

```
// 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;
// pumplevel 3,2,1,0 corresponding to 1000 ms, 1250 ms, 1700 ms and 2500 ms of
pumppause.
int pumplevel = 3;
// pumppause of 1000, 1250, 1700, 2500 ms corresponding to 1 L/h, 0.8 L/h, 0.6
L/h, 0.4 L/h
unsigned long pumppause = 1000;

// heater relais
const int heater=10;

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

// green switch
const int greenSwitch=3;

// green LED
const int greenLED=2;

// yellow switch
const int yellowSwitch=5;

// yellow LED
const int yellowLED=4;
```

```
// red switch
const int redSwitch=7;

// red LED
const int redLED=6;

// baud rate
int datarate = 9600;

// temperature sensors
OneWire myds(8);

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

// bouncer for each switch
Bounce bouncer_red = Bounce();
Bounce bouncer_yellow = Bounce();
Bounce bouncer_green = Bounce();

// Variables
// C - Cooler, H - Heater
// SetpointH in degree C
double SetpointH = 110, MaxTempC, CurrentTempH, CurrentTempC;
unsigned long now, pumpstarttime, HeatStartTime, blinkTime, holdButton;

// templevel and corresponding SetpointH temperatures 0, 95 C; 1, 100 C; 2, 105
C; 3, 110 C
int templevel = 3;

// temperature sensor cooler
byte dsaddrC[8];

// temperature sensor heater
byte dsaddrH[8];

// current program: 0, stop; 1, clean; 2, run; 3, leak; 4, setup
int program = 0;

// temperature sensors read stage
int readstage = 0;

void setup() {
  // setup pin modes
  pinMode(greenLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
}
```

```

pinMode(redLED, OUTPUT);
pinMode(heater, OUTPUT);
pinMode(pump, OUTPUT);
pinMode(redSwitch, INPUT_PULLUP);
pinMode(yellowSwitch, INPUT_PULLUP);
pinMode(greenSwitch, INPUT_PULLUP);
pinMode(leakSensor, OUTPUT);
digitalWrite(leakSensor, HIGH);

// setup bouncer
bouncer_red.attach(redSwitch);
bouncer_yellow.attach(yellowSwitch);
bouncer_green.attach(greenSwitch);

// setup bouncer sensitivity
bouncer_red.interval(2);
bouncer_yellow.interval(2);
bouncer_green.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

// HERE GOES THE ADDRESS OF
// 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;

// HERE GOES THE ADDRESS OF
// 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_red.update();
  bouncer_yellow.update();
  bouncer_green.update();

  // switch to respective program
  if(bouncer_green.read() == LOW && program != 4) {program = 2;} // run
  if(bouncer_yellow.read() == LOW && program != 4) {program = 1;} // clean
  if(bouncer_red.read() == LOW) {program = 0;} // stop
  pinMode(leakSensor, INPUT);
  if(digitalRead(leakSensor) == LOW) {program = 3;} // leak
  pinMode(leakSensor, OUTPUT);
  digitalWrite(leakSensor, HIGH);
  if(bouncer_red.read() == LOW && bouncer_red.fell()) {holdButton = millis();}
  if(bouncer_red.read() == HIGH && bouncer_red.rose()) {holdButton = 0;}
  if(bouncer_red.read() == LOW && millis() > holdButton + 1000) {program = 4;
blinks=0;} // setup

  /*******
  // program STOP
  /*******
  if (program == 0) {
    // switch on red LED, others off
    digitalWrite(redLED, HIGH);
    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, 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(redLED, LOW);
digitalWrite(yellowLED, HIGH);
digitalWrite(greenLED, 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(redLED, LOW);
    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, HIGH);
    // define set point for heater
    SetpointH = 100;
    // define max temperature of cooler
    MaxTempC = 78;
}

```

```

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 SetpointH
if (CurrentTempH < SetpointH + 5 && 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 100 ms
if (now > (100 + HeatStartTime) && heaterstate > 0) {
// switch off heater
digitalWrite(heater,LOW);
}

// allow next heat pulse
// if CurrentTempH is below SetpointH the heater will heat in cycles 100ms on
/100 ms off (ca 100 Watts)
// if CurrentTempH is close to SetpointH the heater will heat with full power
of 250 Watts.
if (now > (100 + HeatStartTime) && heaterstate > 0) {
// heater status
if (CurrentTempH < SetpointH) {delay(100);}
}
}
}

