HO CHI MINH CITY, UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEER

Application Based Internet of Things Report

Student 1: Phạm Quang Lâm - 1812778
Student 2: Võ Phi Trường - 1814582

HỒ CHÍ MINH CITY, 06/ 2021

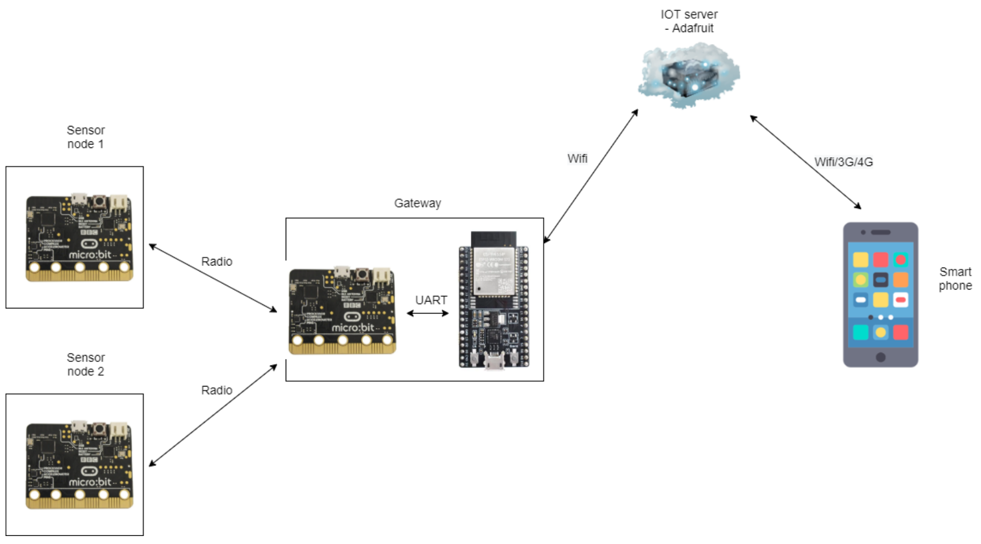
Content

1 Introduction

The system have been build base on basic architecture of an internet of things (IOT) system with: sensor node, gateway, IOT server and control device. The system is build to collect data from sensor and send data to device through IOT server. Furthermore, system also can control some aspect of sensor node (in this system is state of LED). Data and graph after collected show in device. The description each part of system is showing below:

- Sensor node: Using micro:bit as an sensor node. Collect data of temperature and light level base on on-board sensor of micro:bit and send to the gateway base on radio feature of micro:bit.
- Gateway: Include a micro:bit and a ESP32 MCU. Micro:bit to connect with sensor node base on radio feature. It's connect to ESP32 with UART protocol to transfer data. ESP32 after that, send data to IOT server through Wifi.
- IOT server: Use **Adafruit** IOT server as server of system. Gateway and device is subscribe to a feed to send and receive data.
- Device: App build for android smartphone. Display data, graph and button to control sensor node.

The summary of system is showing in this picture:



Hình 1: Structure of system

Short response time is also required. There is only one sleep(10) at the end of the infinite loop. If the sleep time is too long, for example 10 second, so in the worst situation, user interact with system when the system start to sleep, then 10 second later that system can react to user interact. So sleep(10) is for guarantee that the response time of system is 10 millisecond.

2 Sensor implementation

System is built to use multiple sensor node. 2 sensor node are used for implementation. Sensor node use micro:bit to collect data of temperature and light level and send to gateway through radio feature. Sensor send data every 10 second. The frame data format to send data is: **#x-temp-light-*** with **x** is ID number of sensor node. ID in system implementation is 1 and 2. **temp** is the temperature value of sensor. **light** is light level of sensor. The sensor is configured with MicroPython language.

The main source code of the sensor is provided below:

Listing 1: Source code of sensor node 1

```
#send data every 10 second
if (wait == 1000):
    temp = int(temperature())
    light = int(display.read_light_level())
    mess = packet(temp, light)
    radio.send(mess)
    receive_flag = False

#receive ack or control message
control_message = radio.receive()
if (control_message):
    header = decode_message(control_message)
    if (header[0] == '#a1*'):
        receive_flag = True
    elif (header[0] == '#c1'):
        if (header[1] == '10'):
            display.show(Image.SAD)
        elif (header[1] == '11'):
            display.show(Image.HAPPY)
        elif (header[1] == '20'):
            display.show(Image.YES)
        else:
            display.show(Image.NO)
    else: pass
wait = wait-1
if (wait == 0 & receive_flag == True):
    wait = 1000
elif (wait == 0):
    wait = 1 //loop to wait

#reset button
if (button_a.is_pressed()):
    wait = 1000 //reset
    sleep(10)
```

