Part 08
-
Sending emails

Version: 2021-03-23

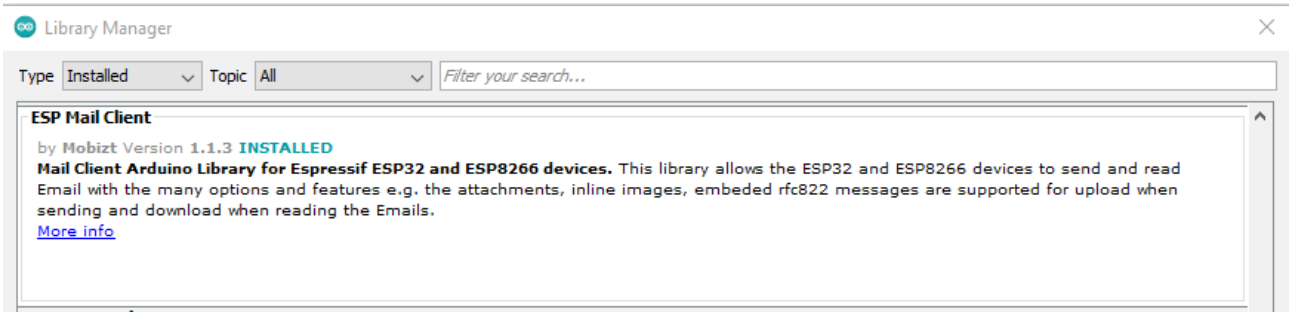
Send Emails using an SMTP Server

This document explains how to send a simple email with raw or HTML text.
The ESP32 board will be programmed using Arduino IDE.

Installing the ESP MailClient Library

Before proceeding, you need to install the ESP MailClient library. This library can be installed through the Arduino IDE Library Manager.
In your Arduino IDE go to **Sketch** > **Include Library** > **Manage Libraries...**
The Library Manager should open. Search for **ESP Mail Client** by Mobizt and install the latest library.

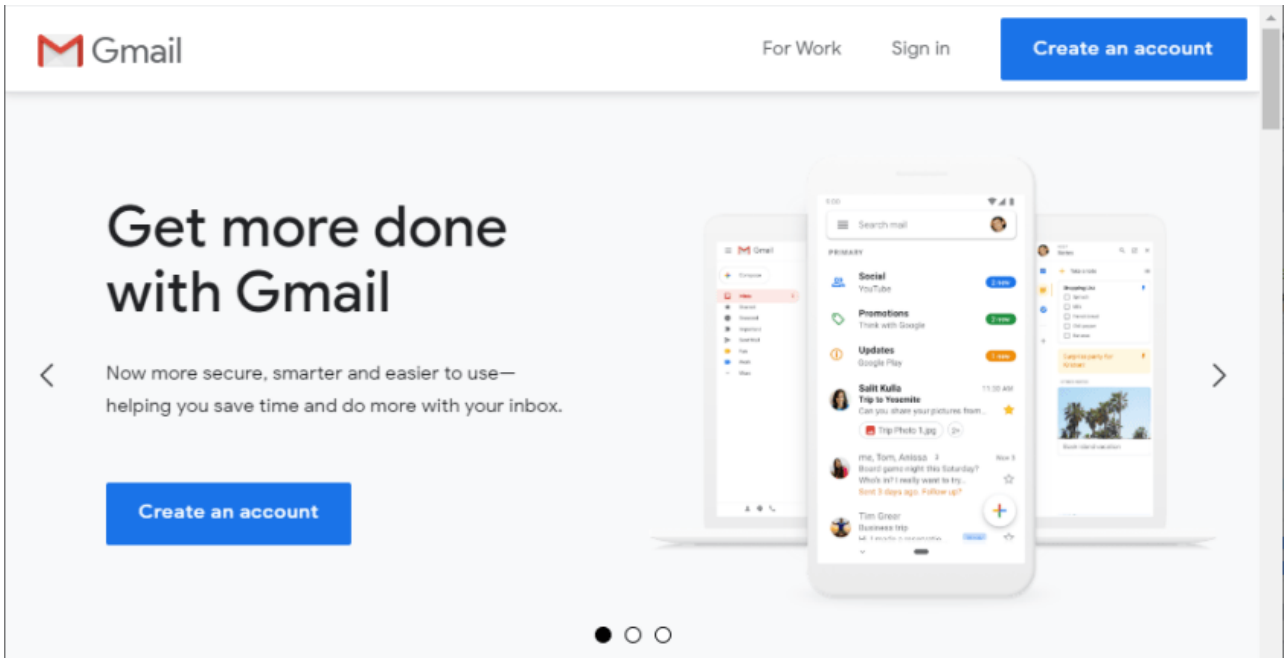
Note: Do not install the ESP32 Mail Client library as that one is obsoleted. Notice the difference between ESP and ESP32 !!!



