MpBus-BACnet Gateway für den Einsatz in der Reinraumtechnik

Mathias Marquardt, HS Wismar, FG CEA/ATM
mathias.marquardt@hs-wismar.de
Christian Bock, HS Wismar, FG CEA/ATM
christian.bock@hs-wismar.de
Olaf Hagendorf, HS Wismar, FG CEA/ATM
Olaf Simanski, HS Wismar, FG CEA/ATM

Zusammenfassung

Moderne Reinraumkomplexe werden mit mehreren miteinander kommunizierenden Teilsystemen zur Druck- und Volumenstromregelung betrieben. Um Skalierbarkeit und Konfiguration neuer Installationen zu vereinfachen und eine zentrale Druckregelung der Reinräume einzurichten, wird auf das BACnet/IP Protokoll gesetzt. Für die Einbindung der für diesen Zweck verwendeten Ventilklappen in die zentrale Regelung wird ein Gateway benötigt, welches die Schnittstelle zwischen dem MpBus Protokoll der Klappe und BACnet/IP bereitstellt. Als Basis des Gateways wurde ein STM32F7 Mikrocontroller verwendet, sowie die nötige Firmware für die BACnet/IP und die MpBus Protokolle entwickelt bzw. portiert. Zusätzlich ist im Rahmen dieses Projektes eine Simulink Blockbibliothek für die Nutzung von BACnet/IP entwickelt worden.

1 Einleitung

Die Integration mehrerer Drosselklappen einer bestehenden Regeleinrichtung zur zentralen Regelung von Volumenströmen einer Reinraumumgebung erfordert es, zwischen der Regeleinrichtung und einem übergeordneten Bussystem zu vermitteln. Die dazu nötigen Bussysteme sind auf Seiten der Regeleinrichtung der von Belimo entwickelte MpBus [5, 6], sowie regelungsseitig, das Ethernet basierte BACnet/IP Protokoll [3].

Für diesen Zweck wird ein entsprechendes Gateway im Rahmen eines FuE Projektes entwickelt. Aufgabe des Gateways ist es, wie Abbildung 1 zu entnehmen, zwischen beiden Systemen zu vermitteln und den derzeitigen Zustand der Regeleinrichtung BACnet/IP-seitig abzubilden, wie auch eine Regelinformation von BACnet/IP Seite an den MpBus weiterzuleiten.

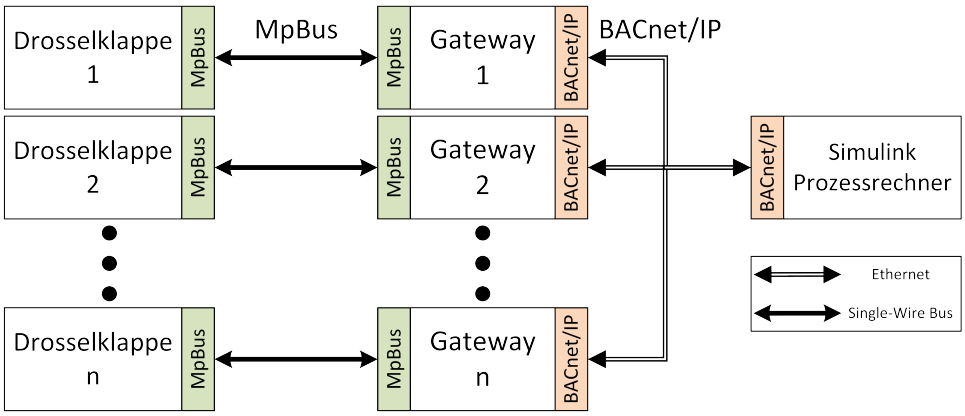


Abbildung 1: Schematischer Aufbau der zentralen Regelung

2 Grundlagen

2.1 BACnet

Das BACnet (Building Automation and Control Networks) Protokoll [3] ist ein Netzwerkprotokoll für den Einsatz in Gebäudeautomationen. Das Konzept von BACnet soll die Interoperabilität von Geräten verschiedener Hersteller gewährleisten. Dazu sind sogenannte BIBBs (BACnet Interoperability Building Block) definiert, welche Services und Prozeduren auf Server- sowie Clientseite beschreiben. Dafür gibt es zu jedem BACnet fähigen Gerät ein PICS (Protocol Implementation Conformance Statement) genanntes Dokument, dass alle unterstützten BIBBs, Objekttypen, Zeichensätze und Optionen auflistet.

