CODESYS STORE

SNMP Library

Mit der SNMP Library können Informationen von Geräten wie Drucker, Router usw. via SNMP von der Steuerung aus abgefragt werden. Die Bibliothek stellt Funktionsbausteine zum Versenden und Empfangen von SNMP Nachrichten bereit. Die Verwendung der Bibliothek wird anhand von mehreren Beispielen demonstriert.

Die Bibliothek ‚SNMP Library' ist jetzt Teil des Produktes IIoT Libraries SL **und ist als Einzelprodukt nicht mehr verfügbar.**

Produktbeschreibung

Lizenzierung:

Arbeitsplatzlizenz

Mit der SNMP Library können Informationen von Geräten, wie Drucker, Router usw. via SNMP von der Steuerung aus abgefragt werden. Die Bibliothek stellt Funktionsbausteine zum Versenden und Empfangen von SNMP Nachrichten bereit. Die Verwendung der Bibliothek wird anhand von mehreren Beispielen demonstriert.

Die Bibliothek unterstützt folgende SNMP Funktionen:

- SNMP GET: Abfrage eines Wertes (Steuerung ist Manager)
- SNMP GET_NEXT: Abfrage von Tabellen (Steuerung ist Manager)
- SNMP Agent: Abfrage von Werten der Steuerung (Steuerung ist Agent)
- SNMP TRAP: Senden und Empfangen von TRAP/INFORM-Telegrammen
- SNMP SET: Setzen von Werten

Das Package `SNMPLibrary.package` enthält folgende Komponenten:

- SNMPLibrary (Namespace: SNMP)
- Projekt mit Beispielapplikationen
- CHM Hilfedatei
- Produktdatenblatt

Die Bibliothek `SNMPLibrary` ist in vier Kategorien aufgeteilt:

`Agent`: Funktionsbausteine für den SNMP-Agent
`GET`: Funktionsbaustein zum Abfragen von Werten
`Trap`: Funktionsbausteine zum Empfangen und Versenden von TRAP Telegrammen
`SET`: Funktionsbaustein zum Setzen von Werten.

Die vier Bereiche sind unter dem Ordner `"Function Blocks"` abgelegt.

Im Folgenden werden die Funktionsbausteine der vier Kategorien beschrieben.

Funktionsbausteine der Kategorie „Agent"

Der Funktionsbaustein `SNMP_AGENT` bildet die Basis des SNMP Agenten. Der Funktionsbaustein empfängt SNMP GET und GET_NEXT-Telegramme, sucht den entsprechenden Wert über die OID und sendet ein Antworttelegramm mit dem Wert an den Master zurück. Über die Funktionsbausteine `SNMP_STRING` und `SNMP_DINT` können beliebige Parameter unter einer OID registriert werden.

Die Applikation `SNMPAgentExample` im Projekt `SNMPExample.project` veranschaulicht die Verwendung der Komponenten.

Funktionsbaustein SNMP_Agent:

Eingänge:

`sOwnIP`: IP-Adresse der Steuerung (eigene IP-Adresse)
`uiPort`: SNMP-Port, Default 161, UDP
`xExecute`: Startet den Agenten

Ausgänge:

`xDone`: True, wenn die Aktion erfolgreich ausgeführt wurde.
`xError`: True, wenn ein Fehler aufgetreten ist.
`xBusy`: True, während die Aktion ausgeführt wird.

Funktionsbaustein SNMP_STRING:

Funktionsbaustein zum Registrieren eines STRINGS.

Eingänge:

`pSnmpAgent`: Zeiger auf Funktionsbaustein SNMP_AGENT an dem der Wert registriert werden soll.
`sOID`: OID des Wertes (z.B. 1.3.6.1.4.1.2001)
`sValue`: Der zu registrierende Wert

Funktionsbaustein SNMP_DINT:

Funktionsbaustein zum Registrieren eines DINT.

Eingänge:

`pSnmpAgent`: Zeiger auf Funktionsbaustein SNMP_AGENT an dem der Wert registriert werden soll.
`sOID`: OID des Wertes (z.B. 1.3.6.1.4.1.2001)
`diValue`: Der zu registrierende Wert

Funktionsbausteine der Kategorie „GET“

Die Abfrage eines Geräteparameters über SNMP erfolgt über den Funktionsbaustein `SNMP_GET_REQUEST`. Die Applikationen `SNMGetNextExample` und `SNMPTimerExample` im Projekt `SNMPExample.project` veranschaulichen die Verwendung der Komponenten.

