Quick Start Guide - AI Sensor Dashboard

🚀 5-Minute Setup

Step 1: Install Dependencies

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

Step 2: Prepare Model Files

Place your trained models in the `models/` folder:

```
models/
├── activity_rf_ucihar.pkl          # Random Forest (scikit-learn)
├── ecg_cnn_win10s_binary.pt        # PyTorch CNN
├── clinical_agent_model.joblib     # Logistic Regression
└── clinical_agent_features.joblib  # Feature names
```

Step 3: Configure Serial Port

Edit `app.py` line 17:

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

Step 4: Upload Arduino Code

Upload `arduino_sensor_code.ino` to your Arduino Nano

Step 5: Run the App

```bash
python app.py
```

Open browser: **http://localhost:5000**

📊 Dashboard Overview

Top Status Bar

- **Green dot** = Connected to Arduino
- **Model status**: ✓ = loaded, ✗ = missing

AI Predictions (8 Cards)

1. **Current Activity** - What you're doing now
    - 6 classes: Walking, Sitting, Standing, Laying, Upstairs, Downstairs
    - Confidence bar shows model certainty
2. **Heart Rate** - Real-time BPM
    - Normal: 60-100 BPM (resting)
    - Updates every 2 seconds
3. **HRV (RMSSD)** - Heart rate variability
    - Higher = more variability (good for fitness)
    - Typical: 20-50 ms
4. **HRV (SDNN)** - Overall heart rhythm stability
    - Typical: 30-100 ms
    - Lower in stress, higher in relaxation
5. **Rhythm Status** - HRV-based irregularity detection
    - NORMAL: Regular rhythm
    - IRREGULAR: RMSSD>50 or pNN50>20
6. **ECG Quality** - Signal reliability
    - Green (>70%): Good
    - Orange (40-70%): Medium
    - Red (<40%): Poor
7. **Arrhythmia Detection** - CNN-based abnormality detection
    - Shows probability % and binary detection
    - NORMAL: <50% probability
    - DETECTED: ≥50% probability (potential arrhythmia)
8. **Clinical Risk** - Static medical vulnerability
    - Based on age, conditions, medications
    - Updates only when profile changes
    - Click "Clinical Profile" to configure

Charts (4 panels)

- **ECG Signal**: Raw heart electrical activity
- **Accelerometer**: 3-axis motion (X, Y, Z)
- **Gyroscope**: 3-axis rotation
- **Raw Values**: Current sensor readings

🔧 Troubleshooting

No data appearing?

```bash
# Check serial port
python -c "import serial.tools.list_ports; print([p.device for p in serial.tools.list_ports.comports()])"

# Update SERIAL_PORT in app.py
```

Activity shows UNKNOWN?

- Wait 6-7 seconds for data accumulation
- Check MPU6050 wiring (SDA→A4, SCL→A5)

Arrhythmia shows 0%?

- Need 10 seconds of ECG data
- Check AD8232 electrode placement
- Verify model file exists

Models not loading?

```bash
# Test models
python test_features.py

# Check files exist
ls models/
```

🎯 Expected Performance

Activity Recognition

- **Accuracy**: ~95%
- **Update Rate**: Every 2 seconds
- **Latency**: ~6 seconds (data collection)

Arrhythmia Detection

- **Sensitivity**: Varies by arrhythmia type
- **Update Rate**: Every 2 seconds
- **Latency**: 10 seconds (data collection)
- **Note**: Not medical grade - for research only

HRV Analysis

- **Accuracy**: Depends on R-peak detection quality
- **Update Rate**: Every 2 seconds
- **Requires**: Good electrode contact

📁 Project Structure

```
project/
├── app.py                      # Flask backend
├── requirements.txt            # Python dependencies
├── test_features.py            # Validation script
├── models/
│   ├── activity_rf_ucihar.pkl  # Activity model
│   └── ecg_cnn_win10s_binary.pt  # ECG CNN model
└── templates/
    └── index.html              # Web dashboard
```

🔬 Model Details

Activity Recognition (Random Forest)

- **Input**: 561 features from 128 IMU samples (2.56s window)
- **Output**: 6 activity classes + confidence
- **Framework**: scikit-learn

Arrhythmia Detection (1D CNN)

- **Input**: 3600 ECG samples (10s window at 360Hz)
- **Output**: Binary probability (0=Normal, 1=Arrhythmia)
- **Framework**: PyTorch
- **Architecture**: 3x Conv1D (16→32→64) + BatchNorm + MaxPool

---

⚠️ Important Notes

1. **Not Medical Device**: This is for research/education only
2. **Signal Quality**: ECG requires good skin contact
3. **Motion Artifacts**: Reduce movement for better ECG
4. **Calibration**: May need adjustment for your sensors
5. **Privacy**: Data stays local, not uploaded anywhere

🆘 Getting Help

If models aren't working:

1. Run `python test_features.py` to validate setup
2. Check console output for error messages
3. Verify model file formats match training code
4. Ensure PyTorch/scikit-learn versions compatible

📈 Next Steps

- Adjust detection thresholds in model checkpoints
- Add more activity classes
- Train on your own ECG data
- Implement data logging/export
- Add real-time alerts for abnormalities