McGill University

ECSE 458 Capstone Design Project

Over the Air Update in Trusted Execution Environment: A secure solution for STM32H5

April 10, 2024

Group 33

Supervisor: Prof. Zeljko Zilic

Alex Wei	Chenyi Xu	Fengqi Zhang	Zhanyue Zhang
260981800	260948311	260963858	260944809
yihui.wei@mail.mcgill.ca	chenyi.xu@mail.mcgill.ca	fengqi.zhang@mail.mcgill.ca	zhanyue.zhang@mail.mcgill.ca

Contents

Abstract

Our project's objective is to develop a Trusted Execution Environment (TEE) supported Over-the-air Update (OTA) system for the STM32H573IIK3Q (abbreviated as H5 hereinafter) microcontroller. Our team of four, having worked together in the ECSE 444 microprocessor course, discovered a shared interest in this field through our lab work with STM32L471 microcontroller. This led us to selecting such a topic. Following discussions with Professor Zilic, we discovered that the relatively new STM32 microcontroller possesses unique security features. This insight led us to refine our project thesis, ultimately choosing to develop cloud-based TEE supported OTA updates for H5 board. Throughout the first semester, we've conducted several small-scale demonstrations to familiarize ourselves with the H5 microcontroller, TEE, and OTA. A significant part of our effort has been devoted to establishing the driver and connectivity for our Wi-Fi module and the bootloader. During the second semester, the team focused on integrating all the components we developed earlier and applying our system onto an AI Interaction Application as a visual for better presentation. In addition, to better showcased the OTA part of our project, the team created a comprehensive dashboard serves as the frontend management page.

List of abbreviations and Keywords

MCU: microcontroller;

TEE: Trusted Execution Environment;

OTA: Over-the-Air update;

TCP/IP: Transmission Control Protocol / Internet Protocol;

STM32H5: STM32H573I-DK Board;

Wi-Fi: Wireless networking protocol;

AI: Artificial Intelligence;

AWS: Amazon Web Services.

Introduction

In the realm of embedded systems and microcontrollers, the advancement of security and update mechanisms is pivotal to ensure the integrity and longevity of devices in a rapidly evolving technological landscape. This project aims to develop a TEE-based OTA updating system specifically for the STM32H5 microcontroller.

Comprised of four members from electrical and computer engineering backgrounds, the team brings together diverse skills and a shared passion for microcontroller and security. Under the guidance of Prof.

Zeljko Zilic, we endeavor to address the challenges associated with implementing secure, reliable, and efficient OTA updates in a TEE context. This involves developing a robust system that not only ensures the security of the update process but also seamlessly integrates with the existing architecture of the H5 microcontroller.

The motivation behind it arises from the increasing necessity for secure and efficient update mechanisms in embedded systems which are widely used in various applications such as consumer electronics and industrial systems. Our H5 board is a rather new commercial model featuring several advanced features like Arm Cortex-M33 core and Trust Zone technology, presenting a unique opportunity to explore and implement OTA updates in a TEE. The significance of this project lies in its technical complexity as well as potential to contribute to the broader field of embedded system security. By successfully implementing a TEE-OTA mechanism, we target to demonstrate the feasibility and benefits of such an approach, and guide future passionate ECE students to ease their workflow with STM32 products.

The project is split into three phase. The first phase is Wi-Fi-based OTA (Over-The-Air) technology, which is designated for the update phase. The application can be updated without us having physical access to the device. The second phase is dedicated to the TEE (Trusted Execution Environment), emphasizing the necessity for security during processes. The top layer is where we implement an application that can be continuously updated with our designed system. And in this project, we decided to design an AI-powered chess game. Players can interact with the game via a touchscreen interface on H5 board and compete with AI.

This report outlines our journey throughout the year, the final design of our project, the challenges we have encountered, and the milestones we have achieved, setting the stage for the exciting developments that lie ahead. Finally, we will address the environmental impact of non-renewable resources used in microcontroller manufacturing and emphasize the societal benefits of OTA updates in reducing electronic waste and enhancing cybersecurity.

Background

Microcontroller

In this project, the STM32H573I-DK Discovery kit is utilized. It provides a thorough platform for showcasing and developing applications with the STMIt 32H573IIK3Q microcontroller. This device is equipped with the Arm Cortex-M33 core, featuring Trust Zone technology tailored for Armv8-M. It also boasts capabilities like digital signal processing (DSP) and a floating-point unit (FPU). The core's operational speed reaches up to 250 MHz, and it can function effectively in temperatures as high as 125°C, thanks to its support for an extended temperature range [1].

The microcontroller is notable for its 34 communication peripherals and its flexible memory expansion capabilities. A key aspect of this microcontroller is its security features, being the first in the STM32 series to incorporate a secure TEE software. Additionally, its embedded switched-mode power supply (SMPS) efficiently manages power by scaling down the supply voltage, enhancing overall power efficiency.

Bootloader

Bootloader is essentially the tool we use to load the data of an operating system kernel into the working memory during the startup of the device [2]. As soon as the firmware initialize bootloader, the boot process will start by loading the main memory. Then the system's kernel contains all storage and processor permissions and drivers will be loaded. It will then process different routine tasks and commands. After completion, it will return system responsibility to the kernel. The bootloader secures the device's state, initiates the Trusted Execution Environment (TEE), and establishes its foundational trust. It checks the integrity of the boot and recovery sections before transferring control to the kernel.

Trusted Execution Environment

The trusted execution environment (TEE) refers to a secure area in the processor of a connected device that ensures sensitive data is stored, processed, and safeguarded within an environment that is both trusted and isolated. Security software with TEE is known as trusted applications (TAs). It offers comprehensive protection by guarding the execution of verified code and maintaining the confidentiality [3].

