Main CODE

```
/// Add plant naming option
#include <Arduino.h>
#include <WiFi.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <ESP_Mail_Client.h>
#include "Adafruit_SHT4x.h"
```

```
#define SMTP_server "smtp.office365.com"
#define SMTP_Port 587
#define sender_email "plant.ee.2024@outlook.com"
#define sender_password "PlantEENotreDame2024"
```

```
//#define NOTIFICATION_LED_PIN 10
#define SUNLIGHT_ANALOG_PIN 4 /// CHANGED
#define MOISTURE_ANALOG_PIN 1 /// CHANGED
#define PUMP_ACTIVATION_PIN 6 /// CHANGED
//#define BUTTON_PIN 9
```

```
int MOISTURE_UPPER = 0;
int MOISTURE_LOWER = 0;
int SUNLIGHT_UPPER = 0;
int SUNLIGHT_LOWER = 0;
int TEMPERATURE_UPPER = 0;
int TEMPERATURE_LOWER = 0;
```

```
const int ARRAY_SIZE = 48; // Number of spots in each array for measurements every 30
minutes for a day
const unsigned long MEASUREMENT_INTERVAL = 60000; // changed to 1 minute //1800000; //
30 minutes in milliseconds
```

```
// Variables
unsigned long lastMeasurementTime = 0;
int tempReadings[ARRAY_SIZE] = {0};
int moistureReadings[ARRAY_SIZE] = {0};
int sunlightReadings[ARRAY_SIZE] = {0};
int readingIndex = 0;
```

```
const int i2cSDA = 19; ///
CHANGED////////////////////////////////////////////////////////////////////////////////
////////////////
const int i2cSCL = 18; /// CHANGED
```

```
const char* AP_SSID = "ESP32-Config-AP";
const char* AP_PASS = "12345678";
const char* MDNS_NAME = "esp32-webserver";
WebServer server(80);
```

```
bool setupComplete = false;
```

```
bool sendEmail = false;
bool moistureFlag = false;
bool temperatureFlag = false;
bool sunlightFlag = false;
bool reservoirFlag = false;
```

```
String selectedPlantType = "";
String enteredEmail = "";
String connectedSSID = "";
```

```
SMTPSession smtp;
ESP_Mail_Session session;
Adafruit_SHT4x sht4 = Adafruit_SHT4x();
```

```
void SendEmail();
void MeasureTemp();
void MeasureSunlight();
void MeasureMoisture();
void WaterPlant();
void SendStatusEmail();
float ProcessReadings(int readings[]);
```

```
void setup() {
delay(4000);
//potentially put the setup complete flag here
Serial.begin(115200);
delay(1000);
Wire.begin(i2cSDA, i2cSCL);
pinMode(PUMP_ACTIVATION_PIN, OUTPUT);
digitalWrite(PUMP_ACTIVATION_PIN, LOW);
//pinMode(NOTIFICATION_LED_PIN, OUTPUT);
//digitalWrite(NOTIFICATION_LED_PIN, LOW);
```

```
WiFi.softAP(AP_SSID, AP_PASS);
```

```
Serial.println("Access Point started");
Serial.print("AP IP Address: ");
Serial.println(WiFi.softAPIP());
smtp.debug(1);
session.server.host_name = SMTP_server;
session.server.port = SMTP_Port;
session.login.email = sender_email;
session.login.password = sender_password;
session.login.user_domain = "";
```

```
if (!MDNS.begin(MDNS_NAME)) {
Serial.println("Error starting mDNS responder!");
} else {
Serial.println("mDNS responder started");
Serial.print("You can now access the ESP32 WebServer at http://");
Serial.print(MDNS_NAME);
Serial.println(".local/");
```

```
}
```

```
server.on("/", []() {
if (setupComplete) {
// If setup is complete, redirect to the summary page
server.sendHeader("Location", "/summary", true);
server.send(303);
} else {
// Show the initial setup page
String html = "<html><head>"
"<style>"
"body {"
" display: flex;"
" justify-content: center;"
" align-items: center;"
" min-height: 100vh;"
" margin: 0;"
"}"
"form {"
" text-align: center;"
"}"
"</style>"
"</head><body>"
"<form method='POST' action='/connect'>"
"<h1>Setup WiFi</h1>"
"SSID: <input type='text' name='ssid'><br>"
"Password: <input type='password' name='password'><br>"
"<input type='submit' value='Connect'>"
"</form>"
"</body></html>";
server.send(200, "text/html", html);
}
});
```

```
server.on("/connect", HTTP_POST, []() {
