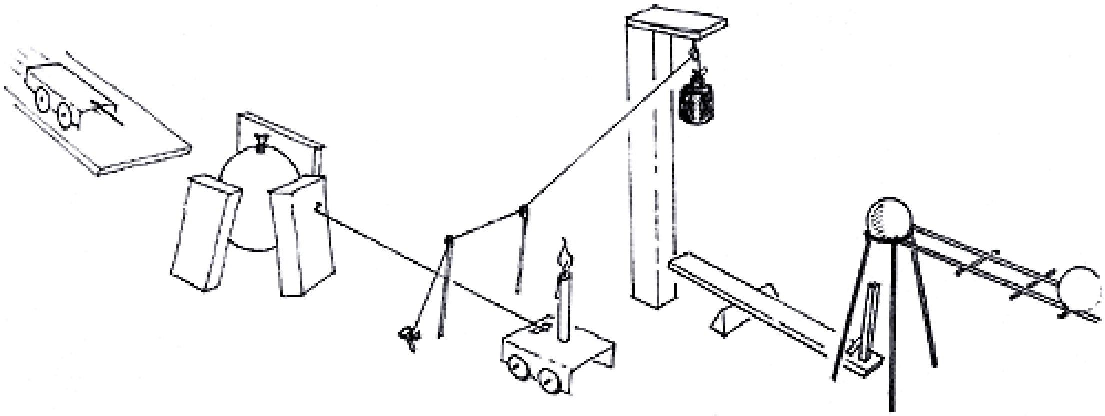
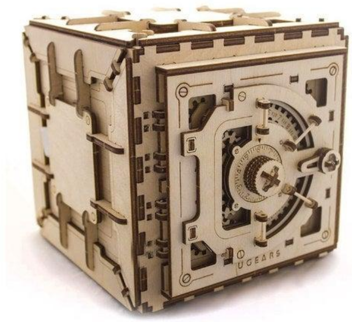
Incredible Machines

Design Document (2024-2025)

Name: Emanuel de Jong

Class: EXM0V.CDTb



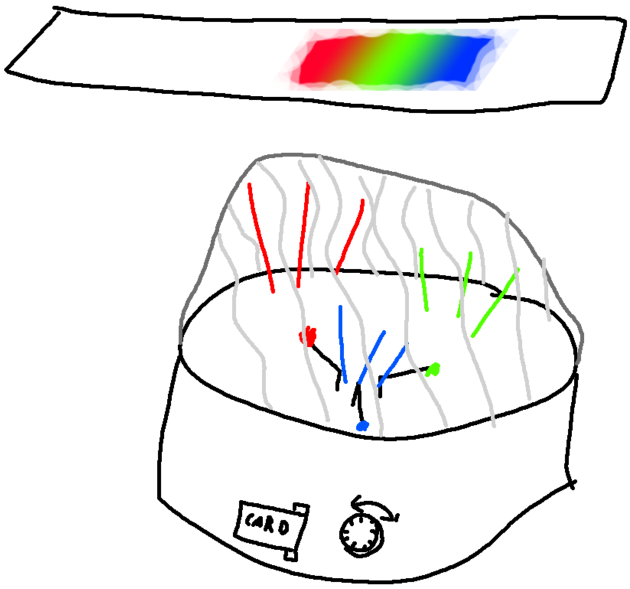


Content:

1 Concept descriptions

1.1 Concepts

Concept A: Aurora projector
A mostly round box with a patterned glass dome or plate on top. A red, green and blue light with a motor for each turning them up to 120 degrees horizontally to create a changing light pattern on any ceiling and partly wall. Putting a card in a card slot starts the machine. There is also a knob to change the speed of the motors.

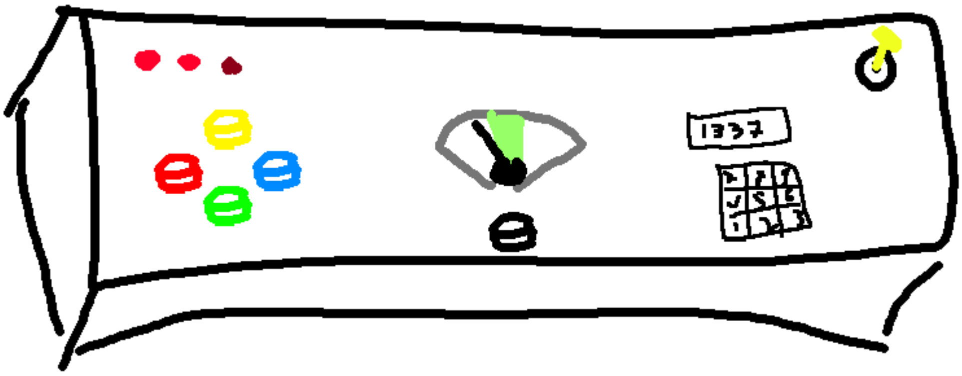


Concept B: Escape box
A rectangular box with challenges. Turning a key in a keyhole turns on the game. Three lights that start turned on indicate lives. When you fail the challenge you lose a live. When you lose all lives you failed the game. When you solved all challenges you win the game. A sound indicates when you won or lost depending on the sound.

The first challenge is Simon says with 3 colour buttons with lights in them. It starts with 1 light increasing the sequence by 1 up to a sequence of 5.

The second challenge is a meter turning left and right quickly. When pressing a button, the meter stops, if it stops in a certain range, you passed. The range is indicated by a colour strip.

