USING RYZE TELLO UAVS IN MULTI-ROBOT SYSTEMS

Matouš Hýbl
Master Degree Programme (2nd year), FEEC BUT
E-mail: xhyblm00@stud.feec.vutbr.cz

Supervised by: Petr Gábrlík
E-mail: xgabrl00@stud.feec.vutbr.cz

Abstract: This paper describes ways of integrating Ryze Tello UAVs (Unmanned Aerial Vehicles) with existing robotic systems. The main focus of this paper is the creation of bridging software and hardware that allows connecting of multiple drones to the robotic system while allowing for both video feed streaming and UAV control. The secondary focus of this paper is a proof of usability of the Rust programming language in robotic applications.

Keywords: unmanned aerial vehicle, multi-robot system, UAV, ROS, Rust

1 INTRODUCTION AND MOTIVATION

Ryze Tello UAVs (Unmanned Aerial Vehicles) (shown in the figure 1) are cheap and small UAVs that are originally aimed at students to study programming or being simply a toy. On the other hand the manufacturer provides a simple API (Application Programming Interface) based on UDP (User Datagram Protocol) datagrams that can be used to control the behavior of the UAV as well as reading some status information from it. Alongside with an on-board camera with video streaming capabilities, this makes the drone a good candidate for a research device, especially for UAV swarm research (an example of such swarm is shown in the figure 2) or research in UAV-UGV (Unmanned Ground Vehicle) cooperation.

There are two versions of the UAV - the Tello and Tello EDU, the latter one being more suitable for introduction to programming as it features a kit that can be used for simple visual robot navigation. The Tello can be controlled wirelessly using a mobile device via Wi-Fi, where it acts as an access point [1]. On the other hand, the Tello EDU allows for being connected to an external access point as a standard Wi-Fi station. This seems promising as it would allow for connecting multiple UAVs to a single access point, therefore allowing for easy implementation of robotic swarms. The problem with this configuration is that the video feed is not accessible in station mode, which disables one of the main advantages of the UAV - the real-time video stream. That means that then it cannot be used for visual navigation or SLAM (Simultaneous Localization and Mapping) and for example for reconnaissance with UGV-cooperation.

This paper aims at removing this problem by creating bridging hardware and software which doesn't require the drone to be switched to the station mode, therefore also eliminating the need to buy the more expensive EDU version.

Figure 1: Ryze Tello UAV [2].

Figure 2: An example of an UAV swarm equipped with lights [3].

2 METHODOLOGY

This section describes the hardware and software used and developed as well as different approaches of doing so, as the bridging software can be implemented using at least three different ways.

2.1 HARDWARE

Hardware used for the software bridge needs to meet following conditions:

- have a working Wi-Fi and ethernet network interface,
- have enough computing power to allow for video encoding and streaming.

For simplicity of implementation, a Raspberry Pi single-board computer was used, more specifically version 3B which was the first one to have a Wi-Fi interface and also enough computing power.

During development, it was also discovered that the UAV is not designed to withstand prolonged periods of being placed on a table with no airflow which results in overheating leading to frequent shutdowns. A simple PC case fan can be placed below the UAV, therefore mitigating the problem. In the future development integrated charging could also be implemented.

2.2 SOFTWARE

There are at least three approaches to developing the bridging software. These approaches differ only in the ways of controlling the UAV, video streaming implementation remains the same for all of them. The approaches are:

- direct retransmission of control and status datagrams,
- control and status datagram translation,
- using the ROS node to control the UAV.

2.2.1 DIRECT RETRANSMISSION OF CONTROL AND STATUS DATAGRAMS

Direct retransmission of control and status datagrams is the most straightforward way of controlling the UAV. The principle is simple, the bridging computer opens the same ports on its ethernet interface as the ports that are opened on the UAV, where it receives the control data originally meant for the

UAV and sends them to it. Then it opens a port on which the UAV sends its status data to and sends the status data back to the control computer. With this configuration, the program on the bridging computer sends whatever data is received from the control computer directly to the UAV and when there are status data available from the UAV, they are sent to the control computer. The whole process is schematically shown in the figure 3. And an example of code in the Rust programming language is shown in the listing 1.

