SkyInvader

Mit einem Arduino sollen LED Strips, welche an der Decke montiert sind gesteuert werden.

Benötigte Hardware:

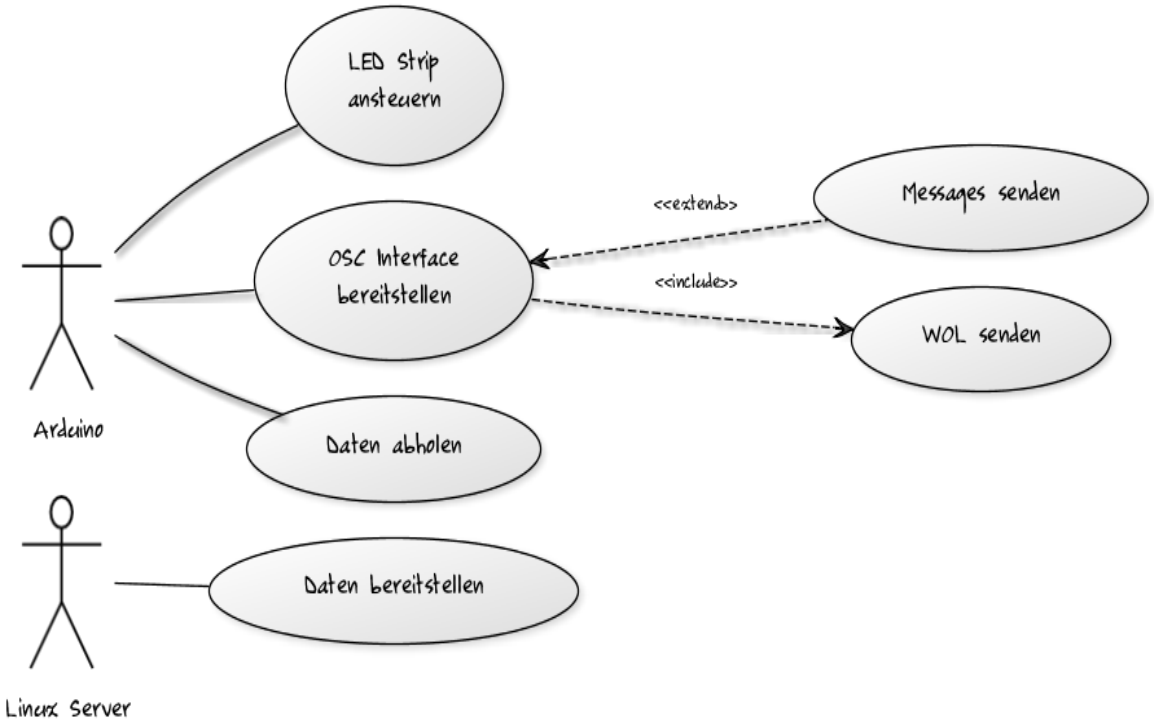
- 1 Arduino Ethernet oder iBoard, evtl Arduino MEGA + Ethernet Shield
- bis zu 10m WS2801 LED Strips, maximal 320 Pixel
- Netzteil 5V/100 Watt.
  Hinweis: 5V sollte an zwei Orten eingespeist werden.

Phase 1

mach ein einfaches softwareInterface. das arduino holt sich das Farbmuster und delay-bis-next-sample auf einer httpadresse ab.
. (Am Anfang einfache Tabellen mit Farbmuster).
Dieses Protokoll soll schlank und dynamisch sein (wakeup, sleep in beide Richtungen)
Der Server sollte ebenfalls in Standby gehen können und per arduino Wake-on-Lan geweckt werden).
ich werde auf einer öffentlichen Adresse gemäss deinen Angaben die HimmelsMuster bereitlegen.

