第一课、感知环境促健康

一、实践情境

在我们的生活中，温度和空气湿度对健康至关重要。温湿度的过高或过低都会影响到我们身体的状态，甚至可能会生病。为此，本节课上，我们将自制一个能够检测环境气温和空气湿度的装置，它能帮助我们实时监测环境温湿度，并且在环境变差时提醒我们。

二、实践目标

本实践项目运用掌控板作为智能终端，通过 DHT11 温湿度传感器采集数据，来实时检测环境温湿度并设置高温提醒。

三、实践过程

在本项目中，我们将利用 DHT11 温湿度传感器，分两步设计一个温湿度检测装置，来实时检测环境的温度与空气湿度，并在环境变差时进行提醒。

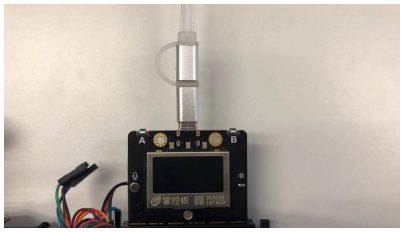
1、实时检测环境温湿度

2、添加提醒功能

任务 1：环境温湿度实时检测

1、硬件连接

通过 USB 连接线将掌控板接到计算机。



2、软件编写

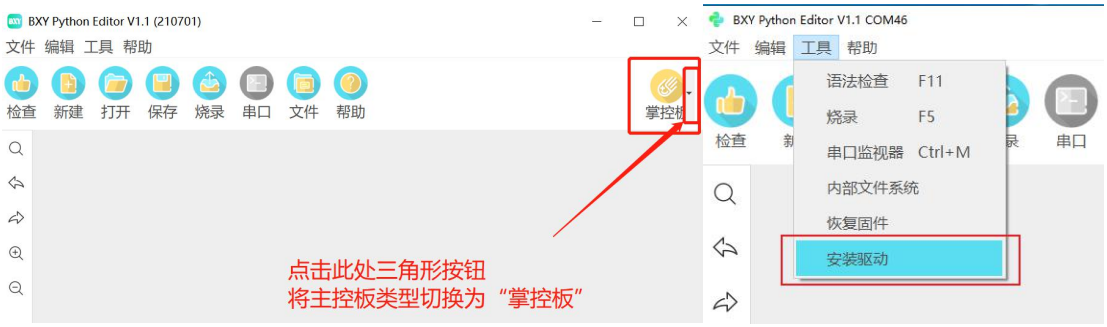
在首次编程时，我们将先对软件进行设置，之后再编写程序。

STEP1：软件设置

1、创建与保存项目文件

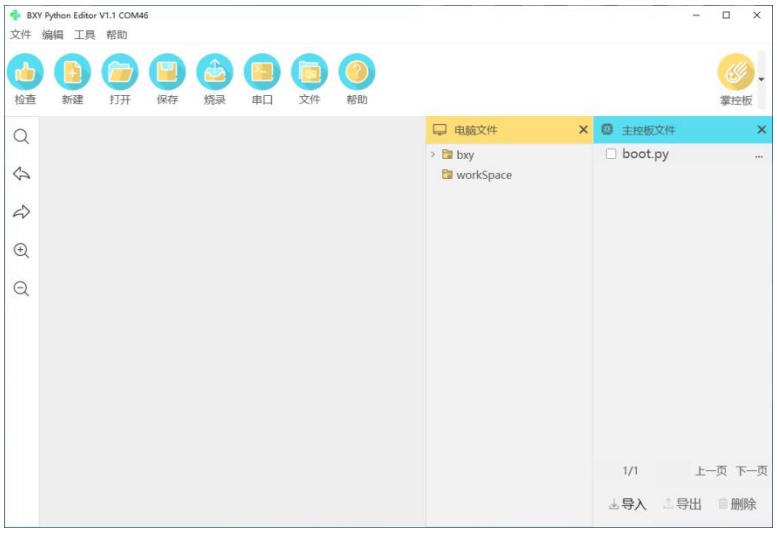
（1）启动 BXY 编程软件，选择主控板类型为“掌控板”

注 1：如果是第一次在电脑上使用掌控板，需要先安装驱动，如下右图。



注 2：如果掌控板中没有固件，则会提示烧录固件，也可以手动在“工具”菜单下选择“恢复固件”。

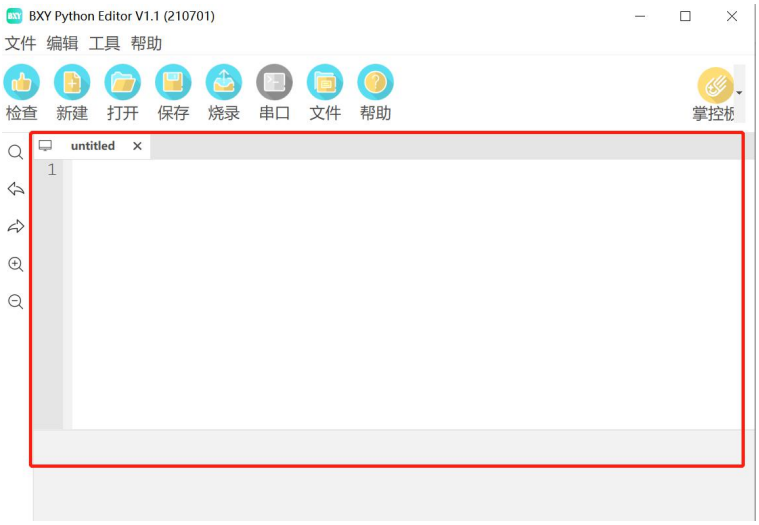
成功连接掌控板后“文件系统”会自动弹出，包含各种示例程序以及掌控板内部的文件列表。



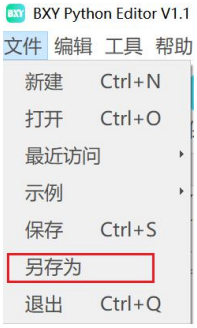
（2）单击左上角的“文件”按钮

（3）在弹出的框内点击“新建”按钮

之后我们会看到有一个名为“untitled”项目文件生成，

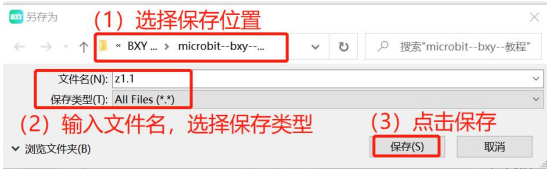


（4）创建完成后，点击菜单栏“文件”中的“另存为”



（5）在弹出的界面中选择保存位置，输入文件名“z1.1”，保存类型默认

Tips：保存位置和文件名称可自选



保存完成后，我们即可开始编程，

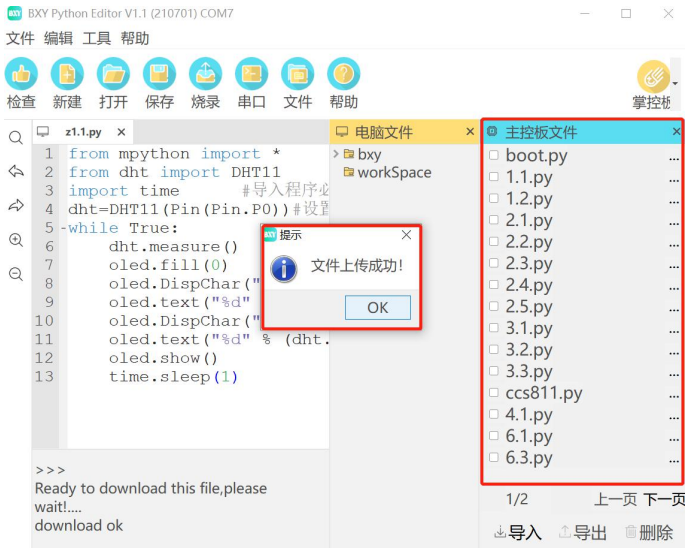
STEP2：程序编写

```
from mpython import *  #导入掌控板相关库
from dht import DHT11  #导入 dht11 库
import time    #导入 Time 时间库
dht=DHT11(Pin(Pin.P14)) #实例化 DHT11 对象
while True:  #永久循环
    dht.measure()  #检测
    oled.fill(0)  #清屏
    oled.DispChar("温 度:",0,10)  #在（0，10）坐标位显示“温度：”
    oled.text("%d" % (dht.temperature()), 48, 14)  #在（48，14）坐标位显示检测到的温度值
    oled.DispChar("湿 度:",0,35) #在（0，35）坐标位显示“湿度：”
    oled.text("%d" % (dht.humidity()), 48, 40) #在（48，40）坐标位显示检测到的湿度值
    oled.show()  #显示屏幕
    time.sleep(1)  #延时一秒
```

