# Engineering Report

|  |  |
| --- | --- |
| **Project Title:** | Fingerprint Lockbox/Safe |
| **Prepared by:** | **Joshua Maxi** |
| **Date:** | July 28, 2025 |

A wooden box on a bed

AI-generated content may be incorrect.

**Figure 1**: Outside view of the Lockbox/Safe

## 1. Summary

This engineering documentation outlines the design, implementation, and testing of a secure, standalone fingerprint-based lockbox system. The device uses an ESP32-CAM microcontroller paired with a fingerprint sensor to authenticate users. If an unauthorized fingerprint is scanned, the system automatically captures and stores a timestamped photo on an SD card. It is built for power-conscious environments and includes features such as safe SD card dismounting and time synchronization via Wi-Fi.

## 2. Physical Build

The project utilizes compact and low-cost components for ease of deployment and flexibility:

**ESP32-CAM:** Microcontroller with integrated camera and Wi-Fi.  
  
**Adafruit Fingerprint Sensor:** Used for fingerprint authentication.  
  
**SD Card (1-bit mode):** Stores photos taken by the ESP32-cam, holds microSDs.  
  
**12v Solenoid Lock (GPIO14):** Electromechanical locking mechanism controlled via GPIO.  
  
**NPN PN2222A Transistor:** Drives the solenoid based on fingerprint match.  
  
**Flash LED (GPIO4):** Illuminates when capturing images.  
  
**Push Button (GPIO13):** Used to trigger re-enrollment of fingerprint with authentication.  
  
**12v Power Source:** External power bank which uses 8 1.5v batteries to power the system.  
  
**10k ohm and 1k ohm resistors:** Use to limit the current/voltage and also as a pull up resistor for the transistor.  
  
**Metal Hinges:** To open the wooden door of the Lockbox/Safe  
  
**3mm Thick Plywood and Acrylic:** Creates the lockbox frame and window for the ESP32-CAM to see outside while not being able to physically touch it.  
  
**Diode Rectifier:** Help to ensure parts are undamaged by the 12v solenoid when it activates.  
  
**Lever Nuts:** Simple way to make connections for the prototype design.  
  
**12v-to-5v Buck Converter:** Steps down the 12v from the power supply to a usable 5v for the ESP32-cam, while being a function power supply for the solenoid.

A wooden box with wires and wires

AI-generated content may be incorrect.

**Figure 2**: Inside view of the Lockbox/Safe, showing the divider between electrical and storage space.

## 3. What I Built

This project integrates hardware, firmware, and custom-designed components to build a fully functioning fingerprint-secured lockbox. Keep the project relatively inexpensive while teaching valuables skills based on prototyping.

**Code Files (Arduino IDE / C++)**

* **Main Firmware (FingerprintLockbox.ino)**
  + Manages fingerprint authentication, camera capture, SD card logging, flash control, solenoid locking, Wi-Fi integration.
  + Also implements a feature to authorize who can enroll and new fingerprint on the device. Time stamping to know when an unknown fingerprint was used to try to open the device. SD card unmount system to ensure that the card doesn’t corrupt before turning off the device.
* **Helper Functions**
  + Three functions, enrollFingerprint(), captureAndSavePhoto(), and authorizeEnroll() to maintain clear structure and readability.

**Design Files (Adobe Illustrator)**

* **Laser Cut Enclosure Panels**
  + Designed for laser cutting acrylic or wood using Epilog Legend 36EXT.

A red outline of a square and a square

AI-generated content may be incorrect.

**Figure 3:** Inside panel design of the lockbox to keep electrical components away from the items stored.

A red line drawing of a square and three circles

AI-generated content may be incorrect.

**Figure 4:** Front panel design with cutouts for the ESP32-cam and fingerprint reader, also cutouts for the handle.

A black and white drawing of a sword and shield

AI-generated content may be incorrect.

**Figure 5:** Outside panel design with engrave design and name.

**3D Printing Files (STL)**

* **ESP32-CAM Holder (ESP32Holder.stl)**

A green objects on a grid

AI-generated content may be incorrect.

