**Středoškolská technika 2019**

**Setkání a prezentace prací středoškolských studentů na ČVUT**

# Chladící tričko s automatickou regulací

**Anastázie Kubásková, Matyáš Levíček**

Gymnázium Joachima Barranda
Talichova 842, Beroun

## Basic concept

So why do we do this in the first place? Essentially, every summer I have a huge problem with the amount of heat that surrounds us. And so at first as a joke I said that I'm making a cooling T-shirt, like astronauts have inside they're spacesuit. A few months later, we had to come up with a project for this conference and the only thing that came to mind was this. So here we go!

The T-Shirt basically works by cooling/heating a water loop, that than passes thru the shirt. We will be using two pre-assemble peltiere units as small thermal pumps to regulate the water temperature and all of that will be controlled by an Arduino style microcontroller. And all of this is gonna be powered by a big 12V battery or an outlet adaptor.

During the writing of this document, we still don't have all the components needed on the table, so we can't jet measure the efficiency, or even say if it was worth it. But until now we had a lot of fun and this definitely moved our practical engineering skills to the next level.

## Hardware and circuitry

The whole T-shirt requires only a few parts listed below, that cost from $50 to $120, depending on where you live.

Parts list:

1. Water loop
    a. T-shirt
    b. 4mm ID, 6mm OD tube
    c. Small 12V aquarium pump
    d. Plastic bag as water reservoir
    e. Water block
    f. (Optional) Quick disconnects
2. Heating / Cooling circuitry
    a. 2x 12V 60W Peltier elements
    b. 2x 5V 30mm PC fan
    c. Screws and nuts
    d. Power supply (AC-DC converter or 12V battery)
    e. 2x relay
3. Control circuitry
    a. Teensy 3.2
    b. 0.96” OLED 128x64px display
    c. Rotary encoder
    d. Thermistor
    e. 10K resistor
    f. 3x 8x8 LED matrixes
    g. 5V power supply
    h. Protoboard
    i. Solder
    j. Sockets for everything
4. A bit of patience

On Picture 1. is the complete circuit diagram for the electrical side of things, and on Picture 2. we have the water loop diagram.

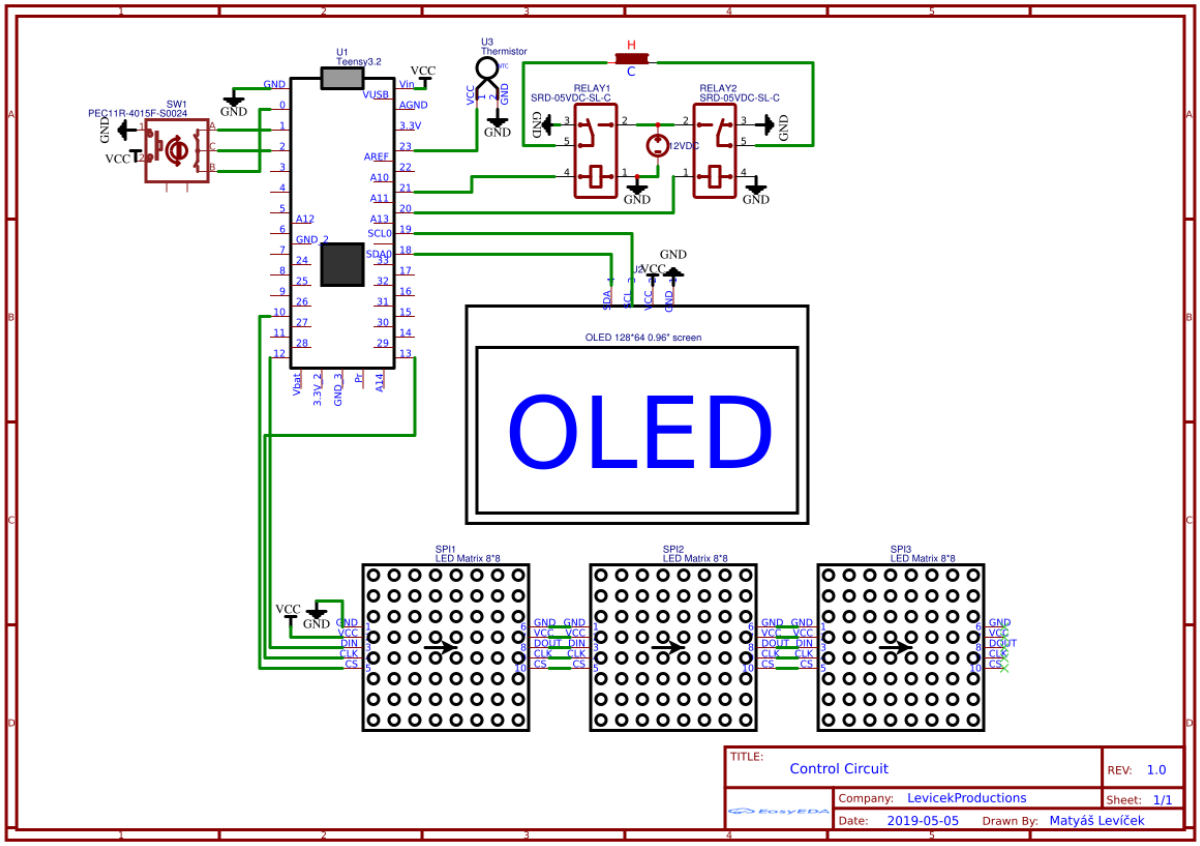
The “brain” of the whole project is the Teensy 3.2 board with an AT Mega processor. The main reason we have a microcontroller is to be able to send – and log – all measured data in my PC. The whole program can be found in the next section (Software), but here is a simplified description: We measure the current water temperature, dependant on the result we heat or cool it, and send the measured data to a computer to be made into a graph.

