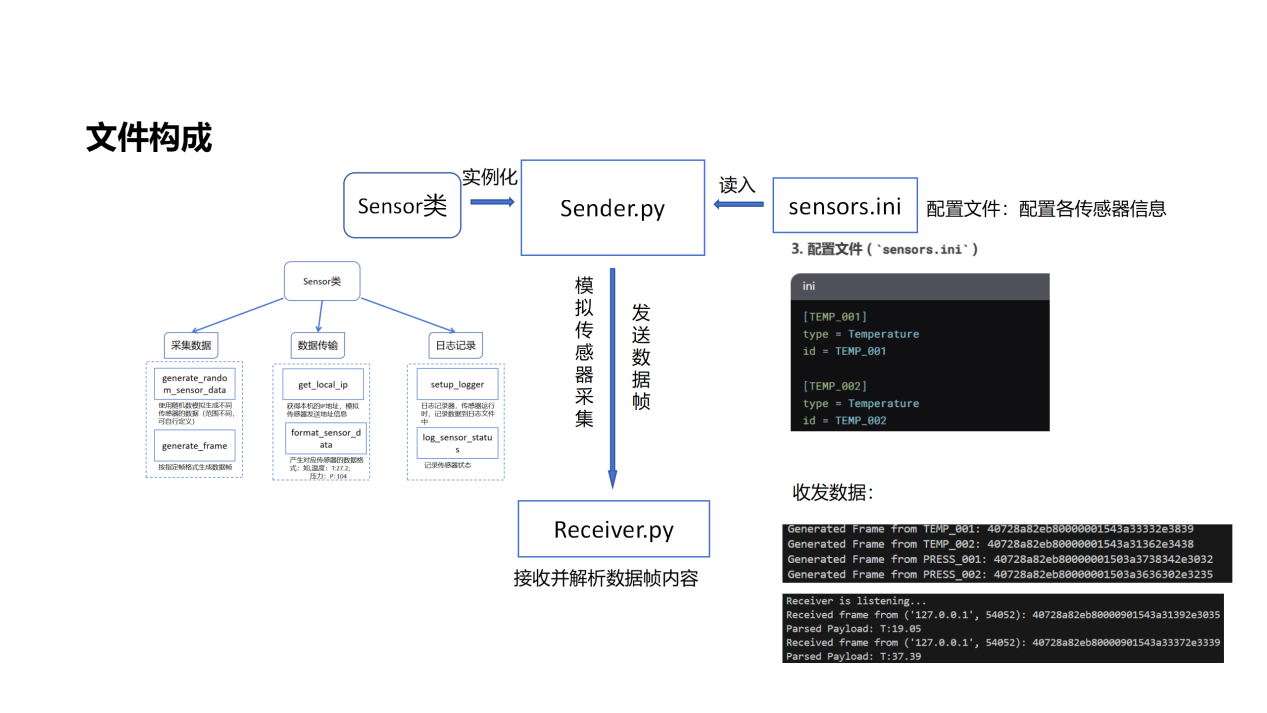
# 1、传感器模拟使用流程

模拟传感器整体流程如下，共涉及三个文件：Sender.py（产生数据）、Receiver.py（数据接收）、sensors.ini（模拟配置）。其中Sender.py模拟部署到建筑物上物联网传感器的采集数据、发送过程；Receiver.py模拟终端接收数据的过程，并完成对数据的解析；sensors.ini用于我们配置各种传感器信息。



## 1.1修改配置文件

配置文件内容形式如下:

[TEMP\_001]

type = Temperature

id = TEMP\_001

[TEMP\_002]

type = Temperature

id = TEMP\_002

现有两个传感器，种类为Temperature，温度传感器。我们可以在中间加入

[PRESS\_001]

type = Pressure

id = PRESS\_001

[PRESS\_002]

type = Pressure

id = PRESS\_002

即完成两个压力传感器的添加。

## 1.2修改发送文件

发送文件主函数如下，读入sensors.ini文件完成传感器的初始化。使用socket库中函数，通过本地端口10000发送数据，现为每五秒发送一次数据。可按自身需求进行修改，完善。

def main():

config = configparser.ConfigParser()

config.read('sensors.ini')

sensors = []

for section in config.sections():

sensor\_type = config[section]['type']

sensor\_id = config[section]['id']

sensors.append(Sensor(sensor\_type, sensor\_id))

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_address = ('localhost', 10000)

while True:

for sensor in sensors:

frame = sensor.generate\_frame()

print(f"Generated Frame from {sensor.sensor\_id}: {frame.hex()}")

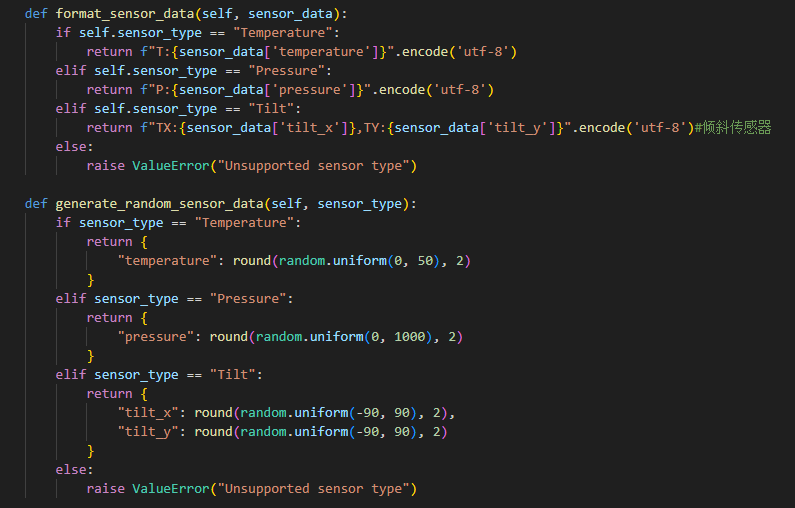
sock.sendto(frame, server\_address)

time.sleep(5)

