Fachhochschule Nordwestschweiz

P5: Happy Flowers

für
Rinesch Murugathas, rinesch.murugathas@students.fhnw.ch
Sacha Schmid, sacha.schmid@students.fhnw.ch

1. Einleitung

Am Institut für Mobile und Verteilte Systeme (IMVS) kämpfen einige Pflanzen mit langanhaltenden Trockenperioden. Das Giessen der Pflanzen soll nun mit diesem Projekt automatisiert werden. Sie verwenden Haskell zur Implementierung einer Applikation mit lowlevel Schnittstellen zu Hardware, Datenbankanbindung und highlevel Webschnittstelle.

2. Aufgabenstellung

Ziel dieser Arbeit ist eine Haskell Applikation, die auf dem Raspberry Pi läuft. Sie soll regelmässig die Feuchtigkeit der Erde über einen Sensor auslesen und bei Trockenheit eine Pumpe einschalten. Die Feuchtigkeitswerte und die Schaltzeiten der Pumpe sollen live über ein Webfrontend visualisiert werden. Zudem soll nach erfolgreichem Login die Pumpe via Web manuell angesteuert werden können.
Die Applikation soll im fprog Unterricht verwendet werden um zu zeigen, dass man durchaus praktische Probleme mit Haskell lösen kann. Entsprechend muss die Qualität der Arbeit stimmen: Verständlichkeit und Wartbarkeit des Codes haben eine hohe Priorität!

3. Vorgehen

Die Studierenden müssen sich in der ersten Phase in die neue Thematik einarbeiten. Das Projekt lässt sich zu Beginn gut in zwei Teile spalten die von den zwei Studierenden separat bearbeitet werden können.

- Webapplikation in Haskell (Webapp)
- Sensor und Aktoren ansteuern auf dem Raspberry Pi mit Haskell (RPI)

Webapp
Ein Studierender soll sich einen Überblick über die verfügbaren Libraries / Frameworks für das Backend und das Frontend verschaffen. Als Backend wird ein leichtgewichtiger Ansatz (REST / Websocket Server) bevorzugt. Für das Frontend muss entschieden werden, ob serverseitig oder clientseitig das HTML erzeugt wird, bzw. die Templates befüllt werden. Statt einem klassischen client-side Ansatz mit React Redux könnte das Frontend auch z.B. mit ReflexDom [1] in Haskell, in Elm [2] oder in Purescript [3] implementiert werden. Hierbei muss berücksichtigt werden, dass die Sensordaten live dargestellt werden sollen, z.B. mit Highcharts [4]. Zwischenziel ist eine minimale Webapp, die vom Server erzeugten Zufallszahlen asynchron aktualisiert und darstellt. Idealerweise lässt sich diese minimale Webapp bereits auf dem Raspberry Pi ausführen.

RPI
Der zweite Studierende soll sich als erstes darum kümmern wie man Haskell Code auf dem RPI ausführen kann. Die Verwendung eines Cross-Compilers hat sich als problematisch herausgestellt. Versuchen Sie direkt auf dem RPI zu kompilieren. Als Build Tool soll Stack verwendet werden.
Zwischenziel ist ein Programm, das eine LED blinken lässt.

Nach diesen ersten Erfolgserlebnissen soll eine Architektur definiert werden, die das Zusammenspiel zwischen der Webapp und dem hardwarenahen Teil definiert.

In der zweiten Phase soll nun die Happy Flowers Applikation implementiert werden. Auf der hardwarenahen Seite muss der Zugriff auf den Sensor und die Pumpe implementiert werden. Die Webapp muss die historischen Daten darstellen und die Darstellung aktualisieren, wenn neue Werte zur Verfügung stehen. Die Datenbank zur Speicherung der Messwerte und Schaltzeiten muss eingebunden werden. Das manuelle Bedienen der Pumpe muss mit einem Login geschützt werden.

4. Projekt Reporting

Für die Ablage von Dokumenten und Source-Code soll ein Git Repository verwendet werden.
Die Studierende soll den Betreuer wöchentlich in einer kurzen Status-Email (oder anlässlich eines Meetings) den aktuellen Stand / die erledigten Arbeiten / Ausblick auf die kommenden Wochen rapportieren.

5. Form des Resultats

Die Resultate der Arbeit sollen in einem Bericht präsentiert werden. Der Bericht soll sauber strukturiert sein, Umfang ca. 30-50 Seiten (ohne Quellcode). Der Bericht richtet sich an ein Fachpublikum. Haskell-Kenntnisse können vorausgesetzt werden.

Für den Bericht steht eine Dokumentenvorlage zur Verfügung:
\\Fsemu18.edu.ds.fhnw.ch\e_18_data11$\E1811_Info\E1811_Info_HT\Anleitungen_Vorlagen_Software\Vorlagen_Semester&Diplomarbeiten
Der Bericht kann aber auch mit LaTeX (oder mit anderen Technologien) gesetzt werden.

Sämtliche für diese Arbeit erstellten Dokumente und der erstellte Code sind auf einem Datenträger (CD-ROM) abzuspeichern und mit der Arbeit abzugeben.

Die Resultate der Arbeit sind auch auf einer Webseite zu veröffentlichen. Beispiele von solchen Webseiten finden Sie unter http://www.fhnw.ch/technik/bachelor/i/studium/studierendenprojekte/. Weitere Informationen zu diesen Webauftritten finden Sie auf der Projektplattform PLONE (falls sie keine Informationen dazu finden dann wenden sie sich direkt an markus.oehninger@fhnw.ch).

Zudem werden Sie die Resultate Ihrer Arbeit in einer halbstündigen Präsentation einem Fachpublikum präsentieren und danach Fragen beantworten.

6. Termine

Ausgabe der Arbeit:	Freitag, 23. September 2016
Zwischenmeetings:	Nach Vereinbarung
Abgabe der Arbeit:	Freitag, 20. Januar 2017
Präsentation:	Wird später definiert, nach Abgabe der Arbeit.

7. Kontakt

Betreuer: Daniel Kröni, Daniel.kroeni@fhnw.ch, IMVS, 5.2B16, +41 56 202 78 17

8. Referenzen

[1] https://github.com/ryantrinkle/reflex-dom
[2] http://elm-lang.org/
[3] http://www.purescript.org/
[4] http://www.highcharts.com/demo/dynamic-update