聚焦异同：物联网与互联网有啥不一样？

目录

物联网与互联网连接方式差异

- 传统互联网体现了人–机交互，人负责输入和接收机器的输出，机器负责计算
- 物联网结构下，设备也可以产生数据、接收数据，即设备也可以完成输入输出

物联网与互联网连接方式差异

物联网设备包含传感器：

- 如温度传感器、位置传感器、激光雷达、相机等，它们可以接收来自外界的数据
- 另一类如马达、机械臂等可以接收用户的数据，完成特定的动作

物联网与传统互联网性能差异

- 传统互联网的服务器功能都要比个人终端性能强大，处理能力强劲
- 物联网终端设备性能有限，往往需要边缘设备配合完成计算任务

总结

1. 物联网的连接方式相比于传统互联网多了设备端，设备端可以产生和接收数据

2. 物联网的设备端性能差，因此需要将数据传送到边缘计算服务器或中心服务器集中处理

课后作业

请你说出至少 3 种接触过的物联网设备，并说明它们是能够产生数据还是能够接收数据执行任务。

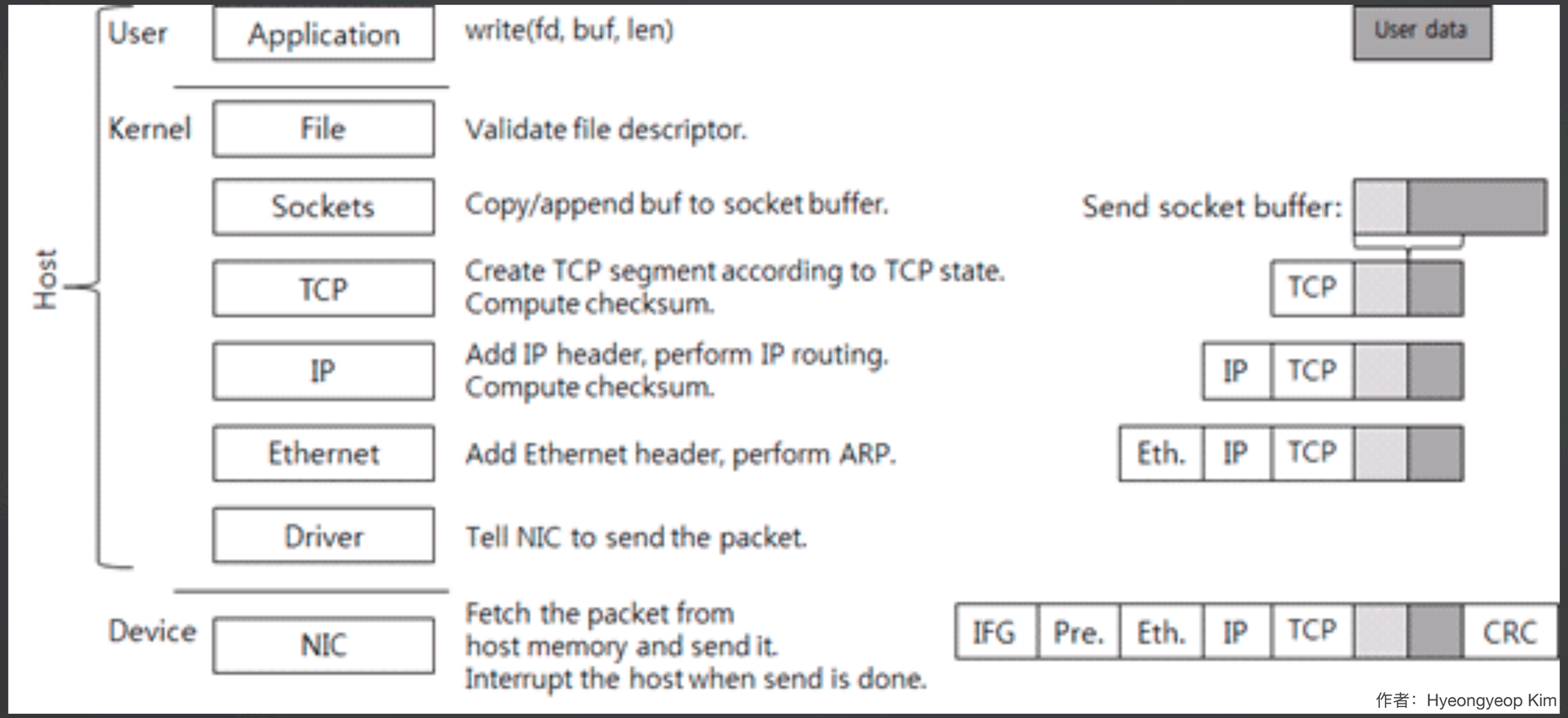
理论盘点：基础但不简单的 TCP 协议

目录

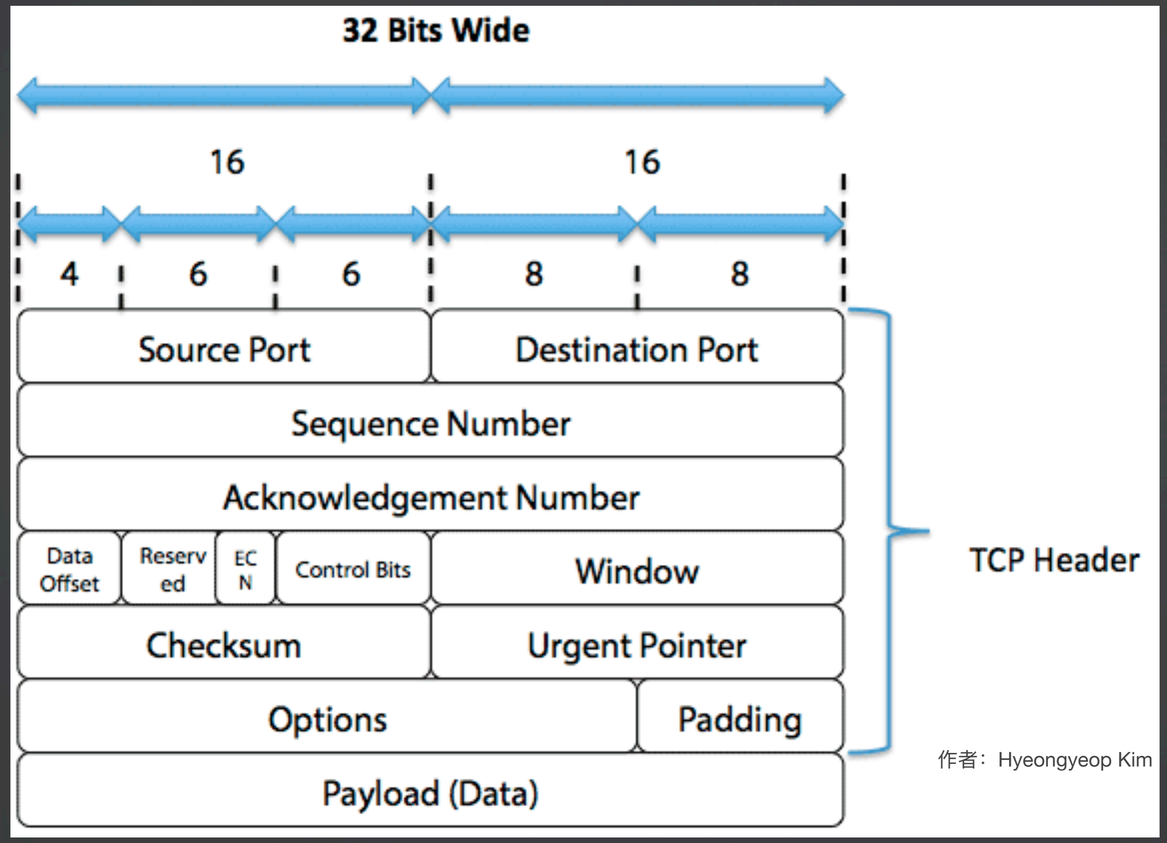
物联网通信为何常用 TCP 协议

- TCP 协议比 HTTP 协议更简洁，在轻量级的设备上占用更少的资源，消耗更低的带宽
- TCP 协议是可靠协议，能够确保设备执行了用户的指令
- 有时为了更加轻量的通信，采用 UDP 协议+自行编写校验来实现可靠通信

