Reverse engineering the Blade family of drones

Chris Venour, March 2018

Introduction

This document is the second part of a three-part series about how to reverse engineer a Blade transmitter so that members of the Blade family of drones, such as the Blade nano, 180QX or 200QX models, can be controlled by a homemade transmitter, and ultimately by a computer. Part 1 described how to 'sniff' or discover the digital signals coming into the antenna module of a Blade transmitter using a logic analyzer. Part 2 (this document) describes how to build a homemade transmitter that uses these digital signals to control a Blade drone. Part 3 explains how to connect a computer to the homemade transmitter so that the drone can be controlled by machine learning algorithms running on a computer.

How a Blade transmitter controls a drone

A Blade transmitter has two joysticks which control the motion of a drone. These joysticks are connected to potentiometers which adjust the voltage of a circuit. For instance, when a joystick is in the down position, the voltage across that joystick's potentiometer drops to 0. When the joystick is all the way up, the voltage across the joystick is at its maximum (in this case 3.3 V). In this way, the voltage read at a joystick represents that joystick's position.

A component in the transmitter measures the joystick voltages and converts them to digital values: bytes ranging from 0 to 255. These bytes are then sent to the transmitter's antenna module using the Universal Asynchronous Receive and Transmit (UART) serial protocol. Radio waves corresponding to the values of the bytes are sent from the antenna to the drone, thereby controlling its motion.

Digital commands that control the drone

As Part 1 explained, 'sniffing' or eavesdropping on the digital signals that a Blade transmitter sends to its antenna module revealed that the drone is controlled by 14-byte commands that are sent to it every 21 ms. Figure 1 shows which parts of the 14-byte command control the throttle (the amount of power sent to the motors), roll (left/right movement), pitch (forward/backward movement) and yaw (rotation) of the drone.

Figure 1. 14 byte commands control the flight of Blade drones.

As the figure demonstrates, pairs of bytes in the 14-byte command control the movement of the drone, and the range of values these bytes can take on is provided here:

- The 1st and 2nd byte are the start of frame bytes and they must equal 88 and 0 respectively.
- The 3rd and 4th byte control the throttle. No throttle is represented by the values 0 170, medium throttle is 01 170 , and maximum throttle is 03 84.
- The 5th and 6th byte control the roll. Maximum roll to the left is 07 84, no roll is 05 255, and maximum roll to the right is 04 170.
- The 7th and 8th byte determine the pitch. Maximum backwards pitch is 08 170, no pitch is 09 255 and maximum forwards pitch is 11 54.
- The 9th and 10th byte determine the yaw/rotation. Maximum rotation to the left is 15 84 , no rotation is 13 255, and maximum rotation to the right is 12 170.
- The 11th, 12th , and 13th byte are ignored by the drone and can take on any value between 0 and 255.
- The 14th byte marks the end of the 14-byte command and it must equal 170.

The command shown in figure 1 applies medium throttle to the drone and the roll, pitch, and yaw are at their neutral settings; that is, the drone is being instructed to simply hover.

To apply maximum throttle to the drone and to get it to make a hard left, for example, the microcontroller would have to change the 3rd and 4th bytes in figure 1 to 03 and 84 (maximum throttle), change the 5th and 6th bytes to 07 and 84 (maximum roll to the left), and leave the values of all the other bytes in the figure unchanged.

Components of the homemade transmitter

The homemade transmitter, shown in figure 2, consists of components that sit on a half-size breadboard. The transmitter is small – it fits in the palm of your hand – and is powered by a small 3.7 volt lithium battery. Push-buttons, rather than joysticks, are used to control the drone's movement and a small potentiometer controls the throttle.

A brief description of each of the transmitter's components is provided here:

Microcontroller (atmega328p): This component – the black chip at the bottom and middle of the breadboard in figure 2 - has a processor that's similar to a computer's CPU. Software on the microcontroller - a C program that I wrote on a computer and downloaded onto the microcontroller – controls the behaviour of the homemade transmitter.

Antenna module: This module was desoldered from a Blade transmitter and appears in the top right corner of the breadboard in figure 2. The 14-byte digital commands described earlier are sent from the microcontroller to the antenna module. Radio waves corresponding to the values of a 14-byte command are then generated and sent by the antenna module to the drone.

Serial connection: In figure 2, a yellow wire to the left of the microcontroller can be seen running from the microcontroller to the antenna module. This wire connects the microcontroller's TX (transmit) pin to the antenna module's RX (receive) pin. The microcontroller transmits the 14-byte commands over this wire, one byte at a time, using the UART serial protocol.

