Project report on

Vehicle Automation using CAN Protocol

Guided by

Mr. Ankush Tembhurnikar

Presented by:

Manish Zine **240844230100**

Sumit Dhapodkar **240844230084**

Rahul Shakya **240844230109**

At

Sunbeam Institute of Information Technology,

Hinjewadi.

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY, HINJEWADI.

This is to certify that the project

Vehicle Automation using CAN Protocal

Has been submitted by

Manish Zine **240844230100**

Sumit Dhapodakar **240844230084**

Rahul Shakya **240844230109**

In partial fulfillment of the requirement for the course of **PG Diploma In Embedded System Design (PG-DESD August 2024)** as prescribed by the **CDAC** ACTS, PUNE.

Place: Hinjewadi Date: 12-02-2025

ACKNOWLEDGEMENT

This project "**Vehicle Automation using CAN Protocal** " was a great learning experience for us and we are submitting this work to SUNBEAM (CDAC).
We all are very glad to mention the name of *Mr. **Ankush Tembhurnikar*** for his valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during the course of project work.
Our most heartfelt thanks goes to *Mr. Devendra Dhande* (Course Coordinator, PG-*DESD*) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day here in C-DAC SUNBEAM,Pune.

ABSTRACT

In modern vehicle automation systems, real-time rain detection, headlight automation, and adaptive control mechanisms are essential for enhancing safety and efficiency. This project presents a Raindrop Sensor and LDR Sensor System with a Servo Motor and LED, utilizing the Controller Area Network (CAN) protocol to automate mechanical responses based on rain and light intensity detection. The system integrates a raindrop sensor and an LDR sensor with a microcontroller unit (MCU) featuring CAN communication capability, along with a servo motor to activate wiper systems and control headlights automatically.

This system aims to enhance vehicle automation by implementing a Controller Area Network (CAN) integrated with the STM32F407 microcontroller for communication between two nodes. CAN is one of the most versatile and widely used technologies in the electronics industry, with applications in various practical scenarios.

The Controller Area Network is a serial, asynchronous, multi-master communication protocol designed for connecting electronic devices and control modules in automotive and industrial applications. CAN was specifically developed for automotive systems requiring a high level of data integrity and supports data rates of up to 1 Mbit/sec.

TABLE OF INDEX

Topic Page No.

1. Introduction ---------- 6

2. Literature Survey---------- 8

1. ARM Cortex M4 Architecture ----------9

3. Controller Area Network---------- 9

1. CAN Network ----------9

2. CAN Bus ----------10

3. CAN Bus Levels ----------10

4.CAN Frames ----------11

5. Bus Arbitration ----------13

4. Requirements---------- -14

4.1 Hardware Requirements ----------14

1.1 Introduction to STM32F407 Discovery Board ---------- 14

1.2Servo motor----------15

1.3 Rain drop sensor ----------16

1.4 LDR ----------17

1.5 LED ----------18

1.6 CAN Transceiver---------- 20

4.2. Software Requirements ----------21

1 STM32CubeIDE---------- 21

5. Project Architecture ---------- 23

5.1 Block Diagram ----------23

5.2 Description ----------23

6. Testing Images----------24

7. Conclusion and Future Scope ----------25

8. References ----------26

1] Introduction

A **"Rainwater Dripping Sensor and LDR Sensor in a Servo Motor and LED Control Using CAN Communication"** refers to a system where a raindrop sensor detects rain and an LDR sensor measures light intensity. Based on this information, a servo motor and LED, controlled via the CAN bus communication, adjust their operation accordingly. For example, the system can automatically activate a wiper mechanism, close a cover, or control a sprinkler system when it starts to rain. Additionally, it can adjust the headlight intensity based on ambient light conditions, enhancing automation and efficiency.

- **Key components: -**

  Rain water sensor: This is the primary component that detects the presence of raindrops. It usually works by measuring the change in electrical resistance when water drops land on its surface.

  LDR Sensor: An LDR (Light Dependent Resistor) changes its resistance based on light intensity, decreasing resistance in bright light and increasing it in darkness.

- **Microcontroller:**

  A small computer that processes the signal from the rain sensor and sends commands to the servo motor based on the detected rain level.

- **CAN bus interface:**

  A module that allows the microcontroller to communicate with the servo motor using the CAN protocol.

- **Servo motor:**

  An electric motor that can be precisely positioned at different angles, allowing it to adjust the position of a mechanism based on the rain sensor input.

Specifications:

- Working voltage: 5V
- Outputs: Digital (0 and 1) and Analog voltage (AO).
- adjustable via a pulse width modulation by varying the Duty cycle
- Utilizes a high-voltage LM393 comparator for reliable performance.
- Produces clean waveform signals with strong driving capacity (over 15mA).
- Boasts excellent resistance to oxidation and conductivity, ensuring a long operational life

LED: An LED (Light Emitting Diode) is a semiconductor device that emits light when an electric current passes through it. It is use to represent headlight in this project.

Specifications:

- **Voltage**: Typically, 1.8V to 3.5V (depends on color/type)
- **Current**: Usually 10mA to 20mA (standard LEDs)

- **Power Consumption**: Varies (e.g., ~20mW for standard LEDs)
- **Wavelength**: Depends on color (e.g., Red ~620-750nm, Blue ~450-495nm)
- **Luminous Intensity**: Measured in millicandela (mcd), varies by LED type
- **Viewing Angle**: Typically, 20°–120°
- **Lifespan**: 25,000–100,000 hours (depending on usage)

Pinout & Parts of Raindrop Sensor Module

The whole Rain drop sensor consists mainly of two parts: a Rain board and a Control Module.

The Rain Board has nothing other than Two pins as shown in the below image which should be connected to the control module.

The control module has a total of Six pins but four of them are major pins which to be connected to the microcontroller.

The other side two pins are to be connected to the Rain Board whereas, on the other side, four pins are:

1. **VCC (or +):** This pin is used to supply power to the raindrop sensor. It should be connected to the 5V output of your Arduino or an external power supply.
2. **GND (or -):** This pin is the ground connection and should be connected to the GND pin on your Arduino or a common ground reference if using an external power supply.
3. **AO (Analog Output):** This pin provides the analog output voltage that varies depending on the quantity of water detected by the sensor. You should connect this pin to an analog input pin on your STM32 microcontroller, such as A0, to read the sensor's output.