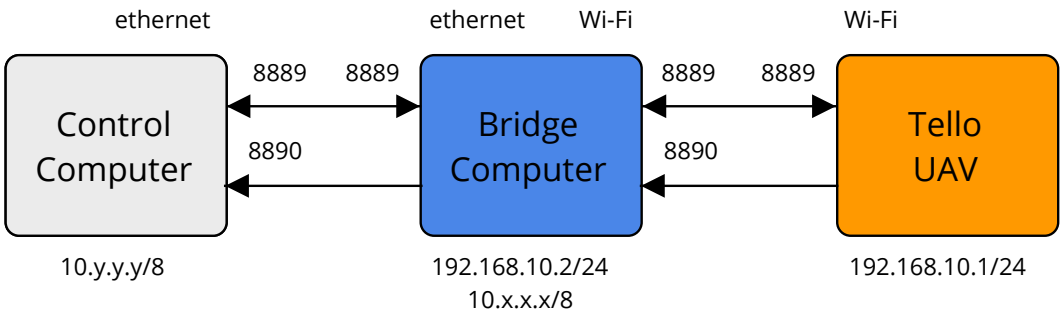

Figure 3: A simple schematic diagram depicting direct retransmission bridge.

```rust
let tello_control_socket = UdpSocket::bind("192.168.10.2:9009").unwrap();
let tello_state_socket = UdpSocket::bind("192.168.10.2:8890").unwrap();
let control_socket = UdpSocket::bind("0.0.0.0:8889").unwrap();
loop {
    let mut buffer: [u8; 1500] = [0; 1500];
    match control_socket.recv_from(&mut buffer) {
        Ok((size, address)) => {
            controller_address = address.ip().to_string();
            tello_control_socket.send_to(&buffer[0..size], "192.168.10.1:8889");
        }
        Err(_) => {}
    }
    match tello_control_socket.recv_from(&mut buffer) {
        Ok((size, address)) => {
            control_socket.send_to(&buffer[0..size], format!("{}:8889", &
                controller_address));
        }
        Err(_) => {}
    }
    match tello_state_socket.recv_from(&mut buffer) {
        Ok((size, address)) => {
            control_socket.send_to(&buffer[0..size], format!("{}:8890", &
                controller_address));
        }
        Err(_) => {}
    }
    std::thread::sleep(Duration::from_millis(10));
}
```

Listing 1: Implementation of the direct retransmission approach.

This approach was tested and seemed to work quite well. The advantage of this approach is that the existing libraries and tools developed to control the UAV can be used, just by changing the IP addresses and the way video is streamed. The disadvantage is that this approach still requires the maintenance of custom software and changes to video streaming.

2.2.2 CONTROL AND STATUS DATAGRAM TRANSLATION

The second approach is similar to the first one, but with the difference that a custom control and status data model is employed. This means that the designer of the robotic system has complete control over the data being sent to the bridging computer the UAV. There is also room for adding, for example, some control logic to the bridge, such as autoland safety features that are not dependent on the control program. This approach was tested and seemed to work quite well. The advantages of this approach are clear - custom functions can be implemented and a standardized communication interface can be used. The disadvantages remain the same as with the previous approach - maintenance of custom software is required.

2.2.3 USING ROS NODE TO CONTROL THE UAV

The last approach is especially suitable for ROS (Robot Operating System) based robotic systems. As ROS is a distributed system a ROS node used to control the UAV can be run on the bridging computer. This is the easiest approach for all ROS enabled robotic systems as nodes for the Tello UAV are already prepared and ready for immediate use [4, 5, 6]. These nodes could also publish images which result in handling the live video stream in a very clean way.

2.2.4 VIDEO STREAMING

Retransmission of the video can be implemented using various audio/video streaming utilities, such as FFmpeg or GStreamer. In this case, GStreamer was used. The goal was to set-up an RTSP (Real-Time Streaming Protocol) streaming server serving the UAV video stream. The GStreamer pipeline was adopted and modified from a tutorial [7], while the RTSP server code in Rust was adopted from GStreamer-rs repository [8]. Based on some experiments, the server should be launched only after the UAV is commanded to begin streaming and with some delay.

3 RESULTS

3.1 BRIDGING DRONE CONTROL, STATUS, AND VIDEO STREAM

In the last section, three different approaches to implement bridging algorithms were described. The first two were tested as a part of this paper and the preliminary testing showed that they work reasonably well. The ROS based approach was not directly tested, however it is believed to be thoroughly tested by other ROS users as well as students of the DCI FEEC Robotics and AI research group [6].

