Tobias Tschinkowitz

Albert-Schweitzer-Str. 3

75031 Eppingen

tobias.tschinkowitz@outlook.com

Immatrikulationsnummer: 1346735

IGP01 – Integrationsprojekt

Assignment

AlmueRaspi - Heimautomation mit dem Raspberry-Pi

09.11.2016

AKAD Hochschule Stuttgart

Inhaltsverzeichnis

1 Einleitung

Das Internet der Dinge oder kurz „IoT“ gewinnt im Alltag immer mehr an Bedeutung. Kühlschränke, Waschmaschinen und Fernseher sind hierbei die Vorreiter dieser Technologie. Sie können durch die Vernetzung mit dem Internet das Leben des Besitzers solcher Maschinen deutlich erleichtern. Beispielsweise könnte ein Kühlschrank fehlende Bestände von Lebensmitteln direkt auf den Einkaufszettel in einer Smartphone-App hinzufügen oder sogar direkt bei einem Lieferanten nachbestellen. Die Vernetzung aller möglichen Geräte birgt natürlich auch ein immenses Risiko, da die Konsumenten dieser, meist wenig Geld hierfür ausgeben wollen und die Software der Geräte dadurch meist sehr unausgereift ist, was diese wiederum angreifbar macht. Dies hat man an den jüngsten DDoS-Attacken gesehen, in denen ein Netz von kompromittierten Haushaltsgeräten dazu verwendet wurde, größere Internetseiten lahmzulegen.

Ziel dieses Assignments ist die Entwicklung einer Software für die Steuerung von Rollläden, welche über eine Android-App gesteuert werden können. Als Hardwareplattform wird hierfür der RaspberryPi verwendet. Das gesamte Projekt bezieht sich darauf eine manuelle Rollladensteuerung mit Tastern, um eine automatische zu erweitern. Es wird also lediglich die bestehende Schaltung um den RaspberryPi erweitert.

Im Grundlagenteil wird zunächst die gewählte Programmiersprache C# bzw. das .NET Framework/Mono näher betrachtet. Danach wird der RaspberryPi vorgestellt und sein Funktionsumfang für diesen Anwendungsfall erläutert. Die Software auf dem RaspberryPi kommuniziert über das IoT-Protokoll MQTT mit dem Smartphone, welches auch grundlegend erklärt wird.

Im Hauptteil des Assignments wird zuerst der Hardwareaufbau vor Ort mit den angeschlossenen Rollläden kurz beschrieben. Das Hauptaugenmerk dieses Assignments liegt jedoch auf der Softwareentwicklung, welche im Anschluss daran umfassend erläutert wird. Abschließend wird die Ausarbeitung noch einmal zusammengefasst und die Android-App vorgestellt.

2 Grundlagen

2.1 Das .NET Framework und die C# Programmiersprache

„Mit dem Erscheinen vom .NET Framework im Jahr 2002 wagte Microsoft einen revolutionären Schritt und stellte eine Plattform bereit, die es möglich machte, nahezu alle erdenklichen Anwendungen mit einer Entwicklungssprache nach Wahl zu Codieren“[1]. Die Sprache C# etablierte sich als Quasistandard für das .NET-Framework, wobei es noch weiter wie Visual-Basic und F# gibt. Für dieses Projekt wurde die Sprache C# gewählt, welche syntaktisch den anderen C Sprachen (C und C++) sehr ähnlich ist. C# gewinnt durch das Open-Source Projekt .NET Core von Microsoft zunehmend an Bedeutung im Linux und macOS Umfeld, wobei früher hingegen lediglich Windows Plattformen unterstützt wurden. Das .NET Framework wurde gewählt, da die Codierung der RaspberryPi Applikation und auch die der Android App im selben Sprachumfeld stattfinden sollte. Leider gibt es momentan von .NET Core noch keine Version, welche auf einer ARM-Architektur läuft, auf welcher der RaspberryPi aufbaut. So musste hier also die Alternative Mono[2] verwendet werden. Mono stellt das .NET Framework für Unixoide Systeme bereit, was bedeutet, dass man Code, welcher unter Windows-Plattformen kompiliert wurde, auch mit Linux und Unix Systemen ausführen kann, welche das Mono-Framework installiert haben. Das .NET Framework erleichtert die Programmierarbeit erheblich, da viele Funktionen, welche unter Ansi-C mühsam programmiert werden müssen schon bereitstehen. C# ist eine rein objektorientierte Sprache. Objektorientierung kann, wenn sie richtig angewandt wird, Code deutlich wartbarer, testbarer und strukturierter machen.

2.2 RaspberryPi

Der RaspberryPi ist momentan so etwas wie der heilige Gral für die Maker-Szene. Unter der Maker-Szene versteht man die sogenannte Do-It-Yourself Kultur, welche es sich zum Ziel gemacht hat, ihre Anwendungen selbst zu entwickeln und nicht von Großkonzernen für teures Geld einzukaufen. Momentan befindet sich der Raspberry Pi in der Version 3 und bietet ein

[1] (Kühnle, 2016, S. 32)
[2] http://www.mono-project.com/

Vielfaches an Leistung im Vergleich zu Desktop-PC´s von vor 10 Jahren, obwohl er lediglich einen Bruchteil seiner Größe aufweist.

Im Folgenden wird der für dieses Assignment verwendete RaspberryPi in der Version 2 abgebildet.

Abbildung 1 - Raspberry Pi 2 Model B V1.1

Der RaspberryPi eignet sich hervorragend um Steuerungsaufgaben im häuslichen Umfeld zu übernehmen. Man kann sowohl ein Windows 10 Betriebssystem als auch ein Linux Betriebssystem hierzu verwenden, welches auf einer SD-Karte gespeichert wird. In diesem Assignment wird sich jedoch auf das Linux Betriebssystem Raspbian[3] beschränkt, welches auf der Distribution Debian basiert. Die GPIO (General-Purpose Input/Output) Pins können mit jeder erdenklichen Programmiersprache angesteuert werden. In Abbildung 1 ist der GPIO-Header mit 40-Pins oben zu erkennen. Diese Pins können, wie der Name schon erahnen lässt, sowohl als Eingänge als auch Ausgänge verwendet werden. Eingänge müssen mit einem Signal von 3,3VDC gespeist werden und Ausgänge geben eine Spannung von 3,3VDC aus. Die Pins können sowohl über das Linux interne sysfs[4] (System-Filesystem) oder über den Linux-Kernel eigenen GPIO-Legacy[5] Treiber direkt gesteuert werden. Die wohl einfachste Art einen GPIO Pin zu steuern, ist die Verwendung des System-Filesystems. In diesem Assignment wird jedoch eine Open-Source Bibliothek für die Steuerung der Pins verwendet, welche direkt über den GPIO-Legacy Treiber agiert.

[3] https://www.raspbian.org/
[4] (Linux Kernel Organization, The Linux Kernel Archives, 2015)
[5] (Linux Kernel Organization, The Linux Kernel Archives, 2016)

2.2 MQTT Protokoll

Das Message Queue Telemetry Transport Protokoll oder kurz MQTT wurde 2013 von der OASIS (Organization for the Advancement of Structured Information Standards) als IoT Protokoll standardisiert. Entwickelt wurde es ursprünglich 1999 von IBM und Arcom und wurde gebührenfrei veröffentlicht. „MQTT ist ein extrem einfaches und leichtgewichtiges Messaging-Protokoll. Die Publish / Subscribe Architektur ist so konzipiert, offen und einfach zu implementieren um mit einem einzigen Server Tausende Remote-Clients zu unterstützen.“[6]

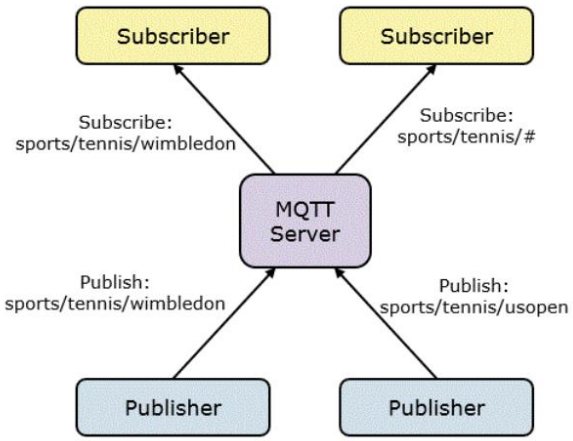


Abbildung 2- Beispiel für die Publish/Subscribe Methode[7]

Der MQTT Server dient hier als sogenannter Broker, also Vermittler zwischen den Publishern und Subscribern. Ein Subscriber, welcher eine Nachricht erwartet, abonniert sein Topic beim Broker, in der Abbildung 2 zum Beispiel „sports/tennis/wimbledon“. Jede Nachricht, die nun von einem Publisher an dieses Topic veröffentlicht wird, wird direkt vom Broker an die Subscriber dieses Topics weitergeleitet. Die Kommunikation erfolgt hier ausschließlich über das TCP/IP Protokoll. Publisher und Subscriber müssen sich also nicht direkt kennen, sondern brauchen sich nur am Broker anzumelden. Da das Protokoll Plattformkompatibel ist, ist es nicht nötig auf Besonderheiten in der Kommunikation zwischen verschiedenen Betriebssystemen zu achten. All das wird vom Protokoll selbst erledigt. Die große Besonderheit ist, dass der Subscriber nicht ständig beim MQTT Server anfragen muss, ob neue Nachrichten vorliegen,

[6] (IBM und WebSphere, 2014, S. 6)
[7] (IBM und WebSphere, 2014, S. 22)

sondern dass der MQTT-Broker ein Event auf dem Subscriber auslöst und dieser nur dann eine Nachricht bekommt, wenn auch eine vorhanden ist. Einzelne Details zum Protokoll werden noch einmal im Hauptteil dieses Assignments erläutert.

3 Dokumentation der gesamten Steuerung

3.1 Hardwareaufbau

Der RaspberryPi unterstützt an seinen Ausgängen nur Spannungen von bis zu 3,3VDC. Mit diesen Werten ist es natürlich nicht möglich, den Rollladen direkt anzusteuern. Im Haus, wo dieses Projekt umgesetzt wurde, sind bereits 10 Rollläden mit Vor-Ort Tastern über Stromstoßschalter EGS12Z-8..230VUC der Firma Eltako verbunden. Das Datenblatt dieser ist im Anhang 3 zu finden. Jeder dieser Stromstoßschalter hat jeweils zwei Eingänge für die Auf- und Abwärtsbewegung der Rollläden. Der Vorteil dieser Schalter ist, dass Sie bereits intern die Eingänge gegeneinander verriegeln und somit einen Kurzschluss verhindern, welcher zu einem Komplettausfall des Motors führen könnte. Um die GPIO Ports an diese Stromstoßschalter zu koppeln, mussten zunächst Relais besorgt werden. Hierzu wurden zwei 8 Kanal Relais-Module und ein 4 Kanal Relais-Modul der Firma SainSmart verwendet. Jedes der einzelnen Relais auf dem Modul ist für eine Aktion zuständig (Auf- oder Abfahrt eines Rollos). Die SainSmart Relais benötigen zum sicheren Schalten der Relaiskontakte, welche bis zu 250VDC bei 10A schalten können, ein 5VDC Signal, was leider mit dem RaspberryPi nicht möglich ist, da dieser nur 3,3VDC an seinen GPIO Ports liefert. Hierzu wurde eine Hilfsplatine mit drei Darlington-Transistor-Arrays ULN2803A erstellt. Die erste Seite des Datenblattes ist im Anhang 4 zu finden. Mit diesen ist es möglich, mit den gegebenen 3,3VDC der GPIO Ports die 5VDC Spannungsversorgung, welche sich in einem Hutschienennetzteil befindet, auf die Eingänge der Relaiskarte zu geben. Da bei den Relais-Modulen die Eingänge invertiert sind, d. h. bei Low-Pegel schaltet das Relais und bei High-Pegel fällt es ab, war dies von Vorteil, da das Darlington-Array durch die NPN-Verschaltung schon eine Signalinvertierung vornimmt. Die Spannungsversorgung für den RaspberryPi und die Relais-Module übernimmt ein 5VDC Hutschienennetzteil, welches bis zu 3A liefert. Da am RaspberryPi keine weiteren USB-Geräte angeschlossen sind, benötigt dieser ca. 330mA und die Relais-Module pro Kanal ca. 15-20mA,

was auf einen Gesamtverbrauch von ca. 730mA schließen lässt. Somit ist das Netzteil ausreichend Dimensioniert. Der gesamte Schaltplan ist im Anhang 2 zu finden.

