

# Back-of-device wheel for mobile one-handed interaction

ADAM GLOBISCH, SARAH LOMP

# Table of Contents

---

- Context
- Research Question
- Related Work
- Prototype
  - Material
  - Sketch
  - Implementation
- Experimental Plan
- Results
- Conclusion

# Context

---

## Problem:

- Hold smartphone horizontally with only one hand
- Not able to reach every corner of the screen
- One-handed video skipping difficult
  - Low accessibility
  - Fat finger problem



# Context

---

## Chances:

- Back-of-Device Tangible User Interface
- Wheel, that is controlled by turning
- Handles the situation of one-handed timeskipping in videos
- Improves accessibility

# Research Question

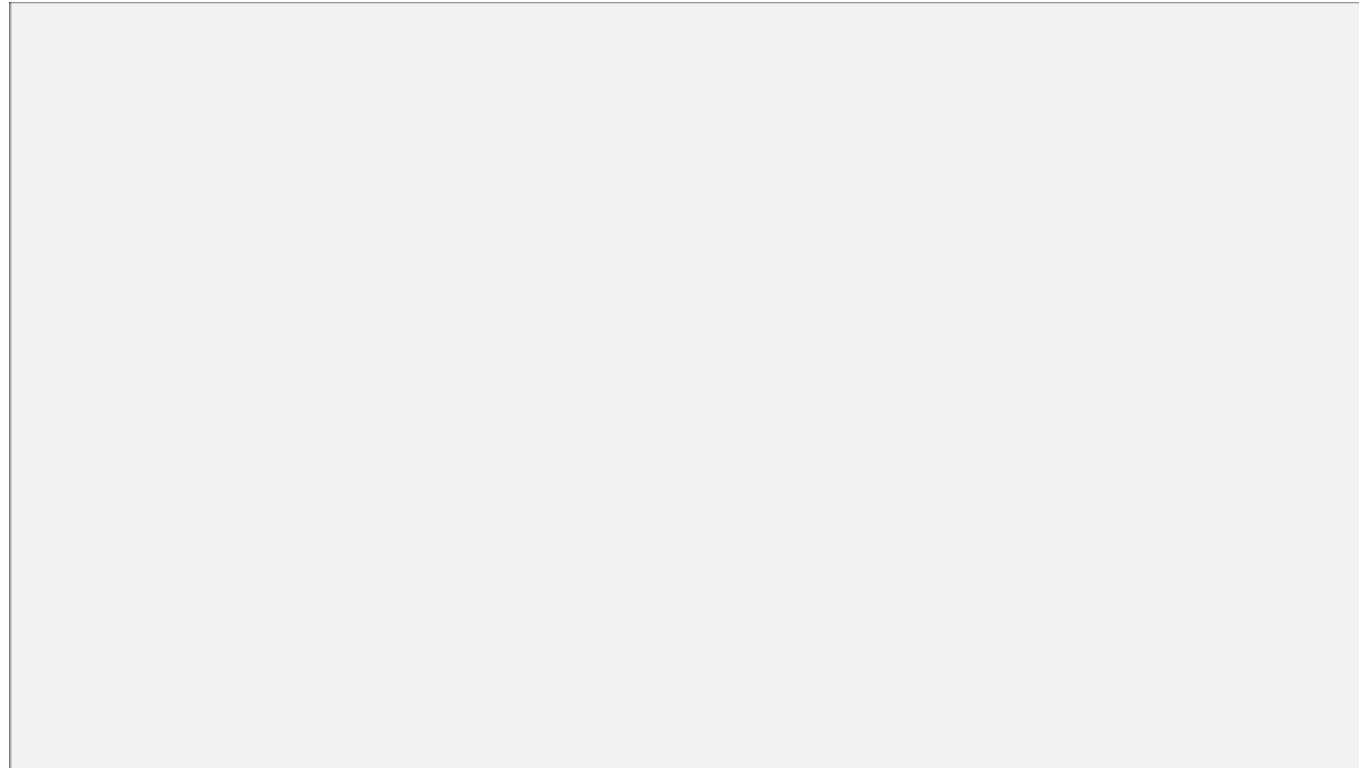
---

Can a back-of-device wheel for mobile one-handed interaction improve precision and overall user-satisfaction in video skipping?

