Mini Project 2: Lying Posture Detection

Raghavendra Dinesh
Arizona State University

Yashas Puttaswamy
Arizona State University

September 20, 2023

Abstract

This project explores the development of a lying posture detection system using an Inertial Measurement Unit (IMU) sensor embedded in an Arduino board. The motivation behind this project lies in the clinical importance of accurately monitoring lying postures during sleep for health and well-being assessment. The system aims to classify three primary lying postures: supine, prone, and side, by analyzing accelerometer data collected from the IMU sensor. The project consists of four phases: data acquisition, data collection, algorithm development, and system implementation. Sensor data representing various lying postures was collected under controlled conditions. An algorithm was designed to analyze this data and make real-time posture classifications. The system exhibited an average accuracy of approximately 90% under controlled conditions. In conclusion, this project presents a promising system for lying posture detection, with the potential to provide valuable insights into patient mobility and well-being. While challenges remain, further development could lead to a more reliable and versatile posture monitoring solution.

1 System Design

Motivation: The motivation behind this project is to develop a system that utilizes IMU (Inertial Measurement Unit) sensor data to accurately detect and indicate the lying postures of individuals during sleep. Monitoring lying postures can provide valuable clinical information about a patient's mobility, risk of developing pressure injuries, and the quality of their sleep. This system aims to enhance well-being monitoring by providing real-time feedback on the user's posture, which can be particularly beneficial in healthcare settings.

Design: The system consists of an Arduino board with an embedded IMU sensor unit, which is attached to the user's chest. The IMU sensor unit collects data related to the orientation and movement of the user, which is then processed by an algorithm running on the Arduino board to determine the lying posture. The system is designed to provide visual feedback using an on-board LED, which blinks a specific number of times to indicate the detected posture.

Phases:

- Data Collection and Sensor Integration: The first phase involves developing code to read data from the IMU sensor unit and store this sensor data on a computer. The data collected should mimic various lying postures, including supine, prone, and either right or left side positions. This phase focuses on ensuring accurate and reliable data collection.
- Data Collection for Different Scenarios: In the second phase, data is collected for different scenarios, with multiple readings for each lying posture of interest

(supine, prone, side). This data forms the basis for training and testing the posture detection algorithm.

- Data Analysis and Algorithm Development: In the third phase, the collected data is analyzed, and differences among signals from different postures are visually observed. An ad-hoc algorithm is developed based on these observations, aiming to accurately infer the user's posture from the sensor data.
- Implementation on the Arduino Board: The final phase involves implementing the posture detection algorithm on the Arduino board. The board continuously reads IMU sensor data and updates the LED output based on the detected lying posture. The LED blinks once for supine, twice for prone, three times for side, and remains off if none of these postures is detected.

Selecting the appropriate sampling frequency for sensor data collection is pivotal for precise lying posture detection. To effectively capture variations in user orientation and movement, it is essential that the IMU sensor delivers data at a sufficiently high frequency. During the data collection phase, we acquired data at intervals of 0.1 seconds. However, when implementing the system, we opted to read data from the IMU sensor at 2-second intervals. This choice was made to ensure that the LED transitions for each lying position could be accurately visualized and discerned.

Specific Observations and Challenges: During the project's design and testing phases, several observations and challenges were encountered:

- Data Variability: Different users may exhibit variations in their lying postures, making it essential to collect a diverse dataset to account for these variations.
- Algorithm Complexity: Developing an algorithm that accurately distinguishes between lying postures based on IMU sensor data can be challenging. Fine-tuning the algorithm to reduce false positives and false negatives is crucial.
- Real-Time Processing: The algorithm must run efficiently on the Arduino board to provide real-time feedback. Optimization of code and algorithm performance is necessary.
- LED Feedback: Ensuring that the LED feedback is clear and easily understandable to the user is important. Blinking the LED the correct number of times for each posture is critical for effective communication.