3、运行调试

STEP1：点击“烧录”上传程序

烧录完成后可以看到有提示字样，

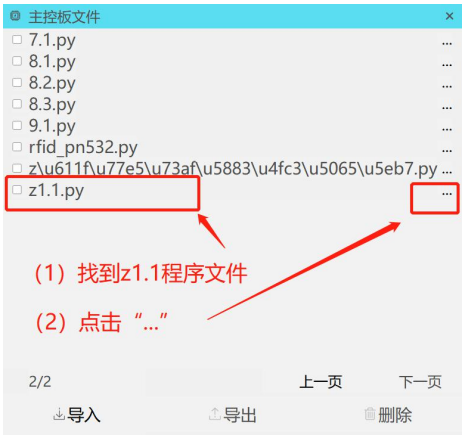


同时，如上图，在软件右侧界面，我们也可以看到主控板内的所有程序文件（具体文件依实际情况而定）

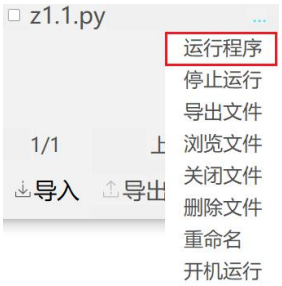
STEP2：运行程序文件

（1）找到上传的程序文件，并点击“...”。

Tips:若程序有多页，可通过上下翻页进行查找。

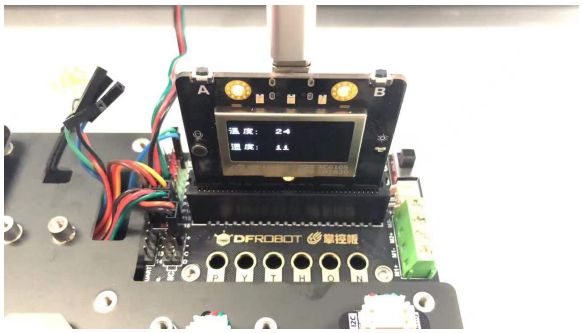


（2）在弹出的操作中，选择“运行程序”



STEP3：观察效果

观察掌控板，我们可以发现测得的温湿度值分别显示在屏幕上



注：由于在 BXY 软件中有关于 DHT11 的示例程序，因此我们也可以直接打开调用而不手动编辑。

任务 2：添加提醒功能

在上个任务中，我们已经完成了对于环境温湿度的实时检测，接下来，我们将在此基础上，为这个装置添加提醒功能，当实时检测到的环境温度过高时，点亮三颗 LED，否则熄灭。

1、软件编写

STEP1：软件设置

1、新建一个项目文件并命名为“z1.2”

STEP2：程序编写

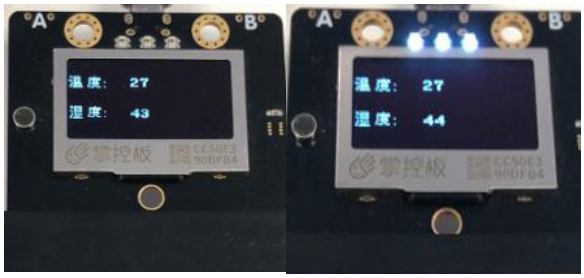
```
from mpython import *  #导入掌控板相关库
from dht import DHT11  #导入 dht11 库
import time  #导入 Time 时间库
dht=DHT11(Pin(Pin.P14))  #实例化 DHT11 对象
while True:  #永久循环
    x = dht.temperature()  #定义温度值为 x
    y = dht.humidity()  #定义湿度值为 y
    dht.measure()  #检测
    oled.fill(0)  #清屏
    oled.DispChar("温 度:",0,10)  #在（0，10）坐标位显示“温度：”
    oled.text("%d" % (dht.temperature()), 48, 14)  #在（48，14）坐标位显示检测到的温度值
    oled.DispChar("湿 度:",0,35) #在（0，35）坐标位显示“湿度：”
    oled.text("%d" % (dht.humidity()), 48, 40) #在（48，40）坐标位显示检测到的湿度值
    oled.show() #显示屏幕
    time.sleep(1) #延时一秒
    if x > 31: #如果温度超过 31℃
        rgb[0] = (16 ,16 ,16) #点亮第一个 LED
        rgb[1] = (16 ,16 ,16) #点亮第二个 LED
        rgb[2] = (16 ,16 ,16) #点亮第三个 LED
        rgb.write()  #写入
    if x <= 31: #如果温度低于 31℃
```

```
        rgb[0] = (0 ,0 ,0)
        rgb[1] = (0 ,0 ,0)
        rgb[2] = (0 ,0 ,0)
        rgb.write() #写入
```

2、运行调试

STEP1：点击“烧录”上传程序
将新编写的“z1.2”程序文件烧录上传至掌控板中。

STEP2：运行程序文件
在弹出的“主控板文件”中找到“z1.2”程序文件，并运行观察效果。我们发现掌控板屏幕上方的三颗 LED 始终处于熄灭状态，如下左图。



STEP3：将装置放在环境温度较高的区域，如夏季的室外等
观察掌控板，三颗 LED 已被点亮，如上右图。
注：程序中的“31”是我们依据实验情况设定的温度达到不宜健康时的临界值，可以依据各自的实际情况设置。

第二课、 电报发报机

一、实践情境

在一些谍战题材的影视作品中，我们常常看到一些人通过发电报的方式来传递信息，“滴滴滴哒哒”的电报声带感又刺激，最重要的，还能提高信息传输的安全性。本节课上，我们将自制一个电报发报机，感受一下其便利性。

二、实践目标

本实践项目运用掌控板作为智能终端，通过触摸掌控版上的金手指，来控制蜂鸣器播放不同的音调。

三、实践过程

在本项目中，我们将利用蜂鸣器，分两步设计一个电报发报机，通过触摸掌控版上的金手指来控制蜂鸣器发声。

1、蜂鸣器自动发声

2、手动控制蜂鸣器发声

任务 1：蜂鸣器自动发声

1、硬件连接

通过 USB 连接线将掌控板接到计算机。

2、软件编写

STEP1：软件设置

1、创建与保存项目文件

（1）启动 BXY 编程软件，选择主控板类型为“掌控板”

（2）新建项目，保存并命名为“z2.1”

STEP2：程序编写

```
# 程序效果：喇叭播放提示音
# 接线顺序：喇叭接 P16

from mpython import * #导入掌控板相关库
import music #导入音乐库

#music.play(music.BA_DING, pin=Pin.P16) #播放声音
music.pitch(131, 500, pin=Pin.P16) #使 16 引脚上的蜂鸣器播放音调 1 低 C/C3  500 毫秒
```

3、运行调试

STEP1：点击“烧录”上传程序
将新编写的“z2.1”程序文件烧录上传至掌控板中。

STEP2：运行程序并观察效果
在弹出的“主控板文件”中找到“z2.1”程序文件，运行后，我们可以听到“嘀”的一声，持续时间大约 500 毫秒。

任务 2：手动控制蜂鸣器发声

在上个任务中，我们已经成功使蜂鸣器在程序运行后自动发出“嘀”的一声，接下来，我们将在此基础上修改发声的方式，实现通过触摸掌控板上的金手指来控制蜂鸣器发出不同的音调。

1、软件编写

STEP1：软件设置
1、新建一个项目文件并命名为“z2.2”

STEP2：程序编写

```
from mpython import * #导入掌控板相关库
import music #导入音乐库

value = 0 #设置变量 value 初始值为 0
while True: #永久循环
    print("value=%d",value) #打印
```

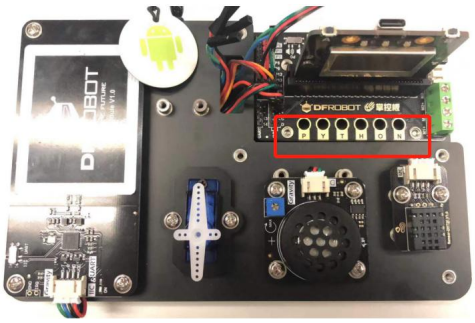
```
    value = value + 1 #自增 1
    if(touchPad_P.read() < 100): #P 引脚被按下
        music.pitch(131, 500, pin=Pin.P16);#使 16 引脚上的蜂鸣器播放音调 1 低 C/C3 500 毫秒
    elif(touchPad_Y.read() < 100): #Y 引脚被按下
        music.pitch(147, 500, pin=Pin.P16);#2 低 D/D3
    elif(touchPad_T.read() < 100): #T 引脚被按下
        music.pitch(165, 500, pin=Pin.P16);#3 低 E/E3
    elif(touchPad_H.read() < 100): #H 引脚被按下
        music.pitch(175, 500, pin=Pin.P16);#4 低 F/F3
    elif(touchPad_O.read() < 100): #O 引脚被按下
        music.pitch(196, 500, pin=Pin.P16);#5 低 G/G3
    elif(touchPad_N.read() < 100): #N 引脚被按下
        music.pitch(220, 500, pin=Pin.P16);#6 低 A/A3
```

