生 产 实 习 报 告

生产实习单位 中国铁路设计集团有限公司

实 习 时 间 2022 年 7 月 4 日 至

2022 年 7 月 17 日 止

指导人员姓名 郭容昱

指导教师姓名 余晓静

学 生 姓 名 林佳奕

电子与信息工程学院 院（系）

计算机科学与技术 专业

大三 年级

目录

一、 实习概述

本次实习，我在中国铁路设计集团有限公司选择了一个课题，进行了为期两周的研究。具体的实习流程为：

第一周：熟悉工作环境，选择课题，配置开发环境，完成课题基本内容。并在第一周周四进行了第一次的验收。

第二周：改进第一周的成果，完成扩展内容，并在第二周周五进行了第二次的验收，验收通过后撰写实验报告。

期间，在第二周的周三、周四，我参加了两次线上的专业实习讲座，收获丰富。

以下是我对本次实习介绍与总结。

二、 实习项目介绍

2.1 课题介绍

我选择的课题是：基于 ESP32 物联网芯片的机器人控制和视频分析。

铁路运维管理市场前景广阔，我们已经研发了数字孪生的底座，并应用于雄安站等多个大型站场。运维管理市场有迫切的需求，解决重点机房、封闭管廊等不适合人员进入的场景巡检问题，尤其是电力系统复杂、空气和温度条件差的地点更加突出。研发智能巡检机器人系统，是我们未来产品产业化的重要方向。

本次实习的主要任务为：

1） 使用 PC 端大疆客户端程序执行 PID 巡线。

2） 将 ESP32 连接到大疆 EP 的数据接口和供电接口。

3） 将 DS18B20 测温模块、MQ 烟雾传感器、TELESKY 火焰传感器模块，连接到 ESP32 或 EP 机器人上用于灾害检测。

4） PC 端发现灾害后，通过 wifi 改变 ESP32 的 ADC 数据，大疆 EP 读取到 ESP32 的数据改变则执行返回起点程序段。

5） 扩展内容：利用 Yolov5 模型，对 ESP32 的摄像头中的视频流进行火焰、烟雾的检测，若检测到烟火则将数据和图片记录到数据库中，并执行返回程序。

6） 给出完整的研究过程、内容、代码，形成实习报告。

2.2 模块设计与实现

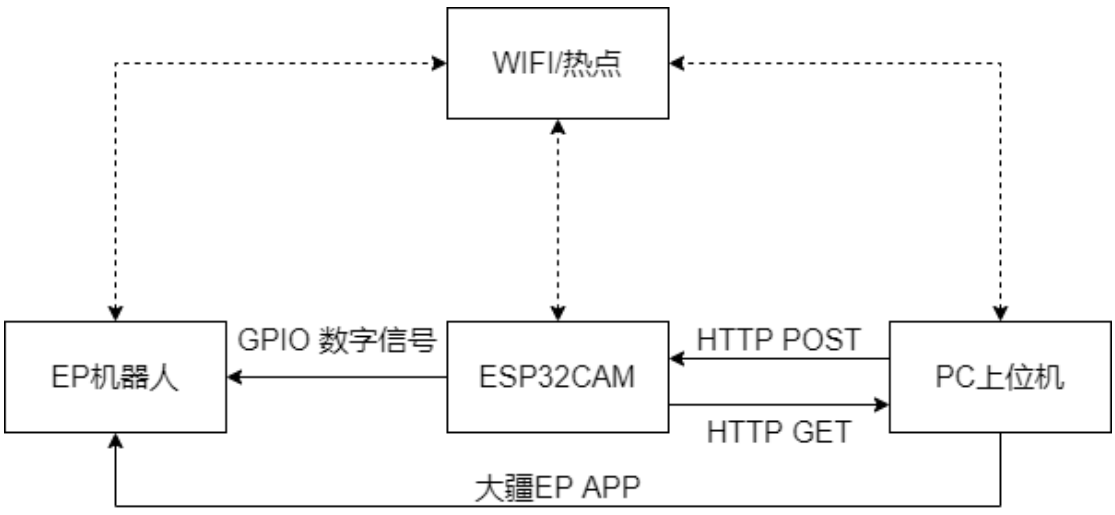
2.2.1 总体设计

不难看出，这个项目主要分为三个部分：EP 机器人、ESP32CAM 芯片、上位机 PC。

其中 EP 机器人负责巡线，接收控制信号和传感器数据；ESP32CAM 芯片接收向 EP 机器人发送控制信号，向 PC 发送视频流和数据；PC 负责对视频流进行目标检测，若发现灾害则向 ESP32CAM 芯片发送控制信号，并将灾害信息存入 MySQL 数据库中。

在信号控制方面，EP 机器人与 ESP32CAM 之间采用**数字信号**进行控制（ESP32CAM 芯片与机器人传感器之间使用杜邦线进行连接），ESP32CAM 与 PC 之间采用 Wifi 进行控制，PC 通过 **HTTP GET** 请求获取 ESP32CAM 的视频流，并通过 **HTTP POST** 请求发送控制信号；PC 通过大疆 EP 机器人 APP，将巡线程序通过 WIFI 下载到机器人中。三者需要连入同一 WIFI，即可在局域网内进行数据和控制的交互

