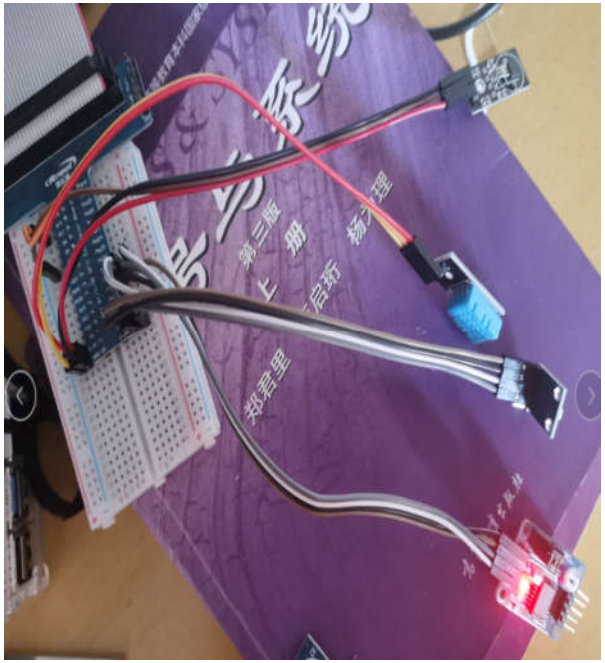
智能系统与控制



环境监测系统2

温度

23.62℃

湿度

40.0 %RH

光照度

295LUX

气压

100161 Pa

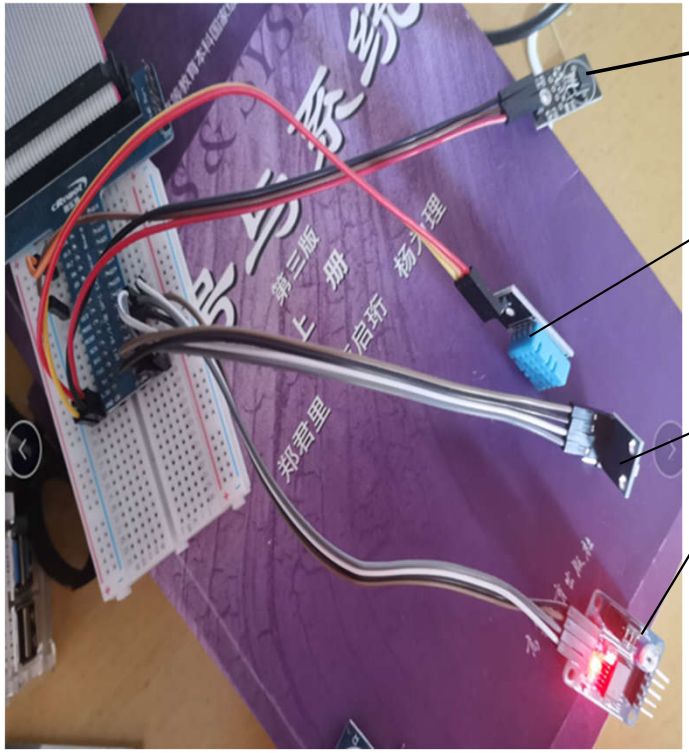
海拔

97 m

树莓派网络控制：Flask 环境监测

于泓
鲁东大学
信息与电气工程学院
2022.5.1

Flask 环境监测任务



1 监测温度： DS18b20

2 监测湿度： DHT11

3 监测光照度： PCF8591+光敏电阻

4 监测气压和海拔： BMP085

/home/pi/EXP-Raspberry/EXP_Flask/

Name	Size (KB)	Last mod
..		
static		2022-05-15 1
templates		2022-05-15 1
__pycache__		2022-05-15 0
main.py	2	2022-05-15 1
PCF8591.py	2	2022-05-15 0
BMP085.py	7	2022-05-15 0
dht11.py	7	2022-05-02 1
ds18b20.py	1	2022-05-02 1
light.py	1	2022-05-01 1
pin_dic.py	1	2022-05-01 1