2、运行调试

STEP1：点击“烧录”上传程序
将新编写的“z2.2”程序文件烧录上传至掌控板中。

STEP2：运行程序文件
在弹出的“主控板文件”中找到“z2.2”程序文件，并运行观察效果。

STEP3：触摸掌控扩展板上的金手指并观察效果。
随机按下扩展板上“P”、“Y”、“T”、“H”、“O”、“N”，我们可以听到蜂鸣器发出了各个不同的音调，每个音调持续时间 500 毫秒。



第三课、午餐选择器

一、实践情境

如今，随着物质生活水平的不断提高，每天的午餐选择渐渐成为许多人的一大难题。本节课上，我们将自制一个午餐选择器，为选择困难症患者解决这一困扰。

二、实践目标

本实践项目运用掌控板作为智能终端，通过按下掌控板上的按钮，来控制舵机随机转至一个角度，而每个位置对应一个午餐选择。

三、实践过程

在本项目中，我们将利用板载按钮和舵机，分两步设计一个午餐选择器，通过按下掌控板上的板载按钮 B 来控制舵机转动，继而选择一款午餐。

1、舵机来回转动

2、按钮控制舵机转动

任务 1：舵机来回转动

1、硬件连接

通过 USB 连接线将掌控板接到计算机。

2、软件编写

STEP1：软件设置

1、创建与保存项目文件

（1）启动 BXY 编程软件，选择主控板类型为“掌控板”

（2）新建项目，保存并命名为“z3.1”

STEP2：程序编写

```
# 程序效果：舵机 180 度来回缓慢转动
# 接线顺序：舵机接 P15

from mpython import * #导入掌控板相关库
from servo import Servo #导入舵机相关库
import time #导入 Time 时间库

s=Servo(15,min_us=500, max_us=2500) #实例化舵机对象，设置舵机脉冲宽度参数的最大值和最小值，引脚为 15 口
while True: #永久循环
    s.write_angle(10) #控制舵机转到 10 度位置
    time.sleep(1) #延时一秒
    s.write_angle(170) #控制舵机转到 170 度位置
    time.sleep(1) #延时一秒
```

3、运行调试

STEP1：点击“烧录”上传程序
将新编写的“z3.1”程序文件烧录上传至掌控板中。

STEP2：运行程序并观察效果
运行程序后，我们可以看到舵机在 10°与 170°之间不停得来回转动。

任务 2：按钮控制舵机转动

在上个任务中，我们已经成功使舵机转动起来，接下来，我们将在此基础上添加按钮控制，使舵机在板载按钮 B 被按下后能够随机转动到一个指定角度。

1、软件编写

STEP1：软件设置
1、新建一个项目文件并命名为“z3.2”

STEP2：程序编写

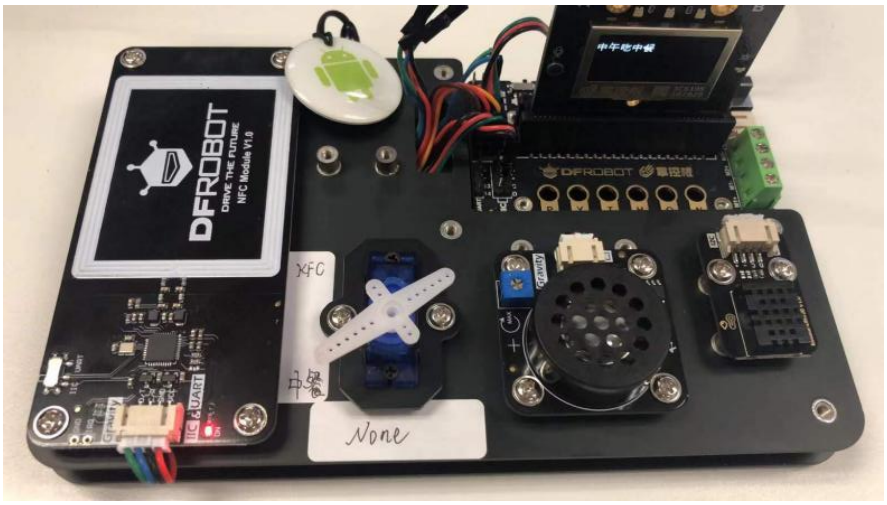
```
from mpython import * #导入掌控板相关库
from servo import Servo #导入舵机相关库
import time #导入 time 时间库
import random #导入随机数库
```

```
s=Servo(15,min_us=500, max_us=2500) #实例化舵机对象，设置舵机脉冲宽度参数的最大值和最小值
s.write_angle(0) #控制舵机转到 0 度位置
time.sleep(1) #延时一秒

a = 60
b = 120
c = 170
d = [a,b,c]

while True:
    if button_b.value() == 0 :      #按键 A 按下
        time.sleep(0.1)
        e = random.choice(d) #随机选择一个数
        print(e)
        s.write_angle(e) #随机选择三个数中的一个作为角度，控制舵机转至该位置
        if e == 60: #如果舵机转至 60°
            oled.fill(0)   #清屏
            oled.DispChar("中午吃 KFC",0,10)   #在（0，10）坐标位显示对应的选择
            oled.show()   #显示屏幕
        elif e == 120: #如果舵机转至 120°
            oled.fill(0)   #清屏
            oled.DispChar("中午吃中餐",0,10)   #在（0，10）坐标位显示对应的选择
            oled.show()   #显示屏幕
        elif e == 170: #如果舵机转至 170°
            oled.fill(0)   #清屏
            oled.DispChar("今天减肥，中午不吃了",0,10)   #在（0，10）坐标位显示对应的选择
            oled.show()   #显示屏幕
    time.sleep(0.1)
```

2、运行调试

STEP1：点击“烧录”上传程序
将新编写的“z3.2”程序文件烧录上传至掌控板中。

STEP2：运行程序并观察效果。
运行程序后，按下掌控板上方的 B 按键，舵机随机转至“60°”，“120°”，“170°”中的其中一个方位，每个方位代表了一种午餐选择，同时，我们也可以在掌控板的屏幕上看到相应的文字显示。


DFROBOT
DRIVE THE FUTURE
NFC Module V1.0
NFC
None

第四课、 智能门禁系统

一、实践情境

住在一些老旧小区时，每次出门，我们都得随身携带一长串钥匙，这为我们的生活带来了许多不便。为此，在这节课上，我们将设计一个智能门禁系统，能够免去带钥匙的烦恼并能记录人员进出门时间。

二、实践目标

本实践项目运用掌控板作为智能终端，借助 NFC 近场通讯模块和来设计一个智能门禁系统，继而结合舵机实现门的开关。

三、实践过程

在本项目中，我们将利用 NFC 近场通讯模块和舵机，分步骤设计一个智能门禁系统。

1、检测 RFID 卡 ID 信息

2、控制舵机转动

任务 1：检测 RFID 卡 ID 信息

1、硬件连接

通过 USB 连接线将掌控板接到计算机。

2、软件编写

STEP1：软件设置

1、创建与保存项目文件

（1）启动 BXY 编程软件，选择主控板类型为“掌控板”

（2）新建项目，保存并命名为“z4.1”

