



PiXtend

Application-Note: Modbus-TCP Kommunikation

Application Note

Modbus-TCP Kommunikation

*Verbindung mehrerer PiXtend-Baugruppen
Einrichten von Modbus-Master & Slave unter CODESYS
I/O-Daten austauschen & Anzeigen*



APP-PX-560

Stand 24.08.2016, V1.01

Qube Solutions UG (haftungsbeschränkt)
Arbachtalstr. 6, 72800 Eningen, Germany

<http://www.qube-solutions.de/>

<http://www.pixtend.de>



Versionshistorie

Version	Beschreibung	Bearbeiter
1.00	Dokument erstellt, Texte & Bilder eingefügt	TO
1.01	- Dokument überarbeitet, Titelbild aktualisiert - Einleitungskapitel erstellt - Tipps & Hinweise eingefügt	TG



Inhaltsverzeichnis

1. Einleitung.....	5
1.1 Voraussetzungen & Vorbereitung.....	6
1.2 Haftungsausschluss.....	7
1.3 Sicherheitshinweise.....	7
2. Modbus Master Programm.....	8
3. Modbus Slave Programm.....	17
3. Inbetriebnahme.....	20



Abbildungsverzeichnis

Abbildung 1: Ethernet-Adapter anhängen.....	8
Abbildung 2: Ethernet Adapter anhängen.....	9
Abbildung 3: Gerätebaum.....	10
Abbildung 4: GVL Liste erweitert.....	11
Abbildung 5: Modbus-TCP-Einstellungen.....	12
Abbildung 6: Kanal Einstellungen.....	12
Abbildung 7: Kanal Einstellungen fertig.....	13
Abbildung 8: E/A-Abbild fertig eingestellt.....	13
Abbildung 9: Buszyklus aktivieren.....	14
Abbildung 10: Auto-reconnect aktivieren.....	14
Abbildung 11: Visualisierung.....	15
Abbildung 12: Variable einer Lampe zuweisen.....	16
Abbildung 13: ModbusTCP Slave einfügen.....	17
Abbildung 14: GVL des Slave.....	18
Abbildung 15: PiXtend-Device - E/A Konfiguration.....	18
Abbildung 16: ModbusTCP Slave Programmcode.....	19
Abbildung 17: ModbusTCP Slave Programmcode.....	19
Abbildung 18: Buszyklus und Variablen Aktualisierung beim Slave.....	19
Abbildung 19: Geräte in CODESYS sauber unterscheiden.....	20
Abbildung 20: Raspberry Pi umbenennen.....	21
Abbildung 21: Modbus läuft.....	21



1. Einleitung

Das Modbus¹-Protokoll hat sich in den letzten Jahren zu einer Art De-facto-Standard in der industriellen Automation entwickelt. Modbus basiert auf einer Master/Slave-Architektur. Es gibt also einen Master und mehrere Slaves.

Als Übertragungsmedium kann RS232 / RS485 (Modbus RTU) oder Ethernet (Modbus TCP) dienen. Wir schauen uns in dieser AppNote das moderne Modbus TCP genauer an.



Bei Modbus TCP haben Sie viele Vorteile. Heute hat jeder Haushalt, jede Firma und Bildungseinrichtung ein Ethernet-basiertes Netzwerk. Egal ob per Kabel oder W-LAN, die Infrastruktur besteht. Mit Modbus TCP können wir genau diese Infrastruktur nutzen, ohne weitere Kabel zu verlegen. Ein großer Vorteil für eine Haus Automatisierung (Smart Home).

In dieser App-Note zeigen wir Ihnen, wie Sie zwei PiXtend-Boards per Modbus TCP miteinander verbinden und Daten austauschen lassen. Ein Gerät fungiert als Master, das zweite als Slave.

Der Master kann die Ausgänge des Slave setzen und Eingänge lesen. Der Slave dient also als E/A-Erweiterung, die bei einer Home Automation sinnvoll ist und problemlos per WLAN angebunden werden kann.

Für die komfortable Bedienung bauen wir auch noch eine kleine Web-Visualisierung auf.

Dann bleibt nur noch zu sagen:

„Viel Spaß beim nachvollziehen der folgenden Schritte und bei Ihrem Automatisierungs-Projekt mit PiXtend!“

1 „Modbus“ und das zugehörige Logo sind geschützte Markenzeichen der Modbus Organization, Inc. - www.modbus.org ;



1.1 Voraussetzungen & Vorbereitung

Es wird vorausgesetzt, dass Sie bereits mit CODESYS und PiXtend gearbeitet haben und den grundlegenden Umgang beherrschen. Ansonsten möchten wir Sie gerne zunächst auf unsere Einführungsdokumente hinweisen:

[App-Note: PiXtend mit CODESYS – Installation der Softwarekomponenten](#)

[App-Note: PiXtend mit CODESYS – Projekt erstellen](#)

Hardware:

- PC mit CODESYS mit Ethernet oder WLAN-Schnittstelle
- 2x PiXtend V1.2 / V1.3, jeweils mit Raspberry Pi
- Netzwerkverbindung aller Geräte per WLAN/Ethernet

Software:

- CODESYS V3 Programmierungsumgebung auf dem PC
- *PiXtend Image CODESYS* aus unserem [Download-Bereich](#)
 - aufgespielt auf beiden SD-Karten der Raspberry Pi Computern
- Zwei Standard-Projekte für PiXtend vorbereitet (laut [AppNote](#))

Eines der Standard-Projekte wird mit dem Namen *Modbus_Master* und das andere mit dem Namen *Modbus_Slave* versehen.

