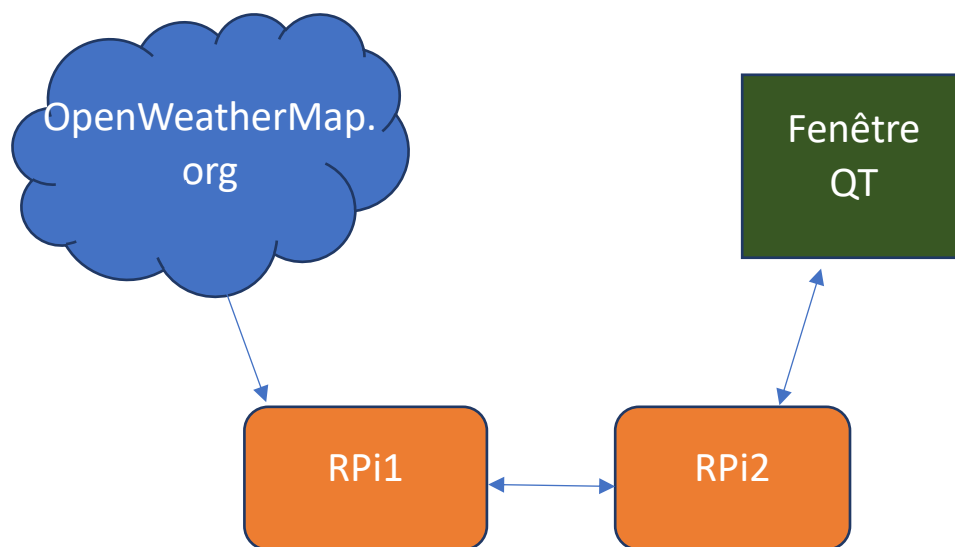


Haute Ecole de la Province de Liège

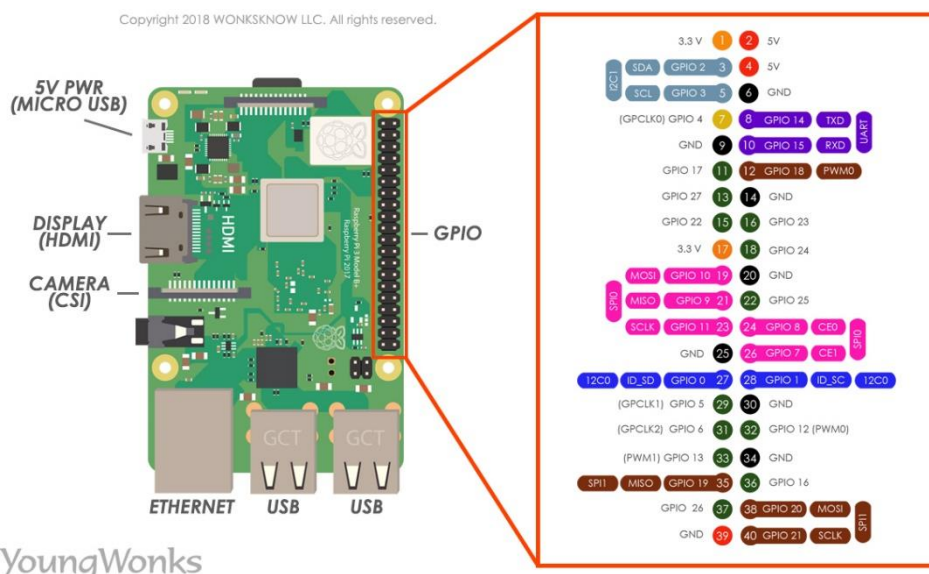


Introduction :

L'objectif de ce laboratoire est d'établir une communication UART avec un Raspberry Pi 3 et d'échanger des données entre deux codes. Dans cette configuration, le premier code sera responsable de la collecte des données météorologiques à partir d'OpenWeatherMap, tandis que le deuxième code se chargera d'afficher ces données sur une interface graphique Qt. De plus, le deuxième code comprendra un bouton qui déclenchera une requête vers le service web pour obtenir les dernières données météorologiques. La communication UART, réalisée via une connexion série entre les broches Rx et Tx du Raspberry Pi, permettra la transmission fluide des données entre les deux codes.



