OTA

Over the air update for smart vehicles

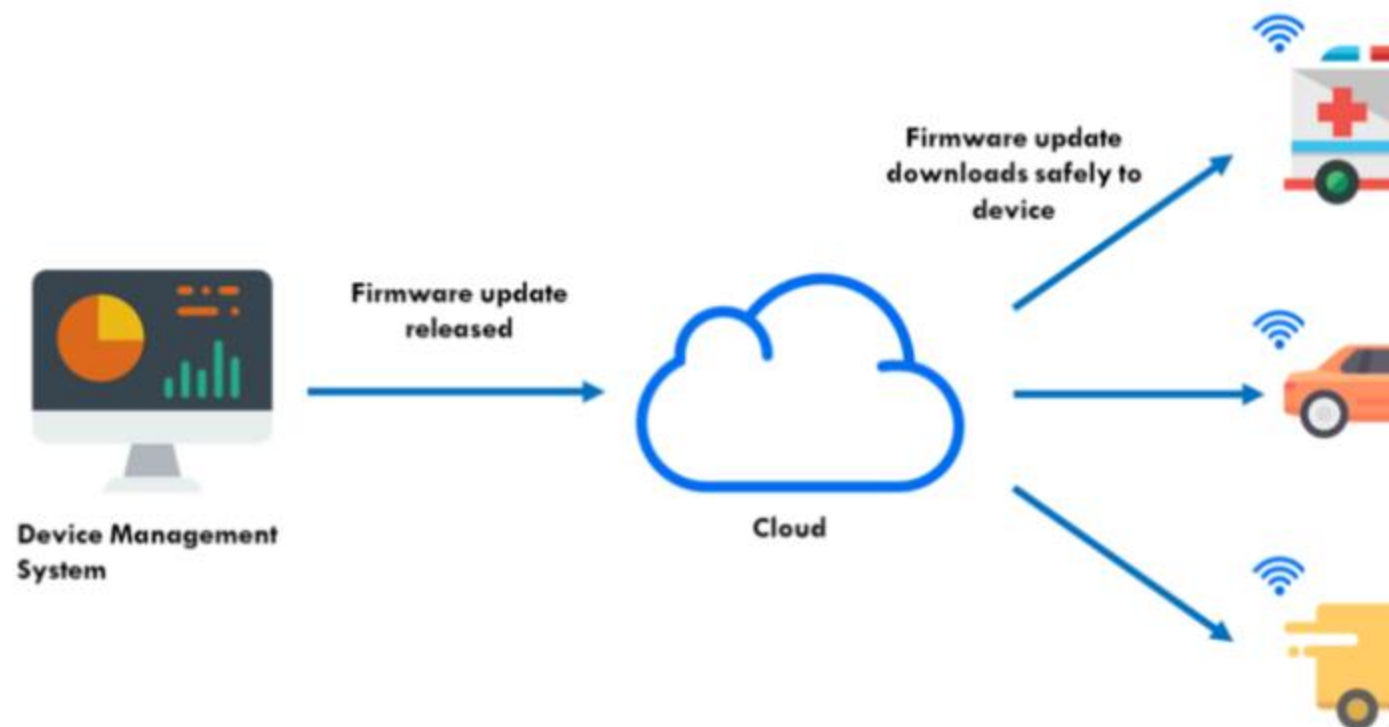Firmware update
released

Device Management
System

Cloud

Firmware update
downloads safely to
device

Supervised by Dr. Hesham Helmy Gad

- Belal Mohamed Abd Elhamid Elmahdy

- Sahar Hossam Elnagar

- Nada Gamal Elsayed

- Hager Mohamed Mohamed

- Yomna Mohamed Mokhtar

Team Members

Project Details

- Advantages
- Work Flow
- GUI
- Challenges
- Boot-loader
- SW design
- Cloud Connection
- Internal Network

Advantages

- Compelling new features to be added to the vehicle at any time.

- Bug fixations

- Save manufacturers money.

Work Flow

Overview of Update Flow

Firmware Control

ECU ID	Current Rev	Update Pending?
ECU_A	v01	N
ECU_B	v04	N
ECU_C	v10	Y

OEM OTA Cloud Servers

Telematics Unit

Gateway (OTA Manager)

ECU A

ECU B

ECU C

Encrypted Update & Rollback Images

NAND Storage

ECU_A_Diff_v01
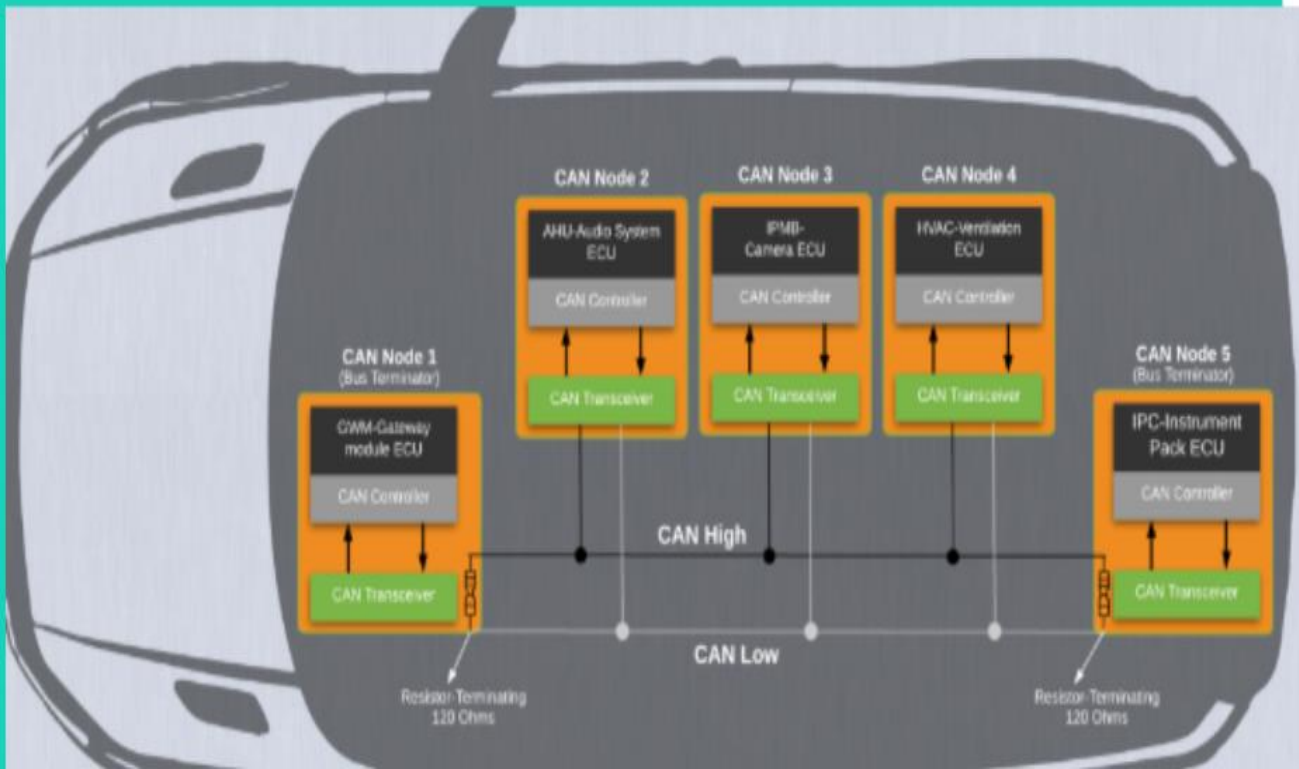ECU_B_Bin_v04
ECU_C_Diff_v11
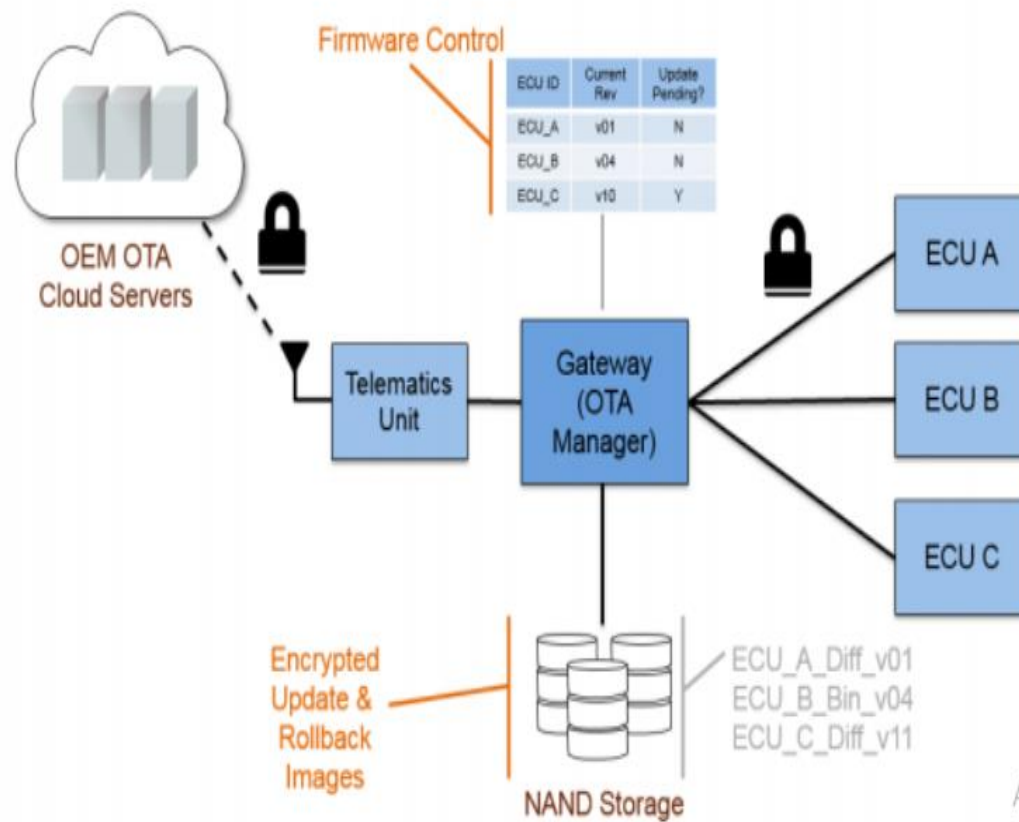
Challenges

