Group No: 4

Car Dashboard with Anti-Theft System

Submitted in partial fulfillment for the award of

Post Graduate Diploma in EMBEDDED SYSTEMS AND DESIGN

From C-DAC, ACTS (Pune)

Guided by:

Mr. Tarun Bharani

Presented by:

Gurditta Singh Creative	**240340130017**
Shubham Falke	**240340130044**
Shivangi Mishra	**240340130041**
Shobhna Kumari	**240340130042**
Jerin Joji	**240340130022**

Centre for Development of Advanced Computing (C-DAC), Pune

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Gurditta Singh Creative	**240340130017**
Shubham Falke	**240340130044**
Shivangi Mishra	**240340130041**
Shobhna Kumari	**240340130042**
Jerin Joji	**240340130022**

Have successfully completed their project on

Car Dashboard with Anti-Theft System

Under the guidance of
Mr. Tarun Bharani

Project Guide **Project Supervisor**

Acknowledgement

The success and final outcome of this project "**Car Dashboard with Anti-Theft System"** required a lot of guidance and assistance from many people and we are extremely privileged to have received this throughout the completion of our project. All that we have done is only due to such supervision and assistance, and we will always remember to thank them.

We are highly indebted to **Mr. Tarun Bharani** for his guidance and constant supervision, and for providing the necessary information regarding the project, as well as for his support in completing it.

We take this opportunity to thank the Head of the department, Mr. Gaur Sunder, for providing us with excellent infrastructure and an environment conductive to our overall development.

We express our sincere and deep gratitude to **Ms. Risha P.R. (Program Head)** and all the staff members for their cooperation many ways. We also thank **Ms. Namrata Ailawar** for her valuable guidance and constant support throughout this work. We would also like to express our gratitude to our parents and the members of CDAC ACTS, Pune for their kind co-operation and encouragement, which helped us in completing this project.

Our most heartfelt thanks go to Mrs. Srujana Bhamidi (Course coordinator, PG-DESD) for her valuable guidance and constant support throughout this work.

Our thanks and appreciations also go to our batchmates for their contribution to the development of the project and to everyone who willingly helped us with their abilities.

From:

Gurditta Singh Creative	**240340130017**
Shubham Falke	**240340130044**
Shivangi Mishra	**240340130041**
Shobhna Kumari	**240340130042**
Jerin Joji	**240340130022**

Abstract

The main purpose of the "Car Dashboard with Anti-Theft System" project is to enhance vehicle safety and driver convenience by integrating real-time monitoring and advanced security features into a single system. In modern vehicles, critical information such as engine temperature and proximity sensor readings for obstacle detection and collision avoidance are essential for safe driving. Traditionally, monitoring these parameters requires manual observation, which can be cumbersome and inefficient.

To address this, our project utilizes two STM32 Discovery boards to create a sophisticated car dashboard system. One STM32 board acts as the data acquisition node, collecting and processing vital data from temperature sensors and proximity sensors. This data is essential for providing real-time information on environmental conditions and potential obstacles. The second STM32 board interfaces with an OLED display to present this information clearly to the driver, ensuring enhanced situational awareness.

Communication between the two boards is managed via the Controller Area Network (CAN) protocol, known for its robustness and reliability in automotive applications. This ensures efficient and dependable data exchange between the nodes.

In addition to monitoring capabilities, the system incorporates an anti-theft feature using a fingerprint sensor. This security measure ensures that only authorized users can activate the dashboard and start the vehicle, thereby preventing unauthorized access and reducing the risk of theft.

Overall, the "Car Dashboard with Anti-Theft System" project integrates advanced monitoring and security technologies to improve vehicle safety and driver convenience, offering a comprehensive solution for modern automotive needs.

Contents

List of Figures

Chapter 1
INTRODUCTION

1.1 About Project:

The "Car Dashboard with Anti-Theft System" project enhances vehicle safety by integrating real-time monitoring and security features into a single system using two STM32 Discovery boards. The first board collects and processes data from temperature and proximity sensors, while the second board displays this information on a 1.5-inch SSD1306 OLED screen, ensuring clear visibility for the driver. Data exchange between the boards is managed via the robust Controller Area Network (CAN) protocol. Additionally, the system includes a fingerprint sensor for anti-theft protection, ensuring only authorized users can activate the dashboard and start the vehicle. This project combines advanced monitoring and security technologies to improve both vehicle safety and driver convenience.

1.2 Project scope:

The scope of the "Car Dashboard with Anti-Theft System" project involves creating an integrated vehicle safety and security solution. This includes designing a system with two STM32 Discovery boards: one for gathering and processing data from temperature and proximity sensors, and the other for displaying this information on a 1.5-inch SSD1306 OLED screen. The project will employ the Controller Area Network (CAN) protocol to facilitate reliable communication between the boards. A key feature of the system will be a fingerprint sensor with a button press mechanism, which ensures that only authorized users can access and start the vehicle. The project will also involve thorough testing to validate system performance, along with providing detailed documentation and support for installation and maintenance. The aim is to deliver an advanced, user-friendly dashboard that enhances vehicle safety and deters theft.

1.3 System Requirements

Hardware Requirements:

1. STM32F407 DISCOVERY BOARD
2. SN65HVD230 CAN Transceivers
3. DHT 11 Temperature Sensor
4. SSD1306 OLED DISPLAY
5. HW-201 IR Proximity Sensor

Software Requirements:

The software used for implementing the project was STM32CubeIDE 1.6.0 for STM32F4

Chapter 2
LITERATURE SURVEY

2.1 Design of Digital CAN Based Car Dashboard Unit

Presented at the Carpathian Control Conference in May 2019, this paper by Tomáš Pawlenka, Jiří Kulhánek, Petr Tomčík, and Rostislav Zapletal from VŠB-Technical University of Ostrava discusses the development and implementation of a digital dashboard for electric vehicles. The study focuses on integrating Controller Area Network (CAN) communication with a graphical LCD display system to enhance vehicle diagnostics and user interaction. Traditional analog indicators are being replaced with digital displays to improve both functionality and aesthetics. The system features two 7-inch LCD displays from 4D Systems connected via a serial interface to a PIC18F25K80 microcontroller, which manages CAN bus messages from the vehicle's control unit. The displays show critical information like speed and vehicle status, designed for high contrast and readability even in sunlight. The microcontroller's firmware was developed using MikroC, and the CAN bus was optimized using the CANCULATOR tool. The paper also details PCB design for the CAN transceiver, focusing on power supply, connectivity, and programmability. Graphical design considerations ensure legibility and adherence to automotive standards. The final prototype was successfully tested in a model of the SCE electric car, demonstrating a cost-effective solution for digital dashboards that enhance functionality and user experience.

2.2 Vehicle Anti-Theft System Based on an Embedded Platform

This paper provides an overview of various vehicle anti-theft systems, categorizing them into theft detection/prevention and owner alerting functions. It reviews systems that use GSM messages for remote vehicle shutdown and location tracking, and those with auto tamper detection features. Modern systems often include immobilization to enhance security, with some utilizing mechanical keys and keypads, and others employing facial recognition technology. The paper discusses fingerprint recognition as a cost-effective and efficient method for vehicle security.

