Rotating LED Strips Using Persistence of Vision

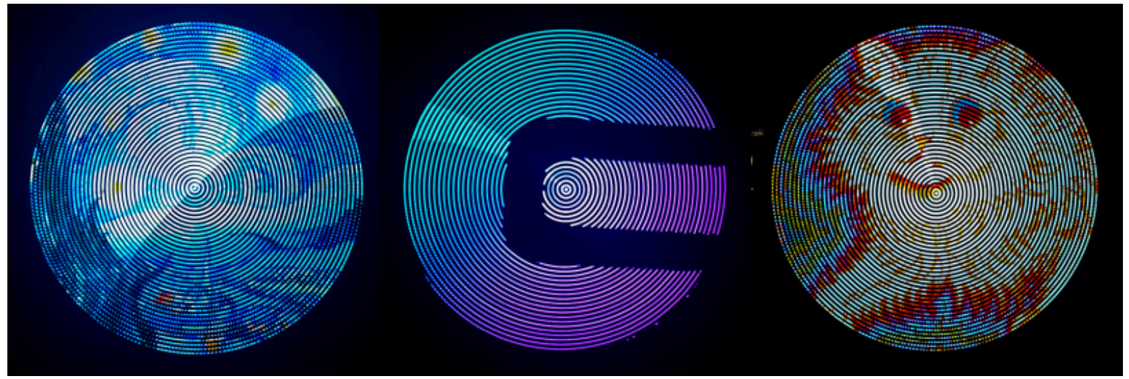
Students: Paul de Lesquen, Walid Megherbi, Florian Babin

Teachers: Christophe Jego, Maxime Gras-Chevalier

BORDEAUX INP Enseirb-Matmeca

Introduction and Context

Examples of similar projects :



Real time constraint : between 333µs and 476µs (for a rotation speed of 700 to 1000 rpm)

Our main goals :

- make it safer than last year
- reducing the rotation speed of the bar
- improve support robustness
- use higher level architectures

