BACHELORARBEIT

zur Erlangung des akademischen Grades

„Bachelor of Science in Engineering“ im Studiengang BEW

Erweiterung eines Tischfußballtisches für automatische Torzählung sowie Spielstandsanzeige via HDMI-Monitor und Webserver

Ausgeführt von: Alexander Peters
Personenkennzeichen: 1310255025

1. Begutachter: Dipl. Ing. Christoph Veigl

Wien 24.1.2015

FACHHOCHSCHULE
TECHNIKUM WIEN

Eidesstattliche Erklärung

„Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Ich versichere, dass die abgegebene Version jener im Uploadtool entspricht."

Ort, Datum

Unterschrift

Kurzfassung

Diese Arbeit behandelt die Entwicklung einer erweiterbaren Plattform für zukünftige Studentenprojekte. Als Träger dieser Plattform dient der Tischfußballtisch (TFBT) in der Mensa der Fachhochschule Technikum Wien.
Die Anforderungen sind die Erfassung geschossener Tore und die Anzeige auf einem extern angeschlossenem HDMI Display sowie über ein Webinterface welches über einen eigenen WLAN Hotspot ausgeliefert wird.

Für die Erfassung der geschossenen Tore kommt ein doppelter Lichtschranken zum Einsatz welcher in der Torwand des Tisches eingebaut ist. Um Fremdlichteinfluss zu vermeiden wird der Lichtschranken mit moduliertem Licht betrieben.
Die niederen für Ansteuerung und Auswertung übernimmt ein Arduino Mega welcher die Informationen zur weiteren Verarbeitung an den Raspberry Pi weiterleitet.
Von diesem Raspberry Pi werden die höheren Funktionen wie die Bereitstellung des WLAN Hotspots sowie die Auslieferung des Webinterfaces und die Ausgabe des HDMI Signals.
Auf dem Pi kommt das auf Debian basierende Raspbian/Linux zum Einsatz. Die Bereitstellung des WLAN Zugangs wird durch die Programme hostapd und dnsmasq bewerkstelligt, als Webserver wird Node.js eingesetzt.
Das im Netzwerk abrufbare Webinterface passt sich automatisch der Bildschirmauflösung des aufrufenden Gerätes an und wird auch für die Anzeige auf dem externen HDMI Display eingesetzt. Hierzu kommt der Browser Chromium zum Einsatz welcher direkt am Raspberry im Kiosk Modus gestartet wird.

Mit Abschluss dieser Arbeit steht der Tischfußballtisch mit der geforderten Ausstattung zur Verfügung. Für zukünftige Arbeiten ist eine Erweiterung um die Erstellung von Bestenlisten oder den Antrieb der verschiebbaren Würfel zur Spielstandsanzeige vorgesehen.

Schlagwörter: Tischfußballtisch, Raspberry Pi, Arduino, Lichtschranke, Linux, WLAN

Abstract

This work examines the development of an extensible platform for future projects done by students. As basis of the platform the canteen's tabletop soccer is used.
The requirements are the detection of shot goals and displaying on an external HDMI connected screen as well as over a web interface delivered via a WIFI Hotspot

For detection of shot goals a double light barrier built in the goal wall is used. To prevent influence of ambient light the light barrier works with modulated light.
The low level functions necessary to drive the light barrier with modulated light and detect the output are done on a Arduino Mega which forwards the information to the Raspberry Pi. This Raspberry Pi provides the WIFI hotspot, the web interface and HDMI signal for the external display.

The Raspberry Pi runs a Debian/Linux based operating system called Raspbian. To provide the WIFI hotspot the programs hostapd and dnsmasq are used. Node.js is used as web and application server.
The web interface accessible within the provided hotspot adapts to the screen size of the viewing device and is also used for the HDMI Display. Therefor the Chromium browser is started locally on the Raspberry in kiosk mode accessing the webserver on local host.

At the time of writing the tabletop soccer is provided with the required features. For future development it is proposed to extend the platform with game statistics and leaderboards or a system to drive the sliding cubes to indicate the state of play.

Keywords: Tabletop soccer, Raspberry Pi, Arduino, light barrier, Linux, WIFI

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit behandelt die Entwicklung einer erweiterbaren Plattform für zukünftige Studentenprojekte. Als Träger dieser Plattform dient der Tischfußballtisch (TFBT) in der Mensa der Fachhochschule Technikum Wien.
Aufgabe der zu entwickelnden Plattform ist es eine die nötige Infrastruktur für Anwendungen im Bezug auf den TFBT zu bieten.
Zusätzlich zur reinen Entwicklung der Plattform wird unter Zuhilfenahme dieser eine einfache Torzählung mit Webinterfache als Beispielanwendung umgesetzt.

1.1 Motivation

Die Motivation für dieses Projekt ist es den Studierenden eine Basis für Projekte im Bereich embedded Systems zur Verfügung zu stellen welche zum einen auch nach Ablauf des Projektes einen Nutzen hat und zum anderen für alle sichtbar ist.
Dies fördert sowohl die Motivation derjenigen die das Projekt umsetzen als auch das Interesse derjenigen welche die Ergebnisse direkt im Studentenalltag erleben können.

Abbildung 1: Fertiger "Wuzzler" am vorgesehenen Aufstellungsort

1.2 Aufgabenstellung

Um sinnvolle und anspruchsvolle Applikationen entwickeln zu können ist es notwendig dass zum einen die Erkennung der relevanten Ereignisse möglich ist und zum anderen eine zeitgemäße Form der Ausgabe ermöglicht wird. Aus dieser Grundannahme ergeben sich die folgenden Anforderungen:

1.2.1 Torerkennung

Wird ein Tor geschossen so muss dies eindeutig und zuverlässig erkannt werden. Hierbei ist darauf zu achten dass der Ball erkannt wird unabhängig von der Stelle an der er die Torlinie passiert.

1.2.2 Webinterface

Die Anzeige von Spielständen und alle anderen Funktionen sollen über ein Webinterface zugänglich sein. Hierbei ist darauf zu achten dass Der TFBT auch von Smartphones und Tabletts bedient werden kann.

