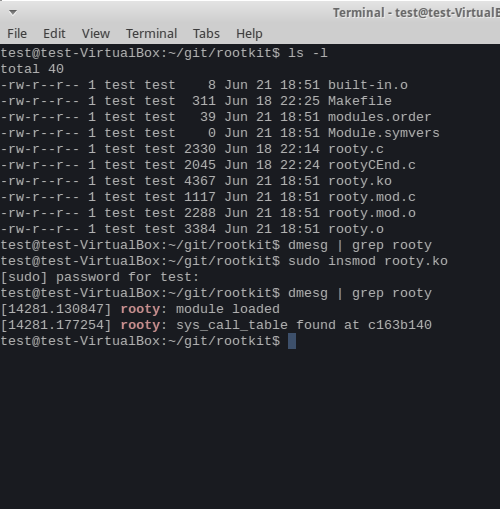
**ReaFID**

**Ein RFID Reaktionsspiel auf Basis eines Arduinos**



**Semesterarbeit Désirée Sacher, ZHAW**

6. /7. Semester 29. Juni 2014

**Danksagung**

Ich möchte mich in erster Linie bei meinem Lebenspartner Patrick bedanken, für seine Geduld und Unterstützung in jeder Lage. Er hat mir nicht nur den Rücken gestärkt und mich im Haushalt entlastet, wir konnten auch Design Entscheide diskutieren und argumentieren, was ich immer als spannend empfinde.

Auch danken möchte ich Alain Lafon für das entgegen gebrachte Vertrauen in der Anfangsphase bis zur Finalisierung der Aufgabenstellung und seine Ratschläge, in Bezug auf Entwicklungsumgebung und Toolkits für RFID/Arduino und Python.

Valentin Zahnd, Informatik Student in Winterthur, möchte ich danken für das teilen seiner Erfahrungen mit Python und Hinweisen, welche die Suche nach Problemlösungen um einiges beschleunigt haben.

Zu guter letzt danke ich allen Entwicklern, welche vor mir schon auf die gleichen Probleme gestossen sind und sie im Internet entsprechend dokumentiert haben. Ihr habt mein Vorankommen massgeblich unterstützt.

**Inhaltsverzeichnis**

[Verzeichnisse 5](#_Toc397784552)

[Abbildungsverzeichnis 5](#_Toc397784553)

[Tabellenverzeichnis 5](#_Toc397784554)

[Zusammenfassung 6](#_Toc397784555)

[1. Einleitung 7](#_Toc397784556)

[2. Umsetzung der Seminararbeit 8](#_Toc397784557)

[2.1. Aufgabenstellung 8](#_Toc397784558)

[2.2. Arbeitsplanung 8](#_Toc397784559)

[2.3. Seminarumsetzung 9](#_Toc397784560)

[3. Über die eingesetzten Technologien 11](#_Toc397784561)

[3.1. RFID 11](#_Toc397784562)

[3.1.1. RFID Technologie und Einsatzgebiete 11](#_Toc397784563)

[3.1.2. Mifare Standard 12](#_Toc397784564)

[3.2. Arduino 13](#_Toc397784565)

[3.3. Python 14](#_Toc397784566)

[4. Eigener Beitrag: ReaFID – Reaktionsspiel auf Basis von RFID und Arduino 15](#_Toc397784567)

[4.1. Einleitung 15](#_Toc397784568)

[4.2. Erfahrungsbericht 15](#_Toc397784569)

[4.2.1. Ausgangslage 15](#_Toc397784570)

[4.2.2. Serielle Verbindung zum Arduino 15](#_Toc397784571)

[4.2.2.1. Implementation Arduino 15](#_Toc397784572)

[4.2.2.2. Implementation Python 16](#_Toc397784573)

[4.2.3. Entwickeln der Spiellogik 16](#_Toc397784574)

[4.2.3.1. Managen der Karten 16](#_Toc397784575)

[4.2.3.2. Zeitmessfunktion 17](#_Toc397784576)

[4.2.3.3. Wahl der Spiellogik 17](#_Toc397784577)

[4.2.4. Entwickeln des GUI 19](#_Toc397784578)

[4.2.4.1. Analyse 19](#_Toc397784579)

[4.2.4.2. Struktur 19](#_Toc397784580)

[4.3. Fazit 20](#_Toc397784581)

[4.3.1. Analyse 20](#_Toc397784582)

[4.3.1.1. GUI Ablauf 20](#_Toc397784583)

[4.3.2. Eingesetzte Pattern 24](#_Toc397784584)

[4.3.2.1. MVC – Model View Control 24](#_Toc397784585)

[4.3.2.2. Singleton Pattern 25](#_Toc397784586)

[4.3.2.3. Factory Pattern 25](#_Toc397784587)

[4.3.2.4. Strategy Pattern 25](#_Toc397784588)

[4.4. Ausblick 25](#_Toc397784589)

[5. Diskussion des eignen Beitrags 27](#_Toc397784590)

[5.1. Studieren der RFID Technologie und Bauen eines Reader mittels eines Arduinos und einem RFID Shield 27](#_Toc397784591)

[5.2. Dokumentieren von Herausforderungen beim Bauen eines RFID Readers 27](#_Toc397784592)

[5.3. Machbarkeitsanalyse zum Auslesen unserer neuen Campus Card 28](#_Toc397784593)

[5.4. Spiel in Python und mit einem Arduino 30](#_Toc397784594)

[6. Literaturverzeichnis 31](#_Toc397784595)

[7. Anhang 32](#_Toc397784596)

[7.1. Glossar 32](#_Toc397784597)

# Verzeichnisse

## Abbildungsverzeichnis

[Abbildung 1: Aufgabenstellung 8](#_Toc397784528)

[Abbildung 2: Übersicht RFID Typen, aus dem RFID Handbuch - Finkenzeller, Abbildung 2.18 11](#_Toc397784529)

[Abbildung 3: RFID Kommunikationsprotokoll nach ISO 14443 12](#_Toc397784530)

[Abbildung 4: Arduino UNO Version 3 13](#_Toc397784531)

[Abbildung 5: Arduino UNO mit einem XBee RFID Shield 13](#_Toc397784532)

[Abbildung 6: UML RFIDReaderClass 16](#_Toc397784533)

[Abbildung 7: UML Karten Factory 17](#_Toc397784534)

[Abbildung 8: UML CardManager 17](#_Toc397784535)

[Abbildung 9: UML timeMeasureClass 17](#_Toc397784536)

[Abbildung 10: UML GamePlayManager 18](#_Toc397784537)

[Abbildung 11: UML GamePlayStrategy 18](#_Toc397784538)

[Abbildung 12: UML MainGUI 19](#_Toc397784539)

[Abbildung 13: UML Dialog GUIs 20](#_Toc397784540)

[Abbildung 14: Flow Chart Bedienung Spiel Start 21](#_Toc397784541)

[Abbildung 15: Flow Chart Bedienung Kartenkonfiguration 22](#_Toc397784542)

[Abbildung 16: Flow Chart Bedienung Konfiguration Serielle Verbindung 23](#_Toc397784543)

[Abbildung 17: Flow Chart Bedienung Spielmoduswahl 24](#_Toc397784544)

[Abbildung 18: Kartentypen im Angebot von InterCard 29](#_Toc397784545)

## Tabellenverzeichnis

[Tabelle 1: Phase 1 des Projekts 8](#_Toc397784546)

[Tabelle 2: Phase 2 des Projekts 9](#_Toc397784547)

[Tabelle 3: Projekttermine 10](#_Toc397784548)

[Tabelle 4: Weiterentwicklungsmöglichkeiten 25](#_Toc397784549)

[Tabelle 5: Literaturverzeichnis 31](#_Toc397784550)

[Tabelle 6: Glossar 32](#_Toc397784551)

# Zusammenfassung

Das ursprüngliche Ziel dieser Arbeit war das Einblicke zu erhalten, wie gut unsere Studentenausweise abgesichert sind. Durch administrative Einschränkungen wurde in der Anfangsphase des Projekts das Ziel verändert und stattdessen ein Reaktionsspiel mit Hilfe eines RFID Readers, einem Arduino und der Programmiersprache Python implementiert. Alle drei Komponenten waren für den Studierenden neue Technologien für welche Interesse bestand, sich einzuarbeiten.

