

**Project Report On**

Hot Swappable Machine Learning Model at the Edge



**Submitted in Partial Fulfillment for the award of**

**Post Graduate Diploma in Embedded Systems and Design**

**(PG-DESD)**

from

**C-DAC, ACTS (Pune)**

**Guided by:**

Mr. Rhugved Rane

|  |  |
| --- | --- |
| **Presented by:** | |
| **Mr. Albin Joy** | **PRN:250840130007** |
| **Mr. Tejas Desale** | **PRN:250840130014** |
| **Mr. Jacob Mathew** | **PRN:250840130020** |
| **Mr. Nidhi Wasule** | **PRN:250840130030** |
| **Mr. Jaydeep Patil** | **PRN:250840130036** |

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

This project was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We are very glad to mention the name of *Mr.Rhugved Rane* for their valuable guidance to work on this project. Their guidance and support helped me to overcome various obstacles and intricacies during the course of project work.

Our heartfelt thanks goes to *Ms. Jubera Khan* (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 ACTS, Pune.

**1. INTRODUCTION**

**2. LITERATURE SURVEY**

**3.** **SYSTEM DESIGN**

1. Methodology

2. Hardware Description

3. Software Modules Description

**4. METHOLOGY AND TECHNIQUES**

1. System Overview
2. Component List and Purpose
3. STM32F407VGT6 Pin Configuration
4. STM32F407VGT6 Clock Configuration
5. Communication Flow
6. Sensor Data Acquisition
7. Communication Protocol
8. **RESULTS**
9. **CONCLUSION**
10. **FUTURE SCOPE**
11. **REFERENCES**

**Abstract**

The rapid growth of Internet of Things (IoT) and edge computing, there is an increasing demand for intelligent embedded systems that can process data locally, adapt to changing requirements, and minimize dependence on cloud resources. This project, titled *Hot Swappable Machine Learning Model at the Edge*, focuses on designing and implementing an edge-based intelligent temperature monitoring system capable of real-time inference and dynamic model updates. The system uses a DHT11 temperature sensor interfaced with an STM32F407VGT6 microcontroller to acquire real-world temperature data, which is then processed locally using a deployed machine learning model to classify the temperature into cold, warm, or hot categories, enabling low-latency decision-making directly at the edge.

The STM32F407VGT6 acts as the primary processing unit, where the sensor data is pre-processed, normalized, and fed into an AI model generated using X-CUBE-AI for efficient execution on resource-constrained hardware. The classification results are displayed locally on an OLED display using I²C communication and simultaneously transmitted to a UART terminal for debugging and monitoring purposes. To enable cloud connectivity, the STM32 communicates with an ESP32 module via SPI communication, where the STM32 operates as the master and the ESP32 as the slave. The ESP32 further forwards the processed temperature classification data to the AWS cloud using AWS Lambda, enabling remote monitoring, data logging, and scalability for cloud-based analytics.

A key feature of this system is the implementation of hot swappability which allows the deployed models to get swapped whilst the system is running. This allows the system to be capable of deploying an improved machine leaning model when they become available. The proposed architecture demonstrates a practical and scalable solution for adaptive edge intelligence, making it suitable for real-world applications such as smart environments, industrial monitoring, and IoT-based predictive systems.

Keywords: Edge Computing, Hot Swappable Machine Learning, STM32F407VGT6, DHT11 Sensor, X-CUBE-AI, SPI Communication,

I2C,UART,ESP32,AWSLambda,IoT

# Chapter 1

# INTRODUCTION

The recent advances in embedded systems and real-time computing have enabled microcontroller-based platforms to perform complex data processing tasks that were previously limited to high-performance computers. Many conventional embedded applications still rely on centralized processing or static logic, which can limit responsiveness, adaptability, and scalability in dynamic environments. Processing data locally at the device level significantly reduces latency, improves reliability, and allows systems to operate efficiently even in the absence of continuous network connectivity. As a result, there is a growing shift toward intelligent edge-based systems capable of making autonomous decisions using on-device computation.

The integration of machine learning into embedded platforms has further enhanced the capabilities of edge systems by enabling real-time inference on sensor data. Lightweight and optimized frameworks now make it possible to deploy machine learning models on resource-constrained microcontrollers while maintaining acceptable performance and power efficiency. However, a critical challenge in embedded machine learning systems is the lack of flexibility after deployment, as updating or modifying trained models typically requires reprogramming the entire firmware or physical access to the device. This limitation becomes a significant drawback in long-term deployments where system behavior must adapt to changing conditions or improved models.

This project, titled *Hot Swappable Machine Learning Model at the Edge*, addresses this challenge by implementing a real-time temperature classification system using an STM32F407VGT6 microcontroller. A DHT11 temperature sensor is interfaced with the STM32 to acquire live environmental data, which is locally processed by a machine learning model generated using the X-CUBE-AI framework. The model classifies temperature values into cold, warm, or hot categories, and the results are displayed on an OLED screen through I²C communication and simultaneously monitored on a UART terminal. This design ensures low-latency inference and provides both user-level visualization and debugging support.

To enable remote data access and system-level updates, the STM32 communicates with an ESP32 module via SPI communication, where the STM32 operates as the master device. The ESP32 forwards the classified temperature data to the AWS cloud using AWS Lambda for remote monitoring and analysis. A key feature of this system is the implementation of hot swappability which allows the deployed models to get swapped whilst the system is running. This allows the system to be capable of deploying an improved machine leaning model when they become available.

# Chapter 2

# LITERATURE SURVEY

# Introduction

# A literature survey is a systematic and comprehensive review of previously published research work related to the proposed project. It involves collecting information from journals, research papers, technical articles, and conference publications in the domain of interest. The primary objective of conducting a literature survey is to understand existing technologies, methodologies, and system architectures relevant to the research problem. This process helps in identifying the strengths, limitations, and research gaps in existing systems.

# The literature survey also assists in selecting appropriate hardware, software, and communication protocols for system design. By analyzing previous research work, the proposed system can be designed more efficiently, avoiding known limitations and incorporating proven techniques. Therefore, conducting a literature survey is an essential step in gathering secondary data and defining the architecture and methodology of the project.

