WS
15/16

TECHNISCHE HOCHSCHULE MITTELHESSEN

Bericht zum Studienprojekt

Konzeption eines Slow Control Systems für Germanium-Detektoren

Verfasser	Cantemir, Raul-Andrei
Matrikelnummer	50 46 030
Studiengang	Maschinenbau Mechatronik Master
Referent (THM Friedberg)	Prof. Dr.-Ing. Sergej Kovalev
Firma	GSI Helmholtzzentrum für Schwerionenforschung in Darmstadt
Firmenbetreuer	Ivan Kojouharov
Eingereicht am	15.10.2015

EIDESSTAATLICHE ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Studienarbeit selbständig und ohne unzulässige fremde Hilfe angefertigt habe. Alle Stellen, die wörtliche oder sinngemäß aus anderen Quellen entnommen wurden, sind als solche kenntlich gemacht.

Datum:_____ Unterschrift_____

VORWORT

Ich möchte mich an dieser Stelle bei all denen bedanken, die mich bei der Anfertigung meiner Studienarbeit unterstützt haben.

Besonderer Dank gilt meinem Firmenbetreuer Ivan Kojouharov, der mir immer mit Rat und Tat zur Seite stand. Außerdem möchte ich mich bei Prof. Dr.-Ing. Sergej Kovalev für seine Betreuung beim Erstellen dieser Arbeit bedanken.

Ein ganz besonderer Dank gilt Herrn Dr. Peter Zumbruch von der GSI, der mir beim Thema der Studienarbeit sehr geholfen hat.

Anschließend möchte ich mich bei meinen Eltern, meiner Schwester und meiner Freundin bedanken, die mich immer und ausnahmslos bis zum Schluss dazu ermutigt haben, weiterzumachen.

Diese Studienarbeit ist meiner Familie gewidmet.

ZUSAMMENFASSUNG

Der vorliegende Bericht entstand im Rahmen des Studienprojekts bei dem GSI Helmholtzzentrum für Schwerionenforschung in Darmstadt. Ziel dieser Arbeit war die Konzeption eines Kontrollsystems für Germanium-Detektoren[1]. Dabei stand EPICS im Vordergrund. Die Softwareumgebung findet bei der Realisierung von verteilten Kontrollsystemen für Großexperimente wie Teilchenbeschleuniger und Teleskope Verwendung [Exp15].

Am GSI Helmholtzzentrum für Schwerionenforschung werden von Forschern aus aller Welt Beschleunigerexperimente durchgeführt. Dazu gehören auch Experimente mit Germanium-Detektoren, bei denen das Messen und Auswerten der Gamma-Strahlung im Vordergrund steht. Um die Stabilität und Sicherheit dieser Detektoren zu gewährleisten, wurde im Rahmen dieser Studienarbeit ein Konzept für die Umsetzung eines Slow Control Systems[2] erarbeitet. Slow Control Systeme werden bei der Überwachung von sicherheitsrelevanten Betriebsparametern, die sich nur sehr langsam ändern, eingesetzt. Werden diese vordefinierten Betriebsparameter überschritten, greift das Kontrollsystem ein und ergreift entsprechende Schutzmaßnahmen. Germanium-Detektoren arbeiten mit sehr hohen Spannungen. Diese Parameter dürfen nur sehr langsam verändert werden, da sie ohne Schutzvorkehrungen unter Umständen die Detektoren zerstören könnten. Das Slow Control System muss daher Anforderungen wie Stabilität, Flexibilität und Zuverlässigkeit erfüllen. Darüber hinaus sollte es über eine Benutzeroberfläche einfach zu bedienen sein. Auch die Möglichkeit der Erweiterbarkeit wurde in Betracht gezogen, um weitere Schnittstellen zur Verfügung zu stellen.

Im Rahmen der vorliegenden Arbeit wurde ein Konzept für die Umsetzung eines Slow Control Systems für Ge-Detektoren erarbeitet. Dabei standen die Identifikation eines geeigneten Computers und die Vorstellung des verwendeten Kontrollsystems im Vordergrund.

[1] Germanium Detektoren (Ge-Detektoren) = Halbleiterdetektoren

[2] Von engl.: slow = langsam und control = überwachen / kontrollieren
(überwacht nicht zeitkritische Vorgänge)

INHALTSVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

A

ASCII.....American Standard Code for Information Interchange

B

BBB.....BeagleBone Black
Bit..... Binary Digit

C

CAN .....Controller Area Network

D

DAC ..... Digital Analog Converter

E

EPICS .....Experimental Physics and Industrial Control System

F

FAIR.....Facility for Antiproton and Ion Research
FPGA .....Field Programmable Gate Array

G

GUI..... Graphical User Interface

H

HDMI.....High Definition Multimedia Interface
HV..... High Voltage

I

I^2C ..... Inter Integrated Circuit
IOC.....Input/Output-Controller

P

PRU .....Programmable Real-time Unit
PV ..... Prozessvariable

S

SCL .....Serial Clock Line
SDA..... Serial Data Line
SPI .....Serial Peripheral Interface

U

UART .....Universal Asynchronous Receiver Transmitter
USB.....Universal Serial Bus

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

1. EINLEITUNG

Dieses Kapitel gibt zunächst einen Überblick über das GSI Helmholtzzentrum für Schwerionenforschung GmbH in Darmstadt und den geplanten Ausbau. Im Anschluss wird die genaue Problemstellung erläutert und auf das Konzeption eines Slow Control Systems eingegangen.

1.1 GSI Helmholtzzentrum für Schwerionenforschung GmbH

Das GSI Helmholtzzentrum für Schwerionenforschung in Darmstadt ist eine Forschungseinrichtung, die 1969 als Gesellschaft für Schwerionenforschung (GSI) gegründet worden ist und eine der weltweit führenden Teilchenbeschleunigeranlagen betreibt. Die GSI beschäftigt etwa 1350 Mitarbeiterinnen und Mitarbeiter. Darüber hinaus hat das Unternehmen über 1000 Gastwissenschaftler aus dem In- und Ausland pro Jahr. Derzeit entsteht das neue internationale Beschleunigerzentrum FAIR. Mit der internationalen Beschleunigeranlage will man mehr über den Aufbau von Materie und von der Entwicklung des Universums erfahren. Das Herzstück ist ein Ringbeschleuniger mit einem Umfang von 1100 Metern. Die bereits existierenden GSI-Beschleuniger dienen dabei als Vorbeschleuniger. Dadurch kann FAIR Ionenstrahlen mit bisher unerreichter Intensität und höheren Energien liefern. Die Bereitstellung von Antiprotonen und Ionen geschieht in bester Qualität [GSI15].

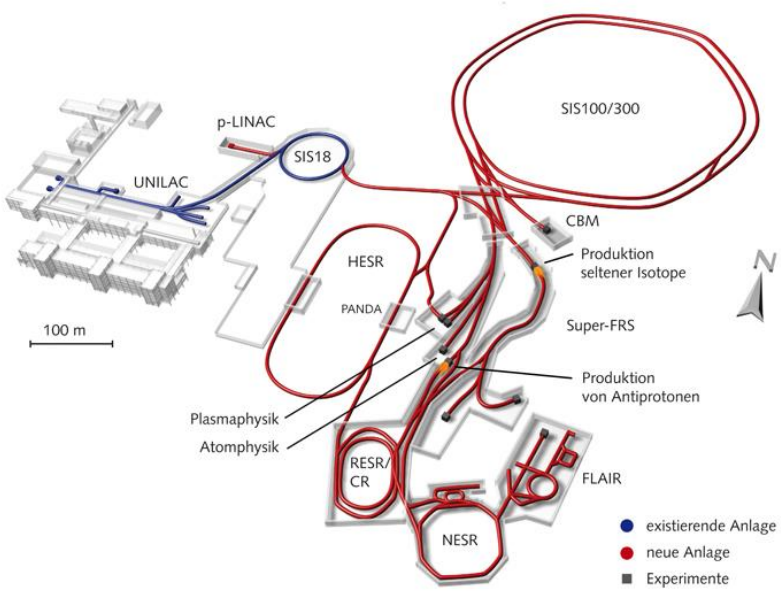

Abbildung 1 - Aufbau von FAIR an der GSI in Darmstadt (Quelle: [GSI15])

1.2 Motivation und Problemstellung

Verteilte Kontrollsysteme, die zur Überwachung von systemkritischen Parametern (Hochspannungen, Temperaturen, Ströme, etc.) eingesetzt werden, sind bei der Durchführung von Großexperimenten wie Teilchenbeschleunigeranlagen nicht mehr wegzudenken. Die Ge-Detektoren in der Abteilung Gammaspektroskopie bei der GSI in Darmstadt werden mit analogen Hochspannung-Modulen (HV-Module) der Firma ISEG [ise15] betrieben. Die Inbetriebnahme der Halbleiterdetektoren ist dadurch sehr aufwendig, da durch das Anlegen der Hochspannung Schäden an den hochwertigen Bauteilen entstehen können. Das Hochdrehen der HV muss daher sehr langsam und unter ständiger Beobachtung der Messsignale geschehen. Es gibt keine Möglichkeit, die verschiedenen Parameter der Detektoren über eine grafische Benutzeroberfläche (GUI = Graphical User Interface) zu steuern. Auch eine Archivierung der Daten zur späteren Rekonstruktion der Betriebsbedingungen ist nicht vorhanden. Die HV-Module für die Ge-Detektoren kommunizieren über den I²C-Bus, womit eine Steuerung über ein netzbasiertes Kontrollsystem realisierbar ist. Mittels eines kompakten Rechners, der direkt an den Ge-Detektoren angebracht wird, können die weniger zeitkritischen Parameter gesetzt und überwacht werden.

