ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

Card Game Token Play Aid

Team #39

NINKETH LAKSHMANAN (nikethl2@illinois.edu)
JACKSON PETERIK (peterik3@illinois.edu)
NATHAN SHIN (nsshin2@illinois.edu)

TA: Angquan Yu

September 30, 2024

1 Introduction

1.1 Problem

Playing trading card games such as Magic: The Gathering (MTG) can be fun, but due to the intricacy of these games, players find themselves needing to use improvised items such as spare cards, dice, or paper scraps to represent what is called a “token”. Tokens act as temporary stand-ins for creatures or other game elements that aren't part of a player's physical deck. The sheer variety and number of tokens can be difficult to manage during a game.

Methods used to keep track of tokens are often inconvenient, messy, and prone to causing confusion—especially when players need to track specific game states like power/toughness, abilities, or counters on each token. Furthermore, as games progress and the number of tokens increases, managing the game board becomes tedious.

Our personal experience playing MTG has led us to frequently face these challenges. Not only is managing tokens cumbersome, but it also interrupts the flow of the game. This inspired us to create a hardware-based solution—a digital token display that streamlines gameplay and reduces the physical clutter on the table

Tokens are integral to the gameplay in *Magic: The Gathering* and similar games. Often, they have different power levels, abilities, and counters, which can change during a game. Tracking all of this manually can lead to errors, slow gameplay, and detract from the overall experience.

While a mobile app could solve part of this problem by displaying token images, it is not a perfect solution. Using an app would tie up a player's phone, and since games can last up to an hour or more, this may be impractical. Phones are often needed for other purposes, such as checking messages, using timers, or referencing rules. Constantly switching between these functions during gameplay would disrupt the flow of the game. In a game where the board state should be visible at all times, picking up your phone for a rule check would mean opponents would be unable to see what your true board state is. A dedicated hardware solution avoids these issues by freeing up the player's phone and providing a specialized, easy-to-use interface tailored for gameplay.

Moreover, hardware allows for faster, more intuitive interactions—such as adding or removing tokens or updating their statuses in real-time—without the hassle of navigating through an app during gameplay.

1.2 Solution

The Tokenizer is a hardware-based solution designed to streamline token management in trading card games like *Magic: The Gathering* by providing a dedicated, digital display for tokens. The device will feature a card-sized screen that dynamically displays token images and their associated attributes, such as power/toughness, counters, and abilities. This display allows players to manage tokens in real-time using simple physical buttons, making it easy to add or remove tokens, modify their attributes, and track game state changes without disrupting the flow of gameplay. By using a specialized device instead of improvised methods, the Tokenizer ensures accurate, organized, and efficient token tracking throughout the game.

The device is designed with practicality and ease of use in mind. It will feature a LCD screen display, physical buttons for real-time interaction, and a rechargeable battery for long gaming sessions. Players can manage tokens quickly and intuitively, even as game states become more complex, with no need to rely on makeshift tokens or smartphone apps that can be cumbersome and disruptive. The Tokenizer will maintain a clean, clutter-free gaming environment, allowing players to focus on strategy rather than worrying about managing various makeshift items to represent tokens.

In addition to the hardware, the Tokenizer will be paired with a companion mobile app, which enables users to upload token images and data directly to the device via USB-C. The app will integrate with online databases, like Scryfall, to provide access to official token art and game data, ensuring that the device stays up-to-date with the latest token types. This seamless interaction between the app and the device enhances the overall user experience, giving players a fast, reliable, and aesthetically pleasing way to manage tokens during gameplay, and ensuring that they never need to compromise on accuracy or convenience again.