# Related Work

---

**Exploring Around-Device Tangible Interactions for Mobile Devices with a Ring to skip videos**



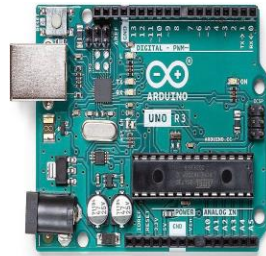
<https://dl.acm.org/doi/10.1145/3173225.3173283>

# Prototype – Material

---



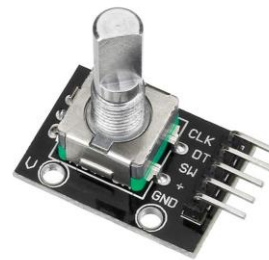
**SMARTPHONE  
+ APP**



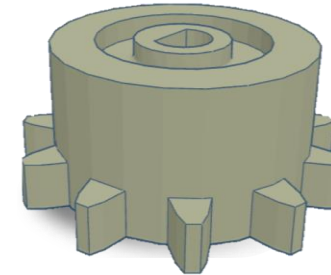
**ARDUINO**



**BLUETOOTH  
MODULE**



**ROTARY  
ENCODER**

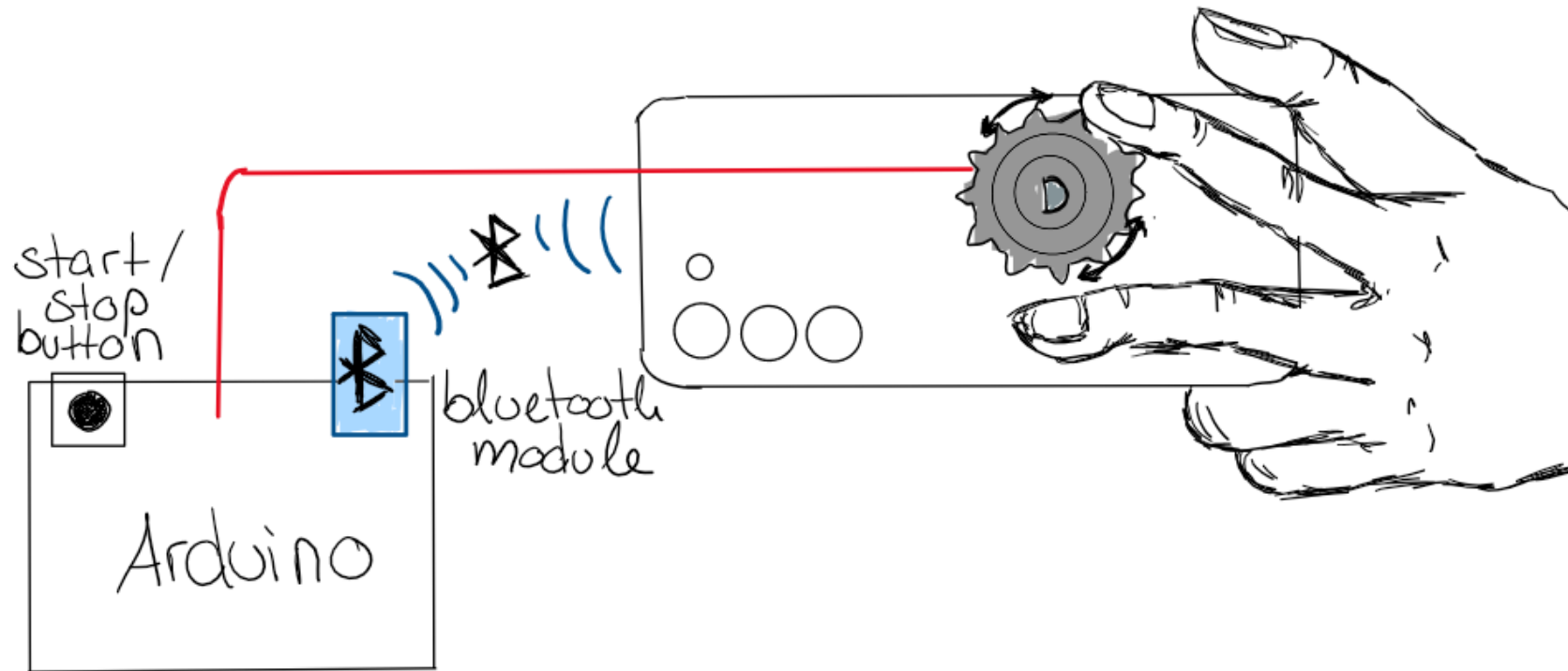


**ATTACHABLE  
WHEEL FOR  
ROTARY  
ENCODER**



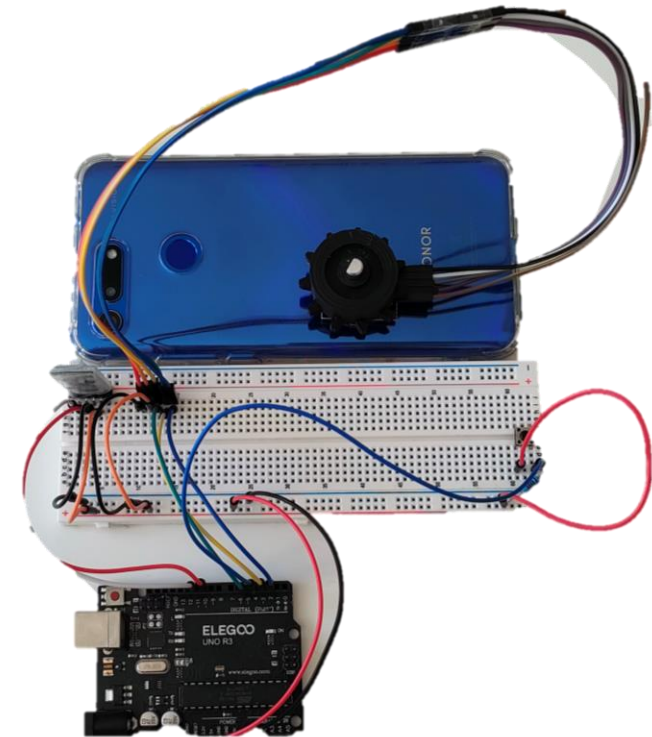
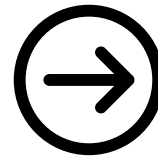
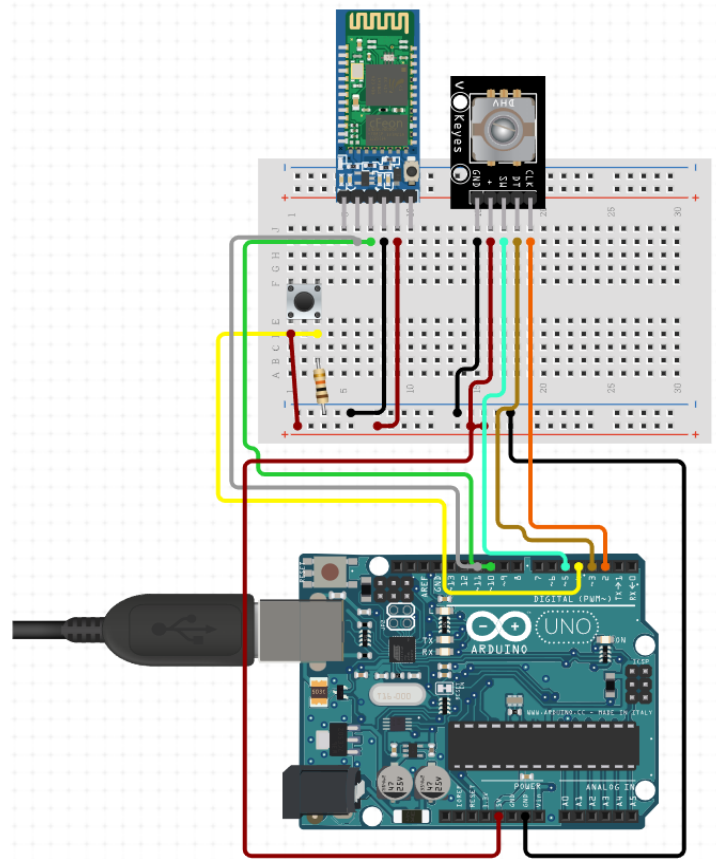
**BUTTON**

