Introduction to Movesense Programming

@ Movesense Meetup Düsseldorf

14.2.2019

Petri Lipponen

Content

- Movesense system overview
- Movesense sensor overview
- How to start developing?
- Data aquisition
- Sensor simulator
- Sensor programming basics
- Mobile programming basics
- Wrap-up

Movesense System Overview

- Movesense sensor
- Sensor software platform
  - REST like sensor API’s
- Mobile connectivity solution for iOS & Android
  - Easy to access all sensor features via same API’s
  - JSON & REST
- Support for other frameworks
  - Unity3D
  - Xamarin

Movesense Sensor Overview

- CR 2025 Coincell battery
- 64MHz NordicSemiconductor MCU (RAM: 64kB, FLASH 512kB)
- Data memory: 384kB
- 9-axis IMU (Accelerometer, Gyroscope, Magnetometer)
- Temperature measurement
- Maxim ECG Analog Frontend
  (ECG, HeartRate, RR-intervals, stud contact detection)
- Maxim 1-wire Master
  - Smart connector detection
  - 1-Wire communication support (1.8 volts)

How to start developing?

- *Have a clear idea on what you want to measure*
- Get to know the sensor using the mobile ”Showcase App”
- Record some data and take a good look at it
  - What does it show?
  - What does it not show?
  - Noise signals?
- Make a simple mobile software that ”does the trick”
  - Easier to debug & find coding help!
- Start development on the simulator and only when that works continue on the sensor device

Data aquisition

- Understand what you are measuring
- Speed & capacity limits by sensor, data-memory and BLE connections!
    - Sensor: G-ranges, sensitivity, etc.
    - Data memory: 400kbps bandwidth, limited size
    - BLE: 12/60 kB/s theoretical maximum (1.7 / 1.8), in practice less
- Measure with Showcase App if possible, using DataLogger as a fallback (see Android *samples/DataLoggerSample*)

Sensor Simulator

- ”Movesense sensor software on Windows & Visual Studio”
- Easier debugging and faster development cycle
- Simulated sensors with data import
- Whiteboard communication using *wbcmd.exe*

- Limitations:
  - No BLE
  - No Mobile communication
  - Not 100% accurate

Preparing data for simulator

- Edit CSV so that the data is in correct format:
  - Comma (,) separator
  - Period (.) as decimal
  - Header row with *LoopingTimestamp* (optional)
  - 1st column is ”Timestamp” (ms since start of the sensor)
  - Rest of columns with data. ColumnHeader from simulator debug output
- Confirm that simulator reads the file correctly
- Confirm that data comes out correctly (use wbcmd)

Sensor programming basics

- C/C++ with some limitations:
  - No dynamic memory (there is but…)
  - No STL
  - Limited resources (thread stack, cstack, RAM)
- Asynchronous API's
  - Code *MUST NOT* hog the execution => No busy-loops!
  - Automatic power optimization
  - Call – callback structure
- REST-like with some additions (Publish-subscribe pattern)

Sensor software structure

- *wbresources* –folder has app specific API definitions
- Source files in app folder
- CMakeLists.txt tells how to build
- app_root.yaml contains execution contexts and lists app API’s
- App.cpp contains applications movesense settings
  - Optional modules
  - Data memory allocations
  - Debug settings

Sensor programming basics: *Whiteboard*

- Services, clients, timers, threading and external communication
- ExecutionContext: Whiteboard threads
- LaunchableModule: ”wb-aware module”
  - Runs in an ExecutionContext (WB thread)
  - Lifecycle callbacks (initModule, startModule, stopModule, deinitModule)
- ResourceProvider: WB REST service
  - API defined using Swagger 2.0 notation (yaml-file)
  - Request callbacks: onGetRequest, onPutRequest,…
- ResourceClient: WB REST client
  - Make requests to internal and external whiteboard services
  - Request methods: asyncGet, asyncPut,…

Mobile programming basics

- MDS: Whiteboard for mobile
- Same sensor REST-api’s available from mobile
- REST verb method calls with async JSON results
- *Least* moving parts when developing on Android / iOS MSD
- Best code samples are for Android
- Alternatively:
  - Use plain BLE & GATT and CustomGattService (requires sensor programming)
  - Use some multiplatform framework: Unity3D, Xamarin, react native, cordova

Questions?