Code Logic Implementation

This section explains how the MicroPython program for the IoT Flood Monitoring System works step by step on the Raspberry Pi Pico W.

---

1. Library Import and Configuration

- The program imports required modules:
  - `machine`, `Pin`, `time_pulse_us`, `time` → to control GPIO pins and measure ultrasonic echo pulse.
  - `network` → to connect the Pico W to WiFi.
  - `ubinascii` and `machine.unique_id()` → to generate a unique MQTT client ID.
  - `socket` and `struct` → for low-level TCP and HTTP communication (MQTT + ThingSpeak).
  - `ssd1306` → to drive the OLED display via I2C.
- WiFi credentials (`WIFI_SSID`, `WIFI_PASSWORD`) are defined.
- MQTT configuration:
  - Server: `test.mosquitto.org`
  - Port: `1883`
  - Topic: `"wokwi/flood/monitor"`
- ThingSpeak configuration:
  - Host: `api.thingspeak.com`
  - Write API key is stored in `THINGSPEAK_API_KEY`.

---

2. Embedded MQTT Client

- A lightweight `MQTTClient` class is implemented inside the code (no external library needed).
- Important methods:
  - `connect()`:
    - Creates a TCP socket to the MQTT server.
    - Builds and sends an MQTT CONNECT packet.
  - `publish(topic, msg)`:
    - Builds an MQTT PUBLISH packet and sends JSON data containing:
      - `"level"` → current water level (distance in cm)
      - `"status"` → SAFE / WARNING / !!CRIT!!
  - `disconnect()`:
    - Sends DISCONNECT packet and closes the socket.

This client is used later to send flood status messages to the MQTT broker, which can be viewed in the MyMQTT app.

3. Hardware Setup

- Ultrasonic sensor:
    - `TRIGGER` pin → GPIO 14 (output)
    - `ECHO` pin → GPIO 15 (input)
- Buzzer:
    - `BUZZER` pin → GPIO 16 (output)
    - Initially set such that the buzzer is OFF at start.
- LEDs:
    - `RED_LED` → GPIO 17
    - `GREEN_LED` → GPIO 18

      These are controlled based on the flood status.
- OLED Display:
    - I2C bus created on `scl = GPIO 1`, `sda = GPIO 0`.
    - Tries to initialize `ssd1306.SSD1306_I2C(128, 64, I2C_BUS)`.
    - If OLED is not detected, it prints an error and continues without display support.

4. WiFi Connection (`connect_wifi()`)

- Activates the WiFi interface in station mode.
- Attempts to connect to the configured WiFi network.
- Waits up to 10 seconds for connection.
- If successful, prints the assigned IP address.
- Returns `True` if connected, otherwise `False`.

This ensures the Pico W is online before using MQTT and ThingSpeak.

5. Distance Measurement (`get_distance()`)

- Sends a short trigger pulse:
    - Sets TRIGGER low (2 µs), high (10 µs), then low again.
- Uses `time_pulse_us(ECHO, 1, 30000)` to measure the duration of the echo pulse (in microseconds).
- Converts the time into distance using the speed of sound:

$$\text{distance (cm)} = \frac{\text{duration} \times 0.0343}{2}$$

- If the pulse fails or times out, returns `-1` to indicate error.

This function provides the current distance from the sensor to the water surface.

---

6. LED Control (`control_leds(status)`)

- Based on the `status` string:
  - `"!!CRIT!!"`:
    - Red LED ON
    - Green LED OFF
  - `"WARNING"`:
    - Red LED ON
    - Green LED OFF
  - `"SAFE"`:
    - Red LED OFF
    - Green LED ON
  - Any other status (e.g., `"ERROR"`):
    - Both LEDs OFF

This gives a clear visual indication of the flood status.

---

7. OLED Display Update (`update_screen(dist, status, mqtt_ok)`)

- If the OLED is available:
  - Clears the screen.
  - Displays title: **"FLOOD ALERT"**.
  - Draws a horizontal line as a separator.
  - Shows:
    - MQTT status → `"MQTT: OK"` or `"MQTT: --"`.
    - Water level as `"Level: XX.X cm"`.
    - Current status string (SAFE / WARNING / !!CRIT!! / ERROR).
  - Updates the display with `display.show()`.