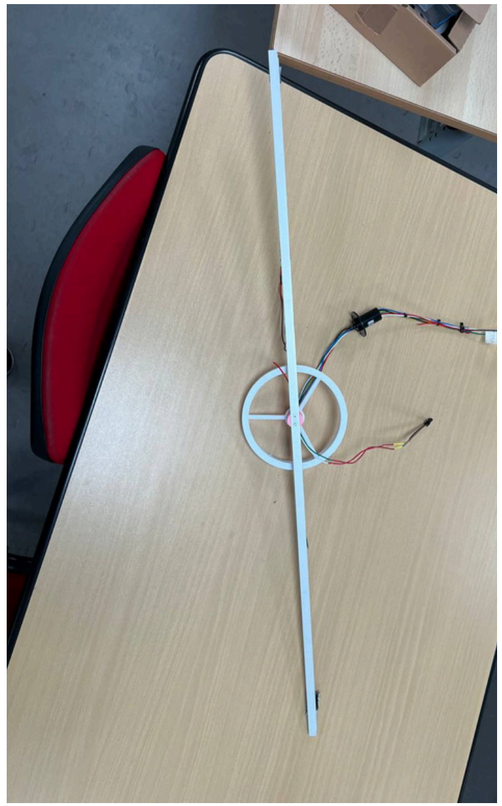
+ shaped LED bar

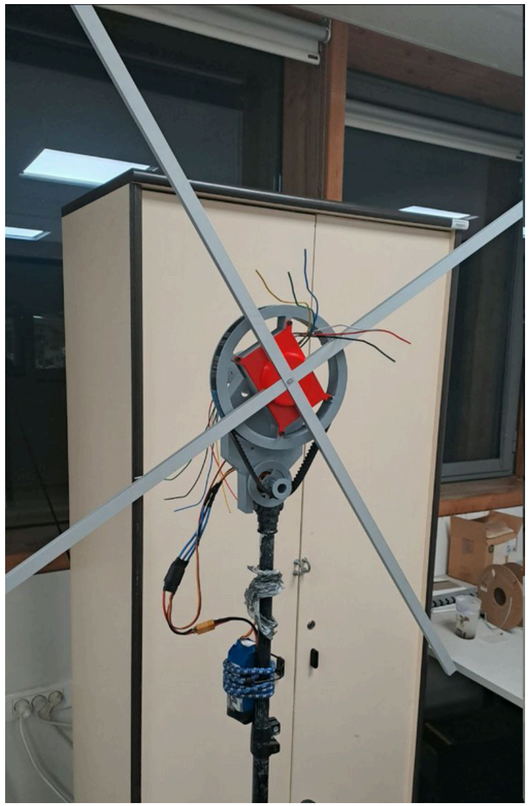
Benefits

- Slower rotation speed
- Better resolution
- Brighter image
- Redundancy

Disadvantages

- More weight
- More LED lights to manage





4 bars instead of 2

Controlling LED lights

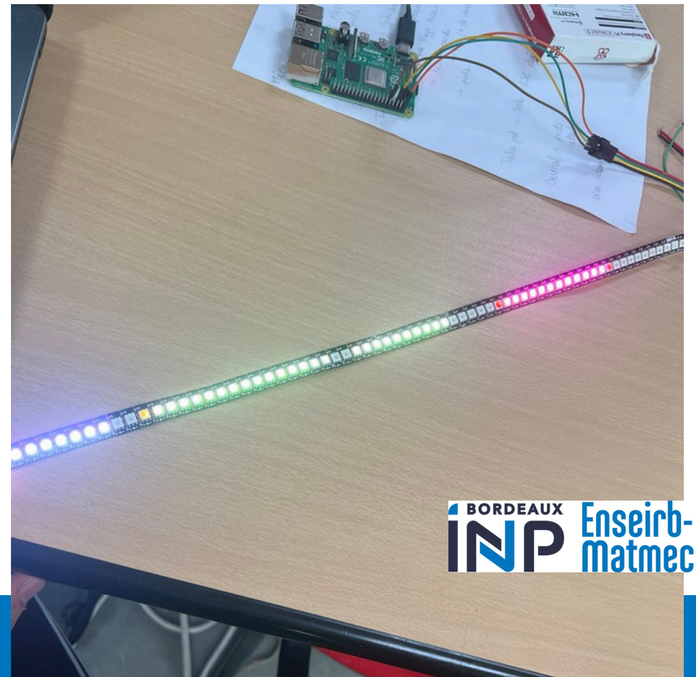
Tools used

- Python with adafruit.dotstar library
- Raspberry Pi 4B board
- 4 LED strips

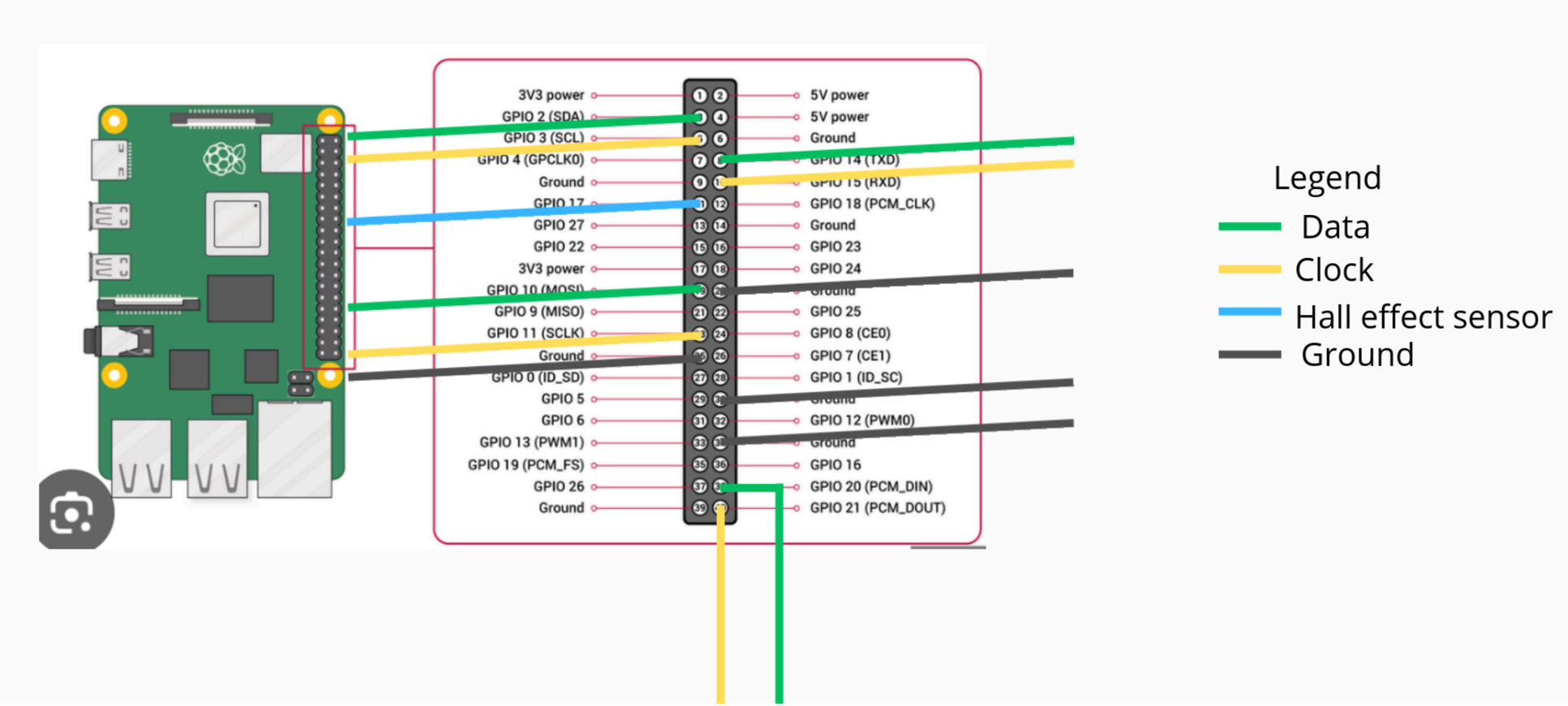
- **4 SPI interfaces activated on the board**

```
17
18  # East
19  strip_e = adafruit_dotstar.DotStar(board.D3, board.D2, NUM_LEDS, brightness=0.1, auto_write=False)
20  # North
21  strip_n = adafruit_dotstar.DotStar(board.D11, board.D10, NUM_LEDS, brightness=0.1, auto_write=False)
22  # West
23  strip_w = adafruit_dotstar.DotStar(board.D21, board.D20, NUM_LEDS, brightness=0.1, auto_write=False)
24  # South
25  strip_s = adafruit_dotstar.DotStar(board.D15, board.D14, NUM_LEDS, brightness=0.1, auto_write=False)
26
```

- Usual SPI frequency : 8MHz



Controlling LED lights



Our image processing algorithm

Reference state

- bars in + shape
- 4 bars : East, North, West, South
- Hall Effect Sensor to sense position and rotation speed

- Use of last year's project

```python
def get_pixel_from_polar(angle_deg, radius):
    angle_rad = math.radians(angle_deg)
    x = int(center_x + radius * math.cos(angle_rad))
    y = int(center_y - radius * math.sin(angle_rad))
    if 0 <= x < width and 0 <= y < height:
        return img.getpixel((x, y))
    return (0, 0, 0)
```

- At 8MHz, SPI is the bottleneck (~1ms response)

- initially worked for 1 bar
- generalized for 4 bars using offsets

```python
# ========================
# Displays a strip
# ========================
def display_strip(strip, angle_deg):
    for r in range(NUM_LEDS):
        strip[r] = get_pixel_from_polar(angle_deg, r)
    strip.show()

# = Main loop
# ========================
global_angle = 0

while True:

    # --- Reference position : angle = 0 ---
    if sensor == 1:
        global_angle = 0
        sensor = 0

    # --- Processing angles ---
    angle_e = (global_angle + 0) % 360
    angle_n = (global_angle + 90) % 360
    angle_w = (global_angle + 180) % 360
    angle_s = (global_angle + 270) % 360

    # --- Displaying in real time ---x
    display_strip(strip_e, angle_e)
    display_strip(strip_n, angle_n)
    display_strip(strip_w, angle_w)
    display_strip(strip_s, angle_s)
```

