ARDUINO SHIELD – BRIGHTNESS CONTROL

Dieses Dokument beinhaltet eine Einführung in die Nutzung des Arduino Shield mit der WinErs Laborversion.

INHALTSVERZEICHNIS

1 EINFÜHRUNG

Das Arduino Shield – Brightness Control ist das kompakte Modell einer Helligkeits-Regelungsstrecke für den Einsatz in der Ausbildung in Regelungstechnik. Bestehend aus einem Arduino MKRZERO, einer dimmbaren LED und einem Helligkeitssensor kann das Arduino Shield in Kombination mit der WinErs Laborversion für Regelkreisoptimierung oder für das Aufsetzen eines Regelkreises genutzt werden.

1.1 WINERS LABORVERSION

Die WinErs-Laborversion ist eine eingeschränkte Version des Prozessleit- und Automatisierungssystems WinErs zum Automatisieren, Simulieren und Experimentieren. Die Anzahl der Signale ist festgelegt auf:

- 32 binäre Eingänge,
- 32 binäre Ausgänge,
- 16 analoge Eingänge,
- 8 analoge Ausgänge,
- 80 binäre Merker,
- 80 analoge Merker.

Über unterschiedliche Prozessschnittstellen ist der Anschluss an den Prozess/Anlage möglich.

Die WinErs-Laborversion eignet sich für die Erstellung dynamischer Simulationen und für die Automatisierung von Anlagen und Prozessen.

Über die Prozessvisualisierung können eigene Bedien- und Beobachtungsoberflächen erstellt werden. Die Steuerungen und Regelungen sowie Simulationen werden auf einfache Weise grafisch durch den Aufbau von Blockstrukturen und Funktionsplänen eingegeben, so dass keine Programmierung notwendig ist. Es steht eine umfangreiche Bibliothek von analogen und binären Blöcken für Prozesssimulationen, sowie Regelungs- und Steuerungsaufgaben zur Verfügung. So existieren neben den aus der Regelungstechnik üblichen Blöcken auch Blöcke für logische Verknüpfungen (und, oder, nicht, excl.-oder, etc.) und mathematische Berechnungen sowie Fuzzy-Controller. Ablaufsteuerungen werden mit Hilfe von GRAFCET-Plänen nach IEC 60848 realisiert. Prozessmodelle können auch direkt als Differentialgleichungen eingegeben und als Block in das Blockschaltbild eingebunden werden. Zusätzlich steht für komplexe Prozessmodelle oder eigene Algorithmen eine DLL-Schnittstelle als Block zur Verfügung.

Signalverläufe werden über die Trenddarstellungen oder über die Messwertspeicherung (grafische und statistische Auswertung gespeicherter Signalwerte) überwacht und ausgewertet.

Mit der integrierten Prozessvisualisierung können zur Präsentation, Bedienung und Überwachung selbst gestaltete Prozessbilder erstellt werden. Der Prozessbildeditor umfasst alle Möglichkeiten, die in der Prozessleittechnik mit dem Prozessleitsystem WinErs für Visualisierungen gegeben sind.

Die WinErs-Laborversion wird mit einem „Beispielprojekt“ ausgeliefert. Das Beispielprojekt beinhaltet:

- Bedienoberfläche für die Praktikumsanlage LC2030 für einen leichten Einstieg in die Entwicklung eines Prozessleitsystems für diese Anlage
- Bedienoberfläche für das Arduino Shield – Brightness Control, zur Regelkreisoptimierung einer Helligkeitsregelung
- Bedienoberfläche für den GRAFCET Kursus, wahlweise mit oder ohne Arduino Shield – GRAFCET
- Simulation einer Füllstandsregelung mit PID-Regler und Messwerterfassung
- Simulation einer Füllstandbehälter-Steuerung über Logik-Bausteine
- Simulation einer Temperaturregelung mit PID Regler
- Simulation einer Ampelsteuerung über GRAFCET

2 KOMMUNIKATION EINRICHTEN

Um die Kommunikation zwischen Arduino Shield und Laborversion einzurichten, sind folgende Schritte notwendig.

2.1 AUSWAHL DES TREIBERS

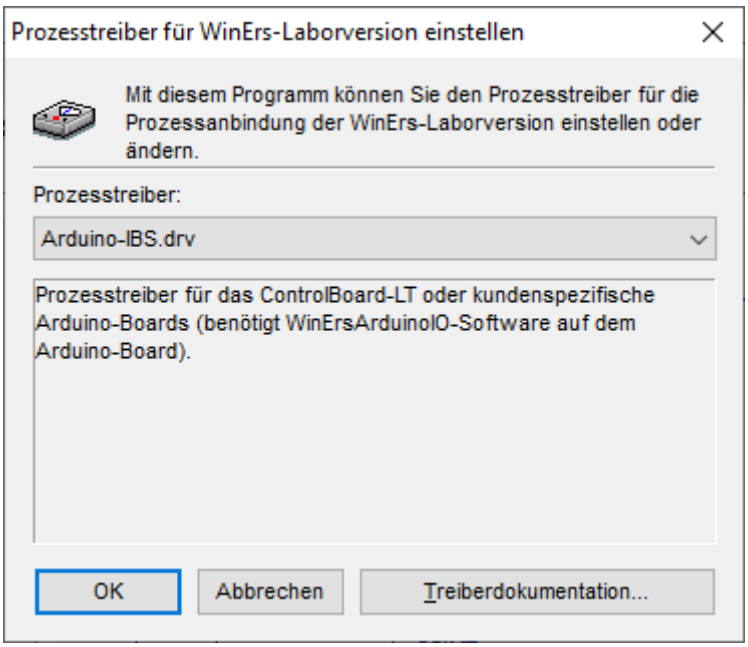


ABBILDUNG 1 AUSWAHLDIALOG DER TREIBERS FÜR DIE WINERS LABORVERSION

Nach Installation der WinErs Laborversion öffnet sich ein Fenster mit dem Auswahldialog des Treibers (Abbildung 1). Ist WinErs bereits auf dem PC installiert kann dieser Auswahldialog über das *Startmenü* unter *WinErs Labor* → *WinErs Labor – Treiber auswählen* geöffnet werden. Das Dialogfeld muss als Administrator ausgeführt werden, da die Einstellung in der Programmpartition gespeichert wird (*rechte Maustaste* → *Mehr* → *Als Administrator ausführen*).

Für die Nutzung des Arduino Shield muss hier der Treiber Arduino-IBS.drv ausgewählt werden.

2.2 Auswahl der COM-Schnittstelle

Die Auswahl der COM-Schnittstelle ist notwendig, damit der PC weiß, welcher USB Port auf den Arduino zugreift.

Die vom Arduino Shield verwendete COM-Schnittstelle wird im Gerätemanager angezeigt (Abbildung 2).