2.3 Communication of Cluster Using CAN Bus

Authored by Matej Kubis, Miroslav Gutten, Daniel Korenciak, and Matus Danko from the University of Zilina, this paper explores communication between car infotainment systems and drive components via the CAN bus. It covers automotive communication buses' features, benefits, and applications, with a focus on developing a LabVIEW-based dashboard that replicates a Volkswagen Golf 5 dashboard. The CAN bus transmits data between control units, such as vehicle speed, to the instrument panel. The paper details the CAN bus's dominant and recessive states, Basic CAN, and Full CAN protocols. It also discusses different CAN bus variants like HSCAN and LSCAN, their connections, and ensuring seamless communication across automotive systems. Practical implementations were tested using tools like the Pico PC Oscilloscope and Kvaser Leaf Semipro to simulate and analyze data transmission for functions such as speedometers and engine temperature indicators.

2.4 Designing Configurable Automotive Dashboards on Liquid Crystal Displays

The paper highlights the importance of the instrument cluster in vehicle passive safety by displaying various signals and warnings. With advanced driving assistance systems like frontal collision warnings and adaptive cruise control, the need for effective visual communication has grown. The ACTIVE project developed software-programmable dashboards using liquid crystal displays (LCDs) to address the challenge of limited display space. These dashboards are designed for flexible customization and real-time configurability, allowing adjustments to the display's number, layout, and appearance based on driving conditions. This flexibility optimizes visual space and integrates information from current and future systems. The paper describes the design process and evaluates the user impact through laboratory and road tests conducted by Robert Bosch GmbH in Germany, showcasing the flexible interface's improvements in user experience and safety.

2.5 Anti-Theft Security System for Vehicles

Advancements in anti-theft security systems for vehicles have integrated technologies such as GPS, GSM, IoT, and biometrics. Key research focuses on real-time location tracking, remote control capabilities, and advanced biometric methods for preventing unauthorized access. Studies highlight the use of GPS and GSM for tracking and alerting, biometric sensors for enhanced security, and microcontrollers like ARM7 for system control. These technologies enable real-time monitoring, alerting vehicle owners, and remote vehicle immobilization. Recent trends include IoT for better connectivity, advanced communication technologies for instant alerts, and the use of machine learning and AI for predictive analytics and anomaly detection. Future developments are expected to enhance AI algorithms, biometric authentication, and smart city infrastructure integration, advancing vehicle anti-theft measures.

Chapter 3
SYSTEM DESIGN

3.1 Implementation

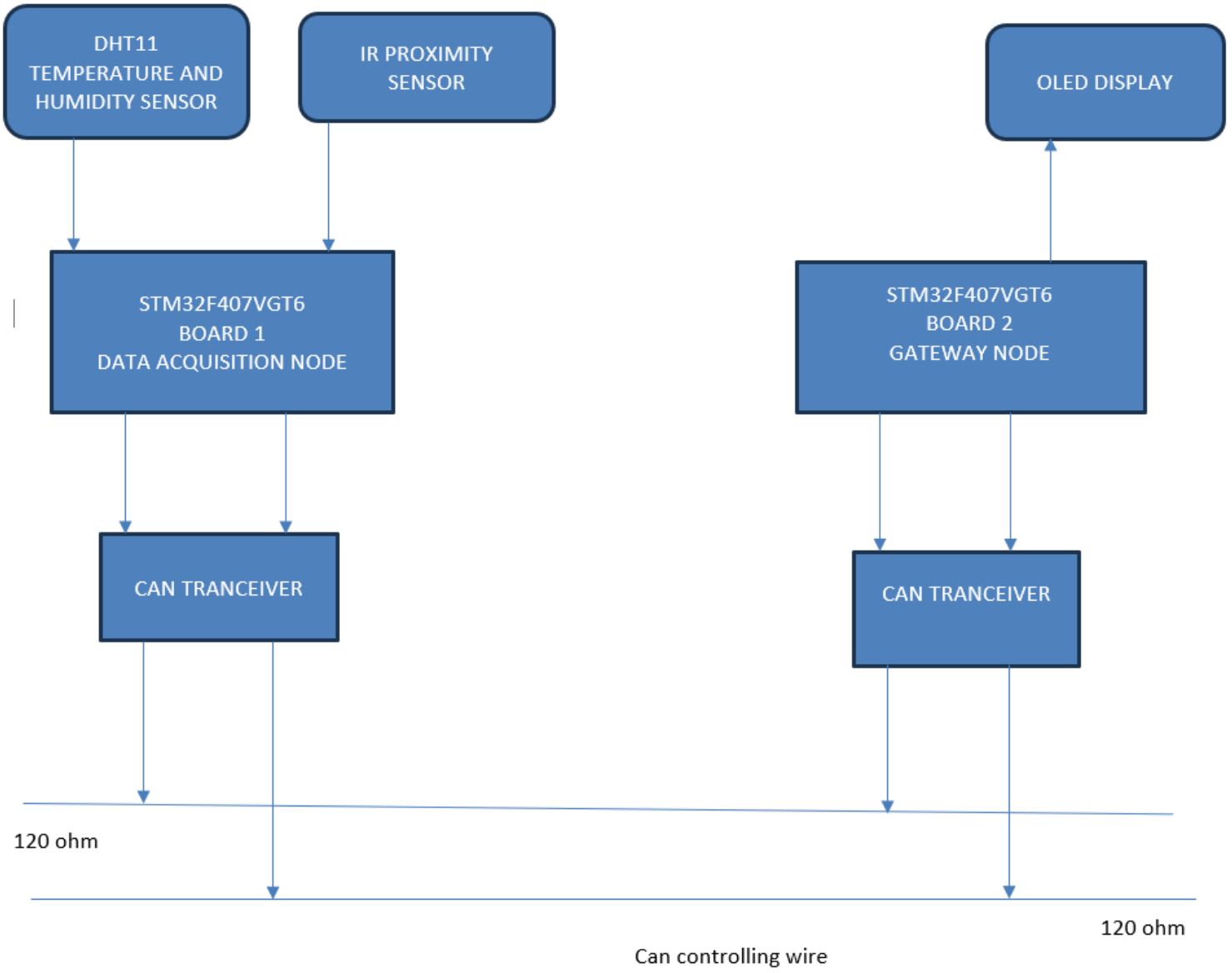


Fig3.1: Block Diagram

Block Diagram Description:

The data captured by the sensor is transferred to STM32 board through the STM32 board using CAN BUS and that is transferred further to the AWS cloud platform.

