1167 物聯網概論(英文授課)
Introduction to Internet of Things

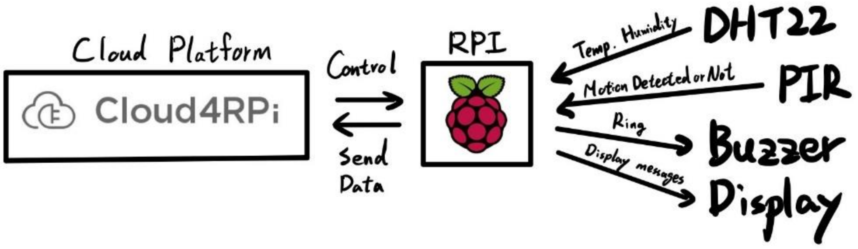
Assignment 2 Report

Student Information

- *109550073 陳宥安*
- *109550164 徐聖哲*

Objectives

The goal of the assignment is to combine the cloud platform and the RPI we have. Through cloud platform, we can monitor and record the collected data. Furthermore, controlling the RPI's actuators become possible by sending signal from the cloud.

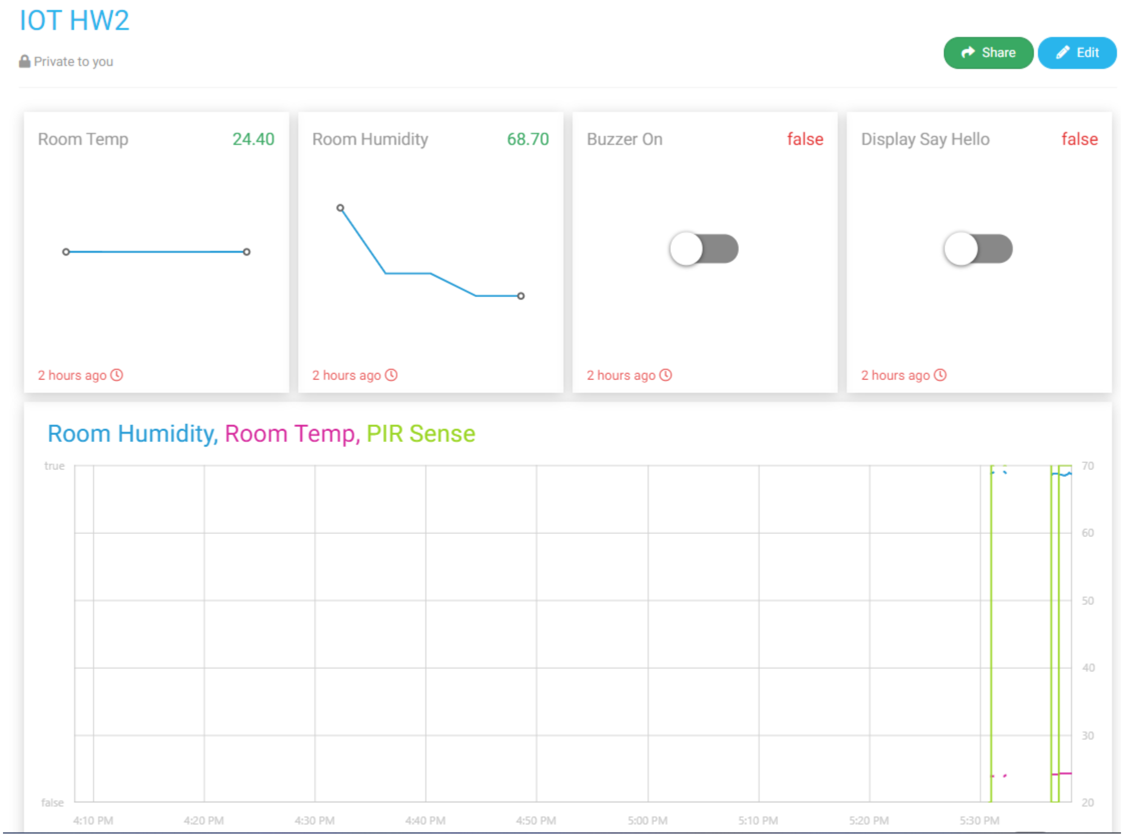


Sensor and Actuator

- Sensors
  - ■ DHT22 -> for getting real-time Temperature and Humidity
  - ■ PIR -> for Motion Detection
- Actuator
  - ■ Buzzer -> for Reminding and Alerting Users
  - ■ Display -> for Displaying Data collecting from the sensors