Die Auswahl der COM-Schnittstelle für die WinErs Laborversion erfolgt im entsprechenden Dialogfenster (Abbildung 3), das über *Startmenü* → *WinErs Labor* → *WinErs Labor – COM-Schnittstelle einstellen.* Nach Auswahl der COM Schnittstelle muss WinErs neu gestartet werden.

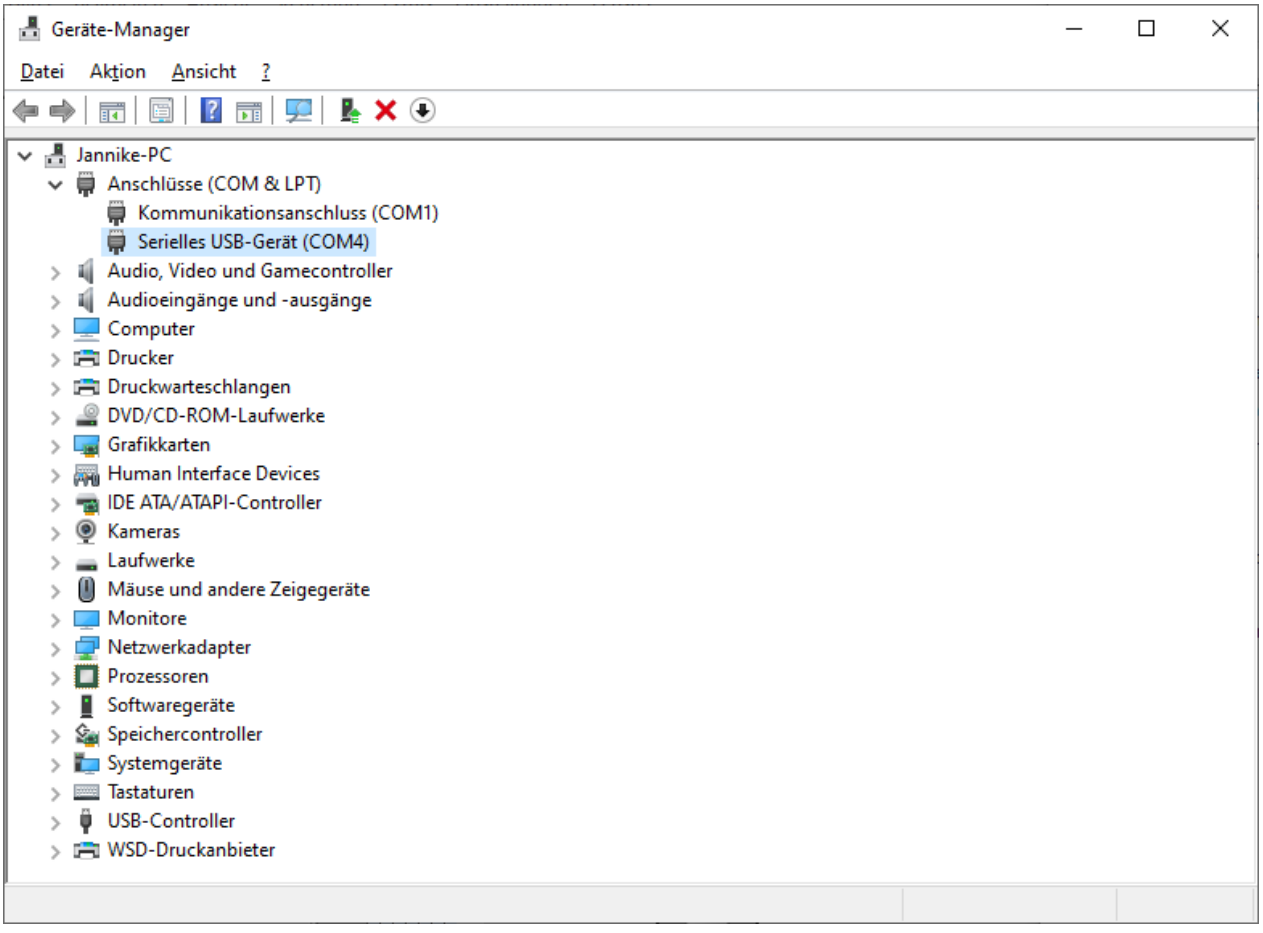


ABBILDUNG 2 ARDUION SHIELD IM GERÄTEMANAGER AN DER COM-SCHNITTSTELLE COM4

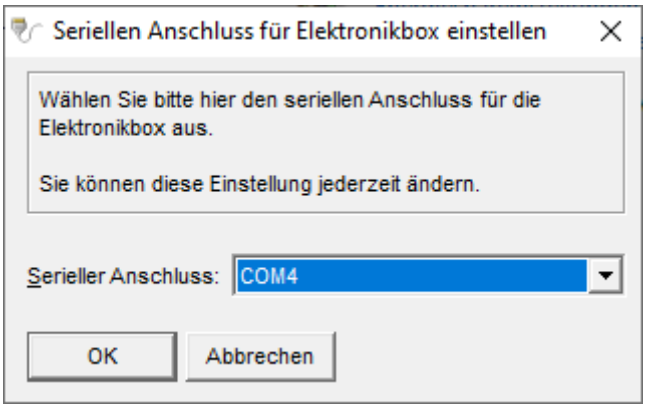


ABBILDUNG 3 COM SCHNITTSTELLE FÜR DIE WINERS LABORVERSION AUSWÄHLEN

2.3 KOMMUNIKATION ÜBERPRÜFEN

Um die Kommunikation zu überprüfen, muss das Hintergrundfenster namens WRPServ in den Vordergrund geholt werden (Abbildung 4).