Die BACnet Norm unterscheidet verschiedene Dienste (Services) zum Beispiel für gemeinsame Datennutzung, Alarm- und Ereignisverarbeitung und die Verarbeitung von Wertänderungen, sowie unterschiedliche Objekttypen, wie analoge und binäre Ein- und Ausgaben, Kalender und Zeitpläne. Alle Datenpunkte eines BACnet Gerätes werden mit Hilfe dieser Objekte im Netzwerk abgebildet. Als komplexe Datentypen bieten die einzelnen Objekte unterschiedliche Eigenschaften (Properties), von denen einige zwingend erforderlich sind und andere optional. Die Properties die jedes Objekt besitzen muss sind die Objekt ID und der Objekttyp, die zusammen einen 32 Bit Wert ergeben, mit dem das Objekt auf einem Gerät eindeutig identifiziert wird. Des Weiteren muss jedes BACnet Gerät ein Device Objekt besitzen, welches Informationen über das Gerät enthält sowie die Geräte ID mit der ein Gerät im Netzwerk identifiziert wird. Der Zugriff eines Clients auf die Eigenschaften von Objekten erfolgt über die Service Requests `Read Property` und `Write Property`.

Für die Sicherungs- und Übertragungsschicht bietet BACnet unterschiedliche Alternativen, wie PTP über RS-232, MS/TP über RS-485, ARCNET, Ethernet, BACnet/IP, LonTalk. Für dieses Projekt wird einzig BACnet/IP genutzt, welches eine komplette BACnet Nachrichten im Datenteil eines UDP Paketes über ein Ethernet Netzwerk versendet.

2.2 Belimo MpBus

Tabelle 1: MpBus Layer

ISO/OSI Layer	Beschreibung
Layer 1 (physisch)	18V LSB first UART, 8n1, 1200 Baud
Layer 2 (Protokoll)	Nachrichtenorientierte Master-Slave Kommunikation
Layer 7 (Anwendung)	Anwendungsebene

Die durch das Gateway anzusprechende Ventilklappe, verwendet als Kommunikationsmedium einen herstellerspezifischen Industriebus, den Belimo MpBus. Der MpBus ist ein Eindrahtbussystem, welches entweder in einer Punkt-zu-Punkt (PP) oder Multipoint (MP) Verbindung genutzt werden kann. In beiden Fällen setzt der MpBus dabei auf eine Halb-Duplex Übertragung in einem RS-232 Frameformat mit 0V Lowpegel und 18V Highpegel.

Die Datenübertragung wird durch den MpBus paketorientiert als Master-Slave Kommunikation realisiert. Dementsprechend agiert das Gateway als MpBus-Master und spricht die MpBus befähigte Ventilklappe als Slave an. Da je Drosselklappe ein Gateway vorgesehen ist, nutzt das Gateway lediglich den PP Anteil des MpBus-Protokollspezifikation.

Der Master startet die Kommunikation mit einem Command Frame. Dieses beantwortet der Slave innerhalb eines durch das Protokoll vorgeschriebenen Zeitrahmens mit einem Answer Frame. Sowohl Command wie auch Answer setzen dabei auf eine identische Framestruktur für den Aufbau einer Busnachricht. Beide Frames bestehen aus bis zu 10 Byte, eingeleitet durch ein Startbyte (STB), welches den Empfänger, den Kommunikationstyp (PP/MP) der Nachricht und deren Länge enkodiert. Dem Startbyte folgt sowohl für Command wie auch Answer ein Datenanteil von maximal 7 Bytes. Im Falle des Commands, siehe Abbildung 2, beschreibt das erste Datenbyte den durch den Master ausgesandten Command.

