

Micropython 温度控制系统

学 院（系）： 软件学院
专 业： 软件工程
班 级： 电计 1801
学 生 姓 名： 聂闻华
学 号： 201999903
完 成 日 期： 2020-08-28

大连理工大学

目 录

| | | |
|-------|--------------------------|----|
| 1 | 文献综述..... | 1 |
| 1.1 | 论文相关背景 | 1 |
| 1.2 | 研究意义 | 1 |
| 2 | Micropython 温度控制系统 | 2 |
| 2.1 | 项目概述 | 2 |
| 2.1.1 | 项目开发背景..... | 2 |
| 2.2 | 项目设计 | 3 |
| 2.2.1 | 人体感应传感器 | 3 |
| 2.2.2 | 超声波传感器..... | 6 |
| 2.2.3 | 温湿度传感器..... | 9 |
| 2.2.3 | 舵机 | 11 |
| 2.3 | 项目实施 | 13 |
| 2.3.1 | 流程图 | 13 |
| 2.3.2 | 数据表 | 14 |
| 2.4 | 主要代码 | 15 |
| 2.5 | 实验结果 | 18 |
| 3 | 源代码..... | 22 |
| 4 | 总结 | 28 |
| | 参 考 文 献..... | 29 |

1 文献综述

1.1 论文相关背景

单片机嵌入式编程经历了汇编、C 语言的发展历程，可以说是一次编程革命，其背后的原因是单片机的速度越来越快，集成度越来越高。而这一趋势并没停止，摩尔定律仍然适用。在未来，单片机上很可能直接跑机器语言。在 2014 年，MicroPython 在英国诞生了，对于电子爱好者来说无疑拉开了新时代的序幕，用 python 这个每年用户量不断增长的编程语言来开发嵌入式，加上无数开源的函数模块，让嵌入式开发变得从未如此的简单。MicroPython 致力于兼容 Python。因此，我们在学习完 MicroPython 后除了可以开发有趣的电子产品外，还可以继续深入使用 Python 语言去开发后台、人工智能等领域。

1.2 研究意义

“MicroPython 是 Python 3 编程语言的精简高效实现，包括 Python 标准库的一小部分，并且经过优化，可以在 Microcontrollers（微控制器）和有限的环境中运行。

MicroPython 包含许多高级功能，如交互式提示，任意精度整数，闭包，列表理解，生成器，异常处理等。然而它非常紧凑，可以在 256k 的代码空间和 16k 的 RAM 内运行。MicroPython 旨在尽可能与普通 Python 兼容，以便您轻松地将代码从电脑传输到微控制器或者嵌入式系统。”

官方说明向我们阐述了，MicroPython 是指在微控制器上使用 Python 语言进行编程，学习过单片机和嵌入式开发的小伙伴应该都知道早期的单片机使用汇编语言来编程，随着微处理器的发展，后来逐步被 C 所取代，现在的微处理器集成度越来越高了，那么我们现在可以使用 Python 语言来开发了。Python 的强大之处是封装了大量的库，开发者直接调用库函数则可以高效地完成大量复杂的开发工作。MicroPython 保留了这一特性，常用功能都封装到库中了，以及一些常用的传感器和组件都编写了专门的驱动，通过调用相关函数，就可以直接控制 LED、按键、伺服电机、PWM、AD/DA、UART、SPI、IIC 以及 DS18B20 温度传感器等等。以往需要花费数天编写才能实现的硬件功能代码，现在基于 MicroPython 开发只要十几分钟甚至几行代码就可以解决。真可谓：“人生苦短，我用 Python 和 MicroPython”

2 Micropython 温度控制系统

2.1 项目概述

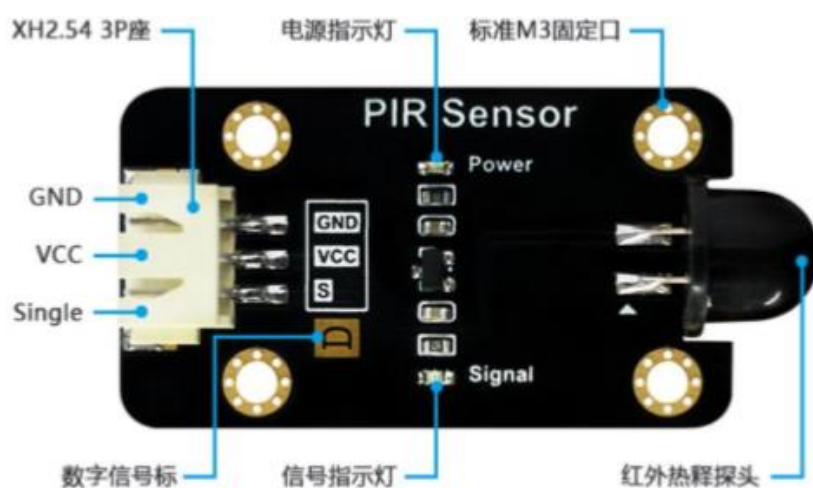
2.1.1 项目开发背景

在目前国家大力发展物联网的大环境之下，智能家居这一部分引起了重视。试想，当人工作了一天，疲惫的回到家中，却又被家中闷热的环境引得心情烦躁。所以我想开发一套自动控制空调进行温度调节的系统。通过人体感应传感器、超声波传感器、温湿度传感器相结合，检测人的位置，控制空调开关。这样既保证了人在进入室内时可以进行温度调节，同样离开时不会对电力造成浪费。

2.2 项目设计

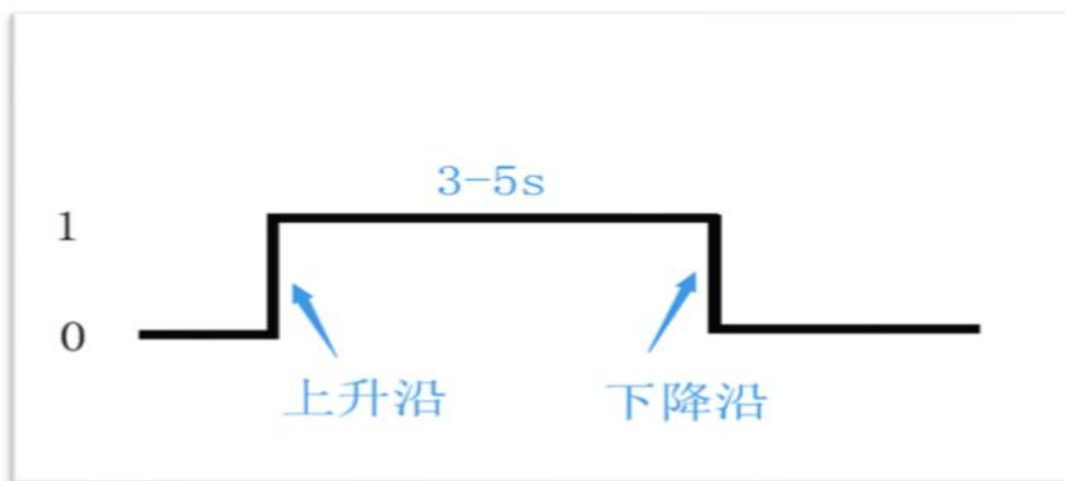
2.2.1 人体感应传感器

人体感应传感器，在室内安防应用非常普遍，其原理是由探测元件将探测到人体的红外辐射转变成微弱的电压信号，经过放大后输出。为了提高探测器的探测灵敏度以增大探测距离，一般在探测器的前方装设一个塑料的菲涅尔透镜，它和放大电路相配合，可将信号放大 70dB 以上，这样就可以测出 5~10 米范围内人的行动。