For user input, we use both serial communication from the PC and a rotary encoder with quadrature encoded signals. I won’t be explaining how any of that works, you can find links to the principals at the end of this article.

The encoder itself is configured to use all of the 4 pulses per tic, so you can ramp the temperature up or down pretty fast, but still with a decent amount of control.

For output, we opted for two methods, one, we used three 8 by 8 led matrixes for a very eye catching display, that can be easily read ever from a distance, this was mainly done to look cool on the show floor, and two, we used a “low resolution” OLED screen, witch displays all of the necessary information, but also a graph to show fluctuations in temperature control, so

we can tune our program, to react faster or slower, so it won't resonate increasing the fluctuations.




Picture 1.

## Software

*All resources used are linked at the end of the script.*

In this section I'm going to explain the very basic concepts of our script and then there is the full, color coded and formatted code, straight form my IDE.

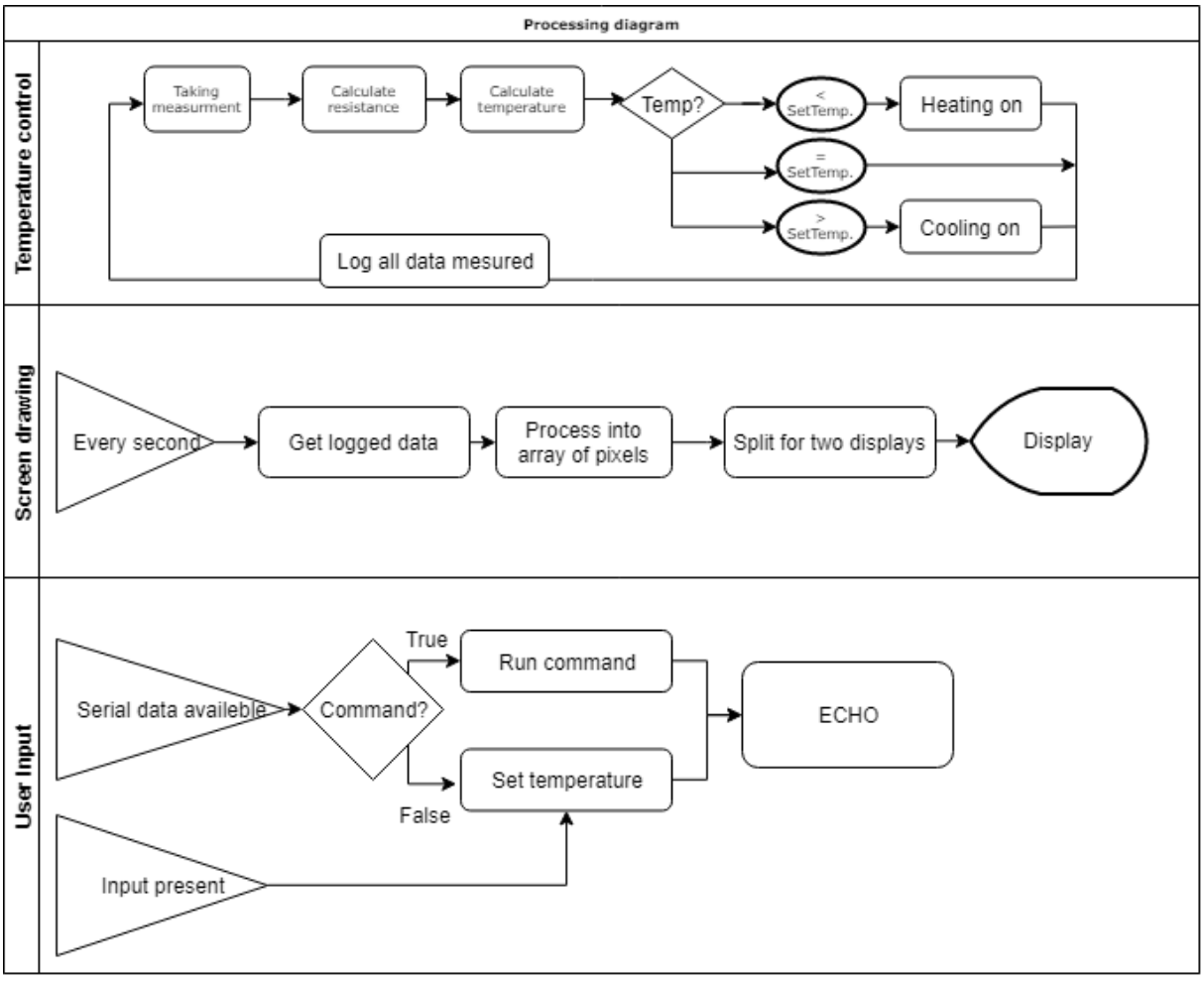
Basically, first we need to measure the temperature of the water. We can do that by using a thermistor – a resistor, that changes its resistance dependant on its temperature. But the struggle is, that our board doesn't directly measure resistance. So, we make a voltage divider and measure at the midpoint and the value we get back is a 10bit number which describes the measured voltage on scale from ground to reference (5V).

To calculate the resistance, we use the formula below – which is outcome of Ohm's law:

$$R_t = 1023 * 10\,000 / V_i - 10000$$

And then to get the temperature, we use this (witch I don't fully understand…):

$$T(R) = \frac{1}{\frac{\ln\left(\frac{R}{R_{25}}\right)}{\beta} + \frac{1}{T_{25}}} \quad [\mathrm{K}]$$




Picture 3. – Script diagram

The program was written for an Arduino compatible development board (Teensy 3.2), and so it is written in the C language, witch I used for my first time – so it can be quite inefficient,

and maybe not using the best practises, but hey! It works! And so, I'm respecting the rule not to fix something that isn't broken!