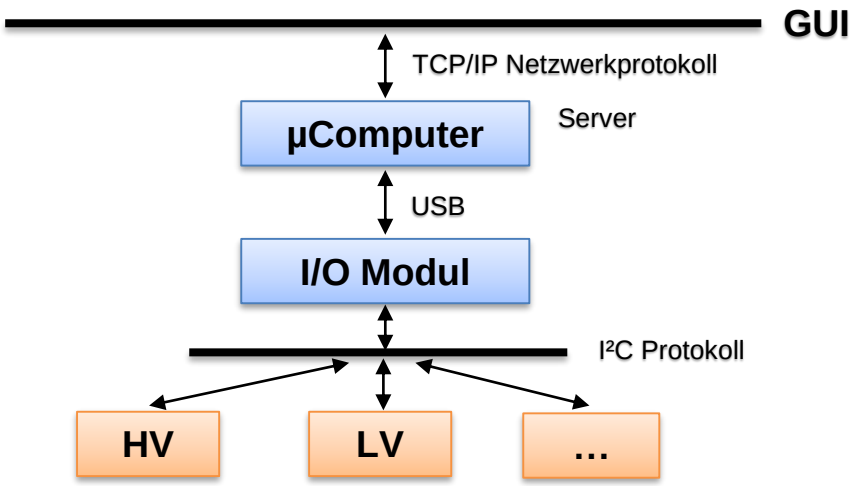

Abbildung 2 - Konzept eines Slow Control Systems

Abbildung 2 veranschaulicht das Konzept eines Slow Control Systems für die Ge-Detektoren. Zur einfachen Bedienung und Überwachung soll eine GUI verwendet werden, die über ein Netzwerk mit dem Server verbunden ist. Der Server ist das Kernstück des Kontrollsystems und verantwortlich für die Kommunikation mit der angeschlossenen Input/Output (I/O) Hardware. Das I/O-Modul HadCon2, das an der GSI in Darmstadt entwickelt wurde, wird mit dem Rechner über USB verbunden. Es stellt bereits eine EPICS-Geräteunterstützung zur Verfügung und lässt sich damit in das Detektor-Kontrollsysteme implementieren [Had15].

2. SLOW CONTROL

Dieses Kapitel befasst sich mit den Aufgaben und Anforderungen eines Slow Control Systems. Anschließend folgt die Identifikation der zu verwendenden Hardware, um einen stabilen und sicheren Ablauf von physikalischen Experimenten mit den Ge-Detektoren gewährleisten zu können.

2.1 Aufgaben und Anforderungen

Ein Slow Control System wird bei experimentellen Aufgaben mit nicht zeitkritischen Parametern eingesetzt. Die zu überwachenden oder veränderlichen Größen können mittels der Softwareumgebung EPICS gesteuert werden. Die Hauptaufgabe des Slow Control Systems wird die Überwachung der Hochspannung sein, die über eine Benutzeroberfläche leicht regelbar sein soll. Das Archivieren dieser Daten sollte für die spätere Auswertung der Betriebsabläufe in regelmäßigen Abständen geschehen. Bei der Ansteuerbarkeit der Hochspannung muss außerdem gewährleistet sein, dass die Software bei einem Ausfall den Fehler meldet und weiterhin stabil läuft.

Die Germanium-Detektoren werden über die HV-Module (Typ-Bezeichnung: ISEG BP 060674p12) mit bis zu 5000V betrieben. Dabei darf die Hochspannung nur langsam und unter ständiger Beobachtung bis zum Endwert erhöht werden. Auch die Anstiegs- und Abstiegszeit sollte nicht zu groß gewählt werden. Deshalb wird eine Aufgabe des Kontrollsystems sein, diese Parameter autonom in kleinen periodischen Schritten und möglichst genau zu kontrollieren. Um sich mit der Kontrollsystemsoftware und der Hardware vertraut zu machen, wurde eine Testumgebung umgesetzt, um das Steuern der Spannung an einem I/O-Modul zu simulieren.

2.2 Identifikation des µComputers

Für ein Kontrollsystem erfüllt ein Einplatinen-Computer[3] alle Anforderungen hinsichtlich Leistung, Zuverlässigkeit, geringe Abmessungen und niedrige Kosten. In der folgenden Tabelle werden die technischen Spezifikationen von vier Einplatinen-Computern, die allesamt auf der ARM-Architektur basieren, miteinander verglichen:

Tabelle 1 - Vergleich von vier Einplatinen-Computern

µComputer	Raspberry Pi 2 Model B [Ras15]	BeagleBone Black [Bea15]	Odroid-U3 [ODR15]	pcDuino 3 [Lin15]
Prozessor	ARM Cortex-A7 (4x 900 MHz)	ARM Cortex-A8 (1GHz)	Cortex-A9 Quad Core (1,7 GHz)	ARM Cortex-A7 Dual Core (1GHz)
Arbeitsspeicher	1 GB	512MB	2 GB	1 GB
Grafikkarte	Dual Core VideoCore IV	Dual Core SGX 530	Mali-400 Quad Core	Mali-400 Quad Core
Betriebssystem (Version)	Raspbian, Arch Linux, OpenELEC	Debian, Android, Ubuntu, Cloud9 IDE	Linux (Xubuntu), Android	Linux (Ubuntu), Android
Mainboard-Steckplätze	GPIO-Port DSI-Display CSI-Kamera	GPIO-Port DSI-Display CSI-Kamera	-	GPIO-Port CSI-Kamera
Speicherkapazität	Extern (microSD)	4 GB 8-bit Onboard Flash (Erweiterung mit microSD)	Extern (microSD)	4 GB Onboard Flash (Erweiterung mit microSD)
Schnittstellen	2x USB 2.0, Cinch (Video), HDMI, LAN (10/100 Mbit/s), Audio, stereo (3.5 mm Klinke)	1x USB 2.0, Cinch (Video), HDMI, LAN (10/100 Mbit/s), Audio, stereo (3.5 mm Klinke)	3x USB 2.0, Cinch (Video), HDMI, LAN (10/100 Mbit/s), Audio, stereo (3.5 mm Klinke)	3x USB 2.0, Cinch (Video), HDMI, LAN (10/100/1000 Mbit/s), Audio, stereo (3.5 mm Klinke)
I/O Gesamt	40	92	(Erweiterungsmodul benötigt)	32
Digital I/O	-	65	-	-
UART	ja	ja	ja	ja
I²C Bus	ja	ja	ja	ja
SPI-Bus	ja	ja	ja	ja
CAN-Bus	nein	ja	-	-
Serielle Anschlüsse	-	5x	-	-
Analoge Ein-/Ausgänge	-	ja	-	ja
Echtzeitfähigkeit	nein	ja (2x PRUs, µController mit 32 Bit)	nein	nein
BxTxH	85x56x17	86x53x20	83x48x20	92x54x25
Zubehör	-	miniUSB	-	-
Anmerkung	-	Ideal für Mess- und Steueraufgaben	-	Ideal für Mess- und Steueraufgaben
Preis	35 EUR	50 EUR	70 EUR	50 EUR

[3] Ein µComputer, oft auch Einplatinen-Computer genannt, ist ein Computersystem, bei dem sämtliche elektronischen Bauteile auf einer Platine angebracht sind.

Wie der Tabelle 1 zu entnehmen ist, deckt der BeagleBone Black von Texas Instruments die wesentlichen Anforderungen eines Kontrollsystems ab. Damit ist die Entwicklungsplatine eine echte Alternative zum beliebten Raspberry Pi 2 Model B. Bei dem BeagleBone Black handelt es sich um eine offene Soft- und Hardwareentwicklungsplattform, die durch eine engagierte Community unterstützt wird.

2.3 Verwendete Hardware

2.3.1 BeagleBone Black

Der kostengünstige Mini-Computer wird von einem 1-GHz-ARM-Cortex-A8-MCU Sitara AM335x von Texas Instruments angetrieben. Ein wesentlicher Vorteil im Vergleich zur Konkurrenz aus Tabelle 1 ist der 4 GB On-Board-Flashspeicher. Dadurch wird enorm viel Platz im Gehäuse des Detektors gewonnen. Der BeagleBone Black verfügt außerdem über die Schnittstellen USB, Ethernet und HDMI. Auch die Anzahl an Erweiterungssteckleisten ist beachtlich. Insgesamt stehen 65 digitale I/O-Anschlüsse und sieben analoge Eingänge zur Verfügung. Die Platine ist außerdem mit zwei PRU-Mikrocontrollern[4] mit 32 Bit ausgestattet. Zu den unterstützten Bussystemen zählen I²C, CAN und SPI. Damit ist der BeagleBone Black in der Lage, mit den HV-Modulen zu kommunizieren [Bea15].

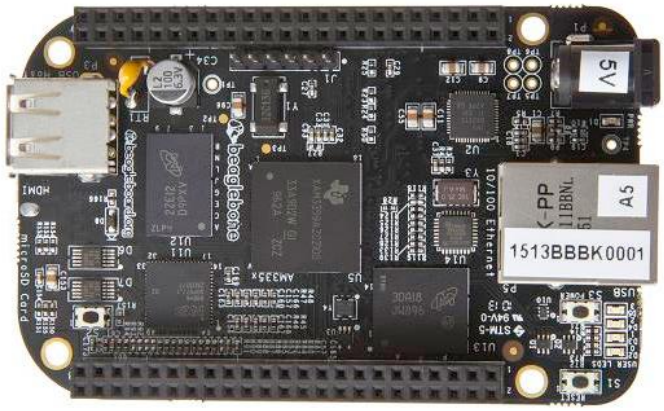

Abbildung 3 - BeagleBone Black (Quelle: [Bea15])

[4] Eine PRU ist ein spezielles Hardware-Element, das sich aus Dual-32-Bit-RISC-Cores zusammensetzt. Es handelt sich hierbei um eine programmierbare Echtzeiteinheit für zeitkritische Systeme.