- **Securing the OTA update process .**

- **Memory Management .**

- **Time Management .**

- **Complex Software design .**

- **Boot-loader App Design .**

Car Internal Network

What is CAN bus?

network inside a typical modern car where control commands for various components inside the vehicle travel

Car ECUs

CAN bus Overview

With & Without CAN

CAN Node 2 — AHU-Audio System ECU — CAN Controller — CAN Transceiver

CAN Node 3 — IPMB-Camera ECU — CAN Controller — CAN Transceiver

CAN Node 4 — HVAC-Ventilation ECU — CAN Controller — CAN Transceiver

CAN Node 1 (Bus Terminator) — GWM-Gateway module ECU — CAN Controller — CAN Transceiver

CAN Node 5 (Bus Terminator) — IPC-Instrument Pack ECU — CAN Controller — CAN Transceiver

CAN High

CAN Low

Resistor-Terminating 120 Ohms

Resistor-Terminating 120 Ohms

Car computer components

- Modern cars consist of a number of different computer components

- Called Electronic Control Units (ECUs)

- A typical car contains from 20-100 ECUs, with each ECU being responsible for one or more particular features of the vehicle

- DCU (Door Control Unit) is the ECU that controls and monitors various accessories in the car door

- Driver DCU offers features like automatic window movement, close-open door, mirror folding, child lock safety, and mirror adjustment
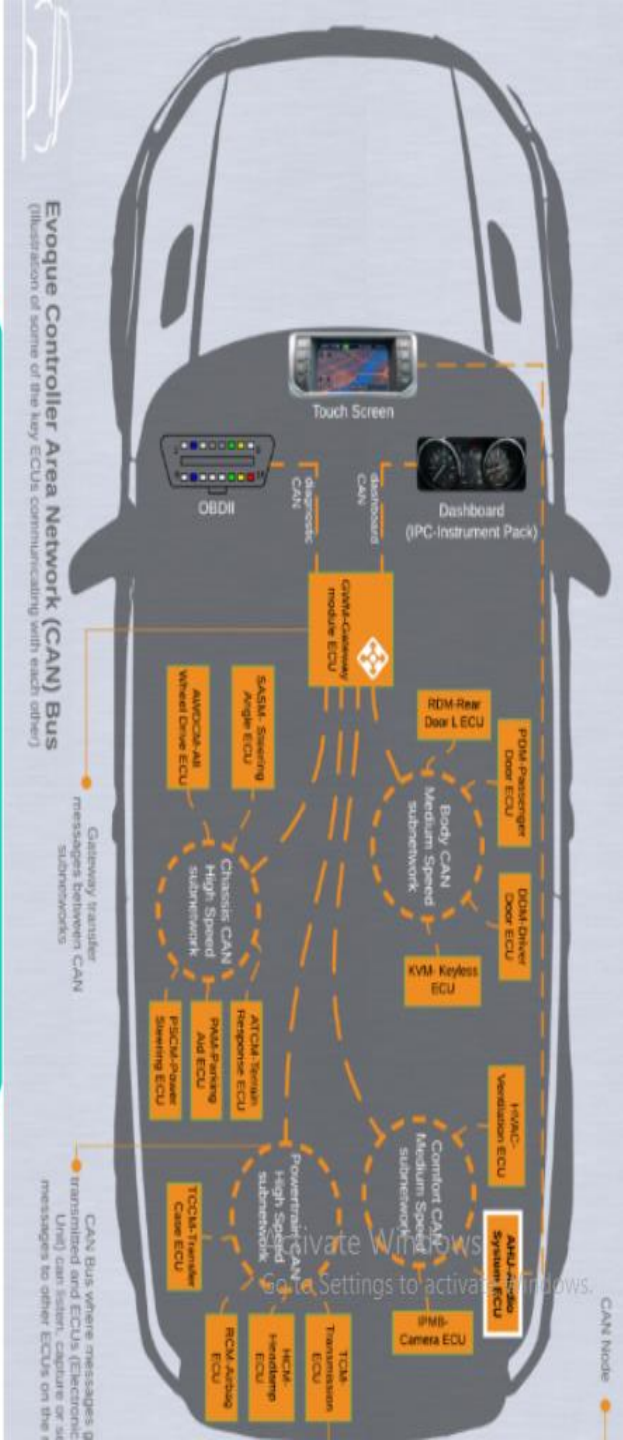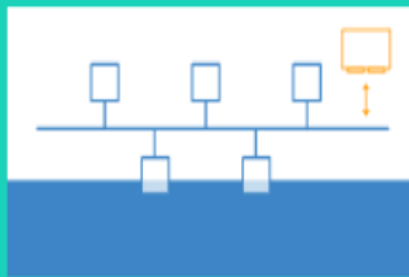
Controller Area Network
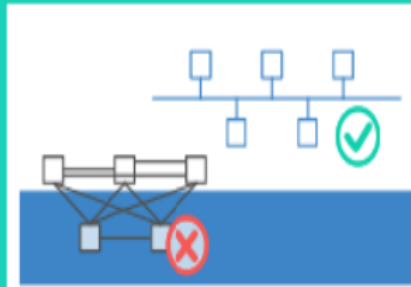
Simple & low cost