| 功能参数 | |
|------|--|
| 供电电压 | 3.3-5V |
| 工作电流 | <20 mA |
| 工作温度 | -20 至 65℃ |
| 接口定义 | XH2.54 防呆接口（3Pin）【GND、VCC、Single】 |
| 输出信号 | 数字信号： 检测到人体：高电平 3.3V；持续 3-8 秒 未检测到人体：低电平 0V。 |
| 感应角度 | 100° |
| 感应距离 | 8 米 |
| 模块尺寸 | 4.5*2.5cm |

从上表可以知，当检测到人体红外的时候，传感器的输出信号为高电平并持续 3-5 秒。如下图所示



由此可见，可以使用外部中断结合上升沿的出发方式来编程实现相关功能。传感器的输出引脚连接 Sensor1 接口，即连接到 pybaord 的“Y11”引脚。编程方法可以是当

“Y11”引脚产生中断时候，说明传感器检测到人体红外线，此时可以在 OLED 上显示相关信息。

因为人体感应传感器使用的是外部中断的写法，同时第一步也是对室内是否有人进行判断，所以我将人体感应设为第一个判断条件。外部中断的方式保证了资源的不浪费，当人进入时，条件触发，才会进行下面的动作，否则会造成超声波传感器和温湿度传感器不停地进行实时检测，出现了资源浪费的现象。

2.2.2 超声波传感器

超声波传感器是一款测量距离的传感器。其原理是利用声波在遇到障碍物反射接收结合声波在空气中传播的速度计算的得出。在测量、避障小车，无人驾驶等领域都有相关应用。



| 功能参数 | |
|-------|--|
| 传感器型号 | HCSR04 |
| 供电电压 | 3.3-5V |
| 工作电流 | <20 mA |
| 工作温度 | -20 至 85℃ |
| 接口定义 | XH2.54 防呆接口（4Pin） 【GND、VCC、ECHO、TRIG】 |
| 通讯信号 | IO 数字接口 |
| 测量距离 | 2-450 cm |
| 测量精度 | 0.5 cm |
| 整体尺寸 | 5.5*4.5*3.0 cm |

超声波传感器模块使用两个 IO 口分别控制超声波发送和接收，工作原理如下：

1. 给超声波模块接入电源和地；
2. 给脉冲触发引脚（trig）输入一个长为 20us 的高电平方波；
3. 输入方波后，模块会自动发射 8 个 40KHz 的声波，与此同时回波引脚（echo）端的电平会由 0 变为 1；（此时应该启动定时器计时）
4. 当超声波返回被模块接收到时，回波引脚端的电平会由 1 变为 0；（此时应该停止定时器计数），定时器记下的这个时间即为超声波由发射到 返回的总时长；
5. 根据声音在空气中的速度为 340 米/秒，即可计算出所测的距离。

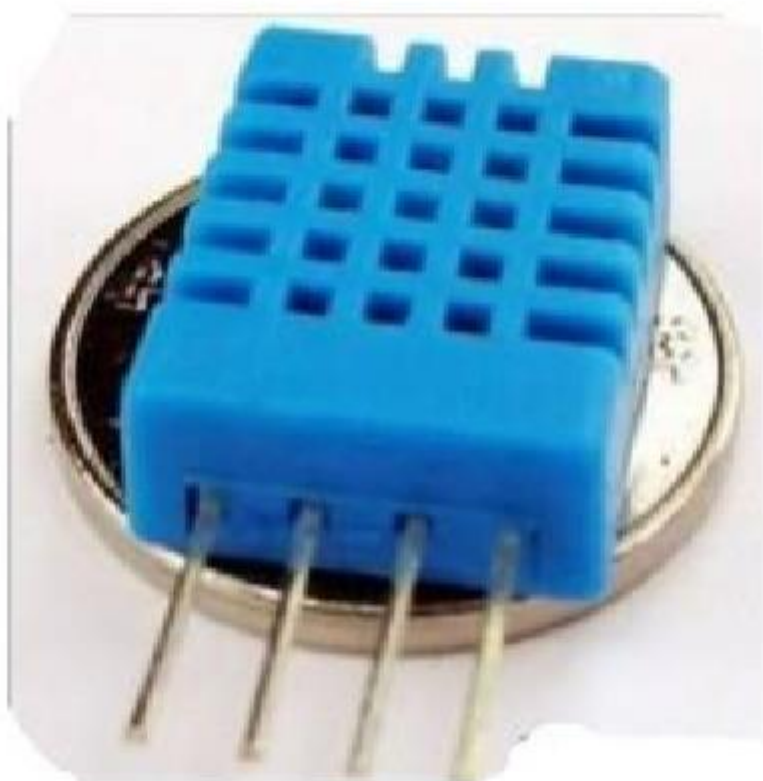
经过在家中的环境模拟可知超声波传感器设置的范围为 100cm。起初调试时设置为 50cm，发现距离太小，超声波传感器无法判断人是否离开，当大于 100cm 时，成年人的步距为 45-60cm，相当于人需要两步才能远离，这样既有了开启空调的提前量，同时还保证了人在近距离有足够多的活动范围。当然此范围需要根据室内具体环境进行调整。

