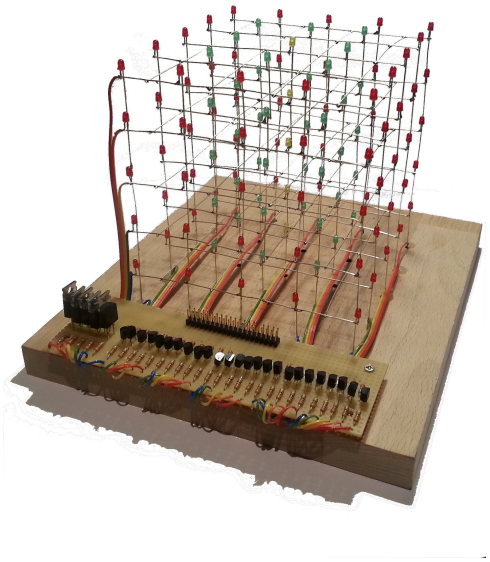
Anwendung und Programmierung von Mikrocontrollern

LED-Cube 5x5x5
mit seriellem Protokoll



3. Juli 2013

Matthias Böffel

Betreuer: Dr.-Ing. Hubert Zitt

Fachhochschule Kaiserslautern | University of Applied Sciences

Inhaltsverzeichnis

1 Einleitung

Bei einem LED-Cube handelt es sich um einen 3-dimensionalen Würfel aufgebaut aus Leuchtdioden und gefestigt durch ein Drahtgeflecht, der beliebige Muster als Standbild oder Animation anzeigt. In diesem Projekt wird der Aufbau und die Ansteuerung eines Würfels mit 5x5x5 (= 125) LEDs der Farben Rot, Grün und Gelb realisiert. Dabei sollen neue Animationsmuster über eine serielle Schnittstelle eingespielt werden können. Außerdem soll eine Testumgebung durch das Computerspiel Minecraft implementiert werden.

2 Vorbereitung

2.1 Material

1x	Holzplatte (220x200x19 mm)	30x	2,2 kΩ Kohleschichtwiderstand
1x	Lochrasterplatine (179x49 mm)	25x	22 Ω Kohleschichtwiderstand
80x	Low-Current LEDs Rot (L-934LID [1])	2x	40 pol. Steckverbinder
40x	Low-Current LEDs Grün (L-934LGD [1])		farbiges Flachbandkabel
5x	Low-Current LEDs Gelb (L934LYD [1])		Silberdraht
5x	NPN Transistor (BUT11A [2])		IDE-Kabel
25x	PNP Transistor (BC327-40 [3])		

2.2 Aufbau

In die Holzplatte werden 3 mm Löcher im Raster 5x5 mit jeweils 3 cm Abstand eingesenkt. Die Kathoden-Beine der LEDs werden auf ca. 3 mm gekürzt und die LEDs in die vorgebohrten Löcher eingesetzt (Abb. 1). Anschließend werden die Kathoden mit einem Geflecht aus Silberdraht verlötet und so die einzelnen Ebenen erstellt (Abb. 2). Die 5 fertigen Ebenen (Abb. 3) werden danach analog dazu durch die Anoden der LEDs vertikal verbunden.

