放射温度センサーを用いた温度監視デバイスの製作とArduino-Ethernet環境におけるOSC通信に纏わる調査

水野 史暁

# 概要

あるシステムの温度を放射温度計で測定し、ある一定の温度以上一定時間経過した場合そのシステムへの電源を遮断するシステムを製作した。

PCとマイコンは双方向に通信し、マイコンは常に放射温度計から取得した温度をPCへ送信する。PCはマイコンへコマンドを送ることで、温度異常時のステータスリセット、SSRのON, OFFを行うことができる。また同様に、異常時温度の設定や異常時温度許容秒数の設定も可能である。

# 背景

Hashilusで製作した筐体の動作試験・耐久試験を行う際に、その筐体の温度など異常ないか常時監視する必要があった。しかしながら、筐体の監視を行う仕組みがなく手動で定期的に測定していたとされている。

ここで、自動で温度を測定しロギングするシステムがあれば動作試験の自動化に有用であると考え、筐体の温度を測定しながらロギングし、また温度異常があれば筐体への電源を遮断できるシステムの開発をした。

# 技術的仕様

ハードウェア構成は以下の通りである。

## Ethernet付きマイコンボード

[Ks0304 Keyestudio W5500 ETHERNET DEVELOPMENT BOARD (WITHOUT POE)](https://wiki.keyestudio.com/index.php/Ks0304_Keyestudio_W5500_ETHERNET_DEVELOPMENT_BOARD_(WITHOUT_POE))

## テキスト, ホワイトボード が含まれている画像 自動的に生成された説明放射温度センサー

192.168.100.1

[MLX90614](https://strawberry-linux.com/pub/SEN-09570-datasheet-3901090614M005.pdf)

192.168.100.65

## SSR

[SSR-40DA](http://akizukidenshi.com/download/ds/fotek/fotek_ssr.pdf)

このシステムはローカルネットワークをもちいたUDPで通信を行う。OSC(Open Sound Control)プロトコルを用いておりオープンエンドで動的なURI風の命名規則を採用しており転送データの柔軟性がある。

192.168.100.7

## 回路図

テキスト, ホワイトボード が含まれている画像

自動的に生成された説明回路図は以下の通りである。

MLX90614は3.3Vで、５Vトレラントはないので注意されたい。Arduinoは5Vだが、2.4VでHIGHレベルなので問題なく動作するはずであり、実際に動作確認できている。

# 電子機器 が含まれている画像 自動的に生成された説明モニター, 室内, ディスプレイ が含まれている画像 自動的に生成された説明

# コマンド

このシステムは以下のOSCコマンドを送信、受信する。

1. 送信：/temp 温度測定値[℃](float32), SSRステータス[true=on, false=off](Boolean)データ送信

常時1秒毎、放射温度センサーで測定した温度を送信する。また、SSRの状態をBooleanで送信する。

1. 受信：/set/maxTemp 設定温度値[℃](Int32)

default: 50[℃]

許容最大温度が設定される。設定値より大きな温度を測定した状態かつ/set/maxTimeで設定された最大温度許容時間以上経過したときにSSRにより筐体への電源供給が遮断される。

３．受信：/set/maxTime 設定最大温度許容時間[s](Int32)

default: 5[s]

最大温度許容時間が設定される。２で設定された最大温度を越えた状態で、

で設定される最大温度許容時間が経過するとSSRが電源供給を遮断する。

４．受信：/set/reset SSRリセット[0=復帰なし, 1>=異常復帰](Int32)

default: 0

温度異常等でSSRが電源遮断した際に、再び電源供給を復帰させるための

コマンド。リセットすることによって再び温度監視をするようになる。

# 注意点

1. 電源投入時は動作が安定せず、PCへOSCメッセージが送られていないことがある。1分程度待つと動作が安定することがある。
   1. それでも動作しなかった場合はリセットをかけると動くことがある。
   2. シャイなので目を離すと動くようになっていることが多い。
   3. それでも動かなかったら一回休ませてあげると動くことがある。
2. ブレッドボードのため、配線が抜けることが多くある。注意されたい。
3. 最終日になって突然、PCからの受信ができなくなる場面が出てきた。コードは何も変更していないため謎である。時間切れとなってしまった。
4. このプログラムはなぜかTimer1, Timer2が動作しない。割り込みの使用は不可能であった。ArduinoOSCのどこかでTimerリソースを使っている可能性もあるがそのような記述は見つけられなかった。

