Edge Vendr: Technical Documentation
Project: Edge Vendr - Snooker Billiard Ticketing Control System
Version: 1.2.0
Date: July 18, 2025

1.0 Overview & Introduction

Edge Vendr is a comprehensive game ticketing and control system designed to digitise and formalise access to physical games such as snooker. The project directly addresses the challenges prevalent in cash-based, informal markets, such as revenue leakage, high maintenance costs of pure mechanical systems, and poor user experience.

This document provides a detailed overview of the system's technical architecture, component design, security protocols, and development process. Our solution is built on the principles of efficiency, security, and scalability, with a strong focus on functionality within resource-constrained environments.

The system consists of three core components:

- **The Smart Electronic Controller Unit (SECU):** The physical hardware that secures the game.
- **The Vending Service**: The cloud-based backend that generates secure one-time tokens.
- **The Mobile Application:** The primary user interface for merchants and players.

2.0 System Architecture

The Edge Vendr ecosystem is designed for robust, semi-offline operation.
The mobile app communicates with our secure Vending Service (API) over the internet to handle payments and generate game tokens.

The token is then delivered to the SECU for validation either manually via a keypad or seamlessly over a direct, offline Bluetooth Low Energy (BLE) connection.

This hybrid architecture ensures that the core gameplay loop (accessing the balls locked up in the snooker table) is not dependent on internet connectivity at the venue.

Vending service / Game merchant systems profile

Over the internet

Mobile app

Bluetooth Low Energy (BLE)

Smart Electronic Controller Unit (SECU)

5V Open or Close signal

Servo or Linear Actuator

3.0 Core Components

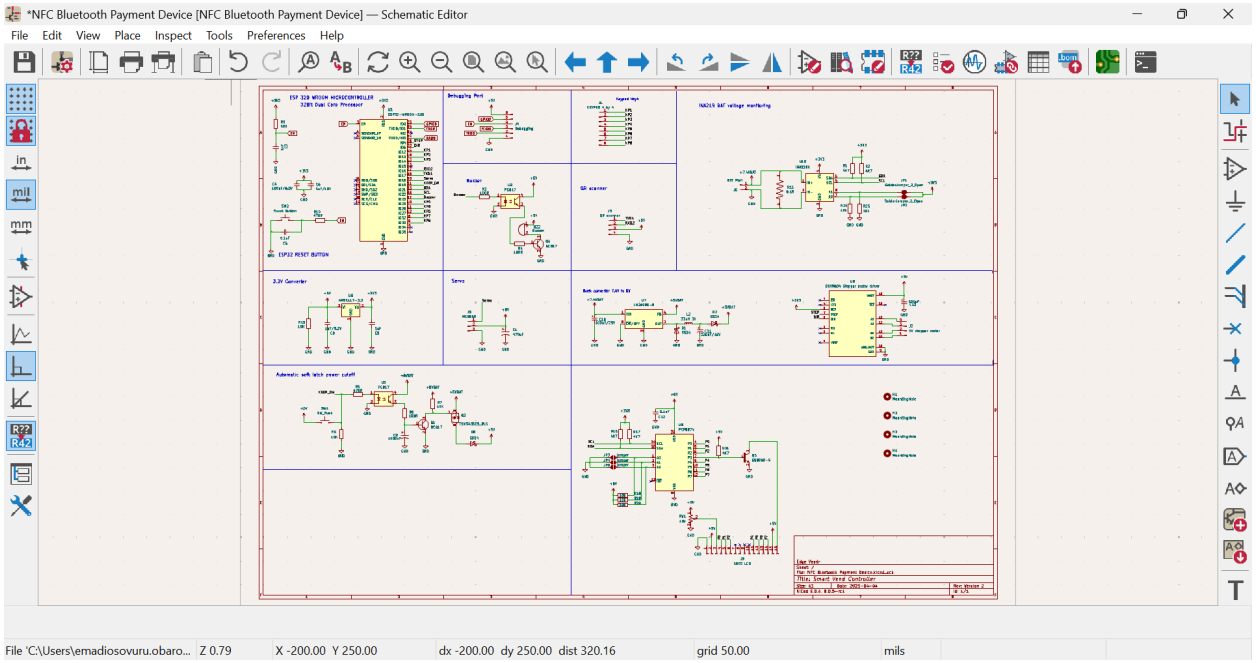
The Smart Electronic Controller Unit (SECU)

The SECU is the heart of our system—an embedded hardware device retrofitted onto the physical game to provide secure, ticketed access control.