Unfortunately, the page size and margins of this document break the text alignment of the code, but at least it is colorful 😊, Enjoy reading!

```
#include <TimerOne.h>           // This library makes it much easyer to deal with timers... It could be done without it, but this is better!

#include <Encoder.h>            // Easier rotary encoder control
#include <LedControl.h>         // LED displays controler file has to be imported for .setLed() to work

#include <SPI.h>                // All below is for the LCD
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// ----------- "Settings" (constants mostly...) --------------

// +-+-+-+-+-+ Pin numbers +-+-+-+-+-+-
const int relAPin = 21;         // Pin of relay 1
const int relBPin = 20;         // Pin of relay 2
const int thermPin = 23;        // The thermistor probably has a resistance of 70K ohm at 25C and Beta 2819K - 46500Ohm při 36C, 121000 při 9.5C

const int DIN_PIN = 12;         // Data pin for the displays
const int CS_PIN = 10;          // Chip select pin (whitch display an I controling now?)
const int CLK_PIN = 13;         // Master clock - to synchronise data streams

const int rotaryButtonPin = 2;  // Thebutton of the rotary encoder
const int rotaryAPin = 1;       // The A output of the rotary encoder (could be swapped with B... that would swap the incremental direction)
const int rotaryBPin = 0;       // The A output of the rotary encoder (could be swapped with A... that would swap the incremental direction)

// +-+-+-+-+-+-+ System settings +-+-+-+-+
const int displayCount = 3;     // number of connected displays - see below "LEDControl"
const int displayBrightness = 1;// brightness of the leds on scale 0-15

const int SCREEN_WIDTH = 128;   // OLED display width, in pixels
const int SCREEN_HEIGHT = 64;   // OLED display height, in pixels
const int OLED_RESET = 4;       // Reset pin # (or -1 if sharing Arduino reset pin)

// +-+-+-+-+- Real User Settings +-+-+-+-+-
String    debug = "temp";       // Serial prints (no, debug, temp, full)

const int samples = 15;         // Number of samples to take while measuring temperature (more = less noise, takes longer)
    float setTemp = 27;         // In degrees celsius, User set trigeret temp - can be nagativ

const int R25 = 62800;          // Resistance of my thermistor at 25C50
const int betaTemp = 3174;      //In kelvin, the beta coeficient - explaination: http://www.giangrandi.ch/electronics/ntc/ntc.shtml

const float splashScreen = 5;   // length of the splashscreen timer in SECONDS
const float graphSpeed = 0.5;   // In seconds, how often do I smaple the temperature for the graph?