16 MHz crystal: The microcontroller has an internal clock that can tick at 8 Mhz but this frequency isn't high enough to ensure that the 14-byte commands are sent to the antenna every 21 ms. Therefore, a 16 MHz external crystal, the silver component at the bottom left of the microcontroller, was added to the circuit to improve the microcontroller's timing accuracy.

Push-buttons: Two buttons control the drone's pitch (forward/backward movement) and two buttons control the drone's roll (left/right movement). These four buttons are electrically connected to four pins on the microcontroller. Interrupt Service Routines (ISRs) running on the microcontroller listen for changes on these pins. When the 'left' button, for example, is pressed, an ISR attached to a microcontroller pin activates and increments the 5^{th} and 6^{th} byte of the 14-byte command. When the microcontroller next sends the 14-byte command, the drone receives instructions to turn left.

Potentiometer: As explained earlier, the voltage output by a potentiometer can represent the direction in which the drone should move. Similarly, the voltage across a potentiometer can symbolize how much power to send to the drone's motors (i.e. the throttle), and that's the function performed by the potentiometer in figure 2 – the white knob in the top left corner of the breadboard.

Battery: A 3.7 volt battery provides power to the components on the breadboard.

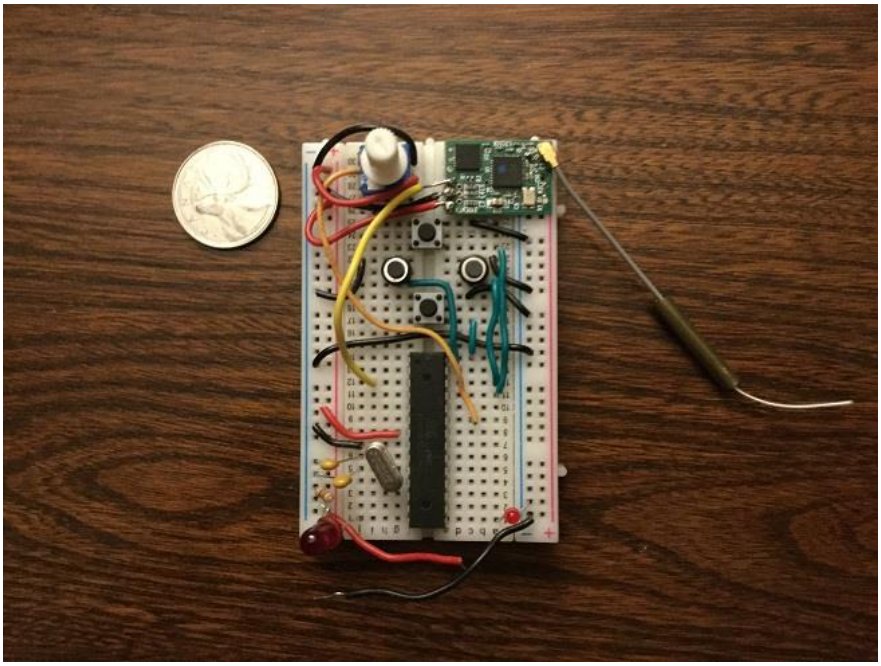

Figure 2. The transmitter built for this project fits on a small breadboard that's the size of a credit card

Programming the transmitter

The homemade transmitter's software consists of a C program that I've written and downloaded onto the microcontroller. The first thing the microcontroller does when it's turned on is run some initialization code which modifies certain bits in some of the microcontroller's registers. The bits in these registers control the behaviour of various hardware peripherals that are built into the microcontroller.

For example, the microcontroller's Analog to Digital Converter (ADC) needs to be activated so that the voltage output by the potentiometer, which is acting as the throttle in this project, can be measured and converted to digital form. Similarly, special serial communication hardware built into the microcontroller needs to be initialized and activated so that the microcontroller can communicate with devices in the outside world, such as this project's external antenna module.

Once some of the microcontroller's internal peripherals have been activated and their behaviour defined, the C program sends a command to bind with the drone. Like the 14-byte commands that control the drone, the bind message was discovered using a logical analyzer, and it too consists of 14 bytes which need to be sent to the drone every 21 ms. To bind properly with the drone, the transmitter needs to send the the bind message to the antenna module (over the serial line) for at least two seconds when the drone is turned on.

When the homemade transmitter has successfully bound with the drone, the C program enters an infinite loop in which the microcontroller sends 14-byte control commands to the antenna module every 21 ms. When someone pushes a button on the breadboard or turns the potentiometer/throttle, Interrupt Service Routines (ISRs) defined in the C program are activated. These ISRs modify the bytes in the 14-byte command that gets sent to the drone, and in this way the drone’s motion is controlled.