



FOTA

Graduation Project (ITI_Advanced_Track)

Team members



Mosatafa Salem



Amr Salem



Saleh Ahmed



Ahmed Saad



Bishoy Ibrahim

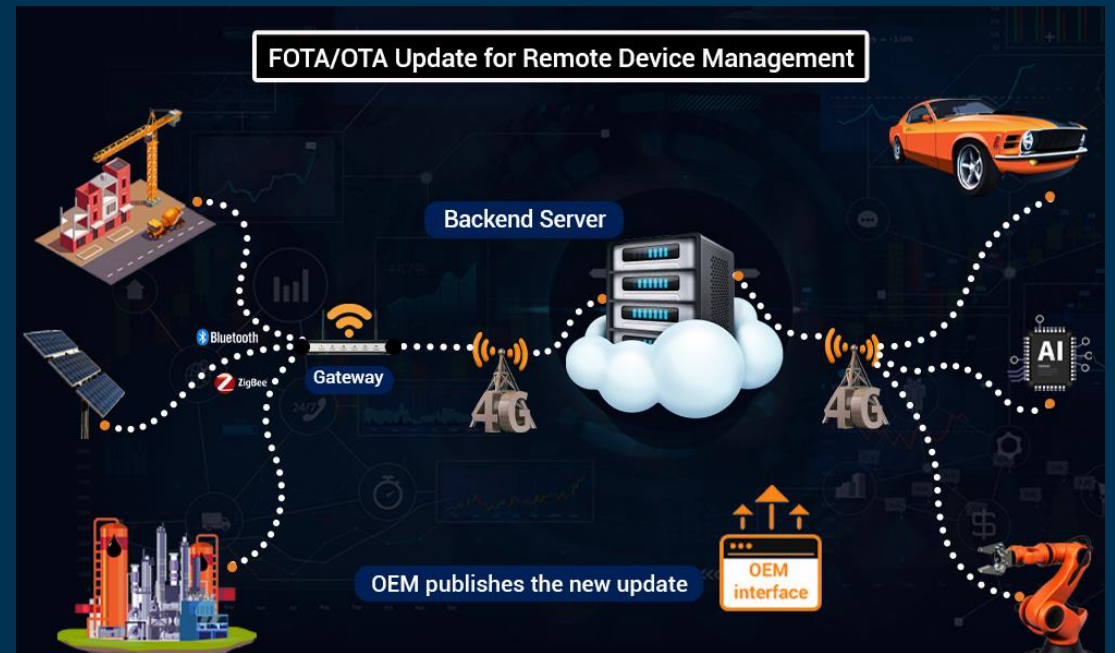
Introduction



What is FOTA ?

What is FOTA ?

- Firmware-over-the-air is a technology that enables the operating firmware of a device to be upgraded and updated wirelessly over the network (“over the air”) without the need to connect directly to the device.





Why we choose FOTA ?

Why we choose FOTA ?

Firmware Over-The-Air (FOTA) is an indispensable tool in the world of IoT, especially in scenarios where numerous interconnected devices demand frequent updates.

Consider a different scenario: A global fleet management company relies on FOTA to enhance its GPS tracking devices. With FOTA, they can remotely deploy critical firmware updates to thousands of devices across the globe. This not only saves time and resources but also ensures that all devices are up-to-date with the latest features and security patches, ultimately improving the overall service quality and customer satisfaction. Without FOTA, they would face the daunting task of physically retrieving and updating each device, leading to logistical challenges and customer inconvenience.



Configure Website




Free Hosting

Free Hosting

We utilize free web hosting services to create a domain and website that serves as a repository for our Application.

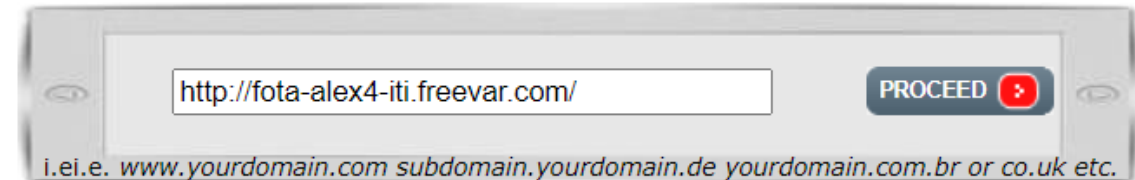
Free SubDomain Hosting



A screenshot of a web form for creating a subdomain. It features a text input field containing 'FOTA-ALEX4-ITI' with a placeholder 'i.e. yourname' below it. To the right is a dropdown menu showing 'freevar.com'. A 'PROCEED' button with a red plus icon is on the right. The text 'www.' is on the left of the input field.

