You:

#include <ESP8266WiFi.h> #include <ESP8266WebServer.h> #include <Servo.h> #include <ESP8266HTTPClient.h> const char* ssid = "KRAKEN"; // Enter your Wi-Fi SSID here const char* password = "Ramboo1234@#"; // Enter your Wi-Fi password here ESP8266WebServer server(80); Servo servo; // Create a servo object int servoPin = 2; // D4 (GPIO2) for the servo control int servoOpenAngle = 90; // Adjust as needed int servoClosedAngle = 0; // Adjust as needed unsigned long lastFeedingTime = 0; // Variable to store the last feeding time unsigned long feedingInterval = 3600; // Default feeding interval in seconds (1 hour) unsigned long feedingFrequency = 3600; // Default feeding frequency in seconds (1 hour) bool isServoOn = false; // Flag to track the servo status bool isWebcamOn = false; // Flag to track the webcam status const char* webcamUrl = "http://192.168.0.104:8080"; // Replace with your IP webcam URL WiFiClient client; // Create a WiFiClient object for HTTPClient HTTPClient http; unsigned long lastWebcamRefresh = 0; // Declare the variable unsigned long webcamRefreshInterval = 10000; // Refresh the webcam feed every 10 seconds void setup() { Serial.begin(115200); delay(100); Serial.println("Connecting to "); Serial.println(ssid); // Connect to your local Wi-Fi network WiFi.begin(ssid, password); // Check if Wi-Fi is connected to the network while (WiFi.status() != WL_CONNECTED) { delay(1000); Serial.print("."); } Serial.println(""); Serial.println("WiFi connected..!"); Serial.print("Got IP: "); Serial.println(WiFi.localIP()); server.on("/", handle_OnConnect); server.on("/servoon", handle_servo_on); server.on("/servooff", handle_servo_off); server.on("/togglewebcam", handle_toggle_webcam); server.on("/updateinterval", handle_update_interval); server.on("/updatefrequency", handle_update_frequency); server.onNotFound(handle_NotFound); servo.attach(servoPin); server.begin(); Serial.println("HTTP server started"); } void loop() { server.handleClient(); if (isServoOn) { // Check if it's time to feed the pet based on the feeding interval if (millis() - lastFeedingTime >= feedingFrequency * 1000) { // Perform the feeding action servo.write(servoOpenAngle); delay(1000); // Adjust as needed servo.write(servoClosedAngle); // Update the last feeding time lastFeedingTime = millis(); } } // Periodically refresh the webcam feed if (isWebcamOn && millis() - lastWebcamRefresh >= webcamRefreshInterval) { refreshWebcamFeed(); lastWebcamRefresh = millis(); } } void refreshWebcamFeed() { http.begin(client, webcamUrl); // Use the WiFiClient object int httpCode = http.GET(); if (httpCode == HTTP_CODE_OK) { // Successfully retrieved the webcam feed, do nothing } else { Serial.println("Failed to fetch the webcam feed."); } http.end(); } void handle_OnConnect() { Serial.print("Servo Status: "); Serial.println(isServoOn ? "ON" : "OFF"); Serial.print("Webcam Status: "); Serial.println(isWebcamOn ? "ON" : "OFF"); String response = SendHTML(isServoOn, isWebcamOn, feedingFrequency); server.send(200, "text/html", response); } void handle_servo_on() { servo.write(servoOpenAngle); Serial.println("Servo Status: ON"); isServoOn = true; String response = SendHTML(true, isWebcamOn, feedingFrequency); server.send(200, "text/html", response); } void handle_servo_off() { servo.write(servoClosedAngle); Serial.println("Servo Status: OFF"); isServoOn = false; String response = SendHTML(false, isWebcamOn, feedingFrequency); server.send(200, "text/html", response); } void handle_toggle_webcam() { isWebcamOn = !isWebcamOn; Serial.print("Webcam Status: "); Serial.println(isWebcamOn ? "ON" : "OFF"); String response = SendHTML(isServoOn, isWebcamOn, feedingFrequency); server.send(200, "text/html", response); } void handle_update_interval() { if (server.hasArg("interval")) { feedingFrequency = server.arg("interval").toInt(); } String response = SendHTML(isServoOn, isWebcamOn, feedingFrequency); server.send(200, "text/html", response); } void handle_update_frequency() { if (server.hasArg("frequency")) { feedingFrequency = server.arg("frequency").toInt(); } String response = SendHTML(isServoOn, isWebcamOn, feedingFrequency); server.send(200, "text/html", response); } void handle_NotFound() { server.send(404, "text/plain", "Not found"); } String SendHTML(bool servoStatus, bool webcamStatus, unsigned long currentFrequency) { String ptr = "<!DOCTYPE html> <html>\n"; ptr += "<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n"; ptr += "<title>Pet