IMTHE1

Dion van den Berg

S1060679

Inhoud

Module: IMTHE1
Dion van den Berg (s1060679)

Inleiding

In dit document beschrijf ik verschillende opdrachten voor het vak **IMTHE1**

Elke opdracht is voorzien van de volgende onderdelen

- Korte uitleg van de opdracht
- Aanpak van de opdracht
- Bronvermelding met Datasheets
- Code
- De Fritzing van de opstelling
- Een foto van de opstelling

De video van elke opdracht is meegeleverd in de opgeleverde zip in de naar de opdracht genoemde map.

Ik wens u veel plezier met nakijken!

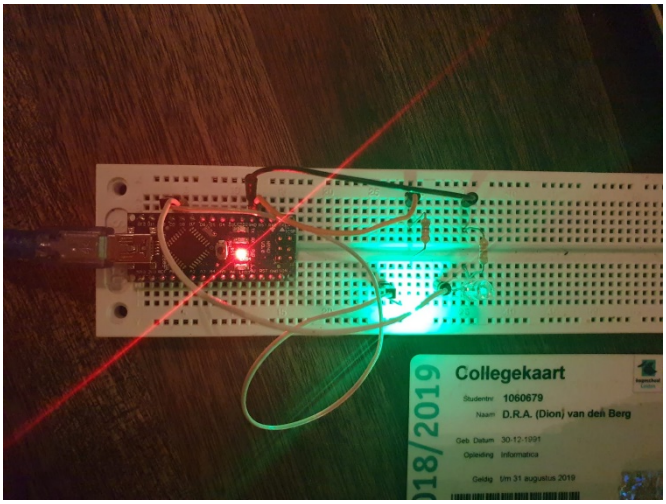
Opdracht 1 Blink once to accept

Voor deze opdracht hadden we net onze eerste les van imthe gehad. Hierin kregen we voorbeelden om een led aan en uit te zetten. Dit heb ik dan daarna ook omgevormd om dit voor deze opdracht te gebruiken door een delay toe te voegen van 250 ms per led waarna ik de andere led weer uitzet zodat het verschil goed zichtbaar is.

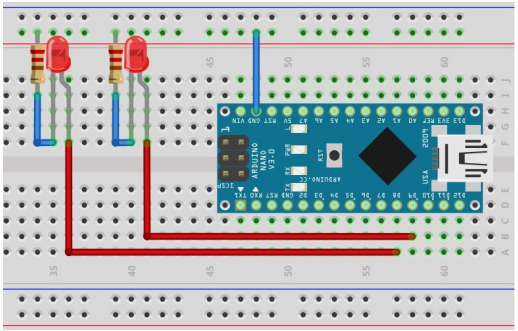
Bronvermeldingen:
Datasheet LED: https://learn.adafruit.com/all-about-leds/the-led-datasheet

```
#include <avr/io.h>

#include <util/delay.h>

int main(void){

        // initialiseer pb0 en pb1

        DDRB = 0b00000011;

        // endless loop

        while(1){

                PORTB = 0b0000010; //zet led 1 aan en 2 uit

                _delay_ms(250);

                PORTB = 0b00000001; //zet led 1 uit en 2 aan

                _delay_ms(250);

        }

        return 0;

}
```

Foto van de opstelling



Fritzing



Opdracht 2.1 Count me in

Voor deze opdracht moesten we tellen van 0 tot 9 op een 7 segmenten display. Voor deze opdracht kon ik aan het begin niet een goed leesbare datasheet vinden. Dus voor mijn aanpak heb ik origineel gewoon de verschillende punten gemapped voor mezelf zodat ik wist wel segment wat deed. Hierna heb ik ze aangesloten en de nummers gemaakt in de code voor elke individueel nummer in een array.

Bronvermelding:
Datasheet 7 segment: https://docs.broadcom.com/docs/AV02-1107EN

```
#include <avr/io.h>
#include <util/delay.h>

#define SEVEN_SEGMENT_PORT PORTD
#define SEVEN_SEGMENT_DDR DDRD
#define POWER_PORT PORTB
#define POWER_DDR DDRB
#define POWERPIN PB0

// Zet de verschillende waardes per nummer in een array voor gemakkelijk gebruik.
uint8_t numberarray[] = {
  0b10001000, // 0
  0b11011011, // 1
  0b10100010, // 2
  0b11000010, // 3
  0b11010001, // 4
  0b11000100, // 5
  0b10000100, // 6
  0b11011010, // 7
  0b10000000, // 8
  0b11000000 // 9
};

// Zet de juiste segmenten aan naar aanleiding van het gevraagde nummer
void segmentchanger(uint8_t n){
  SEVEN_SEGMENT_PORT=n;
}

void setup() {
SEVEN_SEGMENT_DDR=0xFF; // Set all as output
SEVEN_SEGMENT_PORT=0xFF; // Segments off
POWER_DDR=0xFF; // Set our power lines as output
POWER_PORT=0xFF; // Power lines turned off
UCSR0B = 0; // serial pins to output
}

int main(void) {
//Setup
setup();
// Run main loop
while(1) {
// Count
  for(uint8_t count = 0; count < 10; count++) { // counter for the numbers to show
    for(uint8_t time=0; time < 240; time++) { // delay
      POWER_PORT = (1 << POWERPIN);
      segmentchanger(numberarray[count]);
      _delay_ms(2);
    }
  }
}
return 0;
}
```

Fritzing

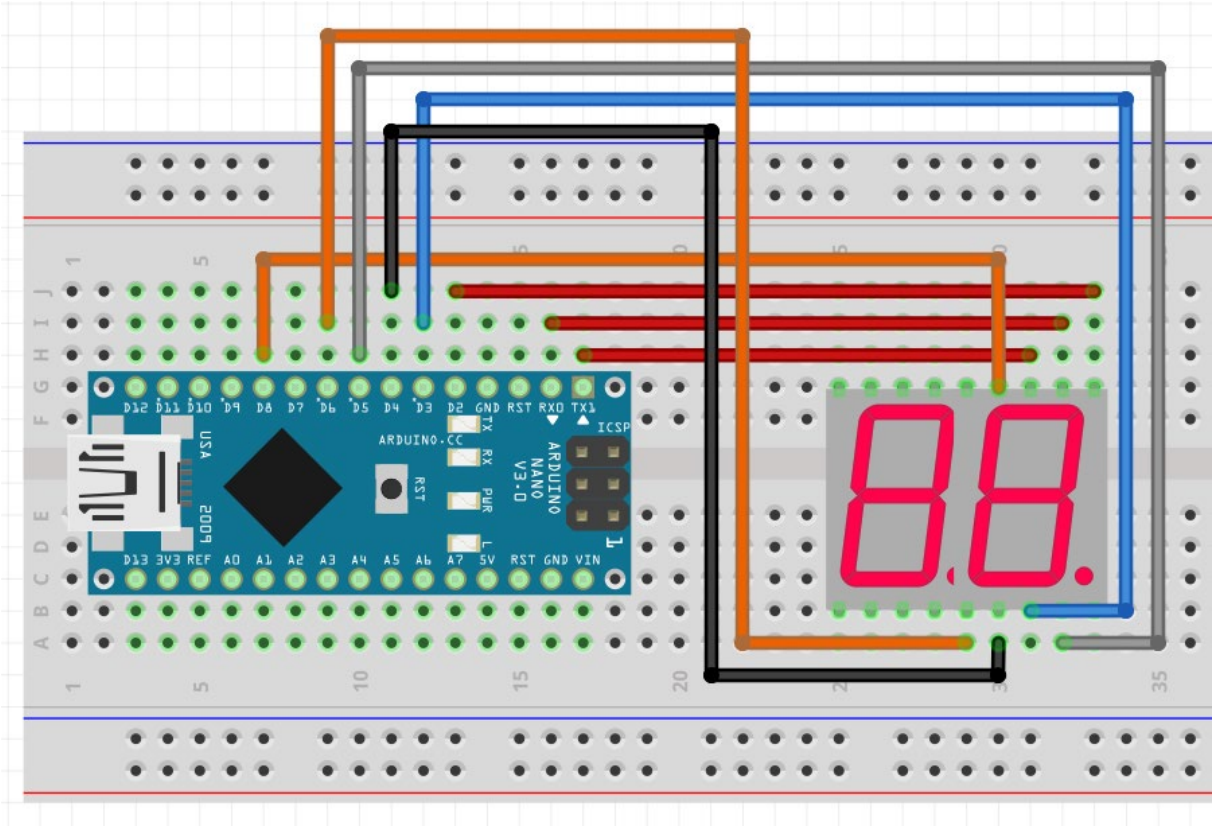
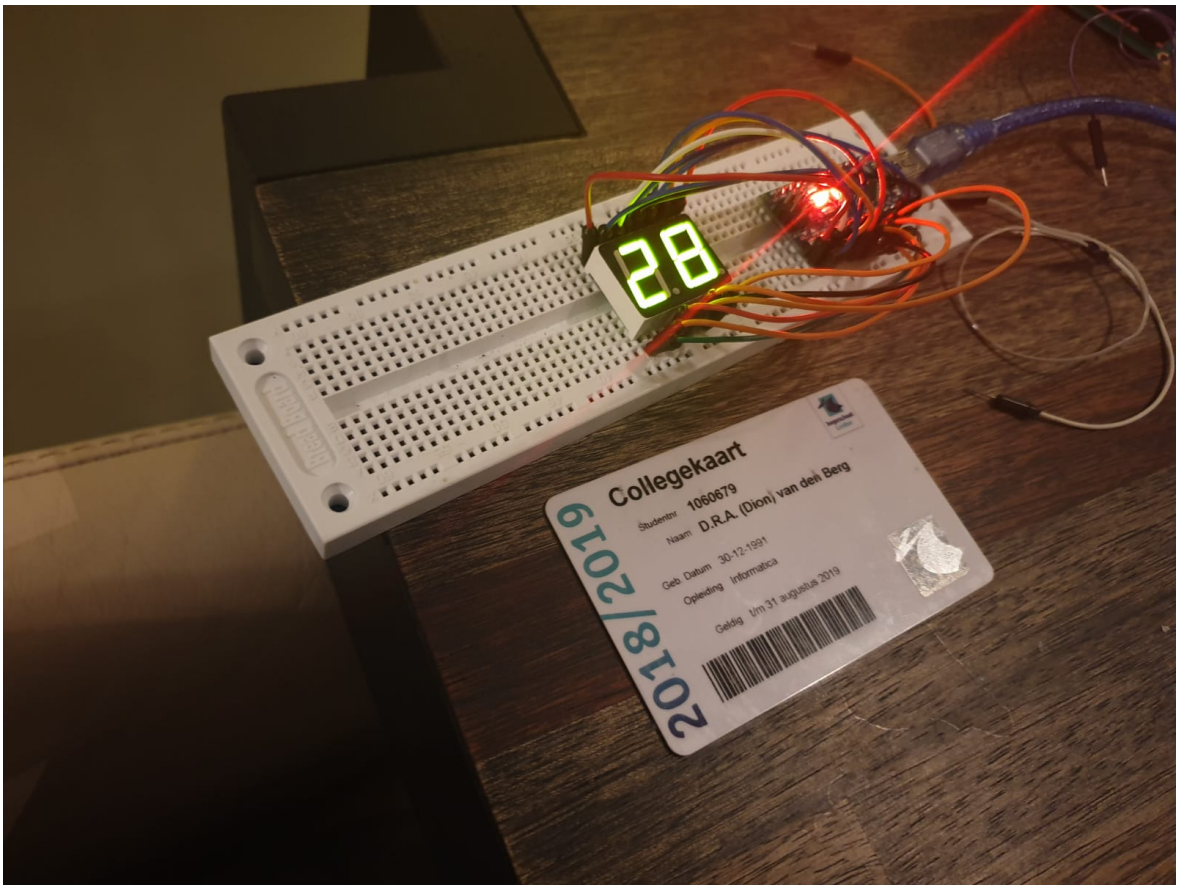