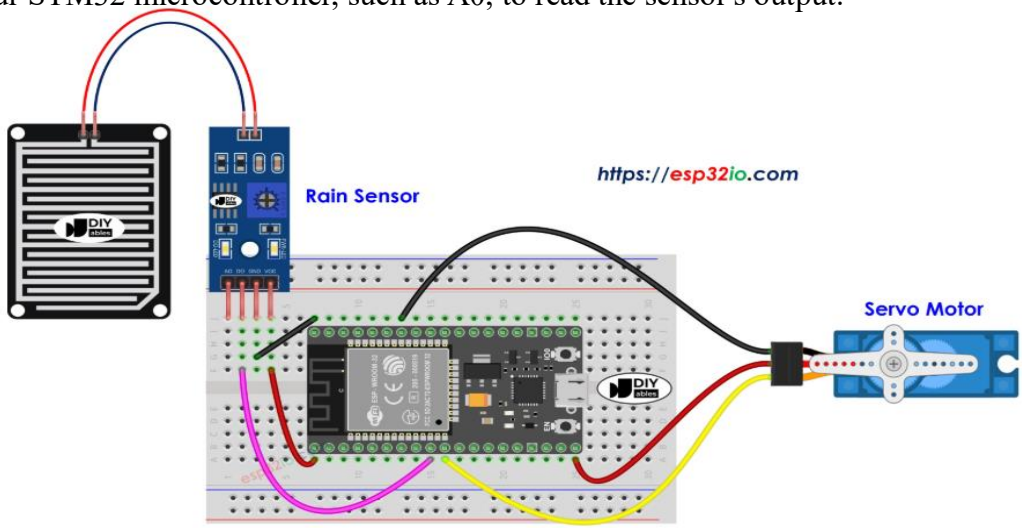


Fig 1.1 Rain Drop sensor with Servo motor

Pinout of LDR Sensor and LED Module

The LDR sensor has two pins: one is connected to **PA0** (configured as an ADC input), and the other is connected to **3V** for the power supply. This setup allows the microcontroller to measure the varying voltage based on light intensity.

On the other side, the **positive (+) terminal** of the LED is connected to **PE9**(configured as TIM2 for PWM), while the **negative (-) terminal** is connected to **GND**, allowing the LED to operate when powered.

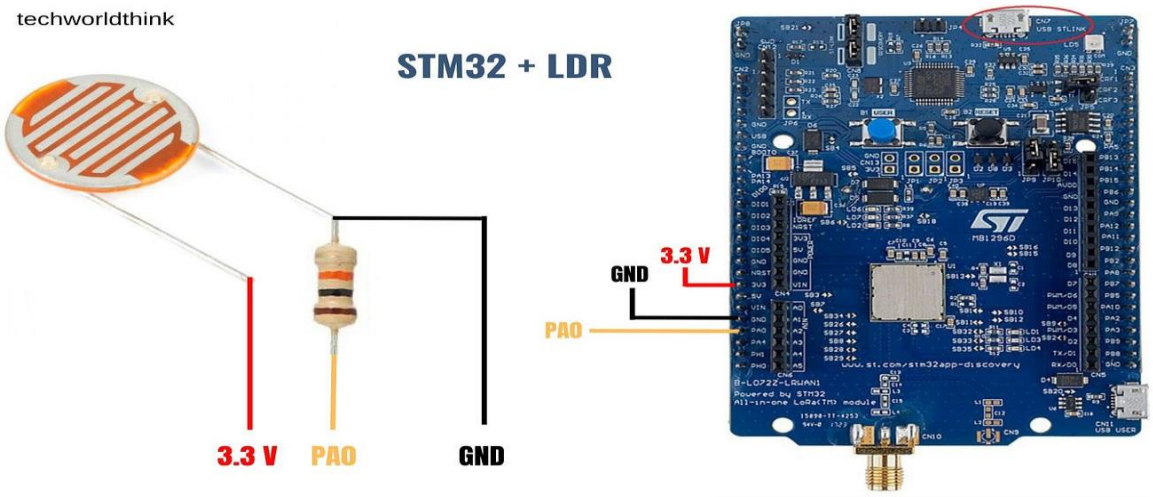


Fig 1.2 LDR

2] Literature survey

1] ARM- CORTEX M4 Architecture

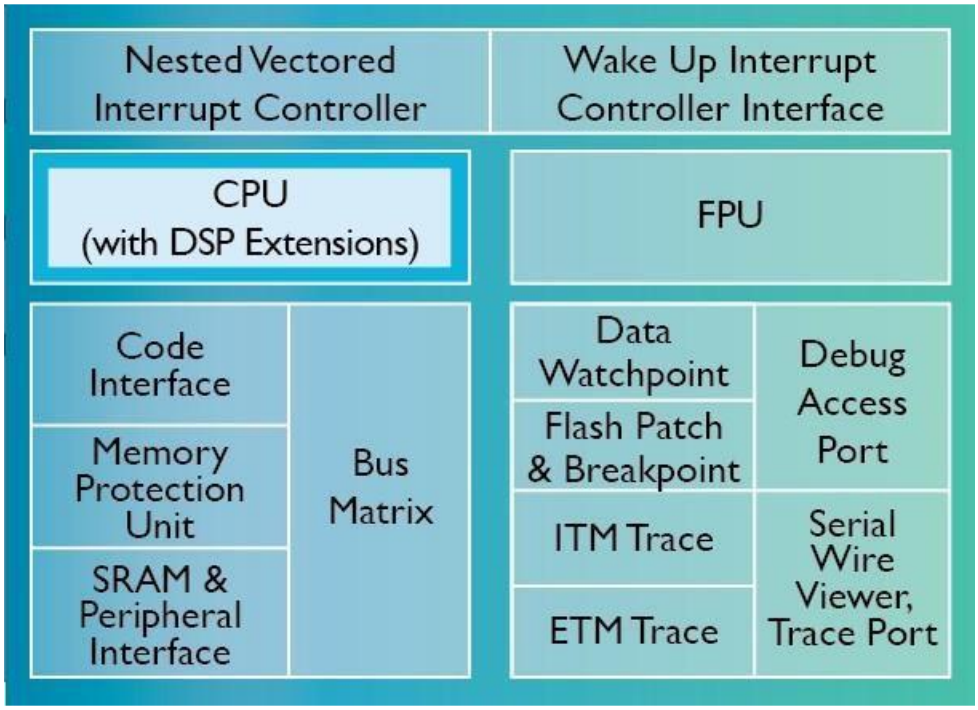


Fig. 2.1 ARM CORTEX M4 Architecture

The ARM cortex-M4 processor is a high-performance embedded processor with DSP instructions developed to address digital signal control markets that demand an efficient, easy-to-use blend of control and signal processing capabilities. The processor is highly configurable enabling a wide range of implementation from those requiring floating point operations, memory protection and powerful trace technology to cost sensitive devices requiring minimal area.

Key benefits:

1. Gain the advantages of a microcontroller with integrated DSP, SIMD, and MAC instructions that simplify overall system design, software development and debug

2. Accelerate single precision floating point math operations up to 10x over the equivalent integer software library with the optional floating-point unit (FPU)

3. Develop solutions for a large variety of markets with a full-featured ARMv7-M instruction set that has been proven across a broad set of embedded applications

4. Achieve exceptional 32-bit performance with low dynamic power, delivering leading system energy efficiency due to integrated software-controlled sleep modes, extensive clock gating and optional state retention.

3] Controller Area Network

The Controller Area Network bus, or CAN bus, is a vehicle bus standard designed to allow devices and microcontrollers to communicate with each other within a vehicle without a host computer.

