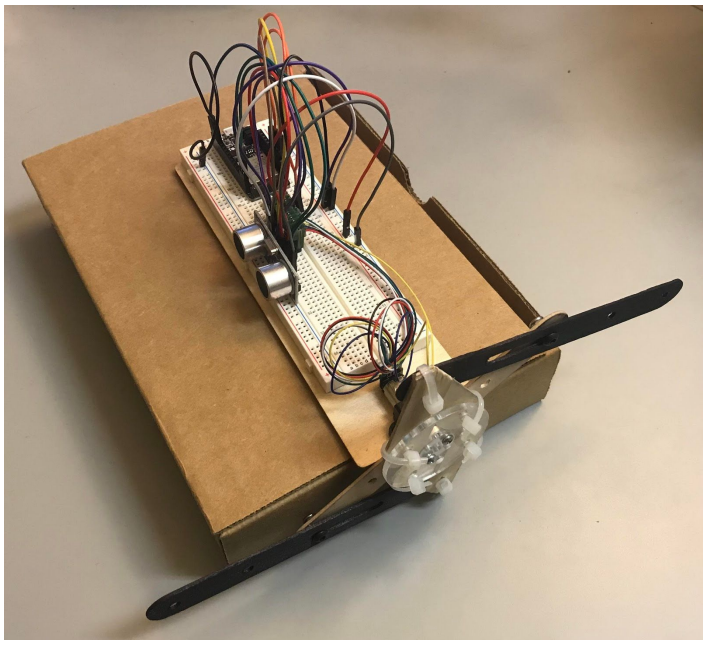
Automatic Paper Rotor

Licheng Yu

Project Description

In our daily life, people often need toilet paper to wipe their wet hands. However, using a wet hand to grab toilet paper directly usually damps some extra paper. Over time, those wet paper will dry and become wrinkly, which affects paper quality and comfort while using. From the experience, we know that this wet process usually happens when we try to rotate the roll. To avoid wet hands directly getting in touch with those extra paper, I designed the automatic paper rotor. This project is a device that automatically rotates the toilet paper roll when it detects hands are nearby. Due to the lack of material and machines, I'm not able to make a full-size prototype, however, it could still show the ideas and principle of the device with complete data collecting and analyzing process.



From the picture, we could see that this device is absolutely too big to fit into the toilet paper roll. And also, the distance sensor is facing the side which should direct along the axial to detect hands on the side of the roll. But that's the best I can get with the limited material I have at home. And hopefully, it clearly shows the idea of the project.