# 使用したライブラリやOSCの雑感

ライブラリは怖い部分が多いと感じた。公開されているものの、例が少ない、そもそもコンパイルが通らないといった問題が多くあった。これまでに試したライブラリは以下のものである。

1. [ArduOSC](https://github.com/recotana/ArdOSC)

そもそもコンパイルが通らないライブラリ。socket.hが足りないと言われるがこのsocket.h自体はArduino1.4xで廃止されているものであり、このライブラリを今更使うのは良くないと判断した。

1. [ArduinoOSC](https://github.com/hideakitai/ArduinoOSC)

これをメインに使った。EthernetシールドでもESP-Arduinoで動かすWiFiModeでもSerialでも動くと謳っているが、実際はExampleのコンパイルが通らなかった。Arduino -AVRの中にあるサンプルのみコンパイルが通るのでこれをベースに今回の開発を進めた。

しかしながら、電源投入してしばらくはosc.send()で3秒くらいかかるという謎な挙動をする。（PCには届いていない）しばらくするとちゃんとすぐにデータが送信できるようになる。（PCには届く）謎である。もしかしたら再送処理のようなことをしているのかもしれない。（ただしOSCにもUDPにも再送処理のプロトコルはない）

サンプルも動いたり動かなかったりとするためあまり信頼できない。

Ethernetではなく、ESP-Arduinoを用いたWiFiでOSC通信を行った場合はなんの問題もなく完璧にプログラムが動作した。Ethernet互換ボードとの相性が悪かった可能性も考えられる。

（FYI: ArduinoEthernetシールド純正はW5100チップを搭載しているが、この互換ボードはW5500を搭載している。因果関係はわからないが、W5500チップを搭載しているボードはEthernet関連のLibraryがうまく動作しなかったというフォーラムの投稿がみられた。）

1. [OSC](https://github.com/CNMAT/OSC)

もっともポピュラーなライブラリと言われている。LibraryをSurveyしたときは様々なマイコンに対応しつつもコンパイルが通らず、非常に複雑な構造をしていたため、深く調べるのを諦めた。が実は一番良く出来ているLibraryだった。最終日にこのLibraryを移植して動作させたところ、マイコンからの送信はうまく行った。しかし受信がOSC Bundleと呼ばれる形式でないと受信ができないようで、鈴木に対応を頼んだが時間切れということで使うことができなかった。