# Research Papers:

# [A] PAPER 1: Embedded AI Using STM32 and X-CUBE-AI

# Author(s): STMicroelectronics

# Title: Artificial Intelligence on STM32 Microcontrollers

# Source: ST Application Note (AN5259)

# KeyContribution: Explains deployment of neural networks on STM32 using X-CUBE-AI, including memory optimization and inference flow.

# RelevancetoProject: Direct technical reference for AI model deployment on STM32F407VGT6.

# [B] PAPER 2: SPI-Based Inter-MCU Communication

# Author(s): Zhang et al.using a real-time operating system to manage multiple vehicle

# Title: *Efficient Inter-Processor Communication in Embedded Systems Using SPI*

# Journal: International Journal of Embedded Systems, 2018

# KeyContribution: Analyzes SPI communication for reliable data exchange between heterogeneous microcontrollers.

# RelevanceProject: Justifies the STM32 (master) to ESP32 (slave) SPI communication architecture.

# [C] PAPER 3: Edge-to-Cloud Data Integration

# Author(s): Shi et al.

# Title: *Edge Computing: Vision and Challenges*

# Journal: IEEE Internet Computing, 2016

# KeyContribution: Describes architectures for edge-to-cloud systems and highlights the benefits of processing data locally before cloud transmission.

# RelevancetoProject: Supports sending processed inference results to AWS Lambda instead of raw sensor data.

# Chapter 3

# SYSTEM DESIGN

# 3.1 Methodology:

# The proposed Hot Swappable Machine Learning Model at the Edge is designed using a modular and layered approach, where sensing, processing, communication, visualization, and cloud connectivity are handled in an organized manner.

# 1. System Architecture and Design

# The system is designed around the STM32F407VGT6 microcontroller, which serves as the central processing unit responsible for sensor interfacing, machine learning inference, peripheral control, and communication management. The architecture is modular, separating sensing, inference, communication, display, and update mechanisms to improve scalability, reliability, and maintainability of the overall system.

# 2. Data Acquisition and Preprocessing

# The A DHT11 temperature sensor is interfaced with the STM32F407VGT6 to acquire real-time temperature data. The raw sensor data is validated and preprocessed through scaling and normalization to ensure compatibility with the machine learning model input requirements. This step improves inference accuracy and consistency by reducing noise and handling variations in sensor output.

# 3. Edge-Based Machine Learning Inference

# The preprocessed temperature data is fed into a machine learning model deployed on the STM32 using the X-CUBE-AI framework. The model performs real-time inference to classify temperature conditions into cold, warm, or hot categories. Executing inference at the edge enables low-latency decision-making and reduces dependence on external processing resources.

# 4. Local Visualization and Debugging

# The inference results are displayed on an OLED display using I²C communication to provide real-time visual feedback. In parallel, the same data is transmitted over UART to a serial terminal, enabling system monitoring and debugging during development and testing. This dual-output approach enhances system observability and validation.

# 5. Hot-Swappability

# The entire system operates in a continuous loop where sensor data is acquired, processed, transmitted over CAN, displayed on the dashboard, and uploaded to the cloud. The modular architecture ensures scalability, fault isolation, and ease of future enhancements such as diagnostics, alerts, and advanced analytics.

# 6. Inter-Controller Communication Using SPI

# The STM32 is configured as the SPI master and communicates with an ESP32 configured as the SPI slave. Classified temperature data is transmitted efficiently over SPI, ensuring reliable and high-speed data exchange between the two microcontrollers while maintaining synchronization and data integrity.

# 7. Cloud Integration via ESP32

# The ESP32 receives the classified data from the STM32 and forwards it to the AWS cloud using AWS Lambda. This enables remote monitoring, data storage, and cloud-based analytics. Sending processed inference results instead of raw sensor data optimizes bandwidth usage and improves system efficiency.

# 

Fig 1 : Block Diagram

# 3.2 Hardware Description

# STM32F40VGT6 Discovery Board:

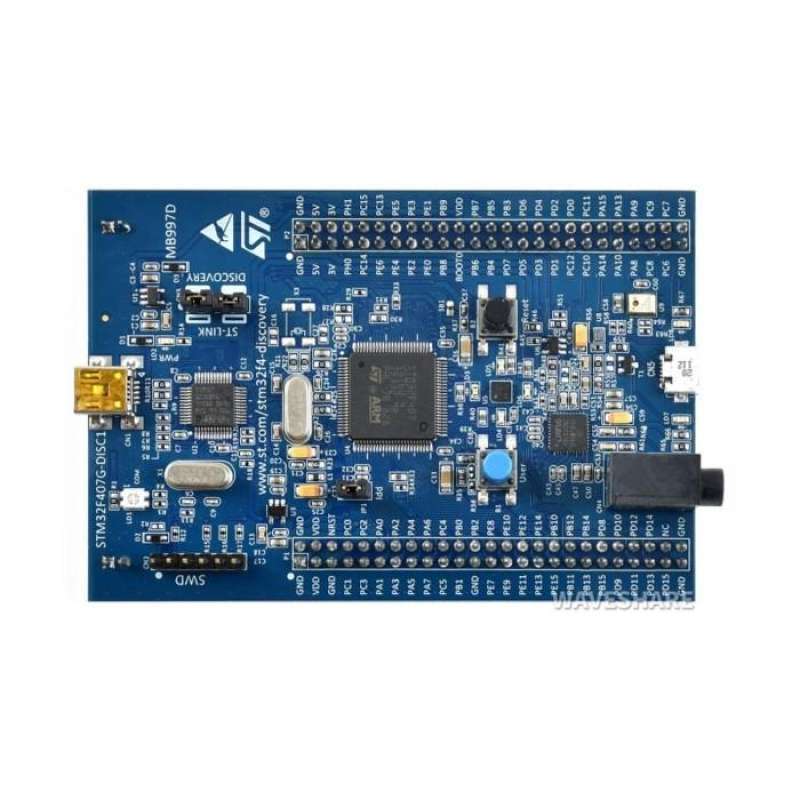


Fig 2 : STM32F407VGT6

# The STM32F407VGT6 is a 32-bit ARM Cortex-M4 based microcontroller with a maximum operating frequency of 168 MHz. In the proposed smart car dashboard system, it functions as the main vehicle ECU, responsible for collecting and processing real-time data such as vehicle speed, battery status, seatbelt indication, and door ajar status. The microcontroller runs FreeRTOS to handle multiple tasks concurrently with deterministic timing and uses its integrated CAN controller to transmit processed vehicle data to the ESP32 gateway. Its high processing speed, real-time capability, and extensive peripheral support make it well suited for automotive and real-time embedded applications.