ECUs communicate via a single CAN system instead of via direct complex analogue signal lines - reducing errors, weight, wiring and costs
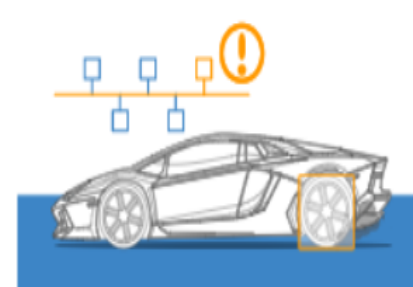
Fully centralized

The CAN bus provides 'one point-of-entry' to communicate with all network ECUs - enabling central diagnostics, data logging and configuration

Extremely robust

The system is robust towards electric disturbances and electromagnetic interference - ideal for safety critical applications (e.g. vehicles)

Efficient

CAN frames are prioritized by ID so that top priority data gets immediate bus access, without causing interruption of other frames
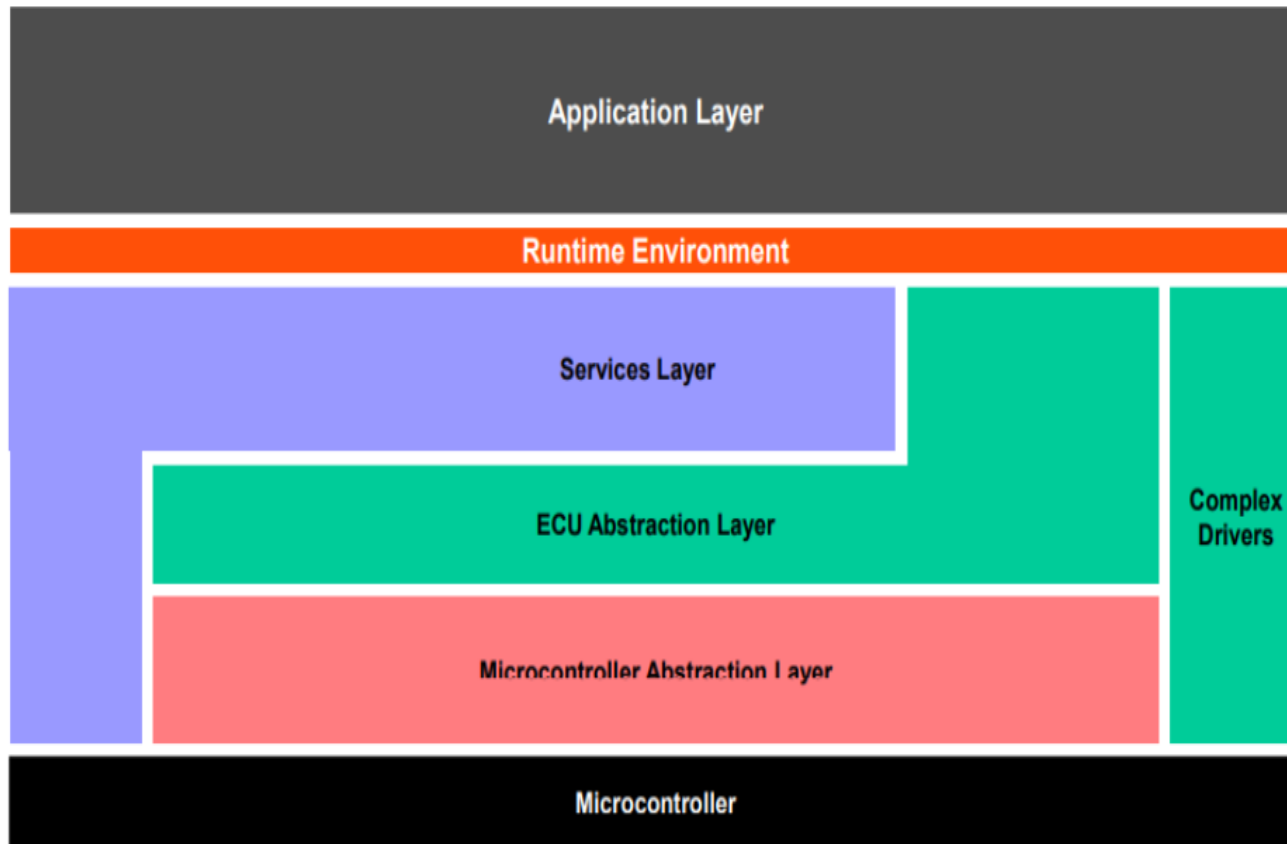
With & Without CAN

- The CAN bus system enables each ECU to communicate with all other ECUs - without complex dedicated wiring.

- Specifically, an ECU can prepare and broadcast information (e.g. sensor data) via the CAN bus (consisting of two wires, CAN low and CAN high).

- The broadcasted data is accepted by all other ECUs on the CAN network - and each ECU can then check the data and decide whether to receive or ignore it.
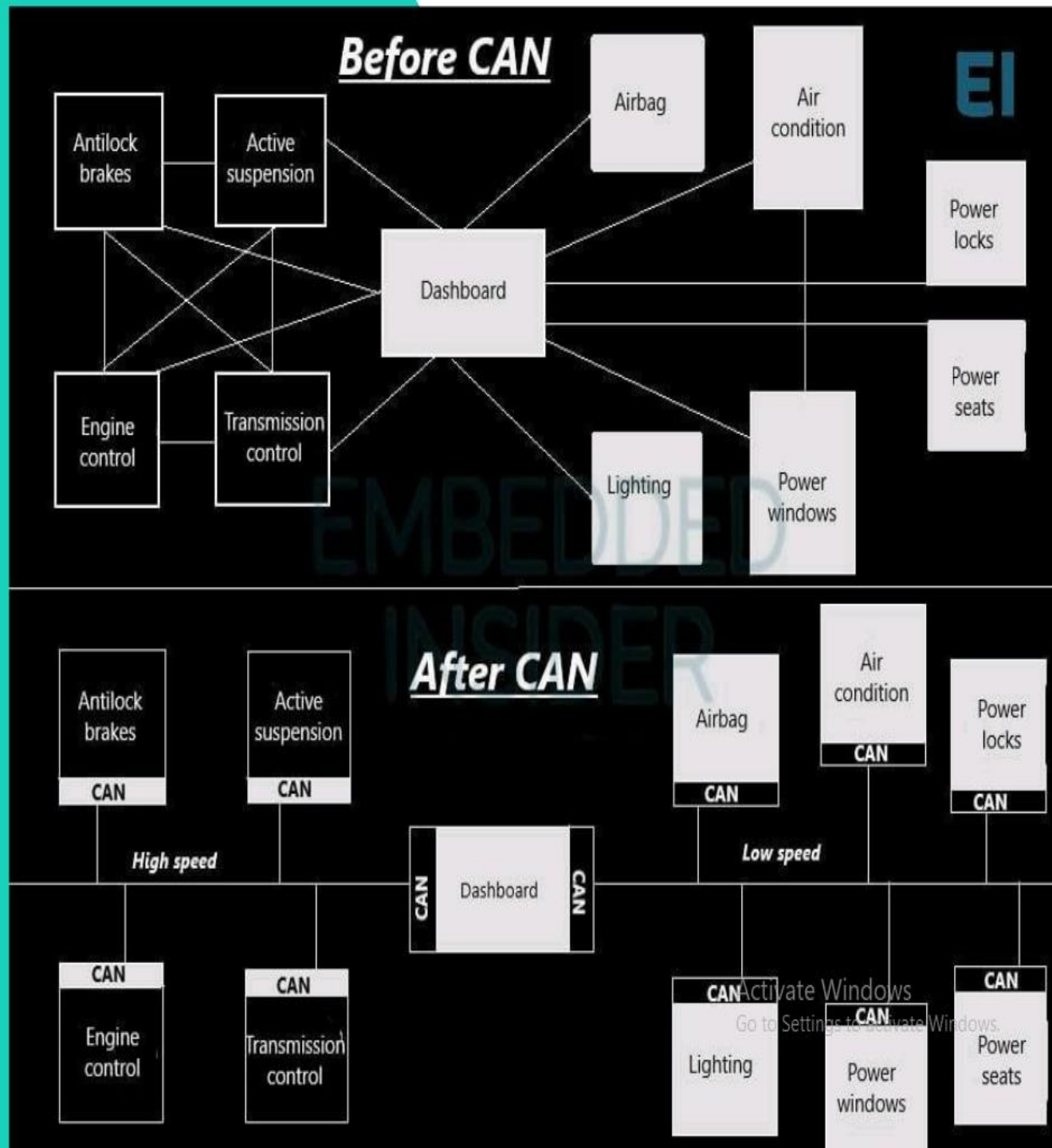
Software Architecture Design