As mentioned before, STM32H573 microcontroller incorporated TEE software. Secure Manager is the framework used for TEE for this microcontroller. The typical starting point when using security feature of this microcontroller is a secure boot. It is initiated right after a reset and ensures user applications are safe and unchanged before they start. It ensures user applications are safeguarded and unchanged before they run and plays a key role in secure firmware installation and updates. The integrity and identity of user application images are verified before installation. The process is known as Root of Trust (RoT). There are two possible boot methods, STiRoT (ST Immutable Root of Trust) and OEMiRoT (Original Equipment Manufacture's Immutable Root of Trust) [4].

A secure boot is ensured by maintaining different levels of isolation levels (HDPL). A protected hardware block System Boot and Security (SBS) controls HDPL. The level is set using a counter that increases monotonically through different stages of system boot. To reset the counter, a power-on reset or system reset is required. Each isolation level has its own access and execution rights. At any given level, the code and related secrets from lower HDPL levels are inaccessible. The level diagram for HDPL is as below.

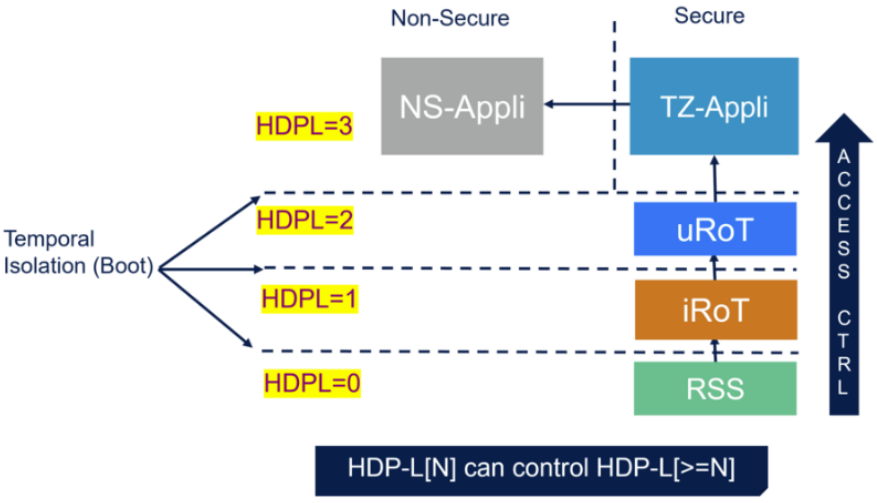

Figure 1. HDPL Level

As shown in the diagram, level zero is the foundational level. It is never erased and is reserved for system flash memory. RSS stands for root security service. It plays a crucial role in securely installing extension like provisioning a device. Level one represents the immutable root of trust which are STiROT or OEMiRoT. Level 2 allows for an updatable root of trust. This is an optional further boot stage and includes OEMuRoT or STuRoT. Level three is the level for user application code, differentiated into secure which is protected through TrustZone and non-secure sections [4].

The possible boot path is shown in the Figure below.

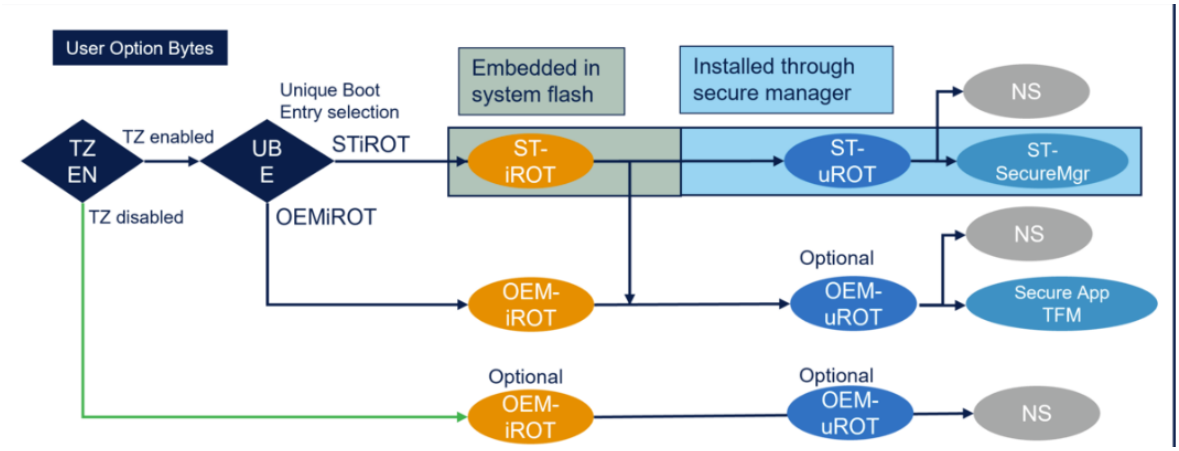

Figure 2. Possible Boot Path

To operate secure manager, secure provisioning is a prerequisite. It is the process of installing keys, certificates, and setting to the application. When developing secure manager, the device is in TZ-closed product state. There are six product state open, provisioning, TZ-closed, closed, locked, and regression. Open indicates for unrestricted development where user is free to experiment. Provisioning represents the intention to establish security, install keys, secure firmware and set the OB. TZ-closed is a state that the secure environment has been set and developers now can only work with the nonsecure part. Closed represents the final product delivery state and developers have no debug access. Locked is same as closed but does not have regression. Regression is the process of reopening for debugging [5].

Over-the-Air Update

Over the air (OTA) is a firmware or operating system updating mechanism to embedded systems using a wireless network connection or Bluetooth [6]. Over-the-air (OTA) updates typically follow a structured process to ensure smooth and secure updates. Initially, a connection is established between the device and the updated server. Following this, an OTA handshake occurs, allowing communication between the device and the server to prepare for the update. The next critical step involves checking the readiness of the device for the update. Once readiness is confirmed, the device proceeds to download the update. After the download, the update is then applied to the device. Post-application, the update undergoes a verification process to ensure it has been correctly installed. Finally, to complete the update process, the device reboots, integrating the new update into its system [7].

