Firmware Coding Challenge

Design a bootloader system with a set of requirements as given below, and implement it in C language.

1. A host device needs to download an image to a target device. It is assumed that the target device code is complete, and you are expected to write code for the host side.
2. The target device has a bootloader, and two partitions (one image per partition). At bootup, the bootloader starts and waits (upto 2 seconds) for any communication from the host. If there is no communication for 2 seconds, it boots into its primary image (which could reside on any partition).
3. The host device would follow these steps:
    a. Reset the target device to start it in bootloader mode
    b. Query the target device to determine which image versions are stored in each partition
    c. Compare both image versions with the new image version (which is to be downloaded) to see if a new download is needed.
    d. If no download is needed, just switch to the right partition.
    e. If a download is needed, proceed as such, and mark the partition as primary.
    f. Restart target device at the end.
4. Mode of communication between the host and the target device could be one of these: UART, I2C, or SPI. The exact protocol implementations can be avoided, but stubs need to be written, and a compile-time protocol selection mechanism should be available.

Other requirements:

1. Please submit this as a project in Github
2. Please follow standard coding guidelines and add comments in code

Tips:

1. You can create a project in any IDE (Eclipse, Atom, VScode, Sublime, etc.) to implement this. But make sure it compiles without errors.
2. Some of the other details (such as download protocol, frame headers, etc.) can be assumed as deemed convenient.