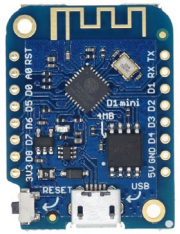
Universidad Politécnica de Madrid

Uso del IoT para construir tú mismo un hogar digital

Diseño software del sensor de temperatura

Iván Pau

Requisitos

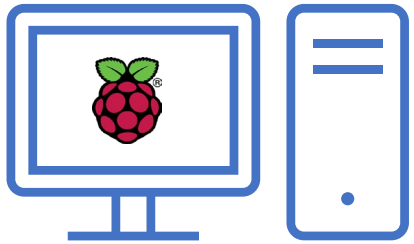


Multisensor

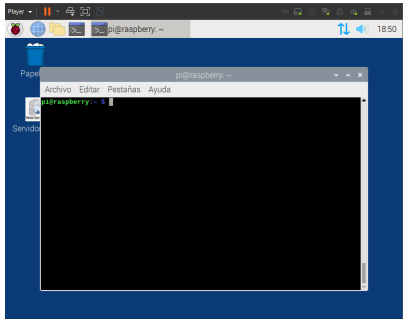
casa/dormitorio/humedad

casa/dormitorio/temperatura

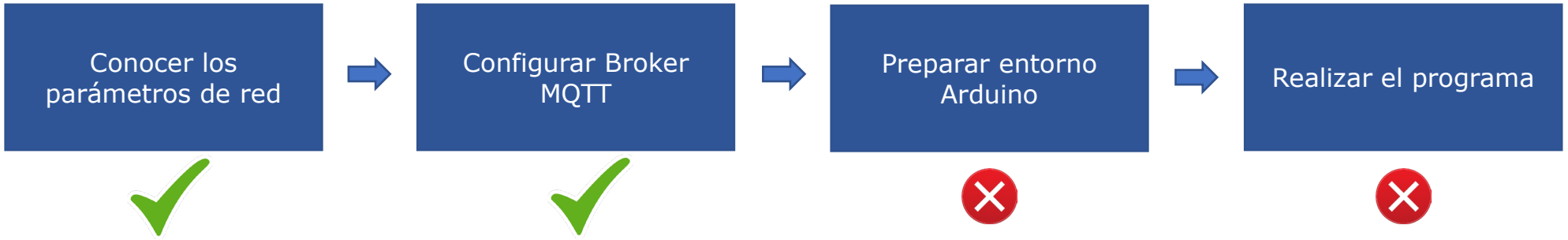
casa/dormitorio/sensacionTermica



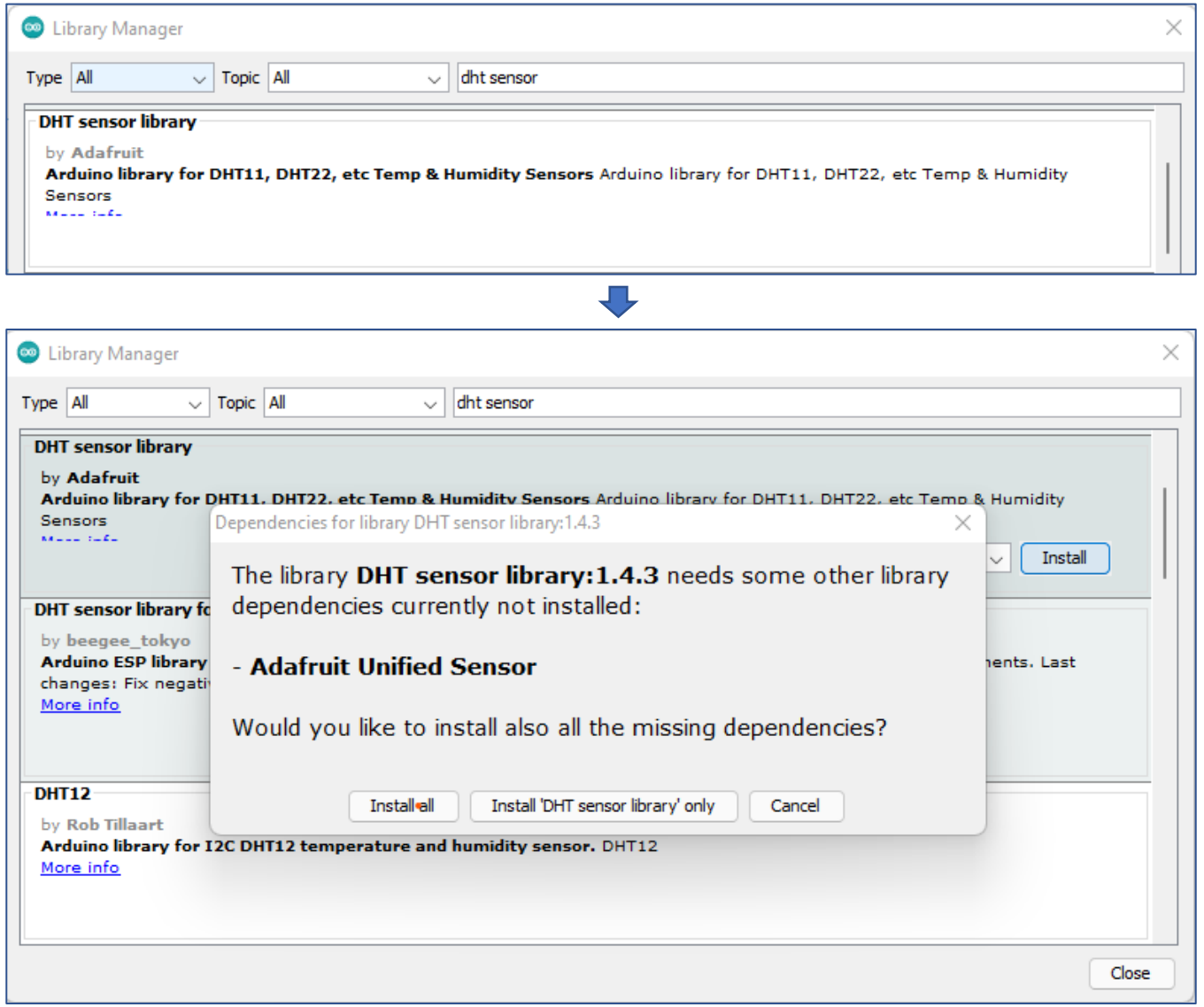
Broker MQTT