Der Einplatinen-Computer zeichnet sich insbesondere durch seine zwei PRUs mit 32 Bit aus. Jedoch erfordert der Einsatz des BeagleBone Black als I/O Hardware die Programmierung von Gerätetreibern und Netzwerkprotokollen. Als Lösung bietet sich die Treiberkarte HadCon2 an, die an der GSI in Darmstadt entwickelt wurde. Das I/O Modul wurde speziell für Detektoren und Kontrollsysteme entwickelt.

2.3.2 HADCON2

Das HadCon2 ist ein I/O Modul im Kreditkartenformat, das für Detektoren und Kontrollsysteme konzipiert und umgesetzt wurde. Das Modul basiert auf dem AT90CAN128-Mikrocontroller der Firma ATMEL und verfügt über viele Anschlüssmöglichkeiten. Es bietet I²C Multiplexer[5], 6 Kanal 1-wire-master, 8-channel 8 Bit DAC, CAN-Bus, 8-channel 10 Bit SAR ADC, SPI und 53 programmierbare I/O Anschlüsse. Über den USB-Anschluss kann die Kommunikation mit jeder Art von Rechner (PC, Raspberry Pi, BeagleBone Black, etc.) erfolgen. Die Kommunikation basiert auf ein ASCII-basiertes Protokoll, um eine einfache Implementierung in Kontrollsysteme für Detektoren wie EPICS oder LabVIEW zu gewährleisten [Had15].

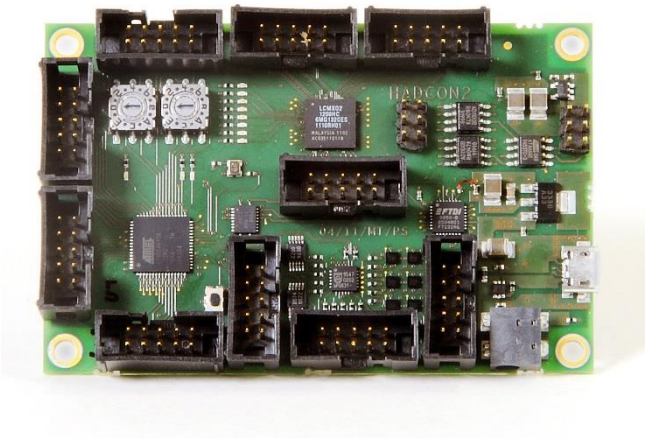

Abbildung 4 - HadCon2 (Quelle: [Had15])

[5] Mit einem I²C-Multiplexer ist es möglich mehrere I²C-Slaves mit gleicher Busadresse am I²C-Bus anzuschließen.

HADCON2 - ÜBERSICHT

- Mikrocontroller: ATMEL AT90CAN128
- I²C-Bus
- CAN-Bus
- SPI
- FPGA: Lattice MachX02-1200-HC
- FTDI USB auf UART-Schnittstelle
- USB 2.0 Anschluss
- USB-Stromversorgung
- 6 x Single-Channel 1-Wire Master
- 1 x 8-Kanal I²C-Bus Multiplexer mit Resetfunktion
- 4-Kanal 8-Bit DAC – Digital-Analog-Wandler
- Reset-Button für ATMEL
- 11 frei programmierbare LEDs

Das HadCon2 verfügt über einen 8-Kanal 8 Bit DAC[6], der einen digitalen Wert in ein analoges Signal wandelt. Auflösung, Genauigkeit und Geschwindigkeit sind wichtige Eigenschaften eines DA-Wandlers. Die Auflösung wird in Bit angegeben und gibt Aufschluss darüber, wie viele Abstufungen der DA-Wandler schafft. Der DAC kann unter anderem genutzt werden, um über die I²C-Schnittstelle digitale Werte in analoge Signale umzuwandeln und an die angeschlossene Hardware zu senden.

Der 8-Kanal 8 Bit DAC hat aber einen großen Nachteilt. Die Hochspannung der Ge-Detektoren darf nur sehr langsam verändert werden. Dabei ist die Auflösung nicht zu vernachlässigen. Für den DA-Wandler des HadCon2 ergeben sich mit 8 Bit $2^8 = 256$ Zustände. Werden die Ge-Detektoren mit einer Spannung von 5000V betrieben, beträgt die Auflösung:

$$\frac{5000V}{256} \approx 20\frac{V}{s}$$

[6] Von engl.: Digital Analog Converter = Digital-Analog-Wandler

Zur Verbesserung der Auflösung kann ein digitales Potentiometer, das mindestens eine Auflösung von 10 Bit aufweist, über den I²C-Bus des HadCon2 verwendet werden. Mit 10 Bit werden bis zu 1024 Werte aufgelöst. Damit ergibt sich bei einer Spannung von 5000V eine viel feinere Abstufung:

$$\frac{5000V}{1024} \approx 5\frac{V}{s}$$

2.4 Das I²C Protokoll

Der I²C-Bus wurde von Philips in den 1980er Jahren für die Datenübertragung zwischen Bausteinen, wie DA-Umsetzer und Mikrocontroller, entwickelt. Seit dem wurde das Protokoll ständig erweitert und findet insbesondere bei der Verbindung von Peripherie-ICs Verwendung. Das Bussystem setzt auf ein Master-Slave-Verfahren und beschränkt sich auf zwei Leitungen. Während eine davon die Taktleitung SCL darstellt, fungiert die andere als Datenleitung SDA. Der Takt wird mit dem Masterbaustein generiert, um die Kommunikation auf dem Bus zu steuern. Der Takt wird nach dem langsamsten Teilnehmer gerichtet [Stu14].

Der I²C-Bus auf der HadCon2-Treiberkarte wird über folgenden Befehl angesprochen:

```
echo "I2C" >/dev/ttyUSBx
```

3. EPICS

In den ersten beiden Kapiteln hat sich der Verfasser mit den wesentlichsten Grundlagen eines Kontrollsystems befasst. Einen wichtigen Teil für den Aufbau einer Slow Control Testumgebung stellt die Kontrollsoftware EPICS dar.

3.1 Was ist EPICS?

Für die Realisierung von verteilten Kontrollsystemen bei Großexperimenten, wie dem Deutschen Elektronen Beschleuniger DESY oder Teleskope, kommt die Softwareumgebung EPICS (Experimental Physics and Industrial Control System) zum Einsatz. EPICS ist als Open Source erhältlich und kann kostenfrei mit freier Lizenz verwendet werden. Abbildung 5 beschreibt die EPICS Architektur, die auch als netzwerkbasiertes Client/Server-Modell bezeichnet wird.

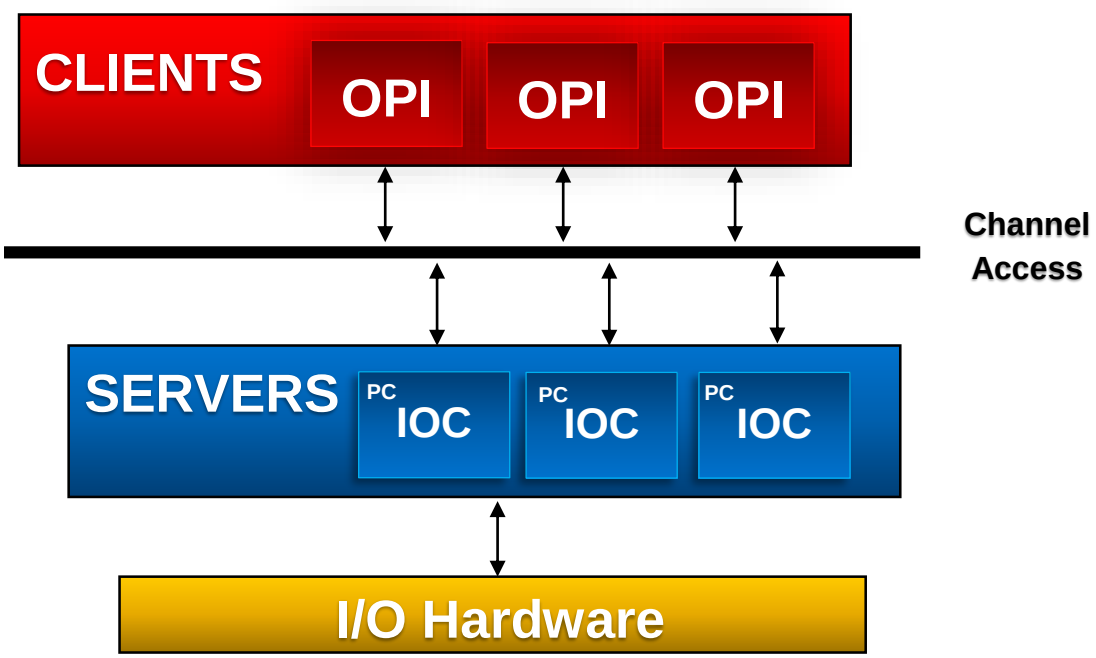

Abbildung 5 - Das Diagramm beschreibt die EPICS-Architektur. Die I/O Hardware tauscht Informationen mit den Servern aus, die über das Channel Access (CA) den Clients zur Verfügung gestellt werden [Joh15].