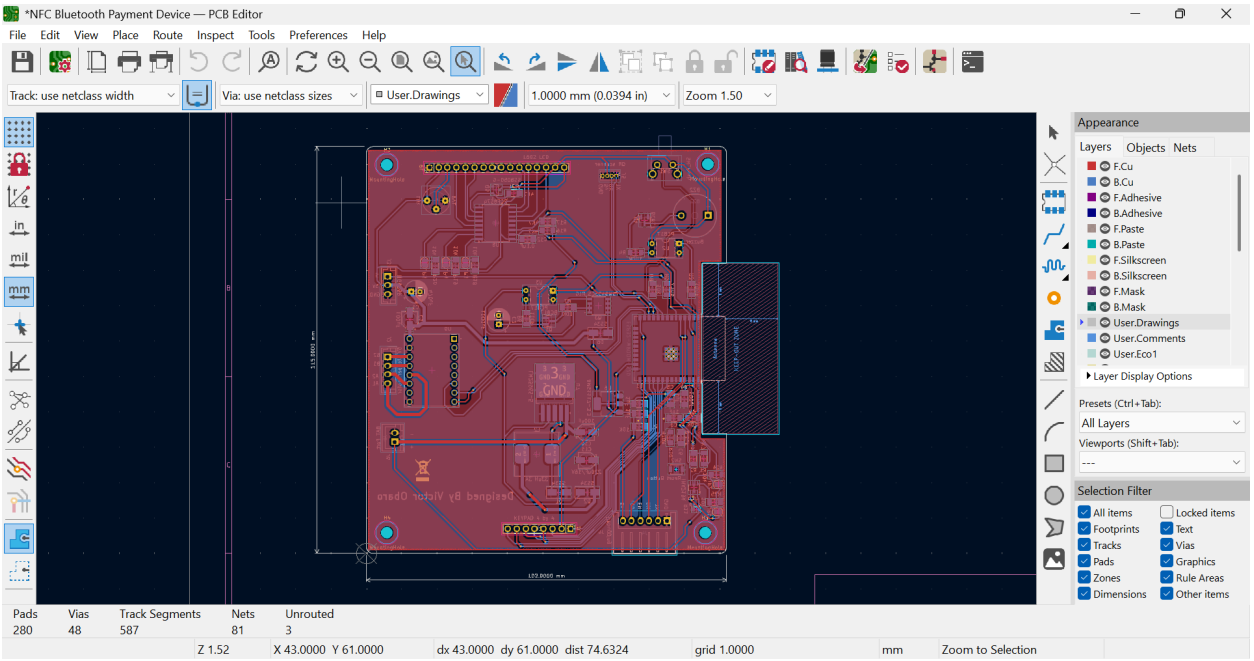
Hardware Specifications

- **Microcontroller:** ESP32 Dual-Core 32-bit MCU.
- **Memory:** 512KB RAM, 4MB Flash.
- **User Interface:** 16x2 LCD with backlight, 12-button keypad, passive buzzer.
- **Connectivity:** Bluetooth Low Energy (BLE).
- **Actuator:** MG996R high-torque servo motor.
- **Power:** 7.4V (8.4V max) 6000mAh Li-ion battery pack (2S 18650).
- **Primary Regulator:** LM2596 DC-DC Buck Converter (7.4V to 5.3V).
- **Secondary Regulator:** AMS1117 LDO Linear Regulator (5.3V to 3.3V).

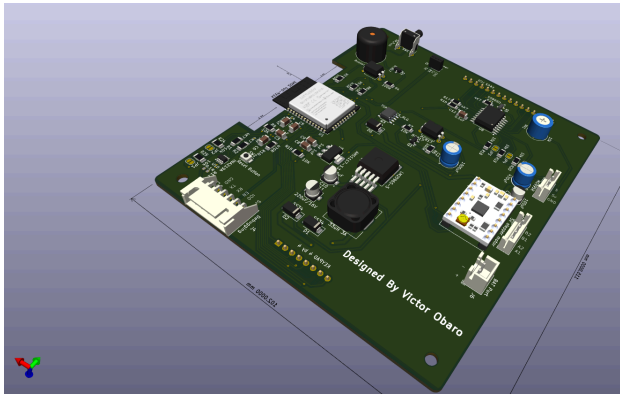
Smart Electronic Controller Unit - KiCAD Schematic layout