Minimax

To enhance user interaction, the team developed an AI application for playing "Four in a row" and "Five in a row." They employed the minimax backtracking algorithm, common in two-player turn-based games, which aims to minimize the potential maximum loss. There are two key players Maximizing player and minimizing payer. The Min Max algorithm assesses all potential moves for both the current and opposing players through recursion. It initiates at the game tree's root, applying Minmax to every subordinate node. The algorithm oscillates between maximizing and minimizing the value of each node at every tree level. Maximizing player would choose the child node with the highest value, while the minimizing player chooses the child node with the lowest value. The algorithm returns when the child node reaches a terminal code or predefined depth [8].

Requirements

OTA updates allow transferring new firmware to devices remotely. However, without adequate security, this process could introduce malicious firmware into devices. This inadequate security is more common for STM32 board. The previous STM32 board before STM32H5, without considering the security, may execute code which are unauthorized or malicious. What’s more, it will not check the integrity of the firmware (such as cryptographic hashing and signature verification). To solve this problem ST company published a tool called Secure Manger. This tool, announced by ST company, creates a trusted execution environment for a 32-bit MCU. It is the first time this manager is being used in STM32H5 board. Secure boot ensures that the microcontroller only runs verified and trusted firmware. Figure 3 shows the architecture of STM32Trust TEE implemented by Secure Manger. However, because it is newly released, secure mange is lack of scalability. AWS and mongoose library, supporting Arm TrustZone and STMicroelectronics secure manager, provides a more suitable solution. The X-CUBE-AWS-H5 Expansion Package keeps the device credentials and settings encrypted in the MCU secure storage [9]. What’s more, the security-sensitive data and operations remain in a secure partition which means they are not exposed to user application ensuring its security. In general, the main goal of the project is to enable OTA update for the firmware (application file) of board STM32H573I-DK under security environment and condition, which is implemented by AWS and mongoose.

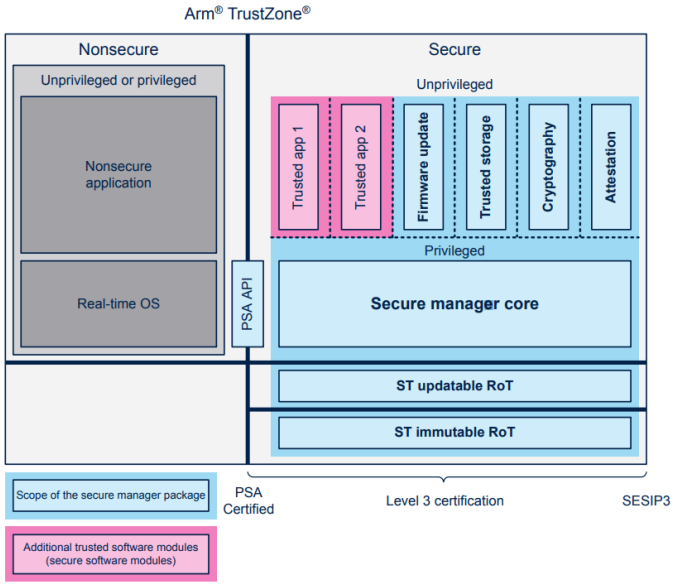

Figure 3. TEE Architecture [4]

Project Requirements

Secure requirement(s):

1. Use verification and cryptographic algorithm provided by AWS to encrypt and verify file transferring between server and board.
2. STM32H573I-DK board should use the function of iRoT and secure storage provided by AWS to ensure its security.
3. Firmware should transfer via Wi-fi or Enternet.
4. Secure have to provide level 3 isolation.

Hardware requirement:

STM32H573I-DK board should have reliable Wi-Fi connection to receive OTA files.

Project Constraints

Hardware constraint(s):

1. Only one STM32H573I-DK board was available during the whole project.
2. Use Wi-Fi model to transfer update files.
3. The size of Touchscreen 240 × 240 pixels.

Time constraint(s):

1. Finish the project before 2024 April 12.
2. Project timelines vary depending on testing.

Budget constraint(s):

1. Economic budget not applicable for this project.
2. Time budget: total maximum 25 hrs / week work session.

User constraint(s):

1. The OTA system should be user-friendly.
2. The transmission time of the update file should not be longer than 10s.
3. The firmware upgrade binary file cannot exceed 140 kB.

Design & Results

Phase I

During the firmware update, bootloader is the most crucial component that decides which firmware to execute. There are two strategies for the bootloader, one is we create two bootloaders residing in the beginning of each firmware binary. The other one is that we can only create a separate bootloader in the beginning of the flash and handles the execution of the two firmware. We can modify the linker script to make the two firmware exist in the two different partitions of the flash.

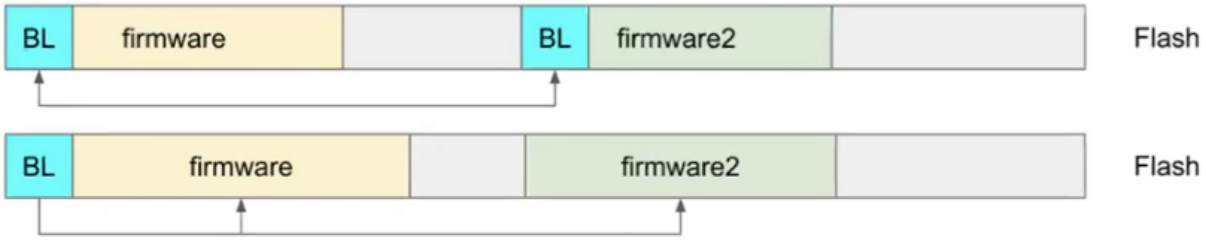

Figure 4. firmware on flash

After knowing how to manipulate the bootloader, we explored how to set up a secure OTA with the support of AWS. AWS IoT core provides services like job management, version control, and certificate verification. Our implementation first uses the internal CA certificate to securely connect to the endpoint of the AWS. This involves the usage of MbedTLS which handles transfer layer security and certificate verification.

