

Information Technology Institute

Egypt Makes Electronics (4 Months)

Embedded Systems Track



Next-Generation Vehicle Technology

Graduation Project

Submitted by:

Hossam Ayoub

Hesham Yasser

Dina Elsayed

Diaa Assem

Ziad Khaled

Seif El-Deen Ashraf

Submitted to:

Eng. Joe Nofal

Eng. Nour Hassan

Nov. 2023

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION	5
CHAPTER 2: MICROCONTROLLERS	7
2.1. STM32F401CC Microcontroller:	7
2.1.1. Overview:	7
2.1.2. Key Features:.....	8
2.1.3. STM32F401CC Pinout Diagram:.....	9
2.2. Raspberry Pi:.....	10
2.2.1. Introduction:	10
2.2.2. Raspberry Pi 3 Components:.....	11
2.2.3. Key Features:.....	12
CHAPTER 3: FIRMWARE OVER-THE-AIR (FOTA)	13
3.1. What is FOTA?	13
3.2. Flashing	14
3.2.1. Flash Memory Interface	14
3.2.2. Hex Parser	15
3.2.3. Bootloader Requirement	16
3.2.4. Installing.....	17
3.2.5. Bootloader Sequence	18
3.3. Raspberry Pi.....	20
3.3.1. Firebase	20
3.3.2. Encryption & Decryption	21
3.3.3. Decide Bank for Programming & Update Level.....	22
3.4. The FOTA Diagram	24
CHAPTER 4: REAL TIME OPERATING SYSTEM (RTOS)	25
4.1. What is RTOS?	25
4.1.1 What are Real Time Systems?	25
4.1.2 What is a Real time OS?	26
4.2. RTOS in Software Layered Architecture.....	27
4.3. Why do we need RTOS in our project?	28

4.4. Designing A Real Time Systems	28
4.4.1. The Most Important Components of Design.....	29
CHAPTER 5: HARDWARE.....	31
5.1. Components:	31
5.1.1. Lane Keep Assistance:	31
5.1.2. Adaptive Cruise Control:	31
5.1.3. Adaptive Lighting System:.....	31
5.1.4. Airbag System:	32
5.1.5. Communication:	32
5.1.6. DC Motors:.....	32
5.2. Schematic & Wiring:	33
5.3. PCB Layout:.....	33
CHAPTER 6: ARTIFICIAL INTELLIGENCE (AI) MODEL.....	35
6.1. What is AI?	35
6.2. Why do we need AI in our project?	36
6.3. Calibration & Training & Results.....	37
CHAPTER 7: SYSTEM APPLICATION.....	38
7.1. Remote Controlled (via Bluetooth).....	38
7.1.1. Working Theory	38
7.2. Adaptive Lighting Control System	39
7.2.1. Working Theory	39
7.3. Adaptive Cruise Control System	40
7.3.1. Working Theory	40
7.4. Automated Emergency System.....	41
7.4.1. Working Theory	41
7.5. Lane Keep Assistance System	42
7.5.1. Working Theory	42
CHAPTER 8: TEST CASES.....	43
CHAPTER 9: CONCLUSION.....	44

TABLE OF FIGURES

Figure 1 STM32F401CC Black Pill	7
Figure 2 STM32F401CC Pinouts and Pin Description	9
Figure 3 Raspberry Pi	10
Figure 4 STM32F401CC Pinouts and Pin description.....	11
Figure 5 Raspberry pi 3 B+ Pinouts	12
Figure 6 Firmware Upload Website.....	20
Figure 7 Real Time Applications.....	25
Figure 8 RTOS	26
Figure 9 AUTOSAR	27
Figure 10 Airbag System.....	28
Figure 11 CPU Load.....	30
Figure 12 RPi Camera Module.....	31
Figure 13 Ultrasonic Sensor.....	31
Figure 14 LDR Sensor	31
Figure 15 FSR Sensor	32
Figure 16 Bluetooth Module	32
Figure 17 DC Gear Motor.....	32
Figure 18 L298N Driver Module.....	32
Figure 19 Schematic & Wiring.....	33
Figure 20 PCB Manufacturing Process	33
Figure 21 Top Silk Layer	34
Figure 22 Bottom Copper Layer	34
Figure 23 Lane Keep Assistance.....	36
Figure 24 Detecting Lane Lines.....	37
Figure 25 Bluetooth Controlled Diagram.....	38
Figure 26 LDR Working Diagram	39
Figure 27 Emergency Airbag System Diagram	41
Figure 28 Lane Keep Assistance Diagram	42

CHAPTER 1: INTRODUCTION

In the realm of automotive technology, our groundbreaking project sets out to redefine the capabilities and user experience of cars. By integrating advanced features and cutting-edge technologies, we have developed a platform that transcends the traditional boundaries of car functionality. Our goal is to provide enthusiasts and hobbyists with an unparalleled driving experience that combines performance, safety, versatility, and convenience.

At the core of our project lies the incorporation of **Firmware Over-The-Air (FOTA)** updates. This revolutionary technology enables wireless updates to the car's firmware, eliminating the need for physical intervention and enhancing convenience. With FOTA, users can effortlessly update their cars with the latest firmware releases, ensuring access to the most up-to-date features and optimizations. Manufacturers can remotely deploy updates, enabling seamless introduction of new functionalities, bug fixes, and performance enhancements.

Our car is equipped with a camera and a Raspberry Pi, enabling the implementation of **Lane Keep Assistance**. Through the power of **Artificial Intelligence (AI)**, our system analyzes the real-time video feed from the camera, identifying lane markings. By actively monitoring the car's position within the lane, our system provides corrective steering inputs when necessary, helping to keep the car safely centered on the road.

In addition to FOTA, our cars boast an array of innovative features that contribute to an immersive driving experience. **Adaptive Cruise Control (ACC)** utilizes ultrasonic sensors to accurately measure distances and dynamically adjust the car's speed. By controlling the motors using timer peripherals and pulse width modulation (PWM) signals, our system maintains a safe and consistent distance from obstacles, ensuring a smooth and secure ride.

To enhance safety, our cars are equipped with an **Automated Emergency Actions System**. A pressure sensor detects critical situations, triggering immediate alerts such as activating buzzers or LEDs. These actions provide real-time awareness of potential dangers, mitigating risks and prioritizing user safety.

Another standout feature is the **Adaptive Light Control System**, which optimizes visibility during nighttime driving. An integrated Light Dependent Resistor (LDR) sensor detects ambient light levels, allowing the system to automatically adjust the intensity and direction of the car's headlights. This ensures optimal visibility while minimizing glare for other drivers, enhancing overall safety on the road.

Furthermore, our cars offer **Remote Control Capabilities via Bluetooth Connectivity**. Users can effortlessly control the car using their smartphones or compatible devices, adding to the convenience and intuitiveness of the driving experience.

To facilitate maintenance and troubleshooting, our cars feature a **Comprehensive Diagnostic System**. Real-time monitoring of key parameters, sensors, and components provides valuable insights into the car's performance. Users can access diagnostic data and receive alerts, enabling timely repairs and ensuring the longevity of the car.

The integration of a **Real-Time Operating System (RTOS)** guarantees efficient task scheduling, priority management, and resource allocation. This enables seamless operation and coordination of multiple features, ensuring a smooth and reliable driving experience.

2.1.2. Key Features:

There are some general features of STM32F401CC Microcontrollers:

- The Flash memory is up to 256 Kbytes.
- The SRAM is up to 64 Kbytes.
- 512 OTP (one-time programmable) bytes for user data.
- An extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB buses and a 32-bit multi-AHB bus matrix.
- Up to 3 UARTs.
- UP to 11 communication interfaces.
- Up to 11 timers: up to six 16-bit, two 32-bit timers up to 84 MHz, each with up to 4 IC/OC/PWM or pulse counter and quadrature.
- Serial wire JTAG debug port (SWJ-DP).
- Clock, reset and supply management.
- General-purpose DMA: 16-stream DMA controllers with FIFOs and burst support.
- CRC calculation unit.
- 96-bit unique ID.
- RTC: subsecond accuracy, hardware calendar.
- Up to 81 I/O ports with interrupt capability.

