一、提出背景和实用价值

随着社会经济和人民生活水平的提高，人们的旅游、出行、文体与社会活动需求日益增加，在道路、场地出入口进行人流量的监控以实现人流量的合理规划具有重要的意义。

当前的人流量监控系统主要有两种类型：第一种是以商场监控摄像头系统为代表的，面向后台服务人员，摄像头固定的监控系统；第二种是以各类智能手机网络地图 APP 为代表的，通过各类手段间接获取人流量的，面向广大用户的人流量监控系统。前者的缺点是摄像头系统布置之后位置固定，缺乏灵活性，并且监控效果不向用户公开。后者的缺点是用户无法了解实际的人流量状况，只能大致的判断“通畅”还是“拥挤”。

针对当前人流量监控系统的不足之处以及需求的痛点，我们提出了基于云端图像处理的可移动多节点人流量监控系统。它应满足以下的指标：

(1) 多个监控摄像头应该是可移动的，能够通过远程的控制使它移动到待监测的目标附近进行监控。

(2) 人流量的判断应该是基于云端图像处理的，即监控画面通过网络传回到后端服务器，通过 AI 进行图像处理进而分析人流量。

(3) 本系统应该要兼顾广大的用户的需要，让用户能够在不同设备上通过网络实时看到人流量的监控地图与每个监控节点的图像画面。

(4) 每个监控节点的移动情况要反映在人流量监控地图上。

二、系统方案

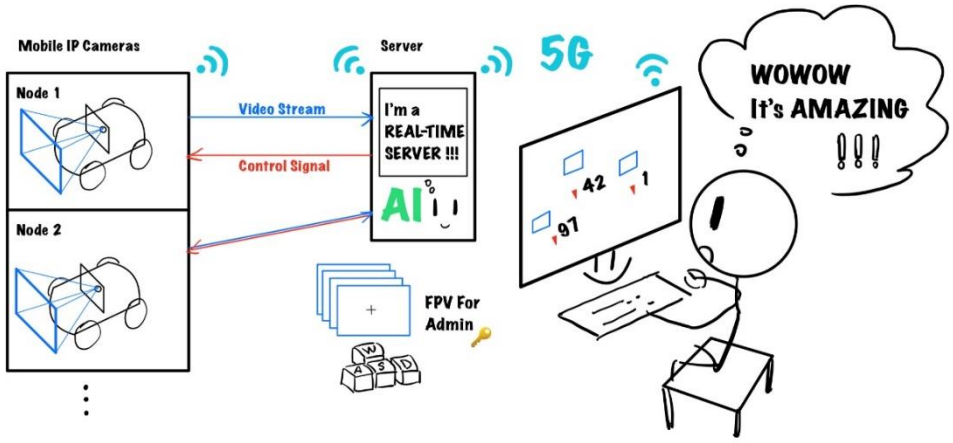


图 1　系统方案概览

本团队设计的移动多节点实时人流量监控系统有两个主要部分。

第一个主要部分是图像采集端。考虑到大部分成品 IP 摄像头并不是为了移动而设计的，我们采用手机作为图像采集设备。移动平台可以是任何载具，为了展示的便捷性，我们使用一个直流减速电机驱动的四轮小车作为移动平台。

第二个主要部分是服务器。在我们设计的系统中，服务器的第一个工作是接收视频、处理视频并进行人数检测。第二个工作是给系统管理员调动 IP 摄像头位置的通路，即让管理员能够通过键盘远程控制小车来达到移动监控设备，实现

不同位置检测人流量的目的。第三个工作是将相关信息传输到网页端，让用户能够看到实际位置的监控信息，了解哪个地方人流量比较大，从而更好地规划自己的行程。在 WEB 方面，我们采用 Flask 框架来实现服务器搭建。

AI 技术主要被用于人数检测，这个问题是机器学习中非常典型的问题；而 5G 技术则主要体现在实时视频传输、远程控制方面。但由于 5G 模块目前普及率不高，所以本小组用局域网来暂时代替 5G 网络。

三、功能与指标

1. 可移动监控节点的硬件设计

(1)图像采集端的设计

可移动监控节点应具备远程遥控管理、图像采集、网络连接、视频流传输、获取 GPS 信息等功能。

当前的智能手机具有丰富的传感器与强大的硬件性能，非常适合作为图像采集端的硬件。因此我们选择手机作为图像采集端的硬件。使用基于 MIT APP Inventor2 的 Wxbit 开发工具，结合 socket 插件、蓝牙与 GPS 模块，编写安卓应用程序，以实现通过网络与服务器通信，处理远程控制移动平台的命令，并回传 GPS 位置信息。

另外，我们选择常见的 IP 摄像头 APP 传输监控画面的视频流。在使用时只需先打开我们开发的服务端 APP，建立连接之后把它放到后台，在运行 IP 摄像头 APP 即可。对于安卓手机的兼容性较好，性能要求不高。

图 2 APP 开发界面

(2)可移动的平台

为了开发调试的简便，我们选择小车作为可移动的平台。手机作为监控节点的核心，已经具备大量的传感器，因此可移动的平台只需要非常少量的硬件，只需包括小车底盘、电机电源与驱动、蓝牙模块（与手机进行通信）、嵌入式最小系统（处理蓝牙通信的信息并控制小车）。

小车使用两节 18650 电池并联供电，可以通过 type-c 接口进行充电；使用 L298N 的 H 桥电机驱动控制直流减速电机，可前后移动并且通过差速转向。小车具有手机支架，便于固定作为图像采集端的手机。小车使用 STM32 作为控制系统，使用 STM32CubeMX，Keil MDK-ARM 开发，程序使用 freeRTOS 作为系统的管理和控制。

2. 服务器功能

服务器的主要功能有三：一是远程控制小车移动，二是不断接收图像并进行人数检测，三是将识别结果通过 Flask 框架呈现在网页上。

为了达到实时检测的性能，我们选用的目标检测算法为 YOLO 算法，使用 Ultralytics 在 COCO 数据集上训练的开源的 YOLOv5s 模型。这个模型整体大小只有 15MB，而且使用 CUDA 加速在 GTX960M（2GB）上面推理任意分辨率的一张图片平均只需要 0.033s，即 30.3FPS。视频传输方面，网页可流畅显示高达 1080p@30fps 的实时视频。