1.2.3 WLAN Hotspot

Zur Verwendung des TFBT unabhängig von vorhandener Netzwerkinfrastruktur ist ein integrierter WLAN Hotspot notwendig. Dieser soll ein freies WLAN anbieten worüber beliebige Clients Zugang zu den angebotenen Oberflächen bekommen. Da in diesem Netzwerk kein Internet zur Verfügung steht können alle DNS Abfragen auf den Webserver der Plattform aufgelöst werden.

1.2.4 HDMI Ausgabe

Es steht ein FullHD TV mit HDMI Eingang zur Verfügung. Zusätzlich zur Ausgabe auf mobilen Endgeräten soll der aktuelle Spielstand und andere zur Benutzung notwendige Informationen auf diesem TV ausgegeben werden.

2 Analyse möglicher Plattformen

Am Markt gibt es eine Vielzahl von embedded Plattformen. Gerade in den letzten Jahren ist die Auswahl stark gewachsen. In diesem Zusammenhang wird bei der Bewertung der Plattform immer die Kombination aus Hardware, Software und den zur Verfügung stehenden Werkzeugen betrachtet.

Zusätzlich zu den Hardwareanforderungen die sich direkt aus der Aufgabenstellung ergeben soll bei der Auswahl darauf geachtet werden dass die erforderlichen Werkzeuge und Informationen frei erhältlich sind. Besonders Augenmerk ist auch auf die vorhandene Community zu legen da eine aktive Community die Entwicklung weiterer neuer Projekte stark vereinfacht.

2.1 Arduino

Arduino [3] ist eine Plattform bestehend aus verschiedenen Mikrocontrollerbasierenden Boards und einer dazu passenden IDE. Ziel des Projektes ist es einen einfachen Einstieg in die Programmierung von embedded Hardware zu bieten. Arduino Projekte benutzen üblicherweise kein Betriebssystem sondern laufen direkt auf der Hardware. Somit sind Echtzeitaufgaben gut lösbar, die Entwicklung komplizierterer Systeme ist jedoch vergleichswiese aufwändig oder aufgrund der limitierten Hardwareressourcen nicht möglich. Zusätzlich zum den verschiedenen Basisboards welche lediglich über den Mikrocontroller und die nötige Grundbeschaltung verfügen gibt es diverse Erweiterungsboards für verschiedenste Anwendungen wie zum Beispiel Motorsteuerungen, Soundausgabe, Displays und Vieles mehr.
Auf der Homepage des Projektes gibt es umfangreiche Beschreibungen und Beispielprogramme welche den Einstieg sehr einfach gestalten. Die Community ist sehr anfängerfreundlich.

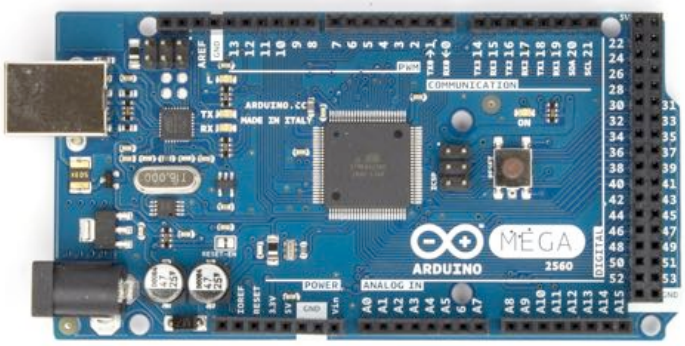

Abbildung 2: Arduino Mega, siehe: [1]

2.2 OpenWRT

OpenWRT [4] ist eine minimale Linux Distribution die vorwiegend für den Einsatz auf Routern und anderer Netzwerkhardware entwickelt wurde. Kompatible Hardware ist ab ungefähr 20€ erhältlich und verfügt üblicherweise über WLAN und mehrere LAN Ports. Aktuelle Hardware verfügt über MIPS Prozessoren mit 400-800MHz, 16-512MB RAM und 4-32MB Flash. USB und teilweise eSATA Schnittstellen sind ebenfalls verbreitet.
Ein Vorteil von OpenWRT gegenüber anderen embedded Systemen dieser Klasse ist der Paketmanager und das schreibbare Filesystem.

2.3 Raspberry Pi

Der Raspberry Pi [5] ist ein Einplatinencomputer basierend auf dem Broadcom BCM2835 System on Chip (SOC). Das Ziel der Entwicklung war es einen günstigen Computer bereit zu stellen um Kindern den Einstieg in die Programmierung zu ermöglichen.

Abbildung 3: Raspberry B, siehe: [2]

Der Raspberry zeichnet sich durch vergleichsweise hohe Leistung zu einem sehr niedrigen Preis aus. Der Raspberry Pi bietet je nach Ausführung bis zu 4 USB Host Ports, einen 100MBit Ethernet Port, HDMI Ausgang, Audio Ausgang, und eine Reihe an GPIO Pins. Er Verfügt über einen ARM6 Prozessor mit 700MHz und 512MB RAM, Zur Speicherung von Programmen und Daten kommt eine SD oder microSD Karte zum Einsatz.
Anders als bei den meisten embedded Plattformen ist es üblich auf dem Pi ein vollwertiges Linux zu installieren. Eine speziell für den Pi modifizierte Version von Debian ist Raspbian.

2.4 Android

Durch die Starke Verbreitung von Smartphones in den letzten Jahren ist Hardware mit Android Betriebssystem stark verbreitet und günstig erhältlich.
Die Leistung reicht je nach Hardware durchaus an einfache x86 Systeme heran. Probleme entstehen durch das nur teilweise offene Betriebssystem und die Notwendigkeit von closed-source Hardwaretreibern.
Die Hardware unterliegt einem sehr kurzen Lebenszyklus und alte Hardware wird oft nicht von neuen Softwarereleases unterstützt.