2.1.3. STM32F401CC Pinout Diagram:

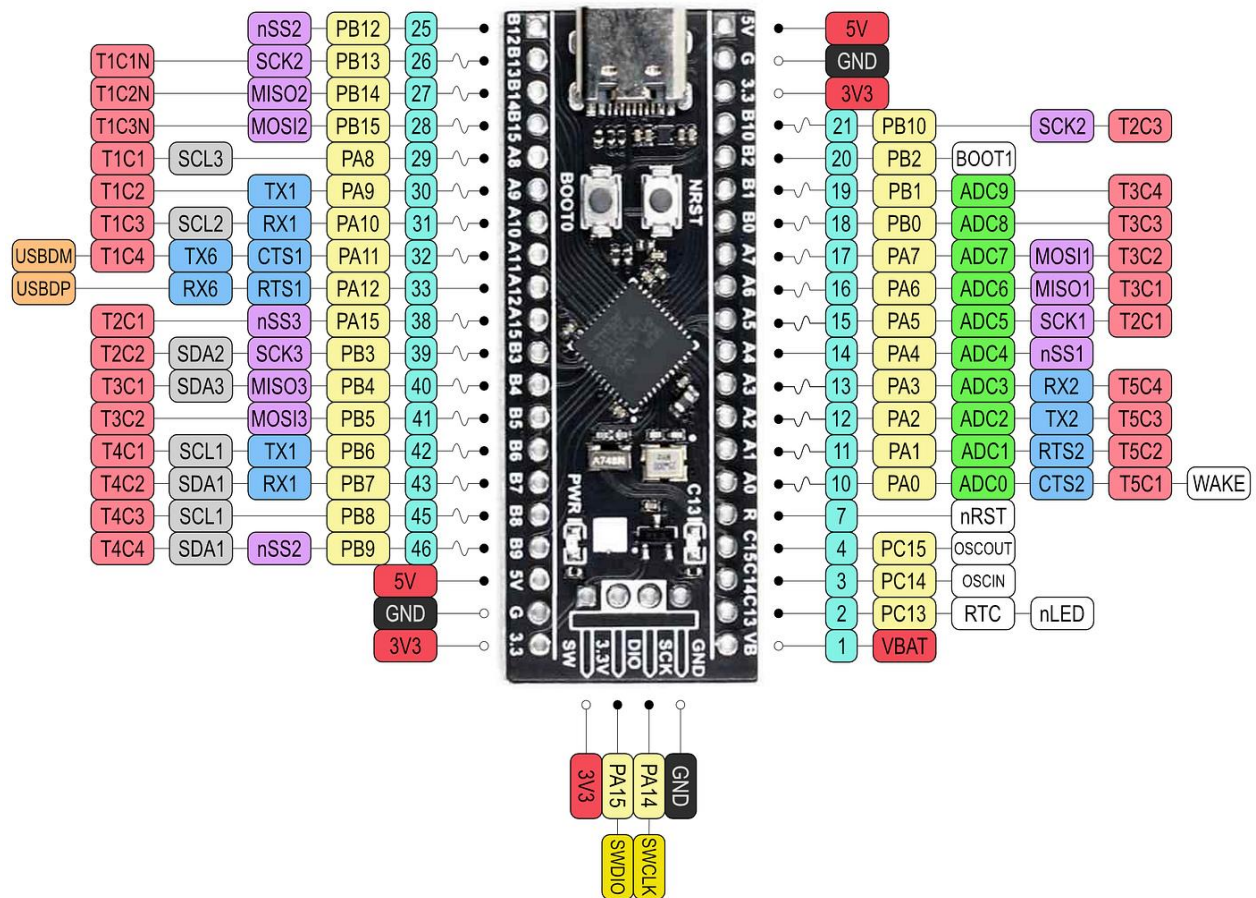


Figure 2 STM32F401CC Pinouts and Pin Description

2.2. Raspberry Pi:

2.2.1. Introduction:

The Raspberry Pi is a series of small, affordable, single-board computers developed by the Raspberry Pi Foundation, a UK-based charity organization. These credit card credit-sized fully functioning computer (System on chip). Although the Raspberry Pi may run a number of operating systems, the official and most popular distribution is Raspbian, which is currently called Raspberry Pi OS. But there are also other operating systems out there, such as Ubuntu, Arch Linux, and versions tailored for certain uses. Like a computer, it has memory, processor, USB ports, audio output, and graphic driver for HDMI output. To meet diverse purposes, the Raspberry Pi Foundation has developed multiple versions with differing specifications over the years. The processor speed, RAM, networking options, and other features of these models could vary.

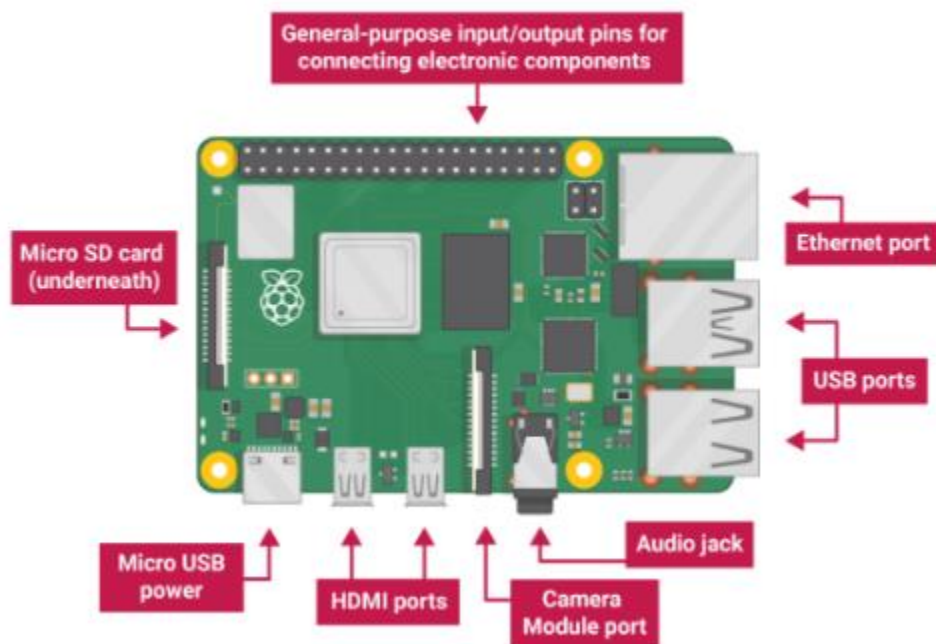


Figure 3 Raspberry Pi

2.2.2. Raspberry Pi 3 Components:

- USB ports: these are used to connect a mouse and keyboard. You can also connect other components, such as a USB drive.
- SD card slot: you can slot the SD card in here. This is where the operating system software and your files are stored.
- Ethernet port: this is used to connect Raspberry Pi to a network with a cable. Raspberry Pi can also connect to a network via wireless LAN.
- Audio jack: you can connect headphones or speakers here.
- HDMI port: this is where you connect the monitor (or projector) that you are using to display the output from the Raspberry Pi. If your monitor has speakers, you can also use them to hear sound.
- Micro USB power connector: this is where you connect a power supply. You should always do this last, after you have connected all your other components.
- GPIO ports: these allow you to connect electronic components such as LEDs and buttons to Raspberry Pi.

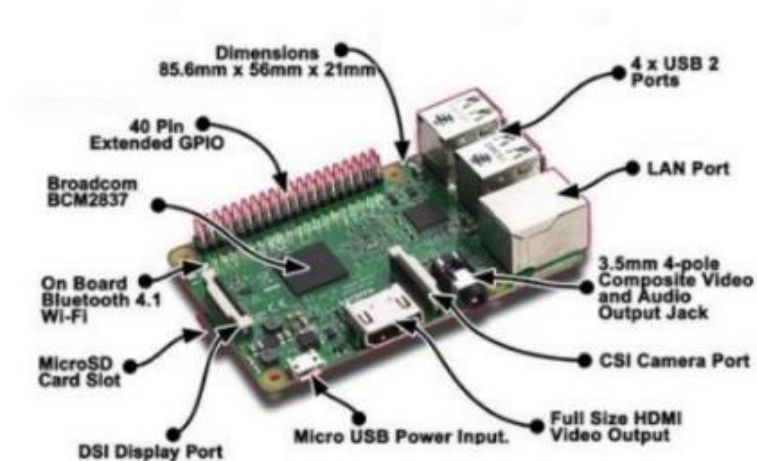



Figure 4 STM32F401CC Pinouts and Pin description

2.2.3. Key Features:

In this project we have used Raspberry pi 3 B+ model with

- 1.4 GHz quad-core ARM Cortex-A53 processor.
- Providing a slight clock speed increase compared to the original Pi 3.
- 1 GB RAM.
- 4 USB.
- Jack HDMI.
- USD.
- Broadcom BCM2837B0.
- 10/100/1000 Mbps ETH(USB) Dual-band Wi-Fi BT

Raspberry pi 3 B+ has 40-pin GPIO header for hardware interfacing.



The image shows a Raspberry Pi 3 B+ board with a red rectangular box highlighting the 40-pin GPIO header on the right side. The header pins are color-coded: red for 3.3v DC Power, blue for DC Power 5v, black for Ground, green for various GPIO pins, and purple for SPI pins.

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I ² C)	DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I ² C ID EEPROM)	(I ² C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Figure 5 Raspberry pi 3 B+ Pinouts

CHAPTER 3: FIRMWARE OVER-THE-AIR (FOTA)

3.1. What is FOTA?

Firmware Over-The-Air (FOTA) in embedded systems refers to the capability of updating the firmware or software of devices wirelessly, without requiring a physical connection. This technology is particularly relevant in embedded systems, where devices may be deployed in remote locations or in situations where physical access is challenging.

FOTA enables manufacturers and developers to remotely update the firmware of devices after they have been deployed in the field. This is particularly useful for several reasons:

- **Bug Fixes and Improvements:** FOTA allows manufacturers to address bugs, vulnerabilities, and other issues discovered after the device has been released. It enables them to send software updates to improve the device's performance or add new features.
- **Security Updates:** In the rapidly evolving landscape of cybersecurity, vulnerabilities are often discovered post-deployment. FOTA allows manufacturers to push security patches and updates to ensure that devices remain secure.
- **Efficiency:** FOTA eliminates the need for physical recalls or manual updates, reducing costs and the inconvenience for users. Devices can be updated seamlessly over the air, without requiring users to take any action.
- **Flexibility:** FOTA provides flexibility for device manufacturers to adapt to changing requirements or standards without requiring users to replace their devices. This is especially important in industries where devices have a long lifecycle.
- **Remote Diagnostics:** it can also enable remote diagnostics and monitoring, allowing manufacturers to collect data on device performance and identify potential issues before they become widespread.

However, it's important to note that while FOTA offers many advantages, there are also concerns related to security and privacy. The process of updating firmware over the air must be secure to prevent unauthorized access or malicious updates that could compromise the device or user data.

Overall, FOTA is a crucial technology for maintaining and improving the functionality of embedded devices, especially in the context of the Internet of Things, where devices are often distributed widely and may be challenging to access physically.

3.2. Flashing

Flashing involves the programming of firmware or software onto the non-volatile memory of a microcontroller or microprocessor-based device. In this process, an application, or a portion of it, provided in hex format, is transferred to the Electronic Control Unit (ECU)'s memory. This transfer occurs through bus protocols such as UART, CAN, LIN, Flex Ray, etc. The Flash Bootloader is responsible for downloading the application as a hex file to the ECU.

To facilitate the transfer of your application to the target platform, you only require the Flash Bootloader, a PC or laptop, and the Flash Tool. Two options are available for flashing an application:

1. **Off-Circuit Programming:** In this method, the chip is removed from the system to be programmed, and the application is flashed.
2. **In-Circuit Programming:** The chip remains in the system, but a debugger is still needed for programming and flashing the application.
3. **In-Application Programming:** This method utilizes an application (Bootloader) to flash a new application without the necessity of a debugger.

3.2.1. Flash Memory Interface

The Flash programming operation of the Flash Memory Interface (FMI) is employed to write data to the Flash memory of a microcontroller. A crucial prerequisite for this operation is that a Flash memory sector must undergo erasure before any new data can be written to it. Attempting to write new data to a sector without prior erasure leads to undefined behavior, as the sector's contents become a mixture of old and new data.

This is due to the organization of Flash memory into sectors, and each sector must undergo erasure before it can be written to again. The erase operation resets all bits in the sector to 1, creating a blank canvas for new data to be written without interference from the old data. Subsequently, the write operation selectively sets specific bits in the sector to 0, effectively storing the new data.

