Titel

Seminararbeit

Version 0.01

Funktionale Programmierung
Zürcher Hochschule für Angewandte Wissenschaften

Simon Lang, Daniel Brun

Date

Versionshistorie

Version	Datum	Autor(en)	Änderungen
0.01	13.04.2015	DBRU	Initiale Version

Daniel Brun (DBRU)

Eigenständigkeitserklärung

Hiermit bestätigt ich, dass vorliegende Semesterarbeit zum Thema „BigData mit RaspberryPi und F#“ gemäss freigegebener Aufgabenstellung ohne jede fremde Hilfe und unter Benutzung der angegebenen Quellen im Rahmen der gültigen Reglemente selbständig verfasst wurde.

Zürich, 24.09.2015 Simon Lang, Daniel Brun

Inhaltsverzeichnis

KAPITEL 1

Einleitung

Diese Arbeit wurde als Seminararbeit zur Vorlesung von funktionalen Programmiersprachen verfasst. In diesem Kapitel wird die Aufgabenstellungen und Rahmenbedingungen der Arbeit erläutert.

1.1 Hintergrund

Einer immer grösseren Beliebtheit erfreuen sich kleine Alltagsgegenstände welche mit dem Internet verbunden sind. Dieser Bereich wird IoT genannt. Diese Gegenstände sind in der Lage Daten zu erheben und weiterzuleiten. Da es zukünftig voraussichtlich immer mehr IoT Gegenstände geben wird fallen immer mehr Daten an. Diese Daten werden wegen ihrer Masse auch BigData genannt.

In dieser Arbeit wird evaluiert wie sich funktionale Programmiersprachen im Bezug auf IoT eignen, um BigData auszuwerten.

1.2 Ziel

Mit einem IoT Gerät sollen Daten aufgezeichnet werden. Diese werden als BigData gesammelt und sollen mit einer funktionalen Programmiersprache ausgewertet und ansprechend ausgegeben werden.

Das Hauptziel der Arbeit besteht darin zu überprüfen wie geeignet funktionale Programmiersprachen für die Auswertung von BigData sind. Als Nebenziel soll evaluiert werden, ob eine funktionale Programmiersprache zum erfassen von Daten auf einem IoT Gerät verwendet werden kann.

1.3 Aufgabenstellung

Die freigegebene Aufgabenstellung lautet wie folgt:

- Projektname: Seminar BigData mit RaspberryPi und F#
- Ausgangslage: Durch die rasante Entwicklung im Bereich IoT ergeben sich viele neue Anwendungsmöglichkeiten. Da der RaspberryPI immer leistungsfähiger geworden ist, soll evaluiert werden ob er sich für den Einsatz von funktionalen Sprachen im Bereich BigData eignet.
- Ziel der Arbeit: Es soll gezeigt werden wie F# Sharp auf einem RaspberryPi im Bereich BigData und IoT eingesetzt werden kann.
- Aufgabenstellung: Es soll gezeigt werden, wie eine funktionale Programmiersprache (F#) im Kontext von BigData und IoT eingesetzt und verwendet werden kann. Es soll eine Anwendung zur Sammlung von Sensordaten auf einem Raspberry PI und eine Anwendung zur Analyse / Auswertung der gesammelten Daten implementiert werden.

1.4 Erwartete Resultate

Gemäss freigegebener Aufgabenstellung werden folgende Resultate erwartet:

- Dokumentation
- Implementation / Prototyp

1.5 Abgrenzung

Aufgrund des Umfanges der Arbeit und der begrenzten Zeitdauer werden folgende Punkte von der Arbeit abgegrenzt:

- **Schnittstellendokumentation**
  In dieser Arbeit werden nicht die Schnittstellendokumentationen und -spezifikationen rekonstruiert. Es werden jeweils die relevanten Aspekte betrachtet und hervorgehoben.

1.6 Motivation

1.7 Struktur

1.8 Planung

KAPITEL 2

Recherche

In diesem Kapitel werden die Grundlagen recherchiert wie die beiden Teilprojekte „Sensordaten sammeln“ und „Sensordaten auswerten (BigData)“ angegangen werden könnten.

2.1 Ausgangslage

Die Vorlesung zu diesem Semniar befasst sich mit den Konzepten der Funktionalen Programmierung. Zur Veranschaulichung dieser Konzepte wurde die Programmiersprache F# des .NET-Frameworks verwendet. Aufgrund dessen haben wir uns entschieden auch dieses Seminar mit der uns nun bekannten Sprache F# umzusetzen. Als IoT Gerät wird ein Raspberry Pi[1] verwendet. Der Raspberry Pi ist ein Einplatinencomputer welcher von der britischen Raspberry Pi Foundation entwickelt wurde. Der Raspberry Pi bietet den Vorteil, dass er sehr weit verbreitet ist[2], er kostengünstig ist und es inzwischen eine sehr grosse Anzahl an Sensoren auf dem Markt gibt mit welchem man Daten sammeln kann[3].

2.2 Der Raspberry Pi

Wie bereits im vorangehenden Kapitel beschrieben, handelt es sich beim Raspberry Pi um einen Einplatinencomputer. Dieser Einplatinencomputer bietet verschiedene zentrale Hardware-Schnittstellen um externe Geräte für Input und Output anzuschliessen.