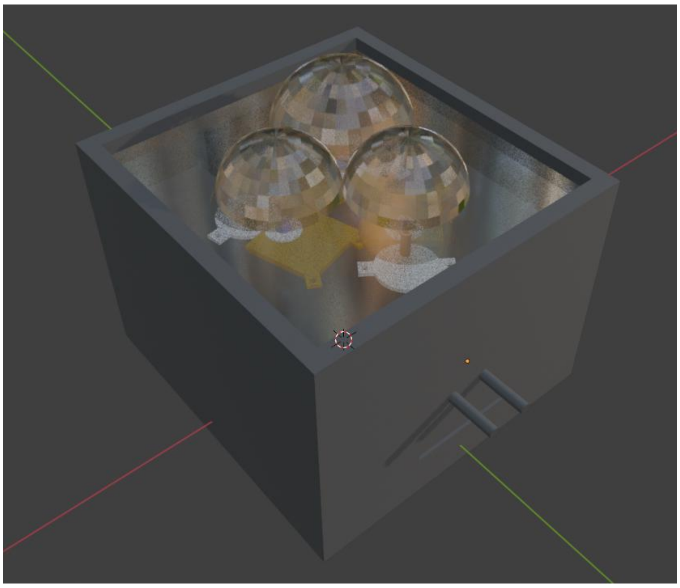
The third challenge is writing a 4 digit code from morse deeps that start after finishing the first challenge. The morse codes for digits is in the manual. When 4 digits are given and it's wrong, you lose a life and the digit display clears.

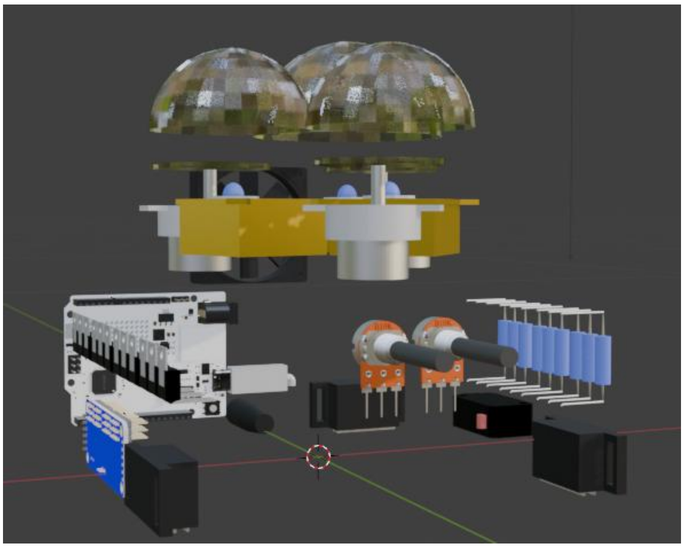


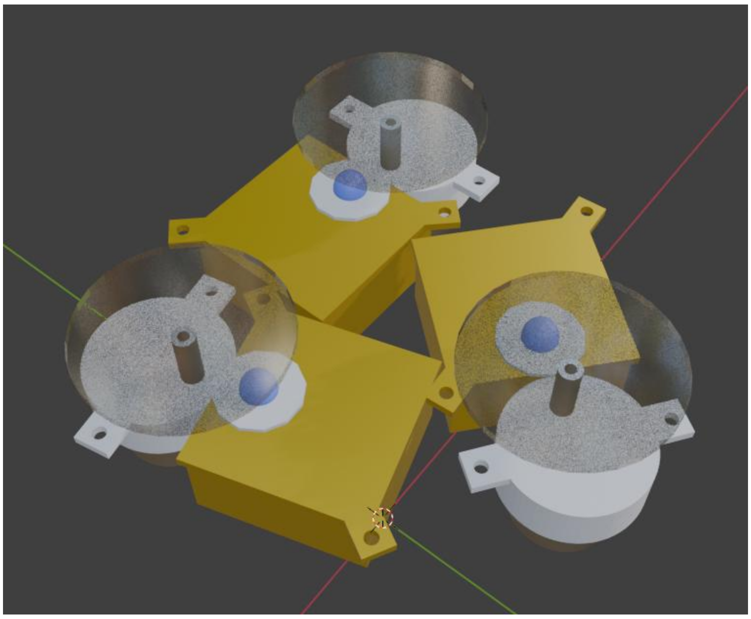
Concept C: Portable fan
A small fan for hot summers. Can be folded to bring with you. Turns on by blowing into a hole. With a button you can toggle between 3 speed settings. 3 lights indicate which speed it's on.

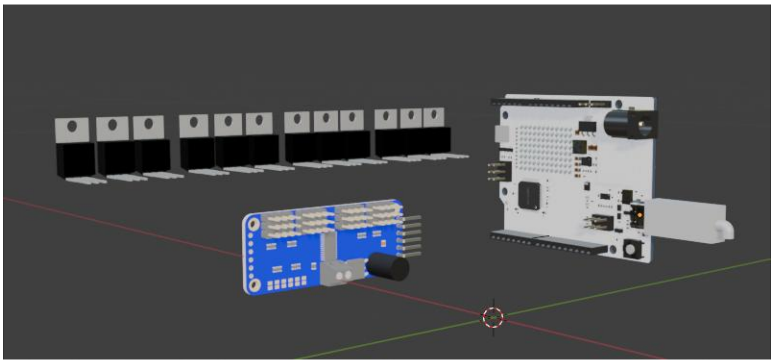
Conclusion
I chose concept A because I'm most interested in it and it's the only concept I will use after the project. It's also simple enough to be manageable in the time of the project, at least I thought so.