```

```

heaterstate = 0;
}

// switch on pump for 40 ms then wait until pumppause is over
// pump is only turned on when SetpointH temperature is reached
if (now > (pumppause + 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 if cooler temperature is too high
    if (CurrentTempC > MaxTempC) {delay(40);}
    if (CurrentTempC > MaxTempC + 3) {delay(40);}
    if (CurrentTempC > MaxTempC + 6) {delay(40);}
    if (CurrentTempC > MaxTempC + 9) {delay(40);}
    if (CurrentTempC > MaxTempC + 12) {delay(40);}
    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 (blinks == 0) {
        blinks++;
        blinkTime = millis();
        digitalWrite(redLED, HIGH);
    }
    if (now > (blinkTime + 200) && blinks > 0) {
        digitalWrite(redLED, LOW);
    }
    if (now > (blinkTime + 400) && blinks > 0) {
        blinks = 0;
    }
    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, LOW);
    // turn off heater
    digitalWrite(heater,LOW);
    heaterstate = 0;
}

```

```
// switch off pump
digitalWrite(pump,LOW);
pumpstate = 0;
} // end program LEAK
```

```
/**/
```

```
// program SETUP
```

```
/**/
```

```
if (program == 4) {
```

```
    // set pumplevel
```

```
    if(bouncer_yellow.read() == LOW && bouncer_yellow.fell() && program == 4) {
```

```
        pumplevel++;
```

```
        if (pumplevel > 3) {pumplevel = 0;}
```

```
        Serial.print(" pumplevel at: ");
```

```
        Serial.println(pumplevel);
```

```
        switch (pumplevel) {
```

```
            case 0:
```

```
                pumppause = 2500;
```

```
                break;
```

```
            case 1:
```

```
                pumppause = 1700;
```

```
                break;
```

```
            case 2:
```

```
                pumppause = 1250;
```

```
                break;
```

```
            case 3:
```

```
                pumppause = 1000;
```

```
                break;
```

```
        }
```

```
        Serial.print(" pumppause at: ");
```

```
        Serial.println(pumppause);
```

```
    }
```

```
    // set templevel
```

```
    if(bouncer_green.read() == LOW && bouncer_green.fell() && program == 4)
```

```
{templevel++;
```

```
    if (templevel > 3) {templevel = 0;}
```

```
    Serial.print(" Templevel at: ");
```

```
    Serial.println(templevel);
```

```
    switch (templevel) {
```

```
        case 0:
```

```
            SetpointH = 95;
```

```
            break;
```

```
        case 1:
```

```
            SetpointH = 100;
```

```
            break;
```

```
        case 2:
```



```

        SetpointH = 105;
        break;
    case 3:
        SetpointH = 110;
        break;
    }
    Serial.print(" SetpointH at: ");
    Serial.println(SetpointH);
}

// blink on yellow, green
now = millis();
if (blinks == 0) {
    blinks++;
    blinkTime = millis();
    // blink zero
    digitalWrite(yellowLED, HIGH);
    digitalWrite(greenLED, HIGH);
    digitalWrite(redLED, HIGH);
}
if (now > (blinkTime + 200) && blinks > 0) {
    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, LOW);
    digitalWrite(redLED, LOW);
}
if (now > (blinkTime + 400) && blinks > 0) {
    // blink one
    if (pumplevel >= 1 ) {digitalWrite(yellowLED, HIGH);}
    if (templevel >= 1) {digitalWrite(greenLED, HIGH);}
}
if (now > (blinkTime + 600) && blinks > 0) {
    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, LOW);
}
if (now > (blinkTime + 800) && blinks > 0) {
    // blink two
    if (pumplevel >= 2 ) {digitalWrite(yellowLED, HIGH);}
    if (templevel >= 2) {digitalWrite(greenLED, HIGH);}
}
if (now > (blinkTime + 1000) && blinks > 0) {
    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, LOW);
}
if (now > (blinkTime + 1200) && blinks > 0) {
    // blink three
    if (pumplevel >= 3 ) {digitalWrite(yellowLED, HIGH);}
    if (templevel >= 3) {digitalWrite(greenLED, HIGH);}
}
if (now > (blinkTime + 1400) && blinks > 0) {

```

```

    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, LOW);
}
if (now > (blinkTime + 2000) && blinks > 0) {
    blinks = 0;
}

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

} // end program SETUP
}

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;
}
```