超声波传感器是一款测量距离的传感器。其原理是利用声波在遇到障碍物反射接收结合声波在空气中传播的速度计算的得出。在测量、避障小车，无人驾驶等领域都有相关应用。

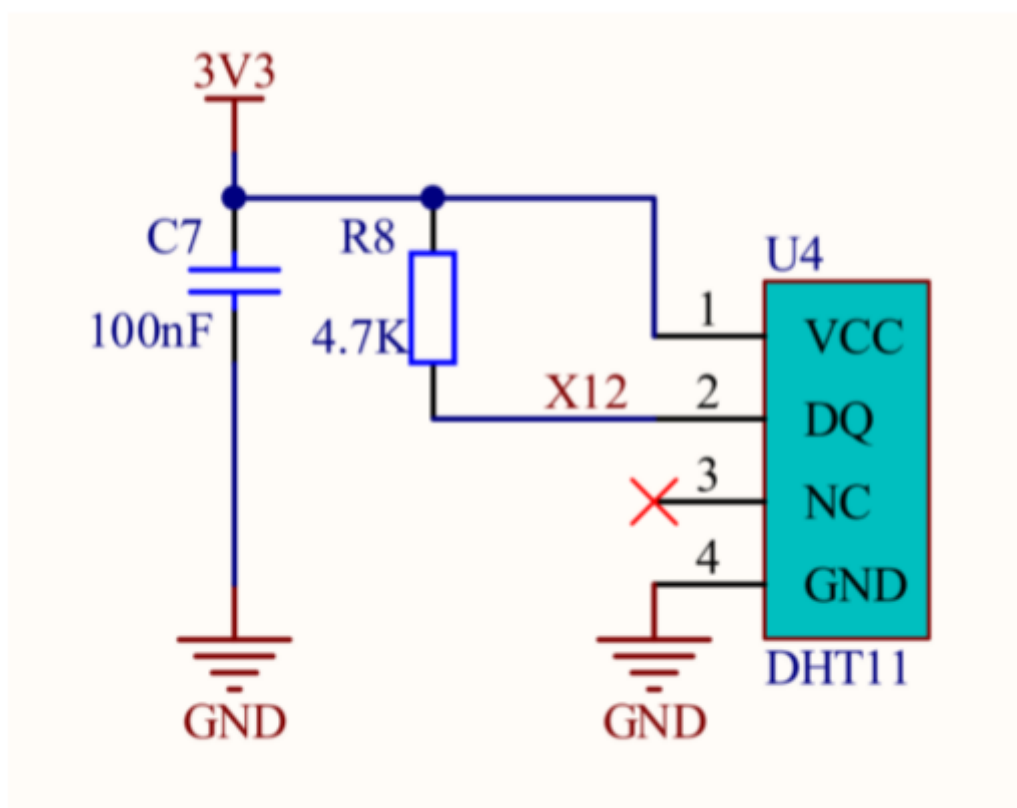
2.2.3 温湿度传感器

温湿度也是我们日常非常常见的指标，我们使用的是 DHT11 数字温湿度传感器。这是一款含有已校准数字信号输出的温湿度复合传感器，它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性和卓越的长期稳定性。

DHT11 具有小体积、极低的功耗，信号传输距离可达 20 米以上，使其成为给类应用甚至最为苛刻的应用场合的最佳选择。产品为 4 针单排引脚封装，连接方便。



DHT11 虽然有 4 个引脚，但其中第 3 个引脚是悬空的，也就是说 DHT11 也是单总线的传感器，只占用 1 个 IO 口。

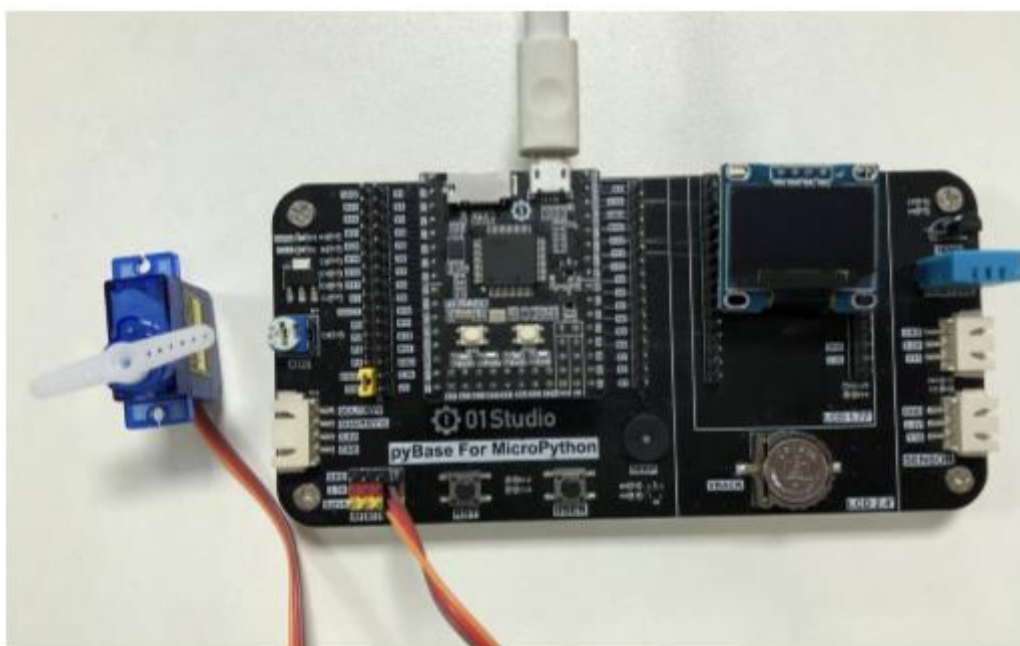


从接线图可以看到 DHT11 连接到 pyboard 的 ‘X12’ 引脚，可以针对该引脚编程来驱动 DHT11 传感器。

通过查阅资料，可以发现使人感到舒适的室内温度为 18-23 度。当人体感应传感器检测到有人时，温湿度传感器才会开始检测室内数据。并且空调开启后，温度并不能瞬间下降，所以我们需要选取一个中间值，经过实验发现，20 度可以保证空调关闭后，温度可以保持一段时间才回到 23 度，同时人不会感到太冷。