# ESP32 Dev Module:

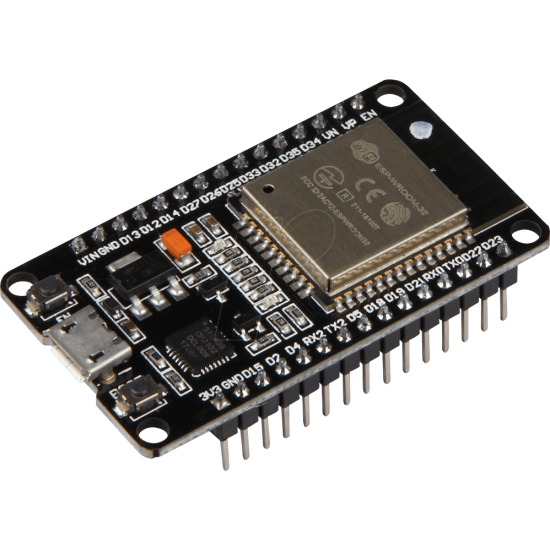


Fig 3 : ESP32 Dev Module

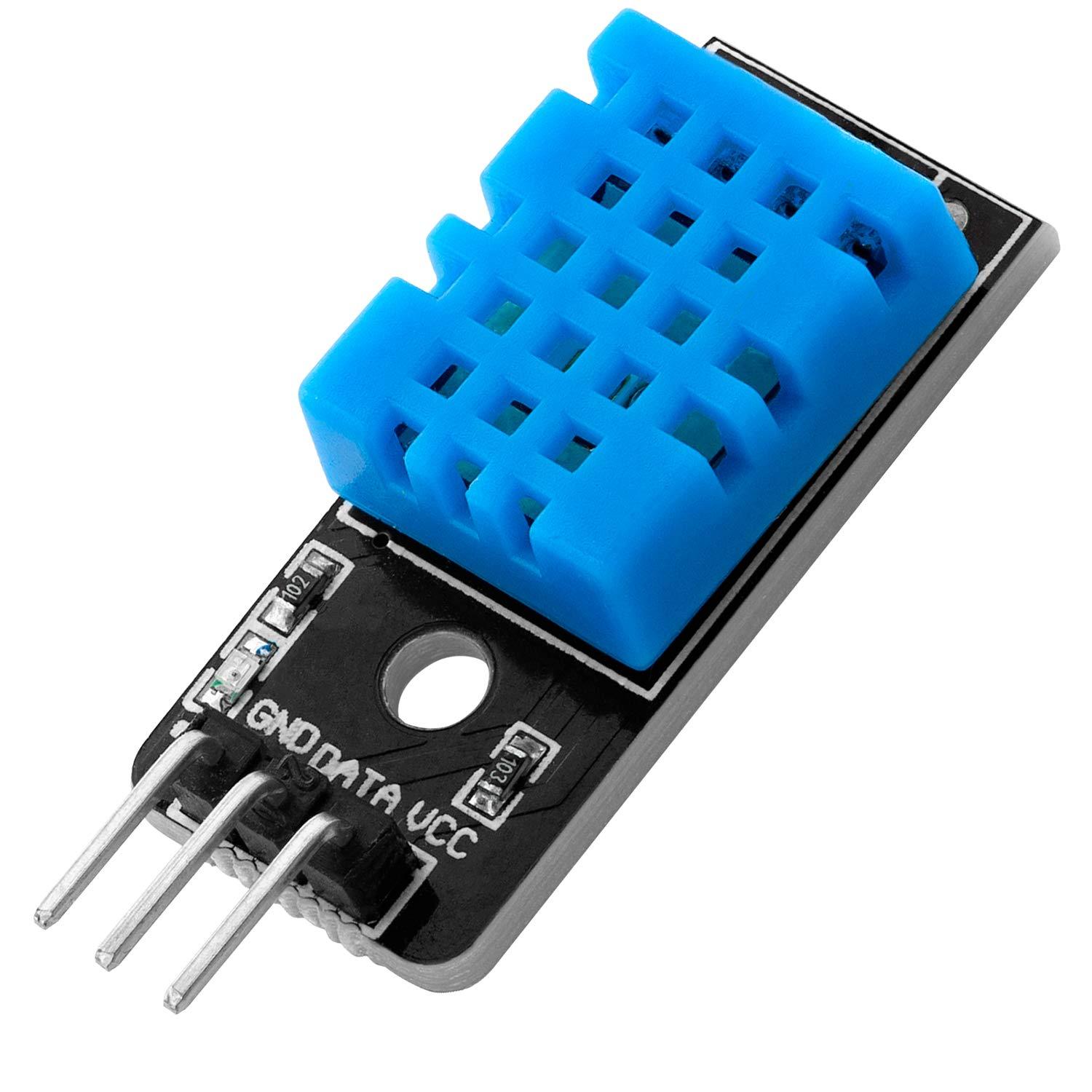
# The ESP32 is a powerful, low-cost microcontroller with built-in Wi-Fi and Bluetooth

# capabilities, widely used in IoT applications. In this smart car dashboard project, the

# ESP32 functions as a gateway ECU between the in-vehicle network and external systems. It receives real-time vehicle data such as speed, battery status, seatbelt status, and door ajar information from the STM32 controller via the CAN bus. The ESP32 processes and formats this data for further communication. Using its integrated Wi-Fi module, the ESP32 transmits the received vehicle parameters to the AWS IoT cloud using the MQTT protocol, enabling remote monitoring and data logging. Simultaneously, it communicates with the DWIN LCD display through a serial interface to provide real-time visualization of vehicle information to the driver. By offloading cloud connectivity and display handling from the STM32, the ESP32 helps maintain real-time performance and improves overall system reliability and scalability.