具体三者之间的关系如下：

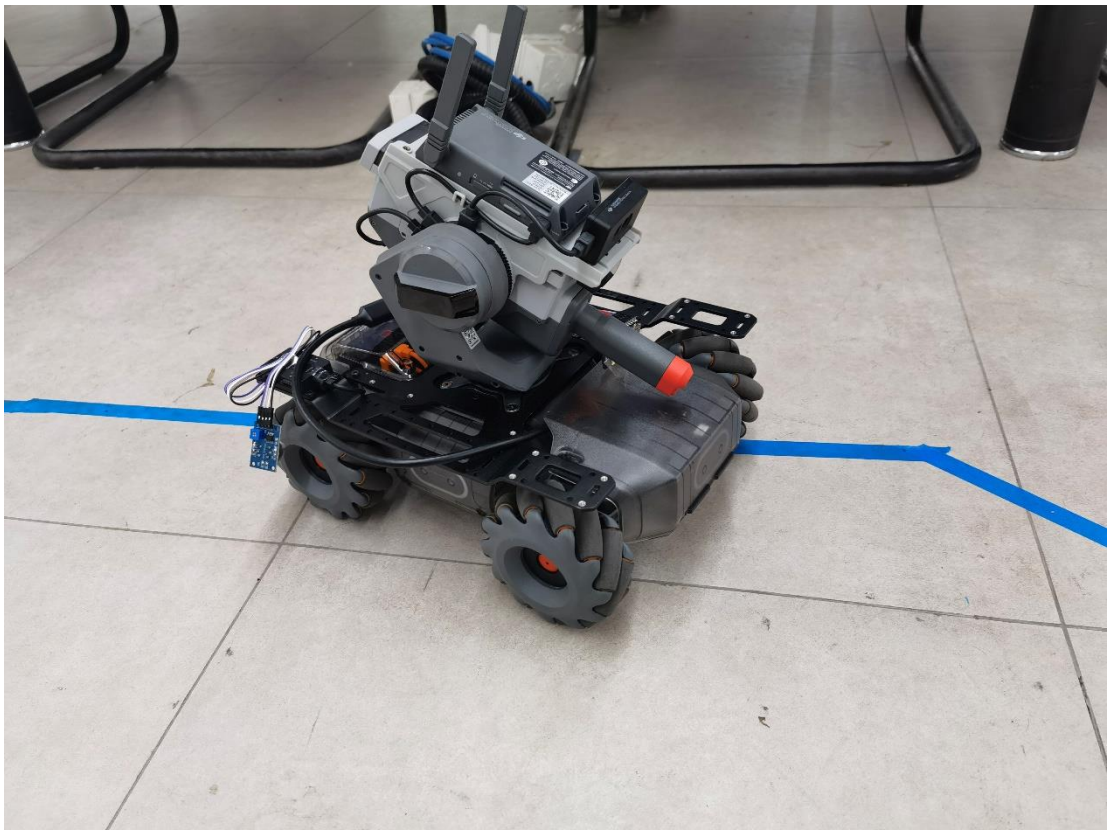


2.2.2 EP 机器人部分

EP 机器人主要负责的工作包括：

1. 沿指定路线进行巡线
2. 安装传感器，当传感器信号达到一定阈值则启动返航程序
3. 接收 ESP32CAM 传来的数字信号，当信号为高电平则启动返航程序

下图是 EP 机器人的外观：



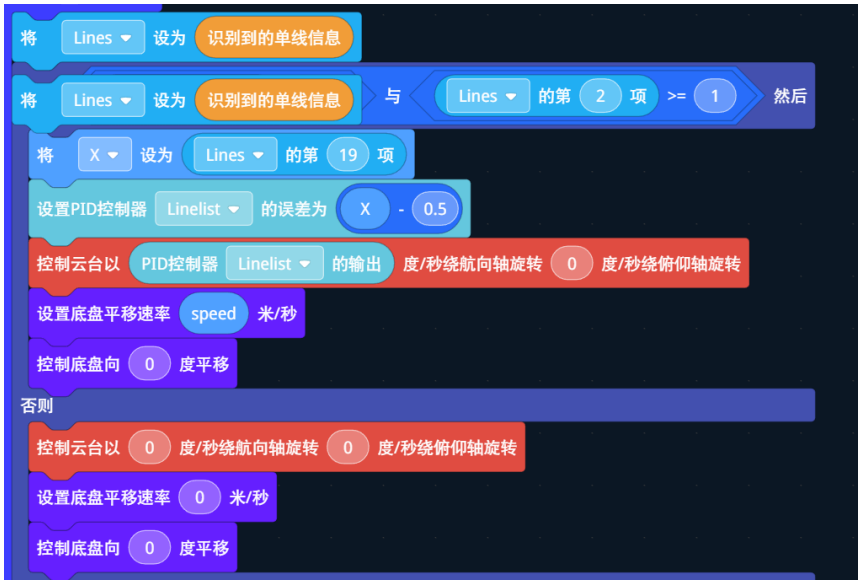
2.2.2.1 巡线工作

巡线的场地布置如下：

这个场地是由闲置的会议室改造成的，也是我本次实习的工作环境。小车会

从红心处出发，并沿着蓝色的边线进行环绕。期间如果检测到灾害，EP 机器人会加速回到红心处，并在返回时闪烁 LED 灯。

巡线的基本原理是通过 EP 机器人摄像头的**线识别功能**，可以追踪蓝色的线，摄像头会返回最近的 10 个点，利用这些点的坐标角度信息控制方向。



（本项目 EP 机器人采用 stretch 语言开发）

当检测到火灾后，小车会进行返航，回到起点处。起点处放置红心标志，这里利用 EP 机器人摄像头的**标签识别功能**，设置识别范围为 0.5m，当检测到红心标志时便停下小车。

2.2.2.2 传感器安装与检测

传感器安装在小车的侧部，每个传感器有 2 个端口，每个端口有 4 个引脚，具体配置如下：

引脚	说明
GND	接地
VCC	供电
I/O	数字信号端口，低电平为 0，高电平为 1
AD	模拟信号端口，数据范围为 0-1023

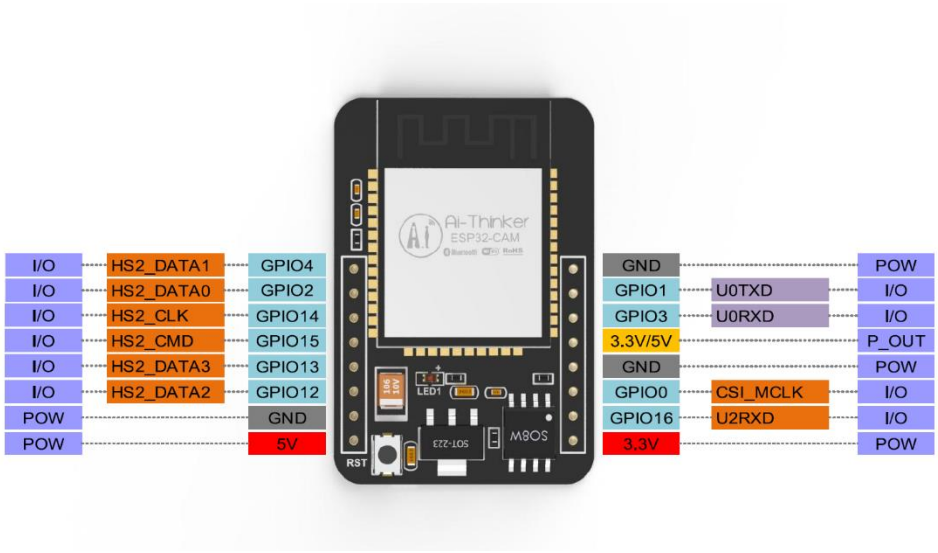
一般的传感器具有 3-4 个引脚，其中必有 GND 和 VCC 引脚，还具有 A0 口和 D0 口（有些没有 A0），这与 EP 传感器是正好对应的，因此可以直接安装。