Für die Kommunikation zwischen zwei oder mehreren Computern werden sogenannte Client-Server-Methoden verwendet. Der IOC (Input/Output-Controller), der sich auf einem oder mehreren Servern befindet und für die Kommunikation mit der angeschlossenen Hardware (z.B. HadCon2 oder HV-Module) verantwortlich ist, zeichnet dabei die Daten in Echtzeit auf und speichert diese. Die verbundenen Clients erhalten anschließend über das TCP/IP-basierte Channel Access (CA) Netzwerkprotokoll alle gesammelten Informationen. Den verbundenen Clients werden die von der Hardware gesendeten Informationen über eine sogenannte Prozessvariable (PV) zur Verfügung gestellt. Es spielt keine Rolle, auf welchem Server diese Informationen abgelegt werden, da die Identifikation dieser Information einzig und allein über die PV geschieht. Eine Installationsanleitung für EPICS steht im Anhang A.1 [Exp15].

3.2 Records und Felder

Ein Record ist eine Zusammenfassung von Prozessvariablen (PV). Eine Prozessvariable ist die Kombination aus einem Record-Namen und einem Feld-Namen mit einem Punkt als Trennzeichen. Wird kein Feld-Name angegeben, benutzt Channel Access als Standard `.Val`, das den Wert eines Records beinhaltet.

`Beaglebone:G:readData.EGU`

Die Funktion bestimmt dabei den Typ dieses Records, wobei auch die Kommunikationsrichtung eine nicht triviale Rolle spielt. Erwartet das Record Informationen von der Hardware, so ist dieser vom Typ `ai` (Analog Input). Im Gegensatz dazu senden Records vom Typ `ao` (Analog Output) Daten an die Hardware oder kommunizieren mit anderen Records. Im Folgenden seien einige Typen aufgezeigt [Phi15]:

Tabelle 2 - Record-Typen

Record Typ	Richtung	Kurzbeschreibung
ai	Input	Bezieht einen analogen Wert
ao	Output	Schreibt eine Variable aus der IOC Datenbank in den DAC der Hardware oder in ein anderes db Record
bi	Input	Erwartet/Verarbeitet einen binären Wert
bo	Output	Schreibt einen binären Wert auf die Hardware oder einem anderen db Record
calc	Input	Führt algebraische, relationale und logische Operationen durch. Das Ergebnis der Operation kann von einem anderen Datensatz abgerufen werden.
calcout	Input	Es bietet neben der Funktionalität eines Calc-Record die Möglichkeit, Output-Links und Output-Events hinzuzufügen, die von der Art des Ergebnisses der Operation abhängig sind.
longout	Output	Speichert Integer-Werte von bis zu 32 Bit und schreibt diese auf die angeschlossene Hardware. Über die Datenbank können mit dem *Soft Channel* Hardware-Support Werte in andere Datensätze geschrieben werden.

Records besitzen die wichtige Eigenschaft, untereinander Daten auszutauschen, Rechenoperationen auszuführen und auf Signale von der Hardware (Interrupts) zu warten. Diese Aktionen werden aber nur dann ausgeführt, wenn die entsprechenden Records prozessiert werden. Für das Prozessieren eines Records gibt es drei verschiedene Möglichkeiten. Der SCAN-Modus bietet ein erweiterbares Auswahlmenü, in dem das Scanintervall festgelegt werden kann. Standardmäßig ist dieses Feld auf `PASSIVE` gesetzt. Das Prozessieren eines Records ist dann entweder über andere Records oder aber durch den Schreibzugriff über einen CA-Client möglich. Für den Fall, dass ein Record einmalig beim Starten der IOC (siehe Kapitel 3.4) prozessiert werden soll, hält EPICS das PINI-Feld bereit. Die Initialisierung eines Records mit einem bestimmten Wert erfolgt dann vor den normalen SCAN-Aufgaben. Als dritte Variante steht das Auslösen eines I/O Interrupts zur Verfügung.

Tabelle 3 - Typische Felder von Records

Feld-Typ	Beschreibung
INP/OUT	Input/Output-Link
DTYP	Gerätetyp
VAL	Technische Größe
EGU	String für eine Technische Größe
RVAL	Rohwert (16 Bit)
PINI	Initialisierung beim Start der IOC
LOPR	unterer Betriebsbereich
HOPR	oberer Betriebsbereich

Um ein Objekt (Record) erstellen zu können, wird mittels eines Texteditors eine Datei mit der Endung `*.db` erstellt. Diese Datei enthält neben der Record-Bezeichnung auch den Typ und den Namen. Dieser Name darf nicht in mehreren Records gleichzeitig vorkommen und sollte sinngemäß gewählt werden. Innerhalb der geschweiften Klammern können Felder mit Attributen definiert werden.

Folgendes Beispiel zeigt die Struktur eines einfachen Records:

```
record(bo, "beaglebone:HVSwitch"){
  field(DESC, "HV On/Off switch")
  field(ONAM, "On")
  field(ZNAM, "Off")
  field(OUT, "beaglebone:write")
}
```

Hierbei handelt es sich um ein Binary Output Record, das die zwei Werte `0` und `1` annehmen kann. Nach dem Start der EPICS IOC enthält die Datenbank die Prozessvariable `beaglebone:HVSwitch`.

3.3 Visual DCT

Eine Alternative zu der Erstellung von Records mit einem Texteditor ist die grafische Methode mittels VDCT (Visual Database Configuration Tools). Das in Java geschriebene Visual DCT erleichtert die Arbeit im Umgang mit Records und bietet eine bessere Übersicht. Insbesondere wenn es um die Verknüpfung mehrerer Records geht, erhöht Visual DCT die Übersicht enorm [Cos15]. Ein Record kann nämlich auch Links zu anderen Records enthalten, was bei umfangreichen Projekten schnell unübersichtlich werden kann.

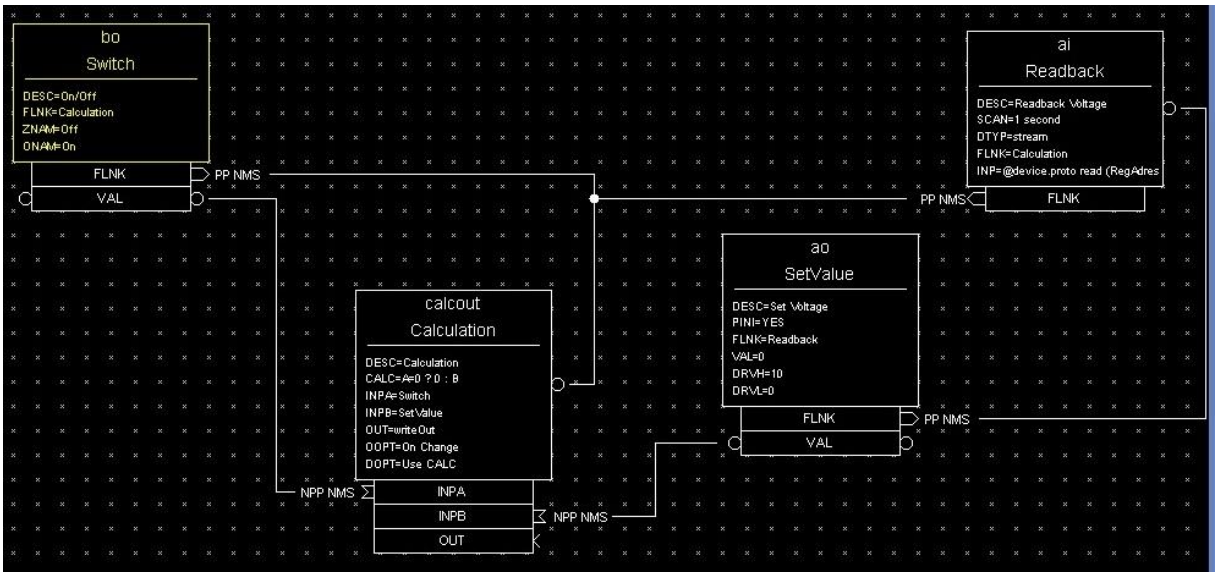

Abbildung 6 - Simulation einer Energieversorgung in VDCT. Die einzelnen Records werden in Kapitel 4.4 beschrieben.

VDCT setzt die Installation der Java Runtime Environment voraus. Anschließend können die `*.db` oder `*.dbd` Dateien, die zu bearbeiten sind, geöffnet werden. Visual DCT Java Archive Package (`*.jar`) ist eine ausführbare Datei, die mit folgendem Befehl vom Kommandofenster aus gestartet werden kann:

```
java -jar VisualDCT.jar [<DBD>* or <DB>*]
```

Als Parameter werden optional sowohl Database Definition Files (`*.dbd`) als auch Database Files (Record-Instanzen) akzeptiert.

3.4 IOC-Server

Als IOC (Input/Output-Controller) bezeichnet man den Teil eines Kontrollsystems, der als Server fungiert, gleichzeitig aber auch ein CA-Client ist. Die an den Server angeschlossene Hardware kann über das EPICS-Kontrollsystem angesteuert werden. Der Datenaustausch zwischen Servern und Clients erfolgt über das EPICS-Netzwerkprotokoll Channel Access (CA). Der Zugriff der Clients auf die Hardware geschieht immer über die IOC. Für die Kommunikation mit den angeschlossenen Geräten werden gerätespezifische Protokolle verwendet. Somit werden keine weiteren Kenntnisse über die Hardware benötigt. Über den Channel Access Security kann auch der Zugriff auf Prozessvariablen eingeschränkt werden. Zu den IOC-Komponenten zählen neben der Datenbank auch der Sequenzer, der bei der Realisierung von Zustandsmaschinen verwendet werden kann. Ein Beispiel zum Erstellen einer IOC Applikation ist unter 4.1 aufgeführt [EPI151].