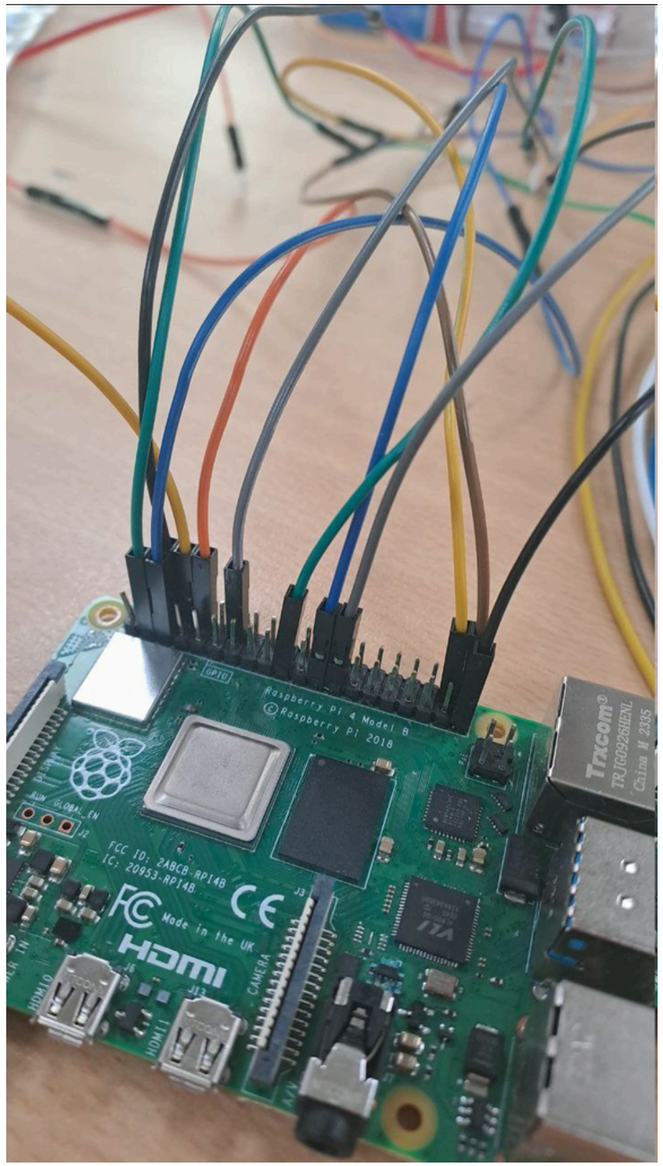
Simulation of the system

Animation:

test the processing algorithm

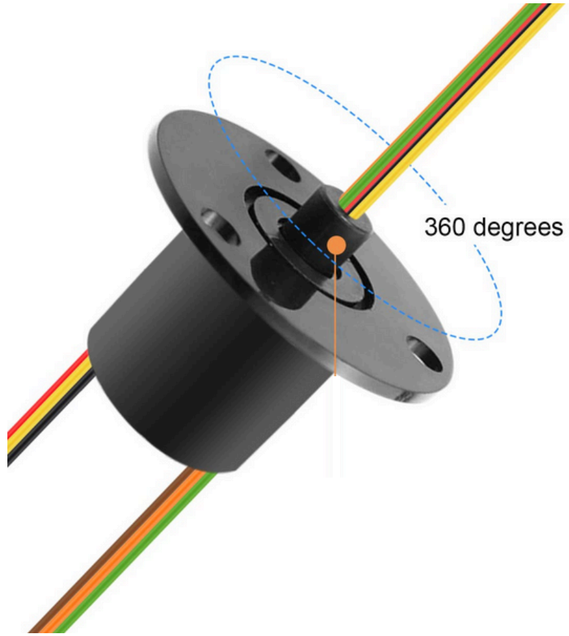
The global System: Fixed Part

- **Raspberry Pi** (Fixed): motor + web interface.
- **ESC & PWM**
- Motor **Power Supply**
- Brushless **Motor**: Drives the rotating system.
- **Hall Sensor**: Synchronizes rotation speed.

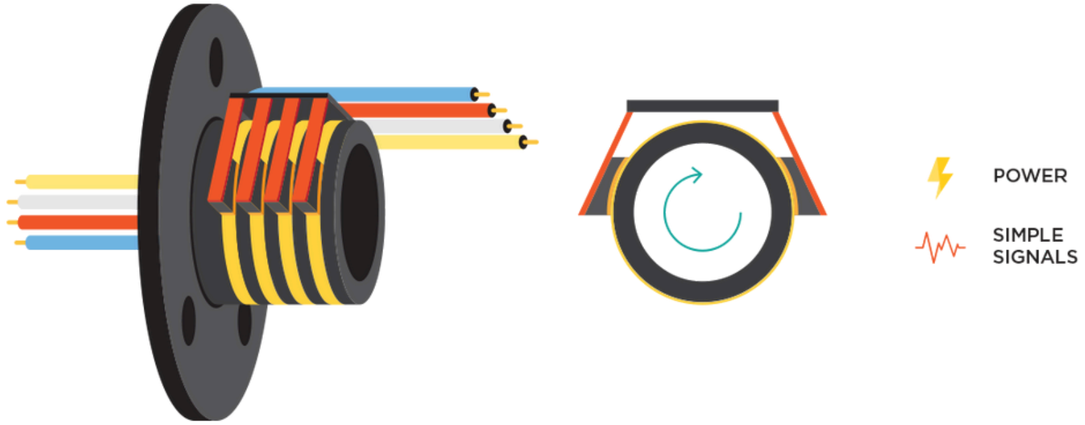


The global System: Rotating Part

- **Raspberry Pi** (Rotating): LED via **SPI**.
- **LED & Power Supply**:
- Mechanical Transmission: **Pulley and belt**
- **Magnets** for **synchronization**

