Hardware:

1. LED Glove
   1. Overview of the LED glove technology.
   2. Application in gesture tracking and interaction.

2. LEDs

* Functionality and characteristics of LEDs.
* Usage in gesture tracking (e.g., flicker patterns for tracking movement).

3. Smartphone Integration

* Role of smartphones in capturing LED movements.
* Integration of rolling shutter camera technology.

4. Rolling Shutter Camera

* **Definition and Functionality**
  + Captures images line by line from top to bottom.
  + Offers low read noise and high-speed data streaming (>50 fps).
* **Advantages**
  + Ideal for high-speed applications without light synchronization.
* **Challenges**
  + Causes spatial distortion with fast-moving objects (skewing/stretched effects).
  + Can be leveraged to infer speed and direction of movement.

5. 3D Mapping

* **Combination of Techniques**
  + Use of optical flow for 2D movement tracking.
  + Tracking size changes to infer relative depth.
* **Approach**
  + Reliable for slower movements.
  + Accurate reconstruction of the LED’s 3D trajectory in space.
* **Utilization of Distortion**
  + Rolling shutter distortion can provide insights into the LED's trajectory.
* **LED Flicker Patterns**
  + Detection of flicker patterns for enhanced tracking of movement.

6. Software Development Frameworks

* **Camera Frameworks**
  + **CameraX**
    - Simplified implementation with built-in lifecycle management.
  + **Camera2 API**
    - Offers greater control for tuning exposure, frame rate, and resolution.
* **Optical Flow Calculation**
  + **OpenCV Integration**
    - Easily integrates into Android apps via the OpenCV SDK.
  + **Optical Flow Algorithms**
    - **Farneback Optical Flow:** For dense motion tracking.
    - **Lucas-Kanade:** For lighter tracking of fewer points.
* **Apparent Size Tracking**
  + **OpenCV Features**
    - Contour detection and blob detection for monitoring size changes.
    - **SimpleBlobDetector:** Detects blobs, useful for isolating the LED.
    - **Contour Detection:** Tracks based on brightness and shape.
* **3D Representation**
  + Combination of 2D data for generating a 3D trajectory.
  + **Rendering Frameworks**
    - **Three.js:** For web-based rendering of 3D paths.
    - **OpenGL ES:** For native 3D rendering in Android.

7. Additional Development Tools

* **Android Studio**
  + Standard development environment for Android app development.
* **NumPy and SciPy**
  + Can be used via Chaquopy or other Python bridges for advanced data processing.

8. General Development Structure

* **Set up Camera Feed**
  + Use CameraX or Camera2 to capture frames.
* **Implement Optical Flow**
  + Integrate OpenCV for tracking LED's 2D movement.
* **Track Size Changes**
  + Use blob detection or contour analysis to monitor LED size variations.
* **Calculate 3D Trajectory**
  + Combine 2D movement with size data for depth inference.
* **Visualize in 3D**
  + Render LED path using OpenGL ES or web-based 3D engines like Three.js.