Erst wenn diese beiden Projekte angelegt bzw. vorbereitet wurden, sollten Sie mit den nächsten Schritten im Kapitel 2. *Modbus Master Programm* fortfahren.

Die IP-Adressen von beiden Raspberry Pis im Netzwerk müssen bekannt sein. Am Besten notieren Sie sich diese im Vorhinein.



1.2 Haftungsausschluss

Weder Qube Solutions UG noch 3S-Smart Software Solutions können für etwaige Schäden verantwortlich gemacht werden die unter Umständen durch die Verwendung der zur Verfügung gestellten Software, Hardware, Treiber oder der hier beschriebenen Schritte entstehen können.

1.3 Sicherheitshinweise



PiXtend darf nicht in sicherheitskritischen Systemen eingesetzt werden.

Prüfen Sie vor der Verwendung die Eignung von Raspberry Pi und PiXtend für Ihre Anwendung.



2. Modbus Master Programm

Als erstes öffnen wir das Projekt, welchem wir den Namen *Modbus_Master* gegeben haben. Im nächsten Schritt wird ein *Ethernet Adapter* eingefügt. Mit einem Rechtsklick im Gerätebaum auf *Device (CODESYS Control for Raspberry Pi)* und dann *Gerät anhängen* kann der Ethernet-Adapter eingefügt werden, wie in Abbildung 1 gezeigt.

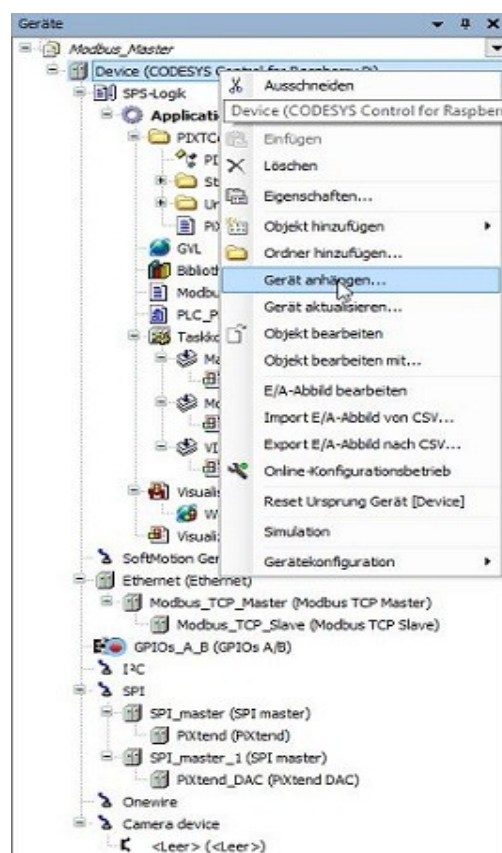


Abbildung 1: Ethernet-Adapter anhängen

Nun wird unter *Ethernet Adapter* das *Ethernet* Gerät ausgewählt.

Über den Button *Gerät anhängen* wird der Ethernet-Adapter an die Device angehängt.

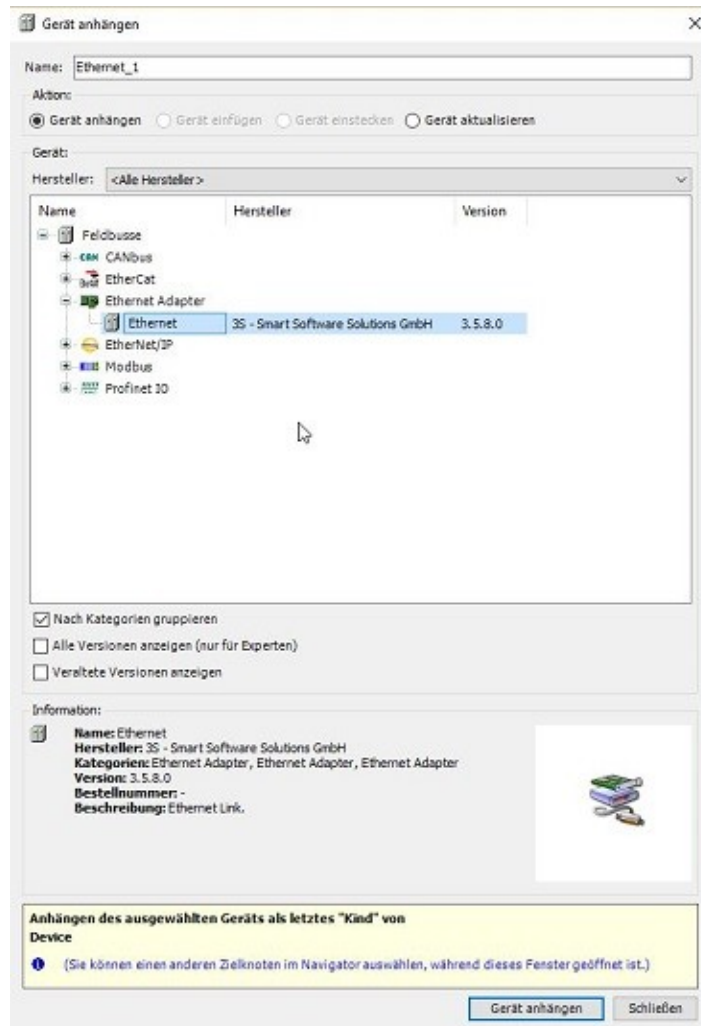


Abbildung 2: Ethernet Adapter anhängen

Der soeben eingehängte Ethernet-Adapter muss noch konfiguriert werden. Wir doppelklicken auf *Ethernet* und landen im Reiter *Allgemein*. Als Schnittstelle sollte hier (*eth0*) eingetragen werden (die Ethernet-Schnittstelle des Raspberry Pi). Falls Sie einen WLAN-Dongle oder das im RPi 3 B integrierte WLAN verwenden möchten, so tragen Sie im Feld *Netzwerkschnittstelle* die entsprechende Bezeichnung des WLANs ein. In der Regel ist das *wlan0*.