# DHT11 Sensor:

# 



# 

Fig 4: DHT11 Sensor

# The SN65HVD230 is a high-speed CAN transceiver designed for reliable communication in automotive and industrial environments. In this smart car dashboard project, it acts as the physical interface between the CAN controller of the STM32/ESP32 and the CAN bus lines (CANH and CANL). The transceiver converts the logic-level CAN signals from the microcontroller into differential signals suitable for transmission over the CAN bus and vice versa. SN65HVD230 provides high noise immunity, low electromagnetic interference, and robust fault protection, which are essential for in-vehicle communication. It supports standard CAN data rates and ensures stable communication between the STM32 ECU and the ESP32 gateway. By using this transceiver, the system achieves reliable and error-resistant data exchange for real-time vehicle parameters such as speed, battery status, and safety indicators.

1. **UART TTL Module:**

Fig 5 : UART TTL Module



# The MCP2515 CAN module is a standalone CAN controller that communicates with microcontrollers using the SPI interface. In this smart car dashboard project, it is used to provide CAN communication capability to the ESP32, which does not have a built-in CAN controller. The module handles CAN protocol functions such as message framing, filtering, arbitration, and error handling. The MCP2515 works in conjunction with a CAN transceiver to transmit and receive vehicle data over the CAN bus. It enables reliable communication between the STM32 ECU and the ESP32 gateway, allowing real-time transfer of parameters such as speed, battery status, seatbelt indication, and door ajar status. Its flexibility and compatibility make it suitable for implementing CAN-based automotive networks in embedded systems.

# OLED Display:

# 

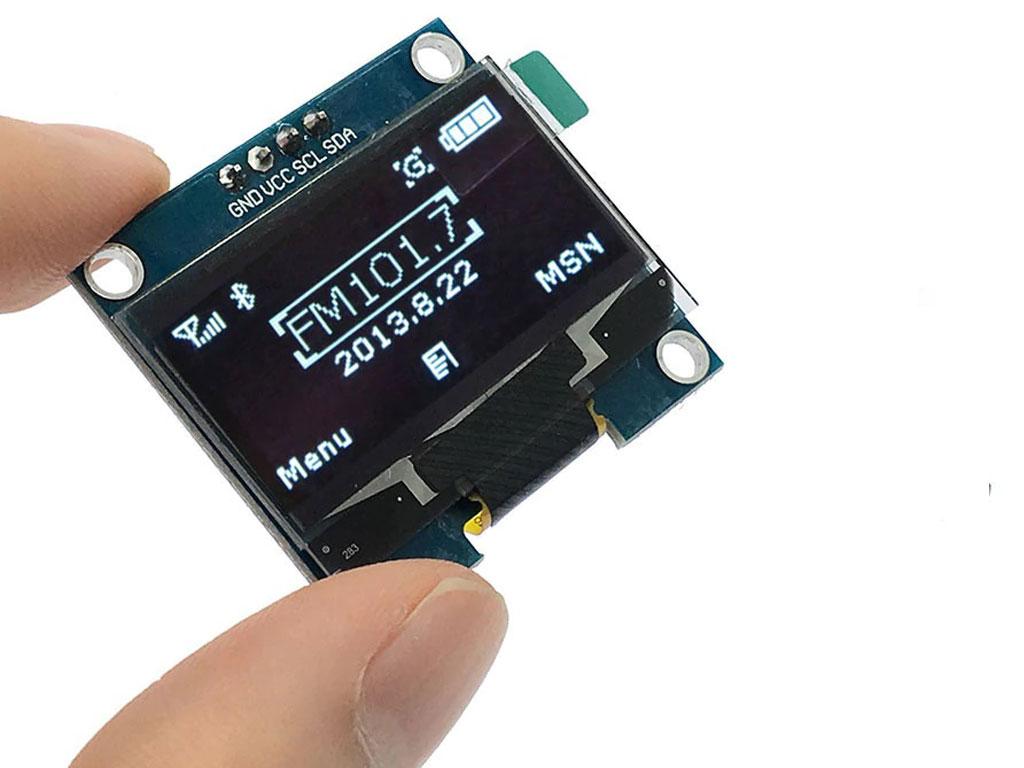


Fig 6 : OLED Display

# The OLED (Organic Light Emitting Diode) display is a compact, low-power visual output device commonly used in embedded systems for real-time data visualization. It is a self-emissive display technology in which each pixel emits its own light, resulting in high contrast, wide viewing angles, and excellent readability without the need for a backlight. In this project, a monochrome OLED display with a typical resolution of 128×64 pixels is interfaced with the STM32F407VGT6 microcontroller using the I²C communication protocol, which minimizes pin usage and simplifies hardware design. The OLED is used to display real-time temperature values and the corresponding classification results (Cold, Warm, or Hot) generated by the edge-based machine learning model, providing immediate visual feedback to the user. Its low power consumption, fast response time, and clear text rendering make it suitable for continuous operation and effective human–machine interaction in embedded applications.

# 3.3 Software Modules Description

# STM32CubeIDE:

# STM32CubeIDE is an integrated development environment used for configuring peripherals (GPIO, UART, SPI, I2C, TIM), writing embedded C code, compiling, debugging, and deploying the edge AI application on the STM32F407VGT6 using STM32CubeMX and X-CUBE-AI middleware.

# X-CUBE-AI:

# X-CUBE-AI is an STM32 expansion package that converts trained neural network models into optimized C code and enables on-device AI inference on the STM32 microcontroller without requiring an external processor.

# Arduino IDE:

# Arduino IDE is used for programming the ESP32, handling SPI data reception from STM32, managing Wi-Fi connectivity, and transmitting temperature classification data to the AWS cloud services.

