almond

autonomous logging and management of networked devices

Lastenheft

und Projektübersicht

Inhaltsverzeichnis

1 Übersicht **1**

2 Teilnehmer und Aufgabenverteilung **2**

3 Controller **2**
3.1 Aufbau . . . 2
3.2 Features . . . 2
3.2.1 Zeitplan . . . 3
3.3 Display . . . 3
3.3.1 Zeitplan . . . 3
3.4 Speicherung (SD-Karte) . . . 3
3.4.1 Zeitplan . . . 4
3.5 mögliche Erweiterungen . . . 4

4 Aktor/Sensor Hardware **4**
4.1 Aufbau . . . 4
4.2 mögliche Erweiterungen . . . 4

5 Protokolle **4**
5.1 Uplink . . . 4
5.2 Downlink . . . 4
5.3 Bluetooth . . . 4

6 Backend **5**
6.1 Aufbau . . . 5
6.2 Backend Bluetooth Interface . . . 5
6.3 Uplink Protokoll . . . 5
6.4 Web Oberfläche . . . 6
6.4.1 Funktionsweise . . . 6
6.5 Zeitplan . . . 6
6.6 mögliche Erweiterungen . . . 6

1 Übersicht

Almond (**A**utonomous **L**ogging and **M**anagement of **N**etworked **D**evices) ist ein System aus vernetzten Aktoren und Sensoren (sog. Nuts (Networked Utilities and Sensors)), die über einen zentralen Controller (Squirrel) gesteuert werden. Zusätzlich gibt es ein Backend für PCs, welches über eine Weboberfläche Zugriff auf Gerätedaten gibt und die Steuerung der Aktoren sowie Konfigurationsänderungen erlaubt.
Die einzelnen Geräte werden über ein dafür entwickeltes Protkoll via Bluetooth mit dem Controller verbunden. Der Controller selbst wird dann ebenfalls über Bluetooth mit dem auf einem Linux/Windows/Mac laufenden Backend verbunden. Dabei werden auch Logdaten (Verlauf der Sensorwerte) mit übermittelt, die dann später auf dem Rechner ausgewertet werden können.

2 Ziele

Designkriterien:

- kostengünstig im vergleich zu bestehenden Systemen
- erweiterbar
- einfache Benutzung
- wartungsarm
- einfache Installation

Es wäre für Entwickler ein Framework um drahtloses Steuern und Messen zu realisieren. Für Heimanwender einfache Möglichkeit Überwachung und Automatisierung kostengünstig zu benutzen.

3 Realisierung

- Drahtlosnetzwerk über Bluetooth
- Modulares Design durch generische Schnittstellen, damit beliebige neue Geräte, wenn auch eingeschränkt, sofort in Betrieb genommen werden können.

Das System besteht aus 3 Komponenten, einem Interface am Rechner für den Endanwender am PC, eine Zentrale Steuereinheit die die Komunikation mit den Sensoren und Steuereinheiten übernimmt und eine Reihe von Sensoren und Aktoren.

4 Teilnehmer und Aufgabenverteilung

Den 9 Teilnehmern sind folgende Aufgabenbereiche im Projekt zugeteilt:

- **Salomon Sickert**: Teamleiter, Downlink-/Uplink Protokoll Design, Controller
- **Pascal Schnurr**: Backend, Settings/Logging, Kommunikation mit Webinterface, Bluetooth Interface
- **Matthias Schwab**: Backend, Webinterface
- **Christian Rupprecht**: I/O Interface für Controller (SD Karte), ... (bitte ergänzen)
- **Seán Labastille**: Controller, Up-/Downlink Protokolldsign
- **Stefan Profanter**: Bluetooth Implementation Aktoren/Sensoren/Backend
- **Linus Lotz**: Boardesign Aktoren/Sensoren/Backend, ... (bitte ergänzen)
- **Thomas Parsch**: Displaysteuerung Controller, ... (bitte ergänzen)
- **Maximilian Karl**: Boardesign Aktoren/Sensoren/Backend, ... (bitte ergänzen)

Die einzelnen Projektteile werden im Folgenden genauer erläutert.

5 Controller

5.1 Aufbau

Der Controller (Squirrel) basiert auf einem Atmel XMega 128a1 AVR Mikroprozessor. Er dient als Zentrale für das System welches autark vom Rechner arbeitet, und auch als Schnittstelle zum Rechner. Am Controller selbst befindet sich zusätzlich ein SD-Kartenleser für die Speicherung von Logdaten, ein Display mit Tasten für Statusüberprüfung und ggf. Konfigurationsänderungen. Wie alle Client-Geräte (Nuts) besitzt der Controller ein Bluetoothmodul.