2.5 x86

Embedded Systeme auf x86 Basis verfügen üblicherweise über Intel Atom oder AMD Turion, selten findet man auch AMD GEODE Prozessoren.
Der Stromverbrauch sowie Preis liegt durchgehend deutlich über den zuvor genannten Systemen dafür ist die Auswahl an Software auch umfangreicher. Am ehesten
konkurrenzfähig ist das zuvor genannte Raspberry Pi welches jedoch aufgrund der ARM Architektur eigens kompilierte Software erfordert.

3 Lösungsstrategie

3.1 Torerkennung

Zur Erkennung der Tore soll eine Lichtschranke zum Einsatz kommen. Um Fremdlichteinfluss zu unterdrücken soll diese moduliert werden.
Am Markt gibt es eine Reihe von IR Empfängern mit eingebautem 38kHz Demodulator. Neben dem fertigen Empfänger IC wird nur noch eine IR Sendediode benötigt welche mit 38kHz gepulst wird. Da das Tor mehr als doppelt so hoch ist wie der Durchmesser des Balles wird es nötig sein 2 Lichtschranken pro Tor einzusetzen um eine Erkennung über die gesamte Fläche gewährleistet werden kann.

3.2 Webinterface

Das Webinterface soll aus einer einzelnen Seite bestehen die den aktuellen Spielstand sowie Hinweise zu den aktuell möglichen Optionen anzeigt. Das design soll so gewählt werden dass es auf allen gängigen Displaygrößen angezeigt werden kann.
Technisch gesehen soll es mit den gängigen Standards HTML5, CSS und JavaScript umgesetzt werden. Um Updates ohne Verzögerung darstellen zu können soll die Kommunikation über einen persistenten Socket abgewickelt werden.

3.3 WLAN Hotspot

Der WLAN Hotspot soll nach Möglichkeit direkt vom Raspberry bereit gestellt werden. Sollte sich dies als nicht möglich erweisen so besteht auch die Möglichkeit diese Aufgabe in einen günstigen WLAN Router auszulagern. Wichtig wäre in diesem Fall dass der DNS Server frei konfiguriert werden kann was einen OpenWRT-fähigen Router nahe legt.

3.4 HDMI Ausgabe

Um Mehrfachaufwand zu vermeiden soll das vorhandene Webinterface auch per HDMI ausgegeben werden. Hierfür ist geplant bei Start automatisch einen Browser zu Starten welcher auf den lokalen Webserver zugreift.

4 Umsetzung

Dieses Kapitel behandelt die Umsetzung des Projekts vom Hardwareaufbau, Einbau in den TFBT sowie das Aufsetzen des Raspberry Pis und einrichten der notwendigen Software.

4.1 Installation des Betriebssystems

Als Betriebssystem kommt Raspbian[7], eine für den Raspberry angepasste Version von Debian Linux, zum Einsatz. Die Installation kann über den NOOBS Installer[8] oder durch manuelles Kopieren eines fertigen Images auf die SD Karte erfolgen. Ein fertiges Image steht wie auch der NOOBZ Installer auf der Webseite der Raspberry Pi Foundation zur Verfügung.

Um das Image auf die SD-Karte zu kopieren kommt das Werkzeug dd zum Einsatz. Dieses steht unter Linux wie den meisten anderen UNIX ähnlichen Betriebssystemen zur Verfügung. Der Vollständige Befehl lautet:

```
#> dd if=2014-09-09-wheezy-raspbian.img of=/dev/disk4 bs=1m
```

Dabei ist 2014-09-09-wheezy-raspbian.img durch den tatsächlichen Pfad des Images zu ersetzen und /dev/disk4 durch den Pfad der zu bespielenden SD Karte.
Zu Beachten ist dass dd während des Vorganges der bis zu einigen Minuten benötigen kann keine Ausgabe erfolgt. Um eine Ausgabe zu erzeugen kann das Werkzeug pv benutzt werden. Der Befehl lautet dann:

```
#> pv 2014-09-09-wheezy-raspbian.img|dd of=/dev/disk4 bs=1m
```

Im Gegensatz zu dd ist pv auf den meisten Systemen nicht installiert kann jedoch über den Paketmanager der jeweiligen Distribution bzw. unter OSX mittels brew installiert werden.

Beim Schreiben auf die SD Karte kann es zu einem Fehler kommen wenn dem Benutzer die entsprechenden Rechte fehlen – in diesem Fall muss der Befehl als Superuser ausgeführt werden.
Sobald der Vorgang abgeschlossen ist kann die Speicherkarte in den Pi eingesetzt und dieser gestartet werden.

Ohne weitere Einstellungen ist es nun möglich sich per SSH anzumelden:

```
#> ssh pi@raspberrypi.local
```

4.2 Erweitern der root Partition

Die zuvor erstellte Speicherkarte verfügt über eine 2GB große root Partition. Um die volle Kapazität der eingesetzten Speicherkarte nutzen zu können muss die root Partition erweitert werden. Dafür steht das Tool raspi-config[9] zur Verfügung. Das Tool startet automatisch beim ersten Boot. Es kann auch nachträglich durch Eingabe des Befehls

```
#> raspi-config
```

gestartet werden. Nach dem Erweitern der Partition kann das Tool beendet und das System neu gestartet werden.

4.3 Aktualisieren des Systems

Nach der Installation des Basissystems ist es ratsam das System auf den neuesten Stand zu bringen. Dies erfolgt über den Paketmanager apt und das zugehörige Werkzeug aptitude:

```
#> sudo aptitude update
#> sudo aptitude upgrade
```

4.4 WLAN Accesspoint

Um einen WLAN Accesspoint konfigurieren zu können muss zuerst die entsprechende Hardware installiert werden. In diesem Fall kommt ein USB Dongle vom Typ TP-Link 723N V2 zum Einsatz. Dieser sollte von Raspbian ohne weitere Maßnahmen unterstützt werden. Erfahrungen in früheren Projekten haben jedoch gezeigt dass manche Hersteller verschiedene Hardware unter der gleichen Produktbezeichnung verkaufen. Es kann also hier also zu Problemen kommen.

