# □ Subtask: "The Plankton Whisper"

*“Before you speak in light, you must learn to whisper.”*

## Objective

Create a system that changes its "Mood" based on incoming commands. You will send a simple text message and a heartbeat speed via Serial (JSON). The screen displays the text, and the LED changes its blinking speed. Refer to the Freertos (https://www.freertos.org/Documentation/00-Overview) Documentation.

---

## WOKVI HARDWARE SETUP

- **Board:** ESP32 Devkit V1
- **Display:** SSD1306 OLED (I2C) - SDA to GPIO 21, SCL to GPIO 22.
- **LED:** Connect to GPIO 2 (Heartbeat).

---

## THE CHALLENGE

### Task 1: The Ear (Input)

- Listens for JSON: `{"msg": "Safe", "delay": 1000}`
- Extracts the text and the delay number.
- Sends this package to the Queue.

### Task 2: The Face (OLED)

- Waits for data from the Queue.
- When data arrives, it clears the screen and prints the new `msg` in large text.
- Updates a **global variable** `currentDelay` so the heartbeat task knows how fast to blink.

### Task 3: The Heart (LED)

- Blinks the LED at the speed of `currentDelay`.
- *Challenge:* This task must never stop running, even when the screen is updating.

---

## SKELETON CODE (Copy & Paste into Wokvi)

C++

```
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ArduinoJson.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// RTOS Handles
QueueHandle_t commandQueue;
TaskHandle_t t_Input;
TaskHandle_t t_Display;
TaskHandle_t t_Blink;

// Shared Global Variable (protected by logic, or use a Mutex if you want extra points!)
int currentDelay = 1000; // Default 1 second blink

// Data Structure for the Queue
struct Command {
  char text[20];
  int blinkRate;
};

// ==========================================
// TASK 1: THE HEART (Blink LED)
// ==========================================
void HeartTask(void *pvParameters) {
  pinMode(2, OUTPUT);
  for (;;) {

    // TODO: Using the global variable(currentDelay) write a task to blink the led(on for 100ms) in intervals of currentDelay

  }
}

// ==========================================
// TASK 2: THE EAR (Serial Input)
// ==========================================
void InputTask(void *pvParameters) {
  Serial.begin(115200);

  for (;;) {
    if (Serial.available() > 0) {
      String input = Serial.readStringUntil('\n');
      input.trim();

      // JSON Parsing (Simplified)
      StaticJsonDocument<200> doc;
      DeserializationError error = deserializeJson(doc, input);

      if (!error) {
        Command cmd;

        // TODO: Extract data from JSON
        // 1. Copy doc["msg"] into cmd.text using strlcpy
        // 2. Copy doc["delay"] into cmd.blinkRate




        // TODO: Send 'cmd' to 'commandQueue'
        Serial.println("Command sent to queue!");
      } else {
        Serial.println("JSON Error");
      }
    }
    vTaskDelay(50 / portTICK_PERIOD_MS); // Yield to other tasks
  }
}

// ==========================================
// TASK 3: THE FACE (OLED Display)
// ==========================================
void DisplayTask(void *pvParameters) {
  Command receivedCmd;

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    for(;;);
  }

  // Initial Screen
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 20);
  display.println("Waiting...");
  display.display();

  for (;;) {
    // TODO: Receive from Queue and wait here indefinitely (portMAX_DELAY) until a message arrives
    if () {

      // TODO: Update the Global Variable for the Heart Task


      // Updates the Screen
      display.clearDisplay();
      display.setCursor(0, 20);

      // TODO: Print the text received from the queue


      display.display();
      Serial.println("Screen Updated.");
    }
  }
}

void setup() {
  // TODO: Create Queue of size 5, element size = sizeof(Command)


  // TODO: Create Tasks

}

void loop() {}
```

## Test Commands (Type these in Serial)

**1. Panic Mode (Fast blink, Urgent text):**

JSON

```json
{"msg": "DANGER!", "delay": 100}
```

**2. Calm Mode (Slow blink, Calm text):**

JSON

```json
{"msg": "Safe Reef", "delay": 2000}
```

**3. Standard Mode:**

JSON

```json
{"msg": "Hello", "delay": 500}
```

## Submission

**1.** Construct the circuit on Wokwi (https://wokwi.com/projects/new/esp32). **2.** Complete the code provided to achieve the desired result. **3.** Test the program with the test commands shown above( you can also use custom commands with the same format). **4.** Submit your code and a video recording of the testing to a github repo(this same repo can be used for the final hackathon).