5.2 Features

Mit dem Downlink-Protokoll spricht die Squirrel seine Nuts an und empfängt Daten von diesen. Die Squirrel führt Buch über die ihr bekannten Geräte und speichert diese intern zwecks Identifizierung, Displayausgabe, Logverwaltung und Konfiguration durch das Backend.
Mit dem Uplink-Protokoll kommuniziert die Squirrel mit dem Backend. Hiermit werden Logdaten zum Rechner geladen und Konfigurationsdaten der Nuts und auch der Squirrel selbst ausgetauscht. Aus dem Entwurf des Downlink-Protokolls ergibt sich theoretisch die Möglichkeit 254 Geräte mit je insgesamt 254 Sensoren, Aktoren und Konfigurationseinstellungen anzusprechen. Die von diesen Geräten ausgegebenen Sensordaten werden geloggt, Aktoren können gesteuert werden und die Konfiguration entsprechend angepasst werden.

5.2.1 Zeitplan

1. Anfangs zielt die Entwicklungsarbeit auf die Übermittlung von Datenpacketen über Downlink zwischen Nuts und Squirrel. Sobald diese Problemlos möglich ist und sofern nicht bereits implementiert müssen bekannte und/oder sichtbare Geräte in einer internen Datenstruktur verwaltet werden. (2 Wochen)

2. Nun sollte, je nach Entwicklungsstand der anderen Komponenten, die Anzeige der bekannten Daten auf dem Display, die Übergabe per Uplink an den Rechner und das Loggen auf SD-Karte implementiert werden. (4 Wochen)

3. Hierauf folgt die Umsetzung von Konfigurationsmöglichkeiten per Uplink oder Display. (1 Woche)

5.3 Display

Die Grafische Benuteroberfläche (GUI) des Squirrels, wird mit Hilfe des ST7565R - ein 65 x 132 Dot Matrix LCD Controller - realisiert. Dieser wird durch ein Atmel Mikroprozessor angesteuert, sodass hier eine zusätzliche Abstraktionsschicht zwischen Display-Controller und Squirrel-Controller eingefügt werden kann. Dies ermöglicht eine leichtere Display-Ansteuerung und eine höhere Kapselung der GUI-Funktionen. Zur ersten Ansteuerung wird zunächst in C ein Grundprogramm entworfen, das die Ausgabe von Fehlercodes und Statusmeldungen bereitstellt. Anschließend wird eine GUI-Schnittstelle entworfen und im Wiki dokumentiert. Danach werden die Schnittstellenfunktionen in C für das Display implementiert. Abschließend werden Fehler beseitigt und die grafische Ausgabe weiter verbessert.

5.3.1 Zeitplan

1. Rascher Entwurf des Grundprogramms zum Ermöglichen erster Tests (1-2 Wochen)
2. Gründlicher Entwurf der Schnittstelle (2 Wochen)
3. Implementierung der Schnittstellen Funktionen (2-3 Wochen)
4. Verbesserungen in der Benutzerführung und der grafischen Darstellung (1-2 Wochen)

5.4 Speicherung (SD-Karte)

Der Controlle besitzt folgende Zugriffsmöglichkeiten auf SD-Karten:

1. SD-Karte mit FAT-32 formatieren
2. Log-File schreiben/lesen (ASCII file)
3. Conf-File schreiben/lesen (dabei werden Name-/Value Paare ausgewertet)
4. beliebige Textdatei schreiben/lesen

So ist es möglich auf die Daten auch außerhalb des Controllers, z.B. vom PC aus zugreifen zu können.

5.4.1 Zeitplan

rumgammeln

5.5 mögliche Erweiterungen

Eine wünschenswerte Erweiterung wäre die Möglichkeit timergesteuert Aktoren auszulösen oder auf Zustandsänderungen bei einem Sensor zu reagieren.

6 Aktor/Sensor Hardware

6.1 Aufbau

Hier kommt Text zum Controller hin..

6.2 mögliche Erweiterungen

7 Protokolle

7.1 Uplink

Verwendet für die Kommunikation zwischen Backend und Controller.
Über Bluetooth werden hier Pakete fester größe ausgetauscht um: Logdaten ans Backend zu übermitteln, die Konfiguration einzelner Nuts abzurufen oder zu ändern und schließlich die Squirrel selbst zu konfigurieren.

7.2 Downlink

Verwendet für die Kommunikation zwischen Aktor/Sensor Hardware und Controller.
Über Bluetooth werden Pakete fester größe ausgetauscht um: Daten von den Nuts abzurufen, d.h. Sensordaten und Geräteinformationen, sowie Konfigurationsänderungen und Aktoren auszulösen.