Außerdem ist bei er Auswahl des Dongles darauf zu Achten dass nicht alle Modelle den Einsatz als Accesspoint erlauben.

Ob der Dongle ordnungsgemäß erkannt wurde lässt sich durch Eingabe des Kommandos:

```
#> ifconfig
```

überprüfen. Wurde der Stick korrekt erkannt wird nun die Schnittstelle wlan0 angezeigt.

Für die WLAN Funktionalität kommt das Programm hostapd[11] zum Einsatz. Dabei handelt es sich um einen Daemon welcher einen IEEE 802.11 kompatiblen WLAN Accesspoint bereitstellt. Es werden verschiedene Authentifizierungsmethoden unterstützt darunter WEP, WPA. In diesem Fall wird jedoch auf eine Absicherung des Netzwerkes verzichtet. Hostapd kann unkompliziert mittels aptitude[10] installiert werden:

```
#> sudo aptitude install hostapd
```

Da der verwendete WLAN Dongle nicht von der im Repository enthaltenen Version von hostapd unterstützt wird muss das gelieferte Binary durch eine angepasste Version ersetzt werden. Der Rest des Packages kann weiterverwendet werden. Die angepasste Version wird anhand dieser [12] Anleitung installiert.

Danach muss die Konfiguration in der Datei ***/etc/hostapd/hostapd.conf*** eingetragen werden:

```
interface=wlan0
ssid=wutzer
channel=6
driver=rtl871xdrv
```

4.5 Dnsmasq

Das Programm dnsmasq[13] ist ein einfacher DNS und DHCP Server der vor allem für kleine Netze entwickelt wurde. Es unterstützt die Vergabe von IP Adressen an festgelegten Interfaces sowie die Auflösung von DNS Anfragen. DNS Anfragen werden entweder anhand der Konfigurationsdateien ***/etc/dnsmasq.conf*** oder ***/etc/hosts*** aufgelöst.

Im vorliegenden Fall werden alle einkommenden Anfragen auf die eigene IP Adresse aufgelöst.

Die Installation erfolgt wieder über die offiziellen Paketquellen:

```
#> sudo aptitude install dnsmasq
```

Folgende Zeilen sind in der Datei ***/etc/dnsmasq.conf*** einzutragen

```
interface=wlan0
dhcp-range=192.168.42.100,192.168.42.200,12h
address=/#/192.168.42.1
```

Danach kann der Service durch Eingabe von

```
#> /etc/init.d/dnsmasq restart
```

neugestartet werden.

Verbindet sich nun ein Client mit dem angebotenen WLAN so sollte dieser eine IP Adresse im 192.168.42.0/24 Netzwerk erhalten. Die Funktion des DNS Servers kann durch einen DNS Lookup überprüft werden. Unabhängig der angegeben Adresse muss immer auf die IP 192.168.42.1 aufgelöst werden:

```
#> nslookup test.com
Server:   192.168.42.1
Address:  192.168.42.1#53

Non-authoritative answer:
Name:     test.com
Address: 192.168.42.1
```

4.6 Node.js

Als Webserver kommt node.js[14] zum Einsatz. Node.js basiert auf der von Google für den Browser Chrome entwickelten V8 JavaScript Engine. Zusätzlich zur reinen Ausführung von JavaScript gibt es eine breite Fülle an nativen Erweiterungen welche den Zugriff auf Systemressourcen wie Netzwerk und Dateisystem ermöglichen. Dadurch eignet sich Node.js hervorragend als Server für dynamische Webanwendungen.

Auch hier findet sich fertiges Paket im Repository von Raspbian welches jedoch veraltet ist. Eine Aktuelle Version steht jedoch als Binary auf der offiziellen Seite des Projektes zur Verfügung.
Die Installation funktioniert wie folgt:

```
#> wget http://nodejs.org/dist/v0.10.12/node-v0.10.12-linux-arm-pi.tar.gz
#> tar xvfz node-v0.10.12-linux-arm-pi.tar.gz
#> sudo mv node-v0.10.12-linux-arm-pi /opt/node/
#> echo 'export PATH="$PATH:/opt/node/bin"' >> ~/.bashrc    source ~/.bashrc
```

Um ein neues Projekt zu beginnen erstellt man zuerst ein Directory und initialisiert dann mittels npm ein neues Paket.

```
#> mkdir wuzzler
#> cd wuzzler
#> npm init
```

Um ein neues Projekt zu beginnen erstellt man zuerst ein Directory und initialisiert darin mittels npm ein neues Paket.
Hierdurch wird die package.json Datei angelegt welche sämtliche Informationen wie unter anderem Name, Version, Autor und Abhängigkeiten enthält. Diese kann nachträglich auch manuell verändert werden.

4.7 Express.js

Express.js[15] ist ein Framework für die Entwicklung von Webapplikationen. Es enthält unter anderem eine Reihe nützlicher Funktionen für das Ausliefern von statischem und dynamischen Content per HTTP(S) sowie ein umfangreiches Session Management.

Um express.js zu laden und den Abhängigkeiten hinzuzufügen wird npm verwendet:

```
#> npm install express --save
```

Danach kann es mittels require() innerhalb des Projektes verwendet werden.

4.8 Websockets und Socket.io

Die Kommunikation mit dem Server wird über eine Websocket Verbindung abgewickelt.
Websockets ermöglichen wie herkömmliche TCP Sockets eine bidirektionale Kommunikation. Der Vorteil gegenüber herkömmlichen HTTP Anfragen ist dass der Server bei Websockets jederzeit Nachrichten an den Client schicken kann während bei reinem HTTP der Server periodisch gepollt werden muss. Dies bedeutet zum einen erheblichen Overhead und zum anderen sind Updates in Echtzeit nicht oder nur bei extrem hoher Pollingfrequenz möglich. Um einen Websocket zu öffnen sendet der Client einen normalen HTTP Request und setzt dabei das Feld „Upgrade“ im HTTP-Header. Unterstützt der Server Websockets so bestätigt dieser das Upgrade und hält den Socket offen. Nun können darüber bidirektional Daten gesendet werden.

