人間環境情報ウェアラブルセンシング最終課題

1. 製作したアプリケーションの仕様

『Self State Detector』自己状態検出器

常時複数のセンサ値を取得しつつ，スマホ本体自身の姿勢・状態を，ユーザの入力により学習ののち，多クラス分類器により自動検出（推定）するアプリケーションを作成した．本アプリケーションは学習モード（図 1）と状態検出モード（図 2）の 2 モードを切り替え動作する．

取得情報・センサ	気圧	Sensor.TYPE_PRESSURE	1 次元
	照度	Sensor.TYPE_LIGHT	1 次元
	重力	Sensor.TYPE_GRAVITY	3 次元
入力	センサ値	n 次元浮動小数点精度ベクトル	$n = 1 + 1 + 3 = 5$
出力	状態	m クラスカテゴリカルデータ	$m = 4$（机面，手の中，頭上，耳元）
状態推定手法	統計的分類	k-NN 多クラス分類器	$k = 10$（任意設定）

図 1　製作したアプリケーションの画面表示：学習モード

検出する本体の状態は机面「OnDesk」(図 3)，手の中「InHand」(図 4)，頭上「OverHead」(図 5) 及び耳元「OnEar」とし，対応する教示として机面「Put on desk and press button」，手の中「Hold in hand and press button」，頭上「Raise over head and press button」及び耳元「Put to your ear and press button」の 4 種類を設定した．

