Dryer Monitoring System Documentation

Vishan Summinga-Sonagadu

Introduction

The Dryer Monitoring System is an innovative solution designed to provide users with real-time updates on their dryer's operational status. Utilizing an ESP32 microcontroller, three strategically placed limit switches, and a web-based interface, this system allows users to remotely monitor whether their dryer is running or off.

Initially, a vibration sensor was considered for detecting the dryer's activity. However, due to the washer being located next to the dryer and causing strong vibrations, the vibration sensor occasionally produced false triggers. To overcome this issue, the system was switched to using limit switches, which proved to be far more reliable in accurately detecting the dryer's status.

By continuously tracking the position of the dryer's dial through the engagement of limit switches, the system determines the current state of the dryer. Data is sent to a server, where it is logged for record-keeping and potential future analysis. This setup allows users to easily monitor their dryer's status and check historical usage. While the system itself does not perform any analytics, the collected data can be analyzed separately to identify usage patterns and recommend optimal drying times. It's worth noting that this documentation does not cover the analytics process, focusing instead on the core functionality of monitoring and data logging.

The web interface displays the current status, system health, and historical data, making it easy for users to access information from any device connected to the network. With features like real-time monitoring, visual indicators, and the ability to integrate with smart home setups, the Dryer Monitoring System offers a practical way to manage dryer operations and contribute to a more organized household routine.

System Overview

The Dryer Monitoring System is designed to provide an easy and reliable way to track the operational status of a dryer. This system comprises hardware components, a microcontroller, and a web-based user interface to ensure real-time monitoring and historical data logging.

Hardware Components

- **ESP32 Microcontroller**: The core processing unit of the system. It reads the state of limit switches to determine the dryer's status, manages Wi-Fi connectivity, and sends status updates to a server.

- **Limit Switches**: **Three** limit switches are strategically placed around the dryer's dial. These switches are triggered by a bump on the dial to indicate the dryer's off position. When none of the switches are engaged, the dryer is considered to be running.

- **LED Indicator**: An onboard LED connected to the ESP32 provides a visual indication of the dryer's status. It lights up when the dryer is running and turns off when the dryer is stopped. It also indicates Wi-Fi connection attempts during startup.

Software Components

- **ESP32 Firmware**: Custom firmware written in C++ runs on the ESP32, handling input from the limit switches, controlling the LED indicator, and managing communication with the server. The firmware checks the status of the dryer every second and sends updates to the server via an HTTP POST request.

- **Server Application**: A Node.js server receives data from the ESP32 and updates the web interface. The server also logs the dryer's status with timestamps for future reference. The server handles various routes, including status updates, system health checks, and static file serving.

- **Environment Configuration**: The system uses a .env file to store configuration settings, such as the server port, date formatting options, and system downtime thresholds. These settings allow easy adjustment without modifying the source code.

Web Interface

- **Dashboard**: A web-based interface accessible from any device on the network. It displays the current status of the dryer (on/off), system health, and historical data logs. The interface provides a user-friendly way to monitor dryer activity and ensure it is operating correctly.

- **System Health Monitoring**: The interface includes a system health page that indicates whether the monitoring system is functioning correctly. It uses a color-coded indicator (green for OK, red for Down) based on the frequency of updates from the ESP32.

- **FAQ and Support**: Additional pages such as an FAQ section provide users with information on system setup, troubleshooting, and common issues.

Data Logging and Usage

- **Data Format**: The system logs data in the format <state> <timestamp>, where 1 indicates the dryer is running, and 0 indicates it is off. Timestamps are recorded in ISO 8601 format for consistency and easy analysis.

- **Data Storage**: Logged data is stored in text files named by the current month and year. This organization facilitates easy access to historical data and supports future analysis efforts.

- **Data Analysis (External)**: While the system itself does not perform any data analytics, the collected data can be analyzed externally to identify usage patterns and recommend optimal drying times. This documentation focuses on the monitoring and logging aspects and does not cover analytics.