1.2 Your Incredible Machine









Because a rectangular shape for the box is more practical, e.g. to fasten it to the wall, I choose not to use a cylindrical shape. The outside vertical walls will be black 3d printed plastic. The bottom is wood with a hole big enough for any finger to open it up for maintenance. The Arduino is held in a small plastic container screwed to the side. Above the Arduino there is a 3D printed plate for the LED's and the rotating lenses. It has a hole in the centre for the cables. The glass domes, not only one, but I choose to use three of them for advanced effects, rests on a transparent horizontal plate as a third level on which the different secondary lenses can be put. As a fourth and final level there is a transparent plate on top of the box.

2 Morphological Chart

Materials

Goal	Option 1	Option 2	Option 3	Reason
Lens	Glass	Plexiglass	Polycarbonate	Only patterned lenses from other sky projectors are good enough as they are made for caustics. The sky projectors I have use polycarbonate lenses.
Casing	Wood	PLA	Metal	As I'd like to use the product after the project, I would like a professional look. Wood feels too DIY. Metal is professional but is hard to work with. With PLA I can easily 3d print it.
Bottom	Wood	PLA	Metal	A thin circular plate is easy to make with normal cutting. Wood has the benefit of being cheap and adaptable for fixing things on it, also it is easy to alter the concept.

Hardware and software

Goal	Option 1	Option 2	Option 3	Reason
Microcontroller	Arduino	ESP	Raspberry Pi	The casing is big enough to fit an Arduino and the tasks are simple enough so that a raspberry would be overkill. It's also what I already have and am used to.
Language	Arduino programming language	Visual	Maschinecode	Gives more freedom and complexity is not a problem for me as software

				engineering is my major.
IDE	Arduino IDE 1	Arduino IDE 2	VS Code	I've had bad experiences with both versions of the official IDE. VS Code can do everything they can and more.

3 Sensors and Actuators

Sensors

Goal	Option 1	Option 2	Option 3	Option 4
Card detection	 Microswitch	 IR proximity sensor	Ultrasonic module	Laser interruption module
First lens Rotation	 DC motor	 Stepper motor	 Servo motor	
Light rotation speed control + Colour changing speed	 Slider	 Potentiometer	 Button (toggle values)	
Light	 Standard LED	 RGB LED	Halogen lamp	Light bulb

Reasons for selection:

Card detection
While an IR sensor is a logical and viable option, nothing beats the reliability and ease of a microswitch being pressed down by the side of the card. It also doesn't suffer from environmental issues like light or sound which are both present inside the product.

First lens rotation
The stepper motor is expensive and needs a driver module , we don't need it's strength either. Cheap versions are also jittery making the light effect less satisfying. The servo motor I received had only a quarter radius forward and back modus, while I need a continuous circular movement. A DC motor is perfect for this and its speed is easily regulated by a potentiometer.

Light rotation speed control + Colour changing speed
A button would be restrictive as it jumps between values. It's also more tedious to go from a middle state to the lowest state having to press multiple times. Although a slider would work, a potentiometer is the more usual regulator in commercial products.

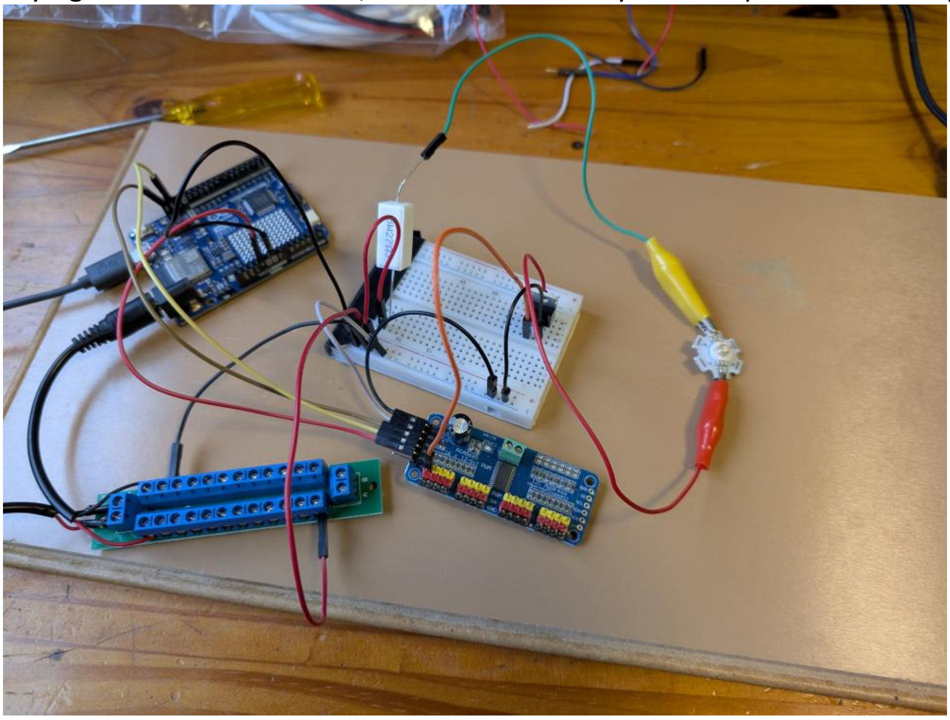
Light
An RGB LED gives more creative freedom for cool effects so I'll choose it. However, single colour LEDs can get brighter without outside power than RGB LEDs . unfortunately You would need more space for them and the light source would be less concentrated on one point, so the projection could be less fluent.