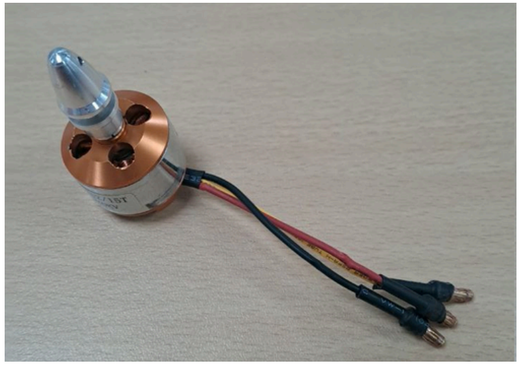


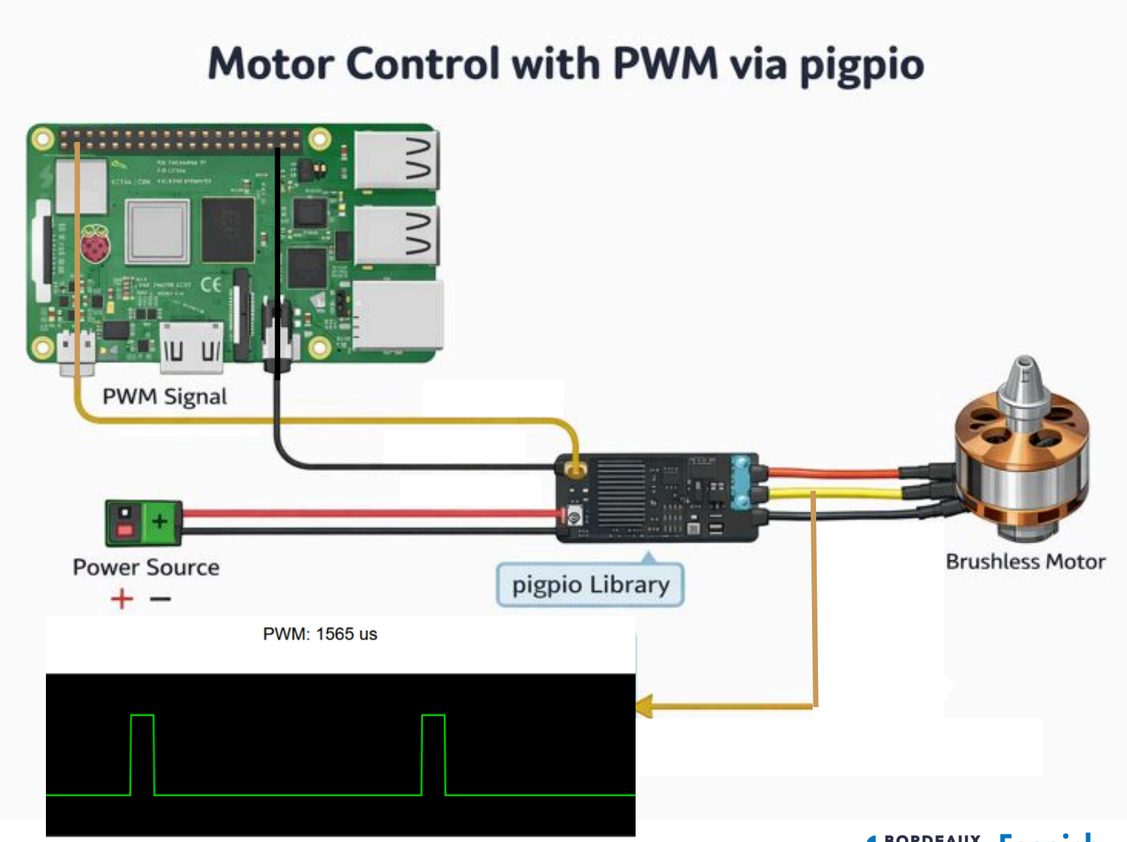
Slip ring: tranmission between 2 parts →



Rasp 1 – Motor control

- Raspberry → **PWM** signals
- **ESC**: Control motor's speed.
- **pigpio**: daemon for real time
- PWM Control:
  - Higher duty cycle = faster speed
  - Lower duty cycle = slower speed.
  - **[1000 µs ; 2000 µs]**

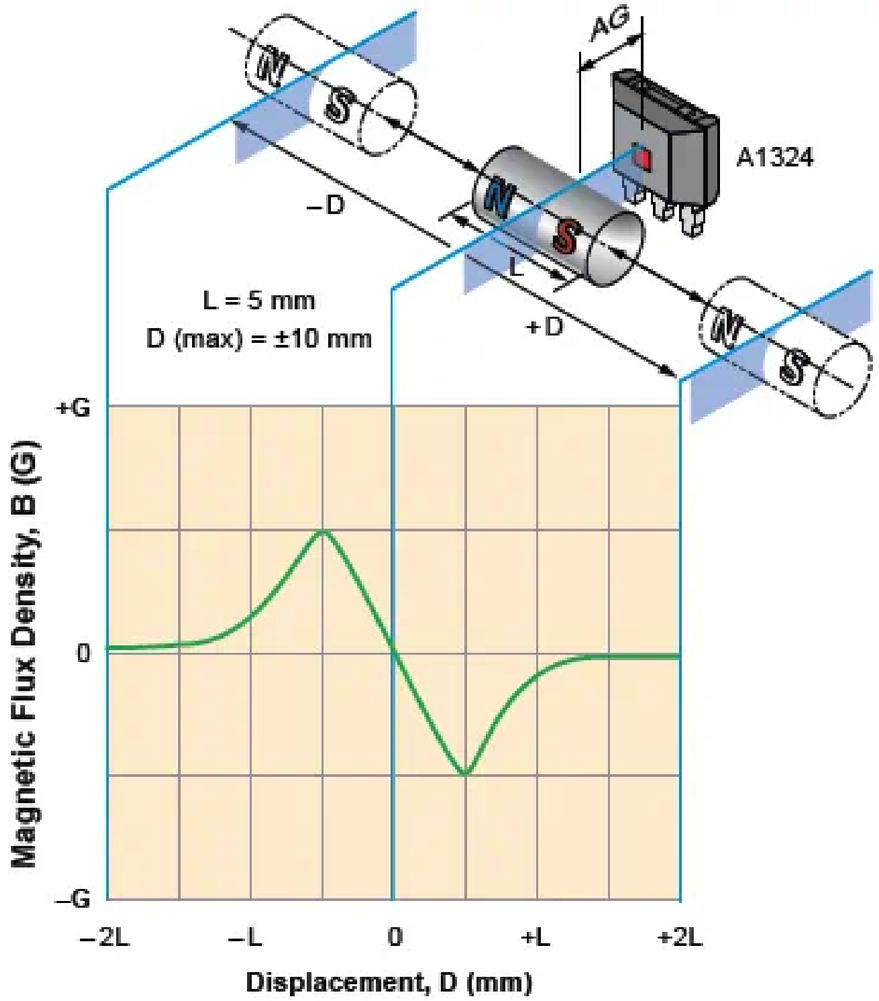




Rasp 1 – Rotation Sensor

Hall Effect Sensor

- Wheel rotation speed.
- Signal Processing: calculates RPM
- **Closed-loop** control system (abandonned)
- **Reset Pulse**: reset and synchronize