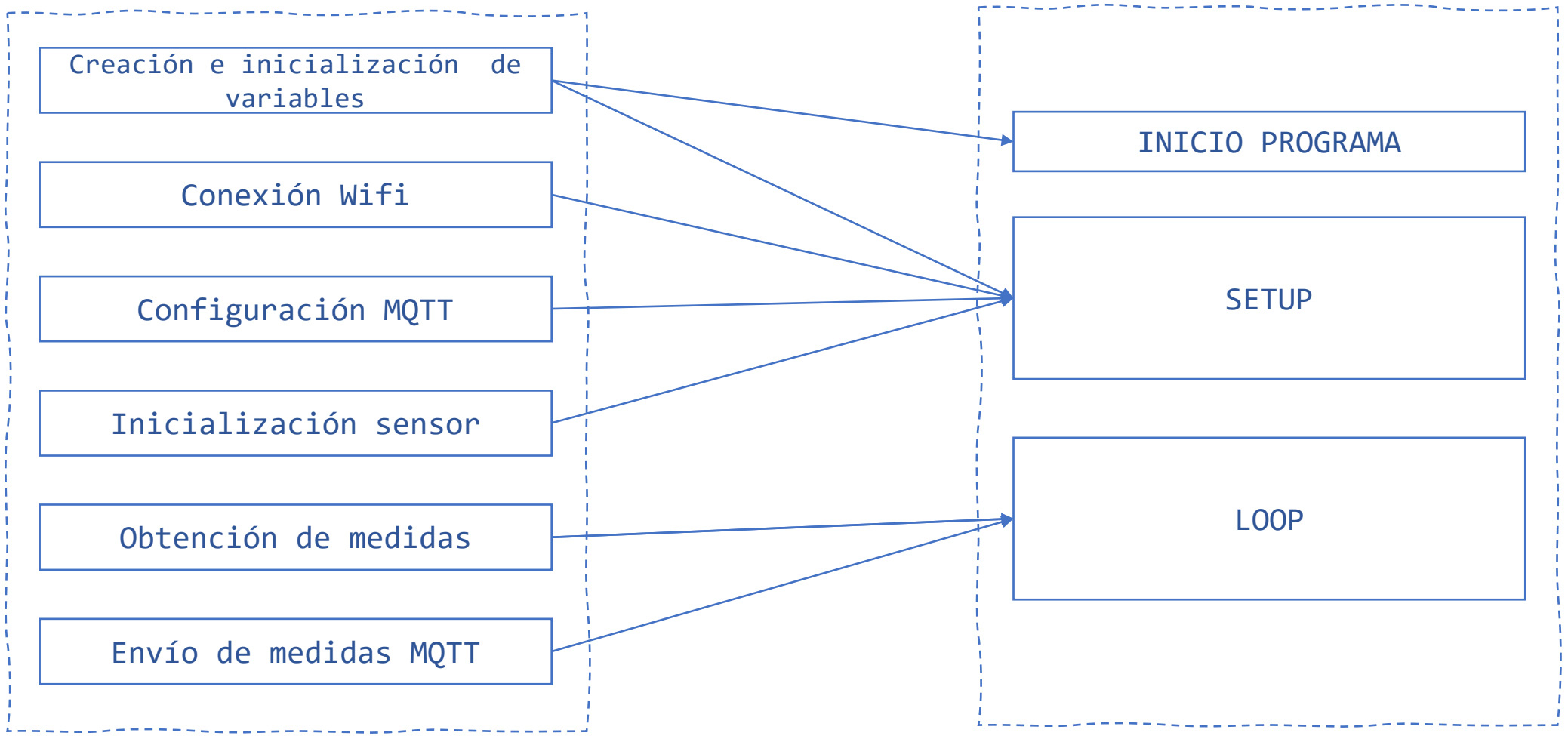
Suscriptores



Biblioteca para MQTT



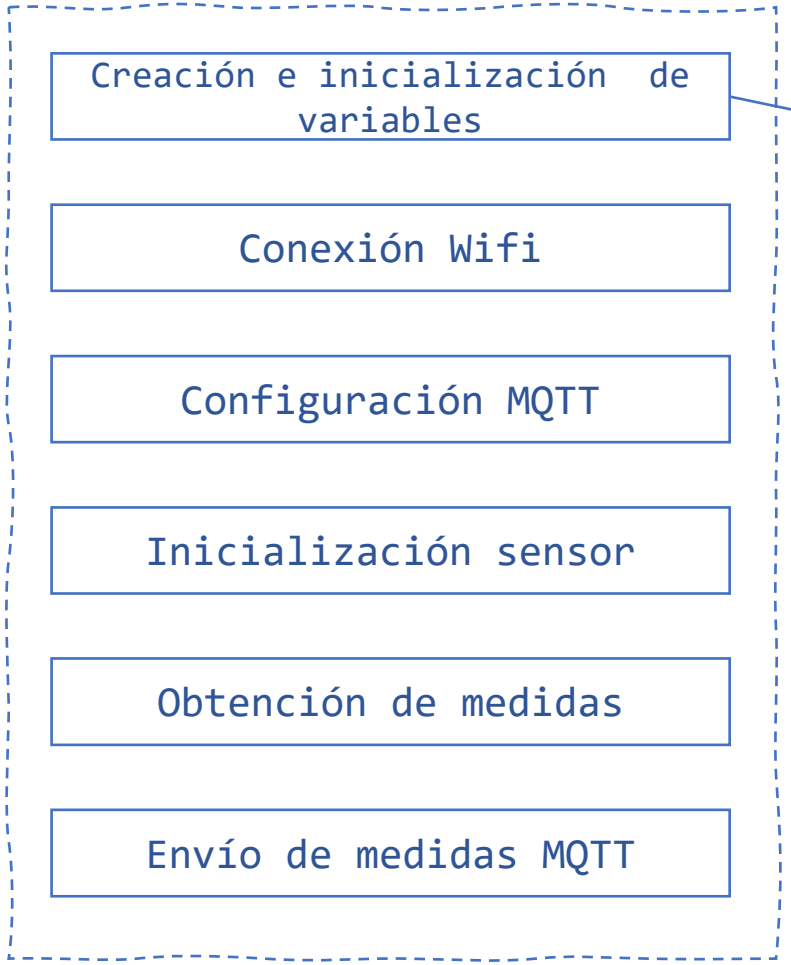
Multisensor



Estructura programa

Bloques programa Arduino

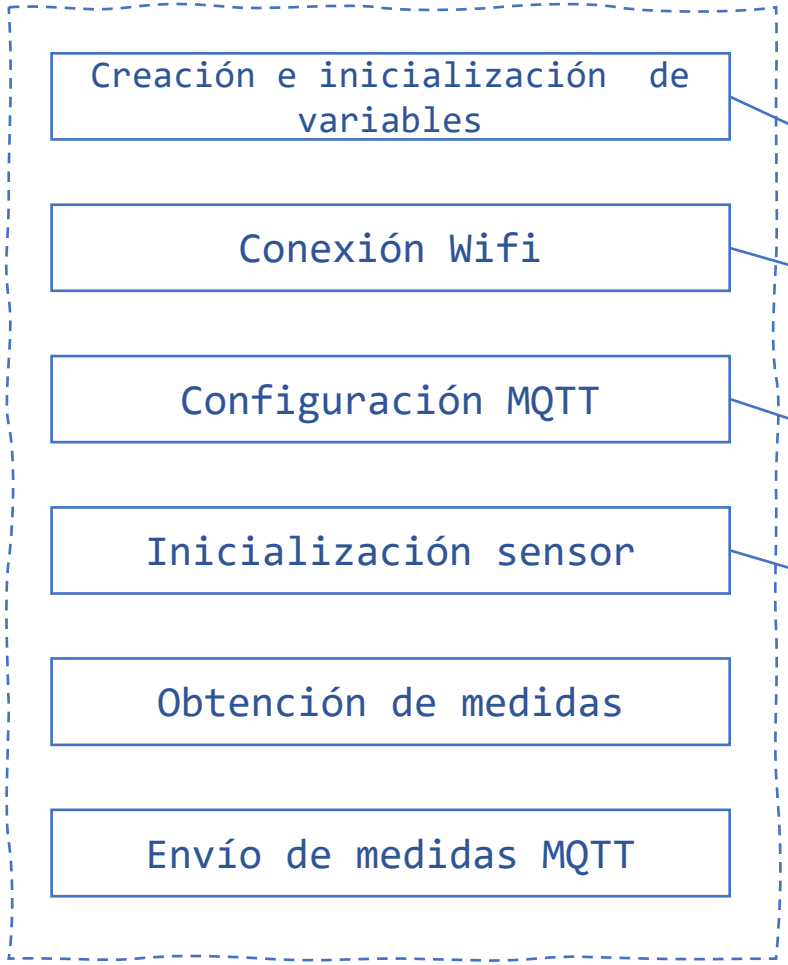
Multisensor



Estructura programa

```
#include <ESP8266WiFi.h>     // Biblioteca wifi de esp8266
#include <PubSubClient.h>    // Biblioteca cliente mqtt
#include <DHT.h>             // Biblioteca del sensor de temperatura

/* Definición de constantes */