Free Domain Hosting (for already registered domains)

NEW - Starting with 13.08.2018, **ALL domain extensions** are accepted, including the new, long TLDs
If you don't have your own domain, register one [here](#) then create your **free hosting account** using the form below.
Our nameservers: [ns1.freewha.com](#) and [ns2.freewha.com](#)



A screenshot of a web form for creating a domain. It features a text input field containing 'http://fota-alex4-iti.freevar.com/'. To the right is a 'PROCEED' button with a red plus icon. Below the input field, there is a line of text: 'i.e. www.yourdomain.com subdomain.yourdomain.de yourdomain.com.br or co.uk etc.'



Using File Zilla

Using File Zilla

To facilitate control and file management, we employ FileZilla, a popular FTP (File Transfer Protocol) client. This combination of web hosting and FTP tools streamlines the process of sharing and updating files within our project

Filename ^	Filesize	Filetype	Last modified		Filename ^	Filesize	Filetype	Last modifi...	Permissi...	Owner/Gr
..					..					
index.html	35,770	Chrome HTML ...	9/13/2023 10:0...		.ftpquota	13	FTPQUO...	9/13/2023 ...	0600	538302 5.
new.html	35,873	Chrome HTML ...	9/13/2023 10:0...		index.html	35,077	Chrome ...	9/13/2023 ...	0644	538302 5.
num_bytes.txt	0	Text Document	9/13/2023 11:4...		new.html	35,180	Chrome ...	9/13/2023 ...	0644	538302 5.
old.html	35,865	Chrome HTML ...	9/13/2023 10:0...		num_bytes.txt	0	Text Doc...			
PHP.png	273,010	PNG File	9/13/2023 10:1...		old.html	35,173	Chrome ...	9/13/2023 ...	0644	538302 5.
script.php	314	PHP File	9/13/2023 9:46...		script.php	294	PHP File	9/13/2023 ...	0644	538302 5.
status.txt	1	Text Document	9/13/2023 10:1...		status.txt	1	Text Doc...	9/13/2023 ...	0666	538302 5.
Selected 1 file. Total size: 0 bytes					Selected 1 file. Total size: 1 byte					



Firmware Update

Firmware Update

- Our FOTA project incorporates an innovative approach to firmware management. Through our HTML-based platform, we've developed an efficient and user-friendly interface to showcase the HEX file that drive firmware updates

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>HEX File Content</title>
</head>

<body>
  <h1>Contents of HEX File</h1>
  <pre>
    <!-- Include the contents of HEX text file here -->
    <!-- Use appropriate line breaks and formatting as needed -->
    :020000040800F2
    :10000000000000120C5020008CB020008CF02000852
    :1000100041030008450300088D03000800000000AC
    :10002000000000000000000000000000D5030008F0
```

Firmware Update

- We 've integrated PHP into our system to effortlessly configure the firmware's status as 'new' or 'old.' This flexibility allows us to efficiently manage updates, ensuring that our devices are always equipped with the latest improvements. Whether it's a fresh release or an upgrade, PHP enables us to stay agile and responsive to our users' needs. Moreover, with PHP, we can automatically check for updates once a day, ensuring that our devices are always up-to-date without manual intervention