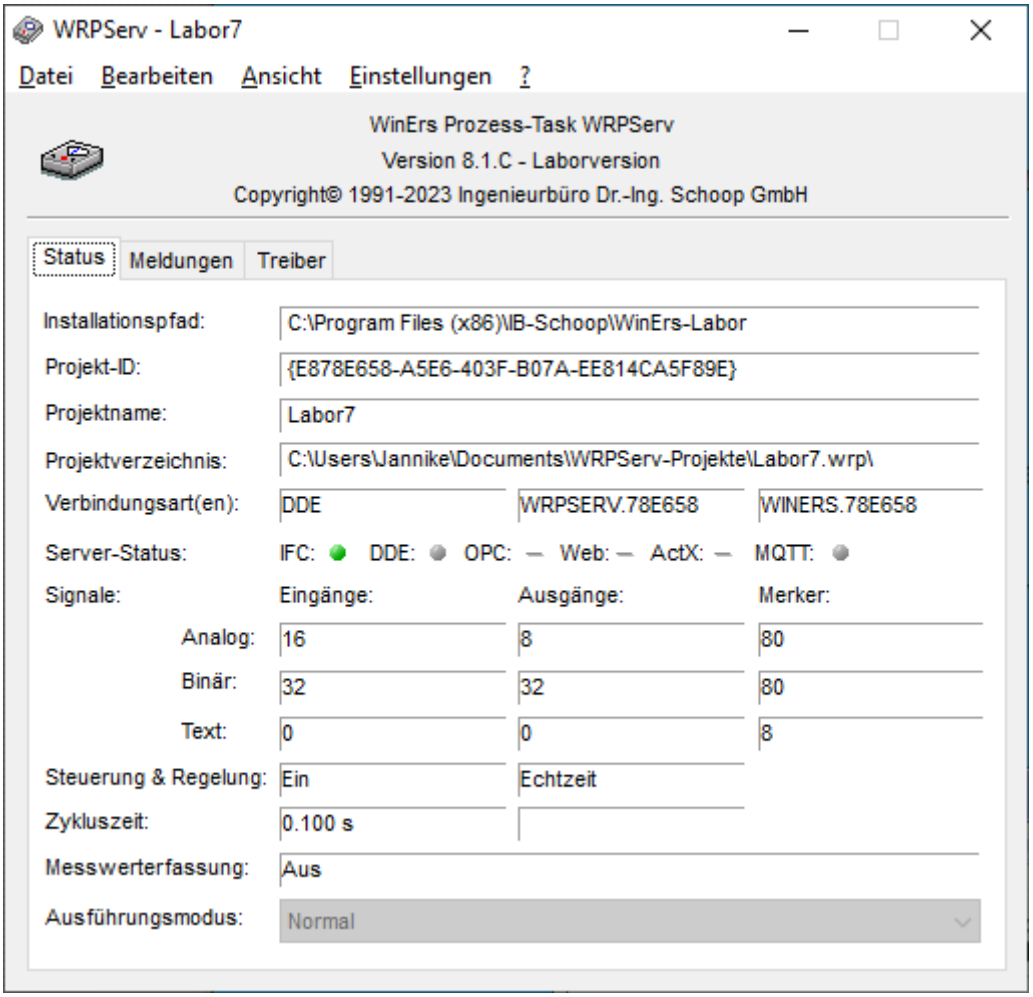


ABBILDUNG 4 DAS WRPSERV DIENT ZUR KOMMUNIKATION MIT DEM ARDUINO

Im Startfenster werden der Dateispeicherort, sowie unter anderem die Anzahl der Signale angezeigt.

In dem Tab Meldungen werden Fehlermeldungen bei Kommunikationsfehlern angezeigt.

In dem Tab Treiber wird der geladene Treiber angezeigt. In diesem Fall der Treiber Arduino-IBS.drv. Durch Doppelklicken auf den Treiber wird das Statusfenster angezeigt (Abbildung 5). Stimmt die Anzahl der gesendeten Pakete mit der der empfangenen überein, ist die Kommunikation in Ordnung.

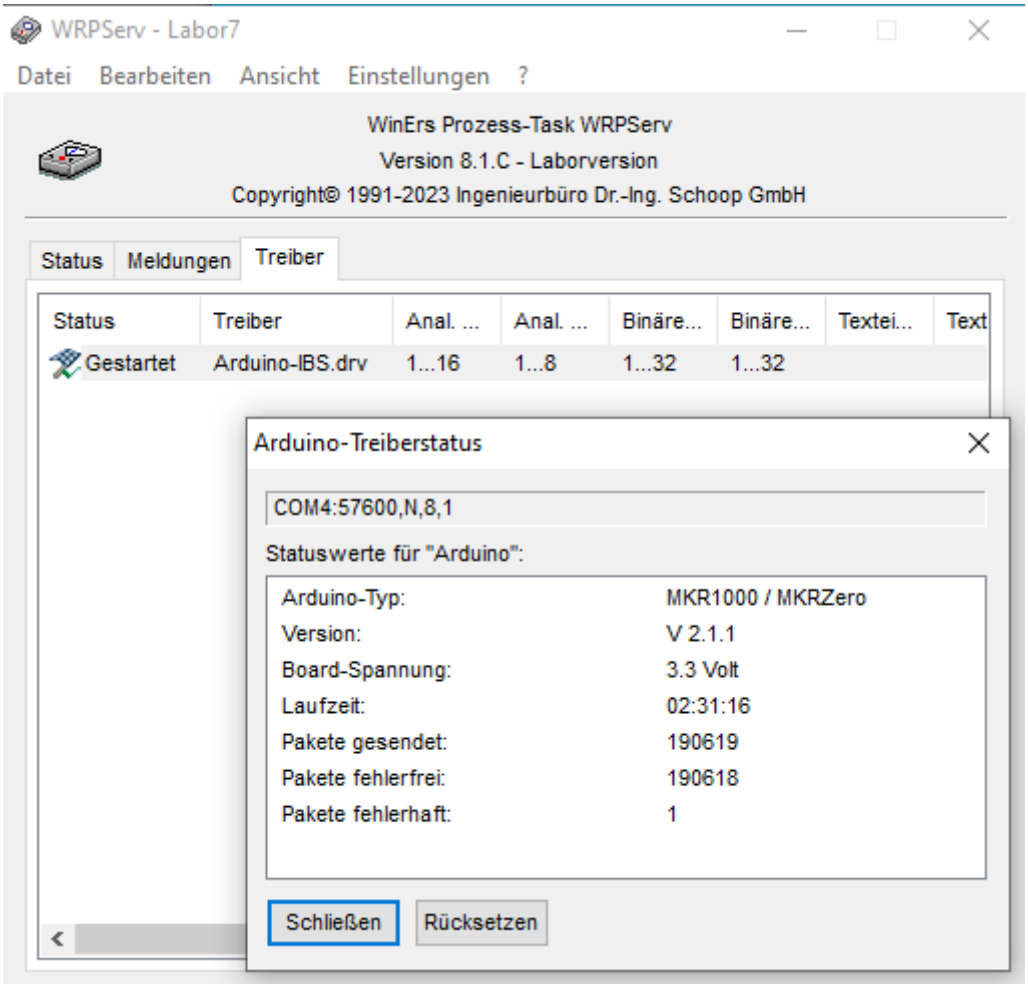


ABBILDUNG 5 ANZEIGE DER TREIBER IM WRPSERV MIT GEÖFFNETEM STATUSFENSTER

2.4 PROZESSTREIBER EINRICHTEN

Der Prozesstreiber muss im Normalfall nicht eingerichtet werden. Die WinErs Laborversion wird mit der richtigen Treiberkonfiguration für das gelieferte Arduino Shield ausgeliefert. Sollte es Probleme bei der Kommunikation geben, lesen Sie das Kapitel 4 Trouble Shooting.

Sollte der Arduino für andere Zwecke genutzt werden, kann es nützlich sein den Treiber anders einzurichten.