Realisiert wurde ein Spiel, bei welchem mit Hilfe des RFID Sensors die Reaktionsgeschwindigkeit getestet wird. Für die Programmierung wurden diverse Design Patterns verwendet und implementiert.

Diese Arbeit enthält Erläuterungen zu den eingesetzten Technologien, welche dem technischen Verständnis dienen können. Herausforderungen und Schwiergkeiten sind dokumentiert, welche meist technischer Natur waren und mit Hilfe von Suchmaschinen und vertiefter Einarbeitung gelöst werden konnten.

Design Entscheide sind in dieser Dokumentation ebenso festgehalten, da schon von Anfang an die mögliche Erweiterungsfähigkeit berücksichtigt und diese entsprechend priorisiert wurden.

Zum Abschluss wird nochmals auf die konkreten Aufgabenstellungen der Seminararbeit eingegangen, wobei nach dem DRY Prinzip zum Teil auf die entsprechenden Kapitel verwiesen wurde.

# Einleitung

Diese Arbeit soll als Möglichkeit dienen, sich mit der RFID Technologie auseinanderzusetzen. Da die RFID Technologie, speziell der in Europa verbreitete Mifare Standard, in den letzten Jahren häufig an Hackerkonferenzen thematisiert wurden, sind dessen Schwachstellen in der Impementation ausreichend bekannt.

Ursprünglich war geplant, die von InterCard hergestellten Studentenausweise der ZHAW genauer zu analyiseren, allerdings wurde diesem Vorhaben von den Sicherheitsbeauftragen der ZHAW einige Hürden in Form von NDAs in den Weg gestellt, welche eine sinnvolle Auseinandersetzung inklusive der Dokumentation der Befunde erschwert hätten.

Stattdessen wurde auf Basis von einem Arduino und einem RFID Shield ein Spiel entwickelt, welches die Technologie kreativ nutzt. Der ursprüngliche Fokus von „Hacken“ und „Reverse Engineeren“ der Technologie wurde verlegt auf das Entwerfen und Programmieren eines Spiels in einer bis jetzt dem Autor unbekannten Sprache. Da die RFID Technologie in beiden Projektzielen verwendet wird, kann trotzdem das Wissen über die Technologie angeeignet und entwickelt werden.

Ziel dieser Arbeit ist entsprechend ein Spiel auf Basis der RFID Technologie mit Hilfe eines Arduinos. Zur Programmierung des Spiels wurde Python verwendet. Die Spiellogik wurde selbst entwickelt.

# Umsetzung der Seminararbeit

## Aufgabenstellung

Die Aufgabenstellung wurde im EBS wie folgt definiert:

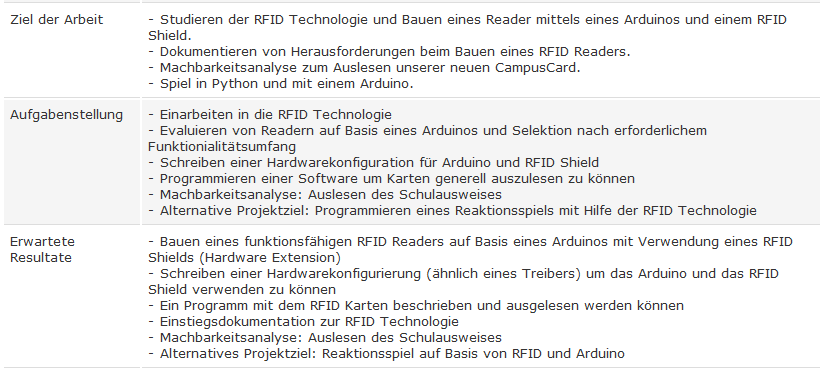


Abbildung : Aufgabenstellung

## Arbeitsplanung

In der ersten Phase wurden die Arbeiten in Stories aufgeteilt und der Aufwand wurde geschätzt. In dieser Phase konnte auch geklärt werden, dass das ursprüngliche Ziel (Auslesen eines Schulausweises) aus rechtlichen Gründen nicht sinnvoll als Semesterarbeit umsetzbar ist. Zum Design Review wurde entsprechend entschieden, dass stattdessen das Alternative Projektziel „Programmieren eines Reaktionsspiels mit Hilfe der RFID Technologie“ verfolgt wird. Aus diesem Grund fand zu diesem Zeitpunkt eine neue Arbeitsplanung über die restliche Zeit hinweg statt, welche hier in der Phase 2 festgehalten ist. Bis zu Projektende wurden 3 Sprints geplant und die Arbeitsschritte auf diese verteilt.

Die Aufwände wurden dabei jeweils in 2-Stundenblöcke (analog zur Pomodoro-Technik) unterteilt und über die zur Verfügung stehenden Wochen der Semesterarbeit verteilt. Der Fortschritt wurde in Jira festgehalten.

Tabelle : Phase 1 des Projekts

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint** | **Jira Bezeichnung** | **Beschreibung** | **Aufwand in Stunden** |
| 1 | RFID-1 | Setup Projekt | 3 |
| 1 | RFID-2 | Learn Python (Code Academy) | 8 |
| 1 | RFID-3 | RFID einlesen Technologie | 8 |
| 2 | RFID-4 | Unterschiede der Arduino RFID Reader ausfindig machen | 6 |
| 2 | RFID-5 | Bestimmen genauer Typ Karte | 4 |
| 2 | RFID-6 | Erstellen des Auslesenscript & auslesen | 12 |

Tabelle : Phase 2 des Projekts

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint** | **Jira Bezeichnung** | **Beschreibung** | **Aufwand in Stunden** |
| 3 | RFID-7 | ReaFID-Funktion – konfigurieren von Karten | 8 |
| 3 | RFID-8 | ReaFID Funktion – lesen von Karten | 8 |
| 3 | RFID-9 | ReaFID Spiel Funktion - Zeitstoppen | 5 |
| 3 | RFID-12 | ReaFID Dokumentation | 10 |
| 3 | RFID-20 | ReaFID Project Management/Admine 1 | 1 |
| 4 | RFID-10 | ReaFID Spiel GUI – Spieloberfläche Hauptfenster | 15 |
| 4 | RFID-11 | ReaFID Spiel GUI – Abbrechen/Stoppen Spiel | 5 |
| 4 | RFID-13 | ReaFID Dokumentation 2 | 10 |
| 4 | RFID-21 | ReaFID Project Management/Admin 2 | 2 |
| 5 | RFID-14 | ReaFID Dokumentation 3 | 20 |
| 5 | RFID-22 | ReaFID Project Management/Admin 1 | 1 |
| Backlog | RFID-15 | ReaFID GUI – Topscorer Liste | 8 |
| Backlog | RFID-16 | ReaFID GUI – Spieloberfläche Kartekonfiguration | 12 |
| Backlog | RFID-17 | ReaFID GUI – Anzeige Auslesen Karte | 6 |
| Backlog | RFID-18 | ReaFID GUI – Menüpunkt Select Modus | 4 |
| Backlog | RFID-19 | ReaFID Spiel Advanced Mode erstellen | 20 |

## Seminarumsetzung

Für das Projekt galten folgende Termine:

Tabelle : Projekttermine

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Termin** | **Beschreibung** |
| **1** | 24.03.14 | Kick Off |
| **2** | 29.04.14 | Präsentationstermin Security BSides London 2014 |
| **2** | 11.07.14 | Design Review |
| **3** | 12.09.14 | Abgabetermin |
| **4** | 05.11.14 | Präsentationstermin |
| **5** | 17.02.14 – 09.03.14 | Sprint 1 |
| **6** | 12.03.14 – 30.03.14 | Sprint 2 |
| **7** | 14.07.14 – 03.08.14 | Sprint 3 |
| **8** | 04.08.14 – 24.08.14 | Sprint 4 |
| **9** | 25.08.14 – 12.09.14 | Sprint 5 |

Zwischen Sprint 2 und 3 wurden nicht nur die rechtlichen Abklärungen getätigt, sondern auch ein 15 Minuten Vortrag für die BSides London („Rookie Track“) vorbereitet. Da diese Arbeiten nur die Erkenntnisse aus den technischen Abklärungen zu RFID und Arduino verwendeten und aufbereiteten, und nicht direkt für die Semesterarbeit waren, wurden diese nicht in Jira festgehalten und sind entsprechend nicht in der Projektplanung berücksicht. Die Informationen aus der Präsentation wurden allerdings in dieser Semesterarbeit miteinbezogen und beeinflussten entsprechend die Story „Dokumentation“ positiv.

