# C#-Desktopapplikation

Die Desktopapplikation verbindet die SQS-Message-Queue mit dem Roboter. Darunter fällt das Erstellen einer Verbindung zu SQS Queue und Roboter, das Empfangen und Übersetzen von Nachrichten, das Senden von Befehlen an den Roboter entsprechend dem Kommando.

## Struktur

Zur Erstellung des Programms werden 4 Dateien benötigt: Die C#-Dateien “Program.cs” und “Ev3Command.cs” und die Konfigurationsdateien “App.config” und “packages.config”.

Die “packages”-Datei enthält eine Liste der benötigten Pakete die Visual Studio benötigt, um das Projekt zu bauen. Mehr dazu im Bereich “Einrichtung von Visual Studio zur Bearbeitung”.

Die “App.config” besteht aus 3 Key-Value-Paaren:

* “Ev3Port” ist der Bluetooth Port, mit dem der Roboter verbunden ist.
* “AwsSqsAddress” ist die Adresse der SQS-Queue
* “StepsPerDegreeTurn” ist ein Faktor um die Drehung des Roboters zu kalibrieren

Die C#-Datei “Ev3Command.cs” enthält eine Klasse zum Speichern der Argumente eines Kommandos.

Die C#-Datei “Program.cs” enthält den gesamten Code außer die Ev3Command-Klasse. Ihr Inhalt und Funktionsweise wird im Bereich “Funktionsweise” erklärt.

## Gebrauch

Um das Programm zu verwenden muss es mittels Visual Studio gebaut werden. Danach kann das Programm ausgeführt werden. Optional kann das Programm für den verwendeten Roboter kalibriert und erweitert werden.

### Einrichtung Installation von Visual Studio Community 2017

Für die Installation und den Download von VS C17 verweisen wir hier auf die offizielle Dokumentation und Downloadbereich von Microsoft, da sich dies ständig ändern könnte:

<https://visualstudio.microsoft.com/de/downloads/>

HINWEIS: Die C#-IDE wird auf jeden Fall benötigt. Alles andere ist in diesem Fall optional bei der Installation.

##### Zusätzliche Paketinstallation

Nachdem man VS erfolgreich installiert hat, werden noch zusätzliche Pakete benötigt, die nicht standardmäßig zur Auswahl stehen.

Hierfür wechselt man zunächst in die NuGet-Paketmanager Konsole. Zu dieser kommt man über die Reiter „Extras“ 🡪 „NuGet-Paket-Manager“ 🡪 „Paket-Manager-Konsole“

In dem aufscheinen Fenster gibt man nun folgenden Befehl ein:

„Install-Package AWSSDK.SQS -Version 3.1.0.4“

### Konfigurieren des Bluetooth Ports und der AWS-Adresse

ALLE KONFIGURATION FINDEN IN DER DATEI „App.config“ STATT!

Zunächst richten wir die Verbindung zum EV3 mittels Bluetooth ein. Hierfür muss man im Programm die COM angeben, auf der die Bluetoothverbindung hergestellt ist.

Dazu sucht man folgende Zeile und ändert das COM entsprechend ab:

<add key="Ev3Port" value="COM3" /> <!-- or for Linux/OSX, "/dev/tty.YourPortName" -->

Des weitern muss man noch einstellen von wo sich das Programm die Nachrichten/Befehle holen soll:

<add key="AwsSqsAddress" value="https://sqs.eu-west-1.amazonaws.com/697251208257/Test2Queue" />

Hier gibt man die ARN-Adresse des SQS-Services an, den man vorher erstellt hat.

WICHTIG: In der Datei „Program.cs“ gibt es unter der Function „private static void Configure()“ die Einstellung, in welcher Region sich der Service befindet! Dieser muss richtig mit dem eigenen Service zusammenpassen:

RegionEndpoint endpoint = RegionEndpoint.EUWest1;

Dies ist in unserem Fall „EUWEST1“!

### Bauen des Programms

Zu guter Letzt muss man das Programm nur noch „Bauen“ oder in Englisch „Builden“. Hierzu drückt man folgende Tastenkombination:

„STRG + SHIFT + B“

Und anschließend um das Programm auszuführen:

„F5“

### Ansteuern des Roboters

Sobald das Programm, Alexa und der EV3 bereit sind, kann angefangen werden.

Zuerst wird das Programm gestartet. Nach einigen Sekunden sollte der Roboter einen Piepston von sich geben und am Terminal am Computer die Nachricht “Connected… Waiting for Commands…” sichtbar sein. Falls nicht ist die Verbindung zum Roboter fehlgeschlagen.

Das Programm wartet nun auf Kommandos. Sobald Alexa ein Kommando übergeben wurde, erscheint der Inhalt der Nachricht am Terminal. Der Roboter sollte den Befehl nun ausführen.

Zum Beenden des Programms kann das Fenster vom Terminal geschlossen werden oder im Terminal “STRG-c” gedrückt werden.

### Kalibrieren der Drehung

Damit der verwendete Roboter sich entsprechend der Angegebenen Gradzahl, die Motoren nicht zu lange oder zu kurz dreht, kann ein Faktor geändert werden.

Der Faktor ist eine Kommazahl die sich in der “App.config” befindet.

Eine Erhöhung des Faktors entspricht einer weiteren Drehung, die Verringerung einer Kürzeren.

Standardmäßig ist dieser Wert auf 3.5 gestellt.