Dafür muss im WRPServ Unter *Einstellungen* → *Prozesstreiber einrichten…* aufgerufen werden. Dann wird der Treiber *Arduino-IBS.drv* markiert und die *Einrichten* Schaltfläche geklickt. Die Nachfrage ob der Treiber gestoppt werden darf, kann mit Ja quittiert werden. Es öffnet sich das Arduino-Treibereinstellungen Fenster (Abbildung 6).

Hier gibt es die Möglichkeit die COM-Schnittstelle zu ändern und die Arduino Pins neu zu konfigurieren. Außerdem können Sie die Arduino Software speichern um einen neuen Arduino einzurichten. Weitere Informationen finden sich in der Hilfe.

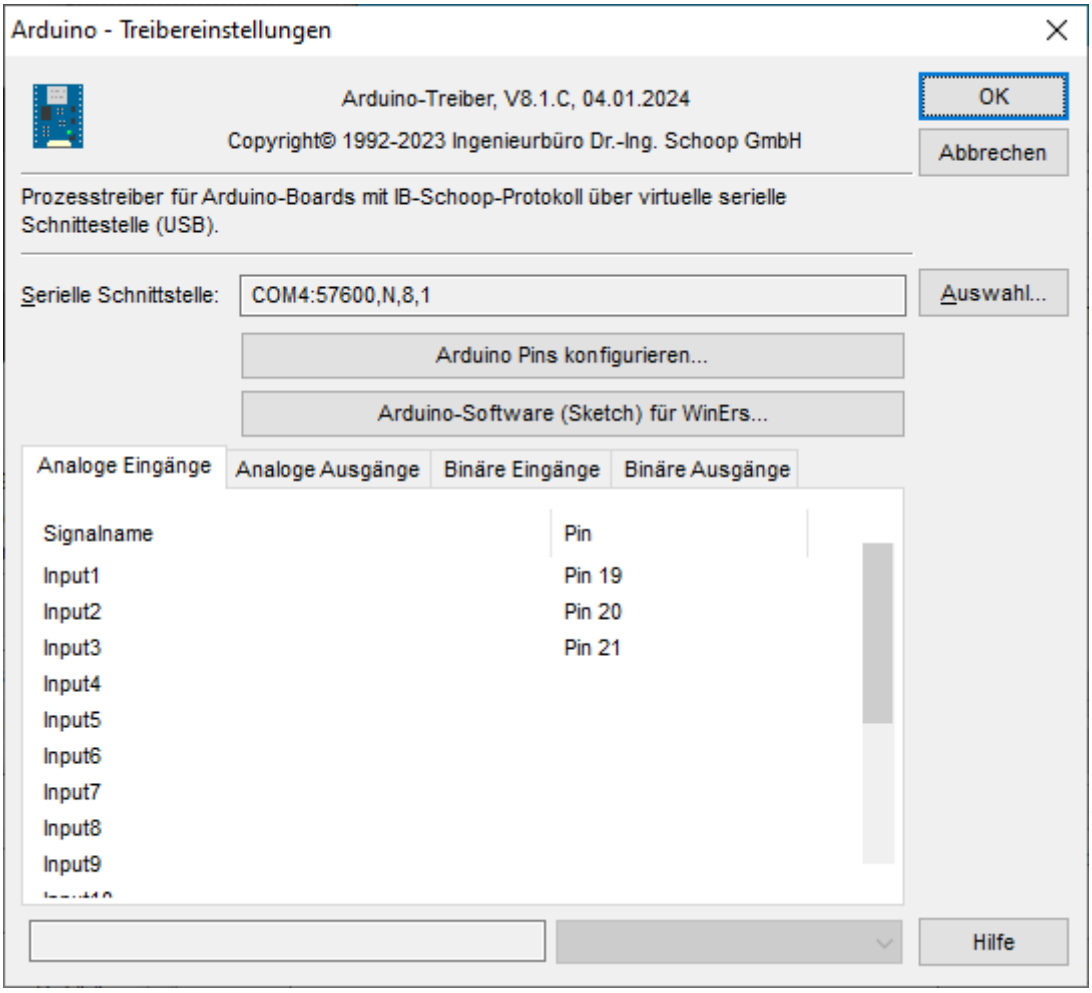


ABBILDUNG 6 ARDUINO TREIBEREINSTELLUNEGN, DIESE WERDEN ERREICHT ÜBER *EINSTELLUNGEN → PROZESSTREIBER EINRICHTEN…*

2.5 SIGNALZUORDNUNG

Für das Arduino Shield – Brightness Control müssen ein analoger Ausgang und ein analoger Eingang zugewiesen sein. Folgende Tabelle zeigt die Zuordnung im Treiber:

TABELLE 1SIGNALZUORDUNG FÜR DAS ARDUINO SHIELD - BRIGHTNESS CONTROL

Signalname	Beschreibung	Zuordnung
Input1	Rohsignal für den Helligkeitssensor	Pin 19
Output1	Rohsignal für die Ansteuerung der LED	Pin 15

3 Arbeiten mit dem Beispielprojekt

Das Beispielprojekt wird über *Startmenü* → *WinErs-Labor* → *WinErs-Labor – Beispielprojekt* geöffnet.

3.1 Interface zur Bedienung

Zur Bedienung des Arduino Shields dienen die Prozessbildseite Brightness Control und die Blockstrukturseite LDR. Alle anderen Blockstrukturen können deaktiviert werden. Das Prozessbild kann über Doppelklick in der Projektleiste geöffnet werden (Abbildung 8).

Hier können die Reglerparameter eingestellt werden und die Führungsgröße eingestellt werden. Möchte man ein Störverhalten erzwingen kann die Umgebungshelligkeit des Shields durch Abdecken manipuliert werden, oder es kann ein Finger zwischen Sensor und LED gehalten werden.

