STMicroelectronics SensorTile Tutorial

Reference Design: Automated New Gesture Detection, Training, and Classification System

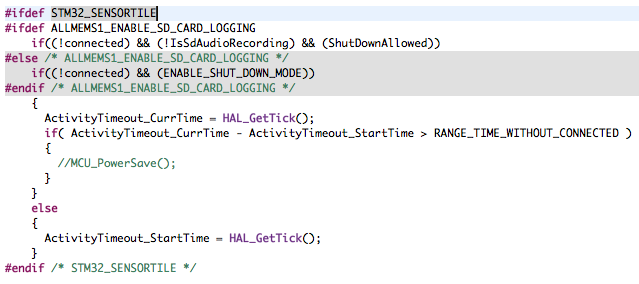
1. Mission Objectives

This reference design focuses on building an automated new gesture detection and training system. In this tutorial, the system begins with a base set of 4 classes and can detect new periodic gestures as they occur, prompt the user to begin collecting data for this new gesture, extract features, and train for this new gesture and update the neural network. This tutorial assumes the prior knowledge with motion classification techniques such as feature extraction and FANN neural network design, but will touch briefly upon feature extraction techniques that help build the base classification set before additional gestures are added.

* 1. List of Required Equipment and Materials

1. 1x STMicroelectronics SensorTile kit
2. 1x STMicroelectronics Nucleo Board
3. 1x Personal Computer with two USB type-A inputs or powered USB hub
4. 1x USB 2.0 A-Male to Micro-B Cable (micro USB cable)
5. 1x USB 2.0 A-Male to Mini-B Cable (mini USB cable).
6. Network access to the internet
   1. Prerequisite Tutorials and Lectures

It is recommended that users have completed and are familiar with the contents of the following tutorials before proceeding:

1. Introduction to SensorTile and the System WorkBench Integrated Development Environment (IDE)
2. Sensor System Signal Acquisition, Event Detection, and Configuration
3. Accelerometer Sensor Systems, Orientation, and Event Detection
4. Introduction to Bluetooth Low Energy (BLE) Interfaces
5. BLE Communication via BlueZ and the GATT Profile
6. Introduction to Motion Data Acquisition via BLE Communication
7. BeagleBone Tutorial: IoT Machine Learning System Reference Design
8. IoT Signal Processing with the LiquidDSP Signal Processing Library
9. Hardware Setup
   1. SensorTile
10. For ease of development and debugging, in main.c of firmware FP-SNS-ALLMEMS1, temporarily comment out the MCU\_PowerSave() function as shown below. This will be the firmware run on the SensorTile for the duration of the tutorial. 
11. Flash the firmware FP-SNS-ALLMEMS1 to your SensorTile according to SensorTile tutorial 8 – STMicroelectronics SensorTileTutorial: Introduction to Motion Data Acquisition via BLE Communication.
12. Power the SensorTile.
    1. BeagleBone
13. Ensure that your BeagleBone has internet access.
14. Install the FANN library as explained in the BeagleBone IoT Machine Learning tutorial.
15. Install the LiquidDSP Signal Processing Library as explained in Lecture 7: IoT Signal Processing with the LiquidDSP Signal Processing Library.
    1. Compiling and Running
16. cd into the directory containing main.c.
17. make main to compile the program.
18. Check the header file main.h to make sure you have ORIG\_TRAIN\_FILE and ORIG\_FANN\_FILE. Subsequent training inputs will be appended to the original train file, which will then be used to generate NEW\_TRAIN\_FILE and NEW\_FANN\_FILE. The main program assumes that you already have a pre-trained FANN network in the form of a .net file to load into the program. See FANN documentation on loading artificial neural networks from file. The default files listed in main.h are provided in the directory:

* motion\_data.dat – Saves raw bytes collected from the SensorTile for parsing; automatically created if it doesn’t already exist
* **train\_orig,csv – Your initial “base set” training data. By default, this contains 40 samples from 4 classes: stationary, walking, running, and jumping. Each data point consists of 12 features collected from each accel axis: RMSx, RMSy, RMSz, corrxy, corrxz, corryz, energyx, energyy, energyz, stddevx, stddevy, stddevz. These features are collected in get\_all\_features() in helpers.c.**
* fann\_motion\_orig.net – The original unmodified saved FANN network trained on the original base set as described above. **Note that both train\_orig.csv and fann\_motion\_orig.net** **will both stay unmodified as new gestures are added to the program so that your base set stays intact.**
* train\_new.csv and fann\_motion\_new.net are both based off train\_orig.csv and fann\_motion\_orig.net and modified as new gestures are added to the system. Both files are created automatically if they do not already exist.

