## 3 Section: The USB virtual comm driver (pg 356)

- **Additional info, page 356**
  - This entry provides info about `CDC_Receive_FS()`, in `usbd_cdc_if.c`
  - How `CDC_Receive_FS()` is created and used:
    - Apparently, `CDC_Receive_FS()` is generated by a tool like STM32CubeMX or STM32CubeIDE.
    - `CDC_Transmit_FS()` can be used to send data over a USB port, and that function is globally available. For example, in Chapter 11, `CDC_Transmit_FS()` is called from `mainRawCDC.c`.
    - However, `CDC_Receive_FS()` is not globally available. It is a static function defined in `usbd_cdc_if.c`, so it cannot be used outside of that file.
    - `CDC_Receive_FS()` gets called from a USB ISR, when data is received over a USB port.
    - The function's parameters specify a pointer to the received data, and its length.
    - The user can add code to `CDC_Receive_FS()`, to store the received data, e.g., the call to `xStreamBufferSendFromISR()`.
  - User-code added to `CDC_Receive_FS()`:
    - These findings include some speculation, so they might not be completely correct.
    - All of the code is user-code (i.e., created by Brian Amos, the book's author), except the return.
    - These two function calls are needed to set-up for getting the next set of USB data that will be received:

```
USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
USBD_CDC_ReceivePacket(&hUsbDeviceFS);
```

    - The call to `USBD_CDC_ReceivePacket()` tells the CDC layer that it's OK to receive the next packet. So, that call might need to be after any code that uses the current data, e.g., after the call to `xStreamBufferSendFromISR()`.
    - In the user-code, `USBD_CDC_SetRxBuffer()` is called, then `xStreamBufferSendFromISR()`. This call-order isn't required. `xStreamBufferSendFromISR()` could be called first.
  - Reference info for `CDC_Receive_FS()`
    - `CDC_Receive_FS()` isn't well documented.
    - Tutorials
      - "Send and Receive data to PC without UART (STM32 USB COM)"
        - https://controllerstech.com/send-and-receive-data-to-pc-without-uart-stm32-usb-com/
      - "Hints for using the CDC USB Serial"
        - https://hackaday.io/project/20879-notes-on-using-systemworkbench-with-stm32-bluepill/log/57048-hints-for-using-the-cdc-usb-serial
    - Forum discussions
      - "embedded - STM32 USB CDC Rx Interrupt - Stack Overflow"
        - https://stackoverflow.com/questions/64878727/stm32-usb-cdc-rx-interrupt
      - "Read data from PC to STM32 via USB CDC"
        - https://community.st.com/s/question/0D50X00009Xkfd1SAB/read-data-from-pc-to-stm32-via-usb-cdc
      - "How best to do CDC Host PC to microcontroller virtual com port communications?"
        - https://www.openstm32.org/forumthread3159

## 4 Section: Using the code (pg 357)

- **Clarification, page 357**
  - This section provides info on installing the packages needed to run `colorSelector.py`.
  - Installing the packages:
    - `requirements.txt` lists the needed packages, including the needed versions, e.g.,

```
pyserial==3.4
```

    - The packages can be installed via: `pip install -r requirements.txt`
      - **But, read the next bullet before running `pip install`**
  - Possible installation problems:
    - Since `requirements.txt` specifies the package versions, the `pip install` will replace packages that are already installed, but have a different version. **This can result in packages being replaced with an older version.**
    - There's two possible solutions for that problem:
      - One solution is to use a Python “Virtual Environment” for `colorSelector.py`.
      - Another solution is to install the most recent version of the required packages.
        - In `requirements.txt`, the version numbers would need to be removed, and the "==" symbols.
        - Then run: `pip install -U -r requirements.txt`
        - This worked using the current packages on 12/2021.