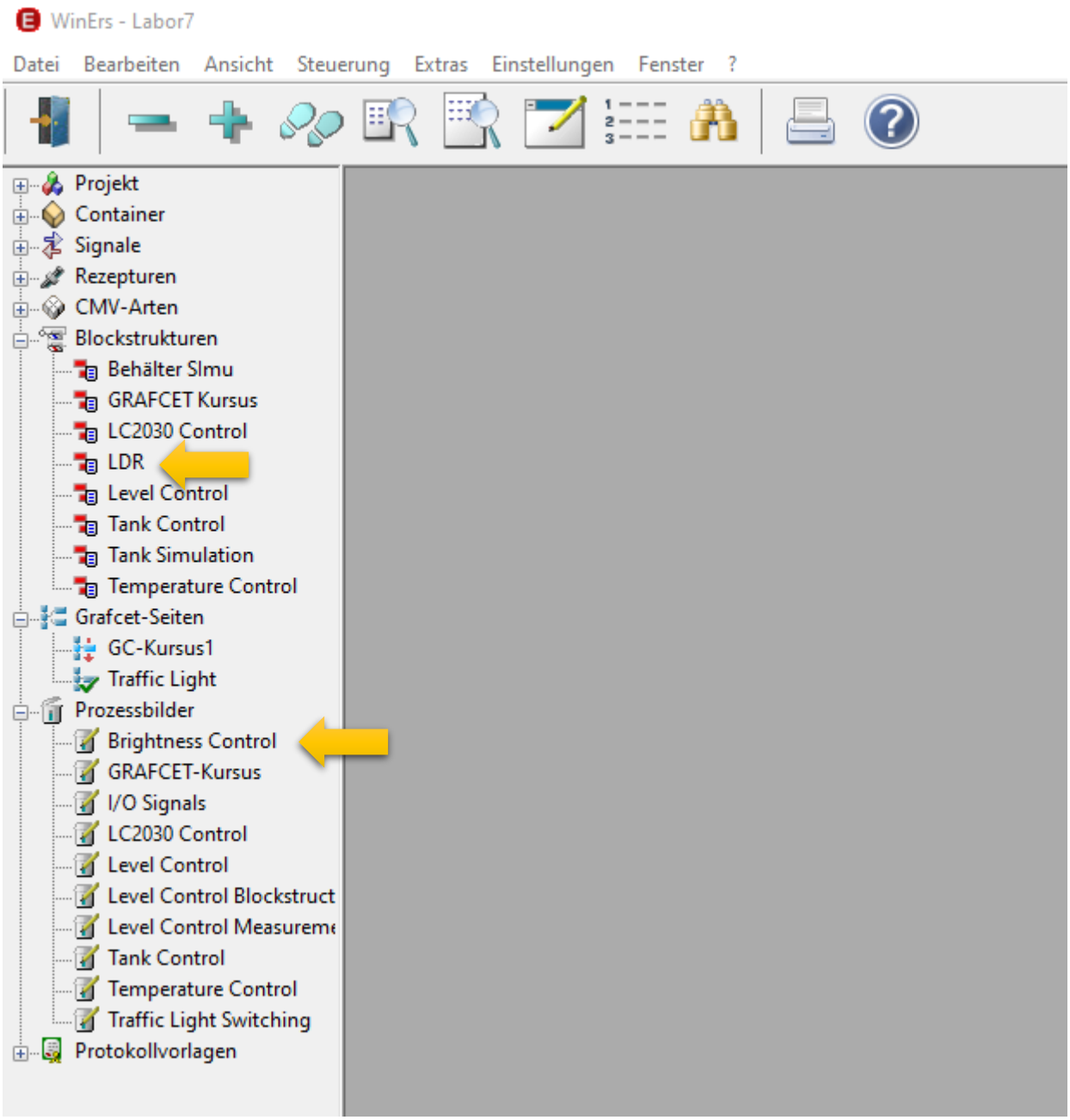


ABBILDUNG 7 WINERS LABORVERSION MIT GEÖFFNETER PROJEKTLEISTE. MARKIERT SIND ALLE ELEMENTE, DIE ZU DER HELLIGKEITSREGELUNG MIT DEM ARDUINO SHIELD GEHÖREN.

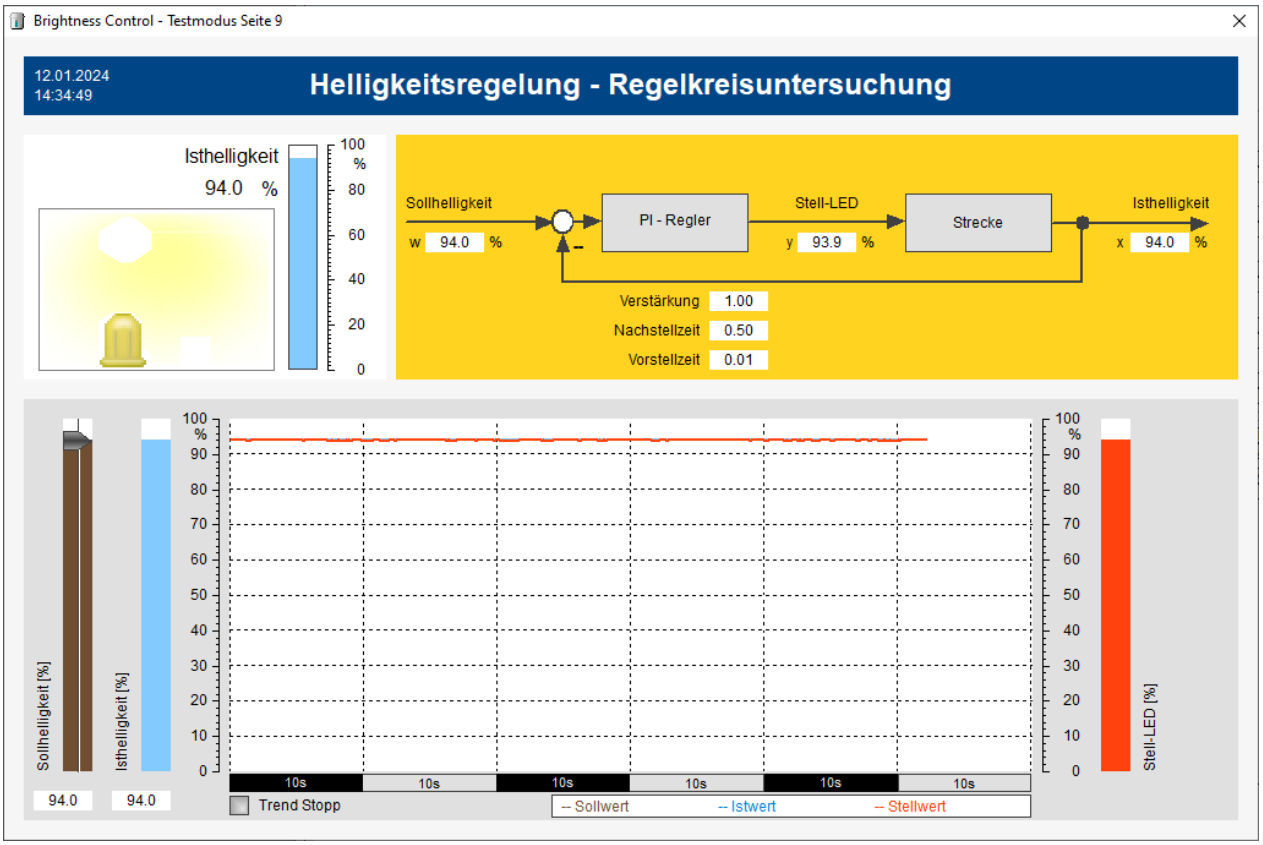


ABBILDUNG 8 PROZESSBILD ZUR BEDIENUNG DES ARDUINO SHIELDS. ÜBER DEN ENTSPRECHENDEN SCHIEBEREGLER KANN DIE SOLLHELLIGKEIT ANGEPASST WERDEN.