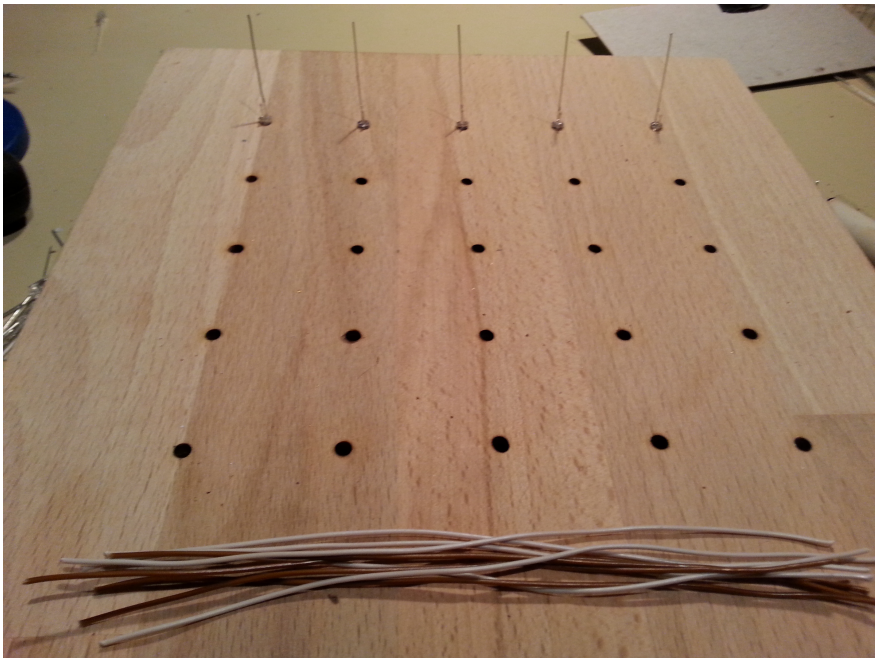


Abbildung 1: Holzplatte

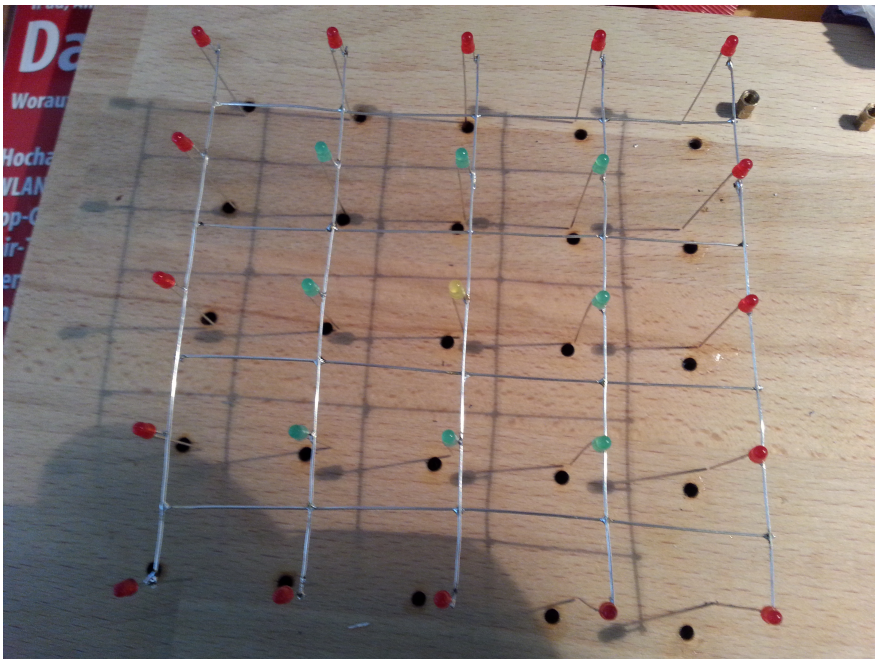


Abbildung 2: Erste Ebene

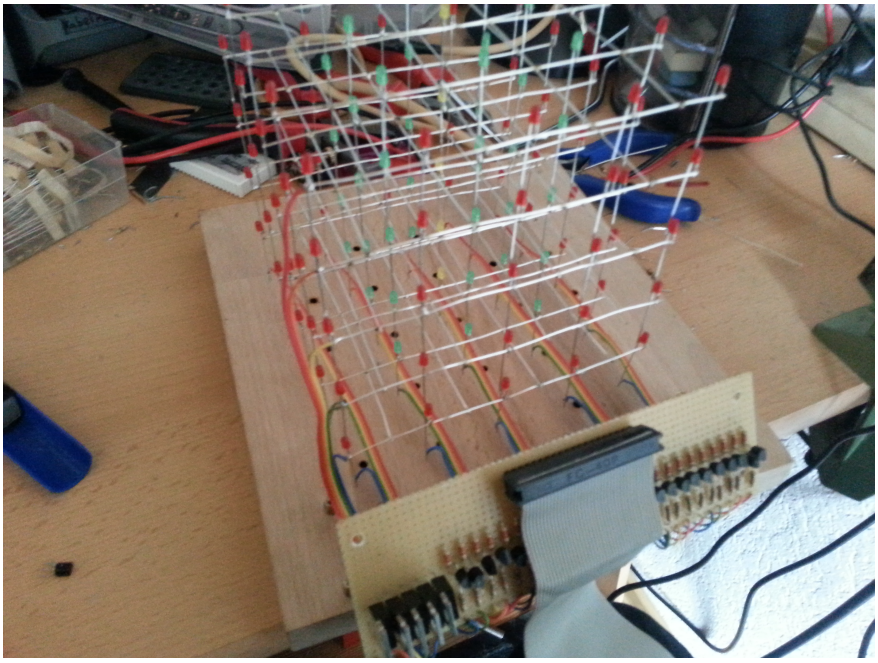


Abbildung 3: Fertiger Cube

3 Technische Realisierung

3.1 Allgemeines

Der Cube selbst besteht aus 25 LED-Säulen und 5 LED-Ebenen. Jede LED teilt sich auf ihrer Kathodenseite die Masse mit den anderen LEDs ihrer Ebene und auf Anodenseite 3,3 v mit den anderen LEDs ihrer Säule. In dieser Konstellation können keine LEDs unterschiedlicher Ebenen und Säulen gleichzeitig betrieben werden, ohne dass die jeweiligen "Schnittpunkte" mitleuchten. Daher wird ein Multiplexing-Algorithmus im Mikrocontroller implementiert, der die 5 Ebenen zeitlich versetzt schaltet. Außerdem werden die verschiedenfarbigen LEDs unterschiedlich gepulst um eine ähnliche Helligkeit zu erzielen.

3.2 Hardware

3.2.1 Mikrocontroller

Der verwendete Mikrocontroller XMC4500 [4] der Marke Infineon (Abb. 4) ist durch seine Anschlussvielfalt gut für dieses Projekt geeignet: Man kann die LED-Treiber direkt ansteuern ohne ein Bus-System (wie z.B. I^2C) zu verwenden. Controller dieses Typs wurden auf der diesjährigen Embedded World an Studenten verteilt. Die Entwicklung erfolgt über Dave3 [5], eine angepasste Eclipse-IDE von Infineon. Bibliotheken für Digital-Pin-Out und USB-zu-Serial sind hier bereits implementiert.