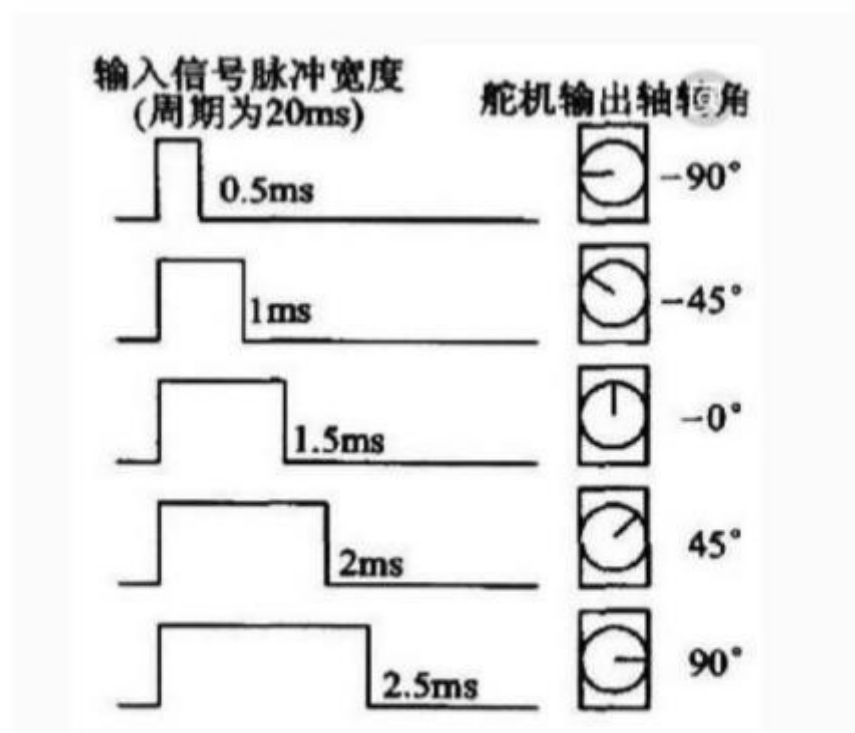
2.2.3 舵机

舵机又叫伺服电机，是一个可以旋转特定角度的电机，可转动角度通常是 90° 、 180° 和 360° （ 360° 可以连续旋转）。我们看到的机器人身上就有非常多的舵机，它们抬手或者摇头的动作往往是通过舵机完成，因此机器人身上的舵机越多，意味着动作越灵活。



实验所使用的是 SG90 舵机，这是一个 180° 舵机，按动开关并不需要特别大的动作幅度，所以舵机应该可以实现。

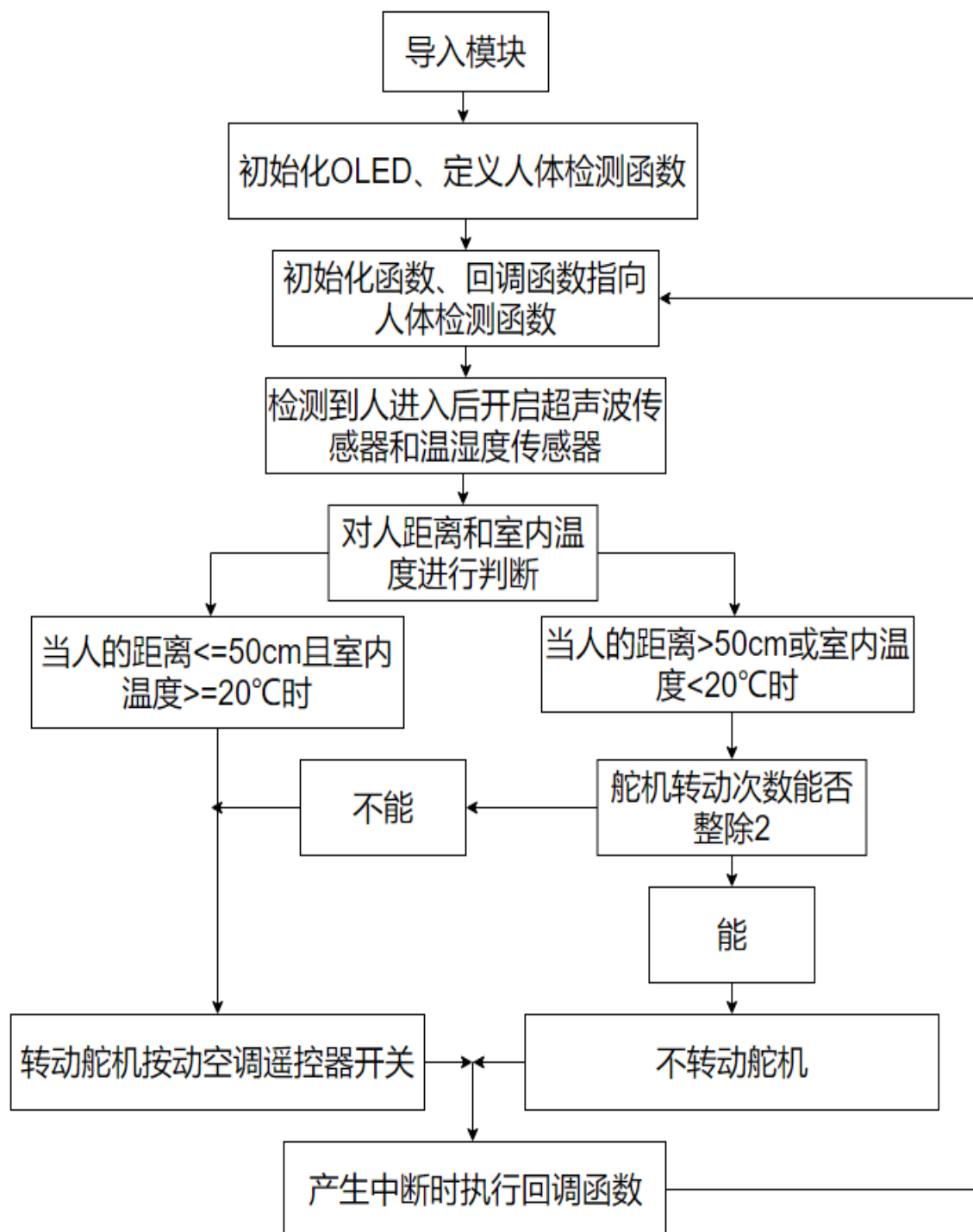
180° 舵机的控制一般需要一个 20ms 左右的时基脉冲，该脉冲的高电平部分一般为 0.5ms - 2.5ms 范围内的角度控制脉冲部分，总间隔为 2ms 。以 180° 度角度伺服为例，在 MicroPython 编程对应的控制关系是从 -90° 至 90° ，示例图如下：



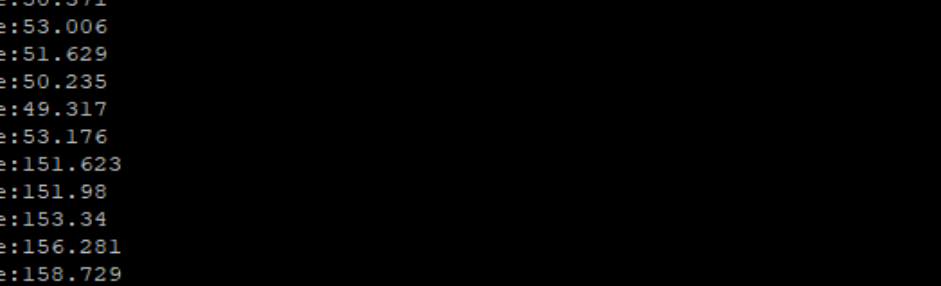
实验较为理想的模式是通过蓝牙或无线模板，通过远程对遥控器实现操作，但是stm32和传感器有限，只能通过舵机实现物理层面的操作模拟，以达到按动开关的效果，但是只是进行了模拟过程，具体的按动遥控器还需调试。

