**Warshooter (FAT)**

**The First Automatic Turret made out of Lego**

# Projektbeschreibung

In diesem Kapitel wird eine kurze Übersicht verschaffen, wie das Projekt aussehen sollte. Mit Hilfe des Programm RobotC soll ein Programm geschrieben werden in welches wir ein vollautomatischen Geschütz über einen Joystick steuern können. Es soll sich nach links, rechts sowie hoch und runter bewegen können. Wird der Joystick gedrückt schiesst es Legosteine. Das Magazin wird über einen RGB-Sensor überwacht.

# Mitwirkende

|  |  |  |
| --- | --- | --- |
| **Name** | **Vorname** | **Funktion** |
|  |  |  |
| Rüedi | Manuel | Hardware- & Softwareentwickler |
|  |  |  |
| Schmeisser | Michael | Scrum-Master, Hardwareentwickler |
|  |  |  |
| Stelling | Fabian | Product Owner |
|  |  |  |
|  |  |  |
|  |  |  |

[Projektbeschreibung 1](#_Toc164168119)

[Mitwirkende 1](#_Toc164168120)

[Product Backlog 3](#_Toc164168121)

[Use Case 3](#_Toc164168122)

[User Stories 3](#_Toc164168123)

[Vergabe der Story Points 3](#_Toc164168124)

[Referenz-Tasks 4](#_Toc164168125)

[Failure Cases 5](#_Toc164168126)

[Hardware / Software 5](#_Toc164168127)

[Definition Of Done 5](#_Toc164168128)

[Anpassungen Use Case 6](#_Toc164168129)

# Product Backlog

## Use Case

|  |  |  |
| --- | --- | --- |
| **Funktion** | **Akteur / Aufgabe** | **Priorität** |
|  |  |  |
| Aufbau | Das Entwicklerteam trifft die richtige Auswahl der Hardware und tut diese dementsprechend Anpassen bei Zusätzen oder Veränderungen. | 1 |
|  |  |  |
| Steuerung | Das Entwicklerteam erstellt Grundfunktionen für Steuerungen. | 2 |
|  |  |  |
| Laden | Das Entwicklerteam erstellt ein Drop-Down System für das Nachladen des Warshooters und eine Überwachung. | 2 |
|  |  |  |
| Schiessen | Das Entwicklerteam sorgt dafür dass der Warshooter vollautomatisch schiessen kann. | 3 |
|  |  |  |
| Dokumentation | Der Scrum-Master soll die Dokumentation stets Up To Date halten. Der Product Owner gibt diese Frei. | 2 |

Prioritäten: 1 – High, 2 – Mid, 3 – Low

## User Stories

Der Product Owner möchte dass der Roboter über einen Joystick steuerbar ist.  
Die Entwickler erstellen einen Joystick welcher für die Bewegung und das schiessen des Roboters zuständig ist.

Der Product Owner würde gerne Wissen wie viel Schüsse noch im Magazin sind.  
Die Entwickler lassen über ein akustisches Signal ein Feedback für die vorhanden Munition geben.

## Vergabe der Story Points

Die Story Points definieren die Schwierigkeit und Dauer einer Story. Dies wird mit einem Punktesystem gehandhabt und kontrolliert.

Punkte werden von 1 – 10 verteilt und geben folgende Informationen wie oben gehabt zurück.

Die Skala ist wie folgt:

1. Sehr Einfach und schnell gelöst
2. Leicht Anspruchsvoll und nicht zu Zeitintensiv
3. Sehr Schwer und Zeitaufwändig

Alle Punkte dazwischen sollten selbstverständlich sein. Die Skala gibt nur eine ungefähre Richtung vor und keine genaue Zeitangabe.

## Referenz-Tasks

**Aufbau**

1. Die Bausteine für den korrekten Aufbau der Hardware musst stets angepasst werden
2. Die Sensoren für die Haupt- und Nebenfunktionen müssen ausgewählt sein
3. Aufbau des Joysticks auf dem Mindstorm-Controller
4. Aufbau des Schiessmechanismus mit Zahnriemen und Motoren
5. Aufbau des Magazins mit Dropdown Möglichkeit mit Munitionsüberwachung
6. Simple Funktionskontrolle des Aufbaus, die mechanische Funktion prüfen
7. Aufbau Plan erstellen, für leichtes Nachbauen

**Steuerung**

1. Joystick Mechanische Funktion testen
2. Joystick Taster Funktionen zuweisen (oben, unten, links, rechts)
3. Betätigen als Funktionen zum schiessen erstellen

**Laden**

1. Dropdown System erstellen welches nach einem Schuss, Munition in den Lauf lässt
2. Ist die Munition knapp, soll eine akustische Meldung ertönen

**Schiessen**

1. Durch betätigen der mittleren Taste soll das Gewehr schiessen solang der Taster betätigt ist (Vollautomatisch)
2. Nach jedem Schuss muss eine Abfrage gewiss vorhandener Munition im Lauf gestellt werden

**Dokumentation**

1. Erstellung / Erweiterung des Product Backlogs
2. Erstellung / Erweiterung des Sprint Backlogs , Sprint Reviews
3. Erstellung / Erweiterung des Aktivitätsdiagrammes
4. Erstellung / Erweiterung der User Stories

Die Referenz-Task werden in ein separates Dokumenten übernommen, welches für die Sprint Backlogs verwendet werden. Für dies muss das Dokument «Sprint Backlog» geöffnet werden. Alles Sprints werden dort definiert.

# Failure Cases

## Hardware / Software

|  |  |  |
| --- | --- | --- |
| **Function** | **Failure** | **Solution** |
|  |  |  |
| Laden | Der RGB Sensor weist komische Werte auf. | Farbkonstruktion des Magazins und der Munition upgraden. |
|  |  |  |
| Schiessen | Der Warshooter schiesst nicht vollautomatisch. | Hardware erneut kontrollieren gegeben falls Software updaten. |
|  |  |  |
| Aufbau | Das Gerüst des Geschützes fällt durch die Kraft der Motoren auseinander. | Stabilisierung durch zusätzliche Bauteile. |
|  |  |  |
| Dokumentation | Die Dokumentation ist nicht vollständig, Up To Date. | Tägliche Dokumentation fixieren. |
|  |  |  |
|  |  |  |

# Definition Of Done

Die Definition Of Done beschreibt wenn ein Inkrement als Fertig (Done) gilt, dies ist der Fall wenn folgende Punkte eingehalten wurden:

* Die Hardware wurde so ausgearbeitet und Überprüft, dass sie hält
* Die Hardware wurde auf Fehler und Ausfälle getestet und im Kapitel Failure Cases Dokumentiert
* Die Software gilt erst als Fertig wenn alles dokumentiert, reviewed und getestet wurde
* Die Software wurde auf Fehler und Ausfälle getestet und im Kapitel Failure Cases Dokumentiert
* Eine Dokumentation gilt als Done wenn diese vom Product Owner reviewed und Freigegeben wurden
* Ein Punkt welcher «Done» nach einem Sprint wäre, wird als «In Progress» und nicht als «Done» gekennzeichnet und in den weiterfolgenden Tag übernommen (Dokumentationen welche stets geupdatet werden etc.)

# Anpassungen Use Case