3.2: Flow Chart

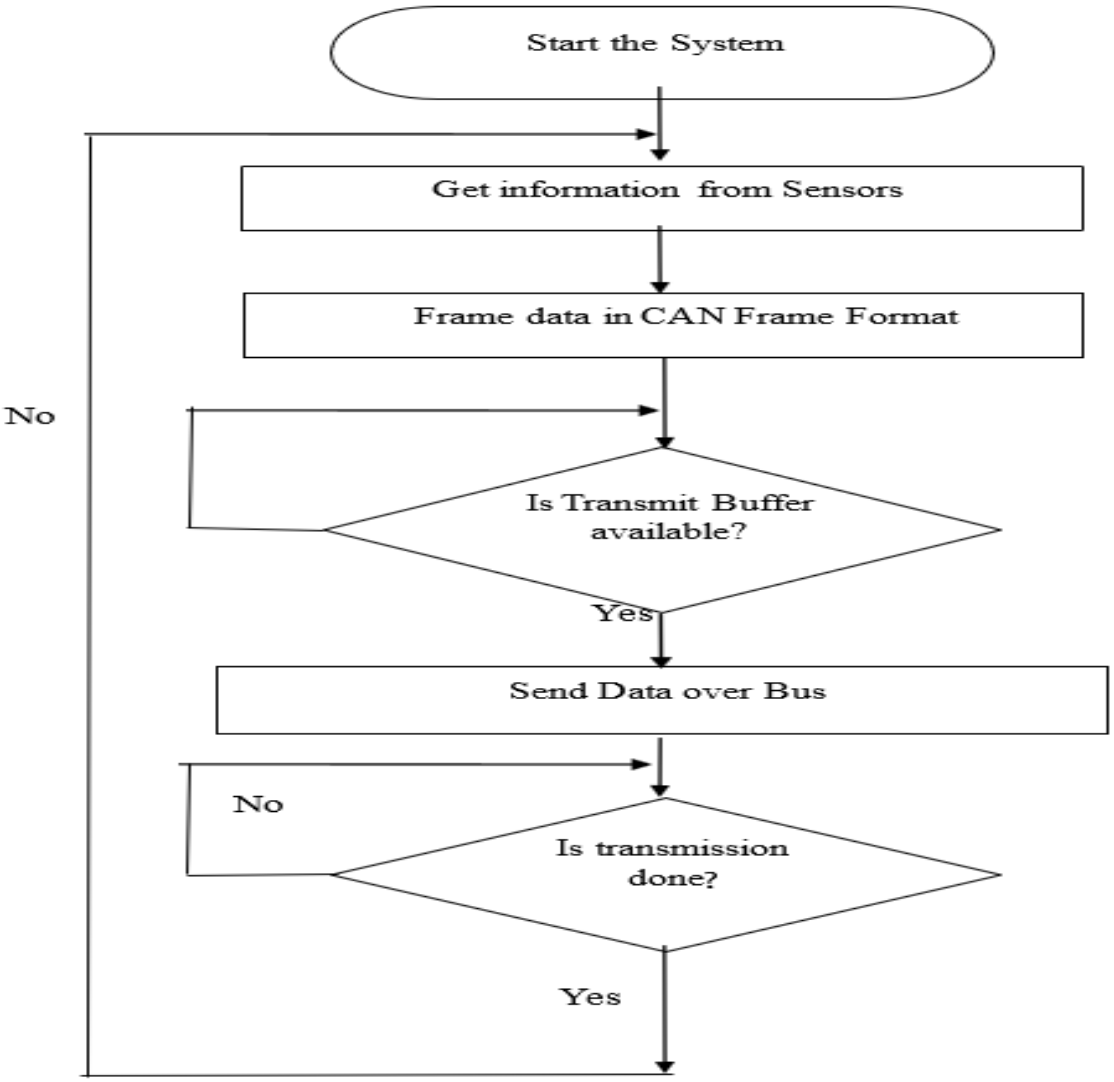


Fig3.2: Flow Charts

3.3 Hardware Components

3.3.1 STM32F407VGT6

The STM32F407VGt6 microcontroller is a high-performance ARM Cortex-M4-based microcontroller unit (MCU) manufactured by STMicroelectronics. It offers a wide range of features and peripherals suitable for various embedded applications, including industrial control systems, consumer electronics, and IoT devices.

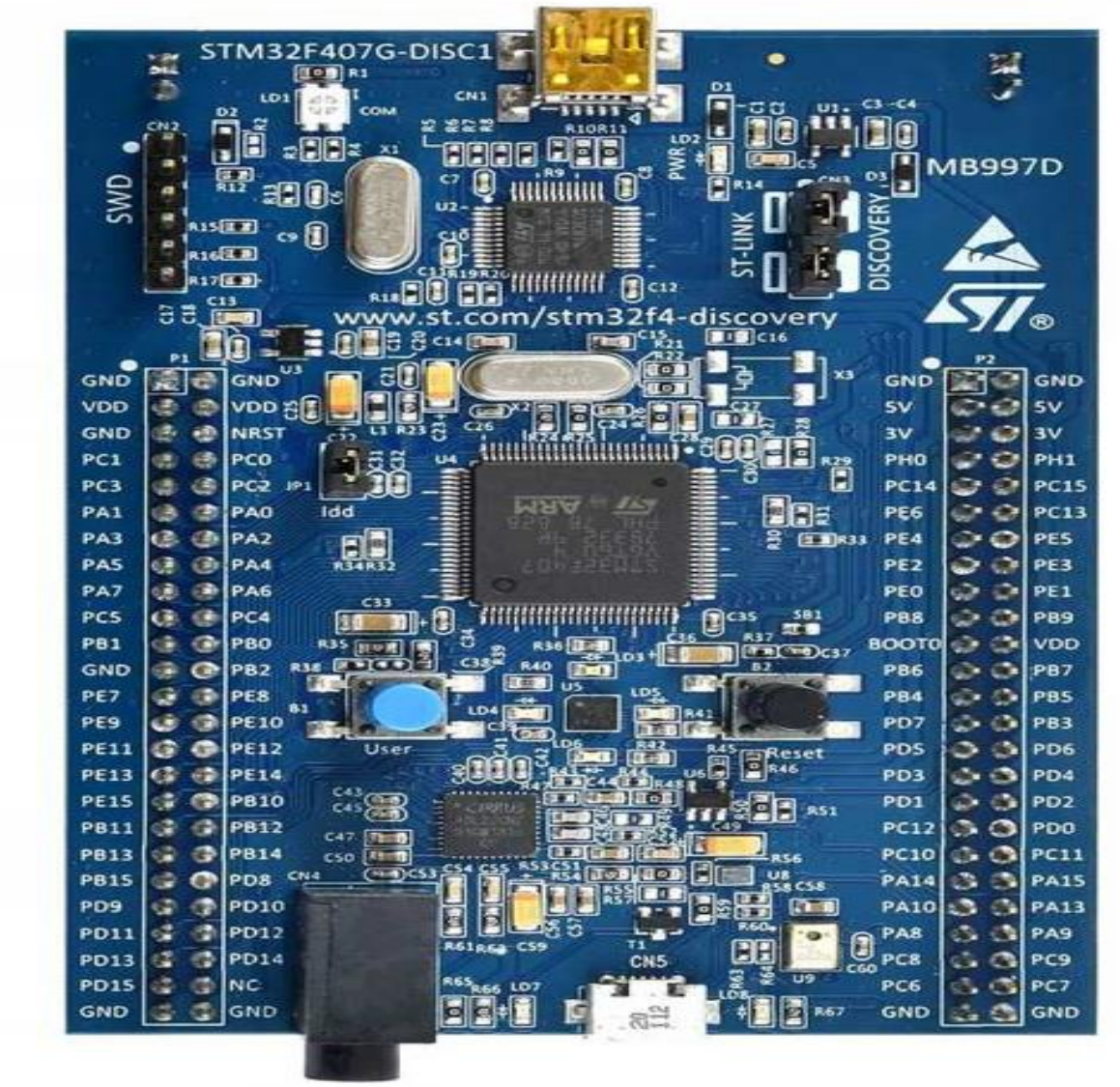


Fig 3.3.1: STM32F4

- The STM32F407VGt6 MCU serves as the central processing unit for the system, handling data acquisition, processing, and communication tasks.
- It features a powerful ARM Cortex-M4 core running at up to 168 MHz, providing sufficient processing power for real-time sensor data processing and control algorithms.
- Peripherals and Interfaces:
- The STM32F407VGt6 MCU is equipped with a rich set of peripherals and interfaces, including multiple UART, SPI, I2C, and CAN interfaces.
- These interfaces facilitate communication with external sensors, modules, and devices, enabling seamless integration into the system architecture.
- Data Acquisition:
- The STM32F407VGt6 MCU interfaces with various sensors, such as temperature sensors, proximity sensor to collect real time data.
- It employs dedicated ADC (Analog-to-Digital Converter) channels to digitize analog sensor readings, ensuring

accurate and reliable data acquisition.