本次实验使用了三个传感器：火焰传感器模块，烟雾传感器模块和温度传感器模块。其中温度传感器模块仅具备数字信号端口，且需要较为复杂的解码工作，因此安装在 ESP32CAM 的端口上；其余两个模块可以直接读出模拟信号，且 ESP32CAM 芯片的引脚不够用，因此直接安装在传感器上。

在程序中，可以直接读取传感器端口的模拟/数字信号值。

2.2.3 ESP32CAM 芯片部分

ESP32CAM 芯片是安信可（AI Thinker）公司开发的一款无线 WiFi 模块，可以即插即用，内置 OV2620 摄像头，芯片外部有 16 个引脚，包含 1 对串口、3 个 VCC、3 个 GND，其余为支持模拟信号和数字信号的 GPIO 管脚；基于 Arduino 开发，上手容易，配备众多例程，便于参考。下图为 ESP32CAM 引脚图。



本项目 ESP32CAM 部分改自于 Arduino studio 中提供的 CameraWebServer 例程，以下是 ESP32CAM 开发中比较重要的部分。

2.2.3.1 HTTP 服务器

在 Arduino stduio 的例程中，HTTP 服务器初始化工作已经做好，因此以下主要介绍如何添加 web 服务器的路由（URL）

首先，我们要定义一个函数，规定访问这个路由时进行的处理，即响应函数

```
static esp_err_t ctrl_handler(httpd_req_t *req)
```

之后，我们构建路由，方式是创建 httpd_uri_t 结构体，将 URL 信息，访问方法，响应函数。

```
/*控制部分*/
httpd_uri_t control_uri = {
    .uri = "/control",
    .method = HTTP_POST,
    .handler = ctrl_handler,
    .user_ctx = NULL
};
```

