

Progetto IoT

Luigi Borriello(0000933539)
Emanuele Orlietti(0000921418)

14 maggio 2021

Indice

1	Introduzione	2
2	Descrizione	3
2.1	Client(Smart Meter IoT)	3
2.2	Gateway	5
2.3	Server Cloud	6
3	Funzionamento	8
4	Librerie utilizzate	11

Capitolo 1

Introduzione

Il progetto consiste nel realizzare un programma con il linguaggio Python che permetta di realizzare una connessione client-server sfruttando i socket TCP, e una connessione client-server utilizzando i socket UDP.

Capitolo 2

Descrizione

2.1 Client(Smart Meter IoT)

I client (dispositivi IoT) sono stati realizzati su 4 moduli separati (**Device1.py**, **Device2.py**,...), i quali però sfruttano in comune delle funzioni definite in un modulo a parte, scritto appositamente.

Nello specifico, il modulo **IoT_Functions.py** contiene due funzioni:

- **readDetections**(client_ip, fileName): dato un indirizzo IP passato in input (*client_ip*), va a leggere i dati nel file contenente le rilevazioni (*fileName*), aggiungendo nella testa di ogni riga, l'indirizzo IP passato, nonché quello dell' IoT specifico. Viene preparato quindi il messaggio, che verrà restituito al chiamante.
- **connectToGateway**(gateway_address, message): data la tupla *gateway_address* passata in input, viene aperta una connessione con il Gateway sfruttando i socket UDP, inviando poi il messaggio (*message*) ricevuto anch'esso dal chiamante (IoT specifico).
Attende poi una risposta dal Gateway e, una volta arrivata, chiude la connessione. La funzione stampa a video sia le dimensioni del buffer utilizzato, che il tempo di trasmissione del pacchetto.

```

def readDetections(client_ip, fileName):
    message = ""
    filePath = "Detections/" + fileName
    file = open(filePath, "r")
    print("Reading the detections from file...")
    time.sleep(2)

    while True:
        # Get next line from file
        line = file.readline()
        # formatting the message
        if(line != ""):
            message = message + client_ip + " - " + line + "\n"
            # if line is empty
            # end of file is reached
        if not line:
            break

    file.close()
    print("Detections readed correctly!")
    return message

```

Figura 2.1: Codice della funzione `readDetections()` del modulo `IoT_Functions.py`

```

# Send info to Gateway
def connectToGateway(gateway_address, message):
    # Create the UDP socket
    sock = sk.socket(sk.AF_INET, sk.SOCK_DGRAM)
    buffer = 4096
    try:
        print("Sending info to Gateway on interface 192.168.1.0...")
        time.sleep(2)
        startTime = time.time()

        sent = sock.sendto(message.encode(), gateway_address)
        print("Waiting the Gateway response...")
        data, server = sock.recvfrom(buffer)
        # Calculate the time to send the message
        finalTime = time.time() - startTime
        time.sleep(2)
        print("Received Message: {}".format(data.decode("utf8")))
        print("UDP message's sending time {} and the size of used buffer is {}".format(finalTime, buffer))
    except Exception as info:
        print(info)
    finally:
        print("Closing Socket")
        sock.close()

```

Figura 2.2: Codice della funzione `connectToGateway()` del modulo `IoT_Functions.py`; `sk` è l'alias assegnato al modulo `socket`.

```

import IoT_Functions as iotF

# Specific IoT informations
client_ip = "192.168.1.2"
fileName = "DetectionsC1.txt"
gateway_address = ("localhost", 10003)
message = iotF.readDetections(client_ip, fileName)
iotF.connectToGateway(gateway_address, message)

```

Figura 2.3: Codice del modulo **Device1.py**

2.2 Gateway

Il gateway, realizzato nel modulo(**Gateway.py**) inizialmente si mette in ascolto sull'interfaccia di rete relativa ai device e attende le rilevazioni dei 4 device. Una volta che ha ottenuto tutte le rilevazioni, apre la connessione TCP con il cloud, aspetta che il cloud conferma la ricezione delle rilevazioni e stampa il tempo impiegato dalla connessione.

```

socket_device = sk.socket(sk.AF_INET, sk.SOCK_DGRAM)
socket_device.bind(("localhost", 10003))
print('Listening on the device interface')

# Waiting devices's detections
for i in range(4):
    data, address = socket_device.recvfrom(4096)
    print(address)
    message = message + data.decode("utf8") + '\n'
    time.sleep(2)
    messageReply = "Detection arrived"
    socket_device.sendto(messageReply.encode(), address)

socket_device.close()
print("Detections arrived. Open connection interface 10.10.10.0")

