SkyInvader

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

Benötigte Hardware:

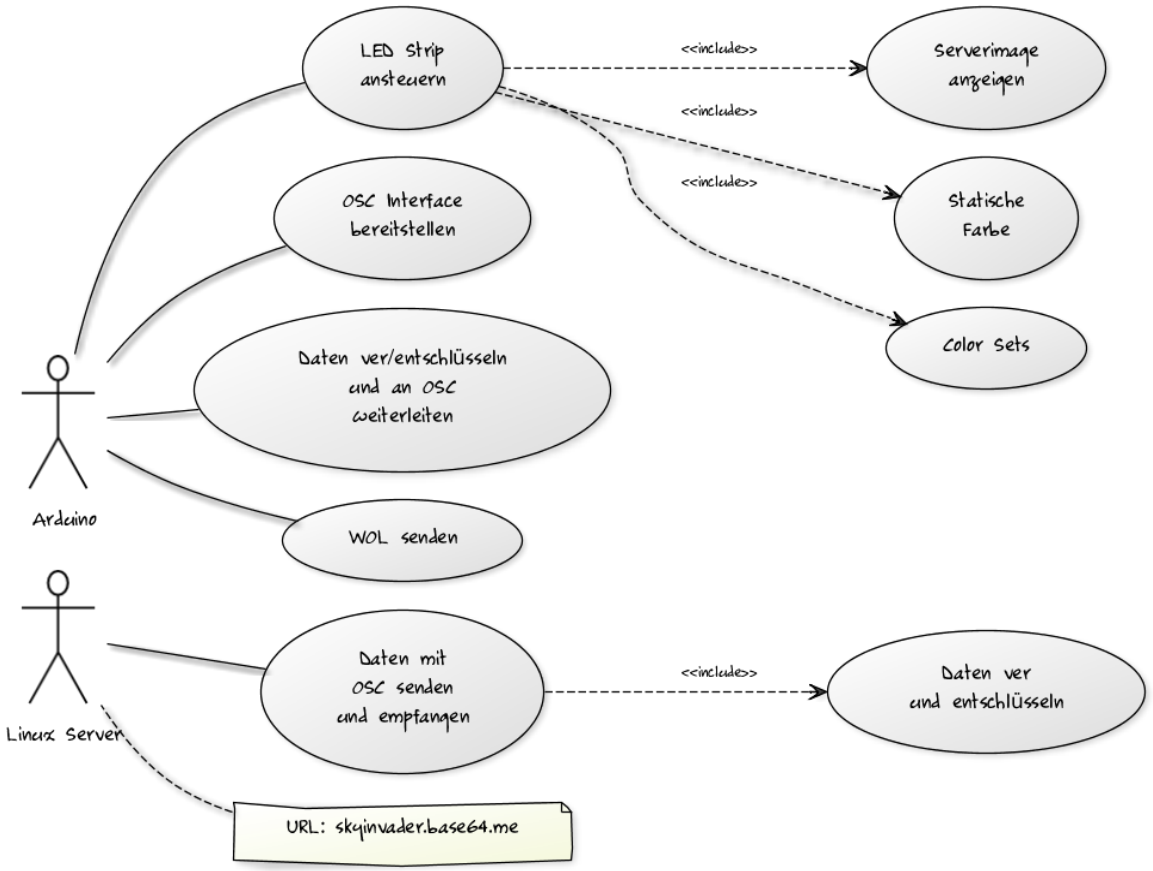
- 1 Arduino Ethernet oder iBoard, evtl Arduino MEGA + Ethernet Shield
- bis zu 10m WS2801 LED Strips, maximal 320 Pixel
- Netzteil 5V/100 Watt.
  Hinweis: 5V sollte an zwei Orten eingespeist werden.

