RF Drone Interim Technical Report

Steve Gillet
Texas Tech University
October 2022

1 Acknowledgement

I would like to thank Dr. Storrs, Rishikesh, Isaac Mondragon, numerous other ECE lab students, all of the contributors to the BetaFlight and ExpressLRS software, and all of the online sources used in this project. This project definitely has been built on the shoulders of giants. This project has long been Rishikesh's brain child and he has poured an immense amount of time and resources into and the vast majority of the credit belongs to him, I am just grateful to be along for the ride and to be able to help and to gain this experience which I consider valuable and truly just fun.

2 Abstract

The project is a remote-controlled flying drone. The PCBs for the flight controller, receiver, and transmitter and the body of the drone are or will be made from scratch. The remote controller will be probably premade but customized for the transmitter. The propellors, motors, ESC, and battery are premade. The software used will be BetaFlight for programming the flight controller and ExpressLRS for programming the receiver and transmitter, both are open-source programs that will be modified slightly for the needs of the project. Specifically, BetaFlight will be customized to allow for the unique deployment mechanism that will be employed.

Contents

List of Figures

3 Introduction

This paper describes the technical details of Dr. Storrs Project Lab 3 Group 4 RF Drone Project. The drone is a pocket-sized, collapsible flying robot. The drone is composed of 4 DALDROP T5045C 5-inch Tri-Blade Propellors, 4 XING 2306 Brushless Motors, a Readytosky 250mm FPV Racing Drone Frame, a SpeedyBee 35A Mini ESC, a CNHL 1500mAh 4S Lipo Battery, a flight controller made with an STM32F722RET6 MCU, an ICM-40609-D IMU, an ICP-20100 barometer, 16MB quad SPI flash memory, and several interfaces and connectors, a receiver made with a SX1280 IC, ESP8566 MCU, and 2.4GHz IPEX Antenna, a transmitter made with a SX1280, ESP8566, and a SMA Antenna, and a Radiomaster Zorro remote control. The flight controller is flashed with SPEEDYBEEF7MINIV2 4.3.1 firmware from BetaFlight and the receiver and transmitter are flashed with BETAFPV 2400 Nano 3.0.1 firmware from ExpressLRS.

4 Ethics and Safety

The primary ethical concerns for this project involve the originality of ideas and the proper credit to sources and the following of Texas Tech and FCC drone flying regulations and the primary safety concern is for the quick moving projectile nature of the drone and the electricity being used. All images and codes used from outside sources are carefully cited in the reference section.

Electrical safety is addressed in our lab safety training and the only major source of electricity we are dealing with is the lipo battery for which regular lab safety protocols are followed including handling and disposal. There is a bin in the lab where we work dedicated to burning batteries that is located near the entrance.

The FCC and Texas Tech regulations are currently being avoided by keeping the drone under half a pound in weight and only flying it in doors for testing. There is an outdoor area that we have already designated as a nearby retired Air Force base.