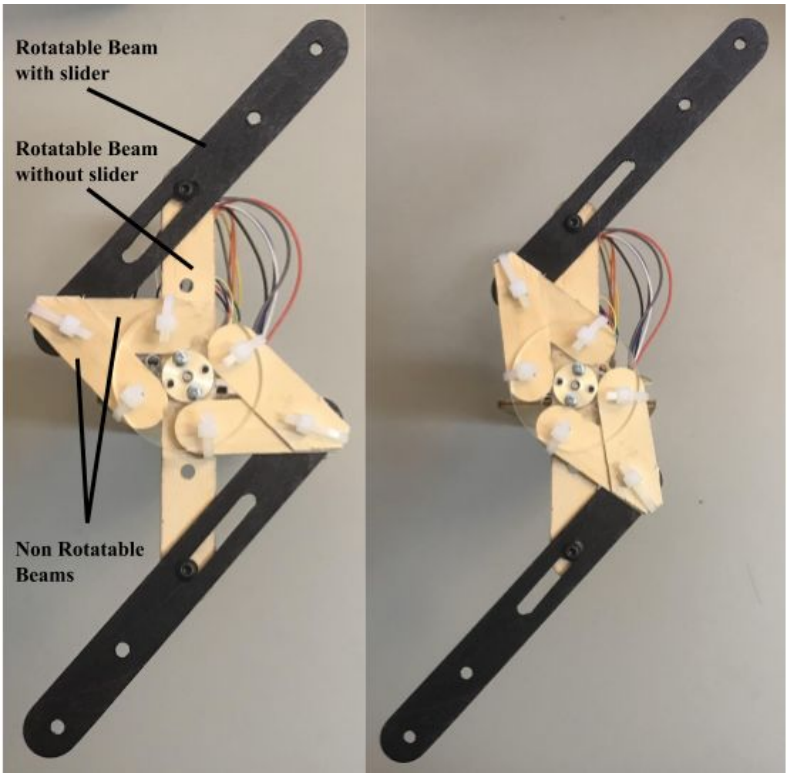
Working Principle

When the distance sensor is unactivated, the rotor is under an unlock mode which allows people to freely rotate the roll in case that they don't want to wait for automatically rotating under some emergency situation. When the sensor detects hands are within 5cm, the device is activated and two lock beams are

moved out against the inner side of the roll to lock it with friction force. Then the motor will rotate a certain angle of time to release a certain amount of toilet paper. Finally, the lock beams are withdrawn and the device goes back to the unlock mode.

Mechanical Part

The mechanical part in this device is mainly used to rotate and fix toilet paper roll. When the device is activated, the motor first finishes a 0.05s locking rotation which drives the mechanism and moves the lock beams outward as shown in the figures below. The mechanism is two simple four beams systems. Each one consists of a disk fixed to the motor shaft, two short non rotatable beams and two rotatable beams with a slider on one of them. With the Gruebler's equation, the degree of freedom of it should be one which agrees with facts.

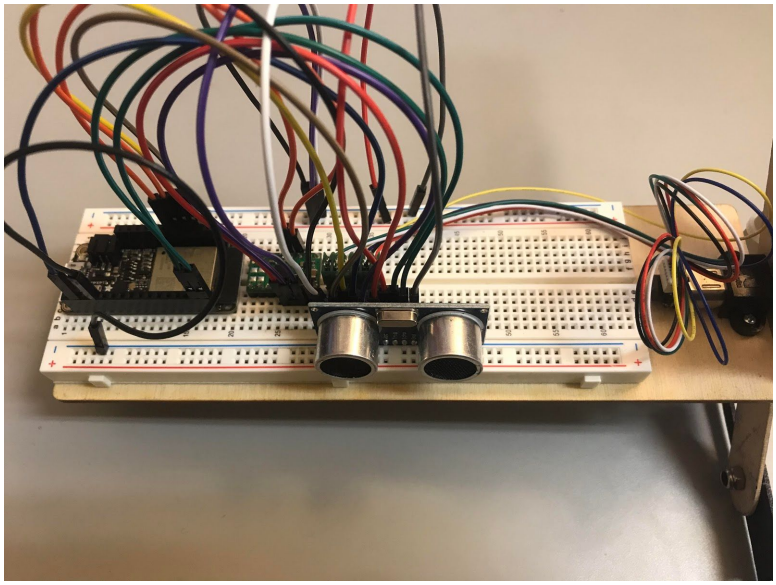


Shown upper are the figures of the mechanism's unlock state and locking state. The problem I found during tests is that the rotatable beam without a slider should not rotate in the first 0.05s locking rotation so that the black lock beams could slide out. However, due to the friction, it rotates with the mechanism. And the same problem happens when the motor tries to withdraw the black lock beams. In conclusion, this prototype doesn't have the ability to release and withdraw the lock beams, to achieve this feature, we need smoother material or a bearing.

Electronic Part

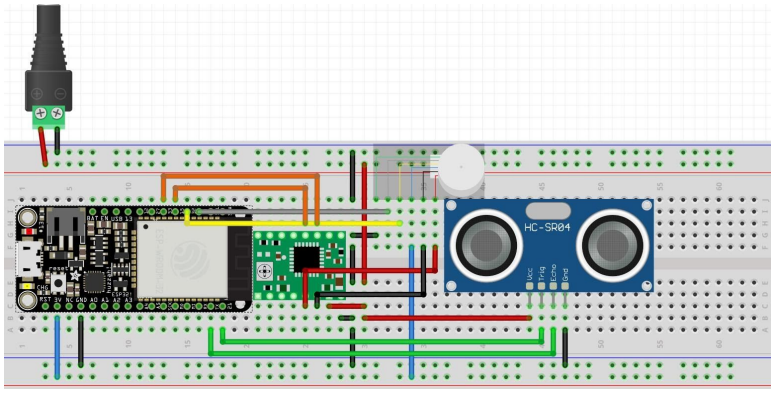
The electronic part in this device mainly consists of a motor, an Arduino ESP32, an encoder and a HC-SR04 ultrasonic distance sensor as listed below. The first three elements have been fully introduced and practiced on using during the lectures and the last is a new element which is introduced below:

- HC-SR04 ultrasonic distance sensor: This is a distance sensor that provides a 2cm - 400cm distance measure range and the accuracy is up to 3mm. It consists of an ultrasonic transmitter, a receiver and a control circuit. Distance is calculated by the time difference between transmission and receive and the speed of the ultrasonic. This calculation should be applied in the sketch.



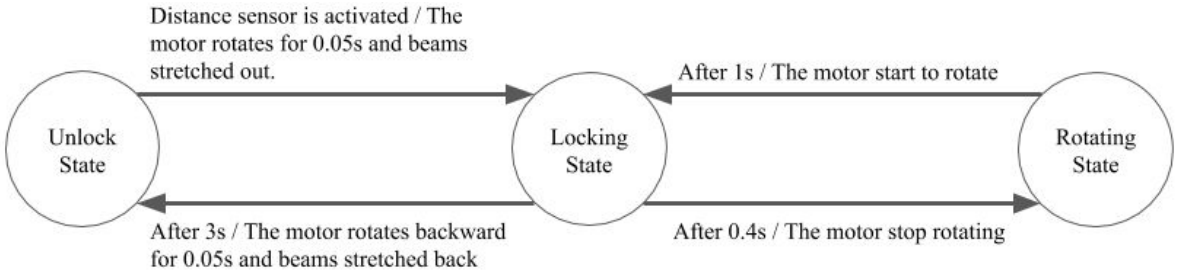
The distance sensor is used to send the activation signal to the arduino and its trigger threshold is set to 5cm. Then the arduino sends a motor signal to the encoder which controls the motor. This is a very simple motor control system just like what we learned in the lecture.

Circuit



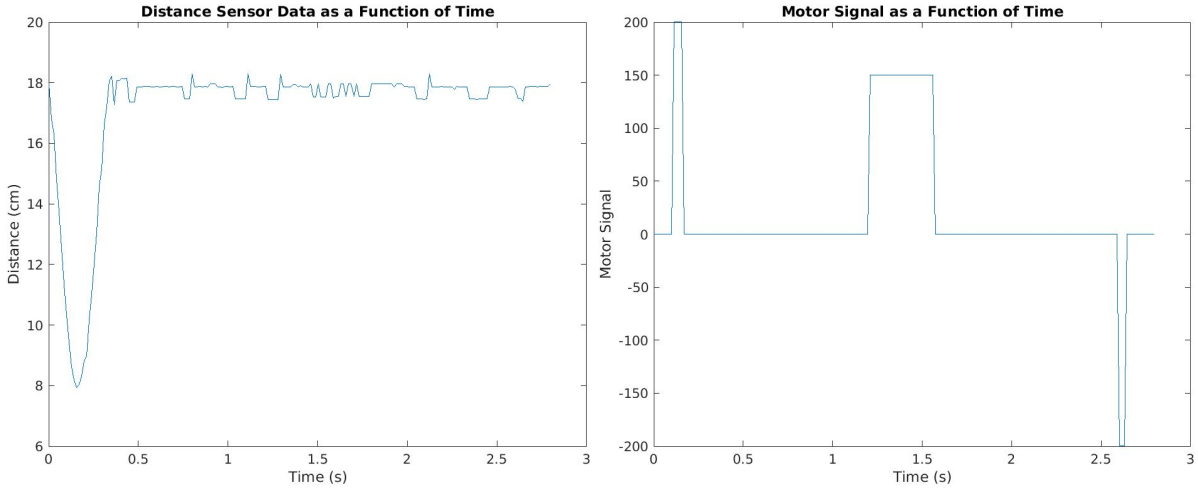
Showing above is the circuit diagram of the device.

Finite State Diagram



Showing above is the finite state diagram of the device.