1.3 Visual Aid

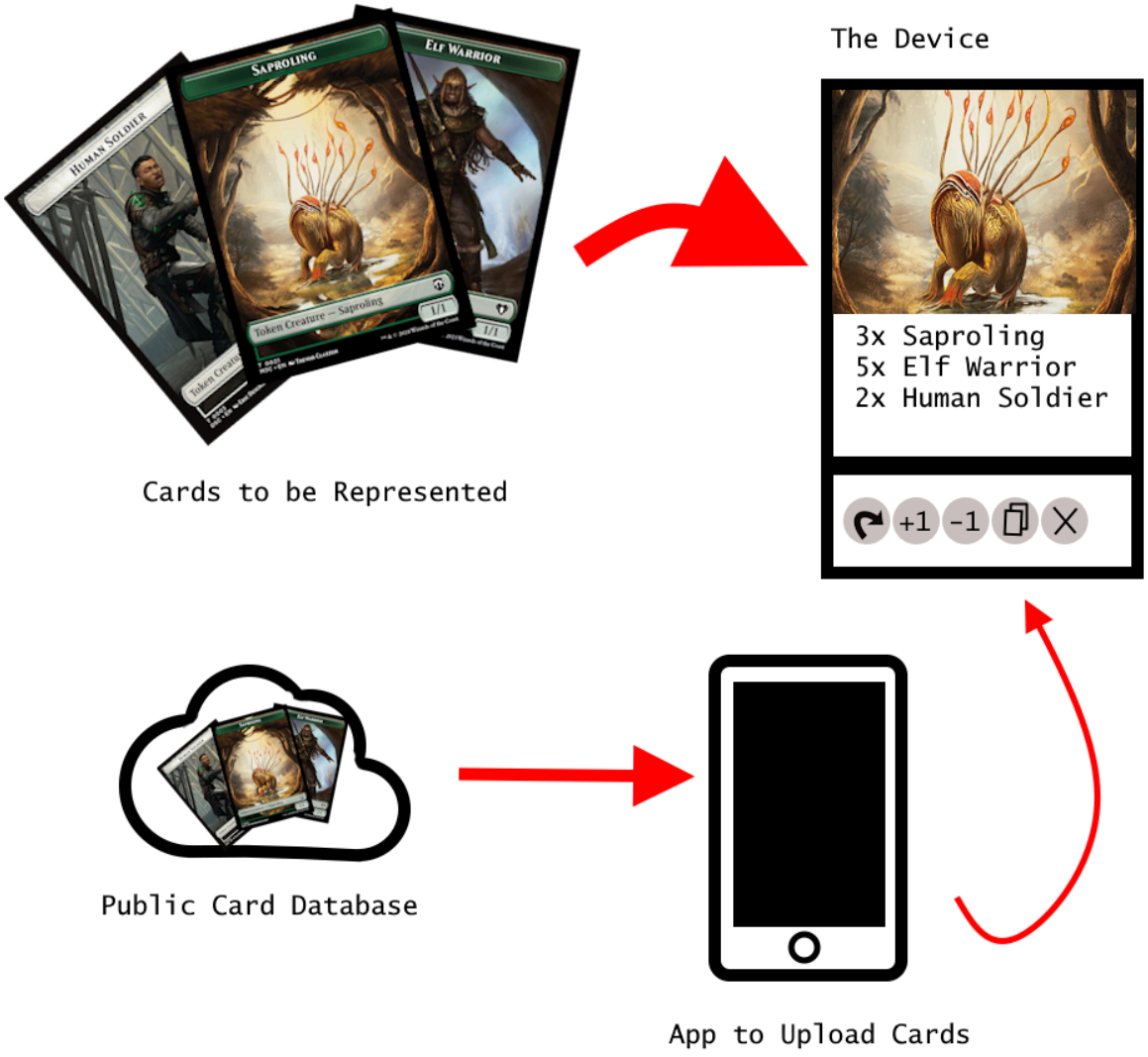


Figure 1: Board State With Device (Located at the Top)