# Prototype – Sketch





# Prototype – Implementation Hardware



# Prototype – Implementation Software

## Arduino Code:

- On 1st Button Press: start timer
- On 2nd Button Press: stop timer and send data to Excel
- If Rotary Encoder registers turn: send turn-direction via Bluetooth
  - encode clockwise turns as "F"
  - encode counter-clockwise turns as "B"

```

44 void loop() {
45   buttonState = digitalRead(buttonPin);
46   if (buttonState == LOW && lastButtonState == 1) {
47     if (startTime == 0) {
48       // start timer
49       startTime = millis();
50       Serial.print("Test ");
51       Serial.println(test++);
52     } else {
53       // stop timer and print the value
54       elapsedTime = millis() - startTime;
55       Serial.print(elapsedTime / 1000); // time in seconds
56       Serial.print(",");
57       Serial.println(elapsedTime % 1000); // milliseconds
58       startTime = 0;
59     }
60   }
61   lastButtonState = buttonState;
62   // Read the current state of CLK
63   currentStateCLK = digitalRead(CLK);
64   // If last and current state of CLK are different, then pulse occurred

```

# Prototype – Implementation Software

## Android Studio Code:

- Initialize Video Player
- Establish Bluetooth connection with the Arduino:
  - Create Bluetooth socket
  - Constantly read Arduino data from incoming Input Stream
- Handle received data:
  - If message contains "F": skip video forward
  - If message contains "B": skip video backwards
- Determine skip duration
  - Measure amount of incoming messages per second
  - Calculate skip duration based on the threshold (min. 0.5 sec, max. 10 sec)

```
private fun calculateAdjustedSkipDuration(messageFrequency: Double): Long {  
    // Define the threshold for faster skipping  
    val threshold = 3.0 // 3 messages received per second  
  
    // Adjust skip duration based on message frequency  
    val range = MAX_SKIP_DURATION - MIN_SKIP_DURATION  
  
    // Determine whether to apply faster skipping based on the threshold  
    val adjustedSkipDuration = if (messageFrequency > threshold) {  
        (MIN_SKIP_DURATION + range * messageFrequency * 0.1).toLong()  
    } else {  
        // If below the threshold, use the regular skip duration  
        MIN_SKIP_DURATION  
    }  
  
    // Ensure the adjusted duration is within the specified range  
    return adjustedSkipDuration.coerceIn(MIN_SKIP_DURATION, MAX_SKIP_DURATION)  
}
```

# Experimental Plan

---

- 8 participants
- Solve time-skipping tasks
- Two groups - first with wheel, then by hand / first by hand, then with wheel
- Measuring:
  - Usability – SUS
  - Time – Excel
  - Precision and risks - Recording scene

# Experimental Plan

---

- Tryout phase:
  - Hold smartphone with wheel and try it out for 5 minutes to get to know the wheel, solve some time skipping tasks
- Testing phase:
  - Holding smartphone horizontally with only one hand
  - First with the wheel, after that just by hand - or the other way around
  - Push button to start - timer start
  - Skip video to certain time stamp
  - Push button to stop - timer stop
  - 30 times per method - 10 skips each - < 10 sec, < 30 sec, > 60 sec
- Review phase:
  - Fill in SUS and questions about personal preferences

