```
/*
 feather_timer_7
 26.10.16
 converted from CH_timer_29


 TIMER PROGRAM FOR Adafruit Feather CENTRAL HEATING or similar


 temp102 by Arduino Playground author
 timer control developed 4.7.15 onwards by Julian Rogers

 */

//timer with two ons and offs for water and heat manual advance and manual over-ride
//Adafruit Feather MO with Adafruit Adalogger SD + RTC
//temperature by TMP102 (sockets for 2 wired sensors - could use network connected sensors
//clock is pcf8523
//clock should be set to GMT, conversion to BST is by software
//reads manual switch positions
//turns on status LEDs as appropriate
//water and heat LEDs will flash if manual off during an "on" timed period
//on off times and thermostat value stored on SD card
//communication via UDP
//timer times, clock setting, thermostat setting set remotely by UDP
//includes data logger function now enough RAM available
//refer to pcf8523 and WiFiUdpSendReceiveString2

/*
 STATUS LIGHTS

 Switch | Timed On | Timed Off
 _____________________________
  T    | Cont Red |   Off
 _____________________________
  O    |Flash Red |   Off
 _____________________________
  C    |Cont Red  | Cont Red    this needs to be confirmed - bit more code needed! - maybe slow flash?
 _____________________________

 OUTPUT FROM SERIAL MONITOR
 Cryptic to reduce RAM usage!
 T plus temp * 10, S or G (Summer or GM time) plus hh:mm, on/off times, ON or OFF depending
 whether roomstat setting is satisfied.

 OUTPUT FROM UDP
 Cryptic to reduce RAM usage!
 time in minutes, / plus on/off times, T plus temp * 10, S plus switch code,
 A plus advance code

*/

//Assignment of Feather pins:

//serial RX (not connected)
//TX1(serial comms TX) (not connected)
//D2 = advance LED - will flash if advance is activated
//A4 = water LED (B)
//D10 = used by  SD card chip select
//A3 = heat LED (A)
//D11 = water advance button B
//D9 = heat advance button A
//D13 = heat relay A
//D12 = water relay B
//SPI (Ethernet Shield, SD card)
//SPI
//SPI
//SPI
//
//D16 = output to self-reset (via an NPN transistor)

//A0 special adc / dac pin (D14) advance led
//A1 = gives water switch position (B)
//A2 = (was self reset) (D16)
//A3 = on led A (D17)
//A4 = on led B (D18)
//A5 = gives heat switch position (A)
//
//
```

```
//I2C SDA (clock and temp sensors}
//I2C SCL

//Libraries:

#include <Adafruit_WINC1500.h>
#include <Adafruit_WINC1500Udp.h>
#include "RTClib.h"
#include <SPI.h>
#include <Wire.h>
#include <SD.h>

#define TMP102_I2C_ADDRESS 0x48 // I2C address TMP102 A0 to GND (0x48 = 72 = 1001000 for GND, 73 for vcc)
#define TMP102_I2C_ADDRESS_2 0x49  //second tmp102 not yet implemented

// Define the WINC1500 board connections below.
// If you're following the Adafruit WINC1500 board
// guide you don't need to modify these:
#define WINC_CS   8     // chip select for wifi
#define WINC_IRQ  7     // irq for wifi
#define WINC_RST  4     // reset pin for wifi - controlled by library
#define WINC_EN   2     // or, tie EN to VCC and comment this out - high to enable wifi
// The SPI pins of the WINC1500 (SCK, MOSI, MISO) should be
// connected to the hardware SPI port of the Arduino.
// On an Uno or compatible these are SCK = #13, MISO = #12, MOSI = #11.
// On an Arduino Zero use the 6-pin ICSP header, see:
//   https://www.arduino.cc/en/Reference/SPI

// Setup the WINC1500 connection with the pins above and the default hardware SPI.
