Compact Firmware Upgrader for Robot System Based on IAP Methodology

Wancheng Zhou (wz233@cornell.edu) Advisor: Dr. Bruce Land

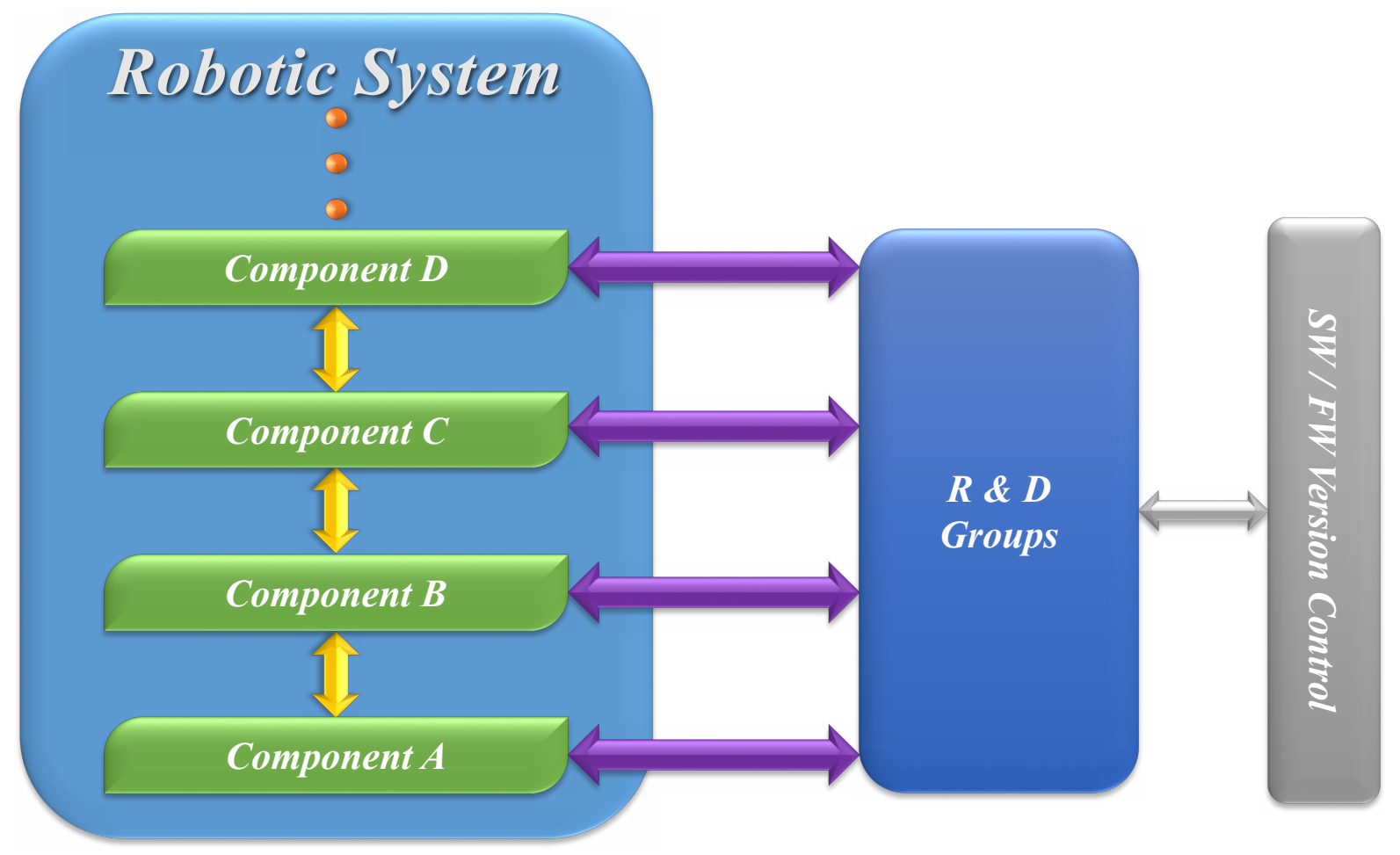
School of Electrical and Computer Engineering, Cornell University