When the device is successfully registered as a Thing on IoT core, we used MQTT protocol to subscribe to the cloud publisher. We can then transfer the message and data block via this connection. We can monitor the state of the device directly in the AWS terminal.

To enhance the security, we used Secure Manager developed by ST company. This is incorporated software which enables Arm PSA APIs. By using these APIs, especially Firmware update libraries, we can have an easier way to handle the data exchange between the secure zone and non-secure zone. As shown below, there is a dispatch method which forwards a request in the non-secure environment to the trust zone for execution and brings the results back. By following this structure, we can have a more convenient while secure way of manipulating flash, performing version checking, and signature checking. When running the OTA job, progress of the job and flash writing will be printed in the serial terminal. We can easily monitor it.

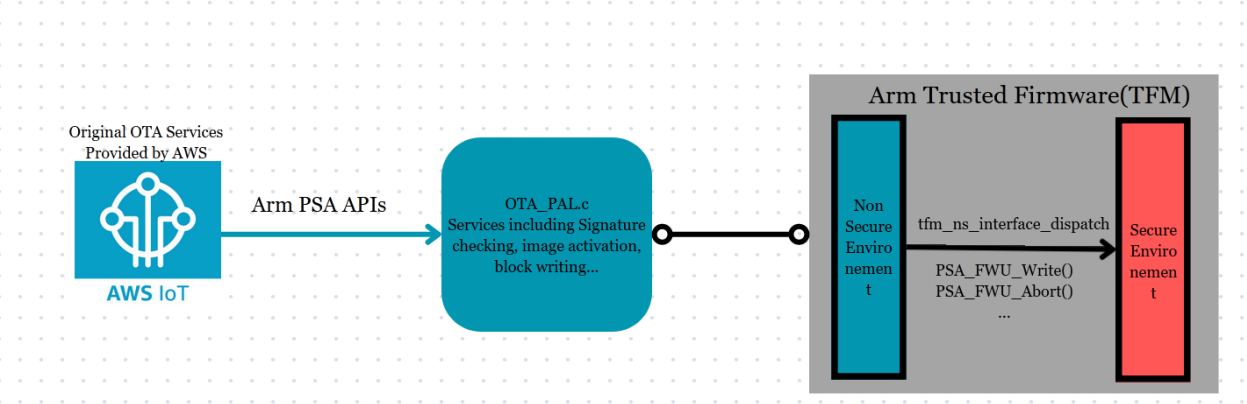

Figure 5. Data exchange between secure and non-secure zone

```
<INF>  8482390 [OTAAgent  ] Received valid file block: Block index=199, Size=2048 (ota.c:2535)
<INF>  8482398 [OTAAgent  ] Number of blocks remaining: 2 (ota.c:2771)
<INF>  8482398 [OTAAgent  ] Sent PUBLISH packet to broker $aws/things/stm32h5xx-user-0117000000000007c627/jobs/AF
_OTA-ota_test1/update to broker. (ota_update_task.c:1160)
<INF>  8482398 [OTAAgent  ] Current State=[WaitingForFileBlock], Event=[ReceivedFileBlock], New state=[WaitingFor
ileBlock] (ota.c:2939)
<INF>  8482399 [OTAAgent  ] Sent PUBLISH packet to broker $aws/things/stm32h5xx-user-0117000000000007c627/streams
AFR_OTA-e5c2f021-f4f7-4b81-ac3a-88c74a81912e/get/cbor to broker. (ota_update_task.c:1160)
<INF>  8482399 [OTAAgent  ] Published to MQTT topic to request the next block: topic=$aws/things/stm32h5xx-user-0
17000000000007c627/streams/AFR_OTA-e5c2f021-f4f7-4b81-ac3a-88c74a81912e/get/cbor (ota_mqtt.c:1166)
<INF>  8482399 [OTAAgent  ] Current State=[WaitingForFileBlock], Event=[RequestFileBlock], New state=[WaitingForF
leBlock] (ota.c:2939)
<INF>  8482522 [MQTTAgent ] Handling callback for task=OTAAgent, topic="$aws/things/stm32h5xx-user-01170000000000
7c627/streams/AFR_OTA-e5c2f021-f4f7-4b81-ac3a-88c74a81912e/data/cbor", filter="$aws/things/stm32h5xx-user-0117000
00000007c627/streams/AFR_OTA-e5c2f021-f4f7-4b81-ac3a-88c74a81912e/data/cbor". (mqtt_agent_task.c:710)
<INF>  8482523 [OTAAgent  ] Received valid file block: Block index=200, Size=2048 (ota.c:2535)
<INF>  8482530 [OTAAgent  ] Number of blocks remaining: 1 (ota.c:2771)
<INF>  8482530 [OTAAgent  ] Current State=[WaitingForFileBlock], Event=[ReceivedFileBlock], New state=[WaitingFor
ileBlock] (ota.c:2939)
<INF>  8482553 [MQTTAgent ] Handling callback for task=OTAAgent, topic="$aws/things/stm32h5xx-user-01170000000000
7c627/streams/AFR_OTA-e5c2f021-f4f7-4b81-ac3a-88c74a81912e/data/cbor", filter="$aws/things/stm32h5xx-user-0117000
00000007c627/streams/AFR_OTA-e5c2f021-f4f7-4b81-ac3a-88c74a81912e/data/cbor". (mqtt_agent_task.c:710)
<WRN>  8482553 [MQTTAgent ] Incoming publish with topic="$aws/things/stm32h5xx-user-0117000000000007c627/jobs/AFR
OTA-ota_test1/update/accepted" does not match any callback functions. (mqtt_agent_task.c:726)
<INF>  8482554 [OTAAgent  ] Received valid file block: Block index=201, Size=1699 (ota.c:2535)
<INF>  8482560 [OTAAgent  ] Received final block of the update. (ota.c:2718)
<INF>  8483109 [OTAAgent  ] Received entire update and validated the signature. (ota.c:2742)
>
```

Figure 6. progress of job and flash writing on terminal

When the version check, digital signature checking, and certificating are all done, writing to flash in an elegant manner is the major thing that we need to consider. Since H5 board supports dual bank flash, this is the most convenient way of maintaining two versions of firmware at the same time. As shown below, the device flash is divided into two equal parts. When the flash memory is mapped to system memory, a certain register is responsible for the order of the mapping. We can change this register in the flash controller to map the first bank first and second bank follows or vice versa. To achieve this, we just need to modify one bit in the register and banks will swap. During the firmware update, we load the firmware and write it to the second bank, then we swap the bank and reboot. New firmware will be in execution while the older firmware will still reside in the second bank, which means we can even roll it back by swapping the bank.