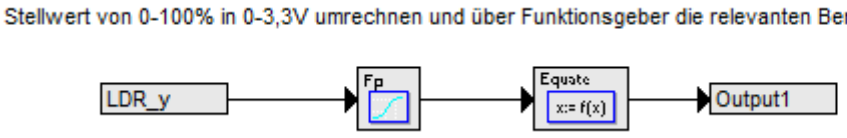
3.2 BLOCKSTRUKTUR MIT REGELKREIS

Die Blockstrukturansicht wird durch Doppelkick auf LDR unter Blockstrukturen geöffnet.

Die Blockstruktur ist in vier Bereiche eingeteilt.

3.2.1 STELLWERT IN EIN AUSGANGSSIGNAL UMRECHNEN

Folgende Blockstruktur dient dem Umrechnen des Stellsignals:



Der Arduino kann einen Eingang von 0 – 3,3V verarbeiten. Das Stellsignal LDR_y ist von 0-100% definiert. Also müssen das Signal entsprechend umgerechnet werden. Dafür wird ein Gleichungsblock verwendet (Abbildung 9).

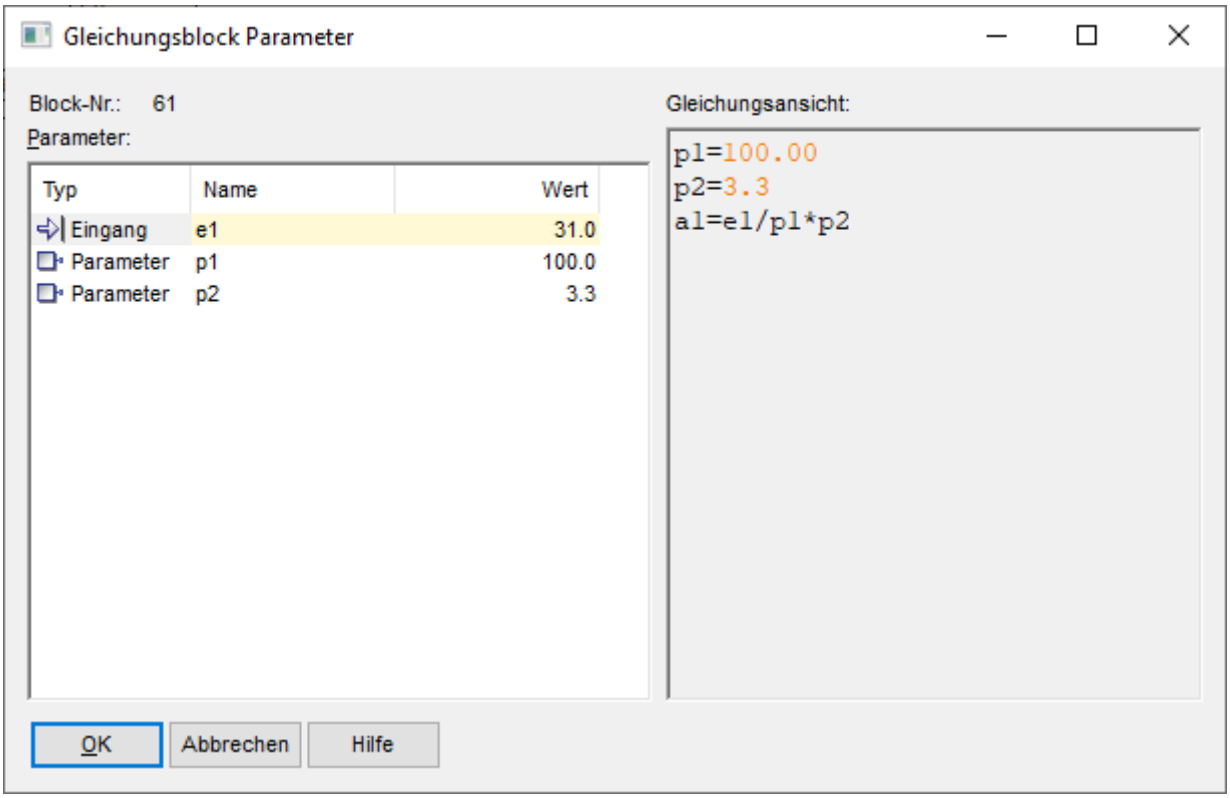


ABBILDUNG 9 UMRECHNUNG DES STELLSIGNALS LDR_Y (0-100%) IN DAS AUSGANGSSIGNAL (0-3,3V)

Zusätzlich zu der Umrechnung wird das Signal skaliert. Das liegt an dem Dimmverhalten einer LED, die nur in einem sehr kleinen Bereich der 0-3,3V auf Änderungen reagiert, dafür aber dann sehr stark.

Um dieses Verhalten zu linearisieren wird hier ein Programmgeber (Abbildung 10) eingesetzt, der je nach Eingangswert den Ausgang abhängig von einer grafisch eingegebenen Funktion setzt. Diese Funktion ergibt sich durch Aufzeichnung der Helligkeit bei Änderung des Stellwertes der LED und anschließender Invertierung, mit dem Ziel einer annähernden Linearisierung des Verhaltens