# STM32CubeProgrammer:

# STM32CubeProgrammer is used to flash firmware, perform memory programming, erase and verify flash, and support firmware-over-the-air (FOTA) validation during hot-swappable updates.

# PuTTY (UART Terminal):

# PuTTY is used as a serial communication terminal to monitor real-time temperature readings and AI classification outputs transmitted from the STM32 via UART.

# AWS Cloud Services:

# Cloud services such as AWS Lambda are used to receive, process, and store data sent by the ESP32, enabling remote monitoring and scalability.

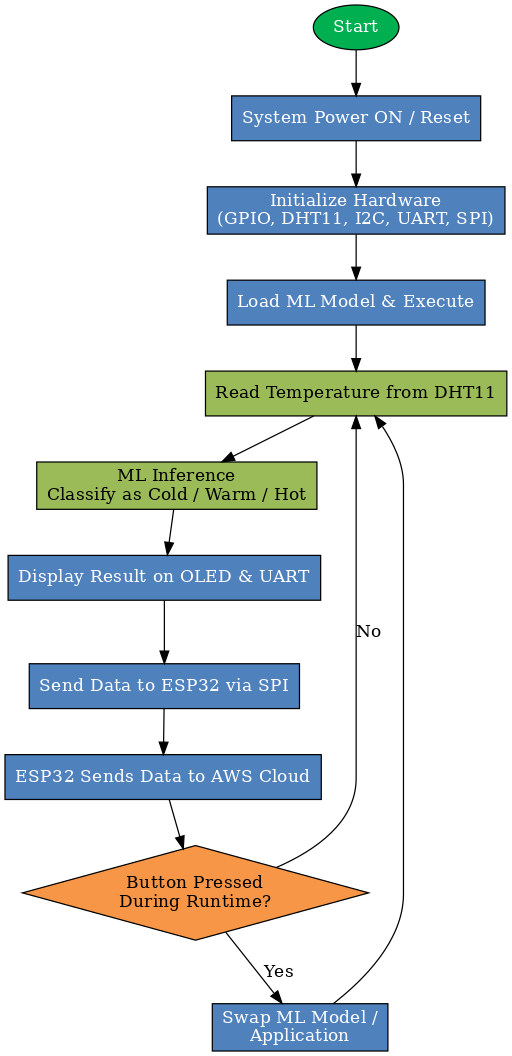
****

Fig 8 : Flowchart

# Chapter 4

# METHOLOGY AND TECHNIQUES

# 4.1 System Overview:

# The *Hot Swappable Machine Learning Model at the Edge* system performs real-time temperature monitoring and intelligent classification using an STM32F407VGT6 microcontroller as the core processing unit, where temperature data acquired from a DHT11 sensor is preprocessed and fed to an onboard machine learning model to classify conditions as cold, warm, or hot, with results displayed locally on an OLED via I²C and simultaneously monitored through a UART terminal, while the classified data is transmitted to an ESP32 over SPI for forwarding to the AWS cloud using AWS Lambda, and the system supports dynamic model updates through Firmware Over-The-Air (FOTA) with formal swapping, enabling seamless replacement of the deployed machine learning model without interrupting system operation.

# 4.2 Component List and Purpose:

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Description** | **Protocol / Interface Used** |
| STM32F407VGT6 | Main Controller | I2C, UART, SPI |
| ESP32 | Communication Gateway | SPI |
| DHT11 | Temperature Sensor | Single-wire digital communication protocol |
| OLED Display | Local visualization and human–machine interface (HMI) | SPI |
| UART TTL | Terminal for monitoring | UART |