这个结构体定义了“/control”这一路由，方法为 POST（即只能向这个路由发送 POST 请求），处理函数为 ctrl_handler。

最后，我们还需要把这个路由注册到服务器中，使用如下函数

```
httpd_register_uri_handler(camera_httpd, &capture_uri);
```

本项目中定义了 4 个路由，如下表：

路由	方法	说明

/capture	HTTP GET	获取摄像头的截图
/stream	HTTP GET	获取持续的视频流（实际未用到）
/control	HTTP POST	向 ESP32CAM 发送控制信号
/temperature	HTTP GET	获取温度

2.2.3.2 摄像头采集

摄像头采集的部分非常复杂，这里不再对原理进行描述，仅说明一下如何设置视频帧大小、清晰度等。

Arduino 中，最重要的两个函数分别是 setup 函数和 loop 函数，其中 setup 函数用于进行一些初始化的操作，仅执行一次且在 loop 函数之前，而 loop 函数可以看做是一个循环体的内部。

因此我们要修改摄像头参数，需要修改 setup 函数内部。摄像头参数的定义在结构体 camera_config_t 中，我们需要修改这个结构体的部分变量

```
camera_config_t config;
```

要修改图片清晰度，我们需要修改变量 jpeg_quality，其值范围在 10-63，越低越清晰

```
config.jpeg_quality = 12;
```

要修改视频帧大小，需要修改变量 frame_size，这个变量类型可以认为是一个二元组，我们不妨用宏定义为其赋值。这里表示帧大小为 480*320

```
config.frame_size=FRAMESIZE_HVGA;
```

2.2.3.3 温度传感器采集

温度传感器我们在前文介绍过，其只有三个端口，没有模拟信号端口。因此我们需要通过一定的协议解析从数字端口获取到的数据，得到温度值。

这里同样参考了 Arduino Stduio 例程，稍作修改得到我们使用的函数

函数如下：

```
float GetTemp(){
  //returns the temperature from one ds18S20 in DEG Celsius
  const int Temperature_Pin=12;
  OneWire wire(Temperature_Pin);
  byte data[12];
  byte addr[8];
  if ( !wire.search(addr)) {
      //no more sensors on chain, reset search
      wire.reset_search();
      return -1000;
  }
  wire.reset();
  wire.select(addr);
  wire.write(0x44,1); // start conversion, with parasite power on at the end
  byte present = wire.reset();
  wire.select(addr);
  wire.write(0xBE); // Read Scratchpad
  for (int i = 0; i < 9; i++) { // we need 9 bytes
    data[i] = wire.read();
  }
  wire.reset_search();
  byte MSB = data[1];
  byte LSB = data[0];
  float tempRead = ((MSB << 8) | LSB); //using two's compliment
  float TemperatureSum = tempRead / 16;
  return TemperatureSum;
}
```

大致过程为：首先向温度传感器写入字节 0x44，并将电平设置为搞；之后重置，在写入字节 0xBE，之后读出 9 个字节，拼接出数值后转换为浮点数，再除以 16 得到温度值。

2.2.4 上位机部分

上位机采用 Python 开发，前后端采用 flask 开发，使用 flask_sqlalchemy 连接 MySQL 数据库，并添加了 Yolov5 接口。

2.2.4.1 数据交互部分

根据前文的介绍，PC 和 ESP32CAM 采用 HTTP 协议通信。这里我们使用 requests 库来发送 HTTP GET 和 POST 请求。

首先是采集温度数据，这里我们使用 HTTP GET 请求。

```
try:
    with requests.get(TEMP_URL) as r:
        json_data=json.loads(r.content)
        temp=json_data["temp"]
except Exception as e:
    #raise e
    print("GET from {} fail!".format(TEMP_URL))
```

其次是采集视频数据，这里我们同样用 HTTP GET 请求。由于采集连续视频流反而不利于检测，因此我们定期采集截图，并且因为图片是二进制编码，我们要设置 stream=True。

```
try:
    with requests.get(IMAGE_URL,stream=True) as r:
        img=r.content
except Exception as e:
    print("GET from {} fail!".format(IMAGE_URL))
```

最后是发送控制信号，我们采用 HTTP POST 请求，发送”0”表示无火灾，发送”1”表示有火灾。

```
try:
    with requests.post(CTRL_URL,code,timeout=3) as r:
        res=r.content
except Exception as e:
    print("POST to {} fail!".format(CTRL_URL))
```