Rasp 1 – Interface controlled by the Rasp

TUI Interface:

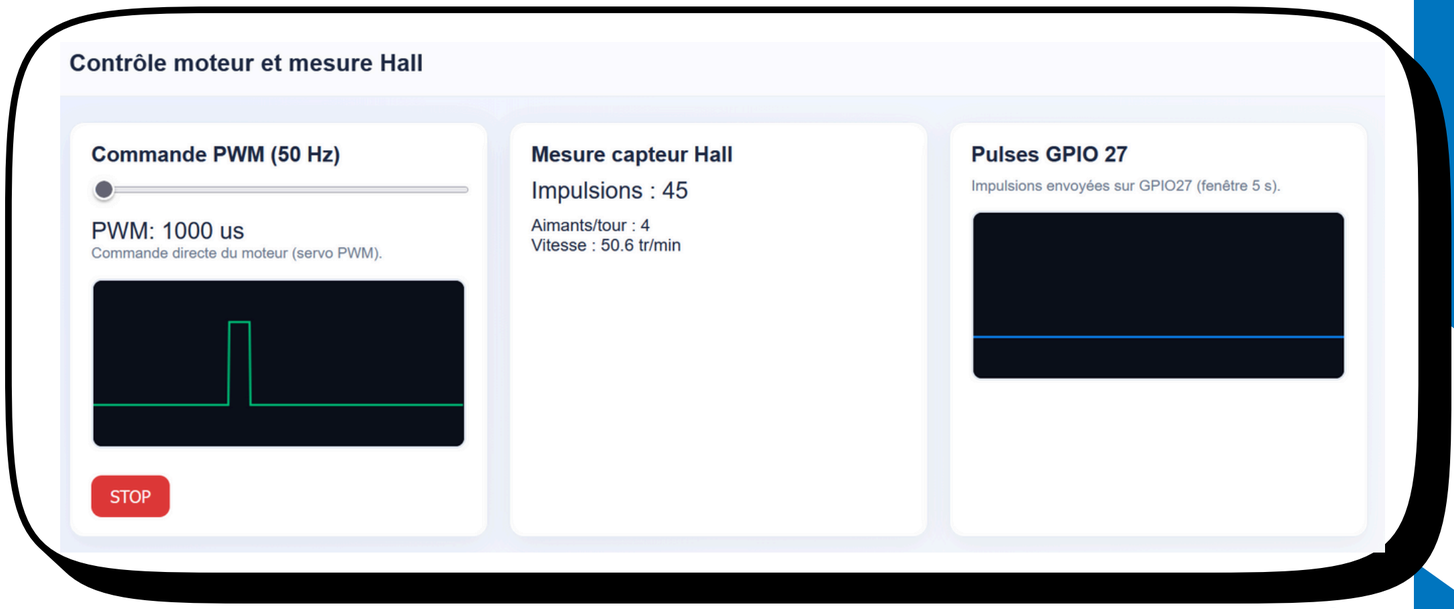
- Command-line interface:

  real-time system control.
- Displays data (PWM, RPM, etc.)
- Motor control

→ **Web Interface**:

- **Flask** web interface
- running locally on the Raspberry Pi

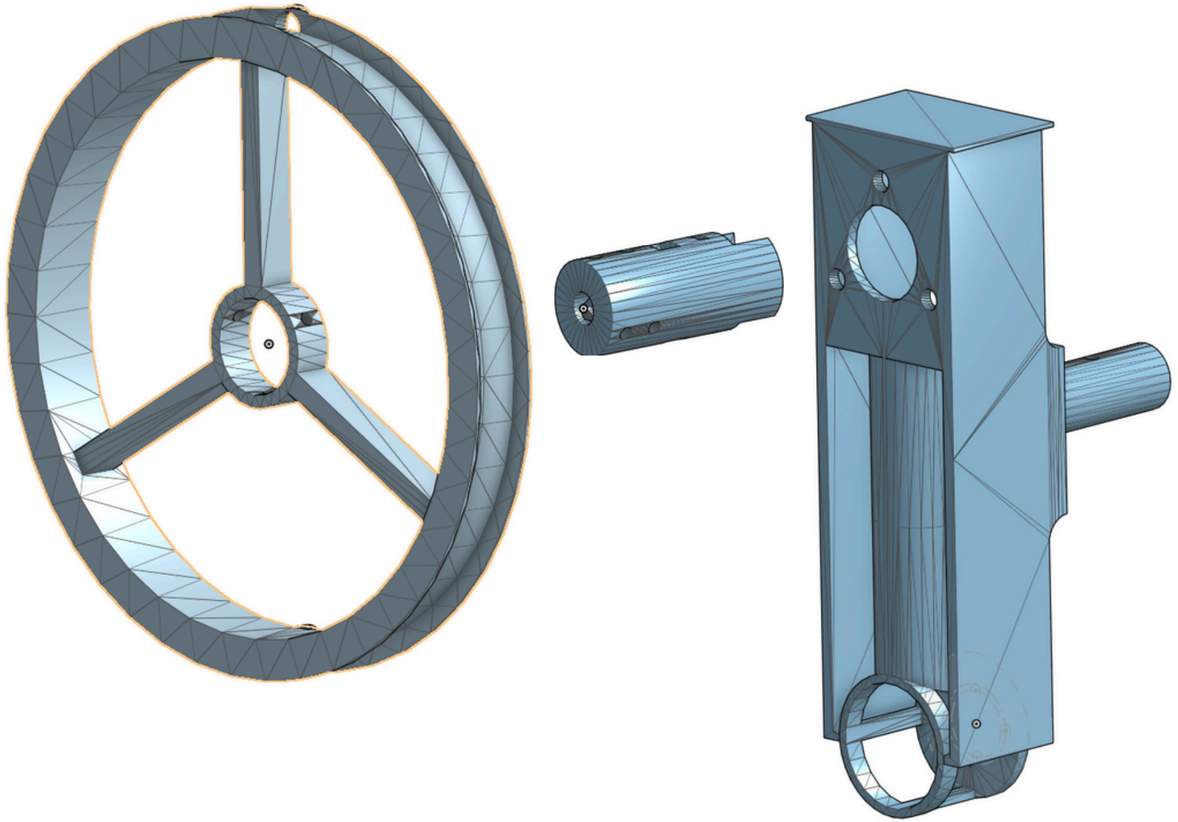
```
projets9 TUI
pigpio: ok | mode: servo
rpm: 0.0 | freq: 0.00 Hz | omega: 0.00 rad/s
period: N/A | pulses: 0
last: N/A
servo: 1000 us

keys: +=up  -=down  x=stop  q=quit
```



