Educational Firmware Over-the-Air (FOTA) Kit for Automotive Systems in STEM Education

"مجموعة تعليمية لتقنية تحديث البرمجيات عن بُعد (FOTA) للأنظمة الخاصة بالسيارات في تعليم العلوم والتكنولوجيا والهندسة والرياضيات (STEM)"

Ghada Abdelhady[1], Emad H. Abdelmalak[2], Omar Elsehity[3]

[1]Faculty of Engineering, October University for Modern Sciences and Arts, Giza, 12572 Egypt.
[2] Swift Act Co., Smart Village, Giza, 12577 Egypt
[3] School of Information Technology, New Giza University, New Giza, 12585 Egypt

Volume Two - Issue Six - November 2024

المجلد الثانى - العدد السادس – نوفمبر ٢٠٢٤

ISSN-Online: 2812-6122 ISSN-Print: 2812-6114

موقع المجلة على بنك المعرفة المصري

https://aiis.journals.ekb.eg/contacts?lang=ar

Abstract: The growing complexity of automotive electronics and the increasing demand for hands-on STEM education highlight the need for practical tools that introduce students to cutting-edge technologies. Firmware Over-the-Air (FOTA) technology, widely used in the automotive industry for remote software updates, remains underutilized in educational contexts. To bridge this gap, we present the Educational Firmware Over-the-Air (FOTA) Kit, a comprehensive learning platform designed to teach foundational and advanced concepts in embedded systems, secure communication, and automotive technologies.

The kit integrates hardware components such as Raspberry Pi, STM32 microcontrollers, and Controller Area Network (CAN) Bus interfaces with software modules covering bootloaders, encryption protocols, and an intuitive Graphical User Interface (GUI). These elements simplify complex automotive technologies for classroom use. Through interactive activities, including writing custom bootloaders, configuring CAN networks, and implementing AES encryption, students gain practical, real-world experience while developing technical proficiency.

Initial deployment of the Educational FOTA Kit has demonstrated its effectiveness in enhancing student engagement, bridging theoretical knowledge and practical application. Quantitative results indicate improvements in problem-solving skills and a deeper understanding of secure firmware updates. By fostering innovation and technical expertise, this kit prepares students for emerging fields such as IoT, automotive engineering, and cybersecurity. The Educational FOTA Kit marks a significant advancement in STEM education, offering a dynamic and versatile platform aligned with the technological demands of modern industries.

Keywords: Automotive Communication Protocols, Educational Kit; Firmware Over-the-Air (FOTA), Infotainment System, Secure Firmware Update.

المستخلص :

تزايد تعقيد الإلكترونيات في صناعة السيارات والطلب المتزايد على التعليم العملي في مجالات العلوم والتكنولوجيا والهندسة والرياضيات (STEM) يبرز الحاجة إلى أدوات عملية تُعرّف الطلاب بالتقنيات المتقدمة. تقنية تحديث البرمجيات عن بُعد (Firmware Over-the-Air - FOTA)، المستخدمة على نطاق واسع في صناعة السيارات لتحديث البرمجيات عن بُعد، لا تزال غير مستغلة بشكل كافٍ في السياقات التعليمية. لسد هذه الفجوة، نقدم مجموعة تعليمية لتقنية تحديث البرمجيات عن بُعد (Educational FOTA Kit)، وهي منصة تعليمية شاملة مصممة لتعليم المفاهيم الأساسية والمتقدمة في الأنظمة المدمجة، والاتصالات الآمنة، وتقنيات السيارات.

تدمج المجموعة مكونات الأجهزة مثل Raspberry Pi، ووحدات التحكم الدقيقة STM32، وواجهات شبكة CAN Bus مع وحدات برمجية تغطي محمّلات الإقلاع (Bootloaders)، وبروتوكولات التشفير، وواجهة رسومية مستخدم (GUI) سهلة الاستخدام. تعمل هذه العناصر على تبسيط التقنيات المعقدة في صناعة السيارات

لتكون ملائمة للاستخدام في الفصول الدراسية. من خلال أنشطة تفاعلية، مثل كتابة محمّلات الإقلاع المخصصة، وتكوين شبكات CAN، وتطبيق تشفير AES، يكتسب الطلاب خبرة عملية مباشرة ويطورون كفاءات تقنية عالية.

أظهرت التجربة الأولية لمجموعة Educational FOTA Kit فعاليتها في تعزيز تفاعل الطلاب وربط المعرفة النظرية بالتطبيق العملي. تشير النتائج الكمية إلى تحسينات في مهارات حل المشكلات وفهم أعمق لتحديثات البرمجيات الآمنة. من خلال تعزيز الابتكار والخبرة التقنية، تُعد هذه المجموعة الطلاب للعمل في مجالات ناشئة مثل إنترنت الأشياء(IoT) ، وهندسة السيارات، والأمن السيبراني. تمثل مجموعة تعليمية لتقنية FOTA تقدمًا كبيرًا في تعليمSTEM ، حيث تقدم منصة ديناميكية ومتعددة الاستخدامات تتماشى مع متطلبات التكنولوجيا الحديثة في الصناعات الحالية.

الكلمات المفتاحية: بروتوكولات الاتصال في السيارات، عدة تعليمية، تحديث البرامج الثابتة عبر الهواء (FOTA)، نظام المعلومات والترفيه، تحديث البرامج الثابتة الآمن

1.Introduction

Firmware Over-the-Air (FOTA) is a transformative technology in the automotive industry [1], enabling secure and efficient updates to vehicle software. With millions of lines of code managing advanced vehicle functions, maintaining and improving firmware has become essential for manufacturers. FOTA eliminates the need for vehicles to visit service centers for software updates, reducing costs, increasing efficiency, and ensuring user convenience while enhancing vehicle reliability and security.