```

Figura 2.4: Codice del Gateway relativo alla connessione UDP con i device

```

# Device detections have arrived and sending everything to the cloud
socket_cloud = sk.socket(sk.AF_INET, sk.SOCK_STREAM)
socket_cloud.connect(('localhost', 8002))
startTime = time.time()
socket_cloud.send(message.encode())
buffer = 4096
data = socket_cloud.recv(buffer)
print("Waiting the server's response...")
finalTime = startTime - time.time()
print("Received Message: {}".format(data.decode("utf8")))
print("TCP message's sending time {} and the size of used buffer is {}".format(finalTime, buffer))
print("Closing connection")
socket_cloud.close()

```

Figura 2.5: Codice del Gateway relativo alla connessione TCP con il cloud

2.3 Server Cloud

Il server cloud, realizzato interamente in un unico modulo (**Cloud.py**) non fa altro che aprire una connessione tramite socket TCP nell'interfaccia "10.10.10.0", mettersi in ascolto, aspettare il messaggio dal Gateway e una volta arrivato, invia un messaggio di risposta e stampa le informazioni ricevute su Console.

```

import socket as sk

def connectToGateway():
    # TCP Server socket
    serverSocket = sk.socket(sk.AF_INET, sk.SOCK_STREAM)
    # Bind Socket to port & ip
    serverSocket.bind(("localhost", serverPort))

    # Listen Client request
    serverSocket.listen(1)
    print("SERVER CLOUD")
    print("Waiting on interface 10.10.10.0 on port {}".format(serverPort))

    # Waiting Gateway connection
    gatewayConnection, address = serverSocket.accept()
    print("Gateway connected!")
    print("Detections")
    message = gatewayConnection.recv(bufferSize)
    print(message.decode("utf8"))
    gatewayConnection.send(("Ok, detections received!").encode())
    gatewayConnection.close()
    serverSocket.close()

# TCP Server port and ip
serverPort = 8002
serverIp = '10.10.10.2'
bufferSize = 4096
connectToGateway()

```

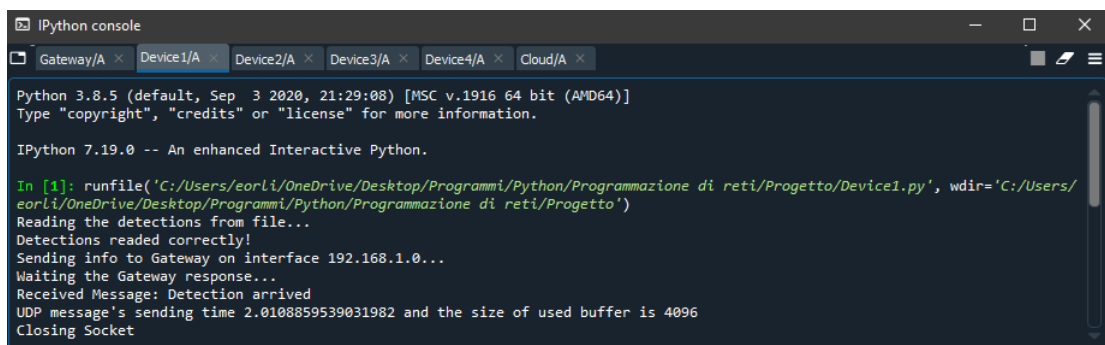
Figura 2.6: Codice del modulo **Cloud.py**

Capitolo 3

Funzionamento

- Per quanto riguarda il funzionamento dovremo innanzitutto avviare il Gateway che si metterà in ascolto sull'interfaccia relativa ai device.
- Successivamente avvieremo i 4 device che invieranno le rivelazioni al Gateway.
- Una volta ricevute tutte le rivelazioni, il Gateway chiuderà il socket con i device e cercherà di mettersi in contatto con il Cloud che dovrà essere già in ascolto.
- Una volta che il Cloud avrà ricevuto il tutto, stamperà le rivelazioni e manderà il messaggio di conferma ricezione al Gateway.

All'interno c'è anche il calcolo del tempo impiegato per le due connessioni, TCP e UDP.

