Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Ubiquitous Computing Lab 1

written by

Isabella Schön

298052

29. April 2021

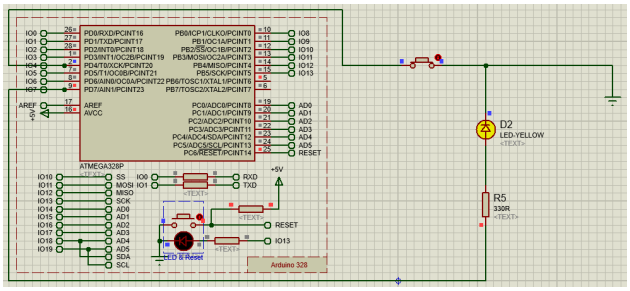
Kapitel 1

From Blinking To Communicate

In the first task the student had to achieve knowledgment about electronic devices and their components such as a LED light, a buzzer and a button.
To be able to control thesedevices in their interaction, a microcontroller is added which is known under the name"Arduino 328".
For these exercises two tools were used: first of all the "Proteus 8" whichis a electronic simulation program, and second the Arduino IDE which is an open-sourceelectronic prototyping plattform for writing the code for each exercise which will be added tothe microcontoller.
To be able to work and use each component, Arduino has the opportunityto implement the standard library for each of these, also when writting a code which involvesa special electronic device.

1.1 LED light with Button

In the first task of the first part, an LED light with a button had to be installed. This was based on the instructions given in the lecture.
First a new project was created with the integration of the "Arduino 328" microcontroller. Then a button was installed, which was connected to the Arduino through PIN 4 (IO4) as well as to a power component. Then the LED light was installed, which was connected on one side with a "Minres330R", a resistor with 5V, to the Arduino via PIN 7 (IO7). The other side of the LED was connected to the connection of the button. Because when the button is pressed, electricity runs through the line, which also supplies the LED light. The following scenario now had to be written in a program which can be seen in the following.
Then the code was integrated and executed by pressing the play button.
What happened? What can you observe? As soon as the program becomes executable, electricity flows through the cables. This could be recognized by the blinking lights on the PINs. If the button is now pressed, the LED light lights up yellow.



(a) LED Light

```
int brightness = 0;
int fading = 5;

void setup() {
  pinMode(4, INPUT_PULLUP); //Ein- und Ausschalter
  pinMode(7, OUTPUT); //Stromzufuhr - blinking
}

void loop() {
  // for LED - blinking
  if(digitalRead(4) == LOW) {
    digitalWrite(7, HIGH);
  } else {
    digitalWrite(7, LOW);
  }
}
```

(b) Code for LED Light

1.2 LED Light with timer

In this task, the previous one was expanded to include a timer. Instead of a constant light when the button is pressed, a timer is now to be integrated, which switches the LED light on and off again after a certain time.
How does your LED Blink code works?
The code was kept simple by adding code to the loop () method. In this a query occurs: is the button below or above? When the button is pressed, the light should go on. If it is not pressed, the light should stay off.
If the button is now pressed and the light goes on, a delay of 10ms should happen immediately afterwards, so that the light goes out again and the light goes back on immediately afterwards. This happens again and again when the button is pressed.

```
void loop() {
  // for LED - blinking
  if(digitalRead(4) == LOW) {
    digitalWrite(7, HIGH);
    delay(10); // timer
    digitalWrite(7, LOW); // timer
    delay(10); //timer
  } else {
    digitalWrite(7, LOW);
  }
}
```

Abbildung 1.2: Code for blinking LED Light - timer

1.3 LED Light with fading Light

After the timer, the task should also be expanded to include a fading light. Instead of a light that is directly on and off again, the light should slowly become brighter when the button is pressed and darker more slowly when it is released.
How does your LED Blink code works?