Despite its widespread industrial success, FOTA technology remains underutilized as a teaching tool in STEM education. Current STEM curricula often prioritize theoretical concepts over practical exposure to real-world technologies, limiting students' ability to bridge academic knowledge with industry requirements (Maspul, 2024). This disconnect hampers the readiness of graduates to meet the technical demands of modern engineering and IoT systems (Gugole et al., 2023). Addressing this gap requires innovative educational tools that integrate emerging technologies into hands-on learning environments.

To fill this void, the Educational Firmware Over-the-Air (FOTA) Kit offers a comprehensive platform designed to teach students both foundational and advanced concepts in embedded systems, secure communication protocols, memory management, and automotive technologies. The kit incorporates accessible hardware components such as Raspberry Pi, STM32 microcontrollers, and Controller Area Network (CAN) Bus interfaces, paired with software modules that include custom bootloaders, encryption protocols, and a user-friendly Graphical User Interface (GUI). By leveraging simplified implementations and intuitive design, the kit ensures that complex automotive technologies are accessible to learners at various educational levels.[٢]

The Educational FOTA Kit not only bridges the gap between theoretical knowledge and practical application but also equips students with essential skills for emerging fields such as IoT, automotive engineering, and cybersecurity. This paper presents the kit's design, implementation, and potential to revolutionize STEM education by fostering innovation and technical proficiency in a practical, interactive environment.

2. State of the Art

Firmware Over-the-Air (FOTA) technology has been extensively explored in industrial and research contexts, focusing on its implementation in IoT, automotive, and sensor networks. Nikolov et al. proposed a cloud-based FOTA system that utilized bootloaders to ensure memory segmentation and reliability [3]. Their system demonstrated the importance of rollback mechanisms for maintaining functionality during updates. However, it lacked security measures to protect data during transmission.

Kerliu et al. implemented AES encryption for secure firmware updates in sensor networks, achieving high reliability under controlled conditions. Although effective, their system did not address memory management or scalability for complex applications [4]. Similarly, Schmidt et al. proposed secure FOTA packages optimized for IoT devices, balancing security, memory usage, and processing speed. While comprehensive, their approach lacked detailed benchmarks for evaluating performance in real-world scenarios.[٥]

Although these studies have advanced the field of FOTA, their focus has been on industrial applications rather than education. There is a lack of educational tools designed to teach FOTA concepts to students. Existing solutions prioritize industrial applications, often requiring expertise beyond the reach of most learners. This gap underscores the need for an educational adaptation of FOTA systems, which this paper addresses by creating a comprehensive kit designed for accessibility and hands-on experimentation. The Educational FOTA Kit builds on these foundations by adapting these technologies for teaching purposes, offering students hands-on experience in areas such as secure communication and memory sectorization.

3. Educational Design of the FOTA Kit

The Educational Firmware Over-the-Air (FOTA) Kit is designed to address the gap between theoretical knowledge and practical applications in STEM education. By incorporating real-world automotive technologies into classroom activities, the kit offers students a dynamic and engaging platform to develop technical proficiency and innovative thinking.

3.1 Integration into STEM Curricula

The FOTA Kit aligns with core STEM objectives by introducing students to cutting-edge technologies in embedded systems, secure communication, and automotive applications. Its design supports multi-disciplinary learning and emphasizes experiential education. Studies have shown that hands-on activities in STEM significantly enhance student engagement and comprehension of complex topics.[٦]

3.1.1 Foundational Concepts

The kit covers essential topics such as memory management, encryption algorithms (e.g., AES), and CAN Bus communication, which are fundamental in courses on embedded systems and automotive engineering.[٧]

By using accessible hardware such as Raspberry Pi and STM32 microcontrollers, the kit makes these advanced technologies approachable for learners at various levels.

The integration of AUTOSAR (Automotive Open System Architecture) [8] with the Raspberry Pi 4 offers significant potential for advancing automotive technologies, especially in the field of autonomous vehicle development. Using the Raspberry Pi 4 as a flexible platform enables the implementation of key features such as real-time data processing and computer vision, that are critical components in the current automotive systems

3.1.2 Project-Based Learning

Students gain hands-on experience in critical areas such as designing bootloaders, configuring CAN networks, and implementing secure over-the-air firmware updates. This project-based approach enhances learning outcomes by encouraging exploration and innovation.[٩]

3.1.3 Interdisciplinary Applications

The FOTA Kit integrates concepts from computer science, electrical engineering, and cybersecurity, providing a holistic learning experience. This interdisciplinary framework prepares students for the challenges of modern IoT and automotive industries.[١٠]

3.2 Implementation in Educational Settings

Figure 1 illustrates the workflow of the Firmware Over-the-Air (FOTA) system, where:

- The system checks for new updates.
- The FOTA master microcontroller unit (MCU) requests an update package from the data center.
- The update package is downloaded to the master MCU.
- The update is securely installed on the target MCU within the vehicle.

This workflow is central to the Educational FOTA Kit, enabling students to engage in hands-on activities that replicate real-world automotive software update processes. By visualizing the system architecture, students can better understand the interaction between hardware, software, and cloud-based services in modern automotive systems.

