**Provisioning Jig v3**

Project by Jesse Jabez Arendse  
Project for UCT EEE Department - Robyn Verrinder & Justin Pead  
  
**Deliverables**:

* Portable and Handheld System for Provisioning UCT Development Boards’   
   - Current Debugger (STM32F103C8) with latest STM32 STLink V2 version   
   - Target (STM32F051C6) with James Gowen’s Demo
* Design New Debugger (STM32F103CB), to be latest STM32 STLink V2.1 version, which introduces USB Re-enumeration, MSD for Drag and Drop Programming, and VCP for built-in USB-UART communication, that is also pin to pin compatible with the Current Debugger.

**Host Machine**:   
Raspberry Pi 3B+ with Raspberry Pi OS 64-bit, with 16GB SD Card

Username: ProvisioningJig  
Password: ProvisioningJig  
Hostname: raspberrypi

**What was done on the RPI**:  
- installed openocd, telnet, and STLink drivers for Linux x64  
- pip installed pexpect and RPI.GPIO  
  
**What does the system need to do:**  
- Auto login and run script on startup  
- Automatically detect which device is plugged in, and the version  
- Flash the device via SWD, with its specific binary file  
- Upgrade the STLink to the latest version, this step needed to be done with the STSW-LINK007 software package, which upgrades the STLink’s firmware via USB DFU   
  
**What could the system not do:**  
- Initially, using Windows on R and installing STLink Drivers for ARM64 was the only way to run the STLink firmware upgrader tool on the RPI, since the drivers and operating system combinations are either Linux/Windows x86\_64 or the above.   
A GUI is not needed to use the tool, simply run the command:  
*$ ./<stlink-firmware-upgrader.exe or .jar > -force\_prog -jtag for V2  
$ ./<stlink-firmware-upgrader.exe or .jar > -force\_prog -msvcp for V2.1*  
Running windows was infeasible on the RPI, and occasionally breaks due to having to install unsigned drivers.  
  
**What does the system actually do:**  
- the debugger upgrade step has been eliminated by flashing pre-upgraded binary files (version specific), once it’s been flashed it’s ready for use.

**How this was accomplished:**  
- flashing unprotected binary files allow you to upgrade the STLink’s firmware and read the devices memory afterwards to create pre-upgraded binaries, this was done for both the current and new debuggers, at the current time the latest version was V2J45.   
The debuggers are flashed with Read-Out Protection enabled, so they are more authentic and cannot be re-read, though the pre-programmed binaries are still unprotected.

**For updates / reproduction:**  
- main script is located at /Desktop/ProvisioningJig\_Code/newMain.py  
- binary file selection can be done from the telnet\_commands.py file  
- use Win32DiskImager to create an .img of the ProvisioningJig’s SD Card, and then PiShrink to compress it (optional)  
- use Win32DiskImager to flash the desired .img file to a SD Card

**Assembly Considerations:**

Figure 1 ProvisioningJig Assembly Stacking Order

* Optional STLink Cap to further prevent damage
* Enclosure is self-locking
* 2x 10mm spacers are needed between the PCBs
* 2x M2.5 Bolts & Nuts should be used to secure the PCBs, or M2.5 Rods can be used to connect the Jig and the entire enclosure for table mounting

**Recommendations:**

* if Debugger RDP is not needed, you could write some more scripting to check the version of a debugger that’s plugged in before flashing it, if it is a newer version than current, then create a new binary from it and use that one, effectively making the ProvisioningJig parasitic and immortal. The original binary from ST is no longer used, so it’s not a breach of that agreement to use unprotected STLinks, but it could be a breach from another agreement.
* further protection against shorting pins on the RPI
* Using a RPI Zero instead of 3B+