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” und “Hardware” hinzugefügt
1.9.2012	v1.03	OSC Messages erweitert, UC6 als ungültig deklariert, UC5 Bearbeitet
4.9.2012	v1.04	UML Diagram aktualisiert, UC1 aktualisisert, UC5 aktualisiert, UC6 hinzugefügt
9.9.2012	v1.05	Augrund Platzproblemen auf dem Arduino wird auf die SD Funktionalität verzichtet. UC1 angepasst, UC5 angepasst, UC6 erneuert. Neue Anforderung: Ein Skyinvader muss ich irgendwo auf der Welt einstecken können. Per DHCP routet er sich ins Internet und meldet sich auf skyinvader.base64.me an. Fertig. Ab jetzt kann er vom Server aus gesteuert werden. ServerInferface wird mit PHP gemacht

Use Case Diagram



http://yuml.me/edit/b7fcff44

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,** ein Server sended via OSC den Bildinhalt

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**)
- /pxl INT INT INT INT INT: Pixel Buffer wird aktualisisert, der erste Parameter ist der Offset der Pixel, die restlichen 4 Parameter sind die Bilddaten (**Serverimage**)
- /wol STRING: Sended ein Magic Packet aufgrund der mitgelieferten MAC Adresse

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 Interface, Daten senden

Der Linux Server sendet dem Arduino Daten via OSC Protokoll. Folgende Daten werden geschickt:

- Farbmuster für den Animationsmode: **Serverimage**
- Sendet der Server eine zeitlang keine Daten (z.B. 10 Minuten), werden vorgefertigte Animationen von der SD Karte abgespielt.
- **Optional**: Animation (z.B. Feuer oder Wasser) wenn das Webcam Bild langweilig ist.
- **Optional**: Colorsets für die Farbanimation, 3 Farbwerte als String oder Long (Animationsmode: Farbanimation), noch nicht implementiert.

Die Software soll in PHP geschrieben werden.

Unter https://github.com/hairmare/PoscHP ist eine nicht fertig implementierte PHP Library vorhanden, wir werden jedoch diese Version verwenden: http://opensoundcontrol.org/implementation/open-sound-control-php (scheint kompletter zu sein)

UC6: Anmelden an Server (skyinvader.base64.me)

Ein Skyinvader muss ich irgendwo auf der Welt einstecken können. Per DHCP routet er sich ins Internet und meldet sich auf skyinvader.base64.me an. Fertig. Ab jetzt kann er vom Server aus gesteuert werden.

Offene Punkte:

- Verschlüsselung, eine Option wäre RC4
- Eine Alternative wäre, dass wir eine HMAC mitschicken - das OSC Paket müsste nicht verschlüsselt werden.
- OSC hat den Vorteil, dass dieses Protokoll offen ist und viele Appliaktionen OSC Unterstützen. Wir müssen die Verbindung zum Server verschlüsseln, OHNE dass bestehende Applikationen nicht mehr funktionieren.
- VPN ist keine Option

Vorschlag: die PHP Library ver und ent-schlüsselt die OSC Nachricht und schickt diese via Internet an den Arduino. Der Arduino überprüft ob das Pakets valid ist, wenn ja wird der entschlüsselte Inhalt an den OSC Port weitergeleitet.

Phase 2

Der Sky-Delivery-Agent kommt später. TBD.

ZigBee/XBee: http://examples.digi.com/category/get-started/
Braucht wohl keine zusätzliche Hardware, da diese Module als Serielle Devices (RX/TX) verwendet werden können.

<=== http://skyinvader.base64.me/?aid=<? $aid ?>&command=**poweroff_delay**
===> 360000 (milisec , nach einer Stunde ohne retrigger Licht aus und Ardunio in standby

<=== http://skyinvader.base64.me/?aid=<? $aid ?>&command=**poweroff_fade**
===> 10000 (milisec , wenn poweroff, langsam wegdimmen, damit ein Bewohner noch den Schalter für ein Retrigger finden kann
->

Das IBoard besitzt einen zigBee Stecker. Es müsste also möglich sein, damit LED-Strip mit IBoard via zigBee anzusteuern statt Ethernet.
Ein IBoard fungiert als Master am Ethernet, alle anderen werden via zigBee vom Master eingeschaltet und gefüttert.
Würde ich gerne haben wollen.
hmm ja tönt noch interessant… dann möchtest du no iboards haben, welche mit zigbee conectivity haben?
yup zigbee, weil Pir und feuchtesensoren und alle steckdosen via zigbee geschaltet werden.
um Dr. Hammond aus Jurassic Parc in etwa zu zitieren:
"wir haben weder Kosten noch Mühe gescheut um eine reales Erlebnis zu vermitteln."