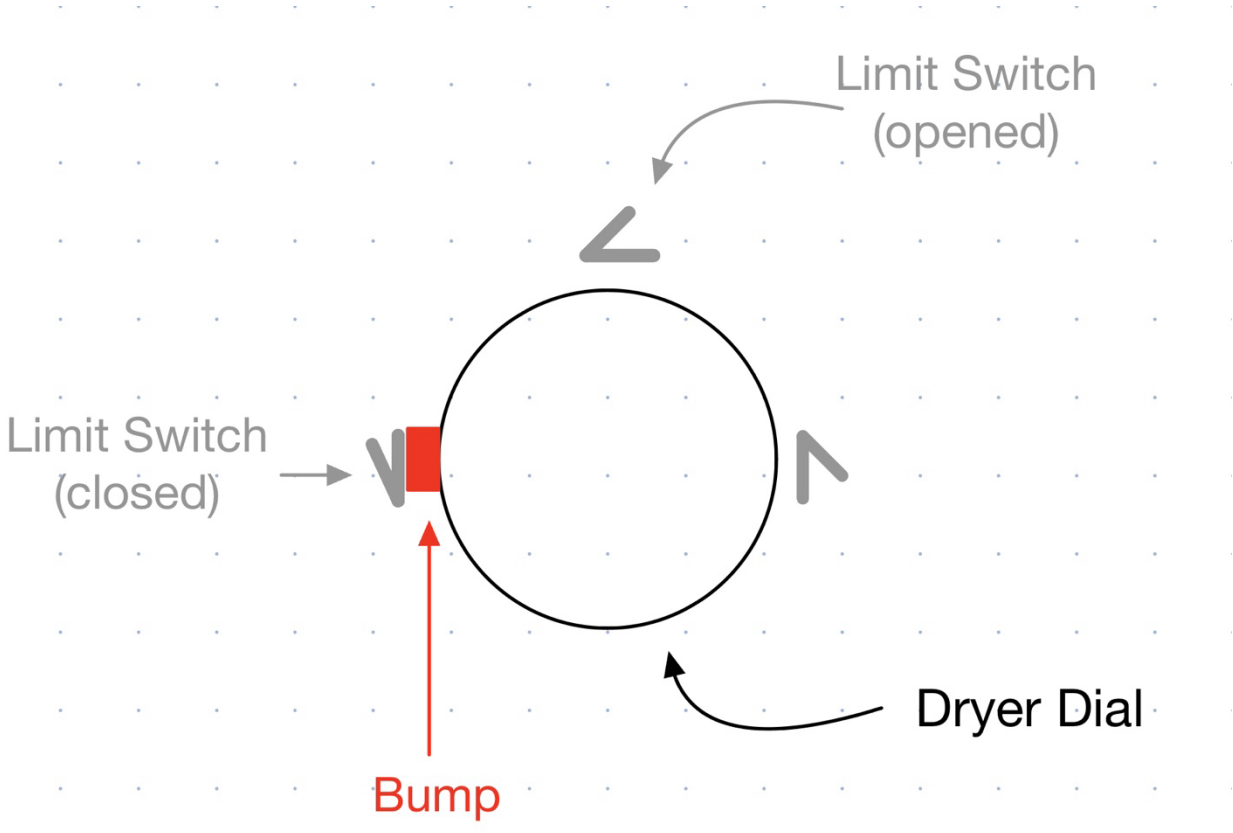
How It Works

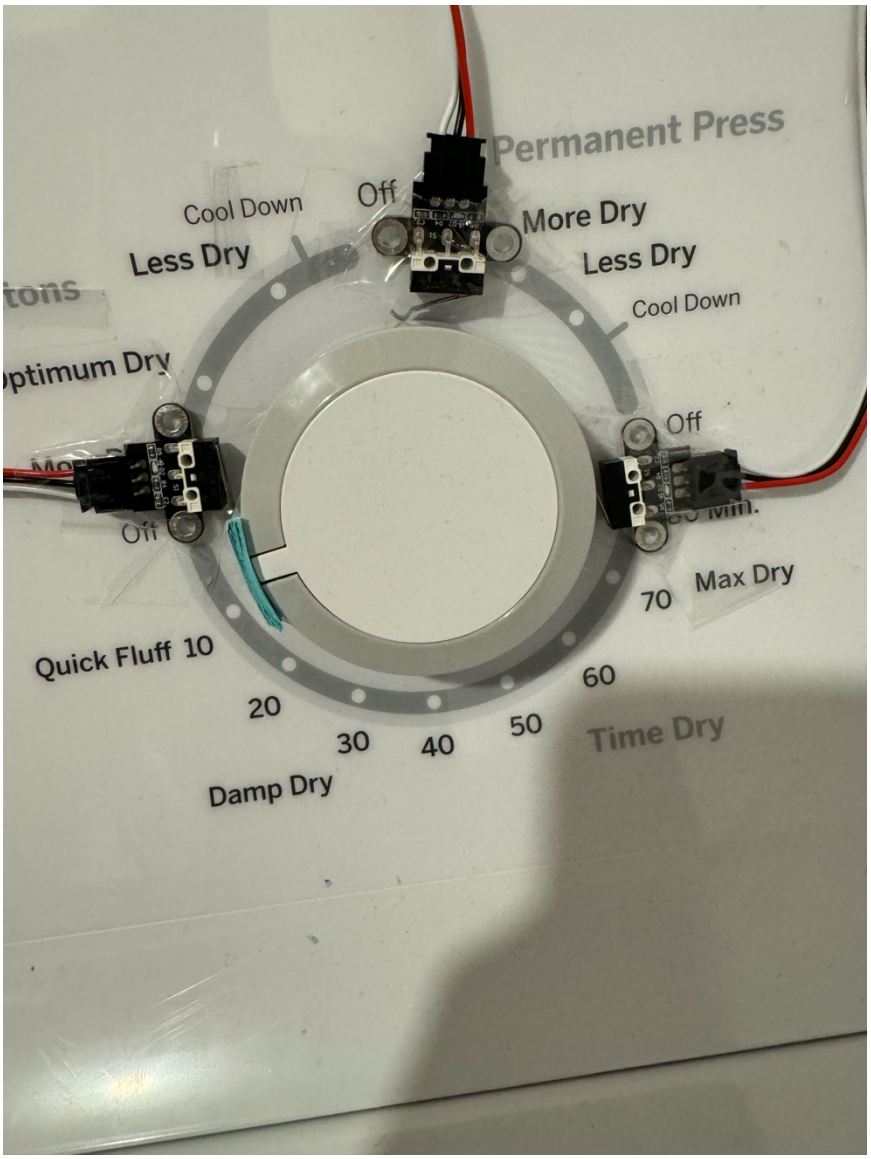
The Dryer Monitoring System operates by continuously monitoring the position of the dryer's dial to determine its operational status. This is achieved through a combination of hardware components and software running on the ESP32 microcontroller, which interacts with a server to provide real-time status updates.