为了实现轮询，我们启用线程来定期进行数据的获取与发送，利用全局变量（可以理解为信号量）控制是否发送数据。

2.2.4.2 flask 部分

本项目中，设置了两个页面，第一个页面是控制页面，用于采集温度和数据的页面。第二个页面用于查看火灾记录。

第一个页面的样式如下：

通过点击“获取温度”，可以更新温度信息；下方的视频区域为 ESP32CAM 摄像头采集到的系信息，再下方的“控制”可以手动设置机器人返航以及重置检测端口。

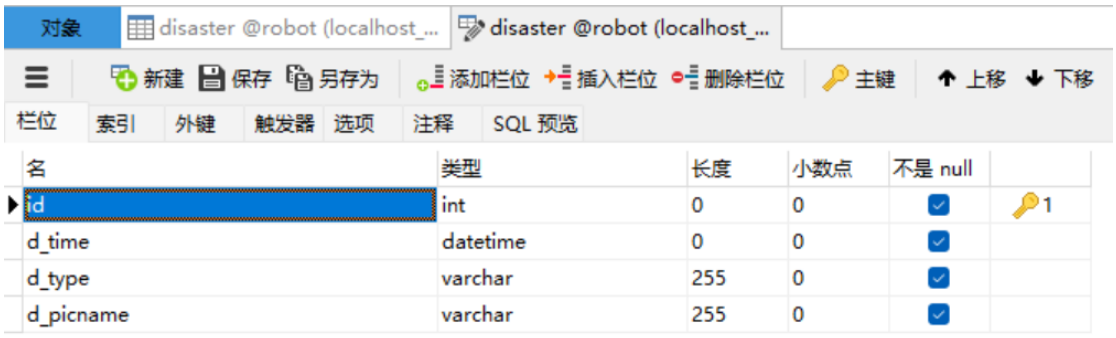
按键部分采用 ajax 进行交互。

第二个页面的样式如下：

可以看到以表格的方式列出了火灾的编号、发生时间、检测类型和现场图片。

2.2.4.3 MySQL 数据库部分

首先是 MySQL 数据库，我们建立名为“robot”的数据库，并建立表“disaster”



字段较少，id 是主键，d_time 为火灾发生时间，d_type 为火灾种类。d_picname

为火灾图片名称，图片存储在上位机的”static/image”文件夹，前端访问的时候，通过 url_for 函数可以加载。

之后是 flask_sqlalchemy 部分，我们采用 ORM 模型，首先定义一个与 disaster 表格式相同的类。

```
class Disaster(db.Model):
    __tablename__="disaster"
    id=db.Column(db.Integer,primary_key=True)
    d_time=db.Column(db.DateTime())
    d_type=db.Column(db.String(255))
    d_picname=db.Column(db.String(255))
    def __repr__(self) -> str:
        return "{{id:{0},d_time:{1},d_type:{2}}}".format(
            self.id,self.d_time,self.d_type
        )
```

连接的 uri 如下：

```
db_uri='mysql+pymysql://{0}:{1}@{2}:{3}/{4}?charset=utf8'.format(
    'root','012704','localhost','3306','robot'
)
```

```
myapp.config['SQLALCHEMY_DATABASE_URI'] = db_uri
```

之后，使用 db.query 查询，使用 db.session 进行增删管理

```
def Disaster_Insert(disa:Disaster):
    db.session.add(disa)
    db.session.commit()

def Disaster_Query_All():
    return Disaster.query.all()
```

2.2.4.4 Yolov5 部分

Yolov5 的原理很复杂，这里并不打算用大量的篇幅介绍 Yolov5 的原理，只介绍如何编写接口以便检测。

首先，我从 github 上下载了 yolov5 的源码，简要阅读后发现其检测函数必须以命令行的方式输入，且每次检测都要加载一遍权重模型，这并不能满足我们上位机的需要。

因此我将检测函数 detect.py 进行了修改，主要分为两个部分：

1. 加载权重模型，初始化参数
2. 导入数据，进行检测。

3. 根据检测结果，决定是否保存

首先第一个部分，我删除了原先的命令行部分，发现大量的参数需要改，为此，我直接命名一个同名类，并将参数写入

```
class opt:
    agnostic_nms=False
    augment=False
    classes=None
    conf_thres=0.4
    device='0'
    img_size=640
    iou_thres=0.5
    output='.\\static\\image'
    save_txt=False
    update=False
    weights=['.\\best.pt']
    webcam=False
    half=False
    device=None
    names=None
    colors=None
```

较为重要的参数包括：

- img_size，规定了图片的大小；
- ouput，规定了输出检测结果的位置
- weights，规定了输入权重模型的位置
- webcam，是否使用网络摄像头，由于其规定的网络摄像头与 ESP32 略有差异，这里设置为 False

初始化的工作包括启用 CUDA，加载权重模型，导入命名和颜色规则等。

之后是检测部分，由于我们上位机获取的是图片二进制流，而 yolov5 提供的接口不能直接使用，最开始我的方式是保存到本地再传入本地目录，但这样效率太低，因此我仿照着 Yolov5 导入模型的接口，自行编写了一个类 LoadBytes。

```
if opt.webcam:
    view_img = True
    cudnn.benchmark = True  # set True to speed up constant image size inference
    dataset = LoadStreams(source, img_size=imgsz)
elif srctype=="other":
    save_img = True
    dataset = LoadImages(source, img_size=imgsz)
else:
    save_img = True
    dataset = LoadBytes(source, img_size=imgsz)
```