Foto van opstelling (zowel 0 tot 9 als 0 tot 99)



Opdracht 2.2 Count me in

Voor deze opdracht moest er, net als bij de vorige opdracht, geteld worden. Dit keer echter naar 99. Hiervoor kwam ik na het vinden van de datasheet erachter dat er in de fabriek gewoon een 2^e identieke 7 segment aan vast zit geplakt de tientallen op het schermpje. Hierdoor wist ik de pin layout al van het eerste nummer en heb ik ze dan ook verbonden aan dezelfde pins op het nano bordje. Alleen de stroom pins heb ik los aangesloten zodat ik door gebruik van pwm op verschillende momenten stroom kon zetten op een kant van het schermpje.

Bronvermelding

Datasheet 7 segment: https://docs.broadcom.com/docs/AV02-1107EN

```
#include <avr/io.h>
#include <util/delay.h>

#define SEVEN_SEGMENT_PORT PORTD
#define SEVEN_SEGMENT_DDR DDRD
#define POWER_PORT PORTB
#define POWER_DDR DDRB
#define POWERPIN_1 PB0
#define POWERPIN_2 PB1

// Zet de verschillende waardes per nummer in een array voor gemakkelijk gebruik.
uint8_t nummers[] = {
 0b10001000, // 0
 0b11011011, // 1
 0b10100010, // 2
 0b11000010, // 3
 0b11010001, // 4
 0b11000100, // 5
 0b10000100, // 6
 0b11011010, // 7
 0b10000000, // 8
 0b11000000  // 9
};

void segmentchanger(uint8_t n){
 SEVEN_SEGMENT_PORT=n;
}

void setup() {
 SEVEN_SEGMENT_DDR=0xFF; // Set all as output
 SEVEN_SEGMENT_PORT=0xFF; // Segments off
 POWER_DDR=0xFF; // Set our power lines as output
 POWER_PORT=0xFF; // Power lines turned off
 UCSR0B = 0; // Turns off serial pins to be used as output
}

int main(void) {
 // Setup
 setup();

 // Run main loop
 while(1) {

  for(uint8_t count1 = 0; count1 < 10; count1++) { // tientallen
   // Count
   for(uint8_t count2 = 0; count2 < 10; count2++) { // enkelen
    for(uint8_t time=0; time < 240; time++) { // doe dit 240 ms lang voor het volgende getal komt
     POWER_PORT = (1 << POWERPIN_2);
     //SevenSegment(count1);
     segmentchanger(nummers[count1]);
     _delay_ms(1);
     POWER_PORT = (1 << POWERPIN_1);
     //SevenSegment(count2);
     segmentchanger(nummers[count2]);
     _delay_ms(1);
    }
   }
  }
 }
            return 0;
}
```

Fritzing

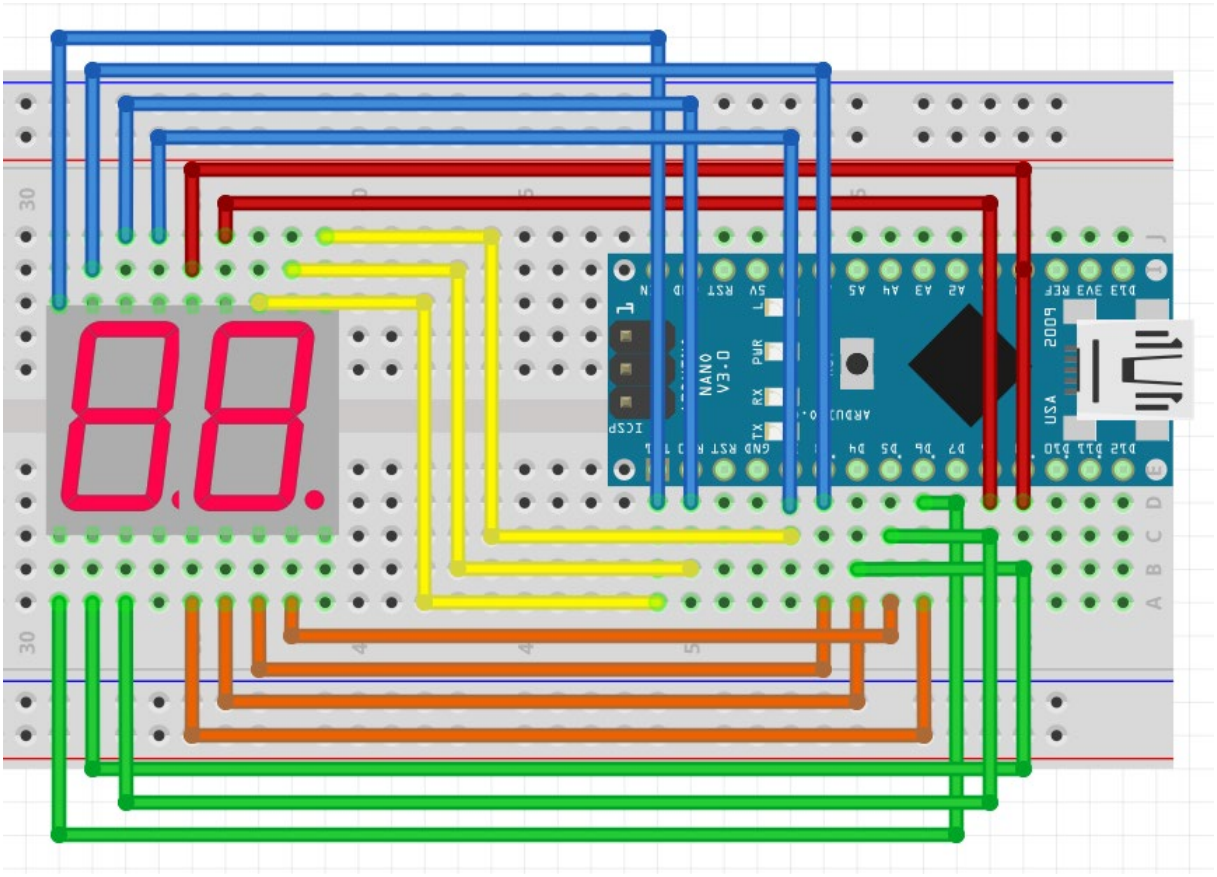
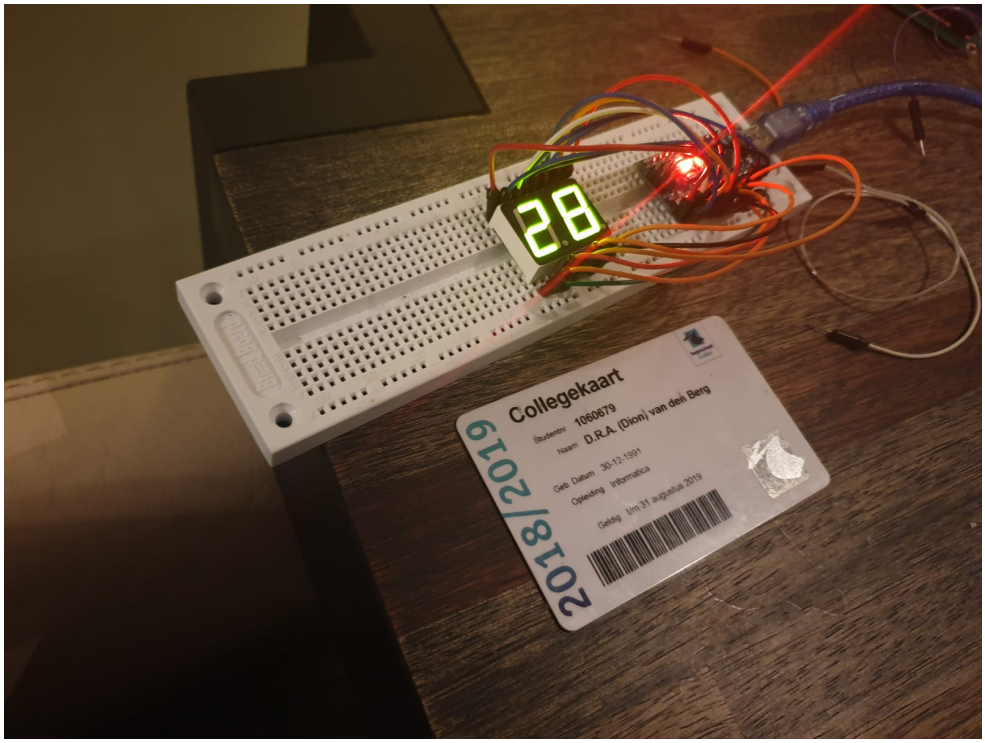
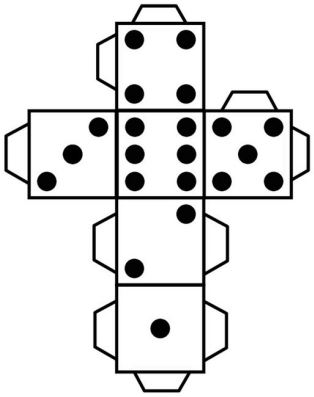


