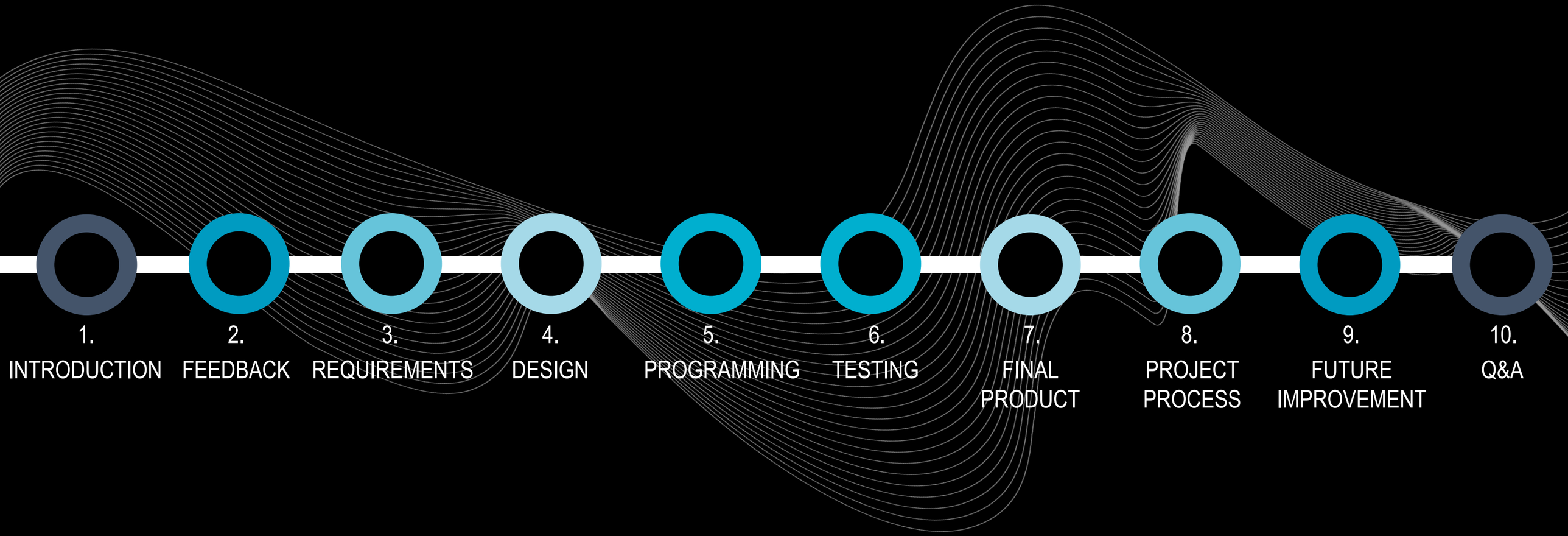
UNIVERSITY OF TWENTE.

T.W.I.P.

TWO-WHEELER INDICATOR PANEL

GROUP 42

IN THIS PRESENTATION:



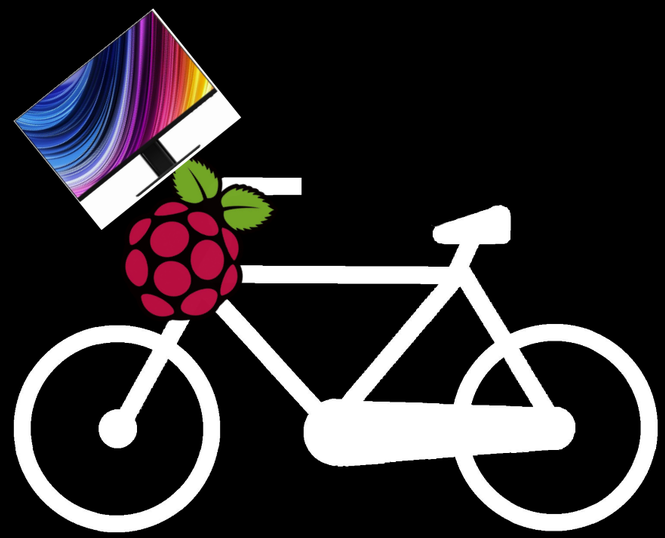
UNIVERSITY OF TWENTE. | T.W.I.P

1. INTRODUCTION

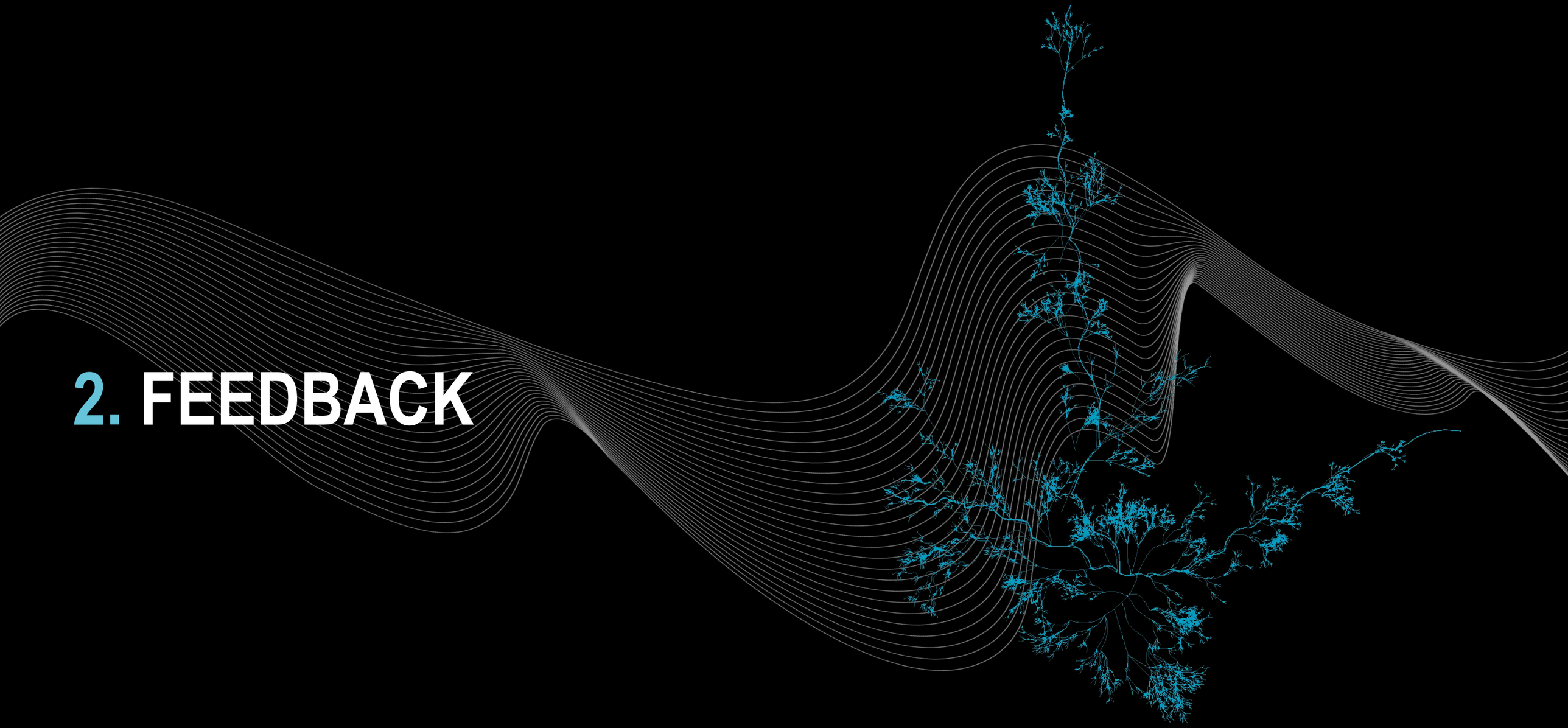
T.W.I.P. offers an advanced personal interface for cycling progress.

Multiple users & performance tracking

Target audience → professional & hobby cyclists



2. FEEDBACK



There was a MVP, but not a bike
→ close to completion

Solve bugs

Authentication

(security) testing not sufficient

3. REQUIREMENTS

Functional requirements

The system should...

- display current & average speeds.
- display current & previous pacing.
- keep track of total distance.
- keep track of trip distance of cycling session.
- automatically turn on/off lights depending on ambient light.
- calculate distance travelled.