TCP协议概述


Host
User
Application
write(fd, buf, len)
User data
Kernel
File
Validate file descriptor.
Sockets
Copy/append buf to socket buffer.
Send socket buffer:
TCP
Create TCP segment according to TCP state.
Compute checksum.
IP
Add IP header, perform IP routing.
Compute checksum.
Ethernet
Add Ethernet header, perform ARP.
Eth.
Driver
Tell NIC to send the packet.
Device
NIC
Fetch the packet from
host memory and send it.
Interrupt the host when send is done.
IFG
Pre.
CRC
作者：Hyeongyeop Kim

TCP协议概述


32 Bits Wide
16
16
4
6
6
8
8
Source Port
Destination Port
Sequence Number
Acknowledgement Number
Data Offset
Reserv ed
EC N
Control Bits
Window
Checksum
Urgent Pointer
Options
Padding
Payload (Data)
TCP Header
作者：Hyeongyeop Kim

总结

1. TCP 协议是物联网通信的最常用协议

2. 掌握 TCP 协议的通信过程，有助于你编写更加高效的物联网设备控制程序

课后作业

请你参考 Python 官方文档关于 socket 库的示例，实现两个程序通过 TCP 协议实现简单通信。

文档参考地址如下：

https://docs.python.org/zh-cn/3.10/library/socket.html#example

理论盘点：物模型与模组

目录

怎样通过物模型定义智能设备

物模型：设备的数据模板

物模型是将设备抽象为数据的模板，和我们学习过的 Django Template 异曲同工

怎样通过物模型定义智能设备

物模型包含属性、事件和方法三个种类：

- 方法一般为设备执行的动作，因此也被称作动作
- 事件是指设备运行过程中产生的信息或故障

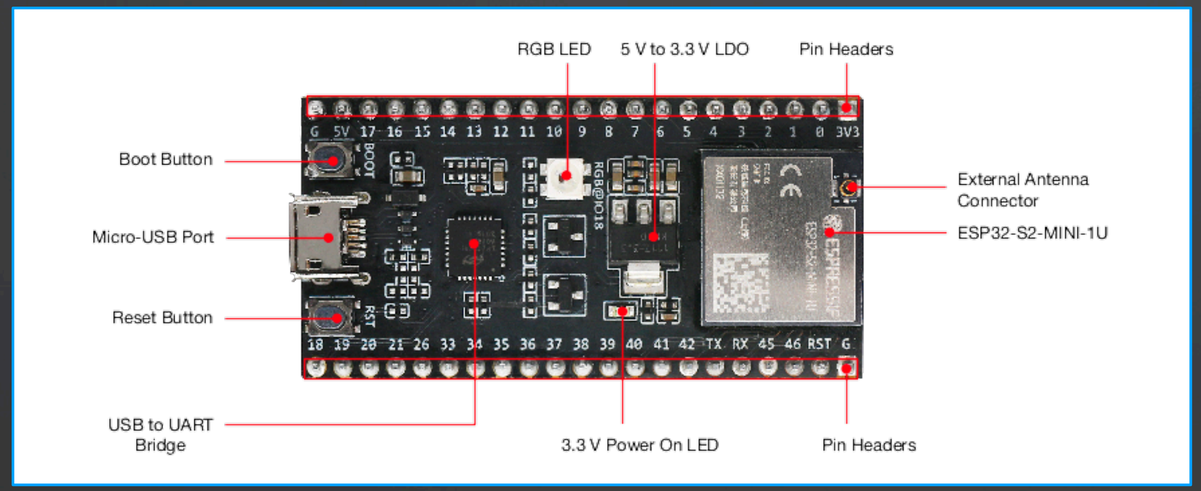
怎样通过物模型定义智能设备

为了体现物模型的扩展性，一般采用 json 格式进行物模型的定义

像搭积木一样掌握模组

物联网芯片负责设备通信和 IO 接口的管理

为了便于使用，通常将芯片与周边硬件组成模组



总结

1. 物模型用于定义设备的数据抽象
2. 模组用于组合芯片与周边硬件，确保物联网设备不必从零开发

课后作业

请你使用这节课我们学习的物模型，尝试为智能电风扇定义一个物模型。

如何为 ESP32 开发板安装 MicroPython?