3.2 DEVELOPING BRIDGING SOFTWARE IN RUST

Another part of this paper was testing the suitability of the Rust programming language for robotic applications. During the work on this paper, an implementation of the Tello SDK (Software Development Kit) in Rust was developed as well as both of the bridge implementations. The area where Rust language stood out was memory safety in multi-threaded applications - the controller was checking for data races, etc. Another strong suit of the language is interoperability with C which allowed the use of the GStreamer Rust bindings, therefore allowing us to use an existing high-level library instead of developing a custom one. A feature of the language that was also used was its fully-featured standard library which conveniently out of the box supports, for example, UDP sockets. The disadvantage of the language was lack of remote development tooling which meant that the bridging code couldn't be run and debugged directly in the bridging computer.

4 CONCLUSIONS & FUTURE WORK

The aim of this paper was to explore approaches of integrating Tello UAVs into various multi-robot systems and to test suitability of Rust programming language for robotic applications.

The paper describes three approaches of integration of the UAVs with their advantages and disadvantages. For easy testing, the first one seems to be an optimal solution as it simply retransmits all communication. The other two approaches are suitable for integrating into an existing robotic system - whether it uses ROS or a custom communication protocol.

As for the suitability of using Rust for robotic applications, so far the safety features have proven useful in avoiding data races. C language interoperability was also successfully demonstrated and tested by integrating GStreamer bindings library. Rust language's standard library which already contains prepared implementations of many useful features such as UDP sockets etc. is extremely useful for jumpstarting software development.

In the future, the UAV will be implemented into the ATEROS robotic system and used for experimenting with indoor navigation and robotic swarms. As for the Rust programming language, it will be further evaluated - more specifically in areas such as direct hardware interfaces, tools, and ROS nodes.

5 ACKNOWLEDGEMENTS

The completion of this paper was made possible by the grant No. FEKT-S-20-6205 - "Research in Automation, Cybernetics and Artificial Intelligence within Industry 4.0" financially supported by the Internal science fund of Brno University of Technology.

REFERENCES

[1] Tello SDK Documentation EN 1.3 1122. In: *DJI CDN* [online]. Shenzen, China: SZ DJI Technology Co., 2018 [ref. 2020-03-02]. Available at: https://terra-1-g.djicdn.com/2d4dce68897a46b19fc717f3576b7c6a/Tello%20%E7%BC%96%E7%A8%8B%E7%9B%B8%E5%85%B3/For%20Tello/Tello%20SDK%20Documentation%20EN_1.3_1122.pdf

[2] DJI RYZE Tello. In: *Robot Advance* [online]. Mornant, France: Robot Advance, 2019 [cit. 2020-03-02]. Available at: https://www.robot-advance.com/EN/ori-drone-dji-ryze-tello-2609.jpg

[3] A Swarm of 800 Drones Create a Giant Airplane in the Sky. In: *Interesting Engineering* [online]. San Francisco, California, USA: Interesting Engineering, 2019 [ref. 2020-03-02]. Available at: https://interestingengineering.com/a-swarm-of-800-drones-create-a-giant-airplane-in-the-sky

[4] Tello_driver. In: *ROS Wiki* [online]. Mountain View, California, USA: Open Robotics Foundation, 2019 [ref. 2020-03-02]. Available at: http://wiki.ros.org/tello_driver

[5] Flock: ROS driver for DJI Tello drones. In: *Github* [online]. San Francisco, California, USA: Github, 2018 [ref. 2020-03-02]. Available at: https://github.com/clydemcqueen/flock

[6] PANSKÝ, Michal. *Autonomous Unmanned Aircraft Tello*. Brno, 2020. Semestral thesis. Brno University of Technology. Thesis supervisor Ing. Petr Gábrlík.

[7] YOUNG, Neil. How to make a Raspberry Pi an RTSP streamer and how to consume this? In: *Github Gist* [online]. San Francisco, California, USA: Github, 2020 [ref. 2020-03-02]. Available at: `https://gist.github.com/neilyoung/8216c6cf0c7b69e25a152fde1c022a5d`

[8] Rtsp-server.rs. In: *Gitlab Freedesktop.org* [online]. Freedesktop.org, 2019 [ref. 2020-03-02]. Available at: `https://gitlab.freedesktop.org/gstreamer/gstreamer-rs/-/blob/master/examples/src/bin/rtsp-server.rs`