**Research how data gathered will be processed and stored : BNO055**

**1. Overview**

This document outlines the methodology for processing data from the BNO055 sensor connected to the ESP8266. The BNO055 sensor outputs various types of orientation and motion data, which will be processed and stored in a SQL database initially, with plans for cloud storage in AWS and PubNub.

**2. Data from the BNO055 Sensor**

The BNO055 provides the following data outputs:

* Absolute Orientation (Euler Vector, 100Hz)
* Absolute Orientation (Quaternion, 100Hz)
* Angular Velocity Vector (100Hz)
* Acceleration Vector (100Hz)
* Magnetic Field Strength Vector (20Hz)
* Linear Acceleration Vector (100Hz)
* Gravity Vector (100Hz)
* Temperature (1Hz)

**3. Data Processing**

Data from the BNO055 sensor will be processed as follows:

* **Data Collection:** Sensor data will be collected at specified intervals (every minute or two). Each reading will include orientation, acceleration and angular velocity, (maybe magnetic field strength).
* **Filtering and Calibration:** Raw data will undergo filtering to remove errors and calibration to ensure accuracy.
* **Data Structuring:** Processed data will be formatted in a structure suitable for SQL storage.

**4. Data Storage**

**Local Storage in SQL:**

* Processed data will be stored in a local SQL database (e.g., SQLite, MySQL).
* Tables will be created to store different types of sensor data, with appropriate data types for each column.

**Cloud Storage:**

* **Initial Local Storage:** Data will be collected and stored in the SQL database on the local device.
* **Cloud Storage with AWS and PubNub:** Data will be periodically sent to AWS for long-term storage. PubNub will be used for real-time data streaming and notifications.

**5. Cron Job Setup**

Cron jobs will be established to automate the following tasks:

* **Read Sensor Data:**
  + **Job:** This cron job will read data from the BNO055 sensor every minute.
  + **Cron Expression:** \* \* \* \* \*
  + **Script:** read\_sensor.py
  + **Function:** Collects and processes sensor data, saving it to the local SQL database.
* **Clean Old Data:**
  + **Job:** This job will run daily to remove old data entries from the SQL database.
  + **Cron Expression:** 0 0 \* \* \*
  + **Script:** clean\_data.py
  + **Function:** Deletes entries older than 30 days to maintain efficient storage.
* **Backup Data to Cloud:**
  + **Job:** This job will back up the latest sensor data to AWS every hour.
  + **Cron Expression:** 0 \* \* \* \*
  + **Script:** backup\_to\_aws.py
  + **Function:** Sends the latest processed data to AWS for long-term retention.
* **Stream Data to PubNub:**
  + **Job:** This job will stream the latest sensor data to PubNub for real-time updates.
  + **Cron Expression:** \* \* \* \* \*
  + **Script:** stream\_to\_pubnub.py
  + **Function:** Publishes the latest sensor readings to a PubNub channel.
* **Generate Reports:**
  + **Job:** This job will generate a daily report of the sensor data at midnight.
  + **Cron Expression:** 0 0 \* \* \*
  + **Script:** generate\_report.py
  + **Function:** Compiles the daily data into a report format, saving it locally or emailing it.

**6. Data Retrieval and Analysis**

* **Local Data Retrieval:** Stored data can be analyzed locally using SQL queries.
* **Cloud Data Retrieval:** Data can be accessed from AWS through API calls for analysis and visualization purposes.

**7. Posture Detection**

**7.1 Posture Analysis**

* **Definition of Posture:** Posture will be evaluated based on the orientation data (Euler angles or quaternion) provided by the BNO055. Key angles to monitor include pitch (forward/backward tilt) and roll (side tilt).
* **Thresholds for Poor Posture:**
  + Define specific thresholds for pitch and roll angles that indicate poor posture. For example:
    - Pitch angle > 15° (leaning forward)
    - Roll angle > 10° (leaning to the side)

**7.2 Prolonged Sitting Detection**

* **Time Monitoring:** Implement a timer to track how long the user remains in a particular posture. If the user remains in a poor posture for more than a specified duration (e.g., 30 minutes), a notification will be triggered.
* **Notification System:** Use PubNub to send real-time notifications to the user’s device, reminding them to adjust their posture or take a break.

**7.3 Data Processing for Posture**

* **Data Collection:** Continuously collect orientation data at 100Hz.
* **Posture Evaluation Logic:**
  + Analyze the collected data to determine if the user’s posture falls within the defined thresholds.
  + If poor posture is detected, log the event in the SQL database with a timestamp.

**7.4 Reporting**

* **Daily Reports:** Include posture data in the daily report generated by the generate\_report.py script. This report will summarize the duration spent in poor posture and provide insights for the user.

**References:**

<https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor>

<https://ieeexplore.ieee.org/document/8952852>

<https://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-on-a-vps>