```

```
const uint64_t IMAGES[] = {     // paterns for different numbers - if written as 8*8-bit array, would resamble the arabic numbers (the numbers humans use)
  0x7088888888888870, // 0
  0x7020202020203020, // 1
  0xf810204080808870, // 2
  0x7088808060808870, // 3
  0x8080f88890a0c080, // 4
  0x70888080780808f8, // 5
  0x7088888878088870, // 6
  0x101010204080f8, // 7
  0x7088888870888870, // 8
  0x708880f088888870, // 9
  0xe0a0202020aeea0e, // C
  0x20202060202eea0e  // F
};

const int IMAGES_LEN = sizeof(IMAGES)/8;  // Just so we don't need to do it later...

const String splash = "11111111111
```

```
77.  String tempDisply = "Current";              // The number displaid ("Current" or "Set")
78.
79.  float lastTemperatures[15] = {};           // Here we store temperatures for averiging
       -
       the 5V line on arduino produces a LOT of noise, so at least 10 samples to average
       are needed
80.
81.  int   graphTemperatures[127] = {};         // Here we can store the temperatures neede
     d for drawing the graph
82.  long  graphPhase = 0;                      // Arduino doesn't allow for dynemic array,
       so I deal with it in this rather stupid way....
83.
84.  String relState = "A";                     // The current state relays are in (A,B,Sta
     gnant)
85.
86.  void setup() {                            //------------ Once upon a start-up... ----
     -----------------
87.                                            // +-+-+-+-+-+ Setup pins +-+-+-+-+-+-+-+-+
88.      pinMode(relAPin, OUTPUT);             // The pin to whitch the relay 1 is connect
       ed is set to output
89.      pinMode(relBPin, OUTPUT);             // The pin to whitch the relay 2 is connect
       ed is also set to output
90.
91.      pinMode(rotaryButtonPin, INPUT);      // The rotary encoders button has to be an
       input
92.      attachInterrupt(digitalPinToInterrupt(rotaryButtonPin), changeTempScale, RISING); /
       / And when it is pressed, it interrupts all actions and changes the temperature sc
       ale
93.                                            // +-+-+-+-+- Setup the graph readout +-+
94.      Timer1.initialize(graphSpeed * 1000000);// Once every set amount of time (delivered
         in seconds has to be made into milis),
95.      Timer1.attachInterrupt(updateGraph);  // run the updateGraph, whitch will automat
       icy add a new step into the temperatures array
96.                                            // +-+-+-+-+- Setup the LED matrixex +-+-
97.      for(int i = 0; i < displayCount; i++){  // Do this for every connected display
98.        display.clearDisplay(i);            // clear the proccesed display
99.        display.shutdown(i, false);         // wake it up
100.       display.setIntensity(i, displayBrightness); // set its brightness (0-15)
101.     }
102.                                           // +-+-+-+-+- Setup the OLED screen +-+-+
103.     if(!displai.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
104.       if(debug != "no" && debug != "temp") Serial.println("SSD1306 allocation failed");
105.       for(;;);                            // Don't proceed, loop forever
106.     }
107.
108.     displai.clearDisplay();               // Clear the display buffer, because on boo
       ting, the adafruit splashscreen is written to it...
109.
110.     for(int x = 0; x < 43; x++){          // For every horizontal pixel of my logo
111.       for(int y = 0; y < 43; y++){        // For every pixel below it
112.          displai.drawPixel(y + (128-42)/2, x + (64-
       42)/2, splash.substring(x*43+y, x*43+y+1) == 1 ? WHITE : BLACK ); // Find if the p
       ixel is lit-up and draw it
113.       }
114.     }
115.
116.     displai.display();                    // Display the logo
117.     delay(splashScreen * 1000);           // Pause for (insert here) seconds, so some
       one can see the splash screen
118.
119.     displai.clearDisplay();               // Clear the buffer
120.     displai.display();                    // Display the empty buffer -
         black screen