STEP2：rfid_pn532 程序烧录

在编写任务程序之前，我们需要先烧录 rfid_pn532 库程序至主控板中。

（1）找到“电脑文件”中“3-4 RFID”文件夹下的“rfid_pn532.py”文件

（2）双击打开后烧录至主控板

STEP3：程序编写

烧录好 rfid_pn532 库程序后，我们再进行任务 1 程序的编写

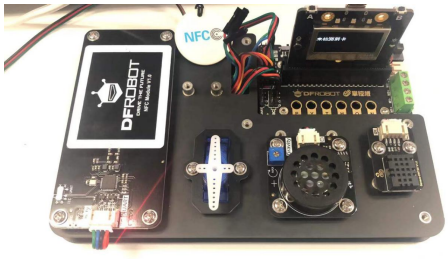
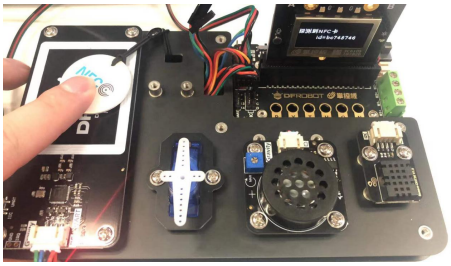
```
# 功能：读取 RFID 卡 ID 信息
# 实验：
#                 将 RFID 读卡器的 TXD 接在 P15      RXD 接在 P16，运行此程序
#                 将 RFID 卡放在读卡器上，读到 NFC 卡，蜂鸣器将响一下
#                 OLED 屏幕上显示对应的信息
'''
注意：  要先烧录 rfid_pn532.py 文件
'''
from machine import UART #导入 machine 相关库
from mpython import * #导入掌控板相关库
import time #导入 Time 时间库
import music #导入音乐库
from rfid_pn532 import RFID_PN532 #导入 rfid_pn532 相关库

u = UART(2,baudrate=115200,rx=33,tx=32,timeout=100) #初始化
pn = RFID_PN532(u) #实例化 rfid 对象

oled.fill(0) #清屏
```

```
oled.DispChar("正在连接读卡器...",0,0) #显示正在连接
oled.show() #屏幕显示

pn.wakeup() #唤醒
oled.DispChar("连接成功", 32,32) #显示连接成功
oled.show() #屏幕显示
oled.fill(0) #清屏

while True: #永久循环
    detected,id = pn.read_tag() #读取检测 NFC 卡
    if detected: #如果检测到
        oled.DispChar('探测到 NFC 卡',0,16) #屏幕显示探测到 NFC 卡
        music.play(music.BA_DING, pin=Pin.P16) #播放音乐
        oled.DispChar(" id=%2x%2x%2x%2x"%(id[0],id[1],id[2],id[3]),32,32) #显示 id 号
    else: #否则
        oled.DispChar('未检测到卡',0,16) #显示未检测到
    oled.show() #屏幕显示
    oled.fill(0) #清屏
    time.sleep(1) #延时一秒
```

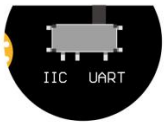
3、运行调试

STEP1：点击“烧录”上传程序
将新编写的“z4.1”程序文件烧录上传至掌控板中。

STEP2：运行程序并观察效果

读卡器没有检测到 NFC 卡片时，掌控板屏幕显示未检测到卡	把 NFC 卡片放到读卡器上，蜂鸣器发出嘟的一声，掌控板屏幕显示 NFC 卡片的 ID
 	 

Tips：须将 NFC 近场通讯模块上的开关拨到 UART 一端。



任务 2：反馈控制

在上个任务中，我们已经实现了 RFID 卡 ID 信息的检测，接下来，我们将在此基础上编写程序实现反馈控制，当检测信息正确时使舵机转动起来实现开门的效果。

1、软件编写

STEP1：软件设置

1、新建一个项目文件并命名为“z4.2”

STEP2：程序编写

```
# 功能：读取 RFID 卡 ID 信息
# 实验：
#                 将 RFID 读卡器的 TXD 接在 P15      RXD 接在 P16，运行此程序
#                 将 RFID 卡放在读卡器上，读到 NFC 卡，蜂鸣器将响一下
#                 OLED 屏幕上显示对应的信息
'''
注意：  要先烧录 rfid_pn532.py 文件
'''

from machine import UART #导入 machine 相关库
from mpython import * #导入掌控板相关库
import time #导入 Time 时间库
import music #导入音乐库
from servo import Servo #导入舵机相关库
from rfid_pn532 import RFID_PN532 #导入 rfid_pn532 相关库

s=Servo(15,min_us=500, max_us=2500) #实例化舵机对象，设置舵机脉冲宽度参数的最大值和最小值
s.write_angle(0) #控制舵机转到 0 度位置
time.sleep(1) #延时一秒

u = UART(2,baudrate=115200,rx=33,tx=32,timeout=100) #初始化
pn = RFID_PN532(u) #实例化 rfid 对象

oled.fill(0) #清屏
oled.DispChar("正在连接读卡器...",0,0) #显示正在连接
```

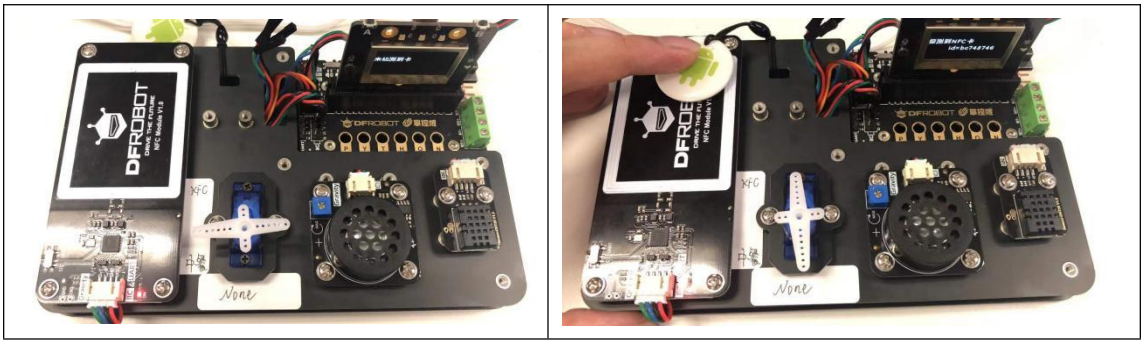
```
oled.show() #屏幕显示

pn.wakeup() #唤醒
oled.DispChar("连接成功", 32,32) #显示连接成功
oled.show() #屏幕显示
oled.fill(0) #清屏

while True: #永久循环
    detected,id = pn.read_tag() #读取 tag
    if detected: #如果检测到
        oled.DispChar('探测到 NFC 卡',0,16) #屏幕显示探测到 NFC 卡
        print(id) #打印原始字节 id
        music.play(music.BA_DING, pin=Pin.P16) #播放音乐
        a = (id[0],id[1],id[2],id[3]) #id 字符
        oled.DispChar(" id=%2x%2x%2x%2x"%a,32,32) #显示 id 号
        if id == b'\xbct\x87F': #如果检测正确
            s.write_angle(180) #控制舵机转到 180 度位置
            time.sleep(1) #延时一秒
    else: #否则
        oled.DispChar('未检测到卡',0,16) #显示未检测到
        s.write_angle(90) #控制舵机转到 90 度位置
        time.sleep(1) #延时一秒

    oled.show() #屏幕显示
    oled.fill(0) #清屏
    time.sleep(1) #延时一秒
```

2、运行调试

STEP1：点击“烧录”上传程序
将新编写的“z4.2”程序文件烧录上传至掌控板中。

STEP2：运行程序并观察效果
运行程序后，舵机转动至 90°横摆，之后检测 NFC 卡片。

读卡器没有检测到 NFC 卡片时，掌控板屏幕显示未检测到卡，舵机转至 90°横摆。	把 NFC 卡片放到读卡器上，蜂鸣器发出嘟的一声，掌控板屏幕显示 NFC 卡片的 ID，同时舵机转动至 180°。

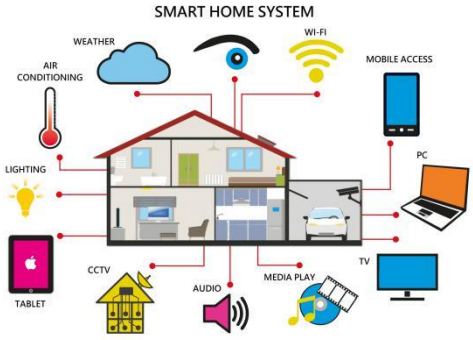

DFROBOT
NFC
None
DFROBOT
NFC
None

第五课、设计智能家居系统

一、实践情境

近年来，智能家居的概念热度一直居高不下，而随着技术的不断进步，许多曾经火热的概念也慢慢有了落地的机会。

在这节课上，我们将模拟一个智能家居系统，体验其带来的便利性。



二、实践目标

本实践项目运用掌控板作为智能终端，借助 SIoT 物联网平台来设计一个智能家居系统，继而将温湿度数据、NFC 数据上传至平台实现远程监测，并结合舵机实现远程控制门的开关。