The AUTOSAR Basic Software is further divided in the layers: Services, ECU Abstraction, Microcontroller Abstraction and Complex Drivers.

Application Layer

Runtime Environment

Services Layer

ECU Abstraction Layer

Complex Drivers

Microcontroller Abstraction Layer

Microcontroller

SW Problems

SW design solutions

Increase memory life time

Memory stack

Old Software design Problem

Single ECU
Single CPU

Time : Help !!

Memory : Many erase cycles due modifications !!

CAN Bus

WIFI connection
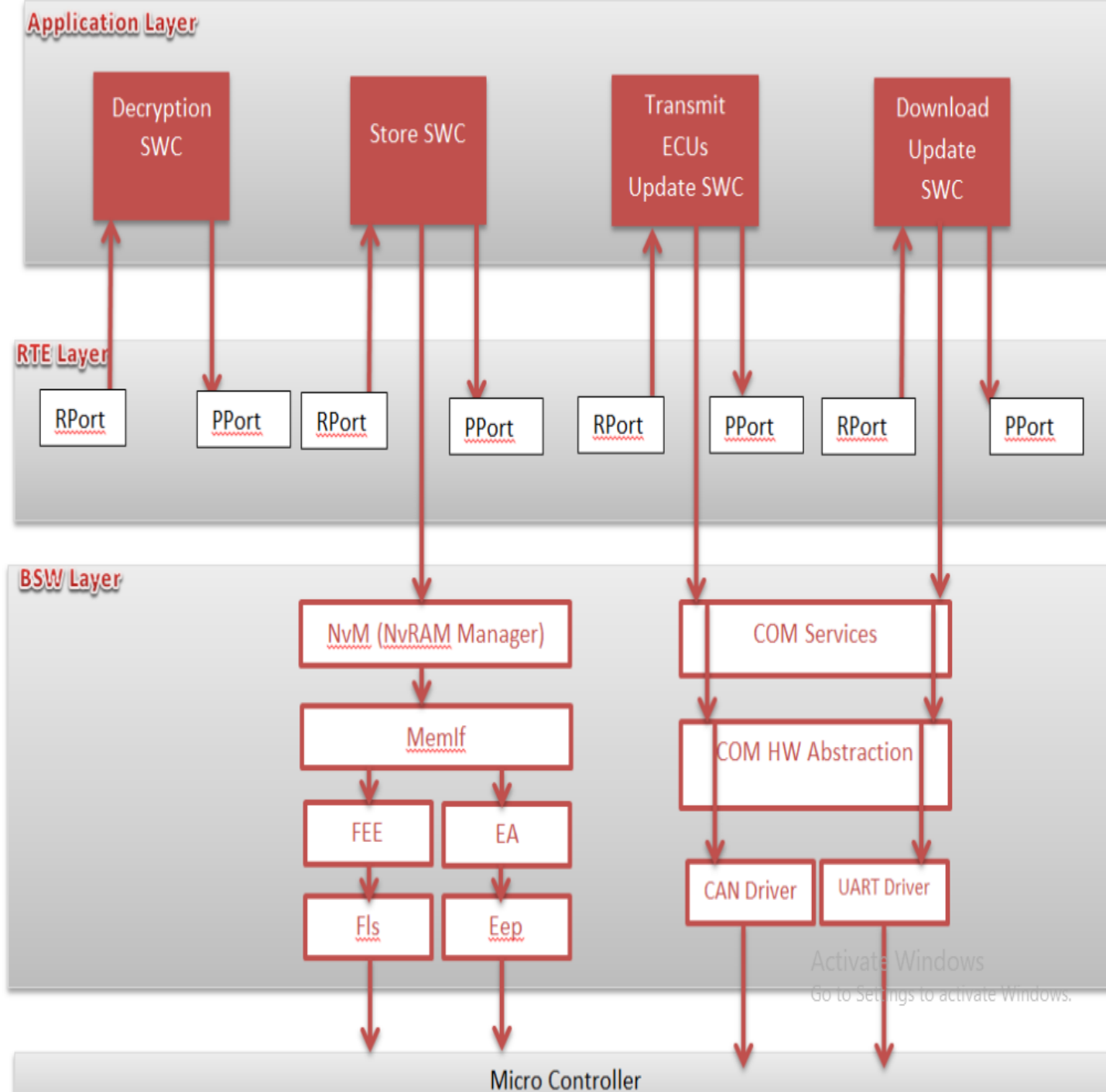
Flash memory

EEPROM

CAN Interface SW

WIFI Interface SW

Transmit Update

Data Packaging

Receive large update data

Decryption

GUI

ECU1 ☹

ECU2 ☹
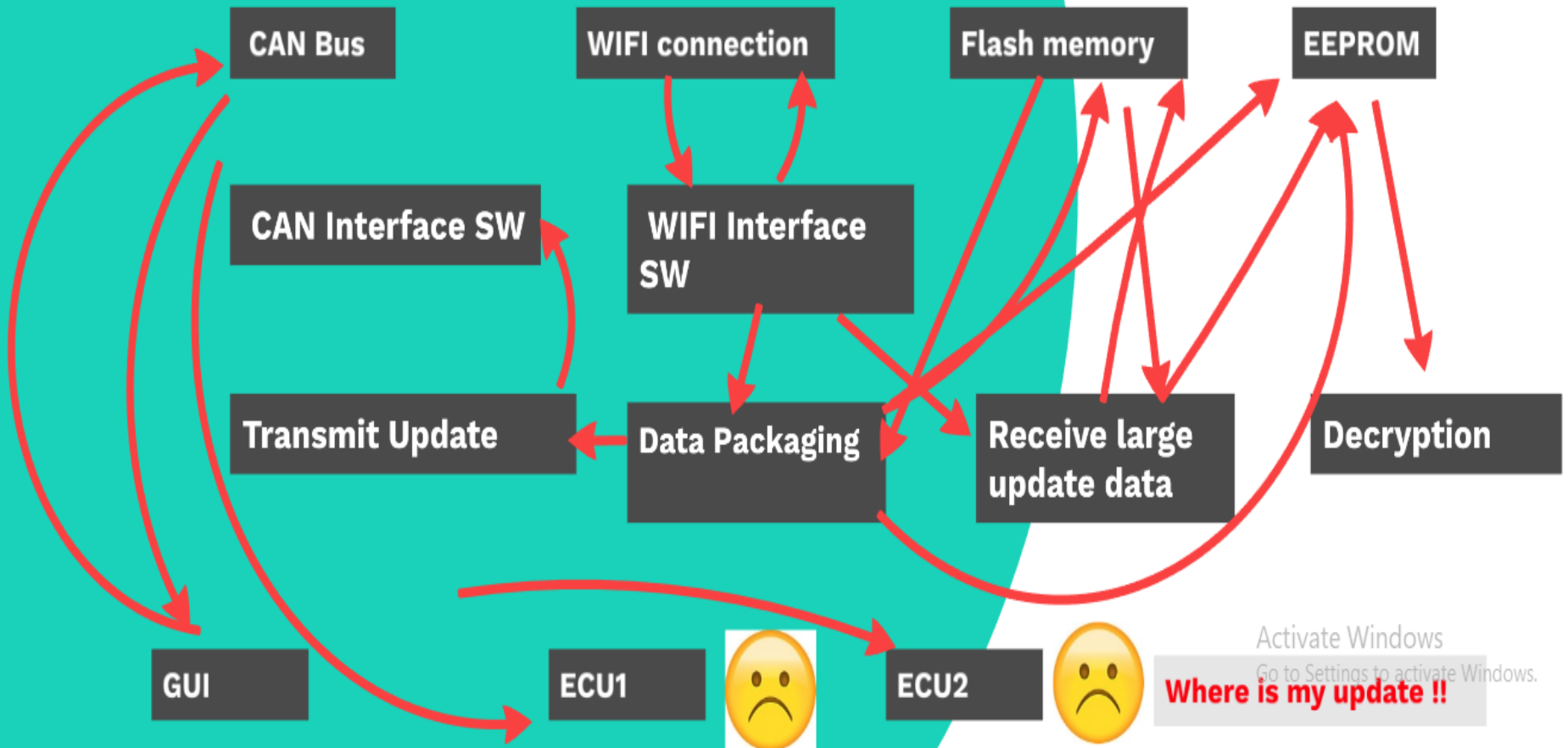
Where is my update !!