3.2 Softwareentwicklung

3.2.1 Die Klasse ConfigurationController

Da es für den Anwender so einfach wie möglich sein soll die Software zu konfigurieren, ohne Änderungen im Quellcode vornehmen zu müssen, wurde die Klasse *ConfigurationController* erstellt. Als Konfigurationsdateiformat wurde hier JSON (JavaScript Object Notation) gewählt, da es hiermit sehr einfach ist, Objekte in einem textuellen Format abzubilden und diese wiederum in C# Objekt umzuwandeln. Der Prozess für die Abbildung eines kompilierten Objektes in einen Bytestream wird als Serialisierung bezeichnet. Die Deserialisierung macht genau das Gegenteil. Für beide Methoden wurde die unter der MIT-Lizenz stehende Open-Source Bibliothek *Json.NET*[8] gewählt. Die Konfigurationsdatei[9] enthält alle nötigen Parameter. Im ersten Abschnitt *broker* werden die Einstellungen für die Verbindung zum MQTT-Broker eingetragen. Da dieser mit der Software *Mosquitto*[10] bereits auf dem RaspberryPi läuft, ist die IP Adresse hier 127.0.0.1 bzw. localhost. Bei Mosquitto ist es möglich bestimmte Benutzerkonten anzulegen, was hier aus Sicherheitsgründen auch getan wurde. Wenn der MQTT-Broker ohne Logins betrieben wird, kann jeder der sich auf den Broker verbindet, unter der Bedingung er kennt die Topics, an diese Nachrichten absetzen, was natürlich nicht wünschenswert wäre. Im nächsten Konfigurationsabschnitt *devices* werden die Geräte definiert. Es sind mit Lichtern und Rollläden momentan zwei Gerätetypen zugelassen, jedoch ist die Software so aufgebaut, dass sie einfach um andere Gerätetypen erweitert werden kann. Bei jeder Änderung an einem Gerät die von einem Benutzer der App durchgeführt wird, sei es der Name oder die Öffnungs/Schließzeit eines Rollladens wird der Geräteteil der Konfigurationsdatei über MQTT mit den Android Geräten welche die Almue-App verwenden synchronisiert. Im Folgenden wird das Klassendiagramm des Controllers dargestellt.

[8] http://www.newtonsoft.com/json
[9] Siehe Anhang 1, S. 12
[10] https://mosquitto.org/

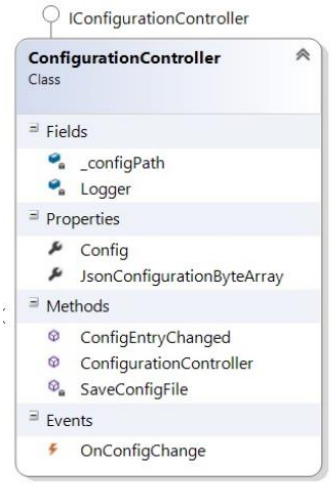


Abbildung 3- Klassendiagramm des ConfigurationController

Die Methode *ConfigEntryChanged* empfängt nach jeder Änderung eines Gerätes, welches die Schnittstelle *INotifyConfigPropertyChanged* implementiert, die Information welche Eigenschaft des Gerätes sich verändert hat. Die Methode wertet dies aus und synchronisiert die geänderte Eigenschaft mit der vorhandenen Konfigurationsdatei. Nach der Synchronisierung wird dem *MQTTController,* welcher im nächsten Abschnitt behandelt wird, mitgeteilt, dass er die Serialisierte Konfigurationsdatei, welche sich in der Eigenschaft *JsonConfigurationByteArray* befindet, via MQTT an das Konfigurationstopic *almue/config* versendet.

3.2.2 Die Klasse MqttController

Diese Klasse stellt die Hauptschnittstelle zwischen der AlmueRaspi Software und dem MQTT-Broker Mosquitto dar. Sie ist dafür verantwortlich, alle nötigen Topics für die Geräte zu abonnieren und Nachrichten, welche auf diesen abgesetzt wurden entgegen zu nehmen und sie entsprechend zu verarbeiten. Für die Kommunikation mit dem Broker wird die Open-Source Bibliothek *M2MQTT*[11] verwendet. Diese dient als MQTT-Client für die Klasse *MqttController*. Dem Konstruktor der Klasse wird ein Interface *IConfigurationController* übergeben, welches vom vorher beschriebenen *ConfigurationController* implementiert wird. Da der *MqttController* nun einen Verweis auf die Konfigurationsdatei bzw. deren Controller besitzt, kann er alle Eigenschaften, welche im Konfigurationsabschnitt *broker* definiert wurden und alle Geräte, welche unter *devices* Konfiguriert wurden am MQTT-Broker abonnieren.

[11] https://m2mqtt.wordpress.com/

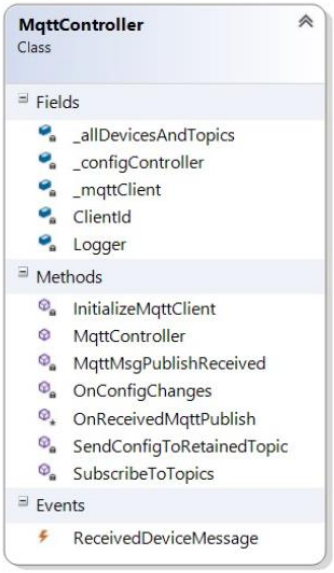


Abbildung 4- Klassendiagramm des MqttController

Im Konstruktor wird des Weiteren das private Dictionary *_allDevicesAndTopics* Initialisiert, welches als Schlüssel den Namen des Gerätes verwendet und als Werte eine Liste aller Topics. Nachdem dieses initialisiert wurde, werden die Topics mit der privaten Methode *SubscribeToTopics* im *MqttClient* registriert. Bei näherer Betrachtung dieser Methode fällt auf, dass für jedes zu abonnierende Topic ein Parameter für die Quality of Service, oder kurz QoS übergeben wird. Dieser ist definiert als QoS-Level 0 was bei dem Mqtt-Protokoll so viel bedeutet wie „At most once delivered“[12]. Vereinfacht gesprochen wird hierbei eine Nachricht nur einmal übermittelt. Es wird jedoch nicht geprüft, ob die Nachricht beim Empfänger auch tatsächlich eingetroffen ist. Dieses QoS Level wurde gewählt, da es das schnellste Kommunikationsverfahren darstellt und für den Anwendungszweck einen Rollladen zu schließen oder zu öffnen völlig ausreichen ist. Anders sieht es jedoch bei dem Topic für die Konfigurationsdatei aus. Hier wird kein Topic abonniert, sondern es wird an ein Topic gesendet, welches von den Clients, also in dem Fall die Android-Apps, abonniert wurde. Dies geschieht in der privaten Methode *SendConfigToRetainedTopic*. Hier wird ein neuer Task erstellt, um den Hauptprozess des Programmes nicht zu blockieren. Der Datenstrom wird an ein sogenanntes Retained-Topic gesendet, welches die letzte Nachricht beibehält. Sobald ein Client dieses Topic

[12] (IBM und WebSphere, 2014, S. 236)

abonniert, bekommt er den letzten Inhalt, der an dieses Topic übermittelt wurde zugesendet. Hier wurde das QoS-Level 1 „At least once delivered“[13] gewählt, da sichergestellt werden soll, die Konfigurationsdatei mindestens einmal übermittelt zu haben. Der Unterschied zum QoS-Level 0 ist hier, dass der MQTT-Broker dem MQTT-Client eine Rückmeldung über den Erfolg der Übermittlung zurücksendet. Wenn der Client keine Rückmeldung erhält, versendet er die Nachricht erneut an den Broker, bis dieser eine Rückmeldung zurückliefert. Die Methode *OnConfigChanges* wurde im Konstruktor des MqttControllers am Event *OnConfigChange* des ConfigurationControllers angemeldet. Somit wird bei jeder Änderung an der Konfigurationsdatei die Methode *SendConfigToRetainedTopic* ausgeführt und sichergestellt, dass AlmueRaspi mit den Almue-Clients dieselbe Konfigurationsdatei teilt. Die nun erstellte Topicstruktur sieht folgendermaßen aus:

```
//KonfigurationsTopic
almue/config

//Topic für Rolläden
almue/shutter/{stockwerk}/{beschreibung}
almue/shutter/{stockwerk}/{beschreibung}/timeron
almue/shutter/{stockwerk}/{beschreibung}/timeroff
almue/shutter/{stockwerk}/{beschreibung}/status
```

Empfängt der MqttClient eine Nachricht auf einem der Abonnierten Geräte-Topics, so wird die Methode *MqttMsqPublishReceived* vom Eventhandler des MqttClienten aufgerufen. Als einfaches Beispiel könnte man nun einen Rollladen auffahren, indem man an das Topic „almue/shutter/erdgeschoss/kueche“ die Nachricht „open“ absetzt. Dies wird dann vom *MqttController* entgegengenommen und in ein EventArgument *MqttDeviceEventArgs* umgewandelt. Diese Klasse stellt intern die Informationen bereit, wie die Beschreibung des Gerätes lautet, welcher Gerätetyp zu steuern ist, welche Aktion auszuführen ist und eventuell weitere Parameter, welche bei der Zeitsteuerung nötig sind. Der *MqttController* stellt öffentlich ein Event namens *ReceivedDeviceMessage* bereit, welchem immer ein *MqttDeviceEventArgs* übergeben wird. Jedes Objekt, welches sich an diesem Event anmeldet, kann nun also verwerten, was an eines der Abonnierten Topics gesendet wurde und muss nichts von der Mqtt eigenen Implementierung dahinter wissen.

[13] (IBM und WebSphere, 2014, S. 236)

3.2.3 Die Klasse DeviceController

Diese Klasse übernimmt sämtliche Steuerungsaufgaben für die Geräte und somit auch das setzen der GPIO-Pins. Hierfür wird die Open-Source Bibliothek Raspberry-Sharp-IO[14] verwendet. Zunächst zur besseren Übersicht das Klassendiagramm:

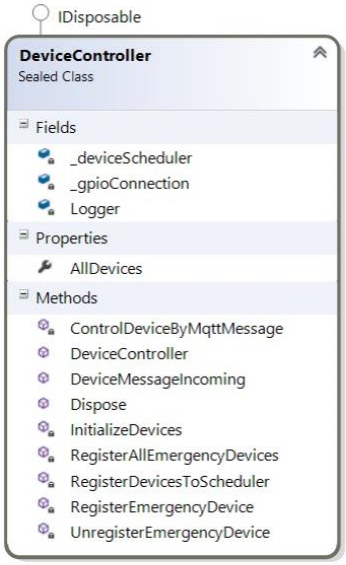


Abbildung 5- Klassendiagramm des DeviceController

Dem DeviceController werden im Konstruktor zwei Schnittstellen übergeben. Zum einen die *IConfigurationController* Schnittstelle, welche von der Klasse *ConfigurationController* implementiert wird und zum anderen ein optionaler Parameter für den *IDeviceScheduler* welcher der von der Klasse *DeviceScheduler* implementiert wird. Auf den *DeviceScheduler* wird im Assignment jedoch nicht näher eingegangen. Dieser dient lediglich dazu, die Geräte Automatisiert zu bestimmten Zeitpunkten mit bestimmten Aktionen zu steuern. Im Konstruktor wird nun die *GpioConnection* instanziiert und in dem privaten Feld *_gpioConnection* abgelegt. Konnte die *GpioConnection*, welche Teil der Raspberry-Sharp-IO Bibliothek geöffnet werden so werden alle konfigurierten Geräteklassen Instanziiert und in der öffentlichen Eigenschaft *AllDevices* abgelegt. Dies übernimmt die Methode *InitializeDevices*. Jedes Gerät bekommt damit auch einen Verweis auf die geöffnete *GpioConnection*. Der *DeviceController* besitzt eine öffentliche Methode, welche auf das im vorhergehenden Teil beschrieben *ReceivedDeviceMessage* Event des *MqttControllers* angemeldet werden kann. Wie vorher schon

[14] https://github.com/raspberry-sharp/raspberry-sharp-io

beschrieben, wird dies ausgelöst, sobald eine neue Nachricht auf einem abonnierten Topic eintrifft. Wird nun das Event ausgelöst, so wird das *MqttDeviceEventArgs* Objekt aufgeteilt und an die Private Methode *ControlDeviceByMqttMessage* übergeben. Diese sucht in der vorher angelegten Liste aller Geräte nach dem zugehörigen Gerät. Danach wird anhand der Enumeration *deviceAction* in einer switch/case Anweisung entschieden, welche Aktion ausgeführt werden muss, und ob diese mit dem angegebenen Geräte-Typ kompatibel ist. Ist dies der Fall, so wird die vom Gerät implementierte Methode zur zugehörigen Aktion ausgeführt. Die Gerätetypen werden im folgenden Abschnitt näher beschrieben.

