Echtzeitbilderkennung zur Steuerung eines Flipperkastens

Johannes Weinbuch

26. Januar 2011

Betreuer
Dr. Ivo Blöchliger

Co-Betreuer
Dr. Adrien Cornaz

Inhaltsverzeichnis

Vorwort

In der heutigen Welt wird immer mehr automatisiert, Produktionslinien werden vollautomatisch Kamerabasiert auf fehlerhafte Produkte hin überprüft, Autos warnen den Fahrer, wenn er unbeabsichtigt eine Sicherheitslinie überfährt und Google Street View ist in der Lage, Personen auf Bildern automatisch zu anonymisieren.

Allen gemein ist, dass hierzu Bilder analysiert werden, teils mit einer gewissen Verzögerung, teils aber auch in Echtzeit, und das mit stetig steigenden Anforderungen. Ohne unverzögerte Reaktionen der Elektronik würden diese Systeme nicht nur nicht funktionieren, sie würden sogar zum Sicherheitsrisiko werden.

Es machte für mich einen grossen Reiz aus, selbst einmal so ein System zu entwickeln, um zu sehen, was dahinter steckt. Auf die Idee, einen Flipperkasten zu steuern kam ich, als ich ein System, dass ich steuern wollte, festlegen musste. Durch meine Affinität zu Flipperkästen fiel die Wahl schnell darauf, einen solchen Kasten autonom spielen zu lassen.

Kapitel 1

Einleitung

Ein Flipperkasten ist ein grosses, auf vier Beinen stehendes Gerät. Im Innern befindet sich ein geneigtes Spielfeld, auf dem eine Stahlkugel rollt. Am unteren Ende sind bewegliche, über aussen montierte Knöpfe bedienbare, Arme mit einer Lücke dazwischen angebracht. Sie können der Kugel neuen Schwung nach oben verleihen. Fällt die Kugel nach unten, so ist das Spiel verloren.

Dies ist eine kurze Grundbeschreibung dessen, was ein Flipperkasten tut. Er ist als Geschicklichkeitsspiel für den Menschen aufgebaut. In dieser Arbeit wird anstelle eines Menschen ein Computer die Steuerung übernehmen und versuchen, die gleiche Aufgabe zu erfüllen. Auf den ersten Blick erscheint dies sinnlos, denn das Ziel dieser Arbeit ist es, einen Computer so zu erweitern, dass er ein Unterhaltungsgerät für den Menschen bedienen kann. Auf den zweiten Blick wird klar, dass der Sinn dieser Arbeit nicht im fertigen Produkt zu suchen ist, sondern in der Arbeit selbst. Die Entwicklung eines Systems, dass diese Aufgabe ausführen kann ist sinnvoll, da daraus Erkenntnisse für andere, ähnliche Systeme gewonnen werden können.

1.1 Zielsetzung

Das Hauptziel dieser Arbeit besteht darin, am Ende einen funktionstüchtigen Flipperkasten zu haben, der die Kugeln ohne Hilfe eines Menschen im Spiel halten kann. Hierbei ist festzuhalten, dass „im Spiel halten“ relativ gesehen werden muss, denn diese Aufgabe ist auch für Menschen nicht immer ganz einfach. Denn genau auf diesem Punkt basieren die Flipperkästen, die an öffentlichen Orten stehen und mit einem Münzprüfer ausgestattet sind. Es wird darauf vertraut, dass der Ball nur so kurz im Spiel bleibt, dass mehr Geld eingeworfen wird, als Wartungskosten entstehen. Somit sind schon wenige Treffer

des Balles als Erfolg zu werten.

Sekundärziel ist es, Möglichkeiten und mögliche Stolpersteine beim Aufbau eines Echtzeitbilderkennungssystems zu dokumentieren. So kann diese Arbeit auch als Grundlage für andere Projekte dienen, die eine Bilderkennung implementieren wollen und helfen, grosse Probleme zu umgehen.

Kapitel 2

Hauptteil

2.1 Hardware

In diesem Kapitel wird die verwendete Hardware dieses Projekts beschrieben. Unter Hardware werden hierbei sämtliche Geräte und Kabel verstanden, die für dieses Projekt nötigerweise physisch vorhanden sein müssen. Für jeden Teil wird dabei angegeben, was genau die zu erfüllende Aufgabe ist, welche Alternativen es in der Auswahl gegeben hätte. Ausserdem wird begründet, weshalb die entsprechenden Teile verwendet wurden.

Dieses Kapitel ist in 4 Unterkapitel aufgeteilt. Das Erste befasst sich mit dem zu steuernden Flipperkasten. Im Zweiten werden die Steuergeräte für den Flipperkasten behandelt, also Computer und zusätzliche Elektronik. Im Dritten wird die verwendete Kamera erläutert. Das Vierte schliesslich dient als Zusammenfassung für sonstige Geräte, die nötig waren.

2.1.1 Der Flipperkasten

Der verwendete Flipperkasten stammt von Williams und ist vom Modelltyp „The Flintstones“. Er stammt aus dem Jahr 1994 und verfügt über so genannte Flipper Opto Boards zur Steuerung der Flipperfinger. Zur elektronischen Steuerung der Finger wird ein Microcontroller verwendet. Die Basisschaltung des verwendeten Microcontrollers stammt aus der Projektarbeit „Datenverarbeitung“ [Koller(2010)] und wurde für dieses Projekt modifiziert. Weiterhin wurden die Lampen im Spielfeld deaktiviert, um eine Störung der Kamera zu vermeiden und die Glasplatte über dem Spielfeld wegen der durch ihr erzeugten Reflexionen.

Es wurden zwei Relais an zwei Outputpins des Controllers gelötet. An die Lastseite des Re-

lais wurden Stiftleisten gelötet, damit der Stecker der Flipper Opto Boards aufgenommen werden kann. Es wurden also Die Signalleitungen mit Grund über das Relais verbunden. Da die Grund- und +12V-Leitungen der Opto Boards zusammen hängen, mussten gewisse Anschlüsse der Stiftleisten miteinander verbunden werden, was in Abbildung 2.1 aus der Dokumentation des Flipperkastens [Wil(1994)] ersichtlich ist.

