Project 20:

Send Data of soil Moisture Via Smpt server

1. Introduction

It is used to send Regular Data in E mail via Smrt Server to get Update of our Farm during Office Hour.

COMPONENTS: -

1. WEMOS
2. SOIL MOISTURE

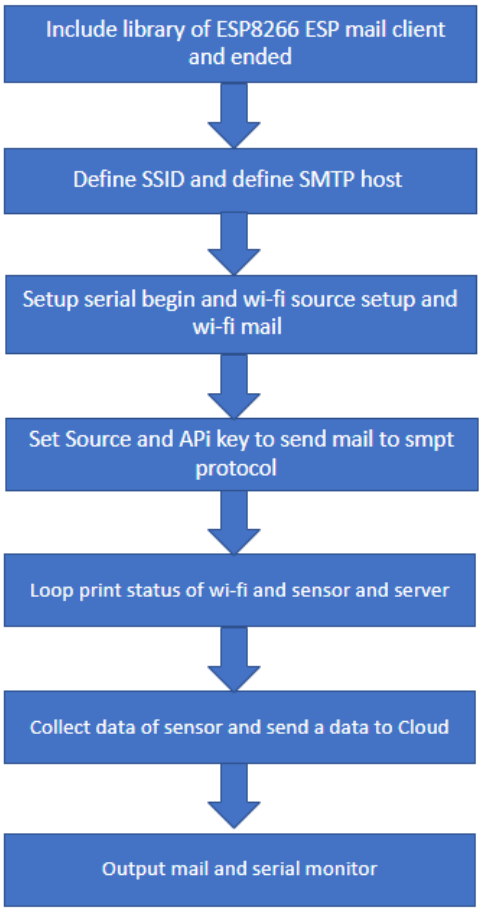
APPLICATIONS: -

The Simple Mail Transfer Protocol (SMTP) is used to deliver e-mail messages over the Internet. This protocol is used by most e-mail clients to deliver messages to the server, and is also used by servers to forward messages to their final destination.

OBJECTIVES: -

SMTP represents Simple Mail Transfer Protocol. SMTP is a set of interaction guidelines that allow the software to transmit electronic mail over the internet, referred to as Simple Mail Transfer Protocol. The main objective of SMTP is used to set up communication rules between servers.