Note: This library does not support simple SMTP but only secure SMTP.

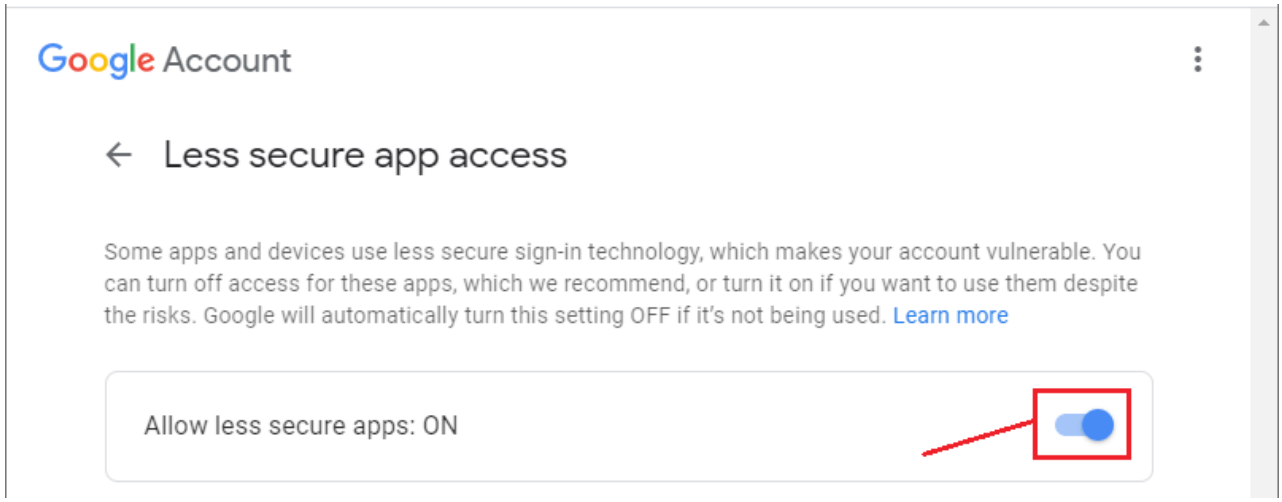
Sender Email (New Account)
For testing purpose, create a new email account to send the emails to your main personal email address. Do not use your main personal email to send emails via ESP32. If something goes wrong in your code or if by mistake you make too many requests, you can be banned or have your account temporary disabled.
A Gmail.com account to send the emails will be used, but you can use any other email provider. The receiver email can be your personal email without any problem.

If you want to use a Gmail account, create a new one.



Allow less secure apps to get access to this new Gmail account, so that you're able to send emails. Enabling less secure apps to access Gmail

- Go to your Google Account.
- On the left navigation panel, click Security.
- On the bottom of the page, in the *Less secure app access panel*, click Turn on access.



Gmail SMTP Server Settings

- SMTP Server: **smtp.gmail.com**
- SMTP username: Gmail mail address
- SMTP password: Gmail password
- SMTP port (TLS): **587**

Outlook SMTP Server Settings

- SMTP Server: **smtp.office365.com**
- SMTP Username: Outlook mail address
- SMTP Password: Outlook password
- SMTP Port (TLS): **587**

Live or Hotmail SMTP Server Settings

- SMTP Server: **smtp.live.com**
- SMTP Username: Live or Hotmail mail address
- SMTP Password: Live or Hotmail password
- SMTP Port (TLS): **587**

If you’re using another email provider, you need to search for its SMTP Server settings.

Now, you have everything ready to start sending emails with your ESP32.

Send a mail in plain text with ESP32

The following code sends a mail via SMTP Server in plain text. For demonstration purposes, the ESP32 sends a mail once when it boots.