这个类的定义如下：

```
class LoadBytes:
    def __init__(self, img_byte:bytes, img_size=640):
        self.byte=img_byte
        self.img_size=img_size
        self.nf=1
        self.cap=None
        self.mode="images"

    def __iter__(self):
        self.count = 0
        return self

    def __next__(self):
        if self.count == self.nf:
            raise StopIteration
        self.count+=1
        path="./output.txt"
        img0=np.asarray(bytearray(self.byte),dtype="uint8")
        img0=cv2.imdecode(img0,cv2.IMREAD_COLOR)
        img = letterbox(img0, new_shape=self.img_size)[0]

        img = img[:, :, ::-1].transpose(2, 0, 1)  # BGR to RGB, to 3x416x416
        img = np.ascontiguousarray(img)

        return path, img, img0, self.cap

    def new_video(self, path):
        self.frame = 0
        self.cap = cv2.VideoCapture(path)
        self.nframes = int(self.cap.get(cv2.CAP_PROP_FRAME_COUNT))

    def __len__(self):
        return self.nf  # number of files
```

其中__init__方法用于导入图片二进制流，__iter__，__next__，__len__方法用于迭代使用（这是其他导入类公有部分，若不写则后面检测函数要大改）。new_video 没有用到。

其余部分不动即可。最后根据检测结果，返回一个字典，字典的键是待检测

的种类，包括火焰、烟雾，值是检测到的种类的数量。

之后只需要如下调用即可。

```
detect_result=detect(img,srctype="bytes")
```

2.3 验收效果

2.3.1 第一次验收

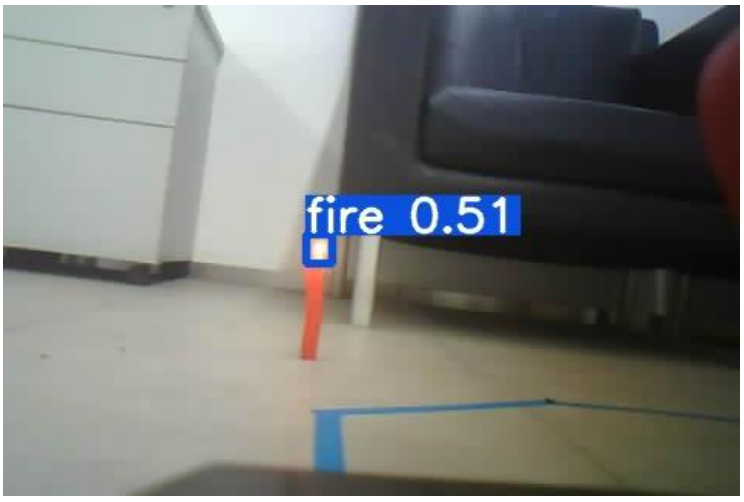
第一次验收主要是对温度、烟雾、火焰传感器进行了测试，由于室内环境和实习单位防火政策的限制，没有完全复现灾害现场，得到的结论如下：

1. 火焰位于火焰传感器探头 1m 范围内时，火焰传感器模拟信号值处于 100 以下，为了防止干扰，将检测阈值调整至 200。
2. 火焰传感器对于太阳光过于敏感，以至于不把窗帘拉下会影响检测，因此这种火焰传感器仅适用于室内。不过考虑到实际应用，这个限制是可以接受的
3. 烟雾被烟雾传感器接收时，模拟信号值处于 600-800，因此将阈值设置为 600。但是由于室内限制，烟雾较小，因此效果并不好。
4. 由于没法模拟高温环境，因此实际没有检测温度，仅仅查看了温度读数，读数符合室内环境。

2.3.2 第二次验收

第二次验收主要是对目标检测效果进行验收，由于训练的不是很好，因此直接采用网上开源的火灾检测权重模型。

如下图，即使火焰较小，也可以正确检测。



当然，也出现过错误检测的情况，如将传白色衣服的行人检测成烟雾。

但指导老师认为总体达到了课题要求，因此通过了第二次验收。

2.4 小结

针对本项目提出一些结论和改进方向：

1. 大疆 EP 机器人的控制方式非常有限，因此可以考虑使用其他厂家的机器人产品或者使用 stm32 开发小车
2. ESP32CAM 芯片的管脚有限，不能容纳全部传感器，因此本人建议换用其他芯片或者结合其他芯片共同开发，芯片之间采用串口/SPI 通信。
3. ESP32CAM 的摄像头较小，且质量不够清晰，可以考虑换用其他摄像头，如 OV7660
4. Yolov5 模型存在着一定的误检的可能，因此在实际应用中，建议以巡视环境为基础采集数据并重新训练。