# Über die eingesetzten Technologien

## RFID

### RFID Technologie und Einsatzgebiete

RFID steht für „radio-frequency identification“, was einer „Identifizierung mit Hilfe elektromagnetischer Wellen“ entspricht. Die Imlpementationen basieren dabei auf der „Low Frequency“ (9 kHz bis 135 kHz), „High Frequency“ (6,78 MHz, 13,56 MHz, 27.125 MHz, 40,680 MHz) oder dem „Ultra High Frequency“ (Europa: 865-869 MHz, USA/Asien: 950 MHz). Während frühere Implementationen (zum Beispiel für Zutrittssysteme von Gebäuden, Hotelzimmern, Verfolgen von Gegenständen) mit Low Frequency durchgeführt wurden, hat in den letzten Jahren der High Frequency einiges an Verbreitung gewonnen.

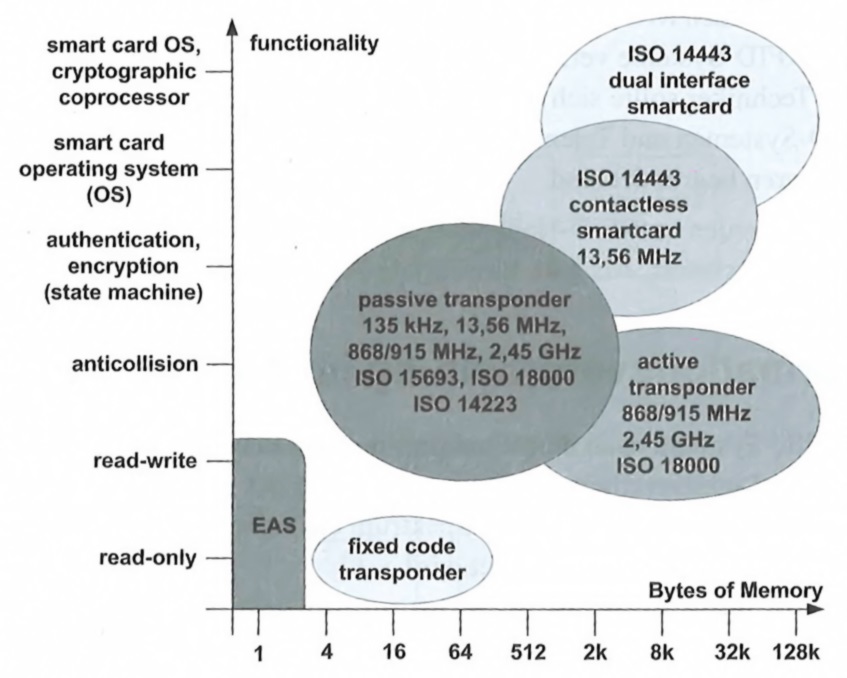


Abbildung : Übersicht RFID Typen, aus dem RFID Handbuch - Finkenzeller, Abbildung 2.18

In den meisten Systemen wird ein aktives Lesegerät und ein passiver Tag verwendet. Sobald der Tag in die Nähe des Lesegeräts gelangt, werden elektromagnetische Wellen erzeugt. Das Protokoll der Übertragung wird in diversen Standards festgelegt, welche inzwischen zum Teil ISO genormt sind. Hersteller verwenden allerdings aus „Sicherheitsgründen“, wobei es sich dabei mehr um „Security by Obscurity“ handelt, oft eigene Implementationsvarianten.

Durch eine zielgerichtete Recherche betreffend des Studentenausweises konnte einfach festgestellt werden, dass es sich bei dem bei uns verwendeten System um ein nach ISO 14443 auf der Frequenz 13,56 MHz festgelegten Standard handelt, wobei sich die ZHAW für die Implementation ohne Chip, also dem „Mifare Classik 1k“ Standard entschieden hatte.

Implementationen mit Chip sind einiges Leistungsfähiger als die ohne, so können zum Beispiel RFID Kreditkarten mit Chip für jede Transaktion einen neuen CVV Code generieren. Dies ist einer der Hauptgründe, warum das RFID Kreditkarten System mit „Chip und Pin“, so wie es in Europa verbreitet ist, einiges sicherer ist als das in den USA verbreitete ohne Chip.

### Mifare Standard

Die Tags welche im Mifare Standard verwendet werden, können in mehreren Formen erhalten werden. Für die Studentenausweise wird zum Beispiel die Form einer Karte verwendet. Die Norm für 1k Tags besagt, dass ein Tag 16 Sektoren hat, wobei jeder Sektor vier Blöcke enthält. Jeweils der letzte Block jedes Sektors enthält den „Access Key“, welcher für verschlüsselte Sektoren nötig ist. Pro Sektor kann es maximal zwei verschiedene Access Keys geben, wobei diese unterschiedliche Zugriffsberechtigungen besitzen können. Der erste Block (Block 0) enthält die UID der Karte und ist normalerweise schreibgeschützt. Karten mit veränderbaren UIDs sind relativ einfach über das Internet erhältlich. Falls ein Sektor verschlüsselt ist, wurde dafür die Crypto1 Verschlüsselung verwendet. Informationen zu dieser Verschlüsselung wurden schon vor Jahren am CCC präsentiert inklusive der Information, wie die verschlüsselten Inhalte tatsächlich ausgelesen werden können.

Die Authentisierung, welche per Sektor möglich ist, findet nach folgendem Schema statt:

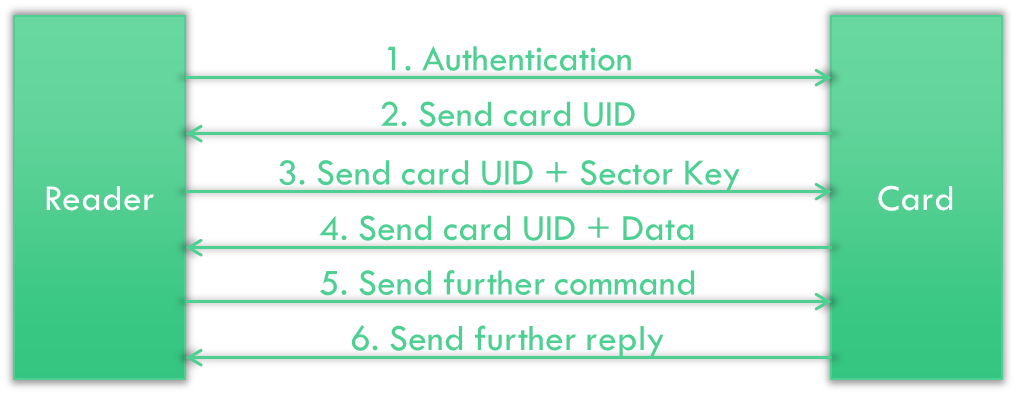


Abbildung : RFID Kommunikationsprotokoll nach ISO 14443

Bereits in den Protokoll Informationen wird der verwendete Sektor mitangegeben. Entsprechend ist dies eine Implementation welche auf dem Reader stattfindet.