Raspberry PI 3 Pinout



Premier code : Transmetteur

Ce code sert à récupérer les prévisions météorologiques pour la ville de Liège à partir de l'API OpenWeatherMap. Il utilise la bibliothèque « requests » pour envoyer une requête « GET » à l'URL de l'API avec les paramètres spécifiés, notamment la clé d'API et les unités de mesure en degrés Celsius. Les données météorologiques sont extraites de la réponse JSON renvoyée par l'API, puis converties en une chaîne JSON. Ensuite, la chaîne JSON est envoyée via une connexion série vers un appareil externe. Enfin, les données météorologiques sont affichées et la connexion série est fermée.

```
import requests
import serial
import json

# Configuration de la connexion série
ser = serial.Serial(
    port='/dev/ttyS0', # Utilisation du bon port
    baudrate=9600, # initialisation du bon baud rate
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
)

# clé API
api_key = "bbcb2b6a701361f7dfc352d70ad3228d"

# Url de l'API qui nous permet d'aller chercher la météo
url = f'https://api.openweathermap.org/data/2.5/forecast?'

# Paramètres de la requête, on demande la ville, on utilise metric pour avoir
les données en degré celsius
parameters = {
    'q': 'Liège',
    'units': 'metric',
    'appid': api_key
}

# Faire la requête à l'API
request = requests.get(url, parameters)

#on extrait les données json
data = request.json()
#On convertit la variable data en une chaîne de caractère json
response= json.dumps(data)

#on envoie sur le port série
ser.write(response.encode())
#On affiche les données
print( data)
# Fermer la connexion série
ser.close()
```

[UART/UARTenvoie.py at main · hepl-hasiuk/UART \(github.com\)](#)