# Experimental Plan

---

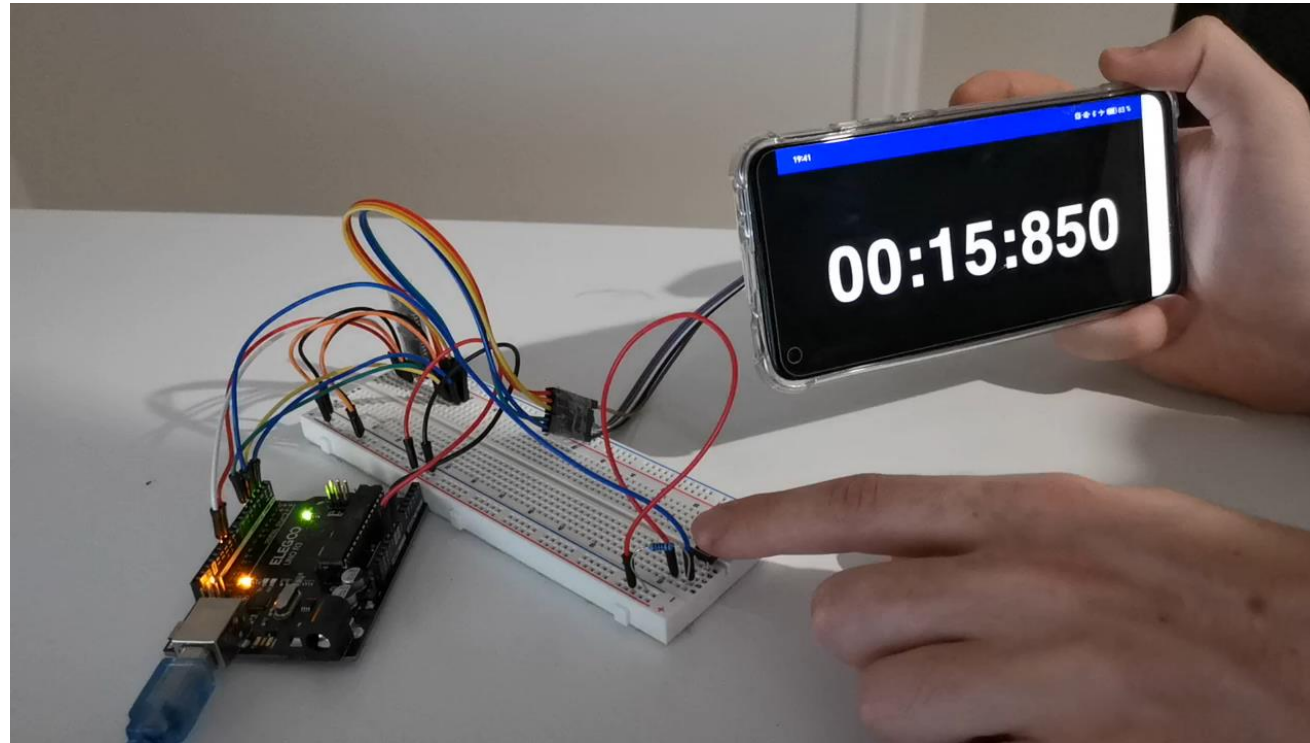
**Video of experiment:**



# Experimental Plan

---

Video of experiment:

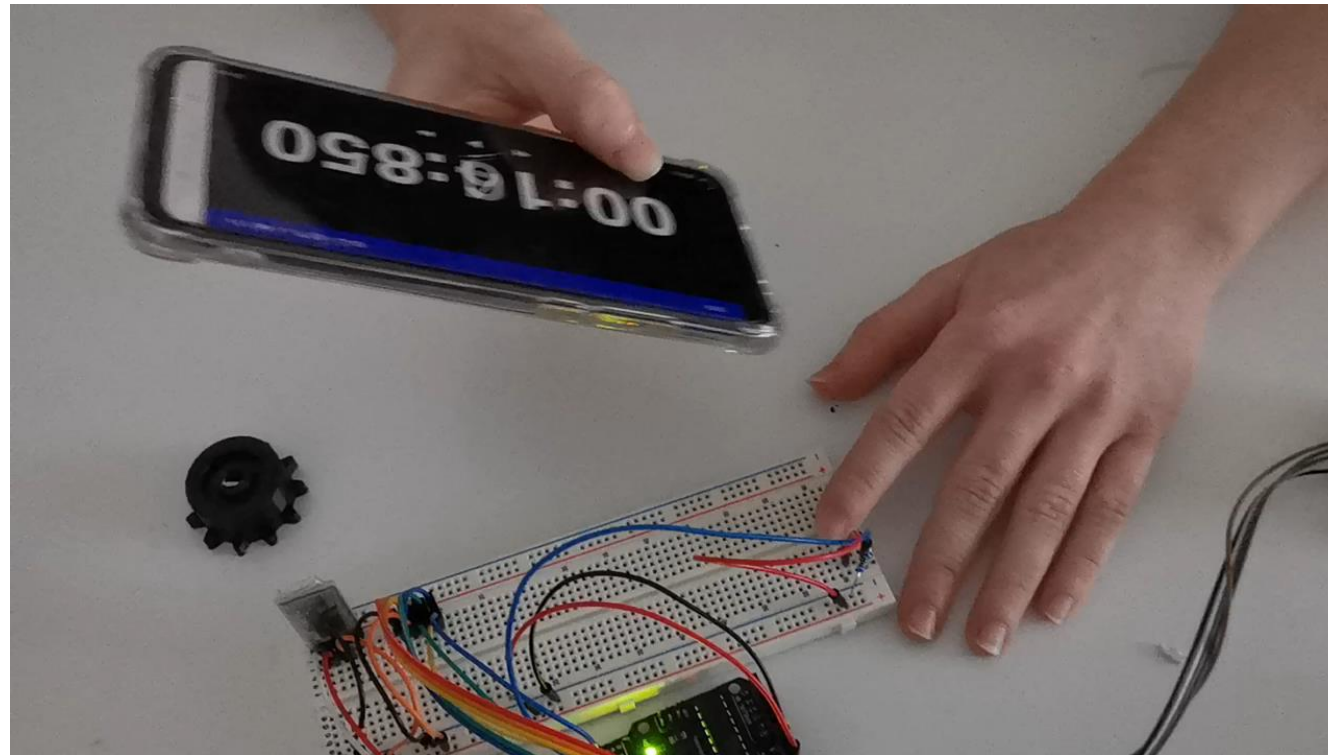




# Experimental Plan

---

Video of experiment:





# Experimental Plan

Timer data was stored in an Excel file:

Round 1	Sek	Msek		
< 30 sec	Wheel			by hand
1	2	956	2,956	4 466
2	5	489	5,489	9 448
3	2	389	2,389	2 731
4	3	569	3,569	5 27
5	2	230	2,230	2 782
< 10 sec				
1	1	621	1,621	1 93
2	2	264	2,264	1 75
3	2	262	2,262	1 40
4	1	415	1,415	2 859
5	2	240	2,240	1 937
> 1 min				
1	5	704	5,704	14 76
2	10	143	10,143	8 724
3	12	178	12,178	7 974
4	4	792	4,792	4 224
5	4	169	4,169	6 501

Skips were asked to the following time-stamps:

< 30 sec:	< 10 sec	> 60 sec
23 sec	11 sec	2,58 min
37 sec	19 sec	4,44 min
9 sec	23 sec	1,02 min
28 sec	17 sec	4,04 min
14 sec	7 sec	2,20 min

# Experimental Plan

## Questionary to be answered afterwards:

- 10 questions System Usability Scale
  - e.g. "I found the system very complicated to use."
  - 5 possible answers each (strongly agree – strongly disagree)
- 4 questions about personal preferences
  - e.g. "Which technique did you prefer for short skips ( $\leq 10$  sec)?"
  - 2 possible answers (by hand or wheel)
- 2 questions about strengths and difficulties with the wheel
  - e.g. "Where do you see strengths of the Wheel?"
  - Textual answer required