CAN is a reliable, real-time protocol that implements a multicast, data-push, publisher /subscriber model. CAN messages are short (data payloads are a maximum of 8 bytes, headers are 11 or 29 bits), so there is no centralized master or hub to be a single point of failure and it is flexible in size. Its real-time features include deterministic message delivery time and global priority through the use of prioritized message IDs.

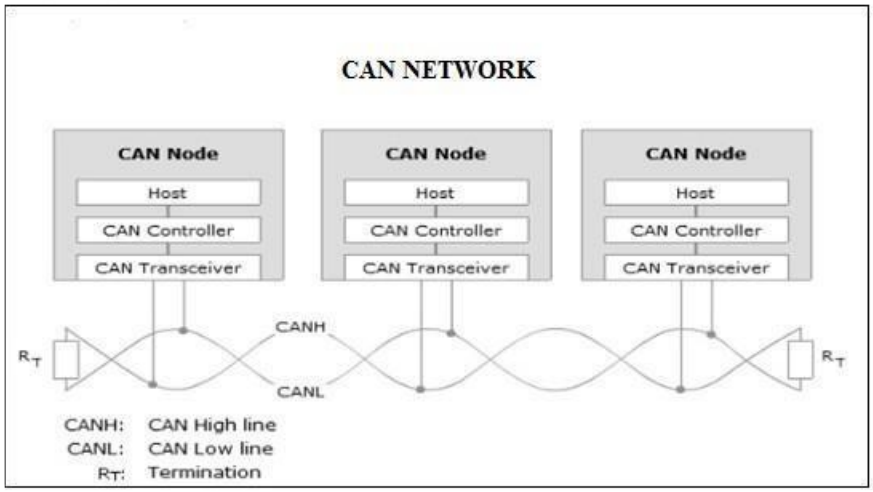


Fig. 3.1 CAN network

2] CAN Bus

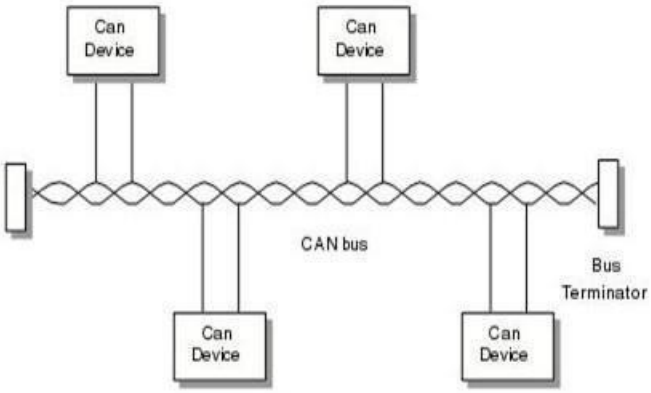


Fig 3.2. CAN Bus

Physical signal transmission in a CAN network is based on transmission of differential voltages **(differential signal transmission**). This effectively eliminates the negative effects of interference voltages induced by motors, ignition systems and switch contacts. Consequently, the transmission medium (CAN bus) consists of two lines: CAN High and CAN Low. Physical signal transmission in a CAN network is based on transmission of differential voltages **(differential signal transmission**). This effectively eliminates the negative effects of interference voltages induced by motors, ignition systems and switch contacts. Consequently, the transmission medium (CAN bus) consists of two lines: CAN High and CAN Low.

Twisting of the two lines reduces the magnetic field considerably. Therefore, in practice twisted pair conductors are generally used as the physical transmission medium. Due to finite signal propagation speed, the effects of transient phenomena (**reflections**) grow with increasing data rate and bus extension. Terminating the ends of the communication channel using **termination resistors** (simulation of the electrical properties of the transmission medium) prevents reflections in a high-speed CAN network.

Twisting of the two lines reduces the magnetic field considerably. Therefore, in practice twisted pair conductors are generally used as the physical transmission medium. Due to finite signal propagation speed, the effects of transient phenomena (**reflections**) grow with increasing data rate and bus extension. Terminating the ends of the communication channel using **termination resistors** (simulation of the electrical properties of the transmission medium) prevents reflections in a high-speed CAN network.

The key parameter for the bus termination resistor is the so-called characteristic impedance of the electrical line. This is 120 Ohm. In contrast to ISO 11898-2, ISO 11898-3 (low-speed CAN) does not specify any bus termination resistors due to the low maximum data rate of 125 kbit/s.

3] CAN bus levels

Physical signal transmission in a CAN network is based on differential signal transmission. The specific differential voltages depend on the bus interface that is used. A distinction is made here between the high-speed CAN bus interface (ISO 11898-2) and the low-speed bus interface (ISO 11898-3).

ISO 11898-2 assigns logical “1” to a typical differential voltage of 0 Volt. The logical “0” is assigned with a typical differential voltage of 2 Volt. High-speed CAN transceivers interpret a differential voltage of more than 0.9 Volt as a dominant level within the common mode operating range, typically between 12 Volt and -12 Volts. Below 0.5 Volt, however, the differential voltage is interpreted as a recessive level. A hysteresis circuit increases immunity to interference voltages. ISO 11898-3 assigns a typical differential voltage of 5 Volt to logical “1”, and a typical differential voltage of 2 Volt corresponds to logical “0”. The figure “High-Speed CAN Bus Levels” and the figure “Low-Speed CAN Bus Levels” depict the different voltage relationships on the CAN bus.

High Speed CAN:

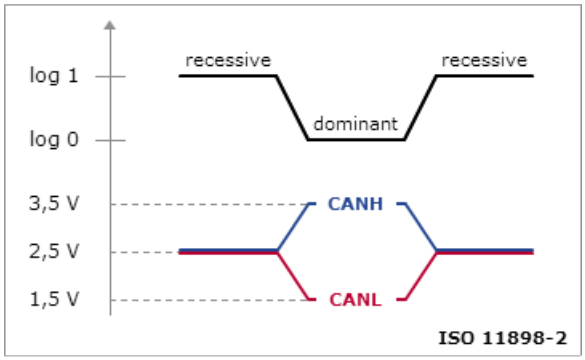


fig 3.3. high speed CAN voltage levels

Low Speed CAN:

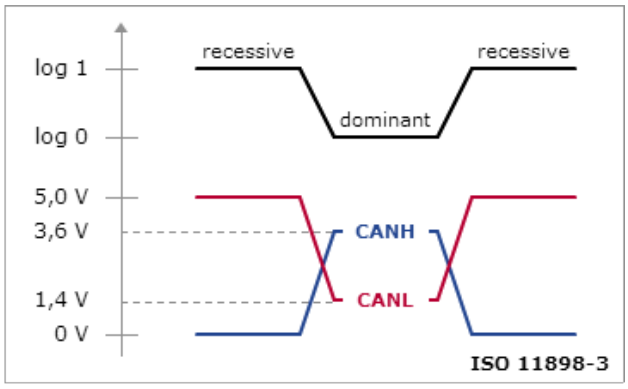


Figure 3.4 Low speed CAN voltage levels

4] CAN Frames

1. **Data frame**

The data frame is the most common message type, and comprises the Arbitration Field, the Data Field, the CRC Field, and the Acknowledgment Field. The Arbitration Field contains an 11-bit identifier and the RTR bit, which is dominant for data frames. Next is the Data Field which contains zero to eight bytes of data, and the CRC Field which contains the 16-bit checksum used for error detection. Last is the Acknowledgment Field.

