Portal Box: Acceptance Test and Datasheet

Project Manager: Axel Andrews

Systems & Performance Engineer: James Powick

Design Engineers: Peter Beaudette, Kevin Duong, Aidan Flynn, Colton Jiorle

Client & Supporter: Matt Lamparter

Bucknell Senior Design Phase 4B Check-In

April 23, 2025

**Abstract**

This datasheet contains information related to the acceptance testing of the Portal Box 4. Our Portal Box replicates existing functionality seen in the MakerE with RPi-based Portal Boxes, but with an ESP32-based implementation. Through a connection with a database and webserver, the Portal Box verifies the identity of a user with an RFID card, then determines whether or not to provide power to the machine they are attempting to use. For machines that require power even when not actively running (i.e. laser cutter), the Portal Box uses a USB interlock to prevent or enable the transmission of data to the machine. The four main new features on the Portal Box 4 compared to past Portal Boxes are the ESP32-C6-DevKit running MicroPython, an LCD screen for a clearer and more informative user experience, a keypad for PINs to enable two-factor authentication, and a 3D printed enclosure. The LCD and keypad enable two new special modes for admins - authenticating users from the box with no website interaction, and displaying RFID card numbers so they can be added to the system. The new enclosure offers a simpler, cheaper assembly which utilizes more commonly available makerspace equipment. The web application/MySQL database combination which powers the system (MakerPortal) has been templatized for usage with AWS CloudFormation, allowing a full-stack deployment in less than 20 minutes with zero coding required. All of our code is publicly available on GitHub for both the box software and web application. The documentation package for the project includes the following guides: User Quick Reference, User/Admin Manual, PCB Ordering/Assembly, Enclosure Printing/Preparation, Final Portal Box Assembly, Portal Box Microcontroller Setup, and MakerPortal Setup. Together and with our publicly available GitHub repository containing code and fabrication files, the project succeeds in simplifying assembly from past iterations. Trials have been ongoing at STEAM Innovation Labs in Sunbury in a continued effort to push Portal Boxes out of Bucknell’s doors.

**Architecture**

Our design replicates many elements of the prior Portal Box 2 and 3, with the PCB and code being very similar. The most notable change is the shift to the ESP32-C6 microcontroller, which comes with a change from CircuitPython to MicroPython. This change has significantly decreased the power consumption of our Portal Box, as well as the cost. The ESP32–C6-DevKit that we are using only costs $9, compared to $35 for the Raspberry Pi 4 used in the 3.0. Both of these benefits support our goals of decreasing the power consumption and cost of Portal Boxes, making them more sustainable and affordable for makerspaces to deploy. The keypad and LCD display enable higher levels of security through PIN requirements to use the machine. These peripherals are configurable, and if a builder wishes to make a Portal Box with just the LCD and a button, or just a button, all they have to do is toggle two booleans in the config file. This provides three levels of functionality and price point for the boxes, providing more flexibility and choice for deployers. The web application and database can be deployed as before, with minor changes being made to Tom Egan’s codebase to allow for PINs, or they can be deployed codeless by using AWS CloudFormation. All a builder has to do is upload the template file, input their OAuth client ID, name, email, and password for their database, and CloudFormation builds the entire stack. They get a functional MakerPortal running on an Apache2 server in an AWS EC2 instance, connected to the MySQL database in AWS RDS, in less than 20 minutes. This provides a great option for inexperienced web developers to deploy the MakerPortal and enable the system (Appendix H).

Table 1: Portal Box Inputs and their Descriptions

| **Inputs** | **Description** |
| --- | --- |
| User Card | User insert RFID card into dedicated spot on the Portal Box enclosure |
| User PIN | User inputs their respective personal identification number using the keypad |
| Electrical Receptacle Power | Power coming from the wall outlet in a room, 120 VAC |
| Control Computer | USB data sent via a cable, provides information about how to control certain machines from a corresponding software interface |
| Database | HTTP responses provide information (Appendix D) |

Table 2: Portal Box Outputs and their Descriptions

| **Outputs** | **Description** |
| --- | --- |
| LCD Screen | Displays messages providing user information, time spent on machine, and updates on Portal Box status |
| DotStar LEDs | Emits different color light that depends on the status of the Portal Box |
| Buzzer | Emits different sounds that depends on the status of the Portal Box |
| Machine Power | 120 VAC power delivered to the machine |
| Machine Data | USB data from a computer which contains information and commands for how to control a given machine |
| Database | HTTP requests ask for information from database (Appendix D) |