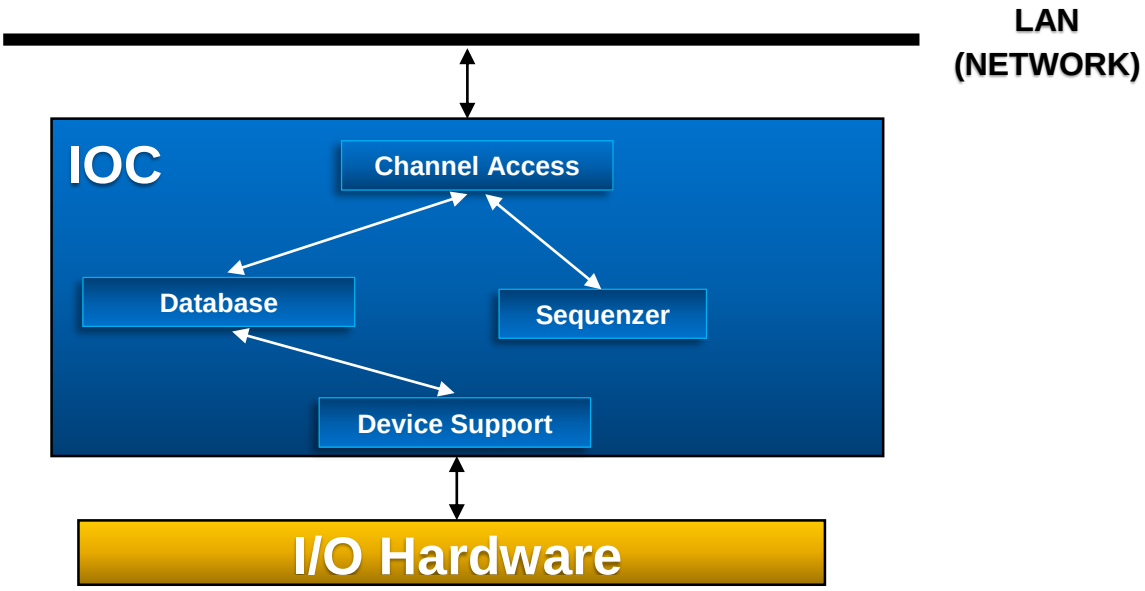

Abbildung 7 - Komponenten eines IOC (Quelle: [EPI151])

Über das Netzwerkkommunikationsprotokoll Channel Access aus Abbildung 7 kann sich der Client direkt mit einer sogenannten Prozessvariable (PV) verbinden. Dabei stehen mit `put` (schreibe), `get` (lese) und `monitor` (überwache) drei grundlegende CA-Kommandos zur Verfügung. Vorausgesetzt die PV ist auf einem im Netzwerk befindlichen CA-Server verfügbar und besitzt einen eindeutigen bzw. einmaligen Namen im Netzwerk. Ist die PV nicht verfügbar, wird die Anfrage durch eine Zeitüberschreitung (Timeout) beendet.

3.5 Alarm Handling

Dem Datensatz eines Records können verschiedene Attribute zugeordnet werden. Das Feld Alarm Severity stellt dabei die Wichtigkeit eines Fehlerzustandes dar. Folgende vier Werte können angenommen werden [Jan04]:

`NO_ALARM, MINOR, MAJOR, INVALID`

Darüber hinaus gibt es ein weiteres Attribut, das den Alarmstatus des Datenbank-Records spezifiziert. Folgende Werte können gesetzt werden:

`HIHI, HIGH, LOW, LOLO, ...`

Diese Zustände können unter anderem mithilfe einer grafischen Benutzeroberfläche überwacht werden. Bei der Verwendung einer GUI sollten die Alarmzustände eines Records immer im Programmcode enthalten sein und nicht erst bei der Erstellung einer neuen Benutzeroberfläche festgelegt werden. Dadurch gelten für alle Clients die gleichen Alarmzustände. `NO_ALARM` ist der Normalzustand eines Records, während `MINOR` bereits die erste Sicherheitsschwelle kennzeichnet. Der obere und untere Schwellwert eines `ao` oder `ai` Records wird in den Feldern `HIGH` und `LOW` festgelegt. Die letzte Sicherheitsschwelle ist `MAJOR` und wird über die Felder `HIHI` und `LOLO` festgelegt. Funktioniert die angeschlossene Hardware nicht korrekt, so wird `INVALID` als Alarmzustand gesetzt.

3.6 EPICS Datenbank

Als EPICS Datenbank wird die Sammlung von mehreren Records bezeichnet, die verschiedene Typen besitzen können. Diese Datei wird beim Hochfahren in den IOC-Speicher geladen und erlaubt die Kommunikation mit dem Channel Access Protokoll. Bei der Definition einer Datenbank sind einige grundlegende Dinge zu beachten. In den Database Definition Files (`*.dbd`) werden Record-Typen, Optionen für den Gerätesupport, Auswahlmenüs und andere Konfigurationen definiert. Während des IOC-Bootprozesses werden eine oder mehrere dbd-Dateien geladen, um die Konfigurationen zu setzen. Des Weiteren gibt es die Database Files (`*.db`), die Beschreibungen von Record-Instanzen enthalten. Auch diese Dateien werden während dem Bootvorgang des IOC geladen. Die Database Definition Files enthalten demnach Definitionen von Record-Typen, die in den Database Files verwendet werden.

4. GRUNDGERÜST

In diesem Kapitel wird das HadCon2 mit dem BeagleBone Black verbunden und anhand einfacher IOC Applikationen eine Testumgebung erprobt, um die Funktionsweise von EPICS zu analysieren. Hierfür werden die Leuchtdioden auf dem HadCon2 I/O Modul angesteuert.

4.1 Erste IOC Applikation

Im Folgenden wird die Funktionsweise von EPICS mithilfe einer einfachen IOC Applikation analysiert [EPI15 S. 13]. Dabei stellt *makeBaseApp.pl*, ein EPICS-eigenes PERL-Script, die Basis für die Entwicklung bereit.

Erstellen von <MyApp>:

```
mkdir <IOC_App>
cd <IOC_App>
<base>/bin/linux-arm/makeBaseApp.pl -t example <MyApp>
<base>/bin/linux-arm/makeBaseApp.pl -i -t example <MyApp>
make
cd iocBoot/ioc<MyApp>
```

Starten der EPICS IOC:

```
../../bin/<Target>/<MyApp> st.cmd
```

Es wird eine IOC Applikation <MyApp> erstellt. Mit der Konfigurationsdatei st.cmd wird die IOC Applikation <MyApp> gestartet. Bei erfolgreicher Initialisierung gelangt der Anwender in die EPICS-IOC-Shell[7], die IOC-Kommandos ausführt.

[7] Eine Shell (auch Terminal oder Konsole) verbindet den Benutzer mit dem Computer.

Nach erfolgreicher Initialisierung erscheint die EPICS-IOC-Shell:

```
< envPaths
epicsEnvSet("ARCH","linux-arm")
epicsEnvSet("IOC","iocApp1")
epicsEnvSet("TOP","/home/debian/MyFirstIOC")
epicsEnvSet("EPICS_BASE","/usr/local/epics/base")
cd "/home/debian/MyFirstIOC"
## Register all support components
dbLoadDatabase "dbd/App1.dbd"
App1_registerRecordDeviceDriver pdbbase
## Load record instances
dbLoadTemplate "db/userHost.substitutions"
dbLoadRecords "db/dbSubExample.db", "user=rootHost"
## Set this to see messages from mySub
#var mySubDebug 1
## Run this to trace the stages of iocInit
#traceIocInit
cd "/home/debian/MyFirstIOC/iocBoot/iocApp1"
iocInit
Starting iocInit
############################################################################
##########
## EPICS R3.14.12.5 $Date: Tue 2015-03-24 09:57:35 -0500$
## EPICS Base built Jun 29 2015
############################################################################
##########
iocRun: All initialization complete
## Start any sequence programs
#seq sncExample, "user=rootHost"
epics>
```

Mit dem Befehl `dbl` werden alle Records aufgelistet. In einem zweiten Kommandofenster können diese Records über einen Channel-Access-Befehl (CA-Befehl) gelesen, gesetzt oder aufgelistet werden:

```
camonitor HOSTNAME:aiExample1
```

Über den CA-Befehl `monitor` wird periodisch das Beispiel-Record `aiExample1`, das zwischen den Werten 0 und 9 einen veränderlichen Wert simuliert, ausgelesen. Der CA-Befehl `get` bietet die selbe Funktionalität, wobei der Wert dann nur einmalig abgefragt wird.

Um den Wert eines Records setzen zu können, wird `put` gefolgt von dem neuen Wert verwendet. Der Wert wird ohne Einheit und unter Berücksichtigung des richtigen Wertetyps eingegeben. Anschließend wird der Wert einmalig zurückgelesen und angezeigt:

```
caput HOSTNAME:VoltSet 5
HOSTNAME:VoltSet 5.0 V
```

4.2 BeagleBone Black, HadCon2 und EPICS

Das HadCon2, das bereits eine EPICS-Geräteunterstützung bereitstellt, wird über die USB-Schnittstelle mit dem BeagleBone Black verbunden.