Datum	Version	Info
21.08.2012	v1.0	Initial Version
22.08.2012	v1.01	UC1, UC3 und UC5 aktualisisert
31.08.2012	v1.02	Kapitel Weitere Informationen hinzugefügt

Use Case Diagram



http://yuml.me/edit/61effabc

UC1: LED Strip ansteuern

Der WS2801 LED Strip wird mit einem definierten Inhalt “gefüllt”. Dabei kann jeder Pixel unabhängig gesteuert werden.
Die Strips werden via BitBanging angesteuert, nicht via SPI. SPI wird für die Ethernet Funktionalität benötigt.
Es gibt drei Animations-Modi:

- **Statische Farbe**, die Farbe wird via OSC Messages eingestellt.
- **Color Set Animation** (Crossfading zwischen 3 Farben), es gibt verschiedene Colorsets (statisch im Code oder auf dem Server abgelegt) welche in einer definierten Geschwindigkeit überblendet.
- **Serverimage,** Infos werden vom Server abgeholt (nächste aktualisierung, Farbinhalt)

Library: https://github.com/neophob/WS2801-Library

UC2: OSC Interface bereitstellen

Das OSC Interface ermöglicht die Steuerung des Arduino uP von einem anderen System wie Android, iOS, OSX, Windows oder Linux. Dabei können verschieden Parameter (Float, Int, String) übergeben werden.
Library: https://github.com/neophob/ArdOSC

Optionale Erweiterung: Bonjour support, damit kann der Arduino mit einem definierten Namen (z.B. skyinvader.local) angesprochen werden. Diese Library benötigt viel Speicherplatz, je nach verwendeten Libraries und Arduino Hardware muss auf dieses Feature verzichtet werden.
Library: https://github.com/neophob/EthernetBonjour

UC3: OSC Messages senden
Via OSC sollen die Animations Modi und die Farbsets geändert werden können.

Folgende Messages könne geschickt werden:
- /mode FLOAT: Animationsmode wird umgestellt (1-3)
- /colR FLOAT: R Farbwert der **Statischen Farbe**
- /colG FLOAT: G Farbwert der **Statischen Farbe**
- /colB FLOAT: B Farbwert der **Statischen Farbe**
- /speed FLOAT: Animationsgeschwindigkeit (**Color Set Animation**)
- /colset FLOAT: Colorset wird umgestellt (**Color Set Animation**)

UC4: WOL Senden
Um den Linux Server remote zu starten, soll der Arduino ein WOL Paket senden.
Das Event wird via OSC getriggert, dabei wird eine OSC Message mit der MAC Adresse als Parameter gesendet.

Beispiele:
https://github.com/mikispag/arduino-WakeOnLan/blob/master/arduino-WakeOnLan.ino
http://arduino.seesaa.net/article/134495683.html

UC5: Linux Server, Daten bereitstellen
Der Linux Server stellt dem Arduino Daten zur Verfügung. Das Protokoll soll schlank und dynamisch sein. Folgende Daten sollten von Anfang an vorhanden sein:
- Delay bis zum nächsten update, in ms
- Farbmuster für den Animationsmode: **Serverimage**
- Optional: Colorsets für die Farbanimation, 3 Farbwerte als String oder Long (Animationsmode: Farbanimation)

UC6: Arduino, Daten abholen
Die Daten werden vom Linux Server abgeholt.

XML wird wohl zu heftig für den Arduino sein, daher schlage ich vor JSON zu verwenden.
- JSON: https://github.com/interactive-matter/aJson und http://interactive-matter.eu/blog/2010/08/14/ajson-handle-json-with-arduino/
- JSON: https://github.com/mikepage/Simple-Arduino-JSON-Client/blob/master/WebClientJson/WebClientJson.ino

JSON kann man z.B. auch mit Node.js verwenden, das vereinfacht das testen.
Der Arduino liefert eine eindeutige ID mit, damit der Arduino eindeutig identifiziert werden kann.

Offene Punkte:
- Wie wird der Linux Server im Arduino referenziert? DNS Name? Bonjour?
  -> Ich würde einen DNS Namen vorschlagen
- DHCP Server ist vorhanden für den Arduino?

Phase 2

Der Sky-Delivery-Agent kommt später. TBD.

Weitere Informationen

Das Arduino Ethernet Board verwendet einen Atmega328 Chip, welcher 2kB SRAM zur Verfügung stellt.