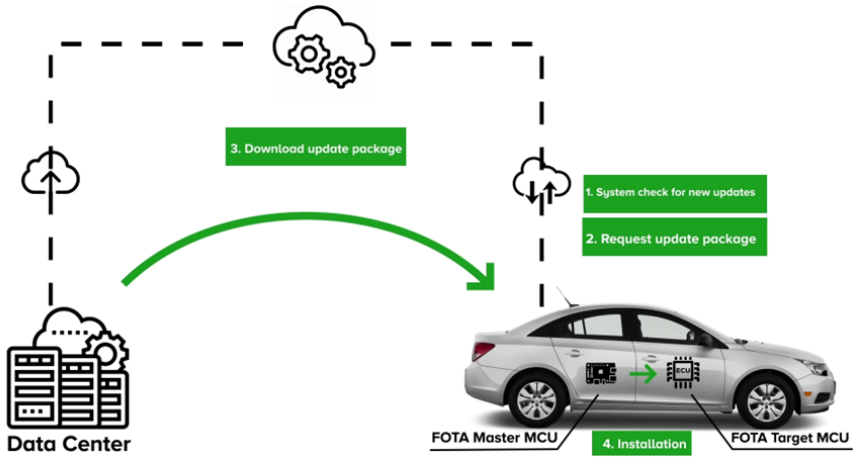

Figure 1: FOTA Workflow between the Data Center and the Vehicle

3.2.1 Laboratory Exercises

The kit includes structured exercises that guide students through tasks like configuring CAN Bus drivers, developing firmware update mechanisms, and implementing encryption protocols. These tasks reinforce theoretical concepts through practical application.

3.2.2 Capstone and Research Projects

The FOTA Kit serves as a platform for advanced student projects, enabling the design and implementation of custom automotive systems. Such projects can inspire innovation while developing critical problem-solving skills [11].

3.2.3 Collaborative Learning

By working in teams, students gain experience in collaborative problem-solving, a key skill in engineering disciplines. This also aligns with active learning strategies proven to enhance student engagement.[١٢]

3.2.4 Assessment and Feedback

Instructors can evaluate student performance through tasks such as successfully deploying firmware updates, debugging CAN communication, and implementing secure communication protocols. This feedback loop ensures alignment with educational goals.

3.3 Advantages of the FOTA Kit for STEM Education

3.3.1 Accessibility and Scalability

The hardware components of the kit are affordable and widely available, making it scalable for use in diverse educational institutions.

Its modular design allows educators to adapt the kit to different levels of complexity, from introductory to advanced courses.

3.3.2 Real-World Relevance:

The integration of technologies like CAN Bus communication and AES encryption mirrors industrial applications, bridging the gap between academia and industry.[١٣]

3.3.3 Innovation and Engagement:

The hands-on nature of the kit fosters curiosity and innovation, motivating students to explore emerging fields such as IoT, autonomous systems, and secure automotive technologies.[١٤]

The CAN Bus module introduces students to automotive communication protocols. By configuring standard and extended CAN frames, students learn to prioritize messages and troubleshoot errors in multi-device networks. The hands-on nature of this module helps students understand the intricacies of reliable data transmission.

Security protocols are another critical aspect of the kit. Students implement AES encryption and SHA-256 hashing to secure firmware updates and verify data integrity.

These exercises provide a practical introduction to cybersecurity concepts, emphasizing their importance in embedded systems. The flowchart of the system is shown in Figure 2.

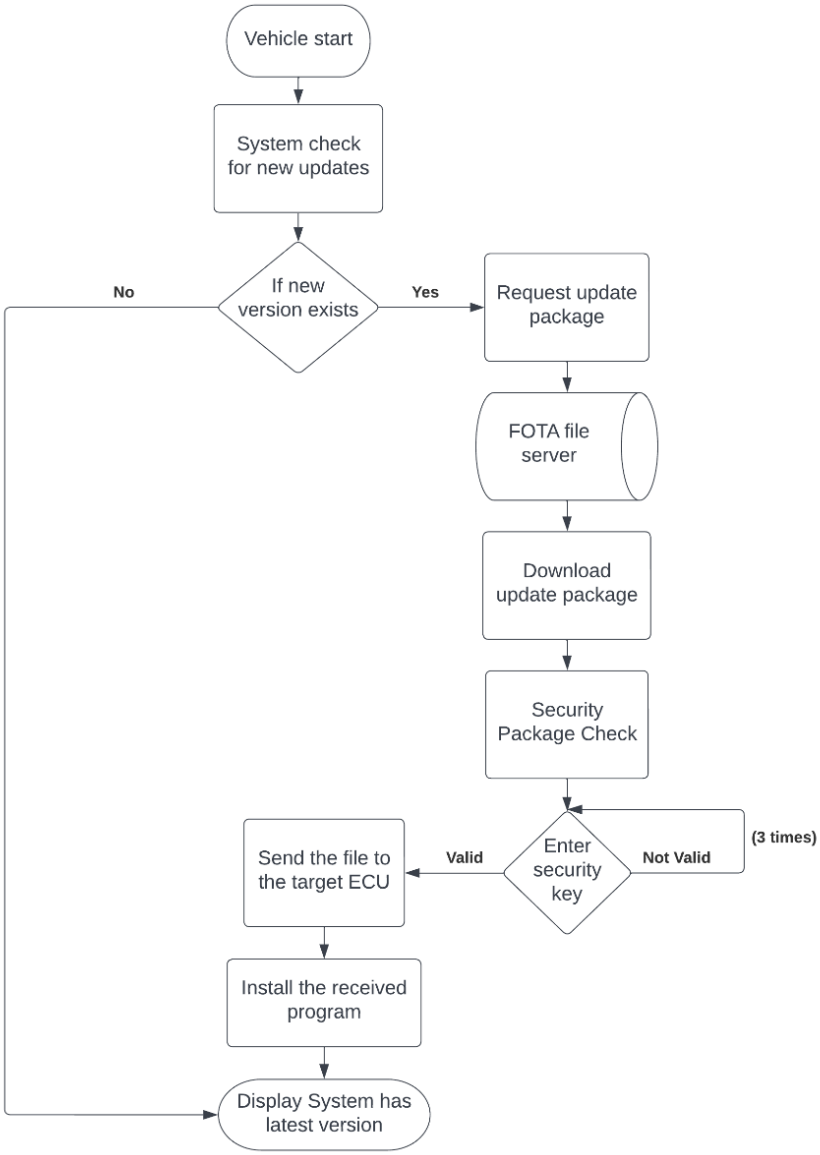

Figure 2: Kit Flow Chart

As illustrated in Figure 2, the system begins by checking for new firmware updates as part of the vehicle's startup process. It connects to the FOTA server and compares the firmware ID currently installed on the vehicle with the version available on the server. If any updates are detected, the kit automatically initiates a request to download the firmware updates package.