Achievements

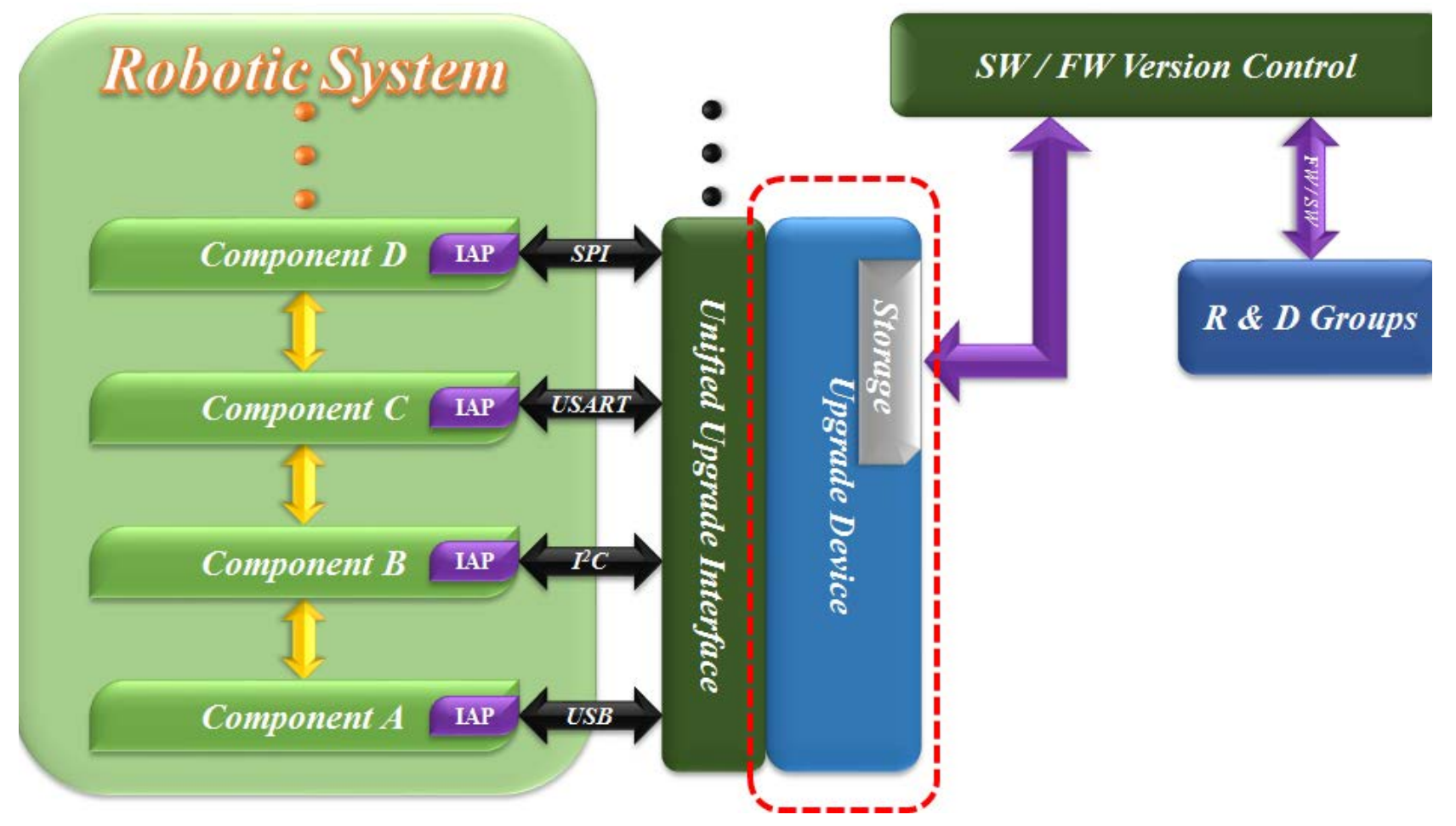
- Designed a low-cost, intelligent system to upgrade the firmware of another system efficiently.
- Designed a head file only protocol (SMART-LINK) to support firmware upgrade in a smart way.
- Designed a USB interface (virtual Disk) for easy firmware import.
- Designed a bootloader template for other different systems (AVR platform is used in demo).

Motivation

In complex Robot System design, efficient firmware upgrade is vital, even though we own a good version control system, firmware upgrade can also be a painful problem. Different hardware platform requires different programming devices. And it becomes worse when the whole system has already been assembled. Engineers often need to tear the assembled system down in order to upgrade some internal modules which is definitely suffering.