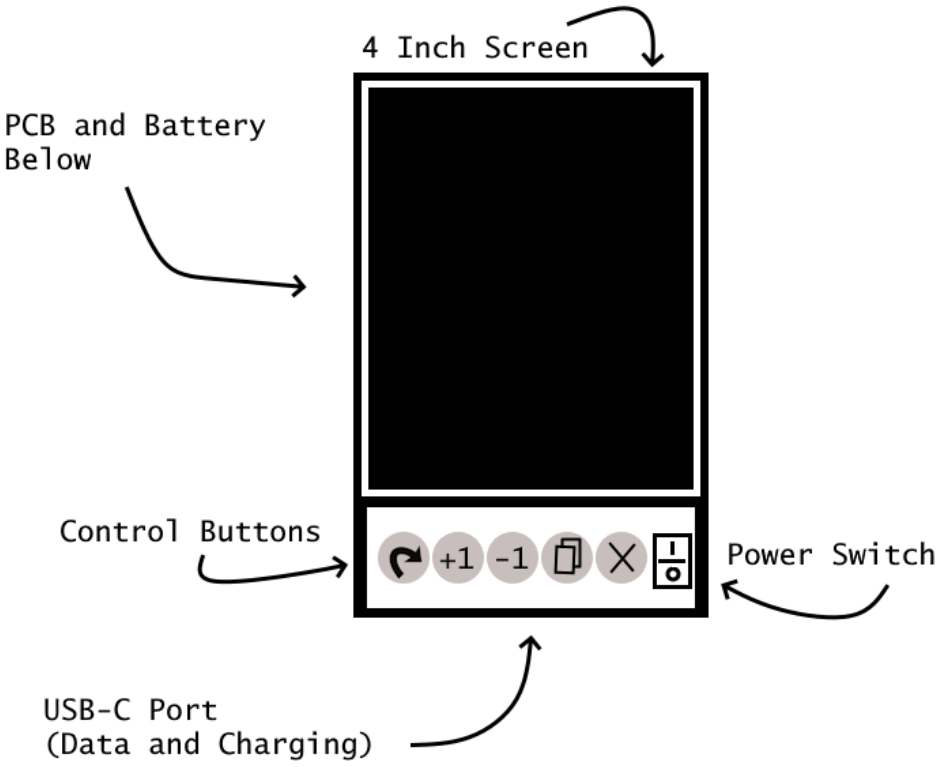


Figure 2: Tokenizer General Layout

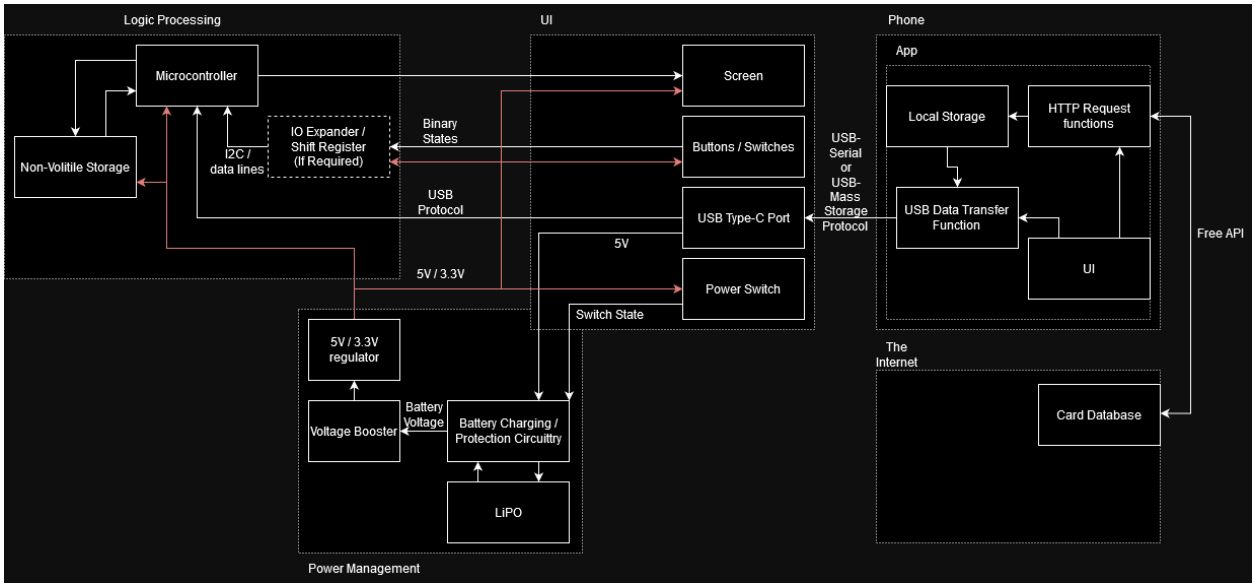
1.4 High Level Requirements

1. **Physical Constraints**:
    - The screen should be at least 4.2 inches to display tokens clearly.
    - Token images and associated data (e.g., power/toughness, counters) should be legible from a distance of at least 60 centimeters.
    - The device should not exceed 100mm x 70mm x 10mm in size, making it compact and portable.
2. **Hardware Features:**
    - The device should process user inputs (e.g., adding/removing tokens, updating attributes) with a response time of less than 1 second.
    - The device should provide at least 4 hours of continuous use on a single charge.
    - The device requires enough storage space for at least 10 different token types at once.
    - Each token type should support up to 16 unique variations, with the ability to manage up to 255 identical copies of each variation.
3. **Data Transferability**:
    - The device should support efficient data transfer through a USB-C device, allowing 10 unique tokens with their attributes to be uploaded within 2 minutes.
    - The mobile app should allow users to search for and select tokens from online databases in under 30 seconds.

These requirements focus on the overall performance, usability, and functionality of the Tokenizer, ensuring that it effectively meets the needs of players in managing tokens during gameplay.

2 Design

2.1 Block Diagram



2.2 Subsystem Overview

2.2.1 Microcontroller Subsystem

The microcontroller is the "brain" of the device, handling data processing and communication between subsystems:

- Microprocessor: This component manages the input from buttons, displays the correct token images on the screen, and communicates with the mobile app for data transfers. It will have built-in USB communication capabilities to enable data transfer between the device and a phone.
- Memory: Extra RAM or flash memory will be used to store multiple token images and any real-time updates, such as token state changes during gameplay.
- USB-C Communication: The microcontroller will also handle data transmission over the USB-C port, allowing users to upload token images and data from the app seamlessly.
- Interactions: The microcontroller interacts with most of the components in the User Interface. Namely, it is responsible for gathering input to the screen, buttons, and power switch to control the entirety of the device. The contents of the screen will be dictated by the token data that is stored within the storage of the microcontroller, and the buttons will be used to change, add, remove, or apply status to the tokens and reflect those updates to the screen. It will also control when the power is turned on or off.