Once the downloading process of the package updates is complete, the system applies a security check to verify the integrity of the file. This process is followed by asking the user to enter a password to confirm their authorization before proceeding with the installation. Based on the user's approval, the update is sent to the target ECU for installation. After completing the installation process, a confirmation message is displayed to inform the user that the vehicle is now running to the latest firmware version.

If the password entered is incorrect, the system allows up to three attempts before canceling the update request. It is important to note that all of these operations will not proceed if the vehicle is in motion to ensure safety during the update process.

4. Methodology and Design

This section outlines the iterative design and implementation process for the Educational FOTA Kit, covering hardware configuration, bootloader development, flash programming, and software integration. This study adopted the Agile Software Development model for kit implementing, starting with bootloader development then flash programming and CAN network, till the Infotainment System.

4.1 Hardware Configuration

The hardware kit includes a Raspberry Pi 4 acting as the FOTA master ECU, STM32F103 microcontrollers as target ECUs, and a serial-to-CAN converter for communication. The Raspberry Pi 4 also functions as a decryption engine and user interface host, utilizing an LCD touchscreen to emulate a vehicle infotainment system. Figure 3 and Figure 4 show the schematic diagram and the simplified circuit diagram of the FOTA kit respectively. The schematic diagram includes microcontrollers, CAN communication modules, and associated power connections. This diagram represents a setup for a distributed control system or communication network, where microcontrollers exchange information over the CAN bus for reliable and real-time data transfer.

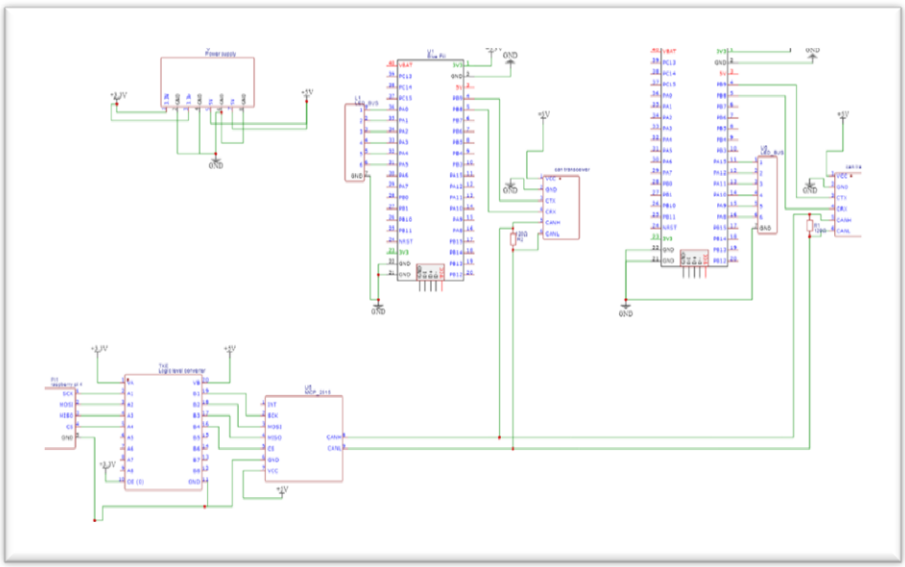

Figure 3: Kit Schematic Diagram

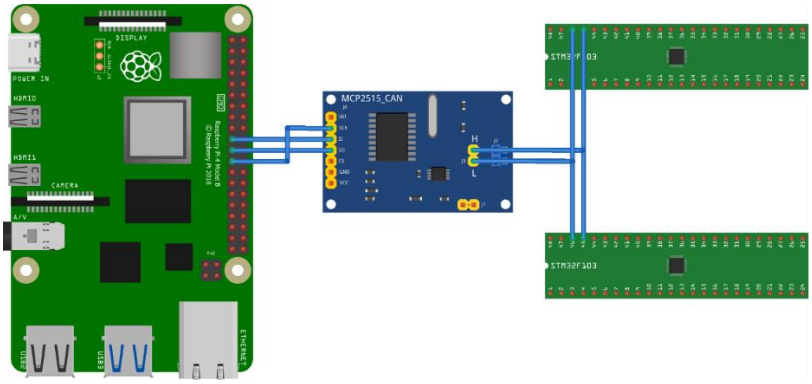

Figure 4: Simplified Circuit Diagram of the FOTA Kit

4.2 Bootloader Development

The bootloader development process began with configuring the GNU C Toolchain, as shown in Figure 5, and modifying the linker script to allocate a dedicated RAM section for bootloader execution, enabling independent operation from ROM .

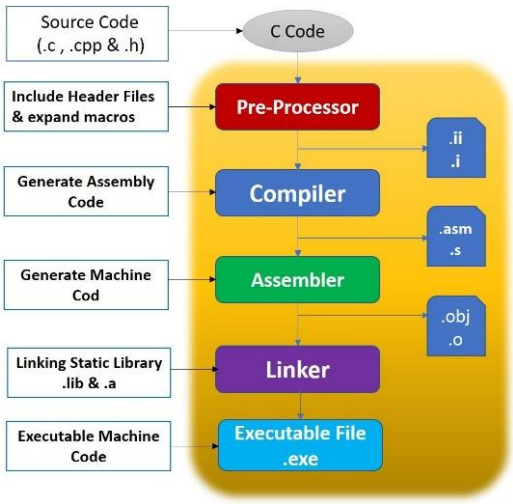

Figure 5: GNU C Toolchain

While working with linker scripts in STM32CubeIDE and Eclipse, various challenges were encountered, including errors when attempting to modify or create new sections within the linker script. To address these issues, the development process transitioned to Keil IDE, where scatter files were utilized as a replacement for linker scripts, providing greater reliability and simplicity. This transition is illustrated in Figures 6 and 7, which show the structure and configuration of a script.Id file in Eclipse.

```
/* Entry Point */
ENTRY(Reset_Handler)

/* Highest address of the user mode stack */
_estack = ORIGIN(RAM) + LENGTH(RAM);    /* end of "RAM" Ram type memory */

_Min_Heap_Size = 0x200; /* required amount of heap  */
_Min_Stack_Size = 0x400;    /* required amount of stack */

/* Memories definition */
MEMORY
{
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 36K
  FLASH    (rx)    : ORIGIN = 0x8000000,   LENGTH = 128K
}