Jetzt kommen die Pir zum Zuge.
Obige Erfassung der Position der LedPunkte wird für mehrere Punkte im Raum errechnet.
Ich wandere mit dem iPhone durch den Raum. Die Pir oder Kinect oder Wiiremote erfassen zusätzlich zur PixelLUT meine Position.
Wenn ich nach der, zugegebenermassen, komplizierten Kalibration durch den Gang laufe wechseln die LUT je nach Position. Ich sehe quasi immer dasselbe "Bild" egal von wo aus ich auf die Led schaue.
Abgesehen von der Detektion im Raum ist die Darstellung danach technisch simpel. Position 5 ergibt LUT-5 ferting

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) habe ich mit einem IBoard (2kB RAM) mit 50 WS2801 Pixel noch 749 Bytes freien Speicher, bei 160 WS2801 Pixel sind es noch 419 Bytes.

Auflistung benötiges Memory:

Name	**Anzahl Bytes**
Arduino RTOS	200
320 WS2801 Pixel	960
WOL Payload	102

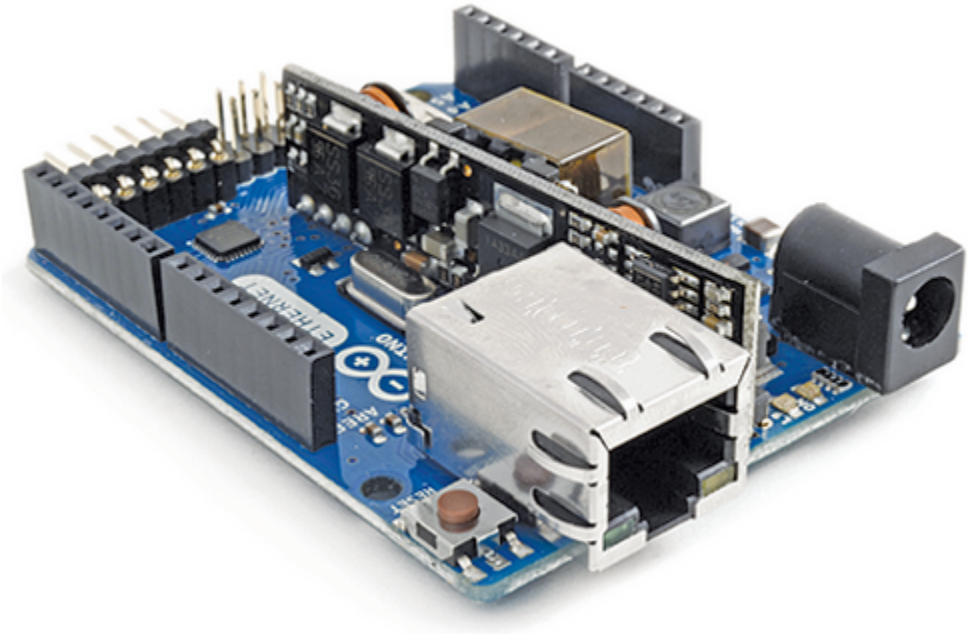
Hardware

Anforderungen:

- Ethernet Anschluss
- mind. 2kb freier Speicher

Arduino Ethernet

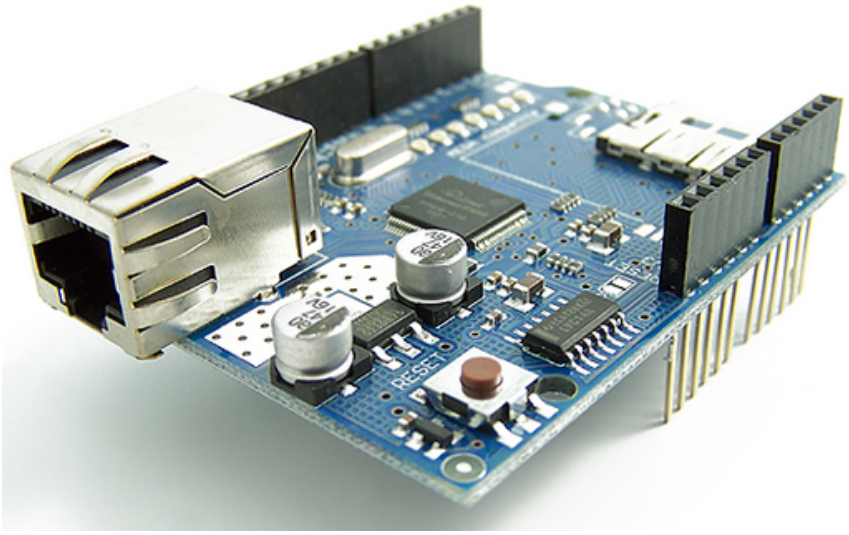
Mehr Informationen: http://arduino.cc/en/Main/ArduinoBoardEthernet



Memory: 2kB, Preis ca. 55 SFr.
Eigenständiges Board (ohne Seriellen Anschluss, braucht Adapter-Board).

