SDI-12 – Ein Protokoll-Dino auf dem Weg ins Ultra-Low-Power IoT

LTE-M, Bluetooth 5 und SDI-12 als idealer Dreiklang

Bild 1: Ein Real-World-IoT Projekt zur Felsüberwachung im Walliser Gebirge (Schweiz) – Das LTE-M-Netz ist bestens, aber der Ort selbst für Bergsteiger nur schwer erreichbar...

Den Begriff „Feldbus“ assoziiert man meist mit etablierten Systemen aus dem industriellen Produktionsbereich, wie z.B. Modbus, CAN oder andere (oft RS485-basierte) Systeme.

Doch speziell fürs Internet der Dinge (IoT) sind aufgrund der Komplexität und des Energieverbrauchs die etablierten Systeme oft nicht die beste Wahl. IoT Geräte werden oft mit Batterien versorgt und im Außenbereich und zur Messung von Umweltparametern eingesetzt. Auch werden meist nur Sensoren gemessen, Aktoren spielen im batterieversorgten IoT häufig nur eine untergeordnete Rolle und werden oft auch nur direkt an das IoT Gerät angeschlossen (z.B. ein Blinklicht für Alarmzustände). Genau in diesem Bereich kann der SDI-12[1] Bus punkten (wobei, als Anmerkung, auch Aktoren per SDI-12 möglich sind).

Möglichst geringer Stromverbrauch und maximal unkomplizierte Wartung, Inbetriebnahme und Austauschbarkeit der Sensoren sind hier die entscheidenden Faktoren zum Aufbau zuverlässiger und wartungsarmer Systeme. Gerade unter Extrem-Bedingungen oder an schwer zugänglichen

Orten lernt man diese Faktoren sehr zu schätzen! Die beiden Fotos stammen von einem IoT Projekt, welches vom Autor mit betreut wird.

Einige typische Beispiele für optimale Einsatzgebiete von SDI-12 sind etwa:

- Wasserpegel-Meßstationen für Oberflächengewässer oder Grundwasser
- Wetterstationen
- Agrar-Sensoren (Bodenfeuchte auf dem Feld)
- Naturgefahr-Überwachung (Hang-Rutschung, Fels-Stabilität, …)
- Überwachung von Schädlingsbefall
- Leckage-Erkennung
- Setzungsmessungen an Brücken
- Füllstände von Containern und Müll
- ...

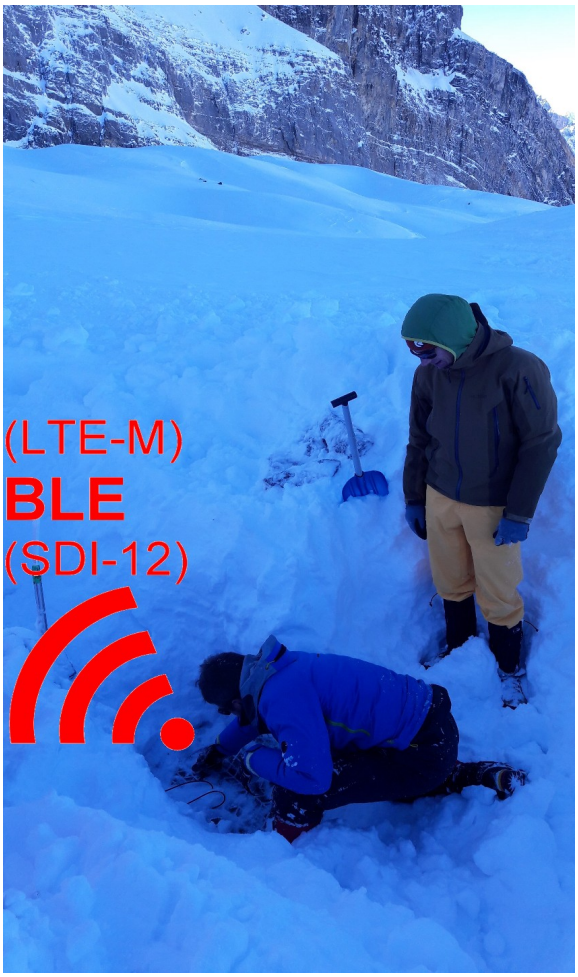

Bild 2: Zum Glück haben die Geräte Bluetooth, sonst wären sie im Schnee wohl nicht zu finden… (bei -20°C auf 3000 mtr.)