String ssid = server.arg("ssid");
connectedSSID = server.arg("ssid");
String password = server.arg("password");
WiFi.begin(ssid.c_str(), password.c_str());
server.send(200, "text/html", "<html><head><meta http-equiv='refresh'
content='5;url=/check-connection'></head><body>Connecting to WiFi...</body></html>");
});
```

```
server.on("/check-connection", HTTP_GET, []() {
if (WiFi.status() == WL_CONNECTED) {
Serial.println("Connected to WiFi successfully.");
server.send(200, "text/html", "<html><body>Connected to WiFi. <a
href='/settings'>Proceed to settings</a>.</body></html>");
} else {
Serial.println("Connecting to WiFi...");
server.send(200, "text/html", "<html><head><meta http-equiv='refresh'
content='5;url=/check-connection'></head><body>Checking WiFi connection
status...</body></html>");
}
```

```
});
```

```
server.on("/settings", HTTP_GET, []() {
String html = "<html><head>"
"<style>"
"body {"
" display: flex;"
" justify-content: center;"
" align-items: center;"
" min-height: 100vh;"
" margin: 0;"
" text-align: center;"
"}"
"form {"
" display: inline-block;"
"}"
"</style>"
"</head><body>"
"<form action='/submit' method='post'>"
"<h1>Plant Settings</h1>"
"Select plant type: "
"<select name='plantType'>"
"<option value='Lilly'>Lilly</option>"
"<option value='Cactus'>Cactus</option>"
"<option value='Ficus'>Ficus</option>"
"</select><br>"
"Email: <input type='email' name='email' required><br><br>"
"<input type='submit' value='Submit'>"
"</form>"
"</body></html>";
server.send(200, "text/html", html);
});
```

```
server.on("/submit", HTTP_POST, []() {
selectedPlantType = server.arg("plantType");
enteredEmail = server.arg("email");
if (selectedPlantType == "Lilly"){
MOISTURE_UPPER = 1500;
MOISTURE_LOWER = 2500;
TEMPERATURE_UPPER = 30;
TEMPERATURE_LOWER = 20;
SUNLIGHT_UPPER = 2200;
SUNLIGHT_LOWER = 0;
}
else if(selectedPlantType == "Ficus"){
MOISTURE_UPPER = 2900;
MOISTURE_LOWER = 1500;
TEMPERATURE_UPPER = 30;
TEMPERATURE_LOWER = 20;
SUNLIGHT_UPPER = 2200;
SUNLIGHT_LOWER = 4000;
}
else if (selectedPlantType == "Cactus"){
MOISTURE_UPPER = 4095;
```

```
MOISTURE_LOWER = 2900;
TEMPERATURE_UPPER = 30;
TEMPERATURE_LOWER = 20;
SUNLIGHT_UPPER = 4095;
SUNLIGHT_LOWER = 4000;
}
setupComplete = true; // Mark setup as complete
lastMeasurementTime = 0;
sendEmail = true; // Mark to send the email
```

```
// Send a page that refreshes after 5 seconds to the summary page
String html = "<html><head><meta http-equiv='refresh'
content='5;url=/summary'></head><body>Setup complete. Redirecting to summary
page...</body></html>";
server.send(200, "text/html", html);
});
```

```
server.on("/summary", HTTP_GET, []() {
String temperatureHtml = "Not available";
String sunlightHtml = "Not available";
String moistureHtml = "Not available";
```

```
// Check if there is at least one reading available
if (readingIndex > 0) {
int lastTempReading = tempReadings[readingIndex - 1];
temperatureHtml = String(lastTempReading) + "°C";
```

```
sunlightHtml = (sunlightReadings[readingIndex - 1] >= SUNLIGHT_LOWER &&
sunlightReadings[readingIndex - 1] <= SUNLIGHT_UPPER)
? "<span style='color:green;'>In Range</span>"
: "<span style='color:red;'>" + String(sunlightReadings[readingIndex - 1] <
SUNLIGHT_LOWER ? "Below Range" : "Above Range") + "</span>";
```

```
moistureHtml = (moistureReadings[readingIndex - 1] >= MOISTURE_LOWER &&
moistureReadings[readingIndex - 1] <= MOISTURE_UPPER)
? "<span style='color:green;'>In Range</span>"
: "<span style='color:red;'>" + String(moistureReadings[readingIndex - 1] <
MOISTURE_LOWER ? "Below Range" : "Above Range") + "</span>";
```

```
}
```

```
String html = "<html><head>"
"<style>"
"body {"
" display: flex;"