The loop () method has also been revised in this code for both states of the button. If the button is now pressed, a query should be made about the brightness. The brightness levels here were selected in the interval [0: 255]. With each query, the lamp becomes one level of five brighter and the brightness is sent to the "analogWrite(pinNumber, PWM)" which is resonpsible for sending the brightness to the LED. When the full brightness level has been reached, the light is completely on .
The opposite happens when the button is no longer pressed. The brightness becomes darker by a level of five with each query until the light is completely dark or off.

```
int brightness = 0;
int fading = 5;

void loop() {
  //for LED - fading
  if(digitalRead(4) == LOW) {
    if(brightness < 255) {
      brightness = brightness + fading;
    }
  } else {
    if (brightness > 0) {
      brightness = brightness - fading;
    }
  }
  analogWrite(3, brightness);
  delay(30);
}
```

Abbildung 1.3: Code for fading LED Light

What is the difference between making a LED blink or fade?
The difference in the codes is the query and integration of the various options. With the LED blinking a delay was integrated, with the LED fading a brightness level with respective query.
What are the technical limitations of each concept?
The limitation of LED blinking lies in the concept of the LED light. There are only two levels: on or off.
With LED fading, on the other hand, there are 256 possibilities of brightness levels for the LED light. The connection to the Arduino 328 is also important here. Instead of using PIN 7, the LED light or the adapter must be connected to PIN 3 which is the "PWM" = analog output to fade an LED.

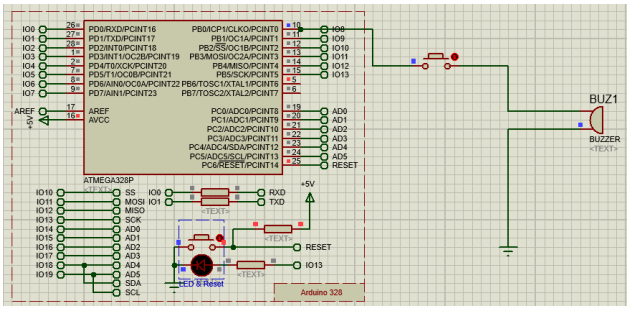
1.4 Buzzer with Button

How does a buzzer work?
In this part, a buzzer was dealt with. The buzzer is an electronic device that has two connection points and makes a sound when it is supplied with power. First, a standard library

is integrated which contains the important functions for the buzzer, such as reproducing the sound from the buzzer. Then an array is created which contains the different tones with their music. Each note of this can be queried in the next array by reproducing it with a number that represents the note. Any number of notes can now be selected, which are queried in the setup () method by querying a for loop. A timer is also built in so that you can hear every note in full. As soon as the button is pressed the whole time the buzzer makes the sound.
What is the difference to turning a LED on and off?
The difference to the LED light lies in the connection with the Arduino and the resulting power supply. The LED light has to be connected to a certain PIN, depending on what is to happen - this is also the case with the buzzer, because each PIN has a different way of handling the component, but the buzzer can be connected to the Arduino without problems. In addition, the buzzer can be controlled more effectively through the different frequencies. Because the buzzer reacts differently to the different tones with the respective pitch and these can be repeated several times.

```
#include "pitches.h"
#include "arduino-timer.h"

//timer created
auto timer = timer_create_default();


// notes in the melody
int melody[] = { NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4 };


// note durations
int noteDurations[] = { 2, 2, 2, 2, 4, 4, 4, 4 };
```

Abbildung 1.4: Code for the buzzer's melody



(a) Buzzer with the Button

```
void repeat_sound(void *) {
  for (int thisNote = 0; thisNote < 8; thisNote++) {

      int noteDuration = 1000 / noteDurations[thisNote];
      tone(8, melody[thisNote], noteDuration); //with Button

      int pauseBetweenNotes = noteDuration;
      delay(pauseBetweenNotes);
      noTone(8); //with Button
  }
  return true;
}
```