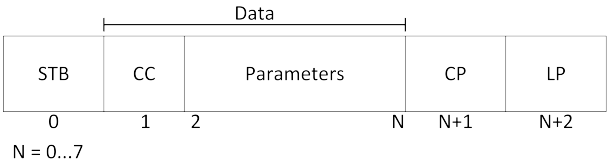


Abbildung 2: MpBus Command

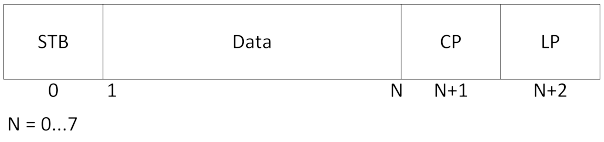


Abbildung 3: MpBus Answer

Je nach Command folgen optionale Bytes mit möglichen Parametern. Der Datenanteil einer Answer, siehe Abbildung 1, ist abhängig vom auslösenden Command, und liefert dem Command entsprechend Parameterinformationen. Sollten durch den Command keine Parameter angefordert worden sein, so ist der Datenanteil des Answerframes leer, und es stellt ein einfaches Acknowledgeframe (ACK) dar. Den Abschluss eines jeden MpBusframes bilden zwei Paritybytes (CP / LP), gebildet aus einer Kreuzparität (CP) und einer Längsparität (LP).

3 Gateway

3.1 Hardware

Die Hardware des Gateway Prototyps ist in Sandwichbauweise bestehend aus drei Platinen aufgebaut.

Den Kern bildet eine NUCLEO-F746ZG Nucleo-144 Entwicklungsplatine der Firma STMicroelectronics. Dieser Cortex-M7 Mikrocontroller besitzt eine Ethernet MAC (Media Access Controller) Einheit, welche die Verwendung des Ethernet basierten BACnet/IP Protokolls ohne externen Ethernet Controller ermöglicht. Die weiteren Komponenten sind eine Basisplatine für die MpBus Ankopplung und eine Shield-Platine mit einem Ethernet PHY bzw. PHY-Switch, der die Anbindung an den physischen Layer übernimmt. Die Verwendung eines 3-Port-Switches ermöglicht das Hintereinanderschalten mehrerer Gateways, was die Verkabelung am Einsatzort vereinfacht.

3.1.1 MpBus Basisplatine

Die Befähigung des Controllers zur Teilnahme an einer MpBus Kommunikation erfolgt durch die Verwendung einer eigens für diesen Zweck entwickelten MpBus-Adapterplatine. Entwickelt wurde die Platine als eine Basisplatine, wie in Abbildung 4 zu sehen, für ein NUCLEO-144 Board des Herstellers STMicroelectronics

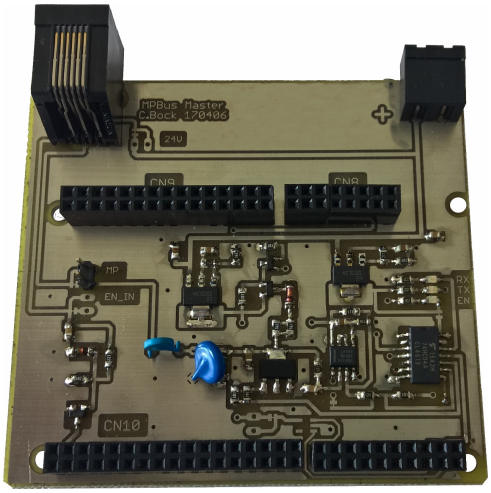


Abbildung 4: MpBus Basisplatine

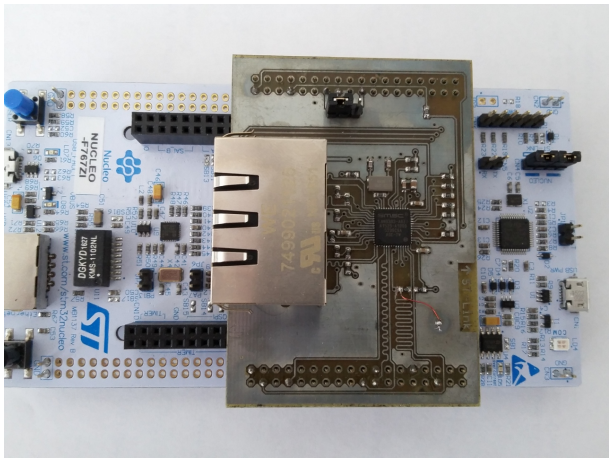
Hauptaufgabe der Platine ist es, ein vom Controller verwendetes UART kodiertes 3,3V Signal in ein MpBus spezifisches 18V Signal zu wandeln. Die Platine fungiert dementsprechend als Buskoppler mit Pegelwandlungsfunktionalität. Wahlweise ist es möglich die Platine über eine durch den MpBus gelieferte 24V Spannungsversorgung, oder aber eine externe Spannungsquelle versorgt werden. Dies ermöglicht entsprechende Flexibilität am konkreten Einsatzort.