Da es sich um eine drahtlose Kommunikation handelt, ist ein Angriff über Mithören der Übertragung und danach separate Abfrage möglich. Zudem können Karten geklont werden. Da die Verschlüsselung in Sicherheitskreisen als unsicher gilt, sind die darauf abge legten Informationen schlecht geschützt. Häufig wird aus diesem Grund bei Implementationen auf das abspeichern auf der Karte verzichtet und stattdessen jegliche Informationen auf dem „Backend System“ gespeichert. Bei der Prüfung der Karte wird lediglich die UID abgerufen und geprüft, ob diese im System konfiguriert ist. Jegliche Kontrollmechnaihanismen müssen entsprechend auf dem Backend System implementiert sein.

## Arduino

Ein Arduino ist eine Open Source Computerplattform, welche auf einem simplen Mikrokontroller (Atmel ATMEGA8) basiert und eine dazugehörige Entwicklungsumgebung mitbringt. Der Mikrokontroller kann Programme in Form einer daueraktiven Statemachine verarbeiten und wird deshalb oft für die Zusammenarbeit mit Sensoren jeglicher Art verwendet.

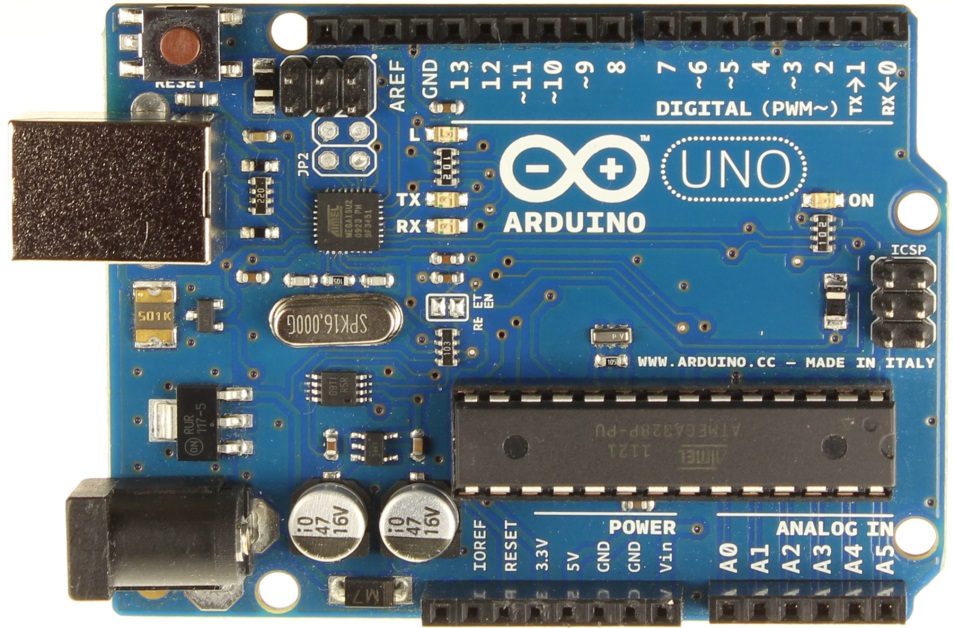


Abbildung : Arduino UNO Version 3

Um die Verknüpfung mit diesen Sensoren (welche generell als Input Systeme verstanden werden können) zu optimieren und Installationen weniger anfällig zu machen, wurden einige „Boards“ oder „Shields“ entwickelt, welche direkt auf das Arduino aufgesetzt werden können. Sie beanspruchen gewöhnlicherweise mehr Schnittstellen, machen die Installationen aber auch ein wenig robuster.

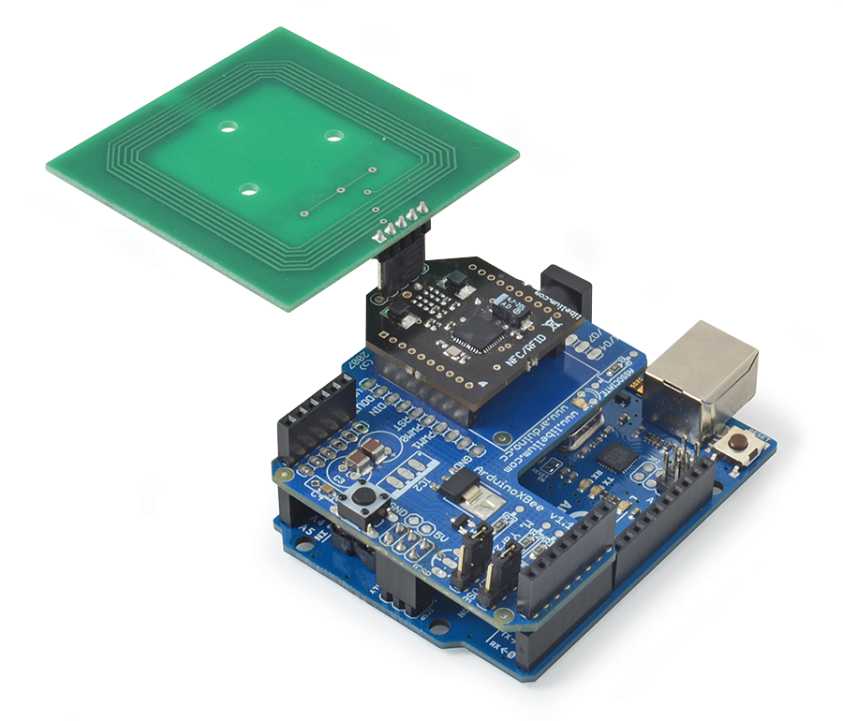


Abbildung : Arduino UNO mit einem XBee RFID Shield

Es gibt heute eine Vielzahl an Shields und Sensoren, welche über normale WLAN Interfaces, über RFID/NFC bis zu Feuchtigkeitsmesser gehen. Falls kein Shield vorhanden ist, können die Sensoren gewöhnlicherweise direkt oder mit Hilfe eines Breadboards angehängt und angesprochen werden. Starter Kits für Arduinos liefern nicht nur eine vielzahl solcher Sensoren mit sondern beinhalten auch detaillierte Anleitungen, wobei diese auch kostenlos Digital auffindbar sind.

Der Output findet normalerweise über Dioden, dem Computerbildschirm (über das serielle Interface) oder einem direkt angehängten Bildschirm statt.

## Python

Python ist eine relativ moderne Sprache (Erscheinungsjahr 1991) zu welcher Entwurfsphilosophie stark die Programmlesbarkeit und Einfachheit der Syntax gehört. In den letzten Jahren fand sie starke Verbreitung durch Online Lernprogramme wie von Codeacademy und den omnipräsenten Einsatz bei Google sowie anderen Open Source Projekten.

Die Sprache unterstützt mehrere Programmierparadigmen, unter anderem auch den Objekt orientierten und den strukturierten Ansatz. Durch diese Vielfältigkeit, kann ein zur Aufgabe passendes Paradigma ausgewählt werden, ohne das neue Syntax erlernt werden müssen.

Zusätzlich ist es hilfreich, dass für Python eine grosse Auswahl an Bibliotheken und Entwicklungsumgebungen existieren, so dass diese jeweils den Bedürfnissen angepasst werden können.

Für dieses Projekt wurde mit der neuester Version von Python (Version 3.4) entwickelt. Durch den grossen Versionsunterschied zwischen Version 2 und 3 sind nicht mehr alle GUI Toolkits mit der aktuellen Version kompatibel. Dies musste entsprechend bei der weiteren Auswahl von Bibliotheken und Anleitungen berücksichtigt werden.

# Eigener Beitrag: ReaFID – Reaktionsspiel auf Basis von RFID und Arduino

## Das Spiel

Die Idee des Spieles ist es, Informationen zu einer Karte einzublenden, wobei daraufhin so schnell wie möglich die korrekte Karte aus den zur Verfügung stehenden Karten herausgesucht werden und an den RFID Sensor hingehalten werden muss. Das Spiel prüft die hingehaltene Karte und zählt die benötigte Zeit. Nach 10 Durchläufen wird die verwendete Zeit addiert und als Summe ausgegeben. Anschliessend kann ein neuer Durchgang gestartet werden.

Durch die Erweiterung auf eine Topscorer Liste kann versucht werden, frühere Zeiten oder Zeiten von Kollegen überboten zu werden.