Foto van opstelling (zowel 0 tot 9 als 0 tot 99)



Opdracht 3.1 ADC Baby



Voor deze opdracht moesten we een dobbelsteen simuleren. Hiervoor werd met ledjes de ogen op een dobbelsteen gevisualiseerd. Voor ik begon heb ik eerst even snel voor de zekerheid er een dobbelsteen bijgepakt om te kijken welke leds ik straks aan moest zetten voor welk nummer.

Net hiervoor hadden we les gekregen in het gebruik van pwm om meerdere leds aan te zetten met het gebruik van dezelfde pinnen. Dit heb ik dan ook toegepast om de 9 leds die ik had onder te verdelen in rijen en kolommen.

Hierna heb ik uitgezocht welke led met welke port combinatie aanging en heb ik de combinatie opgeslagen voor alle ogen op de dobbelsteen. Hierna zette ik deze in een switch case per dobbel combo met een delay zodat de ogen van de steen zichtbaar waren voor het blote oog.

Als laatste heb ik de button verbonden in een interupt die een functie aanriep die een random dobbelsteen nummer genereerde om zo de juiste ogen te laten zien uit de switch case.

Bronvermelding:
Datasheet LED: https://learn.adafruit.com/all-about-leds/the-led-datasheet
Datasheet button: https://components101.com/switches/push-button

Code:

```
#include <avr/io.h>
#include <util/delay.h>
#include <time.h>
#include <stdlib.h>
#include <avr/interrupt.h>

#define POWER_PORT PORTC
#define POWER_DDR DDRC
#define GROUND_PORT PORTB
#define GROUND_DDR DDRB
#define PIN_PORT PORTD
#define PIN_DDR DDRD

uint8_t dice = 1; // start getal

void ledsOff() {
  // zet alle leds uit
  POWER_PORT = 0; // Power lines off
  GROUND_PORT = 255; // Ground lines high
}
ISR(INT0_vect){
  // hardware interupt op de button om te rollen wanneer ingedrukt
  dice = (rand() % 6 + 1);
_delay_ms(200): //delay om niet te triggeren wanneer de button weer omhoog gaat
}

void initINT0(){
  // innitialiseer de hardware interupt
  EIMSK |= (1 << INT0);
  EICRA |= (1 << ISC00);
  sei();
}

void rollDice(uint8_t diceroll) {
  // laat het nummer zien dat gerolt is
  if(diceroll>0 && diceroll<7) {
    switch (dice) {
      case 1:
        POWER_PORT = (1 << PB1);
        GROUND_PORT ^= (1 << PC1);
        break;
      case 2:
        ledsOff();
        POWER_PORT = (1 << PB0);
```

```
      GROUND_PORT ^= (1 << PC0);
      _delay_ms(1);
      ledsOff();
      POWER_PORT = (1 << PB2);
      GROUND_PORT ^= (1 << PC2);
      _delay_ms(1);
      break;
    case 3:
      ledsOff();
      POWER_PORT = (1 << PB0);
      GROUND_PORT ^= (1 << PC0);
      _delay_ms(1);
      ledsOff();
      POWER_PORT = (1 << PB1);
      GROUND_PORT ^= (1 << PC1);
      _delay_ms(1);
      ledsOff();
      POWER_PORT = (1 << PB2);
      GROUND_PORT ^= (1 << PC2);
      _delay_ms(1);
      break;
    case 4:
      POWER_PORT = (1 << PB0);
      GROUND_PORT ^= (1 << PC0);
      _delay_ms(1);
      POWER_PORT = (1 << PB2);
      GROUND_PORT ^= (1 << PC2);
      _delay_ms(1);
      break;
    case 5:
      ledsOff();
      POWER_PORT = (1 << PB0) | (1 << PB2);
      GROUND_PORT ^= (1 << PC0);
      _delay_ms(1);
      ledsOff();
      POWER_PORT = (1 << PB1);
      GROUND_PORT ^= (1 << PC1);
      _delay_ms(1);
      ledsOff();
      POWER_PORT = (1 << PB0) | (1 << PB2);
      GROUND_PORT ^= (1 << PC2);
      _delay_ms(1);
      break;
    case 6:
      ledsOff();
      POWER_PORT = (1 << PB0) | (1 << PB1) | (1 << PB2);
      GROUND_PORT ^= (1 << PC0);
      _delay_ms(1);
      ledsOff();
      POWER_PORT = (1 << PB0) | (1 << PB1) | (1 << PB2);
      GROUND_PORT ^= (1 << PC2);
      _delay_ms(1);
      break;
   };
  } else {
   //voorkom foute worpen
   dice = (rand() % 6 + 1);
  };
}

void setup() {
  POWER_DDR=0xFF; // Power bank as output
  GROUND_DDR=0xFF; // Ground bank as output
  PIN_DDR=(1 << PD2); // Button pin as output
}

int main(void) {
  // Setup
  setup();
  initINT0();
  // Turn off the leds
  ledsOff();

  while(1) {
    rollDice(dice);
  }
  return 0;
}
```

Fritzing

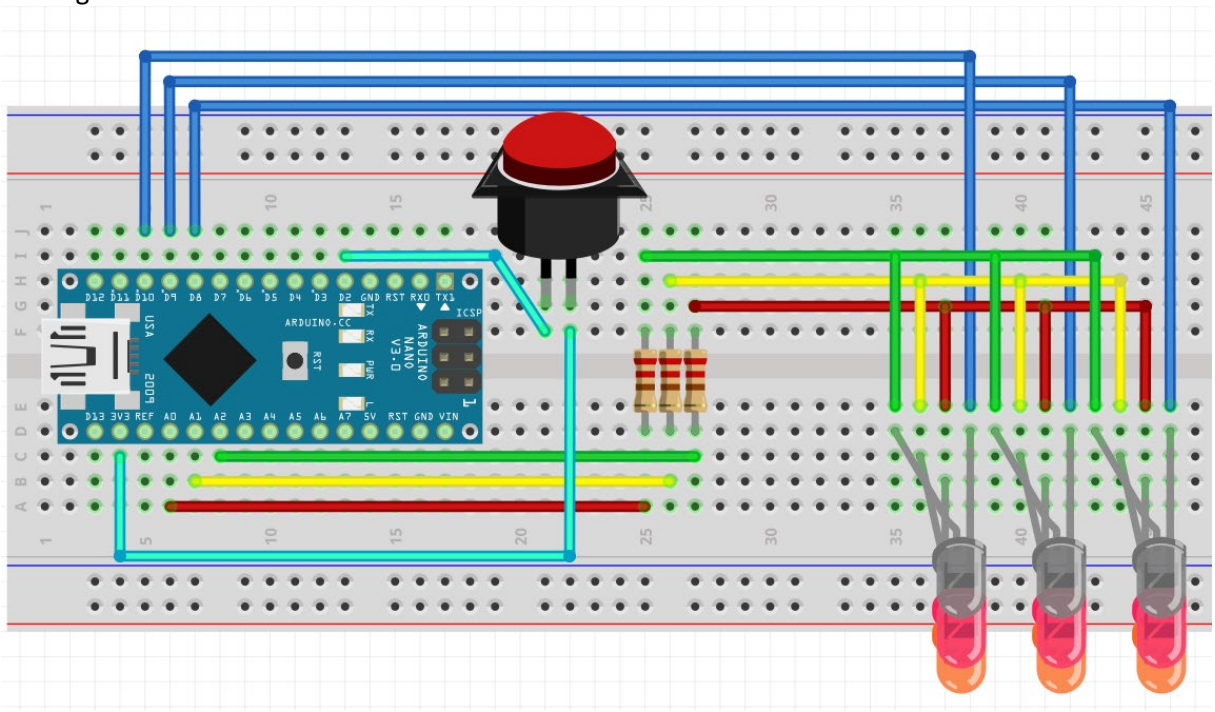
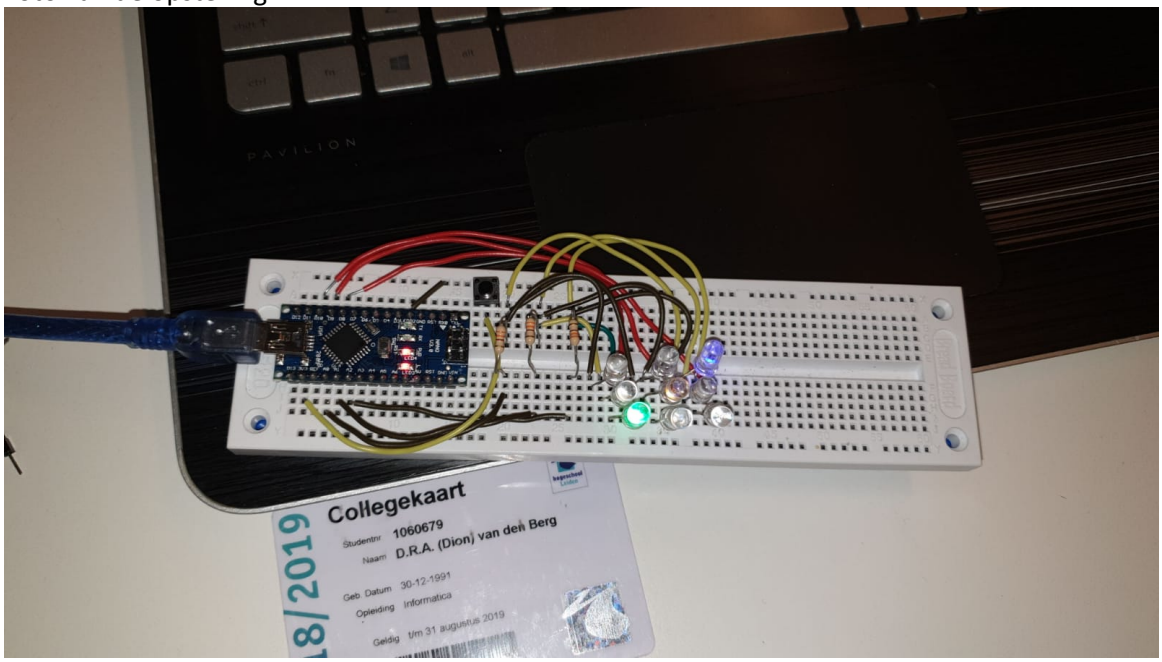