Résultats :

```
Fichier Édition Onglets Aide
AllyasLyra:~/Desktop $ python UARTenvoie.py
{"cod": 280, "message": 0, "cnt": 48, "list": [{"dt": 1684940480, "main": {"temp": 14.7, "feels_like": 13.74, "temp_min": 14.36, "temp_max": 14.7, "pressure": 1023, "sea_level": 1023, "grnd_level": 1015, "humidity": 58, "temp_kf": 0.34}, "weather": [{"id": 883, "main": "Clouds", "description": "broken clouds", "icon": "04d"}], "clouds": {"all": 82}, "wind": {"speed": 3.91, "deg": 11, "gust": 5.72}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-24 15:00:00"}, {"dt": 1684951200, "main": {"temp": 13.07, "feels_like": 12.29, "temp_min": 12.17, "temp_max": 13.07, "pressure": 1023, "sea_level": 1023, "grnd_level": 1015, "humidity": 71, "temp_kf": 0.91}, "weather": [{"id": 884, "main": "Clouds", "description": "overcast clouds", "icon": "04d"}], "clouds": {"all": 90}, "wind": {"speed": 3.05, "deg": 37, "gust": 6.43}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-24 18:00:00"}, {"dt": 1684962000, "main": {"temp": 10.51, "feels_like": 9.78, "temp_min": 10.51, "temp_max": 10.51, "pressure": 1025, "sea_level": 1025, "grnd_level": 1016, "humidity": 83, "temp_kf": 0}, "weather": [{"id": 884, "main": "Clouds", "description": "overcast clouds", "icon": "04d"}], "clouds": {"all": 100}, "wind": {"speed": 2.26, "deg": 41, "gust": 6.24}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-24 21:00:00"}, {"dt": 1684972800, "main": {"temp": 7.16, "feels_like": 6.43, "temp_min": 7.16, "temp_max": 7.16, "pressure": 1025, "sea_level": 1025, "grnd_level": 1016, "humidity": 96, "temp_kf": 0}, "weather": [{"id": 883, "main": "Clouds", "description": "broken clouds", "icon": "04d"}], "clouds": {"all": 58}, "wind": {"speed": 2.05, "deg": 29, "gust": 2.44}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-25 00:00:00"}, {"dt": 1684983600, "main": {"temp": 6.31, "feels_like": 4.93, "temp_min": 6.31, "temp_max": 6.31, "pressure": 1025, "sea_level": 1025, "grnd_level": 1017, "humidity": 98, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 10}, "wind": {"speed": 1.96, "deg": 45, "gust": 2.7}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-25 03:00:00"}, {"dt": 1684994400, "main": {"temp": 9.72, "feels_like": 8.08, "temp_min": 9.72, "temp_max": 9.72, "pressure": 1026, "sea_level": 1026, "grnd_level": 1018, "humidity": 91, "temp_kf": 0}, "weather": [{"id": 881, "main": "Clouds", "description": "few clouds", "icon": "02d"}], "clouds": {"all": 18}, "wind": {"speed": 3.16, "deg": 40, "gust": 6.02}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-25 06:00:00"}, {"dt": 1685005200, "main": {"temp": 14.85, "feels_like": 14.17, "temp_min": 14.85, "temp_max": 14.85, "pressure": 1027, "sea_level": 1027, "grnd_level": 1018, "humidity": 68, "temp_kf": 0}, "weather": [{"id": 802, "main": "Clouds", "description": "scattered clouds", "icon": "03d"}], "clouds": {"all": 44}, "wind": {"speed": 4.92, "deg": 37, "gust": 6.15}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-25 09:00:00"}, {"dt": 1685016000, "main": {"temp": 17.89, "feels_like": 17.22, "temp_min": 17.89, "temp_max": 17.89, "pressure": 1026, "sea_level": 1026, "grnd_level": 1018, "humidity": 57, "temp_kf": 0}, "weather": [{"id": 802, "main": "Clouds", "description": "scattered clouds", "icon": "03d"}], "clouds": {"all": 33}, "wind": {"speed": 4.93, "deg": 35, "gust": 5.81}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-25 12:00:00"}, {"dt": 1685026800, "main": {"temp": 18.74, "feels_like": 18.13, "temp_min": 18.74, "temp_max": 18.74, "pressure": 1026, "sea_level": 1026, "grnd_level": 1018, "humidity": 56, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01d"}], "clouds": {"all": 0}, "wind": {"speed": 3.79, "deg": 30, "gust": 9.66}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-25 15:00:00"}, {"dt": 1685037600, "main": {"temp": 15.45, "feels_like": 15.07, "temp_min": 15.45, "temp_max": 15.45, "pressure": 1027, "sea_level": 1027, "grnd_level": 1019, "humidity": 57, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01d"}], "clouds": {"all": 0}, "wind": {"speed": 4.68, "deg": 22, "gust": 8.43}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-25 18:00:00"}, {"dt": 1685048400, "main": {"temp": 10.61, "feels_like": 9.49, "temp_min": 10.61, "temp_max": 10.61, "pressure": 1028, "sea_level": 1028, "grnd_level": 1020, "humidity": 83, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 1}, "wind": {"speed": 3.79, "deg": 30, "gust": 9.66}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-25 21:00:00"}, {"dt": 1685059200, "main": {"temp": 7.94, "feels_like": 