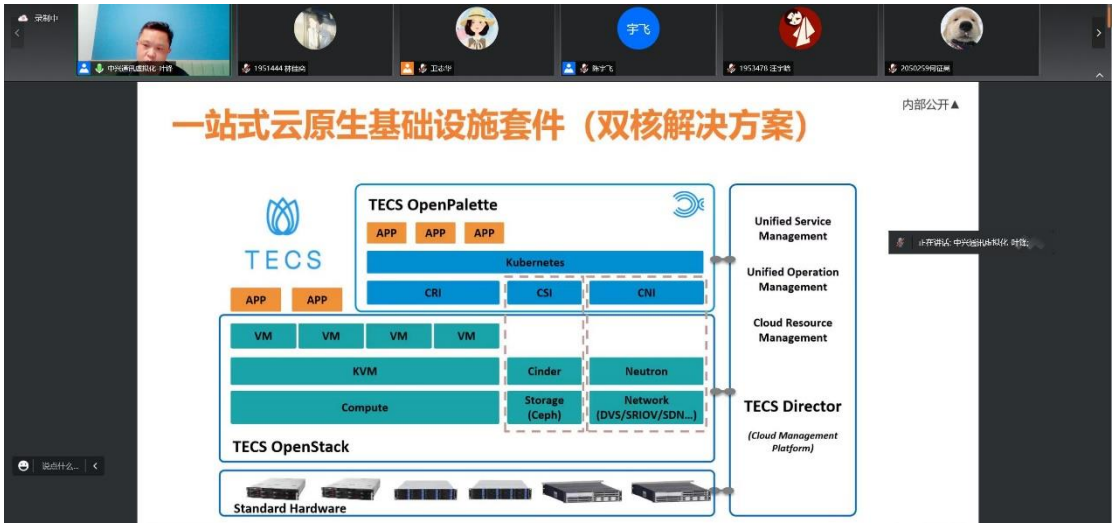
三、 实习讲座介绍

在本次实习期间，我参加了 4 场线上实习讲座，以下是我对每场讲座的总结和体会。

3.1 第一场讲座

第一场讲座是中兴科技集团的老师开展的讲座，讲述内容主要是通信虚拟化技术，包含了诸如操作系统、体系结构等内容。

通过这场讲座我了解到了通信虚拟化技术的相关内容，了解了大型企业是如何开展计算机架构的设计的，并且了解到了中兴集团的工作待遇、薪资等问题，中兴 HR 很热情，给每个同学解答问题，令我印象深刻。