```

Figure 6: Initial Section of the Script.Id file in Eclipse

```
  .mysection :
  {
    . = ALIGN(4);
    __mysection_start__ = .;
    *(.mysection*)
    __mysection_end__ = .;
  } > MY_MEMORY
```

Figure 7: Additional Section of the Script.Id file in Eclipse

Using scatter files, a dedicated RAM section was created specifically for the bootloader, as depicted in Figure 8. This setup ensured effective memory segmentation and allowed the bootloader to operate independently of the ROM.

```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************

LR_IROM1 0x08000000 0x00010000  {    ; load region size_region
  ER_IROM1 0x08000000 0x00010000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
   .ANY (+XO)
  }
  RW_IRAM1 0x20000000 0x00010000  {  ; RW data
   .ANY (+RW +ZI)
  }

}
```

Figure 8: Scatter file configuration in Keil IDE (file.sct)

The scatter file specifies the RAM and ROM sections along with their access permissions, starting addresses, and sizes. It also facilitates essential operations such as copying the vector table into RAM, which prevents bus errors during simultaneous ROM access. To implement this functionality, a new section was added for the vector table, and the table was transferred into RAM using SCB_VTOR, as demonstrated in Figures 9–12.

```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************

LR_IROM1 0x08000000 0x00010000  {    ; load region size_region
  ER_IROM1 0x08000000 0x00010000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
   .ANY (+XO)
  }
  RW_IRAM1 0x20000000 0x00004000  {  ; RW data
   .ANY (+RW +ZI)
  }
  RAM_FUNC 0x20004000 0x00001000{
     .ANY (.myramfunction)
        }
}
```

Figure 9: Allocation of a new RAM section for the bootloader.

```
}
RW_IRAM1 0x20000000 0x00003000  {  ; RW data
 .ANY (+RW +ZI)
}

RAM_FUNC 0x20003000 0x00001000{
   .ANY (.myramfunction)
       }
myvtor 0x20004000 0x0000800{
   .ANY (.VCTORTABLE_)
       }
}
```

Figure 10: Adding a section for the vector table in RAM

```
16 static u32  vctTable[300]__attribute__((__section__(".VCTORTABLE_")))={0};
```

Figure 11: Defining an array to store the copied vector table

```
106   __asm volatile ("cpsid i" );  /* Disable gloabal Interrupt */
107
108   *SCB_VTOR = 0x20004000;
109   ptr= Old_Vtor_Address;
110
111   for (i=0;i<300;i++){          /* COPY VECTOR TABLE FROM data ARRAY INTO SRAM */
112     vctTable[i] = ptr[i];
113   }
114
115   __asm volatile ("cpsie i" );  /* Enable gloabal Interrupt */
```

Figure 12: Transferring the vector table to RAM

To ensure system reliability, vector table copying was implemented as a critical section, disabling the global interrupted flag during execution. This approach minimized unpredictable behavior and ensured smooth bootloader operation.

Developing the bootloader offers students an opportunity to explore memory management and debugging techniques in embedded systems. By encountering and resolving issues with linker scripts, students gain valuable problem-solving skills applicable to real-world software development scenarios.

4.3 Flash Programming and CAN Network

STM32's Flash Program and Erase Controller (FPEC) was used to perform key flash memory operations, including unlocking, erasing, and writing, as illustrated in Figure 13 .

```
181
182 int main(void){
183
184   typedef void (*Function_t)(void);
185   Function_t addr_to_call = 0;
186   static u32 CodeAddress= 0x08001004;
187   while(1){
188     if (DATA_Received==1){
189       BOOTLOADER_RAM(Page_Address,DATA_WriteAddress,DATA);
190     }
191     else if(timer == timeout){
192       addr_to_call = *(Function_t*)(CodeAddress);
193       addr_to_call(); /* Jump on code erea*/
194     }
195     timer++;
196   }
197 }
```

Figure 13: Main Code Logic

The bxCAN module, with its compatibility with CAN protocol versions 2.0A and 2.0B, supported communication via a CAN network. Key operational and test modes, such as silent and loop-back modes, were configured to validate network performance as shown in Figure 14 .

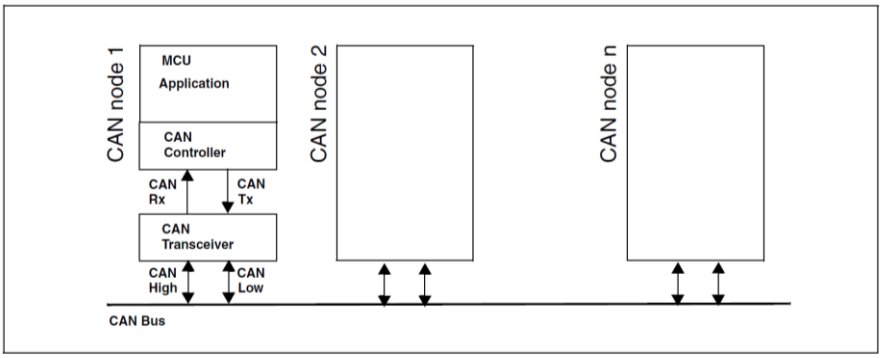

Figure 14: CAN Network Topology

4.3.1 bxCAN Main Features

The bxCAN module is a crucial component of the FOTA system, facilitating efficient communication between devices on the Controller Area Network (CAN). Its robust design ensures seamless data transfer while supporting various operational and testing modes to optimize performance.