Table 3: Limits to Current System

| **Limits to the System** | **Description** |
| --- | --- |
| RFID Reading Distance | The maximum distance the RFID reader can read from is 5 cm. |
| LCD Screen Size | The LCD screen only has 2 rows of 16 characters. |
| PCB Size (Smallest Possible Box) | The smallest size of the enclosure is limited to the size of the PCB. |
| Local Storage | The ESP32-C6 only has 4-6 MB of flash memory. |

**Behavior**

The Portal Box cycles through different states according to the inputs it receives to safely and efficiently provide access to machines in a makerspace. The Portal Box is designed to balance access and security in makerspaces while ensuring seamless power delivery. The Portal Box system allows only authorized users to enable power for machines while preventing unauthorized access. The system follows a structured flow: an authentication request is processed, validated, and, if successful, power is granted. If authentication fails, access is denied, and the event is logged for security tracking.

**Known Issues**

* Emailing is not implemented from the Portal Box. In past versions, if a card is left in the machine and the grace period expires due to a machine timeout, the user gets an email to retrieve their card.

**Performance Metrics & Testing Procedures**

**FSM Flow:** The purpose of the Portal Box is to seamlessly flow through different states in order to provide access to a machine. Measuring the performance of its ability to manage each state is imperative to the final portal box.

* **Testing Procedure:** Full end to end testing to reach all states. (Full procedure and results in Appendix)

**Power Delivery Efficiency**: Ensuring stable and quick power delivery is crucial for user experience and system reliability. If power activation is delayed or unstable, it may affect machine performance and user trust.

* **Testing Procedure:**
  + Authenticate an authorized user using a valid card.
  + Measure the time taken for power to stabilize from the moment of authorization.
  + Record voltage levels at the machine’s input terminals.
  + Press the button to end the session.
  + Measure the time taken for power to return to zero from the moment of button press.
  + Repeat the test multiple times to identify any inconsistencies.
* **Expected Metrics:**
  + Power stabilization time should be less than 1 second.
  + Voltage should match the expected supply level within ±5%.

**Reliability**: The system must consistently activate power for authorized users while preventing unauthorized access. Any failures must be minimal.

* **Testing Procedure:**
  + Only use authorized user cards.
  + Track activations over a test period.
  + Log and compare successful vs. failed activations.
* **Expected Metrics:**
  + Success rate of at least 99%.

**Power Consumption**: The Portal Box should operate efficiently without excessive energy consumption.

* **Testing Procedure:**
  + Measure the system’s current draw under normal operation.
  + Compare power consumption to previous versions of the system.
* **Expected Metrics:**
  + The system should draw less than 1.2 A of current.
  + Total power consumption should be lower than prior implementations.

**Startup and Recovery Timing**: The system must start up and recover from faults quickly.

* **Testing Procedure:**
  + Observe and log how the system handles faults.
  + Check if the system returns to normal function upon receiving correct inputs.
* **Expected Metrics:**
  + The system should provide clear error messages or logs.
  + Recovery time after a fault should be under 30 seconds.

