Temperature Monitoring System(MAX6675 K-thermocouple sensor) - Code Report

Overview

This application runs on the **pcbcupid GLYPH C6 – ESP32C6 DEV Board** and implements a wireless temperature monitoring system that reads temperature data from a MAX6675 K-thermocouple sensor, displays the readings on a TM1637 7-segment display, and transmits the data to a remote receiver using ESP-NOW protocol.

Hardware Components

- **pcbcupid GLYPH C6 – ESP32C6 DEV Board** - Main processing unit
- **MAX6675 thermocouple module** - Temperature sensor (pins 18, 5, 19)
- **TM1637 4-digit display** - Visual temperature display (pins 2, 4)
- **Thermocouple probe** - Connected to MAX6675 for temperature measurement

Key Features

Temperature Monitoring

- Continuously reads temperature from MAX6675 thermocouple in Celsius
- Validates sensor readings (filters out invalid values like NaN, negative, or >1000°C)
- Displays temperature on 7-segment display as integer values

Alert System

- Triggers high-temperature alerts when reading exceeds 50°C threshold
- Alternates between two alert display modes:
    - Blinking actual temperature value (3 times)
    - Blinking "ALRT" text pattern (3 times)
- Provides clear visual indication of dangerous temperature conditions

Wireless Communication

- Uses ESP-NOW protocol for low-latency, peer-to-peer communication
- Automatically scans WiFi channels 1-13 to locate receiver device
- Implements acknowledgment system to verify successful data transmission
- Includes automatic reconnection logic when communication fails

Reliability Features

- **Channel Discovery**: Automatically finds the receiver on any WiFi channel
- **Connection Recovery**: Re-scans and reconnects after 3 consecutive failed transmissions
- **Data Validation**: Filters invalid temperature readings
- **Status Monitoring**: Tracks transmission success/failure with detailed serial output

Data Structure

The system transmits a structured message containing:

- type: Message identifier (1 = Temperature data)
- temperatureC: Actual temperature reading in Celsius

Operation Flow

1. **Initialization**: Scans for receiver device across WiFi channels
2. **Main Loop**:
    - Reads temperature from sensor
    - Updates display with current reading or alert pattern
    - Transmits data to receiver via ESP-NOW
    - Monitors transmission success
3. **Error Handling**: Automatically reconnects if communication fails

Technical Specifications

- **Update Rate**: 1-second intervals
- **Temperature Range**: 0-1000°C (validated range)
- **Alert Threshold**: 50°C
- **WiFi Channels**: Scans channels 1-13
- **Communication Protocol**: ESP-NOW (low-latency mesh networking)

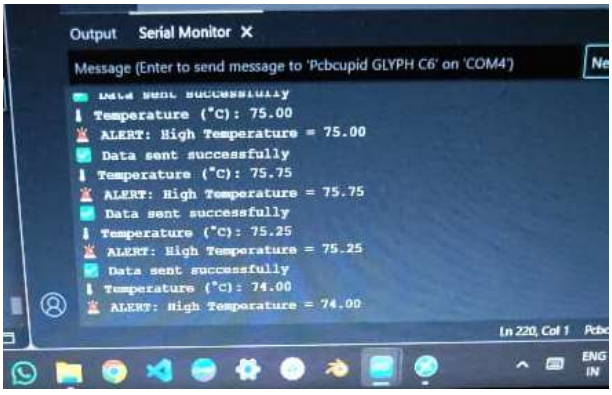
Use Cases

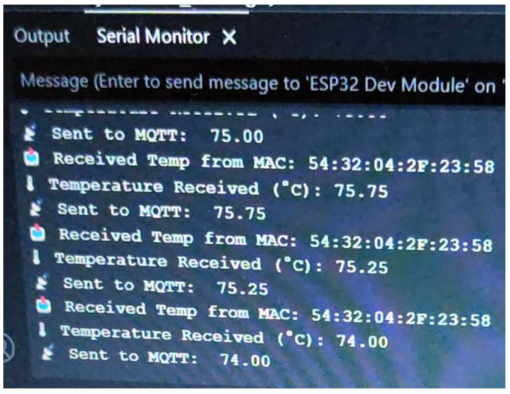
This system is ideal for:

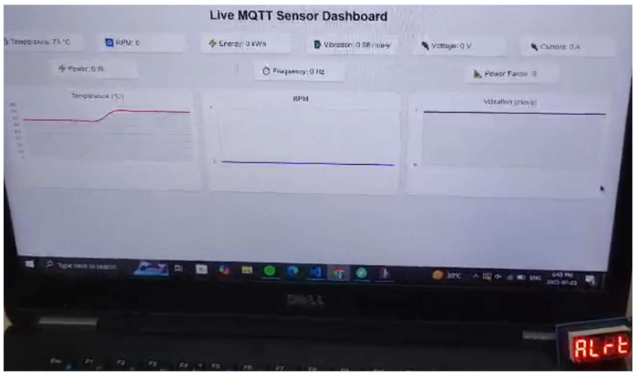
- Industrial temperature monitoring
- Equipment overheating protection
- Remote temperature sensing applications
- Safety-critical temperature alerts

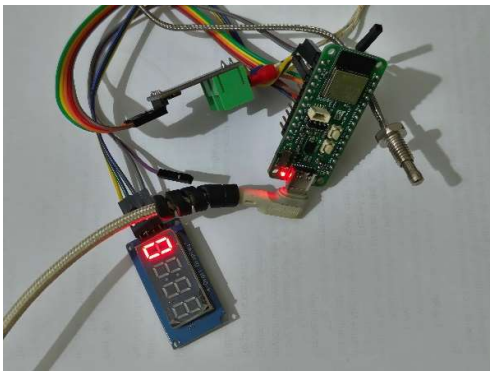
Output Documentation

Detailed images showing the system's operation, serial monitor output, display patterns, and alert behaviors are included to demonstrate the practical implementation and real-world performance of the temperature monitoring system.









```
uint8_t type;     // 1=temp, 2=rpm, 3=vibration, 4=power
float val1;       // voltage
float val2;       // current
float val3;       // power
float val4;       // energy
float val5;       // frequency
```

Serial Monitor

Message (Enter to send message to 'Pcbcupid GLYPH C6' on 'COM14')

No Line Ending | 115200 baud

```
18:49:46.292 -> Data sent successfully
18:49:47.372 -> V=260.40V, I=0.00A, P=0.00W, E=0.00kWh, F=50.00Hz, PF=0.00
18:49:47.575 -> Data sent successfully
18:49:48.630 -> V=253.00V, I=0.00A, P=0.00W, E=0.00kWh, F=50.00Hz, PF=0.00
18:49:48.835 -> Data sent successfully
18:49:49.888 -> V=248.70V, I=0.00A, P=0.00W, E=0.00kWh, F=50.00Hz, PF=0.00
18:49:50.106 -> Data sent successfully
18:49:51.165 -> V=247.30V, I=0.00A, P=0.00W, E=0.00kWh, F=50.00Hz, PF=0.00
18:49:51.369 -> Data sent successfully
18:49:52.435 -> V=246.90V, I=0.00A, P=0.00W, E=0.00kWh, F=50.00Hz, PF=0.00
```

```
PubSubClient client(espC

// Define unique message
typedef struct struct_me
  uint8_t type;  // 1=te
  float val1;    // temp
  float val2;    // curr
```

Serial Monitor

```
18:49:52.435 -> Received from
18:49:52.435 -> Power Data:
18:49:52.435 -> Voltage: 246.90
18:49:52.435 -> Energy: 0.00 kWh
18:49:52.435 -> Sent to MQTT:
18:49:52.435 -> {"temperature":
18:49:52.467 -> Received from
18:49:52.467 -> Temp: 30.75
18:49:52.467 -> Sent to MQTT:
18:49:52.467 -> {"temperature":
```