Project Design

- Widgets


IOT HW2

Private to you

Room Temp 24.40

2 hours ago

Room Humidity 68.70

2 hours ago

Buzzer On false

2 hours ago

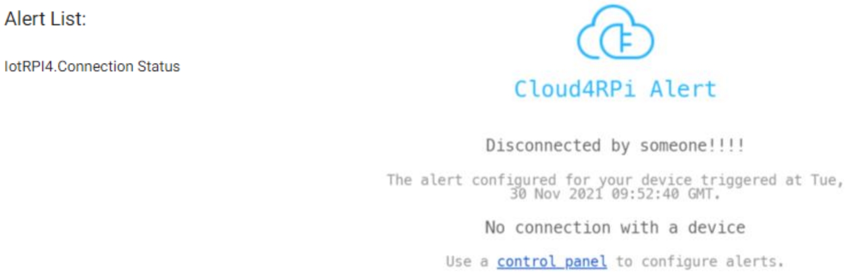
Display Say Hello false

2 hours ago

Room Humidity, Room Temp, PIR Sense

The above screenshot shows the design of the widgets in Cloud4RPi. We have two widgets that display the real-time value of temperature and humidity. Two buttons are added for control the buzzer and the display. The chart below let the users can tell the change of temperature, humidity and PIR data at a glance.

- Alert



Alert List:

IotRPI4.Connection Status

Cloud4RPi Alert

Disconnected by someone!!!!

The alert configured for your device triggered at Tue, 30 Nov 2021 09:52:40 GMT.

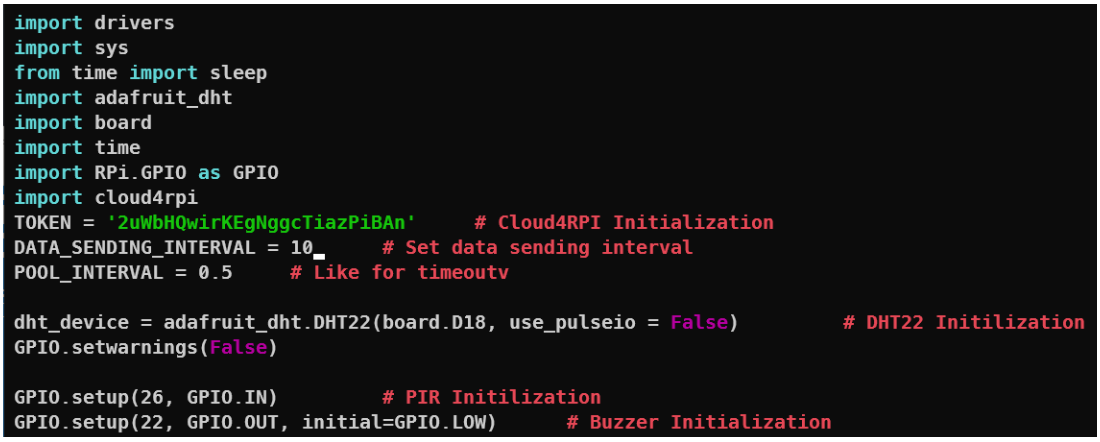
No connection with a device

Use a control panel to configure alerts.

The above screenshot indicates the alert we set for the assignment 2. If the RPI is disconnected from the Cloud4RPi, the cloud platform will raise an alert and send an email as notification.

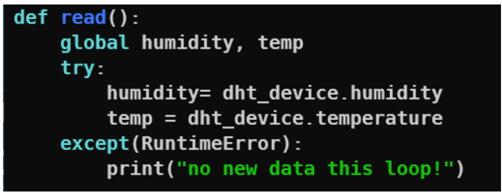
Source Code

The source code aims to uploaded the data acquiring from the sensor to the cloud platform. In this assignment, Cloud4RPi is the specified cloud platform.