Mechanics

New design objectives

- Weight distribution
- Raspberry Pi placement
- Rotational axis and ball bearing
- Pulley and belt transmission
- Easy disassembly



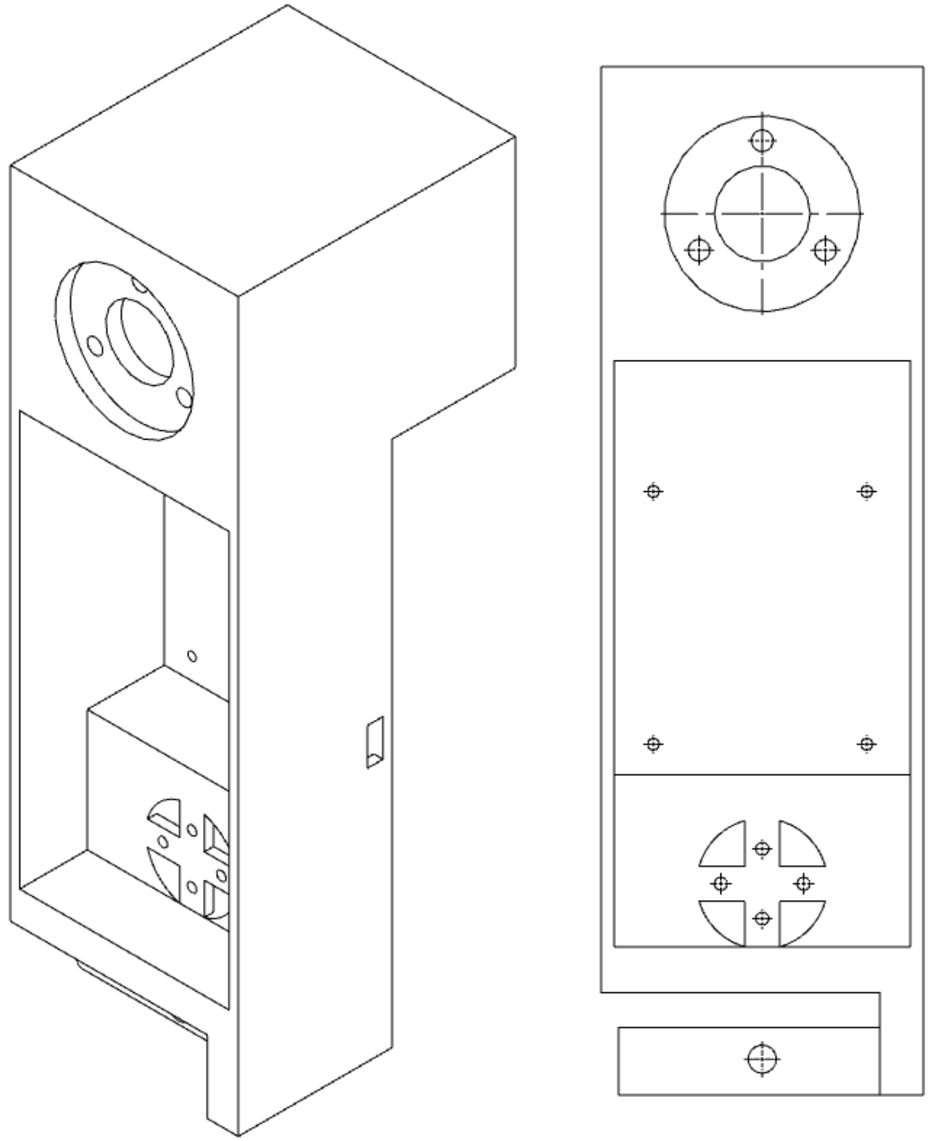
Mechanics

First Design

- Tripod repositioning
- Ball bearing housing
- Raspberry Pi compartment

Disadvantages

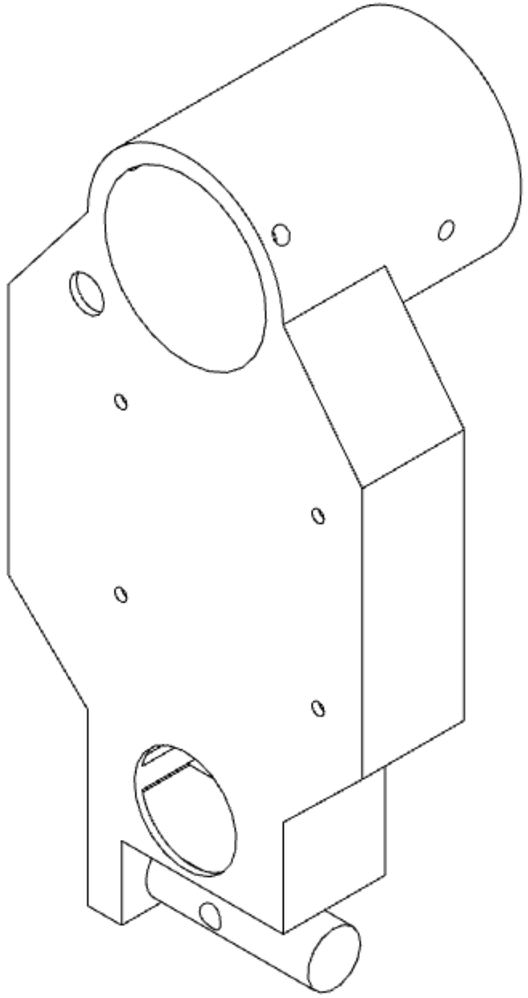
- Motor mounting
- Weight distribution
- Unaesthetic, blocky design

