Software Overview

Year: 2025 Semester: Spring Team: #20 Project: Encrypted USB Drive

Creation Date: ­Jan 30,2025 Last Modified: February 1, 2025

Author: Joshua Wai Email: waij@purdue.edu

Assignment Evaluation: See Rubric on Brightspace Assignment

1.0 Software Overview

The primary software functionalities are:

* UI to display on LCD screen
  + LCD Display Driver: Prompts (Select User, Enter Passcode) and feedback (Unlocked, Access Denied). Uses I2C/SPI for communication.
  + Keypad Input Handler: Debounces inputs using a state machine (with something like 100ms delay to filter noise).
* Authentication
  + Password verification
  + Fingerprint matching
* Security
  + Attempt Counter: Resets to zero on success, increases by 1 on failure, and after 5 failures it will erase data
  + Data encryption
* File System
  + Virtual partitioning: has user specific files as a FAT32 volume when unlocked
  + Read and write data to the USB
    - Setup GPIO pin for communication
    - Using SPI to read/write
* Firmware responsibilities
  + Bootloader: initializes hardware and resets user data when tampering is detected
  + Power management: detects when the USB is unplugged to trigger locked state
  + Polls fingerprint sensor (every 100ms)
  + State machine
  1. User interface management module

The microcontroller initializes the onboard LCD, keypad, and fingerprint sensor on reset. On startup, the system sets the default password for all four users to “0000.” When the USB drive is plugged in, the LCD will display a “Select User” prompt. The user selects their profile by entering 1–4 using the keypad. This module also manages subsequent displays, such as “Enter Passcode” and status messages such as “Unlocked” or error notifications. Use a state machine to debounce input with 100ms delay.

1.2 Authentication and Access Control

Two methods of authentication are supported. First, a passcode entry is handled via the keypad with immediate visual feedback (digits are shown as dots to preserve security). Second, fingerprint authentication is integrated via a sensor that compares the live scan with stored biometric data. If either method validates the user, the firmware unlocks the drive, enabling access to user-specific files. When the user is logged in, they can press a button to update their password or fingerprint.

* 1. Security

To protect data, the firmware maintains an attempt counter for each user. If a user inputs the wrong passcode five times, then the software automatically erases that user’s data and resets the password to 0000. This protects against unauthorized access and brute force attacks. The firmware will also use secure communication protocols for USB data transactions and will use encryption routines such as AES to protect file content when the drive is being used. This will ensure file security.

* 1. File System

The file system uses virtual partitioning to give each user a dedicated FAT32 volume once the device is unlocked. This makes it so that the files remain isolated and secure, and each partition behaves like a standard USB drive volume when accessed by the user. In terms of data operations, this is achieved by setting up specific GPIO pins to facilitate reliable communication between the microcontroller and the USB hardware. Additionally, the firmware utilizes the SPI protocol to handle data transfers. SPI provides a high-speed, full-duplex communication channel, for efficient data transfer.

* 1. Firmware Responsibilities

At startup, the bootloader initializes the hardware components, including the LCD, keypad, fingerprint sensor, and USB interface. It also is programmed to reset user data when tampering is detected if multiple passwords are wrong. The firmware will also detect when the USB is unplugged, and trigger the locked state. It will also poll the fingerprint sensor at intervals of 100 milliseconds. We will use a state machine to dictate the overall behavior of the system which include startup, user selection, authentication (passcode entry/fingerprint scan), unlocked (file access), and password update modes.

2.0 Description of Algorithms

2.1 Input Processing and Debouncing

The firmware continuously polls the keypad for numerical input. To detect the key that’s pressed, two ports of the microcontroller are connected to the rows and columns of the matrix respectively. The port which is connected to the rows of the matrix is configured as an output port, hence making each row logic 0. On the contrary, the port which is connected to the columns is configured as an input port, making the column at logic 1. When a button is pressed, it changes the logic of that column to 0 because the button causes a short circuit between the row and the column. To summarize, when a key is pressed, the column containing the key is driven to logic 0. The microcontroller then checks each row and identifies the row where the button is pressed.

To avoid spurious activations, an input debouncing algorithm ensures that each key press is accurately registered. The microcontroller waits for a certain amount of time and allow the signal to stabilize. It has to also make sure the previous key has been released. The microcontroller scans the matrix after a keypress has been detected. To do this, the microcontroller must make all the rows logic zero and see if all the columns have a logic 1. If all the columns have logic 1, it means that none of the keys are pressed, and the microcontroller looks for a new keypress. See figure below.

2.2 Authentication Decision Logic

The authentication module compares the entered passcode against the stored value or accepts the authentication if the fingerprint match is successful using the built in SDK from the fingerprint sensor. In the event of a failed passcode attempt, a counter increments and triggers a data erasure process once five attempts is reached.

3.0 Description of Data Structures

**3.1 USB Communication Packet (UART)**

For data transfers, we will use s structure packet format:

* Header Byte: identifies the packet type (command, data, acknowledgment)
* Payload: contains the data
* Checksum/error detection

**3.2 Input Buffer and Display Buffer:**

**Keypad Input Buffer:** Temporarily stores numerical inputs until the user submits the passcode.

**Display Buffer:** Holds the current message that is sent to the LCD.

**3.3 User profile structure**

User ID: A unique identifier (1–4).

Passcode: A string or fixed-length array stored in a secure memory area (hashed for extra security)

Fingerprint Template: Data structure or pointer to the biometric template provided by the fingerprint sensor.

Attempt Counter: An integer value tracking consecutive failed passcode entries.

Access Rights/Data Pointer: A pointer or reference to the location of the user’s encrypted file storage.

3.4 Microcontroller Communications

We will be reading and writing using SPI protocols with its SPI running in either of two modes depending on clock polarity (CPOL) and clock phase (CPHA) settings:

CPOL = 0, CPHA = 0 (Mode 0)

CPOL = 1, CPHA = 1 (Mode 3)

Input data is latched in on the rising edge of SCK, and output data is available from the falling edge of SCK for both modes

A line of lines with text

AI-generated content may be incorrect.

*Standard Commands*

A table of command

AI-generated content may be incorrect.

4.0 Sources Cited:

J. Ganssle, "Debouncing, hardware and software, part 2," The Ganssle Group. [Online]. Available: https://www.ganssle.com/debouncing-pt2.htm. [Accessed: Jan. 31, 2025].

Technobyte, "Keypad Matrix and 8051 Interfacing," Technobyte.org, [Online]. Available: https://technobyte.org/keypad-matrix-8051-interfacing/. [Accessed: Jan. 31, 2025].

E. Peña and M. G. Legaspi, "UART: A Hardware Communication Protocol," Analog Dialogue, vol. 54, Dec. 2020. [Online]. Available: https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html. [Accessed: Jan. 30, 2025].

S. Campbell, "Basics of UART Communication," Circuit Basics, Feb. 13, 2016. [Online]. Available: https://www.circuitbasics.com/basics-uart-communication/. [Accessed: Jan. 30, 2025].

Appendix 1: Program Flowcharts

*Main Program*

A diagram of a computer

Description automatically generated

*Keypress and debounce logic*

*A diagram of a algorithm

AI-generated content may be incorrect.*

Appendix 2: State Machine Diagrams

A diagram of a software system

Description automatically generated