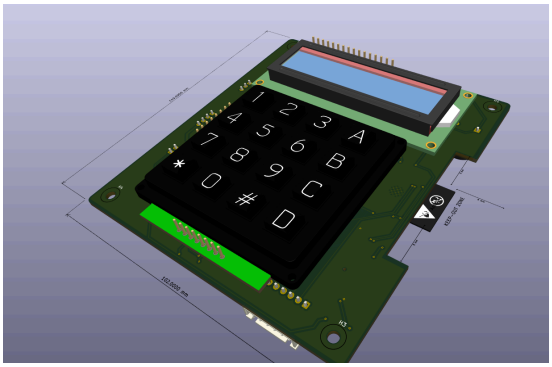


Smart Electronic Controller Unit - KiCAD PCB layout



Smart Electronic Controller Unit - KiCAD 3D view

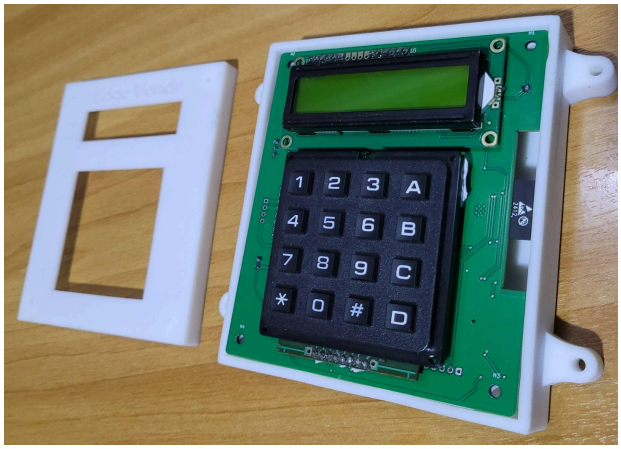


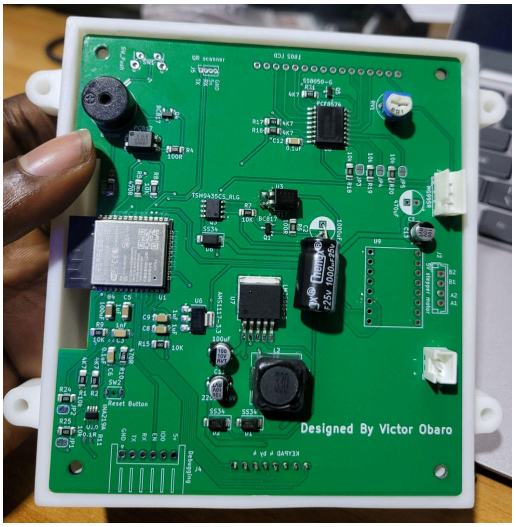


Link to PCB and CAD video:

https://drive.google.com/file/d/1in8p5zkCa-X7aaPHoB6scLgSZE3nBwkC/view?usp=sharing

Manufactured product





Firmware & Offline Operation

The SECU runs a custom firmware built on FreeRTOS using C/C++. This allows for efficient multitasking, managing BLE communications, the LCD driver, keypad input, and controlling the servo motor via GPIO.

The entire operational logic of the SECU is offline, only requiring an internet connection for infrequent Over-the-Air (OTA) firmware updates.

BLE Communication & Callback Mechanism:

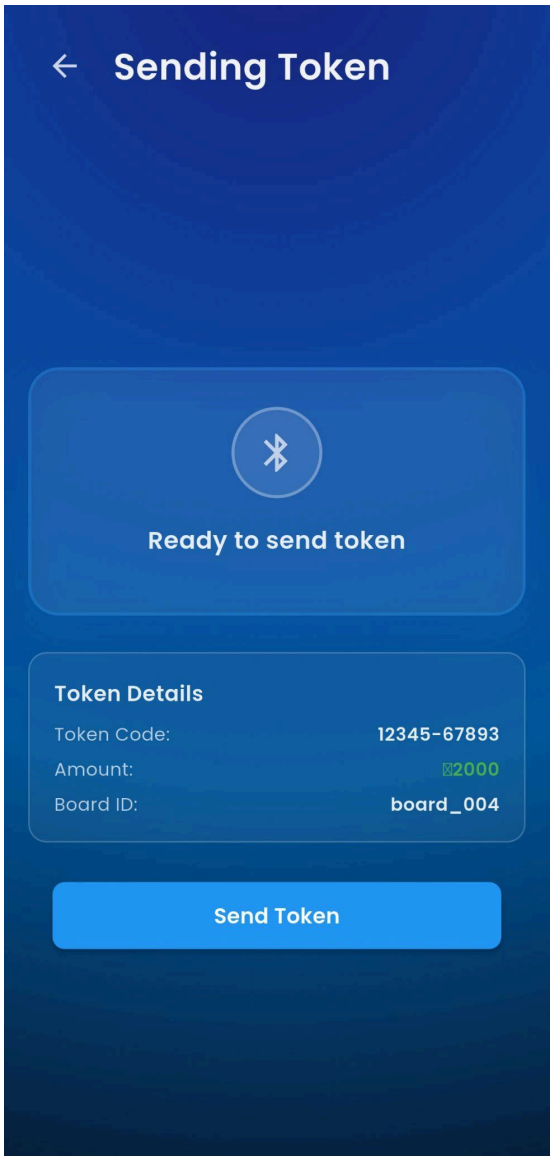
A Bluetooth Low Energy (BLE) service was implemented to achieve a seamless user experience, allowing players to send tokens directly from the mobile app. A JSON-based callback mechanism provides the mobile app with real-time status updates, which are vital for automatically tracking token usage.

```
// Token received by SECU
{"deviceID": "SB-7304250000001",  "type": "token",  "token" : "1234567893"}
```

```
// Response: Success
{"deviceID": "SB-7304250000001",  "status": "success",  "message" : "access granted"}

// Response: Failure (Token already used)
{"deviceID": "SB-7304250000001",  "status": "failed",  "message" : "token used"}

// Response: Failure (Token is not valid for this device)
{"deviceID": "SB-7304250000001",  "status": "failed",  "message" : "token invalid"}
```



Link to BLE Control video: EdgeVendr BLE app control.mp4

4.0 Token System and Security

This system generates secure, short numeric tokens (10-digit strings) that represent a 32-bit encrypted value. These tokens are used for an offline vending system. The system ensures tokens are authentic, have not been tampered with, and cannot be reused. It also allows multiple unique tokens to be generated and validated within the same minute.

4.1 Key Features

- **Fixed-Length Tokens:** 10-digit decimal strings.
- **Offline Validation:** The controller can validate tokens without requiring server communication (after initial key provisioning).
- **Tamper-Proof:** Message Authentication Code (MAC) ensures token integrity.
- **Anti-Replay:** Each token can only be successfully used once.
- **Minute-Collision Resistance:** Multiple unique tokens can be generated and validated for the same timestamp minute.
- **Defined Lifespan:** Tokens are valid up to a 2-year rollover period from a defined epoch.

Token Structure (32-bit Plaintext)

The 32-bit plaintext, before encryption and transposition, is structured as:

Field	Bits	Description
idInMinute	4	Unique identifier (0-15) for tokens in the same TID
TID	24	Timestamp ID (minutes since epoch)
macTag	4	Truncated HMAC-SHA256 tag for authentication
Total	**32**	

- **Bit Order (Big Endian for packing into a 32-bit number):**
  [idInMinute (4 bits)] [TID (24 bits)] [macTag (4 bits)]
  MSB ... LSB

4.2 System Components

Server-Side (Token Generation)

- Responsible for generating tokens.
- Requires ENCRYPTION_KEY_HEX, MAC_KEY_HEX, and AES_FIXED_NONCE_FOR_KEYSTREAM_HEX.
- Uses the current time to calculate TID.

- Can deterministically generate unique idInMinute values if needed.
- Implemented in TypeScript.

4.3 Controller-Side (Token Decryption & Validation)

- Responsible for receiving a 10-digit token string, decrypting it, and validating its authenticity and uniqueness.
- Requires the same keys and nonce as the server.
- Maintains an anti-replay database (UsedTokenDB).
- Implemented in Arduino C++ (guidance provided below).