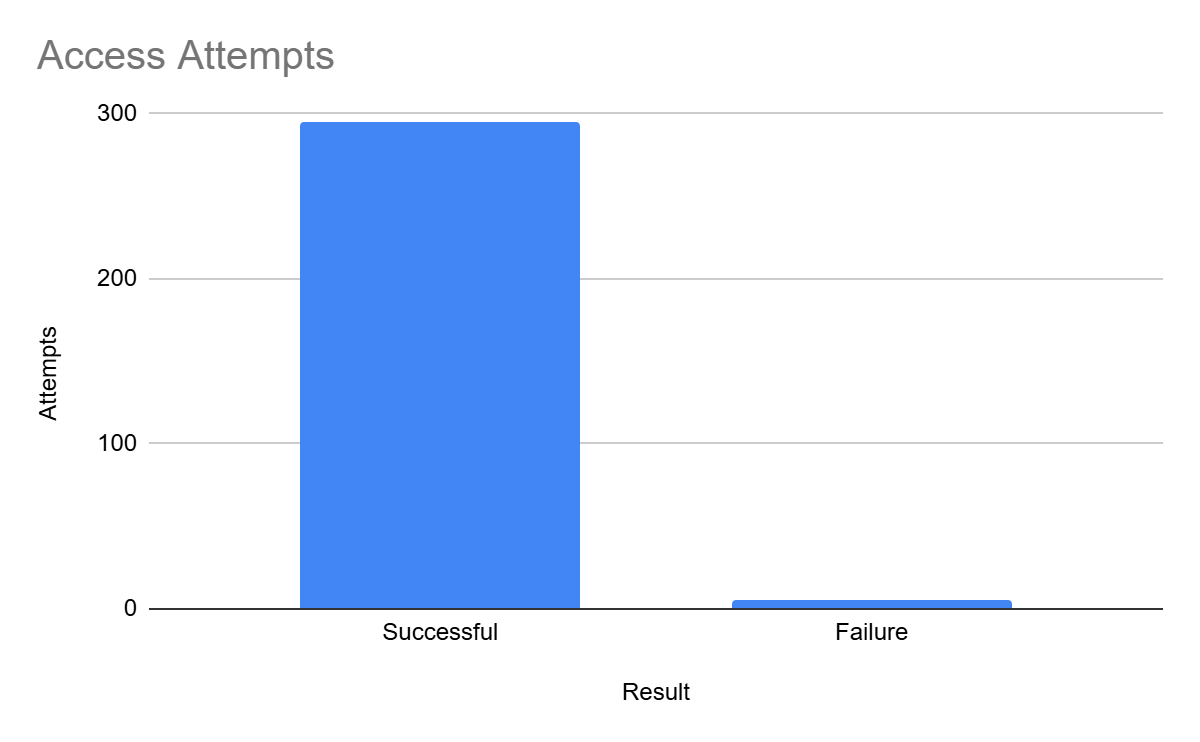
**Performance Data**

**Power Delivery Efficiency**

| **Measurement** | **Value** | **Description** |
| --- | --- | --- |
| RMS Voltage | 125 V | Measured AC voltage |
| Rise Time (0V → 125V) | 560 µs | Time to reach stable voltage |
| Fall Time (125V → 0V) | 65 ms | Time for voltage to decrease to 0 |

Conclusion: The power delivery efficiency test revealed that the Portal Box reaches a stable output of 120V ± 5% in approximately 560 microseconds after being switched on, indicating a rapid response time. Upon switching off, the voltage decay to 0V took approximately 65 milliseconds, suggesting a slower discharge process likely influenced by residual capacitance or inductance in the circuit. These results demonstrate efficient voltage stabilization on power up and a controlled voltage drop on shutdown.

**Reliability**



Conclusion: During the reliability test, we attempted 300 authorization events using the same RFID card over the course of a day. 297 attempts were successful and 3 were unsuccessful. This gives us a success rate of 99%. Failed authorization attempts were due to the RFID card being around 5cm from the RFID scanner.

**Power Consumption**

| **Attempt** | **Current (mA)** | **Power (mW)** |
| --- | --- | --- |
| 1 | 318 | 1590 |
| 2 | 316 | 1580 |
| 3 | 316 | 1580 |
| 4 | 315 | 1575 |
| 5 | 315 | 1575 |
| 6 | 315 | 1575 |
| 7 | 315 | 1575 |
| 8 | 315 | 1575 |
| 9 | 315 | 1575 |
| 10 | 315 | 1575 |

Average value of current: 315.5 mA

Average instantaneous power: 1577.5 mW

Conclusion: Testing the Portal Boxes power consumption revealed that the Portal Box has a very negligible power usage, compared to previous Raspberry Pi based Portal Boxes. This is due to the fact that the ESP32 microcontroller isn’t as power hungry as the RPi. The current draw of Portal Box is less than half that of the RPi version.

