```
//    This program is part of the microstill project.
//    For details and instructions visit <http://www.microstill.net>.
//
//    This program is free software: you can redistribute it and/or modify
//    it under the terms of the GNU General Public License as published by
//    the Free Software Foundation, either version 3 of the License, or
//    (at your option) any later version.
//
//    This program is distributed in the hope that it will be useful,
//    but WITHOUT ANY WARRANTY; without even the implied warranty of
//    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//    GNU General Public License for more details.
//
//    You should have received a copy of the GNU General Public License
//    along with this program.  If not, see <http://www.gnu.org/licenses/>.

#include <OneWire.h>
#include <Bounce2.h>

// pump relais
const int pump=9;

// pump status, 0 for off, >0 for on
int pumpstate = 0;

// heater relais
const int heater=10;

// heater status, 0 for off, > 0 for on
int heaterstate = 0;

// green switch
const int Tastgruen=3;

// green LED
const int LEDgruen=2;

// yellow switch
const int Tastgelb=5;

// yellow LED
const int LEDgelb=4;

// red switch
const int Tastrot=7;

// red LED
const int LEDrot=6;
```

```
// baud rate
int datarate = 9600;

// temperature sensors
OneWire myds(8);

// leakSensor
const int leakSensor = 11;
int blink = 0;

// bouncer for each switch
Bounce bouncer_rot = Bounce();
Bounce bouncer_gelb = Bounce();
Bounce bouncer_gruen = Bounce();


// Variables
// C - Cooler, H - Heater
double SetpointH, MaxTempC, CurrentTempH, CurrentTempC;
unsigned long now, pumpstarttime, HeatStartTime, blinkTime;

// temperature sensor cooler
byte dsaddrC[8];

// temperature sensor heater
byte dsaddrH[8];

// current program: 0, stop; 1, clean; 2, run; 3, leak
int program = 0;

// temperature sensor read stage
int readstage = 0;



void setup() {
  // setup pin modes
 pinMode(LEDgruen, OUTPUT);
 pinMode(LEDgelb, OUTPUT);
 pinMode(LEDrot, OUTPUT);
 pinMode(heater, OUTPUT);
 pinMode(pump, OUTPUT);
 pinMode(Tastrot, INPUT_PULLUP);
 pinMode(Tastgelb, INPUT_PULLUP);
 pinMode(Tastgruen, INPUT_PULLUP);
 pinMode(leakSensor, INPUT_PULLUP);

  // setup bouncer
 bouncer_rot.attach(Tastrot);
```

```
 bouncer_gelb.attach(Tastgelb);
 bouncer_gruen.attach(Tastgruen);

  // setup bouncer sensitivity
 bouncer_rot.interval(2);
 bouncer_gelb.interval(2);
 bouncer_gruen.interval(2);

  // start serial
 Serial.begin(datarate);

  // switch off heater and pump
 digitalWrite(heater,LOW);
 digitalWrite(pump,LOW);

  // Set configuration temperature sensors
 myds.reset();
 myds.write(0xCC); // Skip ROM
 myds.write(0x4E); // Write scratch pad
 myds.write(0);    // TL
 myds.write(0);    // TH
 myds.write(0x3F); // Configuration Register to 10 bit resolution

  // address temperature sensor heater
 dsaddrH[0] = 0x28;
 dsaddrH[1] = 0xFF;
 dsaddrH[2] = 0x7C;
 dsaddrH[3] = 0xDE;
 dsaddrH[4] = 0x37;
 dsaddrH[5] = 0x16;
 dsaddrH[6] = 0x4;
 dsaddrH[7] = 0x93;

  // address temperature sensor cooler
 dsaddrC[0] = 0x28;
 dsaddrC[1] = 0xFF;
 dsaddrC[2] = 0x12;
 dsaddrC[3] = 0xA6;
 dsaddrC[4] = 0x37;
 dsaddrC[5] = 0x16;
 dsaddrC[6] = 0x4;
 dsaddrC[7] = 0x94;
}

void loop() {
  // read switches
 bouncer_rot.update();
 bouncer_gelb.update();
 bouncer_gruen.update();
```

```
// switch to respective program
if(bouncer_gruen.read() == LOW) {program = 2;} // run
if(bouncer_gelb.read() == LOW) {program = 1;} // clean
if(bouncer_rot.read() == LOW) {program = 0;} // stop
if(digitalRead(leakSensor) == LOW) {program = 3;} // leak


//*****************************
// program STOP
//*****************************
if (program == 0) {
     // switch on red LED, others off
     digitalWrite(LEDrot, HIGH);
     digitalWrite(LEDgelb, LOW);
     digitalWrite(LEDgruen, LOW);
     // turn off heater
     digitalWrite(heater,LOW);
     heaterstate = 0;
     // switch off pump
     digitalWrite(pump,LOW);
     pumpstate = 0;
} // end program STOP


//*****************************
// program CLEAN
//*****************************
if (program == 1) {
     // update variable now
  now = millis();
  // switch on yellow LED
     digitalWrite(LEDrot, LOW);
     digitalWrite(LEDgelb, HIGH);
     digitalWrite(LEDgruen, LOW);

  // turn off heater
     digitalWrite(heater,LOW);
     heaterstate = 0;

  // initial switch on
  if (pumpstate == 0) {
      // switch on pump
     digitalWrite(pump,HIGH);
      // record time pump was started