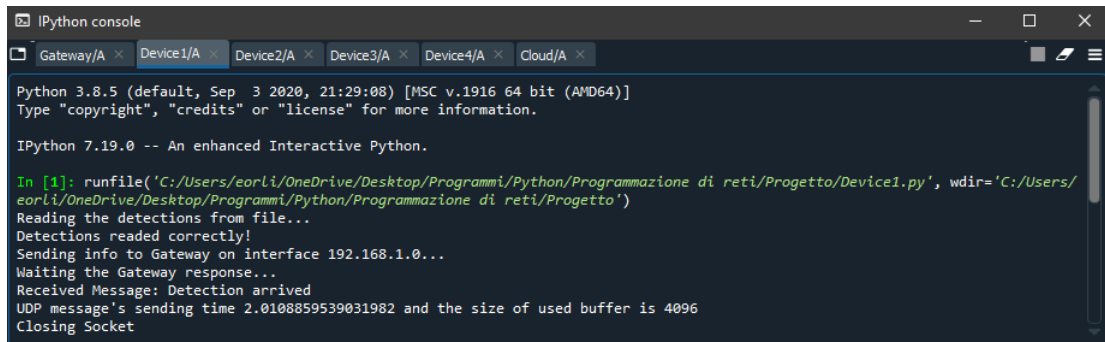


```
IPython console
Gateway/A x Device1/A x Device2/A x Device3/A x Device4/A x Cloud/A x
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/eorLi/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto/Device1.py', wdir='C:/Users/
eorLi/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto')
Reading the detections from file...
Detections readed correctly!
Sending info to Gateway on interface 192.168.1.0...
Waiting the Gateway response...
Received Message: Detection arrived
UDP message's sending time 2.0108859539031982 and the size of used buffer is 4096
Closing Socket
```

Figura 3.1: Console **Device 1**

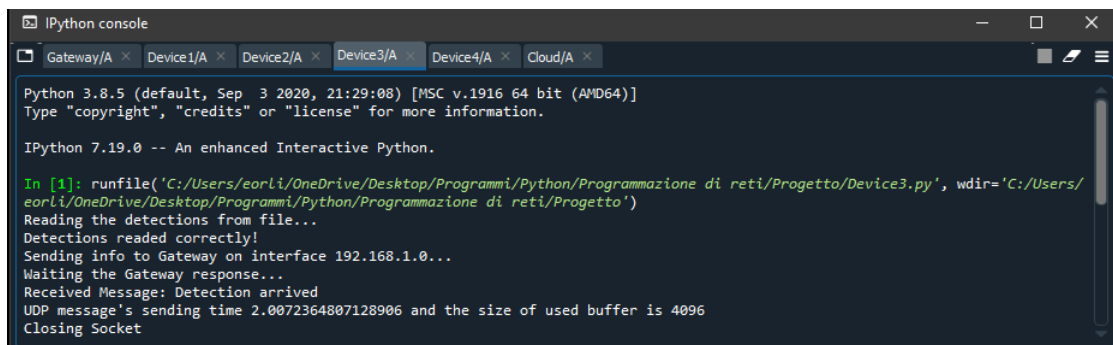
The screenshot shows an IPython console window with tabs for Gateway/A, Device1/A, Device2/A, Device3/A, Device4/A, and Cloud/A. The Device2/A tab is active. The console output shows the execution of a Python script that reads detections from a file, sends information to a Gateway on interface 192.168.1.0, waits for a response, receives a message indicating a detection arrived, and then closes the socket. The UDP message's sending time is 2.0108859539031982 and the size of the used buffer is 4096.

```
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/eorli/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto/Device1.py', wdir='C:/Users/
eorli/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto')
Reading the detections from file...
Detections readed correctly!
Sending info to Gateway on interface 192.168.1.0...
Waiting the Gateway response...
Received Message: Detection arrived
UDP message's sending time 2.0108859539031982 and the size of used buffer is 4096
Closing Socket
```