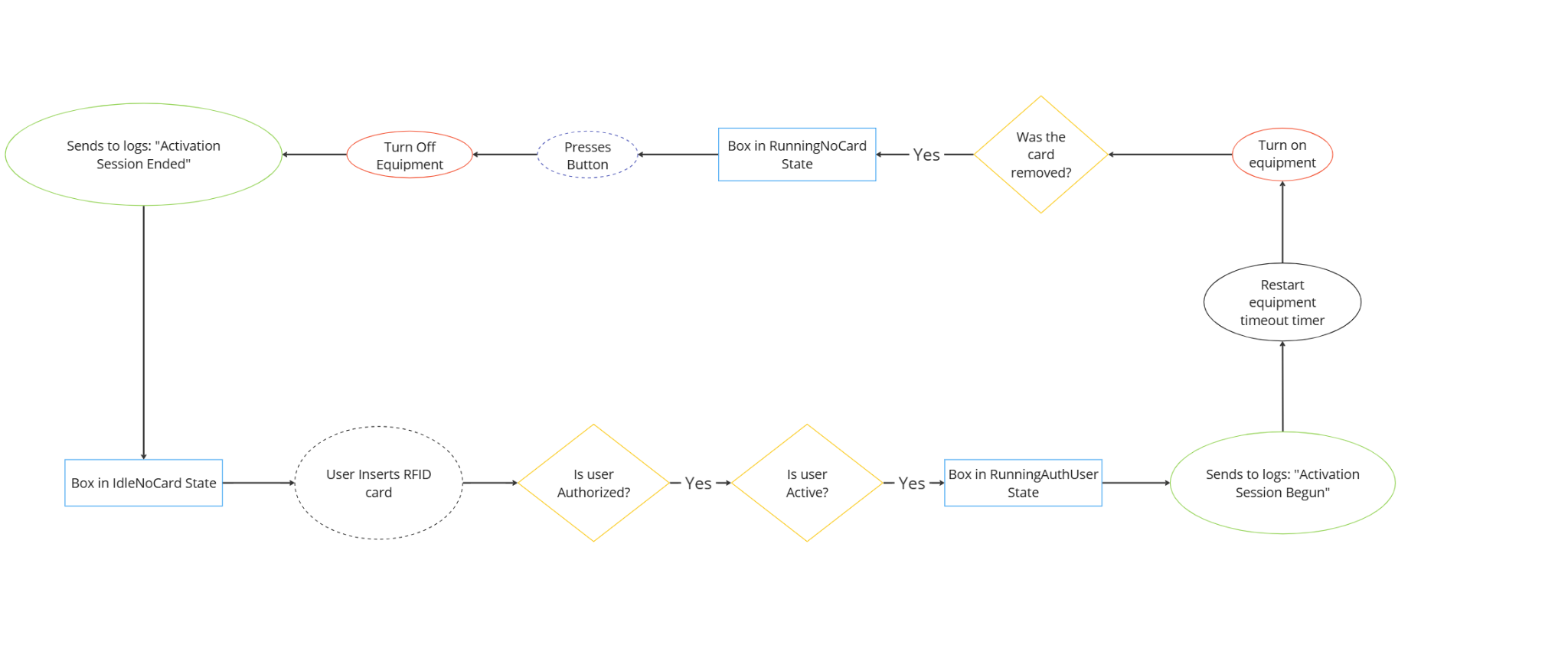
**Startup and Recovery Timing**

| **Fault** | **Expected Recovery** | **Actual Recovery** | **Recovery Time** |
| --- | --- | --- | --- |
| Power Loss (Idle State) | Reboot, PB returns to idle state | Reboot, PB returns to idle state | ~ 13 seconds |
| Power Loss (Running State) | Reboot, PB returns to running state | Reboot, PB returns to running state | ~ 15 seconds |
| Reset Button Pressed | Reboot, PB returns to idle state | Reboot, PB returns to idle state | ~ 15 seconds |
| Shutdown Card Activated | PB enters shutdown state | PB enters shutdown state | < 1 second |