5 Software

5.1 BetaFlight

BetaFlight is the main software used in this project and it does the majority of the heavy lifting. BetaFlight is an open source firmware made by drone hobbyists around the world. The firmware itself consists of C files for all of the drivers and programs necessary to run the different components and functions of quad propeller drones. Figure 1 shows a screenshot of an example of the BetaFlight source code written in C. The functions in the screenshot show how the RX refresh rate is calculated.

```c
void updateRcRefreshRate(timeUs_t currentTimeUs)
{
    static timeUs_t lastRxTimeUs;

    timeDelta_t frameAgeUs;
    timeDelta_t frameDeltaUs = rxGetFrameDelta(&frameAgeUs);

    if (!frameDeltaUs || cmpTimeUs(currentTimeUs, lastRxTimeUs) <= frameAgeUs) {
        frameDeltaUs = cmpTimeUs(currentTimeUs, lastRxTimeUs); // calculate a delta here if not supplied by the protocol
    }

    DEBUG_SET(DEBUG_RX_TIMING, 0, MIN(frameDeltaUs / 10, INT16_MAX));
    DEBUG_SET(DEBUG_RX_TIMING, 1, MIN(frameAgeUs / 10, INT16_MAX));

    lastRxTimeUs = currentTimeUs;
    isRxRateValid = (frameDeltaUs >= RC_RX_RATE_MIN_US && frameDeltaUs <= RC_RX_RATE_MAX_US);
    currentRxRefreshRate = constrain(frameDeltaUs, RC_RX_RATE_MIN_US, RC_RX_RATE_MAX_US);
}

uint16_t getCurrentRxRefreshRate(void)
{
    return currentRxRefreshRate;
}

#ifdef USE_RC_SMOOTHING_FILTER
// Determine a cutoff frequency based on smoothness factor and calculated average rx frame time
FAST_CODE_NOINLINE int calcAutoSmoothingCutoff(int avgRxFrameTimeUs, uint8_t autoSmoothnessFactor)
{
    if (avgRxFrameTimeUs > 0) {
        const float cutoffFactor = 1.5f / (1.0f + (autoSmoothnessFactor / 10.0f));
        float cutoff = (1 / (avgRxFrameTimeUs * 1e-6f));  // link frequency
        cutoff = cutoff * cutoffFactor;
        return lrintf(cutoff);
    } else {
        return 0;
    }
}
```

Figure 1: BetaFlight Source Code Example

The BetaFlight software contains hundreds of C, C++, and Assembly files with thousands of lines of code and within that is the code needed for several different configurations and utilities. The BetaFlight Configurator is used to actually choose and customize the configuration needed and to flash it onto the flight controller. Figure 2 shows the main page of the BetaFlight Configurator. This is where the majority of the work is done software-wise with the drone.

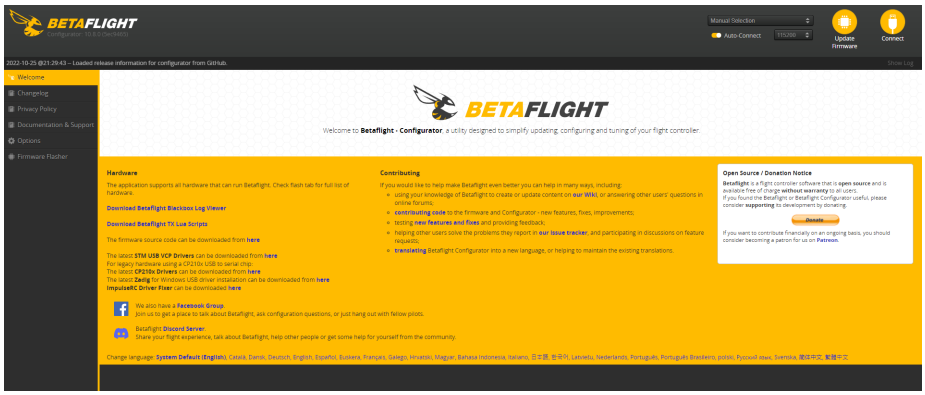

Figure 2: BetaFlight Main Page

Inside of the BetaFlight Configurator various configurations are set by connecting directly to the flight controller. The accelerometer is calibrated according to where it's set on the drone. The ports and types of ports are set, in this project currently only the one UART connecting to the receiver is set. Preset configurations may be chosen and for this project the SupaflyFPV Freestyle 5 Inch preset was experimented with, although that set up has not quite given the desired results. The receiver can be set up for different modes and channels, this project uses a UART connection and CRSF serial receiver provider. The ESC and motor features can be set, the drones motor directions have to be set because with a quad copter the motors have to go in different directions. In Figure 3 it shows the orientation of the motors, as well as the ESC protocol, the gyro input, and the manual motor test. The last configuration tool that is used is the CLI which allows direct command input for various settings, the serialrx_halfduplex option had to be turned off so that the correct UART communication could be used with the receiver.

