Software

ARDUINO - Kommunikation mehrerer ARDUINI miteinander

gaulois

#1/5 Verfasst am: 03 Okt 2018 9:50 Titel: ARDUINO - Kommunikation mehrerer ARDUINI miteinander

Hallo zusammen,
in den anderen Arduino-threads wird teilweise vorausgesetzt, dass mehrere Geräte miteinander kommunizieren.

Das spezielle Kommunikations-Thema sollte diese threads aber nicht "belasten", daher eröffne ich hier mal einen thread, der sich nur damit beschäftigen soll, wie ein Arduino mit einem oder mehreren "redet".

Stichworte sind:

Serielle Schnittstellen,
Verbindung zweier Geräte,
Daisy-Chain mit beliebig vielen Geräten,

Hier sollen nur Fragen zum Datenaustausch der Geräte miteinander diskutiert werden. Ich denke, das ist speziell genug für einen eigenen thread.

gaulois

#2/5 Verfasst am: 03 Okt 2018 11:28 Titel:

Standardmäßig verfügt der Arduino über zwei serielle Schnittstellen, nämlich den USB-Port, über den er auch programmiert wird und über die Pins 0 und 1, die laut Arduino.cc über eine entsprechende Routine zwischen zwei Durchläufen des Hauptprogramms "loop" abgefragt werden. Dazu gibt es folgende Routine, die allerdings mit Strings arbeitet, was in C++ nicht unproblematisch ist:

```
void serialEvent() {
  while (Serial.available()) {
    // neues Byte einlesen:
    char inChar = (char)Serial.read();
    // dem Eingabe-String hinzufügen:
    inputString += inChar;
    // ist der hereinkommende Buchstabe eine Zeilenschaltung wird ein flag gesetzt,
    //sodass die Hauptroutine Loop mit dem Ergebnis weiter arbeiten kann:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
```

Quelle: arduino.cc - von mir redaktionell in den Kommentaren überarbeitet

Dabei scheint es sich um Standard-Routinen aus der Bibliothek zu handeln.

Dieser Code läuft nur mit der Standard-Seriellen-Schnittstelle. Die übrigen Pins eines Arduino können aber ebenfalls als serielle Schnittstellen deklariert werden, es ist aber wohl nicht möglich, diese Software-Seriellen-Schnittstellen so zu verwenden wie die Pins 0 und 1, also so, dass zwischen zwei loop-Durchläufen automatisch die Serielle Schnittstelle eingelesen wird.

Das ist eine Einschränkung für die Verwendbarkeit. Wenn man mit einem Arduino gleichzeitig, also parallel mehrere andere Arduini kontaktieren will, um von diesen eine Antwort auf eine Frage zu erhalten (z. B. ist irgendwo ein Gleis mit der Nutzlänge von XX frei und wenn ja, wo?) dann kann die Hauptroutine diese wohl nur nacheinander aufrufen und auf Antwort warten, aber nicht wie mit einer interrupt-Steuerung damit umgehen.

Ein solcher Master kann dann keine anderen Aufgaben verarbeiten als die Kommunikation mit den Slaves, weil sonst durch die Laufzeit der anderen Aufgaben eine sichere Kommunikation mit mehreren Slaves wohl nicht gewährleistet werden kann, oder liege ich da falsch

Dann wäre die Alternative eine Gänseblümchenkette, also eine Daisy-Chain, bei der die Arduini alle in einem Ring miteinander verbunden sind. Hier ist ein Beispiel dafür:

https://makezine.com/2012/12/03/how-to-daisy-chain-arduinos-via-serial/

Der Nachteil einer solchen Kette ist, dass sie langsamer arbeitet, denn der erste Arduino sendet eine Nachricht zum zweiten, der sendet nach Verarbeitung seine Nachricht zum nächsten usw. bis der erste die verarbeitete Anfrage vom letzten Arduino in der Kette zurück bekommt.

Der Vorteil ist: Will man z. B. eine Abfrage der freien LängenKapazität mehrerer Gleise starten, die je von einem Arduino verwaltet werden, bekommt man als Antwort alle Gleise mit der gewünschten (Mindest-) Länge zurück und kann dann auswählen, welches Gleis genutzt werden soll und dann den entsprechenden Befehl wieder an alle schicken, aber mit dem token, für welchen Arduino das gelten soll, sodass alle anderen das ignorieren. Das ist eventuell schwierig, was die Länge der Nachrichten, die über die Serielle Schnittstelle kommen, angeht.

Dazu muss jeder Arduino eine Adresse bekommen, von der er weiß und auf die er reagieren kann. Im code der einzelnen Geräte muss mithin alles bis auf die Adresse gleich sein, die als Konstante im header des Programms abgelegt wird.

Bei der Form der Software-Schnittstellen müsste die Hauptroutine nacheinander alle Slaves abfragen, bis eine perfekte Antwort (z. B. exakt die gewünschte Gleislänge) zurückgemeldet wird. Dann könnte man die Anfrage abbrechen und den entsprechenden Arduino gezielt ansprechen (z. B. zum Fahrstraßenschalten). Ähnlich ginge es auch über die Daisy-Chain, wenn die Nachricht nur modifiziert wird, wenn ein Arduino die Frage besser beantworten kann als seine Vorgänger in der Kette und bei der eine perfekt gegebene Antwort nur noch weitertransportiert wird ohne weitere Verarbeitung. Dann entfiele das oben dargestellte Problem der maximalen Anfrage-Antwort-Länge. Ein interessanter Programmier-Auftrag für den anderen thread, in dem es um die Frage der nutzbarren, freien Gleislängen geht (soll hier also nicht weiter vertieft werden).