Figura 3.2: Console **Device 2**

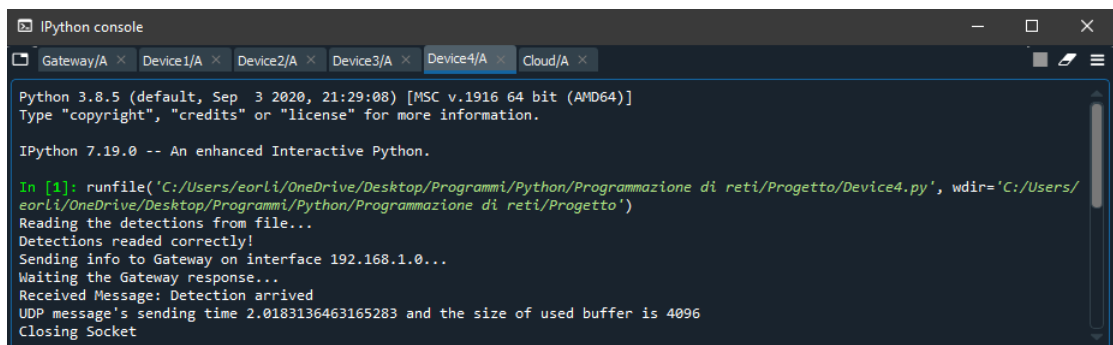
The screenshot shows an IPython console window with tabs for Gateway/A, Device1/A, Device2/A, Device3/A, Device4/A, and Cloud/A. The Device3/A tab is active. The console output shows the execution of a Python script that reads detections from a file, sends information to a Gateway on interface 192.168.1.0, waits for a response, receives a message indicating a detection arrived, and then closes the socket. The UDP message's sending time is 2.0072364807128906 and the size of the used buffer is 4096.

```
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/eorli/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto/Device3.py', wdir='C:/Users/
eorli/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto')
Reading the detections from file...
Detections readed correctly!
Sending info to Gateway on interface 192.168.1.0...
Waiting the Gateway response...
Received Message: Detection arrived
UDP message's sending time 2.0072364807128906 and the size of used buffer is 4096
Closing Socket
```

Figura 3.3: Console **Device 3**

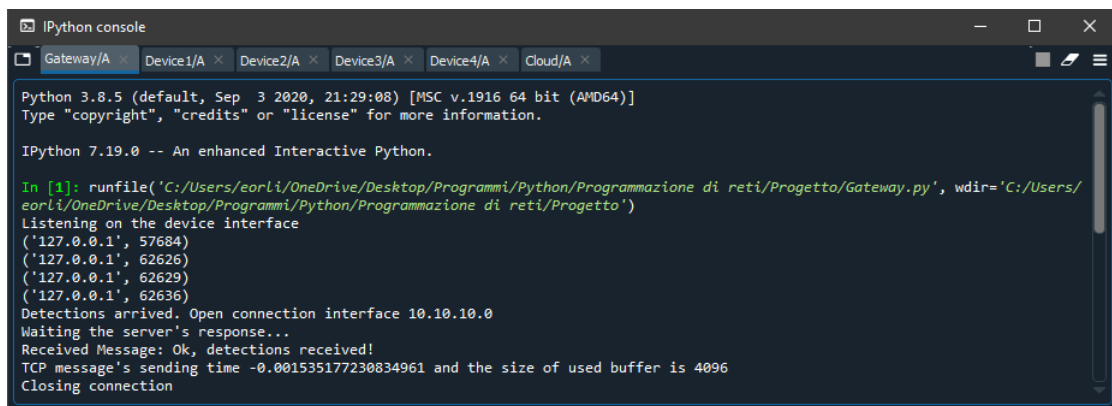
The screenshot shows an IPython console window with tabs for Gateway/A, Device1/A, Device2/A, Device3/A, Device4/A, and Cloud/A. The Device4/A tab is active. The console output shows the execution of a Python script that reads detections from a file, sends information to a Gateway on interface 192.168.1.0, waits for a response, receives a message indicating a detection arrived, and then closes the socket. The UDP message's sending time is 2.0183136463165283 and the size of the used buffer is 4096.

```
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/eorli/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto/Device4.py', wdir='C:/Users/
eorli/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto')
Reading the detections from file...
Detections readed correctly!
Sending info to Gateway on interface 192.168.1.0...
Waiting the Gateway response...
Received Message: Detection arrived
UDP message's sending time 2.0183136463165283 and the size of used buffer is 4096
Closing Socket
```