Monitoring Dryer Status

Limit Switches

- Three limit switches are strategically placed around the dryer's dial. These switches are positioned at specific points corresponding to the dryer's 'off' state.
- The dryer's dial has a physical bump that engages these switches. When the bump on the dial aligns with any of the switches, the circuit closes, signaling that the dryer is off. If none of the switches are engaged, the dryer is considered to be running.





The dryer is running because the bump on the dial is not pressing any limit switch, leaving the circuit open.

ESP32 Microcontroller

- The ESP32 continuously monitors the state of the three limit switches using its GPIO pins. It reads the digital signal (LOW or HIGH) from each switch.

- A logic check is performed to determine the dryer's status:
  - If any of the switches are engaged (reading LOW), the dryer is off (isAnySwitchOn = 0).
  - If all switches are not engaged (reading HIGH), the dryer is on (isAnySwitchOn = 1).
- The built in LED lights up when the dryer is on and turns off when the dryer is off. During Wi-Fi connection attempts, the LED blinks to show connection status.

Communication with the Server

Wi-Fi Connectivity

- The ESP32 connects to a specified Wi-Fi network using credentials defined in the firmware. This connection allows the ESP32 to communicate with the server and send status updates.

- If the ESP32 loses its Wi-Fi connection during operation, it attempts to reconnect, indicated by the fast blinking LED.

Sending Status Updates

- The ESP32 sends status updates to a server located at a predefined IP address. These updates are sent as HTTP POST requests, with the dryer's status appended to the URL.

- The server receives these requests and logs the dryer's status along with a timestamp. The server also updates the web interface, allowing users to see the current status in real time.

Data Logging and Display

Data Logging

- The server logs each state change with a timestamp in text files organized by month and year. This logging allows for historical tracking of dryer usage.

- The data format used is <state> <timestamp>, where 1 indicates the dryer is on, and 0 indicates it is off. The timestamp is in ISO 8601 format, ensuring consistency and easy analysis.

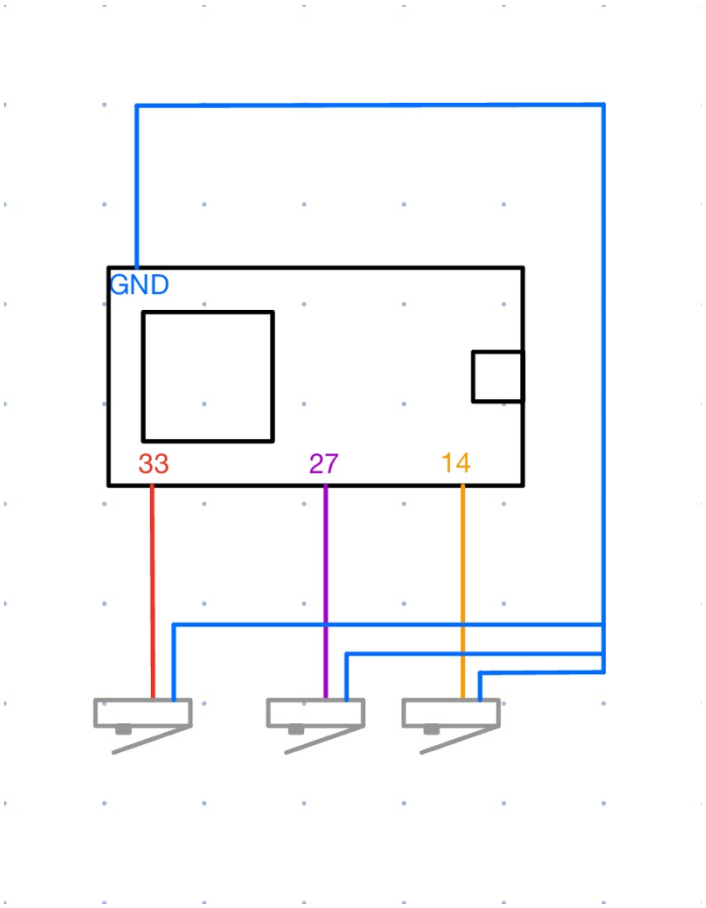
Web Interface

- The web interface displays the current status of the dryer, showing whether it is on or off and the time since. It also includes a system health indicator that reflects the status of the monitoring system.

- Users can access historical data through the interface, which shows a log of past dryer activities. This data can be used to identify usage patterns and optimize dryer operation.

Wiring

The ESP32 is connected to three limit switches through specific GPIO pins: GPIO 14, GPIO 27, and GPIO 33. Each switch is connected to one of these GPIO pins and shares a common connection to the ground (GND) pin on the ESP32.