Vom Raspberry PI gibt es folgende Modelle:

- Raspberry Pi Compute Module
- Raspberry Pi Zero
- Raspberry Pi Model A

1 **Raspberry_Pi_2016-04-24**.
2 **Raspberry_Pi_Erfolgsgeschichte_2016-04-24**.
3 **Raspberry_Pi_Sensor_2016-04-24**.

- Raspberry Pi Model A+
- Raspberry Pi Model B
- Raspberry Pi Model B+
- Raspberry Pi 2 Model B
- Raspberry Pi 3 Model B

Am 29 Februar 2016 ist die neuste Version, der Raspberry Pi 3 (Model B), auf dem Markt erschienen[1]. Einige Zahlen zu dem Gerät:

- 1.2GHz 64-bit quad-core ARM Cortex-A53 CPU (10x die Leistung eines Raspberry Pi 1 und 50-60% die Leistung eines Raspberry Pi 2)
- Integriertes 802.11n wireless LAN und Bluetooth 4.1
- Komplette Kompatibilität zu Raspberry Pi 1 und 2 (Model B)

2.3 Eingesetzte Hardware

Für diese Seminararbeit wurden folgende Hardwarekomponenten verwendet:

- Raspberry Pi 2 Model B
- Raspberry Pi 3 Model B
- GrovePi Sensoren

2.4 F# mit dem Raspberry Pi

Um F# auf dem dem Raspberry PI auszuführen, gibt es grundsätzlich zwei Möglichkeiten, welche nachfolgend erläutert werden.

2.4.1 Linux (Raspbian) mit Mono

Ein weit verbreitetes Betriebsystem für das Raspberry Pi ist das Raspbian[2]. Bei dem Namen handelt es sich um eine Zusammenfassung von Raspberry und Debian. Demnach handelt es sich auch um eine Debian Distribution.

Um F# auf dem Raspbian laufen zu lassen wird Mono benötigt[3]. Dabei handelt es sich um eine Open Source implementierung von Microsoft's .NET Framework, welches für die Ausführung von F# benötigt wird.

1 **Raspberry_Pi_3_2016-04-24**.
2 **FrontPage_-_Raspbian_2016-04-24**.
3 **Mono_2016-04-24**.

2.4.2 Windows 10 IoT

Microsoft hat mit Windows 10 IoT eine Version ihres Betriebsystem herausgebracht, welches speziell für leistungsschwächere Geräte entwickelt wurde[1]. Bei der IoT Version von Windows 10 it ein .NET Framework bereits vorhanden. Demnach sollte es keine Probleme darstellen F# darauf laufen zu lassen.

2.5 Sensoren

Für den Raspberry Pi gibt es viele Sensoren auf dem Markt. Herauskristallisiert hat sich jedoch das Starter Kid GrovePi+[2]. Dieses kommt mit einem Ton-, Temperatur-, Feuchtigkeit-, Lichts- und noch weiteren Sensoren. Vorteil an dem Kit ist es, dass es eine Library für .NET gibt mit welchem die Sensoren angesprochen werden können[3].

2.6 Datenauswertung

Die von den Sensoren gespeicherten Daten werden in einem noch zu definierenden Format abgespeichert und danach für die Datenauswertung ausgelesen. Dafür wurde vom Dozenten die Library F# Data vorgeschlagen[4]. Diese kann Daten im Format CSV, HTML, JSON und XML entgegennehmen und für die Verwendung in F# zur Verfügung stellen.

1 **Windows_IoT_2016-04-24**.
2 **GrovePi_2016-04-24**.
3 **NuGet_GrovePi_2016-04-24**.
4 **Fsharp_Data_2016-04-24**.

KAPITEL 3

Datensammlung

Dieses Kapitel beschäftigt sich mit dem Teilprojekt der Datensammlung. Dazu wird ein Raspberry Pi verwendet welcher mit F# Informationen von verschiedenen Sensoren zusammentragen soll. Diese Daten sollen in darauf folgenden Kapiteln weiterverwendet werden.

3.1 F# auf dem Raspberry Pi

Im Abschnitt 2.4 F# mit dem Raspberry Pi wurden zwei Möglichkeiten aufgezeigt, welche es ermöglichen F# auf einem Raspberry Pi laufen zu lassen. Entweder wird das Linux Raspbian mit Mono verwendet, oder das Windows 10 IoT.

Es war zu erwarten, dass der Weg über Window 10 IoT der einfachere ist, da dort .NET schon mitgeliefert wird. In diesem Projekt wurden beide Methoden ausprobiert. Die folgenden Abschnitte erläutern mit den beiden Herangehensweisen an das Problem.

3.1.1 Raspbian

3.1.2 Windows 10 IoT

Windows 10 IoT ist eine Version des Betriebssystems von Microsoft, welches speziell für kleinere Geräte mit weniger Rechenleistung konzipiert wurde.

Die Installation gemäss der Anleitung auf dem Github Account from Microsoft[1] war nicht erfolgreich. Der Raspberry Pi startete nicht und blieb beim Rainbow Screen[2] hängen. Mit dem NOOBS[3] Installer, welcher von der Raspberry Pi Foundation zur Verfügung gestellt wird, war die installation von Windows 10 Io

1 **install_win10iot_2016-04-25**.
2 **RPi_Rainbowscreen_2016-04-25**.
3 **NOOBS_2016-04-25**.

KAPITEL 4

Schlusswort

4.1 Fazit

4.2 Vergleich: Ist -/ Soll-Planung

4.3 Dank

Abbildungsverzeichnis

ANHANG A

Anhang