Tipp: Wenn Sie mit dem Gerät verbunden sind, kann CODESYS die Schnittstellen auch über den „...“ Button suchen.



Nun fügen wir noch zwei weitere Geräte ein, Modbus TCP Master & Slave.

Zuerst mit Rechtsklick auf den *Ethernet* Adapter der jetzt im Gerätebaum zu sehen ist. Dann wieder über Gerät anhängen im Reiter *Modbus / Modbus TCP Master* das *Modbus TCP Master* Gerät auswählen. Zuletzt muss auf die gleiche Weise ein *Modbus TCP Slave* an den *Modbus TCP Master* angehängt werden.

Der Gerätebaum sollte nun wie folgt aussehen:

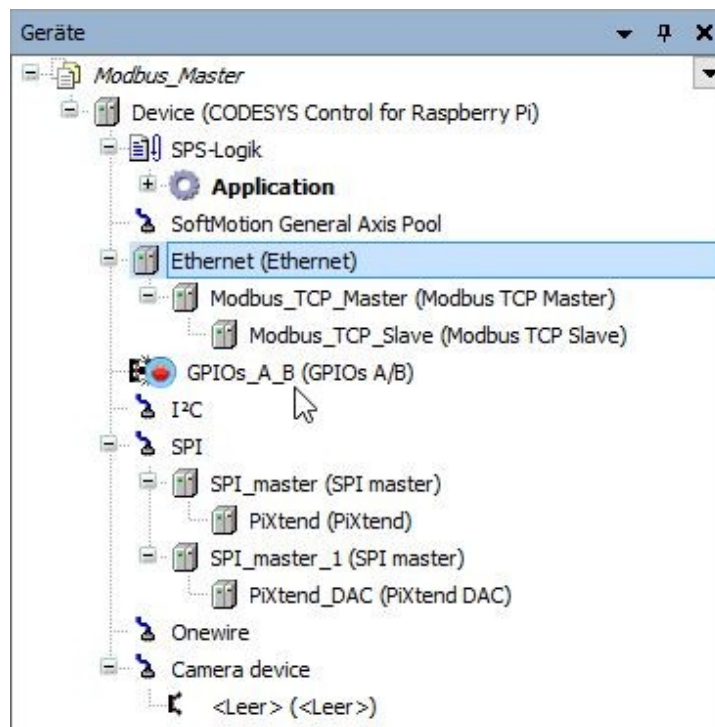


Abbildung 3: Gerätebaum

Bevor die Geräte-Parameter eingestellt werden, legen wir zuerst die Variablen an, die wir für das Auslesen der digitalen Ein- und Ausgänge brauchen. Diese werden in die GVL (Globale Variablen Liste) geschrieben.

Rechtsklick auf *Application* → *Objekt hinzufügen* → *Globale Variablenliste*

Evtl. haben Sie die GVL bei der Vorbereitung der beiden Projekte bereits angelegt. In diesem Fall können wir die Ein- und Ausgänge einfach zur bisherigen GVL hinzufügen. Die folgenden Variablen können Sie einfach per Copy-Paste aus diesem Dokument in CODESYS übernehmen:



//Eingänge

```
xDI0_Slave1 : BOOL;  
xDI1_Slave1 : BOOL;  
xDI2_Slave1 : BOOL;  
xDI3_Slave1 : BOOL;  
xDI4_Slave1 : BOOL;  
xDI5_Slave1 : BOOL;  
xDI6_Slave1 : BOOL;  
xDI7_Slave1 : BOOL;
```

//Ausgänge

```
xDO0_Slave1 : BOOL;  
xDO1_Slave1 : BOOL;  
xDO2_Slave1 : BOOL;  
xDO3_Slave1 : BOOL;  
xDO4_Slave1 : BOOL;  
xDO5_Slave1 : BOOL;
```

Nach dem Einfügen der Ein- und Ausgangsvariablen sollte die GVL in etwa so aussehen:

```
1 {attribute 'qualified_only'}  
2 VAR GLOBAL  
3 //PiXtend Status  
4 byUCStatus: BYTE;  
5 byUCVersionL: BYTE;  
6 byUCVersionH: BYTE;  
7 //Control  
8 byUCControl: BYTE;  
9  
10 xDI0_Slave1 : BOOL;  
11 xDI1_Slave1 : BOOL;  
12 xDI2_Slave1 : BOOL;  
13 xDI3_Slave1 : BOOL;  
14 xDI4_Slave1 : BOOL;  
15 xDI5_Slave1 : BOOL;  
16 xDI6_Slave1 : BOOL;  
17 xDI7_Slave1 : BOOL;  
18  
19 xDO0_Slave1 : BOOL;  
20 xDO1_Slave1 : BOOL;  
21 xDO2_Slave1 : BOOL;  
22 xDO3_Slave1 : BOOL;  
23 xDO4_Slave1 : BOOL;  
24 xDO5_Slave1 : BOOL;  
25 END_VAR
```