AUTOSAR layered architecture

- Managing the growing complexity
- Flexibility
- Scalability
- Quality
- Hardware Abstraction

Application Layer

| Decryption SWC | Store SWC | Transmit ECUs Update SWC | Download Update SWC |

RTE Layer

| RPort | PPort | RPort | PPort | RPort | PPort | RPort | PPort |

BSW Layer

NvM (NvRAM Manager)

MemIf

| FEE | EA |

| Fls | Eep |

COM Services

COM HW Abstraction

| CAN Driver | UART Driver |

Micro Controller

Flash EEPROM Emulation (FEE)

- Many applications require storing small quantities of system related data (e.g., calibration values, device configuration,"App hex update") in a non-volatile memory.

- so that it can be used, modified or reused even after power cycling the system. EEPROMs are primarily used for this purpose. EEPROMs have the ability to erase and write individual bytes of memory many times over and
- the programmed locations retain the data over a long period even when the system is powered down.

What about Flash Memory ??

Flash Memory overview

Suggested solutions

Chosen Algorithm

Flash memory Block segmentation

- Each Sector size is 4KB

- You can only erase 4KB

- 100,000 erase/program cycles

What if you wrote a single word in a sector and need to modify it ??

Block Segmentation

xxFF00h	Sector 15 (4KB)	xxFFFFh
xxF000h		xxF0FFh
xxEF00h	Sector 14 (4KB)	xxEFFFh
xxE000h		xxE0FFh
xxDF00h	Sector 13 (4KB)	xxDFFFh
xxD000h		xxD0FFh
xx2F00h	Sector 2 (4KB)	xx2FFFh
xx2000h		xx20FFh
xx1F00h	Sector 1 (4KB)	xx1FFFh
xx1000h		xx10FFh
xx0F00h	Sector 0 (4KB)	xx0FFFh
xx0000h		xx00FFh

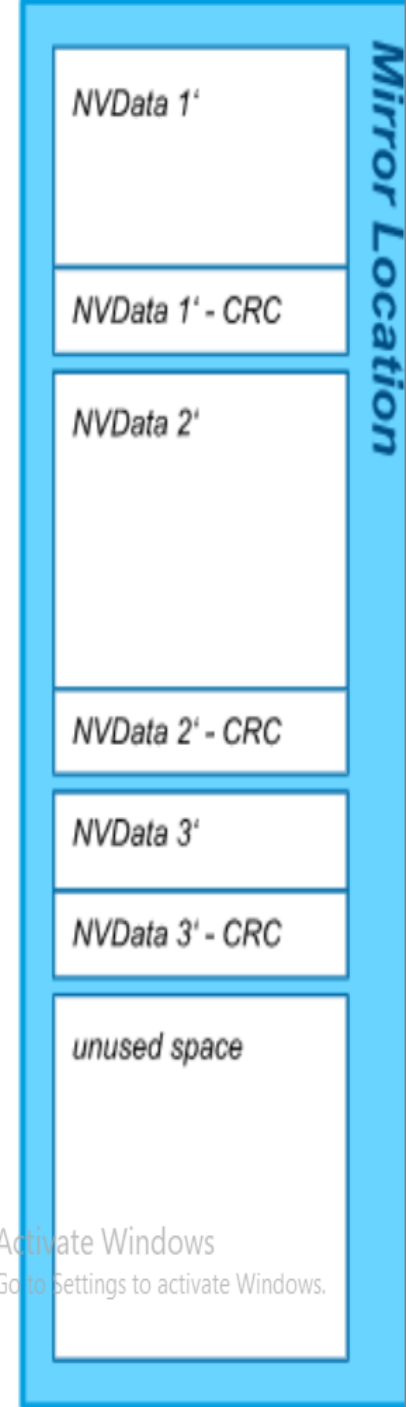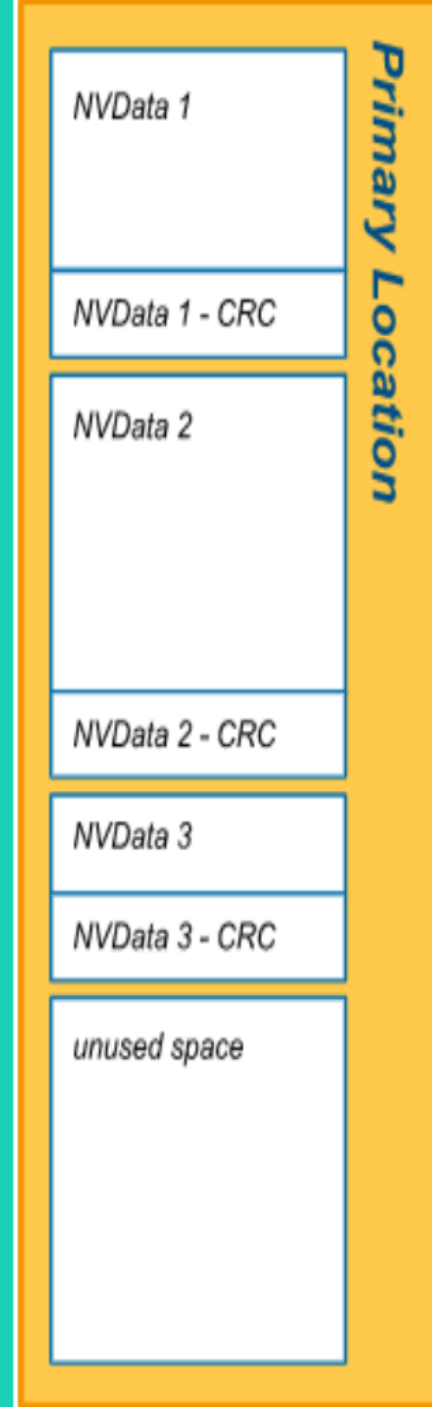# Modify your data without killing flash so fast !