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

// WiFi settings
const char* wifiSSID     = "<SSID>";
const char* wifiPassword = "<PASSWORD>";
const char* hostname     = "esp32test";

// to hold MAC address
String wifiMACAddress    = WiFi.macAddress();

#define SMTP_HOST        "smtp.telenet.be"
#define SMTP_PORT        587
#define AUTH_EMAIL       "<mailaddress>"
#define AUTH_PASSWORD    "<password>"
#define FROM_NAME        hostname
#define FROM_EMAIL       "<mailaddress>"
#define TO_NAME          "<name>"
#define TO_EMAIL         "<mailaddress>"

// The SMTP Session object used for mail sending
SMTPSession smtp;

// Set-up WIFI
//**************************************************************************
void wifiConnect()
{
  WiFi.config(INADDR_NONE, INADDR_NONE, INADDR_NONE, INADDR_NONE); // needed to allow setting
hostname
  WiFi.setHostname(hostname);
  WiFi.mode(WIFI_STA);
  WiFi.begin(wifiSSID, wifiPassword);

  Serial.print(F(" - WiFi attempting connection to ")); Serial.println(wifiSSID);

  uint8_t i = 0;
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print('.');
    // delay non-blocking
    unsigned long time_now = millis();
    while(millis() < time_now + 500){}
    if ((++i % 16) == 0)
    {
      Serial.println(F(""));
      Serial.println(F(" - still trying to connect"));
    }

    if (i > 32)
    {
      Serial.println(F(" - Restarting..."));
      ESP.restart();
    }
  }

  Serial.println(F(""));
  Serial.println(F("WiFi connected"));
  Serial.print(F("  MAC Address: ")); Serial.println(wifiMACAddress);
  Serial.print(F("  IP Address : ")); Serial.println(WiFi.localIP());
  Serial.print(F("  Hostname   : ")); Serial.println(WiFi.getHostname());
}
//**************************************************************************

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

  // Print the sending result
  if (status.success())
  {
```

```
    //Serial.println("----------------");
    //Serial.printf("Message sent success: %d\n", status.completedCount());
    //Serial.printf("Message sent failled: %d\n", status.failedCount());
    //Serial.println("----------------\n");

    for (size_t i = 0; i < smtp.sendingResult.size(); i++)
    {
      // Get the result item
      SMTP_Result result = smtp.sendingResult.getItem(i);
      Serial.printf("Message No: %d\n", i + 1);
      Serial.printf("Status: %s\n", result.completed ? "success" : "failed");
      Serial.printf("Recipient: %s\n", result.recipients);
      Serial.printf("Subject: %s\n", result.subject);
    }
  }
}
//**************************************************************************

//**************************************************************************
void setup()
{

  Serial.begin(115200);
  Serial.println();

  // Startup WiFi
  Serial.print("Connecting to AP");
  wifiConnect();

  // Enable the debug via Serial port
  // no debugging     = 0
  // basic debugging  = 1
  smtp.debug(0);

  // 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       = AUTH_EMAIL;
  session.login.password    = AUTH_PASSWORD;
  //session.login.user_domain = "mydomain.net";

  // Declare the message class
  SMTP_Message message;

  // Set the message headers
  message.sender.name    = FROM_NAME;
  message.sender.email   = FROM_EMAIL;
  message.subject        = "Test sending plain text Email";
  message.addRecipient(TO_NAME, TO_EMAIL);

  message.text.content = "This is simple plain text message";

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

  // The content transfer encoding
  //   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.text.transfer_encoding = Content_Transfer_Encoding::enc_8bit;

  // 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;