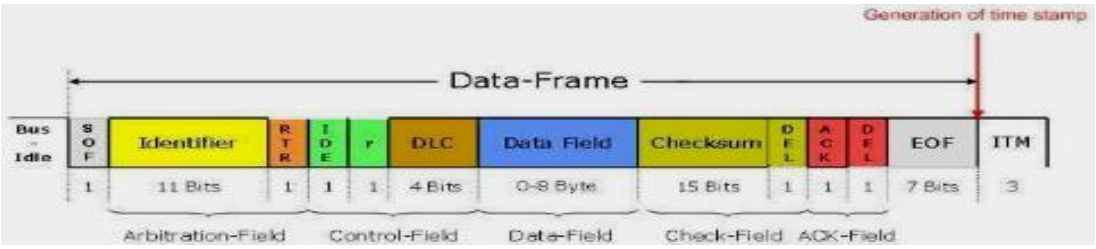
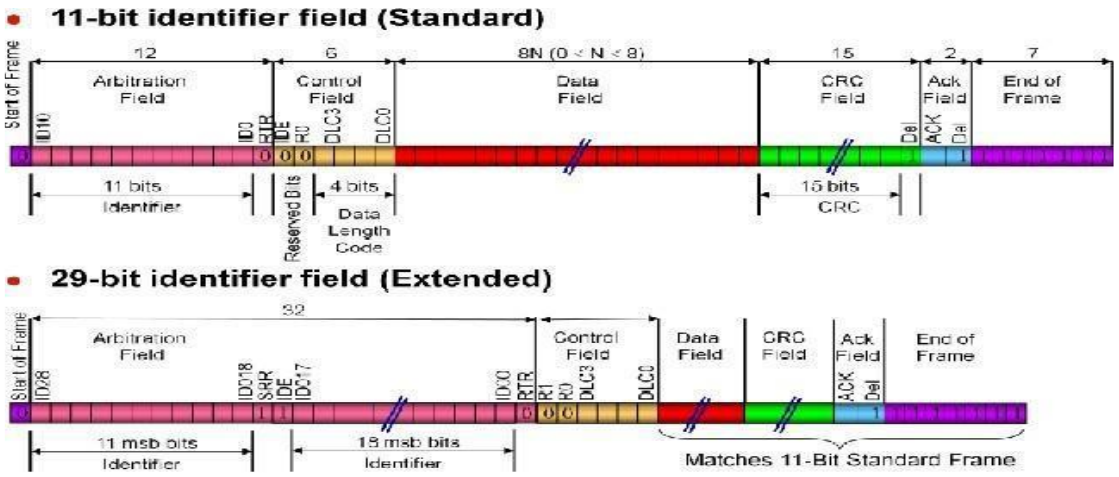


fig. 3.5 data frame



2. Remote frame
   The intended purpose of the remote frame is to solicit the transmission of data from another node. The remote frame is similar to the data frame, with two important differences. First, this type of message is explicitly marked as a remote frame by a recessive RTR bit in the arbitration field, and secondly, there is no data.

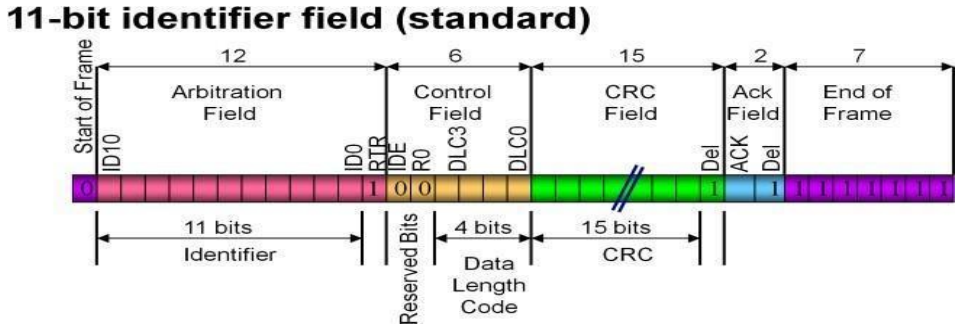


Fig. 3.6 remote frame

3. Error Frame

The error frame is a special message that violates the formatting rules of a CAN message. It is transmitted when a node detects an error in a message, and causes all other nodes in the network to send an error frame as well. The original transmitter then automatically retransmits the message. An elaborate system of error counters in the CAN controller ensures that a node cannot tie up a bus by repeatedly transmitting error frames.

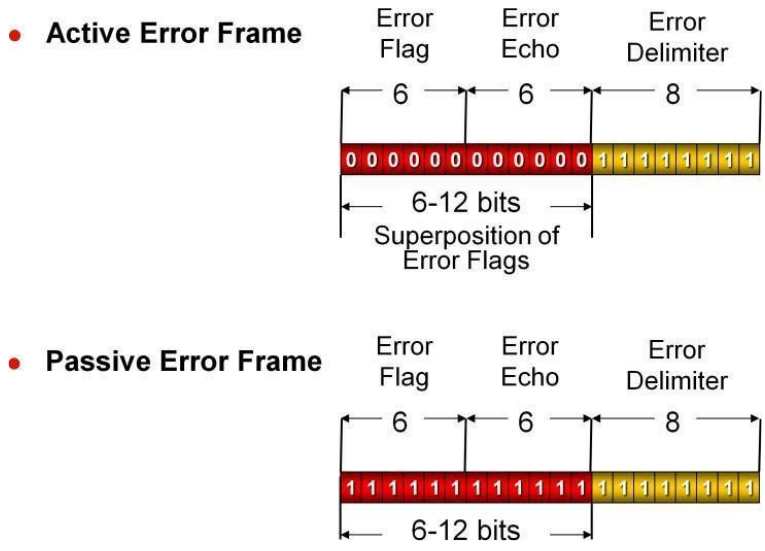


Fig 3.7 error frame

4. Overload frame
   The overload frame is mentioned for completeness. It is similar to the error frame with regard to the format, and it is transmitted by a node that becomes too busy. It is primarily used to provide for an extra delay between messages.

