WAACEYBOARD: Technical Documentation

System Overview

WAACEYBOARD is a wearable Augmentative and Alternative Communication (AAC) device utilizing **eye-tracking**, **head movement recognition**, and **hover-based selection** for text input and speech output. The system consists of two primary components:

- **Low-Level System (Raspberry Pi Zero 2 W):** Processes eye-tracking data and relays positional and sensor data to the user interface.
- **High-Level System (Google Glass running Android 8.1):** Displays the interface, processes gaze-based selection and integrates text-to-speech (TTS) functionality.

---

1. Low-Level System (Raspberry Pi Zero 2 W)

1.1 Eye-Tracking & Image Processing

- **Camera Input:** Up to 200 fps (hardware), **limited by processing power to ~25-30 fps**.
- **Processing Steps:**
    1. Convert frames to **grayscale**.
    2. Apply **edge detection (Canny) + adaptive thresholding.**
    3. Detect pupil using **ellipse fitting (Hough Transform) & Starburst Algorithm**.
    4. **Kalman Filter (2D):** Smooths eye trajectory, ignoring NaN values (blinks) and extrapolating based on past steps (e.g., t-1, t-5, t-9).
- **Output Format:** Serial data tuple (x, y, timestamp) sent to Google Glass via **Bluetooth (SPP Profile)**.

1.2 IMU Sensor Data

- **Captures head orientation, tilt, and acceleration.**
- **Format:** (gyro_x, gyro_y, accel_z) transmitted alongside eye coordinates.

---

2. High-Level System (Google Glass, Android 8.1)

2.1 UI Rendering & Gaze-Based Selection

- **Interface Updates:** Matches Bluetooth input rate (~30 Hz).
- **Calibration Process:** Samples four corners, averages positions, and defines coordinate mappings.
- **Selection Algorithm:**
    1. Uses **K-D Tree Search** or **Spatial Hashing** for nearest neighbor lookup.
    2. When gaze remains within an element boundary, **timer-based hover selection** is initiated.
    3. **Visual Feedback:**
        - **Progressive Ring (arc):** A circular ring fills **clockwise** around the selected element over **1.5 seconds**.
        - **Color Gradient:** Background transitions from **black to yellow** during selection.

2.2 Keyboard Functionality

- **Four Primary Sections:**
    - **Vowels** (5 most common vowels)
    - **Common Consonants** (7 frequent consonants)
    - **Medium Consonants** (7 intermediate consonants)
    - **Rare Consonants** (7 least frequent consonants)
- **Selection Process:**

1. Initial quadrant selection → refined letter selection (nested radial layout).
2. **Hover-based confirmation** for letter entry.
3. TTS activation upon full word confirmation.

2.3 Special Actions

- **NUM/CHAR Toggle:** Switches between numeric and alphabetic mode.
- **Delete (X)/ No:** Delete the last input; When the text string is empty, output "No".
- **Space/ Confirm (✓)/ Yes:** Add space to the text string; Confirms selection and sends text to the TTS module; When the text string is empty, output "Yes".
- **Exit:** Return to the main interface.

---

3. Text-to-Speech (TTS) Integration

- **Library:** Google TTS (gTTS, Python)
- **Process:**
  - Converts finalized text input into an MP3 file.
  - Automatically plays the generated speech.

---

4. Future Considerations

- **Expanding Noise Models in Kalman Filtering:** Explore **non-Gaussian models** for improved accuracy.
- **Enhanced UI Algorithms:** Experiment with **machine learning-based gaze prediction** for more fluid interactions.
- **Customized TTS Model:** Develop a lightweight, patient-specific ML-based TTS system trained on the patient's voice data, allowing for fast and natural speech synthesis.
- **Edge Computing for Real-Time Processing:** Implement **lightweight neural networks** on the Raspberry Pi Zero 2 W to improve processing speed while minimizing energy consumption.
- **Multimodal Input Integration:** Combine eye-tracking with additional sensor modalities (e.g., EEG signals or muscle activity) to enhance input accuracy and robustness.
- **Improved Hardware Compatibility:** Explore alternative hardware platforms beyond Google Glass to ensure long-term support and performance scalability.
- **Adaptive UI Design:** Introduce dynamic interface adjustments based on user preferences and real-time environmental feedback to enhance accessibility.

---

5. Suggested Developer Resources

(a) Eye Tracking Algorithms

- **Eye-Tracking:**

  https://www.pyimagesearch.com/

- **OpenCV Eye Tracking (Using Haar Cascades & Dlib):** **https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/**

- **Starburst Algorithm (More accurate pupil detection):** [**https://ieeexplore.ieee.org/document/1221214**](https://ieeexplore.ieee.org/document/1221214)

(b) Kalman Filtering

- **Understanding Kalman Filters (Stanford):** [**https://stanford.edu/class/ee363/lectures/kalman.pdf**](https://stanford.edu/class/ee363/lectures/kalman.pdf)
- **Python Implementation (FilterPy):** [**https://filterpy.readthedocs.io/en/latest/**](https://filterpy.readthedocs.io/en/latest/)

(c) Google Glass Development

- **Google Glass API (Android 8.1):**

  https://developers.google.com/glass/

- **Google Glass Development Kit:** [**https://developers.google.com/glass/develop/gdk/**](https://developers.google.com/glass/develop/gdk/)
- **Bluetooth Communication on Android:** [**https://developer.android.com/reference/android/bluetooth/BluetoothSocket**](https://developer.android.com/reference/android/bluetooth/BluetoothSocket)

(d) UI Optimization

- **KD-Tree for Nearest Neighbor Search (Scipy): https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html**

Disclaimer: This document is edited with ChatGPT.