AI-Powered Sensor Dashboard Setup Guide

Overview

This system integrates Arduino sensors (AD8232 ECG + MPU6050 IMU) with machine learning models for real-time activity recognition and cardiac health monitoring.

Dataset Analysis

1. UCI HAR Dataset (Activity Recognition)

- **Features**: 561 time and frequency domain features
- **Window Size**: 128 samples (2.56 seconds at 50Hz)
- **Activities**: 6 classes
    1. WALKING
    2. WALKING_UPSTAIRS
    3. WALKING_DOWNSTAIRS
    4. SITTING
    5. STANDING
    6. LAYING
- **Sensors Used**: 3-axis accelerometer + 3-axis gyroscope
- **Model**: Random Forest (300 estimators)
- **Expected Performance**: ~95% accuracy

2. MIT-BIH ECG Dataset (Cardiac Analysis)

- **Signal**: Single-lead ECG
- **Sampling Rate**: 360 Hz
- **Window Size**: 10 seconds (3600 samples)
- **Model**: 1D CNN (PyTorch)
    - 3 Conv1D layers (16→32→64 filters)
    - BatchNorm + ReLU + MaxPool
    - Binary classification: Normal (0) vs Arrhythmia (1)
- **Labels**:
    - 0: All beats in window are Normal ('N')
    - 1: Any non-normal beat in window (V, A, etc.)
- **Features Extracted** (for HRV analysis):
    - **Per Beat**: RR intervals, HR (BPM), SQI
    - **Per Window** (60s with 10s step):
        - HR statistics (mean, min, max)
        - HRV metrics (SDNN, RMSSD, pNN50)
        - Rhythm irregularity flag
        - Signal Quality Index (SQI)
- **Processing**: Bandpass filter (0.5-40 Hz) + per-window normalization

Signal Processing Pipeline

Activity Recognition (MPU6050)

```
Raw IMU Data (128 samples)
  ↓
Extract 561 features:
  - Time domain: mean, std, median, max, min, range, percentiles, energy
  - Frequency domain: FFT magnitude statistics
  - Cross-axis: magnitude, correlations
  ↓
Random Forest Classifier
  ↓
Activity + Confidence
```

ECG Analysis (AD8232)

```
Raw ECG Signal (10 seconds, 3600 samples)
   ↓
Bandpass Filter (0.5-40 Hz)
   ↓
Per-Window Normalization
  (mean=0, std=1)
   ↓
1D CNN Model (PyTorch)
  Conv1D(1→16→32→64)
  + BatchNorm + ReLU + MaxPool
   ↓
Binary Classifier
   ↓
Arrhythmia Probability (0-1)
  + Detection (threshold=0.5)

PARALLEL PATH for HRV:
   ↓
R-peak Detection (NeuroKit2)
   ↓
RR Interval Calculation
   ↓
Compute Metrics:
  - Heart Rate (60/mean(RR))
  - SDNN (std of RR intervals)
  - RMSSD (sqrt of mean squared successive differences)
  - pNN50 (% of RR differences > 50ms)
   ↓
Irregularity Detection (RMSSD > 50 OR pNN50 > 20)
```

Key Corrections Made

1. Feature Extraction Alignment

The code now extracts 561 features matching UCI HAR format:

- 40 features per sensor axis (6 axes × 40 = 240)
- 15 FFT features per axis (6 × 15 = 90)
- Cross-axis features (magnitude, etc.)
- Padded/truncated to exactly 561 features

2. ECG CNN Model Integration

- **Model Architecture**: Exact match to training code
  - 3 Conv1D layers with BatchNorm
  - Kernel size 7, padding 3
  - 16→32→64 filters with MaxPool
  - AdaptiveAvgPool + Linear classifier
- **Preprocessing**:
  - Bandpass filter (0.5-40 Hz)
  - Per-window normalization (mean=0, std=1)
  - Tensor shape: (1, 1, 3600)
- **Output**:
  - Sigmoid probability (0-1)
  - Binary detection using threshold 0.5

3. ECG HRV Processing

- Proper bandpass filtering (0.5-40 Hz) before R-peak detection
- Uses NeuroKit2's `ecg_process()` for robust R-peak detection
- Implements HRV metrics exactly as in MIT-BIH processing code
- Signal Quality Index checks for physiologically plausible RR intervals (0.3-2.0s)

4. Sampling Rate Handling

- Activity recognition expects ~50Hz (20ms per sample)
- ECG CNN expects 360Hz for MIT-BIH compatibility
- Arduino sends at 20Hz (50ms intervals) - adequate for activity
- ECG accumulates 3600 samples (10 seconds) for CNN inference