Foto van de opstelling



Opdracht 3.2 ADC Baby

Deze opdracht wilde een speakertje aansturen door midden van pwm. De waarde die je hieraan meegeeft wordt gegenereerd door een potmeter die ook aangesloten is.

De aanpak die ik heb gebruikt is uit de voorbeeldcode uit het AVR boek over pwm en de settings van de ports.

Bronvermelding:
Piezo speaker: http://www.alliedelec.com/m/d/69ca74a806a82d744141c552ce43700a.pdf
Potentiometer: https://cdn-shop.adafruit.com/datasheets/p160.pdf

```
#include <avr/io.h>
#include <util/delay.h>

#define SPEAKER PD2
#define SPEAKER_PORT PORTD
#define SPEAKER_DDR DDRD
#define POT PC5
#define POT_PORT PORTC
#define POT_PIN PINC
#define POT_DDR DDRC

// homemade delay function
void Playtime(uint16_t count) {
  while(count--) {
    _delay_us(1);
  }
}

// Play sound for t time using PWM
void playSound(uint16_t t) {
  SPEAKER_PORT = (1<< SPEAKER); // Speaker on
  Playtime(t); // Delay for input t
  SPEAKER_PORT = 0; // Speaker off
  Playtime(t); // Delay for input t
}

// Read values from potentiometer
uint16_t readMeter(uint8_t channel) {
  ADMUX = (0xf0 & ADMUX) | channel; // nieuw kanaal
  ADCSRA |= (1 << ADSC); // conversie start
  loop_until_bit_is_clear(ADCSRA, ADSC); // wacht to alle data binnen is

  return (ADC);
}

void setupSpeaker() {
  SPEAKER_DDR |= (1 << SPEAKER); //speaker aanzetten
}

void setupMeter() {
  ADMUX |= (1 << REFS0); // Reference voltage = 5v
  ADCSRA |= (1 << ADPS1) | (1 << ADPS0); // Turn on ADC
  ADCSRA |= (1 << ADEN); // Conversion start
}

int main(void) {
  // Setup
  setupSpeaker();
  setupMeter();

  // Run main loop
  while(1){
    playSound(readMeter(POT)); // Play sound using the pot value.
  }
  return 0;
}
```

Fritzing

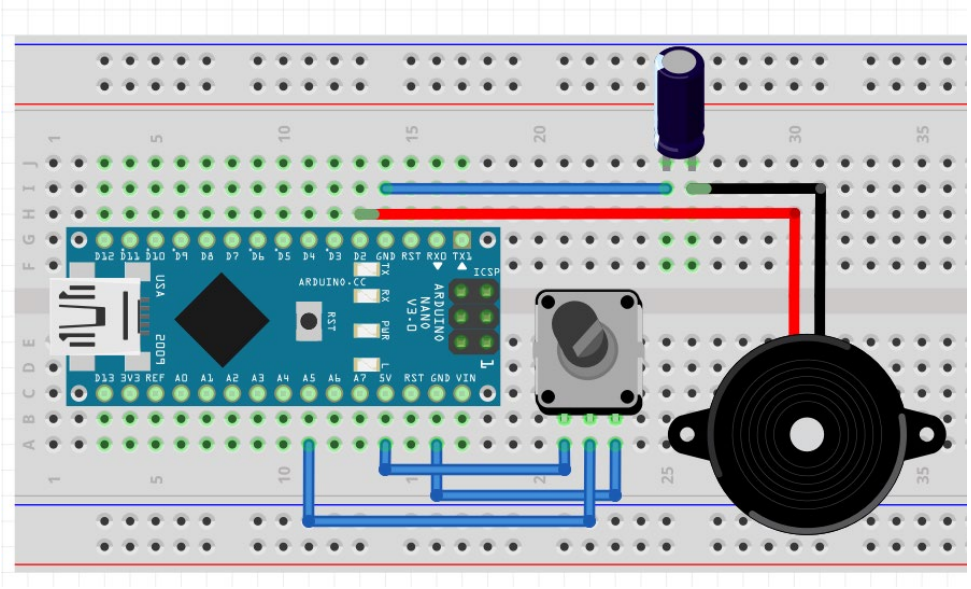
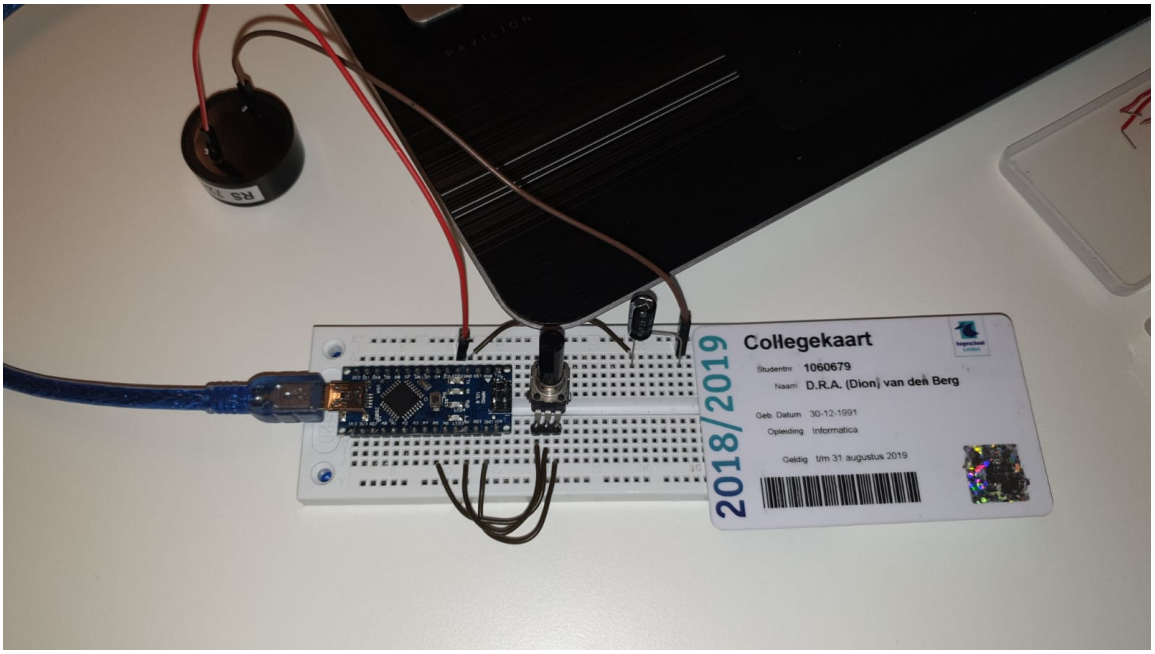


Foto van de opstelling



Opdracht 4.1 Led there be light

Voor deze opdracht moest er een RGB led aangestuurd worden door gebruik van PWM en timers. Om dit voor elkaar te krijgen kon ik mijn oude code hergebruiken om 1 led aan te sturen aangezien een RGB led niets meer is dan 3 ledjes in 1 omhulsel. Omdat ik aan het einde er toch wat leuks mee wilde doen heb ik een voorbeeld gebruikt van James Otron om een leuke regenboog visualisatie erin te stoppen.