#define MSG_BUFFER_SIZE  50
#define TIEMPO_ENTRE_MEDIDAS 10000
#define NUMERO_REINTENTOS 5
#define TIEMPO_RECONEXION 5000
#define ENCENDER LOW
#define APAGAR HIGH
#define TOPIC_PUBLICACION_TEMPERATURA "casa/dormitorio/temperatura"
#define TOPIC_PUBLICACION_HUMEDAD "casa/dormitorio/humedad"
#define TOPIC_PUBLICACION_SENSACION_TERMICA "casa/dormitorio/sensacionTermica"
#define PIN_SENSOR D2

// Parámetros de la red wifi de la casa
const char* ssid = "LabIoT1";
const char* password = "XXXXXXXX";

// Configuración Wifi - IP
IPAddress wifiIP(192, 168, 1, 50);
IPAddress wifiNET(255, 255, 255, 0);
IPAddress wifiON(192, 168, 1, 1);

// IP del broker MQTT
IPAddress mqtt_server (192, 168, 1, 37);

WiFiClient ClienteWifi;
PubSubClient clienteMQTT(ClienteWifi);   // Instanciar objeto cliente MQTT
DHT dht(PIN_SENSOR, DHT11);
char mensajeTemperatura[MSG_BUFFER_SIZE];
char mensajeHumedad[MSG_BUFFER_SIZE];
char mensajeSensacionTermica[MSG_BUFFER_SIZE];
long tiempoUltimoMensaje;
```