図 2　製作したアプリケーションの画面表示：検出モード

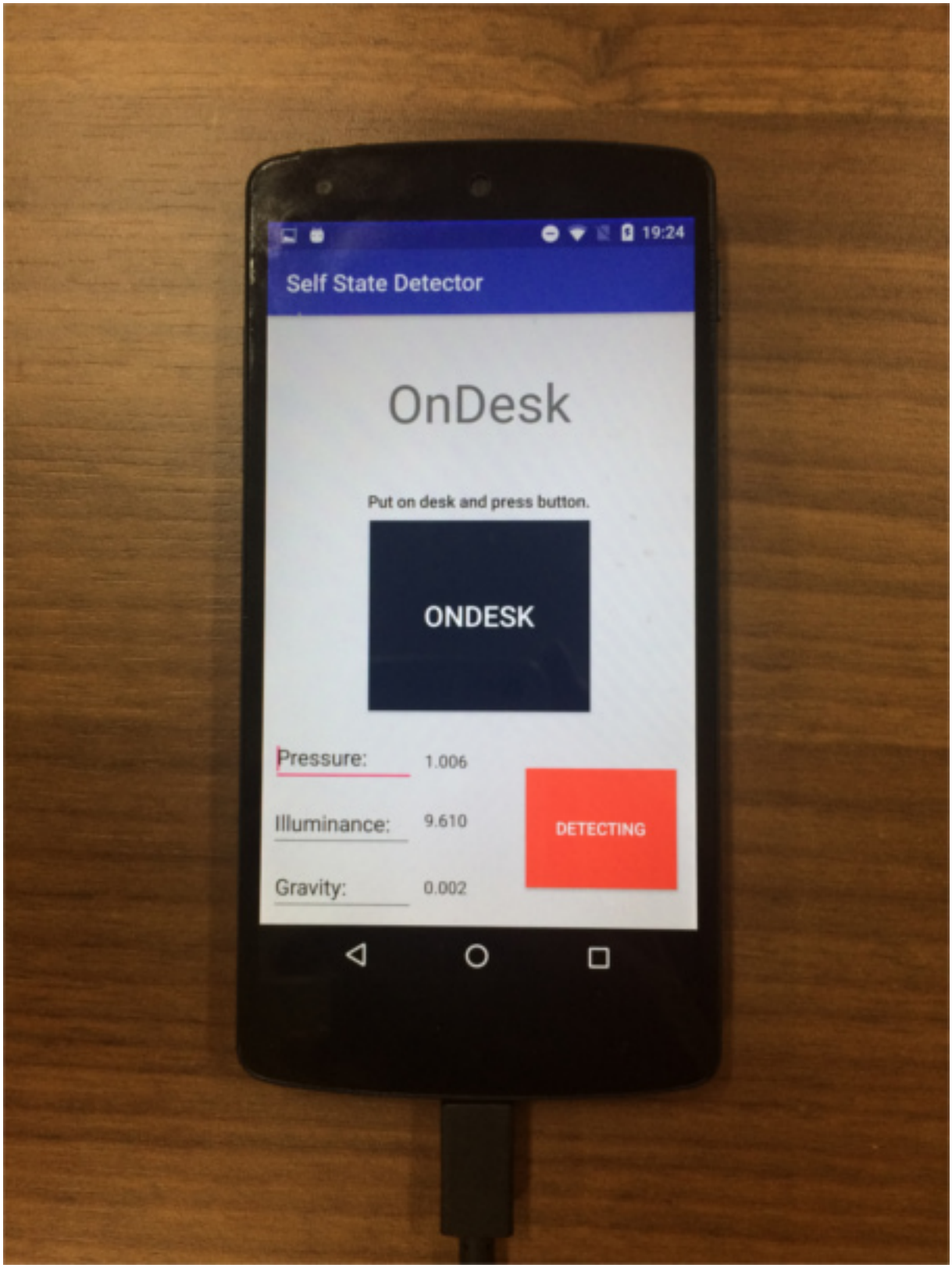

図３　検出モードの実際の動作（机面）

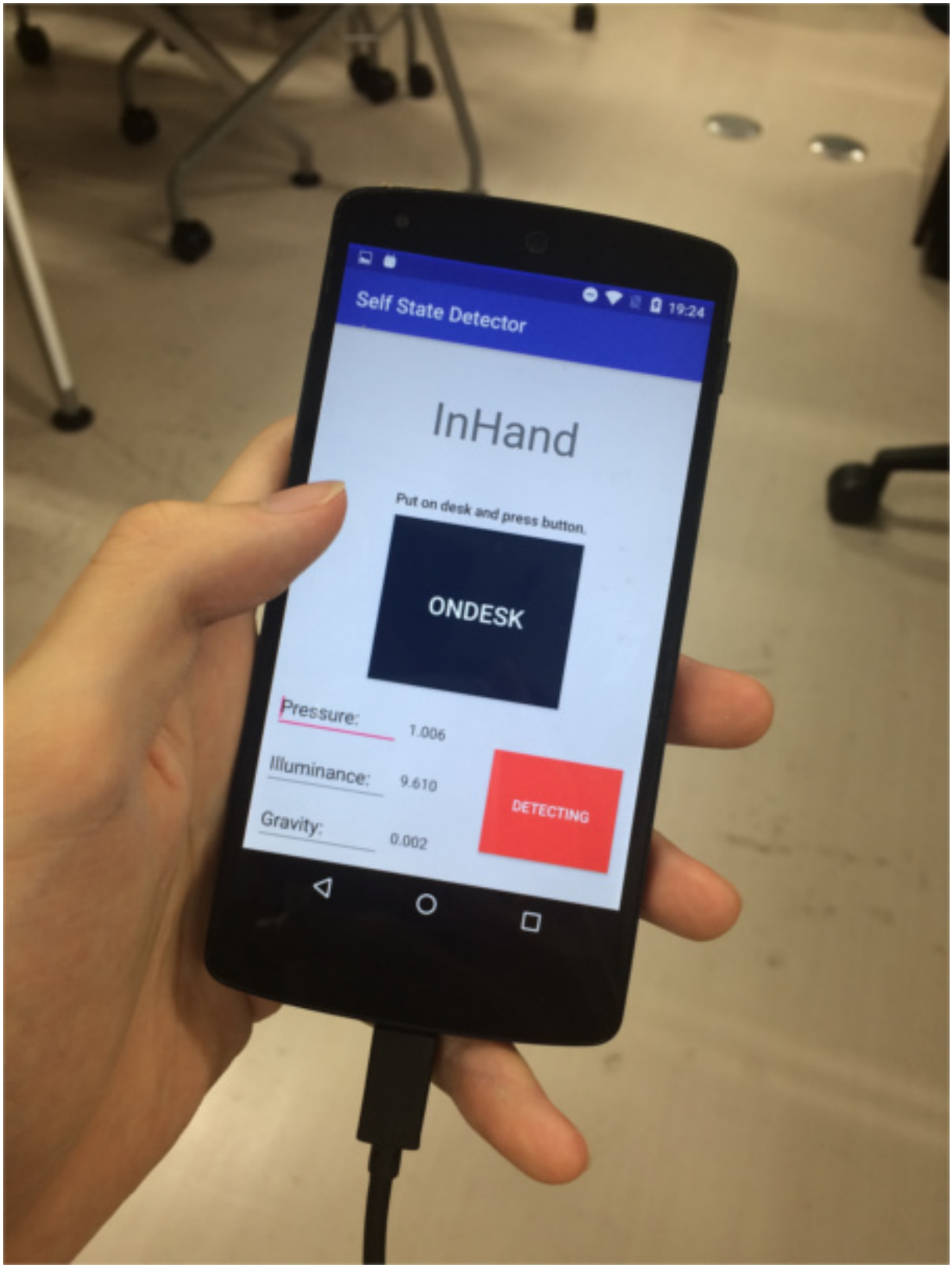

図 4　検出モードの実際の動作（手の中）

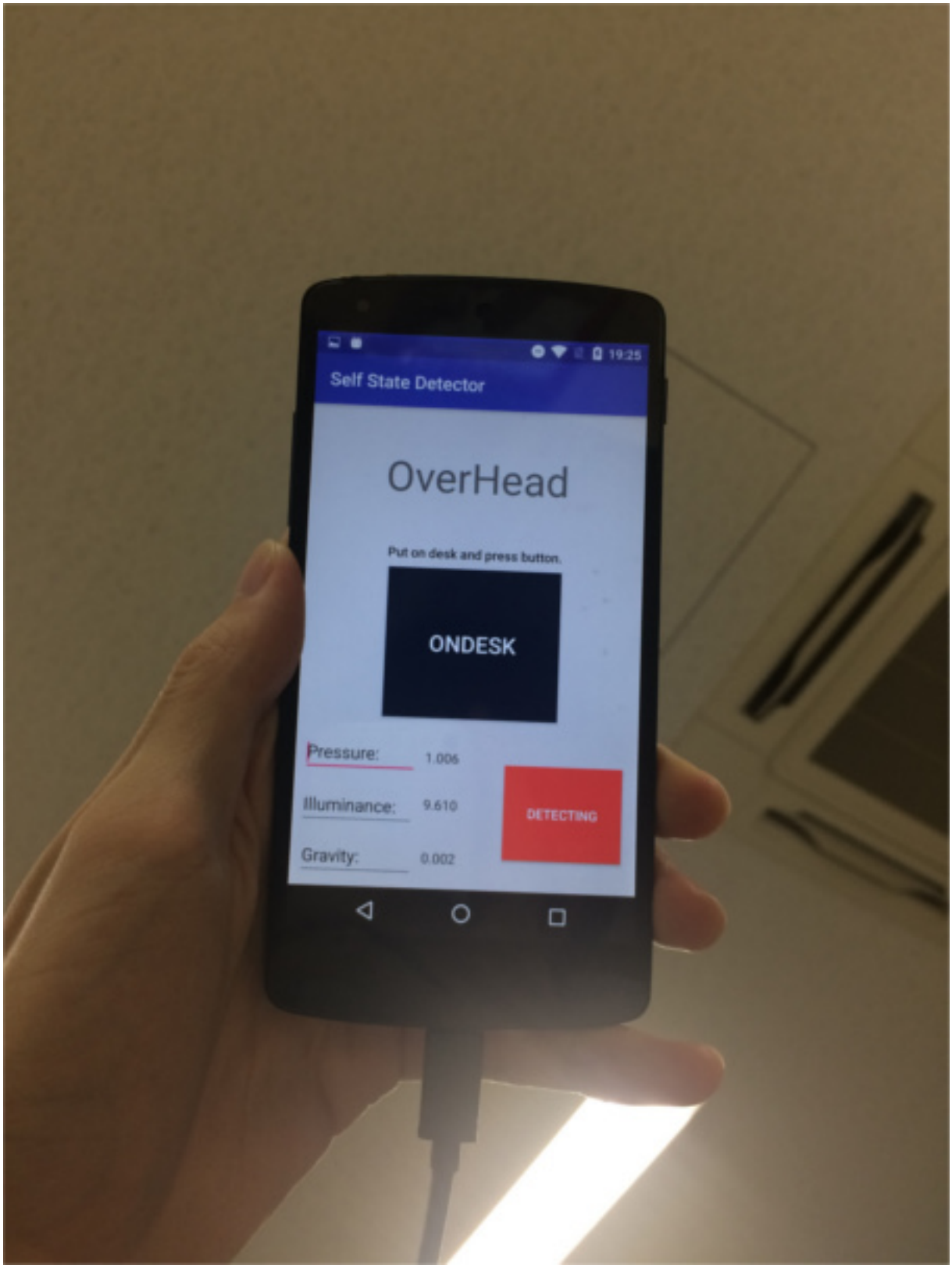

図５　検出モードの実際の動作（頭上）

2. 製作したアプリケーションのアルゴリズム

本アプリケーションは大きく 3 つの部分に分かれる（図 6）．MainActivity.java は UI の描画・制御と k-近傍法による多クラス分類の実行を担う．