Figura 3.4: Console **Device 4**

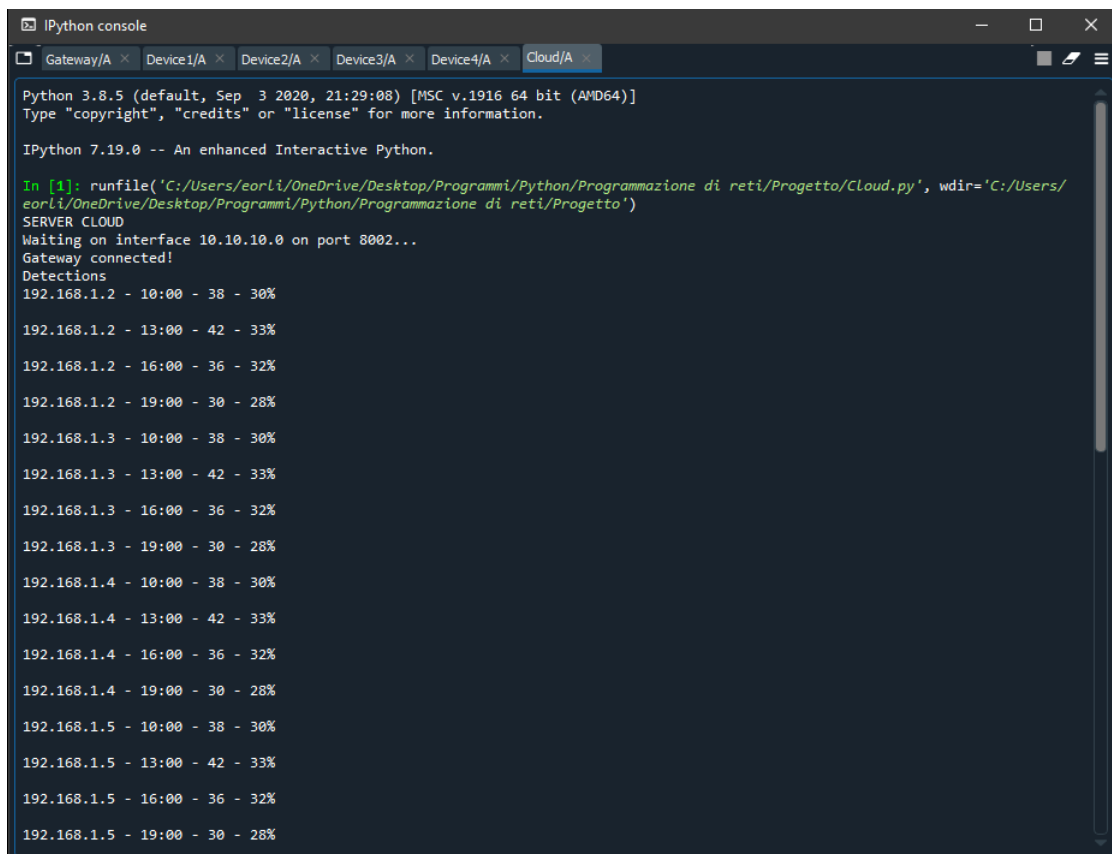


```
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/eorLi/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto/Gateway.py', wdir='C:/Users/
eorLi/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto')
Listening on the device interface
('127.0.0.1', 57684)
('127.0.0.1', 62626)
('127.0.0.1', 62629)
('127.0.0.1', 62636)
Detections arrived. Open connection interface 10.10.10.0
Waiting the server's response...
Received Message: Ok, detections received!
TCP message's sending time -0.001535177230834961 and the size of used buffer is 4096
Closing connection
```

Figura 3.5: Console Gateway



```
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/eorLi/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto/Cloud.py', wdir='C:/Users/
eorLi/OneDrive/Desktop/Programmi/Python/Programmazione di reti/Progetto')
SERVER CLOUD
Waiting on interface 10.10.10.0 on port 8002...
Gateway connected!
Detections
192.168.1.2 - 10:00 - 38 - 30%
192.168.1.2 - 13:00 - 42 - 33%
192.168.1.2 - 16:00 - 36 - 32%
192.168.1.2 - 19:00 - 30 - 28%
192.168.1.3 - 10:00 - 38 - 30%
192.168.1.3 - 13:00 - 42 - 33%
192.168.1.3 - 16:00 - 36 - 32%
192.168.1.3 - 19:00 - 30 - 28%
192.168.1.4 - 10:00 - 38 - 30%
192.168.1.4 - 13:00 - 42 - 33%
192.168.1.4 - 16:00 - 36 - 32%
192.168.1.4 - 19:00 - 30 - 28%
192.168.1.5 - 10:00 - 38 - 30%
192.168.1.5 - 13:00 - 42 - 33%
192.168.1.5 - 16:00 - 36 - 32%
192.168.1.5 - 19:00 - 30 - 28%
```

Figura 3.6: Console Cloud

Capitolo 4

Librerie utilizzate

- time
- socket