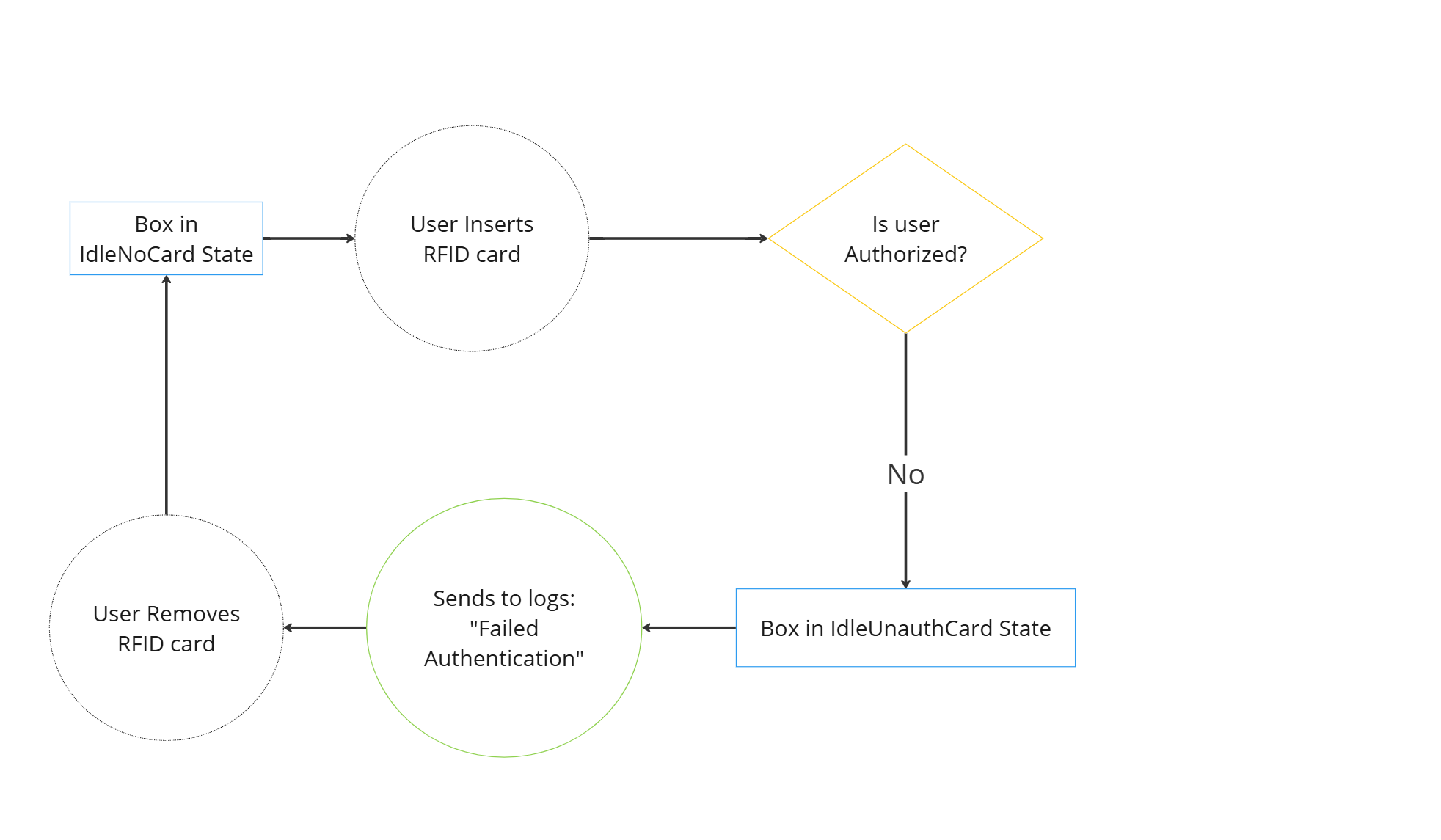
Conclusion: The fault detection and recovery test for the Portal Box demonstrates that the system reliably reboots and returns to its expected state following power loss, reset, or shutdown event. Power loss in both idle and running states resulted in recovery times of 13–15 seconds, while a manual reset required a similar recovery period. Notably, the shutdown card triggered an almost immediate shutdown. All reboot/recovery times are faster than those of RPi based Portal Boxes.