但是，由于 COCO 训练集和标注的原因，YOLO 在人体重叠、俯视人群和人体非常小的时候漏检率很高。这就需要另外一种算法来应对这样的场景。我们采用的是 SS-DCNet 模型。SS-DCNet 需要的显存比较多，若图像分辨率较大，比如推理 720P 分辨率的图片所需的显存基本需要 2GB 以上。同时，推理速度它会比较慢，在 CPU 上（为了防止显存不足）推理一张1024×768的图像需要 0.65s。

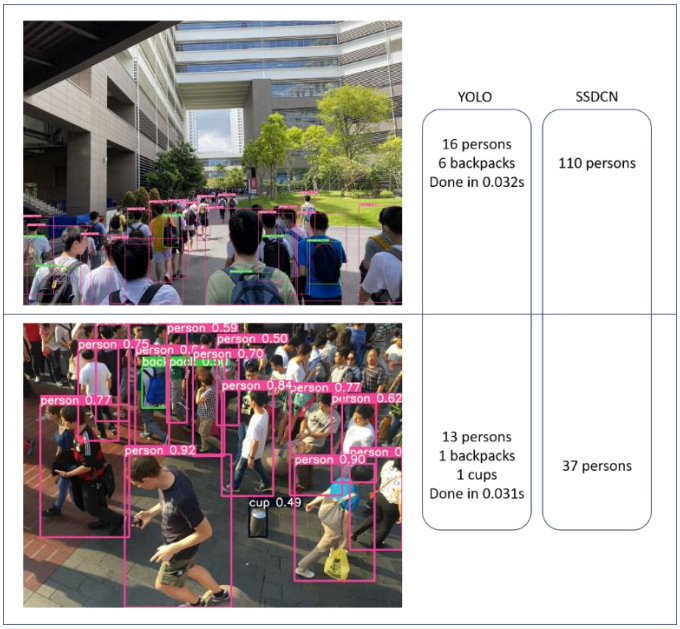


图 3　YOLO 和 SSCDN 识别结果对比

3. 网页部分

网页部分的主要功能有两个：第一是能够接收到服务器传输的移动平台 GPS 信号，并在地图上对其进行标注；第二是能够实时更新并接收服务器传输的视频流和人数检测数据，并将其展示在网页页面上。

当用户进入网站首页，首先会看到地图页面，上面的各个红色标注代表着目前移动平台的位置，旁边的数字代表画面中的人流量。用户可以点击自己想要了解的位置标注，进入目标地的移动平台实时画面来直观地感受当地的人流量。

四、实现原理

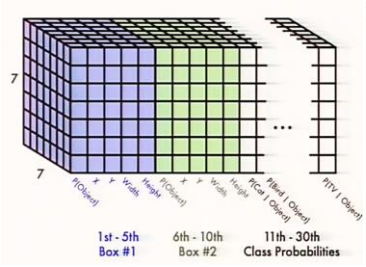
1. 目标识别与人群计数

(1) YOLO：参数化目标检测

目标检测有许多算法，大概可以分为两类，第一类是两步走的算法，即先用

某个网络来预测出图中有意义的物体的位置，再用 CNN 对目标进行分类，这类算法中的豪杰要数 Faster RCNN，速度 7FPS，MAP 为 0.73。第二类是一步到位算法，代表为 YOLO 和 SSD，速度直接杀到 30FPS 以上，满足实时目标检测的需求。YOLO 其实经过了很多版本的迭代了，现在官方的版本已经到了第四代，但核心思想一直都是把两步统一成一个回归问题。

一步到位指的是：将整个图片划分到7×7的格子之中，每一个格子可以看成一个输出单元，负责预测目标框的中心点坐标和宽高、框内包括有效目标的概率和在包括有效目标条件下为各个类别的条件概率。如下图所示，每一个格子实际上负责预测两个目标框，以及 20 个条件概率，避免两步走的方法本质上是把问题参数化，即把位置参数和每个分类的概率参数同时预测出来。



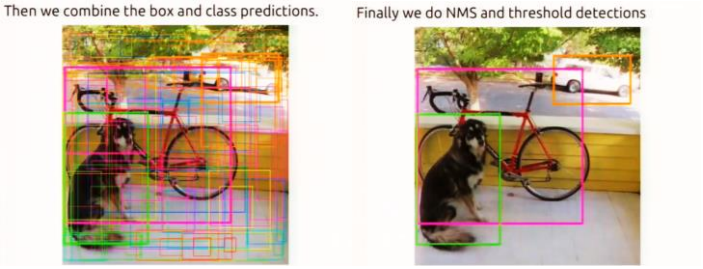


图 4　网络的输出数据(CVPR2016)　图 5　所有目标框和过滤后的结果(CVPR2016)

假设我们已经有一个训练好的网络模型，输出层给出的则是总共 7*7*2=98 个目标框，每个目标框对应了一个物体的分类和对应的置信度，注意这里给出的分类和置信度已经是排序后取最值后的结果。接着，为了从一堆预测框中选取合理的、有意义的预测，用阈值滤除低置信度的目标框，用 NMS 算法滤去重合的多余的框。

(2) YOLO 的网络架构和版本迭代

网络架构是这样子的：24 个穿插了池化层的卷积层——两层全连接层——输出层。卷积层的意义在于提取各种特征和物体与物体间、物体与背景间的关系。显然，这个网络的基础网络，又称特征提取网络，就是一个层数比较多的卷积神经网络。

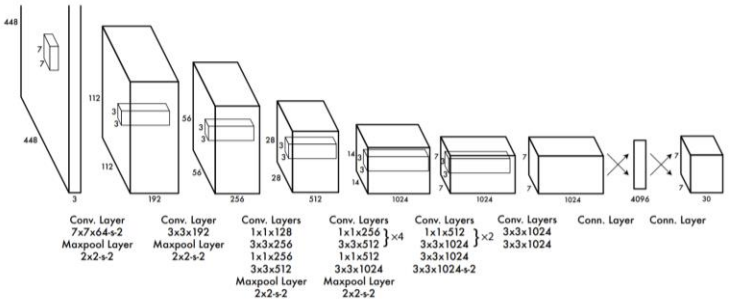


图 6　YOLOv1 网络架构[1]

作者在论文 2.4 节也明确分析了 YOLO 的缺点，大致复述如下。首先，YOLO 对靠的很近的物体，特别是挨在一起且中点都落在同一个格子上的情况，还有很小的群体，检测效果都不好，这是因为一个网格只负责预测两个框，并且只属于