3.2.3 Die Geräteklassen

Ziel bei der Implementierung der Software war es sie so allgemein wie möglich zu gestalten, dass auch weitere über GPIO steuerbare Geräte einfach Implementiert und hinzugefügt werden können. Hierfür wurden einige Schnittstellen eingeführt, welche im Folgenden dargestellt sind:

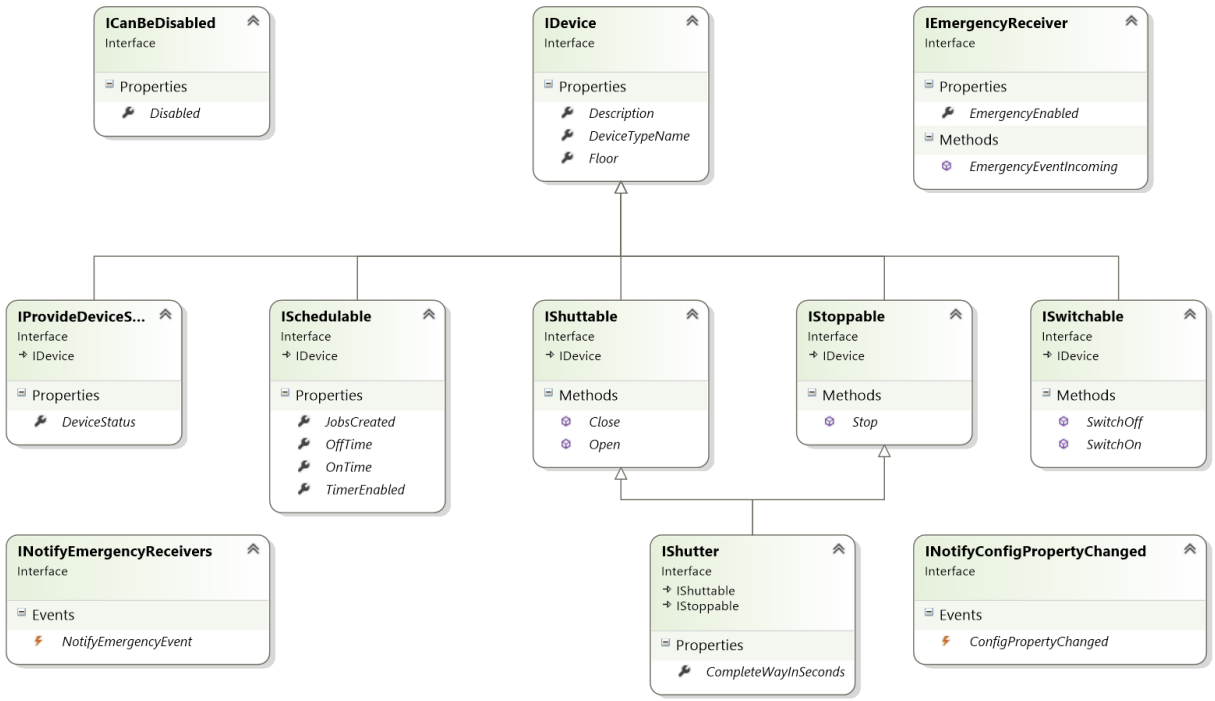


Abbildung 6- Schnittstellen für die Geräteklassen

Eine Geräteklasse, zum Beispiel einen Rollladen, muss nun nur noch die Schnittstellen implementieren, welche für sie von Nöten sind. Des Weiteren wurde eine abstrakte Klasse *GpioDevice* erstellt, die von allen Geräten implementiert werden muss, welche über GPIO-Pins steuerbar sind.

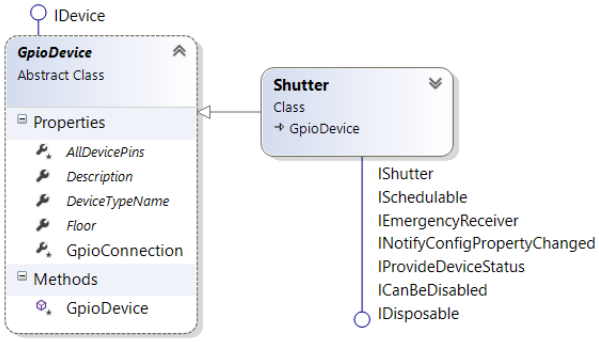


Abbildung 7- Klassendiagramm für die Rollladen-Klasse

Die Allgemeinste aller Schnittstellen ist die *IDevice* Schnittstelle, welche von allen Geräten implementiert werden muss, da die Liste im *DeviceController* nur solche Geräte aufnimmt. Ein Gerät, welches zum Beispiel *IShuttable* implementiert hat somit automatisch zwei Methoden, welche für das Öffnen und Schließen des Gerätes zuständig sind. Die Implementierung für die *Close* Methode eines Rollladens sieht folgendermaßen aus:

```
public void Open()
{
	if (!Disabled && GpioConnection.IsOpened)
	{
		StartUpTimer();
		Logger.Info($"{Description} opens completely");
		DeviceStatus = DeviceStatus.Undefined;
		GpioConnection[_closePin] = false;
		GpioConnection[_openPin] = true;
	}
 }
```

Es wird zunächst sichergestellt, dass das Gerät nicht deaktiviert ist und die *GpioConnection* auch tatsächlich geöffnet ist. Ist diese Bedingung erfüllt so wird ein Timer gestartet, welcher so lange läuft wie der in der Konfigurationsdatei definierte Parameter *completeWayInSeconds*. Ein Logger, auf den nicht näher eingegangen wird, zeichnet dieses Ereignis auf. Der *DeviceStatus* wird auf Undefiniert gesetzt, da jetzt nichtmehr sicher ist, auf welcher Position sich der Rollladen befindet. Die GPIO Pins, welche auch in der Konfigurationsdatei eingestellt wurden, werden nun entsprechend gesetzt und der Rollladen fährt auf. Ist der Timer abgelaufen so wird die Methode *Stop* aufgerufen, welche dafür sorgt, dass die Relais abfallen und das Gerät anhält. Da der Rollladen die Schnittstelle *INotifyConfigPropertyChanged* implementiert, ist sichergestellt, dass der neue *DeviceStatus* auch beim steuernden Client ankommt. Die Schnittstelle *IEmergencyReceiver* ist dafür gedacht, dass wenn zum Beispiel noch ein

Windsensor angeschlossen wird, wessen Klasse die Schnittstelle *INotifyEmergencyReceivers* implementiert, das Gerät automatisch mit der Methode *EmergencyEventIncoming* gesteuert werden kann.

3.2.3 Die Klasse MainApplication

Um nun alle Klassen zusammenzuführen und miteinander zu verbinden dient die Klasse *MainApplication,* welche in jedem lauffähigen C# Projekt enthalten sein muss. Diese ist sozusagen mit der statischen Methode *Main* Haupteinstiegspunkt sobald die Applikation ausgeführt wird. In der *Main* Methode wird zunächst überprüft, ob der Benutzer welcher die Applikation auf dem RaspberryPi ausführt auch die nötigen Rechte (root-Rechte) besitzt, da dies vom *GpioConnection* Treiber gefordert ist. Danach wird überprüft, ob im Verzeichnis der Applikation eine Konfigurationsdatei mit dem Namen *config.json* vorhanden ist. Sind alle Bedingungen gegeben, so wird nun die Applikation initialisiert. Die Methode *InitializeLogger* initialisiert wie der Name schon vermuten lässt den Logger. Die jedoch wichtigere Methode ist die Methode *InitializeApplication*. In dieser werden nun die Instanzen der vorher behandelten Klassen angelegt und miteinander bekannt gemacht. Des Weiteren werden alle Geräte welche die Schnittstelle *INotifyConfigPropertyChanged* implementieren am *ConfigurationController* angemeldet.

4 Zusammenfassung

Im Grundlagenteil dieses Assignments wurde auf die für die Aufgabe nötigen Technologien eingegangen. Hierbei wurde zunächst ein grober Überblick über das sehr umfangreiche .NET Framework erstellt und die zugehörige Programmiersprache C#. Danach wurde der RaspberryPi vorgestellt, welcher hier als Server und MQTT-Broker für die AlmueRaspi Software dient. Die Funktionsweise des MQTT-Protokolls wurde im Anschluss betrachtet und grundlegend erläutert. Im Hauptteil wurde die gesamte Steuerung Entwickelt. Im ersten Teil davon wurde kurz auf die benötigte Hardware eingegangen, da das Hauptaugenmerk dieses Assignments auf der Softwareentwicklung liegt. Danach wurden die wichtigsten Klassen der AlmueRaspi Software erklärt und ihre Funktionsweisen detailliert erläutert. Zuerst wurde hierfür der *ConfigurationController* betrachtet, welcher die ganze Arbeit übernimmt die die JSON-Konfigurationsdatei betrifft. Der *MqttController*, welcher die Kommunikation mit dem MQTT-

Broker übernimmt, wurde daraufhin grundlegend erklärt. Zur tatsächlichen Steuerung der Geräte wurde die Klasse *DeviceController* erstellt, welche die Kommunikation mit dem GPIO-Treiber des Linux-Kernels übernimmt. Um die Software so allgemein wie möglich zu gestalten, wurde im Anschluss an dem Beispiel eines Rollladens gezeigt, wie sich dieses Gerät durch die Vielzahl von bereitgestellter Interfaces einfach in die Software integrieren lässt. Die Klasse *MainApplication* welche alle Klassen zusammenführt, wurde zum Schluss noch einmal kurz erläutert. Die gesamte Software, sowohl AlmueRaspi als auch AlmueAndroid, wurden unter der MIT-Lizenz auf https://github.com/he4d/Almue frei verfügbar gemacht. Zum Abschluss werden noch zwei Screenshots der Android-App präsentiert, welche ihre Oberfläche dynamisch anhand der in der Konfigurationsdatei angegebenen Geräte aufbaut.

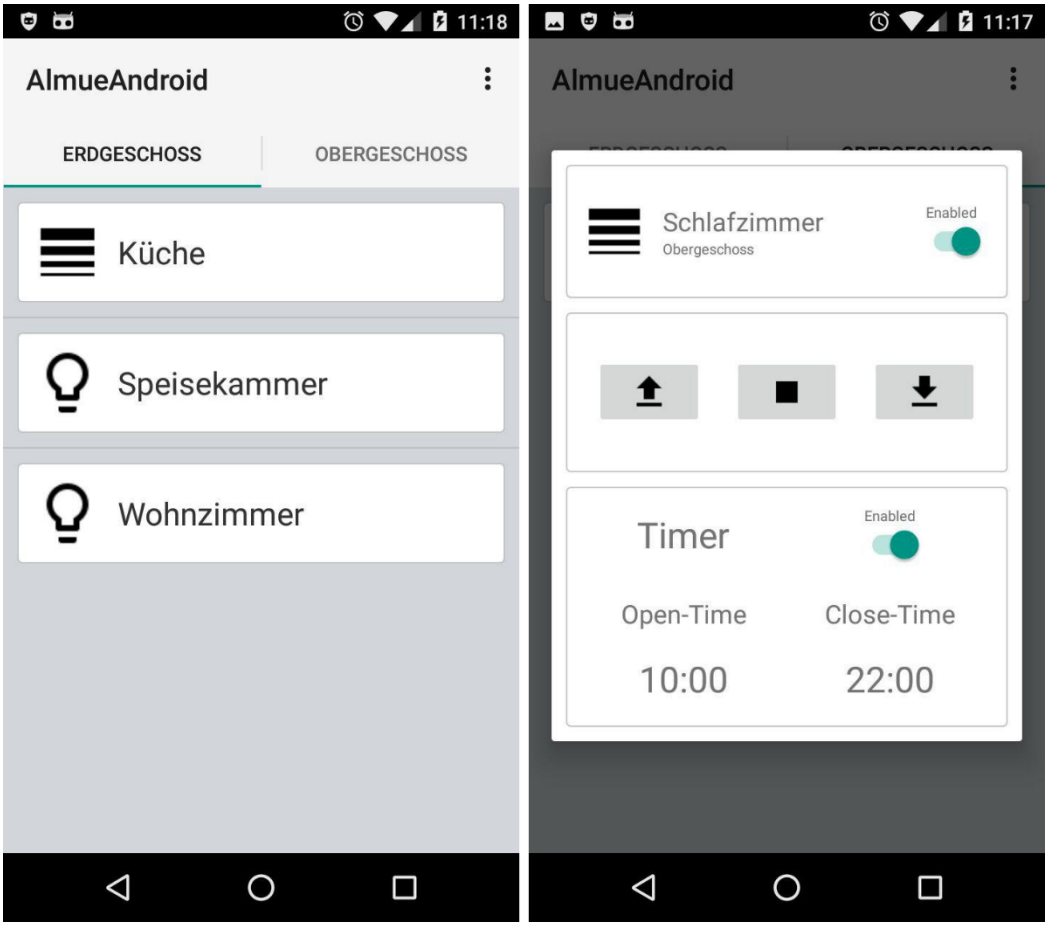


Abbildung 8- Die AlmueAndroid App

5 Literaturverzeichnis

IBM und WebSphere. (2014). *Building Real-time Mobile Solutions with MQTT and IBM MessageSight* (First Edition Ausg.). IBM Corp.