Feeder Control</title>\n"; ptr += "<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n"; ptr += "body{margin-top: 50px; background-color: #f4f4f4;} h1 {color: #444444; margin: 50px auto 30px;} h2, h3 {color: #1abc9c; margin: 30px auto;} h3 {margin-bottom: 10px;}\n"; ptr += ".button {display: block; width: 200px; background-color: #1abc9c; border: none; color: white; padding: 13px 30px; text-decoration: none; font-size: 20px; margin: 0px auto 35px; cursor: pointer; border-radius: 4px;}\n"; ptr += ".button.button-on {background-color: #16a085;}\n"; ptr += ".button.button-on:active {background-color: #16a085;}\n"; ptr += ".button.button-off {background-color: #e74c3c;}\n"; ptr += ".button.button-off:active {background-color: #c0392b;}\n"; ptr += ".menu {display: flex; justify-content: space-around; background-color: #333; padding: 10px;}\n"; ptr += ".menu a {color: #fff; text-decoration: none; font-size: 18px;}\n"; ptr += ".menu a:hover {text-decoration: underline;}\n"; ptr += "p {font-size: 14px; color: #888; margin-bottom: 10px;}\n"; ptr += ".iframe-container {height: 100vh;}\n"; // Updated container height to fill the viewport height ptr += "iframe {width: 100%; height: 100%; border: 0;}\n"; // Updated CSS for full-screen iframe ptr += ".left {float: left; width: 50%;}\n"; ptr += ".right {float: right; width: 50%;}\n"; ptr += "input {width: 60px; text-align: center;}\n"; ptr += "form {margin-top: 20px;}\n"; ptr += "footer {margin-top: 50px; color: #777;}\n"; ptr += "</style>\n"; ptr += "</head>\n"; ptr += "<body>\n"; // Navigation Menu ptr += " Feeder | Webcam | Feeding Time | About "; // Main Content ptr += "<h1>Pet Feeder</h1>\n"; ptr += "<section id='controls'>"; ptr += "<h2>Controls</h2>"; ptr += " Servo Status: "; ptr += servoStatus ? "ON" : "OFF"; ptr += " "; ptr += " Webcam Status: "; ptr += webcamStatus ? "ON" : "OFF"; ptr += " "; ptr += "" : "button-on' href='/servoon'>"; ptr += servoStatus ? "Turn OFF" : "Turn ON"; ptr += "\n"; // Toggle Webcam Button ptr += "<form action='/togglewebcam' method='GET'>"; ptr += "<button class='button "; ptr += webcamStatus ? "button-off'>" : "button-on'>"; ptr += webcamStatus ? "Disable Webcam" : "Enable Webcam"; ptr += "</button></form>\n"; ptr += "</section>"; ptr += "<section id='webcam'>"; // Centered ptr += "<h2>Webcam Feed</h2>"; if (webcamStatus) { // Adjusted CSS for center positioning ptr += " "; ptr += "<iframe src='" + String(webcamUrl) + "' frameborder='0'></iframe>\n"; ptr += " "; } else { ptr += " Webcam is disabled. "; } ptr += "</section>"; ptr += "<section id='feeding'>"; ptr += "<h2>Feeding Time</h2>\n"; ptr += " Current Feeding Frequency: " + String(currentFrequency) + " seconds \n"; ptr += "<form action='/updatefrequency' method='GET'><input type='number' name='frequency' value='" + String(currentFrequency) + "'><input type='submit' value='Update'></form>\n"; ptr += "</section>"; ptr += "<section id='about'>"; ptr += "<h2>About</h2>\n"; ptr += " This IoT Mini project controls a pet feeder remotely using a web interface. It allows you to feed your pet on a schedule and view a live webcam feed of your pet. "; ptr += " Team Members: ALAN, BIJALI, LITTY, NANDHANA \n"; ptr += "</section>"; ptr += "<footer>Designed with love by alan_cyril for IoT Mini Project - Version 5.4 (stable)</footer>\n"; ptr += "<script>\n"; ptr += "function toggleControls() {\n"; ptr += "var servoStatusElement = document.getElementById('servo-status');\n"; ptr += "var webcamStatusElement = document.getElementById('webcam-status');\n"; ptr += "var controlsButtonElement = document.getElementById('controls-button');\n"; ptr += "var currentServoStatus = servoStatusElement.innerHTML;\n"; ptr += "var currentWebcamStatus = webcamStatusElement.innerHTML;\n"; ptr += "var newServoStatus = (currentServoStatus === 'OFF') ? 'ON' : 'OFF';\n"; ptr += "var newWebcamStatus = (currentWebcamStatus === 'OFF') ? 'ON' : 'OFF';\n"; ptr += "servoStatusElement.innerHTML = newServoStatus;\n"; ptr += "webcamStatusElement.innerHTML = newWebcamStatus;\n"; ptr += "controlsButtonElement.innerHTML = newServoStatus + ' | ' + newWebcamStatus;\n"; ptr += "// Additional logic to send requests to the server to toggle the servo and webcam status\n"; ptr += "// You can use AJAX or fetch to make asynchronous requests to the server here\n"; ptr += "}\n"; ptr += "</script>\n"; ptr += "</body>\n"; ptr += "</html>\n"; return ptr; } // ... (Rest of the code remains unchanged) explain in detail