121.
```

```
122.      if(debug != "no") Serial.begin(9600);   // Begin serial comunication over usb, so w
           e can list temperatures, and roger set temperatures (if no debug or log is require
           d, we can skip this and save some processing time)
123.  }
124.
125.  void loop() {                                          //------------
            Every time she closed her eyes... -----------
126.    for(int i = 0; i < samples; i++){                    // sample user requested sample c
           ount
127.      measuredVoltage = analogRead(thermPin);            // sample voltage
128.      resistance = 10230000 / measuredVoltage -
            10000;  // calculate resistance of the thermistor based on the other resistor in
           the voltage devider
129.      temperature = ( 1 / ( (log(resistance / R25) / betaTemp) + (1 / 298.15) ) ) -
            273.15; // calculate temperature -
           this could have been done in one step with the resistance, but this is more pleas
           ant!
130.      lastTemperatures[i] = temperature;                 // store the sample for the next
           round
131.    }
132.
133.    averageTemperature = 0;                              // reset the "average varieble"
134.    for(int i = 0; i < samples; i++){                    // do this to every sample
135.      averageTemperature += lastTemperatures[i];         // add up all samples
136.
137.      if(debug != "no" && debug != "debug") Serial.print(lastTemperatures[i]);  // Print
           each sample
138.      if(debug != "no" && debug != "debug") Serial.print("C, ");                // And it
           s scale
139.    }
140.
141.    averageTemperature = averageTemperature / samples;  // do the actual average
142.
143.    if(debug == "full") Serial.print("average: ");      // print the average
144.    if(debug != "no" && debug != "debug")Serial.println(averageTemperature);
145.
146.    if(averageTemperature < (setTemp - 2)){              // +-+-+-+-+-
            Cooling or heating? +-+-+-+-+-+-
147.      relState = "A";
148.      digitalWrite(relAPin, HIGH);                       // Set the positions of the "swit
           ches" to heating configuration
149.      digitalWrite(relBPin, LOW);                        // -------------------- || ------
           ------------------------------
150.    } else if(averageTemperature > (setTemp + 2)){       // Or the cooling one...
151.      relState = "B";
152.      digitalWrite(relAPin, LOW);
153.      digitalWrite(relBPin, HIGH);
154.    } else if((setTemp -
            0.5) < averageTemperature && averageTemperature < (setTemp + 0.5)){ // To stop th
           e relays from flickering, and thus destroing the mechanical components
155.      relState = "Stagnant";                             // We have to leave out a "dead z
           one" where no heating or cooling is going on
156.      digitalWrite(relAPin, HIGH);
157.      digitalWrite(relBPin, HIGH);
158.    }
159.
160.    // ----------------- User Input handling ----------------
161.    if(Serial.available()){                              // If we have rogered anything fr
           om the serial port
162.      String input = Serial.readStringUntil('\n');       // We read it into a string
163.      if(debug == "full") Serial.println(input);         // Echo it
164.      if(input.toInt() != 0) setTemp = input.toInt();    // If it is converteble to an int
           eger, do it!
165.
166.      if(input == "scaleSwap") changeTempScale();        // If it is a command for swappin
           g scales, do it!
```

```
167.      if(input == "no" || input == "debug" || input == "full" || input == "temp") debug =
            input; // If it is a command for changing debug options, change them!
168.
169.      encoder.write(setTemp);                                  // Write in the set temperature
170.
171.      if(debug == "debug") Serial.print("Temperature set: " );  // Echo it...
172.      if(debug == "debug") Serial.println(setTemp);
173.
174.      input = "";                                              // Prepare for another command :-