Eine Alternative ist das sogenannte Longpolling bei dem der Server eine Anfrage erst beantwortet wenn Daten vorhanden sind.
.
Socket.io[16] stellt für Server wie Client geeignete Libraries bereit welche auch einen Fallback auf Longpolling implementieren. Da alle modernen Browser jedoch bereits Websocket unterstützen sollte dieses Fallback nicht nötig sein.

Die Installation auf Serverseite erfolgt erneut über npm:

```
#> npm install socket.io --save
```

Für den Client stellt socket.io ebenfalls eine Library bereit welche gemeinsam mit dem Server Modul installiert wird und um Client wie folgt eingebunden werden muss:

```
<script src="/socket.io/socket.io.js"></script>
```

4.9 Webinterface

Das Webinterface wird rein mit den üblichen Standards HTML5, CSS und JavaScript umgesetzt.

Um auf verschiedenen Auflösungen und Bildschirmgrößen zu funktionieren wurde die Größe aller Elemente sowie die Schriftgröße relativ zur Bildschirmgröße angegeben. Für Elemente wie Tabellen ist dies seit langem gut unterstützt und sollte auch auf älteren Browsern nicht zu Problemen führen. Wie Angabe der Schriftgröße relativ zum umgebenden Element ist jedoch ein vergleichsweise neues Feature. Um den Entwicklungsaufwand in Grenzen zu halten wurde jedoch ausschließlich auf den WebKit Browsern Chrome und Safari getestet.

Abbildung 4: Screenshot Webinterface

4.10 Portweiterleitung und Autostart

Da Node nicht mit root-Rechten ausgeführt werden soll ist es dem Prozess nicht möglich sich an Port 80 zu binden. Daher muss eine Weiterleitung von Port 80 auf Port 3000 eingerichtet werden. Dies kann sehr einfach mit iptables[17] realisiert werden:

```
#> sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3000
```

Diese Regel bewirkt dass alle eingehenden Verbindungen von Port 80 an Port 3000 Weitergeleitet werden. Der Service ist also nun über beide Ports erreichbar. Da diese Regel nach einem Neustart nicht erhalten bleibt muss sie bei jedem Start erneut eingetragen werden.
Daher werden diese Zeilen in der Datei ***/etc/rc.local*** eingetragen:

```
#forward PORT 80 to 3000 for node applications
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3000

#start wuzzler app on boot
su pi -c "/opt/node/bin/node /home/pi/source/app.js" &> /var/log/wuzzler.log &
```

Dies Bewirkt dass diese Befehle am Ende des Systemstarts ausgeführt werden.
Wichtig ist zu beachten dass alle Befehle mit root rechten ausgeführt werden – node sollte daher per **su** als User pi gestartet werden. „***&>“*** bewirkt eine Umleitung von stdout und stderr ein das angegebene File ***/var/log/wuzzler*** und das abschließende ***„&“*** sorgt dafür dass der Prozess im Hintergrund weiter ausgeführt wird aber das Skript weiter läuft ohne auf das Ende des Node.js Prozesses zu warten.

4.11 Ausgabe per HDMI

Um das Webinterface auf einem fest installierten Display anzuzeigen soll eine Ausgabe per HDMI erfolgen. Dazu kommt der mitgelieferte x11 Server und das Desktop Environment in Verbindung mit dem Webbrowser Chromium[18] zum Einsatz.

Um den Webbrowser automatisch zu starten muss folgende Zeile am Ende der Konfigurationsdatei des Windowmanagers LXDE[19] ***etc/xdg/lxsession/LXDE/autostart*** eingetragen werden:

```
@chromium --kiosk --incognito http://localhost:3000
```

Gibt man nun auf der Kommandozeile den Befehl

```
#> startx
```

ein so startet erst die Grafische Oberfläche und danach der Browser, Der ganze Vorgang dauert fast eine halbe Minute dies ist jedoch verkraftbar da Neustarts relativ selten sind.

Der Autostart der graphischen Oberfläche sowie das automatische Login als Benutzer Pi kann einfach über das Konfigurationstool raspi-config[9] aktiviert werden. Hierzu muss das Tool durch Eingabe von

```
#> raspi-config
```

in der Konsole gestartet werden.

Unter dem Menüpunkt ***„Enable Boot to Desktop/Scratch“*** muss die Option ***„Desktop Login as user pi at the Graphical Desktop„*** ausgewählt werden. Danach kann das Programm geschlossen und das System neugestartet werden. Beim nächsten Boot wird automatisch die graphische Oberfläche und in weiterer Folge Chromium im Kioskmode gestartet.

Abbildung 5: Anzeige am UniScreen der Mensa

4.12 Lichtschranken

Zur Erkennung der Tore kommen jeweils 2 Lichtschranken pro Tor zum Einsatz. Es sind jeweils 2 Schranken nötig da das Tor mehr als Doppelt so hoch ist wie der Durchmesser des Balles. Die Lichtschranken sind direkt im Torrahmen angebracht um den Ball beim Übertreten der Torlinie zu erkennen selbst wenn dieser wieder aus dem Tor springt.

Abbildung 6: Einfräsungen für Lichtschranken

Um Fremdlichteinfluss zu vermeiden werden die Sendedioden mit 38kHz moduliert. Dazu wird direkt am verwendeten Arduino ein 38kHz Rechtecksignal erzeugt welches durch einen NPN Transistor pro Tor verstärkt wird und eine IR Sendediode vom Typ CQY 99 [21] ansteuert. Als Empfänger kommt eine IR Empfänger vom Typ TSSP4P38 [20] mit integriertem 38kHz Filter zum Einsatz. Der Ausgang wird bei ununterbrochenem Lichtschranken auf GND geschalten und bei unterbrochenem Lichtschranken mittels internem Pull-Up auf VCC gezogen.