4 Tests of (small) prototypes (proof of concept)

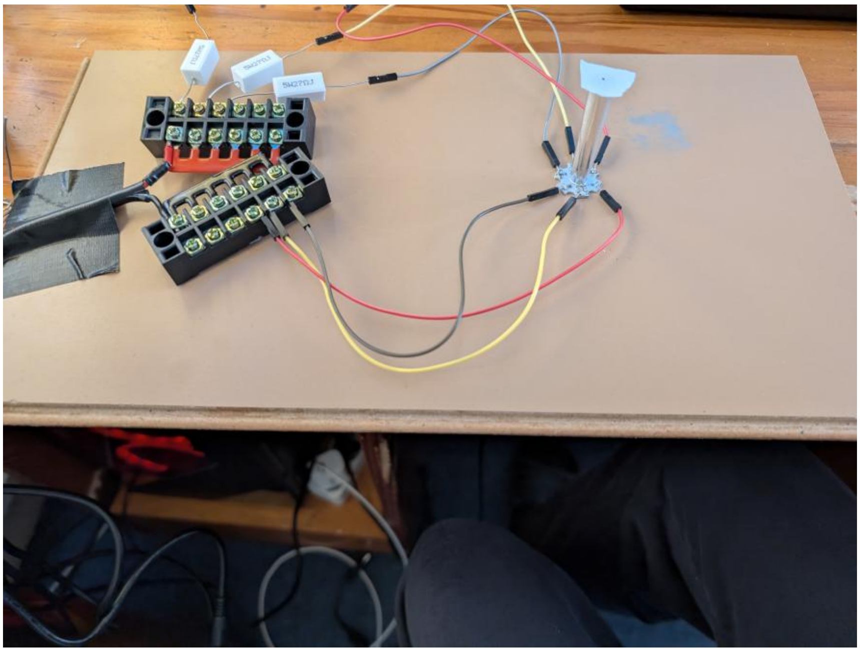
Of course there a many stages, so I only describe the main stages here.
The reason to move to a next stage is mostly extension to the final product.
Changes and their reason are written between the lines.

Preparation
Many, many components

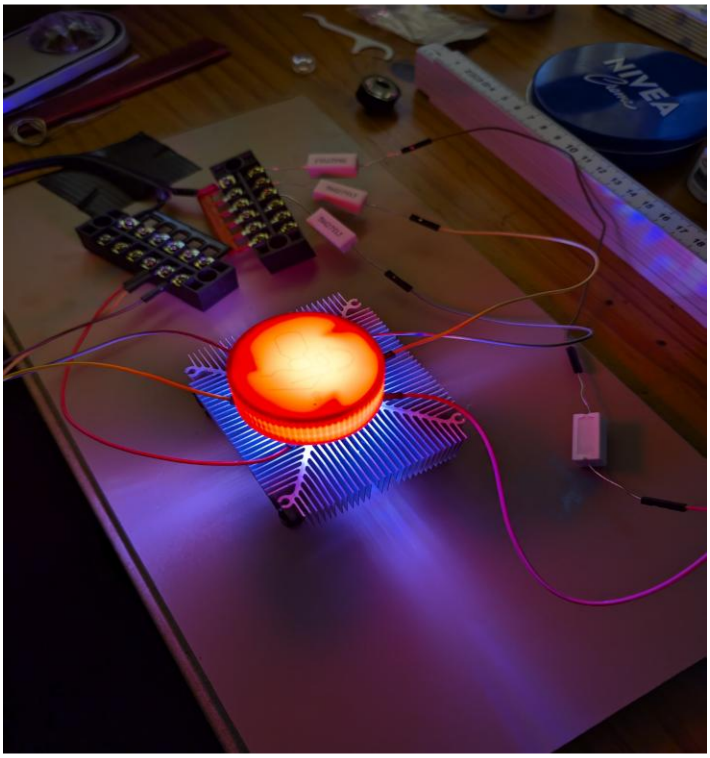
First Prototype
Trying one color of one LED, to test the intensity and the premature setup.



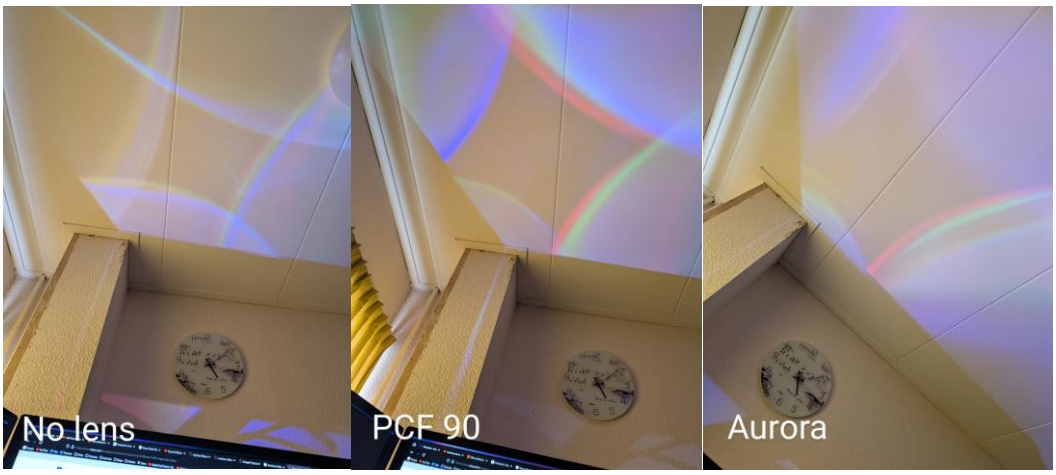
Second prototype
Now all colors of one led. Notice the power resistors (5W). They were used because the light part was initially based on 12V. Later on I changed it to 5V base and therefore 3W resistors were strong enough to use, although they also get a bit hot, so I'm thinking about using a heatsink in the final prototype.