(b) Code for Buzzer with Button

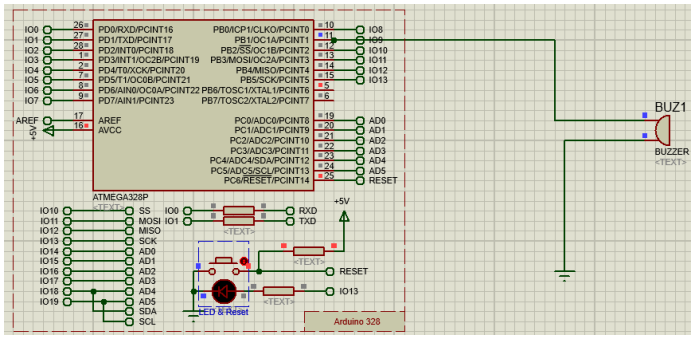
1.5 Buzzer with Timer

How does a buzzer with timer work?
The code from the previous exercise can be used except for a few additional lines. In order

to link the buzzer to a timer, the standard library for the timer must first be integrated. The timer is created with the "timercreatedefault ()" function. In the method "setup ()" this is called with "every (delay, function)". The output must also be taken into account, i.e. from which PIN of the Arduino the power is supplied. And in loop () the timer is activated with "every()".

```
#include "pitches.h"
#include "arduino-timer.h"

//timer created
auto timer = timer_create_default();


// notes in the melody
int melody[] = { NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4 };


// note durations
int noteDurations[] = { 2, 2, 2, 2, 4, 4, 4, 4 };
```

Abbildung 1.6: Code for the buzzer's melody



(a) Normal Buzzer with Timer integrated

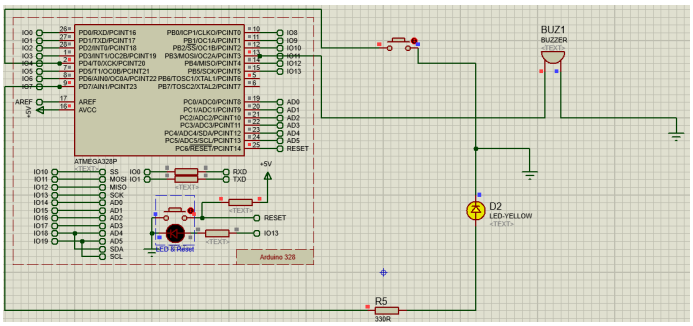
```
void repeat_sound(void *) {
  for (int thisNote = 0; thisNote < 8; thisNote++) {

      int noteDuration = 1000 / noteDurations[thisNote];
      tone(9, melody[thisNote], noteDuration); //with Timer

      int pauseBetweenNotes = noteDuration;
      delay(pauseBetweenNotes);
      noTone(9); //with Timer
  }
  return true;
}



void setup() {
  pinMode(9, OUTPUT);
  timer.every(3000, repeat_sound); //repeat timer
}
```

(b) Code for Buzzer with Timer

1.6 LED Light with Buzzer and Button

The last task of this part is to use all learned components and possibilities and to combine them with each other. For this purpose, an LED light with a button and a buzzer was chosen. If the power supply is done by pressing the button, and the LED light lights up, then the buzzer should make a sound.
Which use case would you plan for a system whit these components?
This type of connection already exists in our everyday life: the traffic light switching. This was specially set up for people with blindness so that they know when the pedestrian lights are green and they can cross the street.

One way to integrate this would be in the factory. For example, if a box is filled with bottles, the amount should be checked. If the box is full, the light does not light up. However, if a bottle is missing, the lamp lights up and the buzzer sounds.



(a) LED Matrixes with button and buzzer

```
void making_sound() {
  for (int thisNote = 0; thisNote < 8; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(11, melody[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration;
    delay(pauseBetweenNotes);
    noTone(11);
  }
}


void loop() {
  if(digitalRead(4) == LOW) {
    //making_sound();
    digitalWrite(7, HIGH);
    making_sound();
  } else {
    digitalWrite(7, LOW);
  }
}
```