Table 1 highlights the primary features of the bxCAN module, focusing on its transmission and reception capabilities. The transmission section details the three transmit mailboxes, configurable priority levels, and time-stamping for data frames. The reception section emphasizes the two receive FIFOs, each with three stages, and configurable FIFO overruns, which enhance data handling in high-traffic scenarios.

Additionally, the 14 filter banks enable precise message filtering, reducing CPU load and ensuring reliable operation.

These features collectively enable the bxCAN module to handle the demands of real-time automotive systems, making it an ideal choice for educational purposes in the FOTA Kit.

Table 1: Transmission and Reception Features of the bxCAN Module

Transmission	Reception
Three transmit mailboxes	Two receive FIFOs each has three stages
Configurable transmit priority	14 filter banks
Time Stamp on SOF transmission	Configurable FIFO overruns

٤,٣,٢ Operating Modes of the bxCAN Module

The bxCAN module operates in three primary modes: initialization, normal, and sleep. After a hardware reset, the CANTX pin has an active internal pull-up resistor, and the bxCAN defaults to Sleep mode to conserve energy. To transition to initialization or sleep mode, the software sets the INRQ or SLEEP bits in the CAN_MCR register. Once the desired mode is engaged, the internal pull-up is deactivated, and bxCAN confirms the transition by updating the INAK or SLAK bits in the CAN_MSR register.

In normal mode, neither INAK nor SLAK is configured, and the bxCAN must synchronize with the CAN bus before operating. Synchronization occurs when the CAN bus is idle, requiring 11 consecutive recessive bits to be detected on the CANRX pin. These operating modes are illustrated in Figure 15.

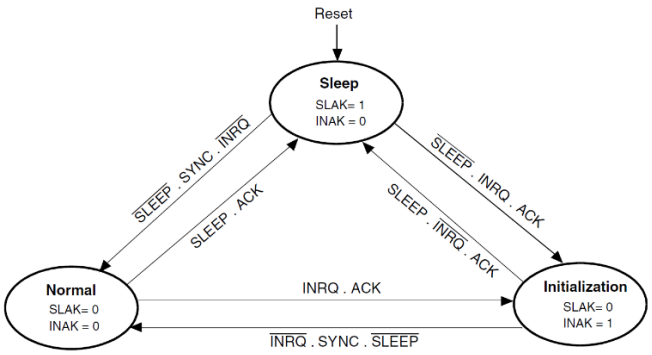

Figure 15: bxCAN's primary operational modes, including initialization, normal, and sleep.

4.3.3 bxCAN Validation and Test Modes

The bxCAN module supports several test modes to verify and validate network functionality. These modes are configured using two bits, SILM and LBKM, in the CAN_BTR register. The test mode must first be selected while the bxCAN is in initialization mode. To switch back to normal mode, the INRQ bit in the CAN_MCR register is cleared. Test modes provide a reliable way to ensure the system's proper operation under various conditions.

4.3.4 Silent Mode for Passive Monitoring

In Silent mode, the bxCAN module can receive both standard data frames and remote frames from the CAN bus but does not actively transmit dominant bits. Instead, it transmits recessive bits only, and any attempt to send a dominant bit is internally redirected to the CAN Core for observation. This mode, as depicted in Figure 16, allows for passive monitoring of the CAN bus traffic without affecting the network. Silent mode is particularly useful for debugging and analyzing network communication.

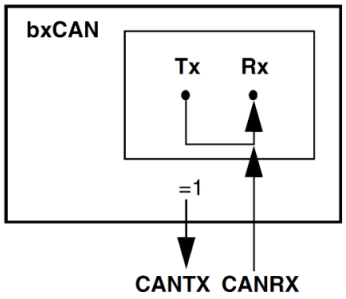

Figure 16: bxCAN operating in Silent mode for passive monitoring of CAN traffic.

4.3.5 Self-Testing with Loop Back Mode

Loop Back mode is designed for self-testing purposes, enabling the bxCAN to operate independently of external events. In this mode, the CAN Core bypasses acknowledge faults and reroutes transmitted data internally, converting the Rx input into Tx output. This functionality allows the bxCAN to ignore the actual state of the CANRX input pin while still validating transmitted messages, as shown in Figure 17. The CANTX pin can also be used to monitor outgoing messages, making this mode ideal for validating network functionality during development.

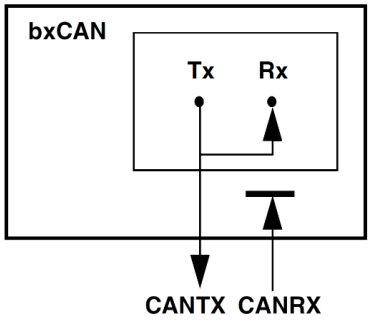

Figure 17: bxCAN in Loop Back mode, internally routing transmitted messages for self-testing

4.4 Software Components

The software architecture of the FOTA Kit comprises four key modules: a custom bootloader for firmware updates, a CAN Bus driver for communication, a Graphical User Interface (GUI) for user interaction, and a PHP-based server hosting encrypted firmware. These components work together to ensure secure, efficient, and reliable over-the-air firmware deployment.

4.4.1 Bootloader

The bootloader is a core component of the FOTA Kit, responsible for enabling the microcontroller to receive firmware updates via communication protocols such as UART, USB, or CAN. It manages the process of flashing the received firmware into the microcontroller's memory. As discussed in Section 4.3, the bootloader also integrates critical functionalities such as memory segmentation, vector table copying, and rollback mechanisms to ensure system reliability.