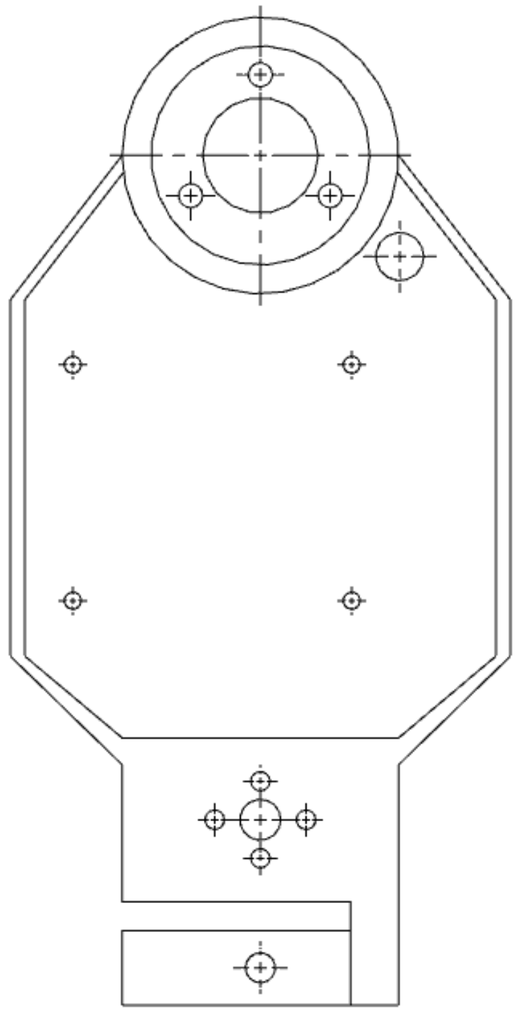


Mechanics

Final Design

- Screws to secure the ball bearing
- Access holes for the Hall sensor
- Improved motor mounting
- More aesthetic design