(b) Code for combining all components

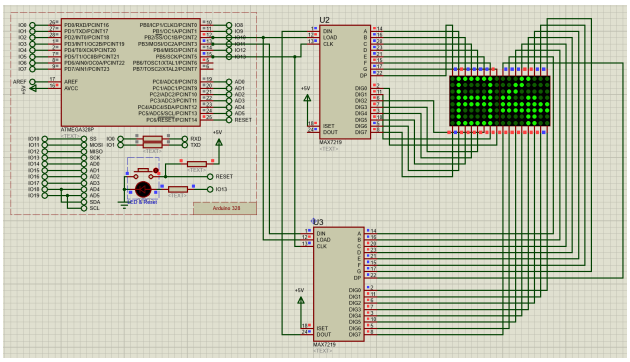
Kapitel 2

Matrix: How to work with a LED Matrix

The second part dealt with an LED matrix. A matrix has a size of 8 columns * 8 rows with a total of 16 pins.
n this exercise, second matrices, i.e. 16 * 8, had to be placed next to each other and viewed as one matrix. In order to have control over a matrix, it was connected to the Arduino via the "MAX7219". This is an LED display driver and is responsible for the control of a dot matrix, communication with a microcontroller and the input and output. Only three PINs are required for the input: DIN - serial data input, LOAD - load data input, CLK - serial clock input. If several matrices are to be connected to one another, this is done via DOUT - Serial-Data Output. The DOUT of one matrix is connected to the DIN of another matrix in order to exchange data with each other and thus to create a link.

2.1 LED Matrix Counter

In the first task, two matrices had to be created which show a random number. A counter had to be used for this.
How did you implement it?
The procedure was as follows: first, the standard library for the matrix was integrated in order to connect the matrices with the microcontroller. Then the numbers 0-9 were written down in binary format. You can choose between binary and hexa. It is important to note, however, that the number of rows and columns match those of the matrix, i.e. 8 * 8. With setup () each matrix has now been updated and the counter for calling up the various numbers has been integrated in loop (). For loops were used for this, so that the interval [0; 9] was initially observed and a random number is displayed on each matrix. The method random () was called for the random number, which contains the minimum value = 0 and the maximum value = 9. At the end of each for loop, it must be ensured that clearDisplay () is called in order to clear the matrix display for further numbers .
Which is your concept idea?
The idea here lies in the for loop, which goes through the created array, which contains the numbers in binary representation, and selects a random number. The counter here is the for loop with the query of the interval.



(a) LED Matrixes showing random numbers

```
int min = 0; //nur Zahlen von  0 bis...
int max = 9; //...9

void loop() {
  int n;

  for(int i=min; i<max; i++)
  {
    n = random(min, max);
    for(int r=0; r<8; r++) {
      lc.setRow(0,r,numbers[n][r]);
    }

    n = random(min, max);
    for(int r=0; r<8; r++) {
      lc.setRow(1,r,numbers[n][r]);
    }

   delay(500);
   lc.clearDisplay(1);
   lc.clearDisplay(0);
  }
}
```

(b) Code for the random numbers

2.2 LED Matrix for output of numbers

The second part of the task refers to the task before with the difference that now the numbers 0-128 are to be displayed. For this purpose, another matrix was added, so that we now have a structure of 24*8, i.e. three matrices.

What functions could be useful for achieving this?

A function was not used for this task. Only the code from the previous task was taken over and the code in the loop() was changed. With the help of the function setRow() was selected which row and which column at which display, so which matrix, should be displayed.

Important to note here is to specify the number of MAX7219 correctly, this is do ne by changing the number at "LedControl(DATA,CLK,LOAD, number of matrices)". This leads to the fact that the matrices are linked together.

How we can add numbers to the matrix so its correctly displayed?

As an aid again for loops were written, altogether four. The first for-loop serves for the correction that the interval [0;128] is kept. The second serves for the first matrix, the third for-loop for the middle matrix and the fourth for-loop for the left matrix. The idea of the modulo calculation was implemented: on the basis of the number 128 follows the explanation...