Adafruit_WINC1500 WiFi(WINC_CS, WINC_IRQ, WINC_RST);

// Or just use hardware SPI (SCK/MOSI/MISO) and defaults, SS -> #10, INT -> #7, RST -> #5, EN -> 3-5V
//Adafruit_WINC1500 WiFi;
int status = WL_IDLE_STATUS;
char ssid[] = "network_name";  //  insert your network SSID (name)
char pass[] = "password";       // insert your network password
int keyIndex = 0;            // your network key Index number (needed only for WEP)

unsigned int localPort = 2390;      // local port to listen on

char packetBuffer[255]; //buffer to hold incoming packet
char  ReplyBuffer[255];       // a string to send back

Adafruit_WINC1500UDP Udp;
RTC_PCF8523 rtc;



//global variables:

byte advCode; //holds manual advance status

int tim;      //current (hours x 60 + minutes) for daily timed periods
int setTemp = 180;  //180 is default temp for thermostat if SD card fails
const byte hysteresis = 3;  //used in thermostat function - is this enough?

char gmtBst[2];   //holds GMT or BST

//boolean sndRec = false;

boolean newChangeH = false;
boolean newChangeW = false;
boolean oldChangeH = false;
boolean oldChangeW = false;
boolean advanceH = false;
boolean advanceW = false;

boolean heat1 = false;
boolean heat2 = false;
boolean water1 = false;
boolean water2 = false;
boolean heat = false;
boolean water = false;

boolean dataAdded = true;
//needed for data logging which does not work on Uno - not enough memory!
//can be implemented on Feather
```

```
char hourStr[7];
char dayStr[7];
char monthStr[7];
char clockSetString[32];
char onOffString[48];
char newonOffString[64];
char tempSetString[4];
char tNowStr[7];
byte sStart[5];
byte sEnd[5];

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
int yr, mon, dofmon, dofwk, hr, mn, sec ;

File myFile;

////////////////////////////////////////////////////////////////////

void setup() {
  pinMode(14, OUTPUT);       // advance led
  pinMode(13, OUTPUT);       // rly A
  pinMode(12, OUTPUT);       //rly B
  digitalWrite(14, LOW);
  digitalWrite(13, LOW);
  digitalWrite(12, LOW);
  pinMode(17, OUTPUT);       // on led A
  pinMode(18, OUTPUT);       //on led B
  digitalWrite(17, LOW);
  digitalWrite(18, LOW);
  //pinMode(2, OUTPUT);      // adv led No! used by wifi module
  pinMode(11, INPUT_PULLUP);  // adv B
  pinMode(9, INPUT_PULLUP);   // adv A
  pinMode(16, OUTPUT);       // could be self reset or motorised valve
  digitalWrite(16, LOW);


   Serial.begin(9600);
/*
  //remove for independent operation
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

 */

   delay(5000);

  #ifdef WINC_EN
    pinMode(WINC_EN, OUTPUT);
    digitalWrite(WINC_EN, HIGH);
  #endif

  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue:
    while (true);
  }

  // attempt to connect to Wifi network:
  while ( status != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }
  Serial.println("Connected to wifi");
  printWifiStatus();
```

```
// load dates for summer time
// start 2015, end 2018
// start in March, end in October

sStart[0] = 29;
sStart[1] = 27;
sStart[2] = 26;
sStart[3] = 25;

sEnd[0] = 25;
sEnd[1] = 30;
sEnd[2] = 29;
sEnd[3] = 28;


Wire.begin();
rtc.begin();

Serial.println("Initializing SD card...");

pinMode(10, OUTPUT); // SD card chip select

if (!SD.begin(10)) {
  Serial.println("SD initialization failed!");
  return;
}
Serial.println("SD initialization done.");

Serial.println("\nStarting connection to server...");
  // if you get a connection, report back via serial:
Udp.begin(localPort);

//////////////////////////////////////////////////
 // open file and get values
myFile = SD.open("TEMPROG1.txt");
if (myFile) {
  Serial.println("TEMPROG1.txt:");

  // read from the file until there's nothing else in it:
  byte index = 0;
  while (myFile.available()) {

    tempSetString[index] = myFile.read();
    index++;
    Serial.println("reading..");
  }
  // close the file:
  myFile.close();

  if(index != 3){
    Serial.println("SD data error");
  }
  else{
    setTemp = atoi(tempSetString);

    }

  //setTemp = atoi(tempSetString);
  }
//////////////////////////////////////////////////
// open file and get values
myFile = SD.open("CHPROG1.txt");
if (myFile) {
  Serial.println("CHPROG1.txt:");

  // read from the file until there's nothing else in it:
  byte index = 0;
  while (myFile.available()) {