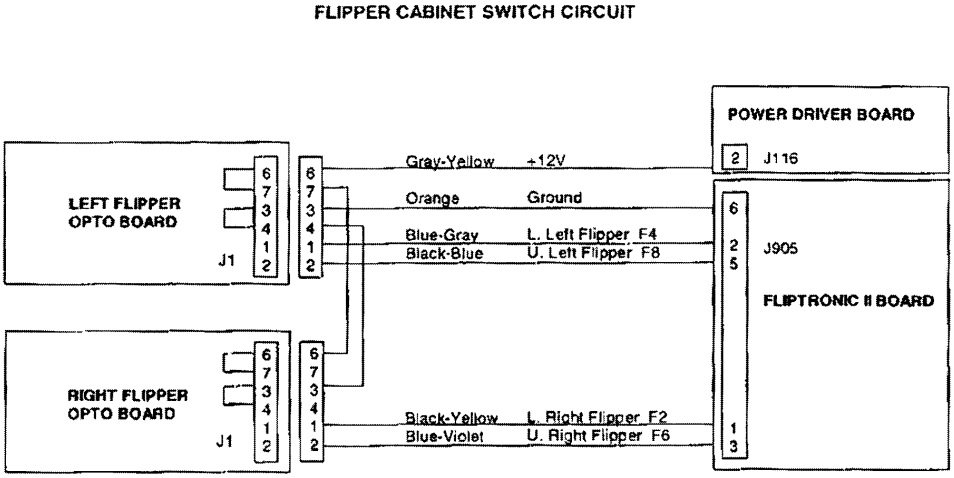

Abbildung 2.1: Schema des Schaltkreises der Flipperknöpfe

Im vorliegenden Gerät war die Verkabelung aber nicht nach Vorgabe der Dokumentation ausgeführt, was sich durch die Belegungen der Stecker gezeigt hat. Auf dem linken Stecker ist an den Pins 6 und 7 je ein grau-gelbes Kabel angeschlossen, so wie beschrieben. Eine Messung ergab, dass das Kabel 12V führt, also wurden die beiden Pins verbunden. Allerdings hat der linke Stecker nur eine Orange Grundleitung. Dafür hat der rechte Stecker zwei Grundleitungen, wobei eine davon mit dem linken Stecker verbunden ist. Also wurden sie auf dem rechten Stecker verbunden.

Prinzipiell würde die +12V-Leitung aber nicht gebraucht, da die elektronische Steuerung auch ohne sie auskommt. Sie wird lediglich benötigt, um die LEDs der Lichtschranken (beschrieben in Kapitel 2.2) zu betreiben. Sie wurde dennoch verbunden, da so auch Experimente möglich sind mit einer Computergesteuerten und einer von Hand gesteuerten Seite.

Die so gebauten Geräte sind also prinzipiell mit anderen Flipperkästen kompatibel, es existieren einige andere Modelle, die auch über Flipper Opto Boards verfügen [ipd(2010)]. Allerdings muss stets zuerst in der Dokumentation geprüft werden, ob das gewünschte Modell die Flipper Opto Boards gleich ansteuert und ob die Dokumentation auch dem realen Sachverhalt entspricht. So lässt sich die Aussage treffen, dass sich dieses Projekt auch mit vielen anderen Flipperkästen hätte durchführen lassen. Der Flintstones-Kasten wurde primär gewählt, weil er bereits vorhanden war.

2.1.2 Die Steuerung

Die Steuerung erfolgt über einen Computer. Ein Computer verfügt über ein Betriebssystem, Ein- und Ausgangsports und kann über einen Bildschirm Informationen ausgeben. So ist die Entwicklung des Projets stark vereinfacht gegenüber der Variante, alles über Microprozessoren zu steuern. Man braucht sich dann nämlich nicht mehr im Gleichen Masse um die Ansteuerung der anderen Komponenten, die Ausgabe von Entwicklerinformationen oder die Algorithmen zur Bilderkennung kümmern.

Gewisse Funktionalitäten muss der Computer aber mindestens mitbringen. So muss er in der Lage sein, die Bilder schnell genug zu bearbeiten, damit es möglich wird, die Situation im Flipperkasten rechtzeitig zu erfassen und darauf zu reagieren. Zudem muss er eine mit der Kamera kompatible Schnittstelle bereitstellen, damit die Bilder auch verarbeitet werden können. Ausserdem muss er in der Lage sein, den Flipperkasten zu steuern. Dies geschieht mit Hilfe eines Adapters und eines Microcontrollers und ist in Kapitel 2.2 genauer beschrieben.

2.1.3 Die Kamera

Als Kamera wird ein Digitaler DV-Camcorder von Sony aus dem Jahr 2004 verwendet. Er kommuniziert über seinen eingebauten i.LINK-Anschluss, was nichts anderes als die IEEE1394 Schnittstelle [iee(2010)], mit dem Computer.

Als alternativen wären zwei verschiedene Webcams vorhanden gewesen. Er wurde aufgrund eines Tests ausgewählt. Auf dem zu verwendenden Computer wurde für jede Kamera nacheinander ein Programm gestartet, dass das Bild von der Kamera ausgibt. Dann wurde die Reaktionszeit der Kamera getestet, indem ein Gegenstand vor der Kamera hin und her bewegt wurde. Für genaue Ergebnisse war dieser Test natürlich nicht zu gebrauchen, jedoch war von blossem Auge zu erkennen, dass das Bild vom Camcorder am schnellsten aktualisiert wurde. Die Webcams waren so viel langsamer, sodass ein weiterer Test nicht nötig war.

Der Camcorder hat zudem den Vorteil, dass sich die Belichtung und der Fokus manuell einstellen lassen. Ausserdem kann man über seinen Ausklappbaren LCD-Monitor eine erste Grobeinstellung des zu filmenden Bildausschnitts vornehmen, ohne auf den Computer blicken zu müssen.

Leider hat das Gerät auch zwei Nachteile. Zum einen haben die Digitalsensoren in den letzten Jahren erhebliche Fortschritte gemacht, vor allem was das Rauschverhalten angeht. Es leuchtet ein, dass ein Bild, welches weniger Störungen durch ein zufälliges Bildrau-

schen enthält, leichter und genauer von einem Computer zu verarbeiten ist. Besonders bei schlechten Lichtverhältnisse offenbart sich das Bildrauschen. Der andere Nachteil ist, dass dieser Camcorder zwar direkt am Stromnetz via Netzteil betrieben werden kann. Allerdings schaltet er sich ohne Aufnahme auf Band nach fünf Minuten automatisch ab. Diese Einstellung kann nur durch das Aufnehmen auf Band umgangen werden. Dies hat die Programmierung der Software etwas erschwert, da man sich neben der Programmierung auch noch regelmässig der Kamera widmen muss.

