Dokumentation

KI – Projekt

„Magic Cube Hacker“

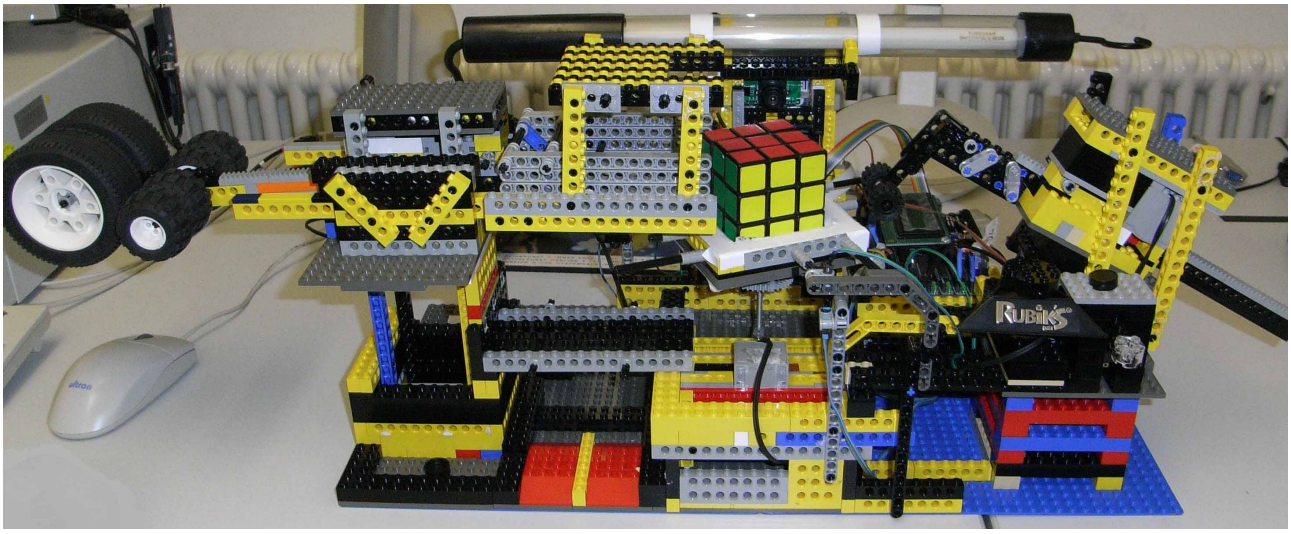

Abbildung 1: Ansicht Front. Links: Festhalter, Mitte: Plattform & Kamera, Rechts: Kipper

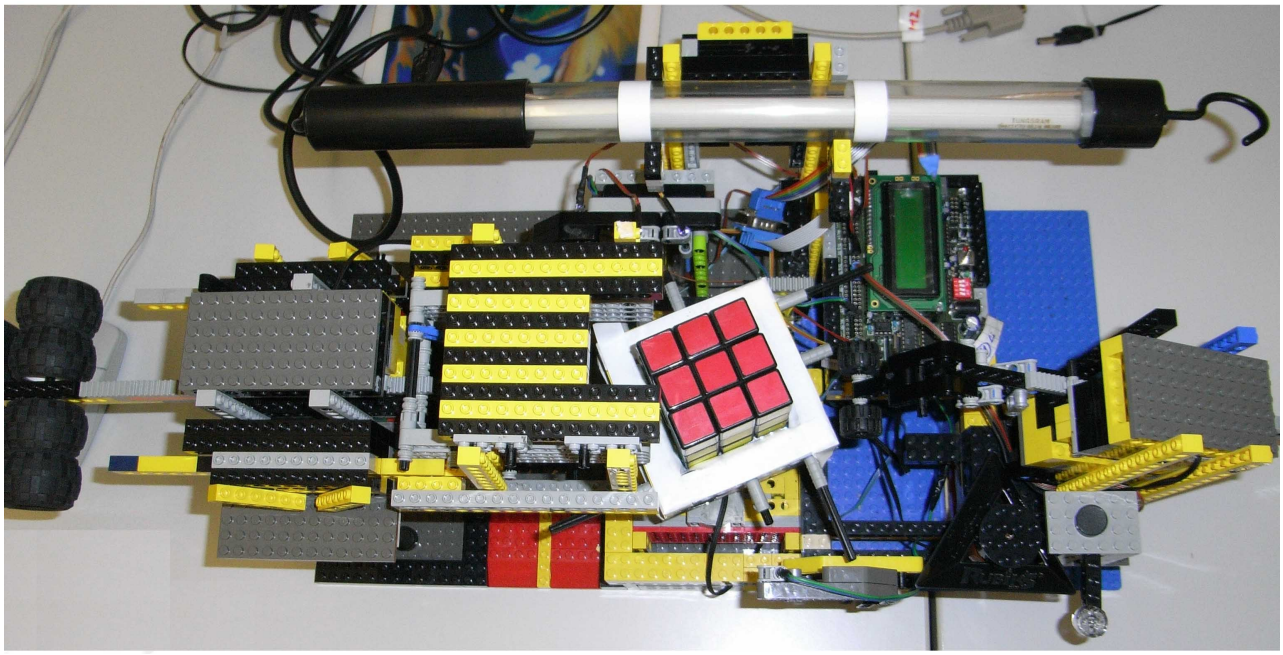

Abbildung 2: Ansicht von oben. Die Leuchtstoffröhre sorgt für ausreichend gute Beleuchtung für den Einlesevorgang.

Inhaltsverzeichnis

1 .Aufgabenstellung.....Seite 4
2 .Idee „Zauberwürfel“.....Seite 4
2.1 Gedanken zur Realisierung der Idee.....Seite 4
3 .Umsetzung der Idee
3.1 Einlesen des Würfels
3.1.1 festen Standort realisieren.....Seite 5
3.1.2 Bildverarbeitung.....Seite 6
3.1.2.1 Foto - Abschnitte bestimmen.....Seite 7
3.1.2.2 Farbe bestimmen.....Seite 8
3.1.3 Kommunikation zwischen AKSEN und CMUCam 3……Seite 8
3.2 Mechanik
3.2.1 Drehung des Würfels.....Seite 9
3.2.1.1 Würfel drehen.....Seite 10
3.2.1.2 untere Ebene drehen.....Seite 10
3.2.2 Festhalten des Würfels (Festhalter).....Seite 11
3.2.3 Kippen des Würfels (Kipper).....Seite 12
3.3 Lösungsalgorithmus
3.3.1 Idee zur Abarbeitung.....Seite 13
3.3.1.1 Phasenbildung.....Seite 13
3.3.2 Umsetzung
3.3.2.1 Internes Modell des Würfels.....Seite 14
3.3.2.2 Methoden zur Ansteuerung der Mechanik.....Seite 14
3.3.2.3 Erstellung eines Abarbeitungsstrings
3.3.2.3.1 Syntax des Programm „Cube-Explorer“……Seite 15
3.3.2.3.2 Umwandlung in Maschinenbefehle.....Seite 15
3.3.2.3.3 Abarbeitung der Befehle.....Seite 16
3.3.3 Optimierung
3.3.3.1 Kürzung der Abarbeitungsschritte (Maschinenbef.)....Seite 16
3.3.3.2 Funktionen um jeweilige Seitendrehungen durchzuführen.
….....Seite 16
3.3.3.3 Kürzungsregeln.....Seite 17
4 .Bedienung.....Seite 18
5 .Resumé / Fazit.....Seite 18

1. Aufgabenstellung

Im Wintersemester 09/10 ist es unsere Aufgabe einen Roboter mit Hilfe des AKSEN – Boards zum Thema „Entertainment-Robots“ zu entwickeln.
Es gab diesmal keine Eingrenzung des Themas, somit war der Kreativität keine Grenzen gesetzt.

2. Idee „Zauberwürfel“

Auf der Suche nach einem passenden Projekt, haben wir ein Video auf Youtube entdeckt, welches sich mit dem Lösen des Zauberwürfels beschäftigt. Nach ein paar Beratungen haben wir uns entschieden einen Roboter zu erfinden, der selbstständig einen verdrehten Würfel löst.

2.1 Gedanken zur Realisierung der Idee

Zur selbständigen Lösung des Zauberwürfels mussten wir uns Gedanken machen, welche Technik wir verwenden. Wir entschieden uns zum einlesen des Würfels die CMUCam 3 zu verwenden. Aufgrund der Komplexität des gesamten Projekts realisieren wir nur die Drehung einer Seite (unteren).
Diese Entscheidung hatte weitere Probleme zur Folge, die wir mit Hilfe eines Kippers und Festhalters versuchen zu lösen. Der Kipper wird zum Wechseln von Würfelseiten verwendet. Der Festhalter dient zum greifen der oberen und mittleren Ebene, um die Drehung der unteren Ebene zu realisieren.

3. Umsetzung der Idee

3.1 Einlesen des Würfels

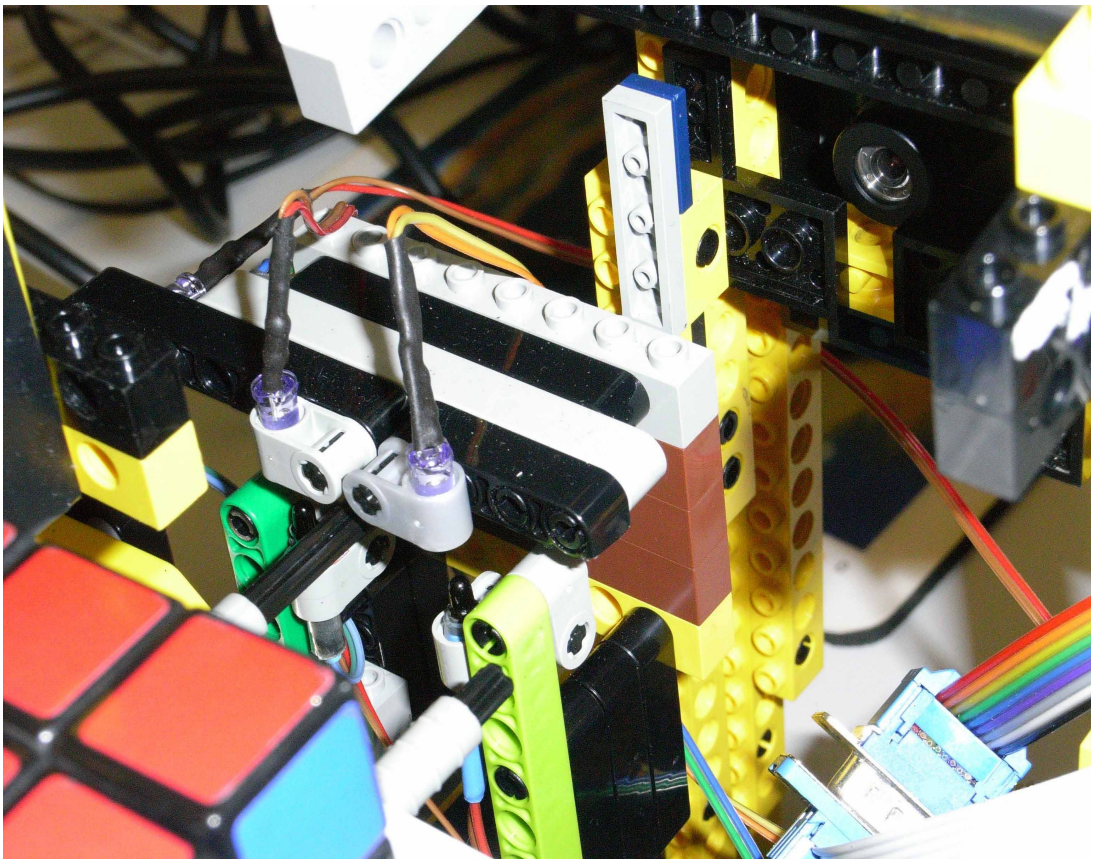

Abbildung 3: Kamera und Lichtschranke für Positionierung der Plattform

3.1.1 festen Standort zum Einleseverfahren

Um einen reproduzierbaren Standort zum Einlesen einer Seite zu realisieren, haben wir an der drehbaren Plattform an jeder Seite eine Legostange angebracht, die beim Drehvorgang durch eine Lichtschranke kommt. Bei Auslösung dieser Lichtschranke stoppen wir den Drehvorgang und geben der Kamera einen Befehl zur Erstellung eines Bildes bzw. Auswertung der momentanen Würfel-Seite. Nach erfolgreicher Auswertung gibt die Kamera ein Feedback über die serielle Schnittstelle. Nach jedem Feedback dreht die Plattform weiter zur nächsten Würfelseite bzw. der Würfel wird gekippt, um auch die obere und untere Seite einlesen zu können.

3.1.2 Bildverarbeitung

Die Bildverarbeitung war mit der schwierigste Teil des Projekts. Es gab viele Störfaktoren, die sich erst nach sehr viel Zeitaufwand und Testbildern auf ein Minimum reduzieren lassen konnten. Eines der komplexesten Faktoren war die ständig wechselnde Belichtung, einmal von Außen und zum anderen die interne automatische Belichtung der Kamera. Mit viel Mühe und Geduld ist es gelungen, die CMUCam so einzustellen, dass ein reproduzierbares Bild entstanden ist. Um dieses Bild zu erstellen, musste man mit den internen Registern arbeiten, welche im folgenden näherer beschrieben werden:

Automatic Gain Control (Reg0) auf 0 gesetzt.(Auomatische Verstärkungsregel)
Blaukanal-Verstärkung (Reg 1) auf 255 gesetzt.
Rotkanal-Verstärkung (Reg 2) auf 255 gesetzt.
Farbsättigung auf 0 gesetzt.
Helligkeit per CC3-Befehl auf 100 gesetzt. (Reg 6)
analoge Schärfen-Einstellung auf 0 gesetzt. (Reg 7)
Internes Register 10 auf 0 gesetzt.
Weißabgleich (Hintergrund) auf 0 gesetzt. [Reg 12] (Blau-Kanal)
Weißabgleich (Hintergrund) auf 0 gesetzt. [Reg 13] (Rot-Kanal)
Automatische Belichtungssteuerung auf 0 gesetzt (Reg 16)

Erst nach der Erstellung eines reproduzierbaren Bildes war es uns überhaupt möglich Farben des Würfels exakt zu bestimmen. Leider mussten wir auf externes Licht zurückgreifen, um bei jeder Licht-Situation den Würfel einlesen zu können. Wir benötigen dazu ein Licht, welches das komplette Spektrum ausstrahlt. Wir haben die Suche nach einer solchen Lichtquelle unterschätzt und stießen immer auf Überschneidungen im RGB-Raum. Eine Leuchtstoffröhre, welche als „Rein Weiß“ gekennzeichnet ist, war unsere Rettung. Das Licht und die Einstellungen gaben uns nun die Möglichkeit die Farben mit Hilfe der CMUCam 3 zu erkennen.

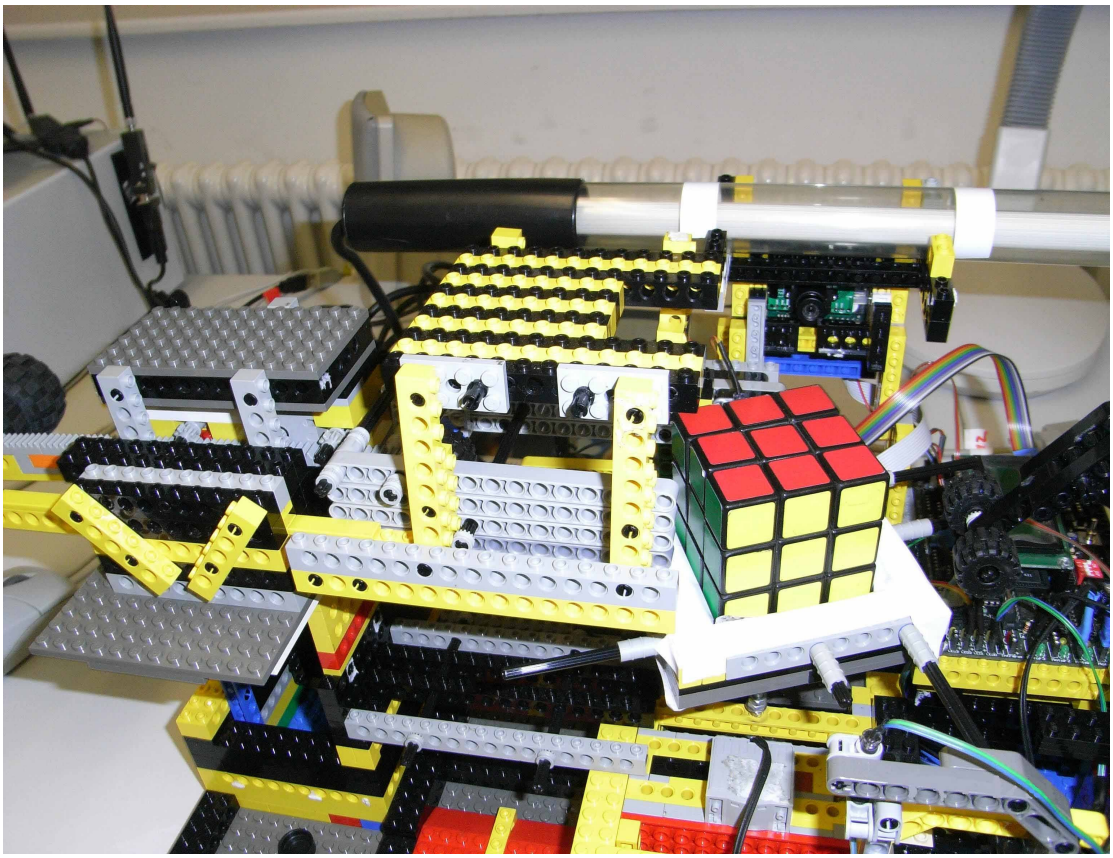

Abbildung 4: Kamera nimmt Fotos des Würfels mit leicht schrägem Winkel auf

3.1.2.1 Foto - Abschnitte bestimmen

Der feste Standort und die Kriterien zum Erkennen der Farben sind erfüllt und es muss nur noch programmtechnisch umgesetzt werden. Ein Zauberwürfel (Original) besteht aus 6 Seiten und jeweils 9 kleinen Feldern/Quadrate. Um eine Farbe exakt zu bestimmen, nehmen wir nur einen kleinen Abschnitt aus dem jeweiligen Quadrat. (30x20 Pixel) Die genauen Positionierungen der Quadrate haben wir durch Nutzung des Programm Adoculus bestimmt. Insgesamt erstellen wir für jedes Quadrat also ein Bild, aber nur noch vom jeweiligen Abschnitt, somit ist es uns nun möglich die Farbe des jeweiligen Quadrats zu bestimmen.

3.1.2.2 Farbe eines Quadrats bestimmen

Um eine Farbe zu bestimmen, benötigen wir Farbwerte der jeweiligen Würfelaufkleber. Diese konnten durch Testbilder erfasst werden. Aufgrund von Reflektierungen und Lichtschwankungen haben wir für jede Farbe einen Bereich festgelegt. Diesen Bereich haben wir mit Hilfe des Programms Adoculus und des kleinen Hilfs-Programms Prozessing erstellt. Mit Hilfe von Adoculus haben wir die Farbwerte der jeweiligen Farbe im RGB-Raum bestimmt. Prozessing bietete uns die Möglichkeit einen RGB-Cube zu erstellen, womit wir eine grafische Sicht auf die jeweiligen Farben besaßen. Durch Ziehen der Farbquader haben wir das Problem der unterschiedlichen Reflektion und der verschiedenen Lichtverhältnisse gelöst. Nach Erstellung des Programms für die CMUCam 3 haben die fehlenden Farbwerte der jeweiligen Farbe hinzugefügt und dann jeden Pixel des fotografierten Ausschnitts verglichen. Nach Erreichung von 300 Pixel einer Farbe, speichern wir diese im Array der jeweiligen Seite zwischen. Um das Array zu bestimmen untersuchen wir immer zuerst das mittlere Quadrat. Die Reihenfolge der Seiten wird auch zwischengespeichert, um bei Übertragung des Würfels zum AKSEN–Board die Ausgangslage des Würfels zu gewährleisten. Diese sind in Front, Left, Back, Right, Up und Down gegliedert.

3.1.3 Kommunikation zwischen AKSEN und CMUCam 3

Die Kommunikation zwischen beiden Geräten wird über eine serielle Verbindung bzw. beider seriellen Schnittstellen realisiert. Das AKSEN – Board schickt nach Drehung des Würfels (siehe fester Standort zum Einleseverfahren) den Befehl „P“ für Photo, dieser gibt der Kamera den Befehl, ein Foto der momentanen Würfelseite zu erstellen und diese auszuwerten. Nach der Auswertung schickt die Kamera den Befehl „K“ für OK zurück und der Einleseprozess wird fortgesetzt.
Nach vollständigen Einleseverfahren (alle Seiten eingelesen) schickt das AKSEN – Board den Befehl „G“ für Get zur Kamera. Der Befehl Get löst im Programm der Kamera eine Funktion an, welche zuerst überprüft, ob 9 Farbquadrate von jeder Farbe auf dem Würfel erkennbar waren. Wenn ja schickt er die ausgewerteten/erkannten Farben zum Aksenboard. (in der gleichen Reihenfolge, wie der Würfel eingelesen wurde) Wenn dies nicht der

Fall ist, wird ein „F“ an das AKSEN – Board gesendet und der Einlesevorgang beginnt von neuem.

3.2 Mechanik

3.2.1 Drehung des Würfels

Die Drehung des Würfels ist ein weiterer, komplexer Faktor, der für die selbständige Lösung des Zauberwürfels wichtig ist. Aus diesem Grund haben wir insgesamt 3 Lichtschranken eingebaut, um die Mechanik stabil zu machen. Wir verwenden drei Lichtschranken, weil bei der Drehung der unteren Ebene länger gedreht werden muss und dieses so genau wie möglich, um eine Verkantung des Würfels zu verhindern. Zwei Lichtschranken sind für Links- bzw. Rechts-Drehungen der unteren Ebene verantwortlich und die Dritte ist für die gerade Position, zum Kippen oder Festhalten des Würfels, zuständig. Weiterhin haben wir eine Metallachse benutzt um die Kräfte auf die Plattform bzw. auf den Würfel zu übertragen. Ein normaler Lego-Motor reicht aus um diese Drehungen auszuführen. Damit wir genug Kraft aufbringen haben wir ein Getriebe entwickelt, welches genau für diese Funktion ausgerichtet ist.

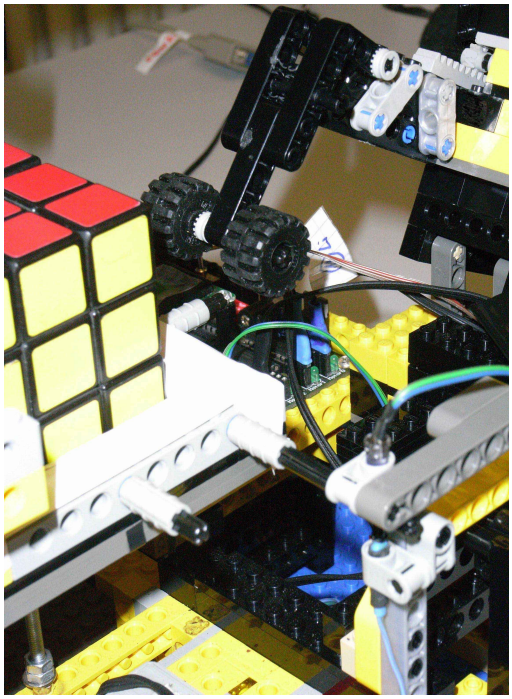

Abbildung 5: Lichtschranke vorne rechts

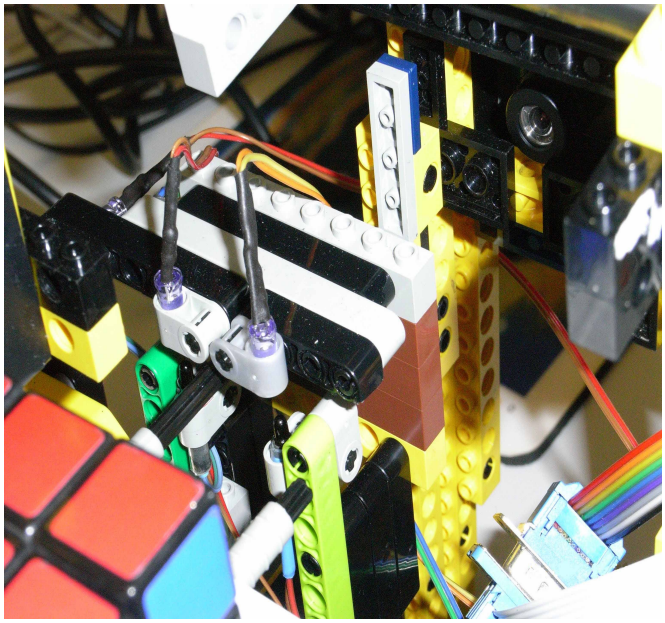

Abbildung 6: Lichtschranke hintere Seite

3.2.1.1 Würfel drehen

Die Drehung des Würfels ist in beiden Richtungen möglich. Die normale Drehung des gesammten Würfels wird, wie oben beschrieben, durch eine Lichtschranke realisiert. Diese Lichtschranke wird von einen der vier angebrachten Legostangen unterbrochen und die Drehung wird gestoppt.
Die etwas wacklige Mechanik ist dank der verwendeten Sensorik so genau, das es fast zu keinem manuellen Eingriff kommen muss.

3.2.1.2 untere Ebene drehen

Die untere Ebene des Würfels ist bei unserem Projekt, die einzigste Seite, die uns erlaubt Änderungen am Würfel vorzunehmen. Aus diesem Grund haben wir, wie auch bei der normale Drehung des Würfels, Lichtschranken verwendet. Diesmal verwenden wir aber zwei Lichtschranken, um beide Änderungen (links, rechts) zu gewährleisten. Die Positionierung der Lichtschranken wurde so vorgenommen, dass die Drehung endet, wenn die untere Ebene exakt einmal um 90° gedreht wurde. Die Kanten sind somit untereinander und gewährleisten, dass es zu keiner Verkantung kommen kann.

3.2.2 Festhalten des Würfels (Festhalter)

Das Festhalten des Würfels ist eine weitere wichtige Funktion unseres Projekts. Wir haben versucht die Spanne zwischen Würfel und Festhaltestangen so gering wie möglich zu halten, um eine flüssige Drehung der unteren Ebene zu gewährleisten. Diesen von uns genannten „Festhalter“ können wir vor und zurück fahren, was wir mit einem Lego-Motor realisiert haben. Im Laufe des Projekts gibt es eine weitere Aufgabe, die der „Festhalter“ für uns übernimmt, nämlich das „reinschubsen“ des Würfels in die Plattform, weil es durch die gerade Positionierung der Plattform öfters zu Problemen bei dem Kippen des Würfels kommen kann. Dieses Problem ist durch die Doppelfunktion nun behoben.

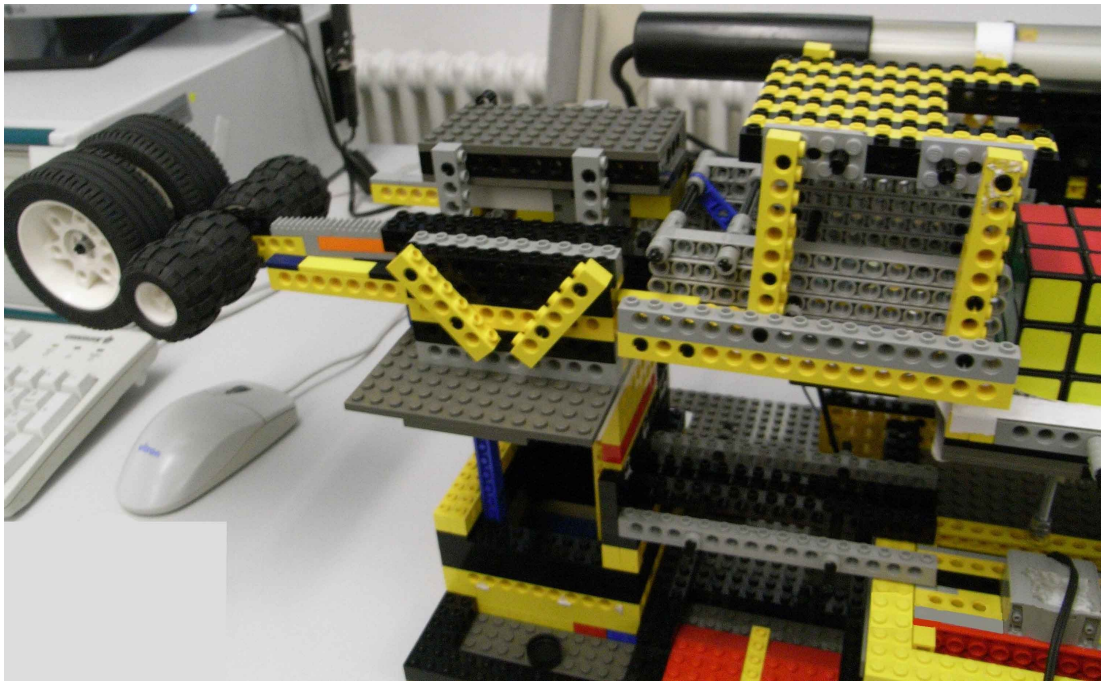

Abbildung 7: Festhalter mit Gegengewicht

3.2.3 Kippen des Würfels (Kipper)

Die letzte mechanische Funktion, die realisiert werden muss, ist das Kippen des Würfels. Dazu haben wir eine Vorrichtung gebaut, die den Würfel schräg oben berührt und Ihn einen „Schubs“ verpasst und somit der Würfel gekippt. Dieser sogenannte „Schubs“ muss schnell erfolgen, um die Kraftübertragung zu gewährleisten. Der Würfel wird vom Kipper durch zwei Lego-Räder (Gummi) berührt, um Reibung zu erzeugen, damit der Kippvorgang nicht fehlschlagen kann. Wie wir im Laufe der Entwicklung des Projektes festgestellt haben, kam es bei der Benutzung des Kippers sehr oft zu Fehlschlägen. Diese konnten wir durch die Erzeugung der Reibung der Räder auf dem Würfel, sowie einer weiteren Lichtschranke zur Steuerung des Kippers minimieren. Ein manueller Eingriff wird nicht mehr sehr oft benötigt.

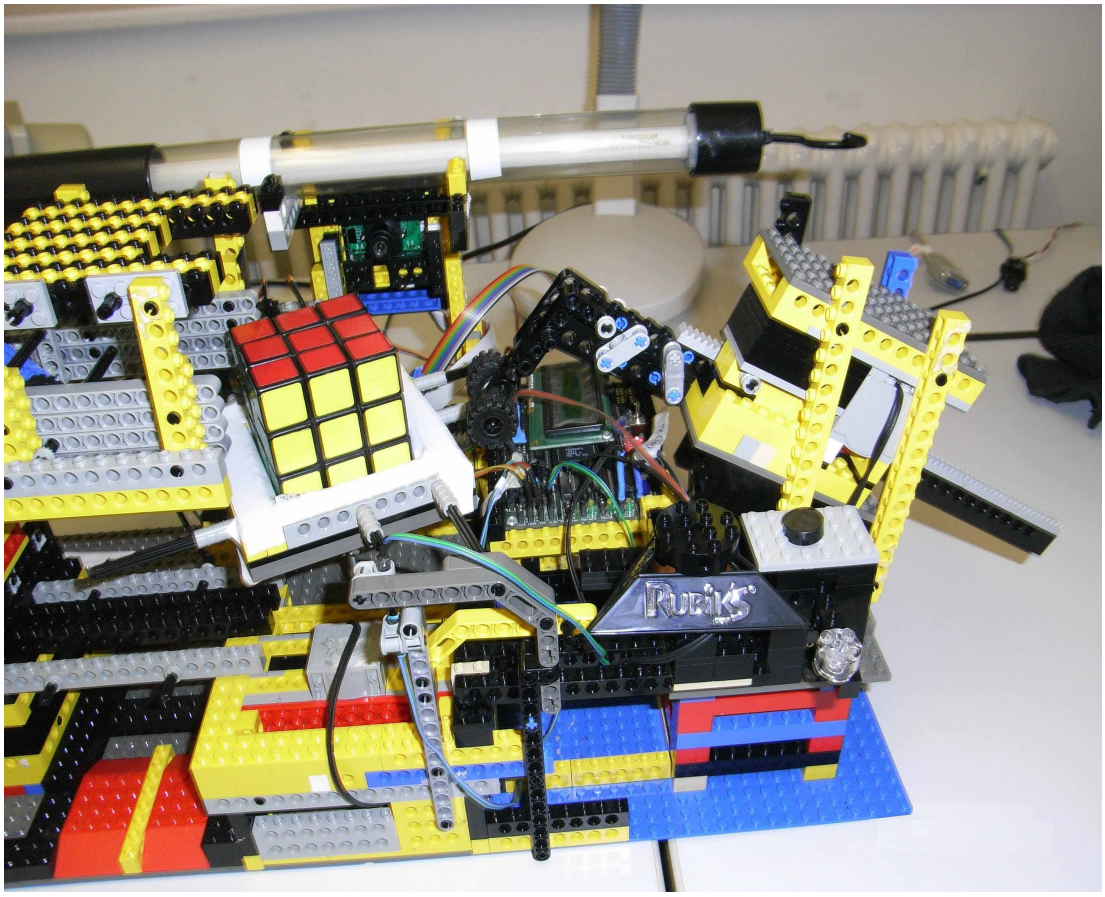

Abbildung 8: Kipper (rechte Seite)

3.3 Lösungsalgorithmus

3.3.1 Idee zur Abarbeitung

Auf der Suche nach einem Lösungsalgorithmus sind wir auf die Homepage „KantenKreuz.de“ gestoßen, welche sich nur auf die Lösung des Zauberwürfels beschränkt. Auf der Seite werden Operationen angezeigt, welche gemacht werden müssen, um einen neuen Zustand des Würfels zu erreichen. Diese Zustände und Fälle haben wir für die Lösung des Zauberwürfels verwendet.

3.3.1.1 Phasenbildung

Aufgrund des geringen Speichers des AKSEN – Boards (32 KB) haben wir uns entschieden die Lösungsphasen zu übernehmen und auch so abzuarbeiten, damit es intern nicht zu keinem Stacküberläufen kommt.
Es gibt insgesamt sieben Phasen:

- oberes Kreuz
- obere Ecken
- mittlere Ebene
- Anordnung des unteren Kreuzes
- Ausrichtung des unteren Kreuzes
- Anordnung der unteren Ecken
- Ausrichtung der unteren Ecken

Diese Phasen sind so aufgebaut, dass man nach Abarbeitung immer genau sehen / prüfen kann, ob der Würfel richtig gelöst wird.

3.3.2 Umsetzung

3.3.2.1 Internes Modell des Würfels

Die verschiedenen Phasen des Lösungsalgorithmus haben uns dazu bewegt ein internes Modell des Würfels zu erstellen. Dieses Modell soll dazu dienen alle Schritte schnell und zuverlässig zu berechnen. Anstelle dieses Modells hätten wir jedes Mal den Würfel neu einlesen müssen, wodurch sehr viel Zeit verloren gegangen wäre. Aufgrund des internen Modells benötigen wir eine 100% funktionierende Mechanik. Somit gibt es Vor- und Nachteile zu beiden Varianten. Wir haben uns für das interne Modell entschieden, was im Laufe des Projekts auch zur Fehlersuche verwendet werden konnte. (Debugging)

3.3.2.2 Methoden zur Ansteuerung der Mechanik

Die Ansteuerung der Mechanik haben wir durch insgesamt acht Methoden realisiert, welche zum einen das interne Modell verändern und zum anderen einen Buchstaben zu einem globalen Char-Array hinzufügen. Diese Buchstaben gehören zu einer Syntax, welche wir im späteren Verlauf der Dokumentation beschreiben werden. Folgende Methoden haben wir realisiert:

- turnFront → Drehung der vorderen Seite
- *turnBack → Drehung der hinteren Seite
- *turnLeft → Drehung der linken Seite
- *turnRight → Drehung der rechten Seite
- turnUp → Drehung der oberen Seite
- turnDown → Drehung der unteren Seite
- kippen → Würfel von rechts kippen
- drehen → Würfel im Uhrzeigersinn drehen (von oben betrachtet)

Aufgrund von Speicherplatzproblemen haben wir die mit * markierten Methoden auf die Methode turnFront abgebildet, was uns weiteren Platz für Optimierungen und anderen Funktionalitäten gegeben hat.

3.3.2.3 Erstellung eines Abarbeitungsstrings

3.3.2.3.1 Syntax des Programm „Cube-Explorer“

Unser Programm verwendet einen bestimmten Syntax, welchen wir im Programm „Cube-Explorer“ kennengelernt haben. Es existieren insgesamt sechs verschiedene Buchstaben, welche jeweils eine bestimmte Drehung des Würfels entsprechen. Folgende Buchstaben stehen zur Auswahl:

- L / L' → linke Seite drehen im / gegen den Uhrzeigersinn
- R / R' → rechte Seite drehen im / gegen den Uhrzeigersinn
- F / F' → vordere Seite drehen im / gegen den Uhrzeigersinn
- B / B' → hintere Seite drehen im / gegen den Uhrzeigersinn
- U / U' → obere Seite drehen im / gegen den Uhrzeigersinn
- D / D' → untere Seite drehen im / gegen den Uhrzeigersinn

Aufgrund unseres Projektes haben wir noch zwei Buchstaben hinzugefügt:

- P → Würfel im Uhrzeigersinn drehen (von oben betrachtet)
- K → Würfel kippen (von rechts)

3.3.2.3.2 Umwandlung in Maschinenbefehle

Nach Erstellung des Lösungsstrings der jeweiligen Phase wird der Syntax in Maschinenbefehle umgewandelt. Diese Befehle dienen im späteren Verlauf dazu Optimierungen vorzunehmen. Welche wir im nächsten Abschnitt näher erläutern. Nach der Optimierung wird der neue String Schritt für Schritt abgearbeitet. Folgende Maschinenbefehle werden von uns verwendet:

- p(x) → Drehung der Plattform (0/1 → im / gegen den Uhrzeigersinn)
- f(x) → Festhalter vor(1) / zurück(0) fahren
- k → Kippen des Würfels inkl. „reinschubsen“ in die Plattform

3.3.2.3.3 Abarbeitung der Befehle

Nach erfolgreicher Umwandlung und Optimierung werden die Befehle abgearbeitet. Für diesen Zweck verwenden wir eine Switch-Case-Lösung, in dem wir für jeden Befehl die jeweiligen Methoden ansprechen, welche für die Motoren-/Sensorik-Steuerung zuständig sind.

3.3.3 Optimierung

3.3.3.1 Kürzung der Abarbeitungsschritte (Maschinenbef.)

Aufgrund der vielen Schritte der jeweiligen Phasen haben wir uns überlegt Muster zu suchen, welche wir entweder komplett entfernen oder vereinfachen. Diese Muster ersparen uns manchmal sogar über die Hälfte der Zeit und ergeben nach Kürzung weitere Muster und es kürzen sich manchmal komplette Drehungen, wie zB. R und R' (eine Drehung im und eine gegen den Uhrzeigersinn). Folgende Muster haben wir bis zum jetzigen Zeitpunkt erkannt und versucht zu vereinfachen:

3.3.3.2 Funktionen um jeweilige Seitendrehungen durchzuführen.

x gibt Drehrichtung an

0 = Plattform gegen Uhrzeigersinn drehen
1 = Plattform im Uhrzeigersinn drehen

L(x) (Linke Seite)

k, f1, p(x), f0, k, k, k

R(x) (Rechte Seite)

k, k, k, f1, p(x), f0, k

D(x) (Untere Seite)

f1, p(x), f0

U(x) (Obere Seite)

k, k, f1, p(x), f0, k, k

F(x) (Vordere Seite)

p0, k, f1, p(x), f0, k, k, k, p1

B(x) (hintere Seite)

p1, k, f1, p(x), f0, k, k, k, p0

3.3.3.3. Kürzungsregeln

{k,k,k,k} → {}	4 mal Kippen ändert nichts
{k,k,k,p(x),p(x)} → {p(x),p(x),k}	anstatt 3 mal Kippen und 2 mal in eine Richtung drehen, 2 mal in eine Richtung drehen und nur noch 1 mal Kippen
{p(x),p(!x)} → {}	drehen und zurückdrehen hebt sich auf
{p(x),p(x),p(x)} → {p(!x)}	anstatt 3 mal in eine Richtung zu drehen einmal in die andere Richtung drehen
{f(0),f(1)} → {}	Greifer weg und wieder hin

4. Bedienung

Die Bedienung unseres Robots ist aufgrund der Automatisierung der Prozesse sehr einfach:

1. Der Würfel muss in die dafür vorgesehene Plattform eingelegt werden.
2. Der Schalter vom AKSEN-Board wird in den User-Modus gekippt.
3. Der Reset-Knopf wird betätigt.

Ab diesem Zeitpunkt läuft alles automatisch ab. Es kann nur, wie schon erwähnt, zu Kipp-Problemen kommen. Diese sollten dann schnellstmöglich manuell getätigt werden.

5. Resumée / Fazit

Wir haben festgestellt, das die Entwicklung des Projekts sehr viel Zeit in Anspruch nahm und die gegebene Projektzeit niemals dafür ausgereicht hätte. Die Bildverarbeitung hat alleine schon die Hälfte der investierten Zeit benötigt. Wenn man aber den Lernfaktor des Projekts betrachtet, ist dieser enorm, wenn man Ihn mit anderen Veranstaltungen betrachtet. Eigenständiges Lösen von Problemen und bearbeiten von nicht erwarteten Situationen sind ein wichtiges Kriterium für die Zukunft in der IT-Branche und wurde im Rahmen des gewählten Projekts komplett behandelt.