SDI-12 wurde bereits 1988 von einer US-Behörde in Kooperation mit einigen Privatfirmen entwickelt und darf frei verwendet werden. Es benötigt lediglich drei Leitungen (Versorgung, Signal, GND) und es können bis zu 62 Sensoren adressiert werden, wobei jeder Sensor mehrere Werte gleichzeitig liefern kann (z.B. Druck und Temperatur). Da die Baudrate recht gering ist (lediglich 1200 Baud) gibt es keinerlei Reflexions- oder Terminierungs-Problematiken, wie etwa bei RS485 basierten Systemen.

Bis zu >500 mtr. Kabel „darf“ in fast beliebiger Topologie verlegt werden. Die Versorgung beträgt traditionell ca. 9.6V – 14V und auf der Signal-Leitung wird mit ca. 0V - 5V Pegeln kommuniziert. Die Kommunikation ist komplett in einem menschlich lesbaren Format, es ist sogar nach den Spezifikationen von SDI-12 explizit gewünscht, dass Menschen direkt mit den Sensoren kommunizieren können! Selbst individuelle, sensorspezifische Kommandos sind in den Spezifikationen vorgesehen (wie beispielsweise das zuvor erwähnte Blinklicht).

Trotzdem gibt es (seit der Version SDI-12 V1.3) auch die Möglichkeit die Daten mit einer CRC16 Prüfsumme fehlergesichert zu übertragen.

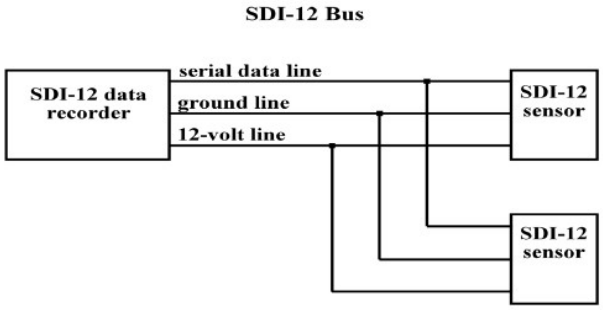

Bild 3: SDI-12 Bus mit 2 Sensoren (Quelle SDI-12.org)

Aber einer der wichtigsten Vorteile von SDI-12 (den es in kaum einem anderen Feldbus gibt) ist es, dass Sensoren nach Bedarf schlafen dürfen! Ein typisches Beispiel hierzu ist etwa ein Regenzähler: er wacht entweder kurz auf, wenn ein Regen-Ereignis detektiert wurde, oder wenn über SDI-12 sein Zählwert abgefragt wird. In den Pausen dazwischen verbraucht er idealerweise „nix“.

Zum Aufwachen kennt SDI-12 ein festes Timing, was maximal 100 msec. dauert. Ganz grob gesagt, wird dazu auf der Signal-Leitung ein '<Break>'-Signal übertragen, gefolgt von der Adresse des betreffenden Sensors ('0' – '9' oder ein Buchstabe) oder der Broadcast-Adresse '?'. Alle „anderen“ Sensoren können sofort wieder schlafen gehen. Danach folgt das Kommando an den Sensor im Text-Format, beendet durch ein '!'-Zeichen. Die Antwort des Sensors ist ebenfalls Text, am Ende mit einem (oder zwei) '<CR><LF>'. Messwerte werden als Standard Zahlenwerte (Fließkomma oder Integer-Zahlen) übertragen.

Einfacher geht es wirklich kaum! Und bei Problemen oder zur Diagnose kann man jederzeit den kompletten Datenverkehr lesen oder auch manuell Kommandos (wie etwa zum Einstellen von Parametern) manuell versenden.

