Knipperen

Het eerste oefenprojectje is het ingebouwde LED-lampje laten knipperen. Hiervoor is geen hardware nodig, naast de Arduino zelf. De LED heeft standaard pinnummer 13; dit wordt opgeslagen in een variabele, samen met de snelheid in milliseconde. Tijdens de setup dient de modus van de LED naar output te worden gezet, zodat deze in de loop aan (HIGH) en uit (LOW) kan worden gezet.

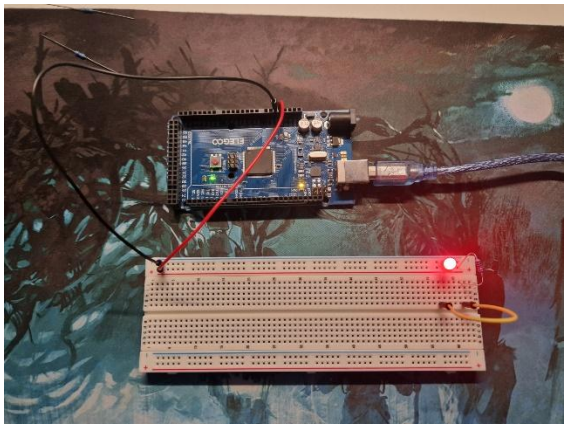
```
// Variables
int led = 13;
int speed = 2000;

// Setup, runs once
// Set the LED pin as an output
void setup() {
  pinMode(led, OUTPUT);
}

// Main loop, runs repeatedly
// Turn the LED on and off, with a delay
// HIGH = on, LOW = off
void loop() {
  digitalWrite(led, HIGH);
  delay(speed);
  digitalWrite(led, LOW);
  delay(speed);
}
```

Bij deze oefening is, naast de Arduino, geen hardware gebruikt.

LED

De eerste hardwarecomponenten die gebruikt worden zijn een LED-lampje en een weerstand. Het doel van deze oefening is om te leren hoe weerstanden werken. Des te hoger te weerstand, des de meer moeite stroom moet doen om een circuit te voltooien. In het geval van een LED betekent dit: des te hoger de weerstand, des te dimmer het lampje brandt.

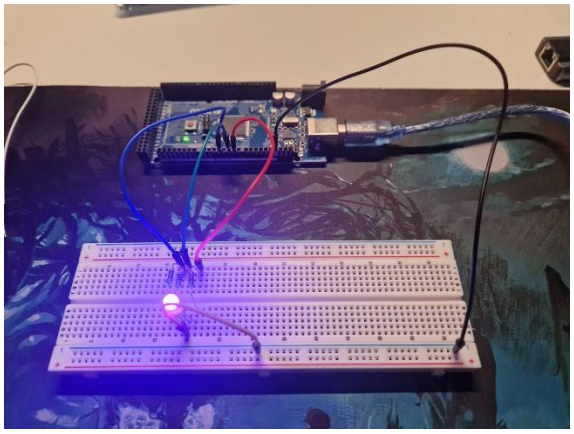
Bij de oefening is geen gebruik gemaakt van code.



RGB

Na te hebben gekeken naar een simpel LED-lampje, was het RGB-lampje aan de beurt. De lampje bevat eigenlijk drie lampje: een rode, een groene en een blauwe. Door een impuls tussen de 0 en 255 naar een lampje te sturen, kun je de intensiteit van die kleur bepalen; en daarmee de uiteindelijke kleur van de lamp.

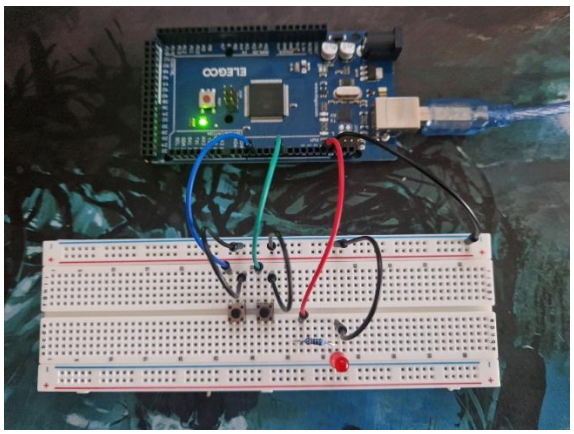
```
// Define the RGB LED pins
#define BLUE 3
#define GREEN 5
#define RED 6

// Setup, runs once
// Make the RGB LED pink
void setup() {
  pinMode(BLUE, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(RED, OUTPUT);

  analogWrite(RED, 255);
  analogWrite(GREEN, 0);
  analogWrite(BLUE, 255);
}

// Loop, runs over and over
// Nothing to update
void loop() { }
```



Switches

De volgende oefening is het aan- en uitzetten van een LED-lampje door middel van schakelaars. De Arduino kan lezen wanneer deze worden ingedrukt, waarna het lampje aan óf uit kan worden gezet.

