IoT Off-Grid, secure data collection from a machine-learning classification using UAV

Abstract

IoT encompasses various objects, technologies, communication standards, sensors, actuators in powered environments, and networked communication. The concept adopted here, IoT Off-Grid, considers an environment without commercial electricity and commercial internet. Managing various utilities with IoT and collecting the relevant information from this environment is the purpose of this project. It uses machine learning to sort and select relevant data. This data is being collected safely using a Drone that travels through the Off-Grid stations. A systematic literature mapping is presented, identifying state of the art. The result is a software architecture proposal, with configurations in the Drone and Off-Grid stations that contemplate security during data collection from the IoT Off-Grid environment. The results are also presented with different selection algorithms used in machine learning and final execution in the prototype.

Summary

Scenery

To validate the procedures reported in this article, the following scenario was used:

DJI AIR 2S drone with a 5000mAh KAI DI model KD-905 power bank attached to its body, powering a Raspberry Pi Zero 2 W and responsible for collecting data from remote stations.

Three different off-grid stations with data to be collected as described below.

Station 1 – consists of Raspberry Pi 3 Model B hardware, 1GB RAM, wireless 802.11 b/g/n configured to be an access point with SSID offg01, and powered by a 20000mAh PINENG model PN-999 power bank.

Station 2 – consists of Raspberry Pi 4 Model B hardware, 8GB RAM, wireless 802.11 b/g/n/ac configured to be an access point with SSID offg02, and powered by a 10000mAh GEONAV model PB10KBK power bank.

Station 3 – consists of Raspberry Pi 4 Model B hardware, 8GB RAM, wireless 802.11 b/g/n/ac configured to be an access point with SSID offg03 and powered by an INOVA model PB03 5000mAh power bank.

Configuration for each Off-Grid Station

Each of the stations had its configuration to be enabled as an access point according to the documentation found on the Raspberrypi documentation page in the following link:

Setting up a Routed Wireless Access Point

https://www.raspberrypi.com/documentation/computers/configuration.html#setting-up-a-routed-wireless-access-point

Station 01 configuration

- SSID – offg01
- Password – 100password
- Network 192.168.4.0/24
- DHCP 192.168.4.100, 192.168.4.110
- Radio channel 7

Station configuration 02

- SSID – offg02
- Password – 100password
- Network 192.168.5.0/24
- DHCP 192.168.5.100, 192.168.5.110
- Radio channel 1

Station configuration 03

- SSID – offg03
- Password – 100password
- Network 192.168.6.0/24
- DHCP 192.168.6.100, 192.168.6.110
- Radio channel 3

For each station, the files with the data collected in the IoT Off-Grid environment must be placed in a directory that will be entirely copied to the drone's micro.

In the configuration of this work, the files were located in /home/pi/archives.

Configuration in the micro coupled to the drone

Wireless network configuration

This Raspberry Pi Zero 2W attached to the drone automatically detects the wireless network of each of the Access Points of the off-grid stations. To allow the drone's micro to identify the SSID of each station, it is necessary to configure WPA_SUPPLICANT as follows:

Edit the /etc/wpa_supplicant/wpa_supplicant.conf file, including the SSIDs of each station.

```
network={
  ssid="offg01"
  psk="100password"
}
network={
  ssid="offg02"
  psk="100password"
}
network={
  ssid="offg03"
  psk="100password"
```

```
        }
```

A link to more details of this configuration:

https://www.raspberrypi-spy.co.uk/2017/04/manually-setting-up-pi-wifi-using-wpa_supplicant-conf/

According to the drone's proximity to the station, when identifying the SSID, the connection is established. Thus, the drone's micro is associated with the same network as the station, which allows the automatic transfer of files from the station to the drone's micro.

Data transfer configuration

Configure SSH

The transfer of files from the station to the micro attached to the drone will be via SSH, and so it is necessary to allow the use of SSH on port 22 on the station via raspberry settings.

Using Preferences / Raspberry settings / interfaces it is possible to enable SSH.

For the SSH connection to occur without the need to provide a username and password, it is necessary to enable login without a password using local/remote keys.

On the station, create an ssh key with

```
ssh-keygen -t rsa -b 4096 -C your_email@domain.com
```

Copy this key to the micro that is attached to the drone

```
ssh-copy-id remote_username@server_ip_address
```

documentation of this procedure can be found at:

https://linuxize.com/post/how-to-setup-passwordless-ssh-login/

File transfer script

The script in python sc06.py to identify if the micro attached to the drone is viewing the wireless network of each station and thus start the transfer of files is as follows:

```
import os

import time

from datetime import datetime

hostname1 = "192.168.4.1" #  primeiro no

hostname2 = "192.168.5.1"  # Segundo no

hostname3 = "192.168.6.1"  # Terceiro no

i = 1

n1 = 0

n2 = 0

n3 = 0

#and then check the response...

while i > 0 :

        response = os.system("ping -c 1 " + hostname1)

        if response == 0 and n1 == 0:

                print (hostname1, 'is up!')

                print ("vai chamar scp")

                data_hora_atual =  datetime.now()

                dh_texto = data_hora_atual.strftime ('%d/%m/%Y  %H:%M')

                print(dh_texto)

                resp= os.system("scp -p pi@192.168.4.1:/home/pi/arquivos/*.*
/home/pi/Doutorado/no_01/" )

                data_hora_atual =  datetime.now()

                dh_texto = data_hora_atual.strftime ('%d/%m/%Y  %H:%M')

                print(dh_texto)

                n1=1



        response = os.system("ping -c 1 " + hostname2)
        if response == 0 and n2 == 0:
                print (hostname2, 'is up!')
                print ("vai chamar scp")
                data_hora_atual =  datetime.now()
                dh_texto = data_hora_atual.strftime ('%d/%m/%Y  %H:%M')
                print(dh_texto)
                resp= os.system("scp -p pi@192.168.5.1:/home/pi/arqs_no_02/*.*
/home/pi/Doutorado/no_02/" )
                data_hora_atual =  datetime.now()
                dh_texto = data_hora_atual.strftime ('%d/%m/%Y  %H:%M')
                print(dh_texto)
                n2=1


        response = os.system("ping -c 1 " + hostname3)
        if response == 0 and n3 == 0:
                print (hostname3, 'is up!')
                print ("vai chamar scp")
                data_hora_atual =  datetime.now()
                dh_texto = data_hora_atual.strftime ('%d/%m/%Y  %H:%M')
                print(dh_texto)
                resp= os.system("scp -p pi@192.168.6.1:/home/pi/arquivos/*.*
/home/pi/Doutorado/no_03/" )
                data_hora_atual =  datetime.now()
                dh_texto = data_hora_atual.strftime ('%d/%m/%Y  %H:%M')
                print(dh_texto)
                n3=1
        print ('intervalo de espera 20 segundos')
        time.sleep(20)
```

This script, to be executed automatically after the boot of this micro that is coupled to the drone, can be configured via crontab with the command

```
crontab crontabe -e
```

```
@reboot /usr/local/src/start.sh
```

So the start.sh script has the following content:

Creating script to clean file*.* and call program that copies in

/usr/local/src/start.sh (with chmod 777)

```
#!/usr/bin/bash
CD /home/pi/Doctorate
Rm file*.*
Python sc06.py >> /home/pi/log
```

Documentation on how to edit crontab:

https://www.raspberrypi.com/documentation/computers/using_linux.html#editing-the-crontab-file