    newonOffString[index] = myFile.read();
    index++;
    //Serial.println("reading..");
  }
  // close the file:
  myFile.close();

  if(index != 47){
    Serial.println("SD data error");
  }
```

```
    else{
      for(index = 0; index < 48; index++){
        onOffString[index] = newonOffString[index];
      }
     }

    //for(index = 0; index < 48; index++){
      // onOffString[index] = newonOffString[index];
    //}


  }
}

// End of setup()


//Functions start here

void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}


////////////////////////////////////////////////////////////////////////////////

//function to extract values from onOffString and convert to minutes
int getTimes(byte index) {
  char selectorString[3];
  selectorString[0] = onOffString[index];
  index++;
  selectorString[1] = onOffString[index];
  int result = atoi(selectorString);
  index++;
  index++;
  selectorString[0] = onOffString[index];
  index++;
  selectorString[1] = onOffString[index];
  result = result*60 + atoi(selectorString);
  return result;
}


////////////////////////////////////////////////////////////////////////
//function to extract values from clockSetString
//see function "adjClock()"
byte getClock(byte index) {
  char selectorString[3];
  selectorString[0] = clockSetString[index];
  index++;
  selectorString[1] = clockSetString[index];
  int result = atoi(selectorString);

  return result;
}
////////////////////////////////////////////////////////////////////////

//function to return the two switch positions coded to numbers 0 to 8
//see documentation / circuit diagram for explanation
//analog val for continuous is >500
//analog val for off is < 50
//analog value for timed is somewhere in between 50 and 500 (approx 317)

byte getSwitchPositions() {
  int valHeat = analogRead(5);  // *changed
  int valWater = analogRead(1); // *changed
```

```
  // 1 = timed, 2 = off, 3 = continuous
  if(valHeat < 50){
    valHeat = 1; // timed
  }
  else if(valHeat > 700){
    valHeat = 2; // off
  }
  else {
    valHeat = 3; // continuous
  }
  if(valWater < 50){
    valWater = 1;
  }
  else if(valWater > 700){
    valWater = 2;
  }
  else {
    valWater = 3;
  }

  return valWater * 3 + valHeat - 4;
}

////////////////////////////////////////////////////////////////////////////
// determine whether it's BST or GMT
boolean isItSummer(byte yr, byte mon, byte dofmon, byte hr) {
  boolean summer = false;
  yr = yr - 15; // list of dates starts in 2015, array index starts at 0
  byte startDate = sStart[yr];
  byte endDate = sEnd[yr];

  if(mon > 3 && mon < 10) {
    summer = true;
  }
  if(mon == 3 && dofmon > startDate) {
    summer = true;
  }
  if(mon == 3 && dofmon == startDate && hr > 1){
    summer = true;
  }
  if(mon == 10 && dofmon < endDate){
    summer = true;
  }
  if(mon == 10 && dofmon == endDate && hr < 2){
    summer = true;
  }
  return summer;

}
////////////////////////////////////////////////////////////////////////////

int getTemp102(byte ADD_TMP102){
  byte firstbyte, secondbyte; //these are the bytes we read from the TMP102 temperature registers
  int val; //an int is capable of storing two bytes, this is where we "chuck" the two bytes together.

  float convertedtemp; //We then need to multiply our two bytes by a scaling factor, mentioned in the datasheet.

  //float correctedtemp;
  // The sensor overreads? I don't think it does!


  //Reset the register pointer (by default it is ready to read temperatures)
  //You can alter it to a writeable register and alter some of the configuration -
  //the sensor is capable of alerting you if the temperature is above or below a specified threshold.

  Wire.beginTransmission(ADD_TMP102); // start talking to sensor
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.requestFrom(ADD_TMP102, 2);
  Wire.endTransmission();


  firstbyte      = (Wire.read());
  //read the TMP102 datasheet - here we read one byte from
  //each of the temperature registers on the TMP102
  secondbyte     = (Wire.read());
  //The first byte contains the most significant bits, and
  //the second the less significant
```

```
  val = firstbyte;
  if ((firstbyte & 0x80) > 0) {
    val |= 0x0F00;
  }
  val <<= 4;
  //MSB
  val |= (secondbyte >> 4);
  // LSB is ORed into the second 4 bits of our byte.