1. Run with ./main. By default, the program begins, it classifies motions using features extracted from accelerometer data, and uses the neural network provided by ORIG\_FANN\_FILE. Once a new gesture is detected, the program will prompt the user before collecting data and automatically training for the new gesture.

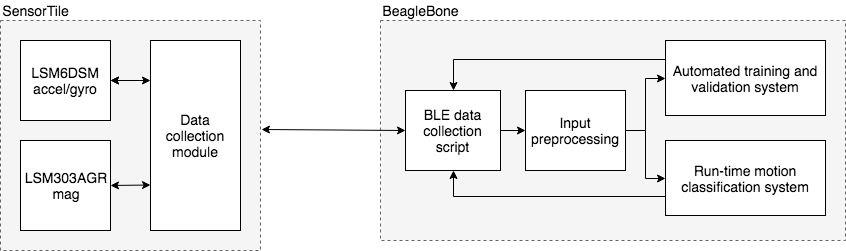


Figure 1. System Architecture

1. System Use and Modification

The system architecture is depicted above in Figure 1. The script collect\_data.sh is run via a forked process from main.c, and remains running for the duration of the main loop. It calls gatttool at 2-second intervals to gather sensor data over BLE from the SensorTile. The main loop awaits a SIGUSR1 signal from collect\_data.sh which signals that the 2-second period of data collection has finished. Once this signal is received, collect\_data.sh waits until the main loop finishes processing the data before collecting another batch of data.

* 1. State Machine

The automated new gesture detection system consists of 3 states: regular run mode, prompt mode, and train mode. Each of these states can be found in the main loop of main.c. In regular run mode, classification is run from the current neural network loaded into the system – a network trained the original training set or an updated network trained on recently added gestures. If a potential new gesture is detected, the system transitions to the prompt mode. In this mode, by default, the program waits until the user presses enter to begin collecting data. However, this can easily be extended to prompting for the user to verify whether or not a new gesture has indeed been detected, and thus the user can decide whether or not to begin data collection and training. If the user decides to continue, regular program execution is paused and the user follows a series of prompts to collect data. Upon conclusion of the data collection phase, the program automatically extracts the necessary features, updates the existing training file, tunes the neural network, and generates a new .net file containing the saved network and parameters. The program then transitions back to the default run mode.

* 1. New Gesture Detection Method

The new gesture detection algorithm is contained in the function detect\_new\_gesture() in helpers.c. The main loop keeps a buffer of size HISTORY\_SIZE (defined main.h) for output MSE and main frequencies along each axis. By default, two conditions have to be satisfied in order for an unfamiliar movement to be identified as a new gesture as opposed to an arbitrary movement:

1. **Misclassification condition:** The system must first be certain that a misclassification has been occurring for a consistent number of frames. This is the first signal of a potential new gesture. By default, inaccurate classification for a single frame is defined as when the mean-squared error between the output score vector and the expected one-hot encoded output exceeds 0.001. If more than HISTORY\_SIZE-1 frames in the window are misclassified, the system concludes that it is dealing with either an arbitrary aperiodic gesture which is likely to surface in between known gestures, or a new periodic gesture which will require training (satisfied by the periodicity condition explained below).
2. **Periodicity Condition:** The main frequency of a user-specified number of sensor axes should remain within a standard deviation of STDDEV\_THRESHOLD and the mean frequency should be above FREQ\_THRESHOLD across all frames in the history buffer. This indicates that the new movements are periodic and are a likely new gesture that can be trained on.

This new gesture detection algorithm can easily be customized. Some suggested starting points:

* Allocate new buffers (in initSensorsBuf() in helpers.c) for other statistics you would like to analyze across time and pass the buffers into detect\_new\_gesture(). For example, you may want to consider standard deviation of the raw signal RMS.
* MSE\_THRESHOLD, STDDEV\_THRESHOLD, and FREQ\_THRESHOLD – can affect sensitivity of system. By default, the system is tuned for more significant exercise gestures such as running, jumping, and kicking.
* Number of sensor axes that must satisfy the periodicity condition
  1. File Descriptions and Key Functions
* dsp.c – contains all feature extraction and filters
* helpers.c – raw BLE data parsing and new gesture detection algorithm
* main.c – main loop, forks new process for collect\_data.sh, contains state machine implementation of mode switching between regular run mode and training mode for new gestures
* main.h – contains all main user-modifiable parameters
* preparetrain.c – contains functions for training and training file formatting
* test.c – standalone file for testing the neural network independently without the full automated gesture detection system
* train.c – standalone file for training the neural network independently without the full automated gesture detection system