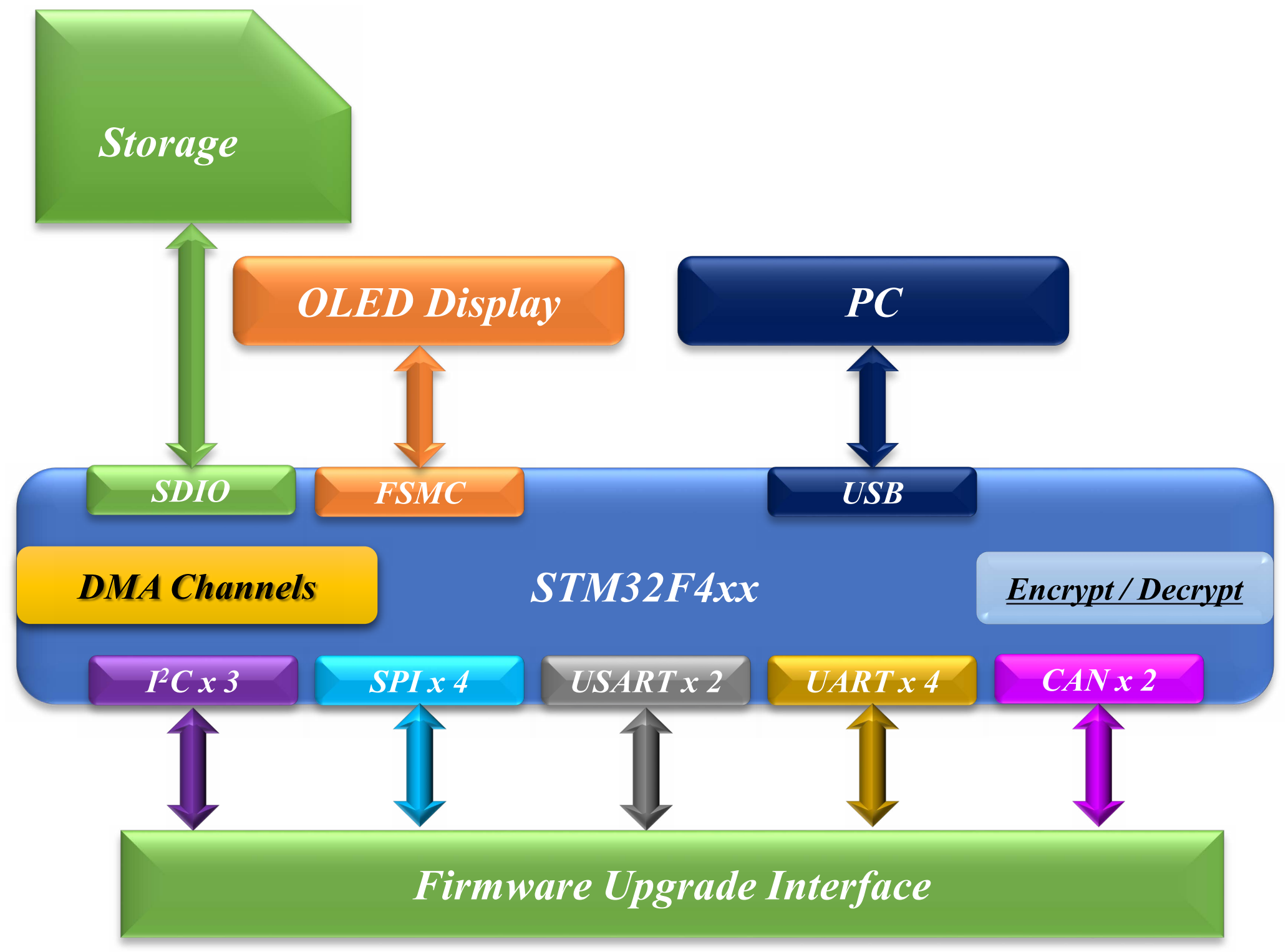
Thie project aims to design a system which is only responsible for firmware upgrade. The sytem is equipped with abundant communication interface that can be connected to multiple components within the robot system. Thus it is applicable to upgrade all components directly or in-directly.