Microcontroller Subsystem Requirements

The way the microcontroller must function is that it must receive data from a phone via a usb-c device or through a non-volatile storage device (in the case of many tokens being saved) while also being able to send that data to the screen. An IO expander and/or shift register may be required for button and switch functionality, but in either case the microcontroller must be able to control these functionalities. Additionally, to save data, the microcontroller must also have the ability to send data to a non-volatile storage device.

Although the image of the token does not have to instantaneously appear, it should be relatively fast, as a slow upload time would defeat the purpose of the design. A reasonable requirement is that the microcontroller must be able to print to the screen within 1 second. For the button controls however, the delay must be very minimal, so an approximate ~250 ms is reasonable for this.

To verify the functionality of the microcontroller, power must be running through the controller proving it is actually being used, then an arbitrary token should be selected to upload. A simple stopwatch time can be used to record how long it takes for the microcontroller to print to the screen. The button requirement is essentially fulfilled as long as inputs do not feel extremely delayed.

2.2.2 Power Subsystem

This subsystem ensures the device remains powered efficiently during long gaming sessions:

- Battery: A flat LiPO battery will provide sufficient power for several hours of gameplay.
- Battery Management System (BMS): This component manages charging and power distribution, ensuring the battery remains healthy and efficiently charges via a USB-C connection. This may end up being two circuits, a battery protection circuit and a charging circuit.
- Power Regulation: A buck-boost converter and voltage regulator will ensure stable voltage to all components, even as the battery depletes, preventing any disruptions during use.
- Interactions: The power subsystem acts in conjunction to all components that require power, which includes every subsystem except the App Subsystem. The power subsystem is responsible for allowing the microcontroller, buttons, power switch, and screen to operate. Although all these components require power, the vast majority of the supply will be allocated to the microcontroller and the screen.