Which technique did you prefer for long skips ( $\geq 60$  sec)? \*

- ☐ BackWheel
- ☐ by hand

Where do you see challenges and difficulties of the BackWheel? \*

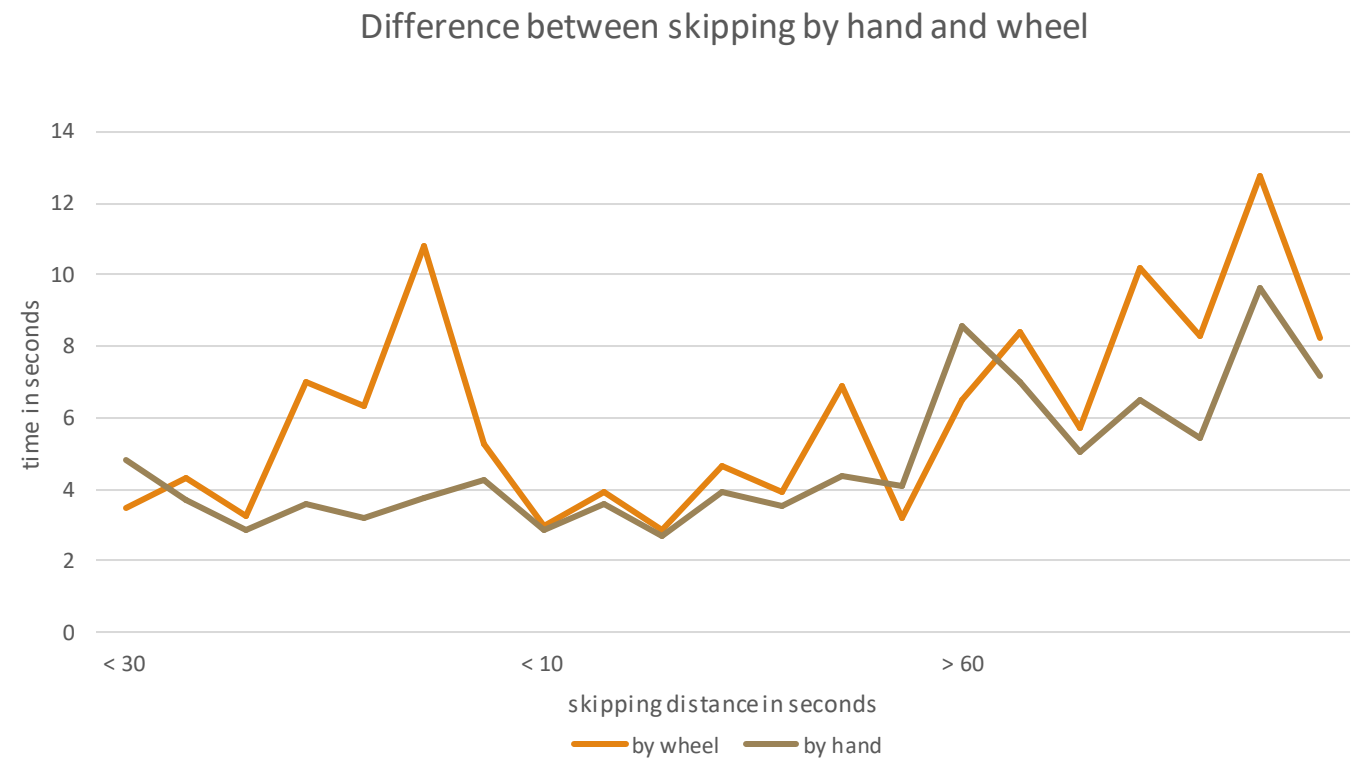
Meine Antwort

Where do you see strengths of the BackWheel? \*

Meine Antwort

# Results

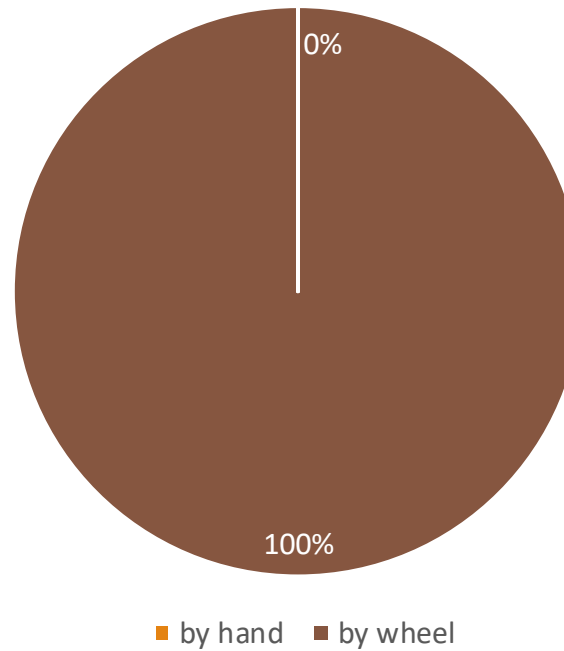
- SUS-Score: 74
  - Intermediate / good usability
- short skips nearly as time efficient as by hand
- mid / long skips by wheel less time efficient