FLOW CHART: -



PROGRAMMING: -

```
#include <Arduino.h>
```

```
#if defined(ESP32)
#include <WiFi.h>
#elif defined(ESP8266)
#include <ESP8266WiFi.h>
#endif
#include <ESP_Mail_Client.h>

//To use only SMTP functions, you can exclude the IMAP from compilation, see ESP_Mail_FS.h.

#define WIFI_SSID "<ssid>"
#define WIFI_PASSWORD "<password>"

/** The smtp host name e.g. smtp.gmail.com for GMail or smtp.office365.com for Outlook or smtp.mail.yahoo.com
 * For yahoo mail, log in to your yahoo mail in web browser and generate app password by go to
 * https://login.yahoo.com/account/security/app-passwords/add/confirm?src=noSrc
 * and use the app password as password with your yahoo mail account to login.
 * The google app password signin is also available https://support.google.com/mail/answer/185833?hl=en
*/
#define SMTP_HOST "<host>"

/** The smtp port e.g.
```

```
 * 25  or esp_mail_smtp_port_25
 * 465 or esp_mail_smtp_port_465
 * 587 or esp_mail_smtp_port_587
*/
#define SMTP_PORT esp_mail_smtp_port_587

/* The log in credentials */
#define AUTHOR_EMAIL "<email>"
#define AUTHOR_PASSWORD "<password>"

/* The SMTP Session object used for Email sending */
SMTPSession smtp;

/* Callback function to get the Email sending status */
void smtpCallback(SMTP_Status status);

void setup()
{

  Serial.begin(115200);

#if defined(ARDUINO_ARCH_SAMD)
  while (!Serial)
    ;
  Serial.println();
```

```
  Serial.println("**** Custom built WiFiNINA firmware need to be installed.****\nTo install firmware, read the instruction here, https://github.com/mobizt/ESP-Mail-Client#install-custom-built-wifinina-firmware");

#endif

  Serial.println();

  Serial.print("Connecting to AP");

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(200);
  }

  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  /** Enable the debug via Serial port
   * none debug or 0
```

```
   * basic debug or 1
   *
   * Debug port can be changed via
ESP_MAIL_DEFAULT_DEBUG_PORT in ESP_Mail_FS.h
   */
  smtp.debug(1);

  /* Set the callback function to get the sending results */
  smtp.callback(smtpCallback);

  /* Declare the session config data */
  ESP_Mail_Session session;

  /* Set the session config */
  session.server.host_name = SMTP_HOST;
  session.server.port = SMTP_PORT;
  session.login.email = AUTHOR_EMAIL;
  session.login.password = AUTHOR_PASSWORD;
  session.login.user_domain = "mydomain.net";

  /* Set the NTP config time */
  session.time.ntp_server = "pool.ntp.org,time.nist.gov";
  session.time.gmt_offset = 3;
  session.time.day_light_offset = 0;

  /* Declare the message class */
```

```
  SMTP_Message message;


  /* Set the message headers */
  message.sender.name = "ESP Mail";
  message.sender.email = AUTHOR_EMAIL;
  message.subject = "Test sending message as embedded files";
  message.addRecipient("Admin",
"change_this@your_mail_dot_com");


  message.html.content = "<span style=\"color:#0055ff;\">This is html
message</span>";


  /** The Plain text message character set e.g.
   * us-ascii
   * utf-8
   * utf-7
   * The default value is utf-8
  */
  message.html.charSet = "utf-8";


  /** The content transfer encoding e.g.
   * enc_7bit or "7bit" (not encoded)
   * enc_qp or "quoted-printable" (encoded)
   * enc_base64 or "base64" (encoded)
   * enc_binary or "binary" (not encoded)
   * enc_8bit or "8bit" (not encoded)
```

```
   * The default value is "7bit"
   */
   message.html.transfer_encoding =
Content_Transfer_Encoding::enc_qp;

   /* Enable to send this message body as file */
   message.html.embed.enable = true;

   /* The name of embedded file */
   message.html.embed.filename = "test.html";

   /** The embedded type
    * esp_mail_smtp_embed_message_type_attachment or 0
    * esp_mail_smtp_embed_message_type_inline or 1
   */
   message.html.embed.type =
esp_mail_smtp_embed_message_type_attachment;



   message.text.content = "This is simple plain text message";
   message.text.charSet = "utf-8";
   message.text.transfer_encoding =
Content_Transfer_Encoding::enc_base64;
   message.text.embed.enable = true;
   message.text.embed.filename = "test.txt";
```

```
  message.text.embed.type =
esp_mail_smtp_embed_message_type_inline;



  /** The message priority
   * esp_mail_smtp_priority_high or 1
   * esp_mail_smtp_priority_normal or 3
   * esp_mail_smtp_priority_low or 5
   * The default value is esp_mail_smtp_priority_low
  */
  message.priority =
esp_mail_smtp_priority::esp_mail_smtp_priority_low;



  /* Set the custom message header */
  message.addHeader("Message-ID: <abcde.fghij@gmail.com>");



  /* Connect to server with the session config */
  if (!smtp.connect(&session))
    return;

  /* Start sending Email and close the session */
  if (!MailClient.sendMail(&smtp, &message))
    Serial.println("Error sending Email, " + smtp.errorReason());
```

```
  //to clear sending result log
  //smtp.sendingResult.clear();

  ESP_MAIL_PRINTF("Free Heap: %d\n", MailClient.getFreeHeap());
}

void loop()
{
}

/* Callback function to get the Email sending status */
void smtpCallback(SMTP_Status status)
{
  /* Print the current status */
  Serial.println(status.info());

  /* Print the sending result */
  if (status.success())
  {
    Serial.println("----------------");
    ESP_MAIL_PRINTF("Message sent success: %d\n",
status.completedCount());
    ESP_MAIL_PRINTF("Message sent failled: %d\n",
status.failedCount());
    Serial.println("----------------\n");
    struct tm dt;
```

```
  for (size_t i = 0; i < smtp.sendingResult.size(); i++)
  {
    /* Get the result item */
    SMTP_Result result = smtp.sendingResult.getItem(i);
    time_t ts = (time_t)result.timestamp;
    localtime_r(&ts, &dt);

    ESP_MAIL_PRINTF("Message No: %d\n", i + 1);
    ESP_MAIL_PRINTF("Status: %s\n", result.completed ? "success" : "failed");
    ESP_MAIL_PRINTF("Date/Time: %d/%d/%d %d:%d:%d\n", dt.tm_year + 1900, dt.tm_mon + 1, dt.tm_mday, dt.tm_hour, dt.tm_min, dt.tm_sec);
    ESP_MAIL_PRINTF("Recipient: %s\n", result.recipients);
    ESP_MAIL_PRINTF("Subject: %s\n", result.subject);
  }
  Serial.println("----------------\n");

  //You need to clear sending result as the memory usage will grow up as it keeps the status, timstamp and
  //pointer to const char of recipients and subject that user assigned to the SMTP_Message object.

  //Because of pointer to const char that stores instead of dynamic string, the subject and recipients value can be
  //a garbage string (pointer points to undefind location) as SMTP_Message was declared as local variable or the value changed.
```

```
    //smtp.sendingResult.clear();
  }
}
```

HARDWARE CONNECTION: -

1. Connect soil moisture to wemos to relay
2. Connect pin D1 to signal
3. Connect pin vcc to vcc
4. Connect pin GND to GND

CIRCUIT DIAGRAM: -