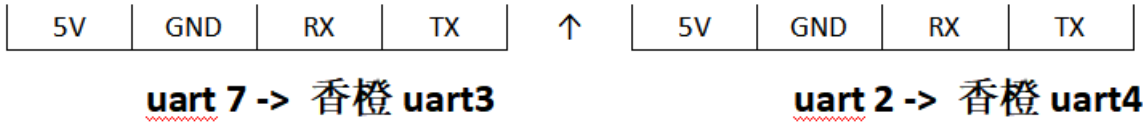
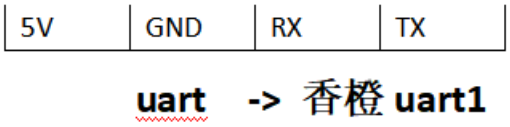
主要控制文件介绍：

飞控板接线：



激光测距：



飞控板：

Drivers/drv_Uart2.c
接收 orangepi uart4发来的消息：1.路径列表，2.动物识别信息
1. uav_car\navigator.py节点:fly_path_callback():ur.uart_sender4_send(self, msg) **路径**发布
2. \uav_car\image_processor.py节点：
sub_camera_callback():self.send_location(msg) 识别到动物发坐标给飞控板？/地面站？ msg = ur.t265_data_to_uart_std(self, "98", "35", data)
ur.uart_sender4_send(self, msg)
3. uav_car\image_processor.py节点中 回调函数sub_t265_loc_callback包含 发送给飞控板的信息： msg = "99 "？为了扫描动物的特殊指令？

路径格式：

- 在 uav_car\uav_car\navigator.py: fly_path_callback()末尾有介绍，例如msg='@1:<长度>;x1,y1;x2,y2;.....;#'
- 后续格式变化：
  - 先 msg = " ".join(format(ord(d), '02X') for d in str(msg))
  - 再 ur.uart_sender4_send(self, msg)
  -

Drivers/drv_Uart7.c
接收 orangepi uart3发来的消息：1. 起飞讯号 2. xyz水平+高度数据

1. UAV_CAR\src\uav_car\uav_car\navigator.py:
takeoff_callback():ur.uart_sender3_send(self, msg) 起飞指令：'52<空格>'
2. UAV_CAR\src\uav_car\uav_car\t265.py: service_t265_open_callback()->timer_pub_t265_callback(): ur.uart_sender3_send(self, t265_date) t265+定高信息：

Drivers/drv_main.c
硬件初始化汇总

Modes/M37.c
起飞，
起飞之后的飞控逻辑

飞行端香橙派：

发送消息，全靠 service 和 topic!

飞行端主要文件即为 UAV_CAR\src\uav_car\uav_car\的py节点文件， 被 launch脚本一起启动。

t265_data_topic： T265Data 航迹信息, 转发给地面站（不转发给飞控板！），同时让飞行端的sub_t265_loc每次回调进行动物扫描判断（会发送给飞控板 特殊指令）

不同于上述过程，UAV_CAR\src\uav_car\uav_car\t265.py 节点创建了一个**t265_data_topic**的话题，发布的数据为**T265Data**。

```
pub_data = self.create_publisher(**T265Data**, name + '_data_topic', topic_stack)，这里 name='t265'
....

    if msg.confidence == 3:
        self.pub_data.publish(msg)
        # t265数据 发给 t265_data_topic 给地面端订阅!
        # t265数据 发给 t265_data_topic 给地面端订阅!

        # 分割！！！--------------------
        # 分割！！！--------------------

        # 之下的是 转化 t265数据，为了给uart3发送个 飞控板做准备
```

```
        t265_date = ur.t265_data_to_uart_std(self, "58", "35", msg)
        # 数据预处理 t265数据 转换
        # 数据预处理 t265数据 转换
        ur.uart_sender3_send(self, t265_date)
        # t265数据 发给 uart_sender3_data_topic
        # t265数据 发给 uart_sender3_data_topic
        # 该话题上的数据，最后通过 uart_sender3_data_topic话题 订阅者uar3
的节点UartSender(Node)的订阅回调，发送给 开发板，下面有记录。
        # uart3： uart_sender3_data_topic 转发 水平和定高xyz 给飞控板！
        # uart3： uart_sender3_data_topic 转发 起飞指令
```