  convertedtemp = val*0.625; // temp x 10
  //correctedtemp = convertedtemp - 0;   //should be 5 according to playground author
  int temp = (int)convertedtemp;
  return temp;
}


////////////////////////////////////////////////////////////////////////////
//function incorporates a thermostatic function

boolean thermostat(int targetT){
  boolean heating;

  int tmp = getTemp102(TMP102_I2C_ADDRESS);
  //Serial.println(tmp);
  if(tmp >= (targetT + hysteresis)){
    heating = false;
  }
  if(tmp < (targetT - hysteresis)){
    heating = true;
  }
  return heating;
}

////////////////////////////////////////////////////////////////////////////
// adjusts clock
void adjClock(){
  byte mn = getClock(0);
  byte hr = getClock(3);
  byte dofwk = getClock(6);
  byte dofmon = getClock(9);
  byte mon = getClock(12);
  int yr = getClock(15);

  Serial.println("Adjusting clock! - New times are:");
  Serial.println(mn);
  Serial.println(hr);
  Serial.println(dofwk);
  Serial.println(dofmon);
  Serial.println(mon);
  yr = yr + 2000;
  Serial.println(yr);


  //rtc.adjust(DateTime(2016, 10, dofmon, hr, mn, 0));
  rtc.adjust(DateTime(yr, mon, dofmon, hr, mn, 0));

}



////////////////////////////////////////////////////////////////////////////

void loop() {

  int tempNow = getTemp102(TMP102_I2C_ADDRESS);
  Serial.print("T");
  Serial.print(tempNow);
  Serial.print(" deg C");
  itoa(tempNow, tNowStr, 10);
  Serial.println(tNowStr);
  int tempNow2 = getTemp102(TMP102_I2C_ADDRESS_2);
  Serial.print("second sensor reads ");
  Serial.println(tempNow2);
  // Here is where the progam checks to see if data is being sent
  // and sends back status data
  // if there's data available, read a packet
  int packetSize = Udp.parsePacket();
  if (packetSize)
```

```
{
 Serial.print("Received packet of size ");
 Serial.println(packetSize);
 Serial.print("From ");
 IPAddress remoteIp = Udp.remoteIP();
 Serial.print(remoteIp);
 Serial.print(", port ");
 Serial.println(Udp.remotePort());

 // read the packet into packetBufffer
 int len = Udp.read(packetBuffer, 255);
 if (len > 0) packetBuffer[len] = 0;

 Serial.println("Contents:");
 Serial.println(packetBuffer);
 for(int x = 0; x < (len + 1); x++){
  ReplyBuffer[x] = packetBuffer[x];
 }




 if(packetSize == 47 || packetSize == 17 || packetSize == 4 || packetSize == 3 || packetSize == 2){

  //get new timer on/off times and save to SD card
  if(packetSize == 47){
  Serial.println("saving on/off times to SD card");
  //save to SD card
  SD.remove("CHPROG1.txt");    //first delete previous file
  Serial.println("deleted previous file");
    myFile = SD.open("CHPROG1.txt", FILE_WRITE);
  if (myFile) {
    myFile.print(packetBuffer);
    //int numChars;
    //numChars = myFile.print(packetBuffer);
    //Serial.write("number of characters written is ");
    //Serial.println(numChars);
    myFile.close();
    for (byte x = 0; x<48; x++){
      onOffString[x] = packetBuffer[x];
    }


  }

  else {
    Serial.println("SD - failed to write data!");
    //add error to UDP transmission
  }
  Serial.println("finished writing new on/off times");
 }

 ////////////////////////////////////////////////////////////
 //get new clock settings
 //format: mm,hh,dd,dd,mm,yy
 if(packetSize == 17){
   for(byte x = 0; x < 18; x++){
     clockSetString[x] = packetBuffer[x];
   }
   Serial.println(clockSetString);

  adjClock();
 }
 ////////////////////////////////////////////////////////////

 //reset Feather?
 //if(packetSize == 4){
  //digitalWrite(16, HIGH);  // thats all folks!
