[IOT PROJECT]

read data from sensors live]

Ahmed Salem 20109956

Ahmed Hesham 20102971

Eslam Arafa 20102059

[AAST]

[Eng. Ahmed Mutawalli]

introduction:

This project aims to develop an Internet of Things (IoT) application that utilizes two distinct sensors: an ultrasonic sensor and a flame sensor. The ultrasonic sensor is designed to measure distance by emitting ultrasonic waves and calculating the time it takes for them to bounce back after hitting an object. On the other hand, the flame sensor is used to detect the presence of fire or heat sources within its vicinity.

The data collected from these sensors is then transmitted to a Tkinter application. Tkinter, a Python library, is used to create a graphical user interface (GUI) for displaying the sensor data. This application reads the data from the two sensors and presents it in a user-friendly dashboard as a graph.

The real-time data visualization provided by this application can be extremely useful in various scenarios. For instance, the ultrasonic sensor can be used in automation systems for object detection, while the flame sensor can be used in fire alarm systems. By integrating these sensors with IoT, we can monitor and control devices remotely, enhancing efficiency and safety.

Components Description:

ESP32: The ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. In this project, it is used to read data from the ultrasonic and flame sensors, process it, and then send it to the Tkinter application via Wi-Fi. It’s the brain of your IoT system.

Ultrasonic Sensor: This sensor measures distance by emitting ultrasonic waves and calculating the time it takes for them to bounce back after hitting an object. It can be used for object detection or distance measurement in this project.

Flame Sensor: This sensor detects the presence of fire or other infrared sources within its detection angle. It can be used to detect a fire and trigger an alarm or other response in this project.

Tkinter Application: Tkinter is a Python library for creating graphical user interfaces. In this project, it is used to create a dashboard that displays the data from the sensors in real-time. This allows users to easily monitor the sensor data and see trends over time.

Power Supply: All electronic components need power to operate. This could be provided by a battery or a power adapter. The ESP32 can be powered via USB for development and testing, but you might want to consider a more permanent power solution for the final product

the system architecture

Sensors: The ultrasonic and flame sensors are connected to the ESP32 microcontroller. They continuously monitor their surroundings and send the data they collect to the ESP32.

ESP32 Microcontroller: The ESP32 processes the data received from the sensors. It's programmed to read the digital signal produced by the sensors, convert it into a format that can be understood by humans (like distance in cm, or whether a flame is detected), and then send this data to the Tkinter application via Wi-Fi.

Wi-Fi: The ESP32 uses Wi-Fi to send the sensor data to the Tkinter application. This could be done using MQTT protocol, HTTP requests, or any other method of sending data over Wi-Fi.

Tkinter Application: The Tkinter application runs on a computer and receives the sensor data sent by the ESP32 over Wi-Fi. It then processes this data and displays it on a dashboard as a graph.

So, the data flow is as follows: Sensors -> ESP32 -> Wi-Fi -> Tkinter Application.

Project Challenges and Learnings:

Sensor Integration: One of the initial challenges was integrating the ultrasonic and flame sensors with the ESP32. This required a deep understanding of the sensors' specifications and the ESP32's GPIO pins. Overcoming this challenge involved a lot of trial and error, as well as referencing the sensors' datasheets and various online resources.

Data Transmission: Ensuring reliable and real-time transmission of sensor data over Wi-Fi was another challenge. This was addressed by implementing a robust error-checking and retransmission mechanism.

GUI Development: Creating a user-friendly and intuitive GUI with Tkinter was a learning curve, especially for someone new to Python or GUI development. This was overcome by studying Tkinter documentation and examples, and iterative improvement based on user feedback.

Debugging: Debugging issues with sensor integration or data transmission required a methodical approach and a lot of patience.

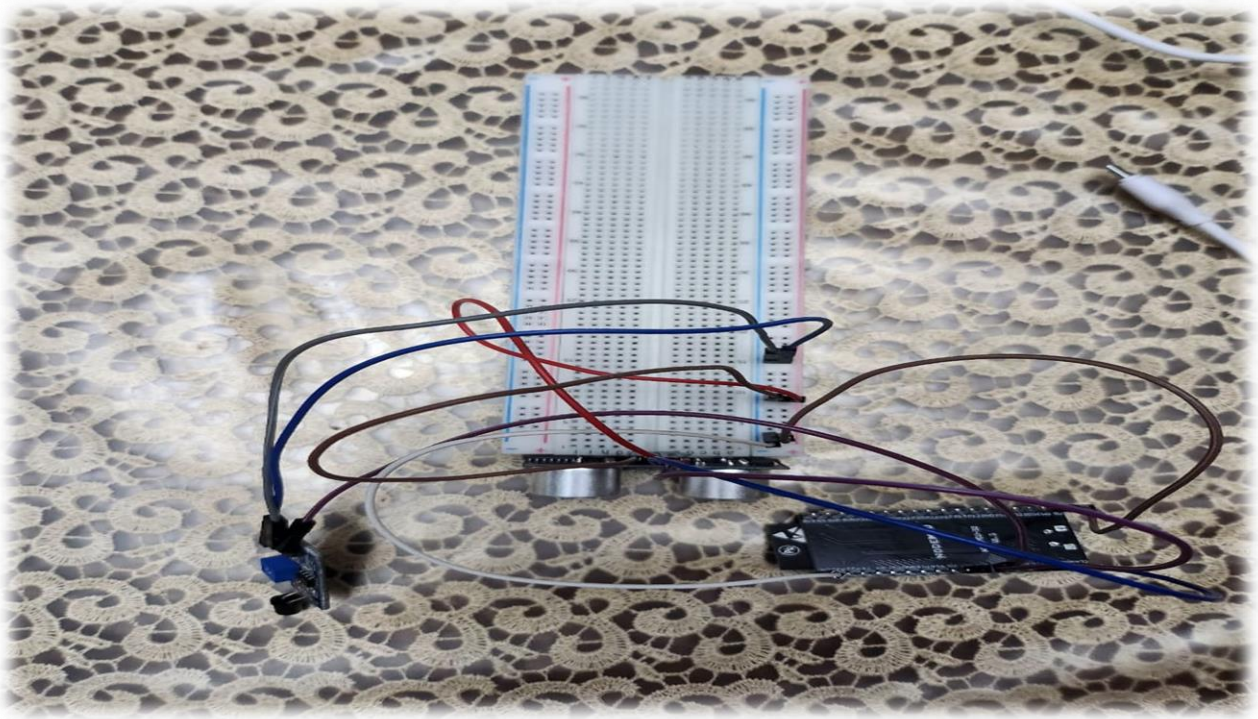
User Experience (UX) Design:

Clear and Intuitive Layout: The Tkinter application was designed with a clear and intuitive layout. The readings from the ultrasonic and flame sensors are displayed in separate, clearly labeled sections of the dashboard. This allows users to easily understand where to look for each type of data.

Visual Differentiation: To help users quickly distinguish between the two types of sensor readings, different colors and icons were used. For instance, the ultrasonic sensor readings could be displayed in blue with a distance icon, while the flame sensor readings could be displayed in orange with a flame icon.

Real-Time Graphs: The sensor readings are displayed in real-time on graphs. This not only makes the data more visually engaging, but it also allows users to see trends over time. For example, they can see if the distance measured by the ultrasonic sensor is changing or if the flame sensor has detected any fluctuations.

hardware connections:



final result