Abbildung 4: Infineon XMC4500 Relax Lite Kit

3.2.2 LED-Cube

Der Mikrocontroller liefert pro Digital Output Pin 5 mA, pro Pin-Gruppe 20 mA und insgesamt max. 100 mA. Jede Säule wird über einen PNP-Transistor angesteuert und hat einen Vorwiderstand der Größe 22 Ω , der aber lediglich zur Strombegrenzung dient. Ursprünglich sollten der PNPs mit 5 v Spannung betrieben werden. Daher ist folgender Rechnung zugrunde liegend den PNP-Transistoren ein 2,2 kΩ Widerstand vorgeschaltet:

$$
\begin{aligned}
I &= 2mA & U_{Abfall} &= U_{Gesamt} - U_{Verbraucher} \\
U_{Gesamt} &= 5v & &=> U_{Abfall} = 4,3v \\
U_{Verbraucher} &= 0,7v & R &= \frac{U_{Abfall}}{I} \\
& & &=> \underline{R = 2150\Omega}
\end{aligned}
$$

Da die Digital-Pins des verwendeten Controllers jedoch nicht 5 v (wie z.B. beim Arduino) sondern 3,3 v liefern und die PNPs nur bei positiver Spannungsdifferenz durchschalten, müssen die Kollektoren dieser Transistoren ebenfalls mit 3,3 v betrieben werden. Diese Änderung ist allerdings unproblematisch im Bezug auf den errechneten Widerstand, da die Transistoren auch bei der geringeren Spannung schalten. Die Masse der Ebenen wird über NPN-Leistungstransistoren gesteuert. Aus diesen Fakten ergibt sich der Schaltplan Abb. 5