The OLED provides a local textual interface showing current system state.

---

8. Sending Data to ThingSpeak (`send_to_thingspeak(level, status_msg)`)

- Checks if a valid API key is set.
- Resolves the ThingSpeak host (`api.thingspeak.com`) and opens a TCP socket on port 80.
- Constructs an HTTP GET request in the form:
  - `/update?api_key=KEY&field1=LEVEL&field2=STATUS`

- Sends the request and closes the socket.
- Prints `"ThingSpeak Updated"` on success or an error message on failure.

ThingSpeak is used for cloud logging and graphing of water level and status.

---

9. System Startup and MQTT Initialization

- Prints a startup message: `"--- STARTING COMPLETE SYSTEM ---"`.
- Calls `connect_wifi()` to connect to WiFi.
- Creates an `MQTTClient` using:
  - Client ID: `"pico-" + <unique_id>`.
  - Server: `test.mosquitto.org`.
  - Port: `1883`.
- If WiFi is OK, tries to connect to MQTT and sets `mqtt_connected = True` on success.
- Two timers are initialized:
  - `last_pub_mqtt` → for periodic MQTT publishing.
  - `last_pub_thingspeak` → for periodic ThingSpeak updates.

---

10. Main Control Loop

The `while True:` loop continuously performs the following tasks:

10.1 Read Water Level

- Calls `get_distance()` to measure the current distance.
- If the distance is valid (`dist > 0`), the program proceeds to evaluate the flood status; otherwise, it treats it as a sensor error.

10.2 Determine Status and Control Buzzer

Based on the measured distance:

- **Critical Level (very near, high water)**
  - Condition: `dist < 8`
  - Status: `"!!CRIT!!"`
  - Buzzer pattern:
    - Turn buzzer ON and OFF with `time.sleep(0.3)` between changes
    - Produces a sustained alarm feel
- **Warning Level (medium distance)**
  - Condition: `dist < 15` and `dist >= 8`
  - Status: `"WARNING"`
  - Buzzer pattern:

        - Turn buzzer ON/OFF with a shorter delay `time.sleep(0.1)`
        - Produces a faster beeping warning
- **Safe Level (water far away / low)**
    - Condition: `dist >= 15`
    - Status: `"SAFE"`
    - Buzzer: kept OFF (no sound)

These thresholds simulate safe, warning, and flood-critical levels.

10.3 Update LEDs

- Calls `control_leds(status)` with the current status.
    - SAFE → Green LED ON
    - WARNING / CRIT → Red LED ON
    - ERROR → both OFF

10.4 MQTT Publishing (Every 10 Seconds)

- Checks if at least 10 seconds have passed since `last_pub_mqtt`.
- If MQTT is not currently connected but WiFi is OK, it attempts to reconnect.
- When connected:
    - Builds a JSON message:
      `{"level": dist_value, "status": "STATUS"}`
    - Publishes this message to the configured MQTT topic.
    - Updates `last_pub_mqtt` on success.
- If an error occurs, it marks `mqtt_connected = False` and tries to disconnect cleanly.

This allows external subscribers (like MyMQTT app) to see live flood status updates.

10.5 ThingSpeak Update (Every 16 Seconds)

- Checks if at least 16 seconds have passed since `last_pub_thingspeak`.
- If WiFi is connected:
    - Calls `send_to_thingspeak(dist, status)` to log the current level and status.
    - Updates `last_pub_thingspeak`.

The longer interval respects ThingSpeak's free account minimum update interval (~15 seconds).

10.6 OLED Screen Update

- Calls `update_screen(dist, status, mqtt_connected)` to show:
    - Current water level
    - MQTT connection status
    - Flood status

10.7 Error Handling for Sensor

- If `dist` is `-1` (measurement failed):
    - Sets `status = "ERROR"`.
    - Calls `control_leds(status)` which turns LEDs OFF.
    - Prints `"Sensor Error"` and waits briefly.

10.8 Loop Delay

- A small delay `time.sleep(0.2)` is used at the end of each loop iteration to stabilize the loop and avoid overloading CPU and network.