```
<?php

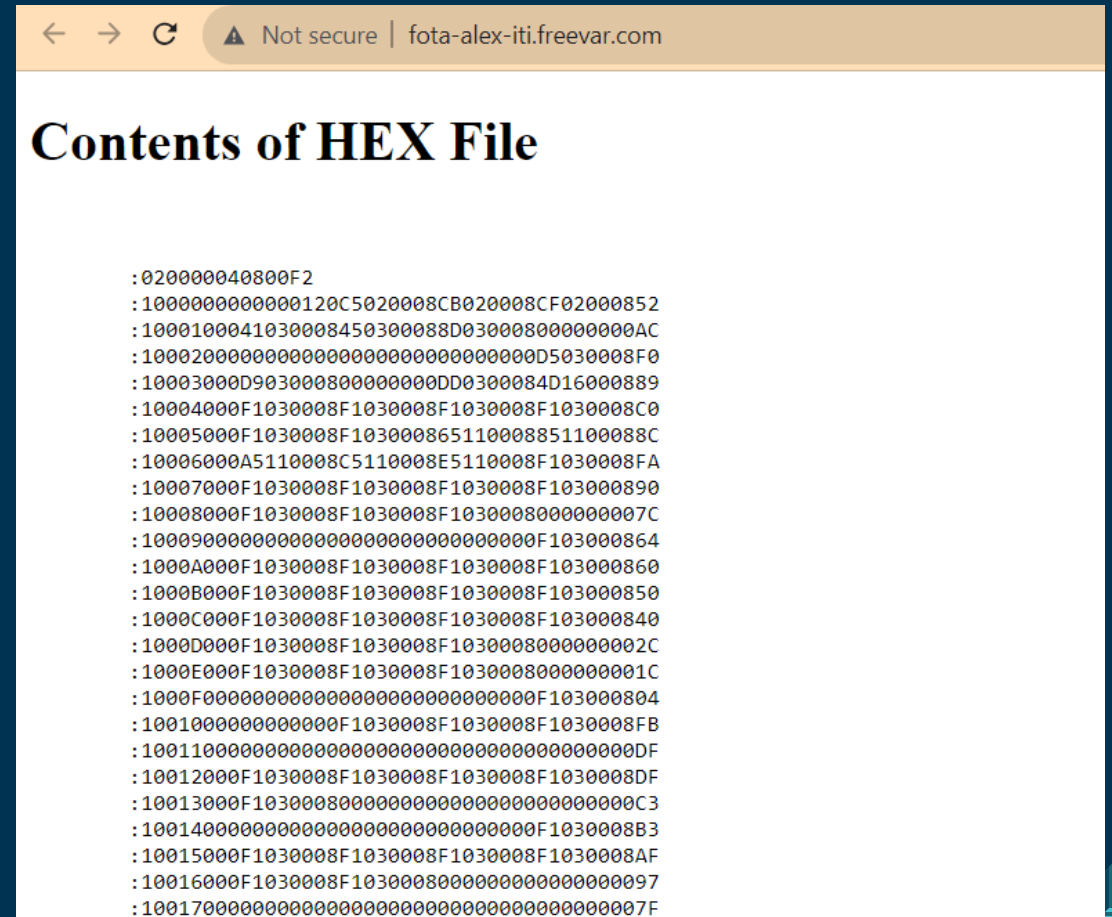
if (isset($_GET['status']))
{
    $myfile = fopen("status.txt", "w");
    if ($_GET['status'] == 'new')
    {
        fwrite($myfile, '1');
        header("location: new.html");
    }

    elseif($_GET['status'] == 'old')
    {
        fwrite($myfile, '0');
        header("location: old.html");
    }
    fclose($myfile);
}