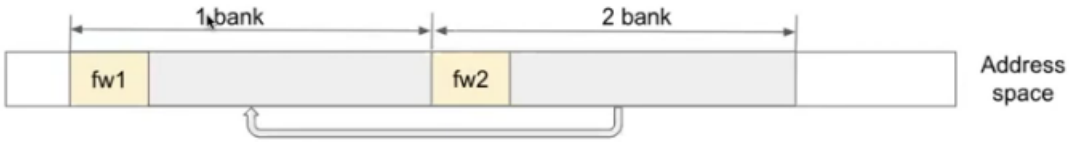

Figure 7. Dual Bank Flash

Phase II Application-wise

After securing a complete workflow for our initial target: a cloud-based OTA mechanism in Trusted Execution Environment, we move forward to next phase of the project. The goal overall is to deliver a working board game intelligence that a user can visually interact with on the onboard touchscreen.

We start by exploring the TouchGFX library. TouchGFX is a software framework used for developing graphical user interfaces (GUIs) on embedded systems, particularly running on STM32 microcontrollers. After initializing and drawing the needed components on STM32CubeMX, it then generated the outline file for TouchGFX. Continuing exploring the TouchGFX library, we explored the functionality of the ClickListener in TouchGFX to detect touch interaction for initialized pushbutton or circle that we draw in STM32CubeMX. The Click Listener in TouchGFX enables specific actions to be performed when the user touches the screen, which is critical to our applications.

In parallel with familiarizing and implementing ToughGFX-based UI and interaction logic, another part of the team starts to find an efficient and appropriate algorithm to search and evaluate the best move. We have chosen to develop a deterministic algorithm, minimax, instead of a more generalized solution that we can further extend to even chess and go: Monte Carlo. This is out of the consideration that the team are mainly formed by electrical engineering students with limited computer science capabilities. Also, this method proves computationally intensive for an embed system with quite limited processor clock frequency and memory. It can indeed produce an acceptable result in polynomial time, yet the length of which is still unacceptable. The only solution would be loading a pre-trained deep neural network model to perform the task. However, we believe this is not a persuasive approach to demonstrate our topic of continuously upgradable system, since a solution using DNN is rather mature and the core to such upgrades is merely the amount of primitive training data. Without enough open-source real-world game data and hardware, it is impractical to go with this solution.

Therefore, we will only develop two elementary games: Connect Four (i.e. Four in a Row) and Gomoku, which is sufficient to serve as a secure and upgradable OTA demonstration. Fig. 8a below depicts the fundamental logic of this famous algorithm. You may refer to minimax above in the background section. It is essentially a recursively tree search trying to maximize the computer's scores while minimizing a player's. By integrating the alpha-beta pruning, it can generally produce a result immediately for Connect Four within difficulty level of 6 and Gomoku of 3. The hardest part of designing this intelligence model eventually turns out to be the heuristic function, whose job is to assign different scores to different linear situations. For instance, in Connect Four, when making a specific move will lead the opponent to be able to drop its next piece that leads to 4 of its pieces linear connected, the heuristic function will assign this move a massively low score. The reason this part of the algorithm proves particularly hard to tune lies in the fact that we always want to make it play like us. However, with a deterministic coding style, there can

only be one way to balance the score (ultimately, aggression habit) at a time. Figure 8b reveals an exemplary score distribution through trial and error in C++. We see that the computer prefers to avoid losing over instant winning.

```
IF (depth = 0 OR depth ≥ remaining unoccupied)
  FOR (every linear n cells)
    score += its score
  ENDFOR and RETURN score
END IF

IF (opponent wins)
  RETURN extremum // COMP: INT_MIN; USER: INT_MAX
ENDIF

FOR (every cell)          // α-β pruning and optimizations omitted
  IF (unoccupied AND near occupied)
    copy ← current board
    score ← MINIMAX(copy, depth-1, opponent)
    IF (player is COMP)
      IF (score > old score)
        update score and record current cell
      ENDIF
    ELSE                  // player is USER
      IF (score < old score)
        update score and record current cell
      ENDIF
    ENDIF
  ENDIF
ENDFOR
```

```cpp
int heuristic(std::vector<int> &v) {
    int favour{0}, neutral{0}, hazard{0}, scores{0};
    for (int i : v) {
        neutral += (i == 0) ? 1 : 0;
        favour += (i == COMP) ? 1 : 0;
        hazard += (i == USER) ? 1 : 0;
    }
    if (favour == 5) {
        scores += 1024;
    } else if (favour == 4 && neutral == 1) {
        scores += 256;
    } else if (favour == 3 && neutral == 2) {
        scores += 64;
    } else if (favour == 2 && neutral == 3) {
        scores += 16;
    } else if (favour == 1 && neutral == 4) {
        scores += 4;
    } else if (hazard == 1 && neutral == 4) {
        scores -= 8;
    } else if (hazard == 2 && neutral == 3) {
        scores -= 32;
    } else if (hazard == 3 && neutral == 2) {
        scores -= 128;
    } else if (hazard == 4 && neutral == 1) {
        scores -= 512;
    } else if (hazard == 5) {
        scores -= 1025;
    }
    return scores;
}
```

Figure 8. Pseudo-structure of minimax algorithm and a well-behaving heuristic logic for Gomoku