Funktionsbaustein SNMP_GET_REQUEST:

Eingänge:

`sHost`: IP-Adresse des Agenten
`sOwnIP`: IP-Adresse der Steuerung (eigene IP-Adresse)
`asOIDs`: OIDs der abzufragenden Werte, z.B. 1.3.6.1.2.1.1.3.0 (Uptime)
`iNumberOfOIDs`: Anzahl der Elemente in `asOIDs`
`eRequestType`: GET bzw. GET_NEXT
`uiPeerPort`: Port des Senders, Default 161, UDP
`uiSendPort`: Port des Empfängers, Default 161, UDP
`xExecute`: Öffnet ein UDP-Socket (wenn nicht schon offen) und startet die Abfrage.
`sCommunity`: „Community“ der Abfrage
`xClosePeer`: Steigende Flanke schließt den UDP-Socket.

Ausgänge:

`aSNMPValues`: Ergebnis der Abfrage. Das Ergebnis der Abfrage ist in einem Array von SNMPValue-Strukturen gespeichert. In der Regel enthält das Ergebnis zwei Werte. Element 0 enthält die OID des Wertes, bei GET_NEXT steht hier die OID des nächsten Wertes. Element 1 enthält den Wert.
`iSize`: Länge des Arrays `aSNMPValues`
`abResponse`: Ergebnis als Byte-Array (BER-Codiert)
`xDone`: True, wenn die Abfrage erfolgreich ausgeführt wurde.
`xError`: True, wenn ein Fehler aufgetreten ist.
`eError`: Fehlercode
`xBusy`: True, während die Abfrage ausgeführt wird.

Funktionsbausteine der Kategorie „SET“

Das Setzen eines Geräteparameters über SNMP erfolgt über den Funktionsbaustein `SNMP_SET`. Die Applikation `SNMSETExample` im Projekt `SNMPExample.project` veranschaulicht die Verwendung der Komponenten.

Funktionsbaustein SNMP_SET:

Eingänge:

`sHost`: IP-Adresse des Agenten
`sOwnIP`: IP-Adresse der Steuerung (eigene IP-Adresse)

snmpVarBindings: Array von SNMPVarBinding. Die Struktur SNMPVarBinding enthält die OID und den entsprechenden Wert.
iNumberOfVarBindings: Anzahl der Strukturen in snmpVarBindings.
uiPeerPort: Port des Senders, Default 161, UDP
uiSendPort: Port des Empfängers, Default 161, UDP
xExecute: Öffnet ein UDP-Socket (wenn nicht schon offen) und startet die Abfrage.
sCommunity: „Community“ der Abfrage
xClosePeer: Steigende Flanke schließt den UDP-Socket.

Ausgänge:

aSNMPValues: Ergebnis der Abfrage. Das Ergebnis der Abfrage ist in einem Array von SNMPValue-Strukturen gespeichert.
iSize: Länge des Arrays aSNMPValues
abResponse: Ergebnis als Byte-Array (BER-Codiert)
bySNMPErrorStatus: SNMP-Error-Status der Antwort
xDone: True, wenn die Abfrage erfolgreich ausgeführt wurde.
xError: True, wenn ein Fehler aufgetreten ist.
eError: Fehlercode
xBusy: True, während die Abfrage ausgeführt wird.

Funktionsbausteine der Kategorie „Trap“

Die Kategorie „Trap“ enthält Funktionsbausteine zum Empfangen und Versenden von TRAP/INFORM-Telegrammen. Die Applikationen SNMPTrapReceiver und SNMPTrapSender im Projekt SNMPExample.project veranschaulichen die Verwendung der Komponenten.

Funktionsbaustein SNMP_TRAP_RECEIVER:

Funktionsbaustein zum Empfangen von TRAP/INFORM-Telegrammen.

Eingänge:

sOwnIP: IP-Adresse der Steuerung (eigene IP-Adresse)
uiPort: SNMP-Port, Default 162, UDP
xExecute: Aktiviert den Funktionsbaustein

Ausgänge:

xDone: True, wenn die Aktion erfolgreich ausgeführt wurde.
xError: True, wenn ein Fehler aufgetreten ist.
xBusy: True, während die Aktion ausgeführt wird.
xReceived True, wenn ein Trap empfangen wurde.
aSNMPValues SNMP-Werte des Traps
iSize Anzahl der empfangen Werte
sSenderIP IP-Adresse des Senders
sEnterprise Enterprise-OID des Senders
udiTimestamp Timestamp des Senders

Funktionsbaustein SNMP_TRAP_SENDER:

Funktionsbaustein zum Senden von TRAP-Telegrammen.