5.2 ExpressLRS

ExpressLRS is another open-source software used for radio control, it is used to program the receiver and transmitter. The receiver is connected to the flight controller and the receiver is programmed through BetaFlight. Figure 4 shows the configuration screen for the receiver and how BetaFlight Passthrough is used to flash the receiver and give it a binding phrase.

The transmitter is flashed in a similar fashion although directly through UART. The remote controller is connected via USB and flashed and then a LUA script is downloaded and flashed onto a micro SD card that goes into the remote controller, the LUA script gives you options for connections and mode selection.

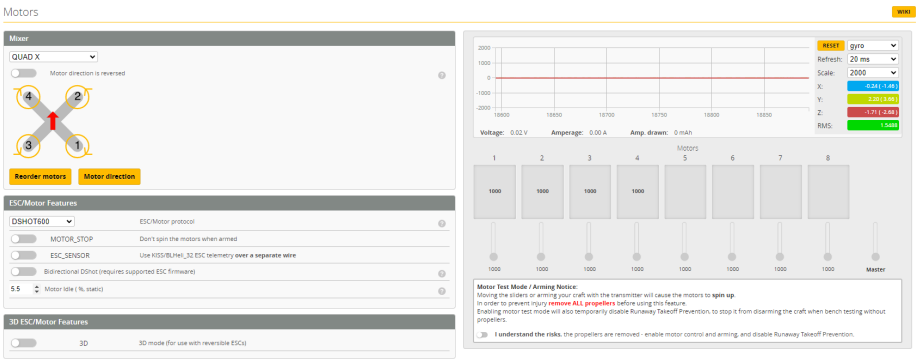

Figure 3: BetaFlight Motor Configuration

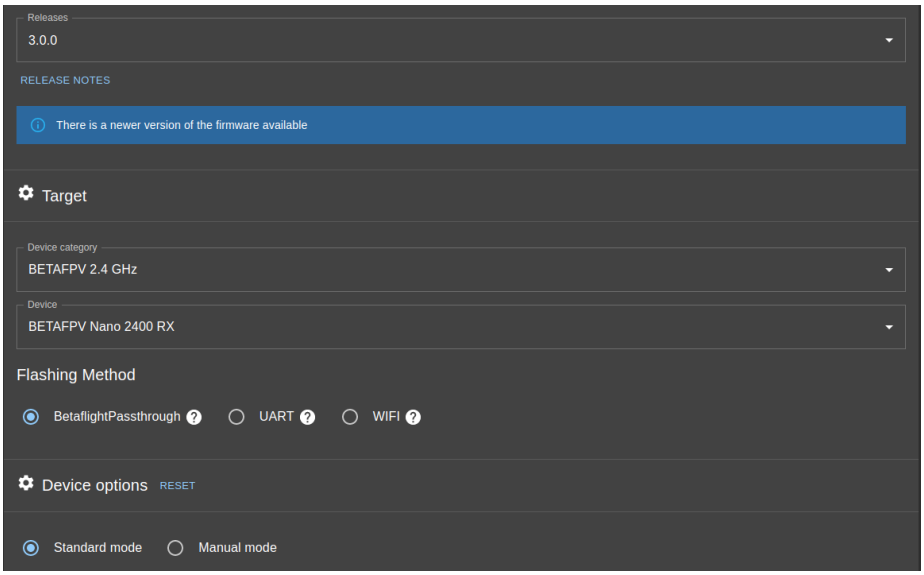

Figure 4: ExpressLRS Configurator

6 Hardware

There is various hardware used in this project. The major components are the flight controller, the drone body, and the remote controller.

6.1 Flight Controller

The flight controller is the main controller of the drone. Connected directly to the flight controller is the ESC which controls and gives feedback from the motors and the receiver which receives signals from the transmitter in the remote controller.