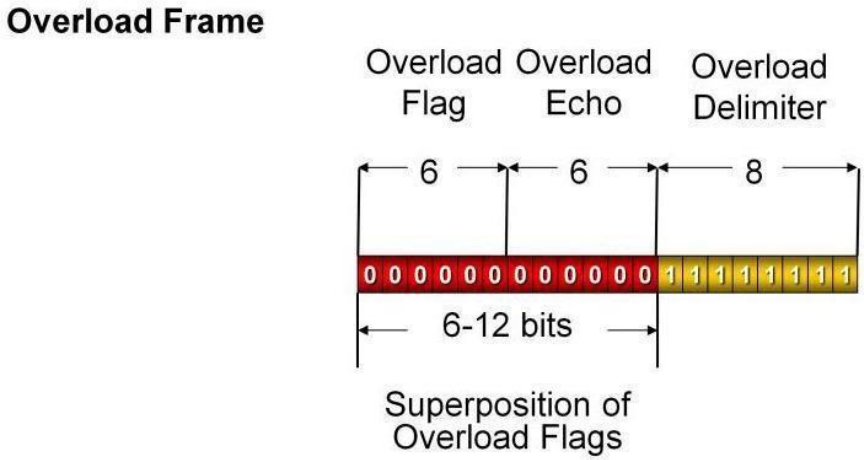


Fig.3.8. overload frame

5] BUS arbitration

The CAN communication protocol is a carrier-sense, multiple-access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP). CSMA means that each node on a bus must wait for a prescribed period of inactivity before attempting to send a message. CD+AMP mean that collisions

are resolved through a bit-wise arbitration, based on a pre-programmed priority of each message in the identifier field of a message. The higher priority identifier always wins bus access. That is, the last logic-high in the identifier keeps on transmitting because it is the highest priority.

Whenever the bus is free, any unit may start to transmit a message. If two or more units start transmitting messages at the same time, the bus access conflict is resolved by bit-wise arbitration using the Identifier. The mechanism of arbitration guarantees that neither information nor time is lost. If a data frame and a remote frame with the same identifier are initiated at the same time, the data frame prevails over the remote frame. During arbitration every transmitter compares the level of the bit transmitted with the level that is monitored on the bus. If these levels are equal the unit may continue to send. When a "recessive" level is sent and a "dominant" level is monitored, the unit has lost arbitration and must withdraw without sending one more bit.

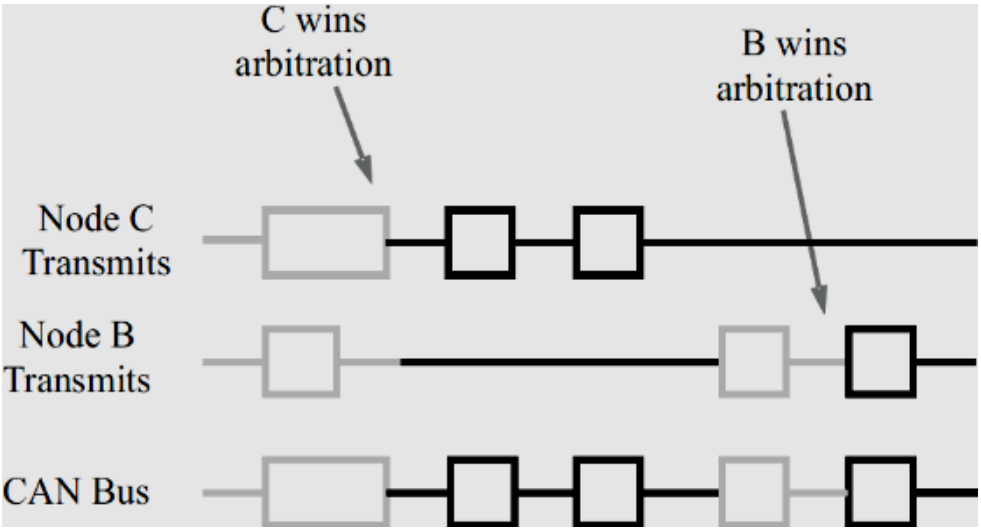


fig. 3.9. Bus arbitration

4] Requirements

1] Hardware Requirements

1.1] STM32F407 Discovery Board

The STM32F407xx family is based on the high-performance ARM® Cortex®-M4 32-bit RISC core with FPU operating at a frequency of up to 72 MHz, and embedding a floating point unit (FPU), a memory protection unit (MPU) and an embedded trace macro cell (ETM). The family incorporates high-speed embedded memories (up to 1 Mbytes of Flash memory, up to 192 Kbytes of RAM) and an extensive range of enhanced I/Os and peripherals connected to two APB buses.

They also feature standard and advanced communication interfaces: up to two I2Cs, up to three SPIs (two SPIs are with multiplexed full-duplex I2Ss), three USARTs, up to two UARTs, CAN and USB.

To achieve audio class accuracy, the I2S peripherals can be clocked via an external PLL. The STM32F303xB/STM32F303xC family operates in the -40 to +85°C and -40 to +105 °C temperature ranges from a 2.0 to 3.6 V power supply. A comprehensive set of power-saving mode allows the design of low-power applications.

Fig 4.1 STM32x407M

1.2] Servo Motor: To make this motor rotate, we have to power the motor with +5V using the Red and Brown wire and send PWM signals to the Orange colour wire. Hence we need something that could generate PWM signals to make this motor work, this something could be anything like a 555 Timer or other Microcontroller platforms like Arduino, PIC, ARM or even a microprocessor like Raspberry

Pie.TowerPro SG-90 Features

- Operating Voltage is +5V typically
- Torque: 2.5kg/cm
- Operating speed is 0.1s/60°
- Gear Type: Plastic
- Rotation: 0°-180°
- Weight of motor: 9gm
- Package includes gear horns and screws

Fig4.2. servo motor

1.3] Rain Sensor: A sensor that is used to notice the water drops or rainfall is known as a rain sensor, this kind of sensor works like a switch. This sensor includes two parts like sensing pad and a sensor module. Whenever rain falls on the surface of a sensing pad then the sensor module reads the data from the sensor pad to process and convert it into an analog or digital output. So the output generated by this sensor is analog (AO) and digital (DO)

Specifications: The specifications of rain sensors like different parameters with values are mentioned below.

- Operating voltage ranges from 3.3 to 5V
- The operating current is 15 mA
- The sensing pad size is 5cm x 4 cm with a nickel plate on one face.
- Comparator chip is LM393
- Output types are AO (Analog o/p voltage) & DO (Digital switching voltage)
- The length & width of PCB module 3.2cm x 1.4cm
- Sensitivity is modifiable through Trimpot
- Red/Green LED lights indicators for Power & Output

Fig 4.3. Rain Sensor with CAN Interface

1.4] <u>LDR:</u>

A light-dependent resistor (LDR) sensor, also known as a photoresistor, is. It changes its electrical resistance based on the amount of light it receives.

How it works:

- LDRs are made of semiconductor materials that absorb photons when light falls on them
- This absorption increases the energy levels of the electrons in the material
- The increased energy allows more electrons to move freely, reducing the material's resistance to electric current
- The brighter the light, the lower the resistance

Applications:

- LDRs are used in automatic lighting control, ambient light detection, and brightness measurement
- They are used in consumer electronics, such as screens and displays
- They are used in automatic security lights

- They can be used in light-activated and dark-activated switching circuits

Advantages:

- LDRs are cost-effective
- LDRs are simple to use and integrate
- LDRs are compact and versatile
- LDRs can reduce energy waste in home appliances, outdoor lighting, and streetlights

Light Dependent Resistor Specifications:

The LDR specifications mainly include maximum power dissipation, maximum operating voltage, peak wavelength, dark resistance, etc. The values of these specifications mentioned below.