5.85, "temp_min": 7.94, "temp_max": 7.94, "pressure": 1029, "sea_level": 1029, "grnd_level": 1020, "humidity": 98, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 4}, "wind": {"speed": 3.31, "deg": 40, "gust": 5.15}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-26 00:00:00"}, {"dt": 1685070000, "main": {"temp": 5.99, "feels_like": 3.72, "temp_min": 5.99, "temp_max": 5.99, "pressure": 1029, "sea_level": 1029, "grnd_level": 1021, "humidity": 95, "temp_kf": 0}, "weather": [{"id": 802, "main": "Clouds", "description": "scattered clouds", "icon": "03n"}], "clouds": {"all": 30}, "wind": {"speed": 2.96, "deg": 37, "gust": 6.86}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-26 03:00:00"}, {"dt": 1685080800, "main": {"temp": 8.43, "feels_like": 5.68, "temp_min": 8.43, "temp_max": 8.43, "pressure": 1030, "sea_level": 1030, "grnd_level": 1021, "humidity": 81, "temp_kf": 0}, "weather": [{"id": 802, "main": "Clouds", "description": "scattered clouds", "icon": "03d"}], "clouds": {"all": 37}, "wind": {"speed": 4.86, "deg": 46, "gust": 9.78}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-26 06:00:00"}, {"dt": 1685091600, "main": {"temp": 13.41, "feels_like": 13.41, "temp_min": 13.41, "temp_max": 13.41, "pressure": 1030, "sea_level": 1030, "grnd_level": 1023, "humidity": 59, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01d"}], "clouds": {"all": 0}, "wind": {"speed": 4.56, "deg": 48, "gust": 11.83}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-26 09:00:00"}, {"dt": 1685102400, "main": {"temp": 16.89, "feels_like": 15.8, "temp_min": 16.89, "temp_max": 16.89, "pressure": 1029, "sea_level": 1029, "grnd_level": 1021, "humidity": 46, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01d"}], "clouds": {"all": 0}, "wind": {"speed": 5.84, "deg": 45, "gust": 7.36}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-26 12:00:00"}, {"dt": 1685113200, "main": {"temp": 10.69, "feels_like": 9.87, "temp_min": 10.69, "temp_max": 10.69, "pressure": 1029, "sea_level": 1029, "grnd_level": 1021, "humidity": 79, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 0}, "wind": {"speed": 4.61, "deg": 43, "gust": 5.35}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-26 15:00:00"}, {"dt": 1685124000, "main": {"temp": 15.21, "feels_like": 14.51, "temp_min": 15.21, "temp_max": 15.21, "pressure": 1028, "sea_level": 1028, "grnd_level": 1020, "humidity": 66, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 0}, "wind": {"speed": 5.11, "deg": 31, "gust": 9.57}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-26 18:00:00"}, {"dt": 1685134800, "main": {"temp": 10.68, "feels_like": 9.39, "temp_min": 10.68, "temp_max": 10.68, "pressure": 1029, "sea_level": 1029, "grnd_level": 1021, "humidity": 70, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 0}, "wind": {"speed": 4.35, "deg": 50, "gust": 5.74}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-26 21:00:00"}, {"dt": 1685145600, "main": {"temp": 7.31, "feels_like": 5.29, "temp_min": 7.31, "temp_max": 7.31, "pressure": 1030, "sea_level": 1030, "grnd_level": 1021, "humidity": 93, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 0}, "wind": {"speed": 2.98, "deg": 48, "gust": 6.86}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-27 00:00:00"}, {"dt": 1685156400, "main": {"temp": 6.41, "feels_like": 4.61, "temp_min": 6.41, "temp_max": 6.41, "pressure": 1024, "sea_level": 1024, "grnd_level": 1016, "humidity": 55, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 0}, "wind": {"speed": 4.61, "deg": 43, "gust": 5.35}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-27 03:00:00"}, {"dt": 1685167200, "main": {"temp": 18.83, "feels_like": 18.2, "temp_min": 18.83, "temp_max": 18.83, "pressure": 1026, "sea_level": 1026, "grnd_level": 1017, "humidity": 85, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01d"}], "clouds": {"all": 2}, "wind": {"speed": 4.94, "deg": 51, "gust": 5.82}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-27 06:00:00"}, {"dt": 1685178000, "main": {"temp": 19.82, "feels_like": 19.29, "temp_min": 19.82, "temp_max": 19.82, "pressure": 1024, "sea_level": 1024, "grnd_level": 1016, "humidity": 55, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01d"}], "clouds": {"all": 3}, "wind": {"speed": 4.61, "deg": 43, "gust": 5.35}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-27 09:00:00"}, {"dt": 1685188800, "main": {"temp": 17.2, "feels_like": 16.73, "temp_min": 17.2, "temp_max": 17.2, "pressure": 1024, "sea_level": 1024, "grnd_level": 1016, "humidity": 67, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 