Hence, before introducing new data to a sector, it is imperative to erase the sector to ensure that no remnants of old data persist. The FMI provides a sector erase operation specifically designed for erasing a particular Flash memory sector before enabling the writing of new data.

Flash memory is segmented into sectors, with each sector typically containing a fixed number of bytes or words. The programming operation involves writing one or more words of data to a designated address in the Flash memory. This address could represent a single byte or an entire sector, depending on the size of the data being written. This systematic approach of erasing and then writing data ensures the reliability and integrity of the Flash memory in the microcontroller.

3.2.1.1. Sector Erase

The Flash memory erase operation can be performed at sector level or on the whole Flash memory (Mass Erase). We used the erase sector to be able to make the 2 banks option

To erase a sector, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register.
2. Set the SER bit and select the sector out of the 5 sectors (for STM32F401xB/C) and out of 7 (for STM32F401xD/E) in the main memory block you wish to erase (SNB) in the FLASH_CR register.
3. Set the STRT bit in the FLASH_CR register
4. Wait for the BSY bit to be cleared

3.2.1.2. Programming

The Flash memory programming sequence is as follows:

1. Check that no main Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register.
2. Set the PG bit in the FLASH_CR register
3. Perform the data write operation to the desired memory address
4. Wait for the BSY bit to be cleared.

3.2.2. Hex Parser

The Hex Parser driver plays a pivotal role in decoding and processing Intel Hex format records, a popular standard for representing binary data in hexadecimal form. This driver encapsulates four distinct functions, with two of them serving as the core components of its functionality. The primary function involves parsing the Hex records, wherein the driver discerns various types of data, identifies the end of a record, extracts extended address information, and retrieves the starting address of the application, among other critical details. This function is integral in decoding the structure and content of the Hex file, providing essential metadata for subsequent operations. The second key function of the Hex Parser driver

is dedicated to parsing the actual data and ensuring its integrity through checksum validation. This process involves scrutinizing the data payload within the Hex records, verifying its accuracy, and safeguarding against potential errors or corruption. By meticulously parsing and validating each record, the Hex Parser driver acts as a crucial intermediary, facilitating the accurate translation of Hex files into executable data for the target microcontroller or embedded system. Overall, the Hex Parser driver serves as a fundamental component in the process of preparing and validating Hex format data, ensuring the reliability and correctness of the information to be utilized in programming microcontroller flash memory.

3.2.3. Bootloader Requirement

The following outlines the necessary hardware and software prerequisites for utilizing the UART bootloader to transfer a hex file byte by byte from the Raspberry Pi to an STM microcontroller:

1. Supported Microcontrollers: Ensure that the bootloader is specifically designed to support the microcontroller in use. For example, in our project, the bootloader should be tailored for the Cortex-M4 STM32f401CC microcontroller.
2. UART Interface Requirements: The bootloader rate must be compatible with the microcontroller's UART interface, considering the following requirements:
 - a) Baud Rate: Used 1562500 baud.
 - b) Data Bits: Used 8 data bits.
 - c) Parity: Used no parity.
 - d) Stop Bits: Used 1 stop bit.

Note: This is the used configuration in the project. It can be changed but both STM UART and Raspberry Pi UART must have the same configuration to avoid problems.

3. Prerequisites: Take into account the following prerequisites:
 - a) Bootloader Application: Ensure that the bootloader application is available and programmed onto the microcontroller before attempting to use the UART bootloader.
 - b) Hex File: Have hex file, containing the firmware for programming onto the microcontroller, ready.
 - c) Serial Communication Program: A serial communication program should be present on the computer used for communication with the microcontroller over UART. This program must be capable of sending and receiving data over the UART interface.

Once these requirements are met, you are prepared to initiate the installation of the bootloader.

3.2.4. Installing

The process of installing a bootloader on a target microcontroller may vary depending on the microcontroller and bootloader in use. However, the following provides a general outline of the steps involved:

1. Obtain Bootloader Software: Begin by acquiring the bootloader software compatible with the target microcontroller. This software may be supplied by the manufacturer or developed by a third party.
2. Configure Microcontroller: Before begin in the bootloader sequence, configure the microcontroller's hardware and software components. This involves setting up GPIO, UART, RCC, FMI, and STK peripherals as required by the bootloader.
3. Program Bootloader onto Microcontroller: Utilize a programming tool, such as JTAG, SWD, or another programming interface, to program the bootloader onto the microcontroller's non-volatile memory. Follow the instructions provided by the manufacturer or third party for this programming process.
4. Configure Bootloader: After programming the bootloader, configure the specific hardware and software components of the target system. This may include setting the UART baud rate, parity, stop bits, and other relevant parameters. Additionally, configure the parser to process the hex file record by record and program it into the microcontroller's flash memory.

Installing a bootloader onto a target microcontroller encompasses configuring the microcontroller's hardware and software, programming the bootloader onto its non-volatile memory, configuring the bootloader, and verifying its functionality. The specific details of each step will depend on the microcontroller and bootloader chosen for the application.

3.2.5. Bootloader Sequence

The Bootloader always starts first in the system. It's located at the beginning of the flash memory present on the Electronic Control Unit (ECU). However, there's no assurance that a valid, executable application is also available in the ECU. Because of this uncertainty, the Bootloader is the initial program executed after a reset. The Bootloader then decides whether to launch the application or remain in the Bootloader.

The fundamental operation of the bootloader involves a sequential process:

1. Waiting for Update:

- The bootloader initiates by checking if there is a new firmware update for 10 milliseconds.

2. Handling No Update Scenario:

- If there is no new update, the bootloader check which memory bank contains a valid application.
- It then jumps to the bank with the valid application and starts executing it.

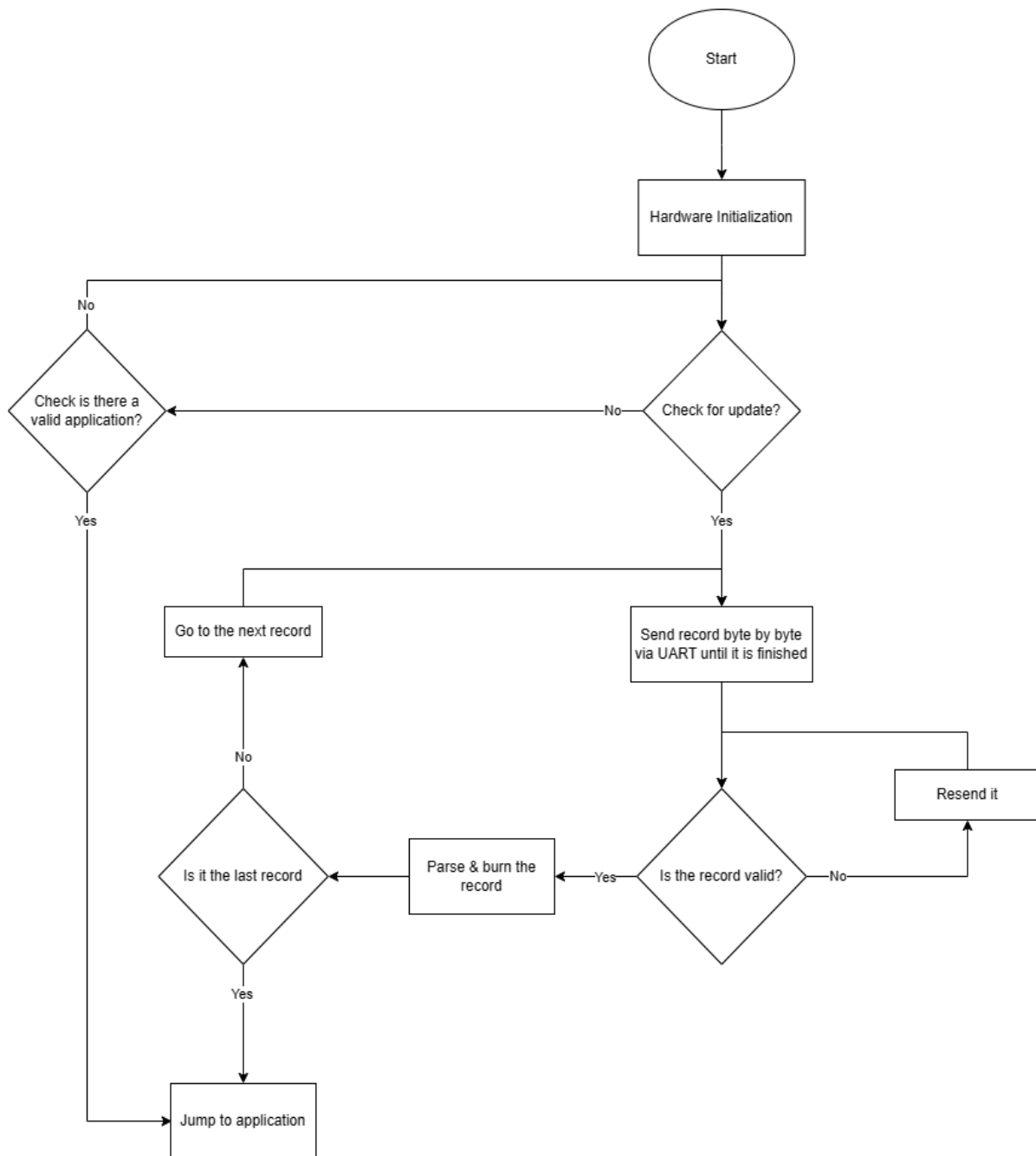
3. Handling Update Scenario:

- In the event of a new update, the bootloader enters a mode where it receives each record byte by byte.
- It continues this process until the entire record is received.
- After receiving a record, it validates the content. If valid, it sends the acknowledgment K; if not valid, it sends N.
- The bootloader then repeats the sequence, awaiting the next byte-by-byte reception until the complete update is received.

4. Handling No Update or No Valid Application:

- If there is neither an update nor a valid application present, the bootloader reverts to the initial state, waiting for any potential updates.

This logic ensures that the bootloader efficiently manages firmware updates, validates receive data, and transitions between different operational scenarios based on the presence of updates and valid applications.



3.3. Raspberry Pi

We talked about the side of STM and how its bootloader work but what about the Raspberry Pi what is its role?