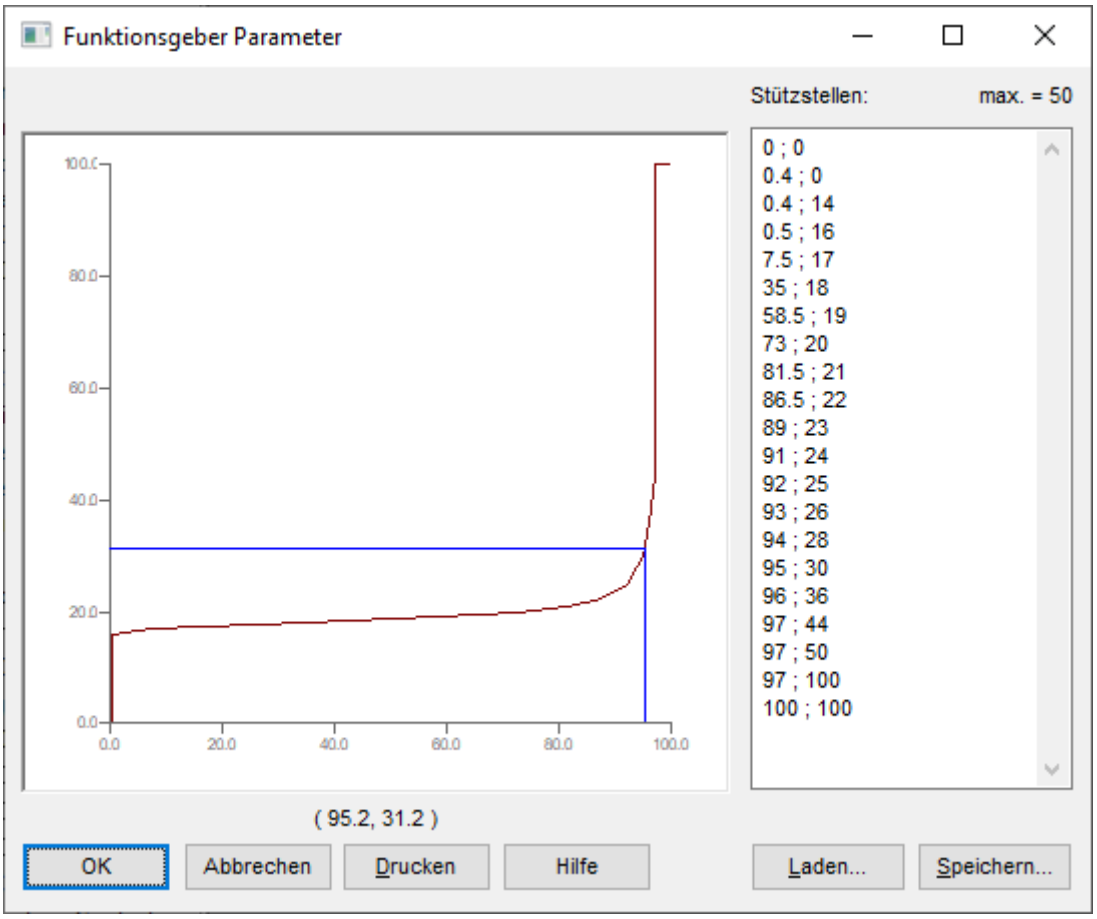
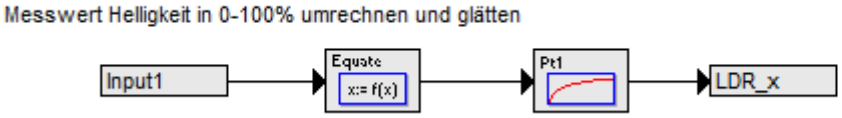


ABBILDUNG 10 PROGRAMMGEBER ZUR LINEARISIERUNG DES ANTWORTVERHALTENS DER LED AUF STELLWERTÄNDERUNGEN

3.2.2 EINGANGSSIGNAL IN DIE HELLIGKEIT UMRECHNEN



Das Helligkeitssignal wird wieder von 0-3,3V umgerechnet in 0-100%, dafür wird ein Gleichungsblock verwendet (Abbildung 11).

Zusätzlich wird das Signal mithilfe eines Pt-1 Blocks geglättet (Abbildung 12), um einen sauberen Signalverlauf zu erhalten und um eine sprungfreie Regelung zu erhalten.

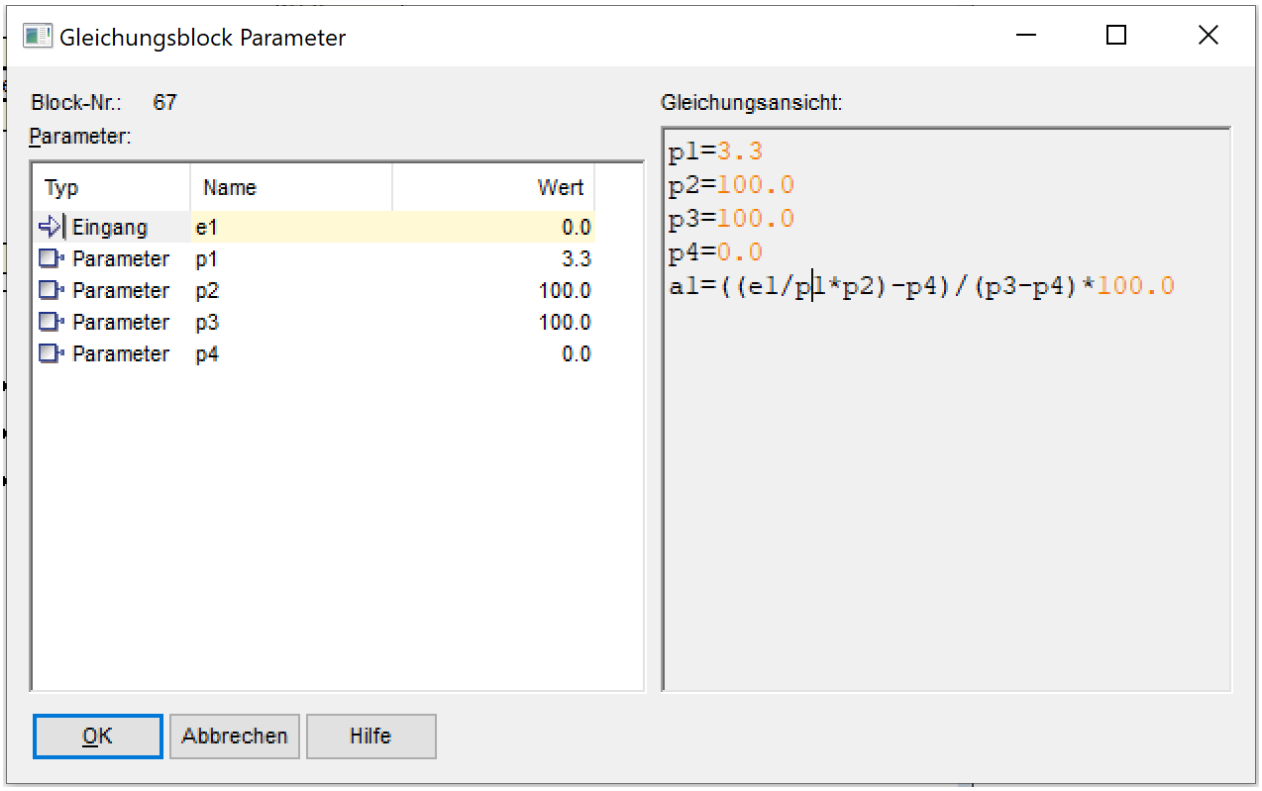


ABBILDUNG 11 UMRECHNUNG DES EINGANGSSIGNALS (0-3,3V) IN DEN ISTWERT DER HELLIGKEIT (0-100%)

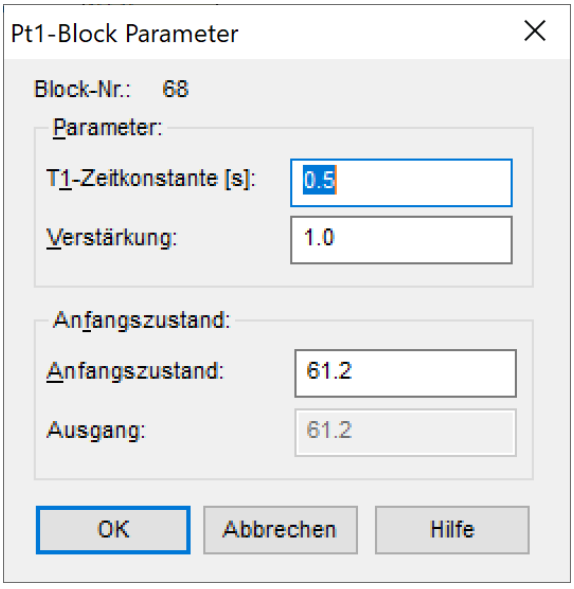


ABBILDUNG 12 PT-1 BLOCK ZUR GLÄTTUNG DES HELLIGKEITSEINGANGSSIGNALS

3.2.3 REGELUNG AUSFÜHREN

Für den Regelkreis wird die Regelabweichung berechnet und dient dann als Eingang in den PID-Regler. Das Ausgangssignal des Reglers ist der Stellwert.