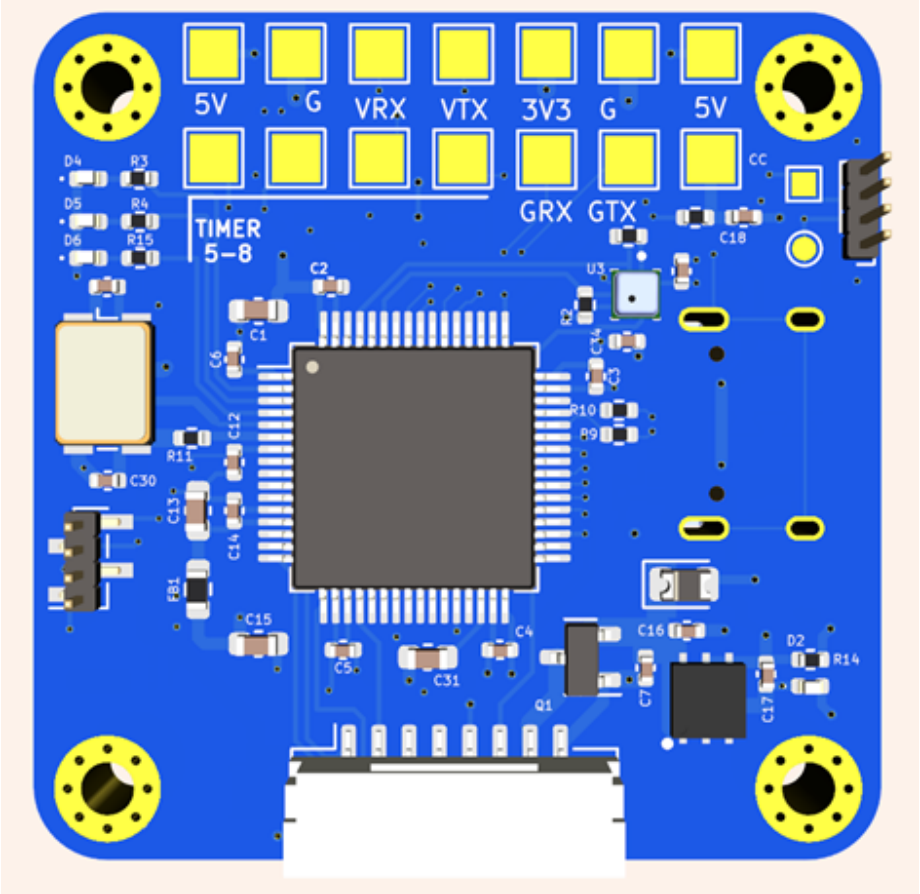

Figure 5: Flight Controller Top

The flight controller is composed of an STM32F722RET6, an ICM-40609-D, an ICP-20100, 16MB quad SPI flash memory, 5 UARTS, 2 SPIs, 1 I^2C, 1 ADC, Serial Wire Debug, a USB-C connector, and 4 general purpose GPIOs. Figure 5 shows the top view of the flight controller. It was designed in house.

The MCU of the flight controller is the STM32F722RET6, this is what gets flashed by the BetaFlight software, this is what does all of the controlling and

math for the other components. The STM32F722RET6 has a ARM Cortex-M7 32-bit single-core processor which has a clock speed of 216 MHz. It has 50 IO pins, 512KB flash memory, 256K x 8 RAM and operates at 1.7-3.6V. At $13.38 this is a mid-priced chip that provides a lot of bang for buck.

The ICM-40609-D is an IMU that provides all of the motion tracking for the drone. It contains a 3-axis gyroscope and a 3-axis accelerometer. The flight controller uses this information to maintain direction and balance. The ICP-20100 is a barometer that is used to determine altitude of the drone. The drone uses this altitude information to determine at when to deploy.

The SpeedyBee 35A BLHeli_S Mini 4-in-1 ESC is a powerful ESC that can run at a continuous 35 amps and burst up to 45 amps. The ESC is used to control the motors, it provides the power and direction for each of the motors and receives feedback from the motors including how much power was sent to the motor and how much it actually moved and this information is sent back to the flight controller where it can be used to calculate compensations that need to be made. Figure 6 shows the ESC in the drone body connected to the motors. The 7 pin connector is used to connect to the flight controller which sits on top.