**Appendix**

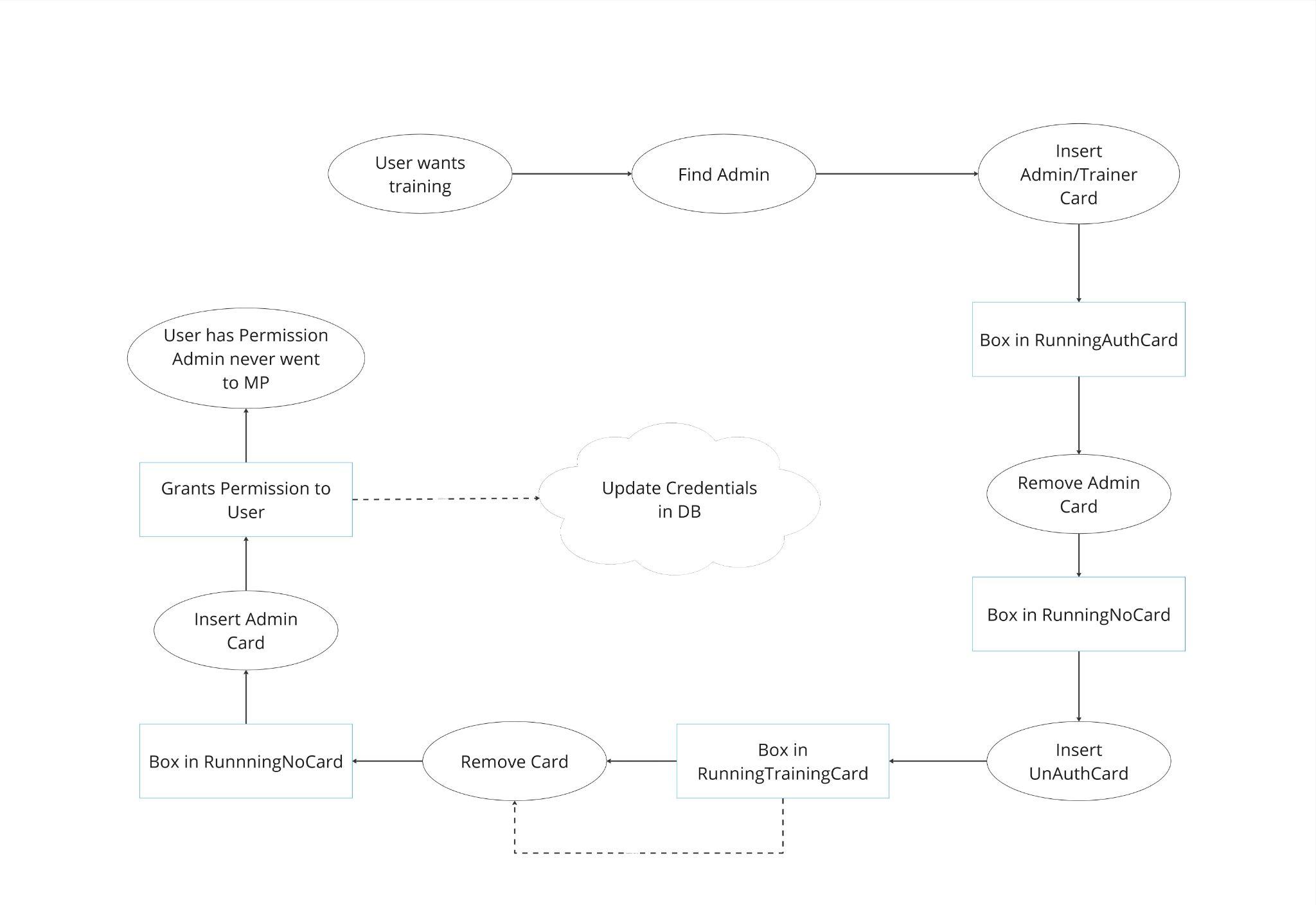
Appendix A: Isolated flow diagram of an authorized user using the Portal Box