Zur Kommunikation mit SDI-12-Sensoren kann ein einfaches Terminalprogramm und eine RS232-Schnittstelle (wie sie immer noch fast jeder PC hat), in Kombination mit ganz wenigen diskreten Bauteilen verwendet werden. Ein passendes Terminalprogramm („SDI12Term.exe“) befindet sich im Quellcode und compiliert auf der Github-Seite des Autors.

Hier ein paar Beispiele (jedem Kommando ist ein '<BREAK>' Signal zum Wecken vorangestellt):

- '?I!' - Identifiziert einen Sensor (unter Verwendung der Broadcast Adresse '?'). Die Antwort ist beispielsweise: '513STS AG 4900001.51157252<CR><LF>' (Der Sensor hat die Adresse '5' und stammt von STS-AG, der Rest ist unwichtig). Das einzige Problem hierbei kann sein, dass, wenn mehrere Sensoren vorhanden sind, alle antworten, was zu Datenmüll führt…

- '5I!' - Das selbe Kommando, hier antwortet nur der Sensor mit der Adresse '5'.

- '5M!' - Startet eine Messung am Sensor '5'. Antwort wäre z.B.:
  '50012<CR><LF>' (Sensor '5' hat 2 Messwerte in 001 Sekunden)

- '5D0!' - Frägt nach dem Ergebnis. Antwort wäre z.B.:
  '5+0.00180+26.15<CR><LF>' ('5' ist die Adresse, die Werte sind 0.0018 bar und 26.15 °C, die Einheiten ergeben sich aus dem (codierten) Sensortyp von oben ('?I!'))

Natürlich existieren noch mehr Kommandos, aber es müssen im Wesentlichen nur die wirklich benötigten Kommandos auch implementiert werden.

Es gibt inzwischen Hunderte von Herstellen von Sensoren mit SDI-12 Protokoll. Die meisten Sensoren unterstützen maximal SDI-12 V1.3. Die ganz aktuelle Variante V1.4 wird wenig verwendet (sie sind in den Spezifikationen komplett in den Kapiteln 5 und 6 enthalten. Tipp des Autors: einfach diese beiden Kapitel ignorieren).

SDI-12 erlaubt es auch, einen Sensor mit eigenen Kommandos zu versehen! Dazu kann das 'X'-Kommando erweitert werden, beispielsweise für Kalibrier-Parameter oder Ansteuerung von Aktoren. Auch reicht es, einen Sensor mit einer Minialmenge an Kommandos zu versehen. Das macht das Ganze sehr flexibel! Natürlich merkt man den SDI-12 Bus seine gewachsene Struktur an, aber das macht ihn aus Sicht des Autors auch sehr sympathisch!

Individuelle Sensoren sind recht einfach und leicht zu bauen. Daher hat der Autor dazu zwei Open-Source Projekte „Open-SDI12-Blue“ (Hardware) und „SDI12Term“ (Terminal für Windows) veröffentlicht[2] [3] (Links am Schluss).

Das Gateway zum Internet / Datenrecorder

Oder auch einfach nur „Datenlogger“ genannt, sammelt unter Verwendung obiger und weiterer (fehlergesicherter) Kommandos die Daten der Sensoren und transportiert sie (sonst wäre es nicht IoT) gleich direkt auf einen Internet-Server, und das oft jahrelang und mit winzigen Batterien. Im obigen Projekt wurden Datenlogger mit LTE-M/2G für Internet, Bluetooth für die lokale Kommunikation mit mit Smartphone und SDI-12 für die Sensoren eingesetzt.

Diese Geräte sind komplex und empfehlen sich weniger für den Eigenbau und es gibt eine Menge hochwertiger Hersteller [4]. Im Eingangs erwähnten Projekt wurden beispielsweise die sehr robusten Geräte des Typs „Aquatos“ von Terratransfer GmbH eingesetzt:

Bild 4: Der Aquatos wurde speziell für wasserdichten Unterflur-Einbau entwickelt (Quelle: TerraTransfer)

Die 2 Wege ins Ultra-Low-Power IoT

1988 war das Konzept von SDI-12 mit seiner ca. 9.6V – 14V Versorgung und ca. 0V - 5V Signalpegel absolut Stand der Technik. Durch das Konzept des Schlafens ist Low-Power also schon quasi seit jeher Bestandteil von SDI-12.