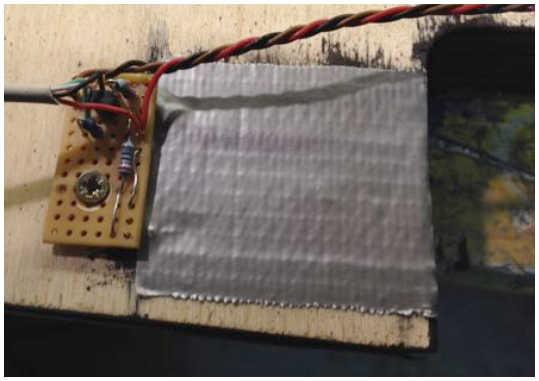

Abbildung 7: IR Sender

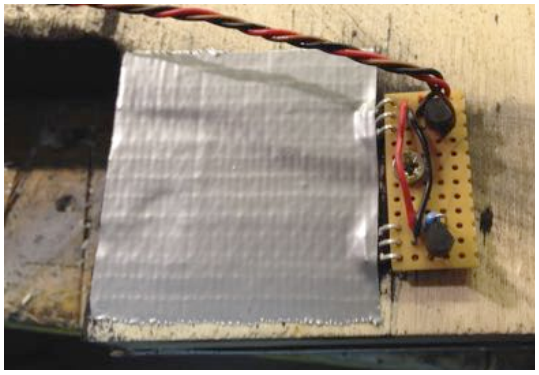

Abbildung 8: IR Empfänger

Da pro Tor 2 Lichtschranken verbaut sind jedoch nur eine unterbrochen werden muss um ein Tor anzuzeigen müssen die Ausgangssignale verknüpft werden. Hierzu schaltet jeder Lichtschranke einen NPN Transistor welche einen gemeinsamen Pull-up gegen Masse ziehen. Wird also einer der beiden Lichtschranken Unterbrochen wird der entsprechende Ausgang high, worauf der nachgeschaltete Transistor den gemeinsamen Ausgang gegen Masse zieht.

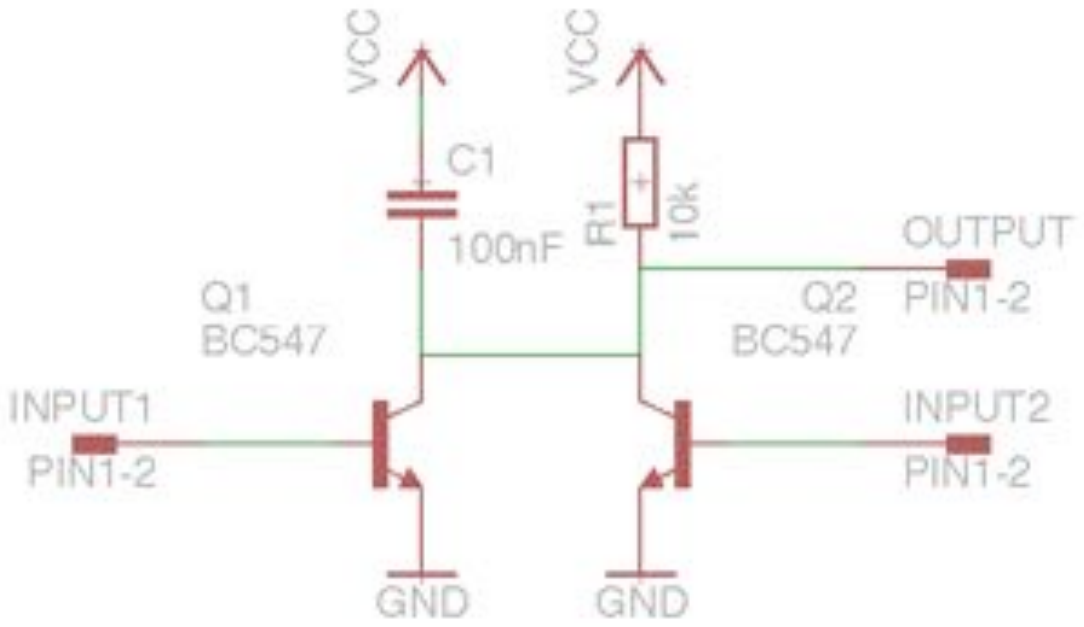

Abbildung 9: Schaltplan Verknüpfung IR Empfänger

4.13 Arduino

Für Echtzeitaufgaben und I/O Aufgaben kommt ein Arduino Mega zum Einsatz.
Folgende Aufgaben werden durch den Arduino übernommen:

- Ausgabe eines 38kHz Signals für die Modulierung der Lichtschranken
- Auswertung der beiden Buttons für Spielstart und Neustart
- Ansteuern der LEDs in diesen Buttons
- Auswertung des Ausgangssignals der Lichtschranken

Die Firmware des Arduino basiert auf der Vorlage ***„FirmataStandard“*** aus der Arduino IDE.

Das benötigte 38kHz Signal für über die tone() Funktion[6] generiert. Diese konfiguriert intern einen der vorhanden Hardwaretimer und gibt ein Rechtecksignal mit 50% Dutycycle und der übergebenen Frequenz am angegeben Pin aus.

```
void setup()
{
  ....

  Firmata.attach(SYSTEM_RESET, systemResetCallback);

  tone(11, 38000);  //insert this line to generate signal

  Firmata.begin(57600);
  ...
}
```

4.14 Cylon.js

Auf Seite von node.js übernimmt das Modul cylon.js[22] die Kommunikation mit dem Arduino. Cylon.js stellt eine Reihe von Konnektoren für verschiedene Hardware Interfaces bereit. Unter anderem auch das hier eingesetzte Arduino Modul[23] welches über das Firmata Protokoll mit dem per USB angeschlossenen Arduino kommuniziert.

Die Installation auf erfolgt wie bei allen anderen node Modulen über npm:

```
#> npm install cylon cylon-firmata --save
```

Im Gegensatz zu den vorhergehenden Modulen nimmt die Installation jedoch deutlich längen in Anspruch da Teile des Moduls cylon-firmata speziell für das Target System kompiliert werden müssen.