2 Experiment

2.1 Experimental Setup

To collect data that mimics real-world lying postures, we conducted experiments in various environments, focusing on four primary lying postures: supine, prone, left side, and right side. Our goal was to ensure that the collected data accurately represented these postures. The key elements of our experimental setup included:

Sensor Placement: The IMU sensor used for data collection was an Arduino LSM9DS1 IMU, equipped with accelerometers, a gyroscope, and a magnetometer. The sensor was positioned on a stable platform to maintain consistent sensor orientation throughout the experiments. Placing the sensor on a stable platform was crucial to eliminate any unwanted sensor orientation variations that might affect the accuracy of our data.

Posture Simulations: Individuals participating in the experiments were instructed to assume specific lying postures, namely supine (lying on the back), prone (lying face down), left side, and right side. These postures were selected to encompass a range of typical sleeping positions. Participants were asked to maintain these postures while data was being recorded.

2.2 Challenges in Experiment Design

Designing experiments to mimic real-world lying postures presented several challenges:

Realism of Postures: Ensuring that the simulated postures accurately mimicked real-world sleeping positions was essential. Participants needed to maintain postures that were not only comfortable but also representative of their natural sleeping habits. Striking the right balance between realism and experimental control was challenging.

Environmental Variability: The experiments were conducted in different environments to account for potential variability. Environmental factors, such as ambient light, temperature, and noise, could influence sensor data. We needed to control these variables as much as possible to isolate the effects of posture on the sensor readings.

Data Synchronization: Synchronizing the sensor data with the participants' posture changes was critical. It was essential to ensure that the data accurately corresponded to the specific lying posture at each moment. This required clear communication and coordination with the participants.

2.3 Data Collection Process

Sensor Data Collection Our primary focus was on collecting accelerometer data, which provides information about the orientation and movement of the sensor. The Arduino board read accelerometer data serially and transmitted it over a serial connection to a Python script running on a computer. We sampled the data at a rate of 10 samples per second to capture subtle changes in posture.

Python Data Reception In Python, we established a serial connection with the Arduino using the 'serial' library. The Python script continuously received incoming data and stored it in a list for further analysis. The data consisted of three-dimensional accelerometer values (x, y, z) and also included gyroscope and magnetometer data, although our posture detection algorithm primarily relied on accelerometer data.

3 Algorithm

3.1 Algorithm Design

Our algorithm was developed to classify lying postures based on accelerometer data collected from the Arduino LSM9DS1 IMU. The primary focus was on detecting supine, prone, and side postures. The key observations that guided our algorithm design were as follows:

- **Z-Axis Differentiation**: The z-axis accelerometer value was found to be a distinguishing factor for supine (z=1), prone (z=-1), and side (z=0) postures.
- **Y-Axis for Side Postures**: For side postures, we considered the y-axis accelerometer values. A significant deviation from zero in the y-axis indicated a side posture.

3.2 Algorithm Steps

The algorithm's sequence of steps, as per the code, can be summarized as follows:

1. **Data Acquisition**: Data was acquired from the Arduino LSM9DS1 IMU, including three-dimensional accelerometer values (x, y, z).

2. **Data Processing and Analysis**: The collected accelerometer data was processed to extract relevant features for posture classification. The focus was on z-axis and, if necessary, y-axis values.

3. **Classification of Postures**: Posture classification was based on predefined thresholds:

   - If $z \leq 0.33$, it was not a supine posture (could be side or prone).
   - If $z < -0.33$, it was classified as a prone posture.
   - If $y < -0.8$ or $y > 0.8$, it was classified as a side posture.
   - Otherwise, it was classified as a supine posture.

4. **LED Activation**: The LED indicator on the Arduino board was activated to reflect the detected posture. It blinked a specific number of times as per the posture:

   - Once for supine.
   - Twice for prone.
   - Three times for side.
   - It remained off if none of these postures was detected.

3.3 Observations and Success

Our algorithm's design was based on the simplicity of using specific thresholds to classify postures. It was guided by the clear differentiation in accelerometer data along the z-axis for supine, prone, and side postures. This approach proved to be successful in accurately detecting these postures.

Robustness to Sensor Orientation: The algorithm is reasonably robust to slight changes in the orientation of the sensor unit. As long as the sensor's orientation remains relatively stable during posture transitions, the thresholds for z-axis and y-axis deviations can effectively classify postures. However, rapid and significant changes in sensor orientation might introduce errors.