Abbildung 4: GVL Liste erweitert



Im nächsten Schritt müssen die Parameter des Slaves angepasst werden. Zu den Einstellungen kommen Sie mit einem Doppelklick auf die *Modbus_TCP_Slave Device*. Es öffnet sich folgendes Menü:

The screenshot shows a software interface for configuring Modbus-TCP settings. On the left is a sidebar with a tree view containing: Allgemein, Modbus Slave-Kanal, Modbus Slave Init, ModbusTCPSlave Parameter, ModbusTCPSlaveE/A-Abbild, Status, and Information. The 'Allgemein' tab is active. The main area is titled 'Modbus-TCP' and contains four input fields: 'Slave IP-Adresse' with the value '192 . 168 . 0 . 24', 'Unit-ID [1..247]' which is empty, 'Response Timeout (ms)' with the value '1000', and 'Port' with the value '502'. A 'MODBUS' logo is in the top right of the main area.

Abbildung 5: Modbus-TCP-Einstellungen

Im Reiter *Allgemein* wird die IP-Adresse des Raspberry Pi eingetragen, welcher der Slave sein wird. Der Port ist schon eingetragen. Der Port 502 wird standardmäßig für den Modbus TCP verwendet.

Im nächsten Reiter *Modbus Slave-Kanal* werden nun zwei Kanäle angelegt. Einen um die Eingänge des Slaves zu lesen und den anderen um dessen Ausgänge zu setzen. Rechts unten im Fenster kann über die Taste *Kanal hinzufügen...* ein neuer Kanal eingefügt werden. Dabei wird folgendes Fenster geöffnet:

The screenshot shows a 'ModbusChannel' dialog box. It has a 'Kanal' section with fields for 'Name' (Channel 0), 'Zugriffstyp' (Read Input Registers (Funktionscode 4)), 'Trigger' (Zyklisch), and 'Zykluszeit (ms)' (100). Below this is a 'Kommentar' field. The 'READ Register' section has fields for 'Offset' (empty), 'Länge' (1), and 'Fehlerbehandlung' (Letzen Wert beibehalten). The 'WRITE Register' section has fields for 'Offset' (empty) and 'Länge' (1). At the bottom right are 'OK' and 'Abbrechen' buttons.

Abbildung 6: Kanal Einstellungen



Hier muss nur der gewünschte Name des Kanals ausgewählt und bei *Zugriffstyp* → *Read Input Registers (Funktionscode 4)* ausgewählt werden. Dieser Kanal dient dem Lesen der digitalen Eingänge des Slaves. Mit OK Bestätigen.

Nun fügen wir einen weiteren Kanal ein, in dem wir den ganzen Vorgang wiederholen. Nur wird dieses Mal beim *Zugriffstyp* → *Write Multiple Registers (Funktionscode 16)* ausgewählt.

Es sind jetzt zwei Kanäle im Reiter *Modbus Slave-Kanal* sichtbar:

Allgemein	Name	Zugriffstyp	Trigger	READ-Offset	Länge	Fehlerbehandlung	WRITE Offset	Länge	Kommentar
Modbus Slave-Kanal	Channel 0	Read Input Registers (Funktionscode 04)	Zyklisch, t#100ms	16#0000	1	Letzen Wert beibehalten			
Modbus Slave Init	Channel 1	Write Multiple Registers (Funktionscode 16)	Zyklisch, t#100ms				16#0000	1	
ModbusTCPSlave Parameter									
ModbusTCPSlaveE/A-Abbild									
Status									
Information									

Abbildung 7: Kanal Einstellungen fertig

Anschließend wird auf den Reiter *ModbusTCPSlave E/A-Abbild* gewechselt. Dort sind die zuvor deklarierten Variablen einzutragen. Abbildung 8 zeigt wie es aussehen sollte wenn Sie fertig sind:

Allgemein	Kanäle	Mapping	Kanal	Adresse	Typ	Einheit	Beschreibung
Modbus Slave-Kanal	Variable		Channel 0	%IW0	ARRAY [0..0] OF WORD		Read Input Registers
Modbus Slave Init			Channel 0[0]	%IW0	WORD		0000:
ModbusTCPSlave Parameter	Application.Modbus_Master.xDI0_Slave1	Bit0		%IX0.0	BOOL		
ModbusTCPSlaveE/A-Abbild	Application.Modbus_Master.xDI1_Slave1	Bit1		%IX0.1	BOOL		
Status	Application.Modbus_Master.xDI2_Slave1	Bit2		%IX0.2	BOOL		
Information	Application.Modbus_Master.xDI3_Slave1	Bit3		%IX0.3	BOOL		
	Application.Modbus_Master.xDI4_Slave1	Bit4		%IX0.4	BOOL		
	Application.Modbus_Master.xDI5_Slave1	Bit5		%IX0.5	BOOL		
	Application.Modbus_Master.xDI6_Slave1	Bit6		%IX0.6	BOOL		
	Application.Modbus_Master.xDI7_Slave1	Bit7		%IX0.7	BOOL		
		Bit8		%IX1.0	BOOL		
		Bit9		%IX1.1	BOOL		
		Bit10		%IX1.2	BOOL		
		Bit11		%IX1.3	BOOL		
		Bit12		%IX1.4	BOOL		
		Bit13		%IX1.5	BOOL		
		Bit14		%IX1.6	BOOL		
		Bit15		%IX1.7	BOOL		
			Channel 1	%QW0	ARRAY [0..0] OF WORD		Write Multiple Registers
			Channel 1[0]	%QW0	WORD		0000:
	Application.Modbus_Master.xDO0_Slave1	Bit0		%QX0.0	BOOL		
	Application.Modbus_Master.xDO1_Slave1	Bit1		%QX0.1	BOOL		
	Application.Modbus_Master.xDO2_Slave1	Bit2		%QX0.2	BOOL		
	Application.Modbus_Master.xDO3_Slave1	Bit3		%QX0.3	BOOL		
	Application.Modbus_Master.xDO4_Slave1	Bit4		%QX0.4	BOOL		
	Application.Modbus_Master.xDO5_Slave1	Bit5		%QX0.5	BOOL		
		Bit6		%QX0.6	BOOL		
		Bit7		%QX0.7	BOOL		

