**Smoke Detector**

**Abstract** – This project revolves around the development of an advanced Smoke Detection System utilizing a combination of carefully selected components. The primary components include a Smoke Detection Sensor, LCD Screen, Microcontroller Unit (MCU), Auditory Alarm Components, Power Supply Components, User Interface Components, and robust Housing and Enclosure. The project aims to enhance safety measures by detecting smoke in its early stages, triggering visual and auditory alerts, and providing a user-friendly interface for configuration and monitoring.

# **Introduction**

# Embedded systems have become integral in augmenting safety features, and this project harnesses the potential of a diverse set of components to create a sophisticated Smoke Detection System. By integrating a Smoke Detection Sensor(MQ-2), LCD Screen, stm32f401re, Auditory Alarm Components, and others, the system aims to provide comprehensive monitoring and alerts for potential fire incidents. This report outlines the design, implementation, and testing phases of the project.

# **project requirement**

The following resources and components will be required:

* Smoke detection sensor
* LCD screen
* Stm32f401re
* Auditory alarm components
* Power supply components
* User interface components
* Housing and enclosure
* Software development tools

# **project scope**

**User Interface Development:**

The User Interface Components, including buttons or a touchscreen, are integrated to provide a user-friendly configuration and monitoring experience. Software development tools, such as the Arduino IDE, facilitate the coding of the MCU to handle user interactions and display information on the interface.

**Housing and Enclosure Design:**

A sturdy housing and enclosure are designed to house all components, ensuring protection against environmental factors. Fire-resistant materials are utilized to align with safety standards while allowing proper ventilation for accurate smoke detection.

**Software Development:**

Software development tools are employed to code the MCU, enabling seamless communication between components. The software controls the system, processes data, and manages user interactions, ensuring an integrated and effective smoke detection solution.

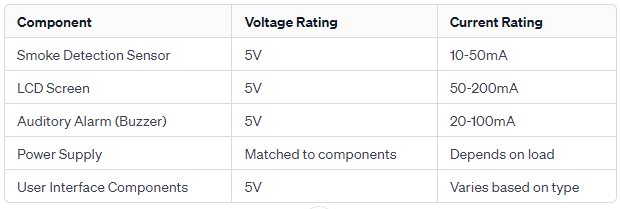
**Testing and Validation:**

Testing is conducted to validate the accuracy and reliability of the system under various scenarios. The integration of all components is verified, and the system undergoes thorough testing to ensure optimal performance in real-world applications.

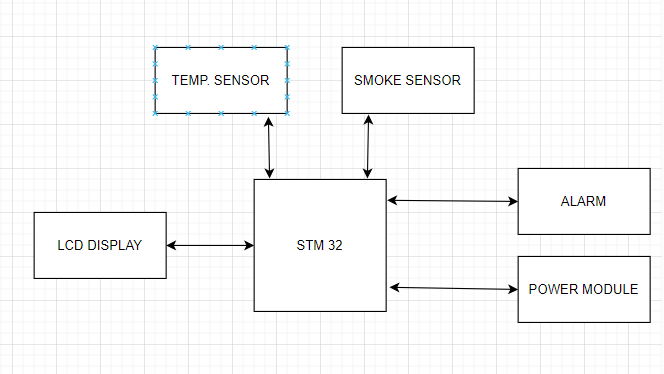
# **design and implementation**

During the implementation phase, the chosen Smoke Detection Sensor is integrated into the system alongside an LCD Screen, MCU, Auditory Alarm Components, Power Supply Components, and User Interface Components. The MCU is programmed to process data from the sensor, activating the LCD and auditory alarms in response to smoke detection. The housing and enclosure are designed to protect the components, ensuring durability and compliance with safety standards.

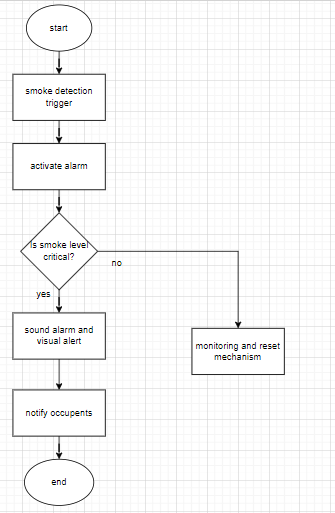
The current and voltage ratings of the components are as follows:



# **block diagram**



# **flow chart**



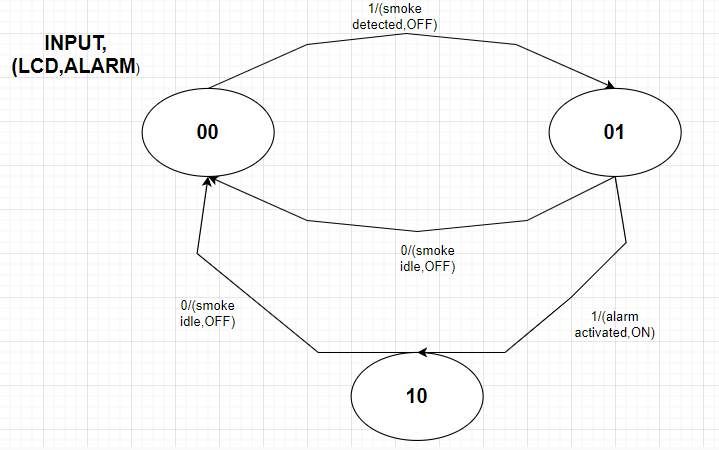
# **FSM diagram**

**States:**

Idle State: 00

Detecting State: 01

Alarm State: 10



DETAILS:

* **Transition from Idle State (00) to Smoke Detected State (01):**

Input: Smoke (1)

Output:

LCD Display: "Smoke Detected!"