2.1 Extra function 1-Protocol using

To control the flow of sending data, **Stop-and-Wait** protocol is used. After sending a frame, the sensor node have to wait an ack frame to send continue data. If after 5 seconds from sent a frame, no ack frame received, a skull image will showed in screen to signal that no ack received. If no ack received, sensor node will not send next data frame. The format of ack message is **#ax*** with **a** is signal for ack message. **x** is the ID number of sensor node. Both sensor node 1 and 2 are received ack message and control message of each other. But it can realize its ack message and control message by ID.

2.2 Extra function 2-Reset button

Whenever button A is press, the sensor node will be reset. After sensor node can't send next data frame due to no ack received, user can press button A to reset sensor node.

2.3 Extra function 3-Control LED display

System can control LED display of micro:bit from smartphone by send frame with format: **#cx-ab-*** with c is signal for control message. **x** is ID number of sensor node. **ab** is control signal: **a** is ID of LED, **b** is state of LED. In implementation, **ab** have meaning is: 10 (turn off led1 implement by show SAD image), 11 (turn on led1 implement by show HAPPY image), 20 (turn off led2 implement by NO image), 21 (turn on led2 implement by YES image).

3 Gateway implementation

Gateway of system have function to send data from sensor node to Adafruit server and give control signal from smartphone through server and send to sensor node. Gateway is include 1 micro:bit connect UART with an ESP32 MCU. Micro:bit use Pin0 as TX and Pin1 as RX. ESP32 use Pin1 as TX ans Pin3 as RX. TX of micro:bit is connect to RX of ESP32, RX of micro:bit connect to TX of ESP32, 2 GND of micro:bit and ESP32 is connect together. Micro:bit have mission to communicate with sensor node and transfer frame to ESP32. ESP32 is communicate with server by Wifi to transfer frame.

3.1 UART received data

Listing 2: Source code for gateway-Micro:bit part

```
    if (microbit.uart.any()):
        return_message = microbit.uart.readline()
        radio.send(return_message)
```

```
  sleep(10)
```

Listing 3: Source code for gateway-ESP32 part

```
while (Serial2.available()){
  send_mess = send_mess + char(Serial2.read());
  if (Serial2.read()=="*"){
    feed.publish(send_mess);
    send_mess = "";
  }
}
```

3.2 Data uploading process

After received UART data from gateway Micro:bit, ESP32 send the frame received to server

Listing 4: Data update process

```
while (Serial2.available()){
  send_mess = send_mess + char(Serial2.read());
  if (Serial2.read()=="*"){
    feed.publish(send_mess);
    send_mess = "";
  }
}
```

3.3 Data received process

To control the LED, ESP32 is subscribe to another feed to received control signal. After received frame of control signal, ESP32 send all the frame to the sensor node. Sensor node will decode the frame for led control.

Listing 5: Data received process-ESP32 part

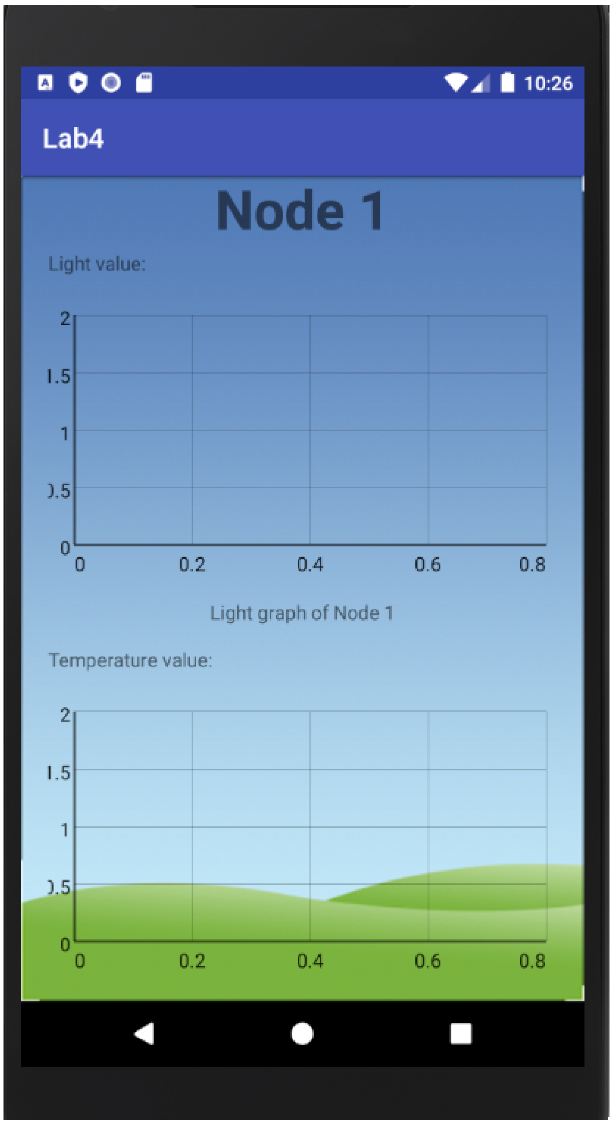
```
Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(10))) {
  if (subscription == &led_control) {
    Serial2.write((char *)led_control.lastread)
  }
}
```