?>
```

Firmware Update

- Our FOTA project incorporates an innovative approach to firmware management. Through our HTML-based platform, we've developed an efficient and user-friendly interface to showcase the HEX file that drive firmware updates



Firmware Update (new vs old)

← → ↻ ⚠ Not secure | fota-alex4-iti.freevar.com/new.html

:1027300061696C65643A2066696C652022257322A4
:102740002C206C696E652025640A00005374616357
:102750006B206672616D653A0A00000020523020DD
:102760003D2020253038580A00000000205231203A
:102770003D2020253038580A000000002052322029
:102780003D2020253038580A000000002052332018
:102790003D2020253038580A0000000020523132F8
:1027A000203D2020253038580A00000000204C5220DF
:1027B0003D2020253038580A0000000020504320DA
:1027C0003D2020253038580A000000002050535288
:1027D000203D2020253038580A000000004653522F73
:1027E0004641523A0A0000002043465352203D2001
:1027F00020253038580A00002048465352203D20FA
:1028000020253038580A00002044465352203D20E0
:1028100020253038580A00002041465352203D20E0
:1028200020253038580A0000204D464152203DA9
:1028300020253038580A00002042464152203D20D1
:1028400020253038580A00004D6973630A000000E3
:10285000204C522F4558435F52455455524E3D200F
:10286000253038580A0000005B4861726446617583
:102870006C745D0A000000005B4275734661756C04
:10288000745D0A005B55736167654661756C745DC4
:102890000A0000000000000000000000000102030424
:1028A00006070809232D302B2000686C4C00656654
:1028B0006745464700303132333435363738394191
:1028C00042434445460030313233343536373839A7
:0728D00061626364656600AC
:1028D8007856341280000200000000000024F40024
:1028E800140000200000000000000000000000AC
:1028F800000000000000000000000000000000D0
:10290800000000000000000000000000000000BF
:10291800000000000000000000000000000000AF
:102928000000000000000000000000000000009F
:102938000000000000000000000000000000008F
:08294800000000032547698F3
:040000050800019559
:00000001FF

To Update the Firmware [press here](#)

To make it old Firmware [press here](#)

Update Firmware Mode

← → ↻ ⚠ Not secure | fota-alex4-iti.freevar.com/old.html

:1027300061696C65643A2066696C652022257322A4
:102740002C206C696E652025640A00005374616357
:102750006B206672616D653A0A00000020523020DD
:102760003D2020253038580A00000000205231203A
:102770003D2020253038580A000000002052322029
:102780003D2020253038580A000000002052332018
:102790003D2020253038580A0000000020523132F8
:1027A000203D2020253038580A00000000204C5220DF
:1027B0003D2020253038580A0000000020504320DA