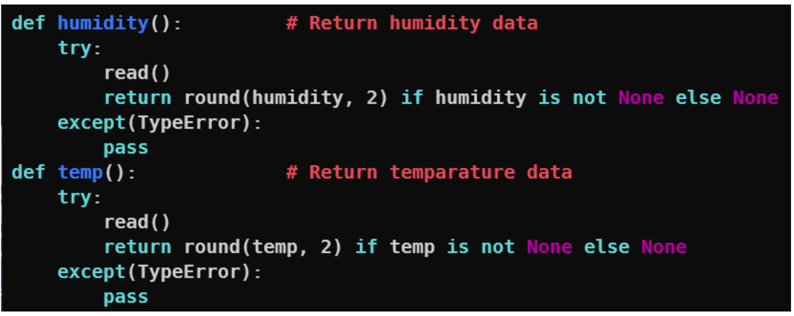


```
import drivers
import sys
from time import sleep
import adafruit_dht
import board
import time
import RPi.GPIO as GPIO
import cloud4rpi
TOKEN = '2uWbHQwirKEgNggcTiazPiBAn'     # Cloud4RPI Initialization
DATA_SENDING_INTERVAL = 10_     # Set data sending interval
POOL_INTERVAL = 0.5     # Like for timeoutv

dht_device = adafruit_dht.DHT22(board.D18, use_pulseio = False)         # DHT22 Initilization
GPIO.setwarnings(False)

GPIO.setup(26, GPIO.IN)         # PIR Initilization
GPIO.setup(22, GPIO.OUT, initial=GPIO.LOW)      # Buzzer Initialization
```

The above screenshot indicates the import libraries and the initialization of different sensors. Comparing to the assignment 1, TOKEN, DATA_SENDING_INTERVAL, and POOL_INTERVAL are added for initializing the connection between the RPI and Cloud4RPi.

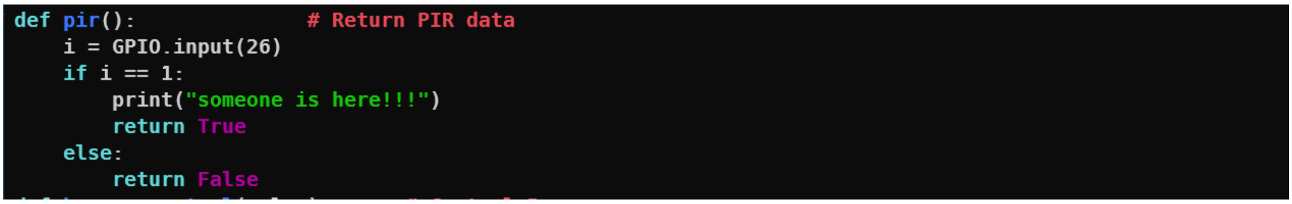


```
def read():
    global humidity, temp
    try:
        humidity= dht_device.humidity
        temp = dht_device.temperature
    except(RuntimeError):
        print("no new data this loop!")
```



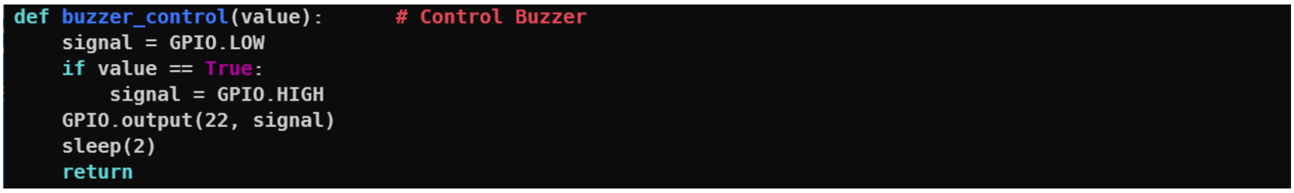
```
def humidity():         # Return humidity data
    try:
        read()
        return round(humidity, 2) if humidity is not None else None
    except(TypeError):
        pass
def temp():             # Return temparature data
    try:
        read()
        return round(temp, 2) if temp is not None else None
    except(TypeError):
        pass
```

The above screenshot shows three functions, read(), humidity(), and temp(). These three functions are used to read data collected from DHT22 and return two respective variables, humidity, and temperature, to Cloud4RPi.