Das Spiel kann nicht mit vorher nicht konfigurierten Karten umgehen. Jegliche im Spiel zu verwendende Karte muss vorhin definiert und als Objekt im Spiel angelegt werden. RFID Karten werden dazu nicht verändert, lediglich die UID wird ausgelesen.

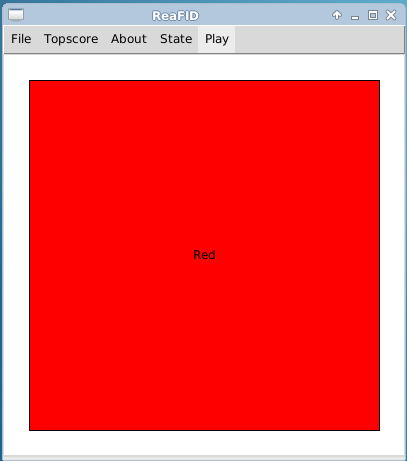


Abbildung : Spiel Hauptfenster nach Start mit zufällig gewählter Farbe

## Ausgangslage

Zu Beginn der Semesterarbeit hatte bereits ein initiales Kennenlernen betreffend Arduinos und RFID stattgefunden. Um allerdings die Technologien weiter kennen zu lernen, war ein Projekt nötig. Die Sprache Python wurde ausgewählt, da sie vielseitig einsetzbar ist und in Zukunft damit schnell Projektideen umgesetzt werden sollen.

Die Entwicklung wurde auf einer Arch Linux Installation durchgeführt, wobei als IDE Eclipse eingesetzt wurde. Zur Interpretation von Python 3.4 wurde das Plugin PyDev verwendet. Das Arduino wurde über die dazu gelieferte IDE programmiert, die Arduino Ausgaben wurden über den Serial Port ausgelesen. Die Kommunikation zwischen RFID Reader und Python findet entsprechend auch über einen Seriellen Anschluss statt.

Zum Thema RFID und Arduino sind im Internet bereits einige Anleitungen und Codebeispiele zu finden. Auf der Basis der Anleitung von Cooking Hacks wurden die Hardwarekomponenten bestellt. Die Beispiel Skripte konnten für das auslesen von leeren RFID Karten verwendet werden. Um den Zugriff für das Spiel allerdings zu optimieren, wurden die Skripte minim angepasst.

Da es sich bei dem Projekt um ein Reaktionsspiel handelt, ist der Faktor Zeit eine wichtige Komponente. Für das Spiel wurde entsprechend das auslesen optimiert und der Handshake wird nach auslesen der Karten UID beendet. Ein verändern der RFID Karten oder der darauf geschrieben Daten findet für unser Spiel nicht statt. Dies ermöglicht es, die Spielkarten beliebig zu erweitern und zum Beispiel mit verschiedenen Studentenausweisen das Spiel durchzuführen. Andere Kategorien wurden in Bezug auf Studentenkarten als eine erste Idee in der Implementation berücksichtigt.

## System Architektur

Um das Spiel zu spielen sind mehrere Komponenten nötig:



Um RFID Karten zu lesen wurde für dieses Beispiel ein Arduino mit einem entsprechenden Modul verwendet. Zudem wird die Anzeige der geforderten Karte sowie sämmtliche Benutzerinteraktion über einen Computer durchgeführt.

Das Arduino übernimmt die Steuerung des RFID Sensors, welcher mit Hilfe eines Communication Shields (welches in unserem Fall Produktabhängig ist und nicht für jeden RFID Sensor benötigt wird) angehängt ist. Zudem kommuniziert das Arduino die gelesenen UIDs über die serielle Verbindung an den Computer.

Auf dem Computer wird bei Start des Spiels der serielle Port geöffnet und während des Spiels auf gelesene Informationen gehorcht. Bei Erhalt eines Signals reagiert das Python auf vordefinierte Weise und lässt das Spiel eine entsprechende Wende oder Ablauf ausführen.

## GUI Architektur

Das GUI besteht aus insgesammt zur Zeit 4 Fenstern. Zum Spielstart wird das Hauptfenster „MainGUI“ geöffnet. Aus diesem Fenster können 3 Dialoge geöffnet werden, in welchem Konfigurationseinstellungen zum Spiel vorgenommen werden können.



Abbildung : GUI Hauptfenster Beschreibung

Die Dialogfenster sind jeweils simpel gestaltet und zeigen nur die nötigen Felder für ihre Anpassungen. Abbildung 8 ist Beispielweise das Fenster zur Konfiguration der Seriellen Verbindung.

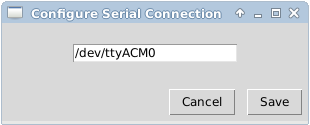


Abbildung : GUI Konfiguration Serielle Verbindung

## Software Architektur

### Pattern

In diesem Projekt kamen einige Patterns zum Zuge. Hier werden sie erläutert.

#### MVC – Model View Control

Als Grundstruktur wurde das MVC Pattern verwendet. MVC hilft durch Kapselung eine saubere Trennung von Anzeige und Hintergrundalgorithmen umzusetzen. Heute wird es allerdings oft nicht mehr in seiner Originalform verwendet da das Control Paket zu künstlichen Kapselungen führt welche keine Effizienzsteigerungen bringen. Aus Konsequenzgründen wurde allerdings eine Controller Klasse implementiert in welcher die Aufrufe aus den View und den Model Klassen zusammengeführt werden.

|  |
| --- |
| src/  | --- model/  | --- --- \_\_init\_\_.py  | --- --- cardFactory.py  | --- --- cardManager.py  | --- --- gameLogics.py  | --- --- gamePlayModes.py  | --- --- rFIDReader.py  | --- --- singleton.py  | --- --- timeMeasure.py  | --- view/  | --- --- \_\_init\_\_.py  | --- --- mainGUI.py  | --- --- subGUIs.py  | --- controller/  | --- --- \_\_init\_\_.py  | --- --- gameController.py  | --- --- starter.py |

Abbildung : MVC Python File Struktur

Durch das \_\_init\_\_.py File in jedem Ordner wird das Unterverzeichnis als Paket erkannt, über welches die Files danach kontrolliert importiert werden können ohne dass der Ausführungspfad von Python verändert werden muss.

#### Singleton Pattern

Das Singleton Pattern erzwingt, dass von einem Objekt nur eine Instanz existiert. Zur Kontrolle der Zugriffe wurde es für das Auslesen des Serial Interfaces, dem Karten Manager, dem Game Controller und dem Spiel Modus Manager verwendet.

Um die bestmöglichste Implementationsart zu finden wurde via Google recherchiert und schlussendlich auf Stakeoverflow eine entsprechende Lösung gefunden.

Für unsere Zwecke wurde im Paket „model“ eine Klasse Singleton erstellt, welche danach mit Hilfe der „metaclass“ Funktion[[1]](#footnote-1) implementiert wird. In dieser Klasse „Singleton“ wird das Verhalten der implementierenden Klassen definiert, was in unserem Fall ein ständiger Verweis auf die das selbe Objekt ist.



Abbildung : UML Singleton

#### Factory Pattern

Das Factory Pattern wird zum leicht modifizierten Erstellen von Objekten verwendet. Für dieses Reaktionsspiel wurde es verwendet um Kartenobjekte mit verschiedenen Typen zu implementieren. Diese unterschiedlichen Objekttypen können unterschiedliche Methoden enthalten. In dieser Implementation hat der Typ Color-Karte eine Variable color und einen entsprechenden getter und setter. Für den Typ Student-Karte wiederum kann der Name erfasst und mit der Methode getName und setName kontrolliert werden.



Abbildung 11: UML Karten Factory

Weitere Factorytypen können mit hinzufügen weiterer Klassen realisiert werden.

Hinweise zur Implementation eines Factory Pattern wurden auch über Google recherchiert[[2]](#footnote-2) und umgesetzt.

#### Strategy Pattern

Das Strategy Pattern wird verwendet um parallel mehrere Strategien zu implementieren. Es eignete sich optimal um die verschiedenen Spielvarianten zu implementieren. Es werden dabei quasi Kopien von Klassen implementiert, wobei die darin enthaltenen Methoden gleich benannt sind.