:1027C0003D2020253038580A000000002050535288
:1027D000203D2020253038580A000000004653522F73
:1027E0004641523A0A0000002043465352203D2001
:1027F00020253038580A00002048465352203D20FA
:1028000020253038580A00002044465352203D20E0
:1028100020253038580A00002041465352203D20E0
:1028200020253038580A0000204D464152203DA9
:1028300020253038580A00002042464152203D20D1
:1028400020253038580A00004D6973630A000000E3
:10285000204C522F4558435F52455455524E3D200F
:10286000253038580A0000005B4861726446617583
:102870006C745D0A000000005B4275734661756C04
:10288000745D0A005B55736167654661756C745DC4
:102890000A0000000000000000000000000102030424
:1028A00006070809232D302B2000686C4C00656654
:1028B0006745464700303132333435363738394191
:1028C00042434445460030313233343536373839A7
:0728D00061626364656600AC
:1028D8007856341280000200000000000024F40024
:1028E800140000200000000000000000000000AC
:1028F800000000000000000000000000000000D0
:10290800000000000000000000000000000000BF
:10291800000000000000000000000000000000AF
:102928000000000000000000000000000000009F
:102938000000000000000000000000000000008F
:08294800000000032547698F3
:040000050800019559
:00000001FF

To Update the Firmware [press here](#)

To make it old Firmware [press here](#)

Old Firmware

Software Architecture

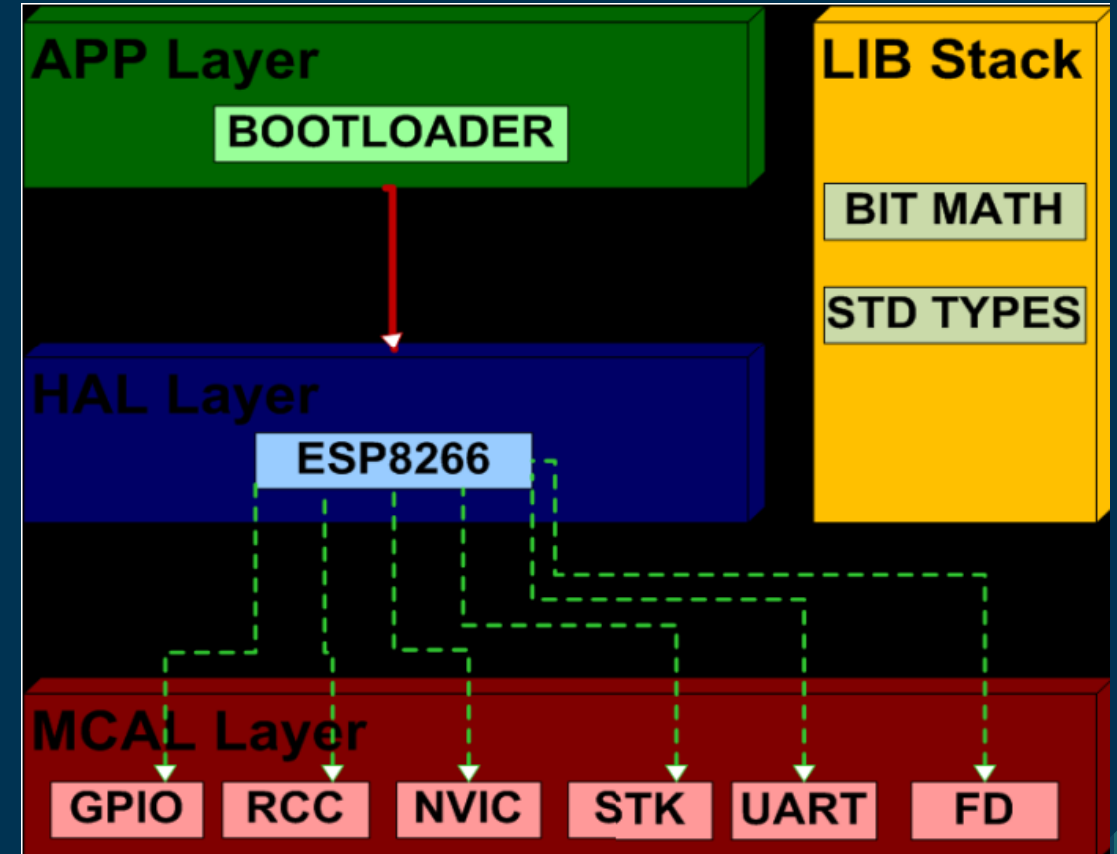
Static Layered Approach



Bootloader Firmware

Bootloader Firmware

- The bootloader is a small piece of code that runs before the main application. Its primary purpose is to facilitate the firmware update process. It's responsible for checking and verifying the integrity of the new firmware (FOTA update) and then flashing it onto the device. The bootloader is typically designed to be more robust and stable than the application firmware to ensure reliable updates





WIFI Module

HAL Layer

WIFI Module

- `/**`
- `* @brief Validate a response from a Wi-Fi module command.`
- `*`
- `* This function validates the response received from a command sent to a Wi-Fi`
- `* module. It checks for specific response patterns, such as "OK"`
 `(indicating`
- `* data reception) or "NDOK" (Indicate error), and extracts data if available.`
- `*`
- `* @param Copy_u32TimeOutm: The timeout period in milliseconds for waiting for a`
- `* response.`
- `* @param data_array: An array to store extracted data if applicable (e.g., for`
- `* +IPD data).`
- `*`
- `* @return 1 if the command is validated and successful, 0 otherwise.`
- `*/`