To enhance fault tolerance, the microcontroller's memory is divided into two sections, each containing a firmware version (e.g., v1.0 and v2.0). If a firmware update is corrupted or incomplete, the microcontroller reverts to the previous version, maintaining functionality. This redundancy ensures the system remains operational even in the event of a failed update.

4.4.2 CAN Bus Driver

The CAN Bus driver facilitates communication between the microcontroller and other devices on the Controller Area Network (CAN). Building on the details provided in Section 4.3, the CAN protocol is a widely adopted standard in the automotive industry, offering features such as cyclic redundancy checks for error detection and an identifier

field for managing multi-device communication. The CAN Bus driver enables the FOTA Kit to handle real-time data exchange with high reliability, making it essential for firmware updates and diagnostic communication.

4.4.3 Graphical User Interface (GUI)

The GUI, developed using Python's CustomTkinter library, provides a user-friendly interface for interacting with the FOTA Kit. Hosted on a Raspberry Pi with an LCD touchscreen, the GUI allows users to monitor the system, fetch firmware updates from the server, and initiate updates. By simulating an infotainment system, the GUI also helps students understand how software updates are managed in modern vehicles, bridging the gap between theoretical learning and practical application.

4.4.4 PHP-Based Server

The server, built using PHP, hosts encrypted firmware update files and handles communication between the system and the cloud. This server ensures that firmware updates are securely deployed using AES encryption and SHA-256 hashing, safeguarding data integrity and preventing unauthorized access during transmission. The encryption and decryption algorithms, discussed briefly in this section, are essential for protecting firmware updates from tampering and ensuring secure delivery.

5. Infotainment System

The infotainment system in the Educational FOTA Kit serves as the primary user interface for interacting with firmware updates. Built using a CustomTkinter-based GUI, it emulates the functionality of a real-world automotive infotainment system, providing students with hands-on experience in managing software updates. By engaging with the system, students gain a deeper understanding of how infotainment systems integrate with other automotive technologies, including bootloaders, CAN Bus communication, and servers.

5.1 Overview of Infotainment System in FOTA Kit

The GUI simulates key functionalities of modern automotive infotainment systems, including tabs for car overview, updates, and settings. Through this interface, users can:

- Monitor the vehicle’s status.
- Receive notifications for firmware updates.
- Initiate or approve installation of new firmware.

This simulation bridges theoretical learning with practical application, enabling students to understand the role of infotainment systems in modern vehicles. Critical updates, such as those addressing security vulnerabilities, are performed automatically when the system is connected to a network. Figure 18 illustrates the dark theme design of the GUI, showcasing its layout and functionality

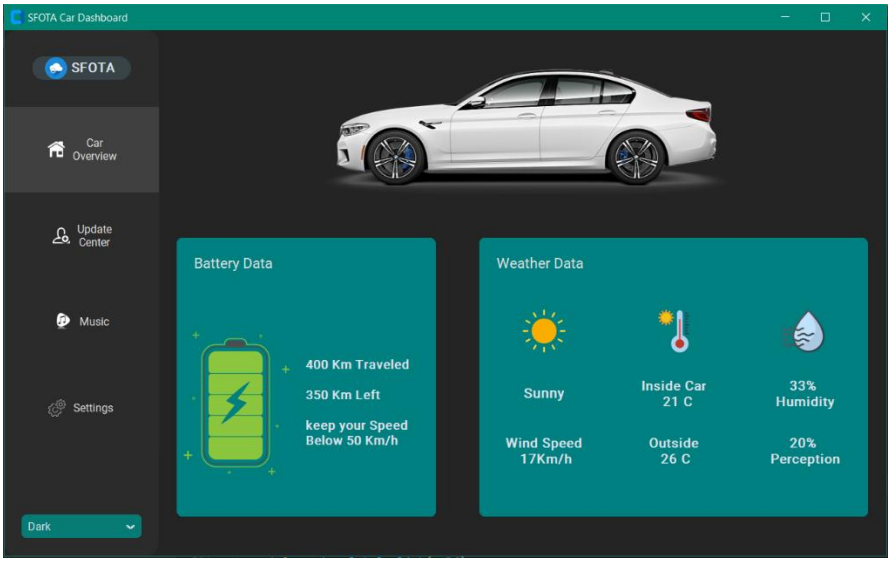

Figure 18: Dark-themed GUI of the FOTA Kit with interactive tabs.

5.2 GUI Design with CustomTkinter

CustomTkinter, an enhanced Python library, is used to develop the GUI for the FOTA Kit. It offers modern, customizable widgets with cross-platform compatibility, adapting to light or dark themes and HighDPI scaling. These features make CustomTkinter suitable for creating visually appealing and interactive interfaces.

The GUI includes four main tabs:

- Car Overview: Displays the vehicle's current status and system information.
- Update Center: Allows users to manage firmware updates, view notifications, and track progress.
- Music: Simulates entertainment functionalities.
- Settings: Provides options for system configuration and preferences.

The dark-themed design of the GUI, shown in Figure 18, provides an intuitive and user-friendly interface for students to engage with the system.

5.3 FOTA Kit Interface Workflow

The FOTA Kit's interface follows a systematic process to manage firmware updates:

- User Login: The system requires user authentication to access the interface.
- Server Connection: Upon login, the system connects to the FOTA server to retrieve the latest firmware version.
- Version Comparison: The retrieved version is compared with the currently installed firmware to determine if an update is available.
- Download Process: If an update exists, the system displays the download progress and retrieves the firmware package.
- Security Check: After the download, AES encryption and SHA-256 hashing verify the package's integrity.
- User Authorization: The user is prompted to enter a password to authorize the installation.
- Installation: The firmware is sent to the target ECU, installed, and verified for successful completion.
- Reboot and Confirmation: The system reboots and displays a confirmation message upon successful installation.

