Software Formalization

Year: 2025 Semester: Spring Team: 20 Project: Encrypted Thumb Drive

Creation Date: ­2/22/2025 Last Modified: February 22, 2025

Author: Brandon Liu Email: liu3394@purdue.edu

Assignment Evaluation: See Rubric on Brightspace Assignment

1.0 Utilization of Third Party Software

Our software will be hosted entirely in the microcontroller, which will be developed using the STM32F4 HAL family driver [1]. Although the chip used is going to be the H7 family, F4 microcontrollers are pin-to-pin compatible with the H7 microcontroller [2]. With an F4 microcontroller available, it will be the dev board hosting development code before it is moved to the H7 chip. The software used to write and develop the HAL code itself will be the STM32Cube IDE with a 1.17 or newer version [3]. Along with the IDE, the STM32Cube MX will also be used for ease of peripheral setup [4]. Currently, the HAL is available to the public via open-source BSD licensing [5] and both the STM32Cube IDE and MX have a software package license agreement that users must agree to during the installation of the development environment. The licenses mentioned are free but require the cooperation of the user to abide by the agreements when using the software.

2.0 Description of Software Components

The microcontroller needs to process user input via a fingerprint sensor and a keypad, interact with flash ICs to read and write data, display user inputs, authentication, and user states on a display, and be able to communicate information with a host computer via USB protocol. All software development on the microcontroller will use the existing libraries, HAL drivers, and code generators provided within the software that we plan to use.

Functions

1. GPIO\_init, SPI\_init, TIM\_init, UART\_init etc. These are some of the peripherals connected to the microcontroller that are to be initialized using STM32 software.
2. GPIO\_WritePin, HAL transmit, HAL receive: These are necessary for the microcontroller to communicate with peripherals using SPI and UART
3. next\_state: Function is necessary to determine how peripheral devices such as LCD as well as the microcontroller itself should behave, display, or proceed depending on its current state
4. authenticate: Function is necessary to compare user inputs on the keypad and fingerprint sensor against all known users to determine whether current user can be authorized and access which data in the file system supported by the flash ICs
5. file\_read, file\_write, file\_delete etc. All file / data functions are necessary to allow authorized users to interact with the data stored in their respective flash ICs
6. fingerprint\_poll, keypad\_poll: Functions are required for the microcontroller to detect user input

Key Data Structures

1. Buffers / Packets: needed for SPI and UART communication to hold data to be sent or received
2. User: structure to hold all attributes of a user from password, id etc. to aid in microcontroller authentication for user access of data in particular flash IC
3. HAL SPI & UART: data structures (creation aided by HAL) to setup SPI and UART communication
4. State Machine: state machine is required for devices that rely on one to know the state transitions for all of its own states

3.0 Testing Plan

All interfacing between the microcontrollers and peripherals, as well as the logical flow of the microcontroller will be rigorously tested to ensure that the encrypted thumb drive has proper functionality and no unintended behaviors. The software implementations below list the order in which components must be functional. The figure 1 diagram below displays what has been completed so far in terms of functionality. Green has been completed, yellow is in progress, and red is to be worked on in the future. The block being the source of the arrow must be completed before the block the arrow is pointing at. The chronological order in which the bolded functionalities are listed under past the diagram indicate the priority of what must be completed.

A diagram of a diagram

AI-generated content may be incorrect.

Figure 1: Functionality Testing

Peripheral & Communication Setup

This functionality must be completed first, as if the microcontroller is unable to communicate with peripheral devices such as keypads, thumbprint sensors, LCDs etc, then the entire project will not meet any of the PSDR requirements at all. This includes all initialization of peripherals or communication protocols, the working of the communication protocols themselves (ability to at least communicate with peripheral devices), packet / buffer setup, the creation of HAL SPI & UART data structures, and successful polling of both the keypad and the thumbprint sensor.

* Testing
  + Test all keypad input to ensure that the microcontroller can differentiate between the different keys that are pressed (via polling); can use different LEDs and have microcontroller light up different ones based on keys pressed
  + Test successful LCD communication via SPI; microcontroller should be able to control what is displayed on the LCD whether it be text, cursors, display on / display off etc – can be tested by sending various commands and ensure that the LCD responds properly based on commands sent
  + Test thumbprint integration; microcontroller should be able to communicate with the thumbprint sensor via UART and learn from the sensor regarding whether thumbprints match (via polling); can be tested by having different people scanning their thumb and see whether or not the microcontroller has received match or no match from the fingerprint sensor since the fingerprint sensor is the device that does the differentiation between thumbprints
  + Test flash IC integration; microcontroller should be able to interact with the data stored in flash IC – testing can be conducted by having microcontroller write data to the different ICs and see if the microcontroller can see what it wrote after interacting with the flash IC. An example would include writing bits / bytes of data into one IC, then reading it, then writing to that same block, and reading it again to see if the data is updated (can be repeated over all 4 ICs – NOT THE SAME AS FILE FUNCTIONS)\*\*\*
  + Test USB & Host computer capability; this test is to ensure that using the USB built into the microcontroller, the host computer can be attached to the thumb drive and potentially be able to read / write files (no flash IC differentiation yet) – as long as it has the same reading capabilities as any empty thumb drive, this test is passed