また，MeasuringService.java により定義される MeasuringService クラスは他の動作と関係なくバックグラウンドで継続的にセンサ値を取得し続ける Service インスタンスとなる．そして，LearningService.java によって定義される LearningService クラスは UI からの操作を受け付けると，取得したセンサ値にラベルを付けて保存する Service インスタンスとなる．

データの保存は GlobalValues.java により定義される GlobalValues クラスのインスタンス”globalValues”に，どのクラスからも setXX() メソッドによって保存・getXX() メソッドによってアクセスできる形で行われる．学習データはさらに ClsModel.java によって定義される ClsModel クラスのインスタンス”clsModel”を GlobalValues クラスのメンバとして持たせることで，clsModel の中にデータ（センサ値）とラベルの配列の形で保存する．

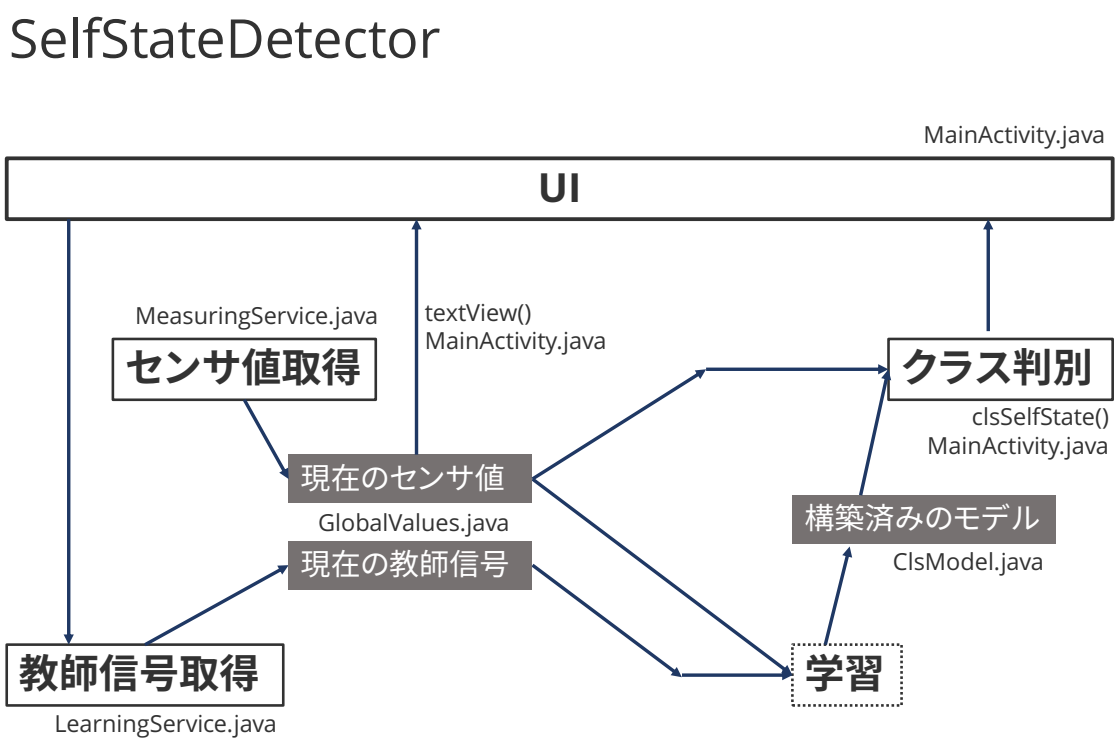

図 6　製作したアプリケーションの構成

MeasuringService クラスは IntentService クラスを継承して定義することで，様々な処理のバックグラウンドでの実行を可能にしている．IntentService クラスは Android においてバックグラウンド処理の実装に用いられるクラスで，UI を伴う実行クラスである Activity クラスに対し UI を伴わない Service ク

