Universität Stuttgart

Studienprojekt **Garment OS** der Abteilung für Mensch-Computer-Interaktion

Abschlussbericht

Kunde:
Prof. Dr. Albrecht Schmidt

Betreuer:
Jun.-Prof. Dr. Niels Henze
M. Sc. Stefan Schneegass
Dipl.-Inf. Markus Funk

Autoren:
Tamara Müller, Lucas Röhrle, Tobias Linn, Sophie Ogando, Ferdinand Pfähler, Velihan Bulut, Martin Root, Oliver Röhrdanz, Vincenz Pauly, Manuel Lorenz

28. April 2015

1 Garment OS Architektur

Betrachtet man Garment OS als Gesamtsystem kann man es in zwei große Schichten aufteilen. Zum einen das SDK (Software Development Kit) und zum anderen der GOSS (Garment OS Service). Das SDK dient als Grundlage für alle Apps die Funktionen von Garment OS verwenden, auch die Garment OS Settings App verwendet eine modifizierte Version des Garment OS SDK, welches es nur als Teil der App gibt und nicht seperat verwendet werden kann.
Das SDK ist aufgeteilt in die verschiedenen Module die es bietet. Zum einen erlaubt das SDK sich für verschiedene Events über einen Callback zu registrieren. Wird das passende Event im Service festgestellt wird der Callback ausgelöst und die App so über das Eintreten des Events informiert. Um Sensordaten abzurufen stehen uns also zwei Möglichkeiten zur Verfügung, zum einen kann dies über die gerade beschriebenen Callback Funktionen erledigt werden, alternativ kann allerdings auch aktiv der aktuelle Sensorwert einer der verfügbaren Sensoren abgefragt werden. Teil des SDK ist auch das Development Module welches den Nutzern erlaubt relativ einfach und ohne große Programmierkenntnisse eigene Apps zu erstellen.
Die verfügbaren SDK Module basieren auf einem APIFunctions Layer der alle benötigten Funktionen bereitstellt. Die Funktionen sind Teil der Interprozesskommunikation und kommunizieren und beziehen ihre Daten aus dem Garment OS Service. Um die Interprozesskommunation herzustellen verwendet das SDK das API Handle, welches dafür sorgt, dass beim Start der App die Verbindung zum Service hergestellt wird. Der GOSS nimmt diese Anfragen von außen über den API Binder entgegen. Dieser ist auch dafür verantwortlich die in den API Functions gebündelten Funktionen wieder auf die verschiedenen Module aufzuteilen.

1.1 Interprozesskommunation

Viele Systeme bieten die Interprozesskommunikation über geteilten Speicher an, auf Android Systemen ist dies allerdings nicht ohne weiteres möglich und ein direkter, prozessübergreifender Speicherzugriff ist nicht möglich. Um dennoch über Prozesse hinweg miteinander kommunizieren zu können gibt es für Android Systeme die AIDL, welche eine Abstraktion des im Android Kernel implementierten Binder Konzepts darstellt. Hierbei kann ein funktionales Interface (Implementierungslose Methoden) definiert werden. Über diese Schnittstelle ist es jedoch nur möglich primitive Datentypen zu übertragen, sodass eigene Objekte auch zerlegt werden müssen um diese über diese Schnittstelle zu übertragen.

2 Speicher- und Verschlüsselungsmodul

Ursprünglich hatten wir die Idee alles in einer Datenbank zu speichern, dass dies allerdings nicht zum gewünschten Ergebnis führte haben wir uns entschlossen, das Speichersystem nochmals zu überdenken. Die Sensordaten werden für jeden Sensor in einer eigenen Datei abgelegt, diese kann eindeutig über die Sensor ID (ebenfalls eindeutig) identifiziert werden.
Zusätzlich gibt es noch zwei weitere Dateien, die eine speichert die Rechtevergabe an Apps, die andere speichert die Sensoreinstellungen. Die Sensordatendatei besteht aus einem Header und vielen Datenchunks. Im Kopf der Datei wird der Zeitpunkt der letzten Änderung gespeichert, und die Dimension der Sensordaten.
Die Datenchunks bestehen zum einen aus dem Zeitpunkt der Datenaufnahme und zum anderen aus den Sensorwerten, die je Dimension in einem 32 bit Fließkommawert gespeichert und in die Datei geschrieben werden. Beim Auslesen kann man sich die chronologische Sortierung der Datei zu Nutze machen und kann auf einen beliebigen Sensorwert in logarithmischer Zeit zugreifen. Die Serialisierung in einzelne Dateien hat den Vorteil, dass sich Sensordaten für einzelne Sensoren leicht importieren oder exportieren lassen.

2.1 Wie werden Daten exportiert und importiert?

Exportieren und Importieren funktioniert live ohne etwas zu deaktivieren. Der Zugriff auf die Dateien wird gesperrt. Das Speichermodul cachet neue Sensordaten bis der Zugriff auf die Dateien wieder freigegeben ist.
Um zu verhindern, dass andere Archiv Dateien importiert werden wird die komprimierte Datei noch modifiziert, genauer gesagt der Kopf der Zip Datei wird so angepasst, dass diese eindeutig als Garment OS Export Datei vom System erkannt wird.
Beim Importieren wird die Datei entpackt und es wird nach möglichen Dateikonflikten gesucht, sofern ein Dateikonflikt gefunden wurde, wird versucht diese zwei Dateien zusammenzuführen.