2.1.4 Sonstiges

Es gibt noch vieles, was für diese Arbeit an Werkzeug und Material gebraucht wurde. Das ist nicht der Inhalt dieses Kapitels. Hier geht es um nicht selbstverständliche Punkte, die noch beachtet werden mussten.

Die Beleuchtung ist einer dieser Punkte. Wie in Kapitel 2.1.3 bereits erwähnt wurde, hat die Kamera bei schlechter Beleuchtung ein erhöhtes Rauschverhalten. Es hat sich gezeigt, dass das Rauschen zu stark war, um den Ball zuverlässig zu erkennen. Daher wurde unmittelbar neben dem Spielfeld ein 500 Watt Strahler platziert, der es ausleuchten soll. Dazu kamen noch diverse andere, kleinere Lampen, welche die Schatten reduzieren sollen. So wurde die Beleuchtung für das Projekt brauchbar. Allerdings ist die Kamera in diesem Aufbau noch immer nicht auf der kleinsten Empfindlichkeitsstufe. Das Rauschen liesse sich durch noch mehr Licht also noch weiter reduzieren. Diese Möglichkeit wurde aber verworfen, da diese Strahler eine erhebliche Wärmeentwicklung aufweisen, die dem Kasten unter Umständen schaden könnte.

Im Verlaufe der Programmierung hat sich auch gezeigt, dass die Software den Kasten überlasten kann. Der Kasten besitzt im inneren Feinsicherungen im amerikanischen Format $\frac{1}{4}$ Zoll auf $\frac{5}{4}$ Zoll. Es ist nun eine 7 Ampere Sicherung diesen Typs durchgebrannt. Sie war nur relativ schwer aufzutreiben und blockierte so die Arbeit, da vor allem auch unklar war, ob noch weitere Bauteile des Kastens in Mitleidenschaft gezogen wurden.

2.2 Elektronische Ansteuerung

In diesem Kapitel wird die Art und Weise beschrieben, wie der Flipperkasten von einem Computer aus elektronisch gesteuert wird. Der praktische Aufbau des Schaltkreises wurde bereits im Kapitel 2.1 beschrieben und wird daher hier nicht mehr näher betrachtet. Der Fokus hier liegt auf den Aufgaben, die die Elektronik übernimmt und wie die Elektronik aufgebaut werden musste, damit die Aufgabe zuverlässig erfüllt werden kann.

Die Elektronische Ansteuerung erfolgt über einen ATMega8 Microcontroller[Atmel(2010)]. Er kommuniziert über seine UART Schnittstelle, welche auch in der Dokumentation des Microcontrollers beschrieben wird, und einen MAX 232 Pegelwandler[Max(2010)] mit dem Computer über die serielle Schnittstelle. Da moderne Computer, insbesondere Laptops keine solche Schnittstelle mehr besitzen, wurde ein FTDI USB zu Seriell Adapter[Fut(2010)] verwendet.

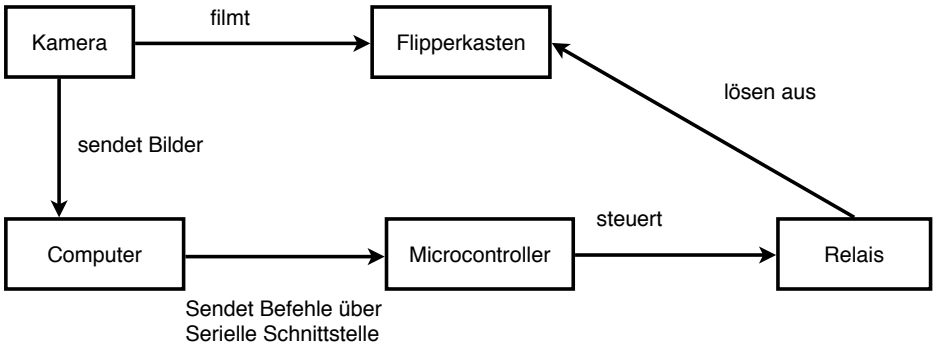

Abbildung 2.2: Aufbau der Anlage, die Pfeile stellen Datenflüsse dar.

So können nun vom Computer aus auf einfache Art und Weise Informationen an den Microcontroller geschickt werden, indem einzelne Zeichen an die Gerätedatei des Adapters geschickt werden. Diese Informationen werden von der Firmware verarbeitet, was später im Kapitel Software noch genauer erläutert wird.

Die Flipperfinger im Kasten lösen aus, sobald der Knopf auf der Aussenseite des Gehäuses gedrückt wird. Auf der Innenseite ist das Flipper Opto Board, auf welchem eine Gabellichtschranke befestigt ist. Diese ist standardmässig unterbrochen, das heisst, von der Infrarot-LED der Lichtschranke fällt kein Licht auf den Fototransistor. Über diesen Transistor ist die Datenleitung zur Hauptplatine, mit Grund verbunden. Wird nun der Knopf gedrückt, fällt Licht auf den Fototransitor. Er wird leitend, und so wird die Datenleitung gegen Grund gezogen. Auf dieses Signal hin löst die Flipperelektronik den entsprechenden Flipperfinger auf dem Spielfeld aus. Dieser Sachverhalt wird auf Abbildung 2.3 aus der Dokumentation von Williams zum verwendeten Flipperkasten [Wil(1994)] ersichtlich.

Um das nun elektronisch auszuführen, wurden die Steckverbindungen von den Flipper Opto Boards getrennt und die entsprechenden Pins mit einem eigenen Stecker verbunden. Die Datenleitung wird nun nicht mehr über die Gabellichtschranke gegen Grund gezogen, sondern über ein Relais, welches vom Mikrocontroller aus gesteuert wird. So ist die galvanische Trennung der Stromkreise gewährleistet. Dies ist sehr wichtig, da der verwendete Flipperkasten relativ alt ist und hin und wieder Kabelbrüche auftreten können. Nur so kann sicher vermieden werden, dass in so einem Fall der Mikrocontroller und der damit

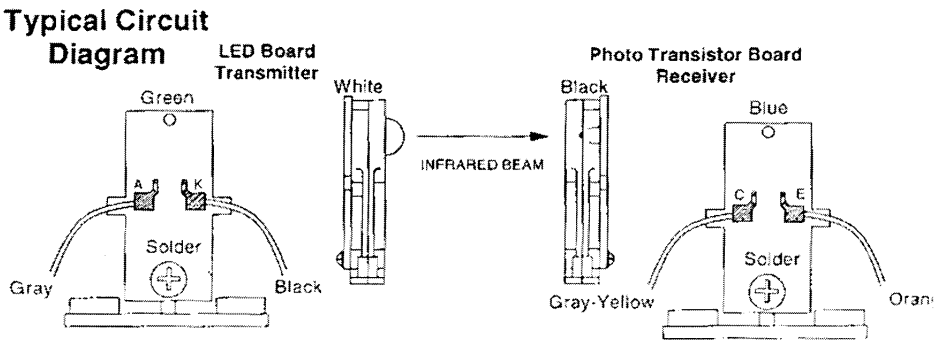

Abbildung 2.3: Aufbau der Gabellichtschranke: In der Mitte Ist der Infrarotstrahl dargestellt, der von der LED auf den Transistor fällt.

verbundene Computer keinen Defekt erleiden.