Bei der Installation kam es anfangs zu Problemen weil sich node-serialport nicht kompilieren ließ. Dies war auf die veraltete Version von node.js welche aus den offiziellen Paketquellen von Raspbian installiert wurde zurückzuführen. Nachdem node.js entfernt und nach Anleitung der node-serialport Github Seite neu installiert wurde war das Problem behoben.

Sobald die Installation abgeschlossen ist kann das Modul wie üblich mittels require() eingebunden werden.

5 Ausblick

Grundlegende Idee bei der Entwicklung ist die Möglichkeit die Funktionalität in zukünftigen Arbeiten erweitern zu können. Im Zuge dieser Arbeit wurden auch bereits einige Überlegungen zur Erweiterung des Systems angestellt. Dieses Kapitel versteht sich als Denkanstoß für zukünftige Arbeiten, die Ausführungen sind daher nicht notwendigerweise vollständig oder durchführbar.

5.1 Antrieb der verschiebbaren Würfel

Die Würfel welche normalerweise für das Zählen der Tore verwendet werden sind durch den Umbau eigentlich zwecklos geworden wurden jedoch nicht entfernt um eine Benutzung auch ohne den Monitor und ohne Stromversorgung zu ermöglichen.
Eine interessante Erweiterung des Projektes wäre diese weiter als Spielstandsanzeige zu verwenden und deren Bewegung zu automatisieren.

Die grundlegende Idee sieht vor in der Unterseite der Würfel jeweils einen staken Magneten zu versenken. Von unten kann dann zum einen über mehrere Hallsensoren detektiert werden wie die Würfel derzeit positioniert sind und andererseits durch Anlegen eines Magnetfeldes die Würfel in die gewünschten Positionen gebracht werden.

Zur Erzeugung des Magnetfeldes gibt es zumindest 2 Möglichkeiten.
Die erste und elegantere Lösung ist es unter den Würfeln eine Reihe von Spulen anzubringen welche entsprechend angesteuert werden. Potentiell problematisch bei dieser Lösung ist der erforderliche Strom bzw. die Anzahl der Wicklungen.

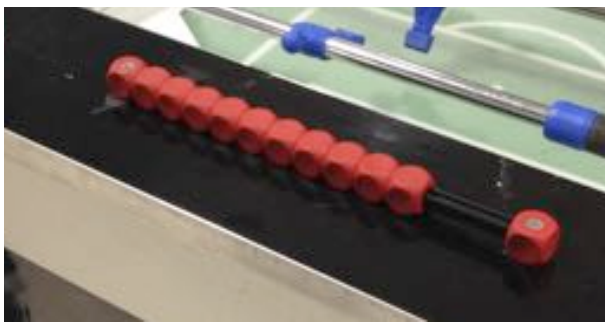

Abbildung 10: Spielstandsanzeige durch verschiebbare Würfel

Die 2. Möglichkeit ist einen Schlitten im Gehäuse des TFBT zu installieren welcher sich über die gesamt länge der Würfel bewegen kann. An diesem Schlitten kann dann entweder ein Permanentmagnet der von einem Servo in eine aktive oder passive Position gebracht wird oder ein entsprechend starker Elektromagnet befestigt werden.
Bei dieser Konstruktion ergeben sich potentiell Probleme durch die erforderliche hohe Dynamik des Schlittens.

5.2 Benutzererkennung und Bestenlisten

Eine Möglichkeit zusätzlichen Nutzen zu generieren ist die Speicherung von Spielen und Spielergebnissen sowie die Erstellung von Bestenlisten.
Hierfür müssen mehrere Teilprobleme gelöst werden die nachfolgend beschrieben werden

5.2.1 Benutzerverwaltung und Authentifizierung

Um Listen führen zu können ist es nötig die Ergebnisse der Spiele einzelnen Spielern oder Teams zuordnen zu können.
Dazu muss die Möglichkeit gegeben werden Benutzer anzulegen und sich als bestehender Nutzer zu authentifizieren. Im einfachsten Fall erfolgt dies über Username und Passwort aber auch neue Technologien wie NFC sollen evaluiert werden.

5.2.2 Entwicklung eines geeigneten Wertungssystems

Um die Leistungen der Spieler darstellen zu können muss eine Metrik gefunden werden nach der die Liste sortiert werden kann.
Bei der Erstellung der Metrik ist zu beachten dass Spieler nicht rein durch vieles Spielen in der Liste aufsteigen kann so wie dies bei der üblichen 2 und 3 Punkteregel der Fall ist.

5.2.3 Darstellung der Ergebnisse

Die Darstellung der Ergebnisse sollte sowohl auf dem HDMI Display geschehen wenn gerade kein Spiel läuft als auch über ein Webinterface von eigenen Geräten abgerufen werden können. Es wird daher nötig sein auf die verschiedenen Auflösungen der Geräte Rücksicht zu nehmen. Dies kann entweder über ein adaptives Interface (responsive Design) oder über 2 getrennte Interfaces für Smartphones und größere Geräte bewerkstelligt werden.

Zusammenfassung

Zusammenfassend ist zu sagen dass in diesem Projekt im Vorhinein kein klares Hauptproblem zu erkennen war. Wenn man einen Punkt als Hauptschwierigkeit nennen möchte so kann man die Diversität des Projektes an sich anführen. Es kommen alleine 2 Hardwareplattformen, Arduino und Raspberry Pi, zum Einsatz. Des Weiteren umfasst das Projekt einen gewissen handwerklichen Aspekt, Design und Aufbau einer simplen Schaltung aus diskreten Bauelementen, Grundlagen der Netzwerktechnik, Webdesign und Webprogrammierung.
Wenngleich jede der Aufgaben vergleichsweise simpel mit vorhandenen, breit eingesetzten und gut dokumentierten Werkzeugen und Programmen gelöst wurde so ist alleine die breite des Themas eine interessante Herausforderung da es ein sehr breites Grundlagenwissen und effiziente Informationsbeschaffung und Verarbeitung voraussetzt.