4.4 Security Considerations & Cryptographic Primitives

- **Keys:**
  - K_ENC_BUFFER (from ENCRYPTION_KEY_HEX): 32-byte key for AES-256-ECB used in keystream generation.
  - K_MAC_BUFFER (from MAC_KEY_HEX): 32-byte key for HMAC-SHA256.
  - **CRITICAL:** These keys are kept secret and securely provisioned onto both the server and each controller. If compromised, the controller's security is broken.
- **AES-ECB for Keystream Generation:**
  - A fixed 16-byte nonce (AES_FIXED_NONCE_BUFFER) is encrypted using AES-256-ECB with K_ENC_BUFFER.
  - The first 4 bytes (32 bits) of the ECB output are used as a keystream.
  - This keystream is XORed with the 32-bit plaintext token to "encrypt" it, and XORed again with the ciphertext to "decrypt". This is a form of a stream cipher.
- **HMAC-SHA256:**
  - Used to generate a Message Authentication Code (MAC) over idInMinute || TID.
  - The full SHA256 HMAC (32 bytes) is truncated to 4 bits (macTag) to fit into the token. This significantly reduces MAC security against brute-force guessing (1 in 16 chance). This is a design trade-off for token length. **A failed MAC attempt on the meter MUST trigger rate-limiting or temporary lockout mechanisms.**
- **Transposition:**
  - A simple byte reversal is used on the 4-byte encrypted block. This is a weak form of obfuscation, but it adds a step to the process. The same reversal is applied during decryption.

5.0 Security & Cryptography in a Constrained Environment

Security is paramount. Our challenge was to implement robust cryptographic security on a constrained 32-bit MCU without sacrificing performance or battery life.
Token Encryption & Integrity:
We developed a proprietary enciphering/deciphering algorithm leveraging well-established cryptographic concepts:
AES-128: Used to encrypt the token payload, providing an excellent balance of security and performance on our 32-bit MCU.
HMAC/CMAC: Used to generate a message authentication code from the SECU's unique secret key, ensuring token integrity and authenticity.
Replay Attack Prevention: The SECU maintains an onboard flash-memory database of all used Token IDs (TIDs) to prevent a valid token from being used more than once.

5.1 Long-Term Security & Key Rollover Mechanism

Flash memory has a finite number of write cycles, and a TID database could theoretically overflow after years of heavy use. To address this, we have implemented a secure key and database rollover mechanism.

Process:
Estimation & Threshold: The SECU's firmware estimates the total number of tokens it can vend before the TID database on the flash memory is full.

Trigger: The backend increments a counter for the total number of tokens generated for a particular device. Once it reaches 98% of the set rollover threshold, vending is restricted for rollover to be carried out. After a successful rollover, the merchant sends a successful feedback.

Code Generation: The Vending Service generates a unique, one-time rollover code. This code is cryptographically signed and contains information to initiate the reset process.

Merchant Action: The merchant is notified and receives this code. They then input this special code directly into the SECU's keypad.

Execution: Upon validating the rollover code, the SECU performs a secure, atomic operation: it erases the TID database, securely clears the old cryptographic keys, and re-initialises itself with a new key set, effectively resetting its 5-year operational lifecycle.
This innovative, in-field solution extends the usable life of the hardware indefinitely without requiring physical servicing or replacement, dramatically enhancing the system's long-term scalability and value.

6.0 Detailed Power Management & Efficiency Analysis

The SECU is engineered for extended operation in locations without constant power.
This final analysis provides our most accurate projection by modelling a worst-case operational scenario that includes converter inefficiencies, battery self-discharge, and the significant power drain from the servo motor's holding torque during an extended inactivity timeout.

6.1 Power Conversion Architecture

The system's two-stage power conversion process is a key factor in our analysis:

- **Primary (LM2596):** 7.4V to 5.3V @ ~80% efficiency.
- **Secondary (AMS1117):** 5.3V to 3.3V @ ~62% efficiency.

6.2 Component Power Profile & Servo States

Component	State	Voltage	Current (mA)	Power (mW)
RISC-V MCU	Deep Sleep	3.3V	~0.15	~0.5
	Active (Idle)	3.3V	~20	66
16x2 LCD	Backlight ON	5V	~25	125
MG996R Servo	Holding	5.3V	~20	106
	Moving (Load)	5.3V	~700	3710

6.3 Energy Calculation per Game Cycle (Worst-Case Scenario)

The actuator does not immediately close and sleep. Instead, it waits for up to 60 seconds of inactivity. To ensure our battery life projection is reliable, we must calculate for the worst-case scenario: a user activates the game and then does not interact with the system again, forcing it to wait the full 60 seconds before sleeping.