Ein WS2801 Pixel braucht pro Pixel 3 Bytes, Theoretisch kann ein Atmega328 Chip also 682 Pixel ansteuern resp. im RAM halten. Praktisch gesehen kann natürlich weniger RAM verwendet werden. Grob geschätzt verwendet ein Arduino rund 200 Bytes RAM (RTOS), zusätzliche Libraries (WS2801 Library, Netwerk Libraries, OSC, JSON..) brauchen natürlich ebenfals Memory.

Aktuell (31.8.2012) haben wir mit 50 Pixel noch 749 Bytes freien Speicher. bei 160 Pixel sind es noch 419 Bytes.

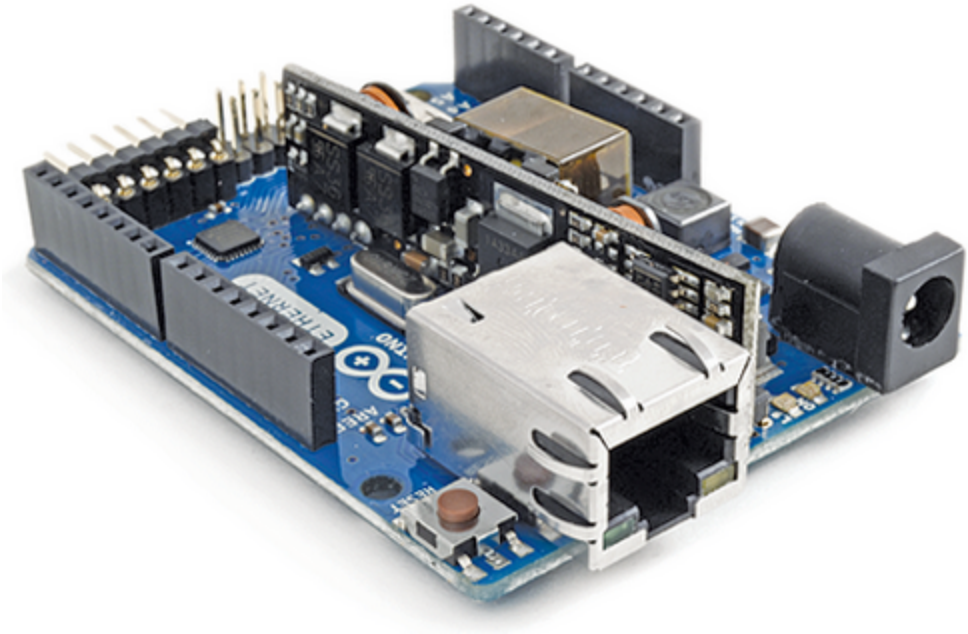
Hardware

Anforderungen:

- Ethernet Anschluss
- mind. 2kb freier Speicher

Arduino Ethernet

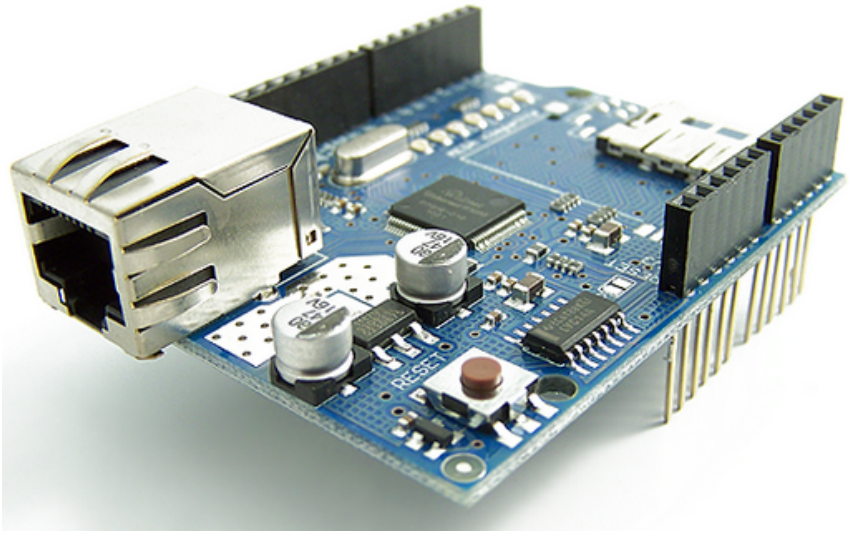
Mehr Informationen: http://arduino.cc/en/Main/ArduinoBoardEthernet



Preis ca. 55 SFr.
Eigenständiges Board (ohne Seriellen Anschluss, braucht Adapter-Board).