Light Dependent Resistor Specifications:

The LDR specifications mainly include maximum power dissipation, maximum operating voltage, peak wavelength, dark resistance, etc. The values of these specifications mentioned below.

- Maximum power dissipation is 200mW
- The maximum voltage at 0 lux is 200V
- The peak wavelength is 600nm
- Minimum resistance at 10lux is 1.8kΩ
- Maximum. resistance at 10lux is 4.5kΩ
- Typical resistance at100lux is 0.7kΩ
- Dark resistance after 1 sec is 0.03MΩ
- Dark resistance after 5 sec is 0.25MΩ

1.5] LED:

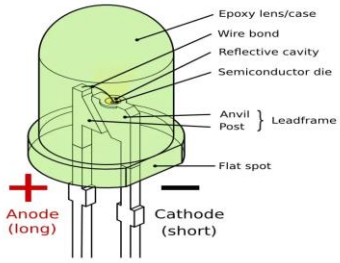


Fig 4.4 LED

LED stands for light-emitting diode, a semiconductor device that produces light when an electric current passes through it. LEDs are more energy efficient than other light sources and are used in many applications, including lighting, computing, and signal processing.

How LEDs work:

- Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.
- The color of the light is determined by the energy required for electrons to cross the band gap of the semiconductor.
- LEDs are encapsulated with a transparent cover so that emitted light can come out.

LED applications:

- LEDs are used in lighting, computing, and signal processing.
- LEDs are used in night lights, flashlights, path lights, task lights, and exit signs.
- LEDs are used in automotive, aeronautic, and water-based vehicles.
- LEDs are used in appliances for dimming of lights and reduce energy consumption.

LED history:

- The first LED was created by Nick Holonyak in 1962.
- IBM implemented LEDs on a circuit board for the first time on a computer in 1964.
- HP (Hewlett Packard) started using LEDs in calculators in 1968.

Features:

- Energy efficiency

- Dimming capability
- Longer led life span
- Viewing angles
- Display quality
- Features of led computer displays
- Leds operate very fast
- Temperature will affect led efficacy

1.6] MCP2551(CAN transceiver)

• Implements ISO-11898 standard physical layer requirements

• Supports 1 Mb/s operation

• Suitable for 12V and 24V systems

• Externally-controlled slope for reduced RFI emissions

• Detection of ground fault (permanent dominant) on TXD input

• Power-on reset and voltage brown-out protection

• An unpowered node or brown-out event will not disturb the CAN bus

• Low current standby operation

• Protection against damage due to short-circuit conditions (positive or negative battery voltage)

• Protection against high-voltage transients

• Automatic thermal shutdown protection

• Up to 112 nodes can be connected

• High noise immunity due to differential bus implementation

• Temperature ranges: - Industrial (I): -40°C to +85°C - Extended (E): -40°C to +125°C

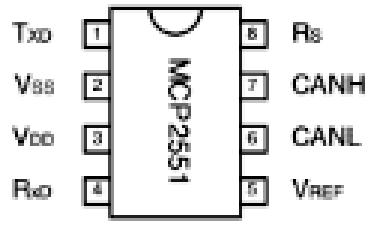


Figure 4.5 MCP2251 Module

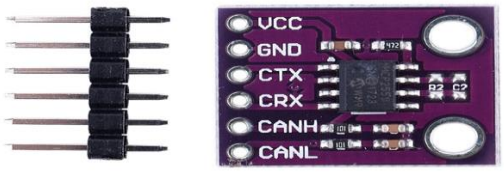


Fig 4.6 CAN module

2] Software Requirements

2.1] STM32CubeIDE

STM32CubeIDE is an all-in-one multi-OS development tool, which is part of the STM32Cube software ecosystem.

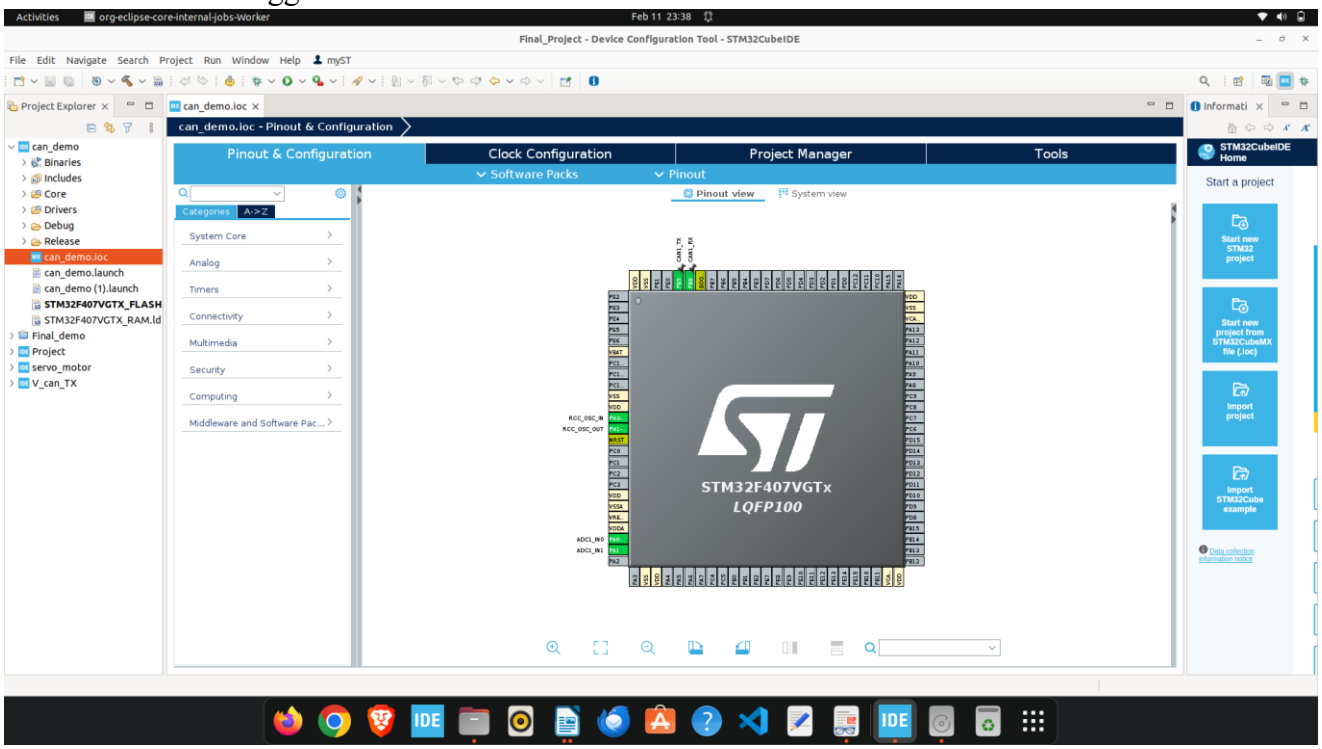
STM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors. It is based on the Eclipse®/CDT™ framework and GCC toolchain for the development, and GDB for the debugging. It allows the integration of the hundreds of existing plugins that complete the features of the Eclipse® IDE. STM32CubeIDE integrates STM32 configuration and project creation functionalities from STM32CubeMX to offer all-in-one tool experience and save installation and development time. After the selection of an empty STM32 MCU or MPU, or preconfigured microcontroller or microprocessor from the selection of a board or the selection of an example, the project is created and initialization code generated.