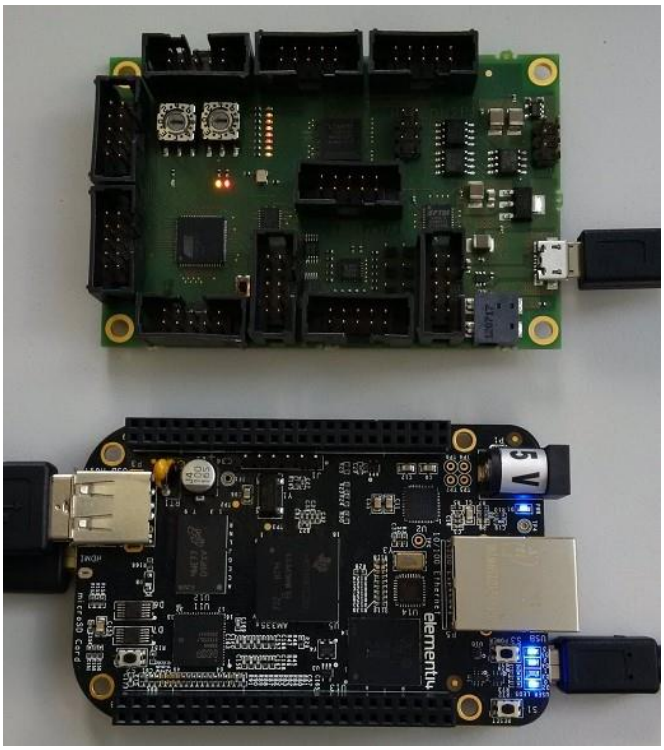

Abbildung 8 - Verbindung von BeagleBone Black und HadCon2

Um das HadCon2 mit dem BeagleBone Black betreiben zu können, wurde eine fertige IOC Applikation verwendet, die bereits Funktionalitäten zur Ansteuerungen der LEDs bereitstellt [Pet15]. Die komprimierte Datei wird im EPICS-Hauptverzeichnis entpackt und die Pfade in den Dateien `RELEASE` und `RELEASE.linux-arm` im Verzeichnis `/config` angepasst. Letztere Datei kann entweder mit dem Inhalt der `RELEASE`-Datei überschrieben werden oder gelöscht werden. Mit dem Befehl `make` von einer Kommando-Shell aus werden die korrekten Pfade in die Datei `/../iocBoot/iocStreamHadcon_Register_IO/envPaths` gesetzt. In einem letzten Schritt muss gewährleistet sein, dass die Datei `~/.bashrc` eine Definition für `HOSTNAME` enthält:

```
export HOSTNAME=$(hostname)
```

Mit dem Befehl `make` im Hauptverzeichnis `HadCon2_AtmelRegisterIO_IOC` ist das Projekt einsatzbereit.

4.3 LED-Applikation zur HadCon2-Ansteuerung

Die Kommunikation zwischen BeagleBone Black, HadCon2 und der Softwareumgebung EPICS soll anhand einer einfachen LED-Ansteuerung demonstriert werden. Diese Applikation basiert auf der HadCon2 General IOControls Dokumentation [GSI151]. Für das Ansteuern der LEDs müssen Register gelesen und beschrieben werden. Hierfür gibt es zwei Kommandos:

`RGRE <register>` (Register auslesen)

und

`RGWR <register> <value>` (Register beschreiben)

Der schematischen Darstellung des HadCon2 [Had15] ist außerdem folgende Pinbelegung zu entnehmen:

Tabelle 4 - HadCon2 LED-Pinbelegung

	Pin	Farbe
AT90CAN128	PG0 (WR)	Rot
	PG1 (RD)	Gelb
	PG2 (ALE)	Orange

Die LEDs folgen dabei einer inversen Logik: `0:Einschalten` und `1:Ausschalten`. Die Register des Mikrocontrollers AT90CAN128 für die LEDs liegen an PORTG.

Die Adresse hierfür findet sich in der Dokumentation des Mikrocontrollers [AT915 S. 408]. Diese lautet `0x34`. Die Registerwerte werden gesetzt, indem auf PORTG geschrieben wird. Das Schreiben einer 1 auf PIN toggelt[8] den Wert von PORTG (AT90CAN128, Kapitel 9.2.2).

Darüber hinaus stellt die APFEL_Hadcon_IOC-Protokolldatei `hadcon_global.proto` [Pet15] eine Debug-Möglichkeit bereit, um einen String[9] zum HadCon2 zu senden und eine Zeile seiner Antwort zu empfangen.

[8] Toggeln nennt man das Hin- und Herschalten zwischen den beiden Zuständen HIGH und LOW

[9] Von engl.: String = Zeichenkette nennt man in der Informatik aneinandergereihte Zeichen.

Die EPICS IOC wird über eine Kommando Shell aus gestartet.

```
cd <HadCon2_base>/iocBoot/iocStreamHadcon_Registers_IO$
../../bin/linux-arm/streamHadcon st.cmd
```

Von einer zweiten Shell aus, der die EPICS Kommandos `caget`, `camonitor` und `caput` bekannt sein müssen, wird das Stringout-Record `<hostname>:1:streamDEBUG` ausgeführt:

```
caput <hostname>:1:streamDEBUG "RGWR 34 0x03"
```

Mit dem Befehl `caput` wird ein Wert in eine Prozessvariable gesetzt. Da dies über einen sogenannten String geschieht, müssen die Anführungszeichen gesetzt sein. Die beiden LEDs `PG0` und `PG1` werden mit `0x03` (`0b00000011`) ausgeschaltet. Soll auch `PG2` ausgeschaltet werden, wird `0x07 (0x00000111)` als Parameter eingegeben.

Die Record-Struktur dazu ist wie folgt aufgebaut:

```
record(longout,
"$(PREFIX)$(HADCON):$(NAME):SetValue_$(RegName)$(SUFFIX)") {
field(DESC, "write $(RegName)@$(HADCON)")
field(DTYP, "Soft Channel")
field(OUT,
"$(PREFIX)$(HADCON):$(NAME):send_$(RegName)$(SUFFIX)_.VAL PP MS")
field(UDF, "0")
field(DRVH, "7")
field(DRVL, "0")
field(IVOA, "Don't drive outputs")
}
```

Es wird ein Record vom Typ `longout` verwendet, der die Soft Channel Device Support Routine verwendet, um die Parameter schreiben zu können. Das OUT-Feld gibt an, an welches Record die Ausgabe gesendet werden soll. Da es sich hierbei um ein Record handelt, das Werte auf ein Gerät schreibt, muss die Adresse des HadCon2 spezifiziert sein. Auch der Name des korrespondierenden Geräteunterstützung-Moduls muss spezifiziert sein. Das Out-Feld verwendet PP (Passive Process), um andere Records, deren SCAN-Feld auf `PASSIVE` gesetzt ist, zu prozessieren. Es werden Werte zwischen 0 und 7 ausgegeben. Eine ungültige Eingabe führt dazu, dass der Wert durch die vordefinierte Aktion `Don’t drive outputs` nicht gesetzt wird.

4.4 Power Supply Simulation

Diese Simulation einer Energieversorgung (Power Supply Simulation) dient als Grundlage für die Umsetzung eines Slow Control Prototypen. Das Ein- und Ausschalten der HV-Module übernimmt ein Record vom Typ Binary Output:

```
  # Set Voltage  ON/OFF
  record(bo,
  "$(PREFIX)$(HADCON):$(NAME):Switch_$(RegName)$(SUFFIX)") {
    field(DESC, "On/Off")
    field(ZNAM, "Off")
    field(ONAM, "On")
    field(FLNK,
  "$(PREFIX)$(HADCON):$(NAME):CalcV_$(RegName)$(SUFFIX)")
  }
```

Das Record `Switch` ist standardmäßig auf `NULL` gesetzt. Ein Record vom Typ `calcout` prüft, ob der Wert dieses Records `1` oder `0` ist:

```
# Calc Value
record(calcout,
"$(PREFIX)$(HADCON):$(NAME):CalcV_$(RegName)$(SUFFIX)") {
  field(DESC, "Write Voltag Out")
  field(CALC, "A=0 ? 0 : B")
  field(INPA,
"$(PREFIX)$(HADCON):$(NAME):Switch_$(RegName)$(SUFFIX)")
  field(INPB, "$(PREFIX)$(HADCON):$(NAME):SetV_$(RegName)$(SUFFIX)")
  field(OOPT, "On Change")
  field(DOPT, "Use CALC")
  field(HIHI, "10")
  field(HIGH, "7")
  field(LOLO, "0")
  field(LOW, "3")
  field(HHSV, "MAJOR")
  field(HSV, "MINOR")
  field(LLSV, "MAJOR")
  field(LSV, "MINOR")
  field(OUT,
"$(PREFIX)$(HADCON):$(NAME):write_$(RegName)$(SUFFIX).VAL PP")
}
```

Entspricht das Ergebnis der Rechenoperation `1` (`ON`), wird es an das Record `write` gesendet, das den Wert von `SetVolt` über ein Geräteprotokoll auf die I/O Hardware schreibt. Das eigentliche Setzen der Werte sowie das Auslesen der I/O Hardware übernehmen zwei Records vom Typ Analog Output und Analog Input.

Das Record `SetV` wird durch das PINI-Feld beim Start der EPICS IOC einmalig prozessiert. Über das Link-Attribut wird gleichzeitig auch das Record `SetVRB` vor allen anderen Records prozessiert, um parallel zur Initialisierung mit dem Wert 0 den aktuellen Wert der I/O Hardware zurückgeliefert zu bekommen. Ein SCAN-Feld sorgt dafür, dass dieser Werte jede Sekunde ausgelesen wird. Über ein Link-Feld wird auch das Calc-Record periodisch prozessiert:

```
# Set Value
record(ao, "$(PREFIX)$(HADCON):$(NAME):SetV_$(RegName)$(SUFFIX)") {
  field(DESC, "User Input Voltage")
  field(PINI, "YES")
  field(EGU, "V")
  field(HOPR, "10")
  field(LOPR, "0")
  field(DRVH, "10")
  field(DRVL, "0")
  field(VAL, "0")
  field(FLNK,
"$(PREFIX)$(HADCON):$(NAME):SetVRB_$(RegName)$(SUFFIX)")
}

# Readback Value
record(ai, "$(PREFIX)$(HADCON):$(NAME):SetVRB_$(RegName)$(SUFFIX)"){
  field(DTYP, "stream")
  field(DESC, "Volt Setpoint Readback")
  field(INP, "@hadcon_RGRE.proto read($(RegAddress)) $(device)")
  field(SCAN, "1 second")
  field(FLNK,
"$(PREFIX)$(HADCON):$(NAME):CalcV_$(RegName)$(SUFFIX).VAL")
}
```

Mithilfe der grafischen Benutzeroberfläche CSS [Cos15], die sich auf einem Client befindet, können die PVs auf dem Server verwendet werden.