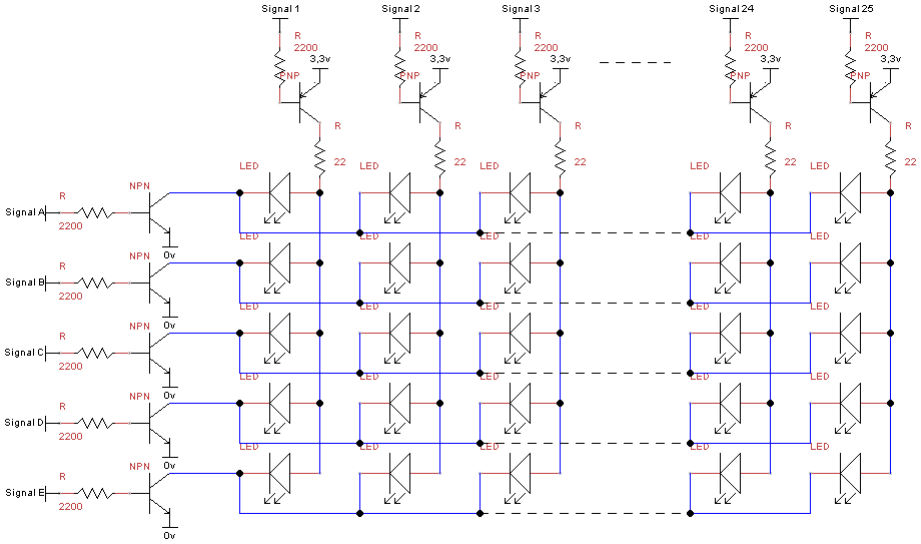


Abbildung 5: Schaltplan LedCube

3.2.3 Verbindung: Mikrocontroller - LED-Cube

Der Microcontroller verfügt über 2 Header mit jeweils 40 Pins. Die Verwendung eines bereits konfektionierten IDE-Kabels (ohne KeyPin 20) liegt nahe. Der Pin Header X2 (Abb. 6) stellt genau 30 Digital-Out Pins für die 25 Säulen und 5 Ebenen (Abb. 7) bereit.

Pin Header X2

		2	1		
		GND	GND		
	4	GND	GND	3	
Signal B	6	P5.7	P2.6	5	Signal A
Signal D	8	P5.1	P5.2	7	Signal C
Signal 1	10	P1.15	P5.0	**9**	Signal E
Signal 3	12	P1.13	P1.14	11	Signal 2
Signal 5	14	P1.11	P1.12	13	Signal 4
.	16	P1.5	P1.10	15	Signal 6
.	18	P1.3	P1.4	17	.
.	20	P1.1	P1.2	**19**	.
	22	P1.9	P1.0	21	.
	24	P0.8	P1.8	23	
	26	P3.4	P0.7	25	
	28	P0.12	P3.3	27	
	30	P0.6	P0.11	**29**	
	32	P0.2	P0.5	31	
Signal 25	34	P0.4	P0.3	33	Signal 24
	36	GND	GND	35	
	38	VDD3.3	VDD3.3	37	
	40	VDD5	VDD5	39	

Abbildung 6: Pinbelegung X2 Header

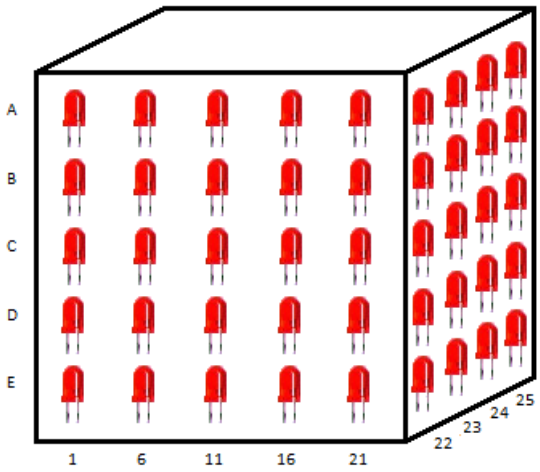


Abbildung 7: Skizze Pin-Mapping

3.3 Protokoll

3.3.1 Allgemein

Das Protokoll ist einfach in der Implementierung und für verschiedene Würfelgrößen ausgelegt. Über die virtuelle, serielle Schnittstelle können Bytes und Byte-Arrays empfangen werden. Die Auswertung hängt vom ersten Byte ab, welches als ASCII-Zeichen interpretiert wird:

a	Animationsspeicher zufällig befüllen und starten
s	Preset aus folgendem Byte-Array übernehmen und Animation starten
r	Leds ausschalten, Animation stoppen und Zähler rücksetzen
+	Delay-Wert dynamisch um +1000 ändern (bis max. 200000)
-	Delay-Wert dynamisch um -1000 ändern (bis min. 1000)

3.3.2 Der Start-Befehl 's'

Der Start-Befehl *s* wird in einem Byte-Array als erstes Byte gesendet. Danach folgen die Würfel-Kantenlänge und die Anzahl der wiederzugebenden Frames jeweils als Byte. Interpretiert werden die Werte als *unsigned int*. Aus diesen Werten werden die verbleibenden Preset-Daten berechnet, die die restlichen Bytes im Array bilden. Die Daten werden anschließend in den Animationsspeicher kopiert. Ein Beispiel in Abb. 8