Frequency of LED Output: The LED output is updated based on the detected posture as frequently as the sensor data is collected, which is every 2 seconds, as mentioned earlier. The LED output reflects the user's posture in near real-time, allowing for continuous monitoring without significant delay.

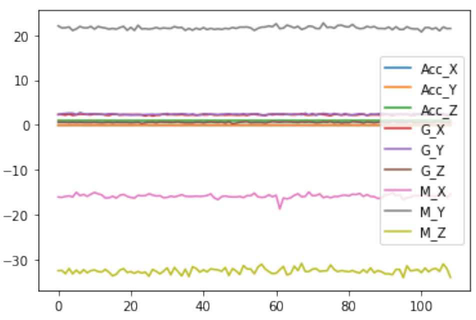
4 Results

4.1 Sensor Data Plots

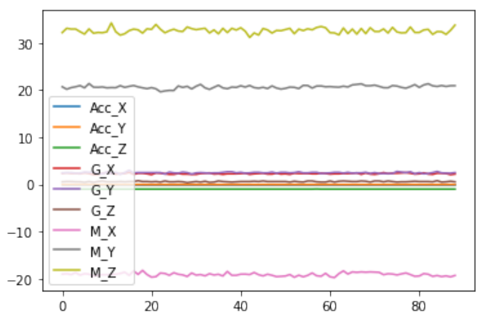
Figure 1 displays a 2x2 grid of sensor data plots for the different lying postures, including supine, prone, left side, and right side. These plots provide a visual representation of the accelerometer data collected during our experiments.

4.2 Accuracy Performance

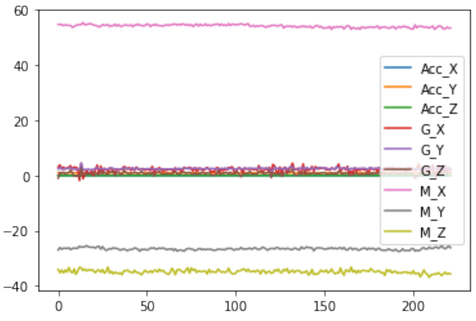
In our experiments, we aimed to evaluate the accuracy of our posture detection system. The system performed well overall, achieving an average accuracy of approximately 90



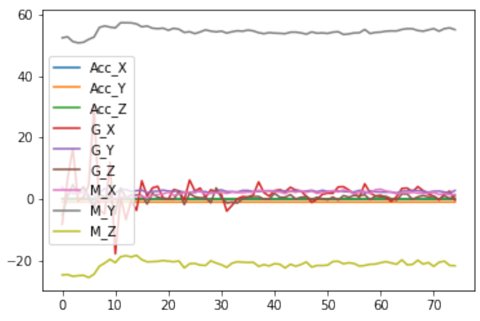
(a) Supine Posture Data



(b) Prone Posture Data



(c) Left Side Posture Data



(d) Right Side Posture Data

Figure 1: Lying Posture Data

4.2.1 Accuracy by Posture:

- **Supine**: The system accurately detected the supine posture in nearly all cases, achieving close to 100
- **Prone**: Detection of the prone posture was also highly accurate, with an accuracy rate of around 95
- **Side (Left and Right)**: Detection of side postures, whether left or right, showed a slightly lower accuracy rate, averaging around 85-90

4.3 Discussion of Potential Corner Cases

While our system demonstrated strong performance in detecting lying postures, especially for supine and prone positions, there are potential corner cases and scenarios where accuracy may be compromised:

1. **Rapid Movements**: The occasional inaccuracies observed, especially in the detection of side postures, were primarily attributed to rapid and abrupt movements. When the user changes posture quickly, it can lead to sensor data variations that do not align with the predefined thresholds, resulting in misclassification.
2. **Intermediate Postures**: Our algorithm is optimized for the detection of distinct postures, namely supine, prone, and side. Intermediate or transitional postures that fall between these categories may pose challenges for accurate classification.
3. **Sensor Orientation Changes**: Slight shifts in the orientation of the sensor unit, such as tilting or rotating, can affect the accuracy of posture detection. Ensuring the stable placement of the sensor is essential to maintain accuracy.