  // The Delivery Status Notifications
```

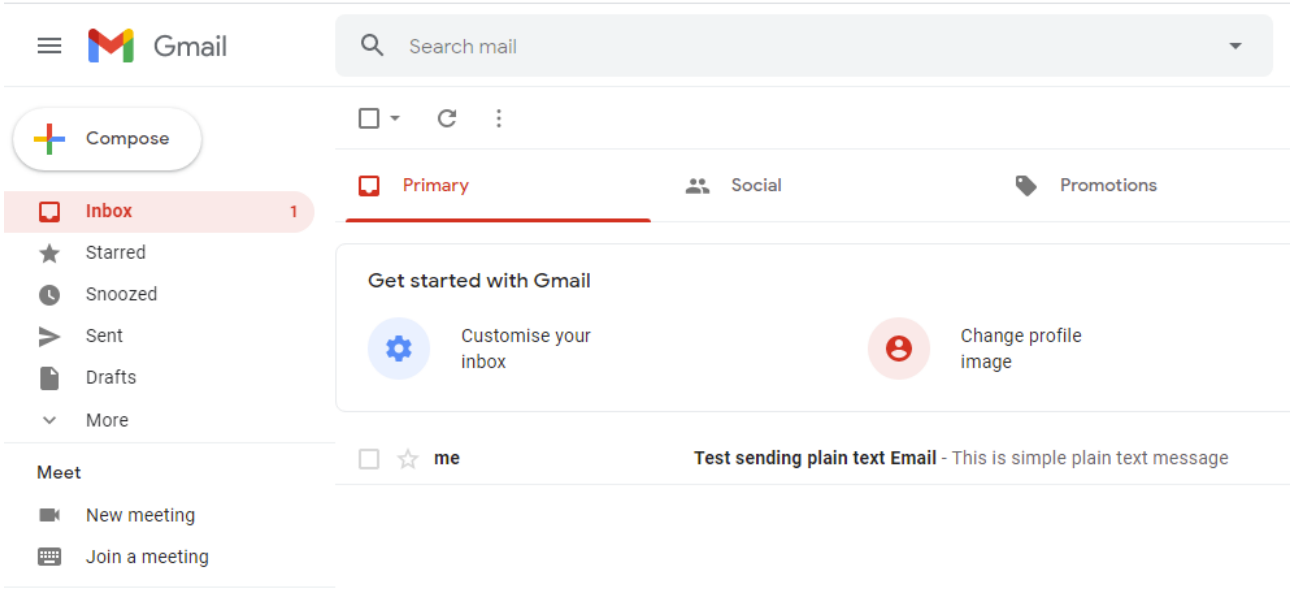
```
  //   esp_mail_smtp_notify_never
  //   esp_mail_smtp_notify_success
  //   esp_mail_smtp_notify_failure
  //   esp_mail_smtp_notify_delay
  // The default value is esp_mail_smtp_notify_never
  message.response.notify = esp_mail_smtp_notify_success | esp_mail_smtp_notify_failure |
esp_mail_smtp_notify_delay;

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

  // Connect to server with the session config
  if (!smtp.connect(&session))
  {
    Serial.println("SMTP failed to connect: " + smtp.errorReason());
    return;
  }
  else
  {
    Serial.println("SMTP connection succesful");
  }

  // Start sending Email and close the session
  if (!MailClient.sendMail(&smtp, &message))
  {
    Serial.println("Error sending mail: " + smtp.errorReason());
  }
  else
  {
    Serial.println("Succes sending mail");
  }
}
//****************************************************************************

//****************************************************************************
void loop()
{
}
//****************************************************************************
```

You need to insert your network credentials as well as setting the sender email, SMTP Server details and recipient.

```
16:57:10.173 -> Connecting to AP - WiFi attempting connection to Engrie-IoT
16:57:10.314 -> .
16:57:10.829 -> WiFi connected
16:57:10.829 ->   MAC Address: AC:67:B2:36:0B:BC
16:57:10.829 ->   IP Address : 192.168.1.80
16:57:10.829 ->   Hostname   : esp32test
16:57:10.829 -> Connecting to SMTP server...
16:57:10.829 ->
16:57:10.829 -> SMTP server connected, wait for greeting...
16:57:11.858 ->
16:57:11.858 -> Sending greeting response...
16:57:13.833 ->
16:57:13.880 -> Send command, STARTTLS
16:57:15.633 ->
16:57:15.633 -> Sending greeting response...
16:57:15.680 ->
16:57:15.680 -> Logging in...
16:57:15.680 -> SMTP connection succesful
16:57:15.680 ->
16:57:15.680 -> Sending Email...
16:57:15.680 ->
16:57:15.680 -> Sending message header...
16:57:15.727 ->
16:57:15.727 -> Sending message body...
16:57:15.768 ->
16:57:15.768 -> Finishing the message sending...
16:57:15.815 ->
16:57:15.815 -> Closing the session...
16:57:15.862 ->
```

```
16:57:15.862 -> Message sent successfully
16:57:15.862 ->
16:57:15.862 -> Message No: 1
16:57:15.862 -> Status: success
16:57:15.862 -> Recipient: marc.engrie@telenet.be
16:57:15.862 -> Subject: Test sending plain text Email
16:57:15.862 -> Succes sending mail
```

Check your email account. You should have received an email from your ESP32 board.



NB: in my test code I set the To and the From equal so account snds an mail to himself :-)