4}, "wind": {"speed": 5.8, "gust": 2.32}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-27 12:00:00"}, {"dt": 1685199600, "main": {"temp": 11.27, "feels_like": 10.62, "temp_min": 11.27, "temp_max": 11.27, "pressure": 1024, "sea_level": 1024, "grnd_level": 1016, "humidity": 83, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 2}, "wind": {"speed": 3.88, "deg": 30, "gust": 7.6}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-27 15:00:00"}, {"dt": 1685210400, "main": {"temp": 9.0, "feels_like": 8.39, "temp_min": 9.0, "temp_max": 9.0, "pressure": 1023, "sea_level": 1023, "grnd_level": 1015, "humidity": 91, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 4}, "wind": {"speed": 1.63, "deg": 78, "gust": 1.65}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-27 18:00:00"}, {"dt": 1685221200, "main": {"temp": 11.27, "feels_like": 10.62, "temp_min": 11.27, "temp_max": 11.27, "pressure": 1023, "sea_level": 1023, "grnd_level": 1015, "humidity": 91, "temp_kf": 0}, "weather": [{"id": 880, "main": "Clear", "description": "clear sky", "icon": "01n"}], "clouds": {"all": 4}, "wind": {"speed": 1.63, "deg": 78, "gust": 1.65}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-27 21:00:00"}, {"dt": 1685232000, "main": {"temp": 8.91, "feels_like": 8.91, "temp_min": 8.91, "temp_max": 8.91, "pressure": 1022, "sea_level": 1022, "grnd_level": 1013, "humidity": 90, "temp_kf": 0}, "weather": [{"id": 801, "main": "Clouds", "description": "few clouds", "icon": "02n"}], "clouds": {"all": 15}, "wind": {"speed": 1.2, "deg": 72, "gust": 1.21}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-28 00:00:00"}, {"dt": 1685242800, "main": {"temp": 12.93, "feels_like": 12.93, "temp_min": 12.93, "temp_max": 12.93, "pressure": 1021, "sea_level": 1021, "grnd_level": 1013, "humidity": 77, "temp_kf": 0}, "weather": [{"id": 802, "main": "Clouds", "description": "scattered clouds", "icon": "03d"}], "clouds": {"all": 39}, "wind": {"speed": 3.33, "deg": 41, "gust": 4.18}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-28 03:00:00"}, {"dt": 1685253600, "main": {"temp": 20.94, "feels_like": 20.47, "temp_min": 20.94, "temp_max": 20.94, "pressure": 1018, "sea_level": 1018, "grnd_level": 1010, "humidity": 53, "temp_kf": 0}, "weather": [{"id": 802, "main": "Clouds", "description": "scattered clouds", "icon": "03d"}], "clouds": {"all": 43}, "wind": {"speed": 3.68, "deg": 31, "gust": 3.73}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-28 06:00:00"}, {"dt": 1685264400, "main": {"temp": 18.51, "feels_like": 17.96, "temp_min": 18.51, "temp_max": 18.51, "pressure": 1020, "sea_level": 1020, "grnd_level": 1012, "humidity": 59, "temp_kf": 0}, "weather": [{"id": 802, "main": "Clouds", "description": "scattered clouds", "icon": "03d"}], "clouds": {"all": 39}, "wind": {"speed": 3.33, "deg": 41, "gust": 4.18}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-28 09:00:00"}, {"dt": 1685275200, "main": {"temp": 20.85, "feels_like": 20.85, "pressure": 1017, "sea_level": 1017, "grnd_level": 1009, "humidity": 57, "temp_kf": 0}, "weather": [{"id": 884, "main": "Clouds", "description": "overcast clouds", "icon": "04d"}], "clouds": {"all": 100}, "wind": {"speed": 4.33, "deg": 25, "gust": 4.47}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-28 12:00:00"}, {"dt": 1685286000, "main": {"temp": 17.38, "feels_like": 17, "temp_min": 17.38, "temp_max": 17.38, "pressure": 1017, "sea_level": 1017, "grnd_level": 1009, "humidity": 70, "temp_kf": 0}, "weather": [{"id": 884, "main": "Clouds", "description": "overcast clouds", "icon": "04d"}], "clouds": {"all": 93}, "wind": {"speed": 4.61, "deg": 21, "gust": 8.36}, "visibility": 10000, "pop": 0, "sys": {"pod": "d"}, "dt_txt": "2023-05-28 15:00:00"}, {"dt": 1685296800, "main": {"temp": 12.04, "feels_like": 11.47, "temp_min": 12.04, "temp_max": 12.04, "pressure": 1018, "sea_level": 1018, "grnd_level": 1010, "humidity": 83, "temp_kf": 0}, "weather": [{"id": 802, "main": "Clouds", "description": "scattered clouds", "icon": "03n"}], "clouds": {"all": 32}, "wind": {"speed": 3.35, "deg": 14, "gust": 7.02}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-28 18:00:00"}, {"dt": 1685307600, "main": {"temp": 8.11, "feels_like": 7.05, "temp_min": 8.11, "temp_max": 8.11, "pressure": 1019, "sea_level": 1019, "grnd_level": 1011, "humidity": 101, "temp_kf": 0}, "weather": [{"id": 801, "main": "Clouds", "description": "few clouds", "icon": "02n"}], "clouds": {"all": 15}, "wind": {"speed": 1.2, "deg": 72, "gust": 1.21}, "visibility": 10000, "pop": 0, "sys": {"pod": "n"}, "dt_txt": "2023-05-28 21:00:00"}]
```