```
static u8 HwWIFI_u8CommandValidate(u32 Copy_u32TimeOutm, u8 *data_array) {
    u8 Local_u8Counter = 0;
    u8 Local_u8ReceivedChar = 0;

    u8 Local_u8Response[100] = {0};
    u32 Local_u32DataIndex = 0;

    while (Local_u8ReceivedChar < 128) {
        Local_u8ReceivedChar = MUART_u8ReceiveDataFromWIFI(Copy_u32TimeOutm);
        Local_u8Response[Local_u8Counter++] = Local_u8ReceivedChar;
    }

    for (Local_u8Counter = 0; Local_u8Counter < 100; Local_u8Counter++) {
        if (Local_u8Response[Local_u8Counter] == 'O' &&
            Local_u8Response[Local_u8Counter + 1] == 'K') {
            return 1;
        } else if (Local_u8Response[Local_u8Counter] == 'N' &&
                    Local_u8Response[Local_u8Counter + 1] == 'D' &&
                    Local_u8Response[Local_u8Counter + 3] == 'O' &&
                    Local_u8Response[Local_u8Counter + 4] == 'K') {
            /* +IPD,1:0CLOSED */

            while (! (
                Local_u8Response[Local_u8Counter + 16 + Local_u32DataIndex] == 'C' &&
                Local_u8Response[Local_u8Counter + 17 + Local_u32DataIndex] == 'L')) {
                data_array[Local_u32DataIndex] =
                    Local_u8Response[Local_u8Counter + 16 + Local_u32DataIndex];
                Local_u32DataIndex++;
            }

            return 1;
        } else {
            return 0;
        }
    }
    return 0;
}
```

WIFI Module