- The MCU supports various communication protocols, such as UART, SPI, and I2C, for serial communication with peripheral devices.
- It facilitates communication with external modules, including the OLED display, and others, enabling seamless data transmission and control.
- Firmware development for the STM32F407VGt6 MCU is carried out using integrated development environments (IDEs) such as STM32CubeIDE or Keil µVision.

3.3.2 HW-201 IR Infrared Obstacle Avoidance Sensor Module

The IR Infrared Obstacle Avoidance Sensor Module detects obstacles using infrared light. It includes an infrared emitter and receiver pair. The emitter sends out infrared light, which reflects off obstacles and is captured by the receiver. The sensor triggers a digital output signal when an obstacle is detected, and the red LED illuminates. The detection range is adjustable from 2 to 30 cm via a potentiometer. Operating at 3.3V to 5V, it interfaces with an STM32F407 Discovery board and is used for object detection and collision avoidance.

Specifications

1. **Detection**:
    - **Range**: 2 ~ 30 cm
    - **Angle**: 35°
    - **Adjustment**: Detection distance adjustable via potentiometer
    - **Type**: Active infrared reflection; effectiveness varies with target reflectivity and size
2. **Output**:
    - **Indicator**: Red LED lights up when an obstacle is detected
    - **Signal**: Digital output (low when obstacle is detected)
    - **Connection**:
        - VCC to VCC
        - GND to GND
        - OUT to STM32F407 Discovery board GPIO PA0
3. **Components**:
    - **Comparator**: LM393 for stable operation
    - **Power Supply**: 3-5V DC, with a red LED indicating power
    - **Board Size**: 3.2 cm x 1.4 cm
    - **Mounting**: 3mm screw holes for secure installation
4. **Module Interface**:
    - **VCC**: 3.3V-5V (connect to STM32F407 power supply)
    - **GND**: Ground
    - **OUT**: Digital output (0 or 1) to STM32F407 GPIO PA0

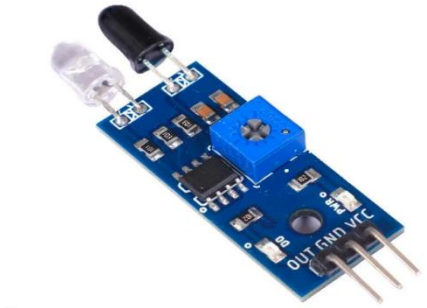


Fig 3.3.2: IR Proximity Sensor

3.3.3 DHT11 Sensor

The DHT11 is a digital sensor designed for measuring relative humidity and temperature with good precision and stability. Manufactured by Aosong Electronics Co., Ltd., it integrates a polymer humidity resistor and an 8-bit single-chip computer to ensure reliable performance. The sensor outputs a calibrated digital signal, providing both temperature and humidity data in a compact, four-pin package that supports long transmission distances up to 100 meters and consumes low power.

Key Features:

- Accuracy: ±5% RH for humidity and ±2°C for temperature
- Resolution: 1% RH for humidity and 1°C for temperature
- Operating Range: Humidity from 20% to 90% RH and temperature from 0°C to 50°C
- Digital Output: Uses Aosong 1-wire bus for communication
- Stability: Outstanding long-term stability with ±1% RH per year

Applications: The DHT11 is suitable for various applications where precise humidity and temperature measurement is needed, especially in environments where long transmission distances and low power consumption are critical.

Technical Specifications:

- Power Supply: 3.3V to 5.5V DC
- Power Consumption: 1.5 mA during measurement, 50 µA in standby
- Communication: 40-bit data transmitted via Aosong 1-wire bus

Considerations:

- Ensure stable power supply and avoid exposure to harsh chemicals or extreme environmental conditions to maintain accuracy and prolong sensor life.
- Avoid using the DHT11 in safety-critical applications where sensor failure could result in personal injury.

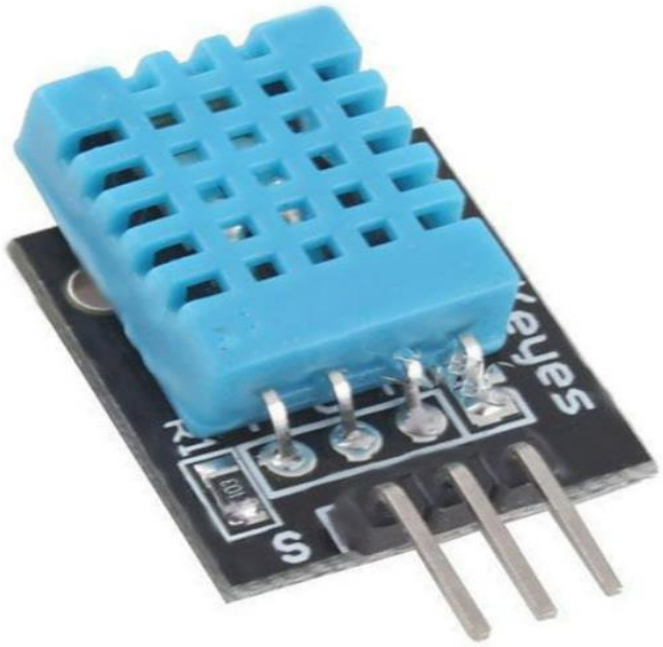


Fig 3.3.3: DHT 11 Temperature Sensor

3.3.4 SN65HVD230 CAN Transceivers

The SN65HVD230is a high-speed CAN, fault-tolerant device that serves as the interface between a CAN protocol controller and the physical bus. The SN65HVD230device provides differential transmit and receive capability for the CAN protocol controller and is fully compatible with the ISO-11898 standard, including 24V requirements. It will operate at speeds of up to 1 Mb/s. Typically, each node in a CAN system must have a device to convert the digital signals generated by a CAN controller to signals suitable for transmission over the bus cabling (differential output). It also provides a buffer between the CAN controller and the high-voltage spikes that can be generated on the CAN bus by outside sources. Some of its features are:

- Supports 1 Mb/s operation
- Implements ISO-11898 standard physical layer requirements
- Suitable for 12V and 24V systems
- 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)
- Up to 112 nodes can be connected
- High-noise immunity due to differential bus implementation
- Temperature ranges: - Industrial (I): -400C to +850C - Extended (E): -400C to +1250C

Fig 3.3.4: CAN Transceiver

CAN Protocol in STM32

The Basic Can protocol in STM32. Here we will see, how to communicate between two STM32 boards using the CAN protocol.