Web User Interface

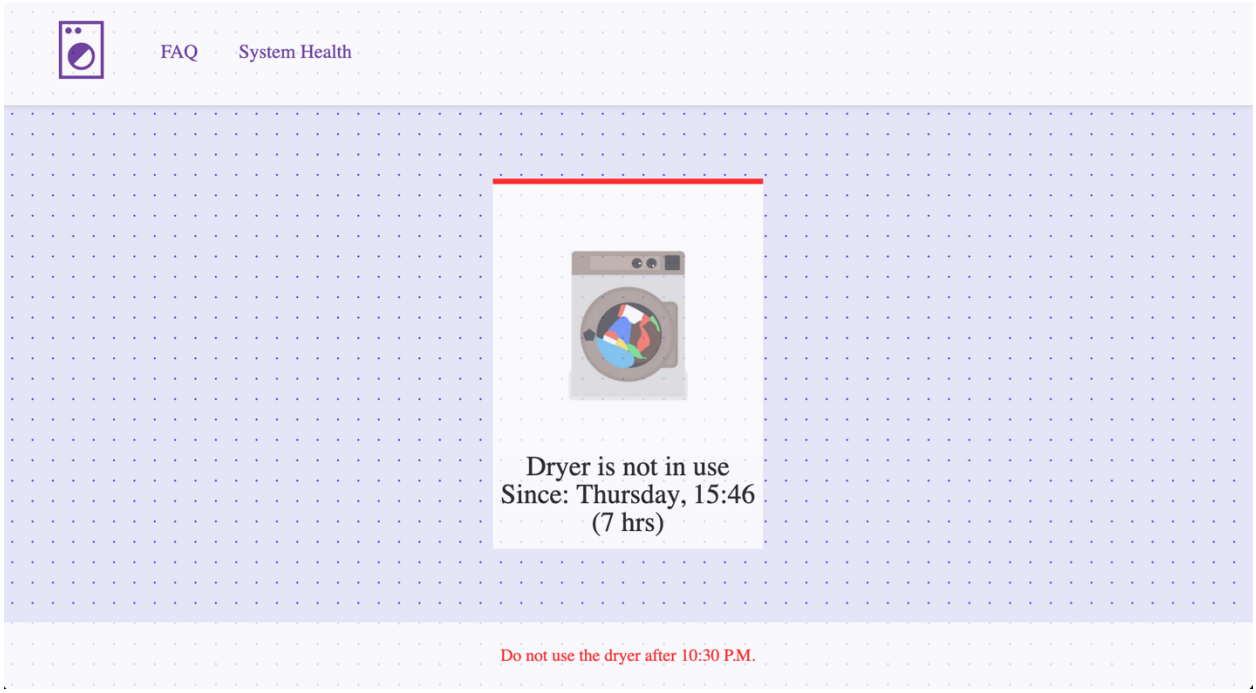
The Web User Interface (UI) is a crucial component of the Dryer Monitoring System, providing users with real-time access to the dryer's status and historical data. The interface is designed to be intuitive and accessible from any device connected to the **same network** as the server, ensuring users can monitor their dryer from anywhere in their home.

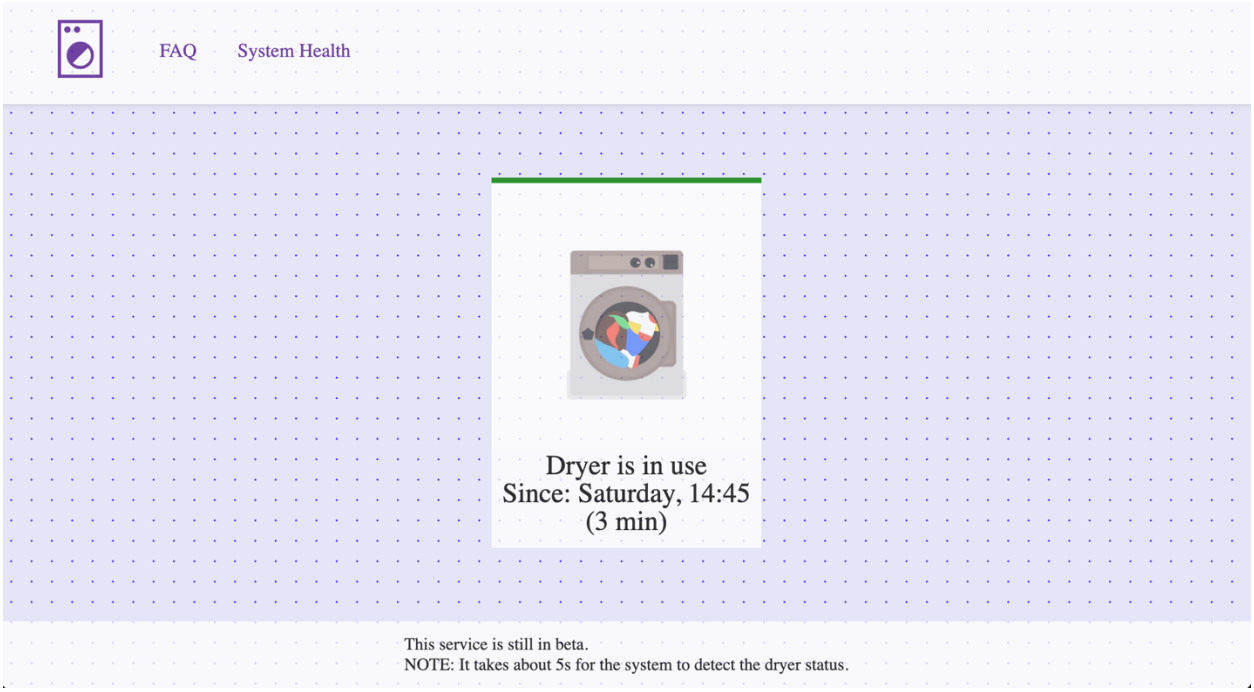
Overview

Main Dashboard

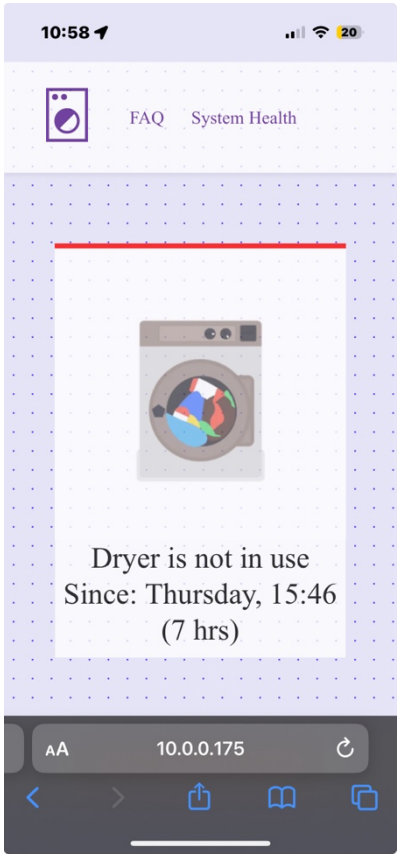
The homepage of the web interface provides a real-time update of the dryer's status. Users can see at a glance whether the dryer is currently running or off. It also shows the last time the dryer's status was updated.

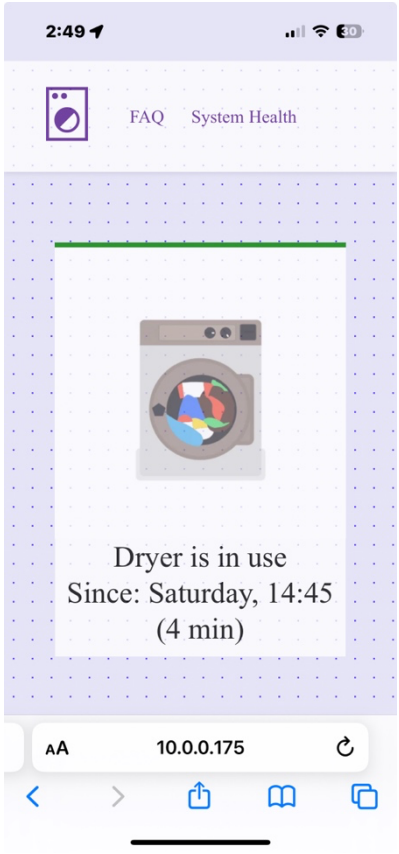
Desktop View




FAQ
System Health
Dryer is in use
Since: Saturday, 14:45
(3 min)
This service is still in beta.
NOTE: It takes about 5s for the system to detect the dryer status.

Mobile View

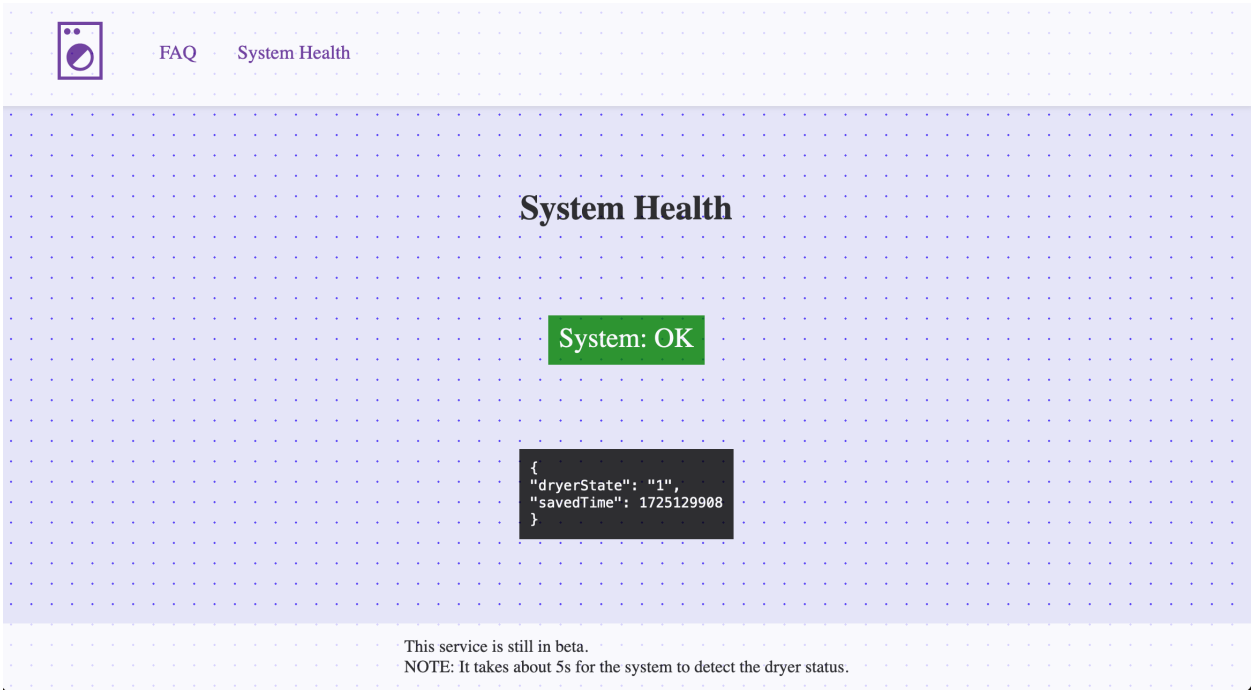

10:58
FAQ
System Health
Dryer is not in use
Since: Thursday, 15:46
(7 hrs)
AA
10.0.0.175


2:49
FAQ
System Health
Dryer is in use
Since: Saturday, 14:45
(4 min)
AA
10.0.0.175

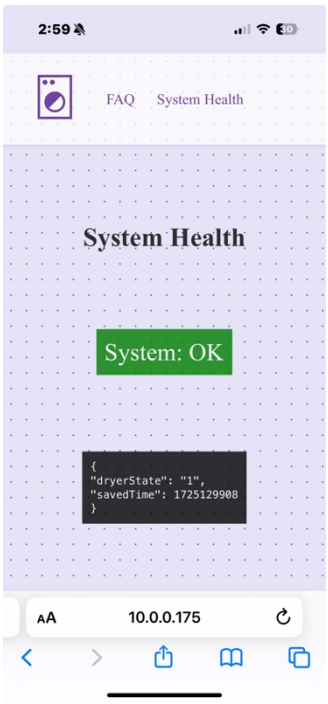
System Health

This page provides detailed information about the monitoring system's health. It indicates whether the system is operating correctly or if there have been any interruptions in the updates from the ESP32. This page helps users identify connectivity issues or potential hardware failures.

Desktop View



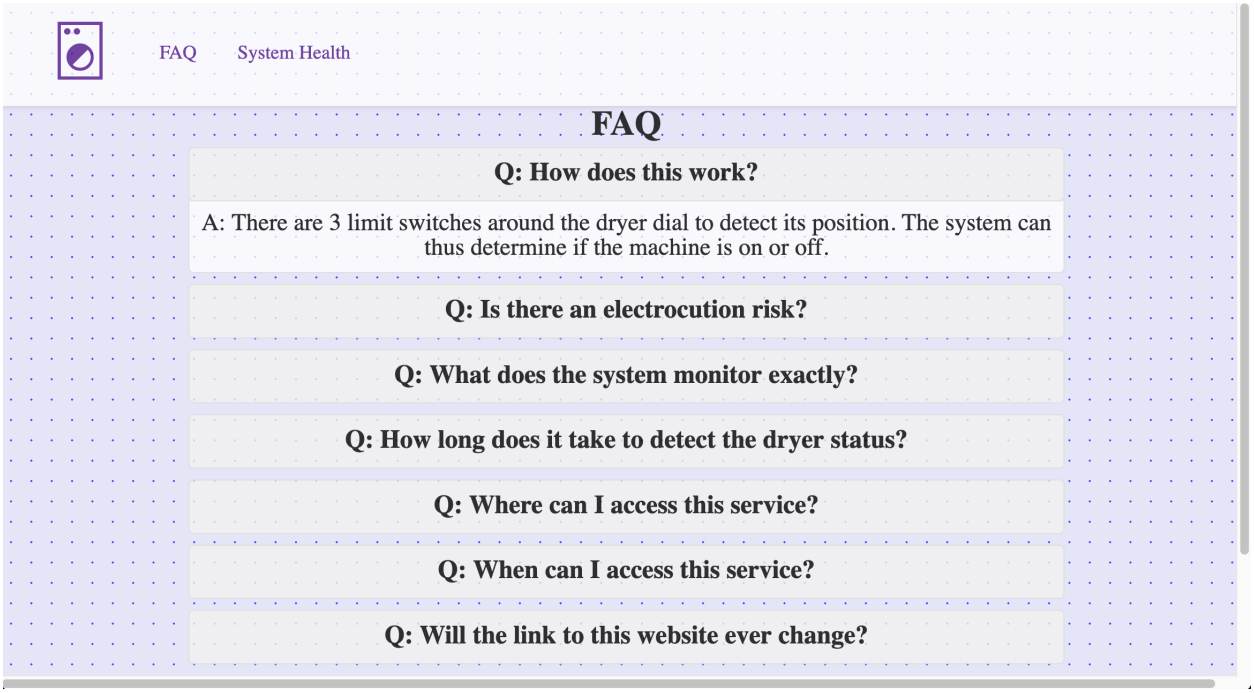
Mobile View



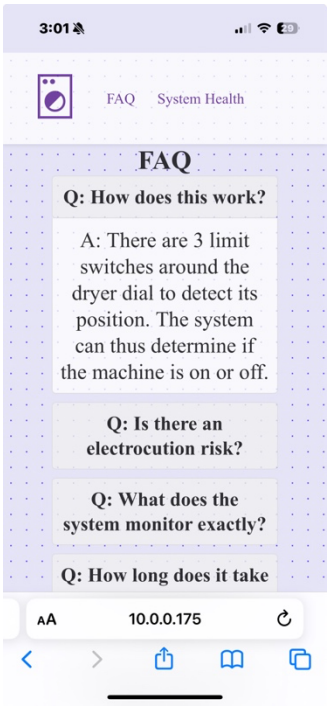
FAQ

A dedicated FAQ page is accessible from the main menu. This page provides answers to common questions about the system's operation, installation, and troubleshooting. It serves as a quick reference guide for users to understand how to use and maintain the system.

Desktop View



Mobile View



ESP32 Firmware

The ESP32 firmware is the core software running on the ESP32 microcontroller, responsible for reading the state of the limit switches, determining the dryer's status, and communicating this information to the server. This section provides an overview of the firmware's key functionalities and how it operates within the dryer monitoring system.

Overview

The ESP32 firmware is written in C++ and uses the Arduino framework. It manages the interaction between the hardware (limit switches and LED indicator) and the server, ensuring real-time monitoring of the dryer's operational status. The firmware includes routines for connecting to WiFi, reading the state of the limit switches, controlling the LED indicator, and sending HTTP POST requests to a server.

Libraries Used

```
#include <WiFi.h>
#include <HTTPClient.h>
```

The firmware relies on several libraries to handle specific functionalities:

- **WiFi.h**: This library provides the necessary functions to connect the ESP32 to a WiFi network. It handles the initialization of the WiFi module, connecting to the network, and maintaining the connection.
- **HTTPClient.h**: This library is used for sending HTTP requests from the ESP32 to the server. It simplifies the process of creating, sending, and receiving HTTP requests, allowing the ESP32 to communicate the dryer's status to the server efficiently.

Key Functionalities

Wi-Fi Connection

The ESP32 connects to a predefined WiFi network using SSID and password credentials stored in the firmware. This network connection is essential for communicating with the server and sending status updates.

```
const char* ssid = "your network's ssid";
```

```
const char* password = "the corresponding password";
```

Limit Switch Monitoring

The firmware continuously monitors three GPIO pins connected to the limit switches. Each switch corresponds to a specific pin:

- switch1Pin on GPIO 14
- switch2Pin on GPIO 27
- switch3Pin on GPIO 33

The firmware checks whether each switch is engaged (LOW signal) or not (HIGH signal), indicating the dryer's status.

Check out the Wiring section above to see how the limit switches are wired with the ESP32.

Dryer Status Determination

The firmware uses a logical check to determine the dryer's state:

- If any of the limit switches are engaged (reading LOW), the dryer is off (isAnySwitchOn = 0).
- If none of the switches are engaged (all reading HIGH), the dryer is on (isAnySwitchOn = 1).

```
int isAnySwitchOn = switch1State || switch2State || switch3State ? 0 : 1;
```

LED Indicator

The onboard LED, connected to GPIO 2, provides visual feedback:

- The LED turns on when the dryer is running (no switches engaged).
- The LED turns off when the dryer is off (any switch engaged).
- During WiFi connection attempts, the LED blinks to indicate connection status.

Server Communication

The firmware sends the dryer's status to a server using an HTTP POST request. The server URL is predefined in the firmware:

- Status updates are sent to http://<server-ip>/value/, where <server-ip> is the IP address of the server.
- The firmware appends the dryer's status (1 or 0) to the server path, allowing the server to log the status change with a timestamp.

Reconnection Logic

If the ESP32 loses its WiFi connection, the firmware includes a reconnection routine that attempts to reconnect to the network. The LED blinks during this process to indicate connection attempts.

Code Structure

Setup Function (setup())

- Initializes GPIO pins for the limit switches and LED.
- Starts the Serial communication for debugging.
- Connects to the Wi-Fi network using the connectToWiFi() function.

Loop Function (loop())

- Reads the state of each limit switch.
- Determines the dryer status and updates the LED indicator.
- Forms the server path with the current status and sends an HTTP POST request.
- Checks Wi-Fi connectivity and attempts reconnection if necessary.
- Includes a delay to prevent excessive server requests.

WiFi Connection Function (connectToWiFi())

- Attempts to connect to the specified Wi-Fi network within a given timeout period.
- Uses the LED to indicate connection attempts.
- Exits the routine if the connection is not established within the timeout. (default timeout is 10 seconds)

Debugging

A DEBUG_MODE flag in the firmware allows for enabling or disabling debugging messages. When enabled, the firmware prints status messages and server responses to the Serial monitor, aiding in troubleshooting and verifying correct operation.