Abbildung 8: E/A-Abbild fertig eingestellt



Die digitalen Eingänge werden bei Channel 0 (Read Inputs Registers) eingetragen, die digitalen Ausgänge bei *Channel 1 (Write Multiple Register)*.

Rechts unten im Reiter *ModbusTCPE/A-Abbild* stellen Sie bei Variablen aktualisieren folgende Einstellung im Drop-Down-Menü ein: *Aktiviert 1, Buszyklus-Task*.

Variablen aktualisieren: Aktiviert 1 (Buszyklus-Task verwenden, wenn in keine ▾)

Abbildung 9: Buszyklus aktivieren

Wichtig: Ohne diese Einstellung findet später keine Datenübertragung statt!

Eine Einstellung nehmen wir nun noch am *Modbus_TCP_Master* Gerät im Reiter *Allgemein* vor:

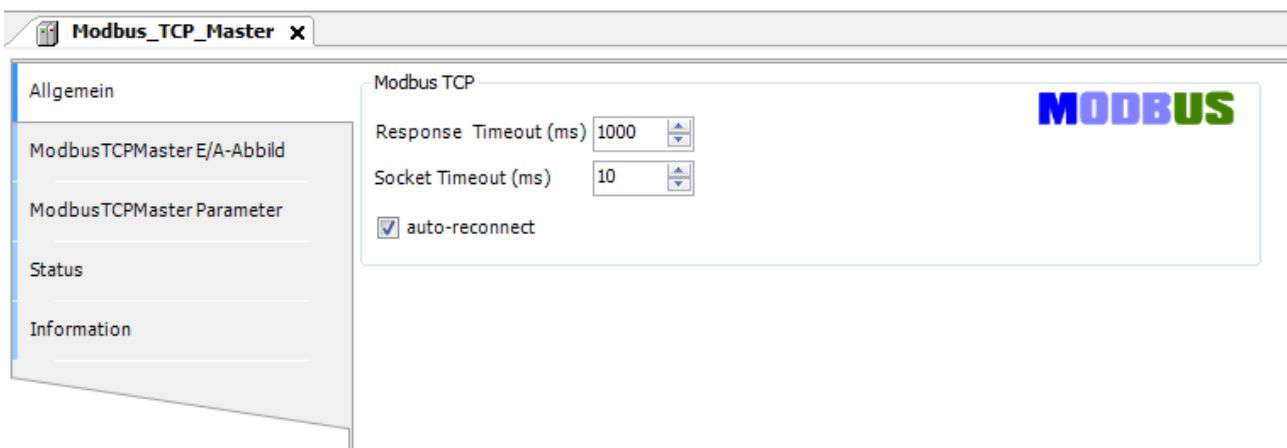


Abbildung 10: Auto-reconnect aktivieren

Die CheckBox *auto-reconnect* sollte aktiviert werden. Ansonsten stellt der Modbus-Master nach einem Bus-Error den Betrieb ein. Ein solcher Bus-Error kann schon dann auftreten, wenn der Slave innerhalb der *Response Time* nicht antwortet (das passiert z.B. wenn der Slave einen Reboot macht oder wir ein neues Programm aufspielen).

Damit ist das Master-Programm nahezu abgeschlossen.



Zum Schluss erstellen wir noch eine kleine Visualisierung, damit wir später die Eingänge des Slaves anzeigen und dessen Ausgänge komfortabel schalten können.

Falls Sie noch keine Visualisierung angelegt haben, so können Sie dies nun folgendermaßen tun:

Rechtsklick auf *Application* → *Objekt hinzufügen* → *VisualizationManager*

Der *Visualizationmanager* legt automatisch ein weiteres Objekt, die *WebVisu*, an. Nun aber noch die eigentliche Visualisierung:

Rechtsklick auf *Application* → *Objekt hinzufügen* → *Visualization*

Damit die gerade erstellte Visualisierung auch als Webvisu angezeigt wird, klicken Sie im Gerätebaum Auf *WebVisu* und tragen im obersten Feld „*Startvisualisierung:*“ den Namen Ihrer Visualisierung ein (Standard-Name ist: *Visualization*).