Installation

1. Install Python Dependencies

```bash
pip install -r requirements.txt
```

2. Project Structure

```
project/
├── app.py                    # Main Flask application
├── requirements.txt          # Python dependencies
├── agents/
│   ├── __init__.py           # Agent module init
│   ├── patient_agent.py      # Patient baseline learning
│   └── clinical_agent.py     # Clinical risk assessment
├── models/
│   ├── activity_rf_ucihar.pkl    # Activity recognition model
│   ├── ecg_cnn_win10s_binary.pt  # ECG arrhythmia model
│   ├── clinical_agent_model.joblib     # Clinical risk model
│   └── clinical_agent_features.joblib  # Feature names
├── data/
│   ├── patient_state.json    # Persistent patient baseline (auto-created)
│   └── clinical_profile.json # Clinical profile (auto-created)
└── templates/
    ├── index.html            # Main dashboard
    └── clinical.html         # Clinical profile page
```

3. Prepare the Models

Place your trained models at:

```
models/
├── activity_rf_ucihar.pkl        # Random Forest for activity (scikit-learn)
├── ecg_cnn_win10s_binary.pt      # PyTorch CNN for arrhythmia
├── clinical_agent_model.joblib   # Logistic regression for clinical risk
└── clinical_agent_features.joblib # Feature names list
```

Activity Model: Trained on UCI HAR dataset with 561 features
ECG Model: PyTorch checkpoint with keys: `model_state`, `threshold`, `win_sec`, `fs`
Clinical Model: Logistic regression on 14 clinical features

4. Arduino Setup

Upload the Arduino code to your Nano with:

- **AD8232**: OUTPUT→A0, LO+→D10, LO-→D11
- **MPU6050**: SDA→A4, SCL→A5

5. Configure Serial Port

Edit `app.py` line 17:

```python
SERIAL_PORT = 'COM3'  # Windows
# SERIAL_PORT = '/dev/ttyUSB0'  # Linux
# SERIAL_PORT = '/dev/cu.usbserial-*'  # Mac
```

Running the Application

```bash
python app.py
```

Then open: `http://localhost:5000`

Understanding the Output

Activity Recognition

- Updates every 2 seconds
- Requires 128 samples (~6.4 seconds of data at 20Hz)
- Shows confidence percentage and predicted activity

ECG Arrhythmia Detection (CNN)

- Updates every 2 seconds
- Requires 3600 samples (10 seconds at 360Hz)
- Shows probability (0-100%) and binary detection
- Threshold: 50% (configurable in model checkpoint)
- **DETECTED**: Potential arrhythmia present
- **NORMAL**: No arrhythmia detected

ECG HRV Analysis

- Heart Rate: Instantaneous BPM from RR intervals
- RMSSD: Short-term HRV variability (higher = more variability)
- SDNN: Overall HRV (standard deviation of RR intervals)
- Rhythm Status:
  - NORMAL: Regular heart rhythm (HRV-based)
  - IRREGULAR: Potential irregularity (RMSSD>50 OR pNN50>20)
- ECG Quality: % of physiologically plausible RR intervals

Typical Values

- **Resting HR**: 60-100 BPM
- **RMSSD**: 20-50 ms (higher in athletes)
- **SDNN**: 30-100 ms
- **Good ECG Quality**: >80%

Troubleshooting

Activity showing UNKNOWN

- Wait for 128 samples to accumulate (~6-7 seconds)
- Check that MPU6050 is connected and sending data
- Verify model file exists: `models/activity_rf_ucihar.pkl`

Arrhythmia always showing 0% or NORMAL

- Need 10 seconds of continuous ECG data (3600 samples)
- Ensure AD8232 leads are properly connected
- Check that ECG model loaded successfully
- Verify model file: `models/ecg_cnn_win10s_binary.pt`
- Check PyTorch installation and device compatibility

Heart Rate showing 0 or --

- Ensure AD8232 leads are properly connected to body
- Check LO+ and LO- pins for lead-off detection
- Need at least 3 R-peaks detected (few seconds of good signal)

Low ECG Quality

- Improve electrode contact
- Reduce motion artifacts
- Check for electrical interference

Models not loading

- Check file paths and ensure models exist
- Activity model: scikit-learn .pkl file
- ECG model: PyTorch .pt checkpoint
- Verify PyTorch version compatibility (2.0+)
- Check model architecture matches training code

API Endpoints

- `GET /` - Main dashboard
- `GET /api/data` - Raw sensor data
- `GET /api/predictions` - ML predictions
- `GET /api/latest` - Latest sensor reading
- `GET /api/status` - Connection status
- `GET /api/clear` - Clear stored data

Notes

- The system processes data in real-time with minimal latency
- **Two independent ML models** run in parallel:
    1. **Activity Recognition**: Random Forest on IMU data
    2. **Arrhythmia Detection**: CNN on ECG waveforms
- Activity predictions use sliding windows for continuous monitoring
- ECG CNN requires 10 seconds of stable signal for reliable detection
- HRV analysis provides additional cardiac health metrics
- Both models update every 2 seconds
- Data is stored in memory (last 3600 points = 10 seconds at 360Hz)
- GPU acceleration available if CUDA is installed (PyTorch will auto-detect)