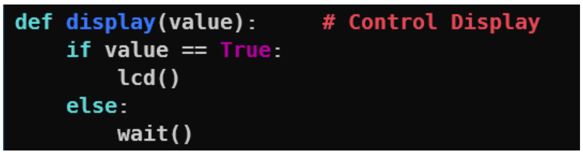
```python
def pir():                  # Return PIR data
    i = GPIO.input(26)
    if i == 1:
        print("someone is here!!!")
        return True
    else:
        return False
```

The above screenshot implies how we return the data that the PIR Sensor collected to Cloud4Rpi.

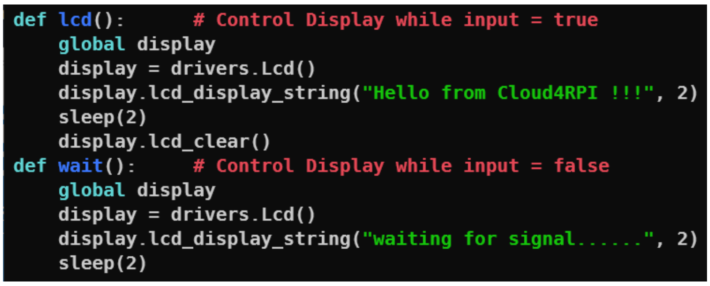


```python
def buzzer_control(value):      # Control Buzzer
    signal = GPIO.LOW
    if value == True:
        signal = GPIO.HIGH
    GPIO.output(22, signal)
    sleep(2)
    return
```

The above screenshot represents how Cloud4RPi Control the buzzer ringing by passing the boolean value of value.

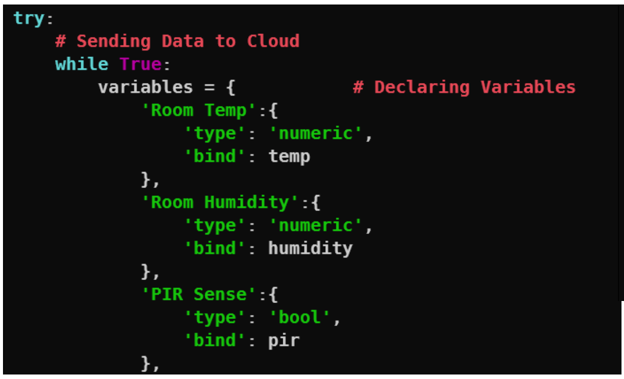


```python
def display(value):      # Control Display
    if value == True:
        lcd()
    else:
        wait()
```

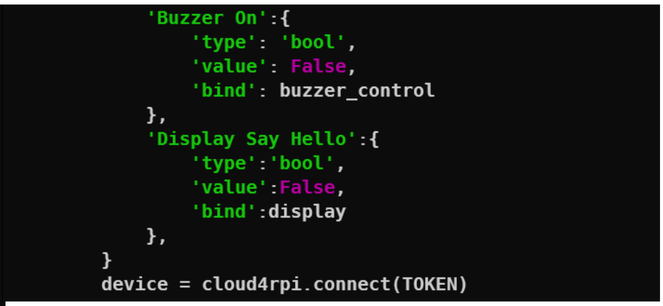


```python
def lcd():       # Control Display while input = true
    global display
    display = drivers.Lcd()
    display.lcd_display_string("Hello from Cloud4RPI !!!", 2)
    sleep(2)
    display.lcd_clear()
def wait():      # Control Display while input = false
    global display
    display = drivers.Lcd()
    display.lcd_display_string("waiting for signal......", 2)
    sleep(2)
```

The above screenshot shows three functions, display(), lcd(), and wait(). These three functions are used to control the display with specific output determined by the Boolean value passing from Cloud4RPi.