Voici le résultat obtenu et envoyer à partir du premier code. On exécute ce programme grâce à l'invité de commande sur notre Raspberry PI 3. Les données que l'on envoie sont les résultats de la requête api. On y retrouve la température, l'heure, l'ensoleillement, ...

Deuxième code : Récepteur

Ce code est une application qui affiche les prévisions météorologiques des cinq prochains jours pour une ville spécifiée (dans cet exemple, Liège). Il utilise la communication série pour recevoir les données météorologiques provenant d'un dispositif externe. Les données sont lues depuis le port série, décryptées et nettoyées pour extraire une chaîne JSON valide. Les données JSON sont ensuite chargées dans un dictionnaire. L'application crée une interface graphique à l'aide de la bibliothèque PyQt5, comprenant un tableau pour afficher les dates, heures, températures et descriptions. Lorsque le bouton "Mettre à jour" est cliqué, les données météorologiques sont récupérées à partir du dictionnaire, les cinq prochains jours sont calculés, et les données correspondantes sont affichées dans le tableau.

```
#importation des bibliothèques
import serial #sert pour la communication série
import sys #gère les fonctionnalités du système
#PyQt5 gère les interfaces graphiques
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout,
QTableWidget, QTableWidgetItem, QHeaderView, QPushButton
from PyQt5.QtCore import Qt
```

```

import json #permet de traiter les données JSON
from datetime import datetime, timedelta #datetime pour la manipulation sur
les dates et time delta pour les opérations sur les dates

#Cette partie configure la communication série
ser = serial.Serial(
    port='/dev/ttyS0', #Le nom du port de notre device
    baudrate=9600, #On règle le débit en bauds de notre device
    parity=serial.PARITY_NONE, #pas de parité
    stopbits=serial.STOPBITS_ONE, #un bit d'arrêt
    bytesize=serial.EIGHTBITS, #une taille en bit
    timeout=1 #un délai d'attente
)

#La boucle est utilisée pour lire les données provenant de la communication
série
#Les données sont lues à partir du port série, décryptées en tant que chaîne
de caractères, puis nettoyées
while True:

    response = ser.readline().decode().strip()

    if response: # Si des données sont reçues, elles sont extraites en tant
que chaîne JSON valide à partir de la réponse.
        start_index = response.find('{')
        end_index = response.rfind('}')
        json_data = response[start_index:end_index + 1]
        #Ensuite, les données JSON sont chargées dans un dictionnaire data2 à
l'aide de la fonction json.loads()
        data2 = json.loads(json_data)
        print("Données météo :")
        print(json_data)
        #Enfin, les données météorologiques sont affichées et la boucle est
interrompue
        break
    else:
        #Si aucune donnée n'est reçue, un message d'attente est affiché.
        print('En attente de données...')

ser.close()#Enfin on ferme la connection série

#Cette fonction permet obtenir les dates des cinq prochains jours à partir de
la date actuelle.
def get_next_5_days():
    days = []
    today = datetime.now().date() #on obtient la date actuelle
    #puis les cinq jours suivants sont calculés en ajoutant un délai de un
jour à chaque itération de la boucle.

```



```

        for i in range(1, 6):
            day = today + timedelta(days=i)
            days.append(day.strftime('%Y-%m-%d'))#format : 'AAAA-MM-JJ' et
puis et ajoutées à une liste days
            #la liste des dates est affichée et renvoyée.
        print(days)
        return days

class MainWindow(QMainWindow):
    def __init__(self, city): #le constructeur initialise la fenêtre
principale avec un titre, une géométrie et une ville spécifiée
        super().__init__()
        self.setWindowTitle('OpenWeatherMap') #titre
        self.setGeometry(100, 100, 800, 600)
        self.city = city #ville
        self.create_widgets() #permet de créer les widgets à afficher dans la
fenêtre.

    def create_widgets(self):
        self.table = QTableWidgetItem(0, 4, self)#création du tableau avec quatre
colonnes représentant la date, l'heure, la température et la description
        self.table.setHorizontalHeaderLabels(['Date', 'Heure', 'Température
(°C)', 'Description'])
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch
)

        #Un bouton "Mettre à jour" est ajouté ainsi que le tableau à un widget
et à un layout pour les disposer correctement dans la fenêtre.
        widget = QWidget(self)
        layout = QVBoxLayout(widget)
        layout.addWidget(QPushButton('Mettre à jour',
clicked=self.update_weather_data))
        layout.addWidget(self.table)

        self.setCentralWidget(widget)

    def update_weather_data(self): #cette méthode est définie pour mettre à
jour les données météorologiques dans le tableau
        data = data2 #Les données récupérées précédemment sont stockées dans
une variable data
        if data is not None:
            next_5_days = get_next_5_days() #Si des données existent, les cinq
prochains jours sont obtenus à l'aide de la fonction get_next_5_days()
            row_count = len(next_5_days) * 8 #Le nombre de lignes du tableau
est calculé en multipliant la longueur des jours par 8
            self.table.setRowCount(row_count) # fix 1
            row = 0

