说明文档

小组成员：

第五小组	2152590	王琳
	2152354	张恺瑞
	2153681	蔡名雅
	2054099	叶洁颖

一、项目背景/内容

MQTT（Message Queuing Telemetry Transport）是一种基于**发布/订阅**模型的消息传输协议（基于 TCP/IP 协议栈构建的异步通信消息协议），具有低开销、高拓展、协议简单、双向通信等优点，可以在不可靠的网络环境中进行扩展，适用于设备硬件存储空间或网络带宽有限的场景。

使用 MQTT 协议，消息发送者与接收者不受时间和空间的限制，可以利用较少的设备资源和网络资源实现可靠、高效的长连接，被广泛应用于物联网领域。

1. **发布端：**从数据文件读取有关传感器数据或其它类型数据通过 MQTT 代理进行发布；
2. **MQTT 代理/服务端：**部署 MQTT 服务器，实现 MQTT 代理和服务；
3. **订阅端：**订阅有关主题数据，通过与 MQTT 代理/服务端连接，接收有关主题数据；
4. **数据处理端：**
    a. 通过订阅方式接收一定的发布数据，进行本地存储，然后进行处理分析，以图表方式展示接收数据和分析结果，模拟分析预测数据曲线与历史数据曲线应具备良好的拟合；
5. **前端界面：**订阅端、发布端和数据处理端需要设计相关图形界面，便于数据发布和订阅，具体界面展示内容和方式尽量美观合理；

注：服务端，代理端和采集发布端可以部署在局域网内不同主机；

二、整体架构设计

本项目将以阿里云的物联网平台为主要服务器部署载体，延伸连接两端的设备（发布端）与业务服务器（订阅端和数据分析）。整体架构如下图所示：

因此为了完成整个项目的搭建工作，我们开通了阿里云的物联网平台公共实例服务，其后主要搭建流程如下所示：

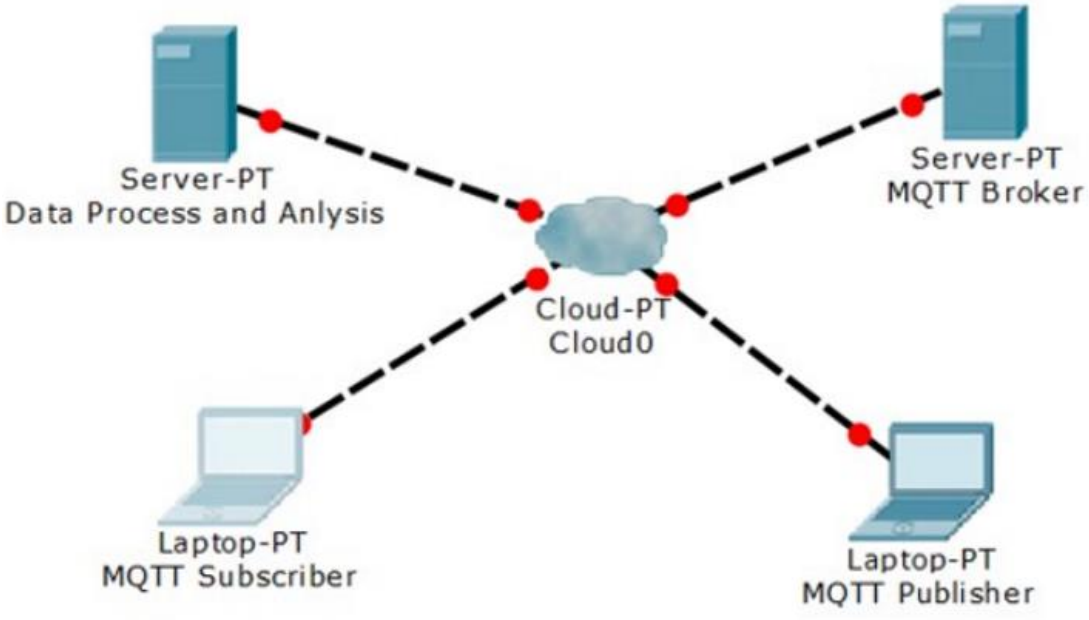
1. 设备上报数据到阿里云物联网平台。

    a. 创建产品与设备：在物联网平台上为设备注册一个身份，获取设备证书信息（ProductKey、DeviceName 和 DeviceSecret）。该证书信息将烧录到设备上，用于设备连接物联网平台时，进行身份认证。

    b. 为产品定义物模型：从属性、服务和事件三个维度定义产品功能。物联网平台根据我们为温湿度气压这些数据定义的功能构建出产品的数据模型，用于云端与设备端进行指定数据通信。

    c. 建立设备与平台的连接：选用 Python 语言开发设备端 SDK，传入设备的证书信息，将设备连接到物联网平台，使设备激活。

2. 服务端订阅设备消息：服务端通过订阅消息类型，接收设备相关消息，如设备上下线通知、设备上报消息等。



- **MQTT 客户端（Client）和 MQTT 代理者（Broker）。**
  - **MQTT 客户端（Client）：**本项目的客户端（Client）指接入物联网平台的设备。设备和用户的服务器不直接建立连接，而是通过代理者（Broker）进行通信。
  - **MQTT 代理者（Broker）：**本项目的 MQTT 代理者（Broker）指阿里云物联网平台。代理者（Broker）是设备和业务服务器消息通信的中介，解耦了设备和业务服务器，实现了设备和业务服务器之间的**异步通信**。
- **MQTT 协议消息的组成部分**
  - **主题（Topic）：**使用正斜杠（/）作为分隔符构造字符串，例如 `/sys/{product_key}/{device_name}/thing/event/property/post`，订阅该 Topic 的所有设备都会收到消息。
  - **消息内容（Payload）：**消息的具体内容。
- **MQTT 协议消息的传输过程，以业务服务器向设备下发消息为例**
  a. 设备订阅相应的主题（Topic）。
  b. 服务器发送消息给物联网平台。
  c. 物联网平台接收消息，根据消息的主题（Topic）确定设备并发送消息。
  d. 业务服务器收到物联网平台的消息，确认消息已成功发送。
- **MQTT 协议的重要参数**
  - **消息服务质量（QoS）：**`QoS=0` 代表物联网平台只推送一次消息给订阅者，`QoS=1` 代表订阅者收到消息后必须返回 `puback` 给发布者，否则会一直推送消息。

本项目仅实现 `QoS=1` 即可满足要求。

- **保活时间：**当设备发起连接时会向物联网平台发送 `CONNECT` 消息，物联网平台使用 `CONNACK` 消息进行响应并保持连接，设备在保活时间间隔内至少需要发送一次报文，否则物联网平台会断开与设备的连接。
- **清除会话：**设备和物联网的会话状态是临时或持久。

三、MQTT 代理/服务器

1. 阿里云物联网平台服务器搭建

2. 创建产品与设备

登录物联网平台控制台。在控制台左上方，选择物联网平台所在地域为“华东 2（上海）”。在实例概览页签，选择我们已经创建过的公共实例。

> 图片略

随后选择设备管理 -> 产品，先创建一个检测系统的产品。在新建产品页面，配置参数后，完成对“智能温湿度与气压检测系统”的创建，具体参数如下图所示。

> 图片略

完成产品创建之后，我们可以进行设备的添加，选择添加设备并输入设备名称、备注名称。至此平台的搭建和设备云端部署就基本完成，信息如下图所示：

> 图片略

创建完成 THP-DataSystems 设备后，在设备详情页面，我们可以获取设备证书，用于后续的发布端和订阅端的开发和连接。其中设备证书包含 ProductKey、DeviceName 和 DeviceSecret，是设备与物联网平台进行通信的重要身份认证。

3. 为产品定义物模型

阿里云的物联网平台支持为产品定义物模型，即将实际产品抽象成由属性、服务、事件所组成的数据模型，便于物联网平台管理和数据交互，产品下的设备将自动继承物模型内容。因此我们为检测系统创建了如下所示的物模型：