Implementation

A. Hardware Design

The hardware of this upgrader must own abundant hardware peripherals for communication purpose. Also, it should contain memory interface (like SDIO) to store and retrieve firmware. Based on above considerations, the core MCU is chosen from ST-Cortex M4 series. Take STM32F407IGH for example, it has abundant peripheral devices, except for essential communication interface above, it owns USB-OTG (Host / Slave), FSMC (Flexible Static Memory Interface), DMA (Direct Memory Access – makes data transmission more efficient), not to mention that it also owns hardware encrypt / decrypt units which are definitely essential for firmware release.

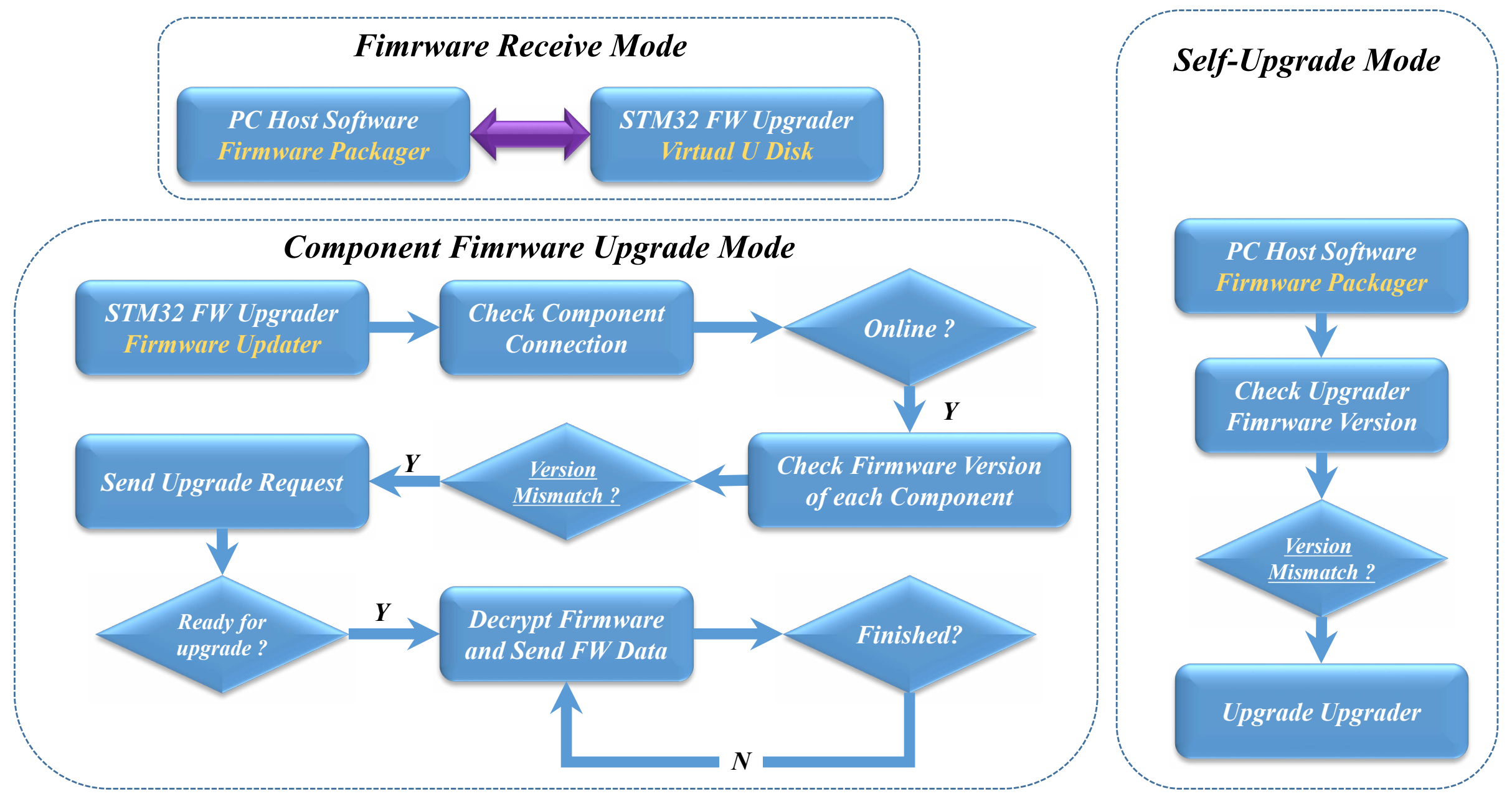


B. Software Design

The main work for software design is to implement a protocol that is independent from hardware. It must be easy to be transplanted to other hardware platforms. Based on this purpose, a head-file only protocol is designed to support the upgrader.