```

```

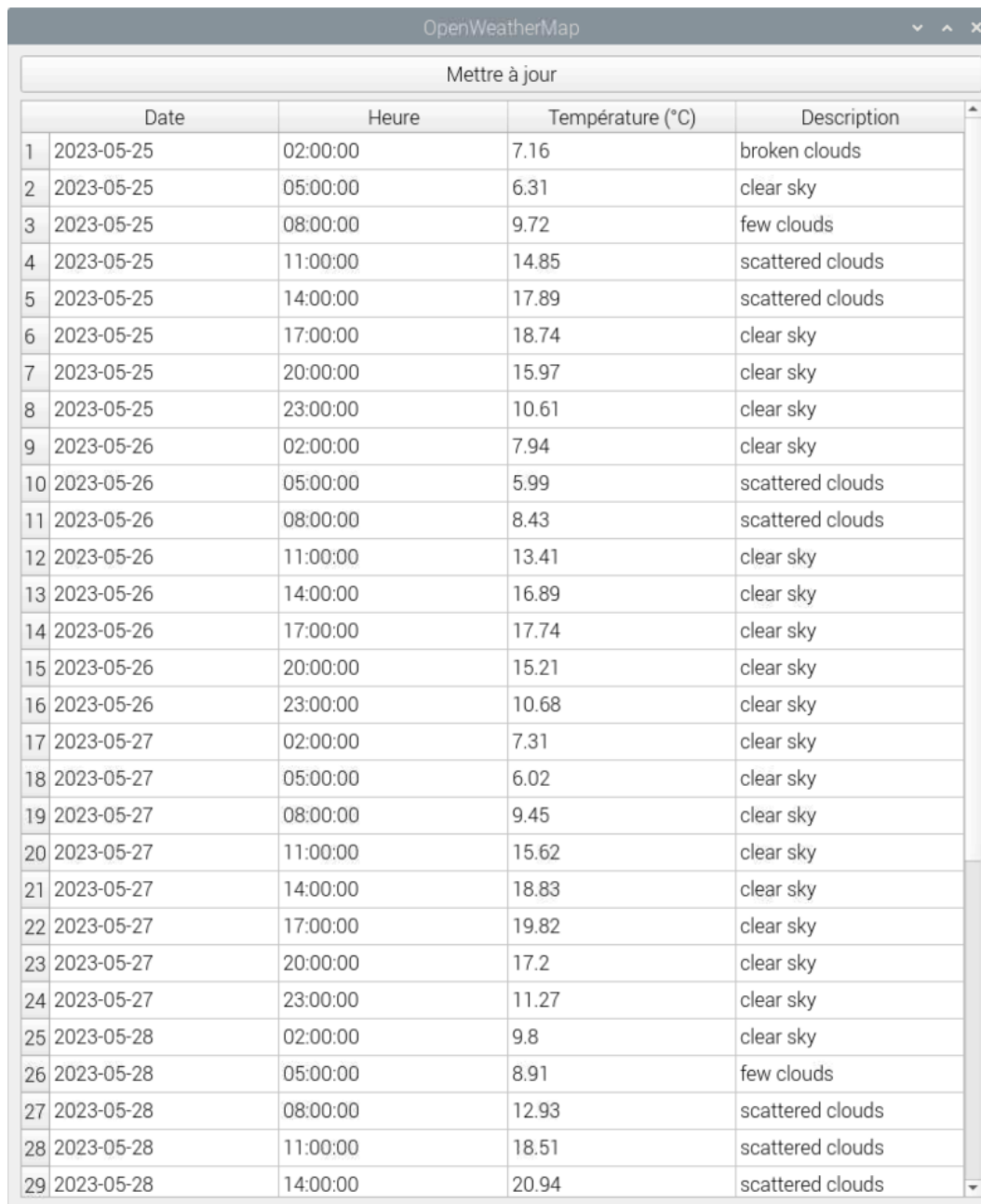
        #ensuite cette boucle parcourt chaque jour et chaque élément de la
liste des données
        for day in next_5_days:
            for item in data['list']: # fix 2
                item_date = datetime.fromtimestamp(item['dt']) #La date de
l'élément est convertie en objet datetime à l'aide de datetime.fromtimestamp()
                #Si la date correspond au jour en cours, les données sont
ajoutées aux cellules appropriées du tableau. L'indice de ligne est incrémenté
à chaque itération
                if item_date.date().strftime('%Y-%m-%d') == day:
                    date_item =
QTableWidgetItem(item_date.date().strftime('%Y-%m-%d'))
                    date_item.setFlags(date_item.flags() ^
Qt.ItemIsEditable)
                    self.table.setItem(row, 0, date_item)
                    time_item =
QTableWidgetItem(item_date.time().strftime('%H:%M:%S'))
                    time_item.setFlags(time_item.flags() ^
Qt.ItemIsEditable)
                    self.table.setItem(row, 1, time_item)
                    temp_item =
QTableWidgetItem(str(item['main']['temp']))
                    temp_item.setFlags(temp_item.flags() ^
Qt.ItemIsEditable)
                    self.table.setItem(row, 2, temp_item)
                    desc_item =
QTableWidgetItem(item['weather'][0]['description'])
                    desc_item.setFlags(desc_item.flags() ^
Qt.ItemIsEditable)
                    self.table.setItem(row, 3, desc_item)
                    row += 1

app = QApplication([])
window = MainWindow('Liège')
window.show() #on affiche la fenêtre
app.exec_() #on execute l'application