Third prototype
The LED is getting rather hot so it needed a heatsink. An old heatsink from a bridge off an motherboard will do.



Testing the effect of angle altering lenses:



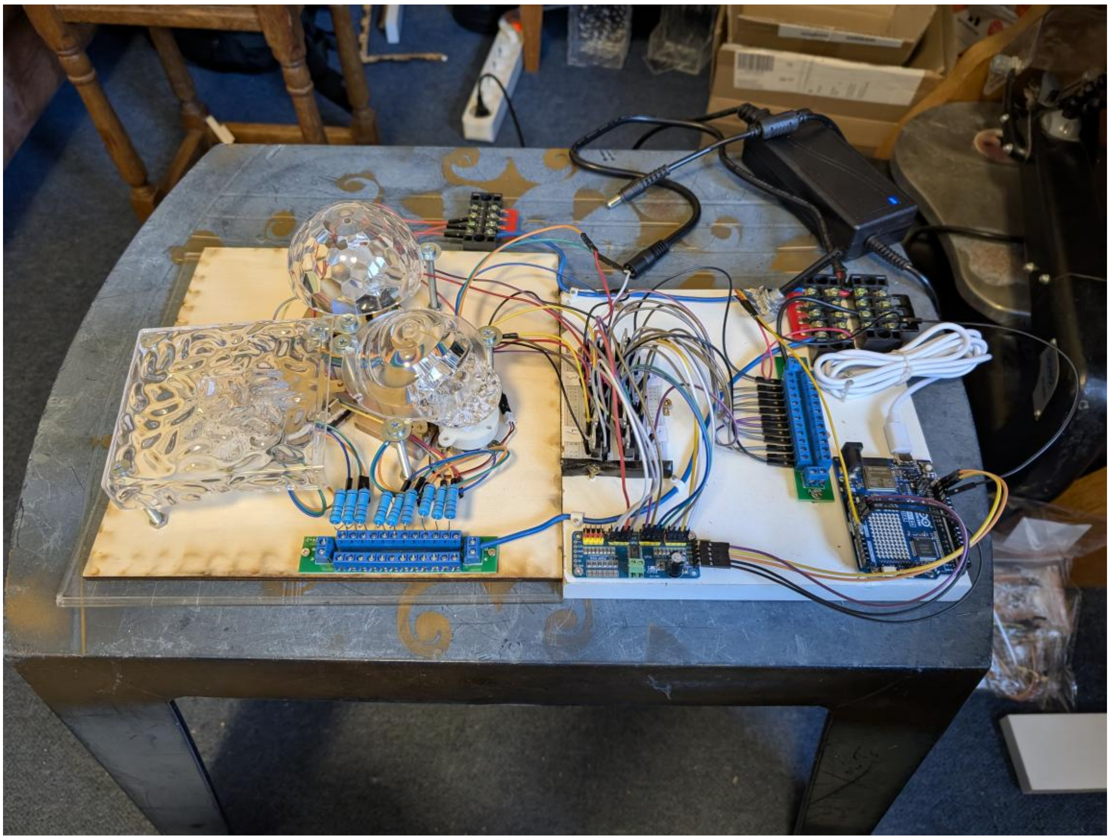
Fourth prototype
Now the Arduino is basically programmed and connected.
This and the next prototypes have no pictures as they went fast due to the experiences I gained from the previous ones.

Fifth prototype
Now rotation comes into play and I connect the dc-motors to irregular lenses that are to be rotated above the individual LEDS's. For regulating their speed I need an RPM module, because the rpm possibilities on the Arduino are less then I need. Additional there are MOSFETs needed to let the whole thing function.

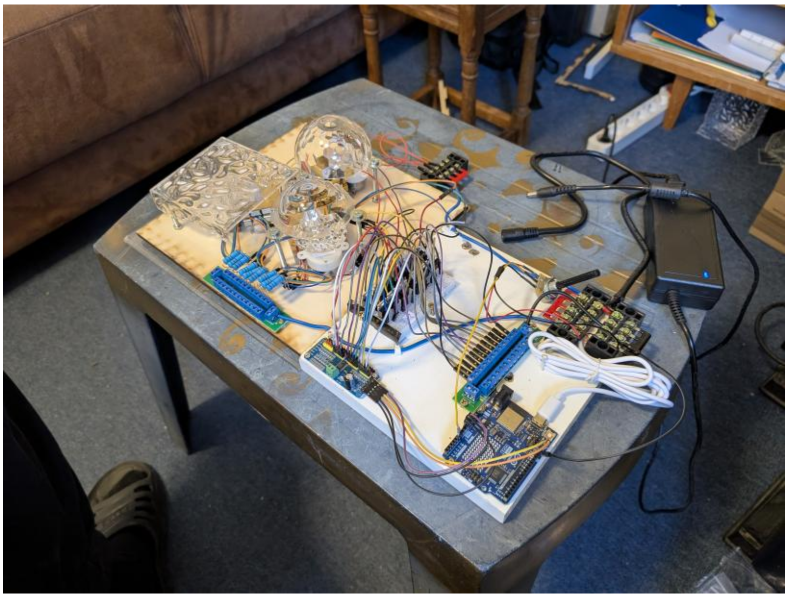
Sixth prototype
Introducing the light effects is the next stage. There are also MOSFETs and (of course other parts of) the rpm driver needed.