Eingänge



Ausgänge



Abbildung 11: Visualisierung



Um die Eingänge anzuzeigen wird z.B. das Element *Lampe* verwendet und um die Ausgänge zu schalten das *CheckBox* Element. Jedem Element wird eine entsprechende Variable zugewiesen. Siehe Abbildung 12:


Eigenschaft	Wert
Elementname	GenElemInst_1
Elementtyp	Lampe
Position	
X	152
Y	134
Breite	70
Höhe	70
Variable	GVL.xDI0_Slave1
Bildeinstellungen	
Transparenz	<input type="checkbox"/>
Transparenzfarbe	 Schwarz
Skalierungsart	Isotropisch
Horizontale Ausrichtung	Links
Vertikale Ausrichtung	Oben
Texte	
Tooltip	
Zustandsvariablen	
Unsichtbarkeit	
Hintergrund	
Bild	Gelb

Abbildung 12: Variable einer Lampe zuweisen

Gleichermaßen wird auch mit den CheckBox-Elementen verfahren. Hier werden die Ausgänge *DO 0* (*xDO0_Slave1*) bis *DO 5* (*xDO5_Slave1*) als Variablen eingetragen.

Somit ist das Programm für den Modbus-TCP Master fertiggestellt und wir können mit dem Slave Programm weitermachen.

Tipp: Die Texte wie „Ausgänge“, „Eingänge“ und die Beschriftungen der Lampen können wir mit dem Element *Textfeld* hinzufügen. Das Textfeld hat standardmäßig einen dünnen Rahmen um den Text. Um diesen Rahmen zu entfernen werfen Sie einen Blick in das Eigenschaften-Fenster des Textfelds. Unter *Aussehen* kann die *Linienart* auf *Unsichtbar* eingestellt werden. Der Rahmen wird dann nicht länger angezeigt.



3. Modbus Slave Programm

Wie im Master Programm muss auch hier zunächst ein *Ethernet Adapter* eingefügt werden. An den *Ethernet Adapter* wird hier allerdings nur eine *ModbusTCP Slave Device* angehängt:

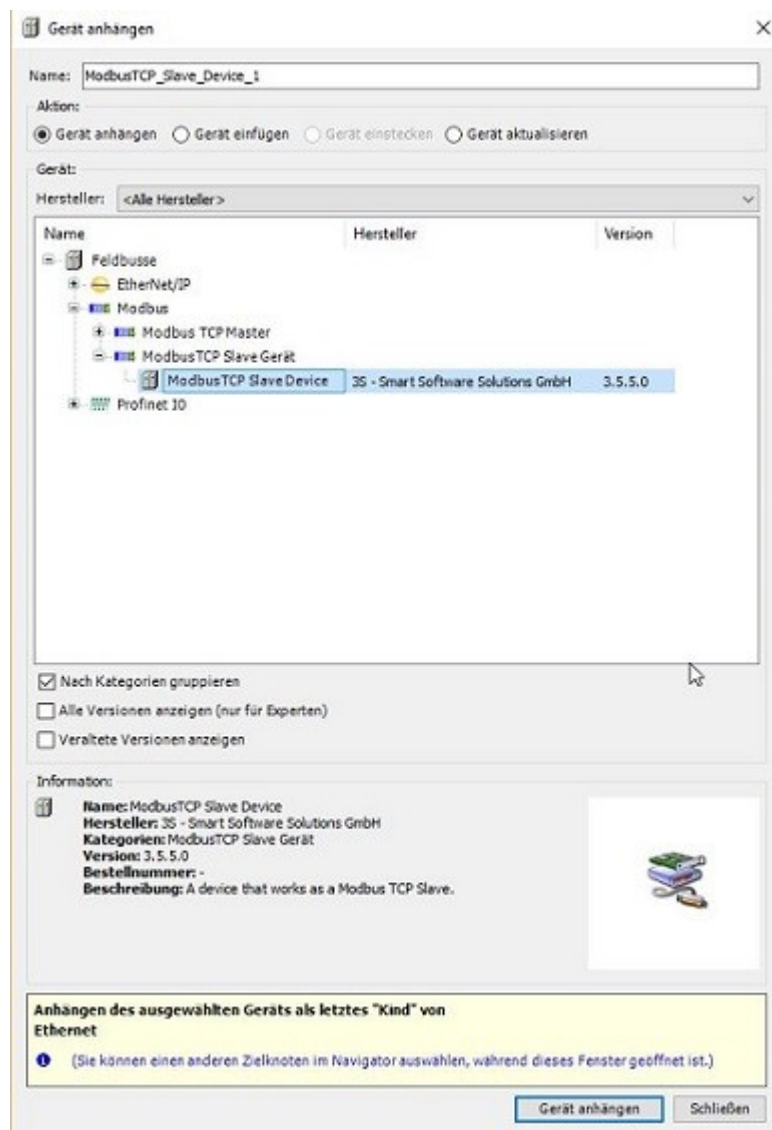


Abbildung 13: ModbusTCP Slave einfügen

Wir legen in diesem einfachen Beispiel keinen Programmcode an, sondern verbinden die Modbus-Daten direkt mit den PiXtend Ein- und Ausgängen des Slave-Gerätes.



Der Einfachheit halber benutzen wir im Slave Programm die gleichen Variablennamen für die Ein- und Ausgänge, wie wir sie bereits im Master Programm genannt haben. Das müsste aber nicht zwingend so sein.