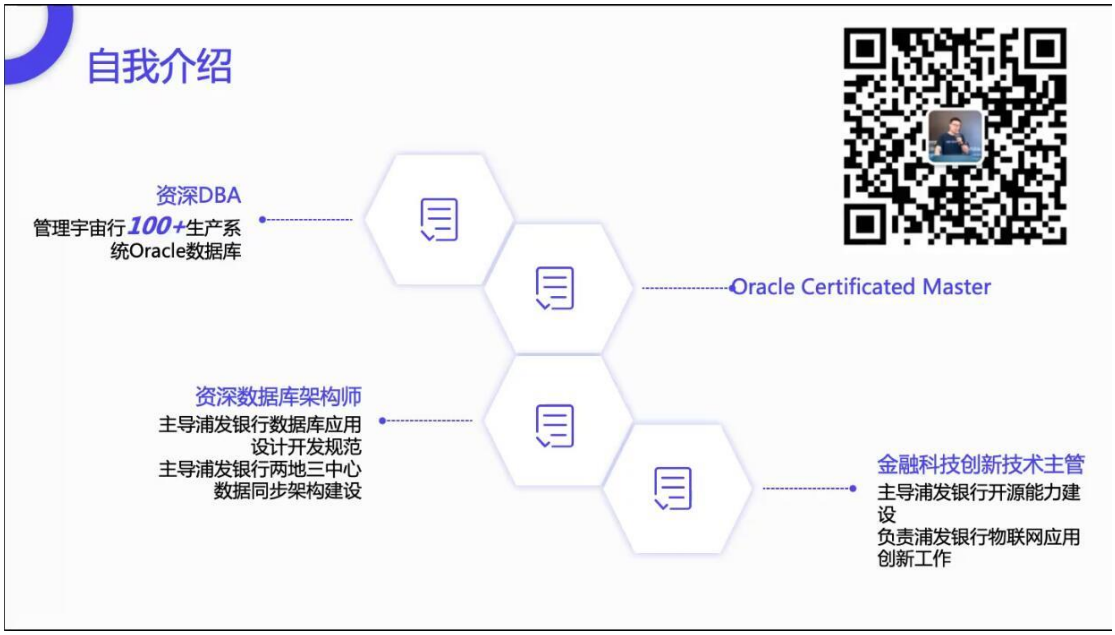


3.2 第二场讲座

第二场讲座是浦发银行开展的，讲述内容主要是物联网金融方向。

演讲人在浦发银行工作多年，是资深 DBA，曾去多个高校讲解数据库领域相关知识，不过这位演讲人在物联网金融方面也是有自己独特的见解。

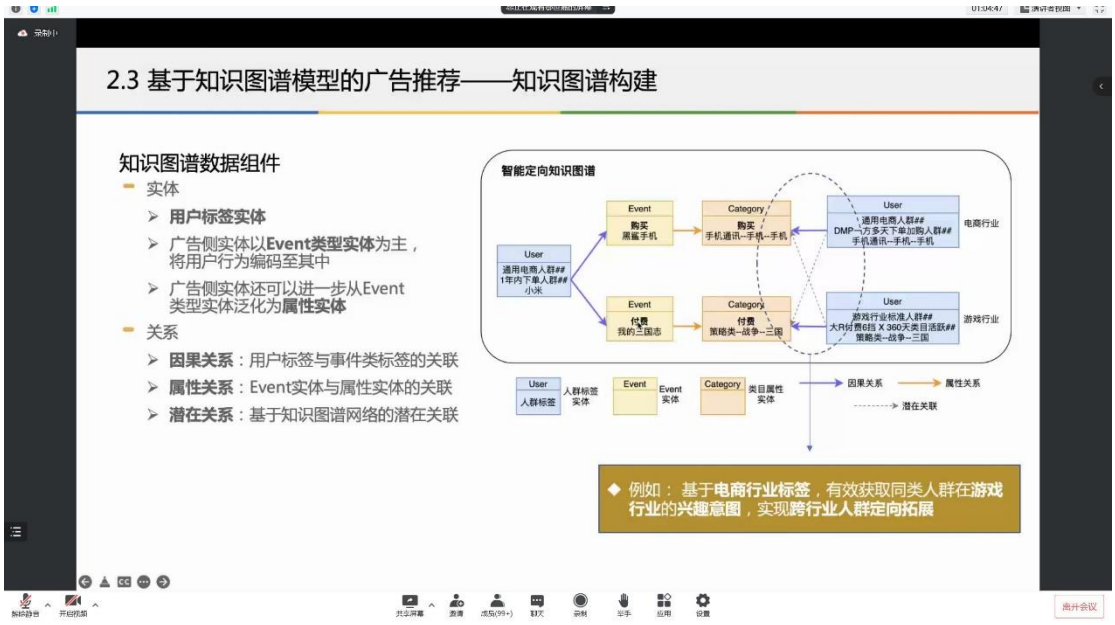
其中，令我印象深刻的是演讲者放出的一张在工作期间吃饭的图片，通过这张图片我体会到了工作的辛苦，因为即使是 leader，也要亲自参与会议，并与普通员工一样，吃着最普通的盒饭。因此这也让我对工作的辛苦有了进一步的认识。



3.3 第三场讲座

第三场讲座是腾讯广告团队开展的，演讲人是同济博士毕业的校友，现在腾讯广告团队算法岗。算法岗是我未来可能考虑的工作方向，因此我较为感兴趣。

演讲人讲述的内容主要是知识图谱，通过图文结合的方式，演讲人将较为复杂的知识简单化，并举了很多贴合实际的例子来进一步讲解，令我收获丰富



3.4 第四场讲座

第四场讲座是由 KLA（世界著名的专业美资半导体（芯片）设备供应商）开展的讲座。其讲述内容主要包括整个公司的介绍和其内部软件技术的应用。

其中，一张图片令我印象深刻（见下图），我发现身边很多电子设备，都离不开这家公司。诚然，随着科技的发展，人们的生活逐渐“电子化”“数字化”，越来越多的电子产品走进我们的视野，而这必然离不开半导体技术的应用。

通过这次讲座我了解到的很多半导体相关的知识，收获丰富。

四、 实习心得体会

两周的实习虽然短暂，却也令我收获丰富。

一方面，我在完成实习单位布置的课题的同时，了解到了所在单位的主要业务。虽然我所在的单位是以铁路工程项目为主的设计单位，与“互联网大厂”的

业务并不一样，一般人也可能学计算机专业的学生也就只能去这样的单位“修网线”，然而并不是这样的。实际上，该公司信息部分的主要的业务分为这么几个方向：一是传统的软件开发，为其他部门开发维护办公套件等；二是利用 3D 引擎渲染，实现施工现场、车站轨道环境的 3D 仿真；三是智慧运维，即基于嵌入式人工智能辅助施工。我所选的课题即属于第三个方向。

实际上这个课题是很有现实意义的。检测火灾，往往通过传感器或人工检查，然而这样的检测可能会有失效的时候，亦或是有“死角”未被检测到，因此使用巡线机器人检测，可以尽量避免这样的意外，同时再使用 Yolov5 目标检测模型可以更好的提高检测率。

另一方面，我体会到的上班的辛苦。尽管作为实习生，我不需要“加班”，但是每天 6 点半起床，晚上 6 点多到家，12 小时在外还是让习惯于睡懒觉的我感到了一些不适应，并且在单位不能太过松懈，需要按时完成项目，因此我感觉到这样的工作是需要一定的注意力的。而实际的入职，工作量要大于实习，并且还需要考虑绩效、考核等诸多因素，还可能因为工作做不完而“加班”，因此会更加辛苦。综上所述，我需要从心理上、身体上、能力上做好准备。

总的来说，实习让我学到了很多课上学不到的知识，令我收获丰富。若有机会我会继续找其他的公司参加学习，提升自己的计算机技术。