### Erweiterung der Funktionalität

Um Kommandos hinzuzufügen oder deren Struktur zu ändern, muss der Code angepasst werden.

Kommandos können in der für diesen Zweck markierte stelle in der Funktion “ProcessCommand” in der Datei “Program.cs” hinzugefügt werden.

Die Struktur kann in der C#-Datei “Ev3Command.cs” geändert werden. Von der Klasse abhängige Funktionen in der “Program.cs” müssen ebenfalls angepasst werden.

## Funktionsweise

In diesem Bereich wird die Implementierung der Funktionalität erklärt. Es wird chronologisch, entsprechend dem Programmverlauf vorgegangen.

### Laden der Konfiguration und Verbindung zur SQS-Queue

Die „Configure“-Funktion wird aufgerufen. Sie ladet Bluetooth-Port, SQS-Adresse und den Faktor zum Kalibrieren der Drehung aus der “App.config”-Datei und stellt eine Verbindung zur SQS-Queue auf.

### Verbindung zum Roboter

Als nächstes wird die Execute-Funktion aufgerufen. Der erste Teil dieser Funktion erstellt verbindet sich mit dem Roboter über Bluetooth und gibt bei Statusinformationen aus. Die Port-Nummer wird separat über die “\_brick\_BrickChanged”-Funktion ausgegeben.

### Nachrichten abrufen

Die Funktion “PollForQueueMessage” wird aufgerufen. Sie sendet eine Anfrage für Nachrichten an die Queue. Wenn sich eine Nachricht in der Queue befindet, wird sie entgegengenommen und aus der Queue gelöscht. Danach wird die Nachricht am Terminal ausgegeben und durch die “GetEv3CommandFromJson”-Funktion bearbeitet.

### Kommando erstellen

“GetEv3CommandFromJson” deserialisiert die Nachricht und erstellt ein EV3-Kommando aus den Argumenten, die über die Nachricht übermittelt wurden.

### Kommando interpretieren und ausführen

Wieder in der “Execute”-Funktion wird das Kommando an “ProcessCommand” weitergegeben. Die Funktion wählt ob ein Kommando und wenn Ja welches an den Roboter gesendet werden. Eine der folgenden Funktionen werden Ausgeführt.

“MoveForward” zeigt eine Statusmeldung am Terminal, erstellt und sendet ein Kommando an den Roboter. Vorherigen Kommandos werden überschrieben. Das Kommando besteht aus einzelnen Einstellungen für zwei Motoren. Falls eine Distanz in Centimeter als Argument angegeben wurde, wird die Distanz berechnet und verwendet. Falls nicht, fährt der Roboter ca. 30 cm vorwärts.

“MoveBackward” funktioniert genau gleich wie “MoveForward”, mit dem Unterschied, dass die Polarität der Motoren umgekehrt ist.

“MoveLeft” und “MoveRight” haben den selben Ablauf wie “MoveForward” und “MoveBackward”, mit dem Unterschied, dass die Motoren in entgegengesetzter Richtung gedreht werden.

### Abschluss

Das Programm durch die Eingabe von “STRG-c” terminiert oder durch das Schließen des Terminals.

## Quelle

Das Programm ist eine modifizierte Version des Programms von Jim Drewes, welches unter diesem Link verfügbar ist:  
<https://github.com/jimdrewes/alexa-to-ev3/tree/master/alexa-to-ev3.console>

Das Programm wurde kommentiert, für unseren Roboter konfiguriert und die Funktionalität erweitert.

### Änderungen

Die beiden C#-Dateien wurden vollständig kommentiert. Zusätzlich wurde die Reihenfolge der Funktionen in “Program.cs” chronologisch entsprechend dem Aufruf sortiert.

Im Rahmen der Konfiguration wurde in der “App.config”-Datei der Bluetooth-Port und SQS-Adresse eingestellt. Außerdem wurde in “Program.cs” in den “Move…”-Funktionen der Port und die Polarität der Motoren angepasst.

Zur Kalibrierung der Drehung des Roboters wurde ein Faktor hinzugefügt. Dieser Faktor wird unter dem Eintrag “StepsPerDegreeTurn” der “App.config” eingestellt. Es ist der Faktor, um den die Räder gedreht werden müssen um den Roboter um x Grad zu drehen. Er wurde hinzugefügt da die Steuerung mit Gradzahl einfacher ist als die Eingabe der Steps, und sie so einfacher als im Code selbst geändert werden kann. Konkret wurde ein Key-Value-Paar in der Config-Datei und im Code eine Konstante mit dem Key-Namen und eine Private Variable für den Wert. Die Configure Funktion im Code wurde ebenfalls erweitert um den Wert der Privaten Variable entsprechend der Config-Datei zu setzen. Die „Moveleft“- und „Moveright“-Funktionen wurden angepasst damit sie den Faktor verwenden.

Die Struktur der EV3-Kommandos wurde geändert um die Angabe des Roboternamens und eines Betrags für die Bewegung zu unterstützen. Dafür wurde die Ev3Command-Klasse um die Einträge “DeviceName” und “Value” erweitert. Die “GetEv3CommandFromJson”-Funktion wurde angepasst um die Werte zu setzen. Der “Value” wird in den “Move…”-Funktionen verwendet um die Weite der Bewegung/Drehung zu bestimmen. Er ist Optional und ein Standardwert wird verwendet falls er nicht angegeben wird.