7.3 Bluetooth

Das Squirrel besitzt ein BTM-222 Bluetooth Modul zur Kommunikation mit den Nuts, welche ebenfalls das BTM-222 Modul verwenden, sowie mit dem Computer. Zur Verbindung mit dem Computer wird wird bereits ein existierendes Bluetooth Modul am PC vorausgesetzt. Um die Verbindung zwischen dem Microcontroller und dem BTM-222 Modul über USART herzustellen, wurde ein eigener Code entwickelt.
Sobal das Squirrel mit einem Nut oder dem PC verbunden ist, wird ein proprietäres Protokoll verwendet, um die Daten auszutauschen. (Siehe: 5.1 sowie 5.2). Da die Datenpakete jeweils eine feste größe haben, muss gewährleistet werden, dass im Falle einer Störung oder eines Übertragungsfehlers das Ende eines Paketes wieder erkannt wird. Deshalb werden am Ende eines jeden Paketes zwei spezielle Byte gesendet: das Special- und Stop-Byte. Dies kann man sich als '\0'vorstellen. Damit nicht zufällig mit den gesendeten Daten ein falsches Ende erkannt wird, wird vor jedem Byte, welches den selben Wert hat wie das Special-Byte ein weiteres Special-Byte eingefügt (z.B. '3\04' wird zu '3\\04' und dann erst übertragen). Beim Empfangen werden dann die doppelten Special Bytes wieder mit einem ersetzt und dadurch erkannt, dass die Bytes kein Ende darstellen.
Zur Kommunikation mit dem PC werden ebenfalls feste Datenpakete verwendet wobei dort auch das Prinzip mit den Special- und Stop-Byte Anwendung findet (weitere Details siehe 6.2).

8 Backend

8.1 Aufbau

Das Backend ist ein Softwarepaket für Linux, Windows oder Mac OS X basierte Computer. Mit dem Backend ist es möglich über eine Web basierte Oberfläche auf die Sensordaten der vernetzten Geräte zuzugreifen. Die Backend Software kommuniziert über ein Bluetooth-Protokoll mit der Controller Hardware. Dabei werden zum einen aktuelle Sensorwerte übermittelt, wie auch auf dem Controlle gespeicherte Log-Daten zum Verlauf der Sonsorwerte seit dem letzten Update.
Als Webserver kommt der **lighttpd** zum Einsatz. Zusätzlich wird eine lokale **MySQL** Datenbank zur Speicherung von Einstellungen und Daten auf der Backend-Seite verwendet.
Die Software zur Kommunikation mit dem Backend wird in C++ entwickelt und schreibt empfangene Daten direkt in die Datenbank.
Für die Darstellung der Web-Oberfläche wird eine in **Perl** entwickelte Software verwendet, die auf die selbe Datenbank zugreift um zB aktuelle Sensorwerte anzuzeigen oder Diagramme und Graphen mithilfe der verfügbaren Log-Daten zu erzeugen oder Einstellungen für den Controller zu ändern.
Der Vorteil der Weboberfläche ist, dass der Backend Server so von jedem Browser aus bedient

werden kann (inklusiver Mobiler Endgeräte) und so eine bestmögliche Integration des Systems ermöglicht wird.

8.2 Backend Bluetooth Interface

Der Backend Daemon muss regelmäßig neue Logdaten vom Controller leeren über Bluetooth und diese in die MySQL DB einspielen.
Hierbei muss natürlich das Bluetooth Interface zur Kommunikation mit dem Netz kompatibel zu der Controller Implementierung sein.
Außerdem muss er auf Inputs des Webinterfaces reagieren und die geänderten Einstellungen oder Aktionen für Aktoren sofort wieder per Bluetooth in das Netz einspielen.
Dies geschieht durch ein Event per Socket sobald Einstellungen in der DB geändert wurden, oder einen Aktion hinterlegt wurde.

8.3 Uplink Protokoll

Hier kommt die Spezifikation des Uplink Protokolls hin...

8.4 Web Oberfläche

8.4.1 Funktionsweise

Die Weboberfläche wird mit Perl über das CGI Interface des lighttpd Servers verfügbar gemacht. Es wird keine Perl Installation auf dem Rechner vorausgesetzt, da eine Version inklusive der benötigten Module bereits mitgeliefert wird. Ein MySQL Server wird auch mitgeliefert und mit dem Backend und Webserver gestartet.

8.4.2 Zeitplan

1. Einrichten der Server-Umgebung (2 Wochen)
2. Entwurf und Implementation des SQL Tabellen Layouts (1 Woche)
3. Entwurf des Backend Bluetooth Interfaces
4. Implementation von Web Oberfläche und Funktionalitäten

8.4.3 mögliche Erweiterungen

1. Benutzerverwaltung für das Webinterface
2. erweiterte Einstellungen und Rechtemanagement
3. Konfifurationseditor im Web
4. evtl. verschlüsselte Kommunikation mit dem Controller