功能类型	功能名称（全部）	标识符	数据类型	数据定义
属性	检测时间 自定义	DetectTime	date (时间型)	-
属性	当前气压 自定义	CurrentPressure	int32 (整数型)	取值范围：300 ~ 1100
属性	当前湿度 自定义	CurrentHumidity	double (双精度浮点型)	取值范围：0 ~ 100
属性	当前温度 自定义	CurrentTemperat...	float (单精度浮点型)	取值范围：-40 ~ 80

确认无误之后，即可完成系统数据的物模型发布，设备将继承最新发布的物模型内容，接下来就可以将阿里云物联网平台与发布端连接、通信。

四、发布端

1. 环境配置

发布端我们使用了 python 3.9 进行开发，环境主要安装包及其版本如下所示：

Package	Version
aliyun-iot-linkkit	1.2.11
paho-mqtt	1.6.1
Flask	2.2.5
Jinja2	3.1.2
pip	23.3.1
stomp.py	8.1.0
schedule	1.2.1

其中较为重要的是安装 Python Link SDK：执行 `pip install paho-mqtt` 命令安装 paho-mqtt，随后执行 `pip install aliyun-iot-linkkit` 安装 Link SDK 最新版本。

2. 建立设备与平台的连接

Python Link SDK 仅支持设备密钥方式进行设备身份认证，本项目采用一机一密的方式进行认证，该方式不涉及注册，而是直接令每台设备烧录自己的设备证书（包括 ProductKey、DeviceName 和 DeviceSecret）。

```
class MQTTClient:
    def __init__(self, product_key, device_name, device_secret):
        self.lk = linkkit.LinkKit(
            host_name="cn-shanghai",
            product_key=product_key,
            device_name=device_name,
            device_secret=device_secret
        )
        self.product_key = product_key
        self.device_name = device_name
        self.device_secret = device_secret
        self.mqtt_topic_post = f'/sys/{product_key}/{device_name}/thing/event/property/post'
```

其中，初始化 linkkit 的相关参数解释如下：

host_name	cn-shanghai	设备接入的地域ID。
product_key	******	设备认证信息。
device_name	THP-DataSystems	
device_secret	******	
port	1883	端口号。
keep_alive	180	保活时间，单位秒，取值范围为60~180。当设备的 活时间内，物联网平台将拒绝设备的连接

点击图片可查看完整电子表格

3. 配置发布端物模型

本项目使用物模型 Topic 实现对设备实际功能的抽象，物模型从属性、服务和事件三个维度，分别描述了该实体是什么、能做什么、能提供什么信息。例如数据中的温度、湿度、气压和检测时间的数值是属性。

初始化

代码中的 `MQTTClient.py` 文件, 在 `MQTTClient` 类的 `init` 函数中完成对象的初始化。

```
class MQTTClient:
    def __init__(self, product_key, device_name, device_secret):
        self.lk = linkkit.LinkKit(
            host_name="cn-shanghai",
            product_key=product_key,
            device_name=device_name,
            device_secret=device_secret
        )
```

配置物模型文件

随后我们需要从云端控制台下载物模型文件并集成到应用工程中，且在连接云端之前调用，这样对应的 Topic 才能正确接收和发送消息：

```
# 从云端控制台下载物模型文件，该文件需要集成到应用工程中
self.lk.thing_setup("tsl.json")
```

随后属性上报、事件上报、设置属性、服务响应等接口的应用，在设备完成认证与连接之后可以顺利进行。

4. 属性上报

通过 python 完成设备发布端的开发，设置设备上报的原始数据格式。设备接入物联网平台后，便可与物联网平台进行通信。

设备的数据上报方式有两种：ICA 标准数据格式（Alink JSON）和透传/自定义。本项目使用的是 ICA 标准数据格式（Alink JSON），即设备按照物联网平台定义的标准数据格式生成数据，然后上报数据。如下所示，我们通过 `mqtt_client.lk.publish_topic` 上报属性，传入参数为属性名称及值，请求 Topic 设置为：`/sys/${productKey}/${deviceName}/thing/event/property/post`

```python
prop_data = {
    "DetectTime": str(int(row[0])),  # 假设 CSV
    "CurrentTemperature": float(row[1]),
    "CurrentHumidity": float(row[2]),
    "CurrentPressure": int(row[3])
}
# 构造上报数据结构
payload = {
    "id": "123",
    "version": "1.0",
    "params": prop_data,
    "method": "thing.event.property.post"
}
# 上报属性
rc, request_id = mqtt_client.lk.publish_topic(
    topic,
    json.dumps(payload)
)
```

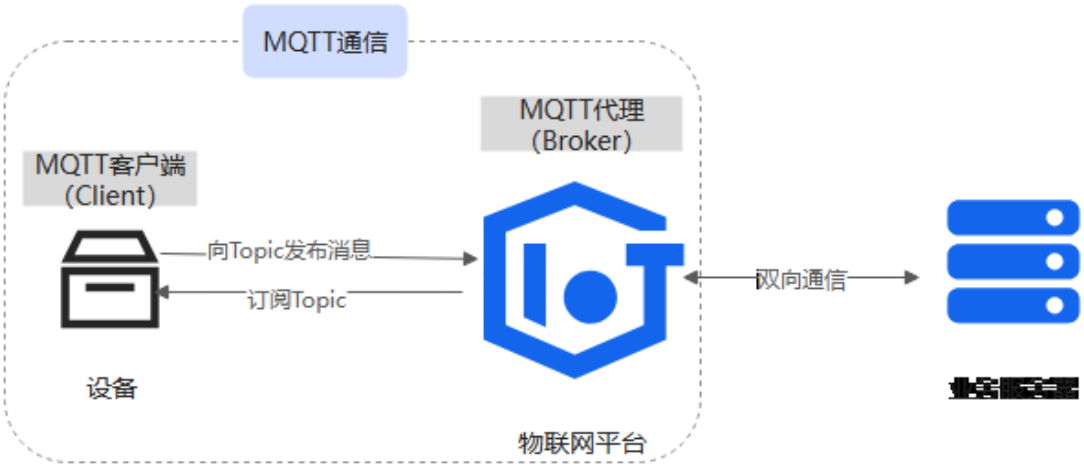
Alink JSON 的相关参数设置，我们做如下解读：

id	String	消息ID号。String类型的数字，取值范围0~4294967295，且每个 设备中具有唯一性。
version	String	协议版本号，目前协议版本号唯一取值为1.0。
sys	Object	扩展功能的参数，如果未设置扩展功能，则无此参数，保持默认配
ack	Integer	sys下的扩展功能字段，表示是否返回响应数据。1：云端返回响应 端不返回响应数据。如果未配置该功能，则云端默认返回响应数据
method	String	请求方法。例如本项目的：thing.event.property.post。
params	Object	请求参数。如以上示例中，设备上报了的四个属性的信息。具体属 含属性上报时间（time）和上报的属性值（value）。若如本例仅 值，无需上传字段time和value。
time	Long	属性上报时间戳，类型为UTC毫秒级时间。若上传time，物联网平 存上传的时间作为属性上报时间。若不上传time，物联网平台的云