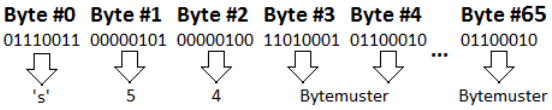


Abbildung 8: Beispiel für ein 's' Byte-Array

3.4 Programmierung - Mikrocontroller

3.4.1 Allgemein

Nach dem Anlegen des neuen Projektes in *Dave* müssen die verwendeten Objekte zugefügt werden. Zum Importieren wählt man die entsprechenden Apps aus einer Liste per Doppelklick (Abb. 9). Zweckmäßig sind hier *IO004* für Digital-Pins und *USBVC001* für den virtuellen COM-Port. Anschließend müssen den Ports die zugehörigen Pins zugewiesen werden. Ein Listing der einzelnen Bibliotheken würde an dieser Stelle den Rahmen sprengen. Die eigentliche Anpassung findet danach ohnehin nur innerhalb der *main.c* statt.

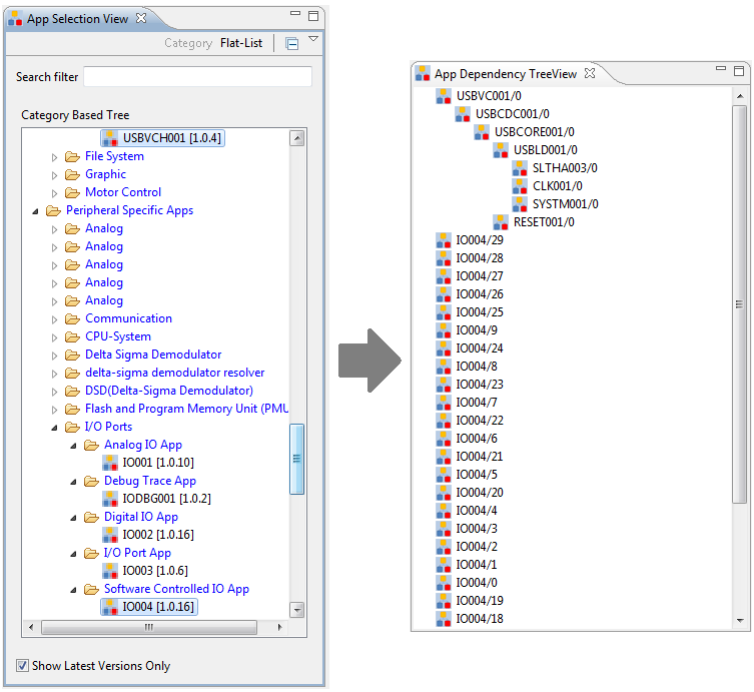


Abbildung 9: Apps hinzufügen

3.4.2 Aufbau main.c

a) Einbinden der benötigten Header

b) Konstanten definieren

c) Deklaration der globalen Variablen

d) Definition der Funktionen

e) Funktion *initHandles*: Pins für Ebenen und Säulen in passender Reihenfolge in jeweilige Arrays zuweisen

f) Funktion *setCubeLed*: Nimmt eine Ganzzahl *send* zwischen 0 und 125 und einen Boolean Wert *state* als Parameter entgegen und schaltet je nach *state* die entsprechende Ebene und Säule an oder aus

g) Funktion *setAllLedsOff*: Durchläuft in 2 Schleifen alle Ebenen und Säulen und setzt sie in den entsprechenden Aus-Zustand

h) Funktion *reset*: Schaltet mittels *setAllLedsOff* alle LEDs aus und setzt die boolesche Running-Variable und den LED-Counter auf 0