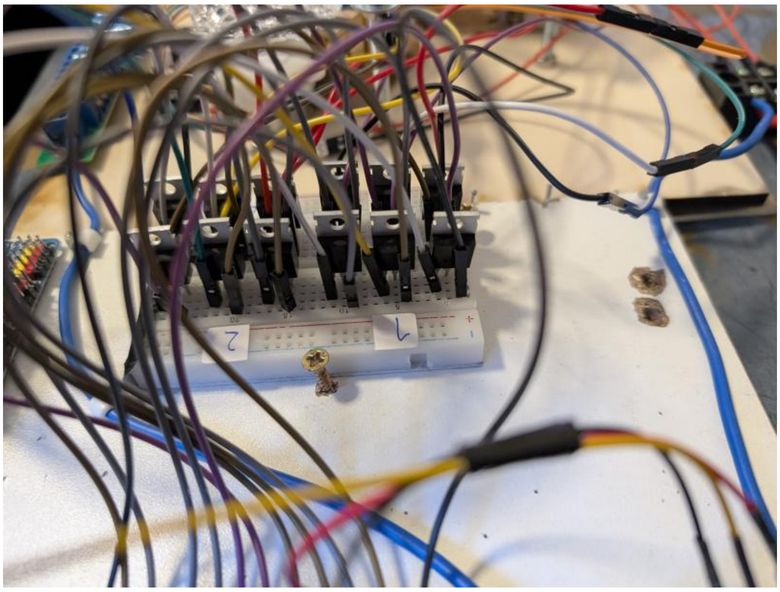
Seventh prototype
Now the combination from in different speed rotating lenses and the speed of light change in combination with colour change is the item. This is a really long path to find the right combinations.

Eight prototype
The experiments with secondary lenses start. The setup is now fully crowdy and it is a lot of fun the experience the possibilities.

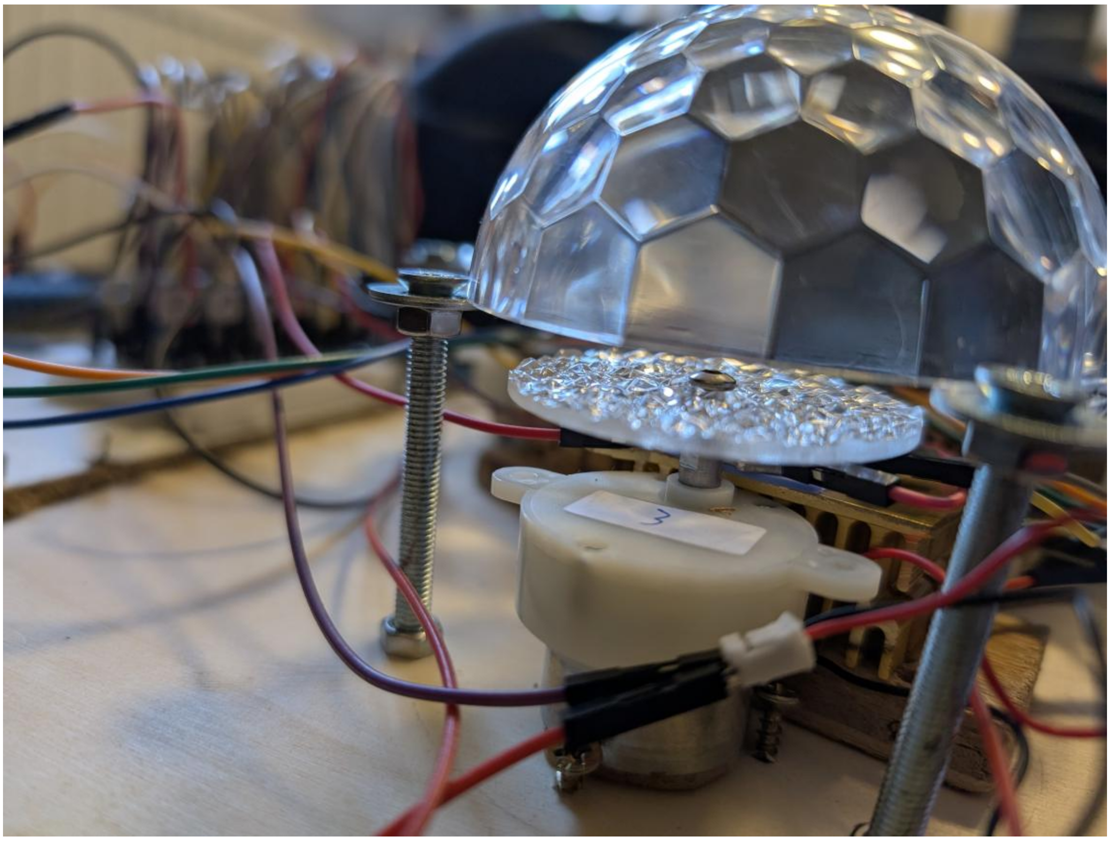


The secondary lenses are flat and/or domes. From right to left you see the Arduino with above it connections. Further left there is the rpm driver at the bottom and the Moffett's behind it. And on the left platform there are the 3 secondary lenses and the resistors, used to give the LED's the right voltage.





The MOSFETs.