            )
175.    }
176.
177.    setTemp = encoder.read();                                  // Since the encoder value is res
          et when you type in the temp thru serial, we can just set the setTemp to whatever
          the encodr is at.
178.
179.    // ---------------- Screendrawing ------------------------
180.    flooredAvgTemp = floor(averageTemperature);                // Floor the temperature, IDK if
          it is really needed thow...
181.    drawMeasured(tempScale, setTemp);                          // Draw it onto the matrixes
182.
183.    displai.clearDisplay();                                    // Get a rid of everything in the
           screen buffer (make the OLED screen black)
184.                                                               // -------- Text settings -------
          ------------
185.    displai.setTextSize(1);                                    // Normal 1:1-pixel scale
186.    displai.setTextColor(WHITE);                               // Draw white text
187.    displai.setCursor(0, 0);                                   // Start at top-left corner
188.    displai.cp437(true);                                       // Use full 256 char 'Code Page 4
          37' font
189.                                                               // -------- Text drawing --------
          ------------
190.    displai.println((tempScale == "C" ? (String)flooredAvgTemp : (String)(flooredAvgTemp
          * 2 + 32))); // Print my temperature, if the scalse is set to fahrenheit, first ca
          lculate the IMPERIAL temperature :{
191.    displai.println((tempScale == "C" ? (String)setTemp : (String)(setTemp * 2 + 32)));
                      // ---------------------------------------- || -------------------
          ----------------------------------
192.    displai.setCursor(15, 0);                                    // Reset the cursor pos, so it
          looks more organized as a table on the screen
193.    displai.println(" " + tempScale + "    Current Temp");// Draw the text
          ||
194.    displai.setCursor(15, 8);                                    // ----------------------------
           || ----------------------------------------
195.    displai.println(" " + tempScale + "    Set Temp.");    // ----------------------------
           || ----------------------------------------
196.    displai.println((String)((int)resistance/1000));     // To draw the resistance, we fir
          st need to make it into kOhms, so it fits...
197.    displai.setCursor(15, 16);                                 // Same shit again
198.    displai.println(" kOhm Termistor");                        // And again
199.                                                               // --------- Graph drawing ------
          -------------
200.    for(int x = 0; x < 127; x++){                              // For every horizontal pixel on
          the screen
201.      displai.drawPixel(x, 64 -
            graphTemperatures[x], WHITE); // Get the walue and draw it
202.    }
203.
204.    displai.drawPixel(graphPhase%127, 25, WHITE);              // To make it more readable, draw
           a phase sync dot on top
205.
206.    displai.display();                                         // Now display the buffer, (becau
          se, we were just filling it previusly...)
207.  }
208.
```

```
209. void updateGraph(){                                    // The routine for graph value ad
         ding (here, just so I can call it with an interrupt...)
210.   graphTemperatures[graphPhase%127] = averageTemperature; // Add a sample to my collect
         ion (or overwrite existing)
211.   graphPhase++;                                       // Increment the phase, because t
         he stupid C prog.lang. has no dynemic arrays! Fuck... My board has enougth memory
         to deal with a dynemic list!
212. }
213.
214. void displayImg(int dis, int num) {                   //---------
          Display drawing routine ------------