```

[UART/UARTreception.py at main · hepl-hasiuk/UART \(github.com\)](#)

Résultat :



The screenshot shows a window titled "OpenWeatherMap" with a "Mettre à jour" button at the top. Below the button is a table with four columns: "Date", "Heure", "Température (°C)", and "Description". The table contains 29 rows of data, starting from 2023-05-25 and ending on 2023-05-28. The temperature values range from 6.02 to 20.94 °C, and the descriptions include "broken clouds", "clear sky", "few clouds", and "scattered clouds".

| | Date | Heure | Température (°C) | Description |
|----|------------|----------|------------------|------------------|
| 1 | 2023-05-25 | 02:00:00 | 7.16 | broken clouds |
| 2 | 2023-05-25 | 05:00:00 | 6.31 | clear sky |
| 3 | 2023-05-25 | 08:00:00 | 9.72 | few clouds |
| 4 | 2023-05-25 | 11:00:00 | 14.85 | scattered clouds |
| 5 | 2023-05-25 | 14:00:00 | 17.89 | scattered clouds |
| 6 | 2023-05-25 | 17:00:00 | 18.74 | clear sky |
| 7 | 2023-05-25 | 20:00:00 | 15.97 | clear sky |
| 8 | 2023-05-25 | 23:00:00 | 10.61 | clear sky |
| 9 | 2023-05-26 | 02:00:00 | 7.94 | clear sky |
| 10 | 2023-05-26 | 05:00:00 | 5.99 | scattered clouds |
| 11 | 2023-05-26 | 08:00:00 | 8.43 | scattered clouds |
| 12 | 2023-05-26 | 11:00:00 | 13.41 | clear sky |
| 13 | 2023-05-26 | 14:00:00 | 16.89 | clear sky |
| 14 | 2023-05-26 | 17:00:00 | 17.74 | clear sky |
| 15 | 2023-05-26 | 20:00:00 | 15.21 | clear sky |
| 16 | 2023-05-26 | 23:00:00 | 10.68 | clear sky |
| 17 | 2023-05-27 | 02:00:00 | 7.31 | clear sky |
| 18 | 2023-05-27 | 05:00:00 | 6.02 | clear sky |
| 19 | 2023-05-27 | 08:00:00 | 9.45 | clear sky |
| 20 | 2023-05-27 | 11:00:00 | 15.62 | clear sky |
| 21 | 2023-05-27 | 14:00:00 | 18.83 | clear sky |
| 22 | 2023-05-27 | 17:00:00 | 19.82 | clear sky |
| 23 | 2023-05-27 | 20:00:00 | 17.2 | clear sky |
| 24 | 2023-05-27 | 23:00:00 | 11.27 | clear sky |
| 25 | 2023-05-28 | 02:00:00 | 9.8 | clear sky |
| 26 | 2023-05-28 | 05:00:00 | 8.91 | few clouds |
| 27 | 2023-05-28 | 08:00:00 | 12.93 | scattered clouds |
| 28 | 2023-05-28 | 11:00:00 | 18.51 | scattered clouds |
| 29 | 2023-05-28 | 14:00:00 | 20.94 | scattered clouds |

Voici le résultat obtenu à partir du deuxième code lancé sur Thonny. La page pour afficher le tableau s'ouvre correctement une fois les données reçues. Grace au bouton mise à jour sur le dessus du tableau on est capable d'afficher les données comme vu sur la capture d'écran ci-dessus.

Conclusion

Dans le cadre de ce laboratoire, nous avons réussi à mettre en place un système opérationnel de communication UART avec un Raspberry Pi. Ce système permet de transférer les données météorologiques collectées depuis OpenWeatherMap d'un code à l'autre via l'UART. Les données sont ensuite affichées sur une interface graphique Qt. Ce système offre la possibilité de collecter et d'afficher en temps réel les informations météorologiques d'une région spécifique.