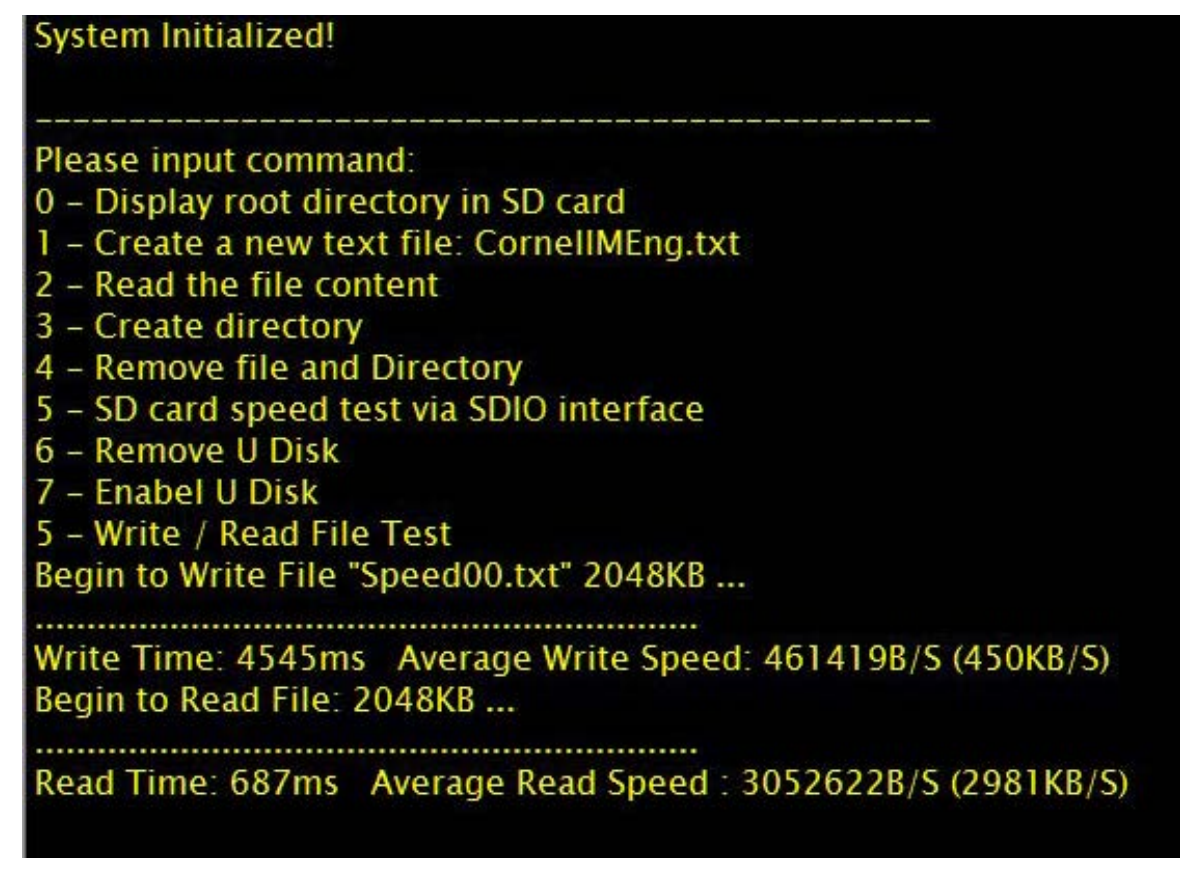
Functions	Description	Owner
SMTLINK_PackFWUpdReqMsg	Pack firmware update request message	HOST
SMTLINK_PackFWUpdRespMsg	Pack firmware update response message	SLAVE
SMTLINK_PackFWDataRcvReqMsg	Pack firmware transmit request message	HOST
SMTLINK_PackFWDataRcvRespMsg	Pack firmware transmit response message	SLAVE
SMTLINK_GetFWDataPkgSize	Get firmware size from message flow.	SLAVE
SMTLINK_PackFWDataTXReqMsg	Pack firmware data sending request message	HOST
SMTLINK_PackFWDataTXRespMsg	Pack firmware data sending response message	SLAVE
SMTLINK_UnPackMsg	Unpack firmware data from message	SLAVE

The system has 3 working modes: Firmware Receive Mode, Component Firmware Upgrade Mode and Self-Upgrade Mode. Firmware receive mode is reserved for future PC-Client software.



