Final Year Project Semester 1 report

Author: Dawid Beres

Supervisor: Brendan Jackman

Student id: 20098868

## Introduction

The automotive industry has always played a big role in society, traditionally cars have been mainly mechanical with the engine being one of the most advanced components however in today’s world where substantial technological advancements are being made frequently it was inevitable that the automotive industry would adopt these discoveries and implement them into vehicles by creating modules which control dynamics such as fuel consumption and suspension setting electronically and programmatically, motorized vehicles have to comply with certain safety standards therefore manufacturers frequently update the modules within these vehicles with the newest software to introduce new features and fix potential bugs.

## The problem

The purpose of this project is to solve a problem in the automotive industry more specifically the problem of updating modules within a vehicle, example – a vehicle could have 3 modules one main module which would be the ECU (electronic control unit) and 2 other modules such as the onboard computer and the unit for controlling the thermostat settings, in this scenario the manufacturer wants to introduce an update to the GPS of the onboard computer and a minor change to the thermostat module however the car has to be in a fully functional state whenever driven and there is no way of knowing when the car is going to be driven (ex: a scenario in which the user isn’t able to use his vehicle because an update is downloading and installing is not allowed the vehicle has to be fully functional whenever started and used by the user).

## Functionality

The main functions of the project are the ability to update and flash specific targets (which aim to be simulated car modules),

* Copy the current working software code to an unused storage as a backup.
* Load the new code into another partition that is not actively being used, ensuring this occurs when the vehicle is in a safe state (not in use)
* After the new software is successfully loaded, the system will switch partitions, making the new code the active version.

If any issues arise with the new software (e.g., bugs or compatibility issues), the system can switch back to the backup partition this ensures that the vehicle can always return to a functional state.

## Assumptions

**Vehicle does not have to be Idle/safe state During Update download and installation:**

* It is assumed that the vehicle does not have to be idle while downloading updates the updates could be downloaded over several driving cycles.

**Sufficient Storage Space:**

* Each module has to have sufficient storage space for a boot loader, the active partition that stores the currently executing code, the backup which stores the old version of the software and the partition on which the new version of the software will be stored.

**The download finishes fully:**

* A stable network connection is available for downloading updates.

**Modules Are Independently Updatable:**

* Each module within the vehicle should operate independently so that one module can be updated without compromising or affecting other modules in any way that would reduce performance or cause unexpected behaviour.

**User Permissions for Updates:**

* TODO: The system assumes that the user has granted permission for updates to be downloaded and installed both while the vehicle is in motion during driving cycles and when it’s in a safe state.

**No Critical Failures During Update:**

* + - During the update download and installation process it is assumed that there are no critical failures such as hardware malfunctions in case of failure the system must be able to return to a previous working state.

**Partition switching can only occur in a safe state:**

* The modules can only switch partitions to the new software versions when the vehicle is not being operated (in a safe state).

**Compliance with Regulatory Standards:**

* + - The update process complies with industry standards and regulations for vehicle software management.

**Backup Version Integrity:**

* + - It is assumed that the backup version of the software remains uncorrupted and intact. In the event of any issues with the new version, the system can safely revert to the older version without complications.

## Functional requirements

A.

Obraz zawierający linia, diagram, Prostokąt, tekst

Opis wygenerowany automatycznie

B. Detailed Requirements List

TODO

**B.1** **Master Module (ECU) Requirements**

1.1 The master module should be able to receive updates from the server TODO?.

1.2 The master module should continue to uphold all of its duties during the update process.

1.3 Master module will communicate with target modules through CAN network.

**B.2** **System overview**

2.1 The system and modules will be able to download and install software without the need to be in a safe state (over driving cycles).

2.2 When setting a different partition as active the vehicle will have to be in a safe state.

2.3 The system must be able to revert to the last functioning version of the software in case of malfunctions such as bugs or errors.

**B.3 Update Management**

3.1 The modules will be able to switch over to the new version of software only when the vehicle is idle.

3.2 The master (ECU) will verify the integrity of the updates before applying it to the specific modules

**B.4 Partition Management**

4.1 There will be three partitions in each module the first partition will be used as a backup to store the old version of code, second partition will be used to store the new version of the software and the third partition will be the active partition on which the running code is stored.