三、实践过程

在本项目中，我们将利用 DHT11 温湿度传感器、NFC 近场通讯模块和舵机，分步骤设计一个智能家居系统。

1、搭建物联网平台

2、检测数据并上传至平台

3、远程开门

任务 1：搭建物联网平台

1、搭建步骤

STEP0：准备好 SIoT 软件包

根据自己电脑的系统，提前下载对应的 SIoT 软件压缩包。下载链接：http://Mindplus.dfrobot.com.cn/siot

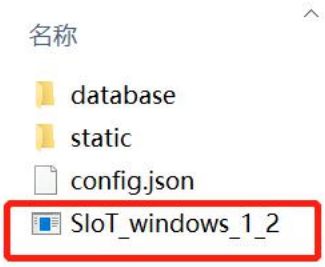
STEP1：电脑连接 WiFi

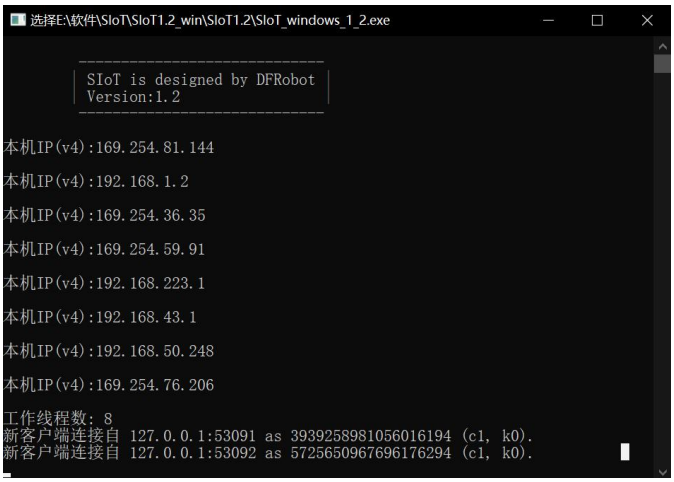
将电脑连接到 WiFi。

Tips：提供 WIFI 的路由器或手机热点可以不连接互联网，因为使用 SIoT 实现物联网应用时，只需要使用路由器或手机热点建立一个局域网即可。

STEP2：运行 SIoT 系统

双击运行 SIoT 应用程序，可以看到一个黑色的窗口。本课程以 win10 系统为例。





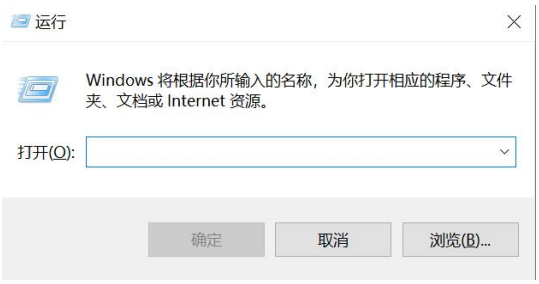
Tips：使用 SIoT 过程中一定不要关闭该窗口。

STEP3：电脑获取 IP 地址

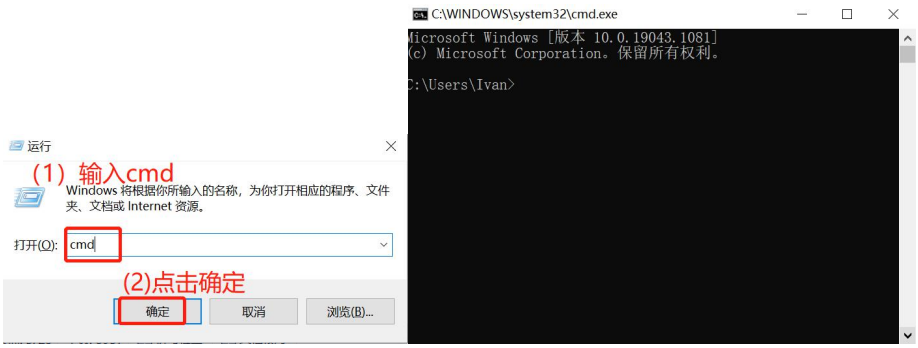
电脑每次连接 WIFI，都会生成一个 IP 地址，每个 IP 地址对应的电脑都是唯一的。运行 SIOT 程序后会在该电脑上建立一个服务器，其他设备要访问它，就需要知道该电脑的 IP 地址。

获取电脑 IP 的方法有很多，可在网页上搜索到，下面我们来介绍其中一种简易操作方法，通过以下 3 步获取电脑 IP。

（1）同时按下键盘上“WIN”+“R”，弹出如下运行窗口。

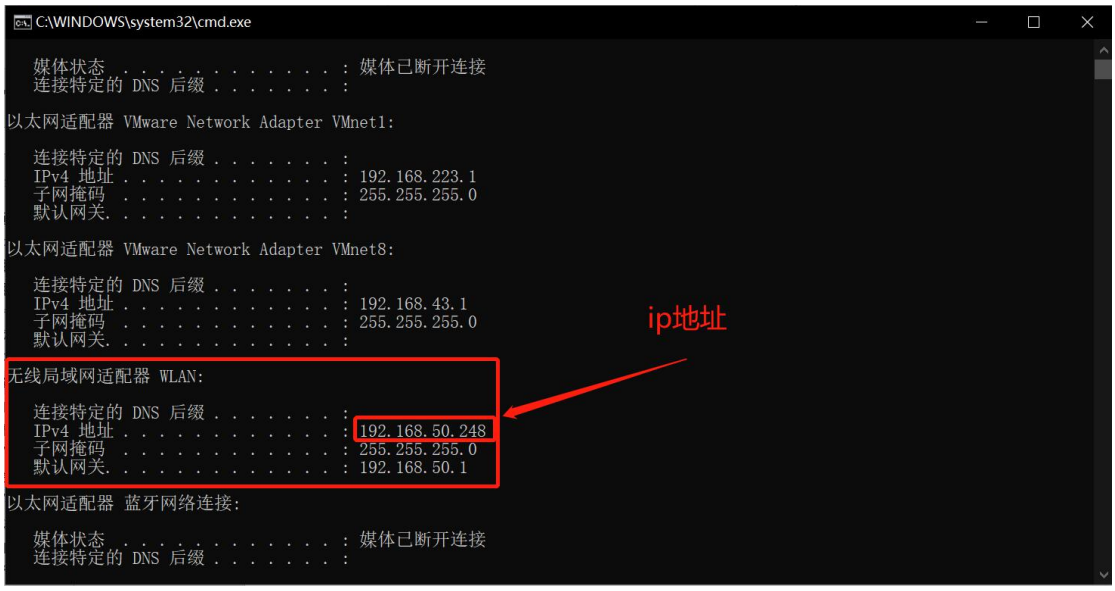


（2）输入“cmd”，点击确定，弹出小黑框。



（3）在小黑框中输入“ipconfig”，点击键盘“enter”，在小黑框中可以看到 IP 地址，如下图 IP 为 192.168.50.248。

Tips：连上不同的 WiFi，IP 地址是不一样的。

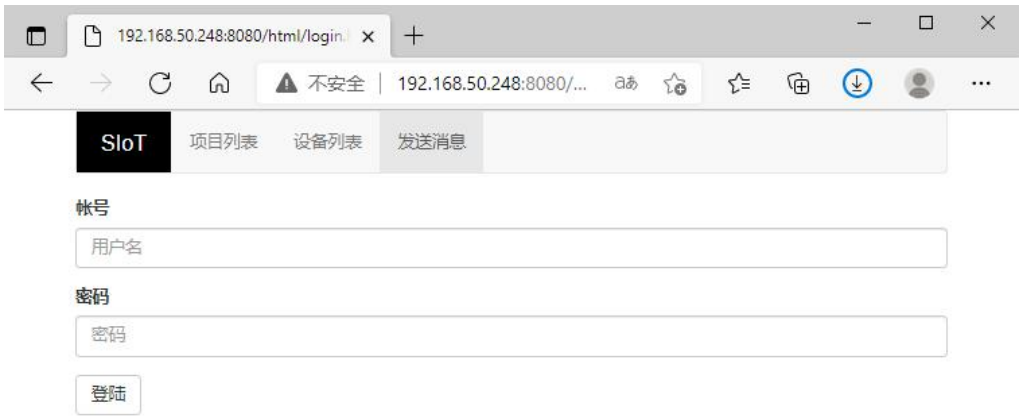


STEP4：打开服务器网页管理界面

（1）打开电脑浏览器，在网址栏输入 STEP3 中获得的 IP 地址，加上 “:8080”，如，192.168.50.248:8080

Tips： “:” 须在英文输入法下。

（2）点击键盘 enter 键，打开即为网页管理界面，如下图：



（3）打不开怎么办？

- 检查 SIoT 的小黑窗是否打开
- 检查 IP 地址是否错误，如果有多个 IP 地址就一个一个尝试
- 关闭网络防火墙

STEP5：登录账号

账号：siot，密码：dfrobot

输入账号、密码并点击登录，如下图，

Tips：网页端账号密码都是统一的

2、硬件连接

通过 USB 连接线将掌控板接到计算机。

3、软件编写

STEP1：软件设置

1、创建与保存项目文件

（1）启动 BXY 编程软件，选择主控板类型为“掌控板”

（2）新建项目，保存并命名为“z5.1”

