Exploring IoT Applications in High School

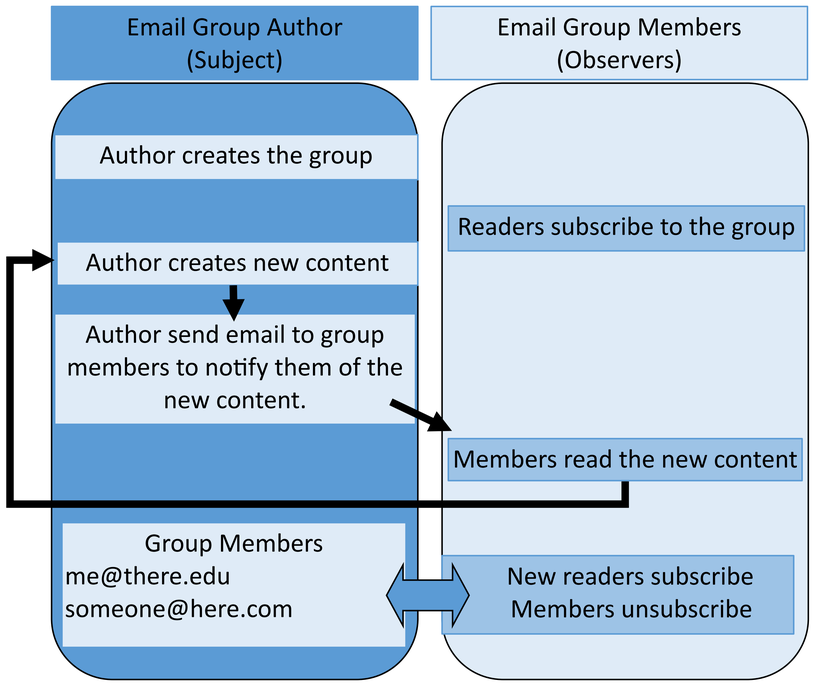
The Subject-Observer Pattern

Explained with a common example

The Subject-Observer pattern is commonly used to respond to changes in the state of some object (such as a button on the screen, a key on a keyboard, or a sensor on a device.)

A ***subject*** keeps track of its state and keeps a list of ***observers*** to notify when its state changes.

Consider an email group list for one-way announcements:



Using the Subject-Observer model eliminates the need to modify the subject when different or additional observers are needed. The observers simply register themselves to receive notifications from the subject. This pattern is used to make the Microsoft Band 2 sensors communicate with the Android App.

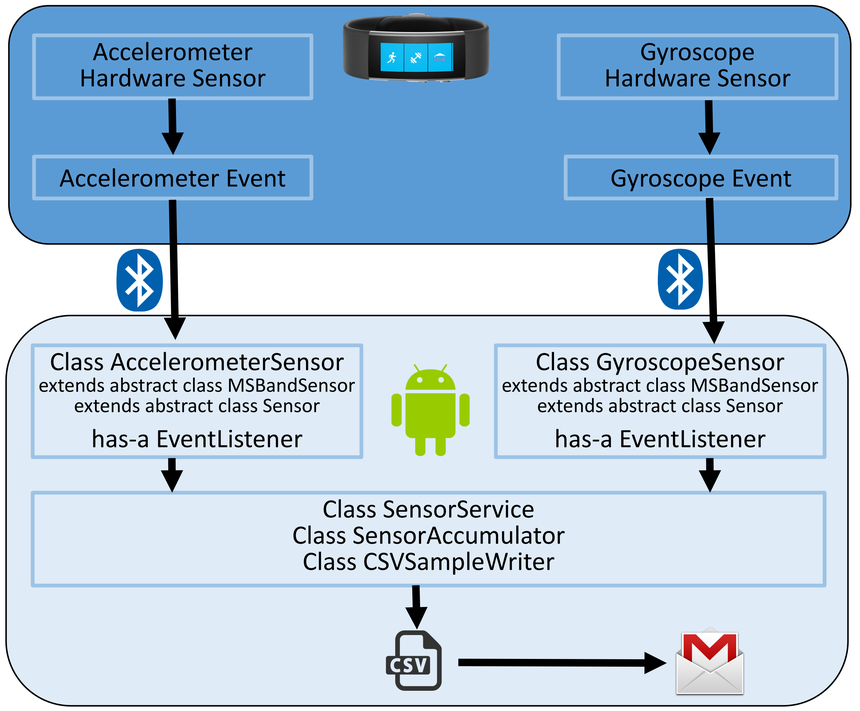
The Subject-Observer Pattern in Java: Events and EventListeners

Java implements the Subject-Observer pattern with Event objects and EventListener classes. The Event object contains the data that describes the subject's current state. When a state-change occurs in the subject, it creates an Event object and generates a state-change event, which is sent to all registered EventListeners (observers).