2.3 项目实施

2.3.1 流程图



2.3.2 数据表



```
COM5 - PuTTY
distance:50.371
distance:53.006
distance:51.629
distance:50.235
distance:49.317
distance:53.176
distance:151.623
distance:151.98
distance:153.34
distance:156.281
distance:158.729
distance:155.329
distance:151.929
distance:151.946
distance:152.881
distance:151.946
distance:150.688
distance:151.623
distance:153.646
distance:152.864
distance:29.342
distance:38.165
distance:32.895
distance:36.142
```

[illegible]

因为温湿度传感器的特性，需要设定延时，同时温度并没有变化的十分频繁，所以温度数据显示主要是在显示屏上，并不会频繁的回传数据。同时因为是使用的 micropython 进行代码编写，可以在代码中直接对数据进行处理判断，所以未将数据统一生成表。

2.4 主要代码

```
60  #pyBoard I2C初始化: sda--> X12, scl --> X11
61  i2c = I2C(sda=Pin("Y8"), scl=Pin("Y6"))
62  dac = DAC(1)      #定义DAC对象名字为dac, 输出引脚为X5
63  #定义4组频率值: 1Hz、200Hz、1000Hz、5000Hz
64  freq=[1000, 1, 1000, 5000]
65  |
66  # 定义8位精度下方波的值。0、255分别对应输出0V、3.3V。需要定义成字节数组。
67  buf = bytearray(2)
68  buf[0]=0
69  buf[1]=255
70  #OLED显示屏初始化: 128*64分辨率, OLED的I2C地址是0x3c
71  oled = SSD1306_I2C(128, 64, i2c, addr=0x3c)
```

对 OLED 屏和蜂鸣器进行初始化。

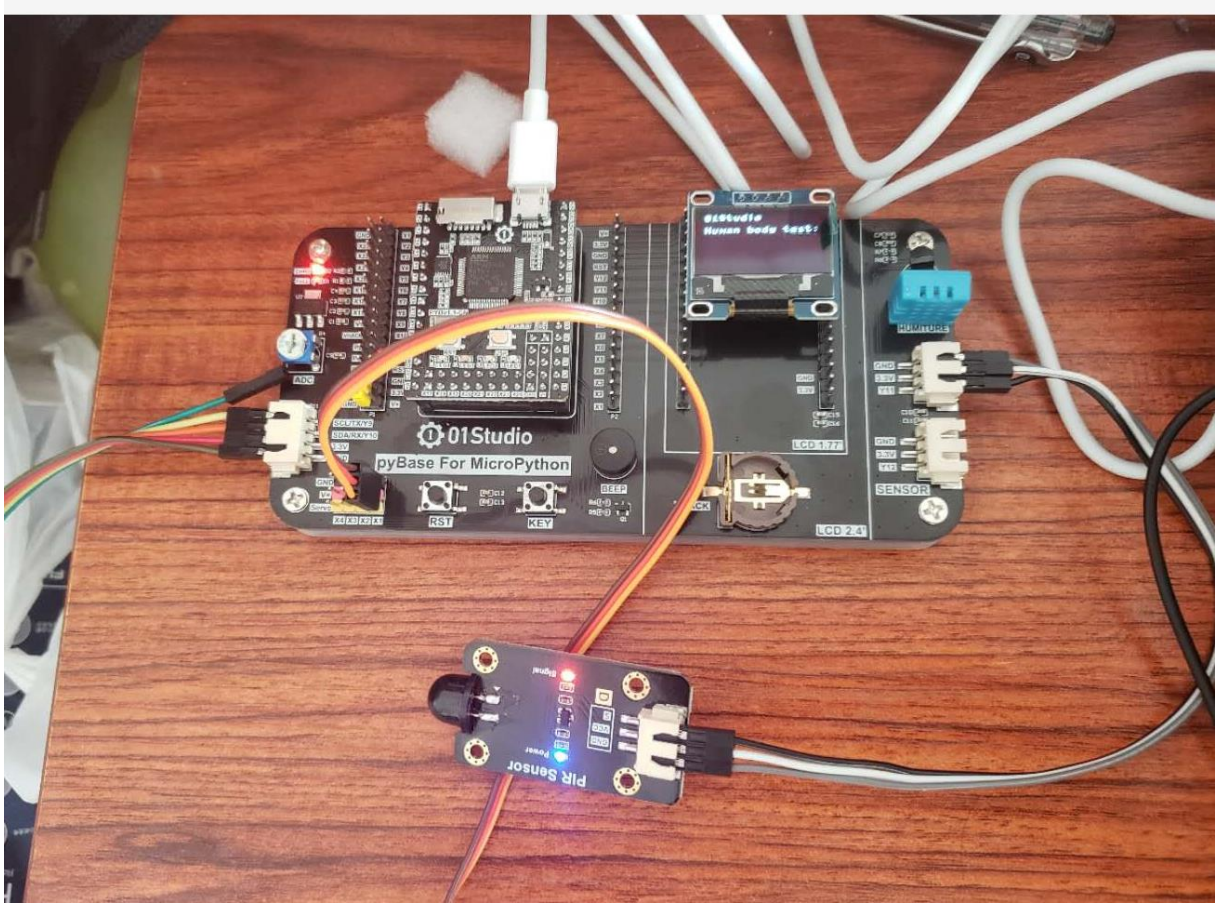
```
99  def ifTurnON():
100      global countTimes
101      global moveFlag
102      countTimes = True
103      if te > 20:
104          if moveFlag is False:
105              dac.write_timed(buf, freq[0] * len(buf), mode=DAC.CIRCULAR)
106              # 将LED(4)-"B4"配置成推挽输出模式
107              p_out = Pin('X5', Pin.OUT_PP)
108              s1.angle(angle[4])
109              delay(1000)
110              s1.angle(angle[0])
111              moveFlag = True
112              dac.write_timed(buf, freq[1] * len(buf), mode=DAC.CIRCULAR)
113          else:
114              dac.write_timed(buf, freq[0] * len(buf), mode=DAC.CIRCULAR)
115              # 将LED(4)-"B4"配置成推挽输出模式
116              p_out = Pin('X5', Pin.OUT_PP)
117              s1.angle(angle[4])
118              delay(1000)
119              s1.angle(angle[0])
120              dac.write_timed(buf, freq[1] * len(buf), mode=DAC.CIRCULAR)
```