Eingänge:

sHost: IP-Adresse des Empfängers
sOwnIP: IP-Adresse der Steuerung (eigene IP-Adresse)
uiPeerPort: Port des Senders, Default 162, UDP
uiSendPort: Port des Empfängers, Default 162, UDP
sEnterprise: OID des auslösenden Events
snmpVarBindings: Array von SNMPVarBinding. Die Struktur SNMPVarBinding enthält die OID und den entsprechenden Wert.
iNumberOfVarBindings: Anzahl der Strukturen in snmpVarBindings.
sCommunity: „Community“ der Abfrage
xClosePeer: Steigende Flanke schließt den UDP-Socket.
bGenericTrapType: Generic type, default: 6 (enterprise specific)
bSpecificTrapType: Specific type, default: 0

`udiTimestamp`: Zeitstempel, default: 0
`eTrapType`: Typ des TRAP-Telegramms (V1_Trap, V2_Trap, Inform). Inform-Telegramme werden mit einem Antworttelegramm beim Sender bestätigt.
`uiTimeout`: Timeout in Millisekunden (nur relevant bei Inform-Telegrammen). Die Zeit in der das Inform-Telegramm bestätigt werden muss.
`xExecute`: Öffnet ein UDP-Socket (wenn nicht schon offen) und startet die Abfrage.

Ausgänge:

`xDone`: True, wenn die Aktion erfolgreich ausgeführt wurde.
`xError`: True, wenn ein Fehler aufgetreten ist.
`xBusy`: True, während die Aktion ausgeführt wird.

Struktur „SNMPValue“

SNMP-Werte werden über die Struktur `SNMPValue` abgebildet.

Aufbau der Struktur SNMPValue:

`byType`: Datentyp des Wertes; Die Datentypen sind in der globalen Variablenliste definiert.
`aValue`: Ergebnis des Wertes als Byte-Array (BER-Codiert)
`iLength`: Länge des Arrays `aValue`
`psValue`: Zeiger auf das Ergebnis als String
`pliValue`: Zeiger auf das Ergebnis als LINT (nur relevant für Integer-Typen)

Struktur „SNMPVarBinding“

Die Struktur `SNMPVarBinding` enthält eine OID und den entsprechenden Wert.

Aufbau der Struktur `SNMPVarBinding`:

`oid`: Object Identifier
`value`: Wert der OID

Beispiel SNMPExample.project / Applikation SNMPAgentExample:

Das Beispiel zeigt, wie der Funktionsbaustein `SNMP_AGENT` verwendet wird und wie Werte registriert werden können. Eine Visualisierung *(siehe Abbildung 1)* zeigt den Status des Agenten an.