点击图片可查看完整电子表格

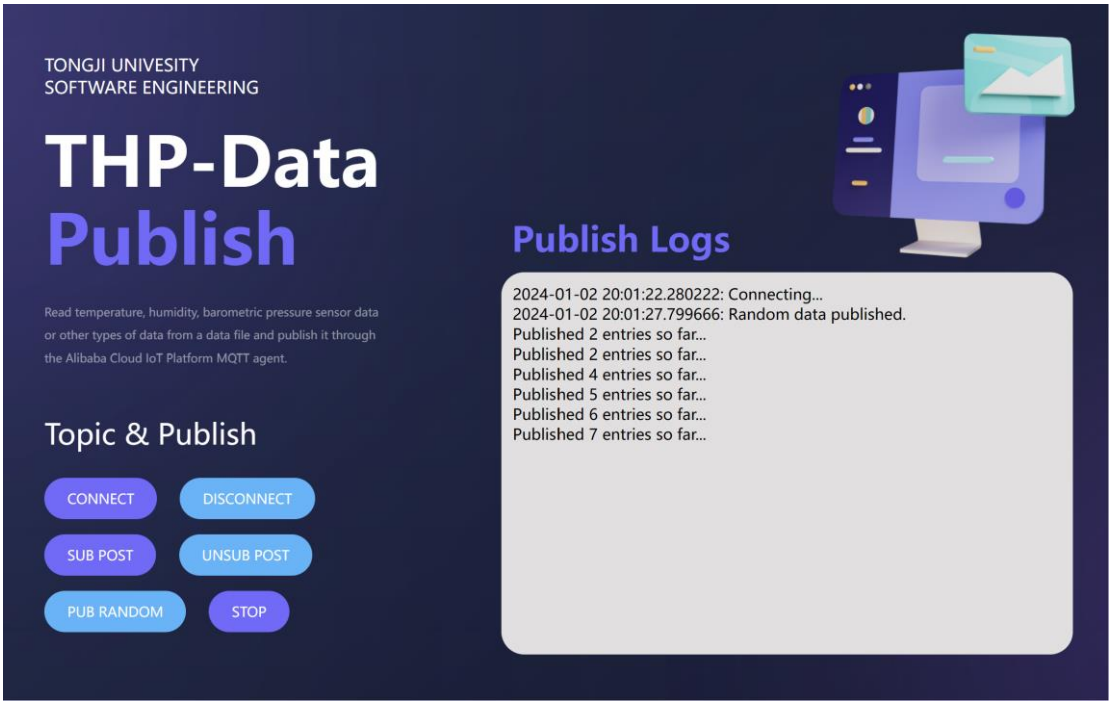
我们通过 topic 发布数据后，服务端将对上报的属性做出处理发出响应，通过 SDK 通知用户，返回值 `rc` 为 0 时，表明请求写入发送缓冲区成功，`rc` 为其它值时，表示写入发送缓冲失败。

在发布/订阅消息的 MQTT 协议中，Topic 用于定义消息传输的通道和路径，设备通过 Topic 将消息发布到物联网平台，物联网平台将消息发送给订阅 Topic 的设备。



项目中还设置了相关函数对输入的数据文件进行了预处理，如温湿度气压按照时间列的一致合并、数据按时间顺序排序等、数据空缺项删除等，保证了传输数据的准确与可靠。

5. 界面设计



发布端界面主要有左方的发布数据区和日志消息显示区，运行发布端进入该界面之后，可以点击 connect 连接到阿里云服务器，点击“Pub Random”可以发布一条数值随机的包含温湿度气压全部属性的 POST 消息，点击“Pub ALL”会发布文件夹内 pubulish_data.csv 内的所有数据。左方按键操作状态都会在右侧日志栏给出日志反馈，以便使用者确定消息发布的成功与否。

五、订阅端

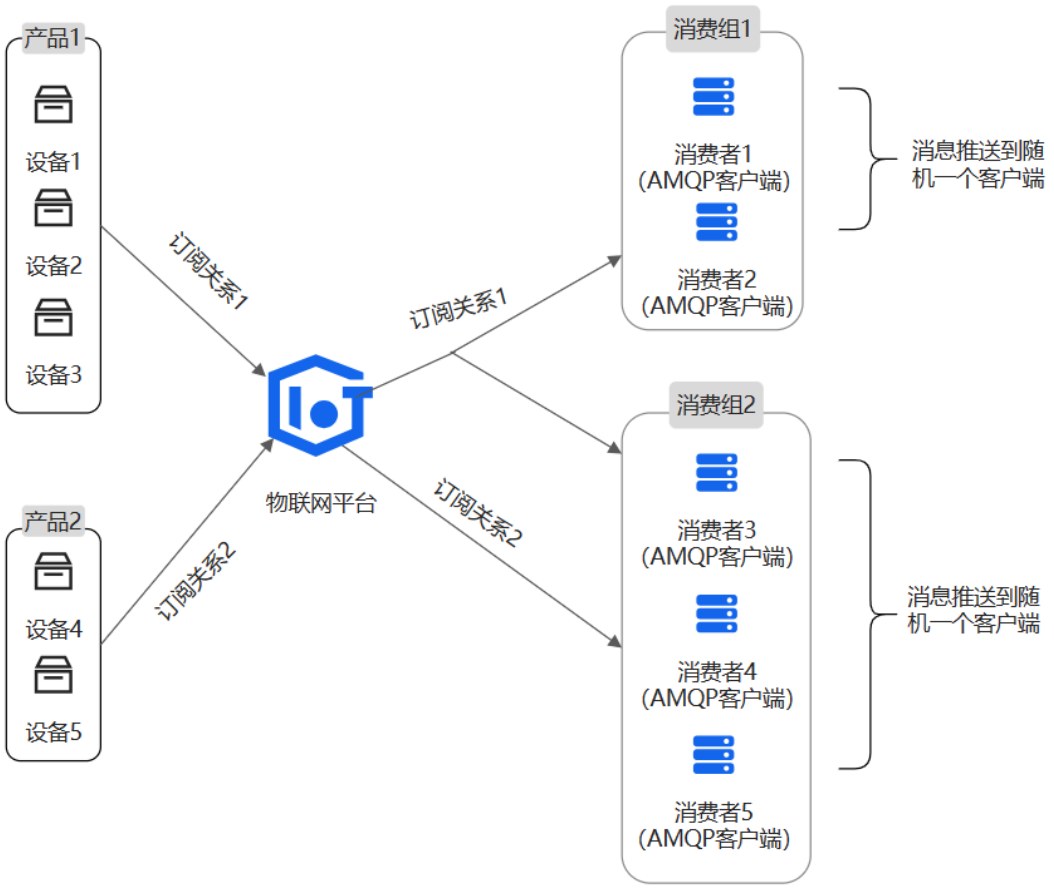
1. 创建消费组并进行服务端订阅

对于订阅端，我们打算创建消费组，然后通过 AMQP 服务端订阅的方式把发布端的数据转发到消费组中。

服务端订阅指的就是 服务端可以接收产品下已订阅的全部设备数据。

它的优点是很简单，缺点是缺乏对数据的过滤和转换能力。

因此，我们订阅端的其中一个任务就是要对接收到的这些数据进行过滤和分类。



2. AMQP 与 MQTT 的区别和联系

接下来介绍一下 AMQP，它的优点是，如果消息发送失败的话会进入堆积队列，等订阅端恢复正常以后，可以再次接收消息，这样能避免数据缺失。这也是一种可靠的信息传输方式，AMQP 客户端与物联网平台需要经过三次握手从而建立 TCP 连接。

AMQP服务端订阅优势：

- 支持多消费组。同一个账号，可以在开发环境下使消费组A订阅产品A，同时在正式环境下使消费组B订阅产品B。

  > **说明** 如果多个不同消费组同时订阅产品B，则不同消费组可同时收到来自设备B的相同信息。

- 方便排查问题。支持查看客户端状态、查看堆积和消费速率。
- 线性扩展。在消费者能力足够，即客户端机器足够的情况下，可轻松线性扩展推送能力。
- 实时消息优先推送，消息堆积不会影响服务。
  设备实时消息直接推送，推送限流或失败时进入堆积队列，堆积态消息采用降级模式，不会影响实时推送能力。
  即使消费者的客户端宕机，或因消费能力不足堆积了消息，消费端恢复后，设备生成的消息也可以和堆积消息并行发送，使设备优先恢复实时推送消息状态。

它和 MQTT 有很多相似性，比如它需在 Open 帧中携带心跳时间，如果超过心跳时间，`Connection` 上没有任何帧通信，物联网平台将关闭连接。MQTT 也有这种心跳间隔时间。

MQTT 协议可以进行异步的发布和订阅，就是说发布端和订阅端不用直接连接，我们的 AMQP 客户端也是这样的，它本质上是与物联网平台建立了联系，而物联网平台与发布端通过 MQTT 协议传递消息。