Bronvermelding

RGB rainbow: https://gist.github.com/jamesotron/766994
RGB LED datasheet: https://www.arduino.cc/documents/datasheets/LEDRGB-L-154A4SURK.pdf

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define LED_PORT PORTB
#define LED_PIN PINB
#define LED_DDR DDRB
#define LEDr OCR2A //PB3
#define LEDg OCR1B //PB2
#define LEDb OCR1A //PB1

// timer function from Make: AVR book
static inline void initTimers(void) {
  // Timer 1
  TCCR1A |= (1 << WGM10); // Fast PWM mode, 8-bit
  TCCR1A |= (1 << COM1A1); // PWM output on OCR1A
  TCCR1A |= (1 << COM1B1); // PWM output on OCR1B
  TCCR1B |= (1 << WGM12); // Fast PWM mode, pt.2
  TCCR1B |= (1 << CS11); // PWM Freq = F_CPU/8/256
  // Timer 2
  TCCR2A |= (1 << WGM20); // Fast PWM mode
  TCCR2A |= (1 << WGM21); // Fast PWM mode, pt.2
  TCCR2A |= (1 << COM2A1); // PWM output on OCR2A
  TCCR2B |= (1 << CS21); // PWM Freq = F_CPU/8/256
}

void setColourRgb(uint8_t r, uint8_t g, uint8_t b) {
  LEDr = r;
  LEDg = g;
  LEDb = b;
}

ISR(TIMER2_OVF_vect) {

  uint8_t rgbColour[3];
  // Start off with red.
  rgbColour[0] = 255;
  rgbColour[1] = 0;
  rgbColour[2] = 0;

  // Choose the colours to increment and decrement.
  for (int decColour = 0; decColour < 3; decColour += 1) {
    int incColour = decColour == 2 ? 0 : decColour + 1;

    // cross-fade the two colours.
    for(int i = 0; i < 255; i += 1) {
      rgbColour[decColour] -= 1;
      rgbColour[incColour] += 1;

      // set leds to specified brightness
      setColourRgb(rgbColour[0], rgbColour[1], rgbColour[2]);
      _delay_ms(10);
    }
  }
}

void initInterruptTimer() {
  // Overflow interrupt enabled when timers overflow
  TIMSK2 |= (1 << TOIE2);
  // Switch interrupts on
  sei();
}

int main(void) {
  // init
  initTimers();
  initInterruptTimer();
  LED_DDR = (1 << PB1) | (1 << PB2) | (1 << PB3);
  while (1) {} //infinite while loop to keep arduino running
  return (0);
}
```

Fritzing

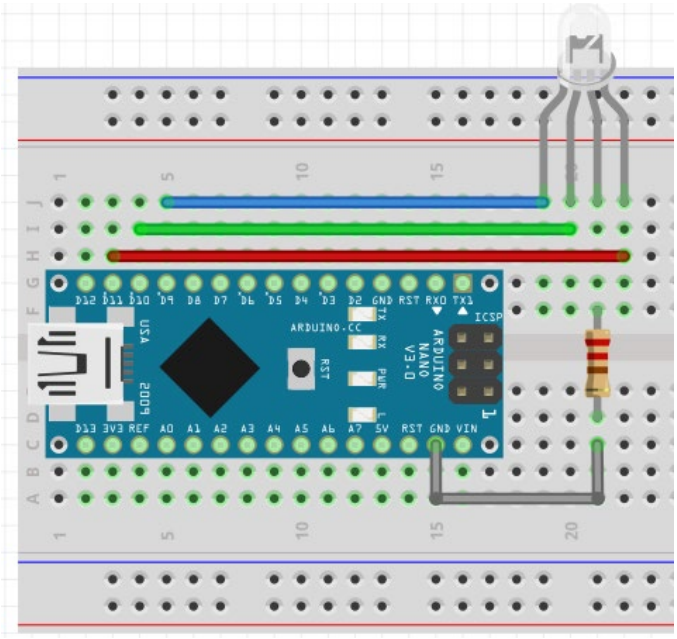


Foto van opstelling

Opdracht 4.2 Led there be light

Voor deze opdracht moest verbinding gemaakt worden met de serial port op je computer. Door het typen van een + of – symbool werd hierdoor de interne led aangezet op D13 van de arduino nano.
Om tot mijn oplossing te komen heb ik gebruik gemaakt van het AVR boek en een stack overflow antwoord waarop ik verder gebouwd heb aangezien ik het origineel niet goed werkend kreeg.

Bronvermelding
Stackexchange oplossing: https://electronics.stackexchange.com/questions/168709/how-to-send-string-in-serial-communication-in-avr-c
Datasheet nano: http://download.arduino.org/products/NANO/Arduino%20Nano-Rev3.2-SCH.pdf

```
#include <avr/io.h>
#include <util/delay.h>
#include <string.h>

#define F_CPU 16000000
#define BAUD 9600
#define BAUD_RATE_CALC ((F_CPU/16/BAUD) - 1)
char rec;
char read;

void serialSetup() {
 // Register settings
 UBRR0H = (BAUD_RATE_CALC >> 8);
 UBRR0L = BAUD_RATE_CALC;
 // Enable transmit and receive
 UCSR0B = (1 << TXEN0)| (1 << TXCIE0) | (1 << RXEN0) | (1 << RXCIE0);
 UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); // Data format = 8 bit
}

// Function that allows sending of a string of chars
void sendString(char* sendString) {
 for (int i = 0; i < strlen(sendString); i++){
  while (( UCSR0A & (1<<UDRE0)) == 0){}; // Wait for buffer to empty
  UDR0 = sendString[i];
 }
}

// Function that allows sending of a single char
void sendChar(char sendChar) {
 while (( UCSR0A & (1<<UDRE0)) == 0){}; // Wait for buffer to empty
 UDR0 = sendChar;
}

// Data received via serial needs to be read and returned in a char
char readData(void) {
 while((UCSR0A & (1<<RXC0)) == 0){}; // wait until read buffer is empty
 rec = UDR0; // Set our last received value
 return rec;
}

int main(void) {
 // Setup internal LED
 DDRB = (1 << PB1);
 // Setup serial
 serialSetup();
 // Print a message that shows the serial interface is ready
 sendString("Sent + to turn on the LED\n");
 sendString("Sent - to turn off the LED\n");

 while(1){
  // Wait until input is read
  read = readData();
  // Print input
  sendString("Input: ");
  sendChar(read);
  sendString(" => ");