Kühnle, A. (2016). *C#6 mit Visual Studio 2015* (7., aktualisierte und erweiterte Auflage 2016 Ausg.). Bonn: Rheinwerk Verlag GmbH.

Linux Kernel Organization, I. (09. 11 2015). *The Linux Kernel Archives.* Von https://www.kernel.org/doc/Documentation/gpio/sysfs.txt abgerufen

Linux Kernel Organization, I. (27. 10 2016). *The Linux Kernel Archives.* Von https://www.kernel.org/doc/Documentation/gpio/gpio-legacy.txt abgerufen

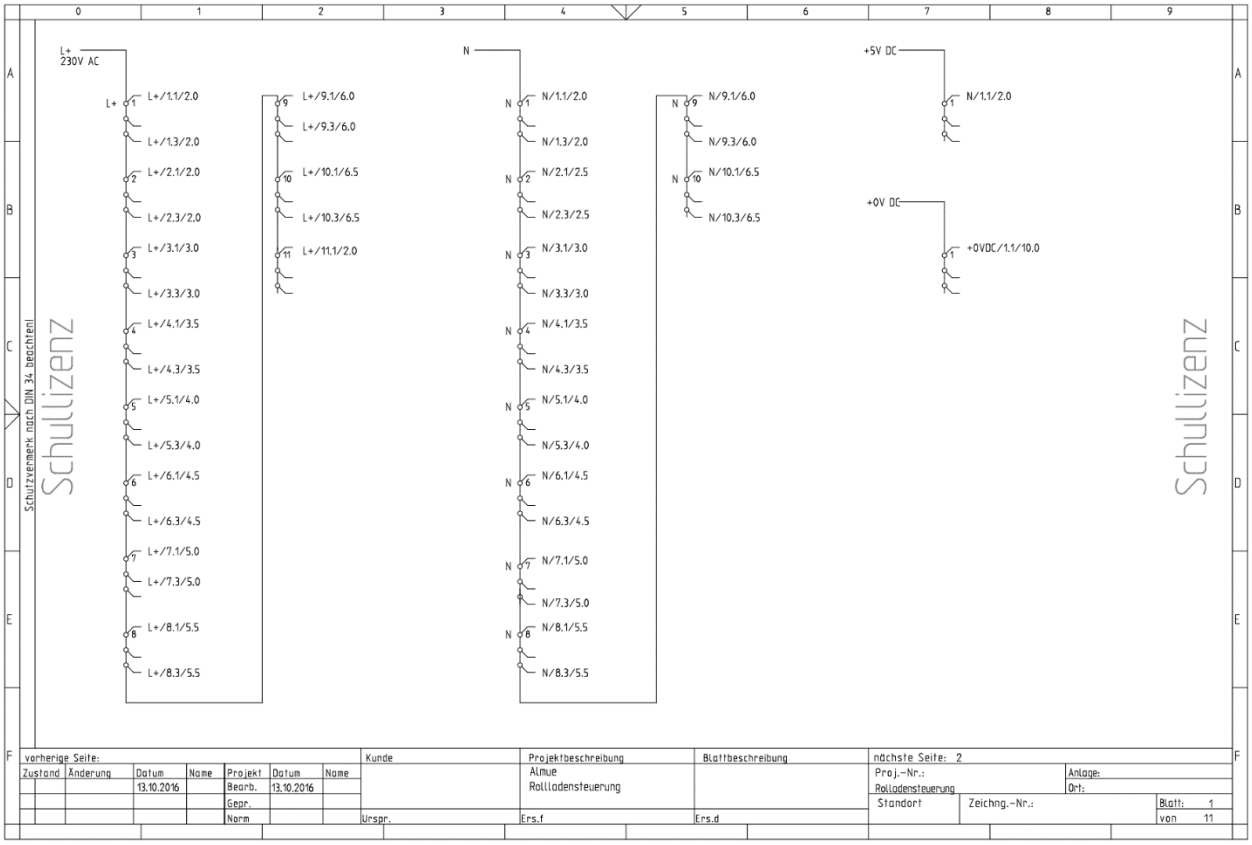
6 Abbildungsverzeichnis

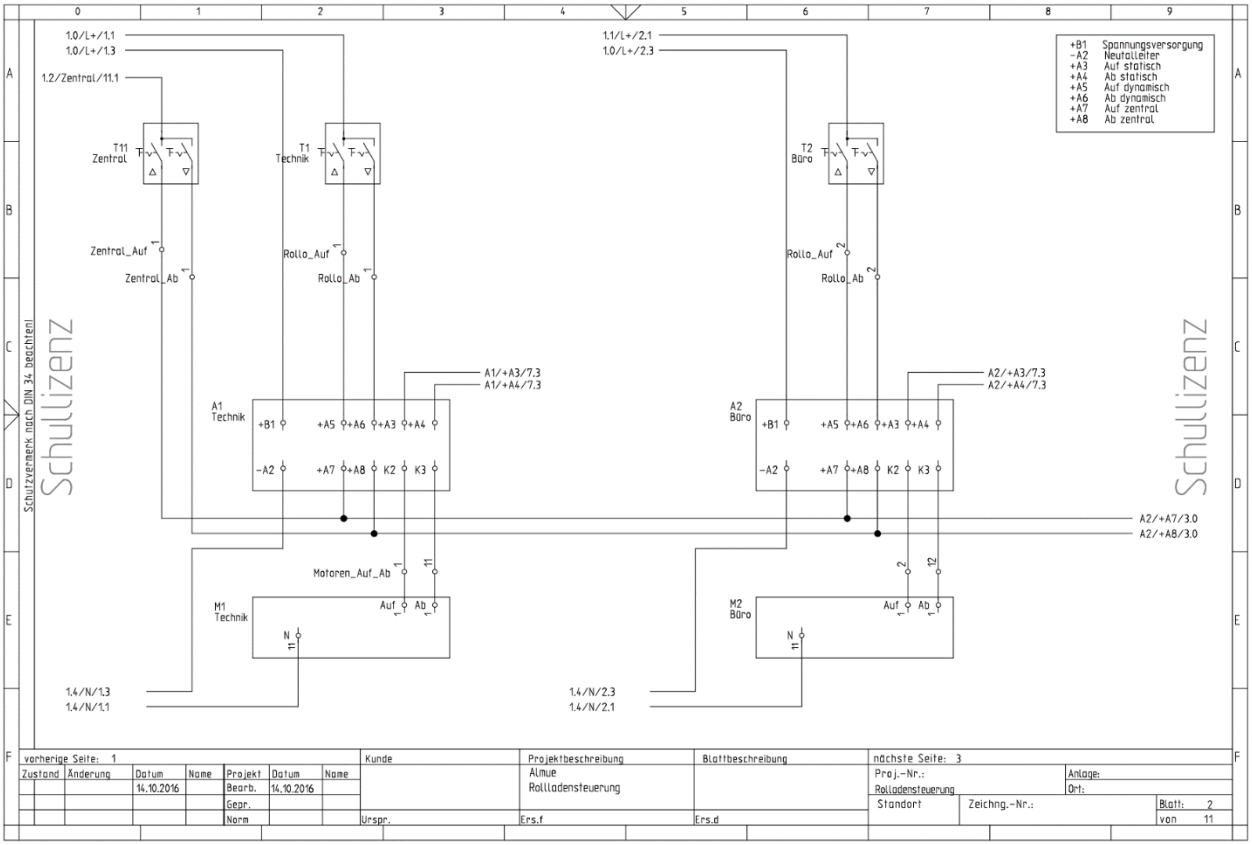
7 Anhang

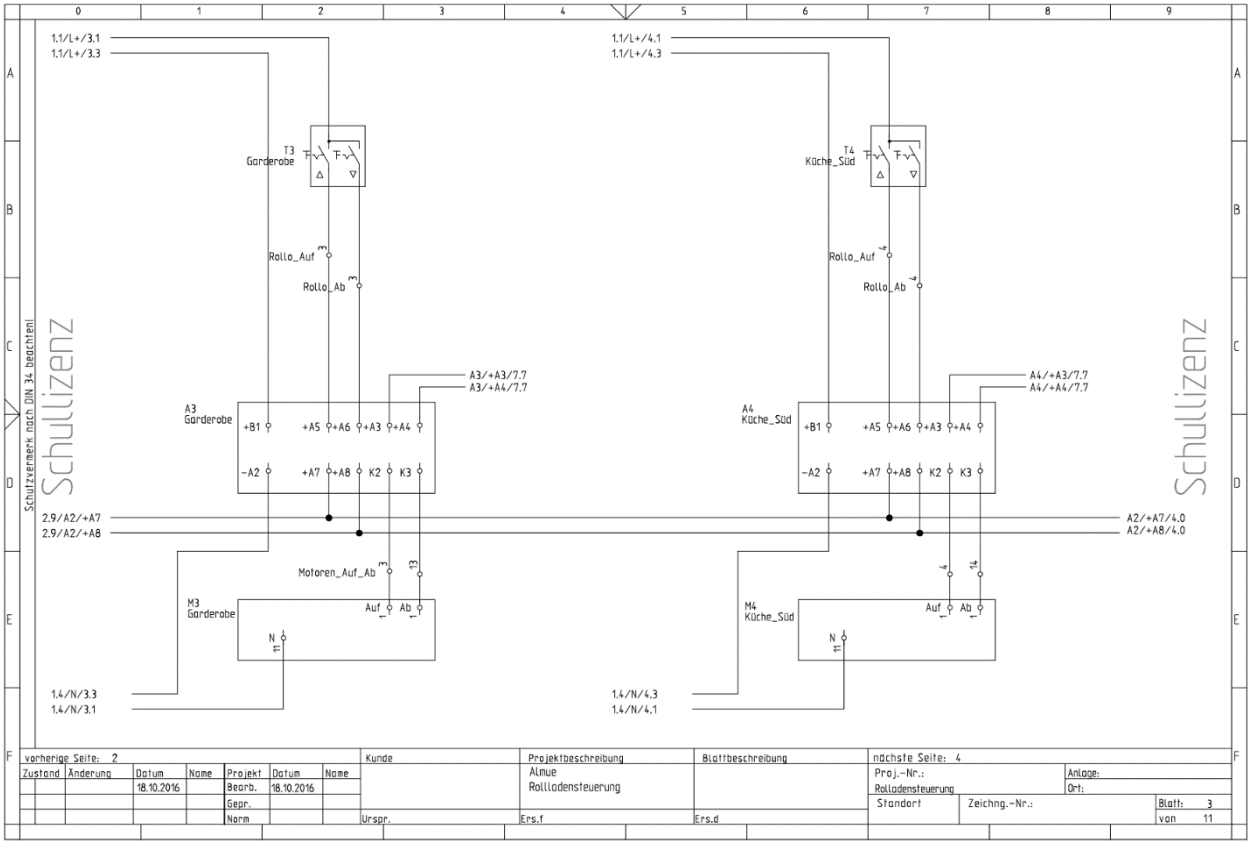
Anhang 1 – Beispielkonfiguationsdatei

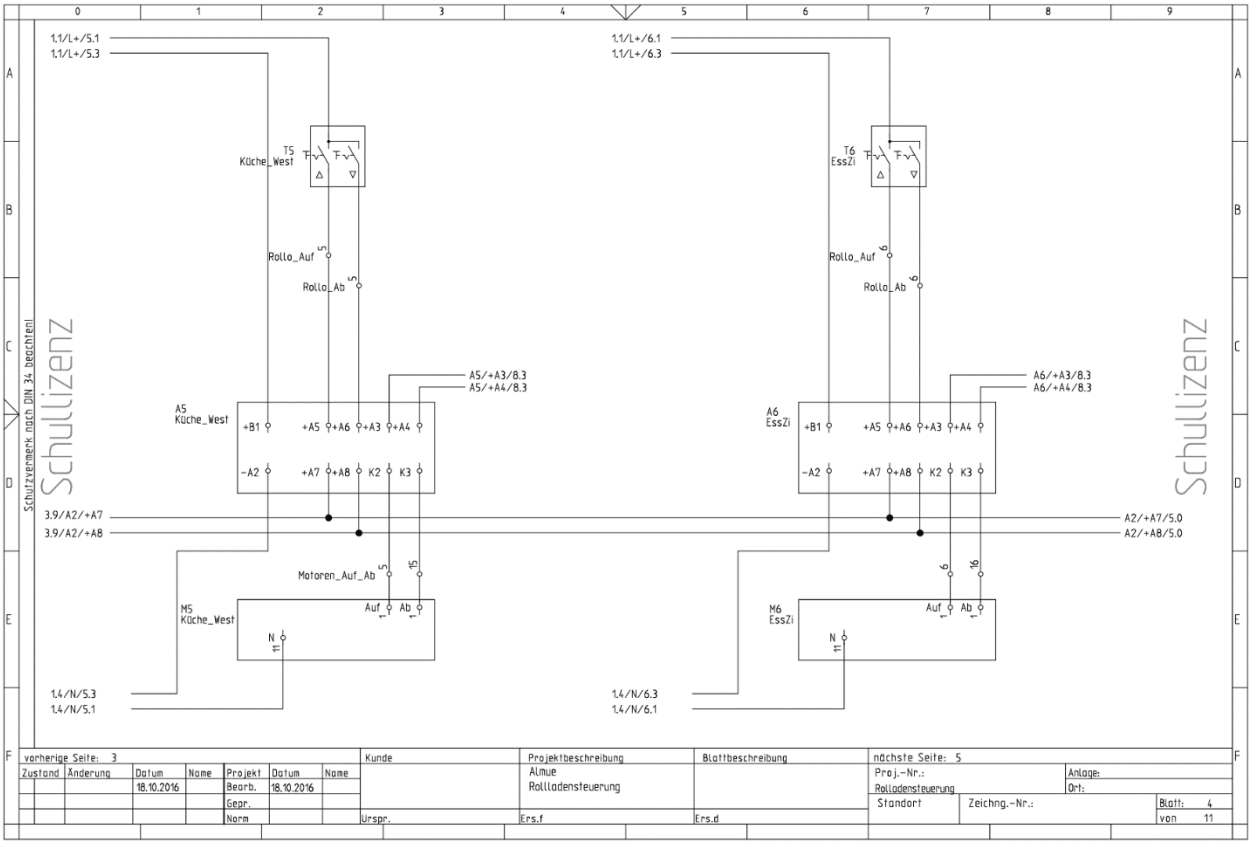

```
{
  "broker": {
    "ip": "127.0.0.1",
    "port": 1883,
    "user": "mustermann",
    "password": "testpw123"
  },
  "devices": {
    "shutters": [
      {
        "description": "Technik",
        "floor": "Keller",
        "openPin": "P1Pin40",
        "closePin": "P1Pin37",
        "completeWayInSeconds": 10,
        "emergencyEnabled": false,
        "disabled": false,
        "timerEnabled": true,
        "openTime": "17:20:00",
        "closeTime": "17:25:00",
        "deviceStatus": "Undefined"
      },
      {
        "description": "Büro",
        "floor": "Keller",
        "openPin": "P1Pin38",
        "closePin": "P1Pin35",
        "completeWayInSeconds": 12,
        "emergencyEnabled": false,
        "disabled": false,
        "timerEnabled": true,
        "openTime": "20:15:00",
        "closeTime": "23:50:00",
        "deviceStatus": "Undefined"
      },
      {
        "description": "Wohnzimmer",
        "floor": "Erdgeschoss",
        "openPin": "P1Pin36",
        "closePin": "P1Pin33",
        "completeWayInSeconds": 12,
        "emergencyEnabled": false,
        "disabled": false,
        "timerEnabled": false,
        "openTime": "20:15:00",
        "closeTime": "23:50:00",
        "deviceStatus": "Undefined"
      }
    ],
    "lightings": [
    ],
    "windMonitors": [
    ]
  }
```