i) Funktion *delay*: Durchläuft eine Schleife so oft wie im Parameter angegeben

j) Funktion *main*: Einstiegspunkt für den Mikrocontroller und Hauptroutine

3.4.3 Vorgehensweise in der Hauptroutine

Nachdem die IO-Pins und der USB Virtual COM-Port initialisiert wurden begibt sich der Controller durch die *while (1)*-Schleife in den Loop-Modus. Im Loop-Modus wird als erstes der USB-Anschluss abgefragt und auf eventuelle Incoming-Bytes geprüft: Sind solche vorhanden oder ist der Controller frisch gestartet so wird die Empfangsroutine durchlaufen. Hier wird geprüft ob einer der Steuerbefehle empfangen wurde und jeweilige Aktionen entsprechend der Protokoll-Tabelle ausgeführt. Wenn der Controller erst gestartet wurde wird der *a*-Befehl durchlaufen, bei dem der 256 Byte-Felder große Empfangsbuffer zufällig befüllt wird um eine Zufallsanimation zu starten. Nachdem die Empfangsroutine durchlaufen wurde wird geprüft ob es zu verarbeitende Bytes gibt (i.d.R. nach *a* und *s*), die dann in ein Bool-Array (Animationsspeicher) geshiftet werden. Nachdem der Animationsspeicher befüllt wurde, wird die Running-Variable true gesetzt, die den folgenden Abschnitt aktiviert. In diesem Abschnitt gibt es erstmal 2 Möglichkeiten: Der Delay-Counter (der in jedem Loop-Durchlauf um +1 erhöht wird) hat seinen gesetzten Maximal-Wert erreicht, was bedeutet, dass das nächste "Bild" in den temporären Bildspeicher geladen wird und der Delay-Counter wieder zurückgesetzt wird

ODER dass der Delay-Counter seinen Maximal-Wert noch nicht erreicht hat und eine Zeile des aktuellen Bilds angezeigt wird. Im nächsten Durchgang wird die nächste Reihe angezeigt usw. Außerdem werden immer nur LEDs einer Farbe gleichzeitig angeschaltet. Jede Farbe hat einen eigenen Delay-Wert, der sich durch mehrere Tests als sinnvoll erwiesen hat, so dass alle Farben relativ gleich hell leuchten.

4 Testumgebung Minecraft

4.1 Allgemeines

Minecraft [6] wurde 2011 vom schwedischen Entwickler Markus Persson veröffentlicht. Es handelt sich dabei um ein Open-World-Spiel (geschrieben in Java [7]). Die 3D-Welt (Abb. 10) in der man sich frei bewegen kann, besteht aus einer endlich großen Menge von quadratischen Blöcken, die jeweils aus programmiertechnischer Sicht einem Typ zugewiesen sind und dementsprechend eine eigene Textur und eigenes Verhalten haben. Der Spieler kann diese Blöcke "abbauen" und daraus Werkzeuge und andere Elemente herstellen, die dann wiederum neue Blöcke (wie z.B. Glas aus Sand) erschaffen können. So sind dem Spieler bei der Erschaffung von Gebäuden, Monumenten oder sonstigen Bauwerken kaum Grenzen gesetzt.

Abbildung 10: Burglandschaft in Minecraft

4.2 Von Minecraft zum Würfel

In Minecraft gibt es unter den vielen Elementen auch Lampen und Schalter, die sich dazu eignen eine Vorlage für die LEDs zu bilden. Im Bündel zu einem großen Würfel zusammengefasst können diese Lampen ein genaues Abbild des LED-Cubes in 3-dimensionaler

Umgebung darstellen (Abb. 11). Auch kann man durch die Positionierung mehrerer Würfel hintereinander eine Abfolge für eine Animation generieren.