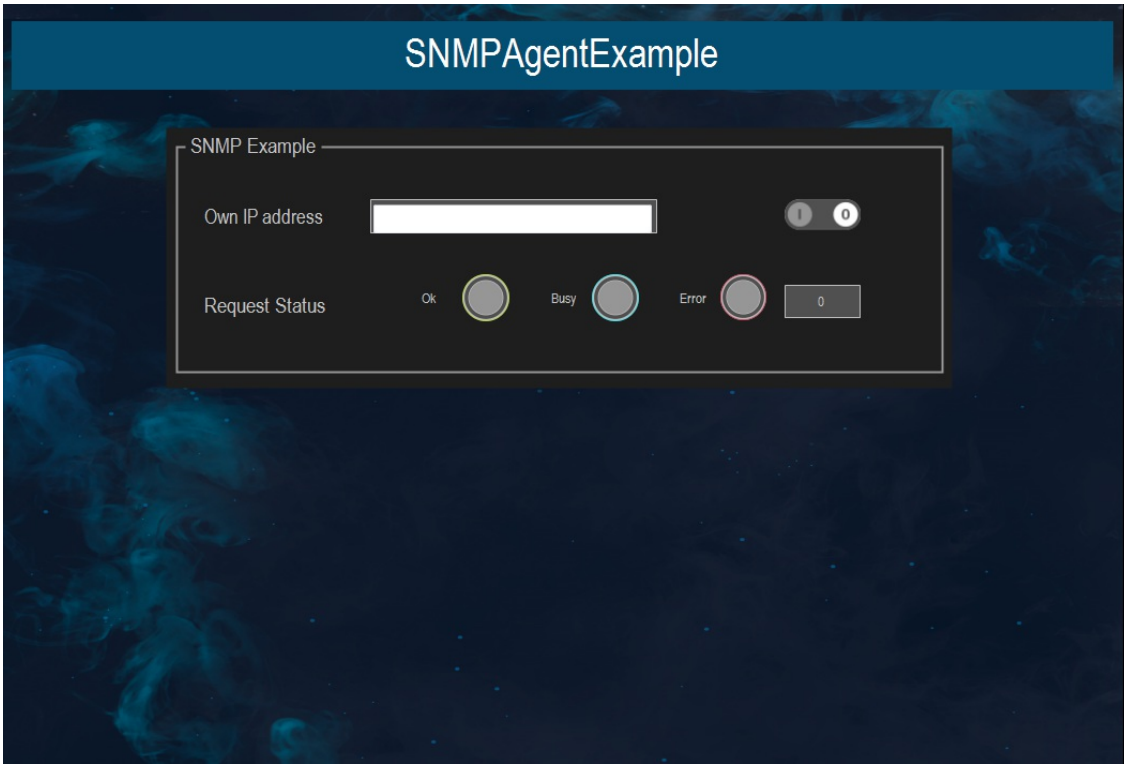

Abbildung 1: SNMPAgentExample

Beispiel SNMPExample.project / Applikation SNMPGetNextExample

Das Beispiel zeigt die Verwendung des Funktionsbausteins `SNMP_GET_REQUEST` für einen GET_NEXT Request. Die Ergebniswerte einer Abfrage werden in einer Visualisierung (siehe Abbildung 2) dargestellt. Es ist zu beachten, dass in dem Beispiel der zugehörige Wert einer OID immer im nachfolgenden Datensatz steht.

Die IP-Adressen und die OID werden über die Visualisierung konfiguriert.

