```
/*
 * BLEでI2C温度計の測定結果を送る。センサーはSTS21。
 * 1秒毎に温度を測定し、サービスキャラクタリスティックスに書き込む。
 * Notifyが有効になっていれば1秒毎に送信する事になる。無効であればReadされる度に書き込まれた内容が送りだされる。
 * 送信中はLEDを点灯。LEDはIO32を使う。
 */
//I2Cを使うためWire.hをインクルード
#include <Wire.h>

//BT系を使うためインクルード
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

BLEServer* pServer = NULL;
BLECharacteristic *pCharacteristic = NULL;
bool deviceConnected = false;
bool oldDeviceConnected = false;
//uint32_t value = 0;

# define SERVICE_UUID "0000181a-0000-1000-8000-00805f9b34fb" // org.bluetooth.service.environmental_sensing
# define CHARACTERISTIC_UUID "00002a6e-0000-1000-8000-00805f9b34fb" // org.bluetooth.characteristic.temperature 0.01℃単位

class MyServerCallbacks: public BLEServerCallbacks {
  void onConnect(BLEServer* pServer) {
    deviceConnected = true;
  }

  void onDisconnect(BLEServer* pServer) {
    deviceConnected = false;
  }
};

bool FLAG = 1;                      //タイマー割り込みが発生したら"1"を立てる。初期値が"1"の理由は、タイムアップする前にデバイスが接続された場合、
                                    //初回データがタイムアップ後になるため。
                                    //(例として、タイマーが10分に設定してあると、ESP32の起動後直ちにコネクトするとReadの結果がゼロになる)

hw_timer_t * timer = NULL;

//割込みサービスルーチン
void IRAM_ATTR LED_Blink() {        //IRAM_ATTRを付ける事により、この関数が内部RAMに配置される。
                                    //(Flashに配置されるとディレイが生じて動作に問題が起きる可能性が有るらしい)
  FLAG = 1;                         //割り込み有ったフラグを立てる
}



//温度計算
float temp_calc(){

  //変数宣言
  float ONDO;                                   //温度の計算結果
  unsigned int SENSOR_DATA;                     //温度センサーから取り出した生データ(16ビット結合)
  unsigned int cmd[2];                          //I2Cで取り出したデータを格納する配列

  //温度センサーに非ホールドマスターモードで測定トリガを送る
  Wire.beginTransmission(0x4A);                 //0x4A -> センサーのアドレス(7ビット)
  Wire.write(0xF3);                             //非ホールドマスターモードの測定トリガ
  Wire.endTransmission(true);
  delay(100);                                   //測定完了時間(＋α)待つ
  Wire.requestFrom(0x4A,2,true);                //2バイト読み込み(チェックサム省略)
  cmd[0]=Wire.read();
  cmd[1]=Wire.read();
  SENSOR_DATA=cmd[0]*256 + cmd[1];             //cmd[0]は上位8ビットなので256倍してから足し算
  ONDO=(float)SENSOR_DATA*175.72/65536-46.85;  //仕様書上の計算式に基づいて摂氏温度に換算

  Serial.printf("温度計算結果  %4.2f  (℃)¥n¥n",ONDO);

  return ONDO;                                  //計算結果をリターン値とする
}


void setup() {
  Serial.begin(115200);

  // Create the BLE Device
  BLEDevice::init("ESP32_Thermo");

  // Create the BLE Server
  pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  //Create the BLE Service
  BLEService *pService = pServer->createService(SERVICE_UUID);
```

```
  //Create the BLE Characteristics
  pCharacteristic = pService->createCharacteristic(
    CHARACTERISTIC_UUID,
    BLECharacteristic::PROPERTY_READ   |
    BLECharacteristic::PROPERTY_NOTIFY
  );

  // https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.descriptor.gatt.client_characteristic_configuration.xml
  // Create a BLE Descriptor
  pCharacteristic->addDescriptor(new BLE2902());

  // Start the service
  pService->start();

  // Start advertising
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(false);
  pAdvertising->setMinPreferred(0x0);  // set value to 0x00 to not advertise this parameter
  BLEDevice::startAdvertising();
  Serial.println("Waiting a client connection to notify...");

 //ピンモードの設定
 //I2Cポート割り当て
 Wire.begin(21,22);                                  //ピンは(SDA,SCL)の、順

 //出力ピン割り当て
 pinMode(32, OUTPUT);

 //出力ピンをオフ (Highでオン、Lowでオフ)
 digitalWrite(32, LOW);

 //割り込み用タイマー設定&スタート
 timer = timerBegin(0, 80, true);                    //timer=1us  タイマー0を80分周、カウントアップモードにする。
                                                     // 源振が80MHzなので、80分周してクロックを1uSにする。
 timerAttachInterrupt(timer, &LED_Blink, true);      //タイマー0のハンドラで、割り込み時呼び出される関数のポインタを指定し、
                                                     //割り込みタイプをエッジに指定
 timerAlarmWrite(timer, 1000000, true);              //タイマー0が100万回(1秒)数えたら割り込み、オートリロード(true)する
 timerAlarmEnable(timer);
}

void loop() {
  float ondo;
  uint8_t data_buff[2];                              //データ通知用バッファ

  if (deviceConnected) {
    if (FLAG == 1) {                                 //温度測定タイミング?
      Serial.printf("温度測定！¥n");                  //UARTに情報出力
      ondo = temp_calc();                            //温度計算する関数を呼ぶ
      digitalWrite(32, HIGH);                        //LED点灯
      data_buff[0] = (uint8_t)(((uint16_t)((ondo + 0.005) * 100)) & 0xFF);      //送るデータの小数部(温度を100倍して下位8ビットを取り出し)
      data_buff[1] = (uint8_t)((uint16_t)((ondo + 0.005) * 100) >> 8);          //送るデータの整数部(温度を100倍して上位8ビットを取り出し)
      pCharacteristic->setValue(data_buff, 2);
      pCharacteristic->notify();
      digitalWrite(32, LOW);                         //LED消灯
      FLAG = 0;                                      //割り込みフラグクリア
    }
  }

  // disconnecting
  if (!deviceConnected && oldDeviceConnected) {
    delay(500); // give the bluetooth stack the chance to get things ready
    pServer->startAdvertising(); // restart advertising
    Serial.println("start advertising");
    oldDeviceConnected = deviceConnected;
  }

  // connecting
  if (deviceConnected && !oldDeviceConnected) {
    // do stuff here on connecting
    oldDeviceConnected = deviceConnected;
  }
}
```