The receiver is the last component that is directly connected to the flight controller. It is made out of an SX1280, an ESP8566, and a 2.4GHz IPEX Antenna. The SX1280 is a long range, low power 2.4GHz transceiver. It's a small package and cheap. The ESP8566 is the MCU for the receiver

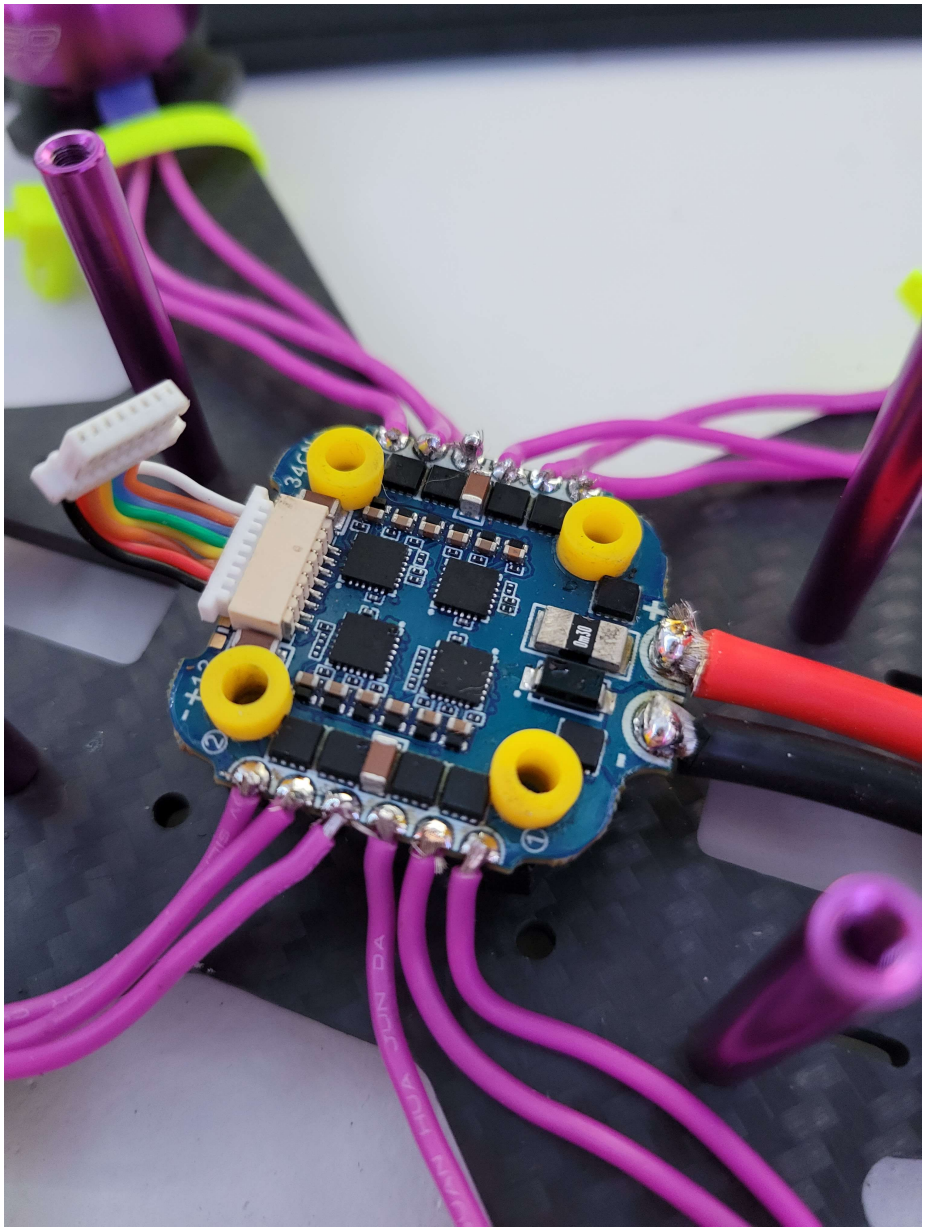

Figure 6: ESC