STEP2：程序编写

```
from mpython import *   #导入掌控板相关库
from umqtt.simple import MQTTClient #导入 mqtt 相关库
import time #导入 Time 时间库

SERVER = "192.168.50.115" # MQTT 服务器 IP，运行 SIoT 的电脑 IP
username='siot'                 # SIoT 默认用户名（user）为：siot
password='dfrobot'              # 默认密码（pwd）为：dfrobot
CLIENT_ID = ""                  # Client ID 默认为空

TOPIC='df/sdf'                  # 自定义 topic，命名方式：项目 ID/设备名，不可缺少符号/

mywifi=wifi()                   # 实例化 wifi 类
```

```
mywifi.connectWiFi("the way out2","dfrobot2017")     # 连接 wifi,替换成你自己的 wifi

try:
    c = MQTTClient(CLIENT_ID, SERVER,1883,username,password,keepalive=30)
    # MQTTClient 类实例,并设置连接保持时间间隔为 30 秒
    c.connect()                                      # 连接 mqtt
    print("Connected to %s" % SERVER)
    while True:
        c.publish(TOPIC,"hello")                     # publish "hello" to 物联网平台
        time.sleep(1)
finally:
    c.disconnect()                                   # disconnect while 异常
```

Tips:上述“SERVER”中输入的须是自己电脑的实际 IP 地址。“connectWIFI”中输入的则是自己电脑所连接的实际 WiFi 和密码。

4、运行调试

STEP1：点击“烧录”上传程序

将新编写的“z5.1”程序文件烧录上传至掌控板中。

STEP2：运行程序并观察效果

（1）打开串口

运行程序并打开软件端的串口，等待两秒后我们可以看到有下图字样，表示已成功连接至平台。

（2）平台查看消息

打开网页界面，先点击“设备列表”，可以看到“项目 ID”、“名称”分别对应为程序中“TOPIC”的信息；再点击“查看消息”，在弹出的页面中可以看到设备的 MQTT 消息记录；最后单击“自动刷新消息记录”，勾选后可以使消息实时自动刷新显示。

可看到程序中对 Topic:“df/sdf”的消息记录，伴有具体的消息内容及发送时间，如下图，

任务 2：检测数据并上传

在上个任务中，我们已经实现了 SIoT 物联网平台的搭建，并成功得每隔一秒发送一条“Hello”消息至平台上，接下来，我们将在此基础上结合 DHT11 温湿度传感器以及 NFC 近场通信模块来检测环境数据，并上传至平台实现远程监测。

1、软件编写

STEP1：软件设置

1、新建一个项目文件并命名为“z5.2”

STEP2：rfid_pn532 程序烧录

同样的，在编写任务程序之前，我们还是需要先烧录 rfid_pn532 库程序至主控板中。

Tips:方法步骤同上节课

STEP3：程序编写

烧录好 rfid_pn532 库程序后，我们再进行任务 2 程序的编写

```
from machine import UART #导入 machine 相关库
from mpython import *  #导入掌控板相关库
from umqtt.simple import MQTTClient #导入 mqtt 相关库
from rfid_pn532 import RFID_PN532 #导入 rfid_pn532 相关库
import time
from dht import DHT11  #导入 dht 库

dht=DHT11(Pin(Pin.P14)) #实例化 DHT11 对象
u = UART(2,baudrate=115200,rx=33,tx=32,timeout=100) #初始化
pn = RFID_PN532(u) #实例化 rfid 对象

oled.fill(0) #清屏
oled.DispChar("正在连接读卡器...",0,0) #显示正在连接
oled.show() #屏幕显示

pn.wakeup() #唤醒
oled.DispChar("连接成功", 32,32) #显示连接成功
oled.show() #屏幕显示
oled.fill(0) #清屏

SERVER = "192.168.50.115" # MQTT 服务器 IP，即运行 SIoT 的电脑 IP
username='siot'              # SIoT 默认用户名（user）为：siot
password='dfrobot'           # 默认密码（pwd）为：dfrobot
CLIENT_ID = ""               # Client ID 默认为空

TOPIC1='df/device1'              # 自定义 topic，命名方式：项目 ID/设备名，不可缺少符号/
TOPIC2='df/device2'
TOPIC3='df/device3'

mywifi=wifi()                  # 实例化 wifi 类
mywifi.connectWiFi("the way out2","dfrobot2017")    # 连接 wifi,替换成你的 wifi

try:
    c = MQTTClient(CLIENT_ID, SERVER,1883,username,password,keepalive=30)
     # MQTTClient 类实例,并设置连接保持时间间隔为 30 秒
    c.connect()                                    # 连接 mqtt
    print("Connected to %s" % SERVER)
    while True:
        dht.measure()  #检测温湿度
        tmp = str(dht.temperature())#温度值
```

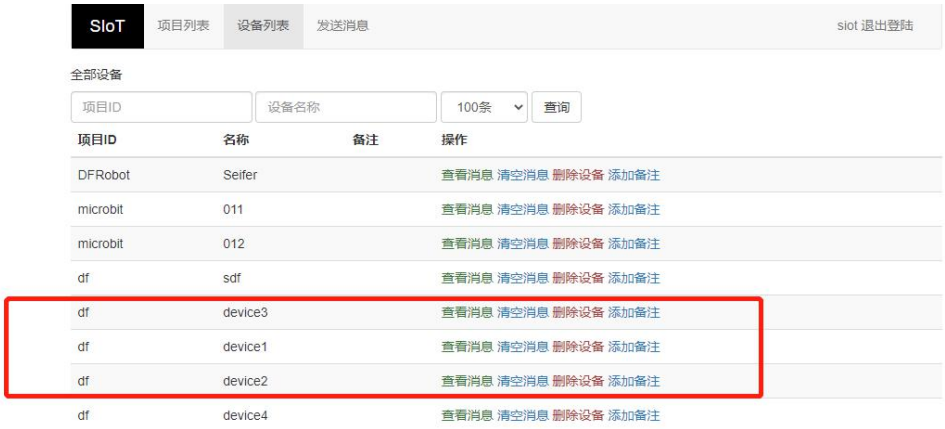
```
            hum = str(dht.humidity()) #湿度值
            oled.fill(0)    #清屏
            c.publish(TOPIC1,tmp)      # publish 消息至 Topic1
            c.publish(TOPIC2,hum)      # publish 消息至 Topic2
            detected,id = pn.read_tag() #读取 tag
            if detected: #如果检测到
                oled.DispChar('探测到 NFC 卡',0,16) #屏幕显示探测到 NFC 卡
                a = (id[0],id[1],id[2],id[3])#id 字符
                oled.DispChar(" id=%2x%2x%2x%2x"%a,32,32) #显示 id 号
                b = str(a)
                c.publish(TOPIC3,b)   # publish 消息至 Topic3
            else:
                oled.DispChar('未检测到卡',0,16) #显示未检测到
                c.publish(TOPIC3,"0") # publish 消息至 Topic3
            oled.show() #屏幕显示
            oled.fill(0) #清屏
            time.sleep(1) #延时一秒

finally:
    c.disconnect()                                  #  异常时,断开 mqtt 连接
```