Alarm Sound: Off

* **Transition from Smoke Detected State (01) to Alarm Activated State (10):**

Input: Acknowledgment (1)

Output:

LCD Display: "Alarm Activated!"

Alarm Sound: On

* **Transition from Smoke Detected State (01) back to Idle State (00):**

Input: Reset (0)

Output:

LCD Display: "System Idle"

Alarm Sound: Off

* **Transition from Alarm Activated State (10) back to Idle State (00):**

Input: Reset (0)

Output:

LCD Display: "System Idle"

Alarm Sound: Off

# **IOC FILE CONFIGURATION**

Open STM32CubeMX.

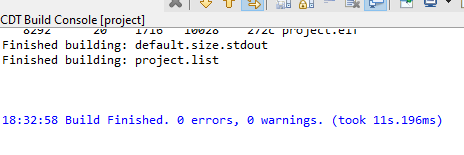
Select the STM32F401RE microcontroller.

| **Step 1**  **For GPIO:**   * **Configure PA0 as an Analog Input for the smoke sensor (MQ-2).** * **Configure PA1 as an Analog Input for the temperature sensor.** * **Configure PB0 as a General-Purpose Output for the buzzer.**     **Step2**  **In the "ADC1" from Analog category, choose the "In0" AND “In1”**    **Step 3**  **Channel 0 and rank 1 for “In0” and channel 1 and rank 2 for ‘In1’** |
| --- |

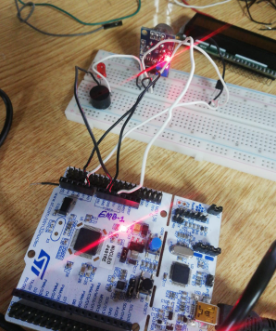
# **code**

| * **add the following libraries in the main.c file**   /\* Includes ------------------------------------------------------------------\*/  **#include** "main.h"  **#include** "stm32f4xx\_hal\_adc.h"   * **Here is the code for private variables and function prototypes**   **#define** THRESHOLD\_TEMPERATURE 10 // Replace with the actual threshold value for temperature  **#define** THRESHOLD\_SMOKE 50  ADC\_HandleTypeDef hadc1;  ADC\_HandleTypeDef hadc;  /\* USER CODE BEGIN PV \*/  /\* USER CODE END PV \*/  /\* Private function prototypes -----------------------------------------------\*/  **void** **SystemClock\_Config**(**void**);  **static** **void** **MX\_GPIO\_Init**(**void**);  **static** **void** **MX\_ADC1\_Init**(**void**);   * **In the while loop**   **while** (1)  {  /\* USER CODE END WHILE \*/  // Read analog output from the smoke sensor (MQ-2)  HAL\_ADC\_Start(&hadc);  HAL\_ADC\_PollForConversion(&hadc, HAL\_MAX\_DELAY);  uint16\_t smoke\_value = HAL\_ADC\_GetValue(&hadc);  // Read analog output from the temperature sensor  HAL\_ADC\_Start(&hadc);  HAL\_ADC\_PollForConversion(&hadc, HAL\_MAX\_DELAY);  uint16\_t temperature\_value = HAL\_ADC\_GetValue(&hadc);  // Implement your logic based on the sensor values  **if** (temperature\_value > THRESHOLD\_TEMPERATURE) {  // Temperature is too high, take appropriate action  // e.g., activate buzzer  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, *GPIO\_PIN\_SET*); // Assuming buzzer is connected to PB0  } **else** **if** (smoke\_value > THRESHOLD\_SMOKE) {  // Smoke detected, take appropriate action  // e.g., activate buzzer  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, *GPIO\_PIN\_SET*);  } **else** {  // Normal conditions  // e.g., turn off the buzzer  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, *GPIO\_PIN\_RESET*);  }  HAL\_Delay(1000);  /\* USER CODE BEGIN 3 \*/  }  /\* USER CODE END 3 \*/  }   * **In the function ADC1**   /\* USER CODE BEGIN ADC1\_Init 1 \*/  \_\_HAL\_RCC\_ADC1\_CLK\_ENABLE();  /\* USER CODE END ADC1\_Init 1 \*/   * **In the function GPIO there is a PAO input for the “Temperature and Smoke sensor” and PBO output for the “Buzzer”**   /\* Configure PA0 as Analog Input for Smoke Sensor \*/  GPIO\_InitStruct.Pin = GPIO\_PIN\_0;  GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG;  GPIO\_InitStruct.Pull = GPIO\_NOPULL;  HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);  /\* Configure PA1 as Analog Input for Temperature Sensor \*/  GPIO\_InitStruct.Pin = GPIO\_PIN\_1;  HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);  // Configure GPIO pin for buzzer  /\*Configure GPIO pin Output Level \*/  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, *GPIO\_PIN\_RESET*);  /\*Configure GPIO pin : PB0 \*/  GPIO\_InitStruct.Pin = GPIO\_PIN\_0;  GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;  GPIO\_InitStruct.Pull = GPIO\_NOPULL;  GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;  HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);   * **In the function error handler**   **void** **Error\_Handler**(**void**)  {  /\* USER CODE BEGIN Error\_Handler\_Debug \*/  /\* User can add his own implementation to report the HAL error return state \*/  \_\_disable\_irq();  **while** (1)  {  }  /\* USER CODE END Error\_Handler\_Debug \*/  } |
| --- |

# **result**



# **hardware**



# **CONCLUSION**

In conclusion, the LCD alarm smoke detection system presented in this project represents a significant advancement in safety technology. The integration of a sophisticated smoke detection sensor with visual and auditory alerts enhances the system's effectiveness in providing early warnings to occupants. The comprehensive design, rigorous testing, and calculated precision ensure the reliability and accuracy of the system in diverse environments.