# 4.3 STM32F407VGT6 Pin Configuration:

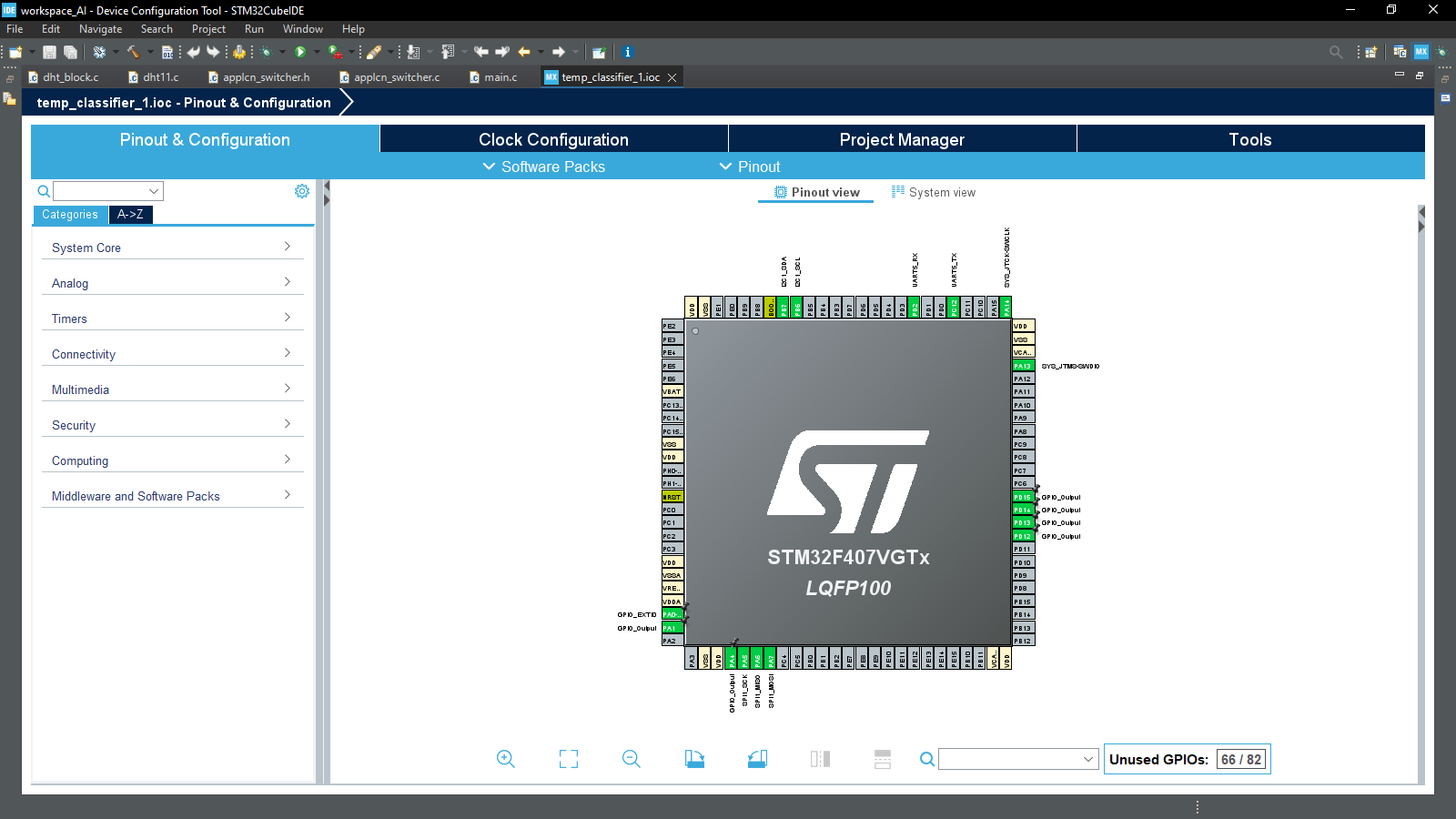


Fig 8 : STM32F407VGT6 Pin Configuration

# 4.4 STM32F407VGT6 Clock Configuration:

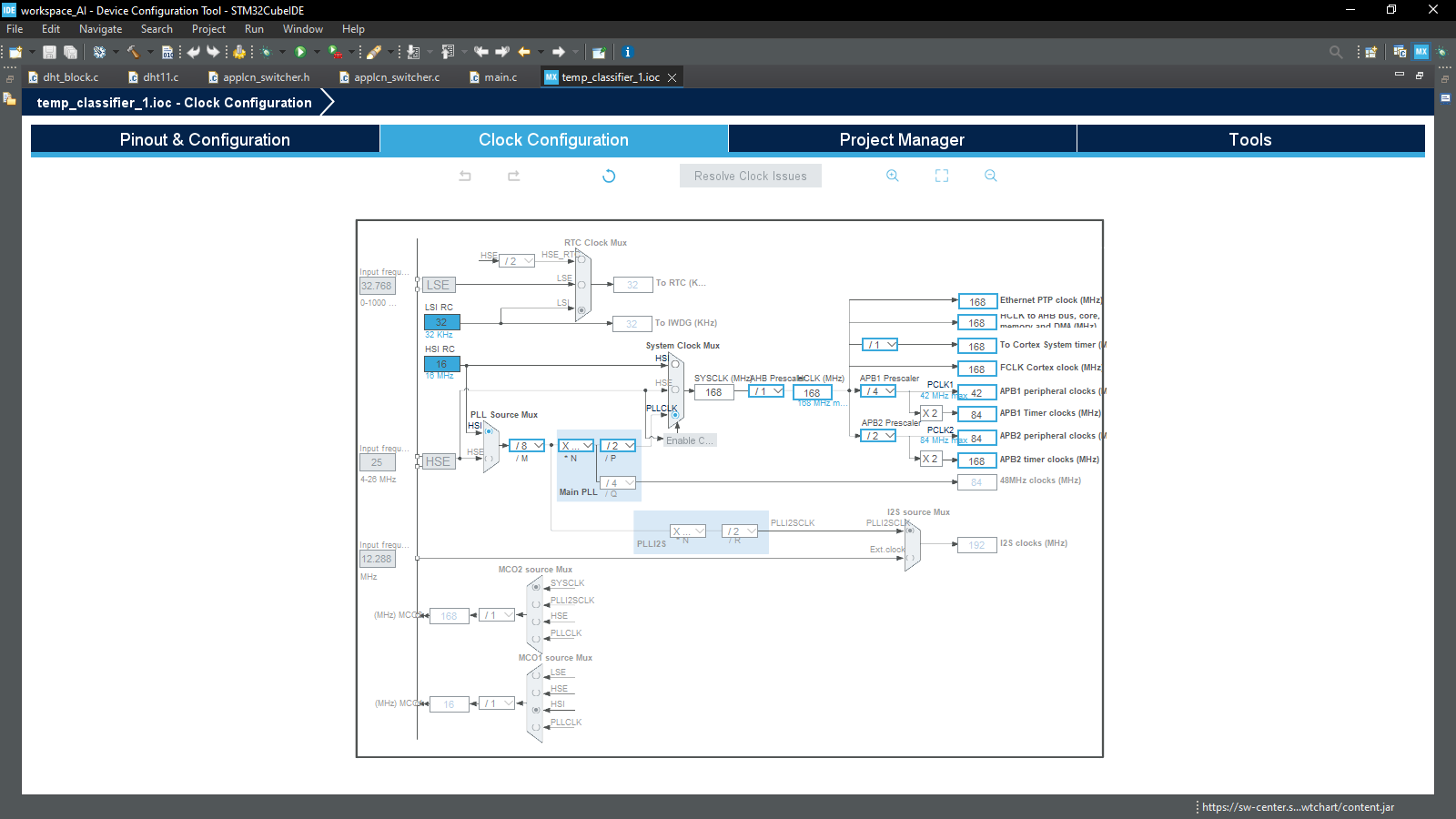


Fig 9 : STM32F407VGT6 Clock Configuration

# 4.5 Communication Flow:

# The communication flow of the system begins with the DHT11 temperature sensor transmitting digital temperature data to the STM32F407VGT6 using a single-wire protocol via a GPIO pin, after which the STM32 preprocesses the data and performs machine learning inference locally; the classified result (Cold/Warm/Hot) is then sent simultaneously to the OLED display over the I²C protocol for local visualization and to a desktop system via UART-TTL using the UART protocol for real-time monitoring and debugging, while in parallel the STM32 operates as an SPI master to transmit the classified temperature data to the ESP32 configured as an SPI slave, and finally the ESP32 forwards this processed data to the AWS cloud using Wi-Fi and AWS Lambda, enabling remote monitoring and cloud-level data handling.