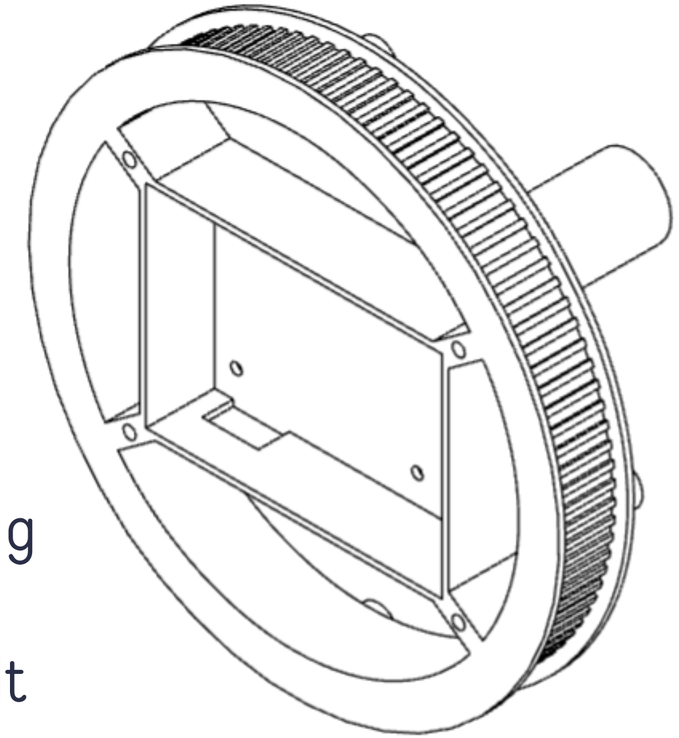


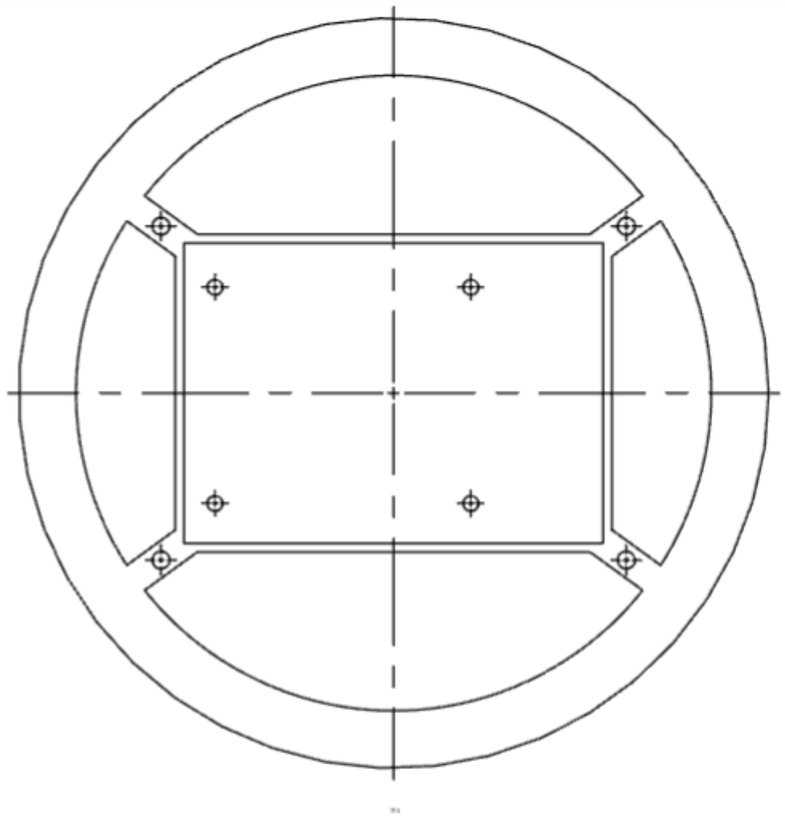


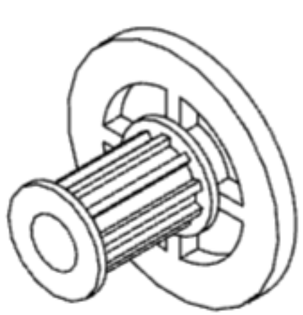
Mechanics

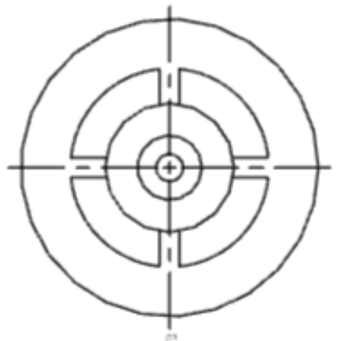
Final Design

- Reduction ratio: 10
- Integrated Raspberry Pi
- Support for bars
- Easy to disassemble using screws
- Adapted to a toothed belt (5 mm pitch)





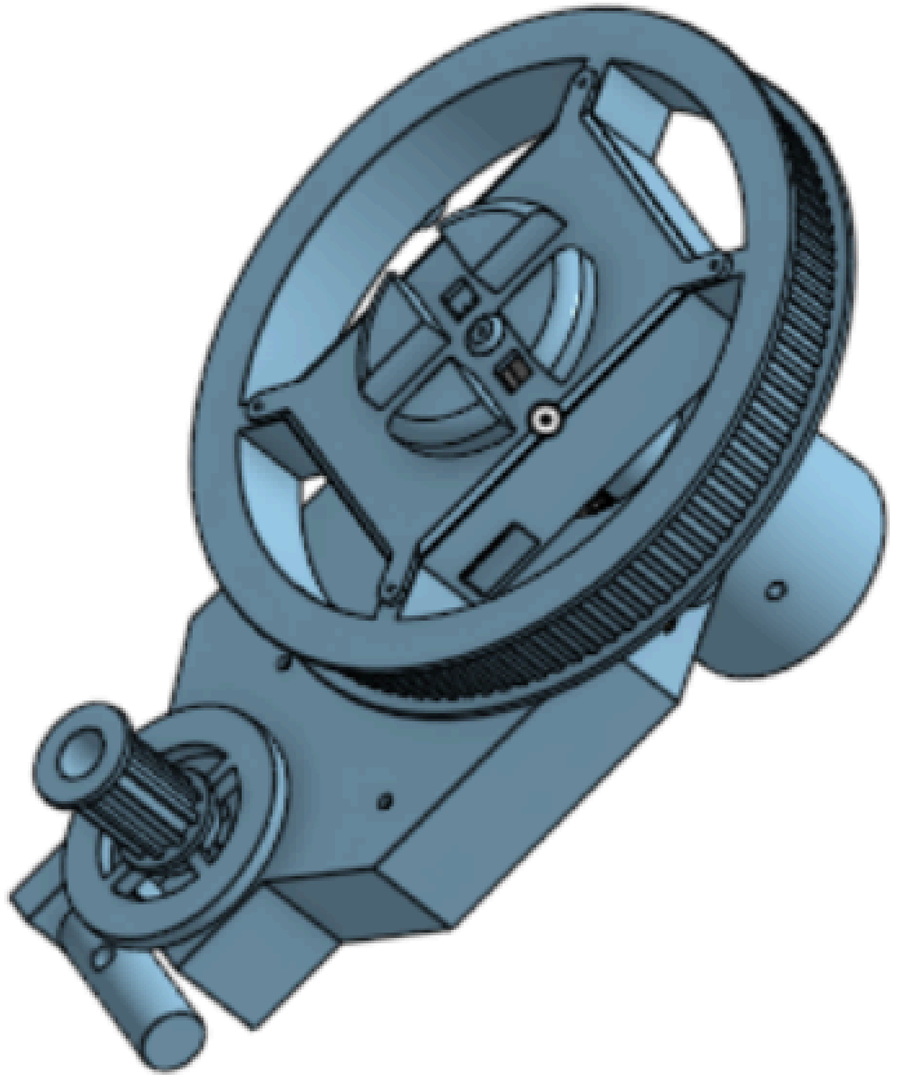




Conclusion and Perspectives

- Mechanics and electronics combined
- 3D design and prototyping
- Motor and LEDs control

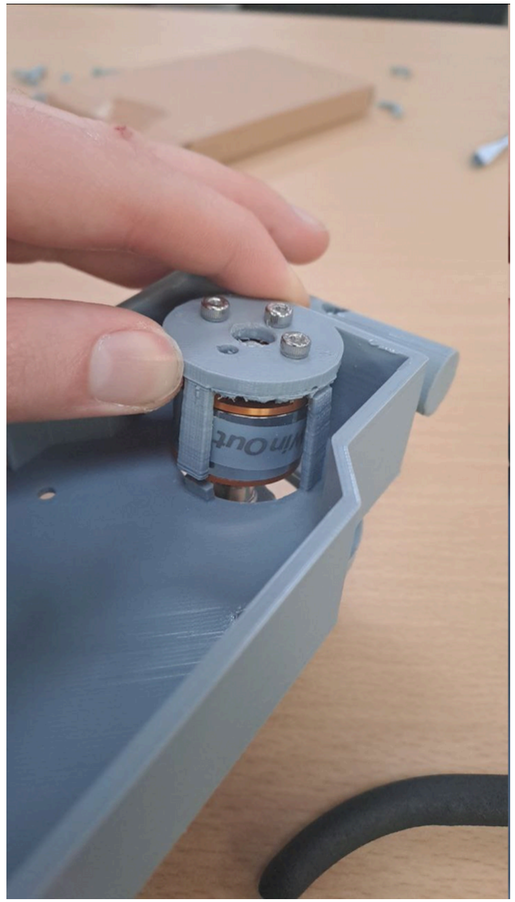
Power for Raspberry Pi



Thank you for your attention !

BORDEAUX INP Enseirb-Matmeca

Appendix



Appendix