Abgesehen von den üblichen Anlaufschwierigkeiten sind bei 3 Punkten unerwartete Probleme aufgetreten.

Beim Einrichten des WLAN Accesspoints stellte sich heraus dass die in den Paketquellen enthaltene Version von hostapd die benötigten Treiber nicht enthält. Außerdem gibt es den eingesetzten USB WLAN Dongle in mehreren Versionen die zwar die gleiche Typenbezeichnung tragen jedoch nicht alle als Accesspoint verwendbar sind.

Ebenfalls als problematisch erwiesen sich die offiziellen Paketquellen bei node.js. Die Enthaltene Version ist mittlerweile veraltet so dass sich node-serialport welches für die Kommunikation mit dem Arduino benötigt wird nicht kompilieren ließ. Es musste daher das von nodejs.org direkt bereitgestellt Binary in der aktuellen Version verwendet werden.

Abbildung 11: Lockpicks

Probleme anderer Art traten beim Einbau der Lichtschranken auf. Um die notwendigen Einfräsungen herstellen zu können musste die Torwand ausgebaut werden. Dies war jedoch nicht möglich da der TFBT versperrt war jedoch kein Schlüssel vorhanden. Da eine Zerstörung des Schlosses vermieden werden sollte wurde es per Lockpicking geöffnet. Glücklicherweise handelt es sich um ein sehr einfaches Modell so dass es selbst ohne große Übung mit einem Handelsüblichen Lockpick-Set zu öffnen war.

Quellenverzeichnis

[1] Arduino.cc, abgerufen am 21.1.2015, http://arduino.cc/en/uploads/Main/ArduinoMega2560_R3_Fronte.jpg

[2] Bohnk Philip, abgerufen am 21.1.2015, http://de.wikipedia.org/wiki/Raspberry_Pi#mediaviewer/File:RaspberryPiModelBRev2.by.Philipp.Bohk.jpg

[3] Arduino Getting Started Dokumentation, abgerufen am 21.1.2015, http://arduino.cc/en/Guide/Introduction

[4] Offizeller Webauftritt des OpenWRT Projekts, abgerufen am 21.1.2015, https://openwrt.org/

[5] Offizeller Webauftritt der Raspberry Pi Fountdation, abgerufen am 21.1.2015, http://www.raspberrypi.org/help/what-is-a-raspberry-pi/

[6] Arduino API Reference, abgerufen am 21.1.2015, http://arduino.cc/en/Reference/Tone

[7] Offizeller Webauftritt der Raspbian Projekts, abgerufen am 21.1.2015, http://www.raspbian.org/

[8] Offizeller Webauftritt der Raspberry Pi Fountdation – NOOBS getting started , abgerufen am 21.1.2015, http://www.raspberrypi.org/help/noobs-setup/

[9] Offizeller Webauftritt der Raspberry Pi Fountdation – raspi-config, abgerufen am 21.1.2015, http://www.raspberrypi.org/documentation/configuration/raspi-config.md

[10] Offizielles Debian Wiki – Aptitude , abgerufen am 21.1.2015, https://wiki.debian.org/Aptitude

[11] Kernel.org Wiki – hostapd , abgerufen am 21.1.2015, http://wireless.kernel.org/en/users/Documentation/hostapd

[12] It-welt.org - Raspberry Pi und Edimax EW-7811Un: Access Point einrichten, abgerufen am 21.1.2015, http://itwelt.org/anleitungen-howto/raspberry-pi/590-raspberry-pi-und-edimax-ew-7811un-access-point-einrichten

[13] Ubuntuusers Wiki – dnsmasq , abgerufen am 21.1.2015, http://wiki.ubuntuusers.de/Dnsmasq

[14] Offizeller Webauftritt der node.js Projekts, abgerufen am 21.1.2015, http://nodejs.org/

[15] Offizeller Webauftritt des Express Projekts, abgerufen am 21.1.2015, http://expressjs.com/

[16] Offizeller Webauftritt der socket.io Projekts, abgerufen am 21.1.2015, http://socket.io/

[17] Ubuntuusers Wiki – iptables , abgerufen am 21.1.2015, http://wiki.ubuntuusers.de/iptables2

[18] Ubuntuusers Wiki – Chromium , abgerufen am 21.1.2015, http://wiki.ubuntuusers.de/Chromium

[19] Offizeller Webauftritt der lxde Projekts, abgerufen am 21.1.2015, http://lxde.org/

[20] Vishay Semiconductors – Datanblatt TSSP4P38, abgerufen am 21.1.2015 http://www.vishay.com/docs/82474/tssp4p38.pdf

[21] Reichelt.de - Datenblatt CQY 99 , abgerufen am 21.1.2015, https://cdn-reichelt.de/documents/datenblatt/A500/CQY_99.pdf

[22] Offizeller Webauftritt der cylon.js Projekts - Hauptseite, abgerufen am 21.1.2015, http://cylonjs.com/

[23] Offizeller Webauftritt der cylon.js Projekts - Arduino, abgerufen am 21.1.2015, http://cylonjs.com/documentation/platforms/arduino/

[24] GitHub – node-serialport, abgerufen am 21.1.2015, https://github.com/voodootikigod/node-serialport

Abbildungsverzeichnis

Abkürzungsverzeichnis

DHCP	**D**ynamic **H**ost **C**onfiguration **P**rotocol
DNS	**D**omain **N**ame **S**ystem
GND	**G**rou**ND** – gemeinsame Masse
HDMI	**H**igh **D**efinition **M**ultimedia **I**nterface
HTTP(S)	**H**yper**T**ext **T**ransfe**r** **P**rotokoll (**S**ecure)
IR	**I**nfra**R**ot
REST	**RE**presentational **S**tate **T**ransfer
SOC	**S**ystem **O**n **C**hip
TFBT	**T**isch**F**uß**B**all**T**isch
USB	**U**niversal **S**erial **B**us
VCC	**V**oltage at the **C**ommon **C**ollector – gemeinsame Versorgungsspannung
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
WWW	**W**orld **W**ide **W**eb