在订阅端使用 AMQP 一方面是因为这是阿里云平台推荐的解决办法，另一方面也是因为发布端已经使用了 MQTT，所以想使用新的方法来对比一下。

从可靠性层面比较的话，MQTT 协议使用 QoS（Quality of Service）级别来确保消息的可靠传输。MQTT 定义了三个 QoS 级别：0、1 和 2。

- QoS 0：最多一次交付，消息可能会丢失或重复。
- QoS 1：至少一次交付，确保消息会被送达，但可能会重复。
- QoS 2：恰好一次交付，确保消息会被送达一次，不会重复。

而 AMQP 提供可靠的服务，它与物联网平台经过三次握手建立 TCP 连接，然后进行 TLS 握手校验。因此大多数时候 AMQP 会有更多的开销。

MQTT 提供的订阅主题功能也是我们所要注意的，它采用的是发布/订阅的消息传输模型，客户端可以选择订阅感兴趣的主题，只接收自己关心的消息，减少了不必要的通信开销。之后我们会试图用 AMQP 建立发布/订阅模型。

接下来我们可以看一下两种通信协议的具体实现。

我们同样都需要通过密钥和产品组 ID 等来建立连接。

这里可以对比一下两种接收消息的方式：

开启接收线程。

```
res = pthread_create(&g_mqtt_recv_thread, NULL, demo_mqtt_recv_thread, mqtt_handle);
if (res < 0) {
    printf("pthread_create demo_mqtt_recv_thread failed: %d\n", res);
    return -1;
}
```

MQTT 创建了一个新线程来接收服务器发送的消息，并将其传递给其他处理逻辑进行处理。相对来说，它比 AMQP 要简单一些。

```
class MyListener(stomp.ConnectionListener):
    def __init__(self, conn):
        self.conn = conn

    def on_error(self, frame):
        print('received an error "%s"' % frame.body)

    def on_message(self, frame):
        print('received a message "%s"' % frame.body)

    def on_heartbeat_timeout(self):
        print('on_heartbeat_timeout')

    def on_connected(self, headers):
        print("successfully connected")
        conn.subscribe(destination='/topic/#', id=1, ack='auto')
        print("successfully subscribe")

    def on_disconnected(self):
        print('disconnected')
        connect_and_subscribe(self.conn)
```

AMQP 提供了监听器服务，因此可以更方便地理解和进行开发，更适合用于大型的复杂系统

3. 基于 AMQP 进行具体功能实现

订阅端需要提供实时通讯服务，MQTT 和 AMQP 都可以用于实现实时通讯，我们可以让发布端发布数据，然后订阅端就能实时接收到并且展示出来。

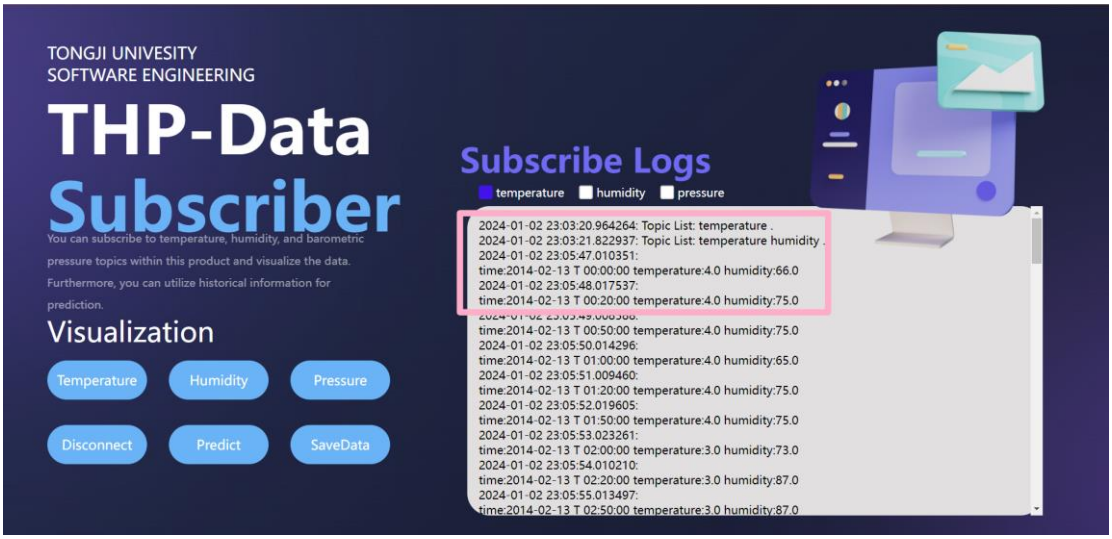
发布端的每个产品可以有多个功能，因此当发布端发布消息的时候，消息里面可能包含多种数据。例如，假设发布端有一个集成式传感器产品，那么它可能包含温度湿度气压等传感功能，如果它采集到数据，就会进行发布，因此它发布的数据可能包括这三种数据。

当订阅端订阅了某个产品以后，它可以获得这个产品发布的每种种类的数据。而订阅端可能只关注其中的一两项指标，所以它可以选择只获取其中的温度数据或者只获取温度和气压数据。

在使用 MQTT 协议的时候可以直接通过 aiot_mqtt_sub 来进行订阅，但因为我们建立了消费组，所以它能接到产品里所有设备的消息，不能取消订阅某种类型的数据，只能在服务端接收到以后再对消息进行处理，比如选择只展示或者只保存某种类型的数据。

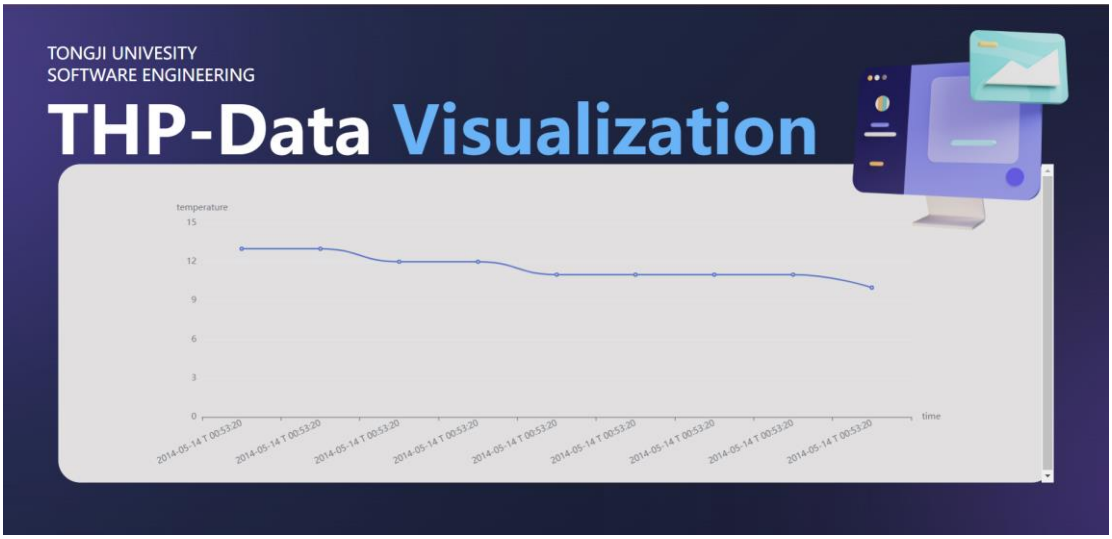
为了方便用户了解自己的操作是否成功，每点击一个按钮，在右侧就会输出相应的日志，显示做了什么操作、状态是成功还是失败。