// }

 ////////////////////////////////////////////////////////////
 //get new thermostat setting
 //format: ttt (TdegC x 10)
 if(packetSize == 3){

   setTemp = atoi(packetBuffer);

  Serial.println("new thermo setting: ");
```

```
    Serial.println(setTemp);

    //save to SD card
    SD.remove("TEMPROG1.txt");    //first delete previous file

      myFile = SD.open("TEMPROG1.txt", FILE_WRITE);
    if (myFile) {

      //myFile.print(tempSetString);
      myFile.print(packetBuffer);

      myFile.close();



    }


    else {
      Serial.println("SD - failed to write data!");
      //add error to UDP transmission
    }


  }
////////////////////////////////////////////////////////////////////////////
    //respond to advance command
    if(packetSize == 2){

     if(packetBuffer[0] == 'Y'){
       advanceH = true;
     }
     else{
       advanceH = false;
     }

     if(packetBuffer[1] == 'Y'){
       advanceW = true;
     }
     else{
       advanceW = false;
     }

     }

    }
    // send a reply, to the IP address and port that sent us the packet we received
    int switches = getSwitchPositions();
    char swStr[7];
    itoa(switches, swStr, 10);
    char tNowStr[7];
    itoa(tempNow, tNowStr, 10);
    char setTempStr[7];
    itoa(setTemp, setTempStr, 10);
    char advCodeStr[7];
    itoa(advCode, advCodeStr, 10);


    char timStr[7];
    itoa(tim, timStr, 10);

    Serial.println("sending reply from remote...");
    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());

    Udp.write(timStr);
    Udp.write("/");
    Udp.write(onOffString);
    Udp.write("T");
    Udp.write(tNowStr);
    Udp.write("S");
    Udp.write(swStr);
    Udp.write("A");
    Udp.write(advCodeStr);
    Udp.write("t");
    Udp.write(setTempStr);

    //should develop this to report status, errors etc
    Udp.endPacket();
```

```
}

// End of section checking whether data is being sent
//////////////////////////////////////////////////////////////////////


//displayTime();
Serial.println();

//retrieve data from pcf8523

   DateTime now = rtc.now();

   yr = int(now.year());
   mon = int(now.month());
   dofmon = int(now.day());
   dofwk = int(now.dayOfTheWeek()) + 1;
   hr = int(now.hour());
   mn = int(now.minute());
   sec = int(now.second());


  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.print(now.day(), DEC);
  Serial.print(" (");
  Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
  Serial.print(") ");
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.print(now.second(), DEC);
  Serial.println();


//////////////////////////////////////////////////////////////////////




boolean summer;
summer = isItSummer(yr,mon,dofmon,hr);
if(summer == true){


  strcpy(gmtBst, "S");

  if(hr == 23){
    hr = 0;
  }
  else {
    hr = hr + 1;
  }

}
else{

  strcpy(gmtBst, "G");

}
Serial.print(gmtBst);
Serial.print(hr);
Serial.print(":");
Serial.print(mn);
Serial.println();
Serial.println(onOffString);

tim = hr*60 + mn;

itoa(hr, hourStr, 10);
itoa(dofmon, dayStr, 10);
itoa(mon, monthStr, 10);

/*
```

```
  if(sndRec){
    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    Udp.write(" T: ");
    Udp.write(hourStr);
    Udp.write(":");
    Udp.write(minuteStr);
    Udp.write(" ");
    Udp.write(gmtBst);
    Udp.endPacket();
    sndRec = false;
  }
  */
  ////////////////////////////////////////////////////////////////////

  //data log section
  //data logged every hour, one minute past hour
  if(mn == 1 && dataAdded == false){
 //save to SD card


       myFile = SD.open("DATALOG1.txt", FILE_WRITE);

      if (myFile) {
        myFile.print(hr);
        myFile.print(":");
        myFile.print(dayStr);
        myFile.print(":");
        myFile.print(monthStr);
        myFile.print(":");
        myFile.print(tNowStr);
        myFile.print(",");


        }

        dataAdded = true;
      /*
       if(!myFile) {
        Serial.println("SD - DLog failed to write!");
        //add error to UDP transmission
       }
      */

    }
    myFile.close();

  if(mn != 1){
    dataAdded = false;
  }



  ////////////////////////////////////////////////////////////////////
  //Main logic for interaction with hardware starts here!