**Figure 6:** Custom bracket design to secure the ESP32-CAM to the enclosure without obstructing the lens or SD card.

* **Fingerprint Sensor Mount (FingerprintHolder2.0.stl)**

A green rectangular object on a black grid

AI-generated content may be incorrect.

**Figure 7:** Custom bracket design to secure the Arduino Fingerprint reader.

Each component was designed and printed to ensure a secure fit and proper alignment within the final physical design. Using a combination of masking and mounting tape to hold electrical parts in place, to ensure connections. Combined, these hardware and digital elements created a responsive, secure, and user-friendly locking system.

## 4. Skills & Knowledge

### Microcontroller Programming

An assortment of libraries and different data structures were used to create the programming for the ESP32-cam. Using Wi-Fi and global time libraries to track the time at which the camera was used. While having to utilize header files that come with the ESP32-cam board to accurately define pins. Most of the enroll process and the general flow of how hardware interacts with each other was solely created by me. Aspects like the camera configuration and how to use the functions from the Arduino fingerprint scanner came from tutorials and threads of other user experiences.

* **Libraries & Header files:** 
  + Adafruit\_Fingerprint.h: Used for the functions that allow the ESP32-cam to function with the fingerprint scanner.
  + Preferences.h: Allows for storage of fingerprint data, allowing the lockbox to store the enrolled fingerprint while powered off.
  + WiFi.h: Enables the WiFi connectivity on the ESP32-cam.
  + Time.h: Gives access to the current time.
  + Esp\_camera.h: Enables the camera on the ESP32-cam.
  + SD\_MMC.h: Supports the SD mount to be launched in 1-bit mode.
  + FS.h: Performs the basic file uploads to the SD.

## 5. Iterative Process

### Initial Testing

The first set of tests done on the hardware was to test if the power supply would work and how to set up the 12v to 5v buck converter so that my ESP32-cam could be powered remotely. The next step in the process was to learn about the fingerprint scanner and how the receive, transmit, and interpret what the fingerprint scanner sends back to the ESP32-cam. Then I implemented the enroll feature to enable the current user to reset the fingerprint to a different finger if needed. Finally, test the solenoid pin using the NPN transistor to control the logic.

A person's legs with a computer mouse and wires on a table

AI-generated content may be incorrect.

**Figure 8**: Prototype of the lockbox components without the ESP32-cam due to troubleshooting.

A notebook with writing on it

AI-generated content may be incorrect.

**Figure 9**: Notebook with initial design thoughts and revisions.

### Failed Ideas

Originally, the first iteration of the lockbox design utilized an LCD1602 screen to help guide the user through the state logic without having to know how it is set up. This is where my first set up problems came, trying to use the LCD1602 or rewire the SDA and SCL pins would cause the rest of my program to start failing. At first, I thought the issue came from the lack of GPIO pins that are usable when using the camera and SD-card mount functions but after various testing this was not the case.

To help alleviate the GPIO pin issue I tried to implement a structure that would allow the ESP32-cam to connect to the internet and send the files over google drive using the google apps script. This did not fix the issue, and I was unable to get the LCD1602 to work with the ESP32-cam. This failure did not cause me to change any hardware components as my components could function without the LCD, but it did cause a lot of struggles during the prototyping process.

### New Ideas

The loss of the LCD1602 was not ideal but this did allow me to think about new ideas and more user-friendly ways I can apply my hardware. This comes in the form of timestamping the files of when they are taken by using the Wi-Fi, enabling the enroll button with the capability to “eject” the SD-card so that it cannot be corrupted, and having the light on the fingerprint scanner blink slowly to show the user that it is in the state of waiting for a new fingerprint to be enrolled. These ideas were not considered the mid-project report or the project proposal, but the workflow continued new ideas were implemented to help the user.

### Results

All parts of the project work as intended in the project proposal except for the LCD screen to tell the user, but I implemented a way to know which state the lockbox/safe is in. The functionality is swift and easy to use, with the only mildly difficult aspect being pressing the enroll button to reset the fingerprint. The outside of the lockbox is flush with little to no unnecessary extruding parts, and the design incorporates a way to prevent unwanted access through pushing or pulling on the door. While having a storage area to keep the users items away from the electrical components while allowing for a full range of motion on the door.