At any time during the development, the user can return to the initialization and configuration of the peripherals or middleware and regenerate the initialization code with no impact on the user code.

STM32CubeIDE includes build and stack analysers that provide the user with useful information about project status and memory requirements. STM32CubeIDE also includes standard and advanced debugging features including views of CPU core registers, memories, and peripheral registers, as well as live variable watch, Serial Wire Viewer interface, or fault analyser.

Features :

- Integration of services from STM32CubeMX:STM32 microcontroller, microprocessor, development platform and example project selection Pinout, clock, peripheral, and middleware configuration Project creation and generation of the initialization code Software and middleware completed with enhanced STM32Cube Expansion Packages.
- Based on Eclipse®/CDT™, with support for Eclipse® add-ons, GNU C/C++ for Arm® toolchain and GDB debugger



STM32MP1 Series: Support for Open ST Linux projects: LinuxSupport for Linux

fig 2.1. STM32 cube IDE interface

- Additional advanced debug features including:CPU core, peripheral register, and memory viewsLive variable watch viewSystem analysis and real-time tracing (SWV)CPU fault analysis tool RTOS-aware debug support including Azure.
- Support for ST-LINK (STMicroelectronics) and J-Link (SEGGER) debug probes
- Multi-OS support: Windows®, Linux®, and macOS®, 64-bit versions only

5] Project Architecture and Block Diagram

5.1] Block Diagram

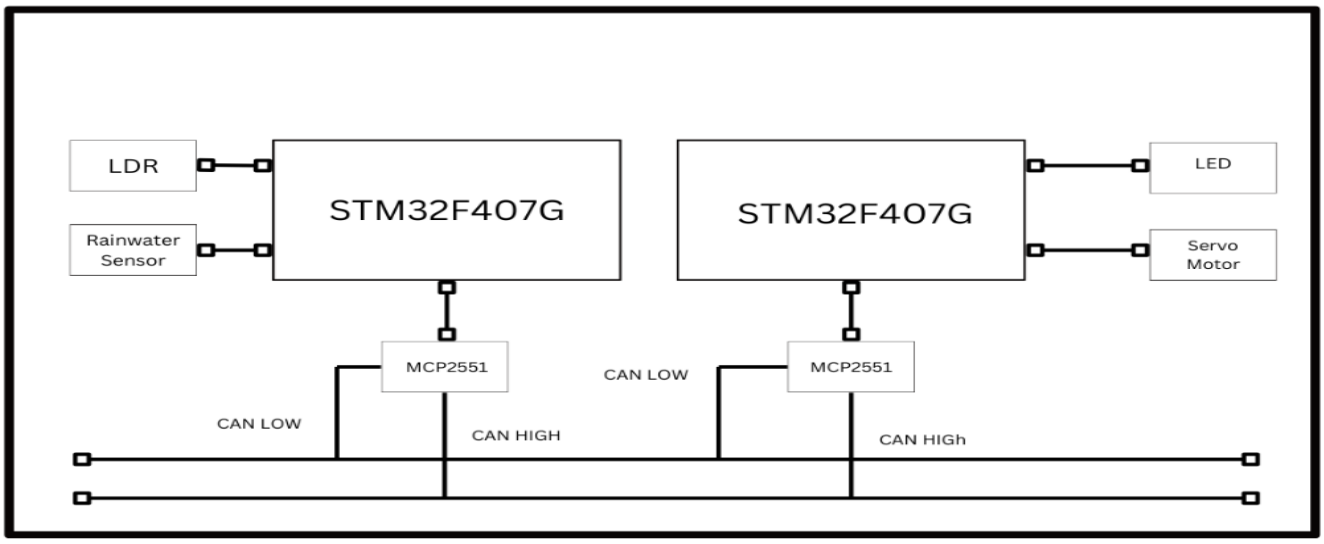


Fig 5.1 Block Diagram of Rain drop sensor and LDR with servo motor Using CAN

5.2] Description

As per block diagram here we are using two STM32F407 discovery board, one is used as a transmitter and another one is used as a receiver

 Rain Detection

- The rainwater droplet sensor detects rain and sends a signal to the microcontroller.
- The sensor's signal is processed and converted into a CAN data frame.

Darkness Detection

- The LDR sensor detects the surrounding light intensity and sends a corresponding analog signal to the microcontroller.
- The ADC within the microcontroller converts the analog signal into a digital signal and transmits it via the CAN protocol.

 CAN Communication

- The first node and second node (sensor node) transmits a rain detected message and led darkness message via the CAN bus.
- The second node (actuator node) receives the message and processes it.

 Servo Motor Activation

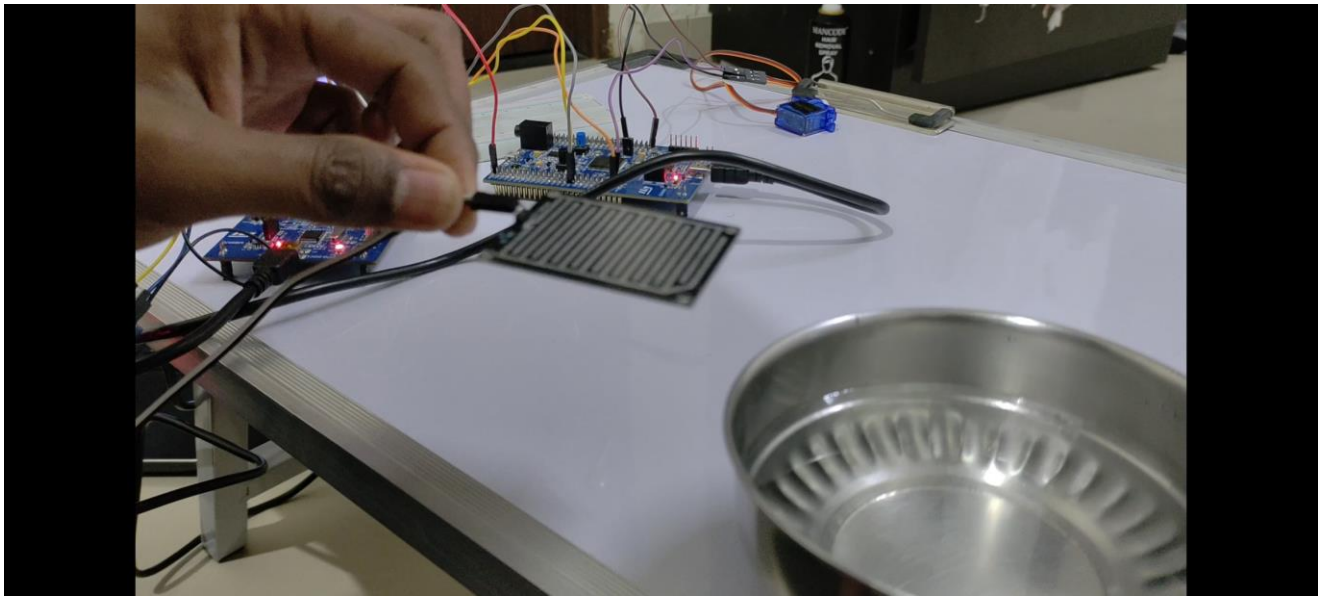
- If rain is detected, the second node instructs the servo motor to rotate to a predefined position
- If no rain is detected, the servo motor returns to its original position.