For critical updates, the system automates the installation process when connected to the network, bypassing user authorization to ensure timely deployment.

6. Security Measures

The Educational FOTA Kit integrates robust security measures to protect firmware updates from unauthorized access or tampering. AES encryption, implemented using the Cryptography library, ensures secure firmware transmission by employing 128-bit keys generated via SHA-256 hashing. This approach guarantees both confidentiality and data integrity during over-the-air (OTA) updates.

Specifically, the system uses AES-CCM mode, which combines encryption and message authentication to provide robust protection. By utilizing a unique initialization vector for each session, the algorithm strengthens resistance against replay attacks and other unauthorized activities. SHA-256 hashing further validates the integrity of firmware packages, ensuring that data remains authentic and unaltered throughout the transmission process.

Incorporating these security measures into the Educational FOTA Kit introduces students to essential cybersecurity concepts, such as secure boot processes, encryption protocols, and digital signatures. Firmware updates, while critical for maintaining

vehicle performance, are vulnerable to malicious interference without adequate security. By working hands-on with these techniques, students gain practical experience in implementing secure OTA updates, preparing them to address the cybersecurity challenges of modern automotive systems.

7. Conclusion and Future Work

The Educational Firmware Over-the-Air (FOTA) Kit represents an innovative and practical approach to teaching FOTA principles within STEM education. By combining accessible hardware and software components, the kit enables students to explore key concepts in automotive systems, embedded systems, and secure communication protocols. Through hands-on activities, students gain valuable skills in areas such as memory management, CAN communication, and AES encryption, equipping them for careers in technology sectors like IoT, automotive engineering, and cybersecurity.

Future enhancements to the FOTA Kit will focus on integrating wireless communication protocols, such as Wi-Fi and Bluetooth, and expanding compatibility with additional microcontroller platforms, such as ESP32 and ARM Cortex-M series. These developments will make the kit even more versatile, enabling it to adapt to the rapidly evolving technological landscape. By continuously aligning with emerging technologies, the kit will remain a valuable resource for fostering innovation and developing technical expertise among students, supporting the advancement of STEM education.

References

[١] C. Plappert and A. Fuchs, “Secure and Lightweight ECU Attestations for Resilient Over-the-Air Updates in Connected Vehicles,” in Annual Computer Security Applications Conference, New York, NY, USA: ACM, Dec. 2023, pp. 283–297. doi: 10.1145/3627106.3627202.

[٢] P. C. Yau, D. Wong, and Q. Hongying, “Educational STEM Laboratory,” in Proceedings of the 2020 6th International Conference on Education and Training Technologies, New York, NY, USA: ACM, May 2020, pp. 9–12. doi: 10.1145/3399971.3399986.

[٣] N. Nikolov, “Research Firmware Update Over the Air from the Cloud,” in 2018 IEEE XXVII International Scientific Conference Electronics - ET, IEEE, Sep. 2018, pp. 1–4. doi: 10.1109/ET.2018.8549628.

[٤] K. Kerliu et al., “Secure Over-The-Air Firmware Updates for Sensor Networks,” in 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW), IEEE, Nov. 2019, pp. 97–100. doi: 10.1109/MASSW.2019.00026.

[٥] S. and T. M. and H. M. Schmidt, “Secure Firmware Update Over the Air in the Internet of Things Focusing on Flexibility and Feasibility Proposal for a Design,” in Internet of Things Software Update Workshop (IoTSU), Jun. 2016.

[٦] P. C. Yau, D. Wong, and Q. Hongying, “Educational STEM Laboratory,” in Proceedings of the 2020 6th International Conference on Education and Training Technologies, New York, NY, USA: ACM, May 2020, pp. 9–12. doi: 10.1145/3399971.3399986.

[٧] S. Corrigan, “Introduction to the Controller Area Network (CAN) Application Report Introduction to the Controller Area Network (CAN),” 2002. [Online]. Available: www.ti.com

[٨] AuToSAR, “AN5247 Application note Over-the-air application and wireless firmware update for STM32WB series microcontrollers,” 2023. [Online]. Available: www.st.com

[٩] M. Gugole, F. Fiore, T. Rosi, G. Zendri, and A. Montresor, “Stem-Kit: An interdisciplinary approach to learning physics and computer science,” in 2023 IEEE Frontiers in Education Conference (FIE), IEEE, Oct. 2023, pp. 1–8. doi: 10.1109/FIE58773.2023.10343208.

[١٠] D. Agarwal, K. C. Teja, and R. Srinivasan, “A Novel Approach to Product Identification in Automobile Sector Using Raspberry pi 4: A Computer Controlled Production System,” in 2023 International Conference on Communication, Security and Artificial Intelligence (ICCSAI), IEEE, Nov. 2023, pp. 940–944. doi: 10.1109/ICCSAI59793.2023.10421150.

[١١] K. A. Maspul, “Exploring STEM Education for Real-World Climate Change Concerns to Empower Students as Change Agents,” Journal of Physics Education and Science, vol. 1, no. 2, p. 12, Jan. 2024, doi: 10.47134/physics.v1i2.249.

[١٢] H. Guissouma, A. Diewald, and E. Sax, “A Generic System for Automotive Software Over The Air (SOTA) Updates Allowing Efficient Variant and Release Management”.

[١٣] K. Kerliu et al., “Secure Over-The-Air Firmware Updates for Sensor Networks,” in 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW), IEEE, Nov. 2019, pp. 97–100. doi: 10.1109/MASSW.2019.00026.

[١٤] N. Nikolov, “Research Firmware Update Over the Air from the Cloud,” in 2018 IEEE XXVII International Scientific Conference Electronics - ET, IEEE, Sep. 2018, pp. 1–4. doi: 10.1109/ET.2018.8549628.