CAN (Controlled Area Network) Protocol is a way of communication between different devices but under certain rules. These rules must be followed when a message is transmitted over the CAN bus. Shown below is the Standard CAN Frame

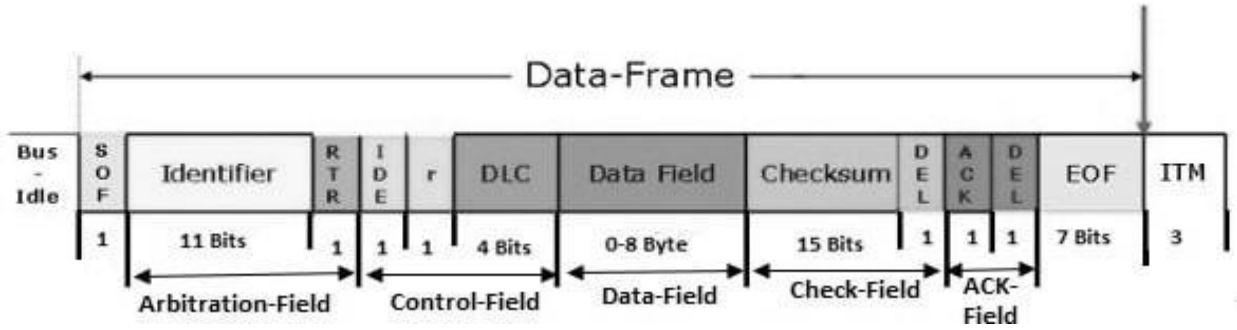


Fig 3.3.5: CAN Data Frame

The CRC and ACK will be handled by the HAL Library.

Here, Identifier is the ID of the transmitting Device

RTR (Remote Transmission Request) Specifies if the data is Remote frame or Data frame

IDE specifies if we are using Standard ID or Extended ID

r is the Reserved bit

DLC specifies the data length in Bytes

Data Field is where we can send the data, which should be up to 8 bytes

Checksum and DEL are the CRC data and it's Delimiter

ACK and DEL is the acknowledgment bit and its Delimiter

Connection between CAN transceiver and STM32f3 board

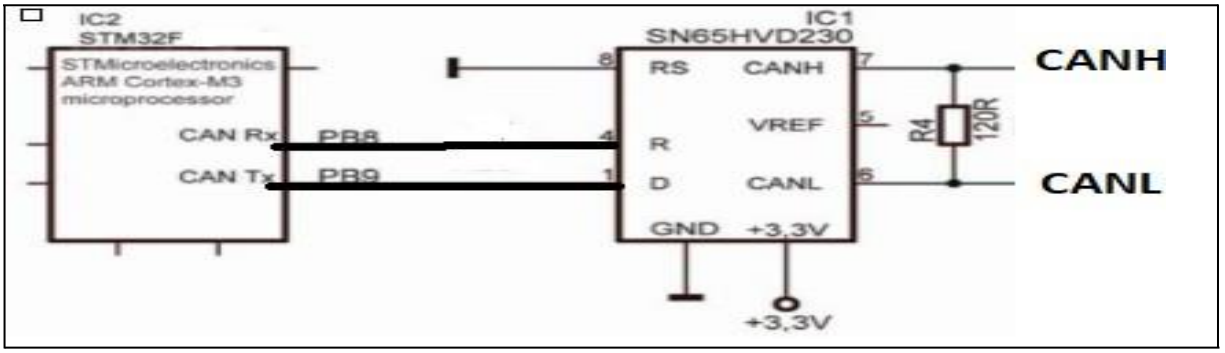


Fig 3.3.6 Connection between STM32f3 board and CAN transceiver

Here the Tx and Rx from the Transceivers are connected to PB9 and PB10 of the Respective controllers CANH and CANL are connected to each other Here the BAUD RATE is set to 62500 bps. The Operating Mode is NORMAL Mode.

3.3.5 OLED Display 1.5 Inch SSD1306

The OLED Display 1.5 Inch SSD1306 is a compact, high-resolution display module designed for various electronic projects. Utilizing the SSD1306 driver IC, this display features a 128x128 pixel resolution with a vibrant, high-contrast OLED panel that ensures clear visibility in diverse lighting conditions. The display operates via I2C or SPI communication, making it versatile and easy to integrate into different systems.

Key Features:

- **Display Type:** Monochrome OLED
- **Resolution:** 128x128 pixels
- **Size:** 1.5 inches diagonal
- **Driver IC:** SSD1306
- **Communication Interface:** I2C or SPI
- **Brightness:** High contrast with deep blacks and bright whites for excellent readability

Applications:

The 1.5 Inch SSD1306 OLED display is ideal for use in handheld devices, digital instruments, and embedded systems where compact size and high visibility are essential. Its ability to render clear and crisp graphics and text makes it suitable for user interfaces, real-time data displays, and status indicators.

Technical Specifications:

- **Operating Voltage:** Typically 3.3V to 5V DC
- **Current Consumption:** Low power consumption, approximately 20 mA during active display
- **Contrast Ratio:** High contrast ratio for enhanced readability

Considerations:

- Ensure proper power supply to avoid display issues.
- Use appropriate communication protocols (I2C or SPI) as per your project requirements to ensure compatibility and optimal performance.

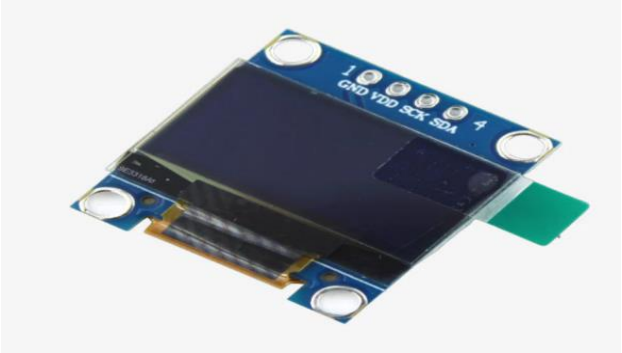


Fig 3.3.7 SSD1306 OLED Display

3.4 Software

3.4.1 STM32CubeIDE

STM32CubeIDE, an integrated development environment (IDE) crafted by STMicroelectronics, serves as a pivotal tool in the project's software development journey targeting STM32 microcontrollers. This environment provides a holistic platform with an array of tools and features aimed at expediting firmware development. Its seamless integration with the STM32CubeMX configuration tool streamlines the process of configuring STM32 peripherals, pin assignments, and middleware components. By visually configuring the MCU's parameters through STM32CubeMX, developers can swiftly generate initialization code, significantly reducing the workload associated with peripheral setup. Furthermore, STM32CubeIDE offers robust project management capabilities, facilitating efficient organization of source files, libraries, and resources within projects. With built-in debugging and testing functionalities, including support for hardware debugging using ST-LINK or JTAG/SWD debug probes, developers can effectively debug firmware code using features like breakpoints and watchpoints.

Additionally, STM32CubeIDE comes bundled with the GNU Arm Embedded Toolchain, providing a robust compiler and toolchain optimized for ARM Cortex-M-based microcontrollers. This toolchain supports advanced compiler optimizations and debugging features, enhancing code efficiency and reliability. Integrated seamlessly with STM32Cube middleware components and software libraries, the IDE enables developers to easily incorporate middleware functionalities into their projects, further accelerating the development process. Through STM32CubeIDE's comprehensive suite of features, developers can efficiently develop, debug, and deploy firmware for STM32 microcontroller-based projects, including the envisioned wireless data transmission system.

Chapter 4

Results

- **Enhanced Security:** The integration of the fingerprint sensor effectively prevented unauthorized access, adding a layer of security to the vehicle.
- **Improved Safety:** The proximity sensor provided real-time distance measurements, contributing to better safety and awareness for the driver.
- **Real-Time Monitoring:** The OLED display offered clear and immediate feedback on the vehicle's status, improving user interaction and system usability.
- **System Reliability:** The robust CAN bus communication and efficient peripheral management ensured a reliable and stable system performance.