设定 ifTurnON 函数，来对温湿度传感器传回的温度进行判断，如果温度达到 20 度以上则转动舵机并且舵机转动的同时，蜂鸣器以 1000HZ 响声提示舵机转动。

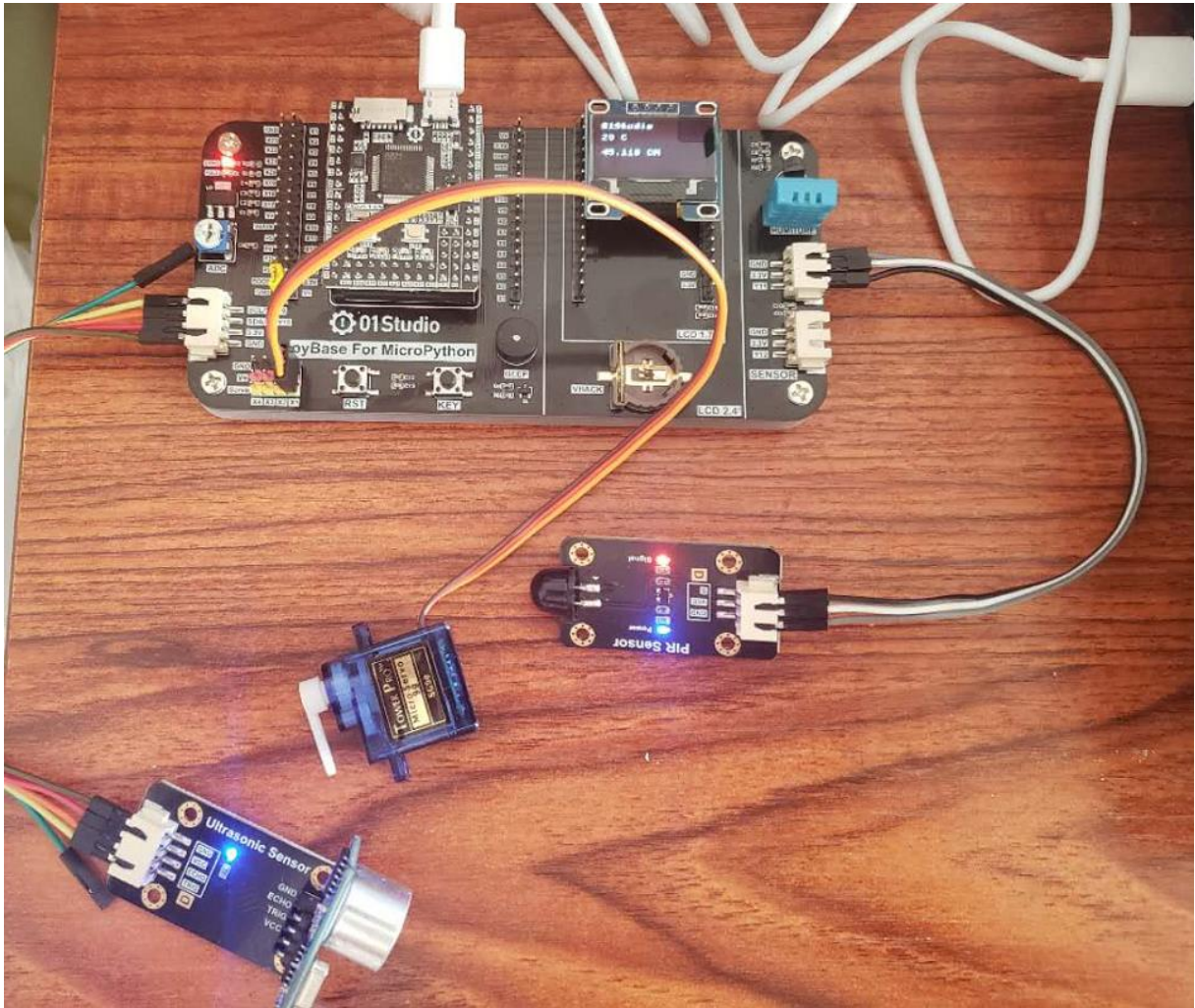
```
145     if p_in.value() == 1 and Distance < 100:
146         ifTurnON()
147     elif p_in.value() == 1 and Distance > 100:
148         moveFlag = False
149         if countTimes:
150             dac.write_timed(buf, freq[0] * len(buf), mode=DAC.CIRCULAR)
151             # 将LED(4)-"B4"配置成推挽输出模式
152             p_out = Pin('X5', Pin.OUT_PP)
153             s1.angle(angle[4])
154             delay(1000)
155             s1.angle(angle[0])
156             countTimes = False
157             dac.write_timed(buf, freq[1] * len(buf), mode=DAC.CIRCULAR)
158         else:
159             s1.angle(angle[0])
160     else:
161         moveFlag = False
162         nothingFlag = False
```

用两层 if 循环嵌套，对人体感应传感器和超声波传感器进行判断，当人体感应传感器感应到人时，进行距离判断，确保在有人的情况下才会按动开关，否则不会开启。和温湿度传感器设定相同，在舵机转动时会有蜂鸣器切换至 1000HZ 响声提示。由于蜂鸣器示例程序是以切换至 1HZ 来假使蜂鸣器关闭，所以采用了同样的方式。具体关闭蜂鸣器的方法没有找到，希望之后能够改进。