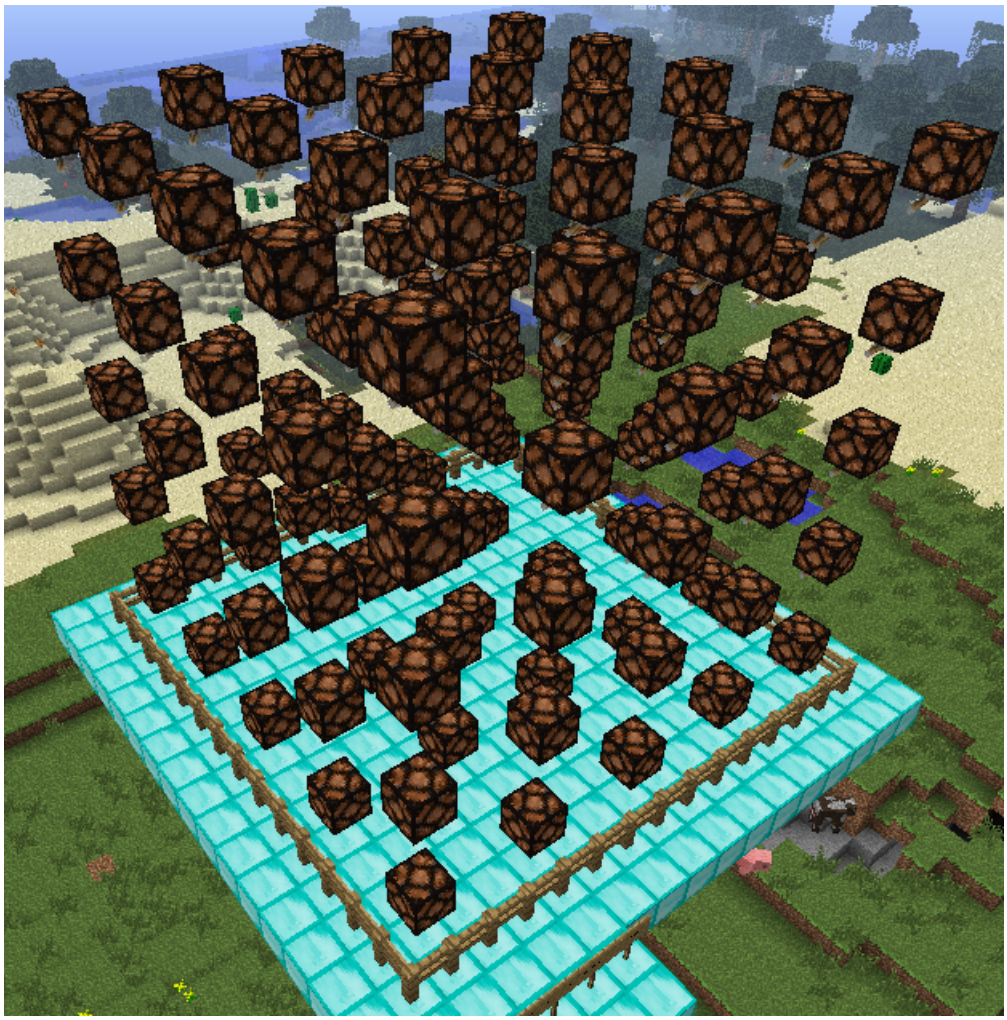


Abbildung 11: LED-Cube "Abbild"

4.3 LED-Cube Plugin

Es gibt diverse offene Serverprojekte für Minecraft die eine Plugin-Schnittstelle bereitstellen. Die entsprechenden Plugins werden ebenfalls in Java programmiert und geben dem Entwickler vielfältige Möglichkeiten in das Spielgeschehen (insbesondere die 3D-Welt) einzugreifen.

So können per Plugin auf Befehl Abbilder des LED-Cubes generiert werden. Diese Abbilder bestehen aus besagten Lampen, die man als Spieler ein- und ausschalten und so ein Animationsmuster erzeugen kann. Zusätzlich kann der Spieler zwischen diversen

Optionen wählen. Das Plugin wird beim Start des Minecraft-Servers mitgeladen und bedient sich der RxTxComm Bibliothek [8] zur Kommunikation mit dem COM-Port. Näheres dazu im Kapitel *Inbetriebnahme*.

5 Inbetriebnahme

5.1 LED-Cube

Der LED-Cube wird über ein 40-pol. IDE-Kabel ohne Keypin mit dem Microcontroller verbunden. Pin-1 ist auf beiden Seiten gekennzeichnet. Anschließend wird der Microcontroller über ein Micro-USB-Kabel mit dem PC verbunden. Der Microcontroller wird als virtuelles COM-Port Gerät erkannt. Gegebenenfalls muss noch ein entsprechender Standard-Treiber installiert werden.

5.2 Minecraft-Server

Der Minecraft-Server stellt im Multiplayer-Modus die generierte Umgebung und gemeinsame Schnittstelle zu Spieler-Transaktionen bereit. Zwar kann Minecraft auch ohne Server lokal gestartet werden, allerdings können dann keine Plugins implementiert werden.

Der Server besteht initial nur aus einer Jar-Datei [9] und wird über die Kommandozeile des jeweiligen Betriebssystems mittels folgendem Befehl gestartet:

java.exe -Xms1024M -Xmx1024M craftbukkit.jar

Der XMS- und XMX-Parameter dienen zur Anpassung des zur Verfügung gestellten Heap-Speichers. Pfade müssen evtl. angepasst werden. Nach dem ersten Start wird eine Ordnerstruktur für den Server angelegt, unter anderem auch ein *Plugin*-Ordner, in den anschließend das LED-Cube Plugin kopiert werden muss. Durch den Konsolenbefehl *reload* wird der Server gezwungen die Plugins neu zu laden. Danach ist er bereit.