Impact: The project successfully demonstrated the potential of integrating advanced sensors and communication protocols in a vehicle dashboard system. It provided valuable insights into sensor integration, real-time data handling, and user interface design, contributing to enhanced vehicle security and driver safety.

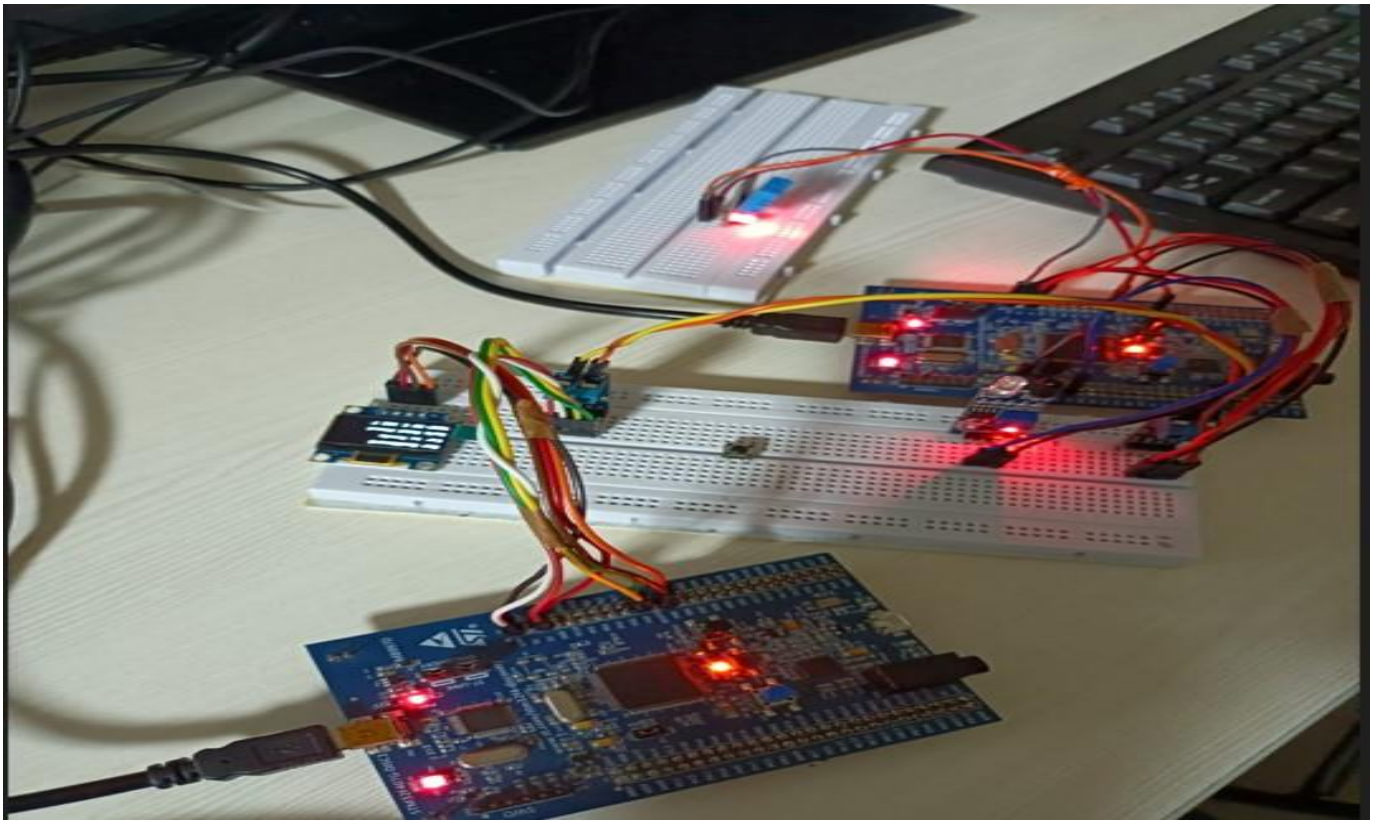


Fig 4.1.: Practical Hardware Connection

Fig 4.2: Transmitting IOC Settings

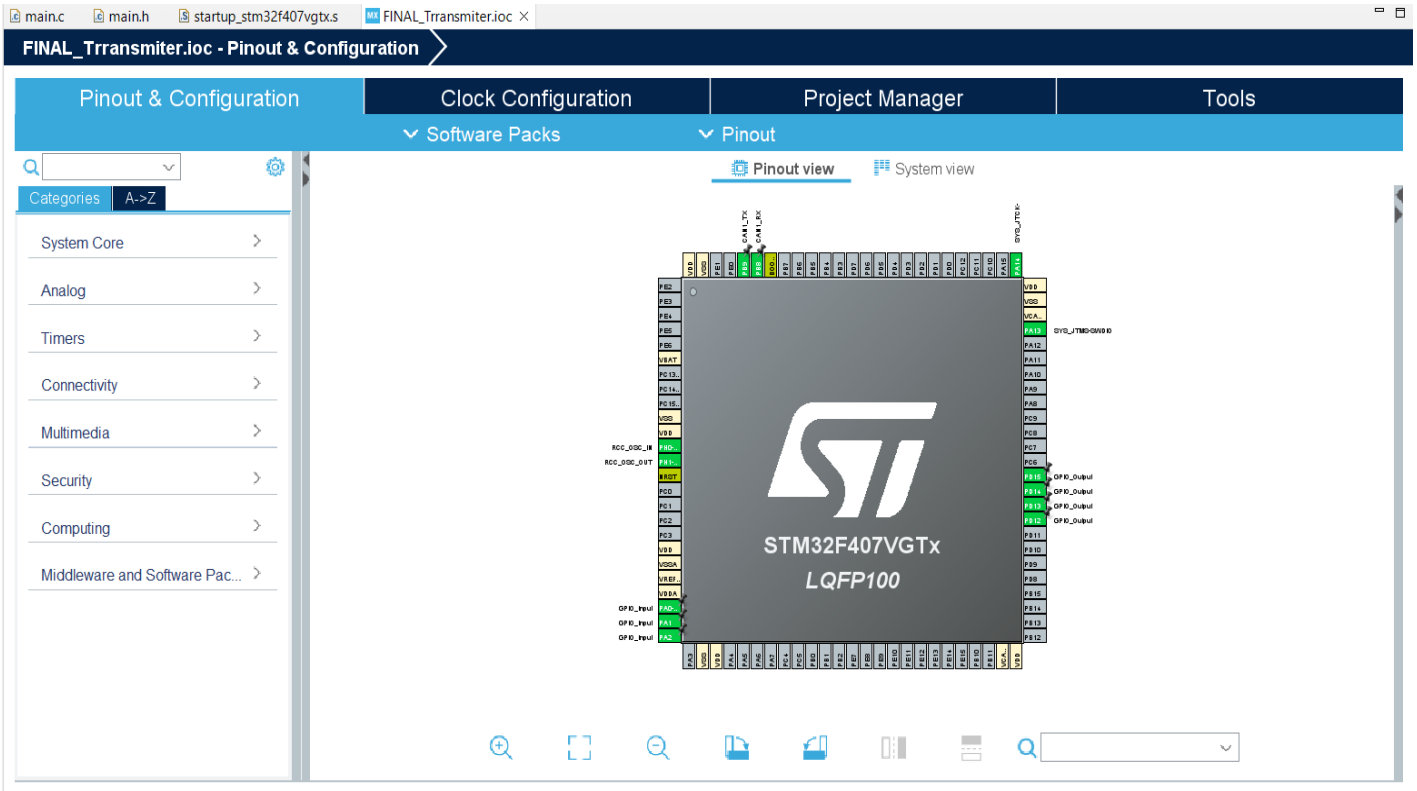
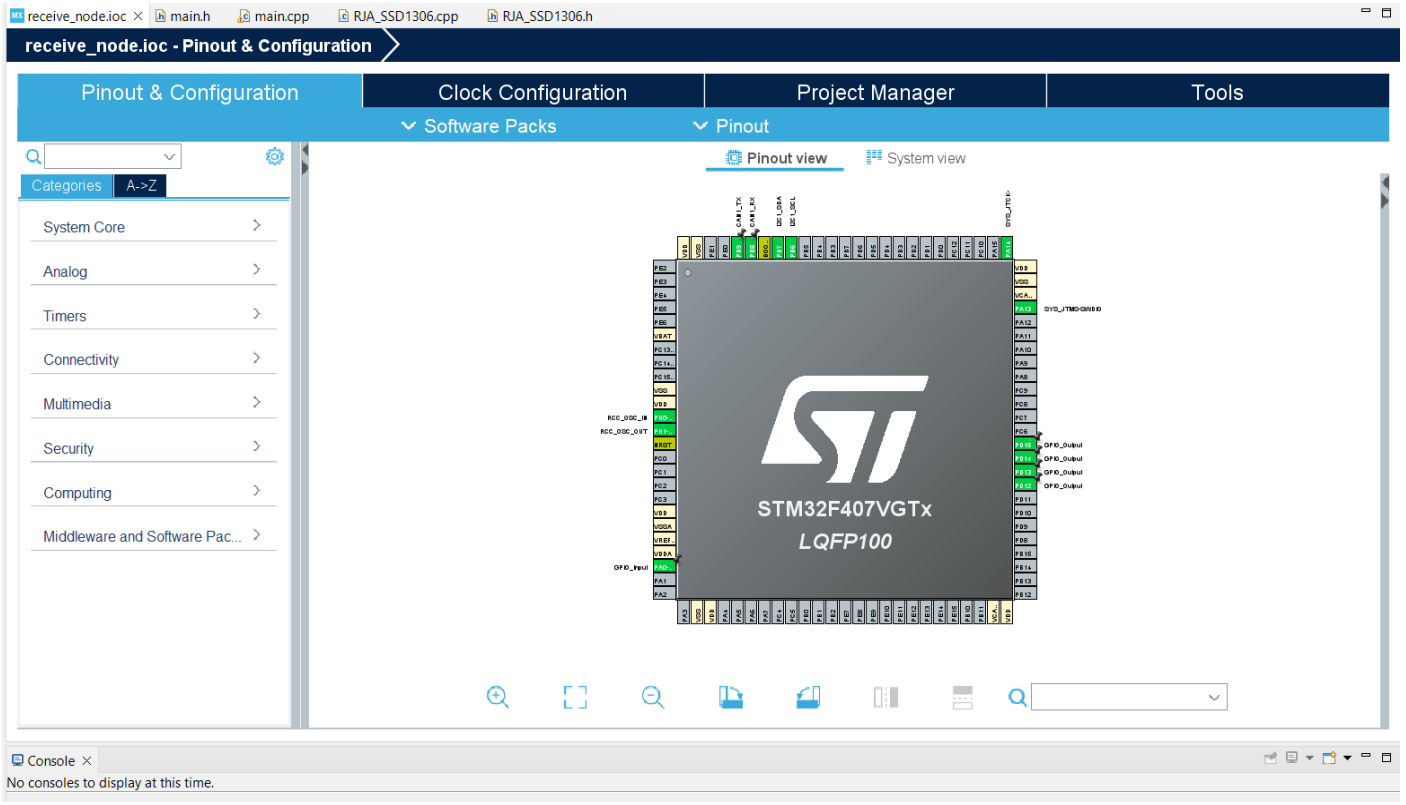