# Results

---

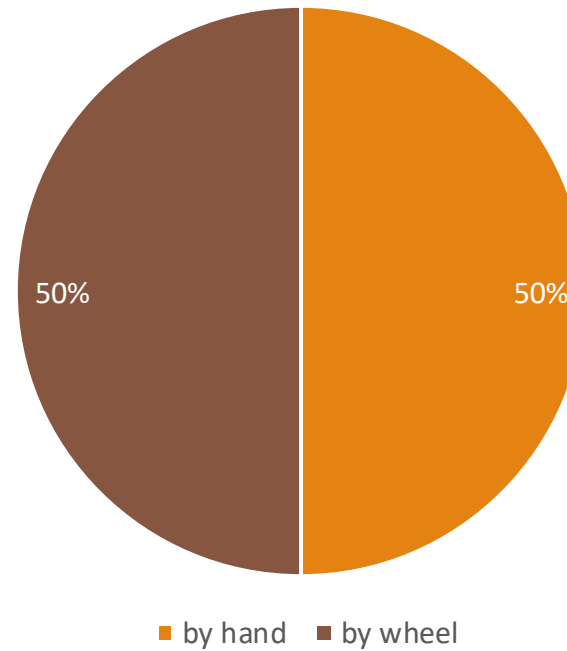
Which technique did you prefer for short skips ( $\leq 10$  sec)?



# Results

---

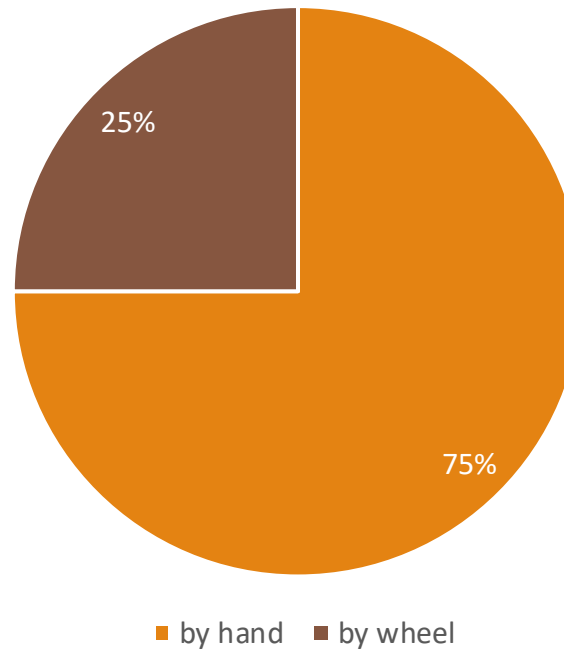
Which technique did you prefer for mid skips ( $\leq 30$  sec)?



# Results

---

Which technique did you prefer for long skips ( $\geq 60$  sec)?



# Results

---

## Overskipping:

- Most overskipping:
  - Wheel: long skips (Skipping too fast/far)
  - Wheel: for mid and long skips overskipping happened almost every time
  - by hand: short skips (Fat-Finger-Problem)
- Some difficulties occurred due to the rotary encoder's inaccuracy

## Risks:

- Both techniques made the smartphone hard to hold
  - Wheel: Prototype still very big and bulky
  - Wheel: ¼ of participants wanted to lay down the phones edge on the table for better grip
  - By hand: One person dropped the phone while skipping

# Results

---

## Strenghts of the wheel:

- “For precise task it was very useful because of the ticks”
  - “It's almost perfect for short skipping, I felt really comfortable using it for short skips”
  - “Its' ergonomical design and natural feel”
- 
- More comfortable for short / precise skips
  - Solves Fat-Finger-Problem



# Results

---

## Difficulties of the wheel:

- “For long skips it still takes a longer time compared to by hand”
  - “Sometimes the time change inconsistently”
  - “Hard to find the right spot to place the wheel”
- Less efficient for long skips

# Conclusion

---

## **User Satisfaction:**

- Back-of-Device wheel could not improve the overall user-satisfaction
- Overall, users still prefer skipping by hand

## **Precision:**

- Wheel could solve the Fat-Finger-Problem
- Users prefer the wheel for precise tasks

## **Prospective:**

- Reduce size of prototype for better grip
- Reduce inaccuracy of rotary encoder
- Improve precision in larger skips (refine scaling of skip duration in the app)