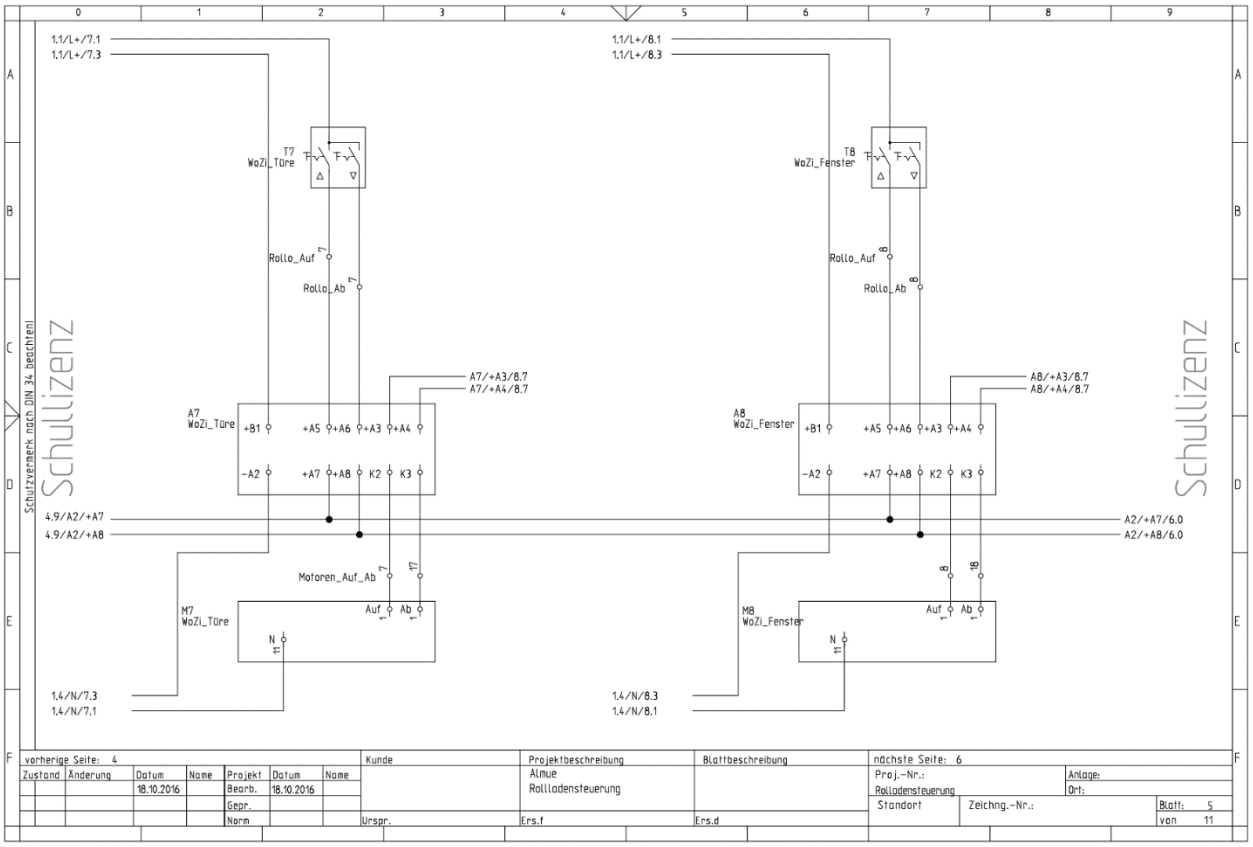
Anhang 2 – Hardware-Schaltplan

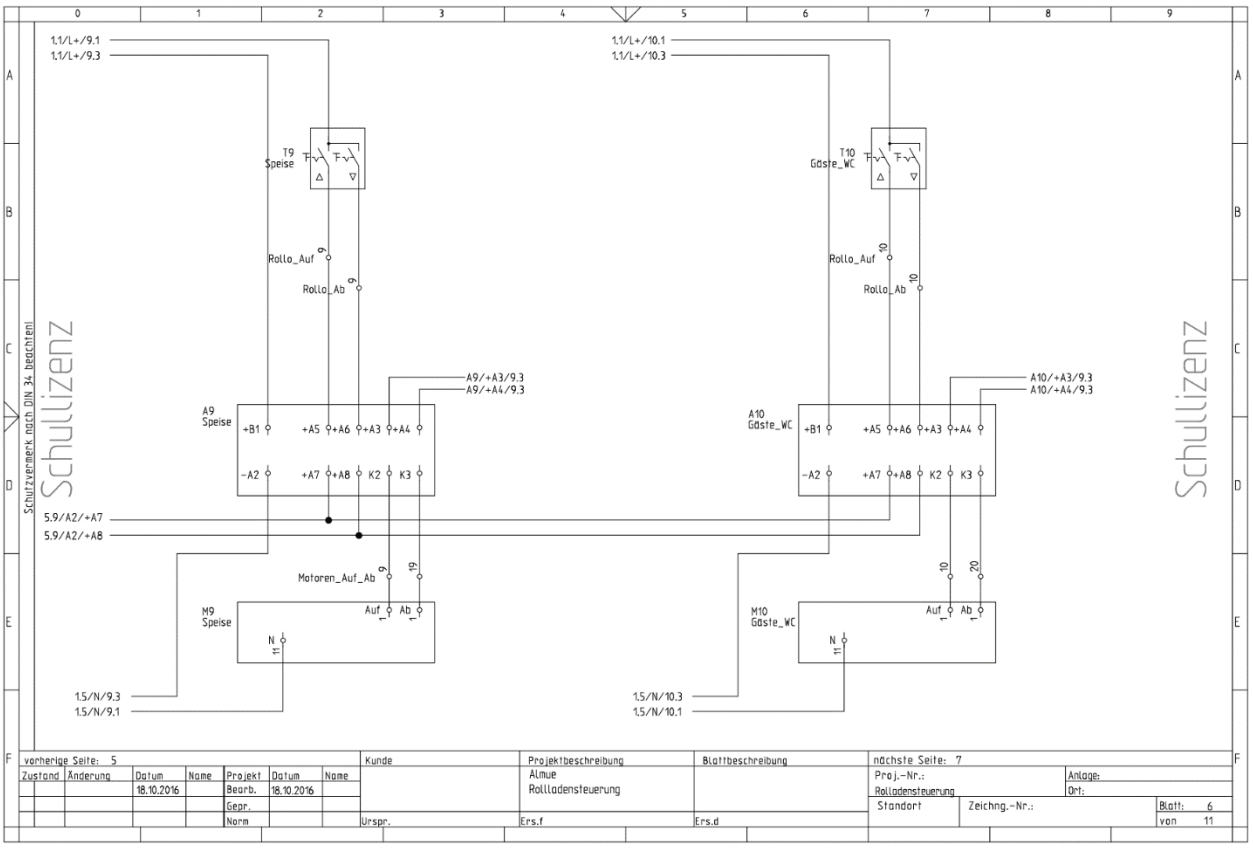


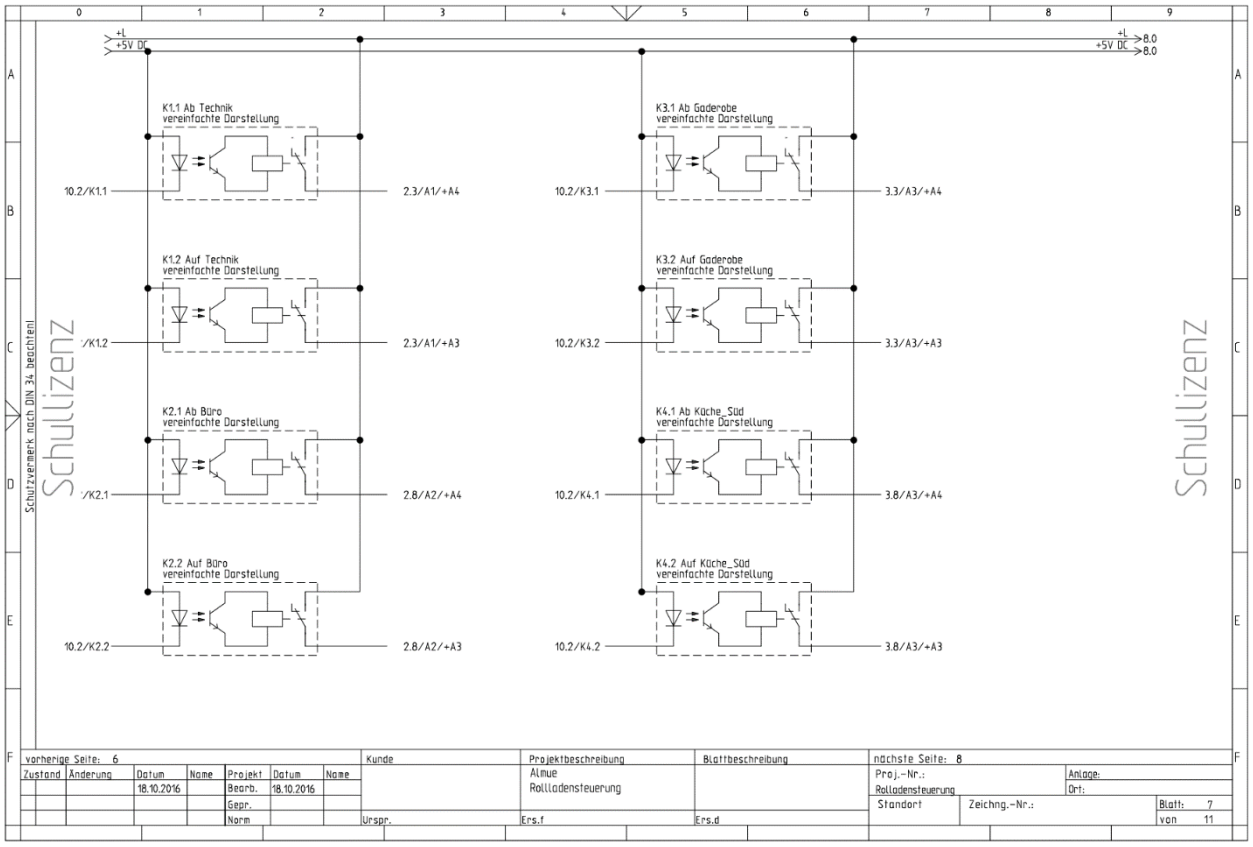


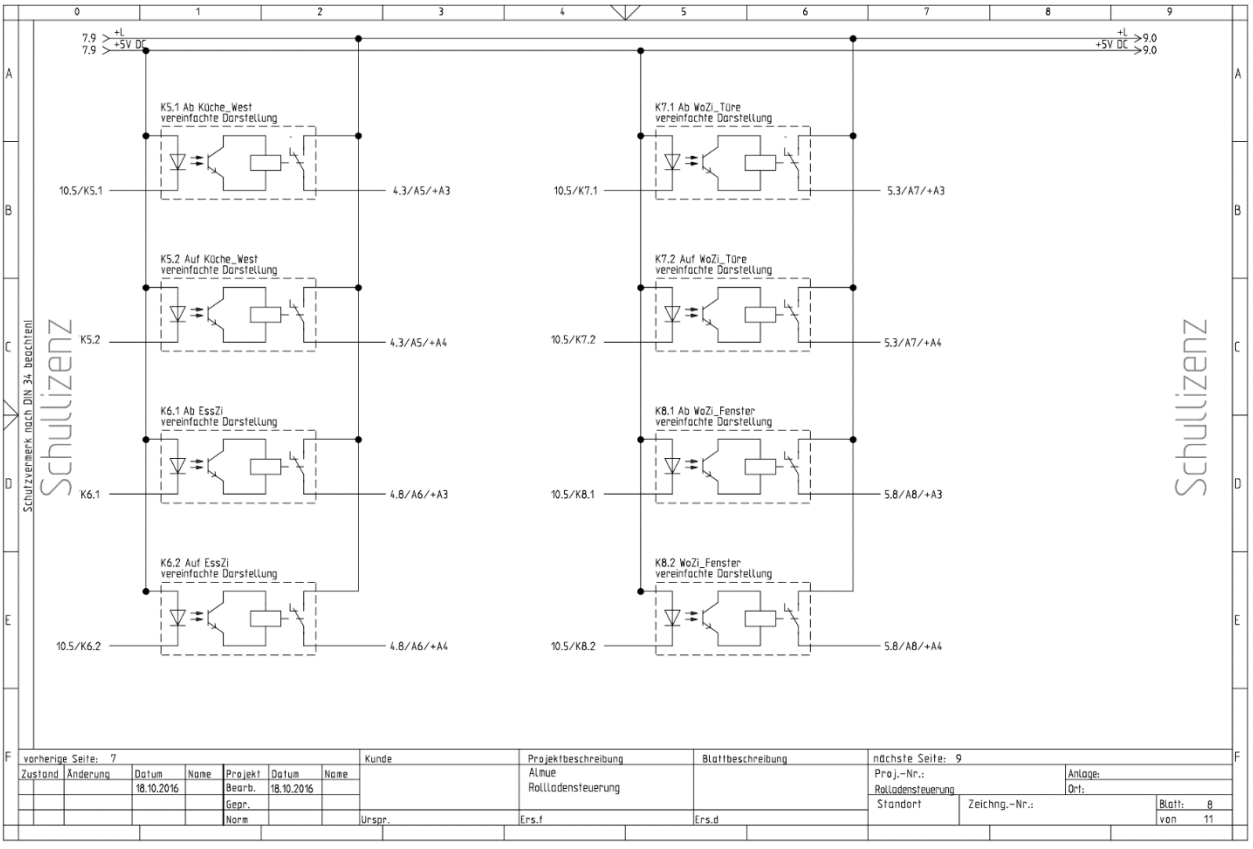

1.1/L+/3.1
1.1/L+/3.3
T3
Garderobe
Rollo_Auf
Rollo_Ab
A3/+A3/7.7
A3/+A4/7.7
A3
Garderobe
+B1
+A5
+A6
+A3
+A4
-A2
+A7
+A8
K2
K3
2.9/A2/+A7
2.9/A2/+A8
Motoren_Auf_Ab
M3
Garderobe
Auf
Ab
N
1.4/N/3.3
1.4/N/3.1
1.1/L+/4.1
1.1/L+/4.3
T4
Küche_Süd
A4/+A3/7.7
A4/+A4/7.7
A4
Küche_Süd
A2/+A7/4.0
A2/+A8/4.0
M4
Küche_Süd
1.4/N/4.3
1.4/N/4.1
Schutzvermerk nach DIN 34 beachten!
Schullizenz
vorherige Seite: 2
nächste Seite: 4
Kunde
Projektbeschreibung
Almue
Rollladensteuerung
Blattbeschreibung
Proj.-Nr.:
Anlage:
Ort:
Standort
Zeichng.-Nr.:
Blatt: 3
von 11

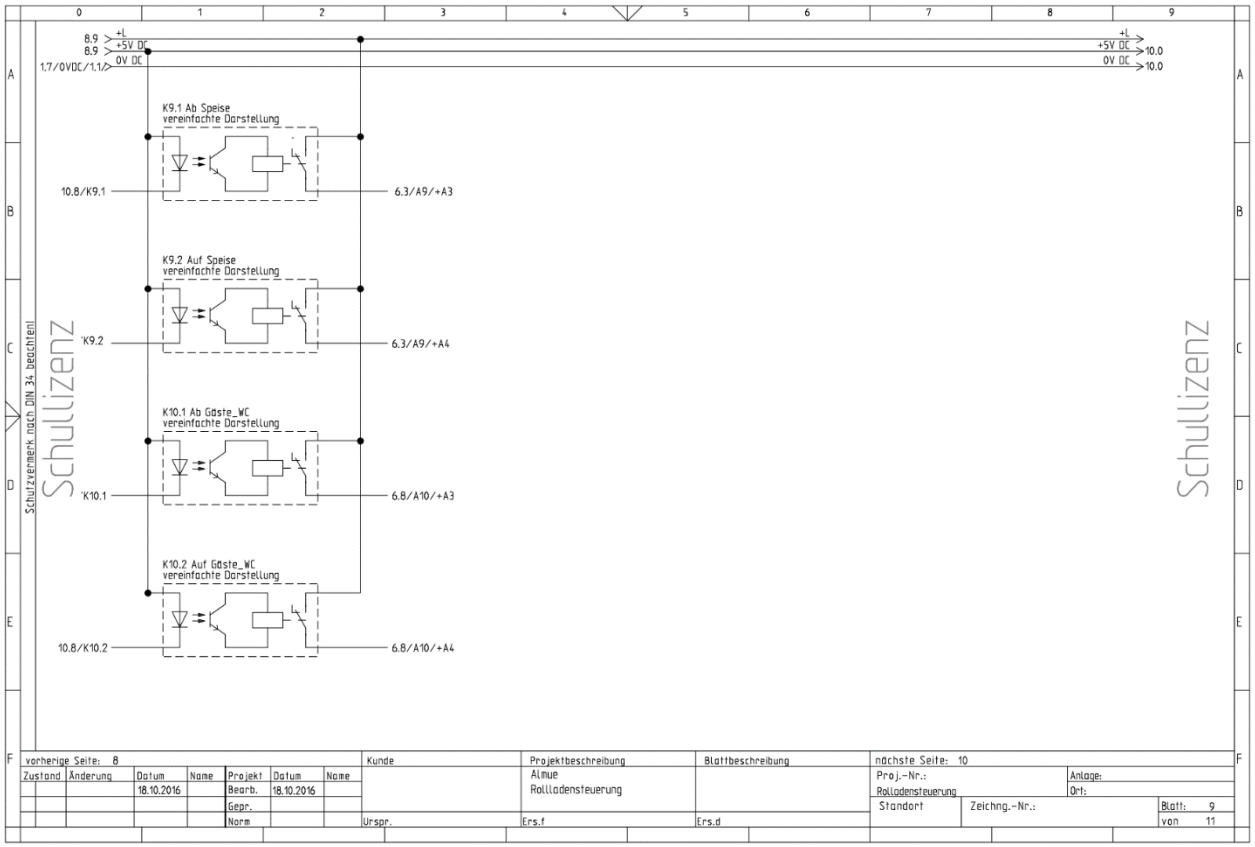

1.1/L+/5.1
1.1/L+/5.3
T5
Küche_West
Rollo_Auf
Rollo_Ab
A5/+A3/8.3
A5/+A4/8.3
A5
Küche_West
+B1
+A5
+A6
+A3
+A4
-A2
+A7
+A8
K2
K3
3.9/A2/+A7
3.9/A2/+A8
Motoren_Auf_Ab
M5
Küche_West
Auf
Ab
N
1.4/N/5.3
1.4/N/5.1
1.1/L+/6.1