3.1.2 Ethernet Shield

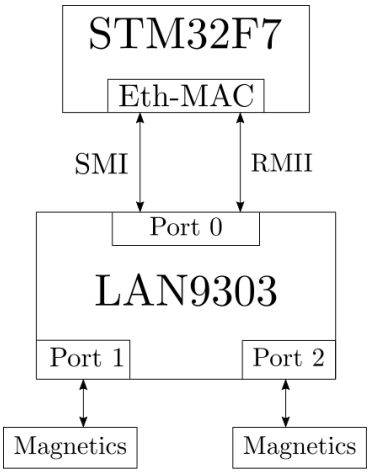
Die Oberseite der Platinenstapels bildet den physichen Ethernet Layer, bestehend aus einem LAN9303 3-Port Ethernet PHY-Switch der Firma Microchip sowie zwei Ethernet Buchsen (siehe Abbildung 5(a)).

Port 1 und 2 des LAN9303 Chips sind normale 100Mbit/s Ethernet Ports, die über Transformatoren mit einer Ethernetbuchse verbunden wurden. Der Port 0 hingegen wurde in dieser Anwendung als Virtual PHY konfiguriert und via RMII und SMI an den Ethernet MAC des STM32F7 Controllers angeschlossen. In diesem Modus gibt sich der

Switch als standard 1-Port PHY gegenüber dem Controller aus, wodurch keine Änderugnen am Ethernettreiber erforderlich sind. Bild 5(b) zeigt den schematischen Aufbau der Shieldplatine.



(a)



(b)

Abbildung 5: a) Ethernet Shield Platine, b) Schematischer Aufbau der Ethernet Shield Platine

3.2 Software

Die Firmware des Gateways basiert auf dem für den IoT Einsatz von ARM entwickelten Open Source Projekt MbedOS 5 [1, 2], welches sowohl einen integrierten Netzwerk Stack (LwIP), als auch ein RTOS Subsystem (RTX5) mit sich bringt. Um einen voneinander unabhängigen, nicht blockierenden Betrieb zu ermöglichen, setzt das Gateway auf die Verwendung des RTOS Systems. Dieses ermöglicht es die Aufgaben des Gateways auf einzelne Threads zu verteilen und getrennt voneinander zu behandeln.

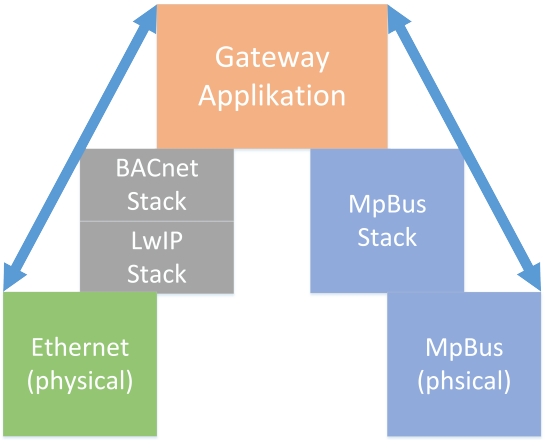


Abbildung 6: Gateway Threading/Netzwerk STacks

Entsprechend teilt das Gateway einzelnen Schnittstellen und Komponenten einen einzelnen Thread zu. Allgemein ist die Verteilung der Threads und ihrer Aufgaben lose an den Aufbau des ISO/OSI Schichtmodells angelehnt (siehe Abbildung 6), so dass einem durch das Gateway verwendetem Layer im ISO/OSI Schichtmodell, wo anwendbar, ein Thread mit spezifischer Aufgabe gegenübersteht.

Bei der Verwendung des MpBus gilt weiter, zu verhindern das unregelmäßige An- und Abfragen der Kommunikationsschnittstelle dazu führen das einzelne Nachrichten sich überschneiden und so zu unvorhergesehenem Verhalten führen. Aus diesem Grund verwendet das Gateway eine EventQueue. Dabei handelt es sich um einen

Mechanismus MbedOSs zur Einreihung einzelner, durch den Nutzer definierbarer, Events zur sequentiellen Abarbeitung durch einen der EventQueue zugeordneten Thread. Diese Funktionalität nutzt das Gateway um einzelne MpBus Kommunikationsabläufe für die Bearbeitung durch einen exklusiv für diese Aufgabe vorgesehenen Thread einzugliedern. Dies stellt sicher, dass stets nur eine MpBus-bezogene Aufgabe zurzeit durch einen Thread abgearbeitet wird.