" flex-direction: column;"
" justify-content: center;"
" align-items: center;"
" min-height: 100vh;"
" margin: 0;"
" text-align: center;"
"}"
```

```
"button {"
" margin: 10px;"
"}"
"</style>"
"</head><body>"
"<h1>Summary</h1>"
"Connected SSID: " + WiFi.SSID() + "<br>"
"Plant Type: " + selectedPlantType + "<br>"
"Email: " + enteredEmail + "<br><br>"
"Current Temperature: " + temperatureHtml + "<br>"
"Sunlight Status: " + sunlightHtml + "<br>"
"Moisture Status: " + moistureHtml + "<br>"
"<p>Refresh this page for the most up to date plant conditions!</p>"
//"<p>If the notification LED is on, refresh this page!</p>"
"<button onclick=\"location.href='/settings'\">Edit Settings</button><br>"
"<button onclick=\"location.href='/check-connection'\">Check WiFi
Connection</button><br>"
"<button onclick=\"location.href='/restart-setup'\">Restart Setup</button>";
```

```
if (reservoirFlag) {
html += "<br><button onclick=\"location.href='/refill-reservoir'\">Reservoir Was
Refilled</button>";
}
```

```
html += "</body></html>";
server.send(200, "text/html", html);
});
```

```
server.on("/restart-setup", HTTP_GET, []() {
setupComplete = false; // Reset the setup complete flag
connectedSSID = ""; // Optionally clear the stored SSID
enteredEmail = ""; // Optionally clear the stored email
selectedPlantType = ""; // Optionally clear the selected plant type
// Redirect to the initial setup page
server.sendHeader("Location", "/", true);
server.send(303); // HTTP status code for redirection
});
```

```
server.on("/refill-reservoir", HTTP_GET, []() {
reservoirFlag = false; // Reset the reservoir flag
//digitalWrite(NOTIFICATION_LED_PIN, LOW);
server.sendHeader("Location", "/summary", true); // Redirect back to the summary page
server.send(303); // HTTP status code for redirection
});
```

```
server.begin();
```

```
}
```

```
void loop() {
server.handleClient();
```

```
if (sendEmail == true) {
SendEmail();
//sendEmail = false;
}
if ((setupComplete == true) && (millis() - lastMeasurementTime >=
MEASUREMENT_INTERVAL)) {
lastMeasurementTime = millis(); // Reset timer
if (readingIndex < ARRAY_SIZE) {
MeasureTemp();
MeasureMoisture();
MeasureSunlight();
readingIndex++;
}
if (readingIndex == ARRAY_SIZE) { // Only process readings when the array is full
float averageTemp = ProcessReadings(tempReadings);
float averageMoisture = ProcessReadings(moistureReadings);
float averageSunlight = ProcessReadings(sunlightReadings);
readingIndex = 0; // Reset readingIndex for new readings
if ((averageTemp > TEMPERATURE_UPPER) || (averageTemp < TEMPERATURE_LOWER)){
temperatureFlag = true;
}
if ((averageSunlight > SUNLIGHT_UPPER) || (averageSunlight < SUNLIGHT_LOWER)){
sunlightFlag = true;
}
if ((averageMoisture > MOISTURE_UPPER) || (averageMoisture < MOISTURE_LOWER)){
moistureFlag = true;
}
if (temperatureFlag || sunlightFlag || moistureFlag) {
SendStatusEmail();
}
```

```
Serial.print("Average Temperature: "); Serial.println(averageTemp);
Serial.print("Average Moisture: "); Serial.println(averageMoisture);
Serial.print("Average Sunlight: "); Serial.println(averageSunlight);
}
}
}
```

```
float ProcessReadings(int readings[]) {
int maxIndex = 0, minIndex = 0;
for (int i = 1; i < ARRAY_SIZE; i++) {
if (readings[i] > readings[maxIndex]) maxIndex = i;
if (readings[i] < readings[minIndex]) minIndex = i;
}
```

```
int sum = 0;
for (int i = 0; i < ARRAY_SIZE; i++) {
if (i != maxIndex && i != minIndex) {
sum += readings[i];
}
}
return (float)sum / (ARRAY_SIZE - 2);
}
```

```
void MeasureTemp(){
sensors_event_t humidity, temp;
uint32_t timestamp = millis();
if (! sht4.begin()) {
Serial.println("Couldn't find SHT4x");
}
else{
sht4.getEvent(&humidity, &temp);
```

```
