UAS Fundamentals - Lab ten

May 8, 2018

Magnus O. C. Liisberg, Thor G. Jensen, Guillaume Quévy

1 mapping

The remote joystick being cheap, their output was not originally perfectly spread other the entire 10 bits range [0:1023], and the idle positions of the analog sticks were not always outputting 512. To solve this problem, we started measuring the most extreme values we could possibly get from the sticks and also the idle values. Each axis has then been mapped like so : The interval between the minimum value and the idle value gets mapped from 0 to 512, and the interval from the idle value +1 to the maximum value gets mapped from 513 to 1023. We are then sure that the remote can output values from 0 to 1023 with the idle position outputting 512. If somehow the sticks reach a minimum lower than the measured minimum or greater than the measured maximum then this value is updated. This mapping method should only be used when using the analog sticks with the SDU UAS TX we used, as the measured values are specific to this one.

2 Computer controlled C2

To enable the us, to control the SDU UAS TX (SUT). We need to create a ROS node, which could transmit data over a serial port to the Arduino. For the Arduino a code, which could interpret the serial communication had to be devised. Which meant to begin with that a communication protocol had to be thought of. It was decided that values between 1 and 1000 would be sent, since that was the value that were read from the joysticks on the transmitter. Values for: Throttle, pitch, aileron and rudder would be sent, delimited by ":". This means a message would look the following way 500:50:50:50, this message would set throttle to 500, pitch, aileron and rudder to 50.

2.1 ROS node

The ROS node was written as a class and consist of only two functions besides the initialize section. The initialize section are used to define the parameters of the serial communication like the baudrate, the bytesize and from which port to send/receive serial communication on.
The first function used thereafter is *initialize_serial* which which only opens the serial communication, which is possible since the parameters are initialized when the class is run.
The last function emulates a transmitter by listing on the input from the keyboard and when key has been pressed the user presses 'enter' and the function packs the different values for each controller into a string and sends it. Throttle is controlled by 'w' and 's', rudder is controlled by 'a' and 'd', elevator is controlled by 'i' and 'k' and lastly the roll i controlled by 'j' and 'l'.

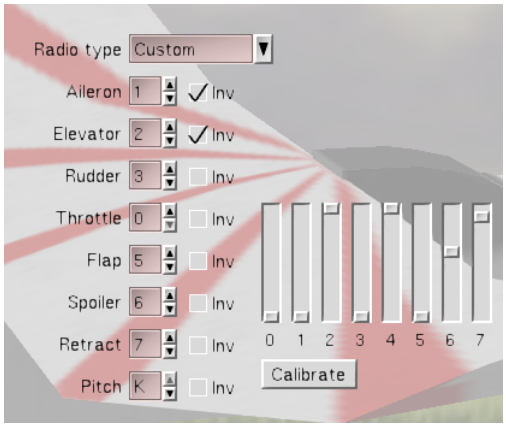
A keystroke on each of these either increase or decrease the current integer value saved in for each of the four controls. One major drawback is that the function used for reading keyboard stroke *raw_ input*, stops the program until 'enter' has been pressed so the user has to hit 'enter' betweem each adjustment of the drone.

2.2 Arduino code

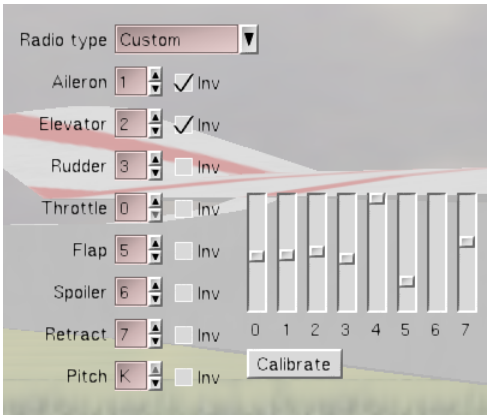
The Arduino was set up the to read the serial input, and push evering into a string using the *readString()* function. Then the string is divided using the function *indexOf()*, which found the index of the delimiter. When the index of the delimiter was found, the string was then divided accordingly, into four substrings. The strings were then converted to integers using the function *toInt()*, and then written to the correct ppm output using the pre-available functions.

2.2.1 Test

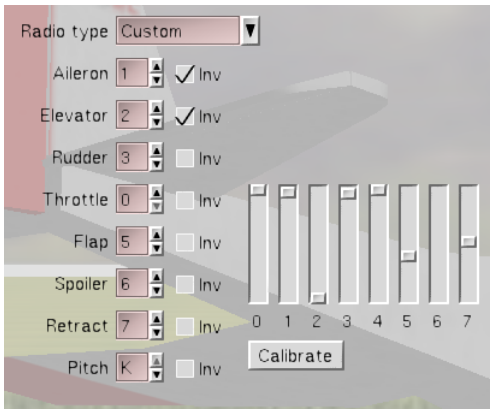
To test the code CRRCSim was use, the result can be seen in the following images. The channels 1,2,3 and 4 represent the sticks values. Channel 2 seems to be inverted but not the channel 1 like the tick boxes suggest.



(a) In this simulation all the values are set to zero on the computer, and we can see that CRRCSIM also detects that.



(b) Here all the values were set to a 512, which would mean that the joysticks are in their middle position.



(c) Here the value were set to 1023 which is the maximum they can reach

Figure 1: Stick simulation test