Wie aus Abbildung 6 ersichtlich, findet ein Kommunikationsablauf des Gateways derart statt, dass die zugrunderliegenden Threads für den Empfang einer Ethernet basierten Nachricht, diese aufnehmen und nach deren vollständigen Empfang diese dem TCP/IP Stack zur Verfügung stellen. Realisiert wird der Stack durch eine Implementation das LwIP Stacks für Mikrocontroller, ebenfalls durch einen eigenen Thread auf dem Controller vertreten. In dem fortgesetzten Fall das eine UDP basierte Nachricht den TCP/IP Stack erreicht und fehlerfrei passiert hat, so wird diese Nachricht dem BACnet Stack übergeben, welcher den Datenanteil entsprechend dem BACnet Protokoll interpretiert und verarbeitet.

Die objektorientierte Organisation des BACnet Protokolls im Umgang mit Datenpunkten ermöglicht es dem Gateway durch individuelle Zuweisung einer Callback-Routine zu einem jeden im Gateway konfigurierten Datenpunkt, mit Eintreffen einer entsprechenden BACnet Anfrage (z.B. ReadProperty oder WriteProperty) eine für das entsprechende Datenpunktobjekt, der Anfrage angepasste, Callbackroutine auszulösen. Kernaufgabe der Callbackroutinen ist es, BACnet Requests und Acknowledges in korrespondierende MpBus Commands oder allgemeine Zustandsabfragen umzuwandeln. Wie zuvor angemerkt, werden diese Anfragen nicht direkt ausgeführt, sondern es ?ndet lediglich ein Eintrag in die o.g. EventQueue statt, welche hier als eine Art Puffer fungiert, um simultane bzw. sich überschneidende Buszugriffe auszuschließen.

3.2.1 MpBus Anwedungsebene

Die gatewayseitige MpBus Kommunikation als Busmaster, wird koordiniert durch eine, in Abbildung 7 dargestellte, Statemachine. Da applikationsseitig durch das MpBus Protokoll nur lose Vorgaben gegeben sind, richtet sich die Implementation vorrangig nach den Anforderungen des Einsatzzweckes. Daher übernimmt es die Statemachine, MpBus Kommunikationsabläufe, beschrieben durch das MpBus Protokoll, zu bündeln und als vollständigen Nachrichtenaustausch zwischen Master und Slave, masterseitig zu realisieren.Die Statemachine wird für jeden individuellen Command seitens des Masters aufgerufen und behandelt sowohl die Auswertung vollständig empfangener Answer Frames, wie auch Timeouts und Parityfehler.

Um eine Verwendung auch in RTOS Systemen zu ermöglichen und eine weitestgehend nicht blockierende Kommunikation zu ermöglichen, setzt die Statemachine auf Zustände wie „awaiting“ und „idling“. Diese werden von der Statemachine aktiviert, wenn keine aktiv zu verrichtenden Aufgaben stattfinden. Beispielsweise ein Nachrichtenversand oder das Warten auf die Freigabe einer Aktion in Folge eines, protokollarisch verlangten, noch ausstehenden Delays in der Kommunikation. In Kombination mit der die Statemachine abhandelnden Schleifenfunktion, erlauben es diese Zustände, den Thread in welchem die Statemachine ausgeführt wird, im Rahmen eines RTOS Zustandswechsels sicher zu verlassen oder zu pausieren und somit CPU-Zeit anderen Aufgaben zur Verfügung zu stellen.

