04. Technical documentation (2023-2024-S015-S023)

Circuit in Tinkercad and its schema

The circuit was designed in Tinkercad, and for the solution of the problem Arduino Nano and continuous Servo motor were used. In addition, a cable had to be added to make the connection to the computer possible (USB to Mini USB).

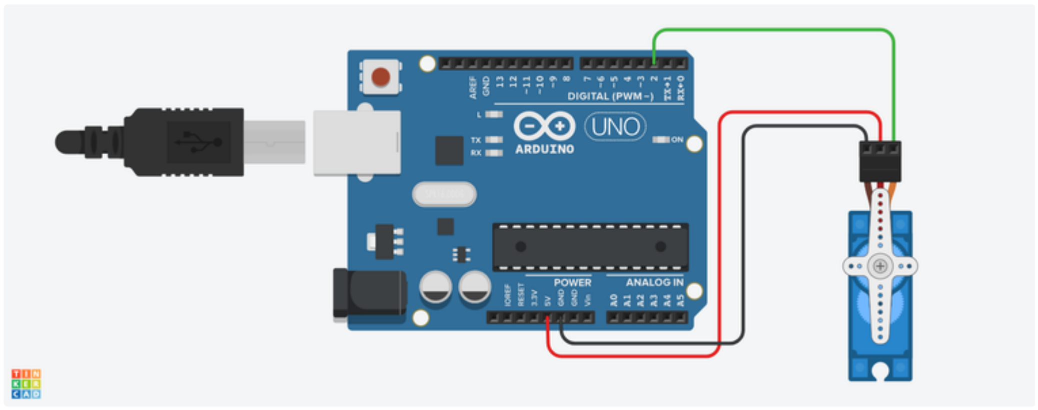

Figure 1. The circuit designed in Tinkercad

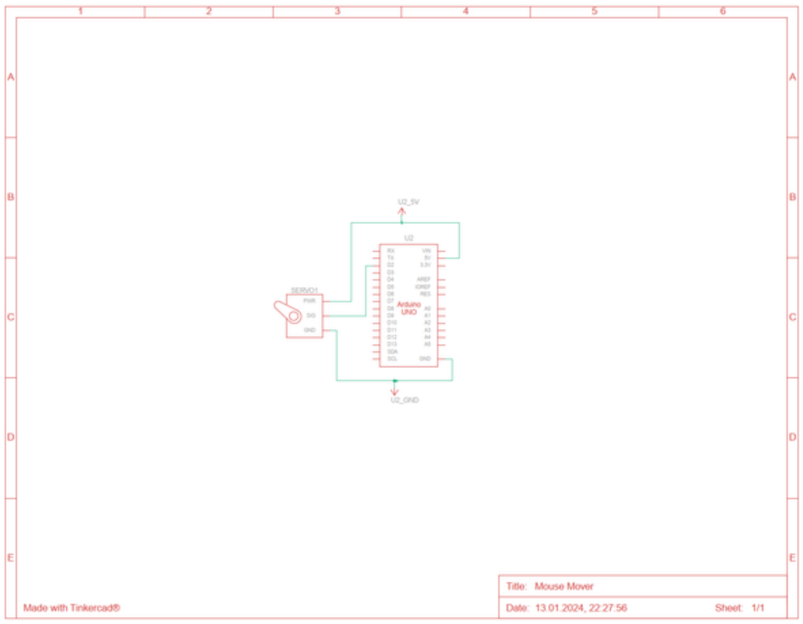

Figure 2. Circuit's schema, designed in Tinkercad

Requirements

- **Hardware**
  - Arduino Nano (where to buy)
    - price ± **8.20€**
  - Continuous rotation servo - we used the FS90R servo (where to buy)
    - price ± **3.10€**
  - Wheel or arm for the servo (this comes in some of the FS90R packages)

        - price **0€**
    - Jumper wires, cables male to female 10-20cm (where to buy)
        - price ± **2.40€**
    - Optical mouse - corded or uncorded - we used our own (where to buy)
        - price ± **2.60€**
- **Tools**
    - Suitable sized box - we used 3 mm thick raw HDF board (where to buy)
        - price ± **6,39 € / m2**
    - Small hand saw - we used our own (where to buy)
        - price **0€**
    - Glue gun - we used our own
        - price **0€**
    - Double sided tape
        - price **0€**
- **Software**
    - Arduino IDE environment (for using Arduino Nano)

In total, **the project ± would cost 32,69€** if we consider with reserve (± 10,0€) that someone does not have for example a glue gun or tape.

Code

The code for the motion simulation was written in C++. The code essentially creates a simple random servo motor movement, making the servo move to a random angle with a random speed, pause for a random duration, and then return to the starting position. The randomness is introduced using the `random()` function, and the Servo library is used to control the servo motor.

Here's a brief overview of what the code does:

1. It includes the Servo library, which provides functions to control servo motors.
2. It declares a Servo object named "myservo" and an integer variable "pos" to store the current position of the servo.
3. It initializes a long variable "randomAngle" and "randomSpeed" to store randomly generated values.
4. In the setup function, it attaches the servo to pin 2 and initializes the random seed using the analog reading from pin A0.
5. In the loop function, it generates random values for the angle and speed of the servo movement.
6. It then moves the servo from 0 to the randomly generated angle with a delay corresponding to the randomly generated speed.
7. After reaching the random angle, it introduces a delay and then moves the servo back to 0 position with a similar delay.

```
#include <Servo.h>

Servo myservo;  // A Servo object named "myservo" to control the servo motor
int pos = 0;    // Variable to store the current position of the servo

long randomAngle;  // Variable to store randomly generated angle
long randomSpeed;  // Variable to store randomly generated speed

void setup() {
  myservo.attach(2);  // Attach the servo to pin 2
  randomSeed(analogRead(A0));  // Initialize the random seed using analog reading from pin A0
}

void loop() {
  randomAngle = random(0, 181);  // Generate a random angle between 0 and 180 degrees
  randomSpeed = random(2, 20);   // Generate a random speed between 2 and 19 (milliseconds)

  // Move the servo from 0 to the randomly generated angle
  for (pos = 0; pos <= randomAngle; pos += 1) {
    myservo.write(pos);  // Set the servo position
    delay(randomSpeed);   // Introduce a delay based on the randomly generated speed
  }

  delay(random(500, 2000));  // Introduce a random pause between movements (500 to 2000 milliseconds)

  // Move the servo back from the randomly generated angle to 0
  for (pos = randomAngle; pos >= 0; pos -= 1) {
    myservo.write(pos);  // Set the servo position
    delay(randomSpeed);   // Introduce a delay based on the randomly generated speed
  }
}

```