Dimensionierung der Schaltung

Abbildung 2.4 zeigt die Beschaltung eines der beiden eingebauten Relais, K1. Da das Relais mit einer Spule über eine Induktivität verfügt, muss der Schaltplan berücksichtigen, dass keine Spannungsspitzen am Mikrocontroller auftreten dürfen. Daher wurde die Freilaufdiode D1 im Schaltplan eingesetzt. Sie ist in Sperrichtung geschaltet. Sie schliesst die Spule kurz, sobald eine Spannung von mehr als 0.7 Volt [Dio(2002)] entgegen der Betriebsstromrichtung anliegt und vermeidet so die Spannungsspitze.

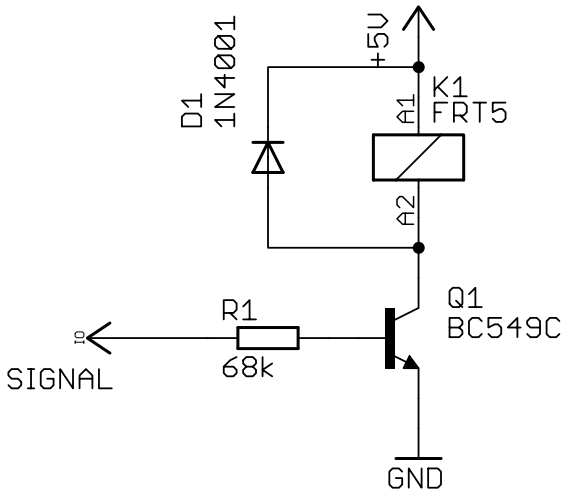

Abbildung 2.4: Schema der Schaltung am Output-Pin des Microcontrollers, erstellt mit Eagle[eag(2010)]

Das Signal vom Controller wird an die Basis von Transistor Q1, der als elektrischer Schalter dient, gelegt. Dieser ist bei einer positiven Spannung leitend, bei 0 Volt blockiert er.

Da der Transistor die selbe Strom-Spannungskennlinie wie eine Diode hat, ist ein Vorwiderstand $R1$ zwingend notwendig. Um ihn zu berechnen, ist zunächst aus dem Datenblatt [Phi(1999)] die Stromverstärkung abzulesen. Sie liegt in diesem Fall ungefähr bei einem Faktor von 480. Der Strom an der Basis des Transistors wird also um das 480-Fache verstärkt. Das Relais, welches am Kollektor des Transistors hängt, hat laut Datenblatt einen Widerstand von 178Ω [rel(2004)] . Bei einer Spannung von 5V fliesst gemäss dem Ohmschen Gesetz ein Strom von $I = \frac{U}{I} = \frac{5V}{178\Omega} = 28\text{mA}$. Folglich muss an der Basis ein Strom von $\frac{28*10^{-3}A}{480} = 5.83*10^{-5}\text{A}$ fliessen, damit das Relais schaltet.

Um nun den benötigten Vorwiderstand zu berechnen, wird wiederum das Ohmsche Gesetz angewendet. Der Microcontroller liefert 5V, abzüglich des Spannungsabfalls des Transistors V_{BE} von 0.7V. Der Spannungsabfall ist wiederum aus dem Datenblatt zu entnehmen. R ist also $R = \frac{5V-0.7V}{5.83*10^{-5}A} = 72.3\text{k}\Omega$. In der Schaltung wurde der nächst kleinere Widerstandswert, 68kΩ verwendet, da so der Transistor sich so in einem übersättigten Zustand befindet, so dass das Relais sicher schaltet, ohne dass der Strom zu gross wird.

Somit lässt sich der Flipperkasten sicher steuern. Das Relais schützt den Computer vor zu hohen Spannungen aus dem Flipperkasten. Die Gefahr durch eine Überlastung der Kastenelektronik durch zu viele Auslösungen ist zwar gegeben, jedoch ist der Kasten entsprechend abgesichert, sodass bei einer Überbeanspruchung der Kastenelektronik lediglich eine Sicherung durchbrennt und so grösserer Schaden vermieden werden kann.

2.3 Software

In diesem Kapitel wird auf die nötige Software für so ein Projekt eingegangen. Unter Software werden dabei die Programme verstanden, die speziell für dieses Projekt nötig sind, konkret sind das die Firmware des Microcontrollers und die Bildverarbeitungssoftware am Computer. Bei der Programmierung wurde darauf geachtet, vorhandene Ressourcen möglichst gut auszunutzen.

2.3.1 Die Firmware des Microcontrollers

Die Firmware wurde zum Grossteil wie die Basisplatine selbst aus dem letztjährigen Projekt Datenverarbeitung an der Kantonsschule Wohlen übernommen [Koller(2010)]. Das hielt den Programmieraufwand in engen Grenzen. Die Firmware konnte bereits auf einen Input von der Seriellen Schnittstelle hin eine analoge Spannungsmessung Starten. Die Kernfunktionalitäten waren also bereits vorhanden. Es konnte bereits ein ASCII-Buchstabe an den Controller gesendet werden und er war in der Lage, abhängig vom gesen-

deten Buchstaben eine andere Aktion auszuführen, in diesem Fall eine Spannungsmessung und das anschliessende zurücksenden des gemessenen Wertes an die serielle Schnittstelle. Die Änderungen in der Firmware beschränken sich also darauf, die Spannungsmessung durch das Schalten von Ausgangspins zu ersetzen. Die Dokumentation hierfür steht im Datenblatt des Microcontrollers [Atmel(2010)]. Da die ursprüngliche Firmware nicht sehr viel Speicher belegte, wurden die meisten Unterroutinen wie zum Beispiel die Funktion für das Senden auf die serielle Schnittstelle unangetastet gelassen. Sie stören nicht und könnten unter Umständen zur Erweiterung der Funktionalität weitergenutzt werden.

2.3.2 Die Software zur Ballerkennung

Die Software zur Ballerkennung ist in C++ geschrieben worden und besteht im Wesentlichen aus zwei Teilen und einigen Hilfsprogrammen. Der erste Teil ist dafür verantwortlich, dass der Computer den Ball erkennt und differenzieren kann, ob nun ein Flipperfinger ausgelöst werden soll oder nicht. Ist in diesem Teil nun eine Entscheidung gefallen, dass der Flipperfinger ausgelöst werden muss, so wird der zweite Teil aktiv. Er kommuniziert mit dem Flipperkasten, aktiviert die gewünschten Flipperfinger, wartet eine definierte Zeitspanne und deaktiviert sie dann wieder. Zu den Hilfsprogrammen gehören unter anderem dvgrab [dvg(2010)], ein Programm, dass die Aufnahme von der verwendeten Kamera steuert und ein selbstgeschriebenes Shellscript, dass die Programme startet, sodass sie so zusammenarbeiten, wie sie sollen.

Der Startpunkt der Programmierung war ein Beispiel aus einer Einführung in die Programmierung mit OpenCV [Agam(2006)]. Von dort aus wurde die Software Stück für Stück weiterentwickelt. Als Hilfsmittel kamen dabei die Dokumentation von OpenCV [ope(2010)] und die dort verlinkten Webseiten zum Einsatz.

Der folgende Teil dieses Kapitels wird nun die Software detaillierter dokumentieren. Verschiedene wichtige Punkte werden in Unterkapitel gegliedert, damit es übersichtlich bleibt. Dabei werde ich mich auf die Erkennung und den Abschuss des Balles an sich, das verwendete Multiprocessing und das Errorhandling beschränken. Zuletzt wird noch kurz auf den Entwicklungsprozess der Software eingegangen.

Erkennung und Abschuss des Balles

Zur Erkennung eines Balles existieren bereits einige Methoden, unter anderem anhand der Farbe oder über Kanten im Bild. In diesem Projekt ist die Erkennungsmethode über ein Referenzbild gewählt worden, da der Ball eine glänzende Metalloberfläche hat. Man sieht also statt dem Ball selbst die Umgebung des Balles auf eine Kugel, also den Ball,

projiziert. Eine Erkennung über die Farbe war somit ausgeschlossen, da ein Ball mit anderen Eigenschaften den Kasten unter Umständen beschädigen könnte. Nicht glänzende und damit nicht reflektierende, farbige Bälle, die sich ausreichend von den Farben dieses Spielfelds abheben, waren mir zum Zeitpunkt dieser Arbeit nicht bekannt. Eine Erkennung über die Kanten im Bild schien auch wenig Erfolg versprechend, da durch die Reflexionen auch auf der Kugel selbst Kanten entstehen würden.

Die Referenzbildmethode beruht auf folgender Überlegung: Wenn man ein Bild von der Fläche ohne Ball und der Fläche mit Ball übereinander legt, muss der Ball also da sein, wo der Unterschied ist. Diese Methode lässt sich durch Bitoperationen im Computer umsetzen. Es wird ein Referenzbild geladen, man nimmt ein weiteres Bild aus dem gleichen Blickwinkel auf, und vergleicht beide Bilder Pixelweise mit einer XOR-Verknüpfung. Die XOR-Verknüpfung hat die Eigenschaft, dass gleiche Pixel eine 0 produzieren und unterschiedliche Pixel eine 1. Somit ergibt sich ein Schwarzweiss-Bild, worauf man nun nach einem Kreis suchen kann.

In der Praxis wurde das mit Hilfe der Programmbibliothek OpenCV [ope(2010)] realisiert. Diese Bibliothek bietet unter anderem einfache Funktionen, um mit Bildern und Streams umzugehen und ermöglicht auch das Erstellen graphischer Benutzerinterfaces. Zu Beginn wird also der Stream von der Kamera initialisiert und als Livebild auf dem Computerbildschirm wiedergegeben. Auf dem Livebild werden ausserdem Hilfslinien eingeblendet, damit die Kamera optimal positioniert werden kann. Durch den Druck auf die Escape-Taste wird nun der Computer angewiesen, das Referenzbild zu erstellen. Dazu wird das aktuelle Bild aus dem Stream gespeichert, sodass es nun permanent das Referenzbild ist.

Nachdem dies geschehen ist, geht das Programm in eine Endlosschleife. Es wird das jeweils nächste Bild aus dem Stream geladen. Das Bild wird in Graustufen konvertiert und ein Gauss'scher Weichzeichner wird auf das Bild angewendet, um das Kamerarauschen zu kompensieren. Danach wird wie zuvor beschrieben, das Bild mit der Referenz mit XOR-Verknüpft. Vor der Speicherung ist natürlich auch das Referenzbild in Graustufen konvertiert und weichgezeichnet worden.

Darauf wird das Resultat noch einmal weichgezeichnet, da die Beiden Bilder aus Graustufen bestehen und durch das Rauschen auch noch Ausschläge entstanden sind, wo kein Ball ist. Diese Ausschläge werden so geglättet. Danach wird das Bild anhand eines Schwellwertes in schwarzweiss konvertiert. Das heisst, es wurde ein Grenzwert festgelegt, bis zu dem Graustufen in schwarze Pixel umgewandelt werden und alles, was heller ist, wird weiss. Die Parameter für die Weichzeichner und die Schwarz-Weiss-Konvertierung sind durch Experimente bestimmt worden.

Das so entstandene Differenzbild wird als Parameter an die Funktion cvHoughCircles

weitergegeben. Diese Funktion ist in der OpenCV Bibliothek definiert und findet Kreise in einem Bild [hou(2010)]. Das ist genau die gesuchte Funktionalität, denn eine Kugel ist unabhängig vom Blickwinkel auf ein zweidimensionales Bild projiziert ein Kreis. Dieser theoretische Ansatz wurde in der Praxis versucht zu erreichen durch eine möglichst gute, das heisst annähernd schattenfreie Ausleuchtung des Spielfelds.