Arduino Ethernet Shield

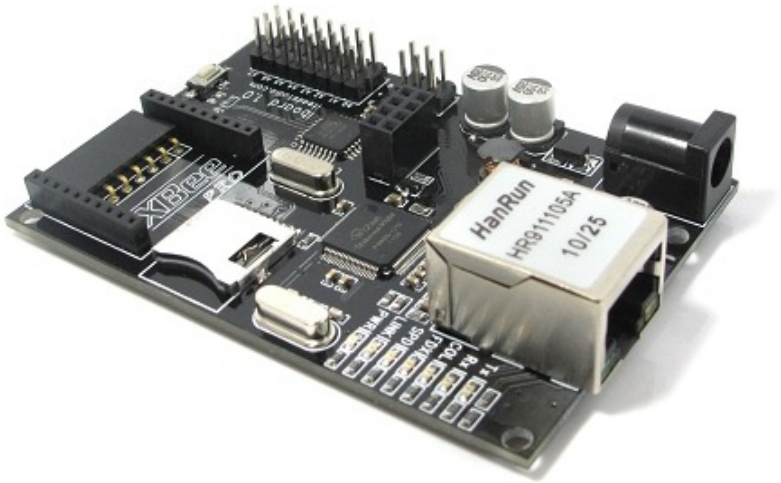
Mehr Informationen: http://arduino.cc/en/Main/ArduinoEthernetShield



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

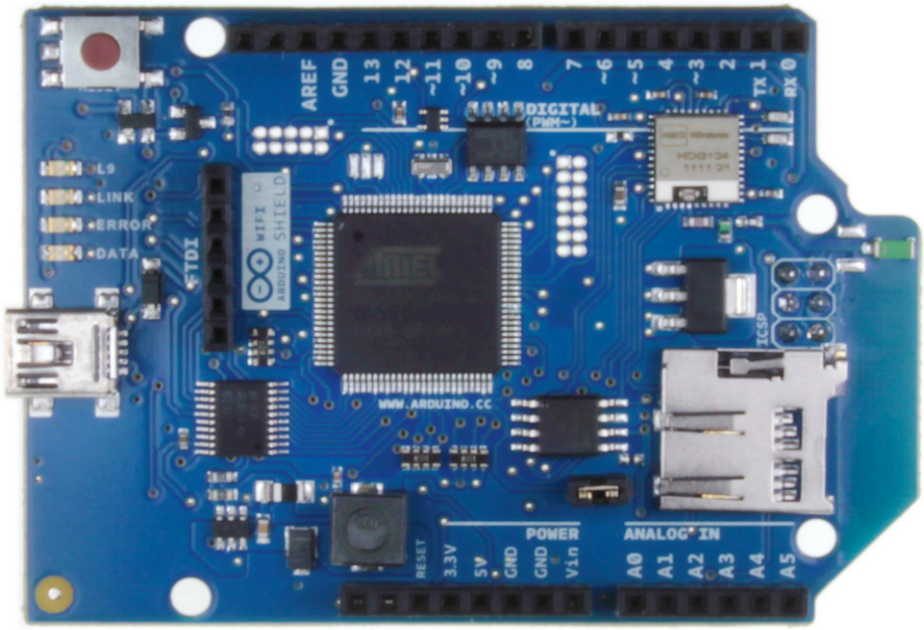
IBoard

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



Memory: 2kB, 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.

Weitere Interessante Boards:

- Arduino Mega 2560, 8kb SRAM, ca. 55SFr.
- Seeeduino Mega (Arduino Mega “klon”), 8kb SRAM, ca. 40SFr.
- Arduino Leonardo, 3,3kB SRAM, ca. 25SFr.

Appendix A, gelöschte Use Cases:

UC6: Aufnehmen

Daten welche vom Server geschickt werden, werden auf der SD Karte aufgenommen. Als Filenamen wird zb. die Frameanzahl verwendet.
Getriggert wird das Aufnehmen via OSC Meldung, aufgenommen werden eine bestimmte Frame Anzahl(z.B. 600 Frames).