Abbildung : UML Strategy Pattern

Implementationsbeispiele wurden über Google recherchiert[[3]](#footnote-3) und analog dieser realisiert.

### Use Cases

## Implementierungsdetails

# old

## Einleitung

Um sich genügend mit der RFID Technologie, sowie Arduinos und Python auseinandersetzen zu können, wurde ein Spiel entwickelt, welches auf den gewählten Konzepten basiert. Die Spielidee wurde selber entwickelt, ist grundsätzlich aber für Kleinkinderspiele in ähnlicher Form Analog bekannt. Neu ist an der Idee, die Umsetzung über RFID durchzuführen. Zudem sind die Varation der Schwierigkeitsstufen in bekannten Implementationen nicht so auswählbar.

## Erfahrungsbericht

### Ausgangslage

### Serielle Verbindung zum Arduino

#### Implementation Arduino

Das Arduino erhält den Code über die Entwicklungsumgebung. Über eine IDE-interne „Hochladungsfunktion“ kann der erstellte Programcode auf das Arduino gespielt werden. Danach wird der programmierte Ablauf in einer ständigen Schlaufe ausgeführt. In unserer Schlaufe ist das das permanente Auslesen von UIDs der Karten, wobei diese entsprechend im Skript markiert werden damit sie anschliessend von Python ohne Probleme erkannt werden können.

Eine Herausforderung der Implementation war das übertragen von „Noise“, welches auch nach optimieren des Skripts nicht vollständig ausgeschlossen werden konnte.

#### Implementation Python

Auf Seite von Python wurde das Auslesen in der Python Klasse RFIDReaderClass realisiert. Dabei wird im Konstruktor der Zugriff auf das Serial Interface geöffnet und erst beim beenden des Spiels wieder entfernt. Um dabei keine parallelen Zugriffe zu erlauben, wurde diese Klasse als Singleton implementiert.

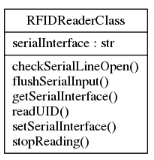


Abbildung : UML RFIDReaderClass

### Entwickeln der Spiellogik

#### Managen der Karten

Bei der Implementation wurden zuerst jegliche Hilfsklassen implementiert. Über ein Factory Pattern wurde so das generieren von Objekten für die RFID Karten abgelegt.

Der Sinn davon war, dass beim Erstellen ein anderer Typ von Karte erzeugt wird und danach jeweils mit Hilfe einer Typenprüfung die richtige Methode aufgerufen wird. So sollen einfach weitere Klassen für weitere Typen von Karten bei Bedarf hinzugefügt werden.

Die Karten selber werden dabei über den CardManager verwaltet. Nur diesem ist es möglich, Karten zu aktivieren, sie zu suchen, zu entfernen oder auszulesen, wie viele Karten überhaupt vorhanden sind.

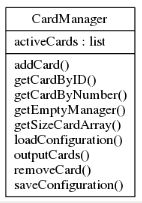


Abbildung : UML CardManager

#### Zeitmessfunktion

Das messen der Zeit wurde in der Klasse timeMeasureClass realisiert. Die separate Klasse ermöglicht eine einfache Kapselung der Funktion und wird nur bei Bedarf vom aktiven Spiel Modus aufgerufen.

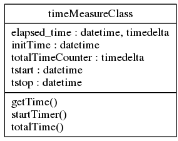


Abbildung : UML timeMeasureClass

#### Wahl der Spiellogik

Im GUI kann der Spielmodus ausgewählt werden, welche danach ausgeführt werden soll. Dieser Modus wird danach über ein Strategy Pattern aktiviert und geladen. In der ersten Phase wurde lediglich der „Easy Modus“ programmiert, welcher mit Farb Karten spielbar ist.

Zuerst wird über das GUI den Spiel Modus Manager (GamePlayManager) angesteuert und kontrolliert.

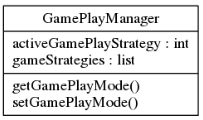


Abbildung : UML GamePlayManager

Im GamePlayManager wird die Strategie gespeichert und als Strategie registriert.

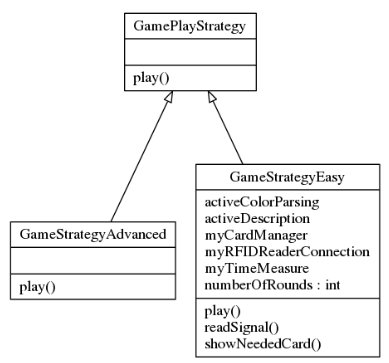


Abbildung : UML GamePlayStrategy

Die Spiellogik der „GameStrategyEasy“ ist, nach Zufall eine Karte aus dem Kartenmanager zu laden und deren Farbe darzustellen. Zeitgleich wird die Zeitmessung gestartet. Sobald die korrekte Karte an das Lesegerät gehalten wird, wird die Zeitmessung gestoppt und die nächste Karte geladen. Die Reaktionszeiten aller 10 Durchgänge (10 ist in einer separaten Variable konfiguriert, ein Ausbau um die Anzahl Runden zu verändern wäre möglich) werden zusammengezählt. Pro falsch hingehaltene Karte werden 5 Strafsekunden hinzugefügt.

Weitere Strategien sind über neue Klassen hinzufügbar. Ideen dafür sind zum Beispiel für weitere Farbkarten, die Anzeige in Schrift und Farbe unterschiedlich zu randomisieren, wobei dann nur eine Farbanzeige wirklich für die korrekte Wertung relevant ist und die andere verwirren soll. Eine weitere Möglichkeit für den Advanced Modus wäre, die Farbanzeige automatisch nach wenigen Sekunden (oder Hunderstelsekunden) zu ändern, so dass dies quasi einer „verpassten Gelegenheit“ gleich käme. Dabei könnten die korrekt hingehaltenen Karten gezählt werden.

### Entwickeln des GUI

#### Analyse

Aus Kompatiblitätsgründen wurde das GUI mit TkInter entwickelt. Eine zusätzliche Herausforderung war der Objektorienterte Ansatz, welcher nicht in allen Hilfestellungen und Anleitungen vorhanden war. Anleitungen dazu waren relativ schwierig zu finden da sie meistens nur sehr spezifisch für ein Ziel waren und es keine Anleitung gab, welches alles beschrieben hätte.

#### Struktur

Jedes Fenster wird über seine eigene Klasse angesprochen. Zugriffe auf die Model Klassen wurde über einen Controller realisiert. Insgesammt gibt es zur Zeit 1 Hauptfenster und 3 Dialogfenster, über welche Spielkonfigurationen vorgenommen werden können.

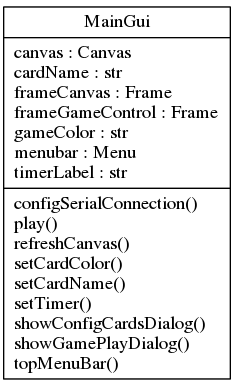


Abbildung : UML MainGUI

Die grösste Herausforderung des Hauptfensters war die automatische Aktualisierung sobald eine neue Karte geladen werden sollte. Es wurde vermieden, das Spiel in mehreren Threats zu realisieren um die Komplexität noch zu erhöhen. Aus diesem Grunde wo sinnvoll ein Singleton verwendet, die Aktualisierung der Anzeige erfolgt über ein „self.canvas.update\_idletasks()“ ausgelöst.