Tips:上述“SERVER”中输入的须是自己电脑的实际 IP 地址。“connectWIFI”中输入的则是自己电脑所连接的实际 WiFi 和密码。

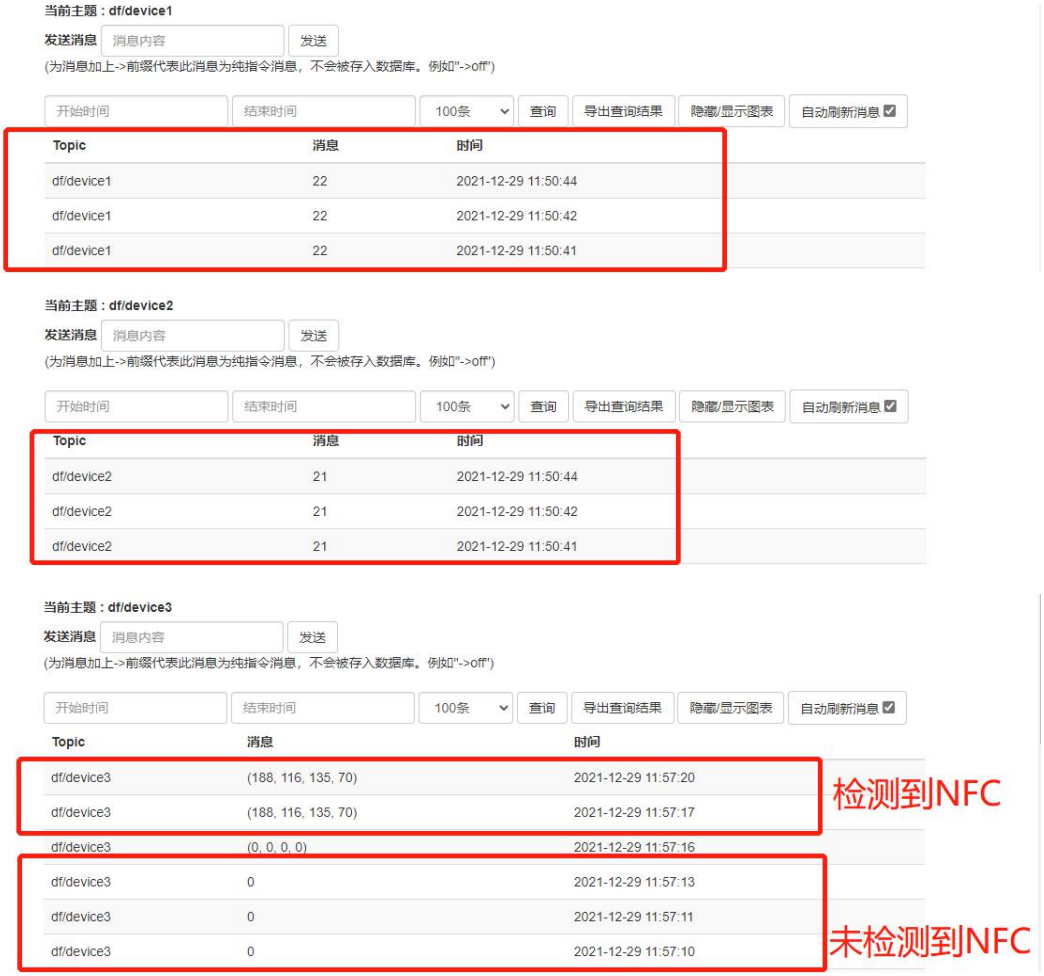
2、运行调试

STEP1：点击“烧录”上传程序
将新编写的“z5.2”程序文件烧录上传至掌控板中。

STEP2：运行程序并观察效果
（1）打开串口
运行程序并打开软件端的串口，等待连接至平台。
（2）平台查看消息
在设备列表中，我们分别查看下列三个设备中的消息。



查看消息后，不难发现，温度数据已传至“device1”中，湿度数据已传至“device2”中，NFC 数据已传至“device3”中。



任务 3：远程开门

在之前的任务中，我们已经成功得将传感器获取到的数据上传至物联网平台并在远程监测，接下来，我们将在此基础上实现反向控制，通过物联网平台发送指定消息后，使舵机转动起来模拟开门的效果。

1、软件编写

STEP1：软件设置

1、新建一个项目文件并命名为“z5.3”

STEP2：程序编写

```
from mpython import *   #导入掌控板相关库
from umqtt.simple import MQTTClient #导入 mqtt 相关库
from machine import Timer #导入计时器相关库
from servo import Servo #导入舵机相关库
import music #导入音乐库
import time   #导入 time 时间库

s=Servo(15,min_us=500, max_us=2500) #实例化舵机对象，设置舵机脉冲宽度参数的最大值和最小值
s.write_angle(0) #控制舵机转到 0 度位置
time.sleep(1) #延时一秒

SERVER = "192.168.50.115" # MQTT 服务器 IP，即运行 SIoT 的电脑 IP
username='siot'                 # SIoT 默认用户名（user）为：siot
password='dfrobot'              # 默认密码（pwd）为：dfrobot
CLIENT_ID = ""                  # Client ID 默认为空

TOPIC4='df/device4'             # 自定义 topic，命名方式：项目 ID/设备名，不可缺少符号/
mywifi=wifi()                   # 实例化 wifi 类
mywifi.connectWiFi("the way out2","dfrobot2017")    # 连接 wifi,替换成你的 wifi
oled.fill(0)    #清屏

try:
    def sub_cb(topic, msg):         # 当接收到订阅消息时的回调函数
        print((topic, msg))         # 打印接收的主题消息
        if topic == TOPIC4.encode():   #
            if msg == b"1": #如果发送“1”
                s.write_angle(170) #控制舵机转到 170 度位置
                time.sleep(1) #延时一秒
                music.play(music.BA_DING, pin=Pin.P16) #播放音乐
            else: #发送其他信息
                s.write_angle(10) #控制舵机转到 10 度位置
                time.sleep(1) #延时一秒
    c = MQTTClient(CLIENT_ID, SERVER,1883,username,password,keepalive=3)
     # MQTTClient 类实例,并设置连接保持时间间隔为 3 秒
    c.connect()                                     # 连接 mqtt
```

```
    c.set_callback(sub_cb)                          # 设置回调函数
    c.subscribe(TOPIC4)                             # 订阅主题
    print("Connected to %s" % SERVER)
    c.publish(TOPIC4,"hello")      # publish 消息至 Topic4
    tim1 = Timer(1)                                 # 创建定时器 1
    tim1.init(period=2000, mode=Timer.PERIODIC,callback=lambda n:c.ping()) # 2 秒间隔发送
Ping,保持连接
    while True:
        c.wait_msg()# 循环等待消息
finally:
    c.disconnect()                                  # 异常时,断开 mqtt 连接
```

Tips:上述“SERVER”中输入的须是自己电脑的实际 IP 地址。“connectWIFI”中输入的则是自己电脑所连接的实际 WiFi 和密码。

2、运行调试

STEP1：点击“烧录”上传程序

将新编写的“z5.3”程序文件烧录上传至掌控板中。

STEP2：运行程序文件

（1）打开串口

运行程序并打开软件上端串口，等待连接至平台。

（2）观察舵机

观察舵机，可以看到舵机重新转回到了 0°。

Tips：如果舵机初始时已经在 0°了，则不转动。

（3）平台查看消息

在设备列表中，我们查看 device4 中的消息，可以看到，有一条“Hello”的消息产生。

（4）发送消息并观察舵机

在输入框中发送“1”，可以看到舵机转至 170°，同时蜂鸣器发出嘟的一声，随后发送“2”，可以看到舵机回到了 10°。

当前主题：df/device4

发送消息 1 发送

(为消息加上->前缀代表此消息为纯指令消息，不会被存入数据库。例如"->off")

开始时间 结束时间 100条 查询 导出查询结果 隐藏/显示图表 自动刷新消息

当前主题：df/device4

发送消息 2 发送

(为消息加上->前缀代表此消息为纯指令消息，不会被存入数据库。例如"->off")