ラスを継承している．IntentService クラスは Service クラスにスレッドとハンドラの処理が扱いやすいよう加わっており，IntentService クラスのインスタンスが呼ばれるときハンドラに渡された処理をスレッドが実行し，処理がすべて完了すると自動でスレッドを破棄して終了するようになっている．本アプリケーションでは MeasuringsService インスタンスに SensorManager の取得，各種 SensorListener の取得・SensorManager への登録，onSensorChanged() メソッドでのセンサ値の取得・保存が処理として渡されており，onSensorChanged() メソッドは明示的に SensorListener の登録解除を指示しない限りセンサ値の変化を待機し続けるため，センサ値取得処理がバックグラウンドでアプリケーション終了まで継続することとなる．

LearningService クラスも同様に IntentService クラスを継承しており，処理は他の動作と関係ないスレッドで実行される．MainActivity インスタンスに設けた (Button)learningButton（UI 中央の紺色のボタン）からラベルを putExtra()・getXXExtra() メソッドによって受け取り起動し，globalValues から現在のセンサ値を取得して 5 次元ベクトルとして，受け取ったラベルとともに glovalValues>clsModel に保存する．

MeasuringActivity インスタンスには onResume() メソッド内の UI 実行中処理としてハンドラと Handler.postDelayed() メソッドによる簡単な定期実行処理を定義しており，ここに書かれた処理が 1,000 ミリ秒ごとに実行され続けるようになっている．具体的な処理として，(Button)modeSwitchButton（UI 右下の灰色・朱色のボタン）の状態に応じた分岐を記述しており，学習モードでは UI 左下部へのセンサ値の表示，一方検出モードでは clsSelfState() メソッドによる自己状態検出処理が定期実行される．

clsSelfState() メソッドは k-近傍法に基づく自己状態推定を行い，推定結果のラベルを返す．まず現在のセンサ値を globalValues インスタンスから取得して 5 次元ベクトルとし，さらに globalValues>clsModel インスタンスに保存されている学習データ配列の内容をすべて参照しながら，現在のセンサ値ベクトルとの距離（差分ノルム）を計算してゆき，距離が大きいほうから k 個の学習データを選び出してそのラベルのうち最多数を占めるものを推定ラベルとする．なお，ラベル返り値の初期値は正常なラベルの範囲”>0” から外れる”-1” としており，推定のための学習データが足りない場合には推定が実行されないよう定義してある．

3. 考察

本アプリケーションでは多クラス分類を k-NN により実装した．この手法は分類実行毎の計算量が比較的多く，またメモリを大きく占有することで知られているが，今回実機において学習データサイズ 100 程度まで実際に動作させてみたところ，特に動作の遅延やメモリ不足などに陥ることなく動作した．k-NN で計算量は学習データの数に対して線形に増大することから，ある程度の学習データサイズの増大に対しては実行に深刻な問題を生じることがないのではないかと推測される．バックグラウンドでの継続処理に関しても，メモリ占有率が肥大することなく，ユーザが他のアプリを使用する大きな妨げとはならないよう設計できるだろう．

このように，本アプリケーションの製作は，Android 端末での連続データ取得・連続データ処理に関する実用可能性が感じられる結果となった．

また，本アプリケーションは瞬時値を利用した自己「状態」の推定を行ったが，これを時系列を利用した自己「様態」の推定に応用すれば（実際，データの保存を一定時間長へと拡張することで実装できると考えられる），Android 端末をセンサの集合と考えた場合にセンサ自身の置かれた状況をラベルとして含めたデータの取得が可能となり，連続的な移動や健康状態のモニタリングに一役買うことも可能となるのではないか．