3.3.1. Firebase

The integration of a Raspberry Pi with Firebase storage establishes a robust and scalable framework for facilitating STM firmware updates. This synergy is achieved through a website designed to streamline the firmware upload process, which is directly connected to Firebase. Firebase, a comprehensive mobile and web application development platform, serves as the centralized storage solution for housing the STM firmware updates. The website acts as an intuitive interface, enabling users to effortlessly upload firmware files. Leveraging Firebase's cloud-based storage capabilities, the uploaded firmware data is securely stored, and its accessibility is extended to the Raspberry Pi.



Figure 6 Firmware Upload Website

3.3.1.1. Why Firebase?

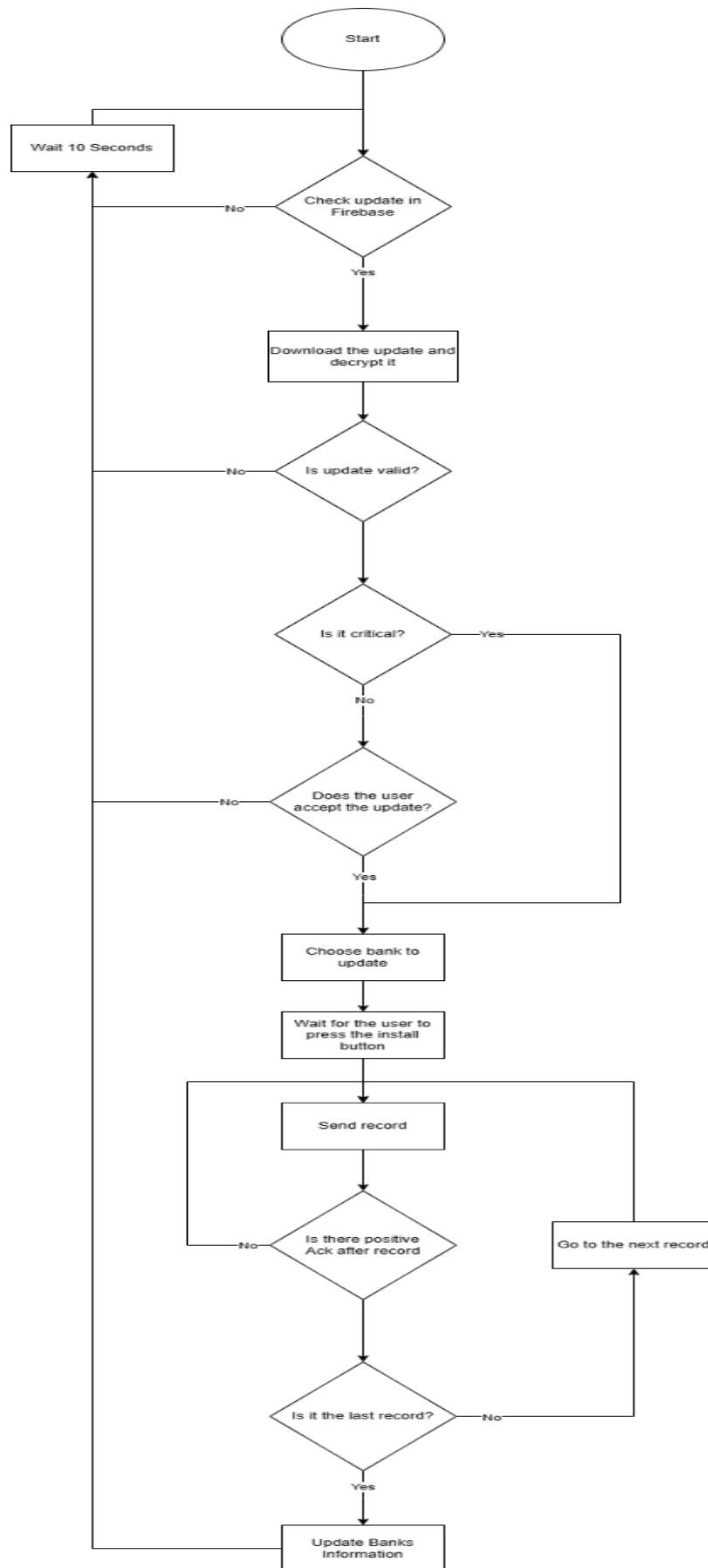
- **Real-time Data Synchronization:** Firebase offers real-time data synchronization, ensuring that firmware updates uploaded through the website are instantly available for retrieval by the Raspberry Pi. This dynamic synchronization ensures that the latest firmware is promptly accessible for deployment.
- **Scalability:** Firebase is well-known for its scalable infrastructure, making it an ideal choice for handling firmware updates across a multitude of connected devices. Whether updating a single STM microcontroller or an entire fleet, Firebase accommodates the scalability requirements seamlessly.
- **User Authentication and Security:** Firebase provides robust user authentication and security features, safeguarding the firmware update process. This ensures that only authorized users can upload firmware updates and that the integrity of the firmware data is maintained.
- **Cross-Platform Compatibility:** Firebase supports cross-platform development, allowing the website and Raspberry Pi to seamlessly communicate and share data. This compatibility simplifies the integration process and enhances the overall efficiency of the firmware update system.
- **Ease of Integration:** Firebase offers straightforward integration with various platforms and devices, streamlining the connection between the website and Raspberry Pi. The ease of integration contributes to the overall agility of the firmware update process.

3.3.2. Encryption & Decryption

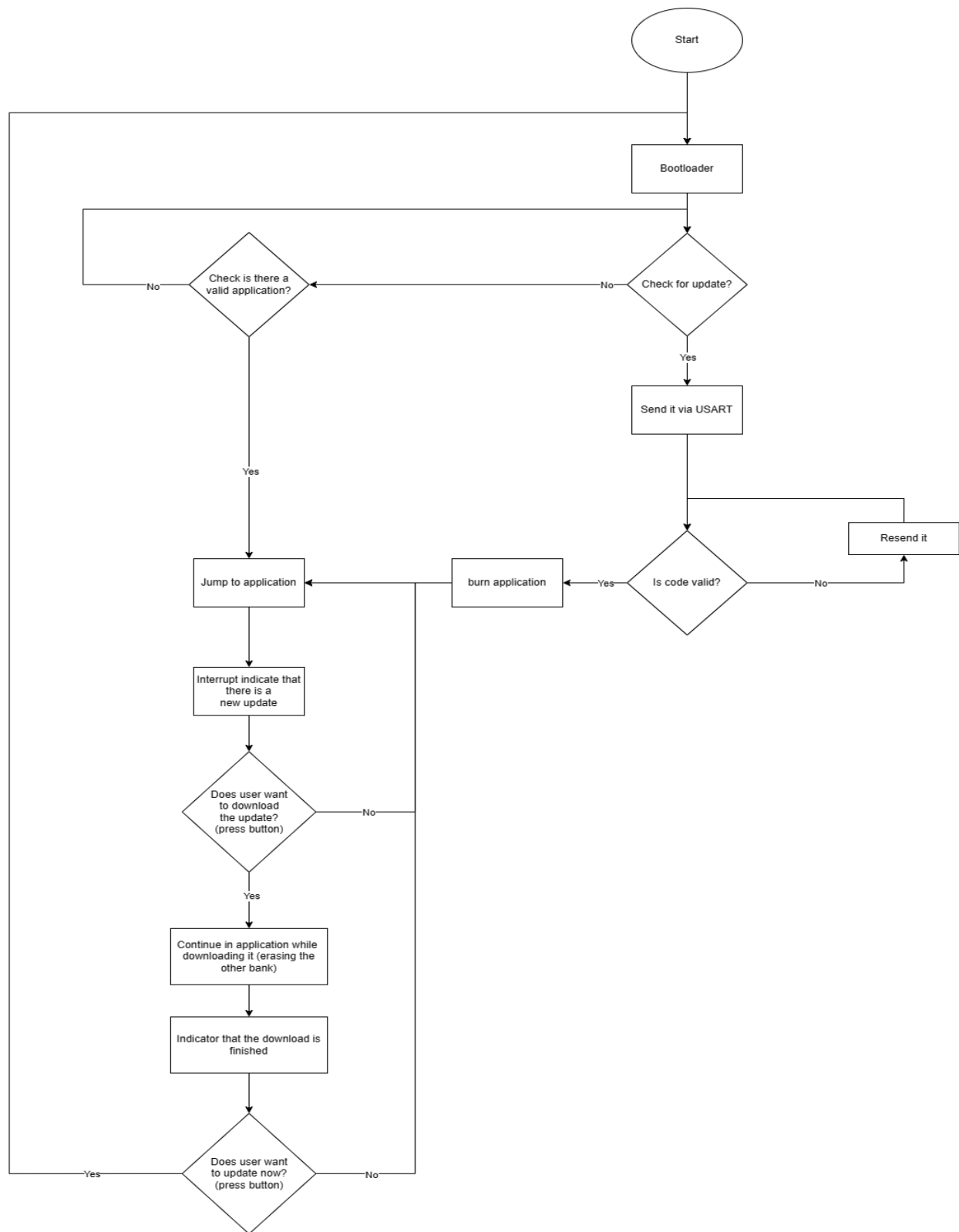
To add more security to the updates we encrypted the updates using the Fernet encryption and decryption constitute a symmetric-key (The key we used consist of 44 digits) cryptographic system, designed for secure and efficient data exchange between parties possessing a shared secret key. Fernet is particularly renowned for its simplicity and effectiveness, making it a popular choice for encrypting sensitive information. In the encryption process, plaintext data is combined with a timestamp and transformed into ciphertext using a shared secret key. The resulting encrypted message, or Fernet token, retains its integrity and confidentiality. Subsequently, decryption involves the reverse process, where the Fernet token is decrypted back into its original plaintext form using the same shared key. Fernet's strength lies in its reliance on the Advanced Encryption Standard (AES) algorithm and its utilization of authenticated cryptography, ensuring that both the sender and receiver can verify the authenticity of the exchanged data. Its ease of use, coupled with strong security measures, positions Fernet as a reliable choice for applications requiring secure communication and data protection.