Daraus erhält man nun ein Array mit sämtlichen erkannten Kreisen. Dieses Array wird nun untersucht. Ist es leer, so wurde kein Kreis gefunden, und man kann wieder zum Beginn der Schleife springen. Enthält es sehr viele Elemente, so wird das Ergebnis auch verworfen, da nicht nur ein Ball, sondern auch noch etwas anderes, wie zum Beispiel ein Flipperfinger in Bewegungals Kreis erkannt wurde. Enthält er nur eines oder sehr wenige Elemente, so handelt es sich um einen Ball, der unter Umständen mehrfach erkannt wurde. Nun werden von sämtlichen erkannten Bällen die Koordinaten, die leicht voneinander abweichen, gemittelt. Liegen sie Innerhalb einer im Programm definiertem heissen Zone, werden die Flipperfinger aktiv. Wenn der Ball auf der linken Spielfeldhälfte liegt, wird an den Microcontroller eine ASCII 2 gesendet, was im Binärsystem 10 entspricht. Das linke Bit ist also eingeschaltet. Analog wird für den rechten Flipperfinger und beide zugleich verfahren.

Multiprocessing

Bei der Entwicklung zeigte sich, dass eine einfache menschliche Taktik auf den Computer übertragen nicht funktioniert. Es ist nicht möglich, den „Knopf“ kurz zu drücken und sofort wieder loszulassen, weil der Computer dies viel zu schnell tut. Der Strom der Spule, die die Flipperfinger betätigt, wird abgeschaltet, solange sie noch nicht ausreichend lange betätigt wurde. Als Folge davon sind die Schüsse deutlich zu schwach, um die Kugel nach oben zu befördern.

Es muss also der Flipperfinger eingeschaltet, eine gewisse Zeit gewartet, und der Finger nach Ablauf dieser Zeit wieder ausgeschaltet werden. Diese Wartezeit wurde frei wählbar gestaltet, sodass ein Ball auch gefangen und gehalten werden könnte. Daraus ergab sich das Problem, dass die Wartezeit vergeudet wurde. In dieser Zeit wurde der Stream nicht weiter verarbeitet, und wurde somit instabil, die langen Wartezeiten führten oft zu undefiniertem Verhalten, wie beispielsweise zu Programmabstürzen.

Ein möglicher Lösungsansatz wäre es gewesen, den Flipperfinger einfach auszulösen, und beim nächsten Durchlauf zu prüfen, ob die Gewünschte Zeit bereits abgelaufen war. Dieser Ansatz erschien zu unflexibel und ungenau, denn die Funktion, die den Ball im Bild sucht, ist zum Teil relativ langsam, wenn sich der Flipperfinger gerade bewegt. Deshalb

wurde der Weg gewählt, das Programm auf zwei Prozesse aufzuteilen. Der Hauptoprozess ist im kompilierten Programmfile „ballrecognition“ untergebracht. Es wird ein Fork erzeugt und der Kindprozess wird durch den Hilfsprozess aus dem Programmfile „shooter“ ersetzt. Dieses Procedere wurde aus einem Tutorial aus den GIDForums übernommen [GID(2004)].

Auf diese Art und Weise konnten nun einfach beliebige Wartezeiten realisiert werden. Der Hauptprozess erkennt einen Ball und teilt dem shooter-Prozess mit, dass er nun für eine bestimmte Zeit, oder solange, bis ein anderes Kommando kommt, den gewünschten Flipperfinger auslöst, während er sich selbst weiter dem Videostream widmet. Die Kommunikation läuft dabei über eine Kommunikationspipe, die nur einseitig ist, da der shooter-Prozess nichts zurückzumeldet.

Errorhandling

Ein weiterer wichtiger Punkt, den es zu beachten gilt, ist das Errorhandling, also das frühzeitige Erkennen und Vermeiden von Fehlern. Und sollte doch einmal ein Fehler passieren, so muss er abgefangen werden, damit vermieden werden kann, dass das Programm in einen nicht vorgesehenen Zustand gerät und dadurch eine Gefahr für die Hardware wird.

Um zu verstehen, wo die Fehlerquelle liegt, muss der Programmstart genauer betrachtet werden. Zunächst wird das Hilfsprogramm dvgrab gestartet, welches einen Stream von der Kamera initialisiert und ihn in eine named pipe [nam()] ausgibt. Dies ist nötig, da es mir nicht gelungen ist, den Stream direkt von der Kamera in OpenCV zu öffnen. Stattdessen wird nun ein Stream von einem File geöffnet, der zuvor erstellten named pipe. Genau dort stürzte das Programm oft ab mit der Meldung, das ein SIGTERM [sig(1993)] von der OpenCV Bibliothek erzeugt wurde, weil eine leere Matrix da war, wo keine sein sollte.

Der Plan war nun, das Signal abzufangen, und daraufhin erneut zu versuchen, ein Bild aus dem Stream zu laden. Es wurde also ein Signalhandler nach einem Beispiel von Douglas C. Schmidt programmiert [Schmidt(1998)] . Nach der Implementierung des Handlers hat sich ein äusserst interessanter Effekt ergeben. Das Programm stürzte nicht mehr beim Start ab, was es zuvor in etwa bei jedem zweiten Start tat. Allerdings, das war das merkwürdige, wurde der Signalhandler nicht aktiv, denn er ist darauf programmiert worden, zuerst Text auf die Kommandozeile auszugeben, der den Benutzer über den Fehler zu informieren. Der Text wurde nie ausgegeben. Der Fehler, den der Signalhandler abfangen sollte, ist also nicht mehr aufgetreten.

In der Folge davon wurde das Signalhandling so angepasst, dass nicht versucht werden

soll, von vorne zu beginnen, sondern das primär eine sichere Beendigung des Programmes gewährleistet ist. Der shooter-Prozess wurde also so umprogrammiert, dass er zusätzlich zu den Abschusscodes für die Flipperfinger auch einen Code zum sofortigen beenden ausführen kann. Die Signalhandler sollten nun eben diesen Code an den shooter-Prozess senden, damit sicher nicht mehr geschossen wird.

Durch einen Tippfehler wurde aber ein anderer Code an den shooter-Prozess gesendet, was fatal war. Das Signal wurde nun bei einem echten Programmabsturz abgefangen. Der shooter-Prozess hat darauf hin einen Code empfangen, mit dem er nichts anfangen konnte. Das Hauptprogramm war abgestürzt, der andere Prozess lief aber weiter, und betätigte aufgrund des falschen Inputs unkontrolliert die Flipperfinger. Unglücklicherweise wurden die Auslösungen dann zufällig so gepulst, dass durch die Spulen, welche die mechanische Kraft für die Flipperfinger liefern, ein zu grosser Strom floss, sodass eine Sicherung im Flipperkasten durchgebrannt ist. Auf dieses Problem wird noch genauer im Kapitel 3.2 auf Seite 19 eingegangen.

Nachdem dies aber korrigiert worden ist, läuft die Software wieder einwandfrei. Bei so komplexen Aufgaben wie Multiprocessing und Bilderkennung kann und muss das Errorhandling aber noch weiter ausgebaut werden, vor allem, wenn man beabsichtigt, die Software produktiv zu nutzen und sie sicherheitsrelevante Funktionen hat. Für dieses Projekt ist eine derartige Erweiterung aber unterlassen worden, da vorerst kein produktiver Einsatz geplant ist.

Kapitel 3

Schluss

Aus dieser Arbeit lassen sich einige Erkenntnisse ziehen. Sie werden im folgenden Kapitel beschrieben. Einen eigenen Platz erhalten die Probleme, die aufgetaucht sind. Aus Problemen können viele Lehren gezogen werden. Da genau diese Lehren äusserst wichtig sind, wird dieses Kapitel auch den Schluss der Arbeit bilden.

3.1 Fazit

Die Zielsetzung dieser Arbeit war es, einen selbst spielenden Flipperkasten zu bauen. Dieses Ziel wurde erreicht. Der Flipperkasten spielt zwar nicht sehr gut, wenn man ihn mit dem Menschen vergleicht. Lässt man allerdings die gleiche Anzahl Kugeln durch den Flipperkasten aufen, ohne dass er ferngesteuert wird, so ergibt sich doch ein signifikanter Effekt. Der Computer macht mehr Punkte als ein Kasten, der nicht bedient wird. Objektiv wurde also das gesetzte Ziel erreicht.

Betrachtet man den Sachverhalt aus der ästhetischen Sichtweise, so kann als Kritikpunkt ein zufällig erscheinendes Auslösen der Flipperfinger angeführt werden. Dies ist auf Bildstörungen zurückzuführen, die in Kapitel 3.2 noch behandelt werden. Ausserdem ist macht die Elektronik einen optisch unfertigen Eindruck, sie sieht sehr provisorisch aus. Dies ist allerdings zweitrangig, da sie die gestellten Aufgaben tadellos erfüllt. Somit hat das Aussehen keinen Einfluss auf die Funktion und ist also völlig in Ordnung. Allerdings ist für den praktischen Gebrauch zu beachten, dass ein Gehäuse in der selben Form wie ein Originalmünzprüfer die Handhabung wesentlich vereinfachen würde. So liessen sich auch die Belastungen auf die Kabel reduzieren, da sie fest eingebaut werden könnten. Die Zuverlässigkeit im Dauerbetrieb liesse sich so steigern, insbesondere gegenüber der aktuellen, lockeren Befestigung in der Kassentür des Kastens.

Das Sekundärziel, das Sammeln von Erfahrungen im Bereich der Bilderkennung wurde ebenfalls erreicht. Wie das folgende Kapitel noch zeigen wird, konnten einige Probleme, egal ob vorhersehbar oder nicht, gelöst werden.

Zusammenfassend lässt sich also sagen, dass die Arbeit ein Erfolg ist. Das Produkt wurde erfolgreich hergestellt und lässt sich erfolgreich einsetzen. Ws gibt auch noch Probleme, die nicht gelöst worden sind. Somit hat diese Arbeit auch neue Fragen aufgeworfen.

3.2 Probleme

Sicherung

Ein grosses, unvorhergesehenes Problem wurde bereits in Kapitel 2.3.2 erwähnt. Durch einen Programmierfehler ist eine Sicherung im Kasten durchgebrannt. Die Sicherung auszutauschen wäre an sich kein Problem gewesen, wenn eine entsprechende Ersatzsicherung vorhanden gewesen wäre. Bei der durchgebrannten Sicherung handelte es sich nämlich um eine 7 Ampere Sicherung im amerikanischen Format also 32 auf 6.3 Millimeter.

Es stellte sich heraus, dass es nicht ganz trivial ist, in der Schweiz rasch einen Ersatz dafür zu finden. Die Elektronikgeschäfte in der Umgebung führten diesen Sicherungstyp nicht und die Elektronikversender hatten Lieferprobleme. Ausser Warten blieb mir also nichts anderes übrig. Die Ratschläge, die Sicherung mit einem Draht einfach kurzzuschliessen wurden ignoriert, da weder bekannt war, ob der Kasten Schaden genommen hatte, noch war sicher ausgeschlossen, dass das nicht einmal passieren könnte. Somit wäre dies mit einem grossen Risiko einhergegangen.

Kamera

Ein Problemfall in vielerlei Hinsicht war die Kamera. Einige Probleme liessen sich recht einfach beheben, andere wiederum gar nicht. Das starke Bildrauschen konnte durch mehr Licht in den Griff gekriegt werden. Die Kommunikation mit dem Computer und der OpenCV Bibliothek wurde durch zusätzliche Software ermöglicht (Siehe Kapitel 2.3.2). Bei der Ausrichtung der Kamera hilft eine Art Fadenkreuz, dass auf dem Computerbildschirm über das Livebild gelegt wird.

Ein Problem, welches nicht gelöst werden konnte, war die Tatsache, dass der Camcorder nach 5 Minuten in den Standby-Modus ging. Dies war nicht weiter tragisch, da der Ball praktisch nie so lange am Stück im Spiel gehalten werden konnte. Bei der Entwicklung der Software und beim Testen störte dies zwar, aber die Lösung dieses Problems hatte

keine grosse Priorität.

Ein grösseres Problem, welches sich erst mit der Verbesserung der Software zeigte, liess sich auch nicht mehr beheben. Es ist festzustellen, dass der Kasten jeweils zu Beginn einer Ausführung der Software besser läuft und die Anzahl der Fehlauslösungen mit der Zeit steigt. Die Ursache dafür ist der Kasten selbst. Durch die Auslösung der Flipperfinger gerät er in Schwingung. Diese Schwingung überträgt sich auf den Boden und so über das Stativ auf die Kamera. Da sich das Problem erst spät zeigte und nicht zufriedenstellend durch eine Verschiebung der Kamera lösen liess, blieb es ebenfalls ungelöst.

Ein möglicher Lösungsansatz dafür könnte eine feste Verbindung der Kamera mit dem Kasten sein, sodass das System als eine Einheit schwingt und sich die Kamera relativ zum Kasten so nicht mehr verschiebt. Eine ander Möglichkeit wäre, die Kamera vom Boden zu entkoppeln, beispielsweise durch eine Montage an der Decke.

Schlusswort

Diese Arbeit stellte mich vor ganz neue Herausforderungen. Ich hatte mich für ein Gebiet entschieden, das Neuland für mich war. Ich hatte zu Beginn zwar bereits Programmierkenntnisse, jedoch überhaupt nicht in C++, der Sprache, die ich wegen der OpenCV Bibliothek wählen musste. Ausserdem hatte ich noch überhaupt keine Erfahrungen mit Multiprocessing und Errorhandling. Bildverarbeitung war mir zwar ein Begriff aus diversen Bildverarbeitungsprogrammen, doch dies aus der Sicht eine Computers zu betreiben war auch neu für mich.

Hinzu kam, dass ich mir persönlich sehr viel höhere Ziele gesetzt hatte. Die musste ich allerdings aufgeben. Erst im Verlauf dieser Arbeit ist mir klar geworden, wie mächtig diese Werkzeuge sind, speziell OpenCV. Es ist als ob man das Land der unbegrenzten Möglichkeiten betreten würde. Werkzeuge müssen allerdings beherrscht werden. Im Verlaufe der Arbeit lernte ich die Grundlagen kennen, was vielleicht nicht genug war. Aus der heutigen Sicht bin ich geneigt zu sagen, dass ich mich zuvor ein Jahr nur mit C++ und OpenCV beschäftigen hätte sollen, ohne überhaupt in die Richtung Echtzeitbilderkennung zu bewegen. Ich bin überzeugt, dass der Kasten dann besser spielen würde.

Diese Spekulationen sollen aber nicht eine Resignation meinerseits andeuten. Durch diese Arbeit habe ich eher Ansporn gefunden, mich noch deutlich tiefer mit diesen Themen zu befassen, da sie sehr interessant sind und auch einen praktischen Nutzen haben. Somit ist es für mich durchaus vorstellbar, dass der Abschluss dieser Arbeit nicht das Ende des Projekts bedeutet, ganz nach dem Grundsatz: To be continued. . . ?

Abbildungsverzeichnis

Literaturverzeichnis

[GID(2004)] Executing programs with c(linux), 2004. URL http://www.gidforums.com/t-3369.html.

[dvg(2010)] Linux digital video, 2010. URL http://www.kinodv.org/.

[eag(2010)] Cadsoft online: Eagle layout editor, December 2010. URL http://www.cadsoft.de/.

[hou(2010)] *Feature Detection - opencv v2.1 documentation*, 2010. URL http://opencv.willowgarage.com/documentation/cpp/imgproc_feature_detection.html#HoughCircles.

[iee(2010)] 1394 trade association: Consumer's faqs, Dezember 2010. URL http://www.1394ta.org/consumers/FAQ.html.

[ipd(2010)] The internet pinbal machine database, 2010. URL http://ipdb.org/.

[nam()] *mkfifo(3) - Linux man page*. URL http://linux.die.net/man/3/mkfifo.

[ope(2010)] Open source computer vision, 2010. URL http://opencv.willowgarage.com/wiki/.

[rel(2004)] *DIL Miniaturrelais FRT5*, 2004. URL http://www.reichelt.de/?;ACTION=7;LA=28;OPEN=0;INDEX=0;FILENAME=C300%252Ffrt5.pdf;SID=15mC32p6wQAQ8AAGPLZfk072b7a64c7cde30876aa85d13ea3f3a0,http://www.reichelt.de/?;ACTION=28;LA=3;ARTICLE=79366;GROUPID=3292;GROUP=C32;SID=15mC32p6wQAQ8AAGPLZfk072b7a64c7cde30876aa85d13ea3f3a0.

[sig(1993)] *The GNU C Library - Signal Handling*, 1993. URL http://www.cs.utah.edu/dept/old/texinfo/glibc-manual-0.02/library_21.html.

[Agam(2006)] Gady Agam. Introduction to programming with opencv. Januar 2006. URL http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html.

[Atmel(2010)] Atmel. *ATmega8(L) Datasheet*, revision x, updated 6/10 edition, Juni 2010. URL http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf.

[Dio(2002)] *Datasheet Silicon Rectifiers*. Diotec Semiconductor, 2002. URL http://pdf1.alldatasheet.com/datasheet-pdf/view/81689/DIOTEC/1N4001.html.

[Fut(2010)] *USB RS232 Cables*. Future Technology Devices International Ltd., Dezember 2010. URL http://www.ftdichip.com/Products/Cables/USBRS232.htm.

[Koller(2010)] Johannes Weinbuch; Zeno Koller. Datenverarbeitung. *Kantonsschule Wohlen*, 2010.

[Max(2010)] *Max232 Datasheet*. Maxim Integrated Products, Juli 2010. URL http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf.

[Phi(1999)] *Data Sheet BC549; BC550 NPN general purpose transistors*. Philips Semiconductors, 1999. URL http://www.reichelt.de/?;ACTION=7;LA=28;OPEN=0;INDEX=0;FILENAME=B400%252FBC549_550_3.pdf;SID=15mC32p6wQAQ8AAGPLZfk072b7a64c7cde30876aa85d13ea3f3a0,http://www.reichelt.de/?;ACTION=28;LA=3;ARTICLE=5012;GROUPID=2881;GROUP=A121;SID=15mC32p6wQAQ8AAGPLZfk072b7a64c7cde30876aa85d13ea3f3a0.

[Schmidt(1998)] Douglas C. Schmidt. C++ report article. April 1998. URL http://www.cs.wustl.edu/~schmidt/signal-patterns.html.

[Wil(1994)] *The Flintstones Operations Manual*. Williams Electronics Games, Inc., 1994.

Anhang A

Eigenständigkeitserklärung

Mit meiner Unterschrift bestätige ich, diese Arbeit selbständig verfasst und nur die im Literaturverzeichnis aufgeführten Quellen verwendet zu haben.

Unterschrift: ..... Ort, Datum: .....

Anhang B

Veröffentlichung

Der Code dieser Arbeit ist auf `https://github.com/jogi91/ballrecognition` zu finden. Dort wird nach der Abgabe auch gegebenenfalls weitere Teile der Arbeit hochgeladen, um sie mit der Öffentlichkeit zu teilen und so das gewonnene Wissen zu verbreiten.