1.1/L+/6.3
T6
EssZi
A6/+A3/8.3
A6/+A4/8.3
A6
EssZi
A2/+A7/5.0
A2/+A8/5.0
M6
EssZi
1.4/N/6.3
1.4/N/6.1
Schutzvermerk nach DIN 34 beachten!
Schullizenz
vorherige Seite: 3
nächste Seite: 5
Kunde
Projektbeschreibung
Almue
Rollladensteuerung
Blattbeschreibung
Proj.-Nr.:
Anlage:
Ort:
Standort
Zeichng.-Nr.:
Blatt: 4
von 11

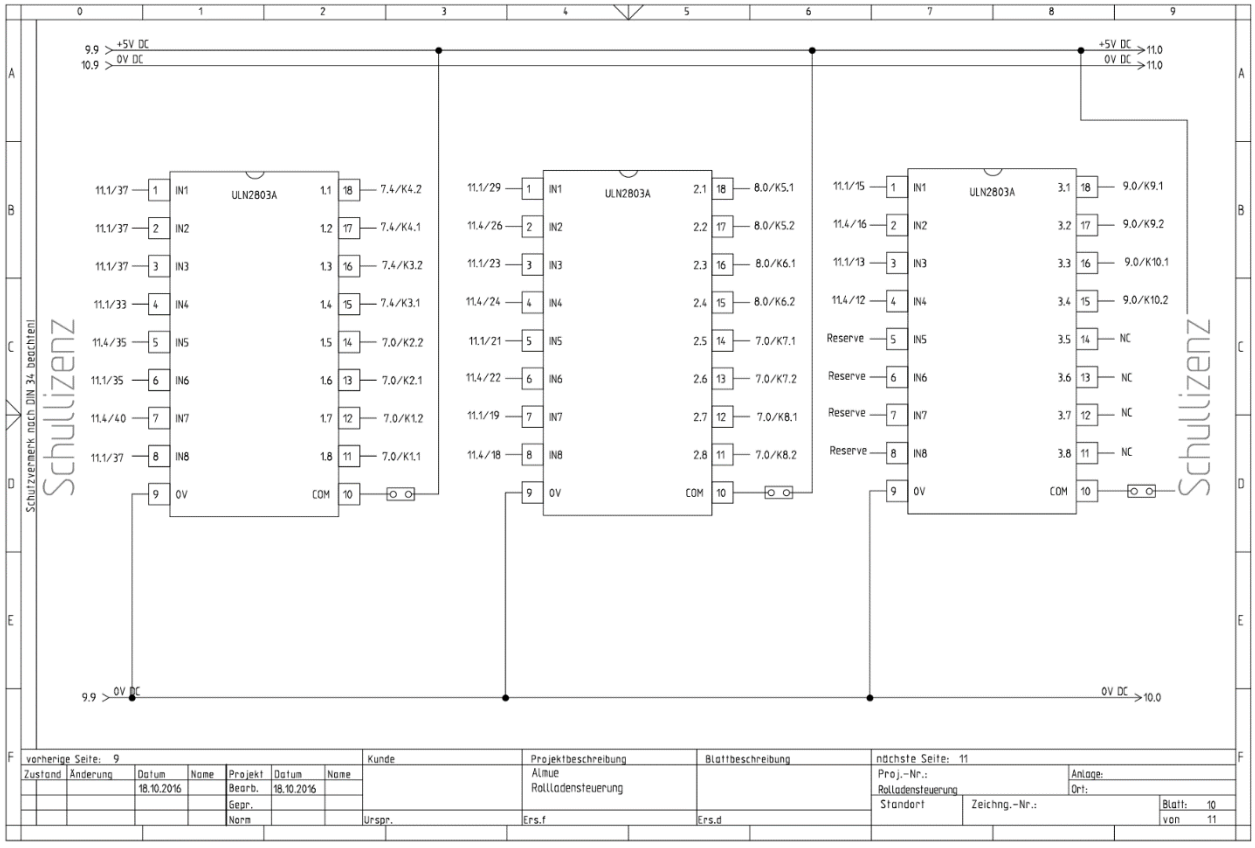

1.1/L+/7.1
1.1/L+/7.3
T7
WoZi_Türe
Rollo_Auf
Rollo_Ab
A7/+A3/8.7
A7/+A4/8.7
A7
WoZi_Türe
+B1
+A5
+A6
+A3
+A4
-A2
+A7
+A8
K2
K3
4.9/A2/+A7
4.9/A2/+A8
Motoren_Auf_Ab
M7
WoZi_Türe
Auf
Ab
N
1.4/N/7.3
1.4/N/7.1
1.1/L+/8.1
1.1/L+/8.3
T8
WoZi_Fenster
A8/+A3/8.7
A8/+A4/8.7
A8
WoZi_Fenster
A2/+A7/6.0
A2/+A8/6.0
M8
WoZi_Fenster
1.4/N/8.3
1.4/N/8.1
Schullizenz
Schutzvermerk nach DIN 34 beachten!
vorherige Seite: 4
nächste Seite: 6
Kunde
Projektbeschreibung
Almue
Rollladensteuerung
Blattbeschreibung
Proj.-Nr.:
Anlage:
Ort:
Standort
Zeichng.-Nr.:
Blatt: 5
von 11
Zustand
Änderung
Datum
Name
Projekt
Bearb.
Gepr.
Norm
18.10.2016
Ursprung
Ers.f
Ers.d

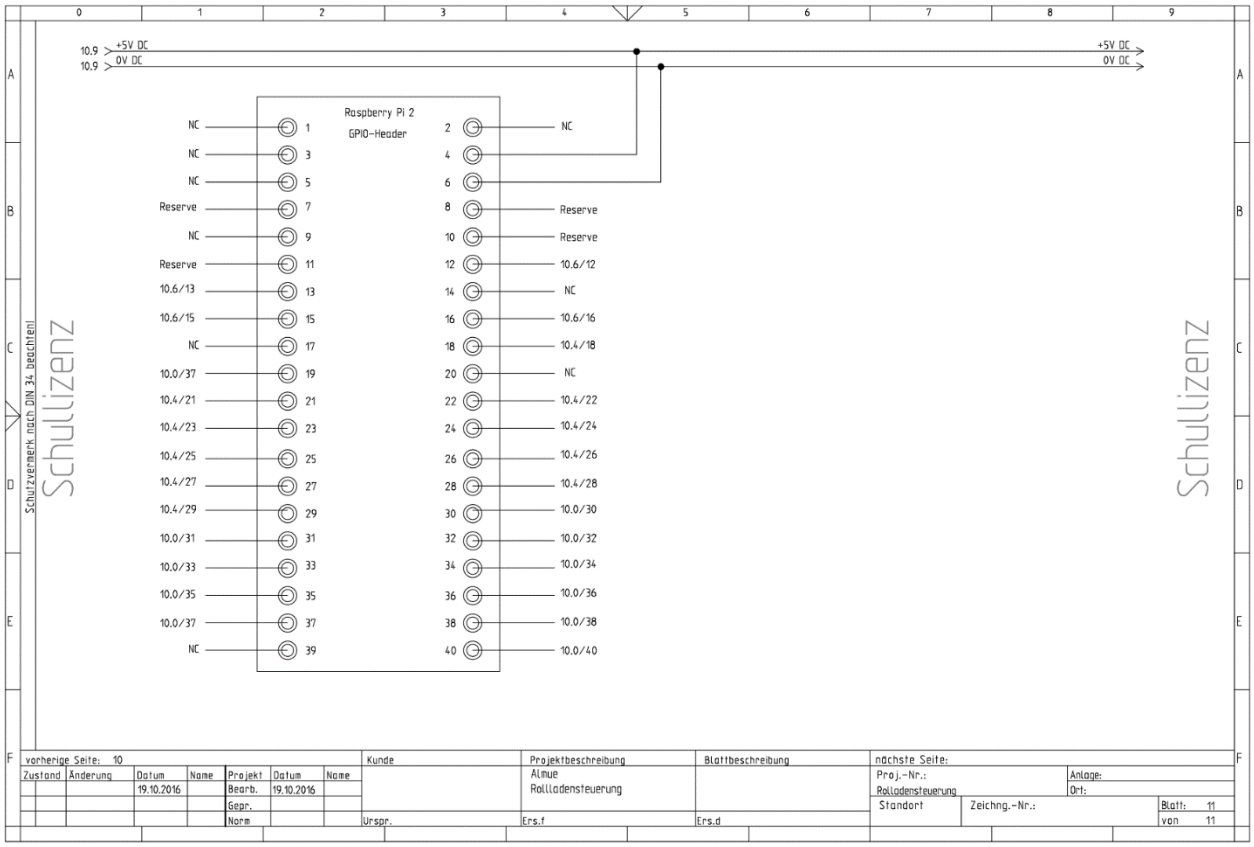

1.1/L+/9.1
1.1/L+/9.3
T9
Speise
Rollo_Auf
Rollo_Ab
A9/+A3/9.3
A9/+A4/9.3
A9
Speise
+B1
+A5
+A6
+A3
+A4
-A2
+A7
+A8
K2
K3
5.9/A2/+A7
5.9/A2/+A8
Motoren_Auf_Ab
M9
Speise
Auf
Ab
N
1.5/N/9.3
1.5/N/9.1
1.1/L+/10.1
1.1/L+/10.3
T10
Gäste_WC
A10/+A3/9.3
A10/+A4/9.3
A10
Gäste_WC
M10
Gäste_WC
1.5/N/10.3
1.5/N/10.1
Schullizenz
Schutzvermerk nach DIN 34 beachten!
vorherige Seite: 5
nächste Seite: 7
Kunde
Projektbeschreibung
Almue
Rollladensteuerung
Blattbeschreibung
Proj.-Nr.:
Anlage:
Ort:
Standort
Zeichng.-Nr.:
Blatt: 6
von 11
Zustand
Änderung
Datum
Name
Projekt
Bearb.
Gepr.
Norm
18.10.2016
Ursprung
Ers.f
Ers.d


+L
+5V DC
+L
+5V DC
8.0
8.0
K1.1 Ab Technik
vereinfachte Darstellung
10.2/K1.1
2.3/A1/+A4
K1.2 Auf Technik
vereinfachte Darstellung
/K1.2
2.3/A1/+A3
K2.1 Ab Büro
vereinfachte Darstellung
/K2.1
2.8/A2/+A4
K2.2 Auf Büro
vereinfachte Darstellung
10.2/K2.2
2.8/A2/+A3
K3.1 Ab Gaderobe
vereinfachte Darstellung
10.2/K3.1
3.3/A3/+A4
K3.2 Auf Gaderobe
vereinfachte Darstellung
10.2/K3.2
3.3/A3/+A3
K4.1 Ab Küche_Süd
vereinfachte Darstellung
10.2/K4.1
3.8/A3/+A4
K4.2 Auf Küche_Süd
vereinfachte Darstellung
10.2/K4.2
3.8/A3/+A3
Schutzvermerk nach DIN 34 beachten!
Schullizenz
Schullizenz
vorherige Seite: 6
nächste Seite: 8
Kunde
Projektbeschreibung
Almue
Rollladensteuerung
Blattbeschreibung
Proj.-Nr.:
Anlage:
Rollladensteuerung
Ort:
Standort
Zeichng.-Nr.:
Blatt: 7
von 11
Zustand
Änderung
Datum
Name
Projekt
Bearb.
Gepr.
Norm
18.10.2016
Urspr.
Ers.f
Ers.d

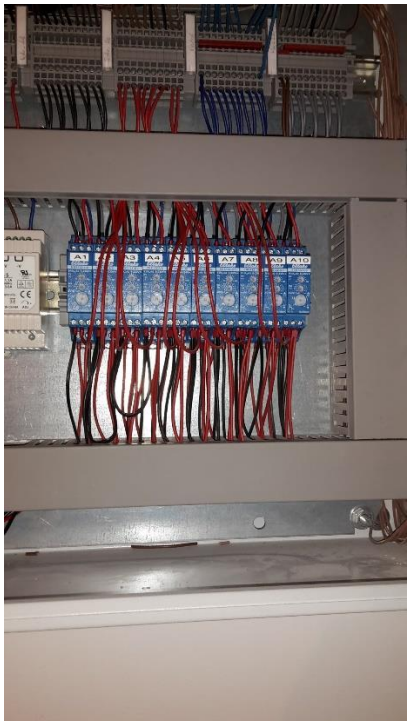

7.9
7.9
+L
+5V DC
+L
+5V DC
9.0
9.0
K5.1 Ab Küche_West
vereinfachte Darstellung
10.5/K5.1
4.3/A5/+A3
K5.2 Auf Küche_West
vereinfachte Darstellung
K5.2
4.3/A5/+A4
K6.1 Ab EssZi
vereinfachte Darstellung
K6.1
4.8/A6/+A3
K6.2 Auf EssZi
vereinfachte Darstellung
10.5/K6.2
4.8/A6/+A4
K7.1 Ab WoZi_Türe
vereinfachte Darstellung
10.5/K7.1
5.3/A7/+A3
K7.2 Auf WoZi_Türe
vereinfachte Darstellung
10.5/K7.2
5.3/A7/+A4
K8.1 Ab WoZi_Fenster
vereinfachte Darstellung
10.5/K8.1
5.8/A8/+A3
K8.2 WoZi_Fenster
vereinfachte Darstellung
10.5/K8.2
5.8/A8/+A4
Schutzvermerk nach DIN 34 beachten!
Schullizenz
Schullizenz
vorherige Seite: 7
nächste Seite: 9
Kunde
Projektbeschreibung
Almue
Rollladensteuerung
Blattbeschreibung
Proj.-Nr.:
Anlage:
Rollladensteuerung
Ort:
Standort
Zeichng.-Nr.:
Blatt: 8
von 11
Zustand
Änderung
Datum
Name
Projekt
Bearb.
Gepr.
Norm
18.10.2016
Urspr.
Ers.f
Ers.d

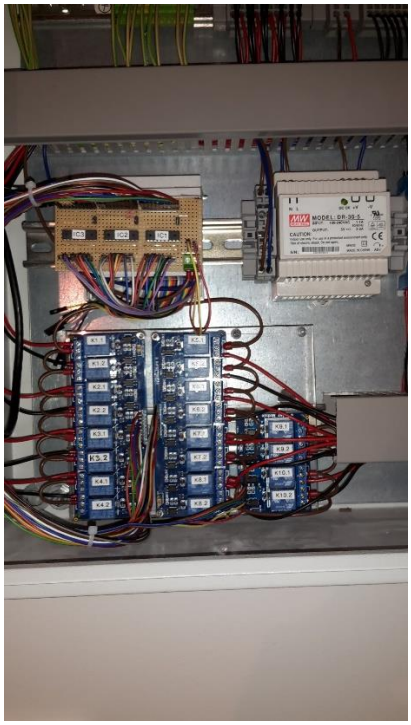

8.9 +L
8.9 +5V DC
1.7/0VDC/1.1/ 0V DC
+L
+5V DC
0V DC
10.0
10.0
K9.1 Ab Speise
vereinfachte Darstellung
10.8/K9.1
6.3/A9/+A3
K9.2 Auf Speise
vereinfachte Darstellung
'K9.2
6.3/A9/+A4
K10.1 Ab Gäste_WC
vereinfachte Darstellung
'K10.1
6.8/A10/+A3
K10.2 Auf Gäste_WC
vereinfachte Darstellung
10.8/K10.2
6.8/A10/+A4
Schutzvermerk nach DIN 34 beachten!
Schullizenz
Schullizenz
vorherige Seite: 8
nächste Seite: 10
Kunde
Projektbeschreibung
Almue
Rollladensteuerung
Blattbeschreibung
Proj.-Nr.:
Anlage:
Ort:
Standort
Zeichng.-Nr.:
Blatt: 9
von 11
Zustand Änderung Datum Name
18.10.2016
Projekt Bearb. Datum 18.10.2016 Name
Gepr.
Norm
Urspr.
Ers.f
Ers.d