目录

ESP32 开发板简介

为了便于验证，我们使用外设丰富的 ESP32-DevKitC 开发板实现模组的选型

ESP32 有 4MB Flash 空间，支持 WiFi 蓝牙等常见的通信方式

安装 MicroPython

MicroPython 是专门为硬件开发实现的 Python 版本

它可以使用 Python 的大部分语法，并支持了丰富的硬件相关的库

烧录命令：

esptool.py --chip esp32 --port **/dev/tty.usbserial-110** --baud 460800

write_flash -z 0x1000 esp32-20220618-v1.19.1.bin

总结

1. MicroPython 是专门为硬件定制的 Python 版本
2. MicroPython 需要烧录到 ESP32 的 Flash 才能正确运行

课后作业

请你将 MicroPython 烧录到 ESP32 开发板，并通过终端连接到开发板。

怎样通过 MQTT 协议构建消息队列?

目录

消息队列在物联网开发中的用途

- 队列机制为设备并发提供了有效的缓存
- 消息队列生产者消费者设计模式：两大角色之订阅者、发布者

使用消息队列实现数据通信

物联网中经常使用 MQTT 协议实现消息队列

目前 MQTT 协议最新版本为 5.0 版本，典型的 EMQX、HBMQTT 等都能完整支持

总结

1. 消息队列有效解决了物联网设备数量多，高并发的问题

2. 消息队列服务器除了保证消息的及时、稳定外，还在 MQTT 协议基础上提供了 QOS、权限验证等基础服务

课后作业

请你使用 MQTT 协议，实现客户端和服务端通信。

注： 服务端可以采用 EMQX、HBMQTT 等 MQTT 服务端实现

在 OLED 屏幕和手机远程同时显示温度

目录

开发基于 MQTT 协议的订阅者

为设备联网：

```
import network
sta = network.WLAN(network.STA_IF)
sta.active(True)
sta.connect(ssid, pwd)
```

开发基于 MQTT 协议的订阅者

订阅 MQTT 服务器：

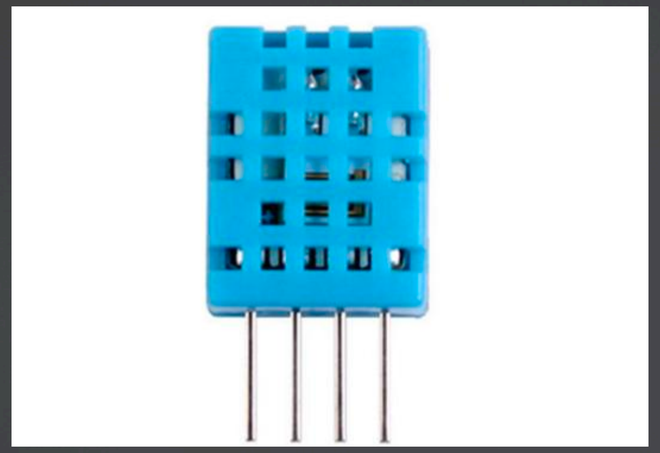
```
from umqtt.simple import MQTTClient
mqtt_client = MQTTClient(服务器信息)
# 订阅并编写(回调)处理逻辑
mqtt_client.set_callback(mqtt_callback)
mqtt_client.connect()
```

使用 MicroPython 控制 IO 引脚

```
from machine import Pin

self.pin = Pin(引脚编号, Pin.OUT)
```

使用 MicroPython 控制 IO 引脚



DHT11 是一款有已校准数字信号输出的**温湿度传感器**。

其精度湿度 ±5%RH， 温度 ±2℃，量程湿度 5~95%RH， 温度 –20~+60℃。

使用 MicroPython 控制 IO 引脚

通过 DHT11 获取温度的方法：

```
import dht
d = dht.DHT11(dht11_pin)
d.measure()
d.temperature()
```

使用 MicroPython 控制 IO 引脚

将温度显示在 OLED 屏幕的方法：

```
from ssd1306 import SSD1306_I2C
i2c = machine.SoftI2C(scl=machine.Pin(18), sda=machine.Pin(19))
oled = SSD1306_I2C(128, 64, i2c, addr=60)
oled.text('Hello World', 0, 20)
oled.show()
```

总结

1. MQTT 协议可以让 MicroPython 成为订阅者的角色，通过 machine 库读取引脚数据，实现网络通讯

2. 温度传感器有多个引脚，要注意引脚的顺序，不要将VCC和GND反向插入

课后作业

请你使用 MQTT 协议控制灯珠，通过远程控制实现灯珠的打开、关闭以及颜色修改。