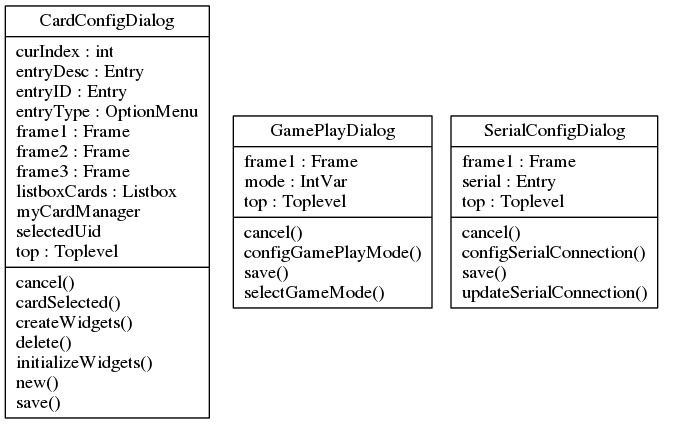


Abbildung : UML Dialog GUIs

Beim Kartenkonfigurationsmenu war die grösste Herausforderung eine Taktik zu entwickeln, wie Karten editiert werden können. Durch eine Folge von Selektieren der Karten und Löschen oder Neu hinzufügen einer Karte ist dies nun möglich.

## Fazit

### Analyse

#### GUI Ablauf

Um ein Spiel mit den Standardeinstellungen zu spielen kann im Hauptfenster der Startknopf in der Menübar gedrückt werden. Weitere Dialogfenster sind dabei nicht nötig.



Abbildung : Flow Chart Bedienung Spiel Start

Um die Karten Konfiguration einzusehen oder anzupassen kann über das „File“ Menu das Konfigurationsfenster geöffnet werden. Darin sind die standardmässig konfigurierten Karten einsehbar. Eine Mutation ist im selben Dialog möglich.



Abbildung : Flow Chart Bedienung Kartenkonfiguration

Um die Serielle Verbindung anzupassen, was in seltenen Fällen möglich sein muss damit die Verbindung zum RFID Reader funktioniert, kann über das „File“ Menü der entsprechende Dialog aufgerufen werden.



Abbildung : Flow Chart Bedienung Konfiguration Serielle Verbindung

Der Spielmodus kann über einen seperaten Konfigurationsdialog aufgerufen werden.



Abbildung : Flow Chart Bedienung Spielmoduswahl

### Eingesetzte Pattern

## Ausblick

Die während dem Projekt programmierte Basis ist ein guter Ausgangspunkt für weitere Implementationen. Es können nicht nur weitere Kartentypen hinzugefügt werden, weitere Spielmodus können genauso implementiert werden. Eine Liste von weiteren Optimierungsbeispielen ist hier zu finden:

Tabelle : Weiterentwicklungsmöglichkeiten

|  |  |  |
| --- | --- | --- |
| 1 | Advanced Mode | Für Farbkarten: „Verwirrungstaktik“ mit Farbeinfärbung anders als dargestellter Text. |
| 2 | Speed Modus | Für Farbkarten: Beschleunigte Anzeige, begrenzt auf Zeitwert wobei der Punktemodus verändert werden müsste |
| 3 | Topscore Liste | Es kann eine Topscorer Liste implementiert werden, welche die besten Resultate festhält und eventuell über eine Serialisierung speichert. |
| 4 | Anzeige ob Serielle Verbindung erfolgreich | Eine Anzeige im Hauptfenster zeigt den Status der seriellen Verbindung an. Falls die Verbindung verloren gehen würde könnte somit das Spiel unterbrochen werden. |
| 5 | Anzeige Zeit im Hauptfenster | Eine Anzeige im Hauptfenster der Zeit kann den Spieldruck erhöhen und das Spiel entsprechend spannender machen. |
| 6 | Countdown vor dem Spiel | Ein kurzer Countdown vor dem Spielstart kann die Vorbereitungsphase verbessern und würde den Spieleffekt erhöhen. |

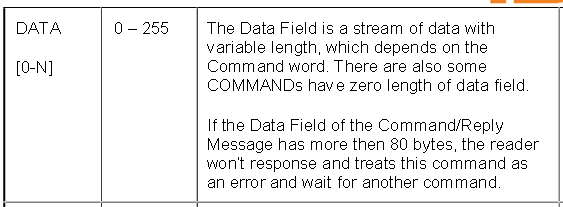
# Diskussion des eignen Beitrags

## Studieren der RFID Technologie und Bauen eines Reader mittels eines Arduinos und einem RFID Shield

Das Protokoll welches für RFID Transaktionen der Studentenkarte verwendet wird ist im ISO Standard 14443 definiert. Die Erkenntnisse zum Protokoll sind im Kapitel 3.1 festgehalten. Um einen RFID Reader selber zu bauen galt es zuerst herauszufinden, welche der erhältlichen Reader für den Einsatz geeignet wären. Ein wichtiger Faktor beim eigenen Zusammenbau war die gute Dokumentation, welche für den Xbee Reader vorhanden war. Zudem sollte ein Chipsatz auf dem Reader verbaut sein, welcher möglichst nicht nur die öffentlich bekannten Funktionalitäten unterstützt, sondern auch Fähigkeiten wie das passive mitlesen von Transaktionen ermöglichen soll. Nach einiger Recherche konnte festgestellt werden, dass der Chipsatz PN532 dies technisch unterstützt, entsprechende Arduino Code Implementationen aber nicht aufgeführt sind.

Durch diese Basis kann dieses Projekt nach Vollendung der Semesterarbeit beliebig fortgesetzt werden. Die Grenzen der Fähigkeiten des Sensors sind noch nicht erreicht.

## Dokumentieren von Herausforderungen beim Bauen eines RFID Readers

Die grössten Herausforderungen waren die versteckten Limitationen, welche gewisse Reader enthielten, zu finden. So hatte der ursprünglich ausgewählte Reader von Seeed Studios eine Datenleselimite von 80 bytes, weshalb er aus Zuverlässigkeitsgründen nicht weiter eingesetzt wurde 

Der jetzt verwendete Reader von Xbee wiederum hat eine Limitation, welche ein gleichzeitiges auslesen und schreiben über das Serial Interface am Arduino unterbindet. Als Konsequenz kann der ausgeführte Code auf dem Arduino nicht von Eingaben über das Serial Interface beeinflusst werden. Reaktionen können einzig über das RFID Modul provoziert werden. Dies hat die Entscheidung massgeblich beeinflusst, auf dem Arduino lediglich ein Code zu aktivieren welche die Karten erkennt aber nicht weiter verändert.

Alternativ hätte die komplette RFID Protokoll Logik ebenfalls über das Serielle Interface übertragen werden können, wobei das Arduino dann nur noch als reine Steckplatine gedient hätte (dieser Modus wird „Transparent Operation“ genannt. Um dies zu erreichen hätte allerdings auf dem Arduino der ARM CHIP entfernt werden sollen, was bei für den Projekt vorgelegene Version nicht möglich war. Zudem wäre der Arduino Teil des Projekts daraufhin zu gering als Nennenswert gewesen, was dem ursprünglichen Ziel, ein Projekt mit Hilfe eines Arduinos zu realisieren, nicht diente.

## Machbarkeitsanalyse zum Auslesen unserer neuen Campus Card

Die Online Recherchen ergaben, dass die Studentenausweise von InterCard erzeugt wurden. Ein Online Kurzportrait und eine Erfolgsstory zur Implementation konnten dies dokumentieren.



Auf der Webseite von InterCard wird präsentiert, welche Kartenmodi zur Verfügung stehen. Ein visueller Abgleich zeigte, dass für unseren Studentenausweis keine Karte mit einem Chip verwendet wurde und entsprechend auf den Typ Mifare Classic 1 Kbyte gesetzt wurde.