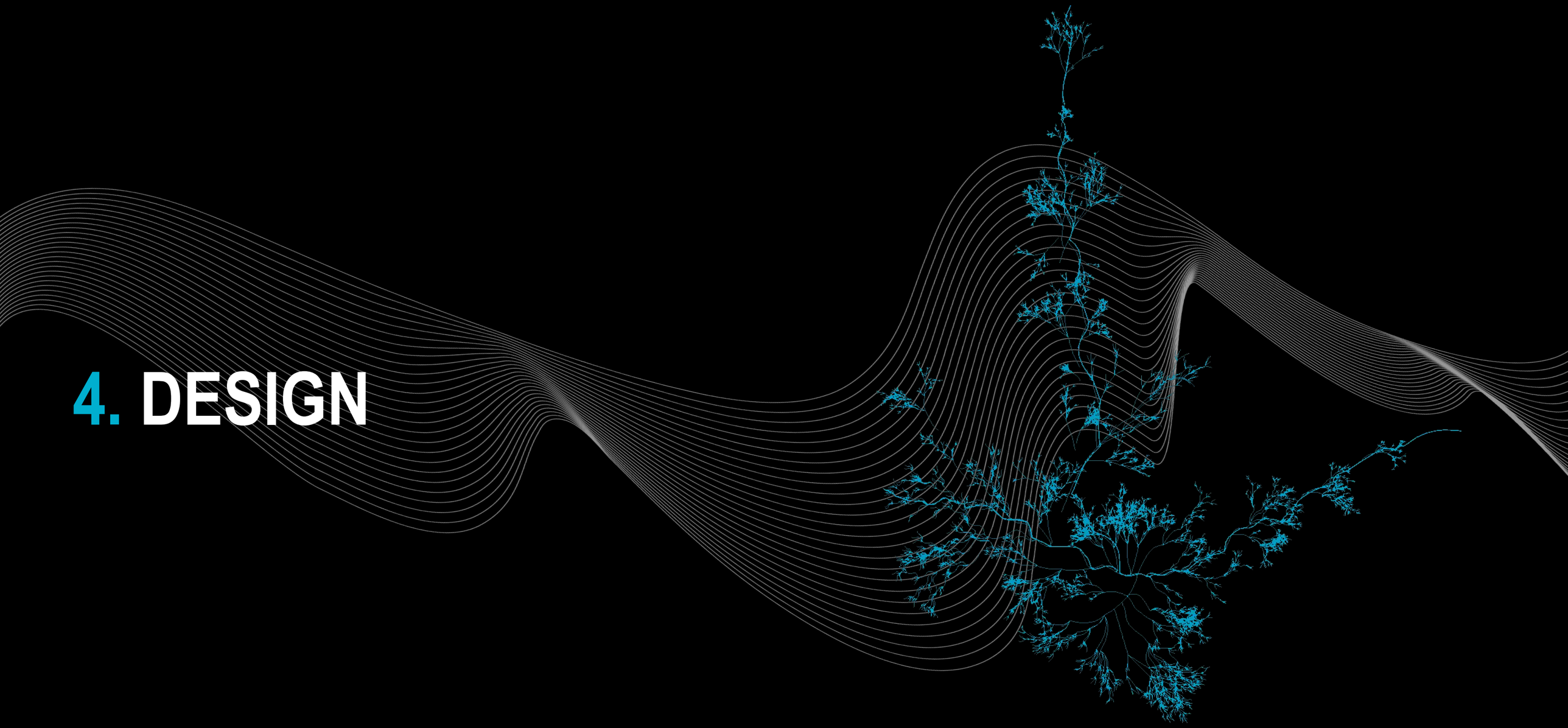
The user should be able to…

- log in using the touchscreen.

Non-Functional requirements

- Speed measure accurately within 1 km/h between 5 km/h and 25 km/h.
- Pacing should be comprehensible at a glance while cycling.
- Distances calculated accurate within 1%.
- Not draw user attention away from road.
- Comprehend data on the screen within 2 seconds of looking.
- Lights shouldn’t randomly flash when LDR is covered.
- LDR output should be consistent for 5 seconds before switching state of lights.
- Users should be able to delete their personal data.
- Complex functions should lock while bike is in motion
- Differentiate between users.