Appendix B: Isolated flow diagram of an unauthorized user using the Portal Box



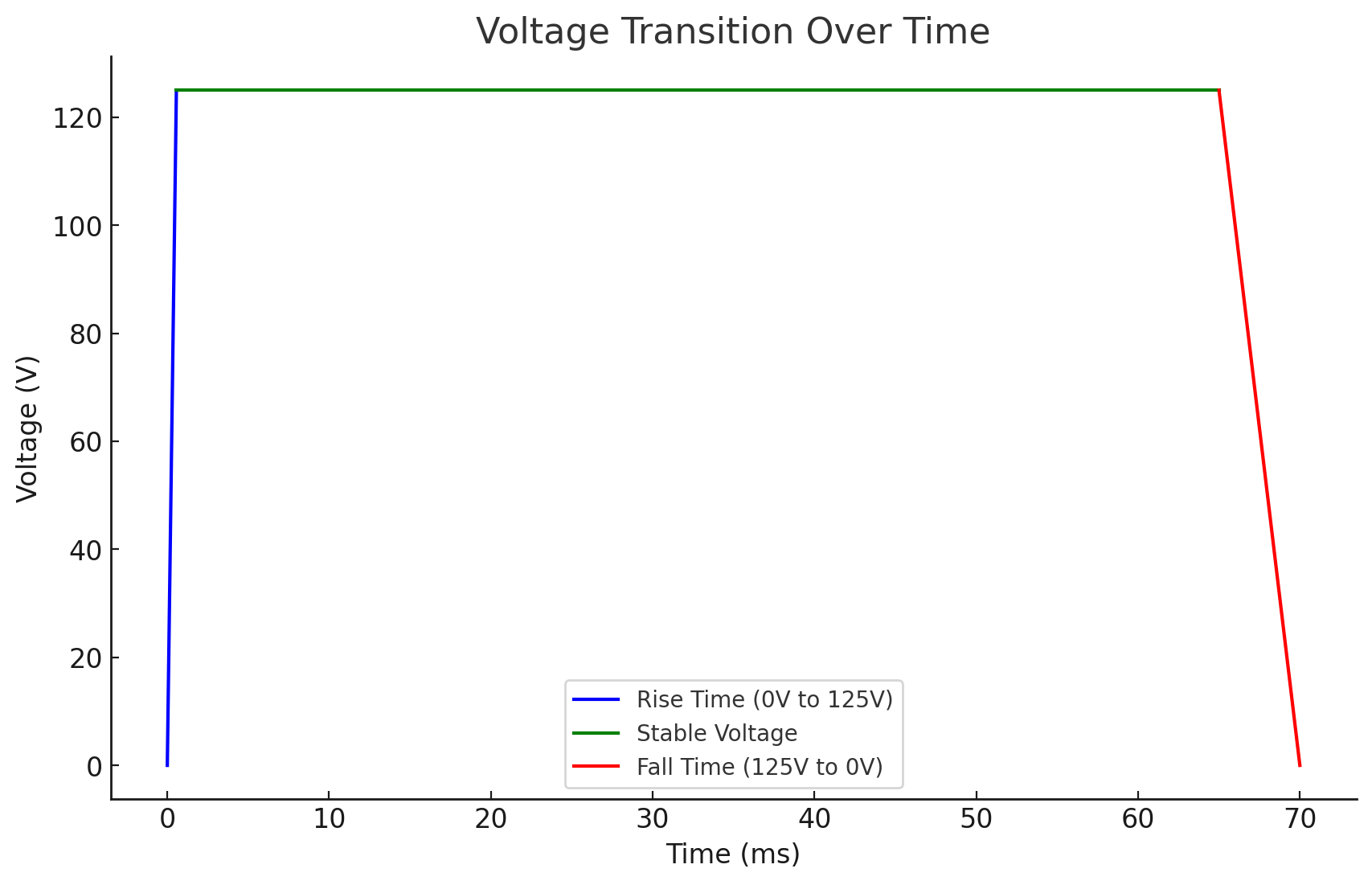
Appendix C: Isolated flow diagram of the new training flow