---

ChatGPT:

Certainly! Let's break down the provided Arduino code step by step:

Libraries

```
cpp
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <Servo.h>
#include <ESP8266HTTPClient.h>
```

- **ESP8266WiFi.h:** Library for connecting ESP8266 to Wi-Fi.
- **ESP8266WebServer.h:** Library for handling HTTP requests on ESP8266.
- **Servo.h:** Library for controlling a servo motor.
- **ESP8266HTTPClient.h:** Library for making HTTP requests.

Wi-Fi Configuration

```
cpp
const char* ssid = "KRAKEN"; // Enter your Wi-Fi SSID here
const char* password = "Ramboo1234@#"; // Enter your Wi-Fi password here
```

- Set the Wi-Fi credentials to connect the ESP8266 to your local Wi-Fi network.

Web Server and Servo Configuration

```
cpp
ESP8266WebServer server(80);

Servo servo; // Create a servo object

int servoPin = 2; // D4 (GPIO2) for the servo control
int servoOpenAngle = 90; // Adjust as needed
int servoClosedAngle = 0; // Adjust as needed
```

- Initialize an HTTP server on port 80.
- Create a Servo object for controlling the servo motor.
- Define the GPIO pin for servo control and servo angles.

Feeding Control Variables

```
cpp
unsigned long lastFeedingTime = 0;
unsigned long feedingInterval = 3600;
unsigned long feedingFrequency = 3600;
bool isServoOn = false;
bool isWebcamOn = false;
```

- Variables for tracking feeding times, intervals, and servo/webcam status.

Webcam Configuration

```
cpp
const char* webcamUrl = "http://192.168.0.104:8080"; // Replace with your IP webcam URL
```

- Set the URL for the webcam feed. Change the IP address accordingly.

Client and HTTPClient Objects

```
cpp
WiFiClient client;
HTTPClient http;
```

- Create a WiFiClient object for the HTTPClient library.

Webcam Refresh Variables

```
cpp
unsigned long lastWebcamRefresh = 0;
unsigned long webcamRefreshInterval = 10000;
```

- Variables for refreshing the webcam feed at regular intervals.

Setup Function

```
cpp
void setup() {
  // ... (Wi-Fi connection setup)

  // Setup HTTP server endpoints
  // Attach servo to the specified pin
  // Start the server
}
```

Loop Function

```
cpp
void loop() {
  // Handle HTTP clients
  // Control servo based on feeding schedule
  // Refresh webcam feed periodically
}
```

Refresh Webcam Function

```
cpp
void refreshWebcamFeed() {
  // Make an HTTP request to refresh the webcam feed
}
```

HTTP Server Handlers

```
cpp
void handle_OnConnect() {
  // Generate HTML response for the main page
}

void handle_servo_on() {
  // Turn on the servo and send a response
}

void handle_servo_off() {
  // Turn off the servo and send a response
}

void handle_toggle_webcam() {
  // Toggle webcam status and send a response
}

void handle_update_interval() {
  // Update feeding interval and send a response
}

void handle_update_frequency() {
  // Update feeding frequency and send a response
}

void handle_NotFound() {
  // Handle 404 Not Found
}
```

HTML Response Generator

```
cpp
String SendHTML(bool servoStatus, bool webcamStatus, unsigned long currentFrequency) {
  // Generate HTML response with current status and controls
}
```

Additional JavaScript

```
cpp
ptr += "<script>\n";
ptr += "// JavaScript function for toggling servo and webcam controls\n";
ptr += "// Additional logic to send requests to the server to toggle the servo and webcam status\n";
ptr += "// You can use AJAX or fetch to make asynchronous requests to the server here\n";
ptr += "</script>\n";
```

Conclusion

This code sets up an ESP8266 as a web server to remotely control a pet feeder and view a webcam feed. The web interface allows toggling the servo, enabling/disabling the webcam, and updating feeding frequency. The JavaScript snippet in the HTML can be extended to handle asynchronous requests for real-time updates without refreshing the entire page.