4.2 When a new update is introduced the active partition (current running code) will be copied into the backup partition.

4.3 If a new update is introduced it will be downloaded and installed on the inactive partition.

4.4 When the new update is fully downloaded and installed on the inactive partition the Inactive partition will be copied into the active partition.

4.5 The Inactive partition can only be copied into the active partition when the vehicle is in a safe state.

4.6 The backup partition should be checked to make sure that it is unaffected by the update.

C.

## 5. Technical Investigations

Technical investigation:

**MQTT:**

MQTT is a lightweight messaging protocol designed for low bandwidth, high-latency, or unreliable networks, it uses a publish and subscribe where devices can publish messages to topics and clients can subscribe to those topics for the purpose of receiving updates and messages.

**CAN:**

CAN is a robust vehicle bus standard designed to allow communication between microcontrollers and devices in applications without a host computer its widely used in automotive and industrial systems, it supports error and fault detection which ensures message integrity.

**Arduino Nucleo:**

The Arduino Nucleo is a microcontroller board which is widely used for simple electronic projects due to its ease of use, open source and flexibility, it is based on the ATmega328P and has the necessary digital and analog components for both MQTT and CAN bus communication.

**ESP8266:**

ESP8266 is a WIFI microchip with TCP/IP stack capabilities this allows microcontrollers such as arduinos to access wifi networks and connect to the internet.

**XMC1400:**

XMC1400 are microcontrollers from Infineon they are based on ARM Cortex-M0 technology and are designed for use in industrial and motor control applications they offer high reliability and performance in embedded systems.

**UDS**:

UDS (unified diagnostic services) is a protocol used for communication it enables essential functions such as error reporting, fault diagnostics and control commands.

This is going to allow standardized communication to the different modules for diagnostics and firmware updates.

## 6. Interface specifications

**TODO: user interface Screenlayout/control actions/navigation ???**

**Database/PC:**

Updates files: Stores software updates for the modules.

Diagnostics: Stores diagnostic data such as errors.

Interface Mechanism:

Communication between the Arduino and PC is going to be achieved with the use of MQTT protocol.

**Network Protocols:**

**MQTT:**

MQTT is going to be used for the communication between the PC and Arduino. Data such as updates files is going to be send using this protocol.

**UDS:**

UDS compliant with ISO 14229 is going to be used for the communication between the Arduino and individual modules. Its going to be used to send data such as control commands for reading and writing individual modules.

**CAN:**

The CAN bus is going to be used for the communication between the Arduino and other modules. Data such as update files and diagnostic data are going to be sent over the CAN bus.

## 7. Non-functional Requirements

TODO: performance = real -time data processing / low latency communication

**7.1 Data Encryption:**

Data such as update files, control commands and diagnostics must be encrypted to prevent access by unauthorized third parties and tampering.

**7.2 Rollback and Integrity checks:**

The system must verify that the integrity of each update before applying it. In the case of any corruption, error or interruption of the update the system must revert to the previous functional state.

**7.3 Partition Switching:**

The partition switching process must be completed successfully to ensure that updates do not enter the vehicle modules in an unstable state. The system should verify partition integrity.

**7.4 Error handling:**

Any detected failure during the update process such as loss of network connectivity or missing files should trigger automatic retries. If the error persists the last fully functional software version should be used until the error is resolved.

**7.5 In-Use updates:** Updates should be downloaded while the vehicle is in use so that the user doesn’t have to leave the vehicle in an idle state and wait for the update to download.

**7.6 Storage requirements:**

The minimum amount of storage each module should have should be able to store 3 partitions and a bootloader.

**7.7 Maintenance:**

The system should perform checks for partition, update and general system integrity.

**7.8 Compliance with industry standards:**

The system should comply with industry standards such as ISO 14229 for UDS communication. This ensures that the design will meet safety and compatibility standards within the automotive industry.

## 8. Quality Assurance Provisions

Software test procedures:

Software validation procedures: