**Supplementary Information**

This Supporting Information provides technical documentation to assist educators, instructors, and makers in replicating, adapting, or expanding the Bunsen burner monitoring system. It includes descriptions of system components, circuit functions, and network configurations to facilitate implementation.

**Additional Files (provided separately)**

* Bill of Materials (BOM) (Excel).
* Arduino Source Code. Available on GitHub:
* PCB Design Files. Gerber files and schematics available on GitHub and PCBWay.
* 3D Printing Files. AutoCAD design files and stl files on GitHub.
* User Manuals. Quick-start and troubleshooting guide.

**Supplementary Demonstration Videos (on YouTube)**

* Video S1: Monitoring system usage. Sensor node detects burner use and begins countdown. LED signage and display node update the burner status.
* Video S2: System alarms. Timeout event triggers local alarms. Active sensor node shuts off the gas valve after the timeout period
* Video S3: Sensor node assembly.
* Video S4: Monitor hub assembly.
* Video S5: Display node assembly.
* Video S6: Relay module assembly.
* Video S6: System setup.

**Resource Links:**

* **GitHub repository**:

https://github.com/yanggaoemail/Bunsen-Burner-Monitoring-System.git

* **PCBWay project page**: https://www.pcbway.com/project/shareproject/A\_Modular\_WiFi\_enabled\_Bunsen\_Burner\_Monitoring\_System\_for\_Smart\_Laboratory\_2ba5f4b7.html
* **YouTube playlist**:

https://youtube.com/playlist?list=PL3LwpViqRsFmU12UfuHDb1hy4t22Aw7wt&si=8kA5nQrR-BTuAZvp

### Table of Contents

[Table of Contents 2](#_Toc212326082)

[1. System Overview 3](#_Toc212326083)

[2. Sensor Node Functionality 3](#_Toc212326084)

[2.1. Modes & Configuration 3](#_Toc212326085)

[2.1.2. Passive Mode 3](#_Toc212326086)

[2.1.3. Active Mode 3](#_Toc212326087)

[2.2. Wi-Fi Function 4](#_Toc212326088)

[2.2.1. Networking 4](#_Toc212326089)

[2.2.2. Application Protocol 4](#_Toc212326090)

[2.2.3. IP Addressing and Hub Discovery 5](#_Toc212326091)

[2.3. Additional sensor node functions 6](#_Toc212326092)

[3. Monitor hub functionality 6](#_Toc212326093)

[3.1. Role & capacity 6](#_Toc212326094)

[3.2. Wi-Fi function 7](#_Toc212326095)

[3.2.1. Monitor hub networking 7](#_Toc212326096)

[3.2.2. Application protocol 7](#_Toc212326097)

[3.2.3. IP addressing and node discovery 8](#_Toc212326098)

[3.3. Additional functions 9](#_Toc212326099)

[4. Desktop Display Node 10](#_Toc212326100)

[4.1. main functionality 10](#_Toc212326101)

[4.2. Wi-Fi function 11](#_Toc212326102)

[4.2.1. Display Node Networking 11](#_Toc212326103)

[4.2.2. Application Protocol 11](#_Toc212326104)

[4.2.3. IP addressing and hub discovery 12](#_Toc212326105)

[4.3. Additional functions 12](#_Toc212326106)

[5. Relay Module 13](#_Toc212326107)

# 1. System Overview

The Bunsen burner monitoring system is modular, comprising up to 16 sensor nodes, one monitor hub, and up to 16 desktop display nodes. The sensor node is the mission-critical component; the hub and display nodes are optional extensions that increase burner-status visibility across the workspace. All networked components communicate over Wi-Fi network. Because most workplace Wi-Fi networks use DHCP (Dynamic Host Configuration Protocol), device IP addresses are assigned dynamically and may change. Therefore, devices cannot be reliably identified by IP. Instead, each sensor and display node is identified by a unique device ID. We use a star (hub-and-spoke) application architecture: sensor and display nodes act as UDP clients that send to a single hub (UDP server) at the given IP/port. The hub validates payloads and replies with either an acknowledgment or the connected sensor status, addressed by device ID.

# 2. Sensor Node Functionality

## 2.1. Modes & Configuration

The sensor node operates in passive (basic) or active (optional) mode (Figure S1a-b), selectable by user configuration. To enable active mode, add a motorized gas valve, a relay module, and a 12 V-capable USB-PD (power delivery) adapter. Then set EleGasVlvEna = true in the source code, build and flash the firmware to the Arduino.

### 2.1.2. Passive Mode

Burner activity is detected by an infrared (IR) proximity sensor positioned behind the manual gas valve (Figure S1c). When the valve opens, the node: (1) starts a countdown with the preselected duration, (2) switches the front-panel LED from green (safe) to red (active), and (3) displays the remaining time on the integrated screen. If the burner remains active at timeout, a local audio-visual alert is triggered (LED flashing and buzzer beeping). The user can silence/reset by either

* momentarily blocking the IR sensor (simulating valve closure), or
* toggling the timer selector (left or right) to restart the countdown with a new preset time.

These resets do not interrupt burner use.

### 2.1.3. Active Mode

All passive functions are retained, with the addition of automatic gas shutoff via a motorized gas valve installed in series with the manual valve. After timeout, a 10-min grace interval allows the same reset procedure as the passive mode. If no action is taken, the motorized valve is actuated to close the gas line. During actuation, a 10-s lockout prevents further input to allow full closure. To safely reactivate after shutoff, the display instructs the user to

* close the manual valve, and
* toggle the timer selector twice (confirmation gesture).

This two-step procedure helps ensure deliberate resets and reduces the chance of false IR triggers (e.g., ambient light or motion), mitigating accidental reopening and unintended gas release.

A close-up of an electrical device

AI-generated content may be incorrect.

Figure S1. Sensor nodes configured as (a) passive and (b) active mode. (c) IR proximity sensor with the valve in the closed position.

## 2.2. Wi-Fi Function

2.2.1. Networking

Wi-Fi connectivity is optional for the sensor node. Enabling or disabling it requires no firmware or hardware changes. On startup, the node attempts a single Wi-Fi join. Regardless of Wi-Fi status, the node begins normal operation. A front-panel switch selects the reconnection mode: ‘A’ (automatic) or ‘M’ (manual). In ‘A’ mode, the node attempts to join the Wi-Fi every 60 s. In ‘M’ mode, the node makes a single Wi-Fi connection attempt at startup. When connected to Wi-Fi, the sensor node sends a status packet to the hub’s IP/port every 1 s.

2.2.2. Application Protocol

The sensor node’s status packet contains two unsigned 32-bit integers:

• Word 0: encodes the device ID and valve status (see Fig. S2 for the layout).

• Word 1: seconds since timeout (count-up).

A black background with a black square

AI-generated content may be incorrect.

Figure S2. Status words and bitfields from the sensor node to the monitor hub. (a) W0 contains sensor ID and valve status, (b) W1 is the timeout counter in seconds.

2.2.3. IP Addressing and Hub Discovery

Because many workplaces do not support static IPs for IoT devices, connected devices obtain an address via DHCP. Thus, the hub must join the network (obtain its IP) before programming nodes with the hub IP used for communications. Additionally, network gear may reboot or reassign IPs (~1-2 times/month in our experience), which can change the device IP address and break node-hub communication. To make recovery automatic, each node stores the current hub IP in on-chip EEPROM and follows a two-step check-up on startup:

* Try the compiled-in hub IP; if no acknowledgment arrives within 15 s, fall back to the EEPROM-stored hub IP.
* Update the EEPROM-stored hub IP only if (i) an acknowledgment is first received 15-45 s after startup and the current hub IP differs from the stored value, or (ii) an explicit IP-change command (See Table S2 for hub commands) is received from the hub and the address differs.

This write strategy avoids unnecessary EEPROM wear while still allowing the hub IP to be changed by flashing the firmware or by the hub command.

## 2.3. Additional sensor node functions

To provide environmental context for research, each device integrates an AHT20 temperature-humidity sensor. Readings are shown in a rectangular panel at the top-right of the display with °C and % for temperature and humidity, respectively.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Format** | **Description** |
| EleGasVlvEna | bool | true / false | Enables active gas shut-off valve when true; when false, system runs passive (audible/visual alarm only). |
| CodeVersion | char[] | Free text (e.g., “vX.Y”) | Source code version displayed on start. Updated at release only. |
| CoreVersion | char[] | Free text (e.g., “0.4.1”) | Arduino core version. |
| SenNum | int | Positive integer (e.g., 1-16) | Device ID for this sensor node; must be unique. |
| ssid | char[] | Wi-Fi SSID string | Network SSID. |
| pass | char[] | WPA/WPA2 passphrase | Wi-Fi password, consult workplace’s IT service for device registration. |
| serverIP | IPAddress | IPv4 address | Hub (server) IP to which clients send UDP datagrams. |
| UDPPort | unsigned int | 1-65535 | UDP port used by hub and nodes; must match on both ends. The 50505 is an unassigned port in the IANA dynamic/private range to avoid conflicts with well-known/registered services. |
| AlmTim[4] | int[4] | Array of 4 positive integers | Alarm timer presets time in minutes. |

Table S1. User-configurable firmware parameters for the sensor node.

# 3. Monitor hub functionality

## **3.1. Role & capacity**

The monitor hub collects status from **sensor nodes (up to 16) and send the status data to desktop display nodes (up to 16)** over the workplace **Wi-Fi. Additionally, the hub** can drive a **relay box** to illuminate external LED signage whenever any **Bunsen** burner is active. The hub display shows real-time node status, the network SSID, and the hub’s IP address. For each node, the status character is “-” (valve closed / inactive) or “!” (valve open / active). Node IDs are shown in **hexadecimal**.

An orange electronic device with white wires and text

AI-generated content may be incorrect.

Figure S3. Monitor hub with optional relay module for two LED signs. The IP address is blacked out.

## 3.2. Wi-Fi function

3.2.1. Monitor hub networking**.**

The hub requires Wi-Fi connectivity (no mode switch). While offline, the hub attempts to reconnect every 60 s. The node telemetry message (sensor nodes and display nodes) serves as the heartbeat to determine node liveness. If no message is received from a node for **> 5 s**, that node is marked **offline.**

### 3.2.2. Application protocol

The hub listens on the predefined UDP port (e.g., 50505). Upon receiving a message of valid length, the hub parses the content to the device ID and/or the timeout length, depending on the device ID. IDs 1-16 are sensor nodes, and IDs 17-32 are desktop display nodes. The hub then replies as follows:

* To sensors: a 1-byte control code 9 as acknowledgment (see Table S2 for control code definitions).
* To display nodes: a telemetry of three unsigned 32-bit integers:

(1) W0, node liveness bitmap (32 bits): bit *n* (0-31)= 1 if device ID *n+1* is online, 0 if offline.

(2) W1, valve-open bitmap (16 bits used): bit *n* (0-15) = 1 if sensor *n+1* valve open, 0 if closed; bits 16-31 are reserved (0).

(3) W2, timeout bitmap (16 bits used): bit *n* (0-15) = 1 if sensor *n+1* is in timeout, 0 normal operation; bits 16-31 are reserved (0).

A black background with a black square

AI-generated content may be incorrect.

Figure S4. Status words and bitfields from the monitor hub to the display nodes. (a) W0 contains sensor ID and valve status, (b) W1 is the timeout counter in seconds.

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Function** | **Notes** | |
| 0-7 | Unused | |  |
| 8 | Hub IP changed | | Let the node switch communications to that address. |
| 9 | Acknowledgement | | Confirms receipt of valid message. |

Table S2. Control codes from the monitor hub to the nodes.

### 3.2.3. IP addressing and node discovery

To mitigate DHCP IP address changing after device or network reboots, the hub stores per-node liveness and IP address in the EEPROM. On each valid UDP datagram from a node, the hub compares the sender’s IP with the EEPROM-stored address. If they differ, the hub updates the EEPROM entries for both the liveness state and IP address. During the 25 s - 60 s after startup, the hub compares current node liveness with the saved state. If a node was previously online but has not been seen in this time window, the hub sends control code 8 (hub IP changed) to the node’s last known IP, prompting the node to reestablish communication and refresh the stored hub IP. The hub does not update its EEPROM upon changes in the node liveness to limit EEPROM wear.

## 3.3. Additional functions

To enable remote, read-only monitoring from laptops and smartphones, the hub runs a lightweight **HTTP server** (port 80). Access is protected by a single shared password stored in the firmware. The hub does not maintain per-client sessions; it validates each request. With the correct password, the hub returns an HTML status page (Fig. S5) with a table showing sensor ID, valve status, and a timeout counter (HH:MM). Remote access is available via the workplace VPN.

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure S5. Web interface for remote monitoring in browsers of (a, b) smartphone and (c, d) computer. IP addresses are blacked out.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Format** | **Description** |
| CodeVersion | char[] | Free text (e.g., “vX.Y”) | Source code version displayed on start. Updated at release only. |
| CoreVersion | char[] | Free text (e.g., “0.4.1”) | Arduino core version. |
| ssid | char[] | Wi-Fi SSID string | Network SSID. |
| pass | char[] | WPA/WPA2 passphrase | Wi-Fi password, consult workplace’s IT service for device registration. |
| UDPPort | unsigned int | 1-65535 | UDP port used by hub and nodes; must match on both ends. The 50505 is an unassigned port in the IANA dynamic/private range to avoid conflicts with well-known/registered services. |
| RmtPsd | char[] | ASCII string (case-sensitive) | Password for the hub’s HTTP status page |

Table S3. User-configurable firmware parameters for the monitor hub.

# 4. Desktop Display Node

## 4.1. main functionality

The desktop display node receives the sensor status data from the monitor hub via the workplace Wi-Fi network. The real-time burner status is shown on the e-ink screen of the display node. Three sensor statuses can be displayed: ‘close’ for closed valve (inactive burner), ‘open’ for opened valve (active burner), ‘TMO’ for alarm timeout status in active burners. If a sensor timeout is received, the backlights of the display node would flash to create a more visible alert to the occupants in the room.

A close up of a device

AI-generated content may be incorrect.

Figure S6. Desktop display node. (a) Front view showing two active Bunsen burners. (b) Rear view of the unit.

## 4.2. Wi-Fi function

### 4.2.1. Display Node Networking

The display node functions similarly to the sensor nodes in the network application. When connected to Wi-Fi, the display node sends a status packet to the hub’s IP/port every 2 s.

### 4.2.2. Application Protocol

The display node’s status packet contains one unsigned 32-bit integer encoding the device ID.

A black background with a black square

AI-generated content may be incorrect.

Figure S7. Status words and bitfields from the display node to the monitor hub containing the display node ID.

### 4.2.3. IP addressing and hub discovery

The display node uses the same DHCP-based IP addressing and hub discovery procedure as the sensor node. See Section 2.2.3 for details.

## 4.3. Additional functions

The display node is mobile and powered by a lithium-ion cell, enabling uninterrupted operation without an external supply. To report an accurate battery state, the node microcontroller monitors the battery voltage via the on-board ADC (analog-to-digital converter). Voltage is derived from the ADC reading by a user-editable linear calibration:

where is the ADC reading, is the y-axis intercept (BatFitoffset), is the slope (BatFit\_slope), and is the battery voltage up-shift when 5 V DC is connected.   
The displayed battery percentage is then derived from using the measured battery voltage limits BatMaxVol and BatMinVol.

A black background with a black square

AI-generated content may be incorrect.

Figure S8. Linear calibration of battery voltage versus ADC reading.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Format** | **Description** |
| CodeVersion | char[] | Free text (e.g., “vX.Y”) | Source code version displayed on start. Updated at release only. |
| CoreVersion | char[] | Free text (e.g., “0.4.1”) | Arduino core version. |
| EinkNum | int | Positive integer (e.g., 1-16) | Device ID for this display node; must be unique among display nodes, can be the same as sensor nodes. |
| ssid | char[] | Wi-Fi SSID string | Network SSID. |
| pass | char[] | WPA/WPA2 passphrase | Wi-Fi password, consult workplace’s IT service for device registration. |
| serverIP | IPAddress | IPv4 address | Hub (server) IP to which clients send UDP datagrams. |
| UDPPort | unsigned int | 1-65535 | UDP port used by hub and nodes; must match on both ends. The 50505 is an unassigned port in the IANA dynamic/private range to avoid conflicts with well-known/registered services. |
| BatFit\_slope | int[4] | Float number unit V | Slope of the linear fit converting ADC readings to battery voltage. Calibrate per batch of Li-ion battery. |
| BatFitoffset | float | Float number unit V | y-axis intercept of the linear fit converting ADC readings to battery voltage. |
| BatMaxVol | float | Float number unit V | Maximum measured battery voltage. |
| BatMinVol | float | Float number unit V | Minimum measured battery voltage. |
| BatDCOffSet | float | Float number unit V | Compensation applied when 5 V DC is connected (corrects the reading up-shift under external power). |
| SenNam | Char[][3] | 2 ASCII chars (e.g., “AB”, “CD”) | Two-character labels are shown in place of numeric sensor IDs (IDs start at 1). If a label is empty (""), the numeric ID is displayed. |

Table S4. User-configurable firmware parameters for the display node.

# 5. Relay Module

The relay box is a snap-on module to allow the sensor node or monitor hub to control power-demanding or higher voltage (> 5V) peripherals. The control side of the relay board connects to the Arduino controller. The isolated side of the relay board is powered by an adjustable USB-PD (power delivery) trigger module, allowing the user to select the appropriate voltage (e.g. 12 V for motorized gas valve, 5 V for LED signs).