3.3.3. Decide Bank for Programming & Update Level

The Raspberry Pi plays a pivotal role in orchestrating the firmware update process for STM microcontrollers. During its interaction with the STM, the Raspberry Pi dynamically evaluates various factors, utilizing the received data to determine which bank of the STM will host the burned code. This dynamic allocation ensures optimal utilization of resources and efficient management of firmware updates. Moreover, the Raspberry Pi enhances its functionality by incorporating an advanced security measure to discern critical updates. Utilizing a lengthy password consisting of 44 characters, the Raspberry Pi introduces an additional layer of authentication specifically for critical updates. This robust password serves as a safeguarding mechanism, allowing the Raspberry Pi to identify whether an incoming update is deemed critical. In the event of a critical update, the user is deprived of the option to refuse it, as such updates typically address crucial vulnerabilities or implement essential features vital for the device's optimal performance and security. The Raspberry Pi's capability to make informed decisions regarding firmware banks and its implementation of stringent security measures for critical updates underscore its role as a sophisticated and intelligent coordinator in the STM firmware update ecosystem.



3.4. The FOTA Diagram



CHAPTER 4: REAL TIME OPERATING SYSTEM (RTOS)

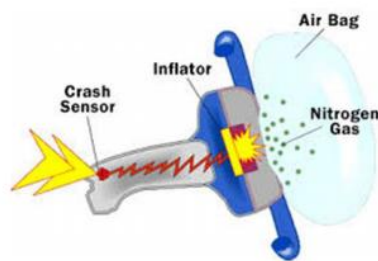
4.1. What is RTOS?

4.1.1 What are Real Time Systems?

A Real time system is one in which the correctness of the computations not only depends upon the logical correctness of the computation, but also upon the time at which the result is produced. If the timing constraints are not met, system failure is said to have occurred.

There is a lot of Characteristics for Real Time Applications (RTAS) like:

- 1- RTAs are not fast executing applications.
- 2- RTAs are time deterministic applications, that means, their response time to events is almost constant.
- 3- There could be small deviation in RTAs response time, in terms of MS or seconds which will fall into the category of soft real time applications.
- 4- Hard real time functions must be completed within a given time limit. Failure to do so will result in absolute failure of the system.



Hard-Real Time application



May be soft-real time application ?
Delay is tolerable

Figure 7 Real Time Applications

Real-Time Application has two main features:

- Time Deterministic – Response time to events is always almost constant.
- You can always trust RTAs in terms of its timings in responding to events.

4.1.2 What is a Real time OS?

It's an Especially designed to run applications with very precise timing. and a high degree of reliability
To be considered as "real-time", an operating system must have a known maximum time for each of the critical operations that it performs. Some of these operations include.

Handling of interrupts and internal system exceptions

Handling of Critical Sections.

Scheduling Mechanism, etc.

RTOS



Figure 8 RTOS

4.2. RTOS in Software Layered Architecture

RTOS in Software Layered Architecture has a different position, it has a vertical

Position because it needs a lot of requirements:

1- RTOS needs to access the hardware registers like HW Timer for Tick Time, Context switching to switch between tasks and stacking because it needs to control the whole system.

2-RTOS provides services to application to make algorithms in application easier by dividing the system to small components and every component has its own logic and every component can talk with another one.

For example, The AUTOSAR-based RTOS emerges as a beacon of standardized excellence, dictating the orchestration of software components, interfaces, and intercommunication within automotive systems. This framework accommodates diverse operating systems, weaving an intricate fabric of compatibility and interoperability between ECUs.

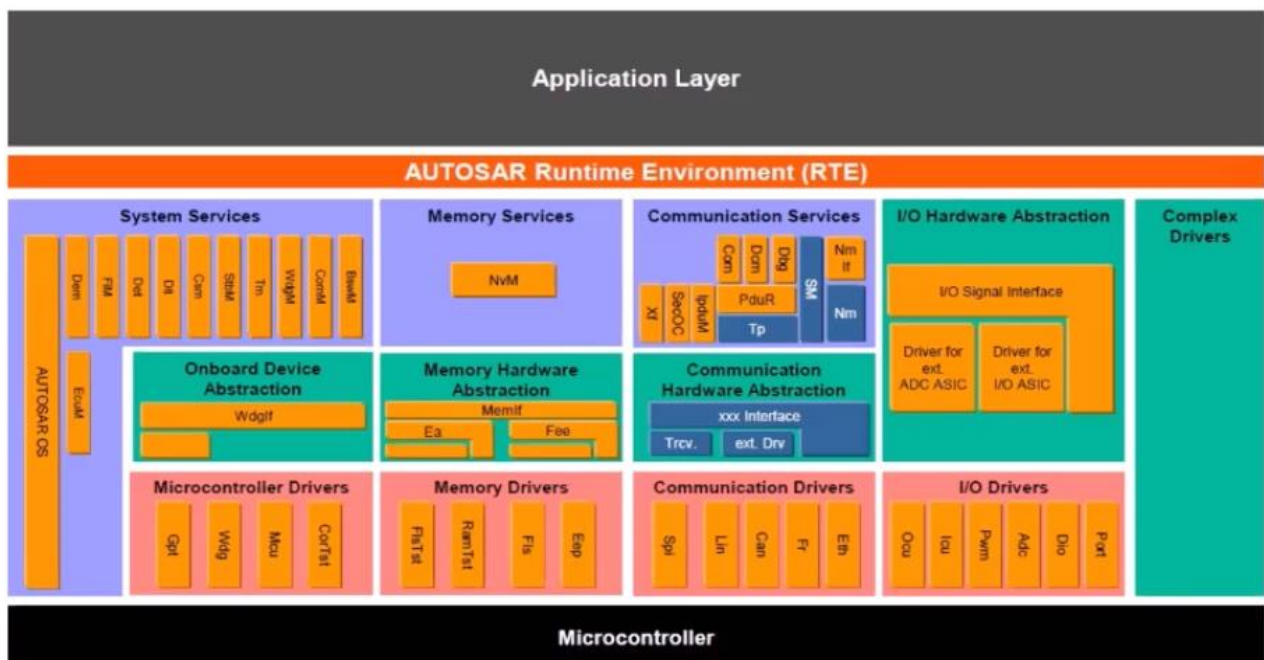


Figure 9 AUTOSAR

4.3. Why do we need RTOS in our project?

In general, we need RTOS in critical application like Missile guidance, anti-lock braking system and air bag, so it's demanding that we use RTOS in Automotive industry to add these privileges:

1. Deterministic: No random execution patterns
2. Predictable response time
3. Time Bound and priority-based scheduling.
4. It works under worst case assumptions.
5. Limited memory resources

Especially in our project we used RTOS to provide some features from used features in automotive industry like:

1. Adaptive Cruise Control
2. Lane Keep Assistance
3. Adaptive Light Control
4. Automated Emergency Actions (Braking System – Airbag System)

To achieve multitasking which means multiple tasks need to perform at the same time because each task takes a finite amount of computation time and resources. This process is repeated with very high-speed giving illusion to the user that tasks are performed the same time or parallelism and to integrate these Features with each other we must use OS to keep these features sharp enough to being closer to reality in the Automotive industry.



Figure 10 Airbag System

4.4. Designing A Real Time Systems

4.4.1. The Most Important Components of Design

To design A reliable Real Time System, we need to put in our heads some considerations to meet the critical Requirements, so scheduling some tasks over a timeline is a very easy task, we only need to define the following parameters to be able to sketch timeline:

1. Scheduling policy
2. System tick rate
3. Task Parameters
 - Periodicity
 - Priority
 - Execution Time
 - Deadline

A good real time design is one that answers the following questions.

- Is the system Deterministic?
- Is the system loaded?
- Is the system feasible?
- Is the system responsive?

So, We do some Calculations in our project for Execution time and we assume that Deadlines of Tasks is its Periodicity ,we decide the System tick based on the smallest periodicity of tasks and it must be bigger than the total time of execution of tasks ,when we calculate the Execution time for every task, we calculate their summation in Hyper period and divided summation by hyper period to calculate the CPU load to see the system is loaded and feasible or not ,The CPU load when all features is add is 50%.

We test every action we take as a user in our project, and we see what will happen after every action to see if our project is Responsive or not.

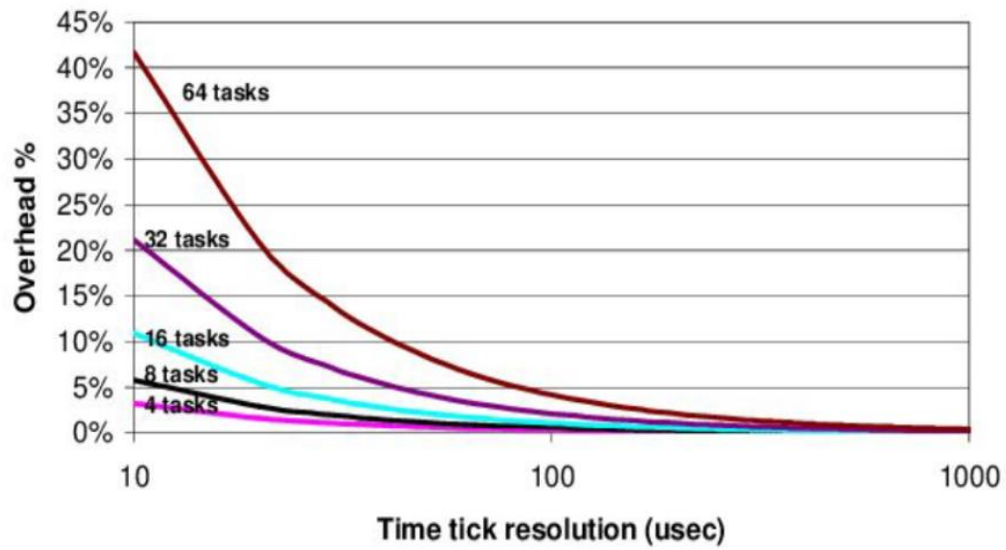


Figure 11 CPU Load

CHAPTER 5: HARDWARE

5.1. Components:

5.1.1. Lane Keep Assistance:

5.1.1.1. RPi Camera Module

The Camera Module 2 can be used to take high-definition video, as well as stills photographs. The camera works with all models of Raspberry Pi 1, 2, 3 and 4.



Figure 12 RPi Camera Module

5.1.2. Adaptive Cruise Control:

5.1.2.1. Ultrasonic Sensor

ultrasonic sensors measure distance by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception.



Figure 13 Ultrasonic Sensor

5.1.3. Adaptive Lighting System:

5.1.3.1. LDR Sensor

Light Dependent Resistor is a special type of resistor that works on the photoconductivity principle means that resistance changes according to the intensity of light. It is often used as a light sensor.