1- Read Modify Write

2- Assign each block of data a fixed place in a sector

What if I'm only modifying One data block at the moment ??

3- Write as long as Sector with free space

Primary Location
NVData 1
NVData 1 - CRC
NVData 2
NVData 2 - CRC
NVData 3
NVData 3 - CRC
unused space

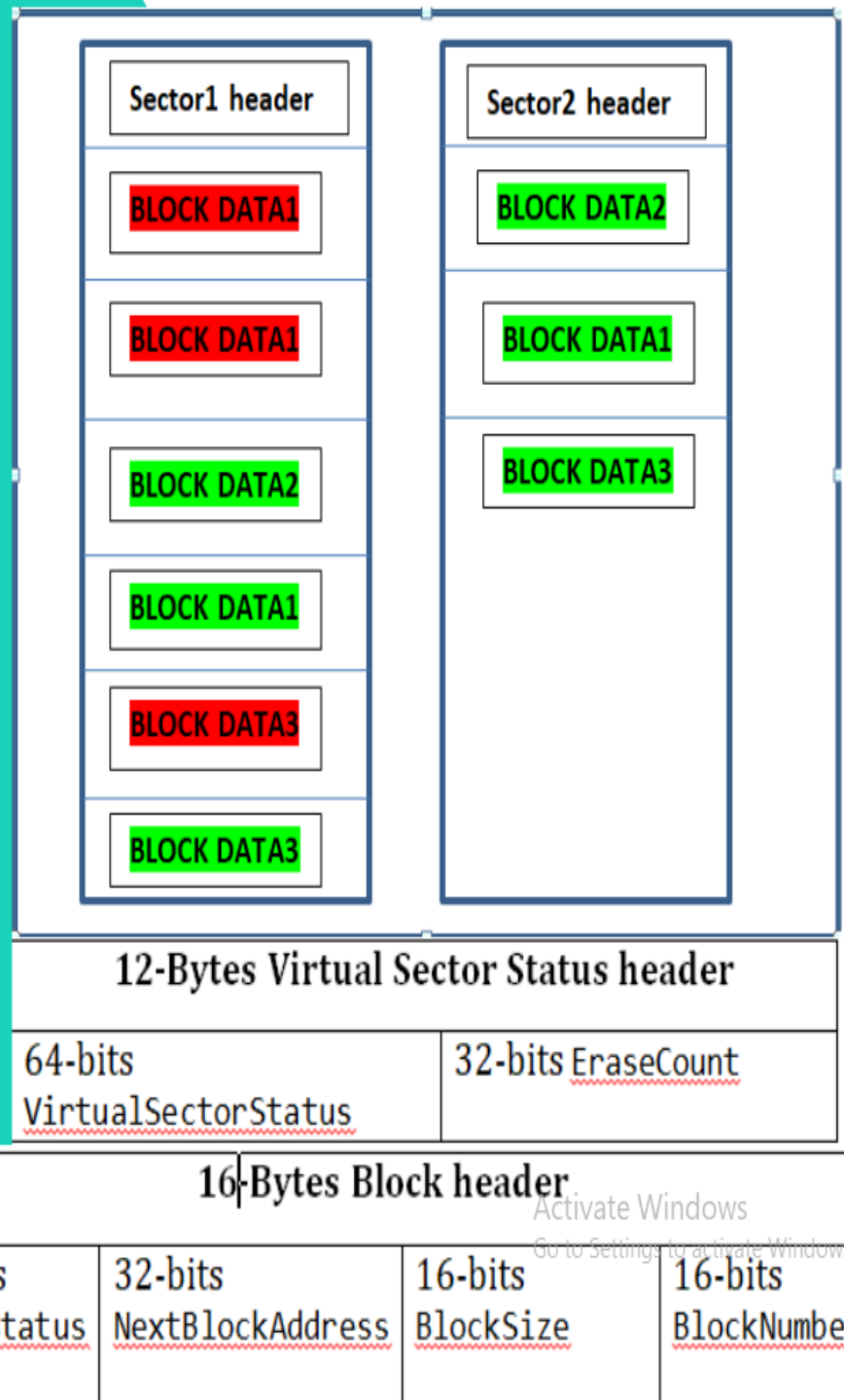| Mirror Location |
| --- |
| NVData 1' |
| NVData 1' - CRC |
| NVData 2' |
| NVData 2' - CRC |
| NVData 3' |
| NVData 3' - CRC |
| unused space |

Write as long as Sector with free space

Wear leveling algorithm to increase emulated EEPROM cycling capability.

The Virtual Sector is the basic organizational unit used to partition the EEPROM Emulation Flash Bank

The Data Block is used to define where the data within a Virtual Sector is mapped

VIRTUAL_SECTOR_SIZE = (NUM_OF_PHYSICAL_SECTORS* PHYSICAL_SECTOR_SIZE)

Erase cycles is increased by number of virtual sectors

Sector1 header	Sector2 header
BLOCK DATA1	BLOCK DATA2
BLOCK DATA1	BLOCK DATA1
BLOCK DATA2	BLOCK DATA3
BLOCK DATA1	
BLOCK DATA3	
BLOCK DATA3	

12-Bytes Virtual Sector Status header

64-bits VirtualSectorStatus	32-bits EraseCount

16-Bytes Block header

64-bits BlockStatus	32-bits NextBlockAddress	16-bits BlockSize	16-bits BlockNumber

AUTOSAR Memory Stack

NvM (NvRAM Manager)	Supports: Prioritization of requests,Requests Queuing, CRC check, Restore default from RAM block, write once option, detection of incomplete write operation, cancelling of running job, different datasets of the same block of data.
MemIf Abstraction Layer	Provides upper layer (NvM) with a virtual segmentation on a uniform linear address space. Regardless of the number of Flash or EEPROM devices.
FEE / **EA**	Abstracts from the device addressing scheme and segmentation. Increases life time of the memory device.
Fls / **Eep**	Accesses the memory device directly. And supports higher layer with an abstract view of the hardware. Initializes memory device and reads/writes to it.

Boot-loader

The boot-loader is the first code that is executed after a system reset. Its goal is to bring the system to a state in which it can perform its main function.