t265_data_topic话题有几个作用：

1. \uav_car\image_processor.py节点的sub_t265_loc 订阅 t265+激光定高数据，也即 **T265Data**
2. 同样，**地面站**也订阅**t265_data_topic**这个话题，更新显示屏上的飞机飞行轨迹
3. uav_car\image_processor.py节点的 **sub_t265_loc**订阅**t265_data_topic**，它的回调函数sub_t265_loc_callback **每次T265Data数据发布，都会进行 更新当前位置、判断是否到达目标点并触发返航、控制激光扫描动物及发送结果。**
4. 回调函数sub_t265_loc_callback包含 发送给飞控板的信息：

```
msg = "99 "
ur.uart_sender4_send(self, msg)
```

---

uart3： uart_sender3_data_topic 转发 水平和定高xyz 给飞控板！

- UAV_CAR\src\uav_car\uav_car\t265.py定义了 **t265_date**

拿 **香橙派飞行端 uart3发送 xyz数据**来说：**uart_sender3_data_topic**这个话题作为中转站
1. navigator.py节点调用ur.uart_sender3_init()创建 publisher发布者，同时也创建了**uart_sender3_data_topic**这个话题
2. **timer_pub_t265定时器**，周期性使用ur.uart_sender3_send()函数(未作处理，字符串发送)发布**t265传感器的t265_date**数据到 话题**uart_sender3_data_topic**

3. **uart_sender3_data_topic**的订阅端，节点**uart_sender3.py**也在并行运行，被 **节点uart_sender3.py**订阅到， 然后用**UartSender类**的回调函数send_uart_data() 借助串口库，发送给**飞控板**
4. 总结： 并行创建 串口数据的发布者+订阅者， 周期发布传感器数据到话题 **uart_sender3_data_topic**，订阅者读话题，然后 从UartSender的订阅回调，串口写数据给**飞控板**！

调用串口**serial**方法发送给**飞控板子**数据：

```
uav_car_unit\uart_sender.py：uart_sender结构如下：uart_sender结构如下：
class UartSender(Node):
    '''飞行端订阅者： 订阅飞行端自己的数据发布者，然后回调转发数据给串口'''

    def __init__(self, name, serial_port, baudrate, topic_stack=50):

        super().__init__(name) # FIXME 串口基类：波特率 串口号
        self.name = name
        self.topic_stack = topic_stack
        self.sub = self.create_subscription(String, name +
"_data_topic", self.send_uart_data, topic_stack) # FIXME 串口基类：订阅
者回调！！！！！
        self.serial_port = serial_port
        self.baudrate = baudrate
        self.sink_serial = serial.Serial(self.serial_port,
self.baudrate)
        self.get_logger().warning(f"init success")

    def send_uart_data(self, msg):
        self.get_logger().info(f"request to send:{msg.data}")
        if self.sink_serial.isOpen():
            hex_msg = bytes.fromhex(msg.data) # FIXME 串口基类：hex转
bit
            send_count = self.sink_serial.write(hex_msg)
            if send_count == len(hex_msg): return True
            else:                          return False
```

uart3: uart_sender3_data_topic 转发 起飞指令

- **飞行端**创建在\uav_car\navigator.py节点创建**服务** takeoff_service = self.create_service(Empty, "takeoff_service", self.**takeoff_callback**)，

takeoff_callback回调时，发送 起飞指令

- **地面站**作为 takeoff_service服务的 客户端， 点击**一键起飞**回调 call_takeoff_service，唤起**飞行端takeoff_service服务的回调**
- 飞行端takeoff_callback如下：

```
    def takeoff_callback(self, request, response):  # FIXME 回调：起飞
        self.get_logger().warning("takeoff service called")
        # wait for camera and t265 to be ready
        msg = "52 "
        ur.uart_sender3_send(self, msg)
        return response
```

- 起飞指令：'52<空格>'，
- 送入uart_sender3_send -> 发布到话题**uart_sender3_data_topic**，
- 被 **节点uart_sender3.py**订阅到， 然后用**UartSender类的回调**

uart4: uart_sender4_data_topic

- 1. 转发动物位置信息，主要是uav_car\image_processor.py节点下的 **send_location(self, data)**， 以及 重置状态 这个指令msg = "99<空格>“ ur.uart_sender4_send(self, msg)
- 2. 转发所有的规划路径： 飞行端 \uav_car\navigator.py节点的 **fly_path_callback**
- 补充：地面站**发布全部飞行路径节点**的话题 **fly_path_topic**， 由飞行端 navigator.py节点的**fly_path_sub**订阅，即为self.fly_path_sub = self.create_subscription(Float64MultiArray, "fly_path_topic", self.**fly_path_callback**, 5)， fly_path_callback为 飞行端fly_path_sub的回调函数。
- 补充：**转发所有的规划路径的 原始数据组装**

```
        # 将一维列表转换为二维列表，每两个元素为一组
        self.fly_path = [self.fly_path[i:i + 2] for i in range(0,
len(self.fly_path), 2)]
        # self.fly_path = [self.fly_path[i:i + 2] for i in range(0, 3,
```

```
2)]
        fly_path_len = len(self.fly_path)
        msg = "@1:" + str(fly_path_len)
        for i in range(fly_path_len):
            msg += ";" + str(self.fly_path[i][0]) + "," +
str(self.fly_path[i][1])
        msg += "#"
        msg = " ".join(format(ord(d), '02X') for d in str(msg))
        # send the fly path to uart sender4
        ur.uart_sender4_send(self, msg)   # FIXME 回调：路径发布
```

摄像头图像传递： camera_data_topic（不是UDP，是摄像头图像，传给rknn识别程序）

+ 位于飞行端 uav_car_unit\camera.py 节点的话题**camera_data_topic** 发布者 pub_camera（不是UDP话题发布者），负责读取摄像头信息。

```
uav_car\image_processor.py节点下的**sub_camera**订阅者摄像头图像，回调
**sub_camera_callback**，进行rknn图像处理 ：
1. 进行动物识别
        -> **animals_recognition**调用rknn，输入订阅的图像 识别动物
        -> 识别到后用self.**send_location**(msg)发送动物位置，调用
        -> 调用 **pub_image_processor**.publish(animal_msg)发送动物识别
信息到 **camera_data_topic**
2. 如果有识别后的图像， 同样会被**udp_sender**.send_frame(draw_image,
stream_id=1) 转发给UDP端口！
```

```
send_location(self, data)的 data组成：
#self.stop_x, self.stop_y = self.get_grid_center(self.now_position[0],
#self.now_position[1])
#msg = T265Data()
#msg.pos_x = self.stop_x
#msg.pos_y = self.stop_y
#msg.pos_z = 1.0
#self.send_location(msg)
#self.get_logger().info(f"Animal found at position: {animal_position},
stopping at ({self.stop_x}, {self.stop_y})")
```

```
def send_location(self, data):
        # config header to send to shufly!
        # data.header = "99"
        # self.pub_t265_uart_send_pos.publish(data)
        msg = ur.t265_data_to_uart_std(self, "98", "35", data)
        ur.uart_sender4_send(self, msg)
        # 发送逻辑同 uart3
        # 发送逻辑同 uart3
        # 发送逻辑同 uart3
        self.get_logger().warning(f'send location data success')
```

UDP图传： udp_info_topic 仅限香橙派之间

- 位于 **地面站！的** 话题**udp_info_topic**， 地面站也创建了它的发布者**udp_info_publisher**，主要转发 ip地址信息
- 而**飞行端**在uav_car\image_processor.py节点创建了订阅者**sub_udp_info**， 接收地面站发来的 ip地址，并在**udp_info_callback**解析，创建**udp_sender**利用UDP互传图像。
- 在rknn识别后，调用udp_sender.send_frame(draw_image, stream_id=1) 传输图像

动物识别信息发布：image_processor_topic

- 位于飞行端 uav_car\image_processor.py节点的话题**image_processor_topic**，
- 有**pub_image_processor** 发布ImageProcessorData动物识别结果，然后被**地面站animal_sub**订阅。