https://www.overleaf.com/project/62c51a29fc44df0f8e7c8f8f

VIETNAMESE - GERMAN UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEER

Information Technology Project Report

Student 1:	Nguyen Nhut Thanh - 10421104
Student 2:	Nguyen Khoi Nguyen - 10421097
Student 3:	Vo Thi Hong Ha - 10421015
Student 4:	Luu Minh Khang - 10421024
Student 5:	Nguyen Ho Tan Dat - 10421071
Student 6:	Huynh Minh Tuong - 10421117

07 / 2022

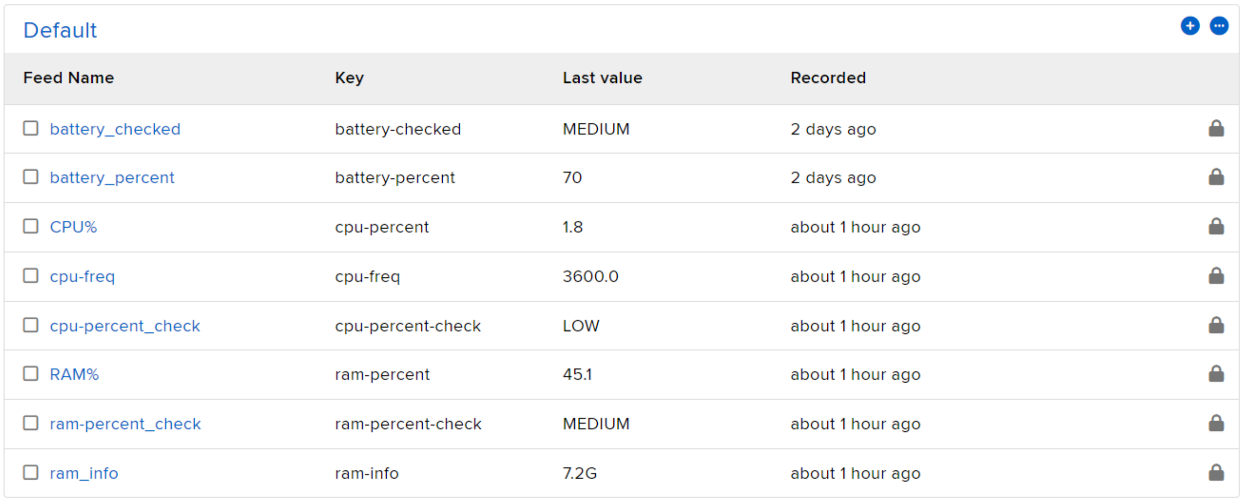
Content

1 Introduction

Our group decided to implement the IoT Gateway Python for the project of the Information Technology course. Briefly explain, the IoT Gateway gains access to the hardware information, sends that data to Adafruit IO Feeds and then displays it on the dashboard.