Server

The server is a crucial component of the Dryer Monitoring System, responsible for receiving data from the ESP32, updating the web interface, and logging the dryer's status. Built using Node.js and Express, the server provides the backend infrastructure that supports real-time monitoring and data storage. This section details the server's architecture, key functionalities, and code structure.

Overview

The server application manages the communication between the ESP32 and the web interface. It listens for incoming HTTP POST requests from the ESP32, processes the dryer status data, and updates the user interface accordingly. Additionally, the server logs each status change with a timestamp to maintain a historical record of dryer usage. This setup ensures that users can monitor their dryer in real time and access past usage data.

Key Functionalities

- **Receiving Dryer Status Updates**: The server listens for POST requests sent by the ESP32. Each request contains the current status of the dryer (on or off), which the server logs and uses to update the web interface.
- **Serving the Web Interface**: The server hosts the HTML, CSS, and JavaScript files that make up the web interface. It serves these files to clients (browsers) that connect to the server, allowing users to access the monitoring dashboard.
- **System Health Monitoring**: The server tracks the frequency of updates from the ESP32. If the updates do not arrive within the predefined SYS_DOWN_TIME threshold (10 seconds), the server flags the system as down, and this is reflected on the web interface.
- **Logging**: The server records each state change of the dryer in a log file. This log file includes the status (on or off) and a timestamp, providing a historical record that can be analyzed for usage patterns.

Code Structure

Environment Configuration (.env file)

- PORT: Defines the port on which the server listens for incoming connections. The default value is 6969.
- FORMAT_DATE: A flag to enable or disable human-readable date formatting. When set to true, timestamps are formatted for readability. The default is true.
- SYS_DOWN_TIME: Specifies the maximum time allowed between updates from the ESP32. If exceeded, the system is considered down. The default value is 10 seconds.

```
PORT=6969
FORMAT_DATE=true
SYS_DOWN_TIME=10
```

Server Initialization

- The server uses the express framework to handle HTTP requests and serve static files.
- The cors middleware is used to enable Cross-Origin Resource Sharing, allowing the web interface to request resources from the server without security restrictions.
- After setting up the Environment Configuration file, use *npm start* to start the server.

Routes and Endpoints

- **/**: Serves the main dashboard of the web interface, showing the current status of the dryer.
- **/faq**: Serves an FAQ page, providing information about the system.
- **/systemHealth**: Serves a page that displays the health status of the monitoring system.
- **/value/:v**: A POST endpoint that receives the dryer's status from the ESP32. The parameter :v represents the current status (1 for on, 0 for off).
- **/dryer**: A GET endpoint that returns the current state of the dryer and the last saved time. This is used to update the web interface in real time.

- **/health**: A GET endpoint that returns the system health status, indicating whether the monitoring system is functioning correctly.

Controller Functions

- **dryerController**: Handles incoming POST requests from the ESP32, updates the dryer status, and logs the change with a timestamp.
- **getDryerState**: Retrieves the current operational state of the dryer.
- **getSavedTime**: Returns the last recorded time when the dryer's status was updated.
- **getSystemHealth**: Provides the current health status of the monitoring system, used to detect issues like lost connectivity or hardware malfunctions.

Data Logging

- The server logs data using the fs (file system) module, appending each status update to a text file named by the current month and year (e.g., August2024.txt).
- The log entries follow the format: <state> <timestamp>, where 1 indicates the dryer is on, and 0 indicates it is off. The timestamp follows the ISO8601 standard.

0 2024-08-25T11:49:48
1 2024-08-25T11:49:49
0 2024-08-25T11:49:51
1 2024-08-25T11:49:52
0 2024-08-25T11:49:53
1 2024-08-25T11:49:54
0 2024-08-25T11:49:57

Example of August.txt

Key Files and Modules

- **server.js**: The main server script that initializes the Express app, configures routes, and starts the server.
- **funcs/dryer.js**: Contains functions related to handling dryer status updates, checking system health, and interacting with the ESP32.
- **funcs/datetime.js**: Provides utility functions for date and time operations, such as getting the current date and time, formatting timestamps, and converting Unix time to human-readable format.
- **funcs/storage.js**: Manages data logging, including appending new status updates to the monthly log files.

Conclusion

The Dryer Monitoring System is a practical and efficient solution for remotely monitoring the operational status of a dryer. By leveraging an ESP32 microcontroller, strategically placed limit switches, and a user-friendly web interface, the system provides real-time updates and historical data logging, enhancing both convenience and efficiency in managing dryer usage.

Initially, a vibration sensor was used to detect the dryer's operation. However, due to the washer being located next to the dryer and causing strong vibrations, the vibration sensor occasionally produced false triggers. To overcome this issue, the system was switched to using limit switches, which proved to be far more reliable in accurately detecting the dryer's status.

This documentation has covered all aspects of the system, from hardware installation and firmware setup to server configuration and user interface navigation. By following these guidelines, users can set up the system, understand its operation, and troubleshoot any issues that may arise.

Although the system does not perform analytics, the collected data can be analyzed separately to gain insights into usage patterns and optimize drying times, potentially saving energy and reducing operational costs. Future enhancements, such as integrating analytics directly into the system or expanding functionality to monitor other appliances, offer exciting possibilities for extending the system's capabilities.

Last revised in August 2024.