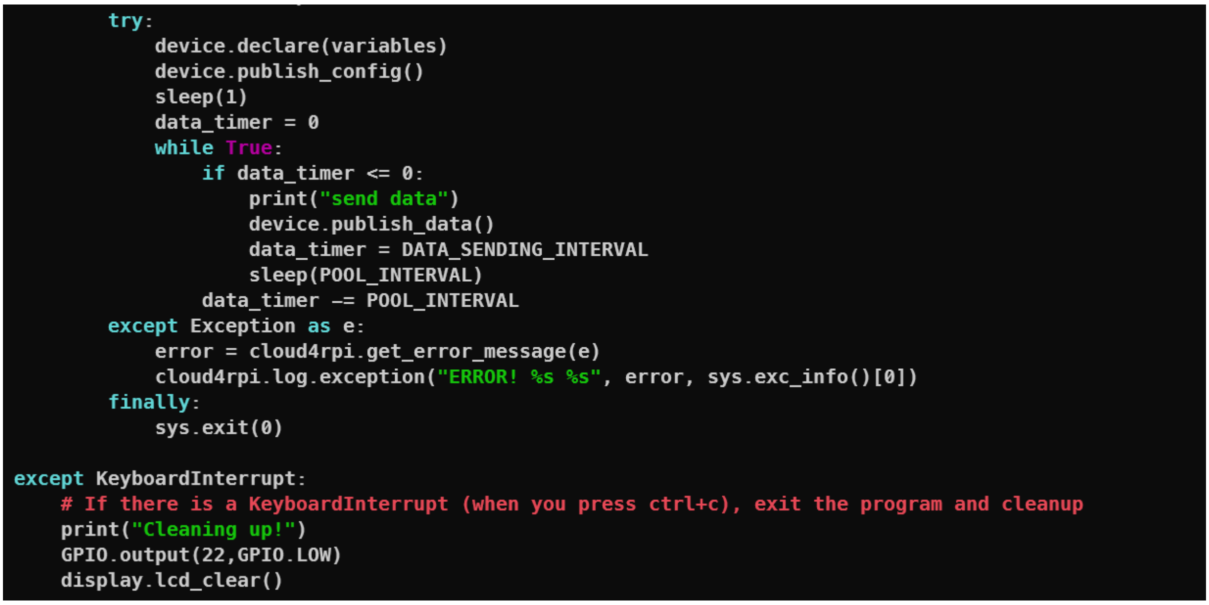


```python
try:
    # Sending Data to Cloud
    while True:
        variables = {           # Declaring Variables
            'Room Temp':{
                'type': 'numeric',
                'bind': temp
            },
            'Room Humidity':{
                'type': 'numeric',
                'bind': humidity
            },
            'PIR Sense':{
                'type': 'bool',
                'bind': pir
            },
```



```python
            'Buzzer On':{
                'type': 'bool',
                'value': False,
                'bind': buzzer_control
            },
            'Display Say Hello':{
                'type':'bool',
                'value':False,
                'bind':display
            },
        }
        device = cloud4rpi.connect(TOKEN)
```

The above screenshots represents the variables we declared in Cloud4RPi with the data type, value initialization, and binding function that we have explained previously. After creating the variables, we connect the RPI with the Cloud4RPi with the token.



```
        try:
            device.declare(variables)
            device.publish_config()
            sleep(1)
            data_timer = 0
            while True:
                if data_timer <= 0:
                    print("send data")
                    device.publish_data()
                    data_timer = DATA_SENDING_INTERVAL
                    sleep(POOL_INTERVAL)
                data_timer -= POOL_INTERVAL
        except Exception as e:
            error = cloud4rpi.get_error_message(e)
            cloud4rpi.log.exception("ERROR! %s %s", error, sys.exc_info()[0])
        finally:
            sys.exit(0)

except KeyboardInterrupt:
    # If there is a KeyboardInterrupt (when you press ctrl+c), exit the program and cleanup
    print("Cleaning up!")
    GPIO.output(22,GPIO.LOW)
    display.lcd_clear()
```

The above screenshots indicates the process after connecting to Cloud4RPi, with fixed package sending interval, the data collected by the sensors connected with our RPI is pushed to Cloud4RPi.

Demo Video Link

https://youtu.be/Aw2cILRIJsU

Thoughts & Comments

In homework 2 ,we learned how to update data from the sensor to the cloud with our raspberry pi.

When we managed to connect the sensor to the cloud, we found that some sensors, like PIR, were too sensitive. We tried to reduce the sensitivity, or add the sleep time; but neither of the two solutions can deal with the problem. Thus, we eventually put it in a dark place so that we could see the difference between high value of infrared or low value.

Meanwhile, the widget may not function as we expected from time to time, so we had to figure out a nice setup of the interface in order to successfully show the data.
After finishing this homework, we learned how to connect the object to the cloud and visualized Information. I thought if we had more time, we could design a more humanized interface and display it.