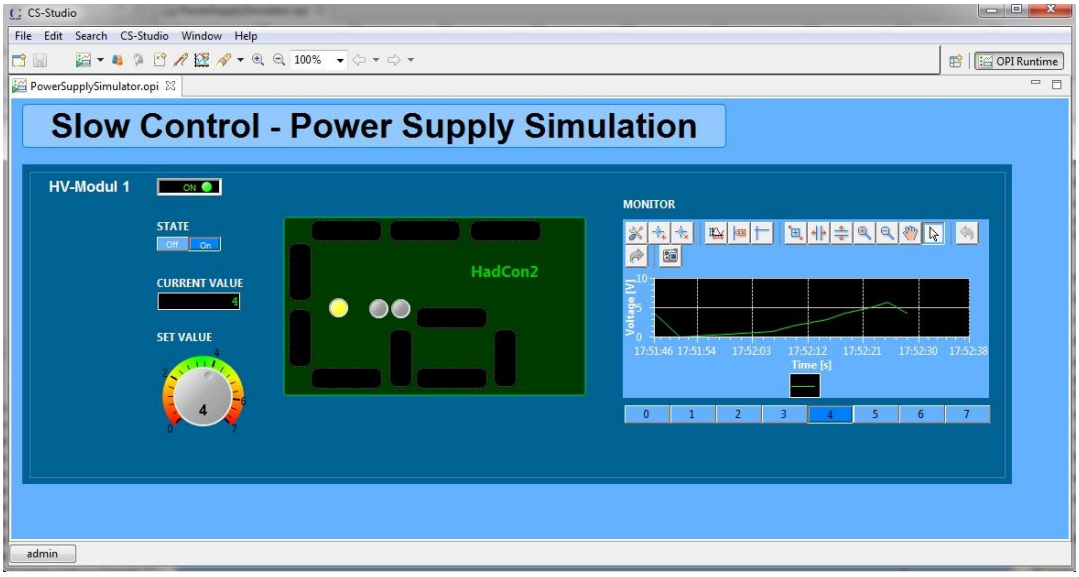

Abbildung 9 - Slow Control Power Supply Simulation

Das Simulationsmodel berücksichtigt jedoch nicht Empfindlichkeit der Detektoren hinsichtlich einer sprungförmigen Erhöhung der Hochspannung. Um die langsame Erhöhung der HV zu simulieren, müsste der vom Benutzer gesetzte Wert an einem Record vom Typ `calcout` übergeben werden, das den aktuellen Wert mit dem neuen Benutzerwert vergleicht und anschließend periodisch inkrementiert bzw. dekrementiert. Das Ergebnis dieser Operation könnte dann in eine Variable gespeichert und auf die Hardware geschrieben werden.

4.5 Periodische Records

Für die Umsetzung von periodischen Records, mit denen beispielsweise das langsame Erhöhen einer Hochspannung simuliert werden kann, bietet EPICS mehrere Möglichkeiten an. Das folgende Calcout-Record zählt nach dem Start der IOC von 0 aufwärts hoch, bis der Endwert 7 erreicht worden ist. In dem Beispiel findet das Inkrementieren periodisch alle 10 Sekunden statt [Phi15 S. Kapitel 10]:

```
record(calcout,
"$(PREFIX)$(HADCON):$(NAME):LEDCount_$(RegName)$(SUFFIX)") {
field(DESC, "LED-Counter")
field(SCAN,"10 second")
field(CALC, "C>0?((D>=A&&D<B)?D+C:A):((D>A&&D<=B)?D+C:B)")
field(INPA, "0x00")
field(INPB, "7")
field(INPC, "1")
field(INPD,
"$(PREFIX)$(HADCON):$(NAME):LEDCount_$(RegName)$(SUFFIX).VAL")
field(EGU, "Counts")
field(OUT, "$(PREFIX)$(HADCON):$(NAME):write_$(RegName)$(SUFFIX)
PP")
field(UDF, "0")
}
```

Es können bis zu 12 INP-Felder (INPA, INPB, … INPL) für das Lesen von Parametern bei Calcout-Records verwendet werden. Dieser Feldtyp kann sowohl Konstanten als auch Werte von anderen Records enthalten. Die Rechenoperation selbst findet im Calc-Feld statt. Solange der Anfangswert von Feld A nicht erreicht ist und der aktuelle Wert von D kleiner dem Endwert ist, wird der aktuelle Werte von Feld D bei jedem Durchgang um `1` inkrementiert. Trifft diese Bedingung nicht zu, so wird nach Erreichen des Endwertes der aktuelle Wert von D wieder auf `NULL` gesetzt.

5. FAZIT UND AUSBLICK

5.1 Fazit

Im Rahmen dieser Studienarbeit wurde ein Konzept für die Entwicklung eines Slow Control System Prototypen für Ge-Detektoren erarbeitet. Der BeagleBone Black stellte sich als zuverlässiger und kompakter Einplatinencomputer heraus, der sich auch für die Umsetzung von echtzeitfähigen Systemen eignet und über wichtige Schnittstellen verfügt. Wesentlicher Nachteil ist dabei allerdings, dass für die Entwicklungsumgebung eine neue EPICS-Geräteunterstützung sowie ein I²C-Netzwerkprotokoll entwickelt werden müssen. Das I/O Modul HadCon2, dass speziell für Detektoren und Kontrollsysteme entwickelt worden ist, verfügt über diese Gerätetreiber. Die Testumgebung für den Slow Control Prototypen wurde mit EPICS als Kontrollsoftware und dem HadCon2 als I/O Hardware realisiert. Mit der Ansteuerung der vorhandenen HadCon2-LEDs wurde eine Steuerung für HV-Module simuliert. Die Steuerung erfolgte über die grafische Benutzeroberfläche CSS.

Es hat sich herausgestellt, dass die Umsetzung eines Slow Control Prototypen in der Konstellation BeagleBone Black, HadCon2 und EPICS realisierbar ist.

5.2 Ausblick

EPICS ist eine leistungsfähige und sichere Softwareumgebung zur Realisierung von Kontrollsystemen für Großexperimente. Das Studienprojekt diente als Vorbereitung für die Entwicklung eines Slow Control Prototypen für Ge-Detektoren. Die Testumgebung für die Ansteuerung der LEDs kann erweitert werden, indem ein I²C-Netzwerkprotokoll für EPICS entwickelt wird. Über den I²C-Bus können die HV-Module direkt angesteuert werden oder aber mit einem digitalen I²C-Potentiometer (Auflösung: 10 Bit), der eine viel feinere Abstufung der Hochspannung erlaubt. Derzeit muss noch geklärt werden, ob die bestehenden Anschlüsse des HadCon2 ausreichen, da eine Liste aller Signale noch nicht vorliegt. Deshalb muss nach der Entwicklung eines Prototyps darüber nachgedacht werden, ob das fertige Slow Control System direkt mit dem BeagleBone Black realisiert wird. Dafür müssen ein Gerätetreiber und ein I²C-Netzwerkprotokoll entwickelt werden.

LITERATURVERZEICHNIS

[GSI15] GSI Helmholtzzentrum für Schwerionenforschung. *GSI-Webseite* [Online]. 11. August 2015. http://www.gsi.de.

[BCD15] BCDA synApps. *BCDA synApps-Webseite* [Online]. 14. Juli 2015. http://www.aps.anl.gov/bcda/synApps/.

[Cos15] Cosylab, Control System Laboratory. *Cosylab-Webseite* [Online]. [Zitat vom: 01. September 2015.] http://www.cosylab.com/solutions/particle_accelerators/VisualDCT/.

[Ras15] Raspberry Pi. *Raspberry Pi-Website* [Online]. [Zitat vom: 20. Juli 2015.] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[ODR15] ODROID. *ODROID-Website* [Online]. [Zitat vom: 20. Juli 2015.] Odroid-U3.

[Lin15] LinkSprite Technologies, Inc. *LinkSprite-Website* [Online]. [Zitat vom: 20. Juli 2015.] http://www.linksprite.com/.

[Stu14] Sturm, Matthias. *Mikrocontrollertechnik: Am Beispiel der MSP430-Familie.* München : Carl Hanser Verlag, 2014. ISBN 978-3-446-42231-5.

[AT915] AT90CAN128 Documentation. *Atmel Corporation-Webseite* [Online]. [Zitat vom: 20. August 2015.] http://www.atmel.com/Images/doc7679.pdf.

[Jan04] Janet Anderson. Alarm Handler User's Guide. *Alarm Handler User's Guide-Webseite* [Online]. 2004. [Zitat vom: 21. September 2015.] http://www.aps.anl.gov/epics/EpicsDocumentation/ExtensionsManuals/AlarmHandler/ALHUserGuide/ALHUserGuide.html.

[EPI15] EPICS Application Developer’s Guide. *EPICS-Webseite* [Online]. 10. August 2015. http://www.aps.anl.gov/epics/base/R3-14/12-docs/AppDevGuide.pdf.

[EPI151] EPICS Input/Output Controller Application Developer's Guide. *EPICS-Webseite* [Online]. [Zitat vom: 10. August 2015.] http://www.aps.anl.gov/epics/base/R3-14/8-docs/AppDevGuide.pdf.