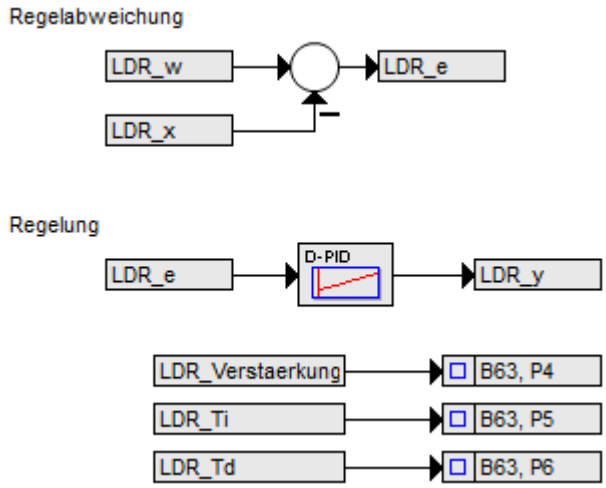


ABBILDUNG 13 BLOCKSTRUKTUR ZUR REGELUNG DER HELLIGKEITSSTRECKE

Der Regler wird in den Automatikbetrieb gestellt und auf einen PI-Regler beschränkt. Der Stellwert wird auf 0-100% begrenzt, weil andere Stellwerte nicht verarbeitet werden können (Abbildung 14). Die Reglerparameter werden von den Signalen LDR_Verstärkung, LDR_Ti und LDR_Td vorgegeben und über Parametersetzblöcke in den Reglerblock geschrieben.

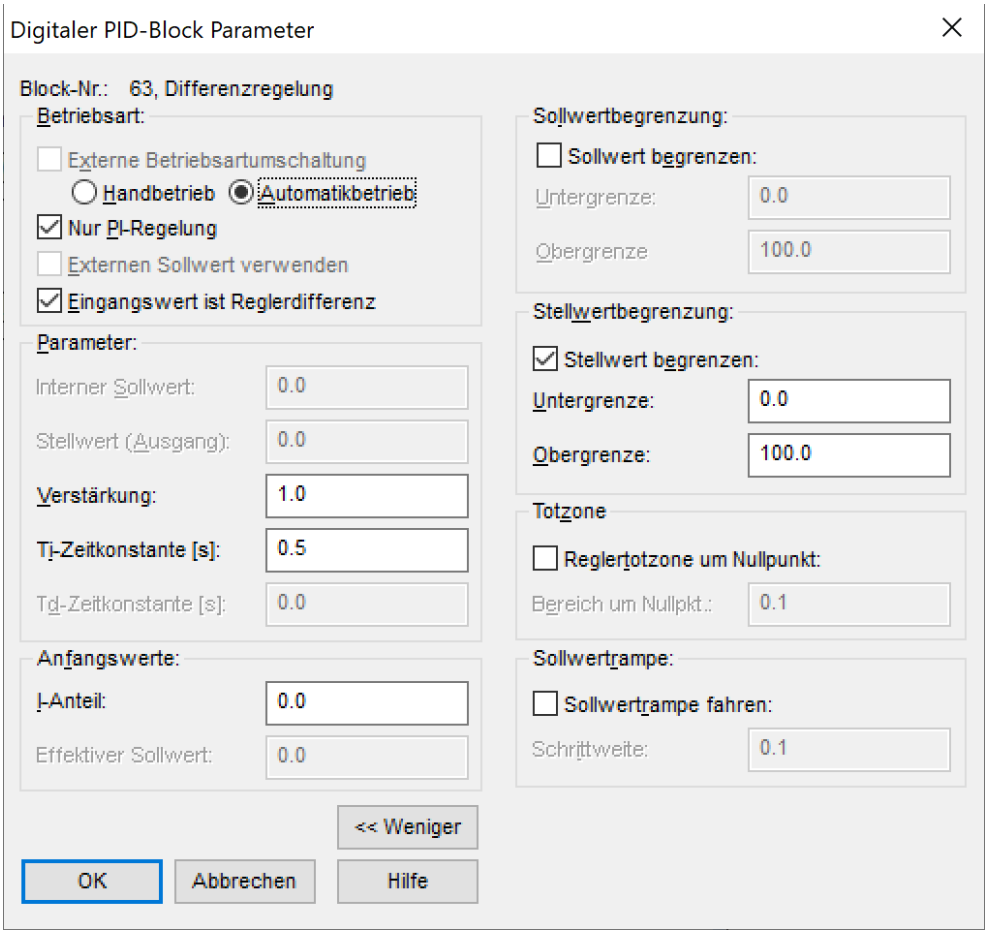


ABBILDUNG 14 EINSTELLUNGEN DES PID-REGLERBLOCKS ZUR HELLIGKEITSREGELUNG

4 TROUBLE SHOOTING

In diesem Abschnitt werden bekannte Schwierigkeiten bei der Inbetriebnahme beschrieben. Bei einem Fehler, der hier nicht beschrieben ist, melden Sie sich bitte bei uns.

4.1 DAS ARDUINO SHIELD VERBINDET NICHT KORREKT

Wenn bereits eine Version der Laborversion auf dem PC installiert war, kann es sein, dass die Treiberkonfiguration nicht korrekt überspielt wird. Dann sollte WinErs deinstalliert werden und das Beispielprojekt, sowie das WRPServ Verzeichnis des Beispielprojekts gelöscht werden. Anschließend kann die Laborversion dann neu installiert werden.

Das Projektverzeichnis ist im Normalfall: C:\ProgramData\IB-Schoop\Labor7. Das Projektverzeichnis kann über *Datei* → *Projektverzeichnis anzeigen…* angezeigt werden.

Das WRPServ Projektverzeichnis liegt im Normalfall unter: C:\Users\User\Documents\WRPServ-Projekte\Labor7.wrp\. Das Projektverzeichnis wird auf der Startseite des WRPServ angezeigt.