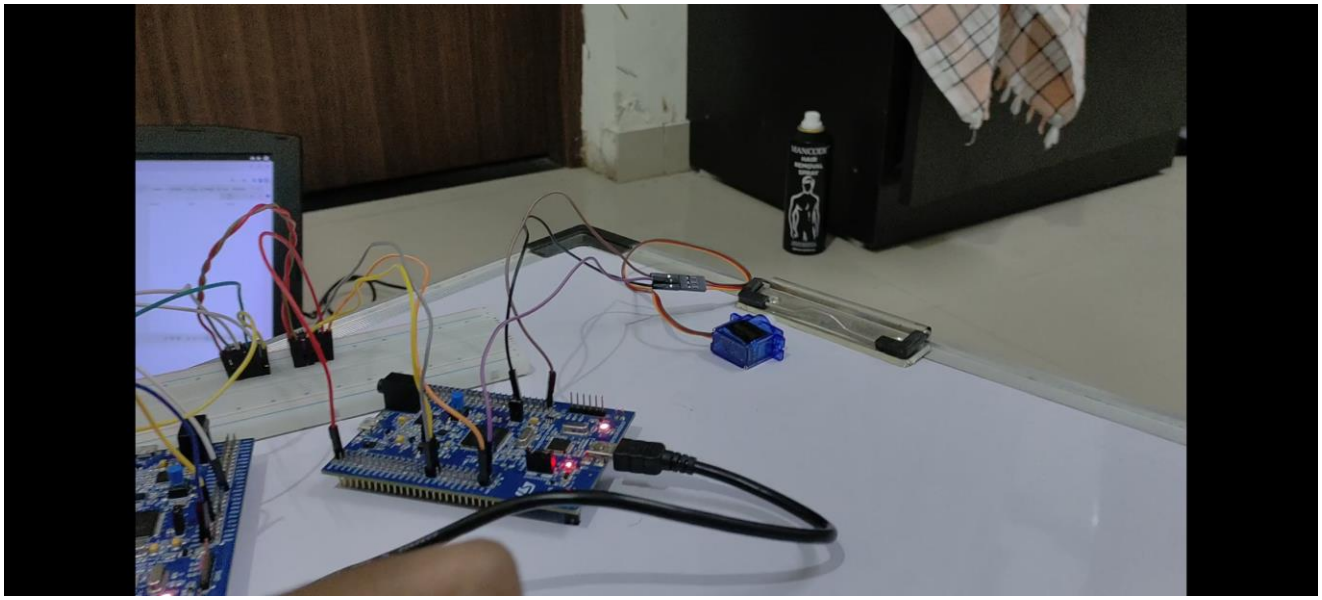
LED intensity control

- After reading the ldr value from CAN the led controls its intensity using PWM technology.

This system uses CAN protocol for communication between sensors and rotate motor and control light intensity at the other end.

6] Testing images





7] Conclusion

This project, **"Vehicle Automation Using CAN Communication Protocol,"** successfully demonstrates the automation of two essential vehicle parameters: **headlight control and wiper activation** using an **LDR sensor and a rainwater sensor**, respectively. By integrating these sensors with a **microcontroller featuring CAN communication**, the system ensures efficient and reliable data exchange between different vehicle nodes.

The **LDR sensor** enables automatic headlight control based on ambient light intensity, enhancing visibility and energy efficiency. Meanwhile, the **rainwater sensor** detects raindrops and triggers the wiper system, improving driver convenience and safety. The implementation of the **Controller Area**

Network (CAN) protocol ensures seamless and real-time communication between these components, making the system scalable for further vehicle automation.

Overall, this project enhances vehicle safety, comfort, and efficiency by leveraging sensor-based automation and robust CAN communication, paving the way for smarter and more autonomous automotive systems.

8] Future Scope

This project, **"Vehicle Automation Using CAN Communication Protocol,"** can be further enhanced and expanded in several ways to improve vehicle automation, safety, and efficiency:

1. **Integration with IoT and Cloud** – The system can be connected to IoT platforms to enable real-time monitoring, remote diagnostics, and data analysis for predictive maintenance.
2. **Advanced Sensor Fusion** – Additional sensors such as **humidity, temperature, and ultrasonic sensors** can be integrated to further enhance automation, such as defogger activation or obstacle detection.
3. **AI-Based Adaptive Control** – Implementing **machine learning algorithms** can improve decision-making by adapting headlight and wiper control based on historical weather conditions and driving patterns.
4. **Smart Vehicle Communication** – Expanding the **CAN network** to interact with other vehicle systems like **automatic braking, lane assist, and adaptive cruise control** can contribute to the development of semi-autonomous vehicles.
5. **Enhanced Power Efficiency** – Using **low-power components** and optimizing **power management strategies** can improve battery life and make the system more energy-efficient, especially for electric vehicles (EVs).
6. **Mobile App Integration** – A mobile application can be developed to allow users to manually control and monitor the system remotely, providing added convenience.
7. **Implementation in Autonomous Vehicles** – The system can serve as a foundation for **fully autonomous vehicles** by integrating with more advanced automation features such as self-driving navigation and smart traffic management.

By incorporating these advancements, this project can significantly contribute to the evolution of intelligent and autonomous transportation systems.

9] References

3.1.1 https://elearning.vector.com/vl_can_introduction_en.html

3.1.2 Renesas CAN pdf

3.1.3 Serial Bus System pdf

3.1.4 CAN primer

3.1.5 STM32F407 Discovery User Manual

3.1.6 https://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php

3.1.7 https://www.ijareeie.com/upload/june/56_Vehicle%20control.pdf

3.1.8 https://www.ijariit.com/manuscripts/v5i2/V5I2-1652.pdf

3.1.9 https://www.ijste.org/articles/IJSTEV7I12004.pdf