  // Turn the led on, off or do nothing, depending on the input
  if (read == '+'){
   // led on
   sendString("LED ON");
   PORTB = (255 << PB1);
  } else if (read == '-'){
   // led off
   sendString("LED OFF");
   PORTB = 0;
  }
  // Make sure next messages get printed on a new line
  sendString("\n");
 }
 return 0;
}
```

Fritzing

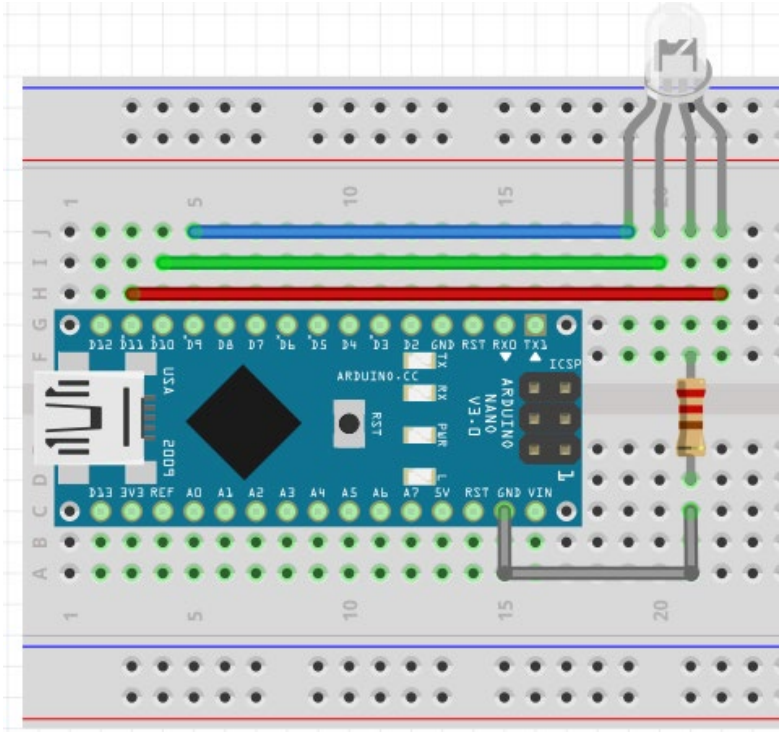
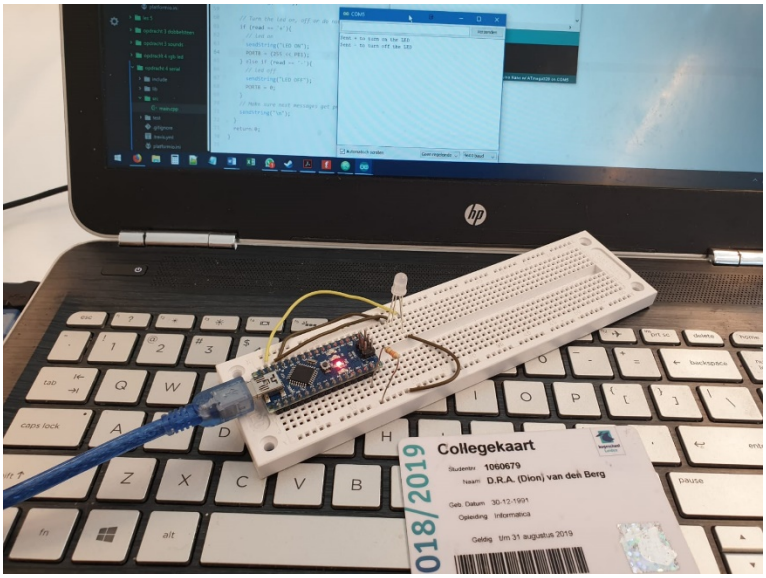


Foto van opstelling



Opdracht 4.3 Led there be light

Voor deze opdracht moest de seriële verbinding van de vorige opdracht gebruikt worden om een RGB led op een bepaalde kleur te zetten

Hiervoor heb ik de verschillende code van 4.1 en 4.2 hergebruikt om dit werkend te maken.

Bronvermelding:

RGB LED datasheet: https://www.arduino.cc/documents/datasheets/LEDRGB-L-154A4SURK.pdf

Code:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <string.h>

#define F_CPU 16000000
#define BAUD 9600
#define BAUD_RATE_CALC ((F_CPU/16/BAUD) - 1)

#define LED_PORT PORTB
#define LED_PIN PINB
#define LED_DDR DDRB
#define LEDr OCR2A //PB3
#define LEDg OCR1B //PB2
#define LEDb OCR1A //PB1

char rec;
char read;
volatile uint8_t brightness = 16;
volatile uint8_t ledWorker = 0;
volatile uint8_t dir = 1;

static inline void initTimers(void) {
  // Timer 1
  TCCR1A |= (1 << WGM10); // Fast PWM mode, 8-bit
  TCCR1A |= (1 << COM1A1); // PWM output on OCR1A
  TCCR1A |= (1 << COM1B1); // PWM output on OCR1B
  TCCR1B |= (1 << WGM12); // Fast PWM mode, pt.2
  TCCR1B |= (1 << CS11); // PWM Freq = F_CPU/8/256
  // Timer 2
  TCCR2A |= (1 << WGM20); // Fast PWM mode
  TCCR2A |= (1 << WGM21); // Fast PWM mode, pt.2
  TCCR2A |= (1 << COM2A1); // PWM output on OCR2A
  TCCR2B |= (1 << CS21); // PWM Freq = F_CPU/8/256
}

void setColourRgb(uint8_t r, uint8_t g, uint8_t b) {
  // zet de kleuren de gevraagt worden naar de pins.
  LEDr = r;
  LEDg = g;
  LEDb = b;
}

void serialSetup() {
  // Register settings
  UBRR0H = (BAUD_RATE_CALC >> 8);
  UBRR0L = BAUD_RATE_CALC;
  // Enable transmit and receive
  UCSR0B = (1 << TXEN0)| (1 << TXCIE0) | (1 << RXEN0) | (1 << RXCIE0);
  UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); // Data format = 8 bit
}

// Function that allows sending of a string of chars
void sendString(char* sendString) {
  for (int i = 0; i < strlen(sendString); i++){
    while (( UCSR0A & (1<<UDRE0)) == 0){}; // Wait for buffer to empty
    UDR0 = sendString[i];
  }
}

// Function that allows sending of a single char
void sendChar(char sendChar) {
  while (( UCSR0A & (1<<UDRE0)) == 0){}; // Wait for buffer to empty
```

```
  UDR0 = sendChar;
}

// Data received via serial needs to be read and returned in a char
char readData(void) {
  while((UCSR0A & (1<<RXC0)) == 0){}; // wait until read buffer is empty
  rec = UDR0; // Set our last received value
  return rec;
}

ISR(TIMER2_OVF_vect) {
  // Wait until input is read
  read = readData();

  // Turn the led on, off or do nothing, depending on the input
  if (read == 'r'){
    sendString("Red on\n");
    setColourRgb(255, 0, 0);
    ledWorker = 0;
  } else if (read == 'g'){
    sendString("Green on\n");
    setColourRgb(0, 255, 0);
    ledWorker = 1;
  } else if (read == 'b') {
    sendString("Blue on\n");
    setColourRgb(0, 0, 255);
    ledWorker = 2;
  }
}

void initInterruptTimer() {
  TIMSK2 |= (1 << TOIE2);
  sei();
}

void pwmLed() {

  if (brightness == 255) {
    dir = -1; // Change dir to low
  } else if (brightness == 0) {
    dir = 1; // Change dir to high
  }
  // Direction influences brightness up or down
  brightness += dir;
  // Turn on LED
  if (ledWorker == 0) {setColourRgb(brightness, 0, 0);}
  if (ledWorker == 1) {setColourRgb(0, brightness, 0);}
  if (ledWorker == 2) {setColourRgb(0, 0, brightness);}

  _delay_ms(1);
  LED_PORT = 0xFF;
}

int main(void) {

  serialSetup();
  initTimers();
  initInterruptTimer();
  DDRB = (1 << PB1) | (1 << PB2) | (1 << PB3);
  //PORTB = (0 << PB1) | (0 << PB2) | (0 << PB3);

  sendString("Sent r to turn on the RED LED\n");
  sendString("Sent g to turn off the GREEN LED\n");
  sendString("Sent b to turn off the BLUE LED\n");

  while(1){
    pwmLed(); //keep looping to increase brightness and decrease to simulate pulse.
  }
  return 0;
}
```

Fritzing

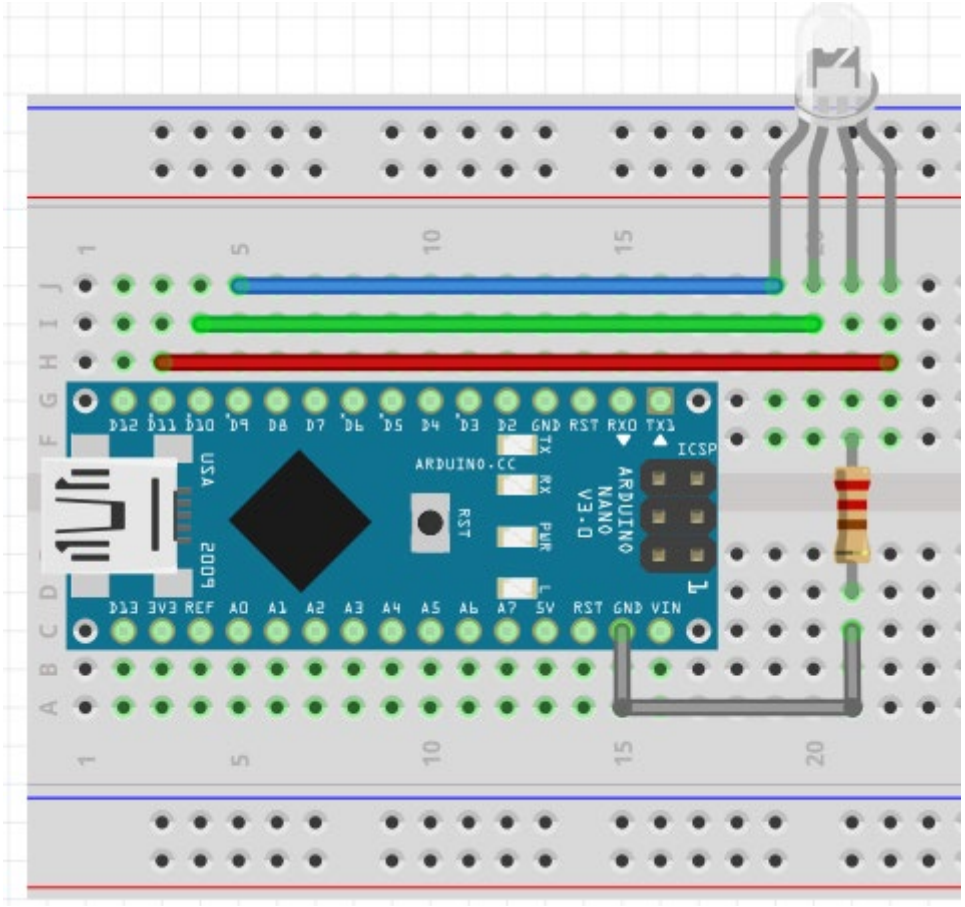
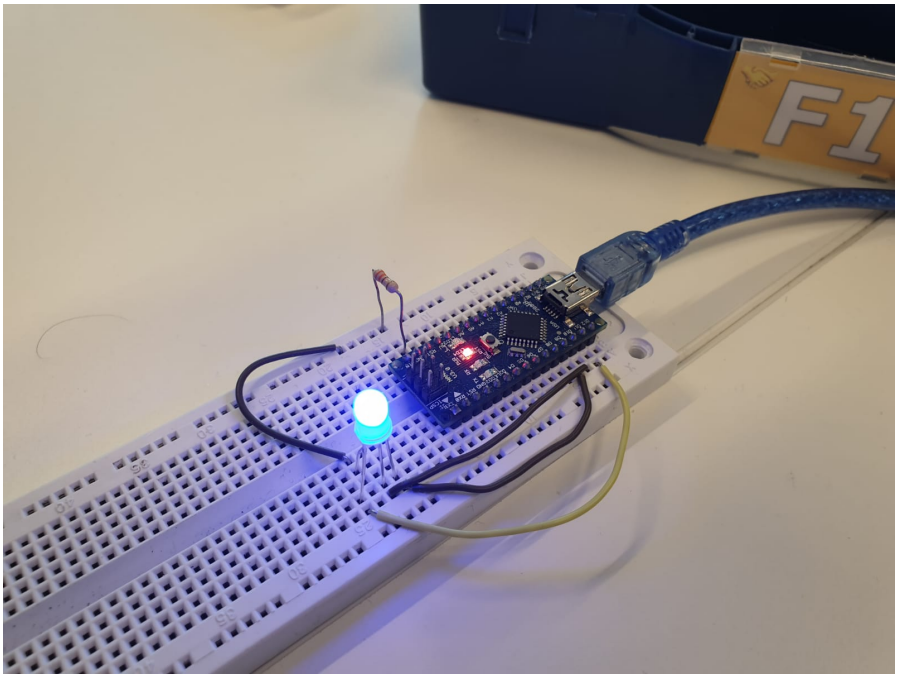


Foto van opstelling



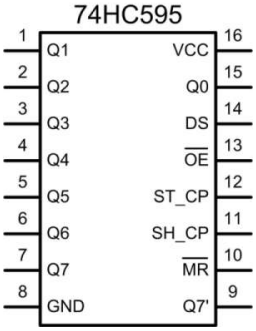
Opdracht 5.1 ET Phone home

Voor deze opdracht moest ik 3 unieke componenten voor mijn imthe box benoemen en zowel de functie als de data sheet beschrijven.
Het ging hierbij om de volgende 3 componenten:

- 74HC595
- MAX6675
- 8x8 LED Matrix

Deze 3 componenten zijn alle 3 zeer verschillende en ik zal ze dan ook proberen zo goed mogelijk te beschrijven betreffende het gebruik in AVR of het nut en functie ervan in de electronica.