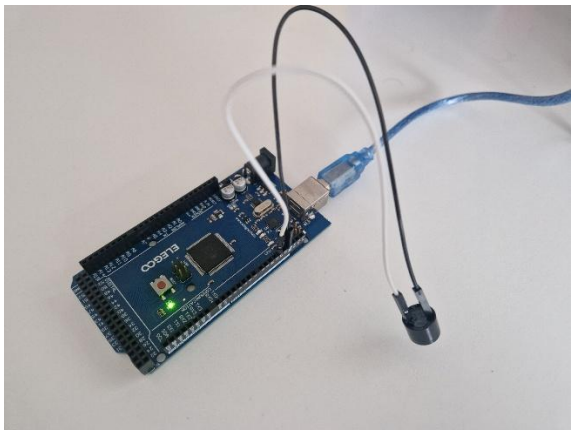
```
// Define the pins
#define SWITCHA 3
#define SWITCHB 7
#define LED 12

// Setup, runs once
// Set the switch pins as inputs
void setup() {
  pinMode(SWITCHA, INPUT_PULLUP);
  pinMode(SWITCHB, INPUT_PULLUP);
}

// Loop, runs over and over
// If A is pressed, turn on the LED
// If B is pressed, turn off the LED
void loop() {
  if (digitalRead(SWITCHA) == LOW) {
    digitalWrite(LED, HIGH);
  }

  if (digitalRead(SWITCHB) == LOW) {
    digitalWrite(LED, LOW);
  }
}
```



Actieve Zoemer

Een actieve zoemer is een zoemer met ingebouwde geluidsproductie hardware om frequenties aan geluid te produceren. Door een analoge waarde in te geven bij een pin, kan de hoogte van het geluid worden bepaald.

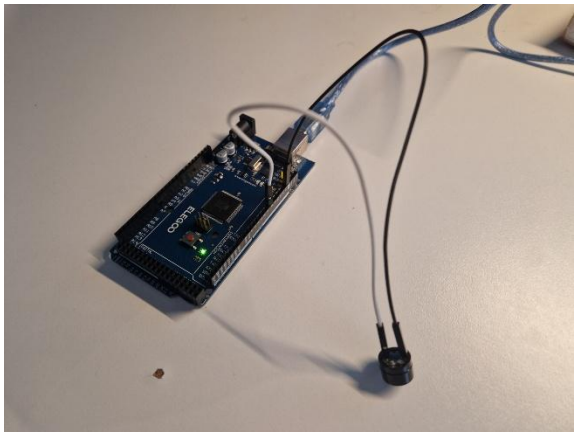
```
// Define the pin
#define BUZZ 12

// Setup, runs once
// Set the pin to a certain frequency
void setup() {
  pinMode(BUZZ, OUTPUT);
  analogWrite(BUZZ, 150);
}

// Loop, runs over and over
// Nothing to update
void loop() { }
```



Passieve Zoemer

De passieve zoemer geeft meer controle over de frequenties. Deze worden gedefinieerd in hertz. Om een noot af te laten spelen dient een andere functie te worden gebruikt dan voor de actieve zoemer.

