- **Contraseña del originario del correo** es la contraseña correspondiente al correo electrónico del originario.
- **Correo electrónico del destinatario** es la dirección de correo electrónico de la persona a quien se le envía el correo electrónico.
- **Asunto del correo** es el "subject" del correo.
- **Mensaje del correo** es el texto del correo.

Referencias

1. SendGrid - Azure
2. SendGrid v3 API Documentation
3. How to Send an SMTP Email
4. JavaMail API - Sending Simple Email
5. JavaMail API - Sending Email With Attachment
6. JavaMail API - Sending an HTML Email
7. JavaMail API - Sending Email With Inline Imagess

Clase del día - 6/06/2023

La clase de hoy haremos un ejercicio de Internet de las cosas (IoT, *Internet of things*); veremos cómo conectar una tarjeta ESP32 a un servicio en Azure Functions.

Internet de las cosas

Internet de las cosas se trata de una red de dispositivos interconectados que se comunican a través de Internet. Estos dispositivos "inteligentes" son microcontroladores dotados de sensores y actuadores.

Los dispositivos de Internet de las cosas pueden recopilar, intercambiar y analizar datos.

El objetivo del Internet de las cosas es hacer que los objetos cotidianos, tales como electrodomésticos, vehículos, equipos industriales, dispositivos portátiles, sistemas de monitoreo, etc., recopilen datos en tiempo real y los envíen a Internet.

Un aspecto muy importante del Internet de las cosas es la seguridad de los datos, ya que gran parte de la información que se recopila y transmite por medio de estos dispositivos "inteligentes" es información sensible, por tanto es muy importante implementar una comunicación segura así como garantizar la seguridad física de los dispositivos.

Creación de un proyecto en Visual Studio

Primeramente vamos a crear un proyecto en Visual Studio; por omisión el nombre de la función será "Function1".

Posteriormente publicamos el proyecto a Azure, en este caso el nombre de la aplicación de funciones será "esp32-iot".

Conexión del ESP32 a Azure Functions

Utilizaremos el IDE de Arduino para compilar y cargar el programa a la tarjeta ESP32.

Para instalar la tarjeta ESP32 en el IDE de Arduino:

- Conectar la tarjeta a un puerto USB
- Seleccionar las opciones: Archivo -> Preferencias
- Ingresar en el campo "Gestor de URLs adicionales de Tarjetas": https://dl.espressif.com/dl/package_esp32_index.json
- Dar clic en el botón OK.
- Seleccionar las opciones: Herramientas -> Placa -> ESP32 Arduino -> NodeMCU-32S
- Seleccionar las opciones: Herramientas -> Upload Speed -> 115200

- Ejecutar el programa "devmgmt.msc" en una ventana Powershell en modo administrador para ver el puerto COM asignado al puerto USB al que se conectó el ESP32. Abrir la sección "Puertos (COM y LPT)"
- Seleccionar las opciones: Herramientas -> Puerto -> (puerto COM asignado al puerto USB).

El programa a compilar y cargar es el siguiente:

```
/*
 * Conexión del ESP32 a Azure Functions
 * Carlos Pineda G. 2023
*/
```

```
#include <HTTPClient.h>
#include <WiFi.h>
```

```
// nombre de la red y contraseña
const char *ssid = "";
const char *password = "";
```

```
// URL de la función en Azure
String url = "https://esp32-iot.azurewebsites.net/api/Function1";
```

```
HTTPClient http;
```

```
void setup()
{
  // retardo para poder abrir el Monitor Serie del IDE de Arduino
  delay(3000);
  Serial.begin(115200);
  // se conecta a la red
  WiFi.begin(ssid,password);
  Serial.print("Conectando a la red ");
  Serial.print(ssid);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi conectada");
  Serial.print("Dirección IP: ");
  Serial.println(WiFi.localIP());
}
int envia_post()
{
    http.begin(url);
    String body = "{'name':'carlos'}";
    Serial.println("Enviando POST");
    int codigo = http.POST(body);
    Serial.print("Codigo HTTP: ");
    Serial.println(codigo);
    String respuesta = http.getString();
    Serial.print("Respuesta: ");
    Serial.println(respuesta);
}
void envia_get()
{
    http.begin(url + "&name=carlos");
    Serial.println("Enviando GET");
    int codigo = http.GET();
    Serial.print("Codigo HTTP: ");
    Serial.println(codigo);
    String respuesta = http.getString();
    Serial.print("Respuesta: ");
    Serial.println(respuesta);
}
void loop()
{
  envia_post();
  delay(3000);
  envia_get();
  delay(3000);
}
```

Referencias

1. ESP32 ADC with Arduino IDE – Measuring voltage example
2. How to connect to a WiFi network with the ESP32
3. HttpClient

Clase del día - 8/06/2023

Anteriormente utilizamos el paradigma de paso de mensajes y el paradigma de objetos distribuidos para desarrollar sistemas distribuidos.

La clase de hoy veremos el **paradigma de memoria compartida distribuida**, el cual permite desarrollar sistemas distribuidos utilizando redes de computadoras de manera similar al desarrollo de sistemas utilizando threads.