Data Analysis



Showing above is the diagram of the data collected when the device is activated. From the distance sensor diagram, we could see when the distance sensor is activated. And from the motor signal diagram we could see how the encoder reacts to that which include a locking process, paper releasing process and a unlock process which represents the three signal changes in the diagram.

Test Record

https://youtu.be/DDoJVqvZRl0

```
#include <ESP32Encoder.h>

// Inputs Pins
#define Encoder1 32
#define Encoder2 14

// Outputs Pins
#define Driver_Output0 33
#define Driver_Output1 15
#define Channel0 0
#define Channel1 1

// Constant Values
#define Frequency 22000
#define Resolution 8
#define Debounce 200

// Distance Sensor Pins
#define trig 21
#define echo 17

// Distance Sensor Data
float Duration = 0; //The length of the sound
float Distance = 0; // Distance data
float DesDistance = 10; //Trigger distance

// Encoder Setting
float Vel_1 = 200;
float Vel_2 = 150;
float Vel = 0;
volatile float CurrentTime = 0;
volatile float StartTime = 0;
int StartCheck = 0;
bool encoderPaused = false;
bool printToggle = true;
ESP32Encoder encoder;
```

```
void setup() {
// Setting pin mode
  pinMode(Driver_Output0, OUTPUT);
  pinMode(Driver_Output1, OUTPUT);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);

// Setting motor driver channels
  ledcSetup(Channel0, Frequency, Resolution);
  ledcSetup(Channel1, Frequency, Resolution);
  ledcAttachPin(Driver_Output0, Channel0);
  ledcAttachPin(Driver_Output1, Channel1);
  ledcWrite(Channel0, 0);
  ledcWrite(Channel1, 0);

// Setting encoder
  ESP32Encoder::useInternalWeakPullResistors = UP;
  encoder.attachFullQuad(Encoder1, Encoder2);
  encoder.setCount(0);

  Serial.begin(9600);
}

void loop() {
// Generate 10-ms pulse to the sending pin
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

// Store the duration when echo pin is high
  Duration = pulseIn(echo, HIGH);

// Calculation of distance
  Distance = (Duration*0.0343)/2;
  Serial.print(Distance);

// Report trigger time
```

```
  CurrentTime = millis();

// When the distance sensor is triggered, record current time
  if (StartCheck == 0 && Distance <= DesDistance) {
    StartTime = millis();
    StartCheck = 1;
  }

// Rotate 50ms to release the arms
  if (StartTime != 0 && CurrentTime-StartTime <= 50) {
    Vel = Vel_1;
    ledcWrite(Channel0, Vel);
    ledcWrite(Channel1, 0);
  }

// Stop for 1s
  if (StartTime != 0 && CurrentTime-StartTime > 50 &&
CurrentTime-StartTime <= 1050) {
    Vel = 0;
    ledcWrite(Channel0, 0);
    ledcWrite(Channel1, 0);
  }

// Rotate 400ms to release the paper
  if (StartTime != 0 && CurrentTime-StartTime > 1050 &&
CurrentTime-StartTime <= 1450) {
    Vel = Vel_2;
    ledcWrite(Channel0, Vel_2);
    ledcWrite(Channel1, 0);
  }

// Wait for 1s
  if (StartTime != 0 && CurrentTime-StartTime > 1450 &&
CurrentTime-StartTime <= 2450) {
    Vel = 0;
    ledcWrite(Channel0, 0);
    ledcWrite(Channel1, 0);
```

```
  }

// Rotate backward 50ms to withdraw the arms
  if (StartTime != 0 && CurrentTime-StartTime > 2450 &&
CurrentTime-StartTime <= 2500) {
    Vel = -Vel_1;;
    ledcWrite(Channel0, 0);
    ledcWrite(Channel1, -Vel);
  }

// Reset varriables
  if (StartTime != 0 && CurrentTime-StartTime > 2500) {
    Vel = 0;
    ledcWrite(Channel0, 0);
    ledcWrite(Channel1, 0);
    StartTime = 0;
    StartCheck = 0;
  }
  Serial.print(",");
  Serial.print(Vel);
  Serial.println(";");
}
```