The API for each sensor in the Microsoft Band includes both an Event interface and an EventListener interface. The Band library (JAR file) contains an implementation class for each event interface. The observer class for each sensor must implement an EventListener.

Data Flow in the MS Band App

Microsoft Band » Android App » Data File



- **SensorService** is a concrete class that extends the Android Service class. It creates background code to manage all of the sensor collection and to run a Thread class to receive asynchronous events from the sensor EventListeners.
- **Sensor** is an abstract class that provides methods for tasks that are common to all sensors on any device - not specifically the Microsoft Band hardware.
- **MSBandSensor** is an abstract subclass of Sensor that provides methods for tasks common to all Microsoft Band hardware sensors. It declares some methods in Sensor that are common to all Microsoft Band sensors.
- **AccelerometerSensor**, **GyroscopeSensor**, etc. (each hardware sensor's class) is a concrete subclass of MSBandSensor. Each of these classes:
  - provides a EventListener of the appropriate type
  - establishes the sensor's meta-data (sensor name, raw data types, sampling rate)
  - provides methods to register and unregister the specific instance of an EventListener

Available Sensors

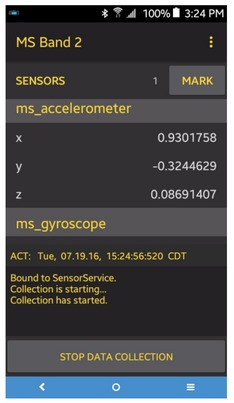
Raw Data from Microsoft Band 2 (Available via Microsoft SDK)

- Accelerometer
- Altimeter
- Ambient Light
- Barometer & Ambient Temp
- Calories
- Contact
- Distance
- Galvanic Skin Resistance
- Gyroscope
- Heart Rate
- Pedometer
- RR (heart beat interval)
- Skin Temperature
- UV Light

Raw Data from the Android Device (Available via the Android SDK)

- Button Touch (listener for buttons on app screen)
- Location (listens for GPS readings from Android device)

Data Collection with the Band App



- The app collects data on all sensors for each run.
- The MARK button allows the user to change the marker number in the data file to identify each trial or each step in a trial.
- After data collection stops, the file can be sent by email or can be retrieved later with a file manager app.

1	ms_gsr_resistance_kOhm	ms_rr_interval_sec	ms_heart_rate_bpm	ms_heart_rate_(
29	340330	0.4148	150	ACQUIRING
30	340330	0.4148	150	ACQUIRING
31	340330	0.4148	150	ACQUIRING

The CSV data file has appropriate column headers to identify the sensor, the value, and the units.

Applications in Science Classes

- Accelerometer
  - Measurement when launching small spring-powered vehicles
  - Direct measurement of forces in scale-model roller coasters
- Gyroscope
  - Measurement of forces and velocity in study of rotational mechanics
- Heart rate, Skin Temp, GSR
  - Measurement of data for correlating responses to sweat production during exercise (in a study of homeostasis)

IoT in Computer Science Classes

IoT programming projects will be introduced in class. This has the advantage that it can be seen by students as more "real world" than traditional projects.

Programming apps for a phone in a high school provide a context (phone interactions) that is familiar to the students, which leads to better learning engagement and interest in pursuing computer science major.

The Fall Detection IoT application created by REU students will be demonstrated. This exposes students to the engineering of a fairy large and complex piece of software.

Students will be instructed to modify the Fall Detection application for collecting other sensor data from the smartwatch. This exposes students to the importance of structuring codes for reusability.

Acknowledgements

- The MS Band 2 app is a modification of an app written by Mario A. Gutierrez, et al, during the Texas State University REU-IoT program in summer 2015. This work was an invaluable starting point for this work.
- REU-IoT students Brock Yarbrough, Andrew Polican, and Joie Wu provided clarification of details based on their experience with the original on which my code is based.
- Dr. Ann Ngu provided direction, support, and encouragement throughout the project, and for to the opportunity to participate in the project.
- We thank the National Science Foundation for funding this research under the Research Experiences for Undergraduates Program, CNS-1358939, and NSF-CRI 1305302 awards, and the supplemental RET (Research Experiences for Teachers) funding.