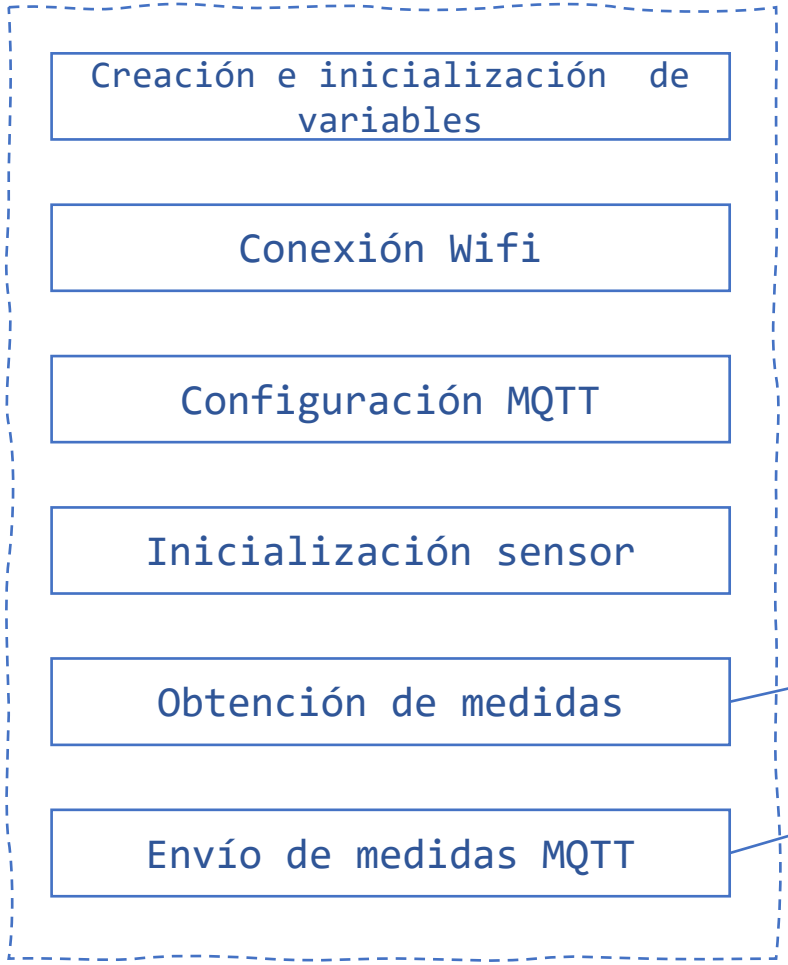
Multisensor




```
void setup(void) {

  Serial.begin(115200);

  conectar_WIFI();

  tiempoUltimoMensaje = 0;

  clienteMQTT.setServer(mqtt_server, 1883);

   // Inicialización del sensor DHT
  dht.begin();

  Serial.println("Listo");
}
```