After completing the design for Connect Four, we are now familiar with the TouchGFX library enough to prepare Gomoku, a traditional game originated from China and Japan is the source of inspiration of Connect Four. Therefore, the core algorithm only requires small modifications to make it functional as expected. However, earlier the computer only needs to evaluate out of 7 possibilities (even less when columns getting full), while now it must deal with 15 × 15 = 225 every loop. Considering the tree search nature of the algorithm, it becomes exponential intensive. This can be manifested by adding a timer in the program prototype. With difficulty 1, in Fig. 9 it takes my computer (single threaded program with 5.4 GHz CPU) 0.04 - 0.16ms to respond for Connect Four and 10-35ms for Gomoku on average. So extrapolating this to a 0.25GHz MCU with difficulty higher than 2 will lead to speed concerns. The solution is straightforward: we have optimized it by not traversing all the remaining cells, instead only the peripheral of the occupied positions, which is consistent to a human player's inclination, enhancing the average efficiency by around 150%. A simple randomization algorithm has also been introduced in early stages to evade minimax' deterministic nature. For detailed codes, please refer to our repository. We then transplanted Gomoku to the STM32 project.

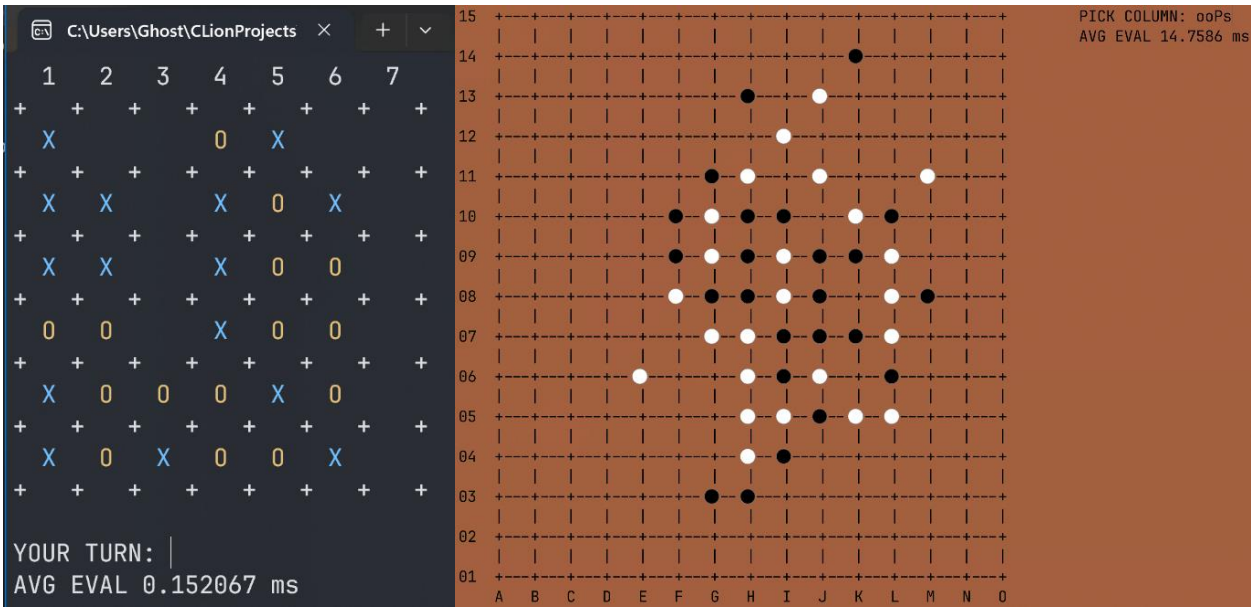

Figure 9. Game debugging on terminal

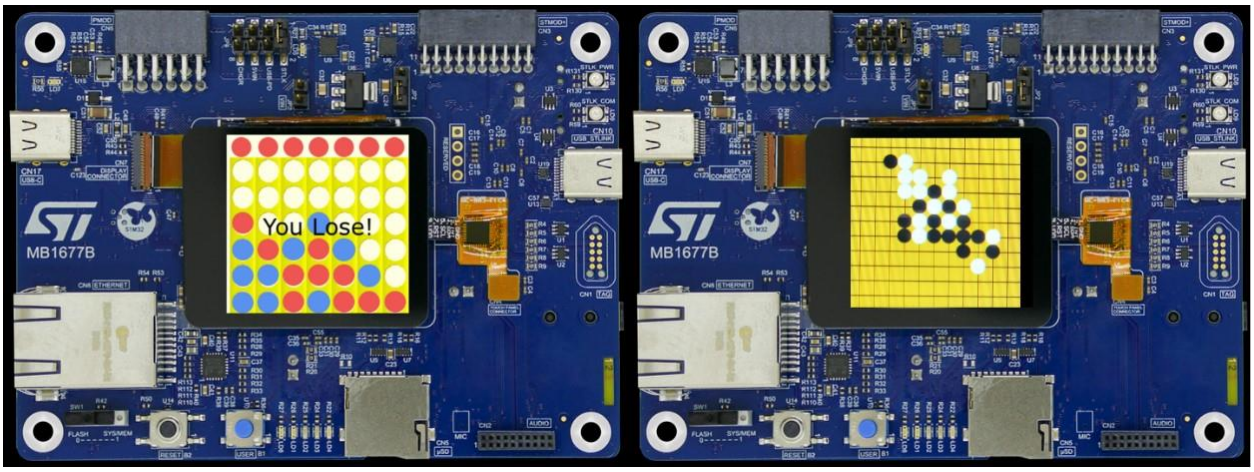

Figure 9. Real game demonstration

Additionally, the game initializes with a starter screen, allowing user to select games. It may also include other common user experiences and security features. TouchGFX has support for multiple screens. The interaction allows the user to configure a function when the trigger happened. After the event listener detecting the click of the start button, it will switch to the chosen game's screen. Similarly, upon detecting a game-over (win / loss / draw) condition, the system will display the message and freeze further interaction. A user may restart the game by pressing the reset button onboard, essentially calling the function gotoScreen1ScreenNoTransition.