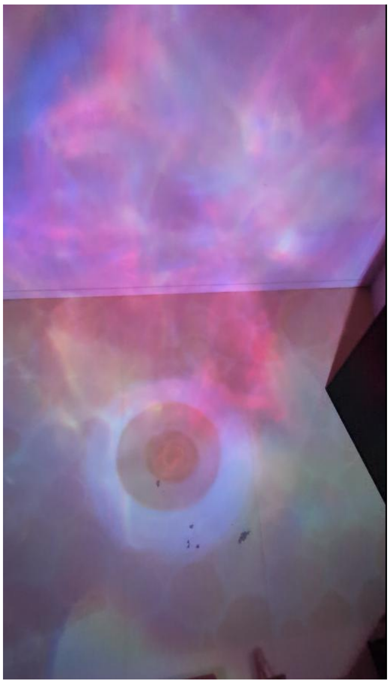


DC motor with turning primary lens and above it the secondary lens, in this case a dome. In the background you can see the cooler on which the LED is placed.

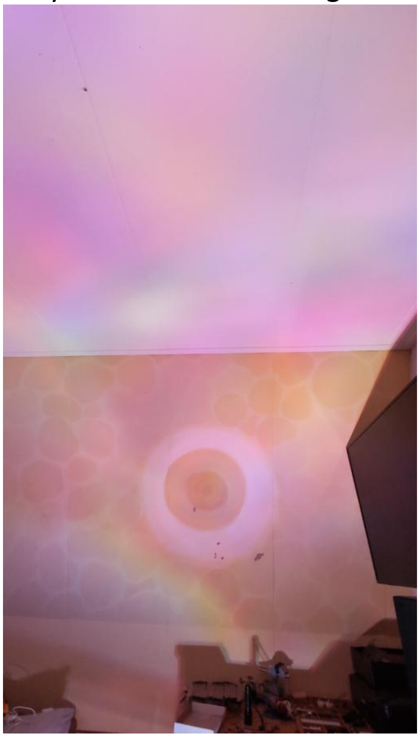
3 directional lenses

3 global/circular lenses

A shared lens



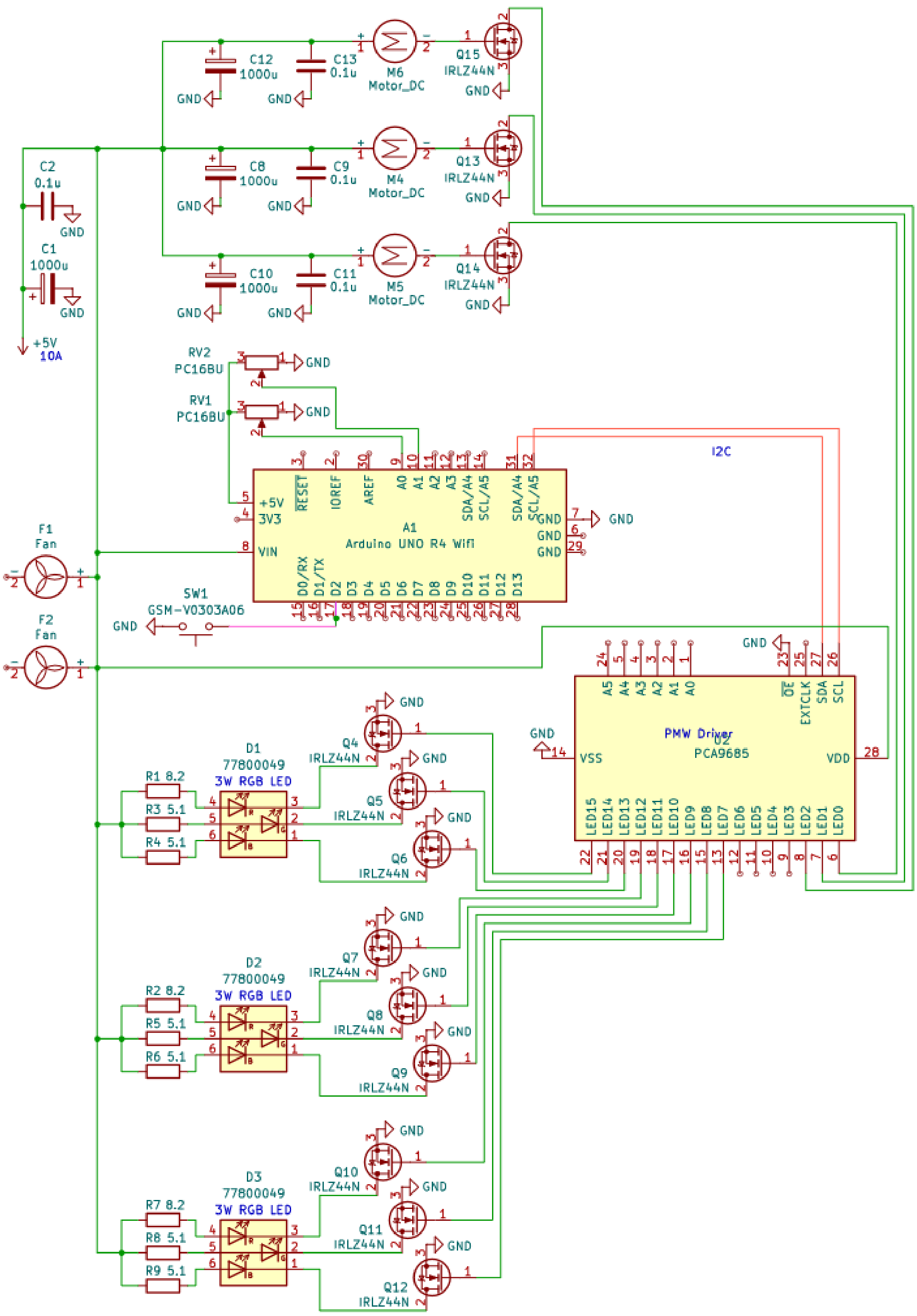
Only the disc lenses for light distribution



Way to final product

The whole construction is to be made fit into a 3Dprinted box with wooden bottom on which the electronics should find their place. Then a second layer in it with LED's, motors and primary lenses. Above that a transparent plate on which the diverse secondary lenses will be placed. And finally on top another transparent plate. This part is still to be finally built.