# 4.6 Sensor Data Acquisition:

# Sensor data acquisition is the initial stage of the system in which real-time environmental temperature data is obtained using the DHT11 temperature sensor interfaced with the STM32F407VGT6 microcontroller. The STM32 initiates communication by sending a start signal on a GPIO pin, after which the DHT11 responds and transmits digital temperature data using its single-wire communication protocol. The received data is decoded based on precise timing of high and low pulses and is validated using the checksum provided by the sensor to ensure data integrity. Once validated, the temperature value is extracted and made available for further processing, where it serves as the primary input to the edge-based machine learning model for real-time temperature classification.

# 4.7 Communication Protocol:

# 4.7.1 Universal Asynchronous Receiver Transmitter (UART):

# UART (Universal Asynchronous Receiver Transmitter) is an asynchronous serial communication protocol used in this project only for debugging and monitoring purposes. It enables serial data transmission between the STM32F407VGT6 and a PC without using a clock signal, using two lines: Transmit (TX) and Receive (RX).

# In the proposed system, the STM32 transmits real-time DHT11 temperature values and the corresponding AI model output (Hot/Warm/Cold) to a PC via UART, where the data is displayed on the PuTTY terminal. Prior to communication, parameters such as baud rate, data length, stop bits, and parity are configured on both the STM32 and the terminal to ensure reliable data exchange.

# UART is chosen for debugging because of its simplicity, low hardware overhead, and ease of integration, making it ideal for observing internal system behavior, validating sensor readings, and verifying AI inference results during development and testing.

# 4.7.2 Serial Peripheral Interface (SPI):

# In this project, the Serial Peripheral Interface (SPI) protocol is used as a high-speed and reliable communication mechanism between the STM32F407VGT6 microcontroller and the ESP32 module, where the STM32 operates as the SPI master and the ESP32 functions as the SPI slave. After acquiring temperature data from the DHT11 sensor and performing on-device AI inference, the STM32 packages the processed temperature value into a fixed-length data frame of four uint8\_t bytes and transmits it to the ESP32 over the SPI bus using MOSI, MISO, SCLK, and CS lines.

# The use of a fixed 4-byte payload ensures deterministic communication timing and simplifies synchronization between the master and slave devices. On the ESP32 side, the received four bytes are reassembled and typecast into a floating-point value, which is then used for cloud transmission and further processing. SPI is selected over other communication protocols due to its full-duplex capability, low latency, and minimal protocol overhead, making it well suited for real-time edge-to-gateway communication in embedded AI applications.

# Overall, the SPI-based master–slave architecture provides a robust, efficient, and scalable communication link between the edge AI node and the cloud gateway, enabling reliable data transfer required for real-time monitoring and cloud integration.

# 4.7.3 Inter-Integrated Circuit (I²C):

# In this project, the Inter-Integrated Circuit (I²C) protocol is used to interface the OLED display module with the STM32F407VGT6 microcontroller for real-time visualization of temperature data and AI inference results. The STM32 operates as the I²C master, while the OLED display acts as the I²C slave device, enabling communication over a two-wire interface consisting of Serial Data (SDA) and Serial Clock (SCL) lines.

# The I²C protocol is used to transmit configuration commands and display data from the STM32 to the OLED controller, where initialization commands configure display parameters such as resolution, contrast, and addressing mode, followed by continuous data updates to display the measured DHT11 temperature and the classified output of the machine learning model (Hot, Warm, or Cold). The use of I²C minimizes GPIO usage, supports multiple slave devices on the same bus, and provides sufficient bandwidth for text-based OLED updates.

# Overall, I²C is selected for OLED interfacing due to its simplicity, low hardware complexity, and reliable master–slave communication model, making it well suited for embedded visualization in real-time edge AI systems.