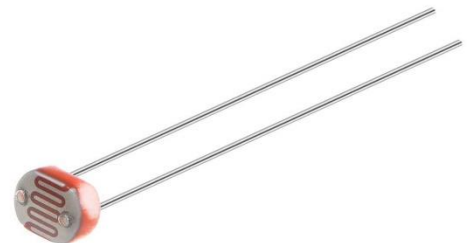


Figure 14 LDR Sensor

5.1.4. Airbag System:

5.1.4.1. Thin Film Pressure Sensor

The Force Sensitive Resistor (FSR) type pressure sensor is a flexible thin film sensor whose resistance decreases along with the pressure increases. The pressure sensor supports static/dynamic pressure sensing and has fast response speed.



Figure 15 FSR Sensor

5.1.5. Communication:

5.1.5.1. HC-05 Bluetooth Module

is designed for wireless communication. This module can be used in a master or slave configuration. It uses serial communication to communicate with devices. It communicates with microcontroller using serial port (USART).

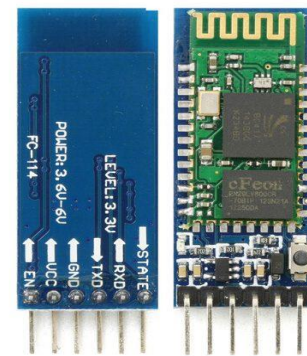


Figure 16 Bluetooth Module

5.1.6. DC Motors:

5.1.6.1. DC Gear Motor

A gear motor is an all-in-one combination of a motor and gearbox. The addition of a gearbox to a motor reduces the speed while increasing the torque output.



Figure 17 DC Gear Motor

5.1.6.2. L298N Motor Driver Module

This **L298N** is a high-power motor driver module for driving DC and Stepper Motors. It can control up to 4 DC motors, or 2 DC motors with directional and speed control.