Die GVL wird genau so angelegt bzw. erweitert, wie im Master Programm:

```
1 {attribute 'qualified_only'}
2 VAR GLOBAL
3   //PiXtend Status
4   byUCStatus: BYTE;
5   byUCVersionL: BYTE;
6   byUCVersionH: BYTE;
7   //Control
8   byUControl: BYTE;
9
10  xDI0_Slave1 : BOOL;
11  xDI1_Slave1 : BOOL;
12  xDI2_Slave1 : BOOL;
13  xDI3_Slave1 : BOOL;
14  xDI4_Slave1 : BOOL;
15  xDI5_Slave1 : BOOL;
16  xDI6_Slave1 : BOOL;
17  xDI7_Slave1 : BOOL;
18
19  xDO0_Slave1 : BOOL;
20  xDO1_Slave1 : BOOL;
21  xDO2_Slave1 : BOOL;
22  xDO3_Slave1 : BOOL;
23  xDO4_Slave1 : BOOL;
24  xDO5_Slave1 : BOOL;
25 END_VAR
```

Abbildung 14: GVL des Slave

Nun kommen wir zum PiXtend-Device. Hier weisen wir die Eingänge und Ausgänge entsprechend zu, wie Abbildung 15 zeigt.

Variable	Mapping	Kanal	Adresse	Typ
Digital In				
Application.GVL.xDI0_Slave1	Bit0	DigIn	%I20.0	BOOL
Application.GVL.xDI1_Slave1	Bit1		%I20.1	BOOL
Application.GVL.xDI2_Slave1	Bit2		%I20.2	BOOL
Application.GVL.xDI3_Slave1	Bit3		%I20.3	BOOL
Application.GVL.xDI4_Slave1	Bit4		%I20.4	BOOL
Application.GVL.xDI5_Slave1	Bit5		%I20.5	BOOL
Application.GVL.xDI6_Slave1	Bit6		%I20.6	BOOL
Application.GVL.xDI7_Slave1	Bit7		%I20.7	BOOL
		GpioIn	%I21	BYTE
Analog In				
Temp In				
Humidity In				
Digital Out				
Application.GVL.xDO0_Slave1	Bit0	DigOut	%Q20.0	BOOL
Application.GVL.xDO1_Slave1	Bit1		%Q20.1	BOOL
Application.GVL.xDO2_Slave1	Bit2		%Q20.2	BOOL
Application.GVL.xDO3_Slave1	Bit3		%Q20.3	BOOL
Application.GVL.xDO4_Slave1	Bit4		%Q20.4	BOOL
Application.GVL.xDO5_Slave1	Bit5		%Q20.5	BOOL
	Bit6		%Q20.6	BOOL
	Bit7		%Q20.7	BOOL
		GpioOut	%Q21	BYTE

Abbildung 15: PiXtend-Device - E/A Konfiguration



Zu guter Letzt widmen wir uns dem Modbus TCP Slave Device.

Wichtig: Was auf der Master-Seite ein Ausgang ist, hier auf der Slave-Seite ein Eingang. Lassen Sie sich dadurch bitte nicht verwirren.

Wir konfigurieren also die DOs (Ausgänge) auf das erste Eingangsbyte unseres Slave.

Ausgänge:

Variable	Mapping	Kanal	Adresse	Typ	Einheit	Beschreibung
		Eingänge	%IW0	ARRAY [0..9] OF WORD		Modbus Holding Registers
		Eingänge[0]	%IW0	WORD		
Application.Modbus_Slave.xDO0_Slave1		Bit0	%IX0-0	BOOL		
Application.Modbus_Slave.xDO1_Slave1		Bit1	%IX0-1	BOOL		
Application.Modbus_Slave.xDO2_Slave1		Bit2	%IX0-2	BOOL		
Application.Modbus_Slave.xDO3_Slave1		Bit3	%IX0-3	BOOL		
Application.Modbus_Slave.xDO4_Slave1		Bit4	%IX0-4	BOOL		
Application.Modbus_Slave.xDO5_Slave1		Bit5	%IX0-5	BOOL		
		Bit6	%IX0-6	BOOL		

Abbildung 16: ModbusTCP Slave Programmcode

Die DI's (Eingänge) legen wir auf das erste Ausgangsbyte des Slave.

Eingänge:

Variable	Mapping	Kanal	Adresse	Typ	Einheit	Beschreibung
		Eingänge	%IW0	ARRAY [0..9] OF WORD		Modbus Holding Registers
		Ausgänge	%QW0	ARRAY [0..9] OF WORD		Modbus Input Registers
		Ausgänge[0]	%QW0	WORD		
Application.Modbus_Slave.xDI0_Slave1		Bit0	%IX0-0	BOOL		
Application.Modbus_Slave.xDI1_Slave1		Bit1	%IX0-1	BOOL		
Application.Modbus_Slave.xDI2_Slave1		Bit2	%IX0-2	BOOL		
Application.Modbus_Slave.xDI3_Slave1		Bit3	%IX0-3	BOOL		
Application.Modbus_Slave.xDI4_Slave1		Bit4	%IX0-4	BOOL		
Application.Modbus_Slave.xDI5_Slave1		Bit5	%IX0-5	BOOL		
Application.Modbus_Slave.xDI6_Slave1		Bit6	%IX0-6	BOOL		
Application.Modbus_Slave.xDI7_Slave1		Bit7	%IX0-7	BOOL		
		Bit8	%IX1-0	BOOL		

Abbildung 17: ModbusTCP Slave Programmcode

Im E/A-Abbild wird auch hier die *Buszyklus* und *Variablen aktualisieren* Einstellung vorgenommen, da ansonsten keine Daten ausgetauscht werden:



Abbildung 18: Buszyklus und Variablen Aktualisierung beim Slave

Der Bus-Zyklus wird auf Ihren Standard-Task (*MainTask*) eingestellt.