Aber SDI-12 wurde schon jeher recht entspannt implementiert. Viele Sensoren kommen auch mit anderen Spannungen klar und oft auch mit niedereren Signalpegeln. Einige Hersteller werben sogar bereits mit Versorgungsspannungen (und damit auch Signalpegeln) von 3.6V an aufwärts und Ruheströmen im µA-Bereich!

Und diese Sensoren dürfen getrost als Ultra-Low-Power bezeichnet werden.

Open-SDI12-Blue – ein Open Source Projekt

95% eines SDI-12 Sensors sind fast immer identisch! Daher hat der Autor das Projekt Open-SDI12-Blue veröffentlicht, in dem verschiedene Sensoren auf SDI-12 umgesetzt werden:

- verschiedene Drucksensoren (Piezo, Keramisch, Barometrisch, ...)
- Zähler für Frequenzen und Ereignisse (Regenmengen, Windgeschwindigkeiten)
- präzise Analogwert-Erfassung mit hochauflösendem A/D-Wandler
- Temperatur/Feuchte-Sensoren

Das Projekt ist vollständig dokumentiert (inklusive Schaltplänen und Software) und wird auch in der Praxis so eingesetzt.

Wichtig: Die kommerzielle Verwendung von Bluetooth (inklusive der Bluetooth-Logos) muss zwingend bei der Bluetooth SIG[5] lizensiert werden! Es gibt inzwischen bereits einige kommerzielle, zertifizierte Anwender des Open-SDI12-Blue, die dabei sicher gerne kooperieren. Die kommerzielle Version des Open-SDI12-Blue verfügt zusätzlich über einen benutzerfreundlichen Secure-Firmware-Downloader (per Bluetooth) und Zugriffs-Schutz. Eine aktuelle Liste findet sich in den Dokus zum Projekt.

Eigene Sensoren können problemlos selbst angepasst werden. Es stehen mehrere Portpins zur Verfügung, die sehr flexibel und einfach programmiert werden können, beispielsweise als I2C-Bus, Interrupt-Eingänge oder als einfacher Analogeingang. Als CPU wird eine SoC der nRF52-Reihe verwendet:

Der Vorteil dieser CPU sind bestechend vielfältig:

- kostenlose, erstklassige Entwicklungsumgebung SES für die nRF52 SoCs von SEGGER[6]
- sehr verbreitetes SoC mit exzellenten Bibliotheken
- sehr hohe Rechenleistung, dank ARM Cortex-M4F-Core @ 64 MHz
- das SoC nRF52 enthält Bluetooth 5
- Treiber für Web-Bluetooth (Javascript/HTML) vorhanden

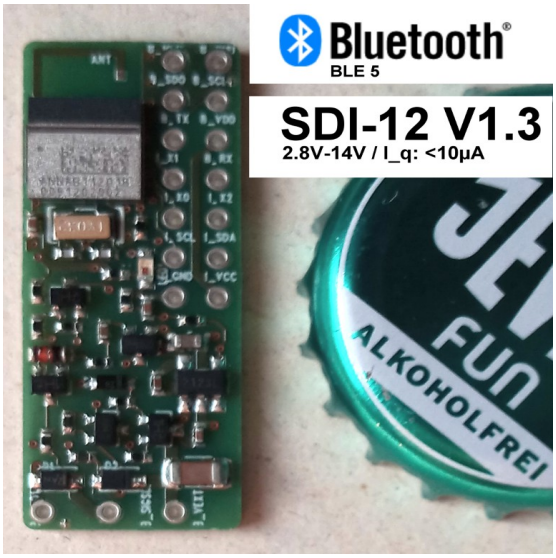

Bild 5: Kern des Open-SDI12-Blue Projekts

Eine der Beispiel-Implementierungen ist etwa Open-SDI12-Blue als barometrischer Drucksensor. Dabei wird der Sensor über I2C-Bus angeschlossen:

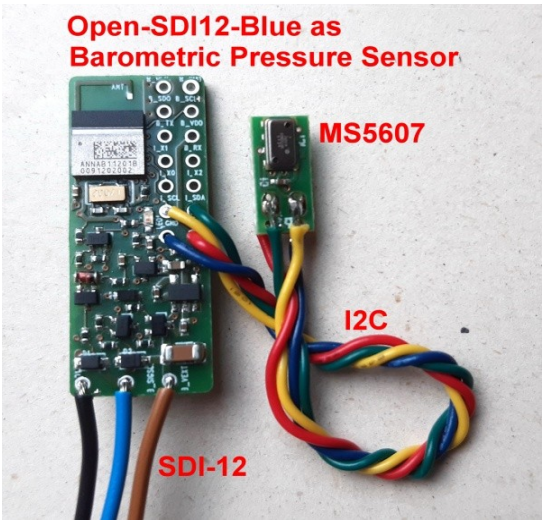

Bild 6: Ein SDI-12 Barometer

Weg 1: Die Kabel-Version

Die Spezifikationen von SDI-12 enthalten auch eine Skizze für einen Signal-Treiber. Allerdings auf 5V-Basis. Der Autor stellt hier ein praxiserprobtes System für einen Treiber vor, der problemlos auch noch mit 3V Versorgung arbeitet, aber auch (recht gut) kompatibel zum 5V Signal-Standard ist. Die einzige Einschränkung ist, dass die 3V Version die Unterscheidung H/L, bzw. 1/0 bei ca. 1.5V macht und nicht, wie in den Original-Spezufukationen bei ca. 3.5V.

Im Schlafmodus benötigt Open-SDI12-Blue <10 µA und die Versorgungsspannung darf 2.8V – 14V betragen. Eine aktive Bluetooth-Verbindung benötigt lediglich ca. 20-30 µA!