一类。第二，当同一类物体出现的不常见的长宽比和其他情况时泛化能力偏弱。第三，由于损失函数的问题，定位误差是影响检测效果的主要原因，尤其是大小物体的处理上，还有待加强。

YOLO的后续版本用了各种技巧，比如添加数据集归一化层、借鉴Faster RCNN思想引入 anchor box、优化特征提取网络和损失函数、细分图片网格以及各种数据集增强……YOLO 随着这些技巧的应用也变得更加准确、更加快速。虽然网络的架构已经和最初差别很大了，但是“一步回归”的基本思想一直都没有变。

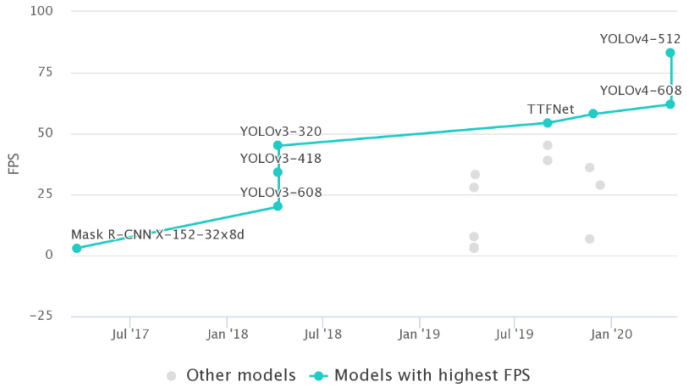


图 7　目标检测速度对比[4]

(3) 人群计数的密度图法

目前前沿的人群计数方法都是训练一个以原图像为输入、人群密度图为输出的、以 CNN 为基础的网络。我们对这个网络得到的密度图求和，得到一个估计的人数。下图是一个专用于人数识别的机器学习框架 C3F 的流程示意图，很好的描述了目前实现人群计数的一般方法。

(4) SS-DCNet

SS-DC 意思是“有监督的空间分治法”，SS-DCNet 的做法是在特征图上实现分治法，即在 CNN 基础网络的输出图（feature map）上实行上采样（upsampling）增大分辨率，然后对每个区域进行预测。有意思的是，这个网络并不是为了生成密度图，事实上它根本用不到密度图，因为它把回归问题转化为了分类问题。基本思想是人数可以被分成有限多个区间——{0}，（0，5]，……（20，inf），每个区间代表了一个类别。每个被分割的特征图被分类器分到一个人数区间，最后取中位数（最后一个区间取下界）进行求和得到最终的人数。

2. 计算机网络基本原理（5G 与 Wi-Fi 之间的异同）

经过查阅相关资料，我们了解了 5G 与最新一代 Wi-Fi 之间的异同。

在相同点方面，Wi-Fi 与 5G 之间的核心技术相同。Wi-Fi6 和 5G 都采用 OFDMA 与 MU-MIMO 技术，使带宽和接入容量都得到了大幅度的提升。

在不同点方面，目前来说，Wi-Fi 与 5G 在应用场景、获取成本以及终端普及程度方面存在较大差异。现阶段而言，Wi-Fi 的应用场景更广，成本低，普及程度高。但由于 Wi-Fi 是一项固定位置的近距离传输技术，满足不了人们的日常雪球，随着 5G 技术发展和 5G 基站的建设，5G 在这些方面终将赶上 Wi-Fi。

但从技术本质来说，5G 与 Wi-Fi 没有较大区别。因此，在本次项目中，从设

备获取的便捷性以及成本方面考虑，本团队使用 Wi-Fi 来模拟日常生活中 5G 技术的使用。

3. 服务器搭建

服务器分为 Web 前端和 Web 后端。

Web 前端是指在 Web 应用中，用户可以看得见、碰得着的内容，这其中包括了 Web 页面的结构、Web 的外观视觉表现以及 Web 层面的交互实现。在前端部分，本团队调用百度地图的 API，使用 HTML、JavaScript、CSS 等编程技术来实现想要的画面与交互。

而 Web 后端则指的是与数据库进行交互以处理相应的业务逻辑，需要考虑如何实现功能、数据的存取、平台的稳定性与性能等，即为前端的展示业务设计逻辑功能。后端部分，本团队则采用 Flask 架构，来实现数据之间的相互传输。