128 mod 10 = 8

128 mod 100 = 28 $--> $ but we only want to display the 2 in the middle matrix, therefore:

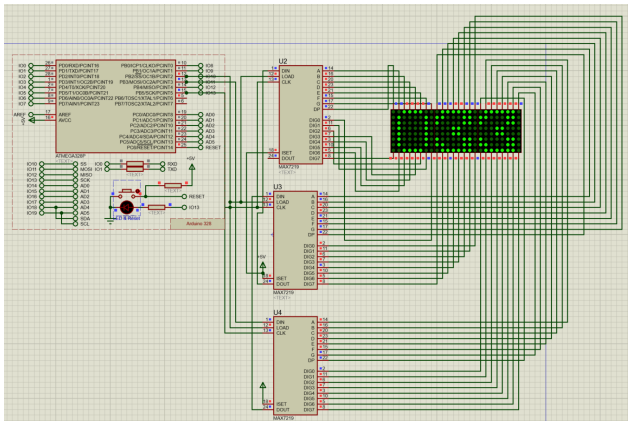
128 / 10 = 12,8 mod 10 = 2

128 mod 1000 = 128 $--> $ but we only want to display the 1 in the first matrix, therefore:

128 / 100 = 1,28 mod 10 = 1

Since we want to start with 0 and count up from there, we therefore start with the last

matrix first and go to the middle matrix and then to the first matrix.



(a) Three LED Matrixes

```
int min = 0; //nur Zahlen von  0 bis...
int max = 128; //...128

void loop() {
  int n = 0;

  for(int i=min; i<=max; i++)
  {
    n = i % 10; //für die letzte Zahl
    for(int r=0; r<8; r++) {
      lc.setRow(0,r,numbers[n][r]);
    }

    n = i/10 % 10; //für die mittlere Zahl
    for(int r=0; r<8; r++) {
      lc.setRow(1,r,numbers[n][r]);
    }

    n = i/100 % 10; //für die vordere Zahl
    for(int r=0; r<8; r++) {
      lc.setRow(2,r,numbers[n][r]);
    }
    delay(100);
  }
}
```

(b) Code for the numbers

2.3 LED Matrix for drawing

The third task combines the first two tasks in that an image, movement or something else is to be displayed on one, two or three matrices. For this, two matrices were chosen and two emojis, a face and a heart.

How does this function work?

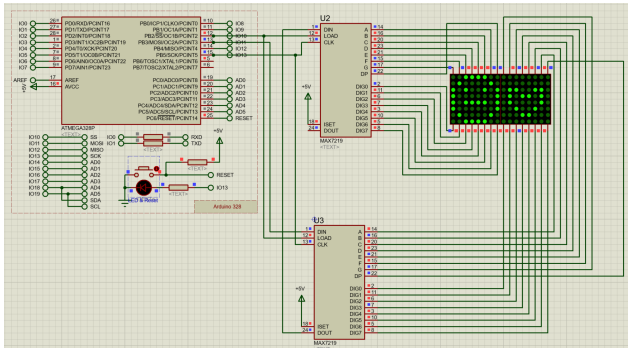
The function works like the representation of numbers with the difference that first hexa was used instead of binary and another function was written.

What else is necessary?

In this function display() a for loop is used in which both displays of the matrices are requested and depending on the matrix the decisive emoji is displayed with setRow(). In the function emojisOnMatrix() the display() function is called by passing the hexacode for the particular emoji. A timer of 500ms each has also been added for better display.

What did you draw?

The left matrix shows a face, which smiles, looks normal and sad. The right matrix shows a heart, which becomes smaller and smaller. Depending on the size of the heart, the expression of the face changes. If the heart is large, the face smiles. If the heart is medium, the face looks normal and if the heart is very small, the face looks sad.