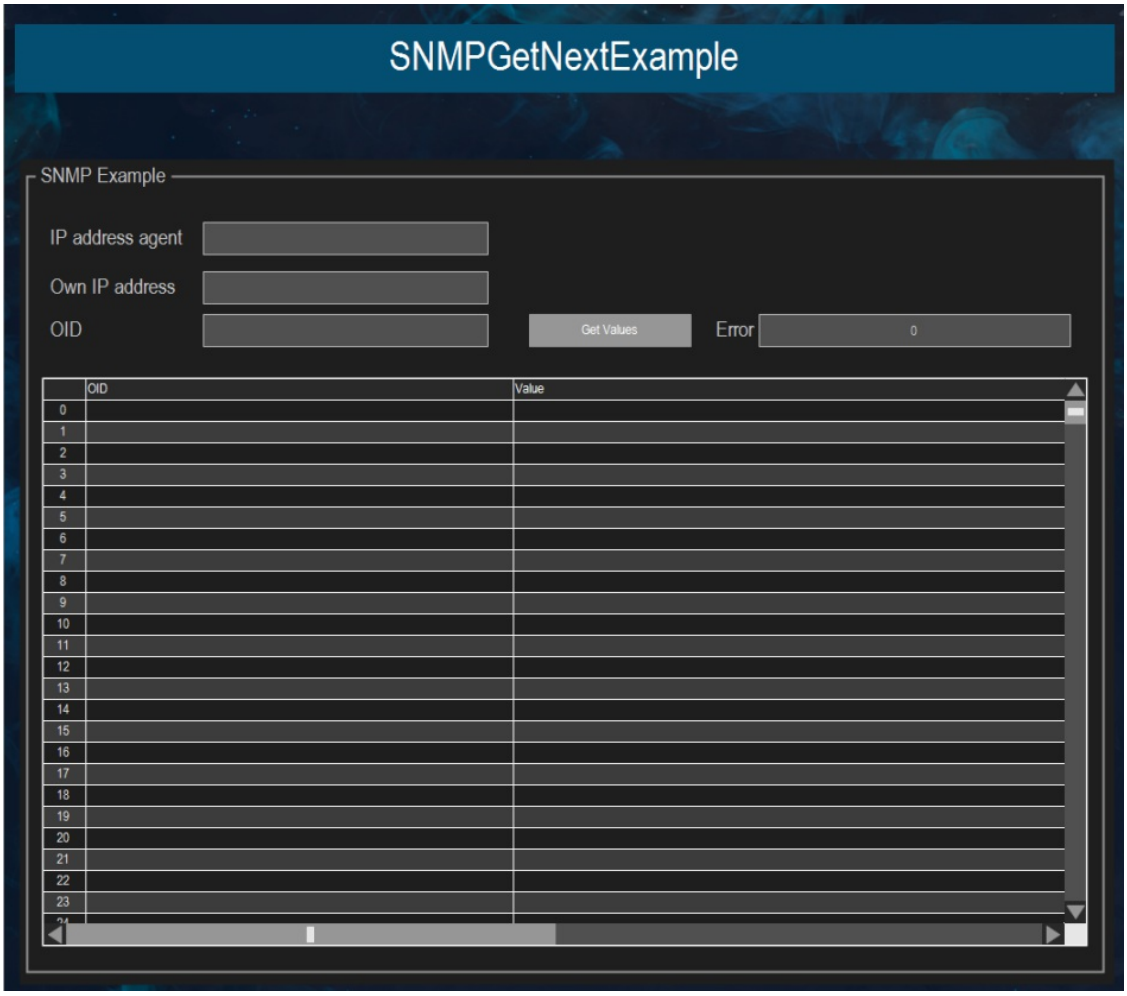

Abbildung 2: SNMPGetNextExample

Beispiel SNMPExample.project / Applikation SNMPMutiGetExample

Das Beispiel wie mehrere OIDs über einen Request abgefragt werden können.

Beispiel SNMPExample.project / Applikation SNMPTimerExample

Das Beispiel zeigt die Verwendung des Funktionsbausteins `SNMP_GET_REQUEST` für einen GET Request. Die Ergebniswerte der Abfragen werden in einer Visualisierung *(siehe Abbildung 3)* dargestellt.

Die IP-Adressen und die OIDs werden in der CSV-Datei `"c:\temp\SNMPConfig.csv"` konfiguriert. Dabei steht in der 1. Spalte die IP-Adresse des Agenten, in der 2. Spalte (getrennt durch ein Semikolon) steht die abzufragende OID. Die letzte Zeile muss einen Zeilenumbruch beinhalten. Die konfigurierten Werte und die Ergebniswerte werden in einer Visualisierung dargestellt. Nachdem die Schaltfäche `"Start Timer"` betätigt wurde werden die Werte zyklisch alle 20 Sekunden abgefragt.