Worst-Case Game Activation Cycle Definition (Total Active Time: ~77 seconds)

- **Phase A: Wake & Pre-Validation (10 seconds):** User wakes the system. The servo is holding the mechanism closed.
- **Phase B: Move to Open (3 seconds):** The token is validated. The servo moves to the open position.
- **Phase C:** Stay in open position for (7 seconds)
- **Phase D: Move to Close (3 seconds):** The servo moves back to the closed position.
-
- **Phase E: Hold Close & Inactivity Timeout (60 seconds):** The system is fully active, waiting for keypad input that never comes. The LCD is on, the MCU is active, and critically, the servo is holding the mechanism in the closed position for this entire duration.

Phase E: Finalise & Sleep Prep (1 second): The system finalises its state before deep sleep. The servo is holding.

Power Draw from Battery during each Phase (accounting for converter efficiencies):

- **During Holding Phases (A, C, E):**
    - **Power @ 5.3V Rail:** 106.5mW (ESP32 from 5.3V) + 125mW (LCD) + 106mW (Servo Hold) = **337.5 mW**
    - **Actual Draw from Battery (via LM2596 @ 80%):** 337.5 mW / 0.80 = **~422 mW**
- **During Moving Phases (B, D):**
    - **Power @ 5.3V Rail:** 106.5mW (ESP32) + 125mW (LCD) + 3710mW (Servo Move) = **~3942 mW**
    - **Actual Draw from Battery (via LM2596 @ 80%):** 3942 mW / 0.80 = **~4928 mW**

Energy Consumed Per Game Cycle:

- **Energy (Phase A, 10s Hold):** 422 mW * (10 / 3600 h) = **1.17 mWh**
- **Energy (Phase B, 3s Move):** 4928 mW * (3 / 3600 h) = **4.11 mWh**
- **Energy (Phase C, 60s Hold):** 422 mW * (60 / 3600 h) = **7.03 mWh** <- The single largest energy consumer.
- **Energy (Phase D, 3s Move):** 4928 mW * (3 / 3600 h) = **4.11 mWh**
- **Energy (Phase E, 1s Hold):** 422 mW * (1 / 3600 h) = **0.12 mWh**
- **Total Energy per Game:** 1.17 + 4.11 + 7.03 + 4.11 + 0.12 = **16.54 mWh**

5.4 Final Projected Battery Life

1. Daily Operational Energy (Assumption: 50 Games/Day):

- **Game Energy:** 50 games/day * 16.54 mWh/game = **827 mWh/day**
- **Total Daily Active Time:** 50 games * 77s/game = 3850s ≈ 1.07 hours
- **Total Daily Sleep Time:** 24h - 1.07h ≈ **22.93 hours**
- **Total Sleep Energy:** 1.01 mW (actual sleep draw) * 22.93 h = **23.2 mWh/day**
- Total Operational Drain: 827 + 23.2 = **850.2 mWh/day**

2. Daily Self-Discharge Energy:

- **Battery Capacity:** 44,400 mWh
- **Loss at 3% per month:** ~44.4 mWh/day

3. Total Combined Daily Drain:

- Total Drain = Operational Drain + Self-Discharge Drain
- Total Drain = 850.2 mWh/day + 44.4 mWh/day = 894.6 mWh/day

Final Projection:

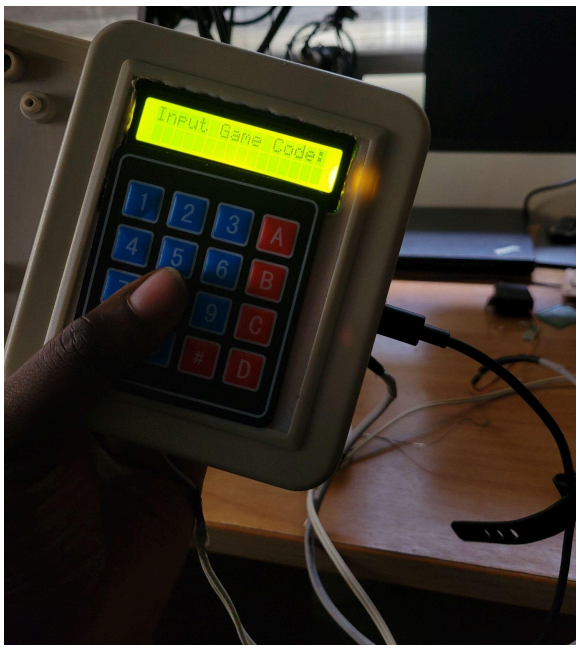
- Total Battery Capacity / Total Combined Daily Drain = Battery Life
- 44,400 mWh / 894.6 mWh/day ≈ **49.6 days**