3. Inbetriebnahme

Das Modbus Slave Programm ist damit abgeschlossen und nun wird es spannend!

Sie können die Programme nun auf die beiden Geräte aufspielen. CODESYS kann auf dem PC mehrfach geöffnet werden (mehrere Programm-Instanzen), so dass Sie ein Fenster für den Master und eines für den Slave zur gleichen Zeit geöffnet haben können.

Wichtig: Achten Sie darauf, dass Sie das jeweilige Programm auf das richtige Gerät aufspielen. Sie erinnern sich daran, dass wir im Master Programm die IP-Adresse des Slaves angegeben haben. Ein Vertauschen der Geräte würde dazu führen, dass keine Kommunikation zustande kommt.

Es ist eine gute Vorgehensweise den Geräten verschiedene Namen zu geben:

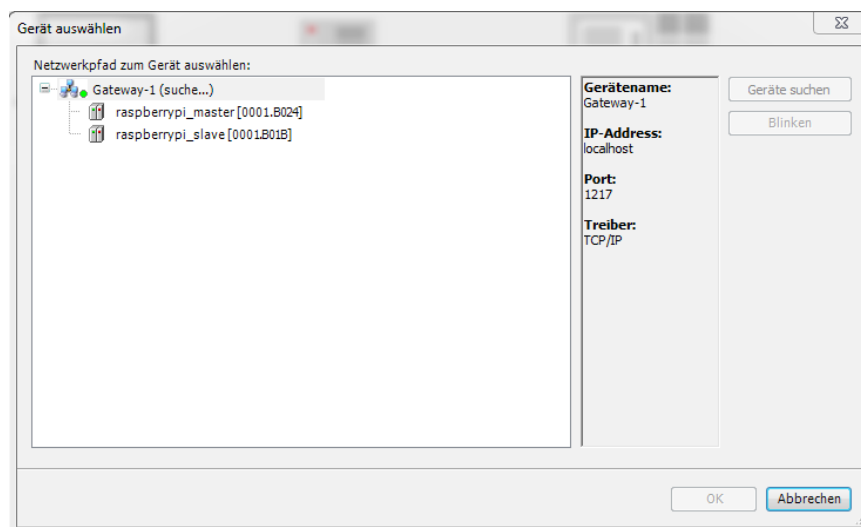


Abbildung 19: Geräte in CODESYS sauber unterscheiden



Ein zuvor ausgewähltes Gerät kann sehr einfach umbenannt werden:

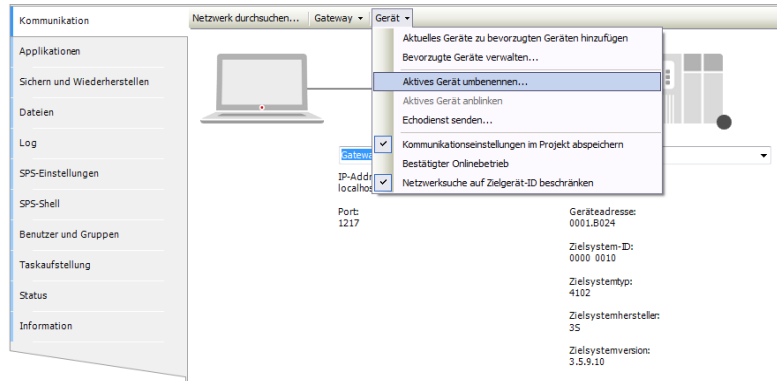


Abbildung 20: Raspberry Pi umbenennen

Wenn Sie keine Fehler in die Programme eingebaut haben, so müssen sowohl auf Master-, als auch auf Slave-Seite die Modbus-Geräte mit einem grünen Symbol gekennzeichnet sein. Dies bedeutet, dass der Bus fehlerfrei läuft.

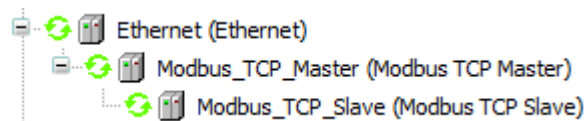


Abbildung 21: Modbus läuft

Sie können auf die Webvisu des Masters zugreifen und dort die Ausgänge des Slaves manipulieren. Ebenso werden Ihnen mit den Lampen die Eingänge des Slaves angezeigt. Es handelt sich hier um ein sehr einfaches Beispiel. Natürlich können Sie Modbus TCP auch zur Übertragung von weitaus mehr Daten verwenden. Es müssen auch nicht nur I/O-Daten sein. Sie könnten beliebige Daten übertragen.

Auch können Sie noch weitere Modbus-Geräte (Slaves) ansprechen. Es gibt eine Vielzahl an Geräten und E/A-Erweiterungen für Modbus TCP.

Wenn es mal nicht funktioniert:

Wir laden Sie zu einer Diskussion und dem Erfahrungsaustausch in unserem Forum (<http://www.pixtend.de/forum/>) ein. Sollte Ihnen hier nicht innerhalb von 2 Tagen geholfen werden, so wenden Sie sich bitte an unseren Support: support@pixtend.de