发送文件中包含sensor类，功能较为基础，现暂未对其封装。如有需要，可进行如下修改：

1. 添加新类别的传感器

如需添加新类型传感器，例如添加震动传感器vibration，在format\_sensor\_data中添加else分支增加vibration相关参数描述。



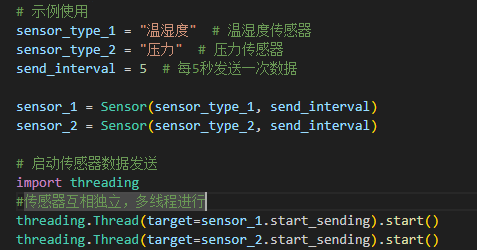
后再在generate\_random\_sensor\_data中添加vibration传感器的数值范围即可。

1. 修改传感器数据范围

同上图所描述，找到generate\_random\_sensor\_data中对应传感器数值范围修改为所需即可；如采用真实数据，结合数据库知识，定义新的传感器数值生成函数。

1. 并发运行传感器

学习使用python中threading库函数，定义多个线程，使得多个传感器同时运行。（不强制要求）



## 1.3修改接收文件

Receiver.py文件主函数如下，监听本地端口10000接受数据，可自行设计save\_to\_database函数，直接将数据储存到数据库中，自行修改。

def main():

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_address = ('localhost', 10000)

sock.bind(server\_address)

print("Receiver is listening...")

while True:

data, address = sock.recvfrom(4096)

print(f"Received frame from {address}: {data.hex()}")

parsed\_payload = parse\_frame\_payload(data)

print(f"Parsed Payload: {parsed\_payload}")

#save\_to\_database(parsed\_payload)

Receiver.py文件中另有解析函数，解析当前LoRaWAN协议帧，根据自身协议需求修改。

def parse\_frame\_payload(frame):

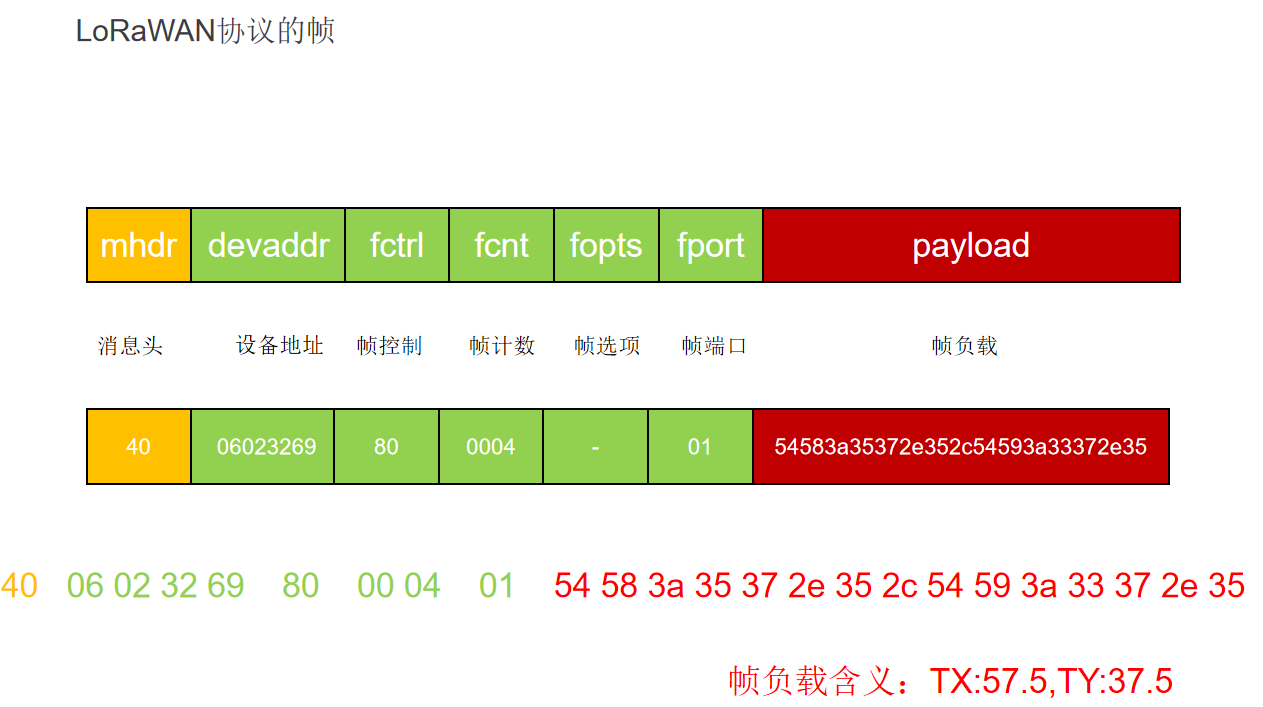
frm\_payload = frame[9:]

payload\_str = frm\_payload.decode('utf-8')

return payload\_str

# 2、Sensor类介绍

在本案例中，我们使用代码模拟物联网传感器进行数据发送，代码中使用的帧格式为LoRaWAN协议（如熟悉其他物联网协议，可修改）。我们在外部函数中，设计了解析帧格式的函数，可直接进行调用。



Sensor类主要实现了三大基本功能：采集数据、数据传输、日志记录。各功能函数如下图所示：