int temperature_reading = temp.temperature;
int humidity_reading = humidity.relative_humidity;
```

```
tempReadings[readingIndex]=temperature_reading;
Serial.println("Temperature value was stored");
delay(3000);
}
}
```

```
void MeasureSunlight() {
int sensorValue = analogRead(SUNLIGHT_ANALOG_PIN);
sunlightReadings[readingIndex] = sensorValue;
readingIndex++;
delay(3000);
}
```

```
void MeasureMoisture() {
int sensorValue = analogRead(MOISTURE_ANALOG_PIN);
if ((sensorValue < MOISTURE_LOWER) && (!reservoirFlag)){
WaterPlant();
Serial.println("Moisture was too low, watered the plant");
}
```

```
// Check if this is at least the third measurement
if (readingIndex >= 2) {
// Check if the last three readings are not increasing
if (moistureReadings[readingIndex - 1] <= moistureReadings[readingIndex - 2] &&
sensorValue <= moistureReadings[readingIndex - 1]) {
reservoirFlag = true;
//digitalWrite(NOTIFICATION_LED_PIN, HIGH); ////////////////////////////////
SendStatusEmail(); // water reservoir empty
Serial.println("Water reservoir is empty, email was sent");
return; // Optionally return after watering to skip this reading or continue as per
logic requirement
}
}
// Store the current sensor value
moistureReadings[readingIndex] = sensorValue;
Serial.println("Moisture Value was stored");
delay(3000);
}
```

```
void WaterPlant(){
delay(5000);
digitalWrite(PUMP_ACTIVATION_PIN, HIGH);
delay(1000);
Serial.println("Pump on");
delay(3000);
digitalWrite(PUMP_ACTIVATION_PIN, LOW);
Serial.println("Pump off");
}
```

```
void SendEmail() {
if (!smtp.connected()) {
if (!smtp.connect(&session)) {
Serial.println("Failed to connect to SMTP server.");
return; // Exit if unable to connect
}
}
SMTP_Message message;
message.sender.name = "Plant Monitor";
message.sender.email = sender_email;
message.subject = "Plant Monitor Confirmation Email";
message.addRecipient("User", enteredEmail);
```

```
// Send HTML message with dynamic content
String htmlMsg = "<div style=\"color:#000000;\"><h1>Email Confirmation!</h1><p>Mail
Generated from ESP32</p><p>Connected WiFi SSID: " + connectedSSID + "</p><p>Selected
Plant Type: " + selectedPlantType + "</p></div>";
message.html.content = htmlMsg.c_str();
message.text.charSet = "us-ascii";
message.html.transfer_encoding = Content_Transfer_Encoding::enc_7bit;
```

```
if (!MailClient.sendMail(&smtp, &message)) {
Serial.println("Error sending Email, " + smtp.errorReason());
}
```

```
sendEmail = false;
}
```

```
void SendStatusEmail() {
if (!smtp.connected()) {
if (!smtp.connect(&session)) {
Serial.println("Failed to connect to SMTP server.");
return; // Exit if unable to connect
}
}
```

```
SMTP_Message message;
```

```
message.sender.name = "Plant Monitor";
message.sender.email = sender_email;
message.subject = "Plant Monitoring Alert";
message.addRecipient("User", enteredEmail);
```

```
String htmlMsg = "<div style=\"color:#000000;\"><h1>Plant Monitoring
Alert</h1><p>Issues detected with your plant settings:</p><ul>";
```

```
if (temperatureFlag) {
htmlMsg += "<li>Temperature out of desired range.</li>";
}
if (sunlightFlag) {
htmlMsg += "<li>Sunlight levels are not optimal.</li>";
}
if (moistureFlag) {
htmlMsg += "<li>Moisture levels need attention.</li>";
}
if (reservoirFlag) {
htmlMsg += "<li>Water reservoir needs to be refilled.</li>";
}
```

```
htmlMsg += "</ul><p>Please check your plant's environment to ensure it is
optimal.</p></div>";
```

```
message.html.content = htmlMsg.c_str();
message.text.charSet = "us-ascii";
message.html.transfer_encoding = Content_Transfer_Encoding::enc_7bit;
```

```
if (!MailClient.sendMail(&smtp, &message)) {
Serial.println("Error sending Email, " + smtp.errorReason());