Die Nutzung der Hardware-Seriellen-Schnittstelle und softwareseitig selbst definierter Schnittstellen sind zwei prinzipiell unterschiedliche Herangehensweisen an die Kommunikation mehrerer Geräte miteinander, die beide ihrer Berechtigung haben, je nach dem, womit bzw. woran man arbeitet. Da z. B. ein Schattenbahnhof sich selten in der Struktur ändert, wäre er über eine Software-Serielle-Schnittstelle ohne weiteres zu steuern, wenn er nicht mehr Gleise hat als Pinpaare zur Verfügung stehen (also 6 Gleise).

Code müsste ich erst noch entwickeln, es sei denn, jemand hat schon was passendes ...

gaulois

#3/5 Verfasst am: 24 März 2019 12:14 Titel:

Hallo zusammen,
nachdem ich die Struktur meiner dezentralen Steuerung für Sassenach in etwa festgelegt habe, beschäftige ich mich seit heute Morgen mit dem Befehls-Protokoll des Masters (Stellwerk Sf) an die verschiedenen Befehlsempfänger (Weichensteller Nord + Süd, Signalsteller Nord + Süd, Rückmelder, Ablaufberg ...)

Ganz schön spannend, ein eigenes Protokoll zu entwickeln.

Festlegungen bisher:

@ beginnt Adresse des angsprochenen Slaves
: beendet Adresse des angesprochenen Slaves
- trennt Start und Ziel (von Einfahrt x nach Gleis y)
? beendet Befehl (jetzt weiß der angsprochene Slave, was zu tun ist)
! quittiert Befehl (damit sagt der Slave dem Master, dass er fertig ist).

Habe ich was vergessen, das man für die Stellbewegungen etc. braucht

gaulois

#4/5 Verfasst am: 24 März 2019 16:41 Titel: Protokollentwurf fertig!

... und das entsprechende debug-Protokoll (das nur angezeigt wird, wenn man auf Fehlersuche ist) zeigt, dass der Weichensteller, den ich exemplarisch vorbereitet habe, unterscheiden kann, was für ihn ist und was nicht:

read address

address set WeichenStellerSued
start EinfahrtSued
dest Gleis3
Befehle gelesen
not ME
nothing to do for WeichenStellerNord
someone else answered

read address
address set WeichenStellerNord
start AusfahrtNord
dest Gleis5
Befehle gelesen
ME
action required
params WeichenStellerNordAusfahrtNordGleis5
WeichenStellerNord operating
params
done!

Natürlich muss das noch mit Leben gefüllt werden. Das ist aber Voraussetzung gewesen dafür, dass der Fahrdienstleiter die im Stellwerk gedrückten Knöpfe in Befehle umsetzt, die dann vom Weichensteller in die Tat umgesetzt werden können.

So kann der Fahrdienstleiter sich permanent mit dem Einlesen von Stellbefehlen und Rückmeldesignalen befassen und muss sich nicht selbst um die zeitintensive Weichenstellung (Servo, langsame Stellbewegung) und Signalstellung kümmern.

In dem Projekt würde ich das mal als Meilenstein bezeichnen, auch wenn es mir rückblickend als eher unspektakulär und einfach zu programmieren erscheint.

Wenn es keine Ergänzungen mehr zum Übertragungsprotokoll aus Sicht des Slaves gibt, also wenn ich keine erforderliche Befehlsübergabe übersehen habe, dann war es das an dieser Stelle.

Die einzelnen Slaves unterscheiden sich ja nur in der Umsetzung der zentralen Befehle.

Die Rückmelder kriegen das gleiche an Befehlen serviert, wobei der Start irrelevant und das Ziel der Rückmeldeabschnitt sein sollen, dessen Status sie zurückmelden sollen.

Bis auf den Part der Auswertung der Befehle und deren Umsetzung ist also das Rahmenprotokoll immer gleich  also wartungsfreundlich

gaulois

#5/5 Verfasst am: 24 März 2019 21:02 Titel:

... so nu is es soweit: Der Weichensteller hat die Syntax für die Weichenstellung in der ersten Version bekommen und antwortet auf die Befehlssequenz

@WeichenStellerNord:AUSFAHRTNORD-9?

mit der Rückmeldung:

read address
address set WeichenStellerNord
start AUSFAHRTNORD
FahrStrasseVon AUSFAHRTNORD
umgewandelt in 4
dest 9
FahrStrasseNach 9
Befehle gelesen
ME
action required
params WeichenStellerNord AUSFAHRTNORD 9

WeichenStellerNord operating
StartNr (Array) 4
nachNorden 6
stelleEinfahrWeichen Weichengruppe 1, Weiche 6
Weichengruppe 1, Weiche 1, Richtung 0
Weichengruppe 1, Weiche 2, Richtung 0
Weichengruppe 1, Weiche 3, Richtung 0
Weichengruppe 1, Weiche 4, Richtung 0
Weichengruppe 1, Weiche 5, Richtung 0
Weichengruppe 1, Weiche 6, Richtung 1
params
done!

Mit dem "!", das als einziges gesendet wird, wenn der debug-modus aus ist, quittiert der Weichensteller den empfangenen Befehl als abgearbeitet

Jetzt müsste man das Ding nur mal einbauen und dem Fahrdienstleiter beibringen, dass er selbst nicht mehr die Weichen stellt, sondern nur noch die Befehle dazu erteilt...