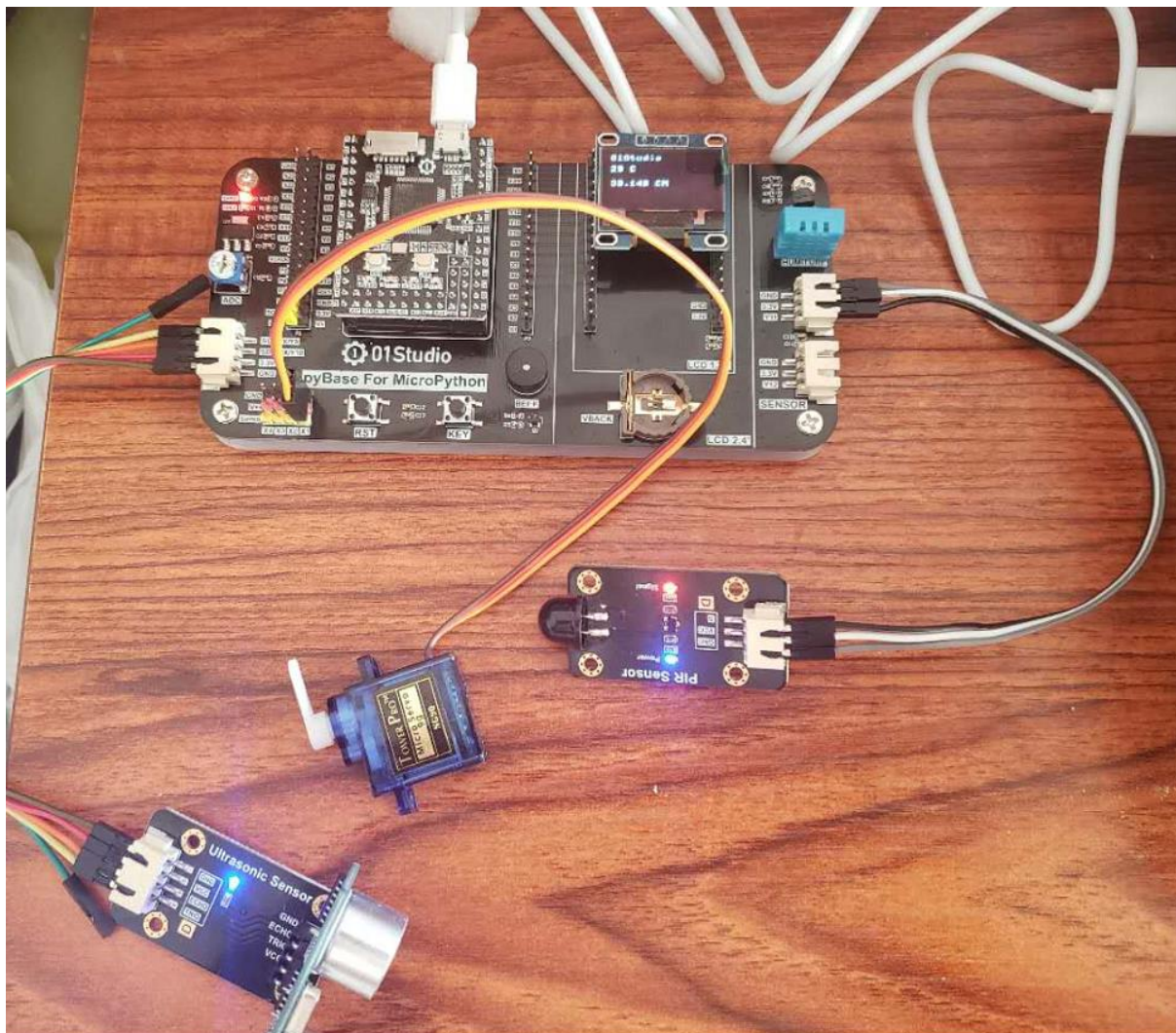
2.5 实验结果



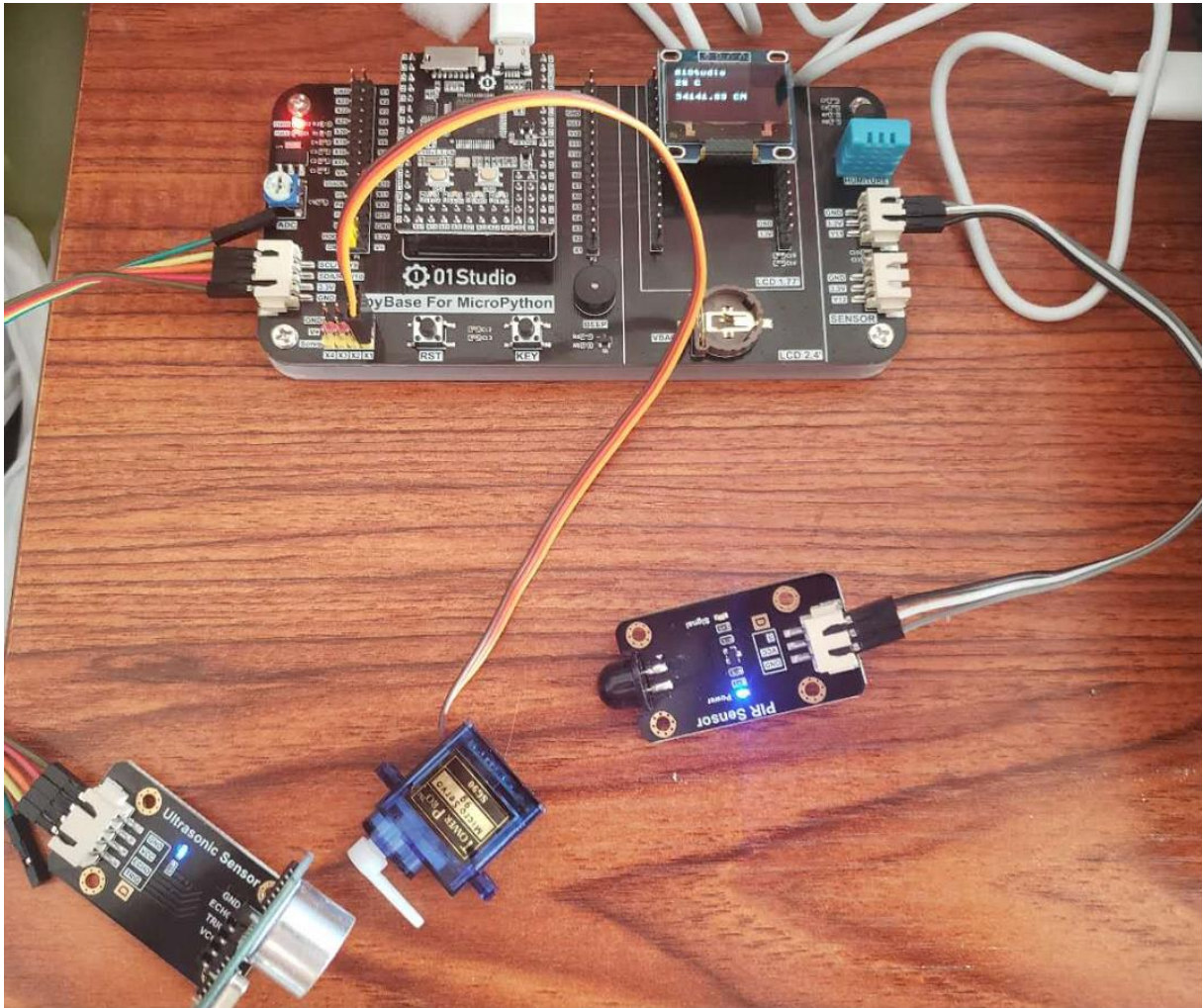
检测到人体之前，OLED 显示屏显示 human body test 字样。



检测到人体后，进入循环，显示屏显示温度和距离。同时进行判断。



判断距离 $<100\text{cm}$ 后，舵机转动，同时蜂鸣器响起。



判断距离 $>100\text{cm}$ 后，舵机再次转动，同时蜂鸣器响起。

3 源代码

```
#导入相关模块
from pyb import Servo,Switch
from machine import Pin,I2C
from ssd1306 import SSD1306_I2C
from pyb import Servo,Switch
from pyb import Pin,delay
from time import sleep_us,ticks_us,sleep
from pyb import dht_readinto
from pyb import DAC

class DHTBase:
    def __init__(self, pin):
        self.pin = pin
        self.buf = bytearray(5)

    def measure(self):
        buf = self.buf
        dht_readinto(self.pin, buf)
        if (buf[0] + buf[1] + buf[2] + buf[3]) & 0xff !=
buf[4]:
            raise Exception("checksum error")

class DHT11(DHTBase):
    def humidity(self):
        return self.buf[0]

    def temperature(self):
        return self.buf[2]

class DHT22(DHTBase):
    def humidity(self):
        return (self.buf[0] << 8 | self.buf[1]) * 0.1

    def temperature(self):
```



```

        t = ((self.buf[2] & 0x7f) << 8 | self.buf[3]) * 0.1
        if self.buf[2] & 0x80:
            t = -t
        return t

class HCSR04():
    def __init__(self, trig, echo):
        self.trig=trig
        self.echo=echo

    def getDistance(self):
        distance=0
        self.trig.value(1)
        sleep_us(20)
        self.trig.value(0)
        while self.echo.value() == 0:
            pass
        if self.echo.value() == 1:
            ts=ticks_us()           #开始时间
            while self.echo.value() == 1: #等待脉冲高电平结
束
                pass
            te=ticks_us()           #结束时间
            tc=te-ts                 #回响时间（单位 us，
1us=1*10^(-6)s）
            distance=(tc*170)/10000 #距离计算（单位
为:cm）
        return distance

#pyBoard I2C 初始化: sda--> X12, scl --> X11
i2c = I2C(sda=Pin("Y8"), scl=Pin("Y6"))
dac = DAC(1) #定义 DAC 对象名字为 dac，输出引脚为 X5
#定义 4 组频率值: 1Hz、200Hz、1000Hz、5000Hz
freq=[1000,1,1000,5000]

```

```
# 定义 8 位精度下方波的值。0、255 分别对应输出 0V、3.3V。需要
定义成字节数组。
buf = bytearray(2)
buf[0]=0
buf[1]=255
#OLED 显示屏初始化: 128*64 分辨率,OLED 的 I2C 地址是 0x3c
oled = SSD1306_I2C(128, 64, i2c, addr=0x3c)

#OLED 初始信息显示
oled.fill(0) # 清屏背景黑色
oled.text("01Studio", 0, 0) # 写入第 1 行内容
oled.text("Human body test:", 0, 15) # 写入第 2 行内容
oled.show() # OLED 执行显示

#初始化接口 trig='Y9',echo='Y10'
trig = Pin('Y9',Pin.OUT_PP)
echo = Pin('Y10',Pin.IN)
HC=HCSR04(trig,echo)

#创建 DTH11 对象 dt