          (whitch display am I currently drawing? which pattern do I need to draw?)
215.   uint64_t image = IMAGES[num];                       // Get the requested patern from
         IMAGES (declared at top)
216.
217.   for (int y = 0; y < 8; y++) {                       // For each row of pixeles
218.     byte row = (image >> y * 8) & 0xFF;               // Mask out the row from the patt
         ern
219.     for (int x = 0; x < 8; x++) {                     // For every pixel in that row
220.       display.setLed(2-dis, 7-y, 7-
         x, bitRead(row, x)); // Draw it! (if the number is negative, reverse colors)
221.     }
222.   }
223. }
224.
225. void drawMinus(int dis){
226.   for (int y = 0; y < 3; y++) {                       // For row of pixeles
227.     display.setLed(dis, 3, 7-y, true);
228.   }
229. }
230.
231. void drawMeasured(String scale, int temp){
232.   if(scale == "F") temp = temp * 1.8 + 32;
233.   displayImg(0, (abs(temp) -
          abs(temp) % 10)/10);     // the first (counting from zero) display should say the
          reminder from deviding set temperature by 100 minus the units(reminder from divis
         ion by 10). (Tens)
234.   if(temp < 0) drawMinus(2);                          // if set temperature is negative
         , draw a small minus sign on top of the first display
235.   displayImg(1, abs(temp) % 10);                      // the second (counting from zero
         ) display should say the reminder from deviding set temperature by 10. (Units)
236.
237.   if(scale == "C") displayImg(2, 10);                 //  the third (counting from zero
         ) display should forn now just say the celsius sigh
238.   if(scale == "F") displayImg(2, 11);                 //  the third (counting from zero
         ) display should forn now just say the fahrenheit sigh
239. }
240.
241. void changeTempScale(){
242.   if(tempScale == "C") tempScale = "F"; else tempScale = "C";
243. }
244.
245. /* Resources used and learning pages:
246. https://navody.arduino-shop.cz/navody-k-produktum/arduino-led-matice.html
247. https://xantorohara.github.io/led-matrix-editor/
248. https://stackoverflow.com/questions/31551888/casting-int-to-bool-in-c-c
249. https://www.norwegiancreations.com/2017/12/arduino-tutorial-serial-inputs/
250. https://playground.arduino.cc/Main/LedControl/#SetupClear
251. https://www.arduino.cc/reference/en/language/variables/data-
         types/string/functions/toint/
252. https://github.com/arduino/Arduino/issues/6714
253. https://www.itnetwork.cz/hardware-pc/arduino/arduino-a-i2c-sbernice
254. https://startingelectronics.org/tutorials/arduino/modules/OLED-128x64-I2C-display/
255. https://www.pjrc.com/teensy/td_libs_Wire.html
256. http://www.giangrandi.ch/electronics/ntc/ntc.shtml
```

```
https://www.reddit.com/r/AskElectronics/comments/3gn3bq/is_it_acceptable_to_wire_digital_grounds_to_agnd/
https://www.pjrc.com/teensy/td_libs_SPI.html#altpins
https://www.pcbcart.com/article/content/clean-pcbs-after-surface-mount-soldering.html

www.giangrandi.ch/electronics/ntc/ntc.shtml
https://cs.wikipedia.org/wiki/Omega
https://www.dcode.fr/binary-image
https://www.gymberoun.cz/galerie

https://www.google.com/search?q=row+of+binary+numbers+into+byte+array&oq=row+of+binary+numbers+into+byte+array&aqs=chrome..69i57.15564j0j7&sourceid=chrome&ie=UTF-8
https://www.arduino.cc/reference/en/language/variables/data-types/string/functions/substring/
https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/
https://playground.arduino.cc/Code/Timer/
https://www.youtube.com/watch?v=U6FLgeLV3j8
https://www.arduino.cc/en/Tutorial/Graph
*/
```

I believe, that the code is already well commented so I won't further explain what it does.

If you need better explanation, send me a mail: matyas@levicek.net

## Conclusion

As said before, we haven’t jet finished the project so we have no way to conclude any observations, but everything so far went as planned and we have learned a lot of new things!

So only if you were on the conference you know how it turned out, the rest of you can write me a mail, or come meet us next year at Stretech 2020!