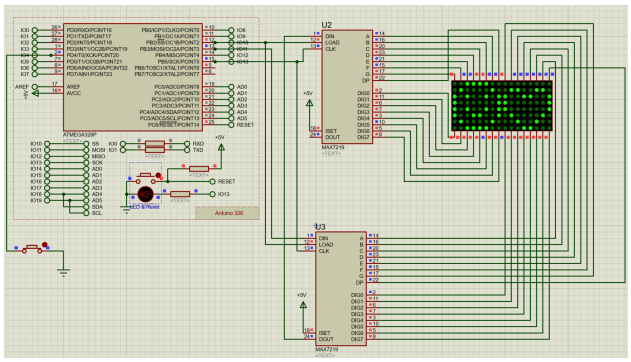
(a) LED Matrixes with emoji

```
void emojisOnMatrix() {

  byte happy[16]=    {0x3C,0x42,0xA5,0x81,0xA5,0x99,0x42,0x3C,0x00,0x66,0x99,0x81,0x81,0x42,0x24,0x18};
    byte neutral[16]= {0x3C,0x42,0xA5,0x81,0xBD,0x81,0x42,0x3C,0x00,0x00,0x66,0x5A,0x42,0x24,0x18,0x00};
    byte sad[16]=      {0x3C,0x42,0xA5,0x81,0x99,0xA5,0x42,0x3C,0x00,0x00,0x24,0x3C,0x24,0x18,0x00,0x00};

  display(happy);
  delay(500);
  display(neutral);
  delay(500);
  display(sad);
  delay(500);
}

void display(byte character[]) {
  for(int i=0; i<16; i++)
  {
    lc.setRow(1,i,character[i]);
    lc.setRow(0,i,character[i+8]);
  }
}


void loop() {
  emojisOnMatrix();
}
```

(b) Code for drawing

2.4 LED Matrix for drawing with Button

In the last task, the previous task must be extended by a button. When the button is pressed, drawing should start. When the button is released, the drawing should stop and when the button is pressed again the drawing will go on.
How does this work?
A large change of the code is not present. Only the code of the button was taken over from the first task part. There is a query whether the button is pressed or not and what should happen then. *What did you had to modify or to add?*
First, a button was added in Proteus, which was connected to the Arduino through PIN 4. Also, the crucial code for the button was added, as already described.



(a) LED Matrixes with button

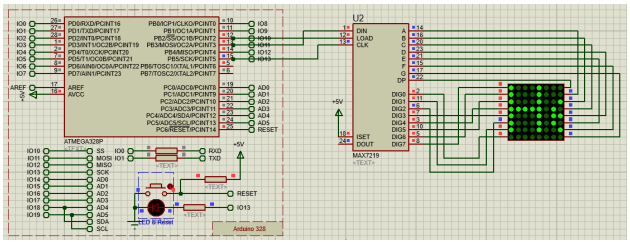
```
  if(digitalRead(4) == LOW) {
    digitalWrite(11, HIGH);
    digitalWrite(10, HIGH);
    digitalWrite(13, HIGH);
    display(happy);
    delay(500);
    display(neutral);
    delay(500);
    display(sad);
    delay(500);
  } else {
    digitalWrite(11, LOW);
    digitalWrite(10, LOW);
    digitalWrite(13, LOW);
  }
}
```

(b) Code for pressing the button

Kapitel 3

LED Matrix as Terminal Output

In the next part, which was an optional exercise, had the output of the Arduino to be combined with the Matrix. When the code runs, the Terminal of the Arduino should appear, you type anything in - normally a senctence - and this senctence should be shown on the display of the matrix.
It was tried with many options, like Serial(), LedController() and at least with Parola(). Final results were done with the Parola Libray. But unfortunately it did not work how it should have. It was only possible to put a sentence as a string to the output and display it in the display of the matrix. But scrolling was not possible as well.
The upcoming pictures show the final results for this exercise.



(a) LED Matrix with text

```
// Including the required Arduino libraries
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

// Uncomment according to your hardware type
//#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define HARDWARE_TYPE MD_MAX72XX::GENERIC_HW

// Defining size, and output pins
#define MAX_DEVICES 1
#define CS_PIN 10

// Create a new instance of the MD_Parola class with hardware SPI connection
MD_Parola myDisplay = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

void setup() {
  // Intialize the object
  myDisplay.begin();

  // Set the intensity (brightness) of the display (0-15)
  myDisplay.setIntensity(15);

  // Clear the display
  myDisplay.displayClear();
}

void text() {
  //myDisplay.setTextAlignment(PA_CENTER);
  //myDisplay.print(String("Hello world!"));
  if(myDisplay.displayAnimate())
    myDisplay.displayText("Hello world!", PA_CENTER, myDisplay.getSpeed(), myDisplay.getPause(), PA_SCROLL_DOWN, PA_SCROLL_UP);
}

void loop() {
  text();
}
```

(b) Code for the terminal

Kapitel 4

Wire layout for temperature measurement

The last major part of the task dealt with the temperature measurement, which was to be displayed on the matrix at the end.