}
```

```
// Reset flags after sending the email
temperatureFlag = false;
sunlightFlag = false;
moistureFlag = false;
}
```

.ini file

```
[env:esp32-c3-devkitc-02]
platform = espressif32
board = esp32-c3-devkitc-02
framework = arduino
monitor_speed = 115200
lib_deps =
mobizt/ESP Mail Client@^3.4.15
adafruit/Adafruit SHT4x Library@^1.0.4
```

DEMO Code

```
#include <Arduino.h>
#include <WiFi.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <ESP_Mail_Client.h>
#include "Adafruit_SHT4x.h"

#define SMTP_server "smtp.office365.com"
#define SMTP_Port 587
#define sender_email "plant.ee.2024@outlook.com"
#define sender_password "PlantEENotreDame2024"

#define SUNLIGHT_ANALOG_PIN 4
#define MOISTURE_ANALOG_PIN 1
#define PUMP_ACTIVATION_PIN 6
#define NOTIFICATION_LED_PIN 10

const char* AP_SSID = "ESP32-Demo-AP";
const char* AP_PASS = "12345678";
const char* MDNS_NAME = "esp32-webserverdemo";
WebServer server(80);

SMTPSession smtp;
ESP_Mail_Session session;
Adafruit_SHT4x sht4 = Adafruit_SHT4x();

void SendEmail();

void setup() {
Serial.begin(115200);
pinMode(PUMP_ACTIVATION_PIN, OUTPUT);
pinMode(NOTIFICATION_LED_PIN, OUTPUT);
digitalWrite(PUMP_ACTIVATION_PIN, LOW);
digitalWrite(NOTIFICATION_LED_PIN, LOW);

WiFi.softAP(AP_SSID, AP_PASS);
Serial.println("AP started, IP: " + WiFi.softAPIP().toString());
if (!MDNS.begin(MDNS_NAME)) {
Serial.println("Error starting mDNS responder!");
} else {
Serial.println("mDNS responder started at http://" + String(MDNS_NAME) + ".local/");
```

```
}

server.on("/", []() {
String html = "<html><body>"
"<h1>ESP32 Plant Monitor Setup</h1>"
"<form method='POST' action='/connect'>"
"SSID: <input type='text' name='ssid'><br>"
"Password: <input type='password' name='password'><br>"
"<input type='submit' value='Connect'>"
"</form></body></html>";
server.send(200, "text/html", html);
});

server.on("/connect", HTTP_POST, []() {
String ssid = server.arg("ssid");
String password = server.arg("password");
WiFi.begin(ssid.c_str(), password.c_str());
server.sendHeader("Location", "/check-connection", true);
server.send(303);
});

server.on("/check-connection", HTTP_GET, []() {
if (WiFi.status() == WL_CONNECTED) {
server.sendHeader("Location", "/demo", true);
server.send(303);
} else {
String html = "<html><body>Connecting to WiFi..."
"<meta http-equiv='refresh' content='5;url=/check-connection'></head><body>"
"<p>Please wait while we try to connect to the WiFi network. If this page does not
refresh, please manually refresh or check your WiFi details and try again.</p>"
"</body></html>";
server.send(200, "text/html", html);
}
});

server.on("/demo", []() {
String html = "<html><head><title>Demo Page</title></head><body>"
"<h1>Select Demo</h1>"
"<ul>"
"<li><a href='/demo-email-confirmation'>Email Confirmation</a></li>"
"<li><a href='/demo-email-update'>Email Update</a></li>"
"<li><a href='/demo-sensor-readings'>Sensor Readings</a></li>"
```

```
"<li><a href='/demo-pumping-water'>Pump Water</a></li>"
"<li><a href='/demo-tank-empty'>Simulate Tank Empty</a></li>"
"</ul></body></html>";
server.send(200, "text/html", html);
});

// Implementing handlers for demos
server.on("/demo-email-confirmation", []() {
String html = "<html><body><h1>Email Sent</h1><p>An email confirmation was sent to:
sazzam@nd.edu. Allow around a minute before receiving</p>"
"<a href='/demo'>Back to Demo</a></body></html>";
server.send(200, "text/html", html);
SendEmail();
// HTML response for the webserver
});



server.on("/demo-email-update", []() {
String html = "<html><body><h1>Update Email Sent</h1><p>An update email was sent to:
sazzam@nd.edu</p>"
"<a href='/demo'>Back to Demo</a></body></html>";
server.send(200, "text/html", html);