dt=DHT11(Pin('X12'))
delay(1000) #首次启动停顿 1 秒然传感器稳定

sw = Switch() #定义按键对象名字为 sw
s1 = Servo(1) #构建舵机对象 s1, 输出引脚为 X1

#定义 5 组角度: -90、-45、0、45、90
angle=[-90,-45,0,45,90]
s1.angle(angle[0])

global countTimes
countTimes = False
print(1)
```

```

def ifTurnON():
    global countTimes
    global moveFlag
    countTimes = True
    if te > 20 :
        if moveFlag is False:
            dac.write_timed(buf, freq[0] * len(buf),
mode=DAC.CIRCULAR)
            # 将 LED(4)-"B4"配置成推挽输出模式
            p_out = Pin('X5', Pin.OUT_PP)
            s1.angle(angle[4])
            delay(1000)
            s1.angle(angle[0])
            moveFlag = True
            dac.write_timed(buf, freq[1] * len(buf),
mode=DAC.CIRCULAR)
        else:
            dac.write_timed(buf, freq[0] * len(buf),
mode=DAC.CIRCULAR)
            # 将 LED(4)-"B4"配置成推挽输出模式
            p_out = Pin('X5', Pin.OUT_PP)
            s1.angle(angle[4])
            delay(1000)
            s1.angle(angle[0])
            dac.write_timed(buf, freq[1] * len(buf),
mode=DAC.CIRCULAR)

#callback=lambda e: ifTurnOn() #执行 Display()函数
#ext = ExtInt(Pin('Y11'), ExtInt.IRQ_RISING, Pin.PULL_UP,
callback) #上升沿触发, 打开上拉电阻
p_in = Pin('Y11', Pin.IN, Pin.PULL_UP)
print(2)

```

```

global moveFlag
moveFlag = False

while True:
    Distance = HC.getDistance() #测量距离
    print('distance:'+str(Distance))
    dt.measure()                #温湿度采集
    te=dt.temperature()         #获取温度值
    dh=dt.humidity()            #获取湿度值
    delay(1000)
    oled.fill(0) # 清屏,背景黑色
    oled.text('01Studio', 0, 0)
    oled.text(str(te)+' C', 0, 15)
    # OLED 显示距离
    oled.text(str(Distance) + ' CM', 0, 35)

    oled.show()
    if p_in.value() == 1 and Distance < 100:
        ifTurnON()
    elif p_in.value() == 1 and Distance > 100:
        moveFlag = False
        if countTimes:
            dac.write_timed(buf, freq[0] * len(buf),
mode=DAC.CIRCULAR)
            # 将 LED(4)-"B4"配置成推挽输出模式
            p_out = Pin('X5', Pin.OUT_PP)
            s1.angle(angle[4])
            delay(1000)
            s1.angle(angle[0])
            countTimes = False
            dac.write_timed(buf, freq[1] * len(buf),
mode=DAC.CIRCULAR)
        else:
            s1.angle(angle[0])
    else:

```

```
moveFlag = False  
nothingFlag = False
```

4 总结

通过这次实验，我学习到了 micropython 的应用，也发现了 micropython 的便捷和库的强大，以往用 c 等其他语言，需要从底层自己编写，而 micropython 有着现成的库，只需要在代码一开始进行引入，极大地简化了编写代码时的工作量。希望以后能够多接触这种便捷的嵌入式开发模式。

参 考 文 献

- [1] MicroPython 从 0 到 1[C]. 01Studio, 2019.