5.3 Minecraft-Client

Der Client ist das Spiel selbst. Das Installations-File erhält man unkompliziert über die Website des Vertreibers [6]. Für den Start ist eine Lizenz erforderlich, die man zuvor erwerben muss. Nach dem Start kann man im Untermenü *Multiplayer* eine IP-Verbindung zum Server anlegen und darauf verbinden.

Im Spiel selbst hat man durch das Plugin *LED-Cube* mehrere Befehle zur Auswahl, deren Eingabe man per Chat-Kommando 't' startet:

$\backslash comconnect\ COM_X$	Verbindet auf den angegebenen COM-Port X Voraussetzung zum Senden von Daten an den LED-Cube

\sqlconnect	Verbindet auf die hard-codierte Datenbank Voraussetzung zum Speichern und Abrufen von Presets
\createcube X Y	Erzeugt in der 3D-Welt ein Abbild des Würfels; *X* ist durch die Kantenlänge und *Y* durch die Anzahl Frames definiert; Um für den LED-Cube eine Animation aus 3 Bildern zu erzeugen würde der Befehl also *\createcube 5 3* lauten

Nach dem Erzeugen eines Abbildes kann man nun die Lampen durch den Schalter darunter nach Belieben an- und ausschalten. Zusätzlich werden folgende Schilder generiert (Abb. 12) um Befehle entgegenzunehmen:

Delete	Löscht das aktuelle Abbild wieder aus der Welt
Start/Stop	Überträgt die Animation zum LED-Cube und startet die Animation
Save Preset	Speichert das aktuelle Preset in der Datenbank
Load Preset	Lädt ein ausgewähltes Preset aus der Datenbank Rechtklick zum durchsuchen der Liste und Linksklick zum bestätigen
Delete Preset	Löscht das ausgewählte Preset aus der Datenbank
Delay	Linksklick zum verringern und Rechtklick zum erhöhen des Delays zwischen den Frames - Kann zur Laufzeit verändert werden

Abbildung 12: Optionen-Schilder

6 Zusammenfassung

Ausgehend von der Idee einen LED-Cube aus einer 3-dimensionalen Testumgebung heraus anzusteuern war das Ziel der vorliegenden Arbeit eine funktionelle Umsetzung dieser Idee. Ein sauberer Aufbau der gewählten Materialien unterstützt die Übersichtlichkeit der Verdrahtung und vermittelt die Qualität, in der auch der Rest des Projektes umgesetzt wurde. Weiter haben sich die Überlegungen zur Implementierung des Microcontroller-programms hinsichtlich des Protokolls und Multiplexings als zweckmäßig und sinnvoll herausgestellt.

Anders als erwartet können die Low-Current-LEDs innerhalb der gegebenen Rahmenbedingungen nicht ihre volle Helligkeit erzielen und der Aufwand für das erstellte Plugin liegt unverhältnismäßig höher als die eigentlichen Microcontroller Programmier-Arbeiten.

Weitere Motivationen könnten sein, das Projekt mit Hilfe des I^2C Datenbus-Systems umzusetzen und den Cube mit RGB-LEDs zu betreiben.

Das Projektziel wurde daher unter wenigen Einschränkungen angemessen erreicht. [10]

Literatur

[1] *Kingbright Low Current LEDs*
http://www.produktinfo.conrad.com/datenblaetter/125000-149999/145998-da-01-en-LED_3MM_ROT_LOW_CURRENT.pdf

[2] *NPN-Transistor BUT11A*
http://www.datasheetcatalog.org/datasheet/stmicroelectronics/5391.pdf

[3] *PNP-Transistor BC327-40*
http://www.datasheetcatalog.org/datasheet/vishay/85132.pdf

[4] *XMC4500 Relax Lite Kit*
http://www.mouser.com/pdfdocs/Infineon_Users_Manual_XMC4500_Relax_KitV1_R11.pdf

[5] *Dave3 IDE*
http://www.infineon.com/cms/en/product/channel.html?channel=db3a30433580b37101359f8ee6963814

[6] *Minecraft*
http://minecraft.net

[7] *Java*
http://www.java.com

[8] *RxTxComm*
http://rxtx.qbang.org

[9] *Craftbukkit Minecraft Server*
http://dl.bukkit.org/downloads/craftbukkit

[10] *Anschauliches Video vom fertigen Projekt*
http://www.youtube.com/watch?v=mCPaRV17-mY