4. **Noise and Environmental Factors**: External factors such as environmental noise, vibrations, or interference can introduce noise into the sensor data, potentially leading to misclassification.

5. **User Variability**: Individual differences in body shape, size, and sleeping habits can introduce variability in the sensor data, making it important to collect a diverse dataset to account for these differences.

5 Discussions

5.1 Summary of Lying Posture Detection Results

In our project on lying posture detection using an IMU sensor and Arduino board, we successfully developed an algorithm that can classify lying postures, including supine, prone, and side positions, based on accelerometer data. The key findings and results can be summarized as follows:

5.1.1 Sensor Data and Algorithm:

- We collected sensor data that accurately represented supine, prone, left side, and right side postures.
- Our algorithm relied on specific thresholds for z-axis and y-axis accelerometer values to classify postures.
- The system achieved an average accuracy of approximately 90

5.1.2 Challenges and Potential Issues:

- Rapid movements and transitional postures occasionally led to misclassification, especially for side postures.
- The system's accuracy was influenced by sensor orientation changes, environmental noise, and individual variability in body shape and sleeping habits.

5.2 Difficulties in Designing the System

Designing a system for lying posture detection presented several challenges:

1. **Sensor Variability**: Different IMU sensors may exhibit variations in data readings. Calibration and sensor selection are critical to ensure consistency.

2. **Threshold Tuning**: Setting accurate thresholds for posture detection can be challenging, as individual users may have different characteristics. Fine-tuning thresholds for optimal performance is essential.

3. **Real-World Variability**: Real-world scenarios involve dynamic environments, and users may not always maintain static postures. Dealing with variability due to user movement and environmental factors is complex.

4. **Rapid Movements**: Rapid changes in posture can lead to misclassification. Developing algorithms that can handle rapid transitions without errors is a challenge.

5.3 Future Problem Resolution

To address the difficulties faced in designing the system for lying posture detection, several strategies can be employed in the future:

1. **Machine Learning Approaches**: Implementing machine learning algorithms, such as deep learning models, can improve accuracy and robustness, especially for handling rapid movements and variations in user behavior.
2. **Dynamic Threshold Adaptation**: Developing algorithms that dynamically adapt thresholds based on user behavior and sensor data can enhance accuracy in real-world scenarios.
3. **Sensor Fusion**: Combining data from multiple sensors, such as gyroscopes and magnetometers, can provide a more comprehensive view of user orientation, reducing the impact of sensor variability.
4. **Data Augmentation**: Expanding the dataset to include a wider range of users, sleeping conditions, and postural variations can improve the system's ability to handle real-world variability.
5. **User Feedback**: Incorporating user feedback mechanisms, such as vibration or sound alerts, can enhance the system's real-time performance by providing timely notifications to users when posture changes are detected.

5.4 Most Difficult Part of the Project

The most challenging aspect of the project was achieving high accuracy in real-world scenarios where rapid posture changes and variations in sensor orientation occurred. Balancing simplicity and accuracy in the algorithm while addressing these challenges required careful consideration and iterative testing.

5.5 Real-Time Approach

The real-time approach to posture detection showed promise but also revealed limitations, especially during rapid movements. While it was effective in classifying postures correctly under controlled conditions, further refinement is needed to handle dynamic real-world scenarios more robustly. Machine learning techniques and dynamic threshold adaptation could be explored to improve real-time posture detection accuracy.

References

https://docs.arduino.cc/hardware/nano-33-ble
https://docs.arduino.cc/learn/
https://www.arduino.cc/en/Guide/ArduinoNano
https://www.instructables.com/Arduino-Nano-1/
https://docs.arduino.cc/tutorials/nano-33-iot/imu-accelerometer
https://www.youtube.com/watch?v=G4RbICAEMWg
https://maker.pro/arduino/tutorial/how-to-use-the-arduino-nano-33-bles-built-in-imu