Peripheral Device State Machine Setup

This functionality must be configured next, as the state machine (and next state functions for all necessary devices) are a part of how the devices behave depending on input microcontroller receives from other devices; this is the beginning of integration of multiple device functionality.

* Testing
  + test state machine functionality is to have an LCD display one thing and depending on what the user inputs (keypad press or thumbprint scan), the LCD displays something else. It doesn’t need to follow authentication logic or anything yet, but if a state machine can be set up and followed, the logic can be changed later once authentication is introduced.

Authentication & User Setup

Once state machines and peripherals are all set up, authentication testing can begin. This includes password / thumbprint comparison based on user ID, only permitting a user to attempt to unlock their respective IC in a certain number of attempts etc. The user data structure to hold all IDs, passwords etc. is to be set up here as well.

* Testing
  + To test authentication, two different users (accessing different ICs) will be set up and passwords and fingerprints will be set for each. Then, authorization (inputting correct password / fingerprint) will be tested to ensure that the microcontroller is able to let the correct user in, and denial (inputting either wrong password / fingerprint) will be tested to ensure that the microcontroller can properly reject unauthorized users.
  + After both cases work, the multiple incorrect attempts will be tested to ensure that the microcontroller either constantly denies improper passwords / fingerprints or finally accepts correct passwords / fingerprints
  + Finally, data erasure (or at least logic set up for it) must be tested and this will be done by having a user set up and consistent improper access to it (correct user ID, but all wrong password / fingerprint attempts) and when this happens, the microcontroller should flag whether or not data will be erased; this can be done with an LED – a red LED turning on to indicate it will erase data
  + Note that for all 3 Authentication and User Setup test cases above, the peripherals (LCD, keypad, and thumbprint sensor) must have logic behaving related to authentication; flash IC does not need to be a part of this if its interface with the microcontroller is set up

Flash IC Interaction / File Functions

Now that the microcontroller knows how to communicate with peripheral devices and have logic setup for authentication, the microcontroller now needs to be able to interact with data in the corresponding flash ICs depending on authentication and user ID

* Testing
  + Two users can be set up with 4 preset bytes of data ready to be written into the flash IC; one user should only be able to write to one IC and other user should only be able to write to one different IC. Then, the \*\*\* functionality in Peripheral and Communication setup will be tested, but this time, each user should only be able to see data written to its own IC as well as the updated data – this is to ensure that one user cannot accidentally write to or read from a flash IC that does not correspond with the user ID
  + Now that flash IC and user ID correspondence is set up, the ability for a user to interact with the thumb drive via a computer to ensure that file writes, reads etc are set up. This may include one user setting up a text file that it can write to only after it is authorized by the microcontroller via keypad / fingerprint
  + Then, multiple users can attempt the same thing and check whether or not users can only interact with the data bank (flash IC) available after authentication

4.0 Sources Cited:

[1]STMicroelectronics, “GitHub - STMicroelectronics/stm32f4xx-hal-driver: Provides the STM32Cube MCU Component ‘hal\_driver’ of the STM32F4 series.,” GitHub, 2020. Available: https://github.com/STMicroelectronics/stm32f4xx-hal-driver [Accessed: Feb. 22, 2025].

[2]“STM32H7 - Arm Cortex-M7 and Cortex-M4 MCUs (480 MHz) - STMicroelectronics,” Available: www.st.com. <https://www.st.com/en/microcontrollers-microprocessors/stm32h7-series.html> [Accessed: Feb. 22, 2025].

[3]“STM32CubeIDE - STMicroelectronics,” STMicroelectronics, 2024. Available: <https://www.st.com/en/development-tools/stm32cubeide.html#> [Accessed: Feb. 22, 2025].

[4]“STM32CubeMX - STMicroelectronics,” STMicroelectronics. Available: <https://www.st.com/en/development-tools/stm32cubemx.html> [Accessed: Feb. 22, 2025].

[5]“STM32CubeF4 - STMicroelectronics,” STMicroelectronics, 2024. Available: <https://www.st.com/en/embedded-software/stm32cubef4.html> [Accessed: Feb. 22, 2025].

‌

‌

‌

‌

‌

Appendix 1: Software Component Diagram

*A diagram of a company

AI-generated content may be incorrect.*