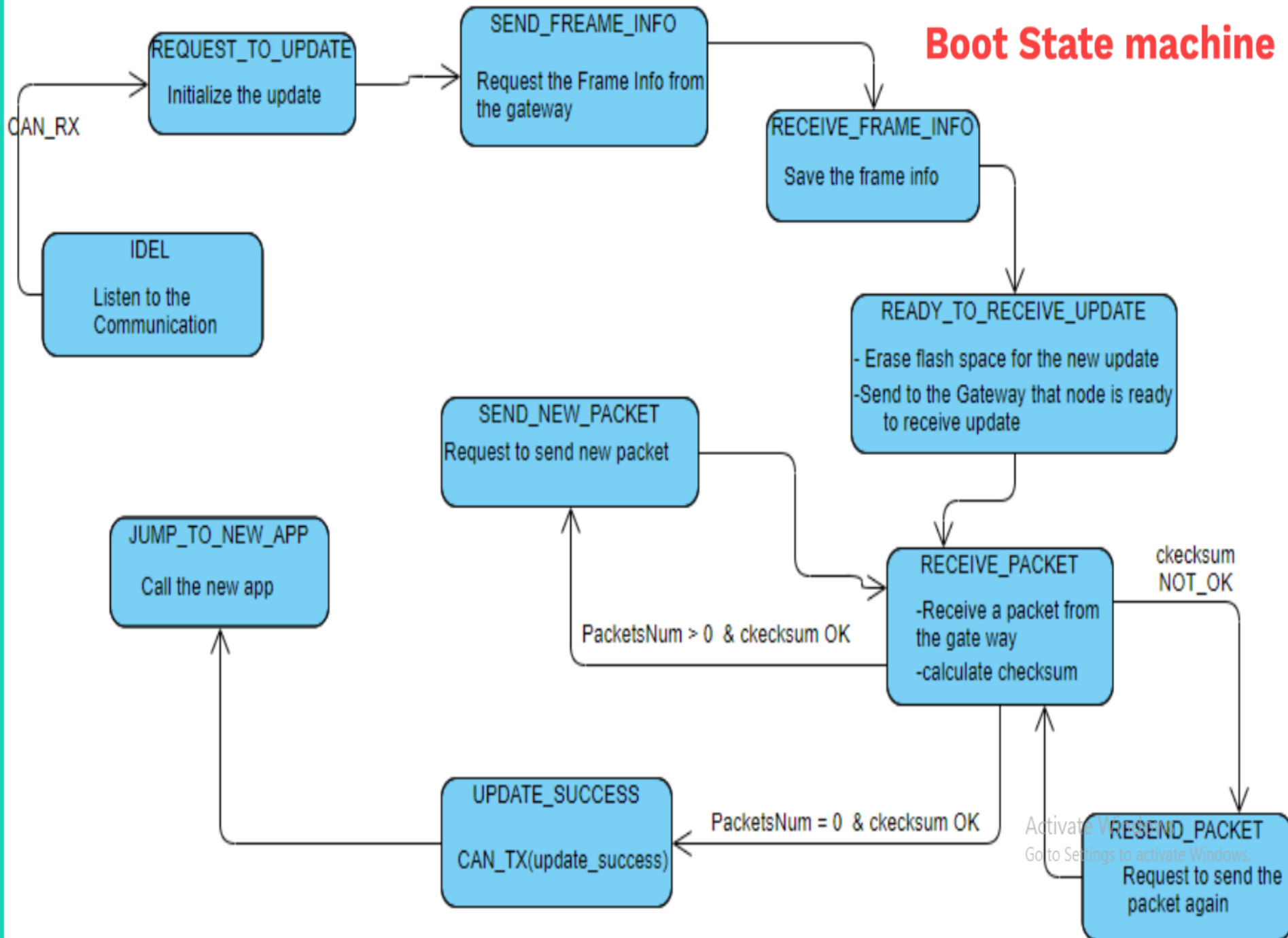Boot-loader design

Update State Machine Diagram

Generic MCU Boot-loader

Branch

Enter Bootloader? — N / Y

Application

Run Application

Enter Bootloader Request? — Y / N

Reset

Soft Reset

Bootloader

Initialize Bootloader

Command Received? — Y / N

Execute Command

Exit To Application? — Y / N
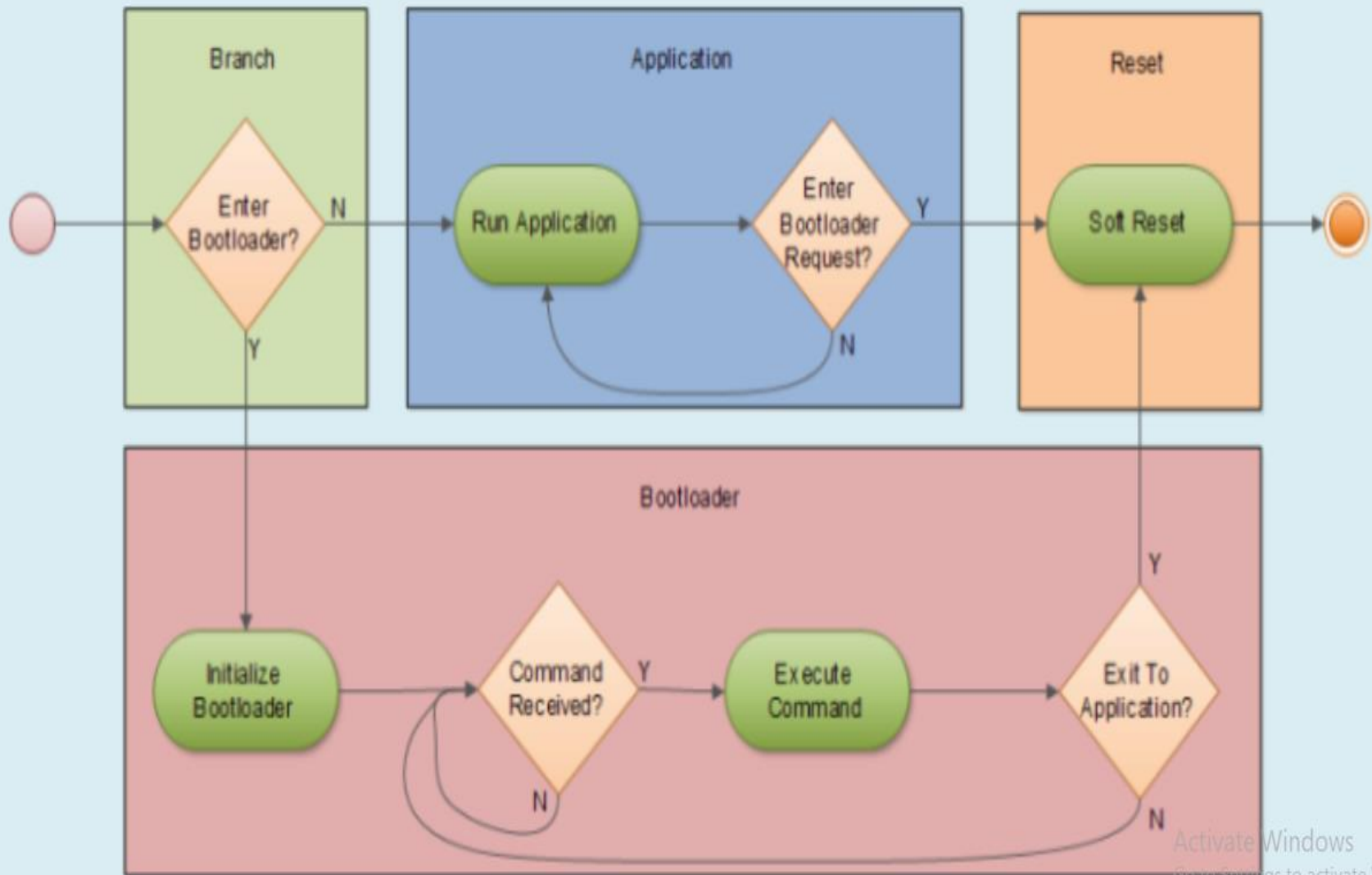
General Boot-loader Operation

Boot State machine

Graphical user interface APP

Features :

- latest software version number.
- connect to the internet.
- notify When updates become available.
- decide the time to start the update.
- cancel the update.