五、硬件框图

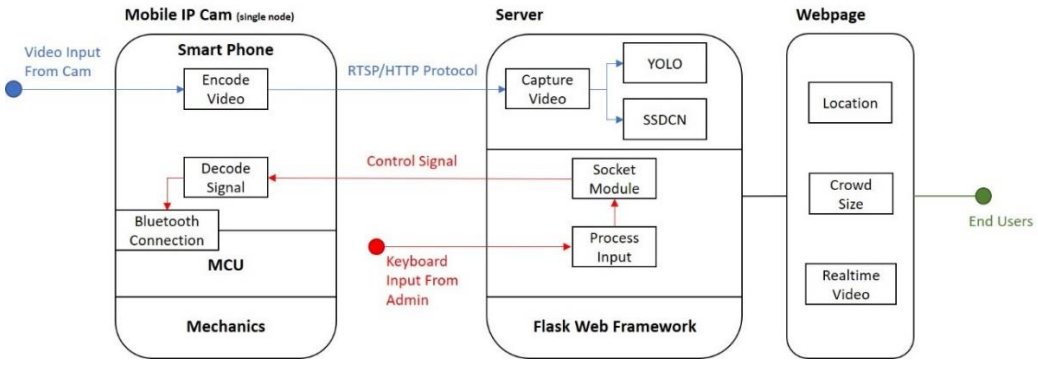


图 10　系统实现框图

小车硬件电路图见附录。

六、软件流程

1. 图像采集流程图

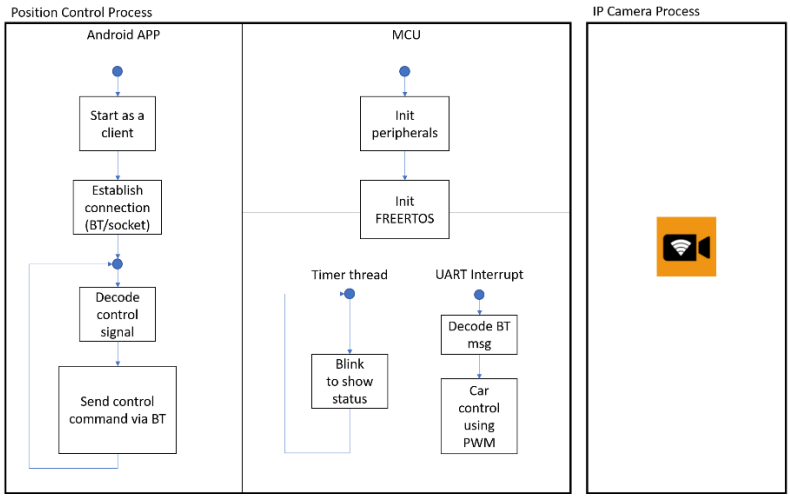


图 11　图像采集流程图

2. 服务器流程图

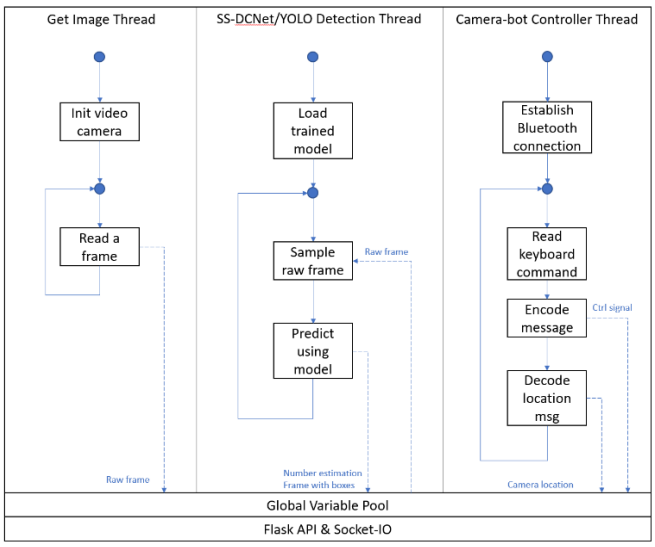


图 12 服务器流程图

七、外观设计

1. APP 界面设计

手机 APP 采用了扁平化简洁的设计风格，使用大量形象的图标代替文字和生硬的按钮，界面间切换使用平滑的过渡动画，并且图标会根据当前的连接状态给予反馈，建立连接的时候显示蓝色，断开连接的时候显示灰色。为使用人员提供了简洁易懂的界面与功能展示。

2. 小车硬件设计

作为移动平台的蓝牙小车。只具有非常少量的硬件设备，为兼顾成本与批量生产的能力，小车的外观非常简洁，核心的电路硬件都在小车内部，整车是透明的亚克力材质，便于观察内部各指示灯的情况。

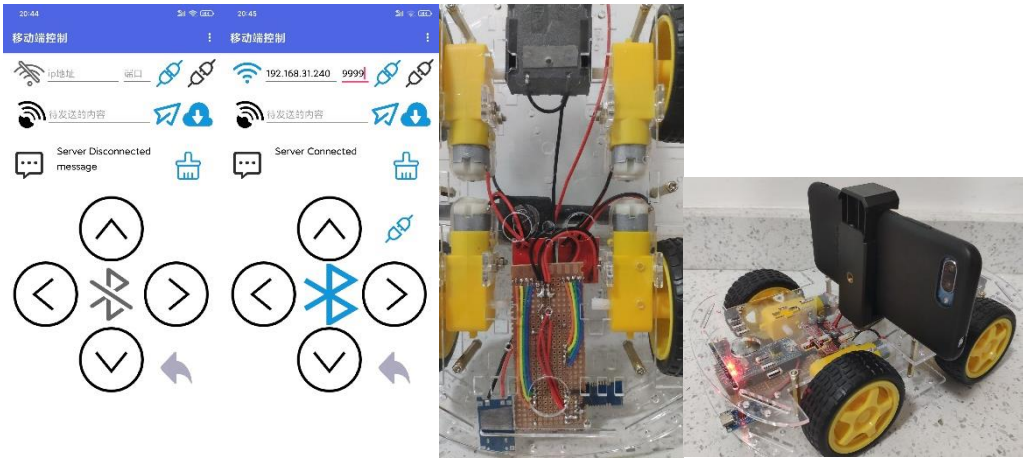


图 13 APP 界面与小车实物

3. 网站设计

在网页的地图搭建方案选择中，经过仔细比较百度地图开发平台与高德地图开发平台，由于百度地图的 API 接口相对后者更加齐全，网上相关资料也更加完善，本团队最终选择了百度地图开发平台来进行地图页面的搭建。

八、系统测试

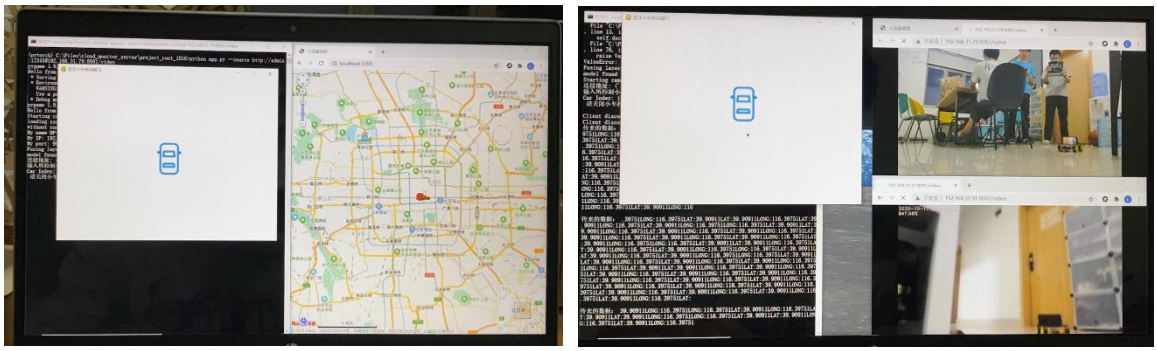


图 14 实际测试画面 1

测试效果如图，上面左图左侧为远程遥控监控节点的用户界面，左图右侧为提供给用户的人流量地图，在红色监控节点的位置显示了实时人数。右图为两个监控节点的情况，可以同时观察到两个监控节点的图像信息。

图 15 实际测试画面 2

上面左图为监控节点的实际运行情况，右图为人流量的监控情况。可以看到 AI 图像处理正常工作，把人框选出来。综上，本团队基于云端处理的可移动多节点人流量监控系统基本满足设计指标。