7.0 Development Journey and Iteration

Proof of Concept (PoC) - 2024

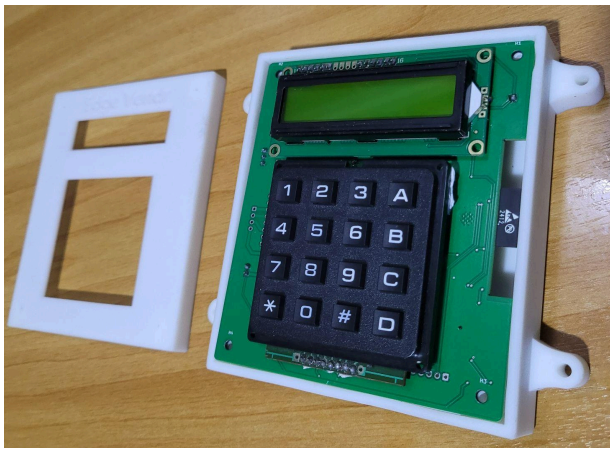
Link to PoC video demo:
https://drive.google.com/file/d/1ir1ZnLV2YqjVHukfVKhljoNtCV2Na32l/view?usp=sharing

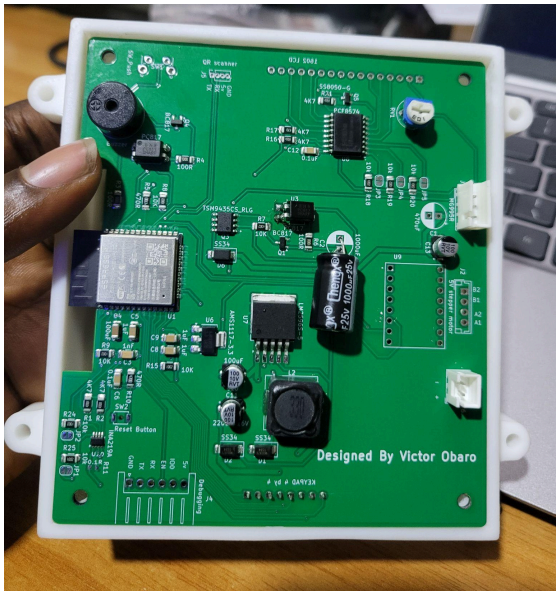
Prototype 1 - April 2025

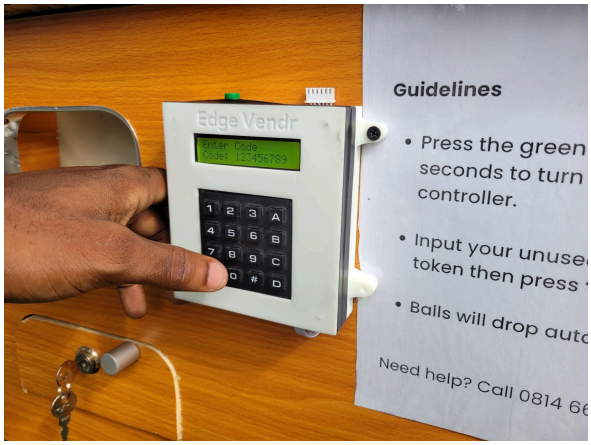


Link to prototype 1 demo:
https://drive.google.com/file/d/1ilFdgFyHDQaeNkaLERNW0P-kfYsfCJuI/view?usp=sharing

Prototype 2 Deployed to a drinking bar for pilot test - June 2025







Link to Prototype 2 demo:
https://drive.google.com/file/d/1ikl_ldPxofSIc0Pd-iqnWs2ijNo-VLsS/view?usp=sharing

Prototype deployment Merchant operation Dashboard

- Onboarded merchant dashboard showing transaction log and total revenue
- Player dashboard showing transaction logs of purchased tokens
- Player dashboard showing purchased tokens/code