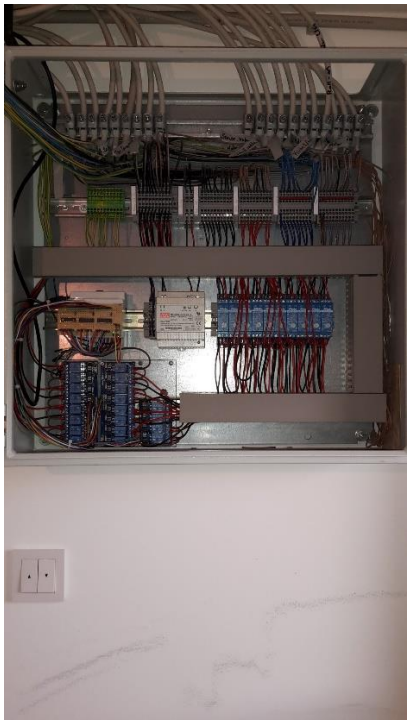

9.9 +5V DC
10.9 0V DC
+5V DC
0V DC
11.0
11.0
ULN2803A
11.1/37 IN1 1.1 7.4/K4.2
11.1/37 IN2 1.2 7.4/K4.1
11.1/37 IN3 1.3 7.4/K3.2
11.1/33 IN4 1.4 7.4/K3.1
11.4/35 IN5 1.5 7.0/K2.2
11.1/35 IN6 1.6 7.0/K2.1
11.4/40 IN7 1.7 7.0/K1.2
11.1/37 IN8 1.8 7.0/K1.1
9 0V COM 10
11.1/29 IN1 2.1 8.0/K5.1
11.4/26 IN2 2.2 8.0/K5.2
11.1/23 IN3 2.3 8.0/K6.1
11.4/24 IN4 2.4 8.0/K6.2
11.1/21 IN5 2.5 7.0/K7.1
11.4/22 IN6 2.6 7.0/K7.2
11.1/19 IN7 2.7 7.0/K8.1
11.4/18 IN8 2.8 7.0/K8.2
9 0V COM 10
11.1/15 IN1 3.1 9.0/K9.1
11.4/16 IN2 3.2 9.0/K9.2
11.1/13 IN3 3.3 9.0/K10.1
11.4/12 IN4 3.4 9.0/K10.2
Reserve IN5 3.5 NC
Reserve IN6 3.6 NC
Reserve IN7 3.7 NC
Reserve IN8 3.8 NC
9 0V COM 10
9.9 0V DC
0V DC 10.0
Schutzvermerk nach DIN 34 beachten!
Schullizenz
Schullizenz
vorherige Seite: 9
nächste Seite: 11
Kunde
Projektbeschreibung
Almue
Rollladensteuerung
Blattbeschreibung
Proj.-Nr.:
Anlage:
Ort:
Standort
Zeichng.-Nr.:
Blatt: 10
von 11
Zustand Änderung Datum Name
18.10.2016
Projekt Bearb. Datum 18.10.2016 Name
Gepr.
Norm
Urspr.
Ers.f
Ers.d



Anhang 3 – Bilder des Hardware-Aufbaus







D
21 200 401 - 2

Eltako ELECTRONICS

Stromstoß-Gruppenschalter für Zentralsteuerung
EGS12Z-8..230 V UC

CE

Diese Geräte dürfen nur durch eine Elektrofachkraft installiert werden, andernfalls besteht Brandgefahr oder Gefahr eines elektrischen Schlages!

Temperatur an der Einbaustelle:
-20°C bis +50°C.
Lagertemperatur: -25°C bis +70°C.
Relative Luftfeuchte: Jahresmittelwert <75%.

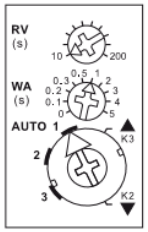
1+1 Schließer nicht potenzialfrei
16 A/250 V AC, für 1 Motor oder für Motor-Relais.
Stand-by-Verlust nur 0,05-0,4 Watt.
Reiheneinbaugerät für Montage auf Tragschiene DIN-EN 60715 TH35.
1 Teilungseinheit = 18 mm breit, 58 mm tief.

Dieser Stromstoß-Gruppenschalter setzt die Befehle der Sensorrelais oder von Schaltern und Tastern um und schaltet je nach Einstellung der Drehschalter auf der Frontseite einen Motor, Motor-Trennrelais MTR12-8..230 V UC oder DC-Motorrelais DCM12-8..230 V UC.
Versorgungs- und Schaltspannung 8..230 V UC an den Klemmen +B1/-A2. Die Steuerspannung an den Klemmen A3 bis A8 muss das gleiche Potenzial haben.

Das Funktionsprinzip dieses Stromstoß-Gruppen-schalters besteht darin, dass einerseits impulsweise 'auf, halt, ab, halt' (Kontakt 1 geschlossen - beide Kontakte geöffnet - Kontakt 2 geschlossen - beide Kontakte geöffnet) gesteuert werden kann, andererseits über zusätzliche Steuereingänge gezielt 'auf' oder 'ab' gewählt werden kann. **Dynamisch** bezeichnet Steuereingänge, bei denen ein Impuls von mind. 20 ms genügt, um einen Kontakt zu schließen. **Statisch** ist ein Steuereingang, bei dem der Kontakt nur solange geschlossen ist, wie der Steuerbefehl anliegt.

Die Bezeichnungen 'auf' und 'ab' gelten für Rolladen, Jalousien und Rollos. Bei Markisen ist
'auf' = aufwickeln, also einfahren, und
'ab' = abwickeln, also ausfahren.
Bei Fenstern ist 'auf' = auf wie öffnen und 'ab'= zu wie schließen.

Funktions-Drehschalter



AUTO 1 = In dieser Stellung des unteren Drehschalters ist die örtliche **Komfortwendefunktion für Jalousien eingeschaltet.** Bei der örtlichen Steuerung mit einem Taster an A3+A4 (mit einer Brücke verbinden) oder einem Doppeltaster an A5/A6 bewirkt ein Doppelimpuls das langsame Drehen in die Gegenrichtung, welches mit einem weiteren Impuls gestoppt wird.

AUTO 2 = In dieser Stellung des unteren Drehschalters ist die örtliche Komfortwendefunktion für Jalousien ganz ausgeschaltet.

AUTO 3 = In dieser Stellung des unteren Drehschalters ist die örtliche Komfortwendefunktion für Jalousien ebenfalls ausgeschaltet. Die bei AUTO 1 und AUTO 2 dynamischen Zentral-Steuereingänge A5 und A6 sind jedoch **zunächst statisch und lassen so das Wenden von Jalousien durch Tasten zu.** Erst nach 1 Sekunde ständiger Ansteuerung schalten sie auf dynamisch um.

▲▼= Die **Handsteuerung** erfolgt in den Positionen ▲ (auf) und ▼ (ab) des unteren Drehschalters. Die Handsteuerung hat Priorität vor allen anderen Steuerbefehlen.

WA = Die **Wendeautomatik** für Jalousien und Markisen wird mit dem mittleren Drehschalter eingestellt.
0 = ausgeschaltet, sonst zwischen 0,1 und 5 Sekunden eingeschaltet mit eingestellter Wendezeit.

Hierbei wird nur bei 'ab' nach Ablauf der mit dem oberen Drehschalter eingestellten Verzögerungszeit eine Drehrichtungsumkehr vollzogen, um z.B. Markisen zu spannen oder Jalousien in eine bestimmte Position zu stellen.
RV = Die **Verzögerungszeit** (Rückfallverzögerung RV) wird mit dem oberen Drehschalter eingestellt. Befindet sich der Stromstoß-Gruppenschalter in der Stellung 'auf' oder 'ab', so läuft die eingestellte Verzögerungszeit, an deren Ende das Gerät automatisch auf 'halt' umschaltet. Die Verzögerungszeit muss daher mindestens so lange gewählt werden, wie das Beschattungselement oder der Rollladen benötigt, um von einer Endstellung in die andere zu kommen. Unter diesem Drehschalter befindet sich die LED-Anzeige für die Verzögerungszeiten WA und RV.

Örtliche Steuerung mit Taster an den Klemmen A3+A4 (mit einer Brücke verbinden). Mit jedem Impuls wechselt der Stromstoß-Gruppenschalter die Schaltstellung in der Reihenfolge auf, halt, ab, halt.

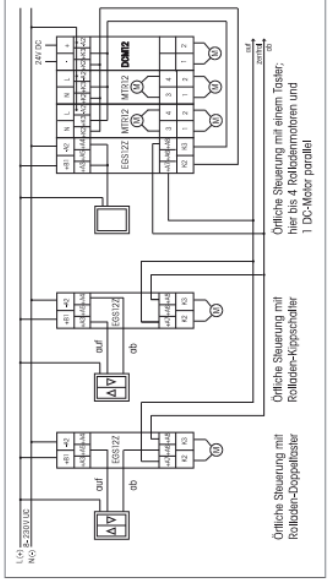
Örtliche Steuerung mit Rolladenkippschalter an den Klemmen A3 und A4.

Örtliche Steuerung mit Rolladendoppeltaster an den Klemmen A5 und A6. Mit einem Tastimpuls wird die Schaltstellung 'auf' oder 'ab' aktiviert. Ein weiterer Impuls von einem der beiden Taster unterbricht den Ablauf sofort.

Zentralsteuerung dynamisch ohne Priorität an den Klemmen A5 (auf) und A6 (ab). Mit einem Steuersignal wird die Schaltstellung 'auf' oder 'ab' aktiviert. Ein weiteres Steuersignal (< 700 ms) an diesem Steuereingang unterbricht den Ablauf sofort, ein weiteres Steuersignal (> 700 ms) führt die laufende Aktion weiter aus.
Ohne Priorität deswegen, weil der örtliche Steuereingang A3+A4 (mit Brücke) und die Zentral-Steuereingänge A7 und A8 sofort übersteuern können, selbst wenn der Steuerkontakt an A5 oder A6 noch geschlossen ist.

Zentralsteuerung dynamisch mit Priorität an den Klemmen A7 (auf) und A8 (ab). **Mit Priorität** deswegen, weil diese Steuereingänge nicht von anderen Steuereingängen übersteuert werden können, **solange** der Zentral-Steuerkontakt geschlossen ist. Sonst Funktion wie die Zentral-Steuerung dynamisch ohne Priorität. Diese Zentral-Steuereingänge A7 und A8 werden von den Sensorrelais MSR12 und LRW12D für die Regen-, Frost- und Wind-Funktion benutzt, da diese absolute Priorität vor den anderen Sensor-Befehlen haben müssen.

Anschlussbeispiel



Technische Daten

Versorgung- und Steuerspannung AC	8..253 V
Versorgung- und Steuerspannung DC	10..230 V
Nennschaltung	16A/250V AC
Indukative Last cos j= 0.6/230 V AC	650 W
Temperatur an der Einbaustelle max./min.	+50° C/-20° C
Steuerstrom A3-A8 bei 12/24/230 V ±20%	0,05/0,11/0,7 mA
Stand-by-Verlust (Wirkleistung)	0,05/0,1/0,4 W bei 12/24/230 V

⚠ Die Zugbügelklemmen der Anschlüsse müssen geschlossen sein, also die Schrauben eingedreht, um die Gerätefunktion prüfen zu können. Ab Werk sind die Klemmen geöffnet.

Zum späteren Gebrauch aufbewahren!
Wir empfehlen hierzu das Gehäuse für Bedienungsanleitungen GBA14.

Eltako GmbH
D-70736 Fellbach
☎ +49 711 94350000
www.eltako.com

10/2012 Änderungen vorbehalten.

Anhang 5 – Datenblatt des ULN2803A Darlington-Arrays

TEXAS INSTRUMENTS

ULN2803A

SLRS049G – FEBRUARY 1997 – REVISED JANUARY 2015

ULN2803A Darlington Transistor Arrays

1 Features

- 500-mA-Rated Collector Current (Single Output)
- High-Voltage Outputs: 50 V
- Output Clamp Diodes
- Inputs Compatible With Various Types of Logic
- Relay-Driver Applications
- Compatible with ULN2800A Series

2 Applications

- Relay Drivers
- Hammer Drivers
- Lamp Drivers
- Display Drivers (LED and Gas Discharge)
- Line Drivers
- Logic Buffers

3 Description

The ULN2803A device is a high-voltage, high-current Darlington transistor array. The device consists of eight NPN Darlington pairs that feature high-voltage outputs with common-cathode clamp diodes for switching inductive loads. The collector-current rating of each Darlington pair is 500 mA. The Darlington pairs may be connected in parallel for higher current capability.

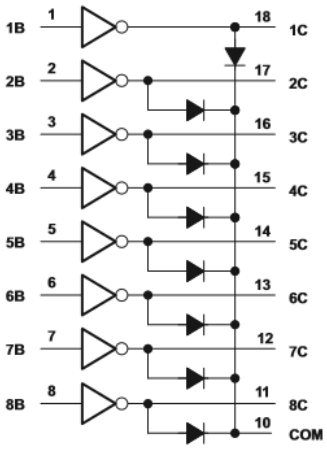
Applications include relay drivers, hammer drivers, lamp drivers, display drivers (LED and gas discharge), line drivers, and logic buffers. The ULN2803A device has a 2.7-kΩ series base resistor for each Darlington pair for operation directly with TTL or 5-V CMOS devices.

Device Information(1)

PART NUMBER	PACKAGE (PIN)	BODY SIZE (NOM)
ULN2803	SOIC (18)	11.50 mm × 7.50 mm
	PDIP (18)	22.48 mm × 6.35 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

4 Simplified Schematics



An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.