Power Subsystem Requirements

The power subsystem essentially needs to not only be able to power various components of the system but also sustain that constant supply of power for a considerable amount of time (at least for the duration of one Magic the Gathering game). Finally, it must also have a battery that can

charge and recharge, giving it an aspect of portability by not being required to be plugged in at all times.

The power subsystem needs to be able to provide 5.0V or 3.3V of power to each of the components it is powering, depending on the voltage level that is required for the specific device. Additionally, the power subsystem needs to be able to be charged via a usb-c cable at 5.0V of power.

The verification for the power subsystem is quite simple. An observation of the screen displaying light is enough to know that it is providing power to the screen, regardless of whether the output on the screen is correct or not. If the screen is able to be changed in any form as represented through the screen, then it is known that the microcontroller is also working (once again regardless whether the changes are correct or not).

2.2.3 User Interface Subsystem

The User Interface subsystem is responsible for user interactions and displaying information:

- Buttons: Physical buttons that allow players to interact with the device, such as adding or removing tokens, updating their attributes, or scrolling through different token displays.
- Switches: A power switch, as well as a switch to enable the editing of all identical tokens simultaneously.
- Display Screen: A 4.2-inch E-Ink or LCD screen that shows the token images and associated statuses (such as counters or abilities). E-Ink is considered for its low power consumption, making it ideal for prolonged use during gameplay. LCD would allow for a full color image, but would increase power consumption.
- Interactions: The user subsystem works mainly with the logic processing along with the UI as a medium for transfer. The user interface acts as an output for the microcontroller, changing based on how the microcontroller functions. Since the usb-c receiver will be present in the UI, it acts as a median of transfer from the app to the storage of the microcontroller.

User Interface Subsystem Requirements

The user interface functions as the front end of the entire system that is controlled by the user. With the subsystem, the user can interact with the interface as well as see the changes on the screen as a result of those interactions. For this to happen, all components within the subsystem need to be functional and work together.

As for specific requirements, firstly, the screen needs to take input from the microcontroller to know which card to display. Then, the buttons must be able to do the add/remove tokens, update the status of tokens, and scroll/select from tokens that are currently saved, and of course the screen should update accordingly to button inputs. The power switch needs to be able to turn the

entire system on or off, and finally the usb-c port needs to both charge the system and receive data from the app subsystem.

To verify that this subsystem works, a “test game” can be run where each UI functionality is used. Starting with the power button, if the screen can turn on and off, then it works. Next is the uploading, if cards can be noticeably added to the storage, then it is functional. Additionally, there needs to be an instance where the battery charge depletes, so that the charging functionality can be verified as working. Finally, each button input needs to be tested to ensure it is doing the correct thing, which includes adding/removing tokens, updating statuses of tokens, and selecting through the currently saved tokens.

2.2.4 App Subsystem

The mobile app will interface with online card databases to retrieve high-quality card images and corresponding game data. Users will be able to select tokens from these databases and upload them to the device via USB-C.

- UI: The UI will allow users to select images from their phone or search cards online, and add them to the local device to be sent over.
- USB Serial Communication: The app will send all card data from the phone to the device over a Serial USB connection.
- API Access: The app will connect with online public databases to get official art for tokens if wanted, such as Scryfall for Magic.
- Interactions: The app subsystem works mainly with the logic processing and the UI as a medium for transfer. This subsystem functions as the data source for unique tokens, as the tokens will be retrieved from an internet database, and then transferred to the device via usb-c.

App Subsystem Requirements

The app subsystem allows users to search for, select, and upload game tokens to the logic processing subsystem. This requires the app to retrieve images and game data from online card databases, enabling users to interact with a dynamic interface where they can search for token images, upload custom images, and send data to the device via a usb-c connection. To be functional, the subsystem must establish a reliable connection to online databases, facilitate efficient file transfers, and provide a user-friendly experience.
Firstly, the user interface must allow for navigation, including the ability to browse or search for tokens and upload them to the device. The app must also support usb serial communication to transfer selected images and associated game data from the phone to the device using a usb-c connection. Then, API access is required to connect to public card databases such as Magic the

Gathering, ensuring high-quality official artwork and game data is available for users to import into the system.