Arduino Ethernet Shield

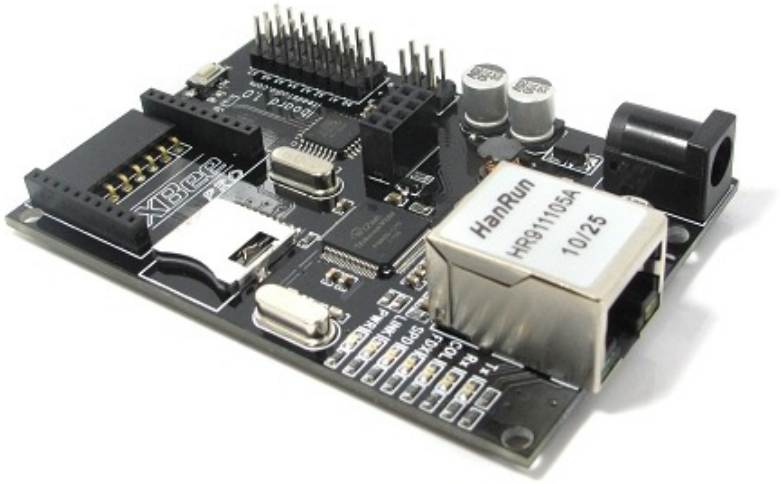
Mehr Informationen: http://arduino.cc/en/Main/ArduinoEthernetShield



Preis ca. 40 SFr.
Shield, braucht also ein Arduino (kompatibles) Board.

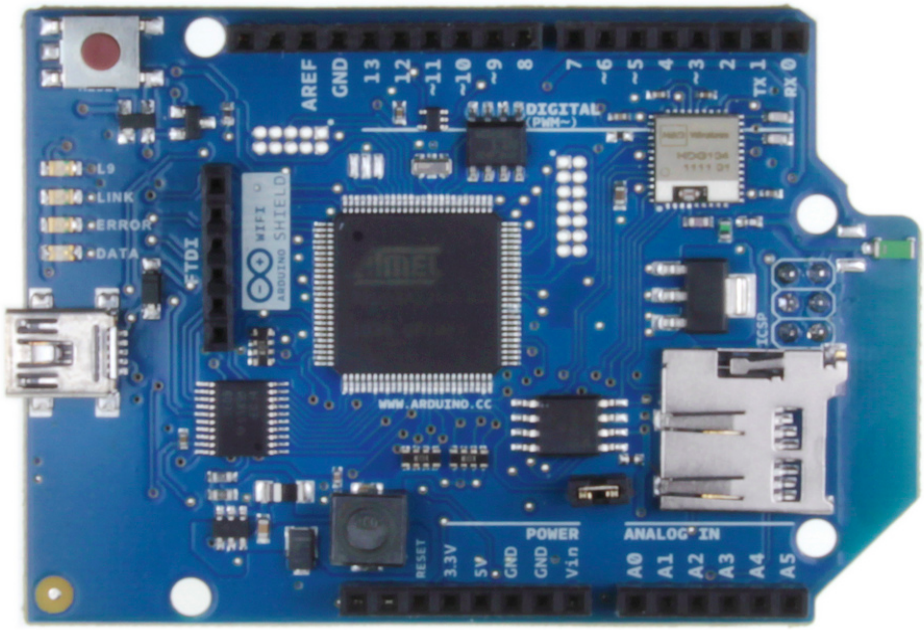
IBoard

Mehr Informationen: http://imall.iteadstudio.com/im120410001.html



Preis ca. 35 SFr.
Eigenständiges Board (ohne Seriellen Anschluss, braucht Adapter-Board), allerdings nicht Shield-kompatibel.

Arduino Wifi Shield



Preis ca. 95 SFr.
Teuer, dafür ohne Kabel.