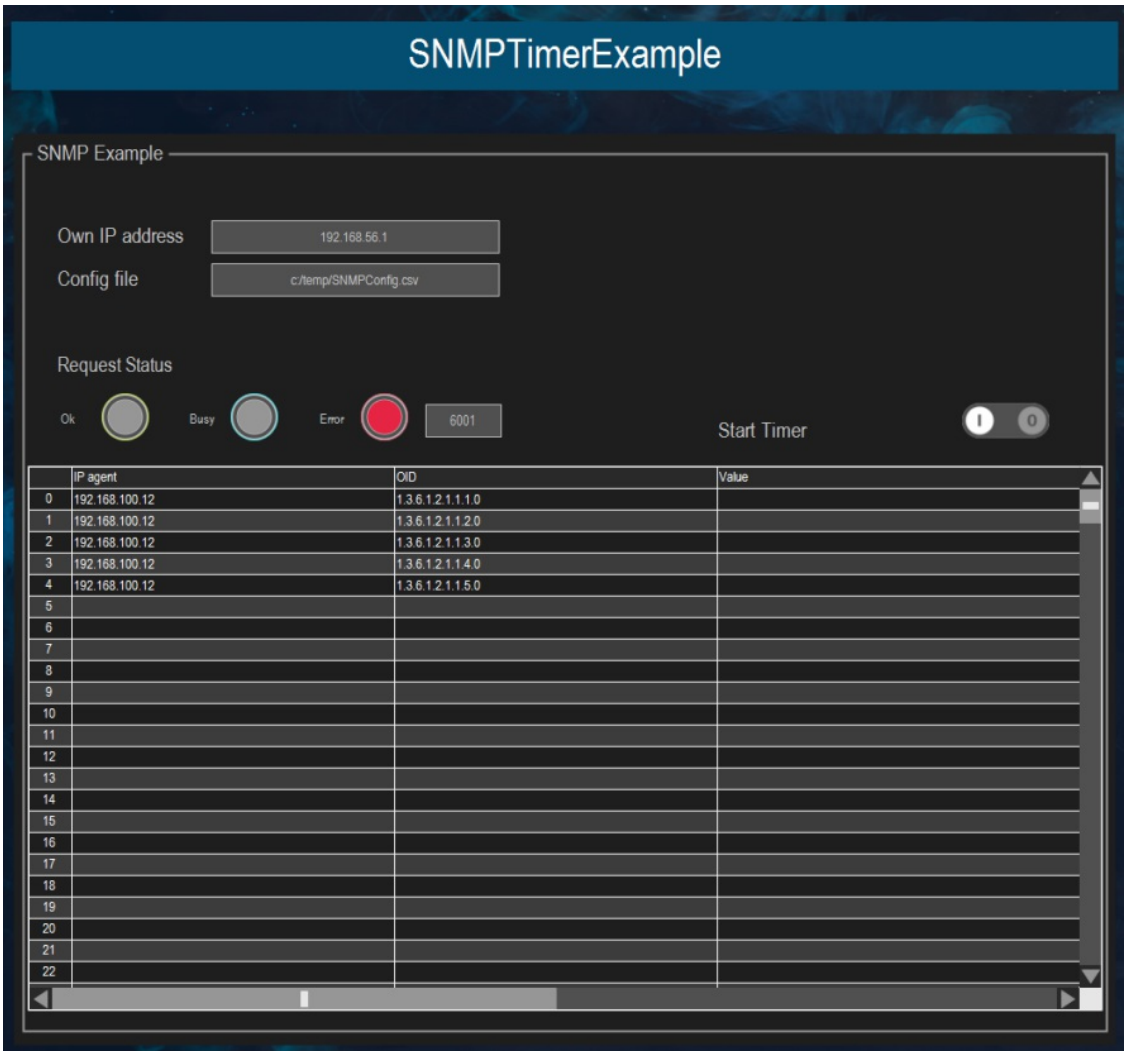

Abbildung 3: SNMPTimerExample

Beispiel SNMPExample.project / Applikation SNMPSetExample

Das Beispiel veranschaulicht die Verwendung des Funktionsbausteins `SNMP_SET`. Über eine Visualisierung *(siehe Abbildung 4)* können SET-Telegramme verschickt werden.

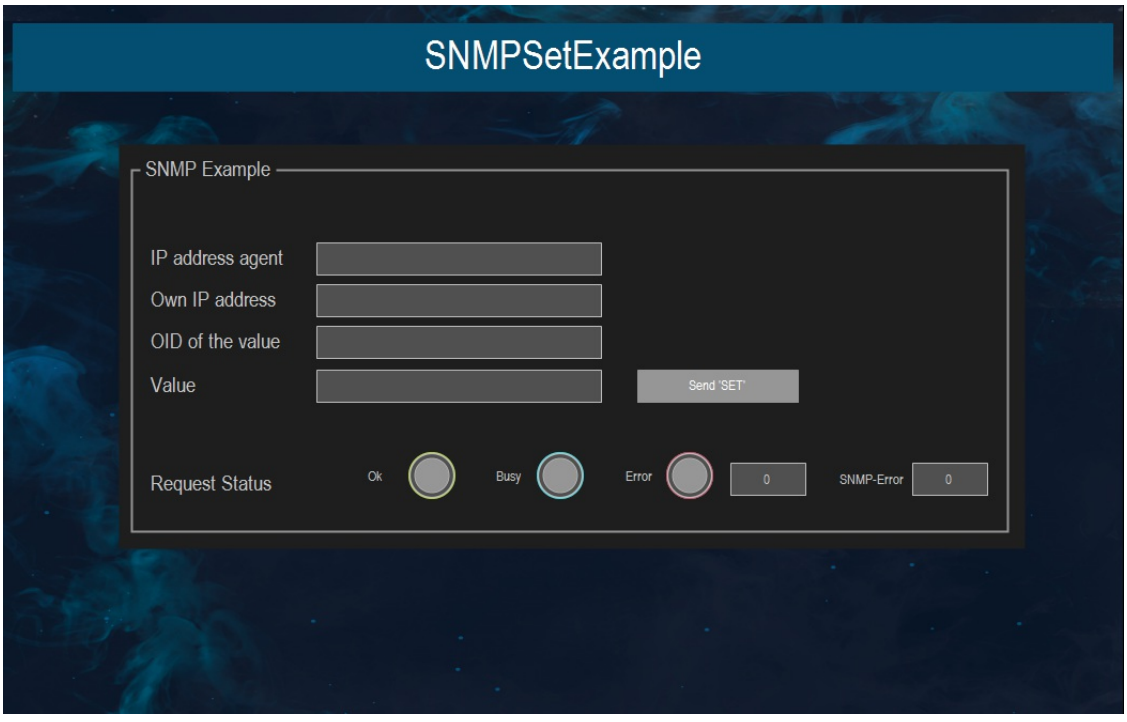

Abbildung 4: SNMPSetExample

Beispiel SNMPExample.project / Applikation SNMPTrapReceiver

Das Beispiel veranschaulicht die Verwendung des Funktionsbausteins `SNMP_TRAP_RECEIVER`. Die empfangenen Werte werden in einer Visualisierung *(siehe Abbildung 5)* dargestellt.