     pumpstarttime = millis();
      // record pump was started
     pumpstate ++;
      }
```

```
  // pump is allowed 2 minute work followed by 1 minute break
  // run pump for 1 minute then switch off pump
  if (now > (pumpstarttime + 60000) && pumpstate > 0) {
      // switch off pump
      digitalWrite(pump,LOW);
      }

  // wait another 30 seconds then switch to program STOP
  if (now > (pumpstarttime + 90000) && pumpstate > 0) {
      // switch to program STOP
      pumpstate = 0;
      program = 0;
      }
} // end program CLEAN


//****************************
// program RUN
//****************************
if (program == 2) {
  // switch on green LED
     digitalWrite(LEDrot, LOW);
     digitalWrite(LEDgelb, LOW);
     digitalWrite(LEDgruen, HIGH);
     // define set point for heater
     SetpointH = 100;
     // define max temperature of cooler
     MaxTempC = 80;
}

if (program == 2) {

// start measurement and conversion
if (readstage == 0) {
myds.reset();
myds.write(0xCC); // Skip ROM
myds.write(0x44); // start conversion
readstage++;
// switch off pump
digitalWrite(pump,LOW);
}

// read temperature after conversion is ready, i.e. myds.read() > 0
if (myds.read() && readstage) {
CurrentTempC = dsreadtemp(myds, dsaddrC, 10);
CurrentTempH = dsreadtemp(myds, dsaddrH, 10);
now = millis();
```

```
 // start heat pulse if temperature below target
 if (CurrentTempH < SetpointH + 10 && heaterstate == 0) {
     // switch on heater
     heaterstate++;
     digitalWrite(heater,HIGH);
     HeatStartTime = millis();
   // print current temperatures
     Serial.print("Heater ON at: ");
     Serial.print(HeatStartTime);
     Serial.print(" ms Temp Heater: ");
     Serial.print(CurrentTempH);
     Serial.print(" C Temp Cooler: ");
     Serial.print(CurrentTempC);
     Serial.println(" C");
      }

 // turn off heater after 1 sec
 if (now > (1000 + HeatStartTime) && heaterstate > 0) {
       // switch off heater
       digitalWrite(heater,LOW);
        }

 // wait 1 more second before allowing next heat pulse
 if (now > (2000 + HeatStartTime) && heaterstate > 0) {
       // heater status
       heaterstate = 0;
        }

 // switch on pump for 40 ms then pause for 960 ms
 if (now > (1000 + pumpstarttime) && CurrentTempH > SetpointH)
 {
       // record time of pump run
       pumpstarttime = now;
       // switch on pump for 40 ms
       digitalWrite(pump,HIGH); // is on
       delay(40); // 40 ms pause
           // extent pause to 240 ms if cooler temperature is too high
           if (CurrentTempC > MaxTempC) {delay(200);}
       digitalWrite(pump,LOW); // is off
 }

 // get new temperature reading
 readstage = 0;
 } // end read temperature

} // end programm == 2

//****************************
 // program LEAK
```

```
 //****************************
  if (program == 3) {
       // blink on red LED, others off
       now = millis();
    if (blink = 0) {
        blink++;
        blinkTime = millis();
        digitalWrite(LEDrot, HIGH);
         }
    if (now > (blinkTime + 200) && blink > 0) {
        digitalWrite(LEDrot, LOW);
         }
    if (now > (blinkTime + 400) && blink > 0) {
        blink = 0;
         }
       digitalWrite(LEDgelb, LOW);
       digitalWrite(LEDgruen, LOW);
        // turn off heater
       digitalWrite(heater,LOW);
    heaterstate = 0;
       // switch off pump
       digitalWrite(pump,LOW);
        pumpstate = 0;
  } // end program LEAK

}

float dsreadtemp(OneWire myds, byte addr[8], byte resolution) {
byte present = 0;
int i;
byte data[12];
byte type_s;
float celsius;

// identify temperature sensor
switch (addr[0]) {
case 0x10:
// DS18S20
type_s = 1;
break;
case 0x28:
// DS18B20
type_s = 0;
break;
case 0x22:
// DS1822
type_s = 0;
break;
default:
```

```
Serial.println(F("Device is not a DS18x20 family device."));
}

present = myds.reset();
myds.select(addr);
myds.write(0xBE); // Read Scratchpad
for ( i = 0; i < 9; i++) { // we need 9 bytes
data[i] = myds.read();
}

// convert the data to actual temperature
unsigned int raw = (data[1] << 8) | data[0];
if (type_s) {
raw = raw << 3; // 9 bit resolution default
if (data[7] == 0x10) {
// count remain gives full 12 bit resolution
raw = (raw & 0xFFF0) + 12 - data[6];
} else {
byte cfg = (data[4] & 0x60);
if (cfg == 0x00) raw = raw << 3;  // 9 bit resolution
else if (cfg == 0x20) raw = raw << 2; // 10 bit resolution
else if (cfg == 0x40) raw = raw << 1; // 11 bit resolution
}
}
celsius = (float)raw / 16.0;
return celsius;
}
```