3 Treibermodul

Fast täglich erscheinen neue Sensoren und Geräte die, sofern der Treiber dafür geschrieben wurde mit Garment OS interagieren können. Es ist also gewünscht einen Treiber auch lange Zeit nach dem deployen von Garment OS dem System hinzuzufügen. Funktionen (neuer Code) einem bestehenden System hinzuzufügen ist aber keine triviale Aufgabe.
Die Möglichkeit den neuen Treiber direkt in die App zu integrieren scheint weniger sinnvoll, da die App neu gepackt und optimiert werden müsste, was vor allem auf leistungsschwächeren Handys keinen akzeptablen Zustand darstellen würde. Alternativ könnte man sich auch ein auf XML Basis existierendes Treibersystem vorstellen, dass zur Laufzeit dann interpretiert wird. Dies ist allerdings nicht leistungsfähig genug (was die Möglichkeiten betrifft) um unsere Anforderungen zu erfüllen.
Daher haben wir uns für Treiber Apps entschieden, die bspw. mehrere Sensoren bündeln und gleichzeitig bei Garment OS registrieren können. Die Funktion dafür ist im SDK integriert. Der Treiber an sich ist ein Interface bzw. eine abstrakte Klasse, in dem vom Nutzer zwei Funktionen implementiert werden müssen.
Das Treibermodul implementiert einen Standarttreiber für Daten mit bis zu 6 Dimensionen. Dazu muss ein Datenpaket wie folgt aufgebaut sein:

[Dim1,Dim2,Dim3,Dim4,Dim5,Dim6]

Die `[]` sind nicht Teil des Pakets. Pakete werden voneinander mit einem Komma getrennt.

3.1 Ablauf

Der Sensor ist dem System bereits bekannt und mit richtigen Treiber am System registriert, dieser empfängt nun die Sensordaten. Diese Rohstrom an Sensordaten leitet der Sensor weiter an den Sensor Thread.
Jeder Sensor hat seinen eigenen Thread, sodass dieser komplett unabhängig von den anderen Sensoren laufen und agieren kann. Dieser Sensor Thread steuert den Sensor und leitet diese Rohdaten weiter an die Treiber App, genauer an den dem Sensor zugewiesenen Treiber. Dieser erzeugt nun aus den erhaltenen Rohdaten die Sensorwerte.
Sofern der Treiber mit der Auswertung der erhaltenen Daten fertig ist, sendet dieser sein Ergebnis weiter an Garment OS, welche nun die Sensor Daten verarbeiten und speichern kann. Ab diesem Schritt gibt es keinen Unterschied mehr zwischen internen Sensoren für die die benötigten Treiber bereits im System implementiert sind oder zwischen nachträglich hinzugefügten Treibern und Sensoren.

4 Activity Recognition Module

Das Activity Recognition Module (im folgenden als ARM bezeichnet) gibt die Möglichkeit anhand der aufgenommenen Sensordaten Aktivitäten, die ein Benutzer ausführt, zu erkennen. Das ARM, welches im internen Service läuft, besteht dazu aus mehreren Teilen, die im Folgenden näher erklärt werden.
Das ARM selber lädt die aufgenommenen Sensordaten aus dem Speicher und schreibt sie in ein Zeitfenster, welche alle die Daten aller Sensoren in einem bestimmten Zeitraum enthält. Die unterstützten Sensoren und der Zeitraum (meistens einige Sekunden) werden dabei von dem Benutzer festgelegt. Das Zeitfenster enthält zudem sein Feature Set in dem die anhand der Daten berechneten Features gespeichert sind. Die Features bestehen aus statistischen (Mittelwert, Abweichung, ...), strukturellen (polynomielle Regression) und vergänglichen (lineare Regression) Daten.
Zusätzlich gibt es einen Klassifikationsalgorithmus und einen entsprechenden Manager, welcher den Algorithmus steuert. Dieser beinhaltet zusätzlich noch Informationen wie die Anzahl der Trainingsepochen, unterstützte Aktivitäten und unterstützte Sensoren. In diesem Fall handelt es sich bei dem Klassifizierungsalgorithmus um ein neuronales Netzwerk. Das neuronale Netzwerk lässt sich über das ARM mit den geladenen Features trainieren. Dazu werden werden die Features und ein entsprechendes Aktivitätenlabel (repräsentativer Wert einer Aktivität) an das Netzwerk übergeben welches darauf hin eine Funktion zu berechnen versucht, die entsprechende Features auf das Label abbildet. Das Training findet dabei über einen längeren Zeitraum statt mit immer neuen Zeitfenstern und dementsprechend immer variierenden Features. Das hat zur Folge, dass der kontinuierliche Stream an Daten in kleine Abschnitte aufgeteilt wird. Entsprechen muss das Netzwerk auch auf alle zu unterstützenden Aktivitäten trainiert werden. Hier sind meist mehrere tausend Trainingsepochen nötig, dass eine entsprechend zuverlässige Erkennung möglich ist.
Das Training des Netzwerkes kann live oder mit bereits zuvor aufgenommenen Daten durchgeführt werden. Hier wird beim live Training angegeben welche Aktivität gerade ausgeführt wird und das Training läuft nebenher, wohingegen beim Training mit bereits gespeicherten Daten, dass Training schnell (abhängig von der Länge des Zeitraums) im Hintergrund durchgeführt wird. Das Netzwerk und die dazugehörigen Daten werden nach einer gewissen Anzahl an Trainingsepochen automatisch gespeichert, damit es nach Beenden des Services nicht gelöscht und neu trainiert werden muss.
Ist das Netzwerk trainiert, kann es letztendlich zur Erkennung benutzt werden. Einziger Unterschied hier zum Training ist, dass jetzt nur die Features übergeben werden und das Netzwerk versuch, diese auf eine Aktivität abzubilden und diese auszugeben. Da die Daten zum Teil um einiges variieren, ist eine gewisse Ungenauigkeit bei der Erkennung zu erwarten. Sollte einmal Bedarf bestehen die Unterstützten Aktivitäten oder oder Sensoren zu ändern, so wird das bestehende Netzwerk mit einem neuen aktualisierten Netzwerk überschrieben. Sämtliche Trainingsfortschritte werden hierbei gelöscht.