Estructura programa

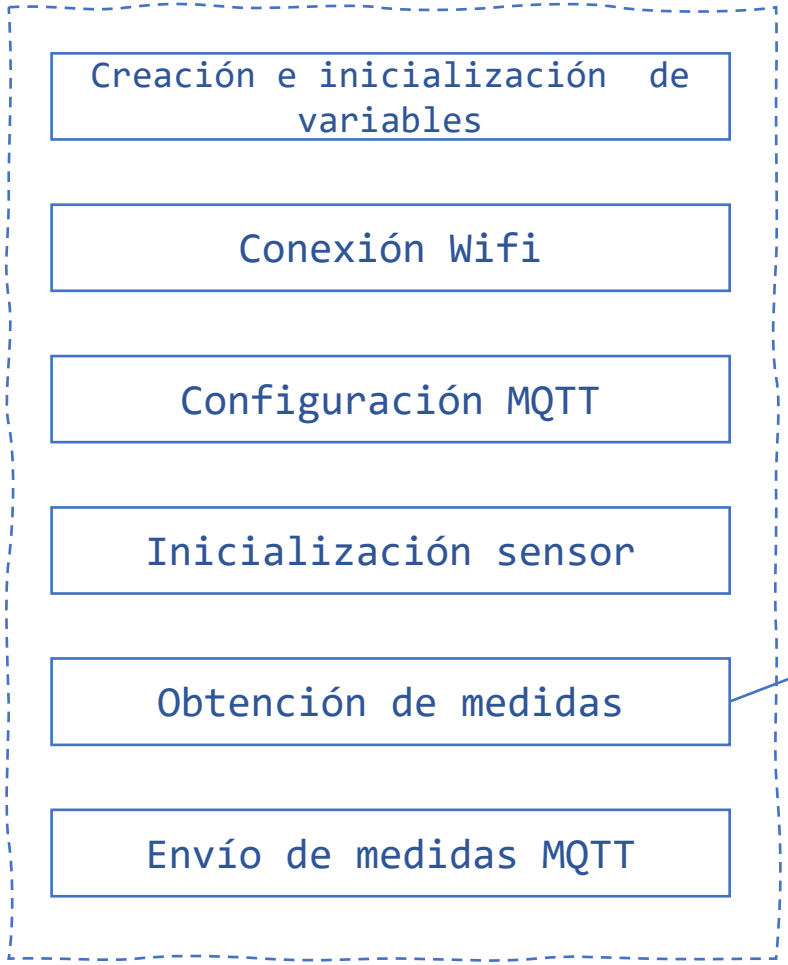
Multisensor




```
void loop(void) {

  if (!clienteMQTT.connected()) {   // Reconectar al broker
    conectar_MQTT();
  }

  long now = millis();
  if (now - tiempoUltimoMensaje > TIEMPO_ENTRE_MEDIDAS) {
    // Se actualiza el último mensaje a la referencia de tiempo actual
    tiempoUltimoMensaje = now;

    float t = medirTemperatura();
    float h = medirHumedad();
    if ( !(isnan(t)) && !(isnan(t)) ) medirSensacionTermica(t,h);
    medirSensacionTermica(t,h);
    mqtt_EnviarMensajes();
  }

}
```