74HC595 schuifregister
Datasheet: http://www.ti.com/lit/ds/symlink/sn74hc595.pdf



Protocol met AVR:

Voor het gebruik van het schuifregister heb ik bij de tutorials gekeken op deze website: http://extremeelectronics.co.in/avr-tutorials/using-shift-registers-with-avr-micro-avr-tutorial/

Hierop wordt op een gemakkelijke manier verteld hoe je de aansluiting van het register moet doen en makkelijk en snel een 8 bit reeks ernaartoe kan sturen voor aansturing. In deze tutorial wordt ook een makkelijke write functie aangemaakt voor gebruik.

Het mooie van deze schuifregisters is ook dat als er meer bits verstuurd worden over de lijn alleen de eerste 8 gebruikt worden door het eerste register waarna de rest doorgestuurd kan worden naar een 2^{e} schuifregister als deze is aangesloten.

Thermocouple module MAX6675
Datasheet: https://cdn-shop.adafruit.com/datasheets/MAX6675.pdf

Protocol met AVR:

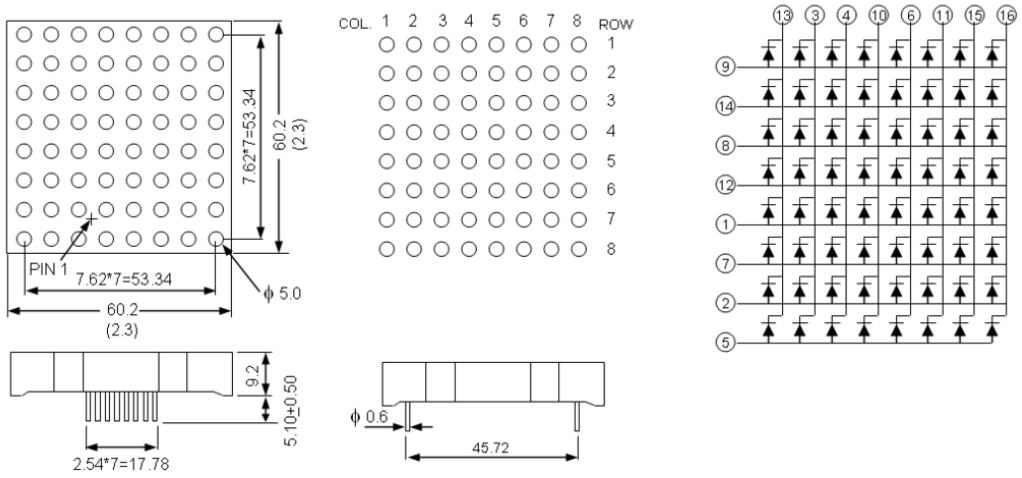
Include de library in de file.
#include "max6675.h"

Initialiseer met de pinnumbers van de Digital pins
int thermoDO = 5;
int thermoCS = 6;
int thermoCLK = 7;

Maak een instance aan
MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);

Lees de Temperatuur met thermocouple.readCelsius(); wat een double getal terug geeft in graden Celcius.

8X8 Led Matrix
datasheet:



Protocol met AVR

Voor dit onderdeel is geen speciaal protocol nodig. Wat wel handig kan zijn is het gebruik van schuifregisters zoals de ovengenoemde 74HC595 hierbij. Dat scheelt het gebruik van een hoop extra pinnen op je Arduino nano. Door gebruik te maken van pwm is gemakkelijk 1 specifieke pin aan te zetten om de matrix. De matrix is onderverdeeld in Rijen en Kolommen. Zoals op de datasheet hierboven is te zien sluit je de linker rij aan als plus signaal en de bovenste rijen als grond. Hierdoor kan je 1 specifieke pin aanzetten.

Een voorbeeld met schuifregisters heb ik hier gevonden op de website van instructables:
https://www.instructables.com/id/Arduino-88-Led-Matrix-Driver-With-2-74HC595-Shift-/

Opdracht 5.2 ET Phone home

Voor deze opdracht werd er gevraagd om een library te importeren voor het gebruik van de lcd. Hierop moest hierna moest er mijn naam en studentnummer opgezet worden.
Om de library te importeren heb ik de readme in de lib folder gelezen van mijn Atom platformIO plugin. Deze liet netjes zien waar ik mijn files moest plaatsen.
Echter was de library geschreven in C in plaats van c++ waardoor het eventjes duurde voor ik erachter kwam waarom hij het niet kon compileren. Na en simpele wijziging van de bestands extensies werkte alles als een zonnetje.

Bronvermelding

LCD datasheet: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

```
// Include our libraries
#include <avr/io.h>
#include <util/delay.h>
#include "hd44780.h"

int main(void) {
  //initialiseer de lcd
  lcd_init();
  // clear het scherm voorgebruikt
  lcd_clrscr();
  // ga naar start van line 1
  lcd_goto(0);
  // zet wat op het scherm
  lcd_puts("Dion van den Berg");
  // skip naar 2e line
  lcd_goto(0x40);
 // zet wat neer op scherm
  lcd_puts("S1060679");
  while(1){} //while to keep things running
  return 0;
}
```

Fritzing

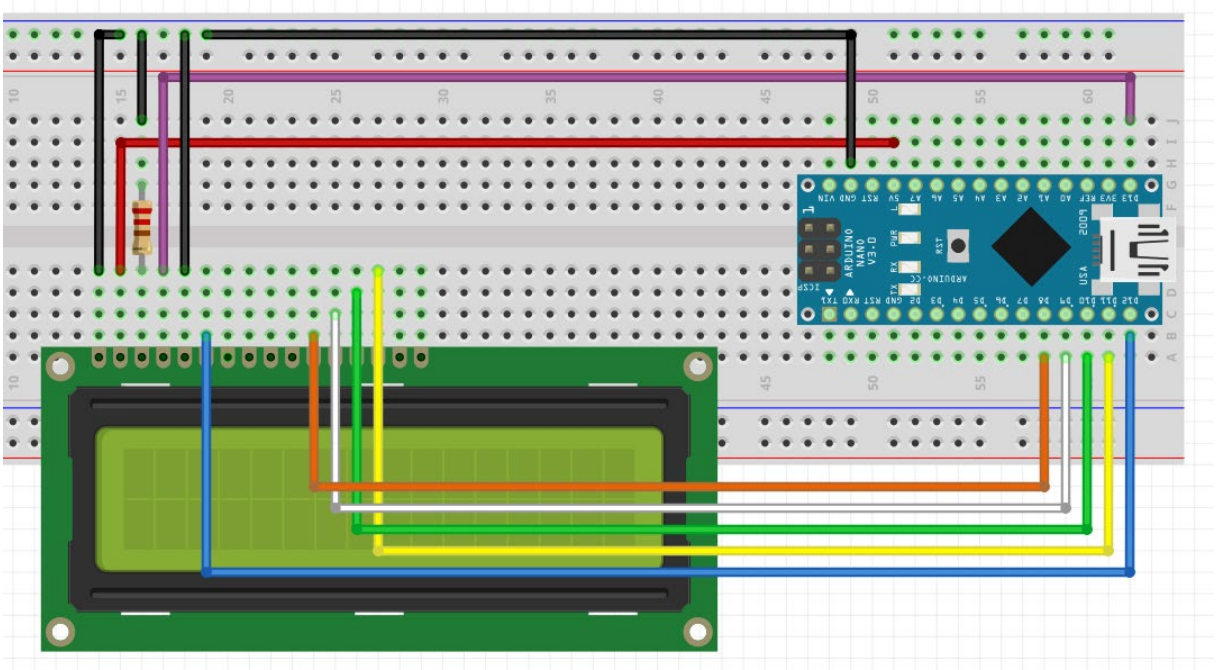


Foto van opstelling

Eind opdracht: Show your moves

Voor deze opdracht moesten we ons eigen concept bedenken om met unieke in en output componenten iets tofs te tonen. Om dit te realiseren heb ik een Temperatuur sensor, een lcd display en een piezo speaker gebruikt.

Concept "Is my tea ready?":
Mijn waterkoker thuis is redelijk geluidloos als hij klaar is met het koken van water. Dit zorgt ervoor dat ik hem soms aanzet en daarna vergeet dat ik heet water heb gemaakt. 30 minuten later loop ik dan langs om hem weer opnieuw aan te zetten waarna hetzelfde weer kan gebeuren. Dit zorgt voor een verspilling van energie en verspilde tijd.

Hierdoor leek het mij leuk om een geluid toe te voegen aan de waterkoker wat afspeelt als hij klaar is. Dit geluid moet storend genoeg zijn dat ik ernaar toe moet lopen om het uit te zetten, omdat je anders geïrriteerd raakt.

Om dit te realiseren en het leuker te maken tussendoor heb ik er een LCD display aan toegevoegd om zo te kunnen zien wat de huidige temperatuur is en of de thee al klaar is.

De thermometer, met gebruik van de MAX6675 library, blijft in een interupt elke 100ms de temperatuur meten. Dit zorgt voor de meest accurate metingen elke keer aangezien hij anders rare waardes gaat geven. Tijdens mijn final versie schiet hij toch nog af en toen 10 graden terug voor hij weer op de juiste temperatuur komt. Dit is jammer genoeg niet op te lossen met mijn huidige competenties of tijd die ik voor dit onderdeel over had.
In de normale loop wordt deze temperatuur op het scherm getoond door gebruik te maken van de HD44780 library. Zodra de temperatuur boven de 100 graden komt zal dit vervangen worden voor "Tea is ready!" en wordt het geluid continu afgespeeld tot deze handmatig wordt uitgezet.