```
/**
```

```
 * @brief Initialize a Wi-Fi module.
```

```
 *
```

```
 * This function initializes a Wi-Fi  
 module by sending AT commands.  
 It ensures
```

```
 * that the module responds and  
 configures it for the desired mode.
```

```
 */
```

```
void HWIFI_voidInit(void) {  
    u8 Local_u8Output = 0;  
  
    while (Local_u8Output == 0) {  
        MUART_vSendString("AT\r\n");  
        Local_u8Output = HWIFI_u8CommandValidate(100000, 0);  
    }  
  
    Local_u8Output = 0;  
    while (Local_u8Output == 0) {  
        MUART_vSendString("AT+CWMODE=1\r\n");  
        Local_u8Output = HWIFI_u8CommandValidate(100000, 0);  
    }  
}
```

WIFI Module

/**

** @brief Connect to a Wi-Fi network.*

** This function attempts to connect to a Wi-Fi network using the provided*

** credentials.*

** @note This function assumes that the Wi-Fi module is already initialized and*

** ready for commands.*

**/*

```
void HWIFI_voidConnectToNetwork(void) {
    u8 Local_u8Output = 0;

    while (Local_u8Output == 0) {
        MUART_vSendString("AT+CWJAP_CUR=\"HUAWEI Y9 2019\\", \"12312345\\\"\\r\\n");
        Local_u8Output = HWIFI_u8CommandValidate(100000, 0);
    }
}
```


WIFI Module

*/***

** @brief Connect to a remote server via TCP.*

** This function attempts to establish a TCP connection with a remote server*

** using the provided URL and port.*

** @note This function assumes that the Wi-Fi module is already initialized and*

** connected to a Wi-Fi network.*

**/*

```
void HWIFI_voidConnectToServer(void) {
    u8 Local_u8Output = 0;

    while (Local_u8Output == 0) {
        MUART_vSendString(
            "AT+CIPSTART=\"TCP\", \"http://fota-alex4-iti.freevar.com/\", 80\r\n");
        Local_u8Output = HWIFI_u8CommandValidate(100000, 0);
    }
}
```

WIFI Module

- `/**`
- `* @brief` *Get firmware status data from a remote server.*
- `*`
- `* This function connects to a remote server and retrieves firmware status data`
- `* from a specific URL. The status data indicates whether the firmware is "new"`
- `* (1) or "old" (0).`
- `*`
- `* @param` *data_array: An array to store the retrieved firmware status data.*
- `* @return` *1 if the data retrieval is successful, 0 otherwise.*
- `*/`

```
u8 HWIFI_u8Getstatus(u8 *data_array) {  
    u8 status;  
  
    HWIFI_voidConnectToServer();  
  
    MUART_vSendString("AT+CIPSEND=50\r\n");  
    status = HWIFI_u8CommandValidate(50000, 0);  
  
    MUART_vSendString("GET http://fota-alex4-iti.freevar.com/status.txt");  
  
    /* 1 indicate new firmware and 0 old firmware*/  
  
    status = HWIFI_u8CommandValidate(100000, data_array);  
    return status;  
}
```

WIFI Module

- */***
- ** @brief Retrieve the number of bytes for a firmware update from a remote server.*
- ***
- ** This function connects to a server and retrieves the number of bytes for a firmware update from a remote*
- ** server by sending appropriate AT commands to the ESP8266 module.*
- ***
- ** @param data_array: A pointer to a variable where the retrieved number of bytes will be stored.*
- ***
- ** @return 1 if the retrieval is successful, 0 otherwise.*
- **/*

```
u8 HWIFI_u8GetNumBytes(u32 *data_array) {  
    u8 status;  
  
    HWIFI_voidConnectToServer();  
  
    MUART_vSendString("AT+CIPSEND=53\r\n");  
    status = HWIFI_u8CommandValidate(50000, 0);  
  
    MUART_vSendString("GET http://fota-alex4-iti.freevar.com/num_bytes.txt");  
  
    status = HWIFI_u8CommandValidate(100000, data_array);  
    return status;  
}
```

WIFI Module

- */***
- ** @brief Retrieve the new firmware's HEX data from a remote server based on status.txt.*
- ***
- ** This function connects to a server and retrieves the new firmware's HEX data from a remote*
- ** server if status.txt indicates an update is available (status = 1) by sending appropriate AT*
- ** commands to the ESP8266 module.*
- ***
- ** @param data_array: A pointer to an array where the retrieved HEX data will be stored.*
- ***
- ** @return 1 if the retrieval is successful, 0 otherwise.*
- **/*