2 IoT Gateway Python

2.1 Adafruit IO Feeds



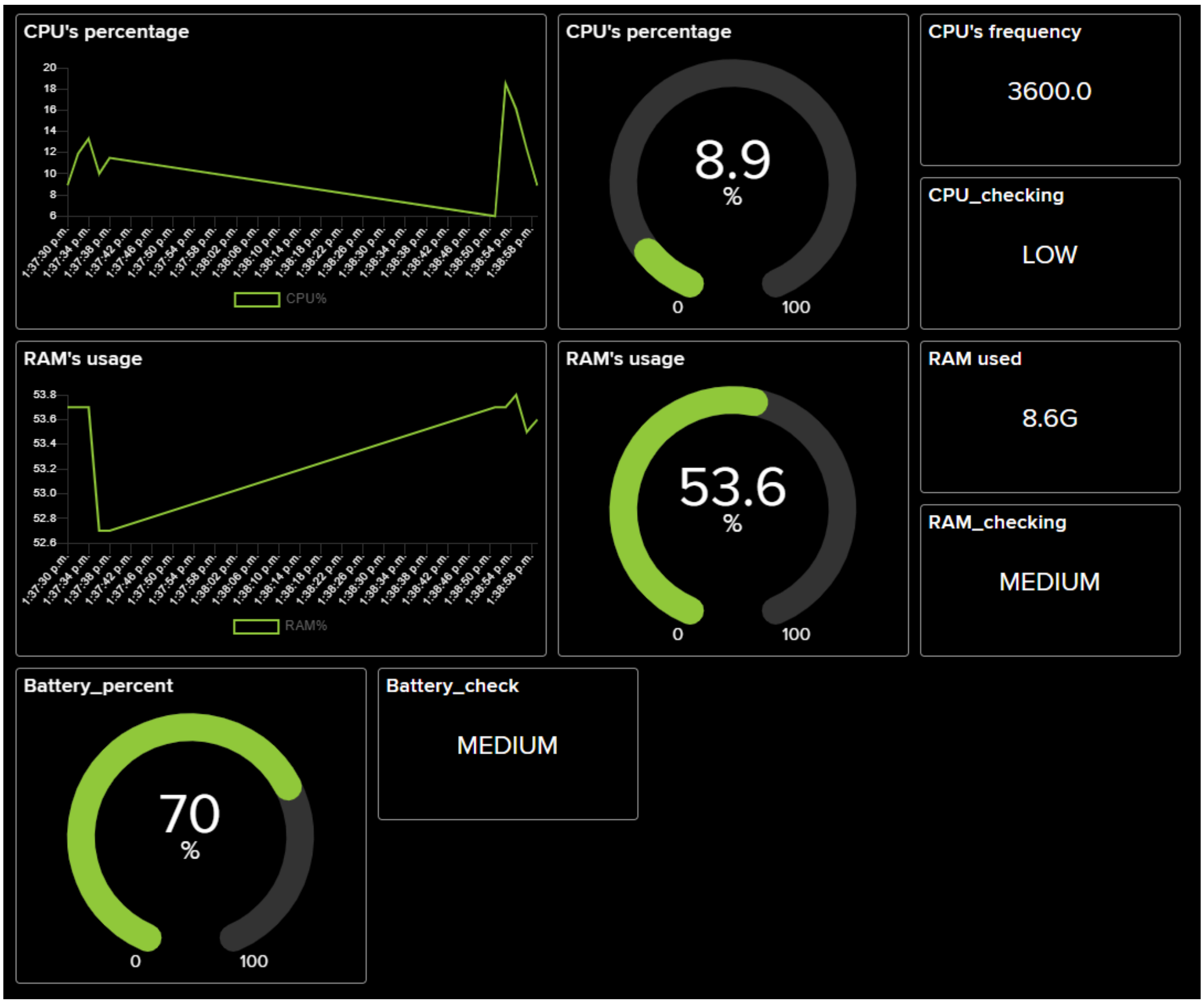
Default

Feed Name	Key	Last value	Recorded
battery_checked	battery-checked	MEDIUM	2 days ago
battery_percent	battery-percent	70	2 days ago
CPU%	cpu-percent	1.8	about 1 hour ago
cpu-freq	cpu-freq	3600.0	about 1 hour ago
cpu-percent_check	cpu-percent-check	LOW	about 1 hour ago
RAM%	ram-percent	45.1	about 1 hour ago
ram-percent_check	ram-percent-check	MEDIUM	about 1 hour ago
ram_info	ram-info	7.2G	about 1 hour ago

Hình 1: *Adafruit IO feeds of the project*

We create 8 feeds that support the following features:
battery_checked: to store the battery level value (low/medium/high).
battery_percent: to store the battery percent value.
CPU%: to store the CPU percent value.
cpu-freq: to store the CPU frequency value.
cpu-percent_checked: to store the CPU level value (low/medium/high).
RAM%: to store the memory percent value.
ram-percent_checked: to store the memory level value (low/medium/high).
ram_info: to store the used memory value.

2.2 Adafruit IO Dashboard



CPU's frequency
3600.0

CPU_checking
LOW

RAM used
8.6G

RAM_checking
MEDIUM

Battery_check
MEDIUM

Hình 2: *Adafruit IO dashboard of the project*

2.3 Python IOT Gateway

```python
import psutil
import time
import sys
from psutil._common import bytes2human
from Adafruit_IO import MQTTClient
```

We implement the 'psutil' library to gain access to the resources data of the system.

```python
def cpu_percentage():
    t = 0
    while (t < 10):
        t += 1
        cpu_percent = float (psutil.cpu_percent())
        client.publish("cpu-percent", cpu_percent)
        client.publish("cpu-percent_check", ("HIGH") if (cpu_percent >
            70) else (("LOW" if (cpu_percent < 30) else ("MEDIUM"))))
        time.sleep(2)
```

This function collects the CPU percent value and then publishes to the "cpu-percent" feed. It also publishes the level of CPU usage to "cpu-percent_check" feed simultaneously. The process is repeated 10 times with 2 seconds delay each.

```python
def cpu_frequency():
    t = 0
    while (t < 10):
        t += 1
        cpu_freq = psutil.cpu_freq().current
        client.publish("cpu-freq", cpu_freq)
        time.sleep(2)
```

This function collects the CPU frequency value and then publish to the "cpu-freq" feed. The process is repeated 10 times with 2 seconds delay each.

```python
def memory_percent():
    t= 0
    while ( t < 10 ):
        t+= 1
        mem_percent = psutil.virtual_memory().percent
        client.publish("ram-percent", mem_percent)
        client.publish( "ram-percent_check", ("HIGH") if (mem_percent
           > 70) else (("LOW" if (mem_percent < 30) else ("MEDIUM"))))
        time.sleep(2)
```

This function collects the memory percent value and then publishes to the "ram-percent" feed. It also publishes the level of memory usage to "ram-percent_check" feed simultaneously. The process is repeated 10 times with 2 seconds delay each.

```python
def memory_used():
    t= 0
    while ( t < 10 ):
        t+= 1
        mem_used = bytes2human(psutil.virtual_memory().used)
        client.publish("ram-info", mem_used)
```

This function collects the used memory value and then publishes to the "ram-info" feed. The process is repeated 10 times with 2 seconds delay each.

```python
def battery():
    battery = psutil.sensors_battery().percent
    client.publish("battery-percent", battery)
    client.publish("battery-checked", ("HIGH") if (battery > 70) else ↩
        (("LOW" if (battery < 30) else ("MEDIUM"))))
    time.sleep(2)
```

This function collects the battery percent value and then publishes to the "battery-percent" feed. It also publishes the level of battery to "battery-check" feed simultaneously. The process is done once per execution.

3 Conclusion

In summary, the IoT Gateway Python project includes 3 main sections: the AIO feeds, AIO dashboard and the Python Gateway with the purpose of monitoring the resource of computer hardwares relating to the CPU, memory and battery. This project is the very basic foundation for further IoT projects and research in the future.