In the future, we may deliver more and enhanced functionalities through OTA like regretting a move using the blue USER button, creating user profile with data stored in dual bank.

Final demonstration overview

A fully functional and satisfactory game is now ready at hand, and we proceed to a solution of demonstrating it in secured environment. Because the Design Day has no wireless connection, we will use ethernet and create a local server to update. This sub-project showcases the integration of the Mongoose Library into an embedded device to create a comprehensive device dashboard. The dashboard boasts authentication features, ensuring a secure login-protected interface for users. Additionally, it supports multiple logins with potentially different permissions, enhancing flexibility and user management capabilities. Notably, the Web UI can be entirely embedded into the firmware binary, eliminating the need for a filesystem to serve it. This design choice enhances resilience to filesystem-related issues.

All modifications made to the dashboard are seamlessly propagated to all connected clients, ensuring real-time updates and consistency across user interfaces. For a detailed step-by-step guide, you may refer to the tutorial available on Mongoose Documentations. The flowchart below aims to serve as a concise user guide.

1	•1. Clone the repository following the instructions provided in the link above
2	•2. Enable Internet sharing in Control Panel for your current connection
3	•Compile the dashboard Web UI
4	•Compile STM32H573I-DK OTA capabilities
5	•Write the generated .elf file to your board in STM32CubeProgrammer (optionally, perform a full chip erase beforehand)
6	•Add the current folder containing `esputil` to `PATH` variables
7	•Establish a connection by running the following command in the command prompt (replace ## with your own portal, which can be found in the device manager) `.\esputil -p /COM## monitor`

Table 1. Demo User Guide

Once the connection is established, open the provided temporary IP address in your web browser. This workflow has been tested for Windows environments. Users can follow these steps to set up and utilize the device dashboard efficiently.

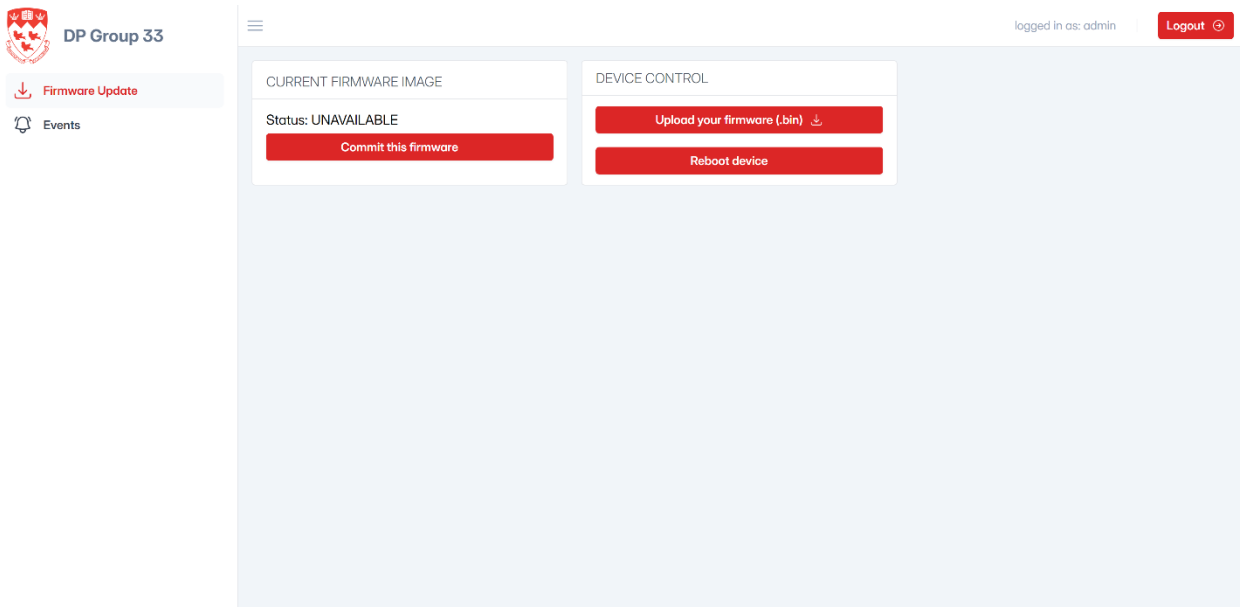

Figure 10. OTA Front End Management

Impacts on Society and Environment Section

Use of Nonrenewable resources

The project utilized the STM32H573 MCUS, which relies on nonrenewable resources from manufacturing to disposal. Its production involves rare metals and synthetic materials, contributing to carbon emissions during distribution and requiring non-renewable energy for operation. Proper recycling can mitigate environmental impacts by recovering valuable materials. Functional MCUS can also be repurposed for new projection, while the rest can be sent to certified e-waste for appropriate handling.

Environmental benefits

Implementing OTA updates on a TEE using STM32H5 provides several environmental benefits, especially when compared to more pollutive technologies. The STM32H573I-DK microcontroller from ST Microelectronics, which features an embedded SMPS, enhances these benefits. SMPS, with its high efficiency in power conversion, significantly reduces the overall system power consumption in all power modes, contributing to energy conservation. This aligns with ST Microelectronics' commitment to sustainability, ensuring their products minimize environmental impacts through their eco-design process [10]. Furthermore, OTA updates eliminate the need for physical displacement for maintenance, thereby reducing carbon emissions associated with transportation. Additionally, the SMPS's ability to dynamically scale voltage according to running requirements further optimizes power usage as in Table 1 reinforcing the microcontroller's eco-friendly profile. Lastly, the enhanced security from TEE reduces the likelihood of device replacement due to security breaches, leading to less electronic waste.