九、参考资料

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, You Only Look Once: Unified, Real-Time Object Detection, arXiv:1506.02640v5 [cs.CV] 9 May 2016

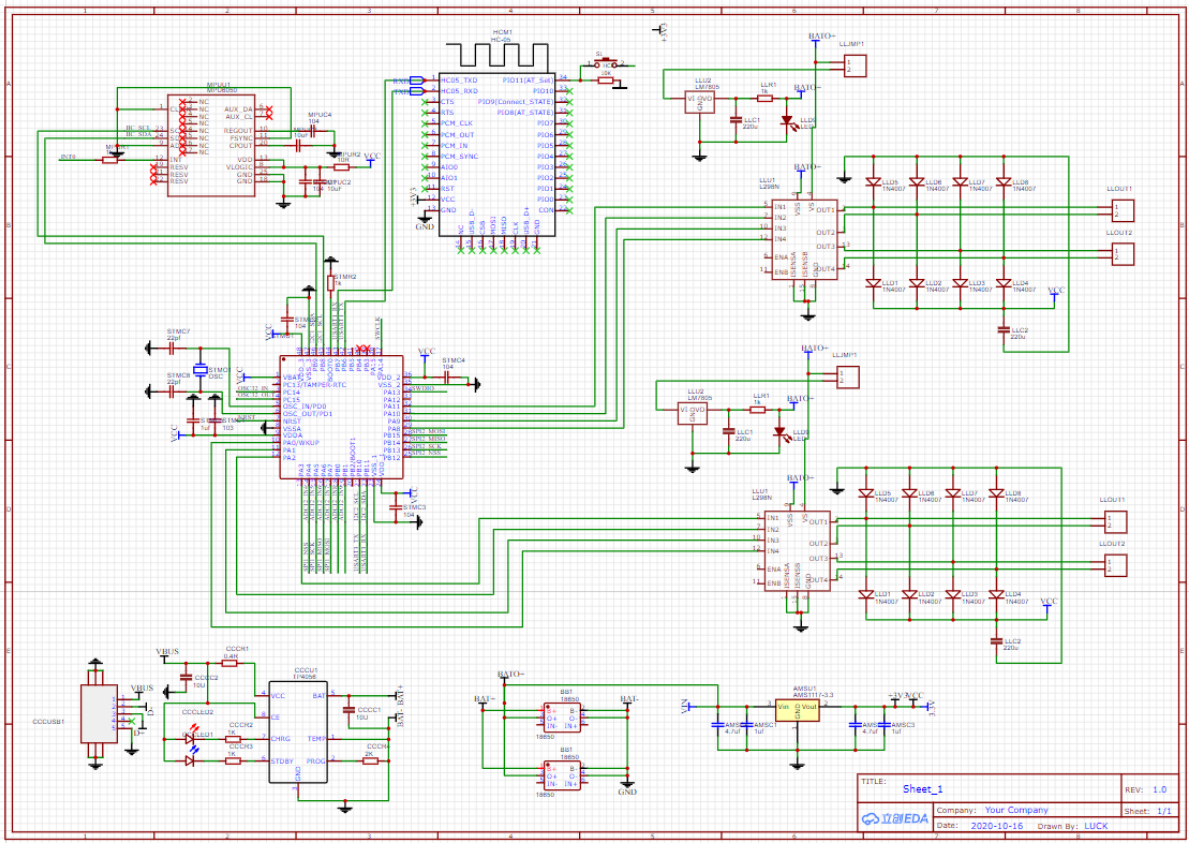
[2] Junyu Gao1, Wei Lin1, Bin Zhao1, Dong Wang1, Chenyu Gao1, Jun Wen, C^3 Framework: An Open-source PyTorch Code for Crowd Counting, arXiv:1907.02724v1 [cs.CV] 5 Jul 2019

[3] Haipeng Xiong, Hao Lu, Chengxin Liu, Liang Liu, Chunhua Shen, Zhiguo Cao, From Open Set to Closed Set: Supervised Spatial Divide-and-Conquer for Object Counting, arXiv:2001.01886v2 [cs.CV] 31 May 2020