5 Bluetooth Modul

Das Bluetooth Modul (im folgenden BT-Modul) stellt die Kommunikation zwischen Garment OS und Bluetooth-Sensoren bereit.
Das Bluetooth Modul verwendet ausschließlich das SerialPortProtokol mit dieser UUID *00001101-0000-1000-8000-00805F9B34FB*. Es ist nicht kompatibel mit Bluetooth Low Energy. Jedoch ist eine Erweiterung auf das LE Protokoll umsetzbar.
Die Übertragung der Nachrichten findet bytestrom statt. Das Bluetooth Modul verwendet entsprechende Treiber die speziell für den jeweiligen Sensor angepasst sind um den bytestrom zu decodieren.

Die Verbindung zu einem Sensor wird in einem eigenen Thread aufgebaut und starte immer dann wenn ein Sensor mit "enabled" erstellt wird oder ein Sensor von "disabled" auf "enabled" gestellt wird. Die Verbindung wird solange aufrecht erhalten bis einer der Fälle eintritt:

1. Sensor wird deaktiviert
2. Sensor wird auf "disabled" gestellt
3. 3 Verbindungsversuche schlagen fehl
4. 3 Versuche die Verbindung nach einem Fehler wiederherzustellen schlagen fehl

Alle 4 Fälle führen dazu das eine Verbindung nicht aufgebaut und der Sensor auf "disabled" gestellt wird.

Nachdem erfolgreich eine Verbindung zum Sensor aufgebaut wurde, wird abhängig von den gewählten Poweroptions der Nachrichtenstrom verarbeitet. Die Verarbeitung findet in einem weiteren unabhängigen Thread statt.
Sollten entweder der Treiber oder der Kommunikationsthread blockieren ist nur diese eine Verbindung betroffen, der Rest des Systems arbeitet unabhängig weiter.
Die Performenceeinbuße fällt dementsprechend gering aus, da einer der Threads für maximal 3 Verbindungsversuchen blockieren kann, danach wird er terminiert.

6 Entwicklungsmodul

Das Entwicklungsmodul ermöglicht es Entwicklern einfach Apps mithilfe des GarmentOS SDKs zu entwickeln. Dazu wird eine Anleitung bereitgestellt in der das Erstellen von Apps mithilfe des Entwicklungsmoduls Schrittweise beschrieben ist.
Das Entwicklungsmodul besteht aus vorgefertigten Modulen die im GUI-Editor (In Eclipse oder IntelliJ) per Drag and Drop einfach ins Layout gezogen werden können.
Somit kann einfach eine App erstellt werden die Sensorwerte anzeigt.
Es gibt zwei unterschiedliche Arten von Modulen, die einen können Sensorwerte in einem Graphen anzeigen, die anderen textuell.
Die Sensoren können entweder vom Benutzer im Popup-Menü ausgewählt werden oder vom Entwickler im XML oder im Code fest gesetzt werden. Wenn ein Sensor vom Entwickler fest gesetzt wird, wird das Popup-Menü ausgeblendet.
Entwickler haben außerdem die Möglichkeit ein Custom Popup-Menü mit beliebigem Inhalt zu erstellen.
Es gibt drei fertige Beispiel-Module, einmal das GPS Module, das den aktuellen Wert eines GPS Sensor anzeigt, dann das Heartrate-Modul das den aktuellen Wert eines Heartrate Sensors anzeigt und das Temperature-Modul, dass den aktuellen Wert eines Temperatur Sensors anzeigt.

7 SettingsApp