[Exp15] Experimental Physics and Industrial Control System. *EPICS-Webseite* [Online]. [Zitat vom: 01. August 2015.] http://www.aps.anl.gov/.

[GSI151] GSI Wiki. *GSI Wiki-Webseite* [Online]. [Zitat vom: 18. 08 2015.] https://wiki.gsi.de/foswiki/bin/view/Epics/HadCon2GeneralIOControls.

[Ins15] Installing EPICS on Raspberry Pi. *Installing EPICS on Raspberry Pi-Webseite* [Online]. [Zitat vom: 22. Juli 2015.] http://prjemian.github.io/epicspi/.

[ise15] iseg Germany. *iseg Germany-Webseite* [Online]. [Zitat vom: 05. Juli 2015.] http://www.iseg-hv.com.

[Joh15] Johnson, Andrew. Channel Access Concepts. *EPICS-Webseite* [Online]. [Zitat vom: 01. September 2015.] http://www.aps.anl.gov/epics/docs/APS2014/05-CA-Concepts.pdf.

[Phi15] Philip Stanley, Janet Anderson, Marty Kraimer. Record Reference Manual. *EPICS Record Reference Manual-Webseite* [Online]. 10. August 2015. http://www.aps.anl.gov/epics/EpicsDocumentation/AppDevManuals/RecordRef/Recordref-1.html.

[Had15] Zumbruch, Peter. HadCon2. *GSI Wiki-Webseite* [Online]. 13. Juli 2015. https://wiki.gsi.de/EE/HadCon2.

[Pet15] —. Zumbruch/HadCon2_AtmelRegisterIO_IOC. *Zumbruch/HadCon2_AtmelRegisterIO_IOC-Webseite* [Online]. 10. August 2015. https://github.com/zumbruch/HadCon2_AtmelRegisterIO_IOC.

[Exp] Experimental Physics and Industrial Control System. *EPICS-Webseite* [Online].

[Bea15] BeagleBoard.org. *BeagleBoard.org-Webseite* [Online]. 13. Juni 2015. http://beagleboard.org/black.

ANHANG

A.1 EPICS Installation

Der Einplatinencomputer BeagleBone Black wird in der Revision C mit dem Betriebssystem Debian 7 (Wheezy) ausgeliefert. Die typische Umsetzung eines Kontrollsystems mit EPICS basiert auf dem Client-Server-Modell, bei dem die Komponenten Server und Client getrennt. Für diese Studienarbeit fungierte der BeagleBone Black sowohl als Server als auch als Client. Erst bei der Verwendung der grafischen Benutzeroberfläche CSS Boy wurden beide Komponenten getrennt.

Für die Installation von EPICS wird das Verzeichnis `/Apps/epics` erstellt und eine Verknüpfung von einem lokalen Ort `/usr/local/epics` zu diesem Verzeichnis erzeugt. Für die Installation wurde das Archiv baseR3.14.12.5.tar.gz, das auf der EPICS-Webseite heruntergeladen werden kann, ausgewählt und im EPICS-Hauptverzeichnis gespeichert. Es gibt zwar eine neuere Version, jedoch wurde für das Projekt die letzte verbesserte Version R3.14.12.5 verwendet. Das Archiv wird im Ordner `/Apps/epics` entpackt. Die Installation basiert auf [EPI15] und [Ins15]:

```
cd ~
mkdir –p ~/Apps/epics
sudo su
cd /usr/local
ln –s /home/debian/Apps/epics
exit
cd ~/Apps/epics
wget
http://www.aps.anl.gov/epics/download/base/baseR3.14.12.5.tar.gz
tar –xzf baseR3.14.12.5.tar.gz
ln –s ./base-3.14.12.5 ./base
```

EPICS ist für verschiedene Computer und Betriebssysteme verfügbar. Das BeagleBone Black System ist ein `linux-arm`, das beim Setzen der Umgebungsvariablen berücksichtigt werden muss. Folgende Zeile setzt die Umgebungsvariable:

```
export EPICS_HOST_ARCH=`/usr/local/epics/base/startup/EpicsHostArch`
```

Anschließend kann der Prozess ausgeführt werden, indem EPICS erstellt und ausgeführt wird. Die Installation setzt jedoch voraus, dass die Tools GNU make, Perl und Unzip bereits installiert sind.

```
cd ~home/debian/Apps/epics/base
make
```

Die Installation von EPICS kann wie folgt überprüft werden:

```
bin/linux-arm/softIoc
iocInit
```

Dass die Initialisierung erfolgreich war, zeigt folgende Zeile:

```
Starting iocInit
############################################################################
##########
## EPICS R3.14.12.5 $Date: Sat 2015-08-08 13:25:50 -0500$
## EPICS Base built Jan 19 2013
############################################################################
##########
iocRun: All initialization complete
epics>
```

Um die Verwendung der EPICS-Tools zu vereinfachen, werden in der Datei `~/.bashrc` folgende Deklarationen gemacht:

```
export EPICS_ROOT=/usr/local/epics
export EPICS_BASE=${EPICS_ROOT}/base
export EPICS_HOST_ARCH=`${EPICS_BASE}/startup/EpicsHostArch`
export EPICS_BASE_LIB=${EPICS_BASE}/lib/${EPICS_HOST_ARCH}
export EPICS_BASE_BIN=${EPICS_BASE}/bin/${EPICS_HOST_ARCH}

if [ "" = "${LD_LIBRARY_PATH}" ]; then
export LD_LIBRARY_PATH=${EPICS_BASE_LIB}
else
export LD_LIBRARY_PATH=${EPICS_BASE_LIB}:${LD_LIBRARY_PATH}
fi
export PATH=${PATH}:${EPICS_BASE_BIN}
```

A.2 synApps Installation

Das große Packet synApps erleichtert die Installation von einzelnen Modulen und enthält darüber hinaus StreamDevice[10], das zum Ansteuern des HadCon2 benötigt wird.

Das synApps-Packet in der Version 5.7 kann unter [BCD15] heruntergeladen werden und in das EPICS Hauptverzeichnis entpackt werden. Um die Lauffähigkeit von synApps auf dem BeagleBone Black zu gewährleisten, müssen folgende Änderungen in der Datei `configure/RELEASE` vorgenommen werden:

```
SUPPORT=/usr/local/epics/synApps_5_7/support
EPICS_BASE=/usr/local/epics/base
```

Die Module, die nicht installiert werden sollen, können auskommentiert werden. Hierzu zählen ALLEN_BRADLEY, DAC128V, LOVE, IP, DELAYGEN, CAMC, VME und AREA_DETECTOR.

Um die Änderung auf alle Module wirksam zu machen, werden folgende Zeilen ausgeführt:

```
cd ~/Apps/epics/synApps_5_7/support
make release
```

Auch die Datei `Makefile` muss editiert werden, indem der Support für nicht benötigte Module komplett entfernt wird.

[10]StreamDevice ist eine Unterstützung für Geräte, um den Datentransfer zwischen einem EPICS I/O Record und einem verbundenen Gerät zu erleichtern.

A.3 XXX-Modul Konfiguration

Das XXX-Modul ist Bestandteil von EPICS IOC und kann verschiedene synApps-Module gleichzeitig konfigurieren. Deshalb kommentiert man in der Datei `/support/xxx-5-7-1/configure/RELEASE` die Zeilen

```
#AREA_DETECTOR=$(SUPPORT)/areaDetector-1-8beta1
#IP=$(SUPPORT)/ip-2-13
```

sowie in der Datei `xxxCommonInclude.dbd` unter `support/xxx-5-7-1/xxxApp/src/` die Zeile

```
#include "ipSupport.dbd"
```

aus.

Danach werden in der Datei `support/xxx5-7-1/xxxApp/src/Makefile` alle Zeilen, die eine Referenz zu `areaDetector` und dessen Komponenten enthalten, auskommentiert:

```
ADSupport.dbd, NDFileNetCDF.dbd, ADBase, NDPlugin netCDF,
simDetectorSupport.dbd, commonDriverSupport.dbd, ip
```

Bevor die Komponenten von synApps generiert werden können, müssen die Erweiterungen `Msi`, `Extensions` und `re2c` heruntergeladen und installiert werden.

```
cd ~/Apps/epics
wget
http://www.aps.anl.gov/epics/download/extensions/extensionsTop_201
20904.tar.gz
tar xzf extensionsTop_20120904.tar.gz

wget http://www.aps.anl.gov/epics/download/extensions/msi1-
5.tar.gz
cd extensions/src
tar xzf ../../msi1-5.tar.gz
cd msi1-5
make
```

Anschließend werden folgende Deklarationen in der Datei `~/.bashrc` eingetragen:

```
export EPICS_EXT=${EPICS_ROOT}/extensions
export EPICS_EXT_BIN=${EPICS_EXT}/bin/${EPICS_HOST_ARCH}
export EPICS_EXT_LIB=${EPICS_EXT}/lib/${EPICS_HOST_ARCH}
if [ "" = "${LD_LIBRARY_PATH}" ]; then
    export LD_LIBRARY_PATH=${EPICS_EXT_LIB}
else
    export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${EPICS_BASE_LIB}
fi
export PATH=${PATH}:${EPICS_EXT_BIN}
```

Voraussetzung für die Installation des EPICS Sequenzer, der Bestandteil eines Kontrollsystems sein kann, ist das Packet re2c:

```
sudo apt-get install re2c
```

Die Basisinstallation wird abgeschlossen, indem folgende Zeilen ausgeführt werden:

```
cd ~/Apps/epics/synApps_5_7/support
make release
make rebuild
```