[4] https://paperswithcode.com/sota/real-time-object-detection-on-coco

十、附录

1. 小车硬件电路图



2. 服务器端部分代码

```
#!/usr/bin/env python
import argparse
from importlib import import_module
import os
from flask import Flask, render_template, session, request, Response
import time
# import camera driver
from camera import Camera
import camera
# socket support
from flask_socketio import SocketIO, Namespace, emit, join_room, leave_room, \
    close_room, rooms, disconnect
import threading
import logging
import socket
import pygame
# ssdcnet
import predict
import cv2
# yolo
import detect
optSource = None
app = Flask(__name__)
thread1 = None
thread2 = None
thread3 = None
thread4 = None
thread5 = None
thread6 = None
latitude = 0
longitude = 0
yolo_count = 0
ssdc_count = 0
bluetooth_ready = False
mutex = threading.Lock()
socketio = SocketIO(app, async_mode=None)
```

```
app.logger.disabled = True
log = logging.getLogger('werkzeug')
log.disabled = True
def gen(camera):
    """Video streaming generator function."""
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
def yolo_process_thread():
    """process video and return result"""
    global yolo_count
    yolov5s = detect.init_yolo()
    while not camera.stream_started:
        time.sleep(1)
    while True:
        camera.boxes, yolo_count = detect.detect_frame(camera.imgSrc, yolov5s, conf=opt.conf_thres, iou=opt.iou)
def yolo_emit_thread():
    while not camera.stream_started:
        time.sleep(1)
    while True:
        socketio.emit('my_response',
                      {'data': yolo_count},
                      namespace='/test')
        time.sleep(1)
def loc_emit_thread():
    while not camera.stream_started:
        time.sleep(1)
    while True:
        socketio.emit('loc',
                      {'lat': latitude,
                      'lon':longitude},
                      namespace='/test')
        time.sleep(1)
def ssdc_process_emit_thread():
    """init ssdcnet and update crowd estimation"""
    global ssdc_count
    enable_cuda = opt.cuda
    ssdc = predict.init_ssdc(cuda=enable_cuda)
    while not camera.stream_started:
        time.sleep(1)
    while True:
        ssdc_count = predict.detect_crowd(
            img=camera.imgSrc, net=ssdc, cuda=enable_cuda)
        socketio.emit('my_response_',
                      {'data': ssdc_count},
                      namespace='/test')
def getImg_thread():
    while not bluetooth_ready:
        time.sleep(1)
    video = cv2.VideoCapture(optSource)
    while True:
        _, camera.imgSrc = video.read()
        # read current frame
        if camera.stream_started == False:
            camera.stream_started = True
def bluetooth_thread():
    global latitude
    global longitude
    global bluetooth_ready
    serversocket = socket.socket(
        socket.AF_INET, socket.SOCK_STREAM)  # 创建 socket 对象
    host = socket.gethostname()  # 获取本地主机名
    port = 9999
    serversocket.bind((host, port))  # 绑定端口号
    serversocket.listen(5)  # 设置最大连接数，超过后排队
    myname = socket.getfqdn(socket.gethostname())
    myaddr = socket.gethostbyname(myname)
    print("My name", myname, "\r\nMy IP:", myaddr, "\r\nMy port:", port)
    car_ctrl = False
    car_ctrl_msg = "hello"
    moving_directionx = 0
    moving_directiony = 0
    while True:
        clientsocket, addr = serversocket.accept()  # 建立客户端连接
        clientsocket.setblocking(0)             # 把接受数据的方式设置为非阻塞的
        print("连接地址: %s" % str(addr))
        car_ctrl = True
        msg = 'BlueToothCar Server Connected' + "\r\n"
        if not bluetooth_ready:
            bluetooth_ready = True
        clientsocket.send(msg.encode('utf-8'))
        car_index = input("输入所控制小车的序号: ")
        print("Car Index:", car_index, "\r\n", "请关闭小车控制的窗口再连接下一台蓝牙小车", "\r\n")
```

```
        pygame.init()
        screen = pygame.display.set_mode([640, 480])
        pygame.display.set_caption("蓝牙小车移动窗口")
        bg_color = [255, 255, 255]
        image = pygame.image.load("./车辆俯视图.png")
        rect = image.get_rect()
        screen_rect = screen.get_rect()
        rect.center = screen_rect.center
        while car_ctrl:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    car_ctrl = False
                    car_ctrl_msg = "zPPm"
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_w or event.key == pygame.K_UP:
                        moving_directiony = -1
                        car_ctrl_msg = "zW1m"
                    elif event.key == pygame.K_a or event.key == pygame.K_LEFT:
                        moving_directionx = -1
                        car_ctrl_msg = "zA1m"
                    elif event.key == pygame.K_s or event.key == pygame.K_DOWN:
                        moving_directiony = 1
                        car_ctrl_msg = "zS1m"
                    elif event.key == pygame.K_d or event.key == pygame.K_RIGHT:
                        moving_directionx = 1
                        car_ctrl_msg = "zD1m"
                    else:
                        print("按键不合法，请重新输入")
                elif event.type == pygame.KEYUP:
                    moving_directionx = 0
                    moving_directiony = 0
                    car_ctrl_msg = "zPPm"
            time.sleep(0.05)
            rect.centerx = screen_rect.centerx + moving_directionx * 30
            rect.centery = screen_rect.centery + moving_directiony * 30
            screen.fill(bg_color)
            screen.blit(image, rect)
            pygame.display.flip()
            # 检查一下是否断开了连接再发送
            try:
                clientsocket.send(car_ctrl_msg.encode('utf-8'))
            except(ConnectionAbortedError, ConnectionAbortedError, ConnectionResetError, TypeError):
                print("连接已断开", "\r\n")
                break
            # 接收手机端传来的数据
            try:
                msg_recv = clientsocket.recv(1024)
                if not msg_recv:
                    continue
                elif msg_recv:  # 有数据的时候
                    msg_recv_str = msg_recv.decode('utf-8')
                    if msg_recv_str.startswith('LAT:'):
                        try:
                            latitude = float(msg_recv_str[4:])  # 纬度
                        except(Exception):  # 有时候 socket 数据阻塞导致要处理一串“39.12323LONG116.1888”会出现错误
                            continue
                    elif msg_recv_str.startswith('LONG:'):
                        try:
                            longitude = float(msg_recv_str[5:]) # 经度
                        except(Exception):
                            continue
                    else:
                        print("传来的数据：", msg_recv_str, "\r\n")
            except(ConnectionAbortedError, ConnectionAbortedError, ConnectionResetError, TypeError, BlockingIOErr
or):
                continue
        pygame.quit()
        clientsocket.close()
@app.route('/video')
def index():
    """Video streaming home page."""
    return render_template('index.html', async_mode=socketio.async_mode)
    # return render_template('index.html')
@app.route('/')
def mapshow():
    """the people shows on the map page."""
    return render_template('mapshow.html', async_mode=socketio.async_mode)
    # return render_template('test1.html', async_mode=socketio.async_mode)
    # return render_template('mapshow.html')
@app.route('/video_feed')
def video_feed():
    """Video streaming route. Put this in the src attribute of an img tag."""
    return Response(gen(Camera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')
```

```
class MyNamespace(Namespace):
    def on_my_event(self, message):
        session['receive_count'] = session.get('receive_count', 0) + 1
        emit('my_response',
             {'data': message['data'], 'count': session['receive_count']})
    def on_my_broadcast_event(self, message):
        session['receive_count'] = session.get('receive_count', 0) + 1
        emit('my_response',
             {'data': message['data'], 'count': session['receive_count']},
             broadcast=True)
    def on_join(self, message):
        join_room(message['room'])
        session['receive_count'] = session.get('receive_count', 0) + 1
        emit('my_response',
             {'data': 'In rooms: ' + ', '.join(rooms()),
              'count': session['receive_count']})
    def on_leave(self, message):
        leave_room(message['room'])
        session['receive_count'] = session.get('receive_count', 0) + 1
        emit('my_response',
             {'data': 'In rooms: ' + ', '.join(rooms()),
              'count': session['receive_count']})
    def on_close_room(self, message):
        session['receive_count'] = session.get('receive_count', 0) + 1
        emit('my_response', {'data': 'Room ' + message['room'] + ' is closing.',
                             'count': session['receive_count']},
             room=message['room'])
        close_room(message['room'])
    def on_my_room_event(self, message):
        session['receive_count'] = session.get('receive_count', 0) + 1
        emit('my_response',
             {'data': message['data'], 'count': session['receive_count']},
             room=message['room'])
    def on_disconnect_request(self):
        session['receive_count'] = session.get('receive_count', 0) + 1
        emit('my_response',
             {'data': 'Disconnected!', 'count': session['receive_count']})
        disconnect()
    def on_my_ping(self):
        emit('my_pong')
    def on_connect(self):
        global thread1
        global thread2
        global thread3
        global thread4
        global thread5
        global thread6
        with mutex:
            if thread1 is None:
                thread1 = socketio.start_background_task(yolo_process_thread)
            if thread2 is None:
                thread2 = socketio.start_background_task(ssdc_process_emit_thread)
            if thread3 is None:
                thread3 = socketio.start_background_task(getImg_thread)
            if thread4 is None:
                thread4 = socketio.start_background_task(yolo_emit_thread)
            if thread5 is None:
                thread5 = socketio.start_background_task(bluetooth_thread)
            if thread6 is None:
                thread6 = socketio.start_background_task(loc_emit_thread)
        emit('my_response', {'data': 'Connected', 'count': 0})
    def on_disconnect(self):
        print('Client disconnected', request.sid)
socketio.on_namespace(MyNamespace('/test'))
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--source', type=str,
                        default='0', help='source')
    parser.add_argument('--conf-thres', type=float,
                        default=0.4, help='object confidence threshold')
    parser.add_argument('--cuda', type=bool,
                        default=False)
    parser.add_argument('--iou', type=float,
                        default=0.5, help='IOU threshold for NMS')
    parser.add_argument('--debug', type=bool,
                        default=False)
    opt = parser.parse_args()
    if opt.source == '0':
        optSource = 0
    else:
        optSource = opt.source
    if opt.debug:
        bluetooth_ready = True
    socketio.run(app, debug=True, host='0.0.0.0', port=5000)
```