4.1 Implementation

Describe shortly the temperature sensor and how does it work?
The temperature sensor is the "DS18B20". This sensor communicates via the so-called One-Wire protocol and only requires a connection to the microcontroller, here "Arduino 328". To use the sensor correctly, among other things, the following is needed:

- 1: the correct initialization with correct standard libraries
- 2: Integration of the One-Wire protocol
- 3: Interaction with the Arduino and the matrix
- 4: Resistor with 4.7k (!)

In this case, the "DallasTemperature" library was included, which contains efficient and sufficient functions so that the temperature can be read from the sensor and displayed in the matrix. For the matrix, the "LedControl" library was included in this case.
What would be an interesting use case for a temperature sensor like this?
The One-Wire protocol offers a good possibility to use this kind of temperature sensor in any field. This protocol needs only one wire interface to be ready for communication. For example, this sensor can be used in factories. Let's take the example of a chewing gum factory. The dough for the chewing gum must be at a certain temperature to be finally processed. Or this sensor can be used to reflect the temperature of a cold store.

4.2 Temperature reading

How did you visualize the results?
The idea from the second part was taken over. With the help of the modulo calculation each

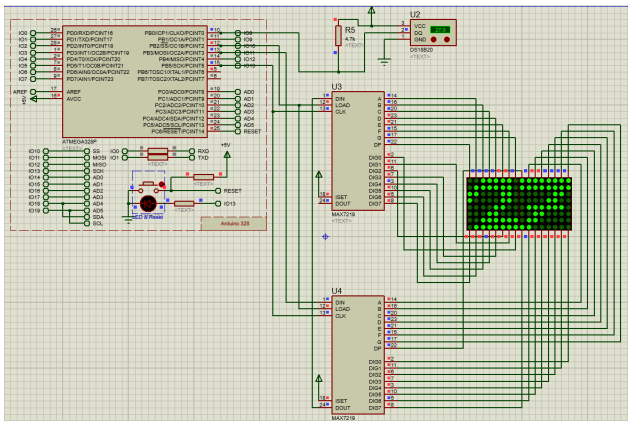
number for each matrix is calculated and provided. With the help of the method setRow() the output happens as LED light on the matrix. For this the library "LedControl.h" was used again.

What is needed and what is your idea?

To map the sensor data from the temperature sensor to the matrix, the following libraries must be included: "DallasTemperature.h" to use the functions to read in the temperature and "LedControl.h" to map the numbers on the matrix. Using the modulo calculation, as mentioned earlier, the number was broken down into its individual parts and displayed on the respective matrix. It is important to mention here that the number was mapped as a whole and the decimal place was ignored.

For what might this be useful?

This sensor can be used, for example, when baking a cake to get the right temperature for the cake dought, or the temperature of the tea water. There are also many more examples just to name a few.



(a) LED Matrixes with temperature sensor

```
void loop() {
  sensors.requestTemperatures();

  // get the temperature from the first sensor only
  float tempCelsius = sensors.getTempCByIndex(0);

  int temp = (int) tempCelsius;

  delay(100);

  if (temp != -127) {
    //sensors.requestTemperatures();
    //tempCelsius = sensors.getTempCByIndex(0);

    int last = temp % 10; //für die letzte Zahl
    for(int r=0; r<8; r++) {
      lc.setRow(0,r,numbers[last][r]);
    }

    int first = temp/10 % 10; //für die erste
    for(int r=0; r<8; r++) {
      lc.setRow(1,r,numbers[first][r]);
    }
  }
}
```

(b) Code for reading the temperatur