Listing 6: Data received process-Micro:bit part

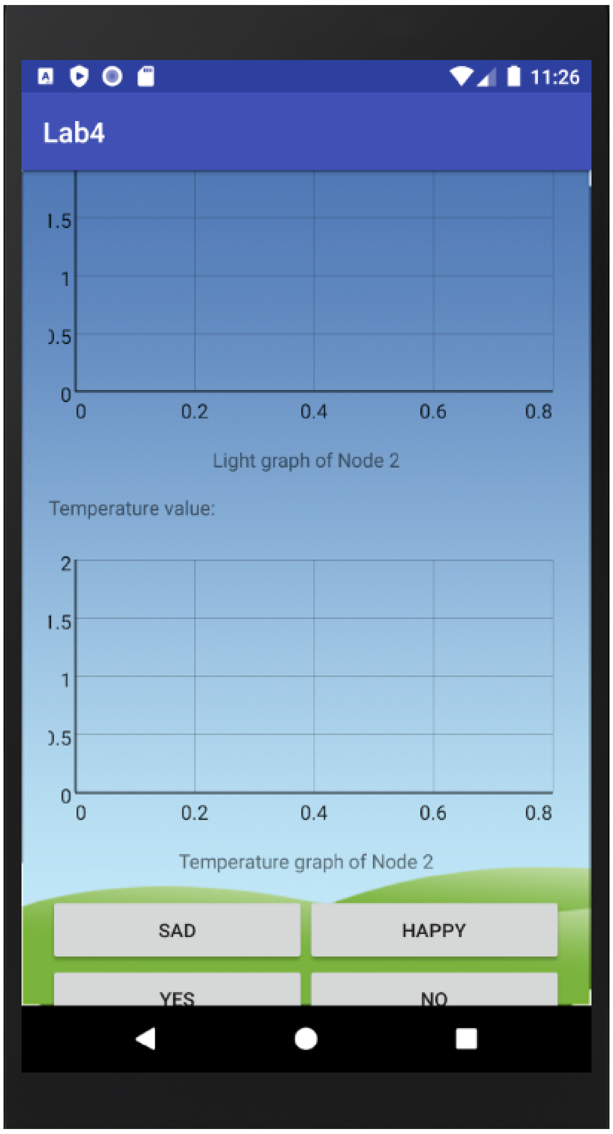
```
    if (microbit.uart.any()):
        return_message = microbit.uart.readline()
        radio.send(return_message)
```

4 Monitoring application

App on smartphone of system is built for show data from sensor node. It's can show value and graph of data collected. Button for control LED at sensor node is also added in the app.



Hình 2: App interface



Hình 3: App interface

4.1 Plus function 1 - Graph

App receives data from **Adafruit** IOT, show current value and graph of data collected into 2 line graph : **Light graph** and **Temperature graph** for each sensor node. Graph include: **x axis** is value of data and **y axis** is number of times received data.

4.2 Plus function 2 - LED control

There are buttons to control the LED of sensor. When a button is press, a corresponding signal is publish to Adafruit sensor (in a different feed of sensor data feed). Gateway received message and send to 2 sensor node. Sensor node read the message to control the LED. The format of signal message is present above.

5 Conclusion

The system is build to work correctly with the requirement and add some extra function. However, all the system is have not been built in reality due to lack of material and the hazardous situation of COVID-19. So the effective and rightness of the system have not been evaluate.

Tài liệu

[Web] Micro:bit, `https://microbit.org/`

[Web] MicroPython, `https://micropython.org/`