3. 网页端部分代码

```
<!DOCTYPE HTML>
<html>
<head>
  <title>cloud_server</title>
  <script src="//code.jquery.com/jquery-1.12.4.min.js" integrity="sha256-
ZosEbRLbNQzLpnKIkEdrPv7lOy9C27hHQ+Xp8a4MxAQ="
    crossorigin="anonymous"></script>
  <script src="//cdnjs.cloudflare.com/ajax/libs/socket.io/2.2.0/socket.io.js"
    integrity="sha256-
yr4fRk/GU1ehYJPAs8P4JlTgu0Hdsp4ZKrx8bDEDC3I=" crossorigin="anonymous"></script>
  <style type="text/css">
    * {
      margin: 0px;
      padding: 0px;
    }
    .div_one {
      position: relative;
      margin: 0 auto;
      text-align: center;
    }
    .div_one p {
      font-size: 20px;
      text-align: center;
      line-height: 100px;
    }
    .div_two {
      position: relative;
      margin: 0 auto;
      text-align: center;
    }
  </style>
  <script type="text/javascript" charset="utf-8">
    $(document).ready(function () {
      namespace = '/test';
      var socket = io(namespace);
      socket.on('connect', function () {
        socket.emit('my_event', { data: 'I\'m connected!' });
      });
      socket.on('my_response', function (msg, cb) {
        $('#log').text(msg.data);
        if (cb)
          cb();
      });
      socket.on('my_response_', function (msg, cb) {
        $('#log_').text(msg.data);
        if (cb)
          cb();
      });
      socket.on('loc', function (msg, cb) {
        $('#lat').text(msg.lat);
        $('#lon').text(msg.lon);
        if (cb)
          cb();
      });
      var ping_pong_times = [];
      var start_time;
      window.setInterval(function () {
        start_time = (new Date).getTime();
        socket.emit('my_ping');
      }, 1000);
      socket.on('my_pong', function () {
        var latency = (new Date).getTime() - start_time;
        ping_pong_times.push(latency);
        ping_pong_times = ping_pong_times.slice(-30); // keep last 30 samples
        var sum = 0;
        for (var i = 0; i < ping_pong_times.length; i++)
          sum += ping_pong_times[i];
        $('#ping-pong').text(Math.round(10 * sum / ping_pong_times.length) / 10);
```

```
      });
    });
  </script>
</head>
<body>
  <h1 class="div_one">Video stream</h1>
  <div class="div_two">
    <img src="{{ url_for('video_feed') }}">
  </div>
  <p></p>
  <p class="div_one">Average latency: <b><span id="ping-pong"></span>ms</b></p>
  <p class="div_one">YOLO 算法检测人数为：<b><span id="log"></span> 人</b></p>
  <p class="div_one">SSDC 算法检测人数为：<b><span id="log_"></span> 人</b></p>
  <p class="div_one">经度位置为：<b><span id="lon"></span></b></p>
  <p class="div_one">纬度位置为：<b><span id="lat"></span></b></p>
</body>
</html>
```