Voltage scaling range	V_{CORE}	Maximum frequency
VOS0	1.35 V	250 MHz
VOS1	1.2 V	200 MHz
VOS2	1.1 V	150 MHz
VOS3	1.0 V	100 MHz

Table 2. SMPS voltage scaling cascade [10]

Safety & Risks

Cybersecurity is crucial in engineering projects like ours, necessitating strong encryption and secure channels to mitigate cyber threats. Protecting data, such as network credentials in mx_wifi_conf.h, from unauthorized access is essential. Currently, we prioritize setting correct access limits for our GitHub repository and uphold the confidentiality of personal data to prevent serious consequences of data leaks. As engineers, maintaining a vigilant stance on data protection is imperative.

Benefits to Society

OTA updates minimize the need for physical handling, saving costs for manufacturers and consumers. Employing a Trusted Execution Environment (TEE) for these updates boosts security, addressing current cybersecurity challenges and reducing the frequency of device replacement or re-encryption processes. Moreover, as companies invest in OTA and TEE technologies, the demand for skilled engineers rises, leading to job creation and economic benefits through technological advancement and employment growth.

Teamwork

Throughout the second semester, our team demonstrated strong collaboration, with each member fully contributing to our success. We initiated the year by forming pairs to conduct preliminary research and minor demonstrations, ensuring a solid foundation for our project. Subsequently, we pooled our efforts towards researching and crafting a concrete application: board game. The task was mainly divided into two sections: algorithm development / testing, along with TouchGFX integration.

To maintain our effective teamwork, we upheld key practices that proved instrumental in our success. Firstly, we ensured an equitable distribution of tasks, resulting in each member fulfilling their responsibilities effectively and extending support to others whenever necessary. Secondly, we meticulously tracked project milestones, which provided us with a clear roadmap of our progress. From determining our topic to completing preliminary demos and finally achieving the satisfactory chess intelligence, monitoring these milestones enabled us to gauge our advancement accurately. Effective communication remained paramount throughout our collaboration. Leveraging a social media group and a shared GitHub workspace facilitated continuous updates on project progress, ensuring everyone remained informed and engaged. Moreover, storing all files and codes on the cloud eliminated any physical constraints imposed by limited hardware availability, enabling seamless participation from any location.

As we conclude our capstone project, we reflect on our accomplishments and the teamwork that fueled our success. Moving forward, we appreciate the need to adapt and enhance our practices as electrical and computer engineers for future endeavors. By maintaining our commitment to effective collaboration in future career, we are confident in our ability to tackle any challenges that lie ahead.

Conclusion

During the initial semester, the team dig into various methodologies for advancing our project, becoming familiarized with the H5 microcontroller, TEE, and modern OTA technologies. A significant portion of our time was allocated to testing distinct components, including the bootloader and Wi-Fi module. Progressing into the second semester, the team managed to integrate the entire system. Furthermore, we developed a User Interface frontend for OTA updates and an AI interaction module to visually demonstrate the system's capabilities. The team takes pride in our achievement and is grateful for everyone who helped us along this journey. The capstone project paved a leading path into the industry and provided us with a precious opportunity to integrate the knowledge of circuit analysis, network communication, cybersecurity, data structure and so on. Moving forward, we are dedicated to exploring deeper in the field of electronics during our graduate studies as future engineers.

Acknowledgements

We would like to express the deepest gratitude to Professor Zilic for providing us with this fascinating yet challenging opportunity, and his invaluable guidance throughout the duration of this project, even during our Design Day. Additionally, we extend our heartfelt credit to his graduate team Shaluo Wu and Guanyi Heng, for their resourceful and altruistic experience-sharing. Last, big thanks to all team members for their dedication and hard work. Each contribution has been vital to achieving our common goal.

References

[1] "STM32H563/573", STMicroelectronics, 2023. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32h563-573.html#st-featured-products. [Accessed 2 12 2023].

[2] IONOS, 11 5 2022. [Online]. Available: https://www.ionos.ca/digitalguide/server/configuration/what-is-a-bootloader/. [Accessed 4 2024].

[3] "Introduction to Trusted Execution Environments", Global Platform, [Online]. Available: https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf.

[4] "Secure Boot for STM32H5", STMicroelectronics, 10 2023. [Online]. Available: https://wiki.st.com/stm32mcu/wiki/Security:Secure_Boot_for_STM32H5. [Accessed 2 12 2023].

[5] Security features on STM32H5 MCUs, 11 2023. [Online]. Available: https://wiki.st.com/stm32mcu/wiki/Security:Security_features_on_STM32H5_MCUs. [Accessed 2 12 2023].

[6] "Over-the-Air Update", Zephyr Project, 6 9 2023. [Online]. Available: https://docs.zephyrproject.org/latest/services/device_mgmt/ota.html. [Accessed 2 12 2023].

[7] "OTA Update Process (Classic Bluetooth)", Alexa, 14 2 2022. [Online]. Available: https://developer.amazon.com/en-US/docs/alexa/alexa-gadgets-toolkit/classic-bluetooth-ota-update-process.html. [Accessed 3 12 2023].

[8] V. Bansal", Min-Max Algorithm in Artificial Intelligence," [Online]. Available: https://www.scaler.com/topics/artificial-intelligence-tutorial/min-max-algorithm/. [Accessed 10 4 2024].

[9] S. Company", AWS IoT software expansion for STM32Cube targeting STM32H573I-DK," [Online]. Available: https://www.st.com/en/embedded-software/x-cube-aws-h5.html.

[10] "Sustainable Technology", ST Microelectronics, 2023. [Online]. Available: https://www.st.com/content/ccc/resource/corporate/company_promotion/corporate_brochure/group0/84/9e/cc/c8/25/26/4c/23/Sustainable_Technology_Brochure/files/ST_Sust_Tech_Brochure.pdf/_jcr_content/translations/en.ST_Sust_Tech_Brochure.pdf.. [Accessed 3 12 2023].

[11] "Guidelines for power management on STM32H5 MCUs", ST Microelectronics, [Online]. Available: https://www.st.com/resource/en/application_note/an5930-guidelines-for-power-management-on-stm32h5-mcus-stmicroelectronics.pdf. [Accessed 12 2023].