もし、またOSCをチャレンジする機会があったらこのLibraryを使ってBundle形式で通信を行いたい。

（FYI: Bundleとは時刻や送信データ形式やバイト数などを梱包した形式で、いまやっているMessageという形式よりも信頼性が高いと言われる。）

# 参考文献

開発したコード

<https://github.com/fumimaker/Hashilus_temperature_ethernet>

以下、ソースコード: ./Hashilus\_temperature\_ethernet.inoを添付する。

Hashilus\_temperature\_ethernet.ino

|  |
| --- |
| #include <ArduinoOSC.h>  #include <Wire.h>  #include <Adafruit\_MLX90614.h>  *//#include <Ethernet.h>*  *//#include <SPI.h>*  *//#include <MsTimer2.h>*  *// Ethernet stuff*  const IPAddress ip(192, 168, 100, 65);  uint8\_t mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};  *// for ArduinoOSC*  OscEthernet osc;  Adafruit\_MLX90614 mlx = Adafruit\_MLX90614();  const char\* host = "192.168.100.7";  *//const char\* host = "192.168.100.155";*  const int32\_t recv\_port = 9999;  const int32\_t send\_port = 9999;  int32\_t maxTemp = 50, maxTime = 5;  int32\_t counter = 0, tempCounter = 0;  uint32\_t delta\_time = 0;  boolean SSR\_status = false;  void subscribeTrigger(void){  osc.subscribe("/need/reply", [](OscMessage &m) {  Serial.print("/need/reply: ");  Serial.print(m.ip());  Serial.print(" ");  Serial.print(m.port());  Serial.print(" ");  Serial.print(m.size());  Serial.print(" ");  Serial.print(m.address());  Serial.print(" ");  Serial.print(m.getArgAsInt32(0));  Serial.println();  Serial.println();  maxTime = m.getArgAsInt32(0);  Serial.println(maxTime);  });  osc.subscribe("/set/maxTemp", [](OscMessage &m) {  Serial.print("/set/maxTemp: ");  Serial.print(m.ip());  Serial.print(" ");  Serial.print(m.port());  Serial.print(" ");  Serial.print(m.size());  Serial.print(" ");  Serial.print(m.address());  Serial.print(" ");  Serial.print(m.getArgAsInt32(0));  Serial.println();  Serial.println();  maxTemp = m.getArgAsInt32(0);  Serial.println(maxTemp);  });  osc.subscribe("/set/maxTime", [](OscMessage &m) {  Serial.print("/set/maxTime: ");  Serial.print(m.ip());  Serial.print(" ");  Serial.print(m.port());  Serial.print(" ");  Serial.print(m.size());  Serial.print(" ");  Serial.print(m.address());  Serial.print(" ");  int32\_t tmp = m.getArgAsInt32(0);  Serial.print("Set max temperature: ");  Serial.print(tmp);  Serial.println();  Serial.println();  maxTime = tmp;  });  osc.subscribe("/set/reset", [](OscMessage &m) {    Serial.print("/set/maxTemp: ");  Serial.print(m.ip());  Serial.print(" ");  Serial.print(m.port());  Serial.print(" ");  Serial.print(m.size());  Serial.print(" ");  Serial.print(m.address());  Serial.print(" ");    int32\_t tmp = m.getArgAsInt32(0);    Serial.print("Set max time: ");  Serial.print(tmp);  Serial.println();  Serial.println();    if(tmp>0){  SSR\_status = true;  }  else{  SSR\_status = false;  }  });  }  void setup(){  pinMode(2, OUTPUT);  digitalWrite(2, LOW);    Serial.begin(115200);    Serial.println("Initialize Ethernet with DHCP:");  if (Ethernet.begin(mac) == 0){  Serial.println("Failed to configure Ethernet using DHCP");  if (Ethernet.hardwareStatus() == EthernetNoHardware){  Serial.println("Ethernet shield was not found. Sorry, can't run without hardware. :(");  }  else if (Ethernet.linkStatus() == LinkOFF){  Serial.println("Ethernet cable is not connected.");  }  while (true){  delay(1);  }  }  Serial.print("My IP address: ");  Serial.println(Ethernet.localIP());  Ethernet.begin(mac, ip);  subscribeTrigger();  osc.begin(recv\_port);  mlx.begin();  }    void loop(){  if((millis() - delta\_time) > 1000){  Serial.println(millis() - delta\_time);  float objectTemp = mlx.readObjectTempC();  float ambientTemp = mlx.readAmbientTempC();  Serial.println(objectTemp);    OscMessage msg(host, send\_port, "/temp");  msg.push(objectTemp).push(SSR\_status);  osc.send(msg);      if(objectTemp > maxTemp){  tempCounter++;  Serial.print("Over temperature ++ ");  Serial.println(tempCounter);  }  if(tempCounter > maxTime){  SSR\_status = false;  tempCounter = 0;  Serial.print("tempCounter > maxTime: ");  Serial.println(SSR\_status);  }    digitalWrite(2, SSR\_status);  delta\_time = millis();  }  osc.parse();  } |