  // ensures valve is deactivated at end of day (if D16 is free)

  /*
  if(tim == 0){
    digitalWrite(16, HIGH);
  }
  else{
    digitalWrite(16, LOW);
  }
  */

  //initialise variables holding on/off times


  int waterOn1 = getTimes(0);
  int waterOff1 = getTimes(6);
  int waterOn2 = getTimes(12);
  int waterOff2 = getTimes(18);

  int heatingOn1 = getTimes(24);
  int heatingOff1 = getTimes(30);
  int heatingOn2 = getTimes(36);
  int heatingOff2 = getTimes(42);
```

```
digitalWrite (14, LOW);        //advance LED *changed
// get switch positions
// if off or cont, LEDs will flash during a timed period set in memory
int valHeat = analogRead(5);  // *changed
int valWater = analogRead(1); // *changed

if(valHeat >50)     //off or cont
{

  digitalWrite(17,LOW);      //heat LED off - LED will flash during an "on" timed period *changed
  delay(200);
}
if(valWater >50)   //off or cont
{

  digitalWrite(18,LOW);    //water LED off - LED will flash during an "on" timed period *changed
  delay(200);
}

////////////////////////////////////////////////////////////////

//Check to see if advance button (heating) is pressed and
//check times to see if boiler should be on
//set various boolean flags accordingly viz. heat1, heat2 advanceH, oldChangeH and newChangeH

byte val = digitalRead(9);    //heat advance button changed
delay(100);

if (val == LOW){
  advanceH = true;

}
oldChangeH = newChangeH; //used to detect status chages due to the timer
                         //to decide if advance is valid

if (tim >= heatingOn1 && tim < heatingOff1)
{
   heat1 = true;
}
else
{
  heat1 = false;
}

if (tim >= heatingOn2 && tim < heatingOff2)
{
   heat2 = true;
}
else
{
  heat2 = false;
}

////////////////////////////////////////////////////////////////////////
//Do the same for water

val = digitalRead(11);   //water advance button *changed
delay(100);
if (val == LOW){
  advanceW = true;

}
oldChangeW = newChangeW;

if (tim >= waterOn1 && tim < waterOff1)
{
   water1 = true;
}
else
{
  water1 = false;
}
if (tim >= waterOn2 && tim < waterOff2)
{
   water2 = true;
}
else
{
  water2 = false;
```

```
}

//////////////////////////////////////////////////
//Work out if heat should be on according to timed periods and advance

if (heat1 == true || heat2 == true)
{
  heat = true;
}
else
{
 heat = false;
}
newChangeH = heat;

if (newChangeH != oldChangeH)       //on/off status has changed
{
 advanceH = false;
}
if (advanceH)
{
 heat = !heat;     //if advance is true, negate the boiler state set by the state of the timer
}

// check to see if boiler should be on comparing temp sensor against target temp

if (heat)
{
 digitalWrite(17, HIGH); // heat LED *changed
 boolean thermo = thermostat(setTemp);
 if(thermo){         // if temperature below target less hysteresis
   digitalWrite(13,HIGH); // heat relay on *changed
   Serial.println("ON");
  }

 if(!thermo) {
   digitalWrite(13, LOW); // heat relay off *changed
   Serial.println("OFF");
 }

}
if(!heat)
{
 digitalWrite(13, LOW); // heat relay off *changed
 digitalWrite(17, LOW); // heat LED off *changed
}
//////////////////////////////////////////////////
//Similarly for water except that, with regard to temperature,
//the themostat is wired into the boiler circuitry
//and not subject to computer control

if (water1 == true || water2 == true)
{
  water = true;
}

else
{
 water = false;
}
newChangeW = water;
//check to see if advance is still valid
if (newChangeW != oldChangeW)
{
 advanceW = false;
}
if (advanceW)
{
 water = !water;

}
if (water)
{
 digitalWrite(12, HIGH); // water relay *changed
 digitalWrite(18, HIGH); // water led *changed
}
else
{
 digitalWrite(12, LOW);  // *changed
```

```
   digitalWrite(18, LOW);  // *changed
  }

  if (advanceH || advanceW)
  {
   digitalWrite (14, HIGH); // advance led *changed
  }

//encode the switch positions ready to broadcast over the network

   advCode = 0;
   if(advanceH){
   advCode = 1;
  }
   if(advanceW){
   advCode = 2;
  }
   if(advanceH && advanceW){
   advCode = 3;
  }


delay(1000); //slow down for testing
}

//end of loop
//and program!
```