# Fig 10 : Connection Diagram

# Chapter 5

# Results

# C:\JD_BACKUP\PROJECTS\hot swap ml\Screenshot (32).png

Fig 11: PUTTY UART output

# C:\JD_BACKUP\PROJECTS\hot swap ml\Screenshot 2026-01-29 021147.png

Fig 12: ESP32 SERIAL MONITOR

# C:\JD_BACKUP\PROJECTS\hot swap ml\Screenshot 2026-01-29 021125.png

Fig 13: cloud integration

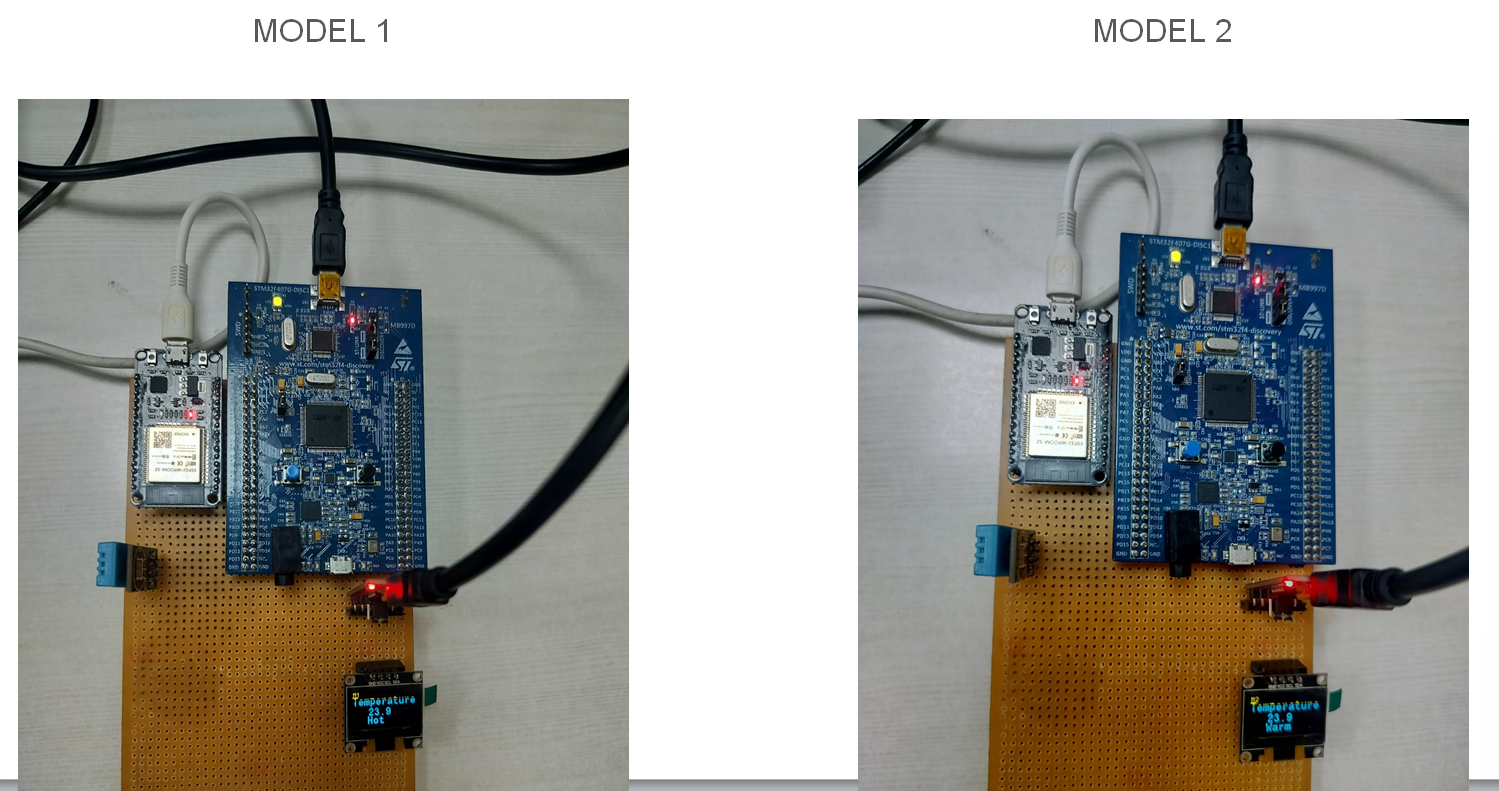


Fig 14: model swapping

# Chapter 6

# Conclusion

# The project successfully demonstrates a real-time edge-AI based intelligent IoT system in which temperature data acquired from a DHT11 sensor is processed locally on an STM32F407VGT6 microcontroller using an embedded machine learning model to classify environmental conditions into hot, warm, and cold categories, thereby eliminating dependency on continuous cloud connectivity and ensuring low-latency decision making. The classified results are simultaneously visualized on an OLED display for local monitoring and transmitted over UART to a PuTTY terminal for debugging and validation, enabling clear observation of system behavior during runtime.

# Furthermore, the system architecture incorporates SPI-based communication between the STM32 and ESP32, where the ESP32 acts as a connectivity gateway to the cloud, ensuring efficient and reliable data transfer from the edge device to cloud services. The ESP32 publishes the classified temperature data to the cloud using HTTP-based communication, where AWS Lambda is integrated with Amazon API Gateway to support both HTTP GET and POST methods, enabling secure, scalable, and serverless data ingestion, processing, and remote monitoring of edge-node status. This cloud integration allows real-time visualization, storage, and further analytics without imposing computational overhead on the edge device.

# A key contribution of this project is the implementation of hot swappablity of machine learning models, which allows replacement of a low accuracy machine learning model and deployment of an improved model. Overall, the project validates the practical feasibility of a low power, low latency upgradable edge ML system.

# Chapter 7

# Future Scope

# The future scope of this project can be extended by enabling true runtime model fetching and hot swapping, where multiple trained machine learning models are stored in the cloud and dynamically downloaded to the STM32 during operation based on application requirements, environmental conditions, or performance metrics. Using secure HTTP/HTTPS APIs through Amazon API Gateway integrated with AWS Lambda, the ESP32 can periodically check for model updates, download the selected model, verify its integrity, and transfer it to the STM32 flash memory without halting the system, enabling adaptive intelligence at the edge.

# Additional future enhancements include implementing a model versioning and rollback mechanism, where multiple AI models are stored in separate flash partitions on the STM32, allowing safe switching between models and automatic rollback in case of failure. The system can also be extended to support secure boot and encrypted model transfer, ensuring authenticity and protection against tampering in production environments.

# The project can further evolve by incorporating adaptive model selection, where the cloud analyzes long-term temperature trends and dynamically decides which optimized model (low-power, high-accuracy, or fast-inference) should be deployed to the edge device. Integration of additional environmental sensors such as humidity, gas, or pressure sensors can enable multi-parameter inference, making the system suitable for advanced predictive analytics.

# From an implementation perspective, the future steps include defining a memory map for model storage, developing a bootloader or model manager, implementing checksum or digital signature verification, and enabling runtime pointer redirection to switch inference engines without restarting the MCU. Overall, these enhancements would transform the system into a fully autonomous, cloud-orchestrated, hot-swappable edge-AI platform capable of large-scale industrial deployment and continuous intelligence evolution.

# Chapter 8

# References

# [1]STMicroelectronics,*STM32F407xx Reference Manual (RM0090)*, STMicroelectronics, 2023.

# [2]STMicroelectronics,*STM32CubeIDE User Guide*, STMicroelectronics, 2023.

# [3] STMicroelectronics, *X-CUBE-AI Expansion Package for STM32*, STMicroelectronics, 2023.

# [4] Aosong Electronics, *DHT11 Temperature and Humidity Sensor Datasheet*, Aosong Electronics Co., Ltd.

# [5] Solomon Systech Ltd., *SSD1306 OLED Display Controller Datasheet*.

# [6] Espressif Systems, *ESP32 Technical Reference Manual*, Espressif Systems, 2023.