不同传感器的驱动程序
为了解决异步触发问题，
将所有的类设置成**线程**的形式

Ds18b20：

```
import threading

class Ds18b20(threading.Thread):

    def __init__(self,str_id):
        super(Ds18b20, self).__init__()
        self.str_id = str_id
        self.str_temperature = " "

    def read(self):
        # 读取温度传感器的数值
        location = os.path.join( "/sys/bus/w1/devices",self.str_id,"w1_slave")

        if os.path.exists(location):
            with open(location) as tf:
                lines = tf.read().splitlines()

            text = lines[-1]
            temperaturedata = text.split(" ")[-1]

            temperature = float(temperaturedata[2:])

            return temperature/1000.0
        else:
            return False
```

继承自 threading.Thread

定义一个存储数据的变量

添加一个数据获取的方法

```
def get_temperature(self):
    return self.str_temperature
```

增加一个循环执行的run方法

```
def run(self):

    while True:

        t = self.read()

        if t:
            self.str_temperature = "%.2f"%(t)
        time.sleep(1)
```

测试程序

```
if __name__ == "__main__":

    str_id = "28-0300a2794829"
    m_ds18b20 =  Ds18b20(str_id)
    m_ds18b20.setDaemon(True)
    m_ds18b20.start()

    try:
        while True:
            print(m_ds18b20.get_temperature())

            time.sleep(1)

    except KeyboardInterrupt:
        print("\n Ctrl + C Quite")
```

类似的 测湿度

```
import threading

class DHT11(threading.Thread):

    def __init__(self,pin_D):
        super(DHT11, self).__init__()
        GPIO.setmode(GPIO.BOARD)
        self._pin = pin_D
        self.str_humidity = "    %%"



    def run(self):

        while True:
            flag, result = self.read_DHT()

            if flag:
                self.str_humidity = "%-3.1f %%"%(result[1])
            time.sleep(2)

    def get_humidity(self):
        return self.str_humidity
```

测光强

```
import threading
import RPi.GPIO as GPIO
class PCF8591(threading.Thread):
    # 初始化输入器件的物理地址Address，以及I2C的通道编号
    def __init__(self,Address=0x48,bus_id=1):
        super(PCF8591, self).__init__()
        self.bus_id = bus_id
        self.Address = Address
        self.bus = smbus.SMBus(self.bus_id)
        self.str_LUX = "      "

    def compute_LUX(self,N):

        if N == 255:
            N= N-1

        R = N/(255-N)*1000

        LUX = 40000*pow(R,-0.6021)

        return LUX


    def run(self):
        self.AD_read(0)
        while True:
            N = self.AD_read(10)
            LUX = self.compute_LUX(N)

            self.str_LUX = "%d"%(LUX)

            time.sleep(1)

    def get_LUX(self):
        return self.str_LUX
```

```python
class BMP085(threading.Thread):
    def __init__(self, mode=BMP085_STANDARD, address=BMP085_I2CADDR,bus_id =1):
        super(BMP085, self).__init__()
        # 检验模式是否正确.
        if mode not in [BMP085_ULTRALOWPOWER, BMP085_STANDARD, BMP085_HIGHRES, BMP085_ULTRAHIGHRES]:
            raise ValueError('Unexpected mode value {0}.  Set mode to one of BMP085_ULTRALOWPOWER, B
        self._mode = mode

        self._bus_id = bus_id

        self._device = smbus.SMBus(self._bus_id)

        self._address = address

        self._load_calibration()

        self.str_pressure = "     "
        self.str_altitude = "     "
```

```python
def run(self):

    while True:
        pressure = self.read_pressure()
        self.str_pressure = "%d Pa"%(pressure)
        time.sleep(2)
        altitude = self.read_altitude()
        time.sleep(2)
        self.str_altitude = "%d m"%(altitude)

def get_pressure(self):
    return self.str_pressure

def get_altitude(self):
    return self.str_altitude
```

准备用来进行数据呈现的html 文件

/home/pi/EXP-Raspberry/EXP_Flask/templates/

Name	Size (KB)	Last mod
..		
env_timer.html	1	2022-05-15 1
env.html	1	2022-05-15 0
index.html	1	2022-05-01 1

{{ }}包裹的变量
可以接受来自服务端
的信息

```html
<!DOCTYPE html>
<html>
    <head>
    <title>环境检测</title>
    </head>
    <body>
        <h2>环境监测系统</h2>

        <div >
        温度
        <h2>{{ tem }}℃</h2>
        </div>

        <div >
        湿度
        <h2>{{ hum }}RH</h2>
        </div>

        <div >
        光照度
        <h2>{{ lux }}LUX</h2>
        </div>

        <div >
        气压
        <h2>{{ press }}</h2>
        </div>

        <div >
        海拔
        <h2>{{ altitude }}</h2>
        </div>
    </body>
</html>
```