Korte reflectie:
Tijdens het maken van dit concept heb ik een lange tijd geprobeerd een soort fluitketel te maken van de speaker. Echter waren de geluiden die eruit kwam door de onderbreking van de temperatuur sensor niet egaal. Hierdoor werd het geluid niet goed genoeg overgebracht naar mijn mening.
Ook zoals ik hierboven al noemde heb ik een tijd geprobeerd de temperatuur sensor wat minder afwijking te laten hebben. Na een tijd googelen heb ik hier niet direct een goede oplossing voor gevonden. Op een eerdere forum post (http://forum.arduino.cc/index.php?topic=108415.0) was te lezen dat sommige problemen met de sensor komen door het niet goed gronden van de arduino. Dit is moeilijk op te lossen met mijn huidige concept aangezien ik deze niet direct aan de netstroom heb zitten. Een ander forum melden dat de delay tussen metingen minstens 250 ms moest zijn voor 1 meting (https://www.avrfreaks.net/forum/max6675-thermocouple-chip-how-elegantly-handle-values). Dit heb ik uitgetest met verschillende tussenstappen maar ik kon niet vinden dat dit accurater zou moeten zijn.
Uiteindelijk bedacht ik mij pas na het inleveren van de imthe doos dat ik er misschien een losse knop op had kunnen zetten om de buzzer uit te zetten en de meting te resetten. Misschien zelfs met een timer zodat het water af kan koelen voor hij gelijk weer gaat piepen. Dit is een leuke upgrade voor een volgende versie.

Bronvermelding
Thermometer support docs: https://learn.adafruit.com/thermocouple/
LCD datasheet: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf
Piezo speaker: http://www.alliedelec.com/m/d/69ca74a806a82d744141c552ce43700a.pdf
MAX6675 Datasheet: https://cdn-shop.adafruit.com/datasheets/MAX6675.pdf

Code:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "hd44780.h"
#include "max6675.h"

#define F_CPU 16000000
#define BAUD 9600
#define BAUD_RATE_CALC ((F_CPU/16/BAUD) - 1)

#define SPEAKER PD3
#define SPEAKER_PORT PORTD
#define SPEAKER_DDR DDRD
#define POT PC5
#define POT_PORT PORTC
#define POT_PIN PINC
#define POT_DDR DDRC

// initialiseer variabelen
double temp = 0;
char printbuff[100];
int sound;

// thermocouple library needs to know the pin numbers
int thermoDO = 5;
int thermoCS = 6;
int thermoCLK = 7;

// create object voor thermometer
MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);

void Playtime(uint16_t count) {
  while(count--) {
    _delay_us(1);
  }
}

// Play sound for t time using PWM
void playSound(uint16_t t) {
  SPEAKER_PORT = (1<< SPEAKER); // Speaker on
  Playtime(t); // Delay for input t
  SPEAKER_PORT = 0; // Speaker off
  Playtime(t); // Delay for input t
}

// Read values from potentiometer for debugging
uint16_t readMeter(uint8_t channel) {
  ADMUX = (0xf0 & ADMUX) | channel; // nieuw kanaal
  ADCSRA |= (1 << ADSC); // conversie start
  loop_until_bit_is_clear(ADCSRA, ADSC); // wacht to alle data binnen is

  return (ADC);
}

void setupSpeaker() {
  SPEAKER_DDR |= (1 << SPEAKER); //speaker aanzetten
}

void setupMeter() {
  ADMUX |= (1 << REFS0); // Reference voltage = 5v
  ADCSRA |= (1 << ADPS1) | (1 << ADPS0); // Turn on ADC
  ADCSRA |= (1 << ADEN); // Conversion start
}

void setup() {
  // wait for MAX chip to stabilize
  _delay_ms(500);
}

ISR(TIMER1_OVF_vect){
// read thermometer
  temp = thermocouple.readCelsius();
// extra delay to let the Meter settle
  _delay_ms(100);
}

void initTimer1(){
  TCCR1B |= (1 << CS12);
```

```
 TIMSK1 |= (1 << TOIE1); //timer overflow interupt enable
 sei();
}

int main(void) {
 setupSpeaker();
 //setupMeter(); //potmeter for debugging
 initTimer1();
 setup();
 lcd_init();
 lcd_clrscr();
 // go to start of line 1
 lcd_goto(0);
 lcd_puts("Temp of Tea");

 while(1){
  // convert double from temp value to string to put on the lcd
   dtostrf(temp, 10, 2, printbuff);
  //setup screen text
   lcd_clrscr();
   lcd_goto(0);
   lcd_puts("Temp of Tea");
   lcd_goto(0x40);
  // print temp to 2nd row
   lcd_puts(printbuff);

  //extra while loop to fire once temperature reaches 100 degrees celcius
   while (temp >= 100.0){
  // play annoying beep
     playSound(100);

  // show on screen that tea is hotand ready
     lcd_clrscr();
     lcd_goto(0);
     lcd_puts("Temp of Tea");
     lcd_goto(0x40);
     lcd_puts("Tea Ready!");
   }
 }
 return 0;
}
```

Fritzing

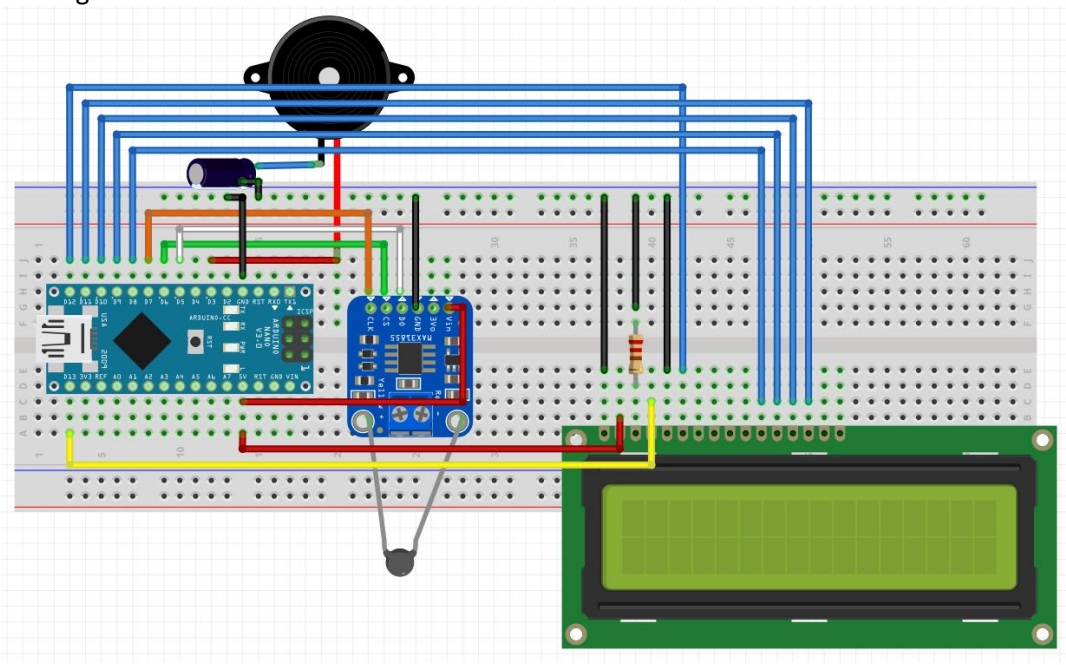
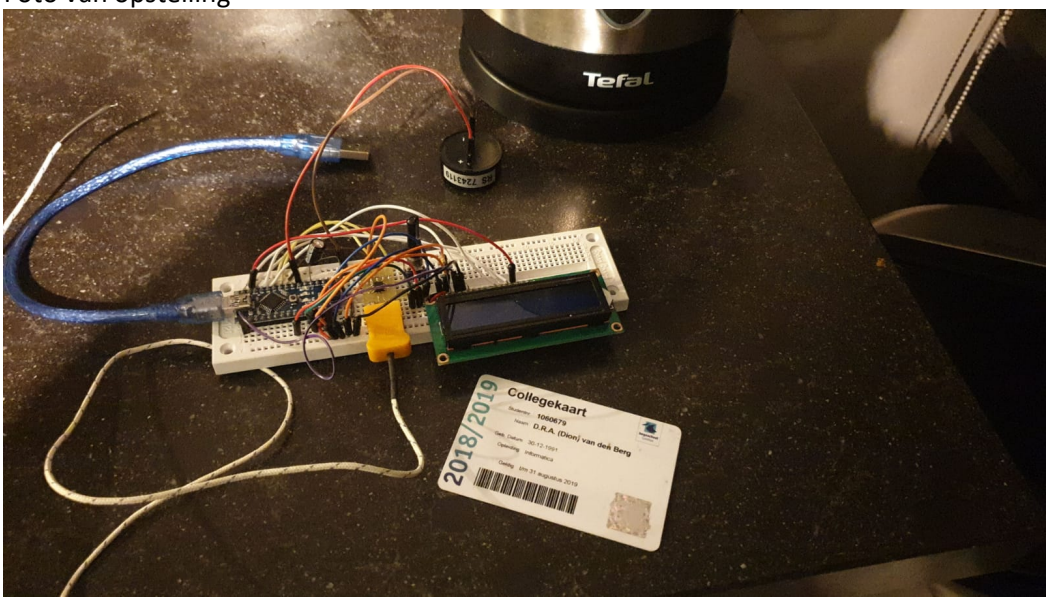


Foto van opstelling



Bewijs van inlevering

Hogeschool Leiden

Interactie Box

Bewijs van Inlevering

10 - April - 2019

Dion van den Berg

Jeroen de Meij / Fred van Krimpen

Bewaar dit certificaat zorgvuldig tot op zijn minst na de definitieve vaststelling van het eindcijfer.