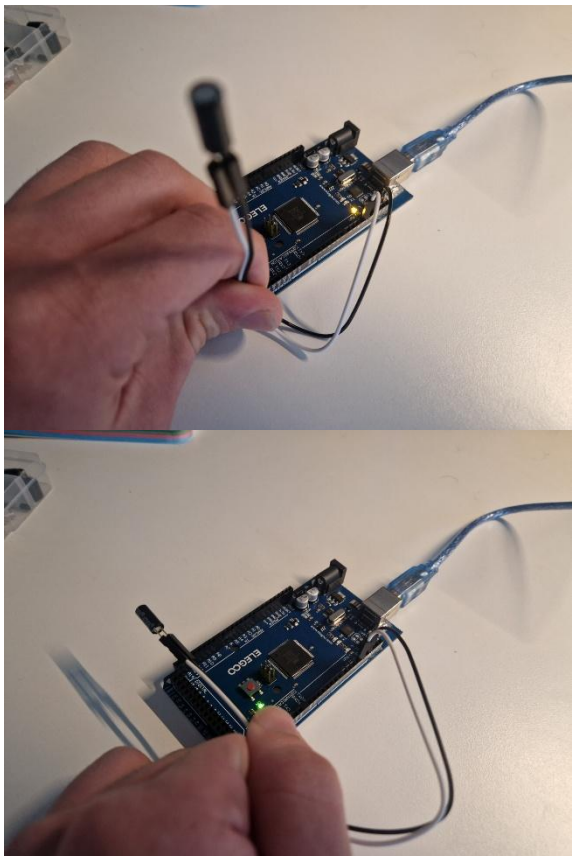
```
// Define the buzzer and import pitches
#include "pitches.h"
#define BUZZER 8

// Define the melody and note duration
int duration = 200;
int melody[] = {
  NOTE_A2, NOTE_A2,
  NOTE_A2, NOTE_D1,
  NOTE_D1, NOTE_A2,
  NOTE_A2, NOTE_D1
};

// Setup, runs once
// Nothing to setup
void setup() { }

// Loop, runs over and over
// Play the melody
void loop() {
  for (int note = 0; note < 8; note++) {
    tone(BUZZER, melody[note], duration);
    delay(200);
  }

  delay(500);
}
```



Balswitch

Een balswitch is een component dat een signaal geeft zolang deze rechtop staat. Op deze manier kan er ontdekt worden of er een hoek is of niet. Dit gebeurd door een balletje in de staaf die contact maakt als hij onderin de staaf zit.

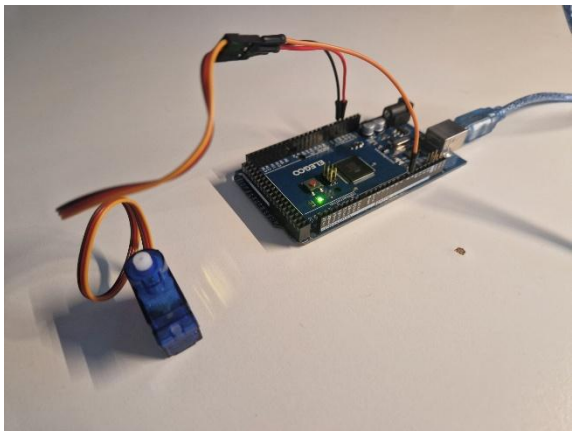
```
// Define the pins
#define LED 13
#define BALLSWITCH 12

// Setup, runs once
// Set the pins as in/output
// Turn on the ball switch
void setup() {
  pinMode(LED, OUTPUT);
  pinMode(BALLSWITCH, INPUT);
  digitalWrite(BALLSWITCH, HIGH);
}

// Setup, runs over and over
// Shine the LED if the ball switch is up
void loop() {
  if (digitalRead(BALLSWITCH) == LOW) {
    digitalWrite(LED, HIGH);
  }

  else {
    digitalWrite(LED, LOW);
  }
}
```



Servo

De servo is een aparaat dat 180 graden kan draaien afhankelijk van de waarde die wordt gebruikt. De servo bibliotheek wordt gebruikt om een servo aan te sturen.

```
// This needs the Servo library
#include <Servo.h>

// Define the servo
#define SERVO 9
Servo myservo;

// Setup, runs once
// Attach the servo to the pin
void setup() {
  myservo.attach(SERVO);
}

// Setup, runs over and over
// Move the servo around
void loop() {
  myservo.write(90);
  delay(500);
  myservo.write(30);
  delay(500);
}
```



Ultrasonische Afstandsmeter

Om een afstandsmeter te gebruiken wordt de HCSR04 bibliotheek gebruikt. Na het initialiseren wordt een loop gemaakt die via seriële output de afstand geeft.

```
// Requires HCSR04 library
#include <HCSR04.h>

// Define the ultrasonic pins
#define T 12
#define E 11
UltraSonicDistanceSensor sensor(T, E);
long a;

// Setup, runs once
// Start the serial monitor
void setup() {
  Serial.begin(9600);
}

// Loop, runs over and over
// Get the distance and print it
void loop() {
  a = sensor.measureDistanceCm();
  Serial.print(a);
  Serial.println(" cm");
  delay(1000);
}
```