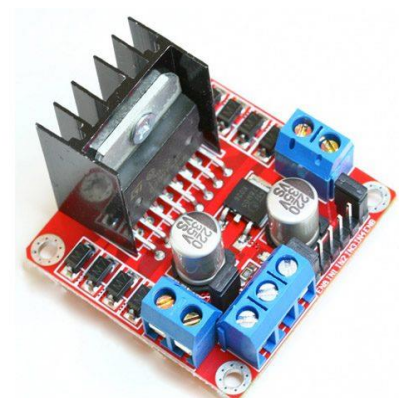


Figure 18 L298N Driver Module

5.2. Schematic & Wiring:

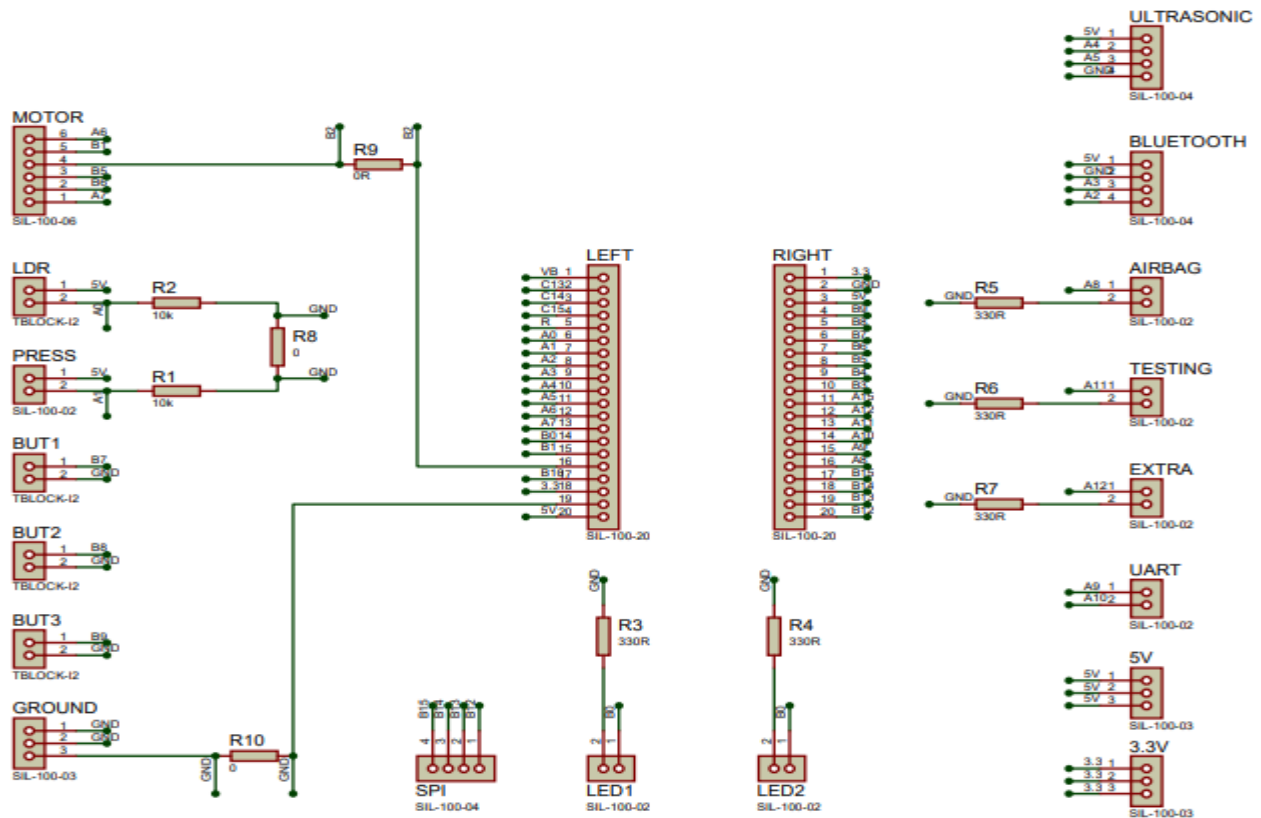


Figure 19 Schematic & Wiring

5.3. PCB Layout:



Figure 20 PCB Manufacturing Process

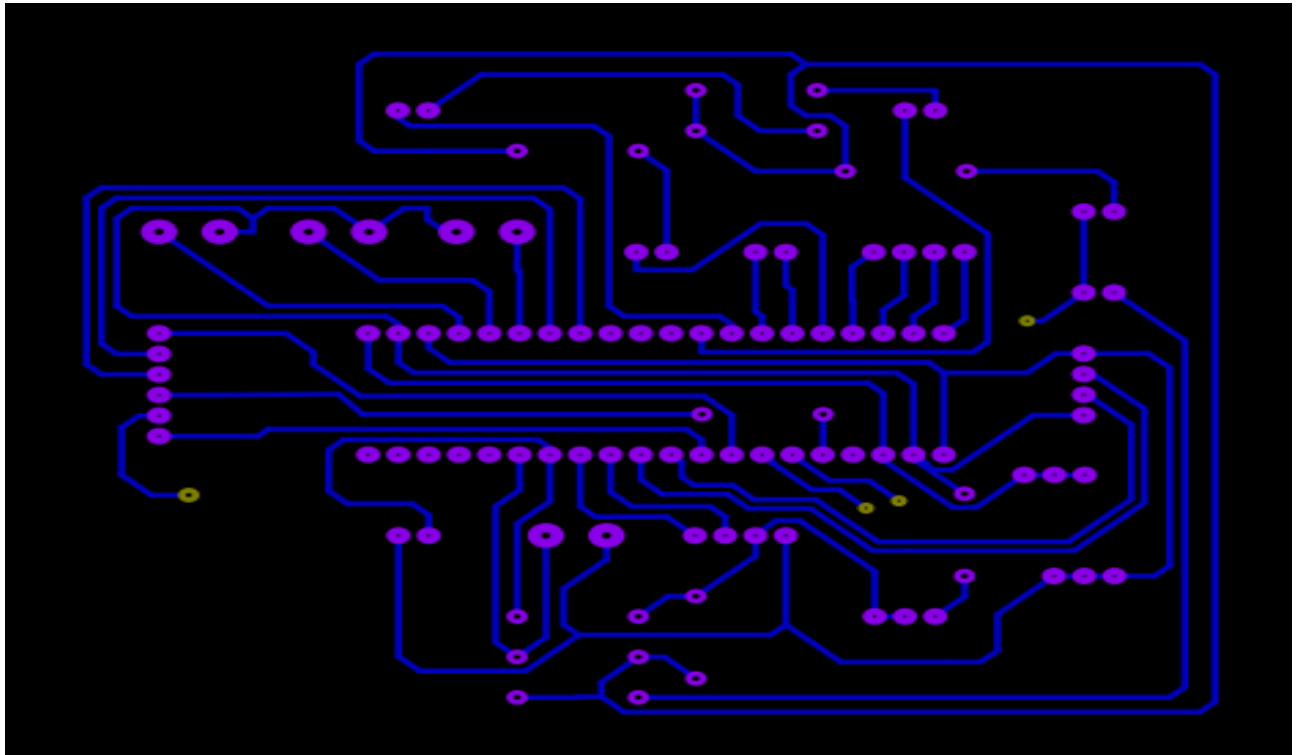


Figure 22 Bottom Copper Layer

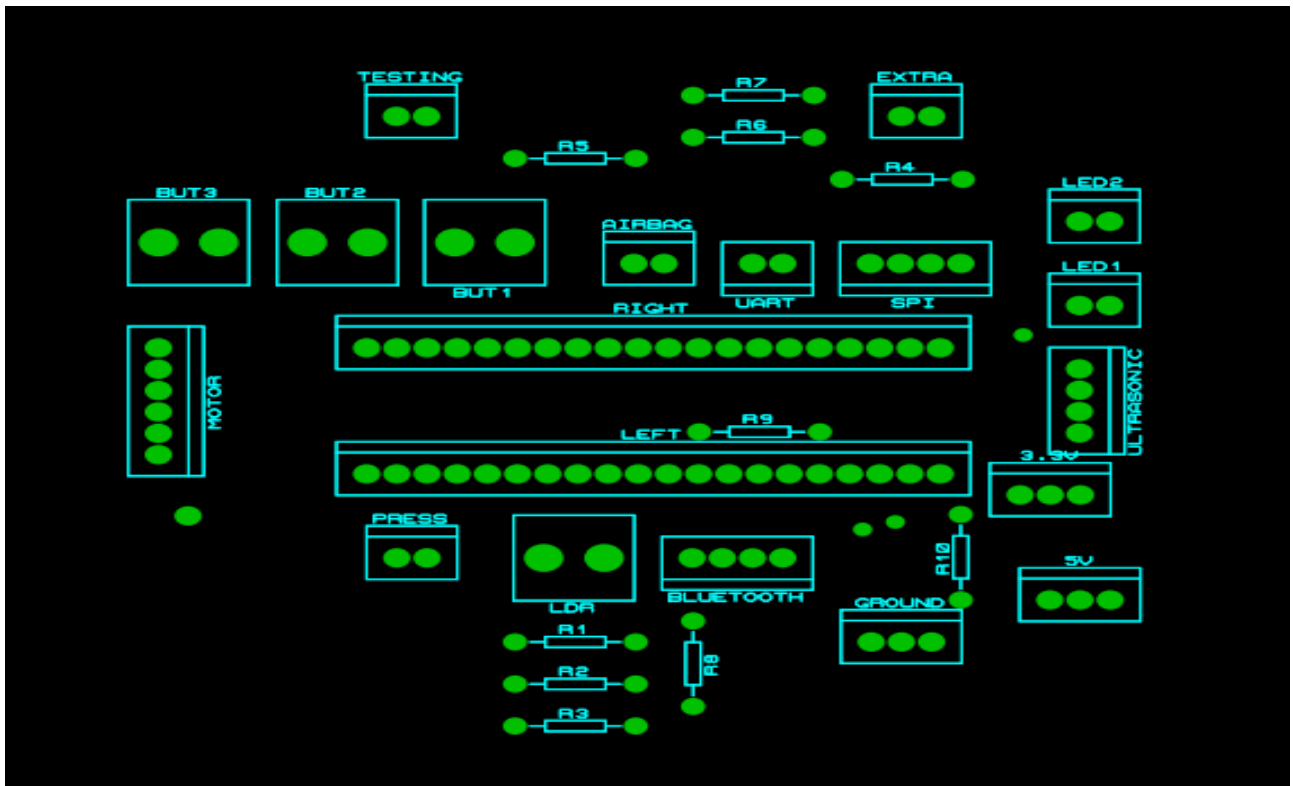


Figure 21 Top Silk Layer

CHAPTER 6: ARTIFICIAL INTELLIGENCE (AI) MODEL

6.1. What is AI?

AI, or artificial intelligence, refers to the development of computer systems that can perform tasks that typically require human intelligence. AI encompasses a broad range of technologies and approaches that enable machines to perceive their environment, reason, learn, and make decisions or take actions based on that information.

There are two main types of AI: Narrow AI and General AI.

Narrow AI: Also known as weak AI, narrow AI is designed to perform specific tasks within a limited domain. Examples of narrow AI include voice assistants like Siri and Alexa, image recognition systems, recommendation algorithms, and self-driving cars. These systems are focused on solving particular problems and are not capable of generalizing their knowledge to other domains.

General AI: Also referred to as strong AI or artificial general intelligence (AGI), general AI aims to replicate human-level intelligence and abilities. This type of AI would possess the cognitive capabilities to understand, learn, and apply knowledge across different domains, similar to human intelligence. However, the development of true general AI is still a subject of ongoing research and remains a hypothetical concept.

AI algorithms and models can be trained through machine learning techniques, such as supervised learning, unsupervised learning, and reinforcement learning. These algorithms learn from large amounts of data, identify patterns, and make predictions or decisions based on the learned information.

AI finds applications across various fields, including healthcare, finance, transportation, manufacturing, and entertainment. It has the potential to revolutionize industries, improve efficiency, and address complex problems. However, ethical considerations and responsible development are crucial to ensure AI benefits society while minimizing potential risks.

6.2. Why do we need AI in our project?

AI plays a crucial role in Auto NXT project for several reasons:

Object Detection and Recognition: AI algorithms can analyze sensor data from cameras, LiDAR, and radar to detect and recognize objects such as vehicles, pedestrians, and obstacles. This information is essential for ADAS features like automatic emergency braking, lane departure warning, and blind-spot detection.

Decision Making: AI enables ADAS systems to make real-time decisions based on the perceived environment. By processing sensor data and applying algorithms, AI can determine appropriate actions, such as adjusting vehicle speed, changing lanes, or initiating evasive maneuvers to avoid collisions.

Predictive Analysis: AI algorithms can analyze patterns in sensor data and make predictions about future events or behavior. This capability is valuable for ADAS systems to anticipate potential hazards or risky situations and take proactive measures to prevent accidents.

Driver Assistance and Augmentation: AI in ADAS systems is designed to assist drivers and enhance their capabilities. It can provide warnings, alerts, and guidance to help drivers stay focused, avoid potential dangers, and make informed decisions on the road.

Overall, AI in ADAS systems improves road safety by enhancing situational awareness, enabling proactive decision-making, and assisting drivers in avoiding accidents. It has the potential to reduce human error, mitigate risks, and ultimately contribute to the goal of achieving autonomous driving in the future.



Figure 23 Lane Keep Assistance

6.3. Calibration & Training & Results

Using a Raspberry Pi and a Camera module to implement lane assist for our RC car. Here's a high-level overview of the steps involved:

Hardware Setup: Connect the Camera module to the Raspberry Pi and ensured that the camera is properly configured and functioning.

Image Acquisition: Use the Raspberry Pi to capture real-time video frames from the camera module. This achieved by using OpenCV libraries in Python.

Image Processing: Apply computer vision techniques to analyze the video frames and extract relevant information, such as lane markings. Techniques like edge detection, color thresholding, and perspective transformation can be used to identify the lanes.

Lane Detection: Implement algorithms to detect and track the lane markings within the video frames. This involved techniques like Convolutional Neural Networks (CNNs) for more advanced approaches.

Lane Tracking and Control: Based on the lane detection results, develop control algorithms to steer the RC car to follow the detected lane. This achieved by adjusting the steering angle based on the position of the lanes within the image.

Integration: Connect the control mechanism to the RC car's steering system, allowing the Raspberry Pi to send commands to control the car's movement.

Testing and Refinement: Experiment with the lane assist system in a controlled environment, such as a track. Fine-tune the algorithms and parameters to improve the system's accuracy and responsiveness.

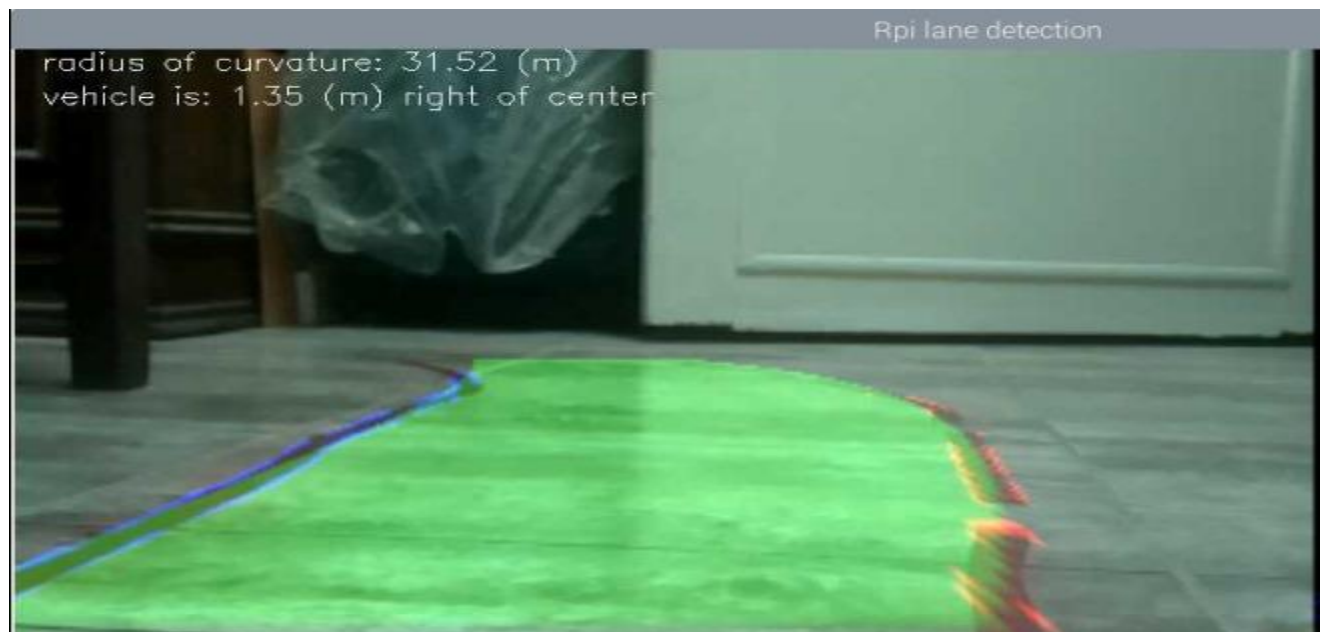


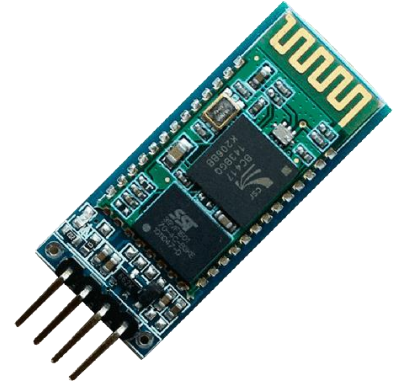
Figure 24 Detecting Lane Lines

CHAPTER 7: SYSTEM APPLICATION

In the following chapter, we will discuss the system app and its features.

7.1. Remote Controlled (via Bluetooth)

At the heart of our innovation is Bluetooth technology, a wireless communication standard that enables seamless connectivity between devices. By leveraging Bluetooth, our RC car can be controlled remotely using a smartphone or a dedicated controller, offering a level of flexibility and interactivity that was once reserved for larger-scale vehicles.



7.1.1. Working Theory

7.1.1.1. Pairing and Connection:

The RC car is equipped with a Bluetooth module that establishes a connection with the controlling device, whether it's a smartphone, tablet, or a custom-made controller.

Pairing is a one-time process, creating a secure and reliable link between the controller and the RC car.

7.1.1.2. Intuitive Control Interface:

The controlling device becomes the interface for steering, acceleration, and other functionalities.

Through a user-friendly app or controller, operators can enjoy precise control over the RC car's movements, creating a more immersive driving experience.