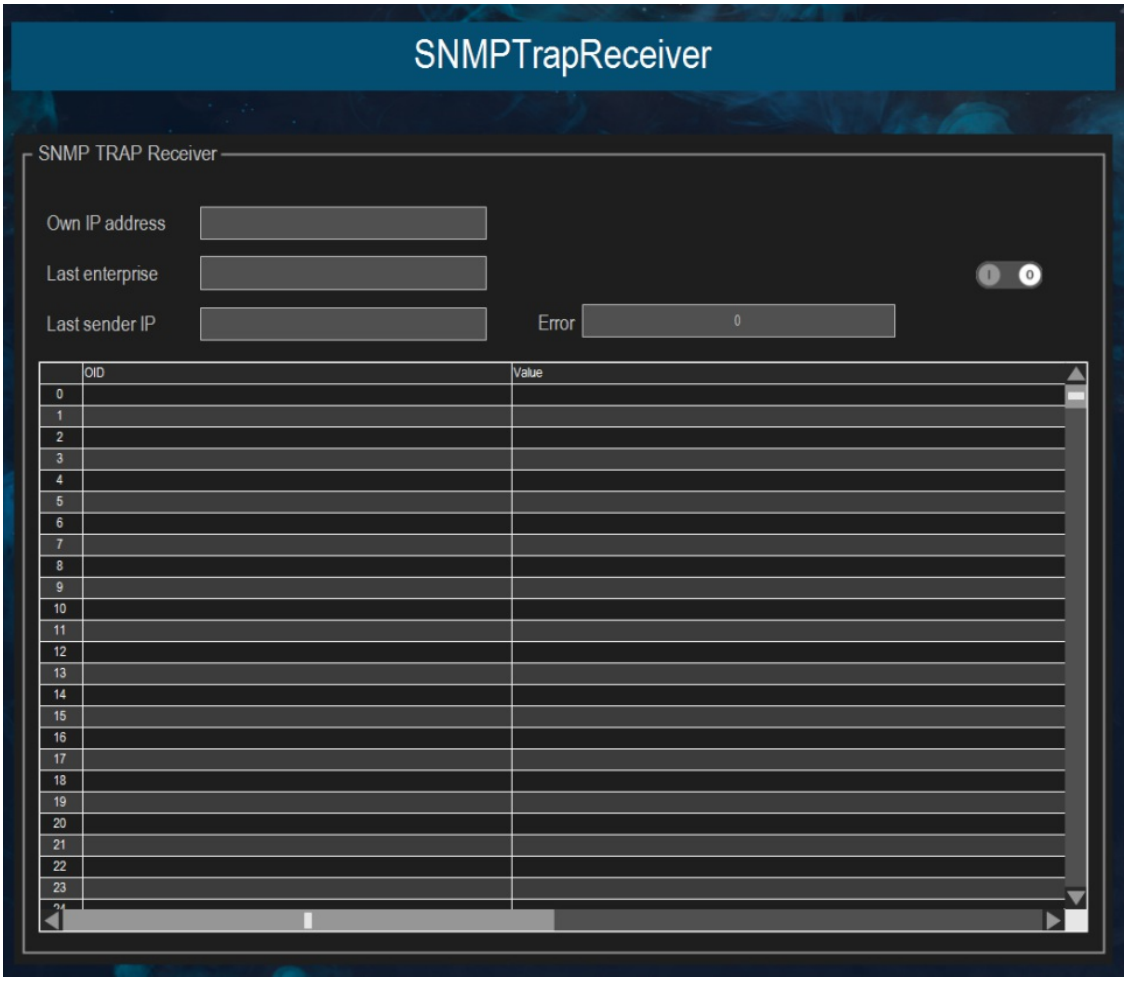

Abbildung 5: SNMPTrapReceiver

Beispiel SNMPExample.project / Applikation SNMPTrapSender

Das Beispiel veranschaulicht die Verwendung des Funktionsbausteins `SNMP_TRAP_SENDER`. Über eine Visualisierung *(siehe Abbildung 6)* können TRAP-Telegramme verschickt werden.