环境监测系统

温度

{{ tem }}℃

湿度

{{ hum }}RH

光照度

{{ lux }}LUX

气压

{{ press }}

海拔

{{ altitude }}

服务端代码:

```
from flask import Flask, render_template, Response, request,jsonify
from pin_dic import pin_dic
from light import light
from BMP085 import BMP085
from PCF8591 import PCF8591
from dht11 import DHT11
from ds18b20 import  Ds18b20


# 定义小灯对象
m_light = light(pin_dic["G17"])

# 定义环境监测对象，并启动线程

# 温度检测
m_ds18b20 =  Ds18b20("28-0300a2794829")
m_ds18b20.setDaemon(True)
m_ds18b20.start()

# 光照度检测
m_PCF8591 = PCF8591(Address=0x48,bus_id=1)
m_PCF8591.setDaemon(True)
m_PCF8591.start()

# 湿度检测
m_dht11 = DHT11(pin_dic['G13'])
m_dht11.setDaemon(True)
m_dht11.start()

# 气压海拔检测
m_BMP085 = BMP085(mode=1, address=0x77,bus_id =1)
m_BMP085.setDaemon(True)
m_BMP085.start()
```

启动环境检测的线程

增加路由

```
@app.route('/env',methods=['GET', 'POST'])
def env():

    print('t    ',m_ds18b20.get_temperature())
    print('h   ',m_dht11.get_humidity())
    print('lux   ',m_PCF8591.get_LUX())
    print('p   ',m_BMP085.get_pressure())
    print('al   ', m_BMP085.get_altitude())
    templateData = {
            'tem': m_ds18b20.get_temperature(),
            'hum': m_dht11.get_humidity(),
            'lux': m_PCF8591.get_LUX(),
            'press':m_BMP085.get_pressure(),
            'altitude':m_BMP085.get_altitude()
            }

    return render_template('env.html', **templateData)
```

利用字典进行数据返回

这种方案的主要缺点在于 **数据无法刷新**

增加动态刷新功能

增加一个路由
只返回数据不返回页面

```html
<!DOCTYPE html>
<html>
    <head>
    <title>环境检测</title>
    </head>

    <script>
    function getEnv() {
            var xmlHttp = new XMLHttpRequest();
            var baseUrl = "/get_env_info";
            xmlHttp.open( "GET", baseUrl, false );
            xmlHttp.send( null );
            return xmlHttp.responseText;
        }

     function writeEnv() {
            var Ino_Env = getEnv();
            var json = JSON.parse(Ino_Env);
            document.getElementById("temp_18b20").innerText = json.tem;
            document.getElementById("hum_DHT11").innerText = json.hum;
            document.getElementById("lux_PCF8591").innerText = json.lux;
            document.getElementById("press_BMP085").innerText = json.press;
            document.getElementById("altitude_BMP085").innerText = json.altitude;
        }
     function addTimer() {
            setInterval(writeEnv, 5000);
        }
    </script>
```

```python
@app.route('/get_env_info')
def responds_env_all():
    dic_env_data = {
            'tem': m_ds18b20.get_temperature(),
            'hum': m_dht11.get_humidity(),
            'lux': m_PCF8591.get_LUX(),
            'press':m_BMP085.get_pressure(),
            'altitude':m_BMP085.get_altitude()
            }

    return jsonify(dic_env_data)
```

```
<body onload="addTimer();">
    <h2>环境监测系统2</h2>

    <div >
    温度
    <h2><span id = "temp_18b20">{{ tem }}</span>℃</h2>
    </div>

    <div >
    湿度
    <h2><span id = "hum_DHT11">{{ hum }}</span>RH</h2>
    </div>

    <div >
    光照度
    <h2><span id = "lux_PCF8591">{{ lux }}</span>LUX</h2>
    </div>

    <div >
    气压
    <h2><span id = " press_BMP085">{{ press }}</span></h2>
    </div>

    <div >
    海拔
    <h2><span id = "altitude_BMP085">{{ altitude }}</span></h2>
    </div>


</body>
```

为每个变量起个名字