Weg 2a: „SDI-12 over Bluetooth“ - HTML/Javascript

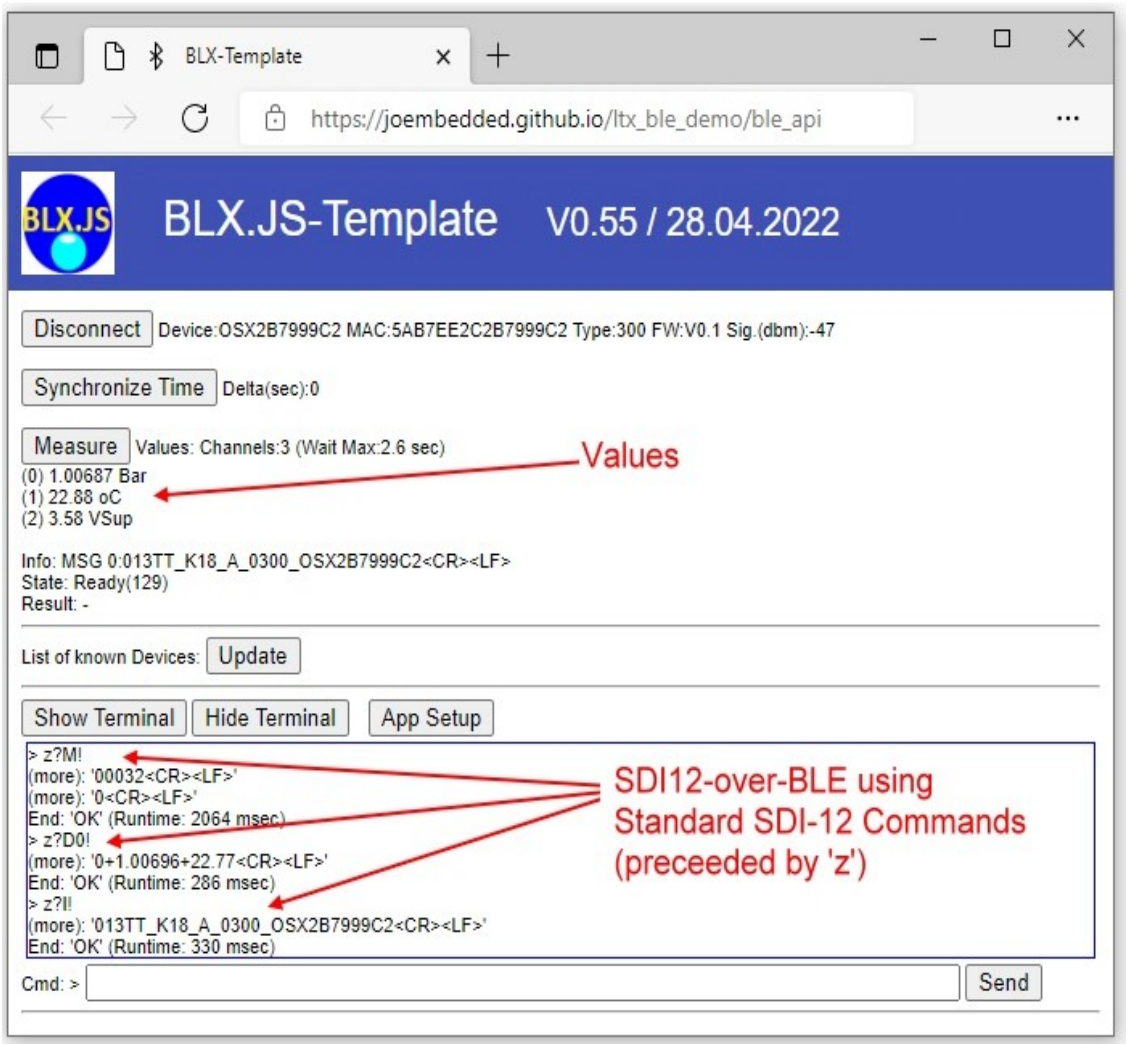

Bild 7: „SDI-12 over Bluetooth“ per Javascript/HTML

Ein noch eleganterer (oder einfach auch nur zusätzlicher) Weg ist es, SDI-12 Kommandos über Bluetooth zu senden! Dazu ist nicht einmal eine APP notwendig, denn inzwischen kann Bluetooth auch über „einfache“ Javascript/HTML Webseiten angesteuert werden. Selbst Firmware kann gegebenfalls per Bluetooth aufgespielt werden. Das Schicken von SDI-12 Kommandos über Bluetooth geschieht exakt im selben Format wie über Kabel! BLX.JS[7] APP ist frei verfügbar (MIT-Lizenz).

So können z.B. die lokalen Einstellungen problemlos über Bluetooth gemacht werden, während ein Datenlogger mit dem selben Sensor gleichzeitig per Kabel kommuniziert! Entsprechende Beispiele und eine API sind in der Doku zum „Open-SDI12-Blue“ enthalten.

Weg 2b: „SDI-12 over Bluetooth“ – Desktop APP für Windows

Als Alternative zur HTML/Javascript API kann ab Windows 10 auf jedem PC die BlueShell als verwendet werden:

Bild 8: Die BlueShell kann u.a. auch zusätzlich mit den Aquatos (TerraTransfer) Datenloggern kommunizieren.

Homepage der BlueShell (kostenlos): https://joembedded.de/x3/blueshell [8]

***

[1] SDI-12 Support Group: https://www.sdi-12.org

[2] Das Open-SDI12-Blue-Projekt: https://github.com/joembedded/Open-SDI12-Blue

[3] Ein einfache SDI-12 Terminal für PC: https://github.com/joembedded/SDI12Term

[4] Eine kleine, unvollständige, Auswahl kommerzieller IoT-Datenlogger mit SDI-12:
https://www.terratransfer.org
http://geoprecision.com
https://www.ecotech.de

[5] Bluetooth SIG: https://www.bluetooth.com

[6] Das für nRF52 kostenlose SES - Segger Embedded Studio: https://www.segger.com
Achtung: Es wird empfohlen eine SES-Version V6.22a oder darunter zu verwenden!

[7] BLX.JS APP: https://github.com/joembedded/ltx_ble_demo

[8] Homepage der BlueShell: https://joembedded.de/x3/blueshell