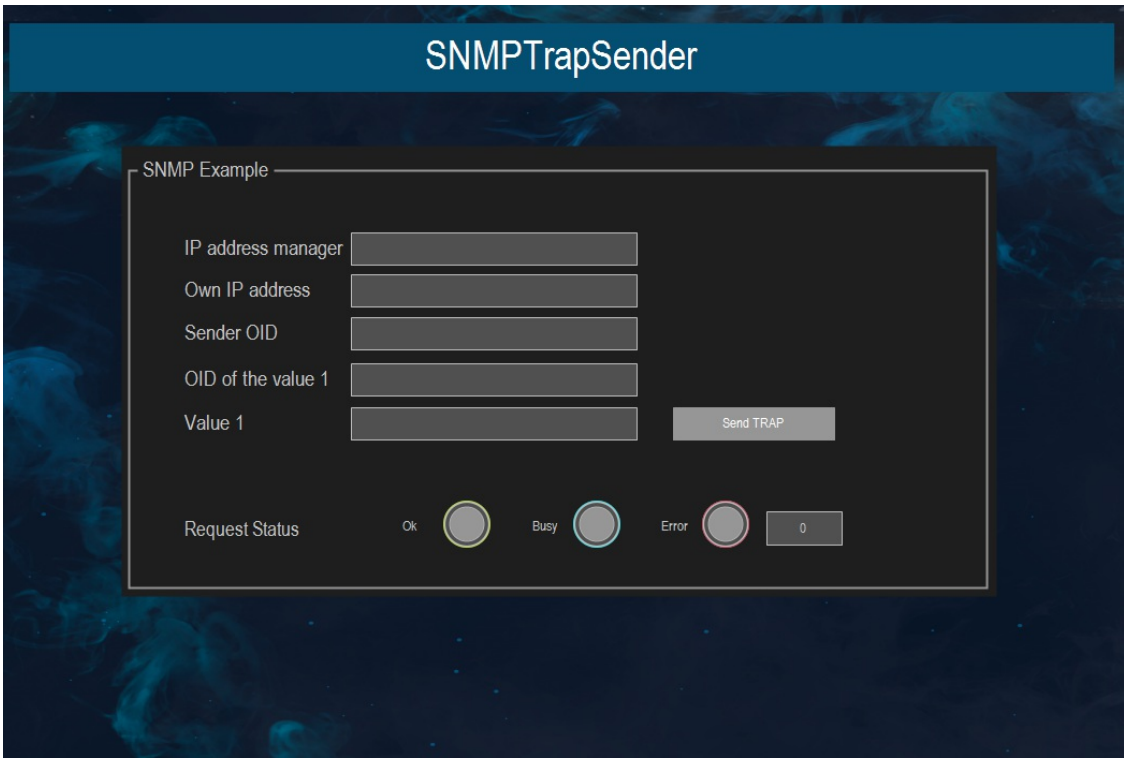

Abbildung 6: SNMPTrapSender

Allgemeine Informationen

Lieferant:

CODESYS GmbH
Memminger Straße 151
87439 Kempten
Deutschland

Support:

https://support.codesys.com

Artikelname:
SNMP Library
Artikelnummer:
2111000005
Vertrieb:

CODESYS Store
https://store.codesys.com

Lieferumfang:
CODESYS Package mit Bibliothek und Beispielen

Systemvoraussetzungen und Einschränkungen

Programmiersystem	CODESYS Development System V3.5.14.0 oder höher
Laufzeitsystem	CODESYS Control V3.5.9.0 oder höher
Unterstützte Plattformen/ Geräte	Alle Hinweis: Verwenden Sie das Projekt ‚Device Reader', um die von der Steuerung unterstützten Funktionen zu ermitteln. ‚Device Reader' ist kostenlos im CODESYS Store erhältlich.
Zusätzliche Anforderungen	-
Einschränkungen	Unterstützte SNMP-Versionen: SNMP V1 und SNMP V2c
Lizenzierung	Arbeitsplatzlizenz: Die Lizenz kann auf dem Arbeitsplatz verwendet werden, auf dem Sie das CODESYS Development System installieren und ausführen. Die Lizenzaktivierung erfolgt auf einem softwarebasierten Lizenz-Container (Soft-Container), der fest an den Arbeitsplatz gebunden ist. Alternativ kann die Lizenz auf einem CODESYS Key (USB-Dongle) hinterlegt werden. Durch Umstecken des CODESYS Keys kann die Lizenz an einem anderen Arbeitsplatz genutzt werden.
Erforderliches Zubehör	CODESYS Key for CODESYS < 3.5.14.0

Bitte beachten Sie: Nicht alle CODESYS-Funktionen sind in allen Ländern verfügbar. Weitere Informationen zu diesen länderspezifischen Einschränkungen erhalten Sie unter sales@codesys.com.
Bitte beachten Sie: Technische Änderungen, Druckfehler und Irrtümer vorbehalten. Es gilt der Inhalt der aktuellen Online-Version dieses Dokuments.