Fig4.3: Receive IOC Settings



Code: Transmitting node

```
/* USER CODE BEGIN Header */

/**
  ******************************************************************************
  * @file        : main.c
  * @brief       : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file * in the root directory of this software
    component.
  * If no LICENSE file comes with this software, it is provided AS-IS.

   Clock configuration for APB1 CLOCK-> 24MHZ
Activate Sys,RCC,AND CAN1
In Can1> SET PRECSCALER 24
BIT SEGMENT 1> 13
BIT SEGMENT 2> 2
Sjw> 1
Activate Can interrup from Nvic parameters TX for trasmitter node and RX for receiver node
Activae all the GPIOS leds GPIO OUTPUT
Activate for PA0 FOR GPIO INPUT FOR fingerpring sensor
temperature sensor is not added in code but shared random data
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
```

```
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
CAN_HandleTypeDef hcan1;

TIM_HandleTypeDef htim1;

/* USER CODE BEGIN PV */
uint8_t proximity_sensor = 0; uint8_t
fingerprint = 0;

uint8_t RHI, RHD, TCI, TCD,
SUM; uint32_t pMillis, cMillis;
float Temperature = 0; float
tFahrenheit = 0; float RH = 0;
uint8_t TFI = 0; uint8_t TFD = 0;
/*RHI: Relative Humidity Integer
RHD: Relative Humidity Decimal
TCI: Temperature Celsius Integer
TCD: Temperature Celsius Decimal
SUM: Checksum
pMillis: Previous Milliseconds cMillis:
Current Milliseconds Temperature:
Temperature (in Celsius) tFahrenheit:
Temperature (in Fahrenheit) RH: Relative
Humidity
TFI: Temperature Fahrenheit Integer
TFD: Temperature Fahrenheit Decimal*/
CAN_TxHeaderTypeDef TxHeader; uint8_t
TxData[8]; // CAN payload uint32_t
TxMailbox; // Buffer for Tx messages
*/
CAN_TxHeaderTypeDef TxHeader; uint8_t
TxData[8]; // CAN payload uint32_t
TxMailbox; // Buffer for Tx messages /*
USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/ void SystemClock_Config(void); static void
MX_GPIO_Init(void); static void MX_CAN1_Init(void); static void MX_TIM1_Init(void); /* USER CODE BEGIN
PFP */

/* USER CODE END PFP */
```

```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
#define     DHT11_PORT     GPIOD
#define DHT11_PIN GPIO_PIN_7


void microDelay (uint16_t delay)
{
  __HAL_TIM_SET_COUNTER(&htim1,   0);   while
  (__HAL_TIM_GET_COUNTER(&htim1) < delay);
}

uint8_t DHT11_Start (void)
{
  uint8_t Response = 0;
  GPIO_InitTypeDef GPIO_InitStructPrivate = {0};
  GPIO_InitStructPrivate.Pin = DHT11_PIN;
  GPIO_InitStructPrivate.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStructPrivate.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStructPrivate.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(DHT11_PORT, &GPIO_InitStructPrivate); // set the pin as output
HAL_GPIO_WritePin (DHT11_PORT, DHT11_PIN, 0);// pull the pin low
HAL_Delay(20);    // wait for 20ms
HAL_GPIO_WritePin (DHT11_PORT, DHT11_PIN, 1);// pull the pin high
microDelay (30);      // wait for 30us
  GPIO_InitStructPrivate.Mode = GPIO_MODE_INPUT;
  GPIO_InitStructPrivate.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(DHT11_PORT,  &GPIO_InitStructPrivate);  //  set  the  pin  as  input
  microDelay (40);
  if (!(HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)))
  {
   microDelay   (80);   if   ((HAL_GPIO_ReadPin   (DHT11_PORT,
   DHT11_PIN))) Response = 1;
  }
  pMillis   =   HAL_GetTick();
  cMillis = HAL_GetTick();
  while ((HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)) && pMillis + 2 > cMillis)
  {
   cMillis = HAL_GetTick();
  }
  return Response;
}

uint8_t DHT11_Read (void)
{
  uint8_t a,b;
  for (a=0;a<8;a++)
  {
   pMillis   =   HAL_GetTick();
   cMillis = HAL_GetTick();
```

```
  while (!(HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)) && pMillis + 2 > cMillis)
  { // wait for the pin to go high cMillis
   = HAL_GetTick();
  }
  microDelay (40); // wait for 40 us if (!(HAL_GPIO_ReadPin
  (DHT11_PORT, DHT11_PIN))) // if the pin is low
   b&= ~(1<<(7-a));
  else

   b|= (1<<(7-a));
  pMillis = HAL_GetTick();
  cMillis = HAL_GetTick();
  while ((HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)) && pMillis + 2 > cMillis)
  { // wait for the pin to go low cMillis
   = HAL_GetTick();
  }
 }
 return b;
}

/* USER CODE END 0 */

/**
 *  @brief The application entry point.
 *  @retval int
  */
int
main(void)
{

 /* USER CODE BEGIN 1 */

 /* USER CODE END 1 */

 /* MCU Configuration--------------------------------------------------------*/

 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */ HAL_Init();
 /* USER CODE BEGIN Init */

 /* USER CODE END Init */

 /* Configure the system clock */
 SystemClock_Config();

 /* USER CODE BEGIN SysInit */

 /* USER CODE END SysInit */

 /* Initialize all configured peripherals */
 MX_GPIO_Init();
```

```
MX_CAN1_Init();
MX_TIM1_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start(&htim1);


if (HAL_CAN_Start(&hcan1) != HAL_OK)
{
 /* Start Error */
 Error_Handler();
}


TxHeader.StdId = 0x123;
TxHeader.RTR = CAN_RTR_DATA;
TxHeader.IDE = CAN_ID_STD;
TxHeader.DLC = 4; // Data length code, updated to 3 as we are sending 3 bytes TxHeader.TransmitGlobalTime
= DISABLE;

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
 // Update fingerprint status
 if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET)
 {
  HAL_Delay(50); // Debouncing Delay
  if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET)
  {
   fingerprint = !fingerprint; // Toggle fingerprint status
  }
 }
 // Update proximity sensor status
 uint8_t new_proximity_sensor = (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) == GPIO_PIN_SET) ?
 1 : 0; if (new_proximity_sensor != proximity_sensor)
 {
  proximity_sensor = new_proximity_sensor;
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, (proximity_sensor == 1) ? GPIO_PIN_SET :
GPIO_PIN_RESET); // Debug LED
 }
 if(DHT11_Start())
     {
      RHI = DHT11_Read(); // Relative humidity integral
      RHD = DHT11_Read(); // Relative humidity decimal
      TCI = DHT11_Read(); // Celsius integral
      TCD = DHT11_Read(); // Celsius decimal
      SUM = DHT11_Read(); // Check sum
      if (RHI + RHD + TCI + TCD == SUM)
```

```
        {
          // Can use RHI and TCI for any purposes if whole number only needed
          Temperature = 10*((float)TCI + (float)(TCD/10.0));
          tFahrenheit =( Temperature * 9/5 + 32);
          RH = (float)RHI + (float)(RHD/10.0);
        }
       }
       HAL_Delay(500);

   // Prepare CAN data
   TxData[0] = fingerprint;        // Fingerprint status
   TxData[1] = proximity_sensor;    // Proximity sensor status
   TxData[2] = Temperature;              // Additional data if needed
    TxData[3]  =  tFahrenheit;
   // Transmit CAN data

   if (HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox) != HAL_OK)
   {
    /* Transmission request Error */
    Error_Handler();
   }

   HAL_Delay(100); // Delay for better tuning
   /* USER CODE END WHILE */

   /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */                              void
SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;

  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
```

```
 RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
 RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
 RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
 RCC_OscInitStruct.PLL.PLLM = 8;
 RCC_OscInitStruct.PLL.PLLN = 72;
 RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
 RCC_OscInitStruct.PLL.PLLQ = 4;
 if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
 {
  Error_Handler();
 }

 /** Initializes the CPU, AHB and APB buses clocks
 */
 RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
 RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
 RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;


 RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
 RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
 {
  Error_Handler();
 }
}

/**
 * @brief CAN1 Initialization Function
 * @param None
 * @retval None
 */ static void
MX_CAN1_Init(void)
{

 /* USER CODE BEGIN CAN1_Init 0 */

 /* USER CODE END CAN1_Init 0 */

 /* USER CODE BEGIN CAN1_Init 1 */

 /* USER CODE END CAN1_Init 1 */
 hcan1.Instance = CAN1;
 hcan1.Init.Prescaler = 36; hcan1.Init.Mode =
 CAN_MODE_NORMAL;
 hcan1.Init.SyncJumpWidth =
 CAN_SJW_1TQ; hcan1.Init.TimeSeg1 =
```

```
CAN_BS1_13TQ;  hcan1.Init.TimeSeg2  =
CAN_BS2_2TQ;
hcan1.Init.TimeTriggeredMode = DISABLE;
hcan1.Init.AutoBusOff      =      DISABLE;
hcan1.Init.AutoWakeUp      =      DISABLE;
hcan1.Init.AutoRetransmission = DISABLE;
hcan1.Init.ReceiveFifoLocked  =  DISABLE;
hcan1.Init.TransmitFifoPriority = DISABLE;
if (HAL_CAN_Init(&hcan1) != HAL_OK)
{
 Error_Handler();
}
/* USER CODE BEGIN CAN1_Init 2 */

/* USER CODE END CAN1_Init 2 */

}

/**
*    @brief TIM1 Initialization Function
*    @param None
*    @retval None

 */          static          void
MX_TIM1_Init(void)
{

 /* USER CODE BEGIN TIM1_Init 0 */

 /* USER CODE END TIM1_Init 0 */

 TIM_ClockConfigTypeDef sClockSourceConfig = {0}; TIM_MasterConfigTypeDef sMasterConfig = {0};

 /* USER CODE BEGIN TIM1_Init 1 */

 /*  USER  CODE  END  TIM1_Init  1  */
 htim1.Instance = TIM1;
 htim1.Init.Prescaler = 71;
 htim1.Init.CounterMode        =        TIM_COUNTERMODE_UP;
 htim1.Init.Period = 65535;
 htim1.Init.ClockDivision            =            TIM_CLOCKDIVISION_DIV1;
 htim1.Init.RepetitionCounter = 0;
 htim1.Init.AutoReloadPreload                                    =
 TIM_AUTORELOAD_PRELOAD_DISABLE;                                 if
 (HAL_TIM_Base_Init(&htim1) != HAL_OK)
 {
  Error_Handler();
 }
 sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
```

```
 if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) !=
 HAL_OK)
 {
  Error_Handler();
 }
 sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
 sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE; if
 (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) !=
 HAL_OK)
 {
  Error_Handler();
 }
 /* USER CODE BEGIN TIM1_Init 2 */

 /* USER CODE END TIM1_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */ static void
MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
```