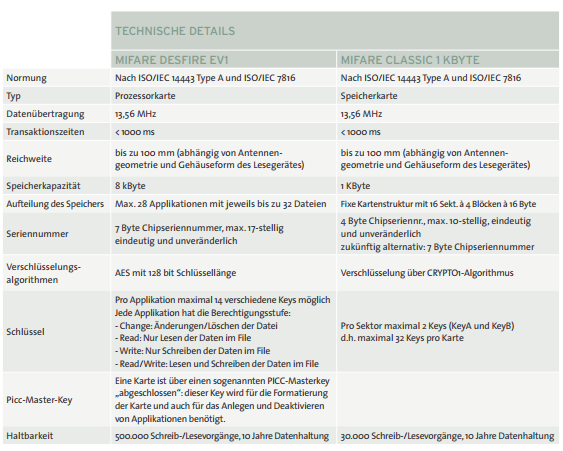
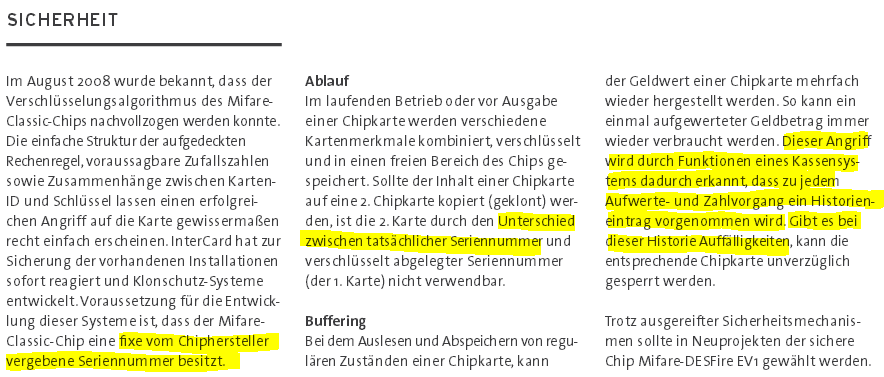


Abbildung : Kartentypen im Angebot von InterCard[[4]](#footnote-4)

Für den CRYPTO1 Algorithmus existiert laut Wikipedia Eintrag[[5]](#footnote-5) seit 2008 genug Verständnis um die „Verschlüsselung“ zu brechen.

Nach Abklärungen mit den Sicherheitsbeauftragten der ZHAW wäre es mir nur nach Unterzeichnen eines NDA erlaubt gewesen, die Sicherheit der auf den Karten abgelegten Daten zu testen. Um für diese Semester keine zusätzliche Schwierigkeit einzubauen wurde entsprechend auf diese Herausforderung verzichtet.

Ein entsprechendes auslesen wäre allerdings mit den heute technisch verfügbaren Mitteln machbar gewesen. Aus diesem Grund weisst auch die InterCard in ihrem technischen Prospekt darauf hin, dass stattdessen die Verwaltungssoftware im Hintergrund Validitätsprüfungen durchführen soll. Der Hinweis auf die „vom Chiphersteller vergebene Seriennummer“ ist naiv, da im Internet ohne Probleme Karten erhältlich sind, auf welchen der Seriennummereintrag verändert werden kann.



## Spiel in Python und mit einem Arduino

Als Alternative wurde ein Spiel realisiert. Die entsprechenden Aufwände und Entscheidungen sind im Kapitel 4 dokumentiert.

Nach Abschluss der Arbeit kann ich sagen, dass das Projekt in weiten Teilen Spass gemacht hat und lehrreich war. Mein Hauptziel des erlernen der neuen Technologien konnte erreicht werden. Mit meinen Designentscheiden bin ich sehr zufrieden und denke, diese bei einer Wiederholung wieder so gelöst zu haben. Einzig beim GUI des Spiels hätte ich bei einem nächsten Versuch mit grosser Wahrscheinlichkeit die Aufgabe vereinfacht und stattdessen ein HTML/CSS GUI verwendet, welches über Python angezeigt worden wäre. Unsicher bin ich bei der Wahl der Python Version. Normalerweise ist es immer die beste Idee auf der neuesten, verfügbaren Version zu entwickeln. Da bei den Python Bibliotheken aber eine so grosse Vielfalt besteht, hätte ein genaues abklären von Anforderungen allenfalls eine andere Versionswahl ergeben.

Gerne hätte ich noch mehr am Spielumfang weiter entwickelt, für das wäre der Zeitaufwand aber durchaus noch 40 Stunden länger gewesen, was in anbetracht des vordefinierten Zeitbudget unrealistisch war. Ein Alternativprojekt wäre dann sinnvoller gewesen.

Schade finde ich, dass meine ursprünglichen Versuche zur Sicherheit von RFID Karten nicht im Rahmen dieser Arbeit tätigen konnte. Ein Interesse dafür wäre durchaus immer noch vorhanden, kann aber nicht legal verfolgt werden da eine veröffentlichung von Ergebnissen nicht erlaubt ist.

Betreffend Innovativer Charakter ist festzuhalten, dass meine Recherche keine Spiel zu Tage gebracht hat, welches die RFID Technolgie verwendet und mit Hilfe des Computers deshalb alleine gespielt werden kann. Alle Spielmodi (die implementierten und nicht implementierten) waren eigene Ideen analog zu einfachen und bekannten offline Spielvarianten. Diese Entwicklung kann so einem sonst in Ruhestand tretenden RFID Sensor zu einem weiteren Daseinszweck verhelfen. Einzige Voraussetzung ist das lesen Karten und ausgeben von UIDs in einer zuverlässigen Form.

# Literaturverzeichnis

|  |  |
| --- | --- |
| Nr. | Verweis |
| L1 | Klaus Finkenzeller: *RFID Handbuch*. Hanser, 6. Auflage, 2012 |
| L2 | Link zu Vortrag an der BSides London  <http://www.youtube.com/watch?v=MGPGv2HnGtU> |
| L3 | Link zu den Slides des Vortrags an der BSides London: <http://www.slideshare.net/d3sre/bsideslondon-rookie-talk-rfid-hacking-an-introduction> |
| L4 | Link zu Vorschlägen zur Singleton Implementation: <http://stackoverflow.com/questions/6760685/creating-a-singleton-in-python> |
| L5 | Link zu Vorschlägen zur Factory Pattern Implementierung :  <http://stackoverflow.com/questions/21025959/factory-design-pattern> |
| L6 | Link zu Vorschlägen zur Strategy Pattern Implementierung:  <http://codereview.stackexchange.com/questions/20718/the-strategy-design-pattern-for-python-in-a-more-pythonic-way> |
| L7 |  |
| L8 | Bild Arduino R3 Front: <http://arduino.cc/en/uploads/Main/ArduinoUno_R3_Front.jpg> |
| L9 | Bild Arduino mit Xbee RFID Shield:http://www.cooking-hacks.com/documentation/tutorials/rfid-13-56-mhz-nfc-module-for-arduino |
| L10 | InterCard Karten Technologien: <http://www.intercard.org/de/produkte/IC_Karten_Technologien.pdf> |
| L11 | Crypto-1 Verschlüsselungs Algorithmus und Sicherheit: <http://en.wikipedia.org/wiki/Crypto-1> |
| L12 |  |
| L13 |  |
| L14 |  |

Tabelle : Literaturverzeichnis

# Anhang

## Glossar

| ID | Begriff | Beschreibung |
| --- | --- | --- |
| G1 |  |  |
| G2 |  |  |
| G3 | Pomodoro Technik | Methode zum Zeitmanagement, welche Arbeiten in Blöcke unterteilt und Pausen miteinrechnet. Das System wurde für die vorliegende Arbeit angepasst und stattdessen mit zweimal 60-Minutenblöcken gearbeitet.  Siehe: <http://de.wikipedia.org/wiki/Pomodoro-Technik> |
| G4 | Scrum | Agiler Projektmanagementprozess basierend auf Iterationen mit festen Iterationsfenstern.  Siehe: <http://de.wikipedia.org/wiki/Scrum> |
| G5 | Kanban | Agiler Projektmanagementprozess, der ebenfalls wie Scrum mit Iterationen arbeitet, allerdings nicht feste Iterationsfenster nutzt, aber die Anzahl in Bearbeitung befindlicher Tasks limitiert.  Siehe: <http://de.wikipedia.org/wiki/Kanban> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Tabelle : Glossar

1. Siehe Literaturverzeichniseintrag: L4 [↑](#footnote-ref-1)
2. Siehe Literaturlistenverzeichniseintrag: L5 [↑](#footnote-ref-2)
3. Siehe Literaturverzeichniseintrag: L6 [↑](#footnote-ref-3)
4. Siehe Literaturverzeichniseintrag [↑](#footnote-ref-4)
5. Siehe Literaturverzeichniseintrag [↑](#footnote-ref-5)