Results and Conclusions

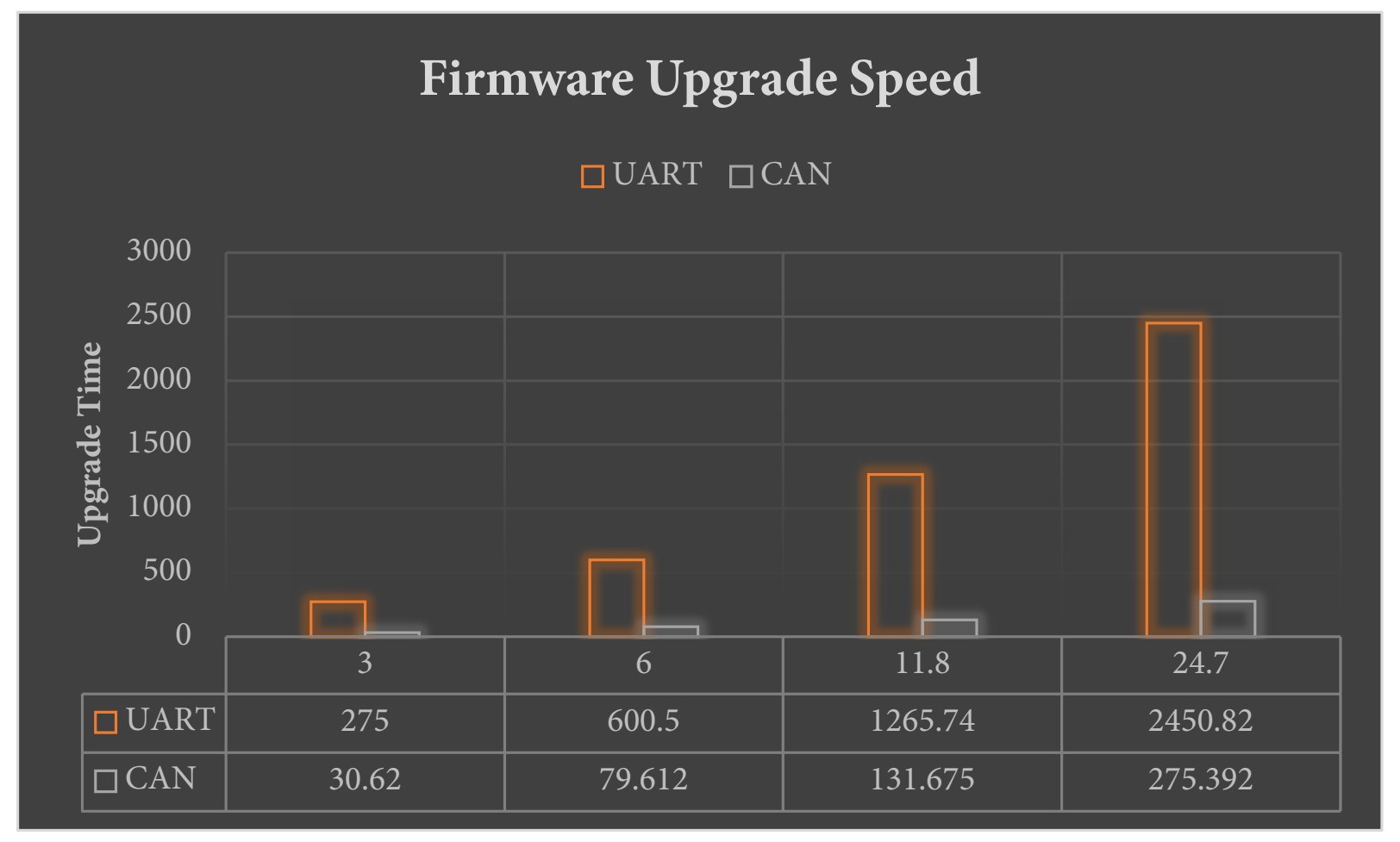
A. Firmware Write Speed Test (PC => Upgrader



```
System Initialized!

----------------------------------------------
Please input command:
0 - Display root directory in SD card
1 - Create a new text file: CornellMEng.txt
2 - Read the file content
3 - Create directory
4 - Remove file and Directory
5 - SD card speed test via SDIO interface
6 - Remove U Disk
7 - Enabel U Disk
5 - Write / Read File Test
Begin to Write File "Speed00.txt" 2048KB ...
.............................................................
Write Time: 4545ms   Average Write Speed: 461419B/S (450KB/S)
Begin to Read File: 2048KB ...
.............................................................
Read Time: 687ms   Average Read Speed : 3052622B/S (2981KB/S)
```