当发布端发布消息以后，订阅端如果有订阅相应数据，那么只要有接收到就会输出出来。

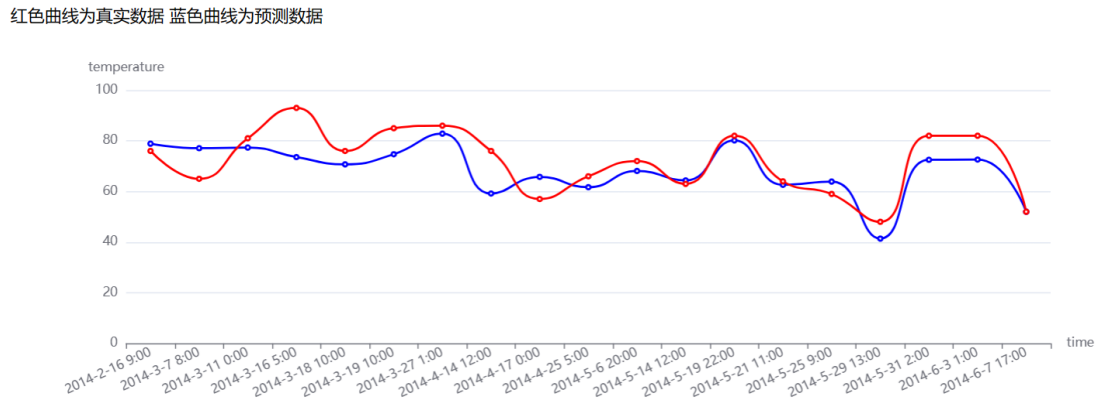


用户可能希望将接收到的数据进行保存，以便后续查看。我们提供了按钮来让用户把数据保存到一个固定的 csv 文件。后面如果接收到了新的数据，数据会添加到原有数据之后，原来的数据不会被覆盖。

用户可以对历史数据进行可视化。例如，点击“Temperature”按钮后，软件将读取 csv 文件，获得所有数据的日期和温度值，然后把它们按照日期先后顺序排好次序，再用平滑曲线的方式进行展示。通过这种方式，即使发布端发布数据的时候是乱序的，订阅端也能自动整理好，并且能提供给用户温度随时间的变化曲线。



我们也提供了数据预测功能，我们事先对项目里的温度、气压、湿度，划分出训练集和测试集，然后分别根据训练集建立了预测模型，再放到测试集上面检查预测值和真实值的区别。这里用红色的线展示了真实值，用蓝色的线展示了预测值，可以看出我们的模型质量是比较好的，然后我们也用 MSE 等来具体地量化了模型的准确度。



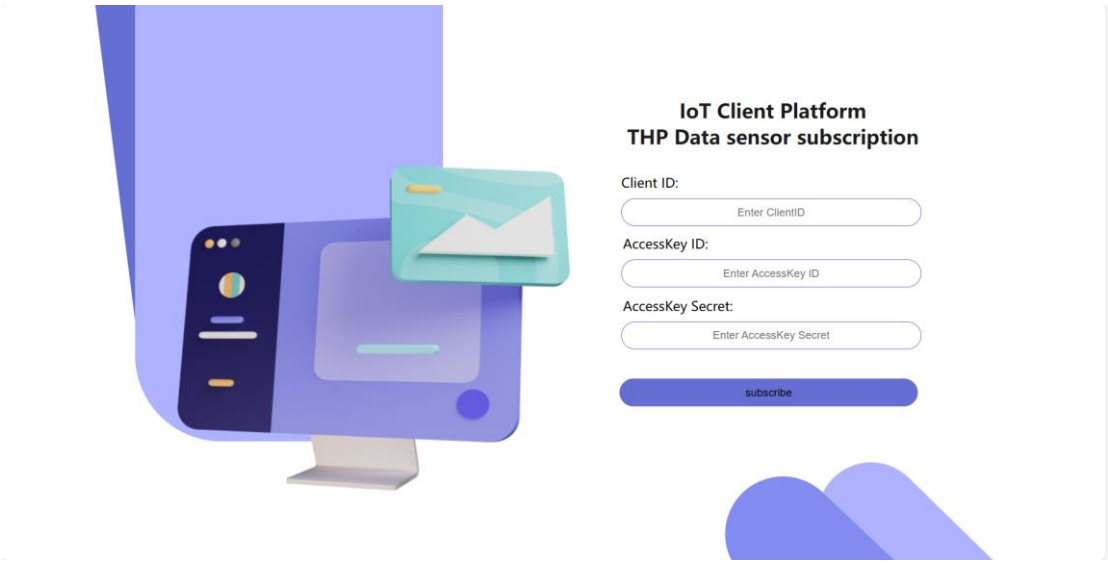
AMQP 和 MQTT 都可以有多个订阅端，我们进行了测试，发现当多个设备分别登录，并订阅同一个服务器的时候，如果服务器发布消息，两个订阅端设备能接收到相同的响应。从而验证了 MQTT 的服务端确实能将消息正确地传递给所有的订阅端。

为了确保安全性，每个订阅端用户只能在一台设备上进行登录。如果有人尝试在另一台设备上使用已经登录的账号进行连接，系统将拒绝建立连接。我们也可以在云服务器平台上查看当前有哪些订阅端用户登录。

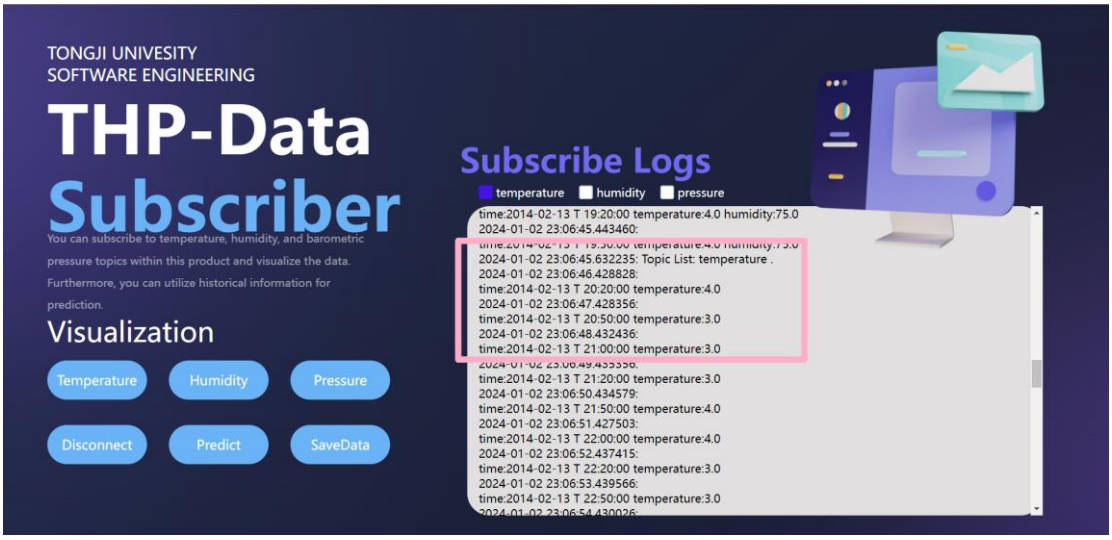
我们提供了取消连接的功能，比如说这个时候用户想更换订阅端设备，那么它可以点击 Disconnect 按钮，就会退出当前 client 的登录，回到登录页面。这个时候就可以在另一个设备上使用账号密码登录。

界面设计

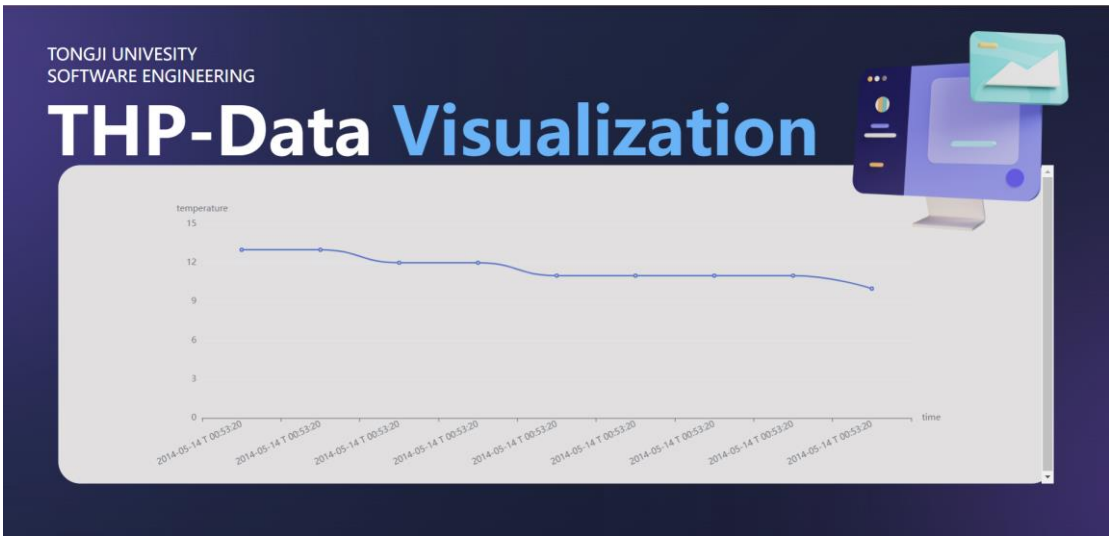
登录界面:



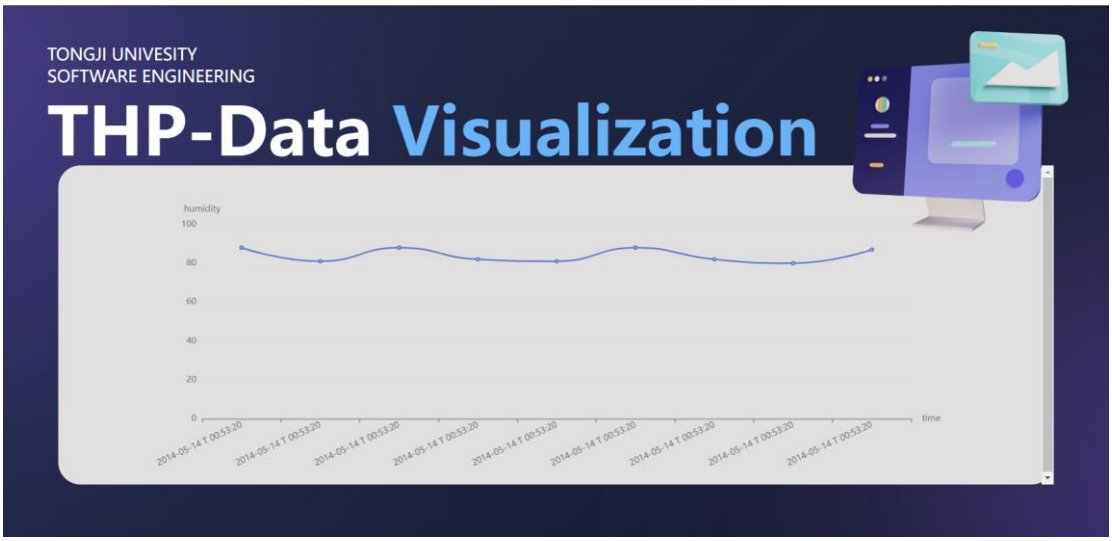
可以选择只接收 temperature 类型的数据，日志窗口会输出数据信息和相应的时间戳



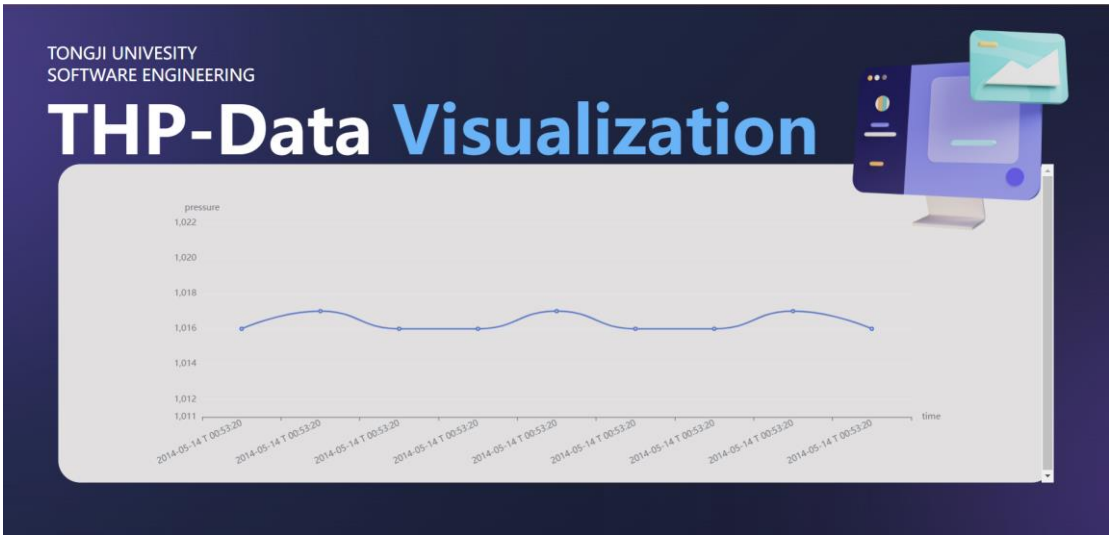
温度数据可视化



湿度数据可视化



气压数据可视化



六、数据处理端

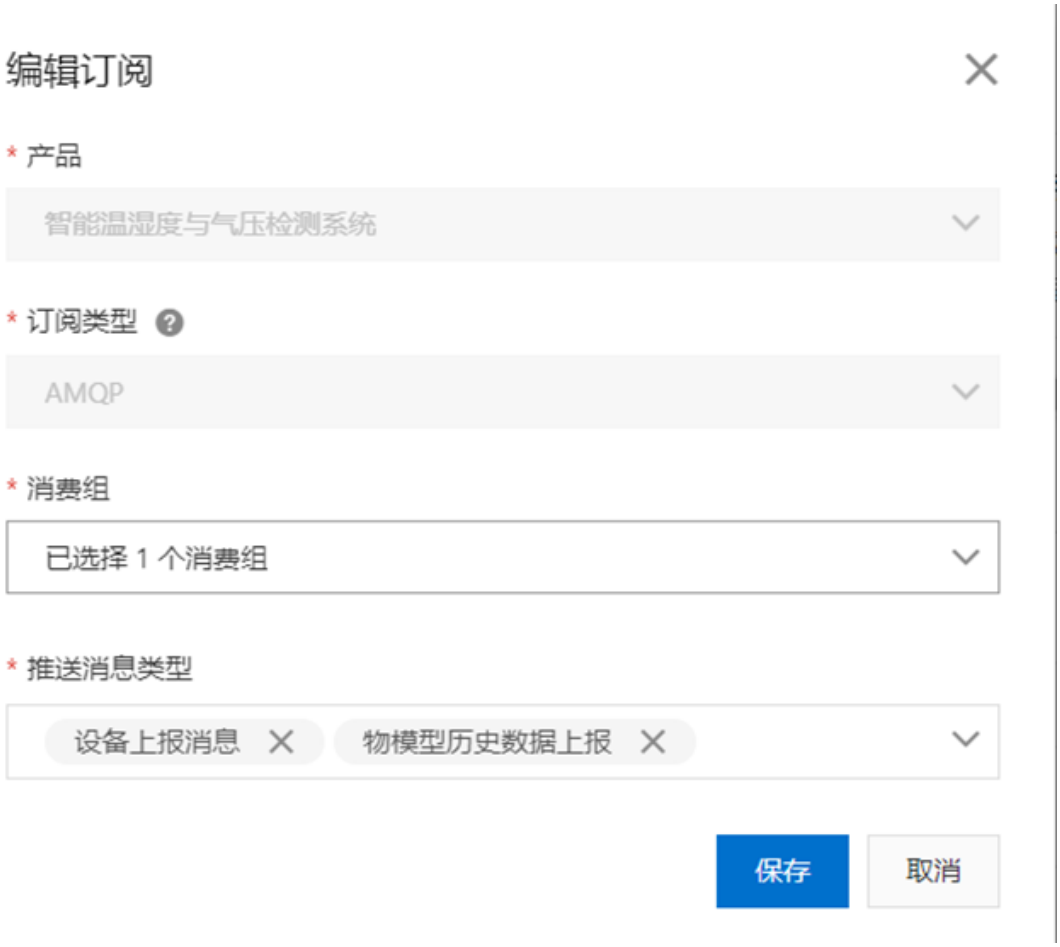
1. 整体介绍

数据处理端通过订阅方式接收数据，进行本地存储，并进行分析处理。

- 数据订阅：使用阿里云物联网平台的服务端订阅功能创建消费组，进行设备的消息转发
- 本地存储：接收的消息通过 python 代码进行处理，存储到本地 csv 文件中
- 分析处理：读入接收到的数据，使用 DNN 神经网络进行训练，实现基于日期时间对温度、湿度和气压分别进行预测。