For verification of the subsystem, the UI functionality can be user tested, ensuring that token search, selection, and upload features work within the app. USB serial communication can be tested by simulating data transfers between the app and microcontroller, confirming successful uploads without data corruption. Finally, API access can be validated by connecting the app to various online databases, retrieving data, and confirming that the correct images and information are imported into the app.

2.5 Tolerance Analysis

In order for the project to meet all of the subsystem requirements, certain mathematical precision checks would need to be put in place. Namely, analysis on the battery life to ensure that it can last for an entire game duration.

Having the battery last 4 hours may be difficult, considering the limited space available for a battery, and the current draw of a screen. The estimated battery life (in hours) of the device can be calculated using these equations

$$T = \frac{C}{I}, I = \frac{P}{V}$$

V = voltage of battery (V), P = total power consumed(mW), I = current(mA), C = battery capacity(mAh)

A battery of a size that would fit beneath the screen is rated at about 3.7V and 1400mAh, or 5.18 Wh.

In order to approximate the total power consumed, the sum of power used by each device must be added.

$$P_{\square} = P_1 + P_2 + P_3$$

P_1 = power used by screen, P_2 = power used by logic processing, P_3 = power used by buttons. (All in mW)

P_1 and P_2 can be calculated via the equations, respectively.

$$P_1 = k \cdot W \cdot H \cdot f, P_2 = V \cdot I$$

k = power drawn per pixel update (mW), W = width of screen in pixels, H = height of screen in pixels, f = refresh rate (Hz). V = voltage rate(V), I = current drawn (mA)

P_3, although part of the equation, would likely be negligible as power drawn from buttons is very minimal.

An E-ink display on the high end consumes about .5μW of power per pixel update and has a refresh rate of about 15 Hz, calculating the power usage with the screen size needed, this value is derived

$$P_1 = 0.0005 \cdot 400 \cdot 300 \cdot 15$$

Which approximates P_1 at about 900mW. P_2 can be calculated by assuming a typical microcontroller current of 50mA and 5.0V rate.

$$P_2 = 50 \cdot 5.0$$

Which approximates P_2 at about 250mW. Going back to the initial equation, we can calculate the estimated battery life.

$$T = \frac{1400}{\frac{1150}{3.7}}$$

The equation evaluates to approximately 4.504 hours, which is slightly more than the subsystem requirement of 4 hours. While these calculations are not tested values, a 100% tolerance is more than enough to feel safe in saying the project is feasible.

3 Ethics and Safety

3.1 Ethical Concerns

The ethical concerns of this project involve privacy. The user can gather data from any public database, and this could potentially lead to private information being leaked. Another is that the app must be sure to not collect any personal information of the user. To address the first issue, we need to ensure our project can not collect data from databases that are not trustworthy. As for the second issue, there will simply be no account feature for the app, and as a result personal information will not be collected.

3.2 Safety Concerns

Many of the safety concerns are the possibility of any part of the system malfunctioning and possibly electrocuting the user or exploding. To minimize this risk, the system should be well enclosed, guaranteeing that if a part becomes damaged that it will not cause harm to the user. Battery charging is managed by a well tested and trusted charging IC, and we are also using a

battery with a built-in protection circuitry, to ensure that the battery is safe. The case will be designed in such a way to prevent movement of components, in order to protect from shorts.

Reference

"ACM Code of Ethics and Professional Conduct." Code of Ethics, Association for Computing Machinery, n.d., https://www.acm.org/code-of-ethics.

"IEEE Code of Ethics." IEEE, Institute of Electrical and Electronics Engineers, n.d., https://www.ieee.org/about/corporate/governance/p7-8.html.

Scryfall. https://scryfall.com/.