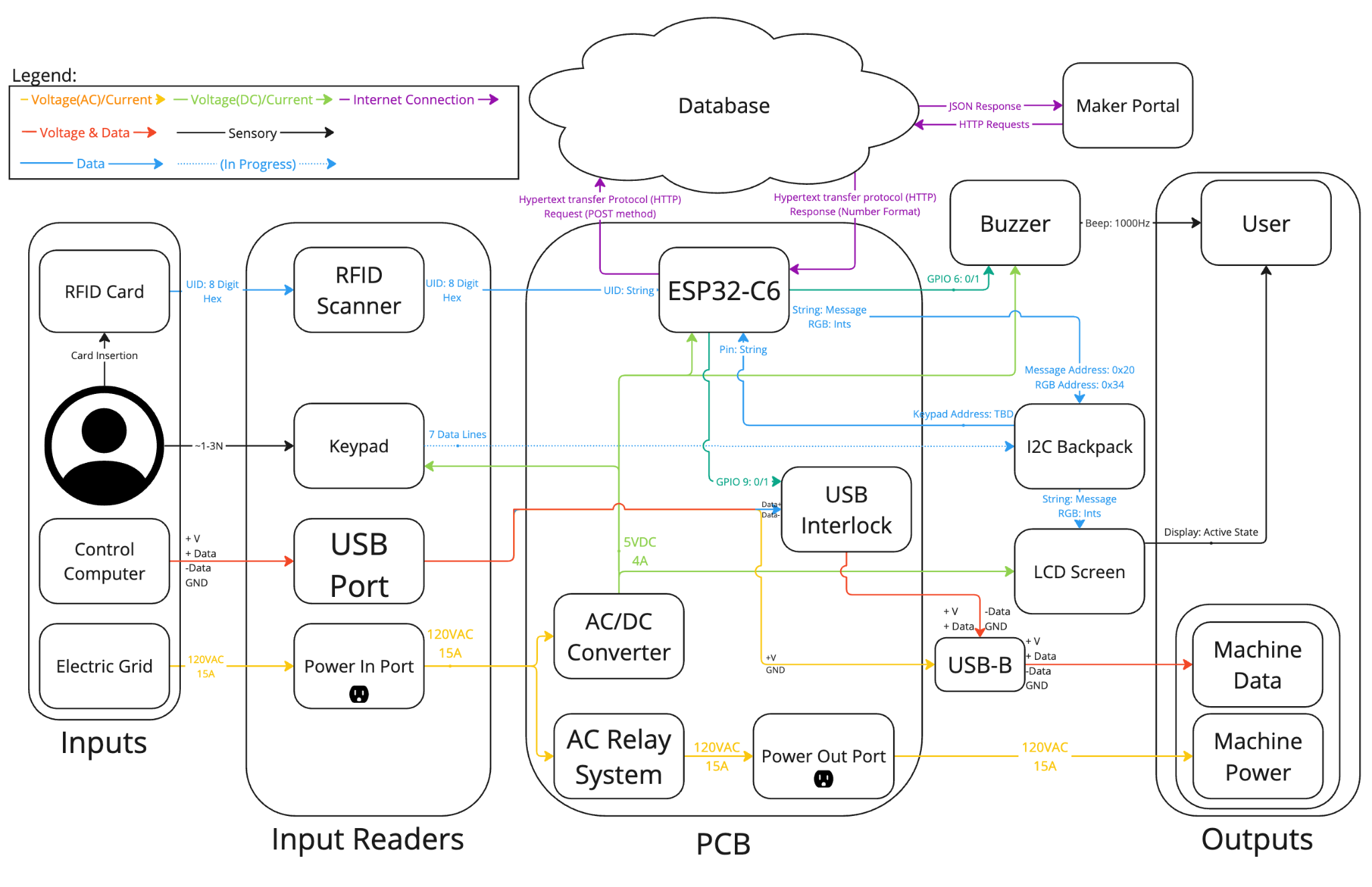
Appendix D: Portal Box API

| **HTTP Method** | **API Mode** | **Parameters** | **Response** |
| --- | --- | --- | --- |
| GET | check\_reg | mac\_adr: MAC address of the device | Integer (1 if registered, 0 if not) |
| PUT | register | mac\_adr: MAC address of the device | Boolean (success status) |
| GET | get\_profile | mac\_adr: MAC address of the device | JSON array with equipment profile: [{"id", "type\_id", "name"[0], "location\_id", "name"[1], "timeout", "allow\_proxy", "requires\_training", "charge\_policy"}] |
| POST | log\_started\_status | equipment\_id: ID of the equipment | Success status |
| POST | log\_shutdown\_status | equipment\_id: ID of the equipment  card\_id: ID of the card used (or 0 if not applicable) | Success status |
| POST | log\_access\_attempt | equipment\_id: ID of the equipment  `card\_id`: ID of the card used  successful: 1 if successful, 0 if not | Success status |
| POST | log\_access\_completion | equipment\_id: ID of the equipment  card\_id: ID of the card used | Success status |
| GET | get\_card\_details | card\_id: ID of the card  equipment\_id: ID of the equipment type | JSON array with card/user details: [{"user\_balance", "user\_auth", "user\_active", "card\_type", "user\_role"}] |
| GET | get\_user | card\_id: ID of the card | JSON array with user details: [{"name", "email"}] |
| GET | get\_equipment\_name | equipment\_id: ID of the equipment | JSON array with equipment name: [{"name"}] |
| POST | record\_ip | equipment\_id: ID of the equipment  ip\_address: IP address of the device | Success status |

Appendix F: Voltage Stabilization Graph



Appendix G: L2 Block Diagram



Appendix H: Cloud Infrastructure