Estructura programa

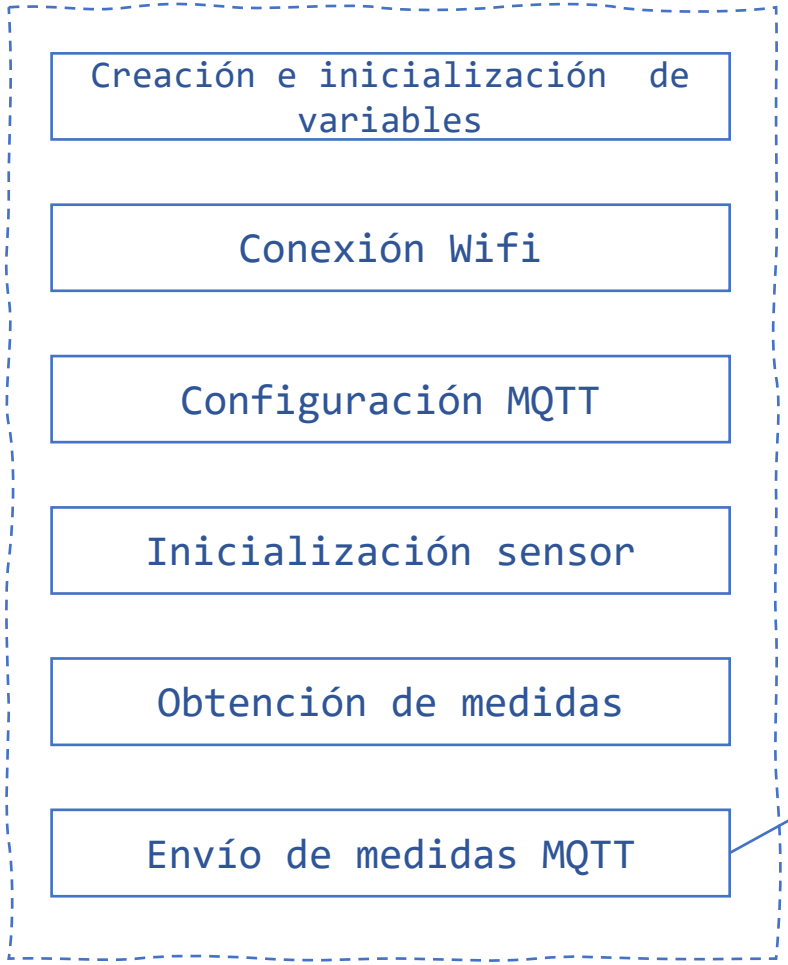
Multisensor



Estructura programa

```
float medirTemperatura() {
  // Comunicarse con el DHT11 para solicitarle el dato de la temperatura
  float temperatura = dht.readTemperature();

  if (isnan(temperatura)) {
    Serial.println("Error al leer del sensor de temperatura");
    snprintf (mensajeTemperatura, sizeof(mensajeTemperatura), "%s", "Error");
  } else {
    snprintf (mensajeTemperatura, sizeof(mensajeTemperatura), "%f", temperatura);
  }
  return temperatura;
}

float medirHumedad() {
  // Comunicarse con el DHT11 para solicitarle el dato de la humedad
  float humedad = dht.readHumidity();

  if (isnan(humedad)) {
    Serial.println("Error al leer del sensor de humedad");
    snprintf (mensajeHumedad, sizeof(mensajeHumedad), "%s", "Error");
  } else {
    snprintf (mensajeHumedad, sizeof(mensajeHumedad), "%f", humedad);
  }
  return humedad;
}

float medirSensacionTermica(float t, float h) {
  // Cálculo de la sensación térmica
  float sensacionTermica = dht.computeHeatIndex(t, h, false);

  if (isnan(sensacionTermica)) {
    Serial.println("Error al calcular la sensación térmica");
    snprintf (mensajeSensacionTermica, sizeof(mensajeSensacionTermica), "%s", "Error");
  } else {
    snprintf (mensajeSensacionTermica, sizeof(mensajeSensacionTermica), "%f", sensacionTermica);
  }
  return sensacionTermica;
}
```

Multisensor




```
void mqtt_EnviarMensajes(){
    clienteMQTT.publish(TOPIC_PUBLICACION_TEMPERATURA, mensajeTemperatura);
    clienteMQTT.publish(TOPIC_PUBLICACION_HUMEDAD, mensajeHumedad);
    clienteMQTT.publish(TOPIC_PUBLICACION_SENSACION_TERMICA, mensajeSensacionTermica);
}
```

Estructura programa

Multisensor

```
}

float medirSensacionTermica(float t, float h) {
  // Cálculo de la sensación térmica
  float sensacionTermica = dht.computeHeatIndex(t, h, false);

  if (isnan(sensacionTermica)) {
    Serial.println("Error al calcular la sensación térmica");
    snprintf (mensajeSensacionTermica, sizeof(mensajeSensacionTermica), "%s", "Error");
  } else {
```

```
Done Saving.
Writing at 0x00028000... (84 %)
Writing at 0x0002c000... (92 %)
Writing at 0x00030000... (100 %)
Wrote 290368 bytes (212238 compressed) at 0x00000000 in 18.6 seconds (effective 124.6 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

70 LOLIN(WEMOS) D1 R2 & mini, 80 MHz, Flash, Disabled (new aborts on oom), Disable