2. 数据订阅模块

通过管理阿里云平台的消费组，进行不同 Topic 消息的订阅。



对接收消息进行处理，得到所含数据，主要代码如下：

```
Python
def transform_data(frame):
    str_result = frame.body
    print(type(frame.body))#str 类型

print(str_result)#{"deviceType":"CustomCategory","iotId":"6kKJ5Dcs
ITYMMbJdQvYdk0kqp0","requestId":"123","checkFailedData":{},"produc
tKey":"k0kqpFwvVRy","gmtCreate":1703778965509,"deviceName":"THP-
DataSystems","items":{"CurrentHumidity":{"value":83,"time":1703778
965505},"CurrentTemperature":{"
    result= json.loads(str_result)
    prop_data= {}
    prop_data["time"] =
timestamp_to_time(int(result['items']['DetectTime']['value']))
    prop_data["temperature"] =
float(result['items']['CurrentTemperature'].get("value", None))
    prop_data["humidity"] =
float( result['items']['CurrentHumidity'].get("value", None))
    prop_data["pressure"] =
```

```
int( result['items']['CurrentPressure'].get("value", None))
    ans_data={}
    print(str(len(gv.global_var.topic_list)))
    for topic in gv.global_var.topic_list:
        if prop_data[topic] is not None:
            ans_data[topic] = prop_data[topic]
    if (ans_data != {}):
        ans_data['time'] = prop_data['time']
        ans_data["printed"] = False
        gv.global_var.receive_data.append(ans_data)
        print(format_topicData(ans_data))
```

3. 数据预测模块

使用五层 DNN 神经网络进行数据预测，网络结构如下：

```
Training instances   6561, Training features   3
Validation instances 821, Validation features 3
Testing instances    821, Testing features    3
DNN(
  (layer1): Linear(in_features=3, out_features=64, bias=True)
  (layer2): Linear(in_features=64, out_features=128, bias=True)
  (layer3): Linear(in_features=128, out_features=256, bias=True)
  (layer4): Linear(in_features=256, out_features=128, bias=True)
  (layer5): Linear(in_features=128, out_features=1, bias=True)
)
```