4. 小车端嵌入式部分代码

```
/* Includes ------------------------------------------------------------------*/
#include "FreeRTOS.h"
#include "task.h"
#include "main.h"
#include "cmsis_os.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "motor_driver.h"
#include "bluetooth.h"
#include "mpu6050.h"
#include "sd_hal_mpu6050.h"
#include "communicate.h"
/* USER CODE END Variables */
osThreadId LoopTaskHandle;
osThreadId MPU6050TaskHandle;
osTimerId LowSpeedTimerHandle;
osTimerId HighSpeedTimerHandle;
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN FunctionPrototypes */
/* USER CODE END FunctionPrototypes */
void StartLoopTask(void const * argument);
void StartMPU6050Task(void const * argument);
void LowSpeedTimerCallback(void const * argument);
void HighSpeedTimerCallback(void const * argument);
void MX_FREERTOS_Init(void); /* (MISRA C 2004 rule 8.1) */
/* GetIdleTaskMemory prototype (linked to static allocation support) */
void vApplicationGetIdleTaskMemory( StaticTask_t **ppxIdleTaskTCBBuffer, StackType_t **ppxIdleTaskStackBuffer, uint3
2_t *pulIdleTaskStackSize );
/* GetTimerTaskMemory prototype (linked to static allocation support) */
void vApplicationGetTimerTaskMemory( StaticTask_t **ppxTimerTaskTCBBuffer, StackType_t **ppxTimerTaskStackBuffer, ui
nt32_t *pulTimerTaskStackSize );
/* USER CODE BEGIN GET_IDLE_TASK_MEMORY */
static StaticTask_t xIdleTaskTCBBuffer;
static StackType_t xIdleStack[configMINIMAL_STACK_SIZE];
void vApplicationGetIdleTaskMemory( StaticTask_t **ppxIdleTaskTCBBuffer, StackType_t **ppxIdleTaskStackBuffer, uint3
2_t *pulIdleTaskStackSize )
{
  *ppxIdleTaskTCBBuffer = &xIdleTaskTCBBuffer;
  *ppxIdleTaskStackBuffer = &xIdleStack[0];
  *pulIdleTaskStackSize = configMINIMAL_STACK_SIZE;
  /* place for user code */
}
/* USER CODE END GET_IDLE_TASK_MEMORY */
/* USER CODE BEGIN GET_TIMER_TASK_MEMORY */
static StaticTask_t xTimerTaskTCBBuffer;
static StackType_t xTimerStack[configTIMER_TASK_STACK_DEPTH];
void vApplicationGetTimerTaskMemory( StaticTask_t **ppxTimerTaskTCBBuffer, StackType_t **ppxTimerTaskStackBuffer, ui
nt32_t *pulTimerTaskStackSize )
{
  *ppxTimerTaskTCBBuffer = &xTimerTaskTCBBuffer;
  *ppxTimerTaskStackBuffer = &xTimerStack[0];
  *pulTimerTaskStackSize = configTIMER_TASK_STACK_DEPTH;
  /* place for user code */
}
/* USER CODE END GET_TIMER_TASK_MEMORY */
/**
  * @brief  FreeRTOS initialization
  * @param  None
  * @retval None
  */
void MX_FREERTOS_Init(void) {
```

```
  /* Create the timer(s) */
  /* definition and creation of LowSpeedTimer */
  osTimerDef(LowSpeedTimer, LowSpeedTimerCallback);
  LowSpeedTimerHandle = osTimerCreate(osTimer(LowSpeedTimer), osTimerPeriodic, NULL);
  /* definition and creation of HighSpeedTimer */
  osTimerDef(HighSpeedTimer, HighSpeedTimerCallback);
  HighSpeedTimerHandle = osTimerCreate(osTimer(HighSpeedTimer), osTimerPeriodic, NULL);
  /* USER CODE BEGIN RTOS_TIMERS */
  /* start timers, add new ones, ... */
  osTimerStart(LowSpeedTimerHandle,200);
  osTimerStart(HighSpeedTimerHandle,50);
  /* USER CODE END RTOS_TIMERS */
  /* USER CODE BEGIN RTOS_QUEUES */
  /* add queues, ... */
  /* USER CODE END RTOS_QUEUES */
  /* Create the thread(s) */
  /* definition and creation of LoopTask */
  osThreadDef(LoopTask, StartLoopTask, osPriorityNormal, 0, 128);
  LoopTaskHandle = osThreadCreate(osThread(LoopTask), NULL);
  /* definition and creation of MPU6050Task */
  osThreadDef(MPU6050Task, StartMPU6050Task, osPriorityIdle, 0, 128);
  MPU6050TaskHandle = osThreadCreate(osThread(MPU6050Task), NULL);
  /* USER CODE BEGIN RTOS_THREADS */
  /* add threads, ... */
  /* USER CODE END RTOS_THREADS */
}
/* USER CODE BEGIN Header_StartLoopTask */
/**
  * @brief  Function implementing the LoopTask thread.
  * @param  argument: Not used
  * @retval None
  */
/* USER CODE END Header_StartLoopTask */
void StartLoopTask(void const * argument)
{
  /* USER CODE BEGIN StartLoopTask */
  /* Infinite loop */
  for(;;)
  {
    HAL_GPIO_TogglePin(GPIOC,GPIO_PIN_13);
    osDelay(100);

  }
  /* USER CODE END StartLoopTask */
}
/* USER CODE BEGIN Header_StartMPU6050Task */
/**
* @brief Function implementing the MPU6050Task thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_StartMPU6050Task */
void StartMPU6050Task(void const * argument)
{
  /* USER CODE BEGIN StartMPU6050Task */
  /* Infinite loop */
  SD_MPU6050_Result result ;
//  result = SD_MPU6050_Init(&hi2c1,&mpu1,SD_MPU6050_Device_0,SD_MPU6050_Accelerometer_2G,SD_MPU6050_Gyroscope_250s
);
  HAL_Delay(500);
//  if(result == SD_MPU6050_Result_Ok);
  for(;;)
  {
    osDelay(100);
  }
  /* USER CODE END StartMPU6050Task */
}

/* LowSpeedTimerCallback function */
void LowSpeedTimerCallback(void const * argument)
{
  /* USER CODE BEGIN LowSpeedTimerCallback */
//  HAL_GPIO_TogglePin(GPIOC,GPIO_PIN_13);
  /* USER CODE END LowSpeedTimerCallback */
}

/* HighSpeedTimerCallback function */
void HighSpeedTimerCallback(void const * argument)
{
  /* USER CODE BEGIN HighSpeedTimerCallback */
  char move_order = data_rx.rx[1];
  int speed_set = 0;
  if(data_rx.rx[2] == '1') speed_set = 1000;
  switch(move_order)
  {
    case 'W':
```

```
      move_forward(speed_set);
      break;
    case 'S':
      move_backward(speed_set);
      break;
    case 'A':
      turn_left(speed_set);
      break;
    case 'D':
      turn_right(speed_set);
      break;
    case 'P':
      motor_stop();
      break;
    default:
      break;
  }
  /* USER CODE END HighSpeedTimerCallback */
}

/* Private application code --------------------------------------------------*/
/* USER CODE BEGIN Application */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  COMM_RxCpltCallback(huart);
}
/* USER CODE END Application */

struct Data_TX_t data_tx;
struct Data_RX_t data_rx;
uint8_t rx_buf = 0;
void COMM_init(void)
{
  HAL_UART_Receive_IT(&comm_huart,data_rx.rx,sizeof(data_rx.rx));
  data_tx.start_flag = START_FLAG;
  data_tx.end_flag = END_FLAG;
}
void COMM_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(huart == &comm_huart)
  {
    HAL_UART_Receive_IT(&comm_huart,&rx_buf,1);
    COMM_ReceiveHandle();
  }
}

void COMM_ReceiveHandle(void)
{
    static int data_count = 0;
    data_rx.rx[data_count] = rx_buf;
    data_count++;
    if(data_rx.rx[0] != START_FLAG)
    {
        data_count = 0;
        return;
    }
    if(data_count < COMM_LEN) return;
    if(data_rx.rx[COMM_LEN - 1] != END_FLAG)
    {
        data_count = 0;
        return;
    }
}
```