4. DESIGN



Minimal distraction interface

Information at glance

Additional features

Isolated system

5. PROGRAMMING

Front-end

Functionality:

- User input
- User display
- Human-Computer Interaction

Used technologies:

- HTML/CSS
  - Jquery

Faced challenges:

- Modals

Solved problems:

- Connection Front-end with Back-end

Estimation:

- Low complexity
- High quality

Back-end

Functionality:
- Data requests
- User authentication
- Hardware readout
- Data forwarding

Used technologies:
- Rust
  - Rocket
  - Sha2
  - chrono
  - postgres
  - hex
  - ...

Faced challenges:
- Rust
- Database
- Authentication

Solved problems:
- All of the above

Estimation:
- High complexity
- Average quality

Database

Functionality:

- Storing security data
- Storing user data

Used technologies:

- Postgresql

Faced challenges:

- Accessing database

Solved problems:

- All of the above

Estimation:

- Low complexity
- High quality

Hardware

Functionality:
- Speed calculation
- ~~light detection~~

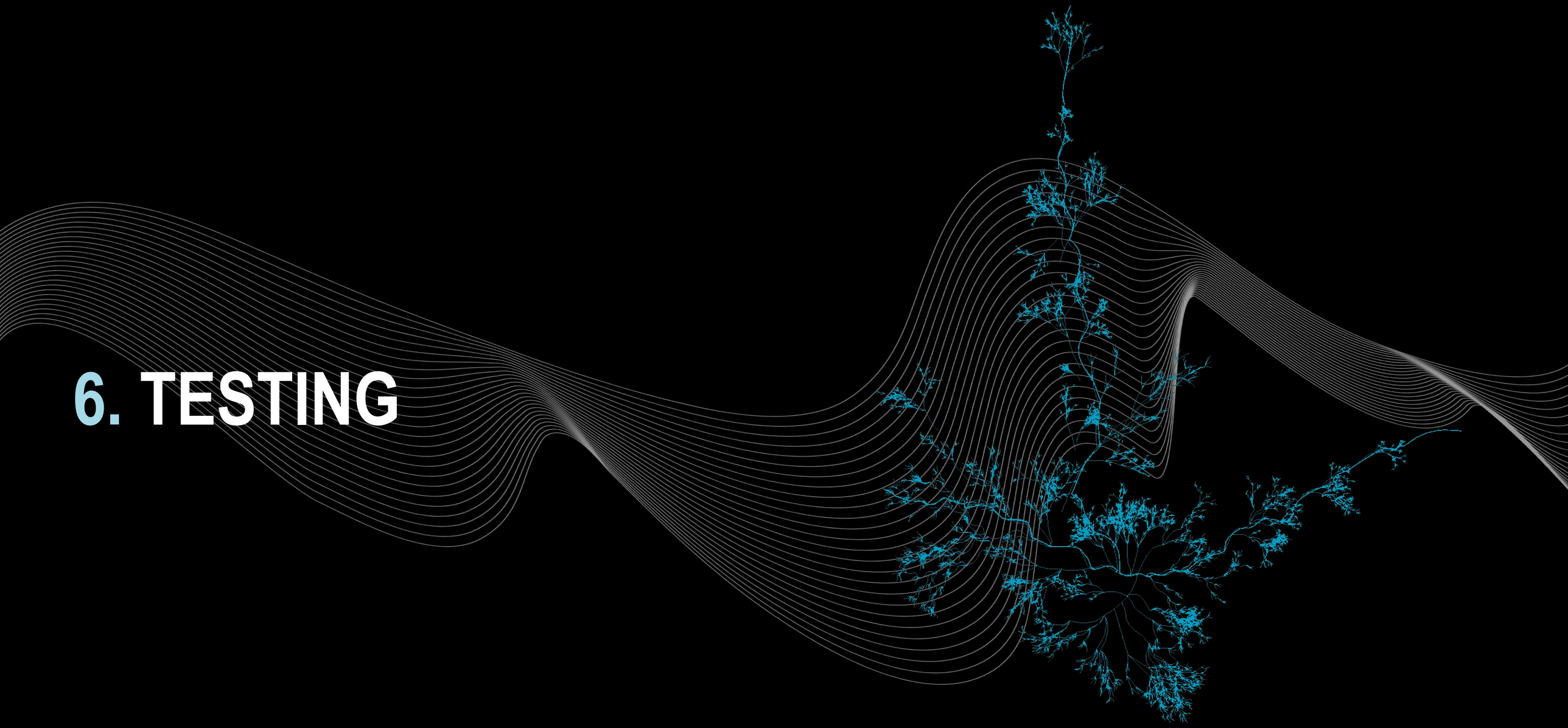
Used technologies:
- Rust
  - gpio

Faced challenges:
- Sensor Flutter
- Unreliable connection
- On-screen keyboard

Solved problems:
- Sensor Flutter

Estimation:
- Low complexity
- Medium quality

6. TESTING



API Testing

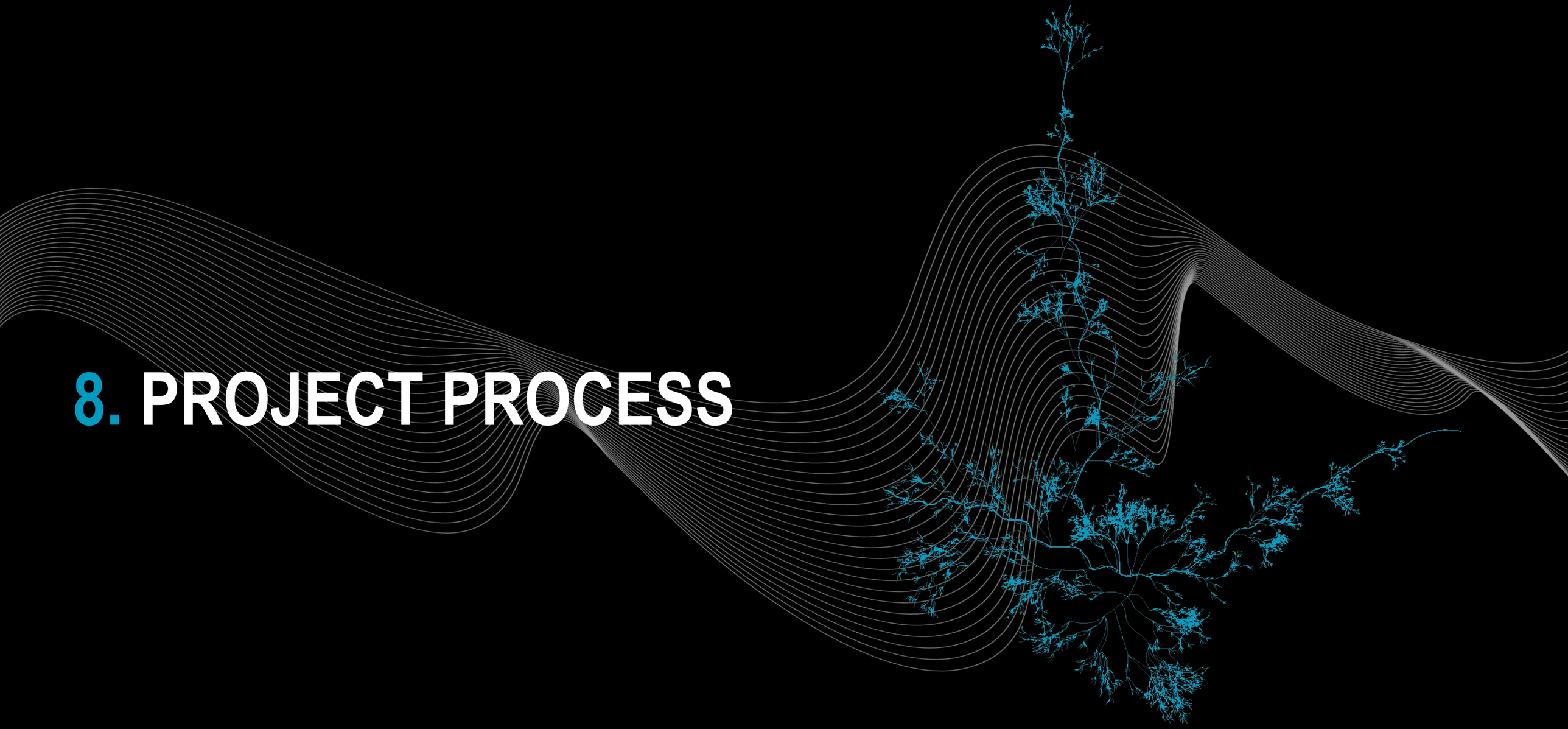
Hardware Testing

Authentication Testing

System Testing

7. FINAL PRODUCT

8. PROJECT PROCESS



Conflict resolution:

- Warning
- Talk with TA’s

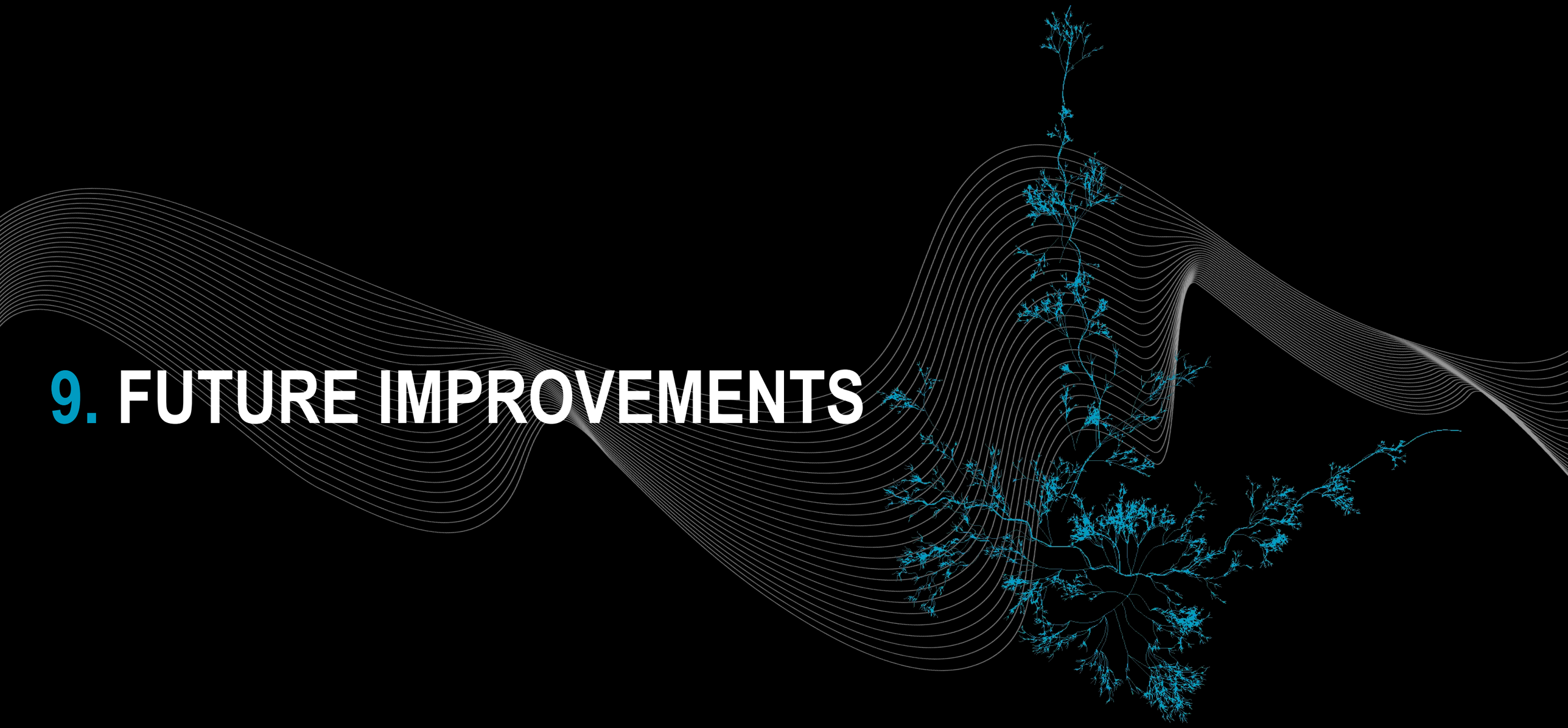
Motivation:

- Medium
- Other priorities

Communication:

- Medium but functional

9. FUTURE IMPROVEMENTS



Additional casing

Better clamp

Higher quality sensor

On screen keyboard

9. Q&A