```
u8 HWIFI_u8GetHexData(u8 *data_array) {
    u8 status;
    u8 *update;
    u32 *num_bytes;
    HWIFI_voidConnectToServer();
    /*if status.txt have 1 then update with new firmware*/
    if (HWIFI_u8Getstatus(update)) {
        status = HWIFI_u8GetNumBytes(num_bytes);
        // Construct the AT command to set the number of bytes to retrieve
        // char at_command[50];
        // snprintf(at_command, sizeof(at_command), "AT+CIPSEND=%lu\r\n",
        // *num_bytes); MUART_vSendString(at_command);

        MUART_vSendString("AT+CIPSEND=50\r\n");
        status = HWIFI_u8CommandValidate(50000, 0);
        MUART_vSendString("GET http://fota-alex4-iti.freevar.com/hex_file.hex");

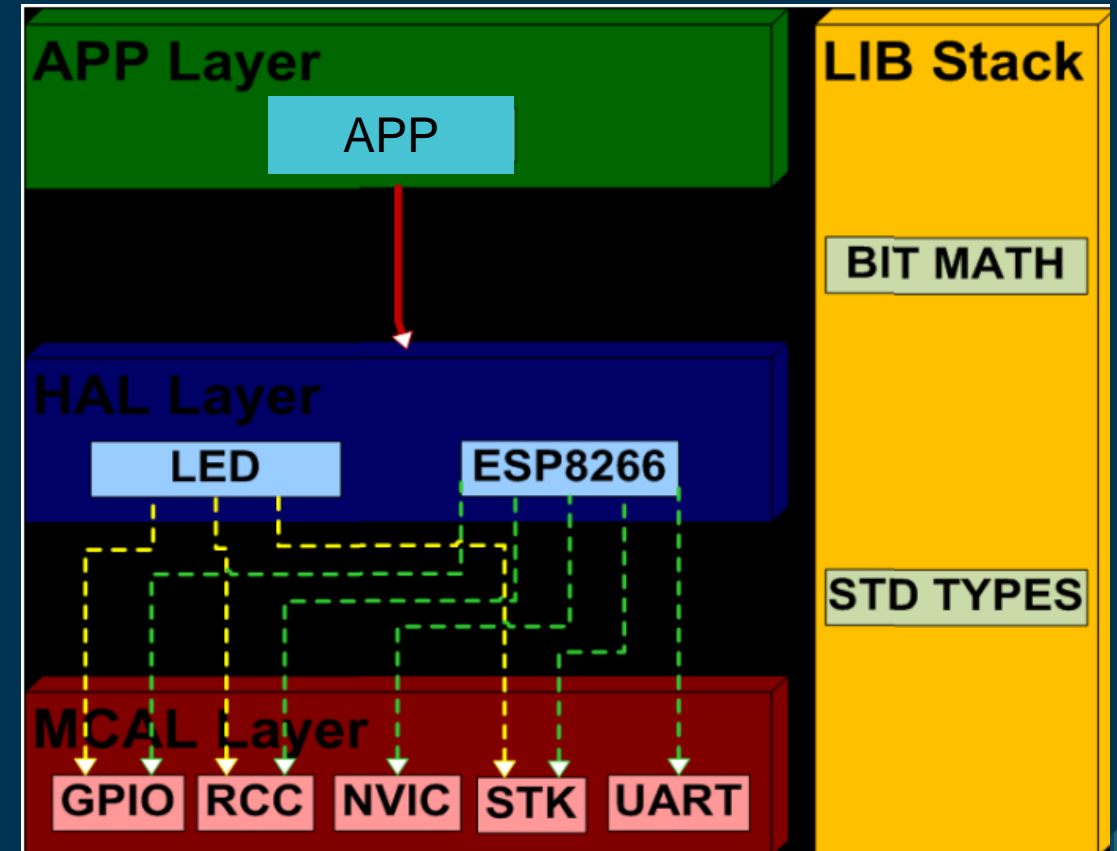
        status = HWIFI_u8CommandValidate(100000, data_array);
    }
    return status;
}
```



Application Firmware

Application Firmware

- This is the main firmware that runs on your device and provides its intended functionality. It's the software that you want to update

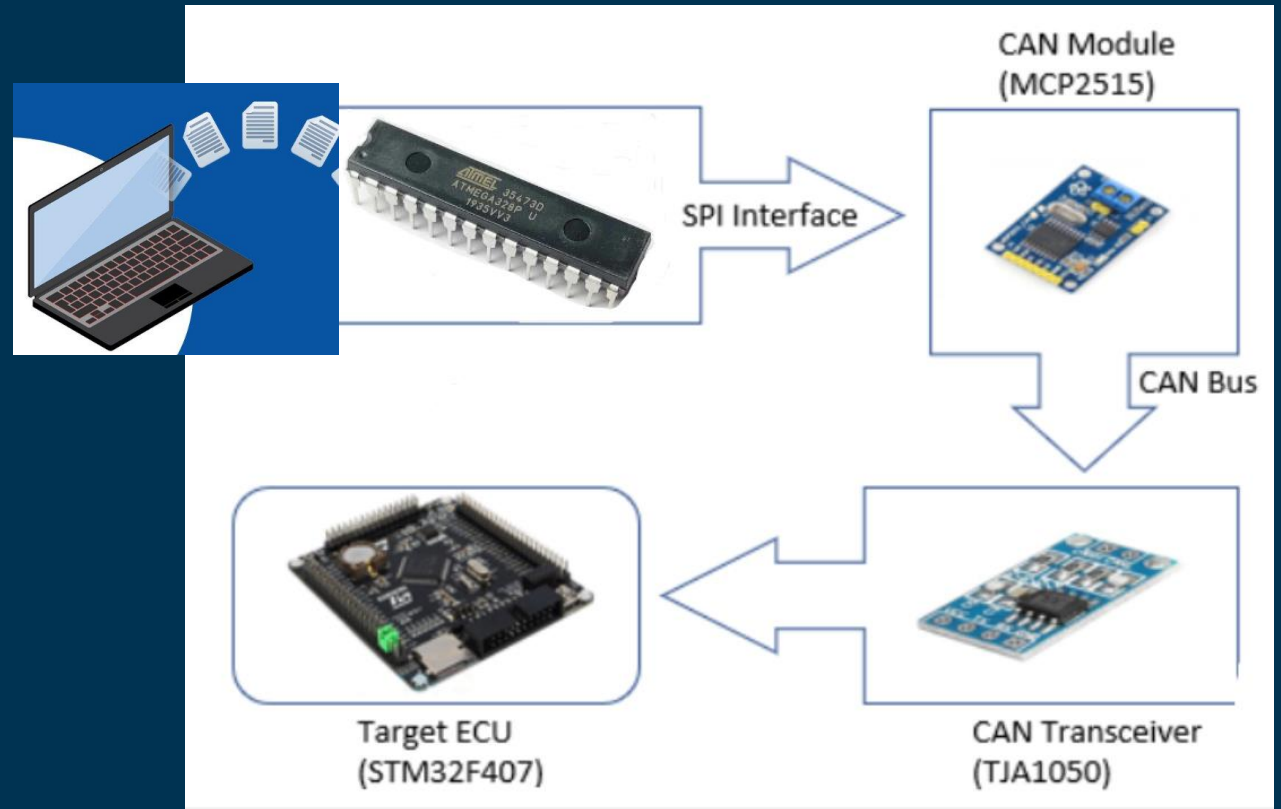


Booting using CAN

Second Project

Booting using CAN

- Source PC: The PC that has the app.hex firmware file and a USB-to-TTL converter to communicate with the ATmega32.
- ATmega32: Acts as an intermediary between the source PC and the CAN network. It receives the firmware (app.hex) from the source PC through USB-to-TTL communication.
- CAN Module (MCP2515): A CAN controller that interfaces with the ATmega32. It's responsible for sending and receiving CAN messages.
- CAN Transceiver (TJA1050): This component is used to convert the logic levels of the CAN controller (MCP2515) into the differential voltage levels required for transmission over a CAN bus.
- Target ECU (STM32F407): The microcontroller that will receive and install the new firmware.





Booting Process

Booting Process