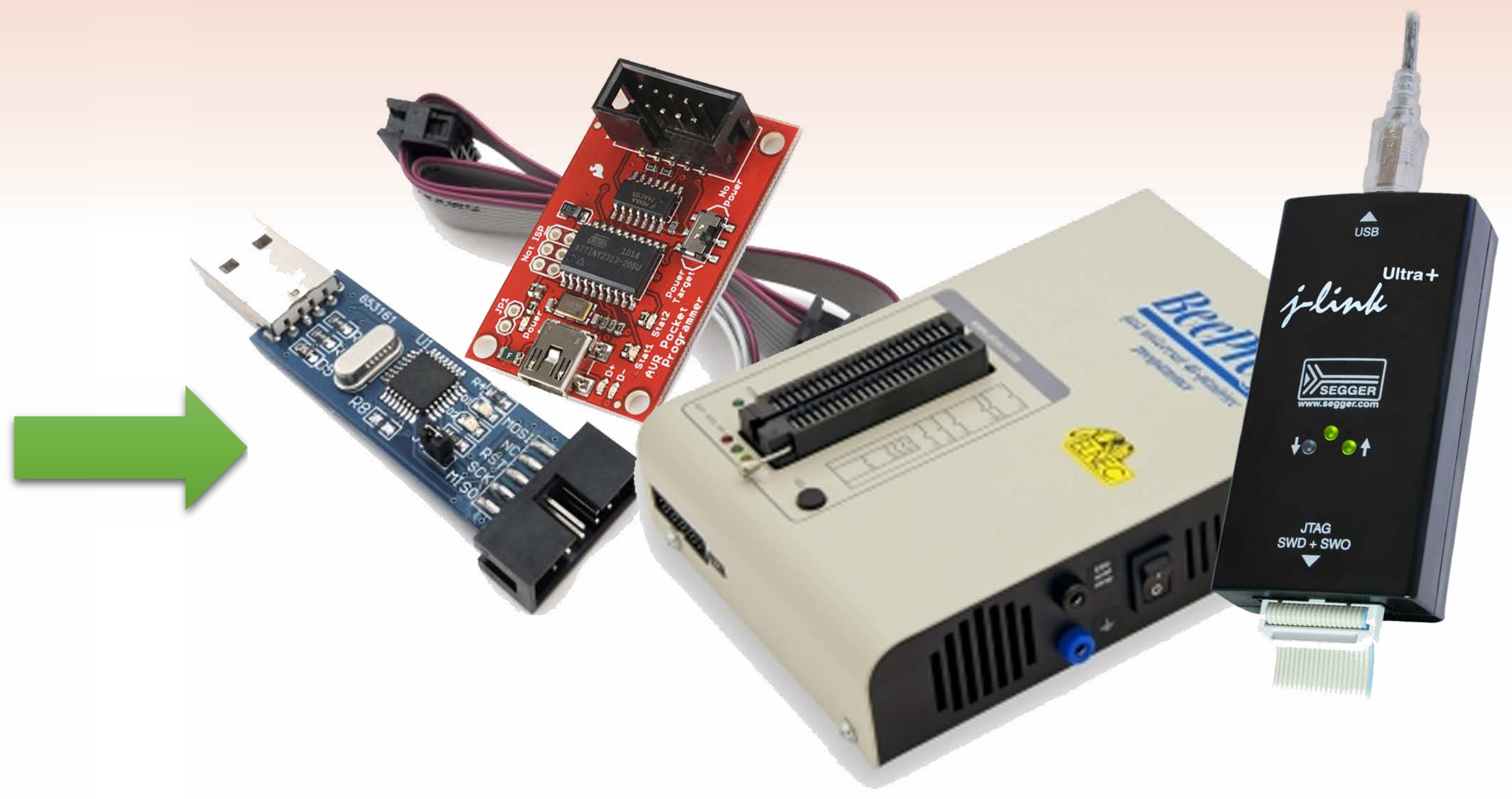
B. Firmware Upgrade Speed Test



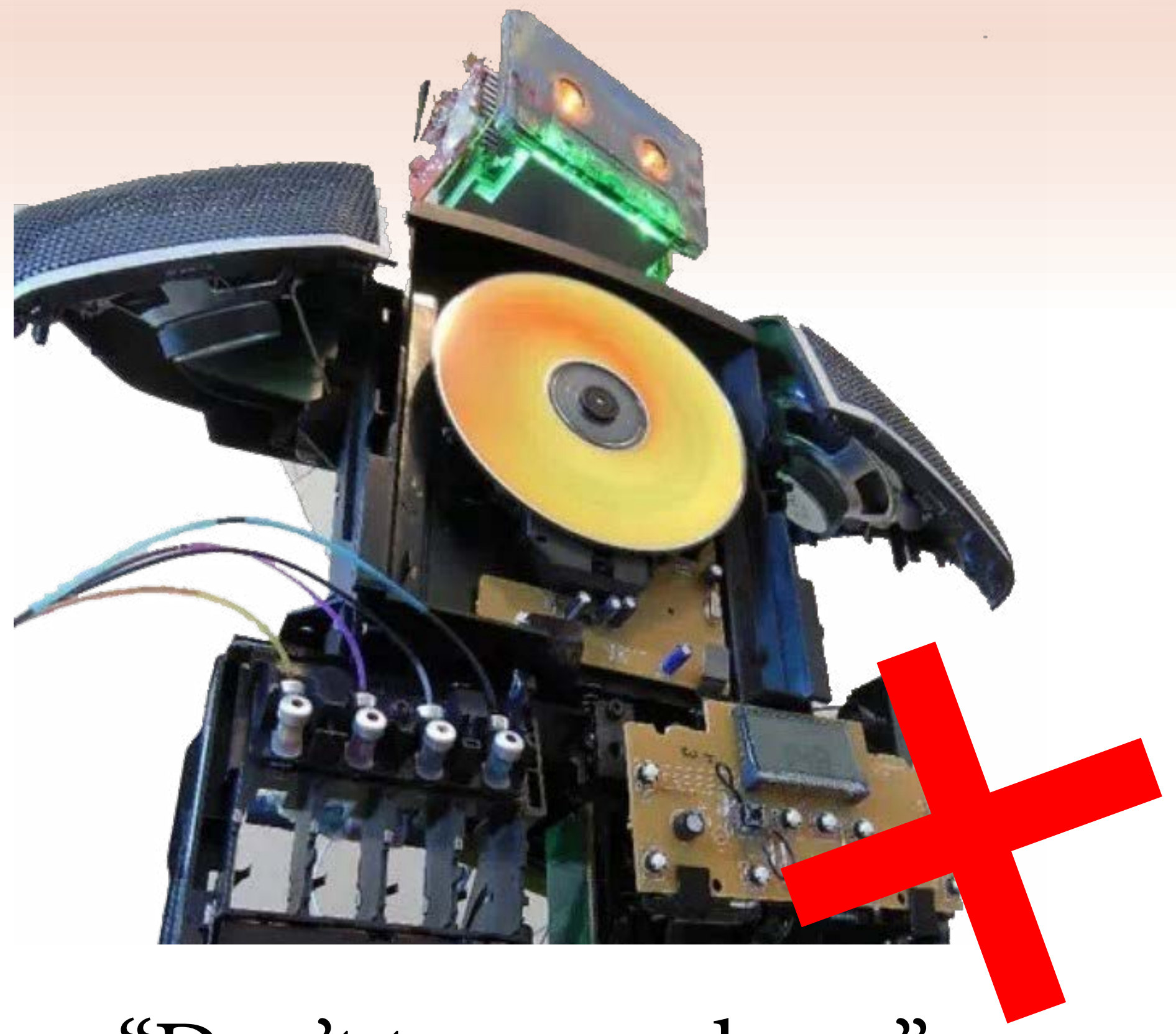
Firmware upgrade speed is primarily limited by specific upgrading interface. Faster communication port (or normal ports with DMA support) can achieve higher upgrade speed.

There still reamains a lot of work to do to finally perfect this system. For instance, PC Client software design with GIT-SYNC function; Utilize state machine to achieve parallel firmware update. Improve encrypt function in SMTLINK to raise efficiency.

“I need upgrade...”



Different kinds of programmers



“Don’t tear me down”