5 Connection schematics



The projector is powered by a 5V 10A power adapter, though A 6A one should suffice.

The electrolytic and ceramic capacitors are for stability and filtering high frequency noise. There is a pair at the entrance and at each motor as you don’t want trailing

motors and they can stall which might draw a high amount of current for a short period.
The ground pin of each motor goes to it's own MOSFET and connects to its drain. The MOSFETs source is connected to ground and the gate goes to a pin on the PWM driver so the Arduino can alter the motors speed.

Each colour channel of each RGB LED works in the same way with the addition of a 3W resistor (8.2 ohm for red and 5.1 for green/blue) to lower the voltage.

The Arduino has 2 potmeters connected like usual. The microswitch uses the Arduino's built in pullup for noise reduction. The Arduino connects to the PWM driver using I2C. The Arduino's ground is connected to common ground to equalize for safety.

Lastly there are 2 fans. One blows air through the whole projector, the other is directed to the resistors as they generate a high amount of heat.

6 3D printing and laser cutting

The walls of the box and the floor separating the main electronics and the LEDS/motors/lenses are 3D printed.
The two transparent plates and the top are laser cut.

7 Used software

I used the Arduino programming language, because the microprocessor used is an Arduino. The software does its work as expected.

Each colour of each LED fades at different speeds. Each DC motor has its own speed. The microswitch to turn the product on and the potmeters for the movement speed and fade speed are still to be added.

Code snippets

```
#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwmDriver = Adafruit_PWMServoDriver();

const int LED_MIN_PWM = 200;
const int LED_MAX_PWM = 4095;
const int LED_BASE_STEPS = 6;
const int LED_SPEED_VARIANCE_RANGE = 1;
int LED_BASE_SPEEDS[] = {10, 12, 14};

int ledPwms[3][3];
int ledSpeeds[3][3];
```

I used a library to easily control the PWM driver without having to worry about I2C connections and such. All I have to do is use the functions of pwmDriver.

The capitalized variables are configuration parameters to easily tweak some behaviours.

```
void setup() {
  randomSeed(analogRead(A5));

  pwmDriver.begin();
  pwmDriver.setPWMFreq(500);

  // Motors from slow to fast
  pwmDriver.setPWM(0, 0, 4095); // First motor is slow of itself
  pwmDriver.setPWM(1, 0, 3000);
  pwmDriver.setPWM(2, 0, 4095);

  // Starting led pwms
  for (int led = 0; led < 3; led++) {
    for (int color = 0; color < 3; color++) {
      ledPwms[led][color] = random(LED_MIN_PWM, LED_MAX_PWM);
    }
  }

  // Create ledSpeedVariances
  // For example: with LED_SPEED_VARIANCE_RANGE=3 it will be -3, -2, -1, 0, 1, 2, 3
  int ledSpeedVariancesLen = LED_SPEED_VARIANCE_RANGE * 2 + 1;
  int ledSpeedVariances[ledSpeedVariancesLen];
  for (int i = 0; i < ledSpeedVariancesLen; i++) {
    ledSpeedVariances[i] = i - LED_SPEED_VARIANCE_RANGE;
  }

  // Led speeds
  for (int led = 0; led < 3; led++) {
    shuffleArray(ledSpeedVariances, ledSpeedVariancesLen);
    for (int color = 0; color < 3; color++) {
      ledSpeeds[led][color] = LED_BASE_SPEEDS[led] + ledSpeedVariances[color];
    }
  }
}
```

The DC motors are set at different speeds. This can be done in the setup as I won't be changing them over time for now.

The rest is randomly choosing starting frequencies and speed modifiers for the LEDs.

```
void loop() {
  for (int led = 0; led < 3; led++) {
    for (int color = 0; color < 3; color++) {
      int pwm = calcLedPwm(led, color);
      if (pwm > LED_MAX_PWM || pwm < LED_MIN_PWM) {
        ledSpeeds[led][color] *= -1;
        pwm = calcLedPwm(led, color);
      }

      // The + 3 is because the first three are the motors
      int ledDriverIndex = led * 3 + color + 3;
      pwmDriver.setPWM(ledDriverIndex, 0, pwm);

      ledPwms[led][color] = pwm;
    }
  }

  delay(50);
}
```

Every 50 milliseconds, the PWMs of the LEDs is changed.

8 Final result

As mentioned earlier, the final build is still being worked on.

9 Reflection

The Arduino Shield was not useful for this setup, so nothing to tell about that.
The microswitch I used is a very simple example of a place/situation sensor. The potentiometer is a useful sensor for setting your preferences to influence the outcome of the effects.

During the programming I was confronted for the first time with a PWM driver. That this PWM module is also useful for the light part was a surprise for me.

I learned a lot about resistors and how to calculate current and voltages so the LED becomes the right Voltage. I also learned this creates a lot of heat.

Further on I was studying the world of relays, switches, transistors and MOSFETs, quite a bit of electronics for a software student!

The soldering of the LEDs, the wiring of the MOSFETs and getting the lenses from other projectors took a lot longer than I anticipated. This made it hard since I needed a fully working prototype to test out effect so that I could decide on one, so that I could think of how the final product will look so that I could finish this document… In future products I think I will choose something with less connections.