if (!smtp.connected()) {
smtp.connect(&session);
}
SMTP_Message message;
message.sender.name = "Plant Monitor";
message.sender.email = sender_email;
message.subject = "Alert: Sensor Readings Out of Range";
message.addRecipient("User", "sazzam@nd.edu"); // Using sender's email for demo

message.text.content = "Warning: The sunlight and temperature readings are out of the
specified range. Move the plant to a cooler room with less sunlight";

if (!MailClient.sendMail(&smtp, &message)) {
Serial.println("Error sending Email: " + smtp.errorReason());
}


});
```

```
server.on("/demo-sensor-readings", []() {
// Simulate reading sensors
/*int temperature = random(20, 30); // Simulate temperature reading
int moisture = random(300, 700); // Simulate moisture reading
int sunlight = random(100, 800); // Simulate sunlight reading
*/

String html = "<html><body><h1>Current Sensor Readings</h1>"
"<p>Temperature: 23.2 °C</p>"
"<p>Moisture: In range </p>"
"<p>Sunlight: Above range </p>"
"<a href='/demo'>Back to Demo</a></body></html>";
server.send(200, "text/html", html);
});

server.on("/demo-pumping-water", []() {
String html = "<html><body><h1>Pumping Water</h1><p>The Plant has been watered!.</p>"
"<a href='/demo'>Back to Demo</a></body></html>";
server.send(200, "text/html", html);
digitalWrite(PUMP_ACTIVATION_PIN, HIGH);
delay(3000); // Pump water for 3 seconds
digitalWrite(PUMP_ACTIVATION_PIN, LOW);

});

server.on("/demo-tank-empty", []() {
//digitalWrite(NOTIFICATION_LED_PIN, HIGH); // Turn on notification LED to simulate
tank empty
String html = "<html><body><h1>Tank Empty</h1><p>The water tank is empty. An alert
email has been sent.</p>"
"<button onclick=\"location.href='/reset-tank-empty'\">Refill and Reset</button>"
"<a href='/demo'>Back to Demo</a></body></html>";
server.send(200, "text/html", html);

if (!smtp.connected()) {
smtp.connect(&session);
}
SMTP_Message message;
message.sender.name = "Plant Monitor";
message.sender.email = sender_email;
message.subject = "Alert: Water Tank Empty";
```

```
message.addRecipient("User", sender_email); // Using sender's email for demo

message.text.content = "Alert: The water tank is empty. Please refill.";

if (!MailClient.sendMail(&smtp, &message)) {
Serial.println("Error sending Email: " + smtp.errorReason());
}

});

server.on("/reset-tank-empty", []() {
//digitalWrite(NOTIFICATION_LED_PIN, LOW); // Turn off the notification LED

String html = "<html><body><h1>Tank Refilled</h1><p>The tank status has been
reset.</p>"
"<a href='/demo'>Back to Demo</a></body></html>";
server.send(200, "text/html", html);
});

server.begin();
}

void loop() {
server.handleClient();
}

void SendEmail() {
if (!smtp.connected()) {
if (!smtp.connect(&session)) {
Serial.println("Failed to connect to SMTP server.");
return; // Exit if unable to connect
}
}
SMTP_Message message;
message.sender.name = "Plant Monitor";
message.sender.email = sender_email;
message.subject = "Plant Monitor Confirmation Email";
message.addRecipient("User", "sazzam@nd.edu");

// Send HTML message with dynamic content
String htmlMsg = "<div style=\"color:#000000;\"><h1>Email Confirmation!</h1><p>Mail
Generated from ESP32</p></div>";
```

```
message.html.content = htmlMsg.c_str();
message.text.charSet = "us-ascii";
message.html.transfer_encoding = Content_Transfer_Encoding::enc_7bit;

if (!MailClient.sendMail(&smtp, &message)) {
Serial.println("Error sending Email, " + smtp.errorReason());
}

}
```

.ini file

```
[env:esp32-c3-devkitc-02]
platform = espressif32
board = esp32-c3-devkitc-02
framework = arduino
monitor_speed = 115200
lib_deps =
mobizt/ESP Mail Client@^3.4.15
adafruit/Adafruit SHT4x Library@^1.0.4
```