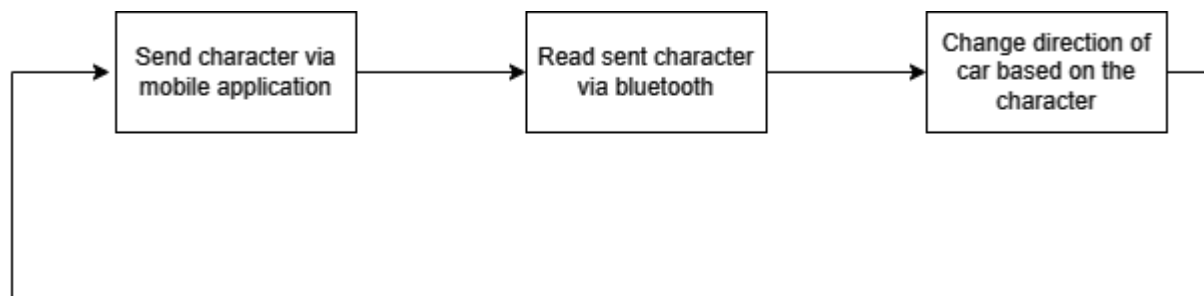


Figure 25 Bluetooth Controlled Diagram

7.2. Adaptive Lighting Control System

Adaptive Light Control transcends the limitations of traditional static headlights. Instead of a fixed beam, ALC harnesses the power of LDR sensors to intelligently modulate the direction and intensity of light. The result is a dynamic lighting system that adapts to the ever-changing conditions of the road.



7.2.1. Working Theory

7.2.1.1. Sensing Ambient Light:

LDR sensors monitor the ambient light conditions, gauging factors such as natural sunlight, streetlights, and oncoming headlights.

7.2.1.2. Dynamic Adjustment:

The data gathered by LDR sensors is processed in real-time by the Adaptive Light Control system.

The PWM signal, generated by the Adaptive Light Control system, regulates the power supplied to the headlights.

In response to changes in ambient light, the system adjusts the intensity of the vehicle's headlights to optimize visibility

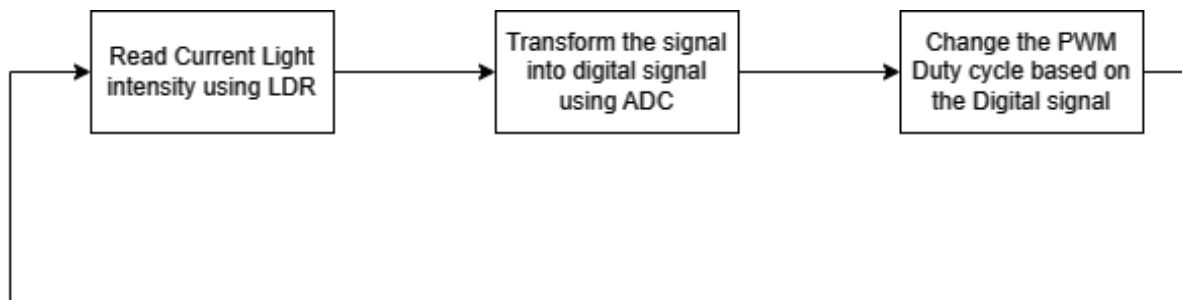


Figure 26 LDR Working Diagram

7.3. Adaptive Cruise Control System

At the heart of our Adaptive Cruise Control systems lie ultrasonic sensors – unassuming devices that emit high-frequency sound waves and interpret their echoes. These sensors function as the vehicle's silent guardians.

7.3.1. Working Theory

7.3.1.1. Sensing the Environment:

Ultrasonic sensors continuously emit sound waves, which bounce off objects in the vehicle's vicinity.

By measuring the time it takes for the echoes to return, the sensors calculate the distance to these objects.

7.3.1.2. Dynamic Speed Adjustment:

In the context of ACC, this real-time distance data is used to assess the vehicle's proximity to the one ahead.

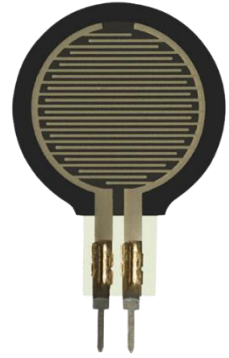
If the system detects a slower-moving vehicle in the same lane, it intelligently adjusts the speed to maintain a safe following distance.

7.3.1.3. Seamless Driving Experience:

The beauty of ACC lies in its seamless integration with the driver's intentions. If the road ahead is clear, the system allows the driver to enjoy a constant speed. However, when traffic conditions change, ACC steps in to dynamically adapt the speed, offering a stress-free and safer driving experience.

7.4. Automated Emergency System

At the core of our emergency system lies the Force-Sensing Resistor (FSR), a remarkable sensor capable of detecting changes in pressure and force. Placed strategically within the vehicle, FSR sensors act as sensitive touchpoints that can assess the severity and location of impact, offering a level of precision crucial for effective emergency response.



7.4.1. Working Theory

7.4.1.1. Constant Monitoring:

FSR sensors continuously monitor the force exerted on them, providing real-time data on the pressure distribution.

7.4.1.2. Impact Assessment:

During a collision or sudden deceleration, the FSR sensors detect changes in force and pressure.

If any collision is detected, a task with the highest priority in the RTOS-based system is enabled to start the Airbag system and all system tasks is suspended.

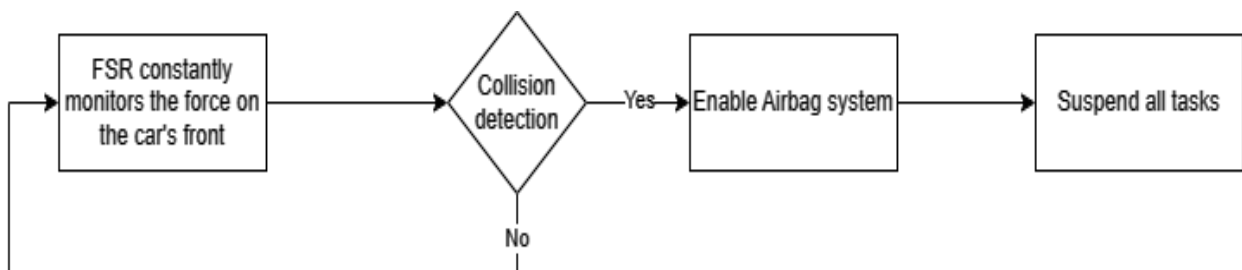


Figure 27 Emergency Airbag System Diagram

7.5. Lane Keep Assistance System

Lane Keep Assist (LKA) is a feature designed to prevent unintentional lane departure, offering a helping hand to drivers in maintaining proper lane positioning. Traditional systems rely on predefined rules and thresholds, but our project takes a leap forward by integrating a simple machine learning model that adapts and refines its performance based on real-world driving data.

7.5.1. Working Theory

7.5.1.1. Data Collection:

Sensors, such as camera, continuously collect data on the vehicle's position within the lane.

This data is fed into the machine learning model, allowing it to learn patterns and correlations between driver inputs and lane-keeping behaviors.

7.5.1.2. Real-Time Decision Making:

During actual driving scenarios, the model analyzes incoming data in real time.

If it detects signs of unintentional lane departure, the Lane Keep Assist system intervenes by gently adjusting the steering to guide the vehicle back into the lane.

7.5.1.3. Communication with the main ECU:

Communication with the main ECU is done through the SPI communication protocol.

Based on the received data the ECU, the steering rotation is changed to stay in the same lane.

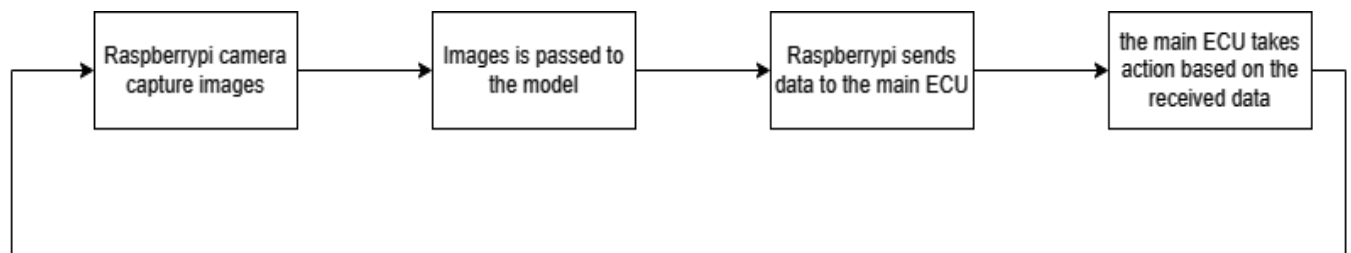


Figure 28 Lane Keep Assistance Diagram

CHAPTER 8: TEST CASES

Test Case ID	Test Case	Pass/Fail
T1	Check for update in firebase, download it and decrypt it.	Pass
T2	Check if update valid and does it critical.	Pass
T3	Check if user can accept or deny the update if it's not critical	Pass
T4	Check if FSR sensor detect the changes in force and pressure.	Pass
T6	Check if the airbag is enabled if there are any changes in force and pressure detected.	Pass
T7	Check if Breaking system is enabled when car detecting an obstacle	Pass
T8	Check if car changes its speed according to the distance to maintain a safe following distance from the vehicle ahead.	Pass
T9	Check if AI model can detect lane and car changes its direction.	Fail

CHAPTER 9: CONCLUSION

In conclusion, our project has revolutionized the automotive industry by introducing a range of advanced features and cutting-edge technologies. The integration of Firmware Over-The-Air (FOTA) updates enables seamless wireless updates, ensuring access to the latest features and optimizations.

Our project encompasses a comprehensive set of features, including Adaptive Cruise Control (ACC) for precise speed control, automated emergency actions for real-time safety alerts, remote control capabilities for enhanced convenience, and a comprehensive diagnostic system for proactive maintenance.

Furthermore, the integration of an AI model with lane keep assistance adds an intelligent layer to our project. By leveraging real-time video feeds and AI algorithms, our system ensures precise lane keeping, enhancing safety and reducing driver fatigue.

These features, combined with adaptive light control, real-time operating systems (RTOS), and other advancements, have elevated the performance, safety, and convenience of vehicles.

In summary, our project represents a significant milestone in the automotive industry, showcasing the potential for continuous innovation. By integrating advanced features and cutting-edge technologies, we have set a new standard for driving experiences, delivering enhanced performance, safety, and convenience to drivers and manufacturers alike