Version number

Internet connection

Update

⚙

Receive new
update

Update
software

software version :

Wi-Fi

SSID

Password

Connect

Cloud connection & Security

- **Edge-to-cloud OTA updates**
- **Gateway-to-cloud OTA updates**
- **Edge-to-gateway-to-cloud OTA updates**
- **We used Edge-to-gateway-to-cloud**
- **OTA updates**

Telematics unit

Firebase

Firebase & Nodemcu Connection

Files & Download URL

File Encryption & Security

Real Time Database & Downloading

OEM OTA
Cloud Servers

Telematics
Unit
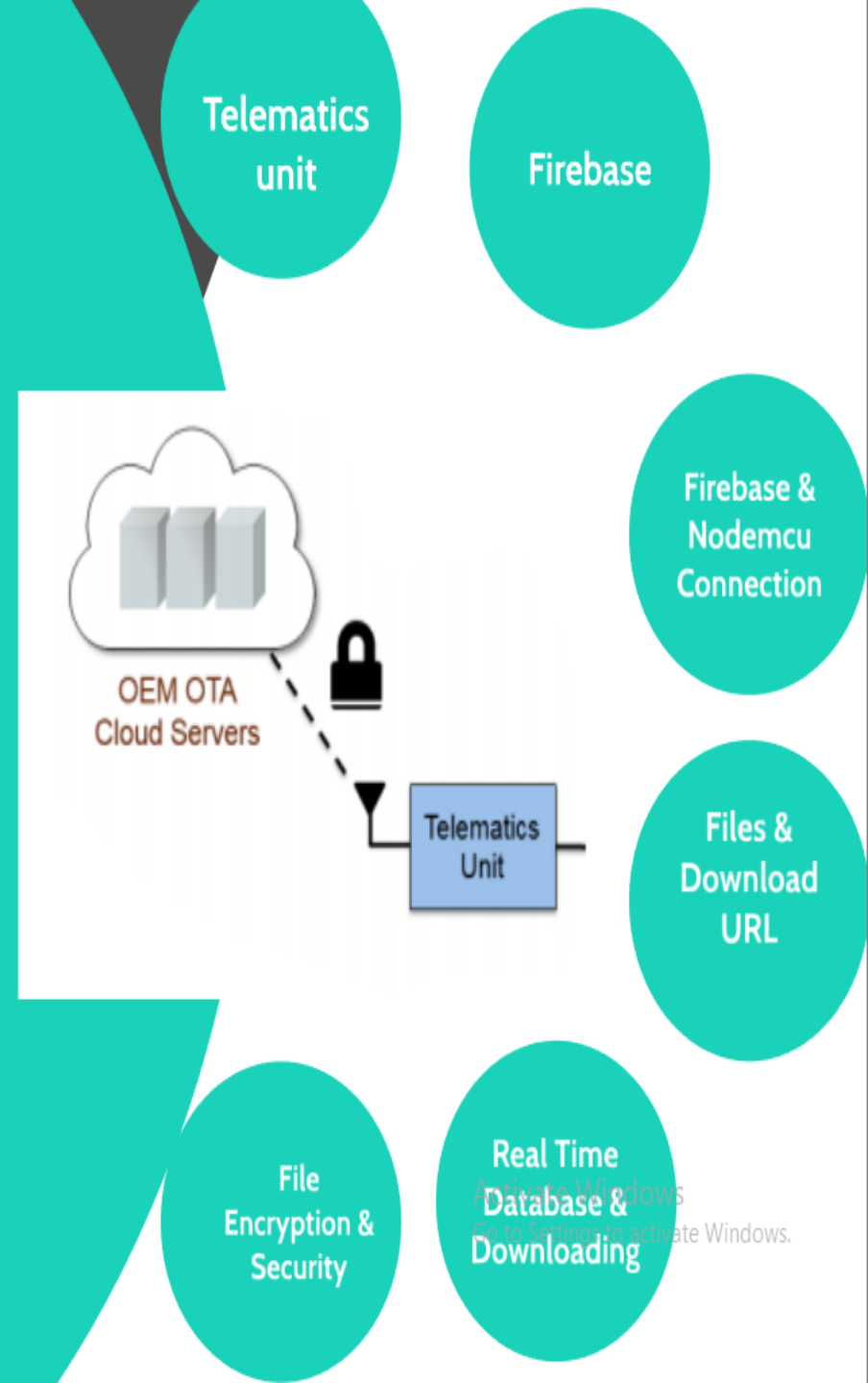
Nodemcu

- **NodeMCU is an open-source firmware and development kit**
- **Main advantage of Nodemcu is that it can directly connect to the internet without using any additional peripheral**
- **Can connect to cloud using HTTP or MQTT protocols**

Firebase

Firebase is google platform, free and easy to use

Became a google platform in 2014 and then google added some more features like firebase cloud storage and real time database

Firebase lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests

Firebase & Nodemcu Connection

The main usage we need firebase for is uploading firmware file that we need to perform ota update

One of Firebase's key features is Firebase Cloud Storage, which lets you store files which can be retrieved over the Internet using a simple HTTP request.

Files & Download URL

Real Time Database & Downloading

we can't include the URL in our code every time we need to update since the Nodemcu will be installed in our vehicle.
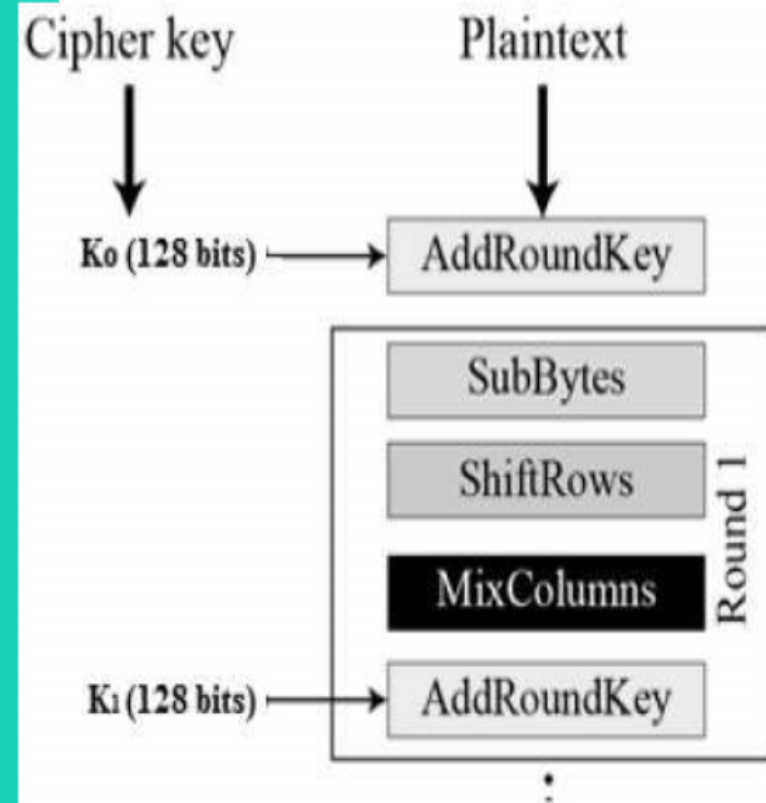
That's why we need to use real time database to send

a notification to the Nodemcu that there is a new update

and also send the URL needed to download the file

- create records in this database
- a flag that tells Nodemcu to start
- receiving the URL to start download
- the default will be "false" and then changed to "true" as shown.

Project Overview

Develop

- Authentication
- Database
- Storage
- Hosting
- Functions
- ML Kit

My First Project ▾   Go to docs 🔔

Database 🖥 Realtime Database ▾

Data   Rules   Backups   Usage

🔗 https://my-first-project-15e18.firebaseio.com/   ◉ ⊖ ⋮

my-first-project-15e18

── FIRMWARE_URL: "https://firebasestorage.googleapis.com/v0/b/esq

── Fingerprint: "93:5C:FF:D0:59:31:EE:0B:12:26:0F:D0:B5:C8:F9:5I

── new_update: "false"

Activate Windows
Go to Settings to activate Windows.

File Encryption & Security

- we have to ensure that the code is safe and have a high security by encrypting the code file before uploading it
- The algorithm we used to encrypt the file is AES 128 which refer to the Advanced Encryption Standard

- **Pure C library**
- **Input: 128-bit blocks**
- **Output: 128-bit blocks**
- **Key: 128 bits**

- We restrict to description of a typical round of
- AES encryption.
- Each round comprise of four sub-processes
- Byte Substitution
- Shiftrows
- MixColumns
- Addroundkey

Thank You