神经网络输入：月份、日期、小时 输出：温度（湿度、气压）

训练神经网络时，首先进行数据处理：

1. 对空值数据进行清除
2. 打乱数据集
3. 将时间转成月份、日期、小时三个参数传入网络

温度、湿度、气压三个预测网络训练的损失值：

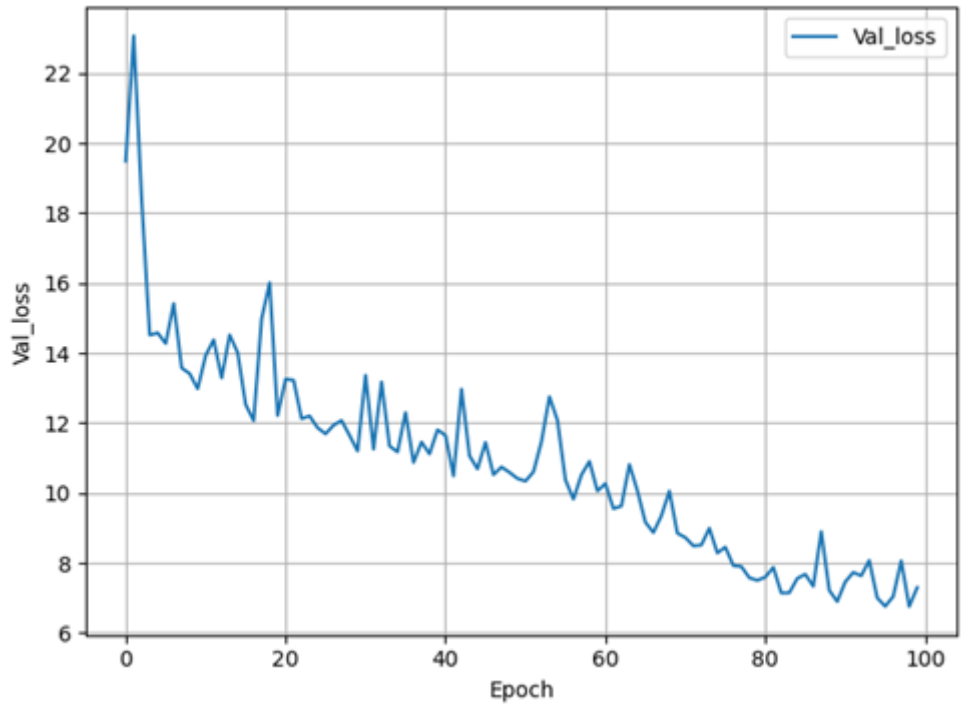


Figure 1 温度训练损失图

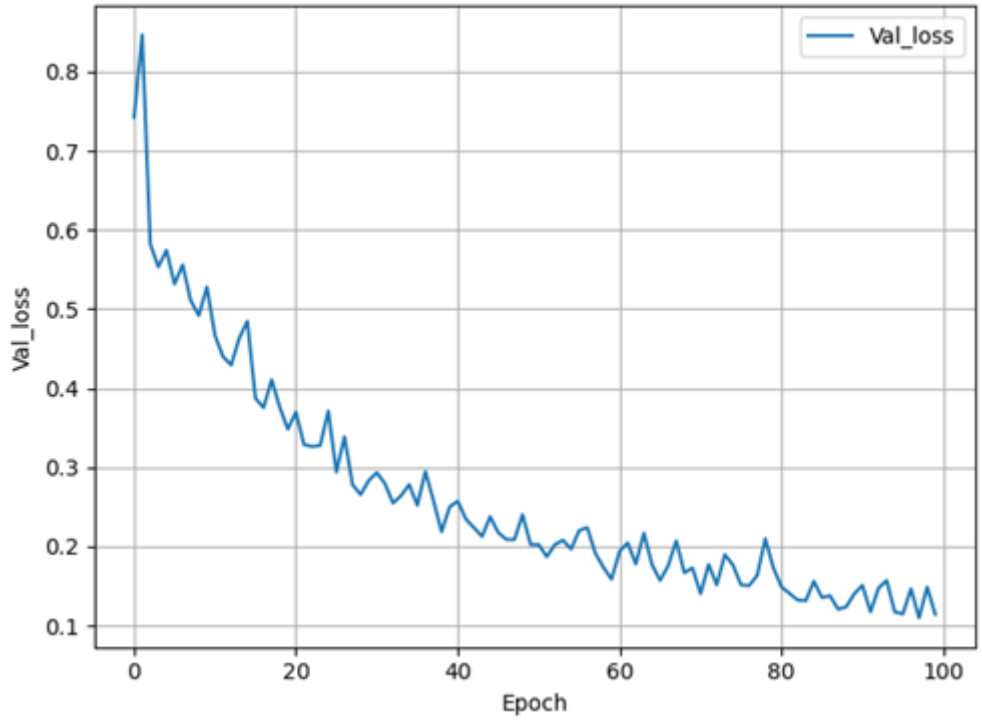


Figure 2 湿度训练损失图

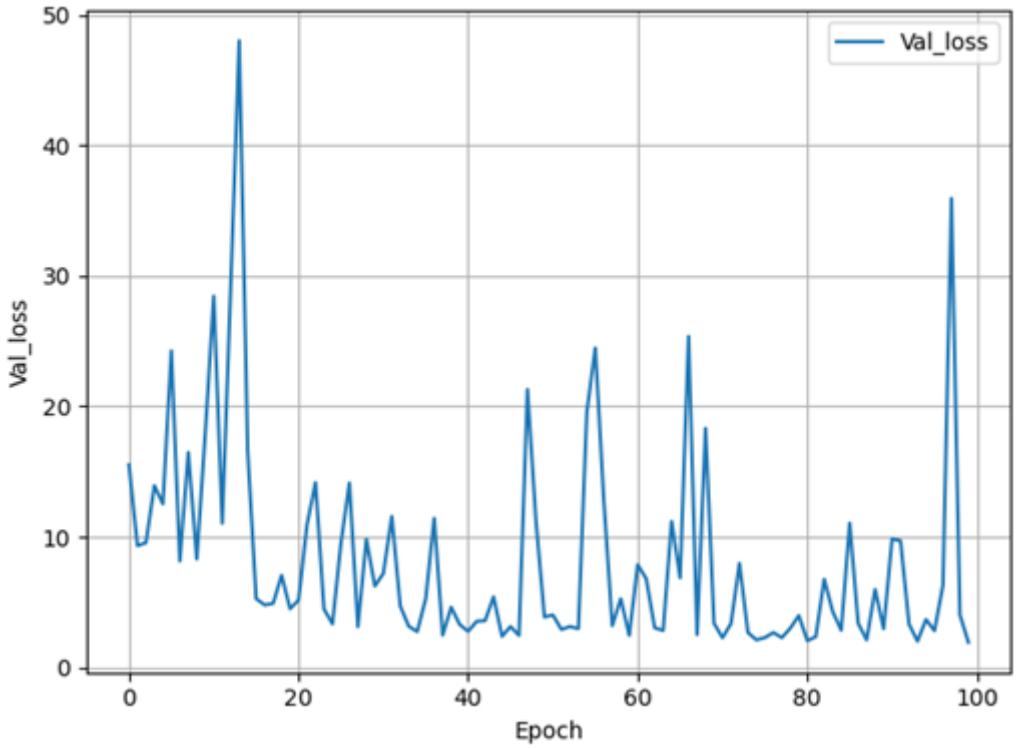


Figure 3 气压训练损失图

神经网络使用 MSE 进行评估:

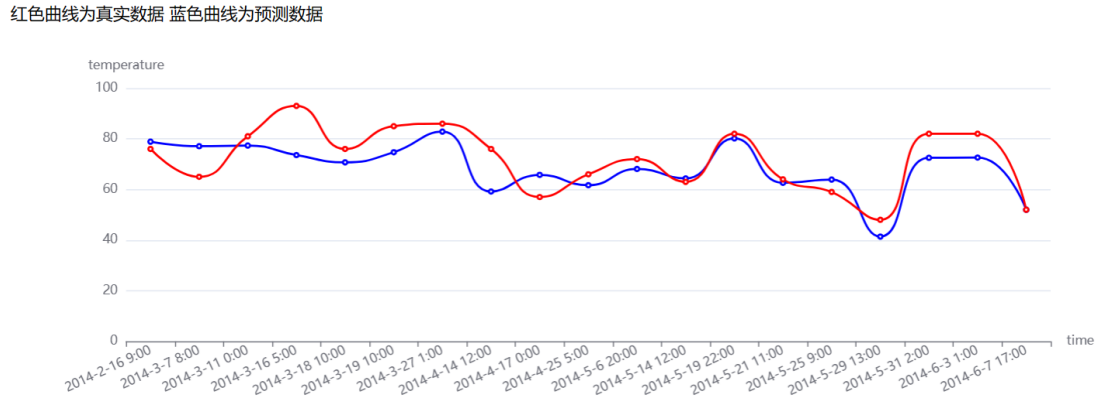
温度 MSE：7.8835

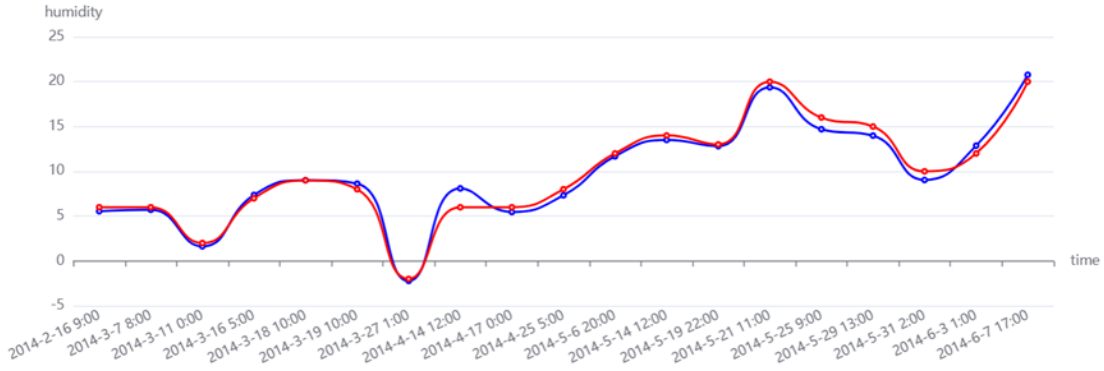
湿度 MSE：0.1051

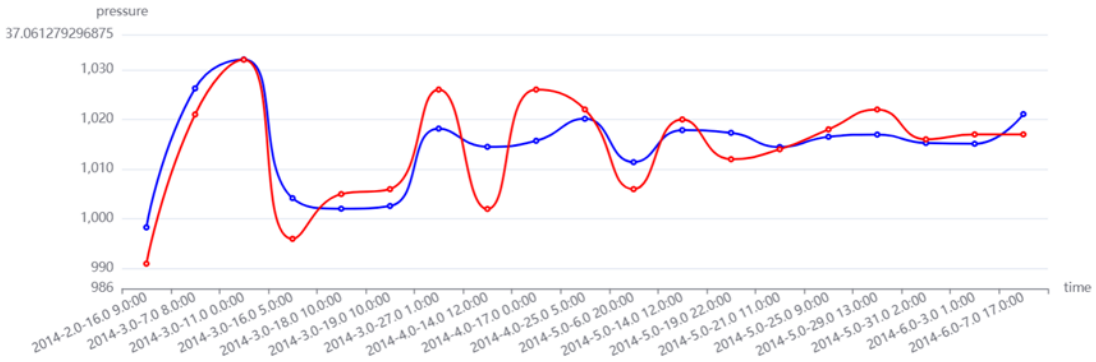
气压 MSE：1.9156

可以看出，使用神经网络进行数据预测的方式比较可行，并且在湿度参数上具有良好的预测能力。

对预测数据进行图表可视化:




humidity
25
20
15
10
5
0
-5
time
2014-2-16 9:00
2014-3-7 8:00
2014-3-11 0:00
2014-3-16 5:00
2014-3-18 10:00
2014-3-19 10:00
2014-3-27 1:00
2014-4-14 12:00
2014-4-17 0:00
2014-4-25 5:00
2014-5-6 20:00
2014-5-14 12:00
2014-5-19 22:00
2014-5-21 11:00
2014-5-25 9:00
2014-5-29 13:00
2014-5-31 2:00
2014-6-3 1:00
2014-6-7 17:00


pressure
37.061279296875
1,030
1,020
1,010
1,000
990
986
time
2014-2-0-16.0 9:0:00
2014-3-0-7.0 8:0:00
2014-3-0-11.0 0:0:00
2014-3-0-16.0 5:0:00
2014-3-0-18.0 10:0:00
2014-3-0-19.0 10:0:00
2014-3-0-27.0 1:0:00
2014-4-0-14.0 12:0:00
2014-4-0-17.0 0:0:00
2014-4-0-25.0 5:0:00
2014-5-0-6.0 20:0:00
2014-5-0-14.0 12:0:00
2014-5-0-19.0 22:0:00
2014-5-0-21.0 11:0:00
2014-5-0-25.0 9:0:00
2014-5-0-29.0 13:0:00
2014-5-0-31.0 2:0:00
2014-6-0-3.0 1:0:00
2014-6-0-7.0 17:0:00