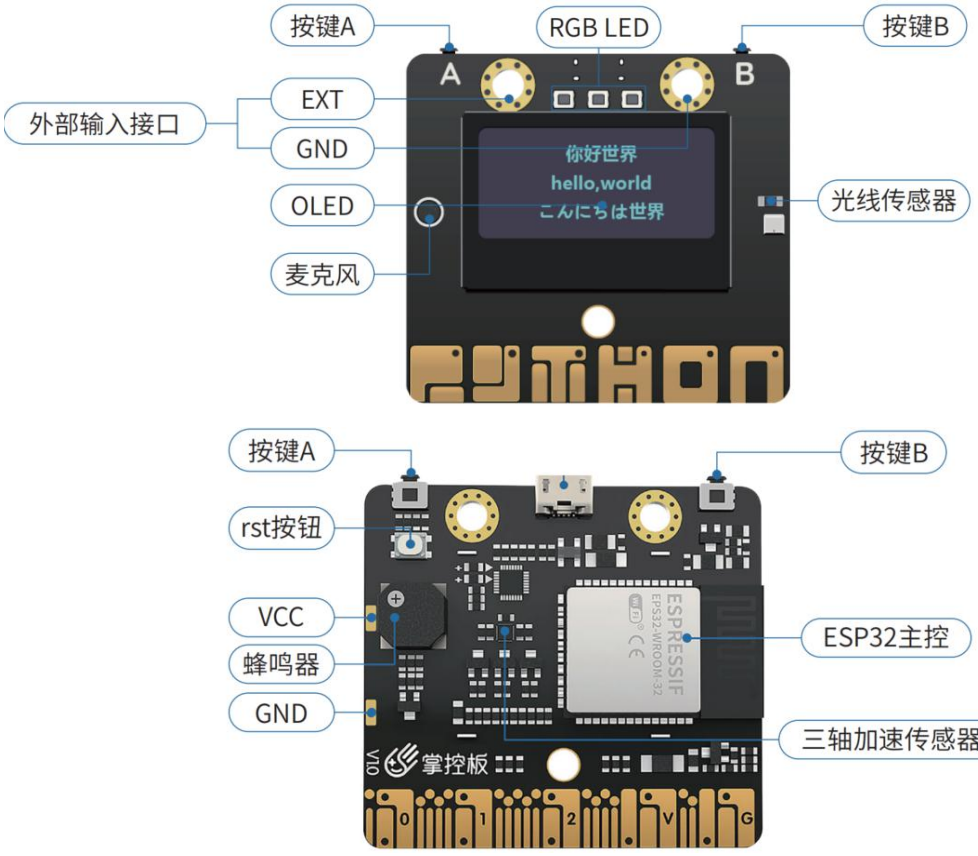
开始时间 结束时间 100条 查询 导出查询结果 隐藏/显示图表 自动刷新消息

附录、知识卡片

一、掌控板

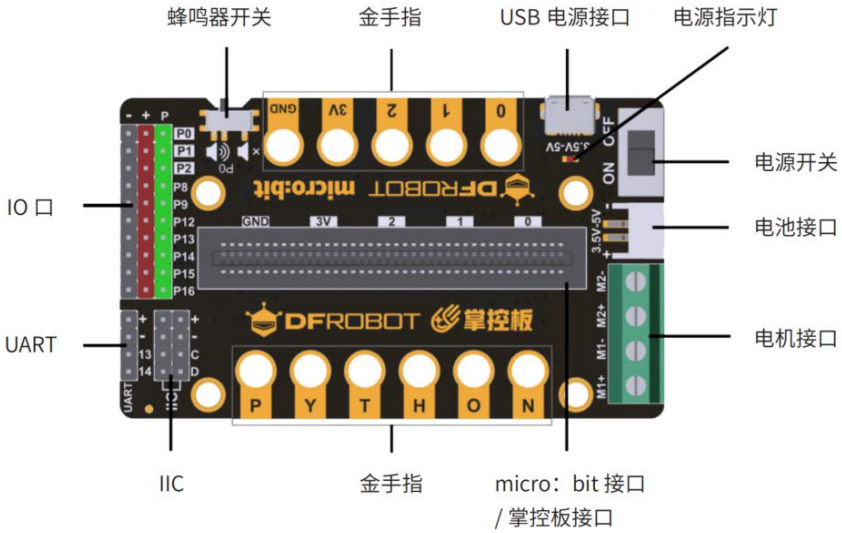
掌控板由创客教育专家委员会推出，是一款专为编程教育而设计的开源硬件!

掌控板上集成了 OLED 显示屏、RGB 灯、加速度计、麦克风、光线传感器、蜂鸣器、按键开关、触摸开关、金手指外部拓展接口等。同时，它支持图形化及 python 代码编程，可实现智能机器人、创客智造作品等智能控制类应用。利用掌控板上丰富的传感器，结合它小尺寸的特点还可以做很多之恩穿戴、电子饰品等各种 DIY 作品应用。掌控板外观布局如下，



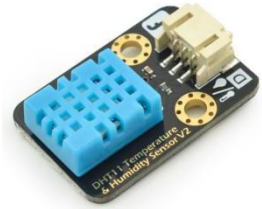
二、I/O 扩展板

这里的“I、O”分别是“input”输入和“output”输出的缩写，I/O 扩展板的作用是将掌控板上的 IO 口以引脚口的形式引出，方便我们接入传感器、执行器等模块。IO 口与掌控板上的接口一一对应。



三、DHT11 温湿度传感器

DHT11 温湿度传感器是传感器的一种，它的功能则是用于采集温度和湿度数据，让我们可以通过掌控板看到环境的温湿度数据及变化。



四、NFC 近场通信模块

近场通信（Near Field Communication，NFC），又称近距离无线通信，是一种短距离的高频无线通信技术，允许电子设备之间进行非接触式点对点数据传输（在十厘米内）交换数据。

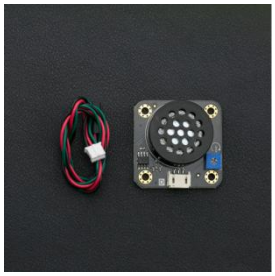
NFC 近场通讯模块采用 NXP PN532 进口高集成 NFC 通信芯片，支持市面上常见的各类 MIFARE Classic S50/S70 系列（即 M1 卡）和 NTAG21x 系列等工作频率在 13.56Mhz 的 NFC 电子标签或卡片。采用 Gravity 标准的 PH2.0-4P 接口，除了让接线更方便，在保留 UART 的基础上额外复合了 I2C，通过开关轻松切换两种不同的接口，使用更灵活。当使用 UART 串口时，可以利用市面上常见的 USB to UART 转换器和第三方上位机软件轻松读写操作各类 NFC 卡。当使用 I2C 接口时，则可用于 Arduino、micro:bit、FireBeetle ESP32、FireBeetle ESP8266 等各类 3V3/5V 主控系统。

五、NFC 圆形配件

这是一款只有硬币大小的 NFC 标签，你可以将其别再手机、背包上作为挂件，既可以作为装饰又简单方便。本标签内嵌 NXP S50 芯片和天线，能被 NFC 读写设备快速识别。NXP 出厂的 Mifare 卡是目前世界上使用量最大、技术最成熟、性能最稳定、内存容量最大的一种感应式智能 IC 卡。
此类 Mifare 1 卡几乎能被所有 13.56MHZ RFID/NFC 兼容设备所识别。该芯片可以通过 EEPROM 来读写多达 1KB 的数据，修改次数可达 100000 次。每个标签带有 4B 的 ID,用于不同标签之间的识别，并且不可被更改。用户可以使用 The NXP TagWriter application 来整合以及存储通讯录、书签、短信和邮件等信息到 NFC 标签当中。

六、带功放喇叭模块

新款带功放喇叭模块，基于高保真 8002 功放芯片制作，在输出音乐的同时，能够确保输出音频不失真。支持音量调节功能，可通过电位器调解输出音量大小。支持宽电压输入，模块可以工作在 2~5.5V 电压环境下，兼容 3.3V 和 5V 主控器。



七、DMS-MG90 金属 9g 舵机（1.8Kg)

舵机是一种可以指定控制位置（角度）的电机，可以通过程序来指定控制舵机旋转的角度。我们最常用的舵机大多最大旋转角度是 0°~180°，也有 90°或者其他角度的。也有比较特殊的 360°舵机，但是 360°舵机不能够控制其旋转到指定的角度。这里使用的是 180°舵机。

八、物联网相关

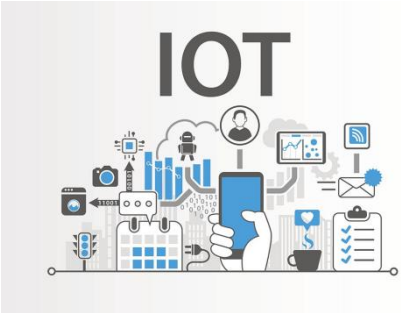
知识链接

SIoT

简介： SIoT 是一个为中小学 STEM 教育定制的跨平台的开源 MQTT 服务器程序，S 指科学（Science）、简单（Simple）的意思。

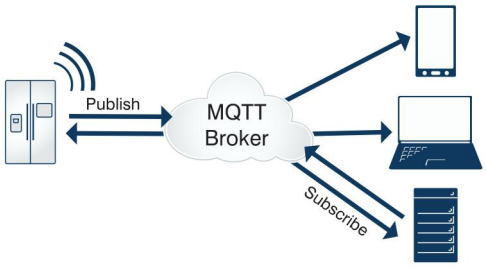
物联网

简介： 物联网（Internet of Things，缩写 IoT）是互联网的一个延伸，互联网的终端是计算机（PC、服务器），而物联网的终端是硬件设备，无论是家电、工业设备、汽车、监测仪器，所有这些终端都可以互联，可以总结为万物互联。



MQTT

简介： MQTT（Message Queuing Telemetry Transport，消息队列遥测传输协议）是一个基于客户端-服务器的消息发布/订阅传输协议。MQTT 协议是轻量、简单、开放和易于实现的，这些特点使它适用范围非常广泛。



九、BXY 软件简介

掌控板和电脑间，我们可以用 USB 线这一硬件构建物理连接，但仅仅做到这一步就好比是买来了各式各样的硬件、组装好了电脑，但是没有软件无法使用这些硬件。那么要如何建立这两者之间信息的连接呢?

答案便是 BXY！它为两者架起了虚拟的桥梁，从而实现代码的烧录、串口的连接，实时数据流的传输等功能。

BXY 是 BXY Python Editor 的缩写，它是一款运行于 Windows 平台的 MicroPython 编程 IDE，界面简洁，操作便利。内置了许多基础操作库，为众多 MicroPython 爱好者提供了一个简洁实用的平台。我们需要下载 MicroPython 编程 IDE-BXY，说明及**下载地址**：https://bxy.dfrobot.com.cn/

在 BXY 网站中，选择“下载 BXY”，然后点击 exe 文件，即可下载。下载完成后，双击 exe 文件安装即可。

注：关于掌控板、BXY 软件的具体使用，在教程中再详细介绍。