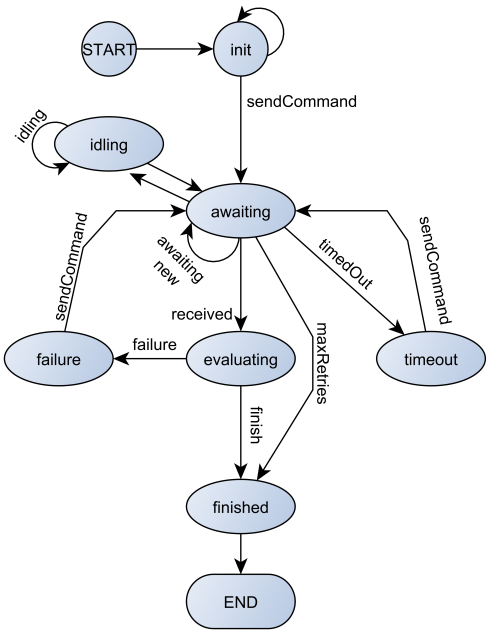


Abbildung 7: MpBus Statemachine

3.2.2 BACnet Stack

Für die Implementierung des BACnet Protokolls wurde der Open Source BACnet Stack von *Steve Karg* [4] auf das MbedOS portiert. Der Protokoll Stack enthält alle nötigen Mittel, um BACnet Requests und Acknowledges zu bearbeiten sowie Hilfsfunktionen für die Manipulation stackinterner Datentypen. Für die Portierung auf einen physischen Layer muss eine Schnittstelle geschaffen werden, die ankommende Datenpakete an den BACnet Stack übergibt und die Antworten des Stacks an die physische Schnittstelle. Im Falle des MbedOS wurde das vorhandene Ethernet Interface in Kombination mit einem UDP Socket genutzt, um so eine BACnet/IP Kommunikation bereitzustellen.

Um den BACnet Stack nutzen zu können, müssen eigene Handler Funktionen für Client Requests sowie eine eigene Implementierung der BACnet Objekte bereitgestellt werden. Die diversen Beispielprojekte, die dem Download des Stacks beiliegen, vereinfachen das Erstellen der Handler Funktionen, die teils unverändert in das Projekt übernommen werden konnten.

Der für dieses Projekt eingebundene Funktionsumfang des BACnet Protokolls lehnt sich an das B-ASC (Application Specific Controller) Geräteprofil an. Dazu gehört die Bearbeitung von `WHOIS`, `WHOHAS`, `READ PROPERTY` und `WRITE PROPERTY` Client Anfragen. Als Objekte wurden das erforderliche `Device Object` sowie `Analog-Input`, `-Output`, `-Value`, `Binary-Input`, `-Output`, `-Value` und `Multistate-Value` implementiert.

4 BACnet Simulink Blockset

Um das Prototyping der zentralen Druckregelung zu vereinfachen, wurde eine Simulinkbibliothek erstellt, welche die Kommunikation mit BACnet/IP Geräten ermöglicht. Der Funktionsumfang ist auf das, für eine Regelung benötigte Lesen, Schreiben und Subscriben der Present Values eines BACnet Objektes beschränkt.

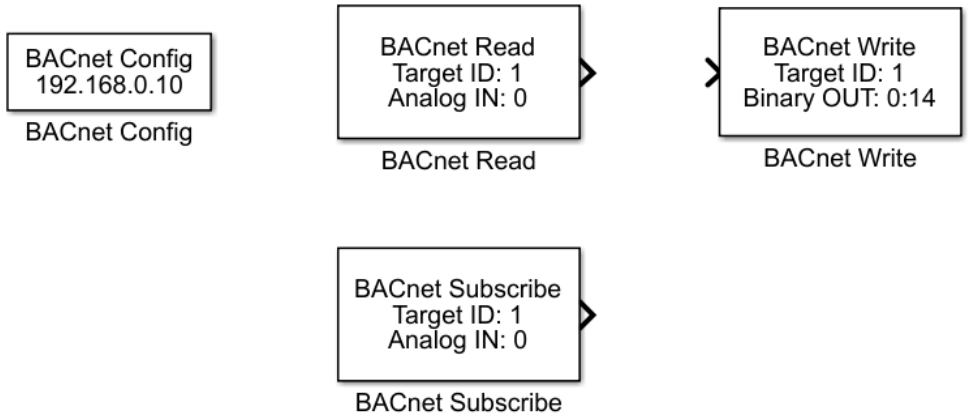


Abbildung 8: BACnet/IP Simulink Blockset

Abbildung 8 zeigt das Blockset für die BACnet/IP Kommunikation. Dies beinhaltet den `BACnet Config` Block, welcher die Konfiguration des Netzwerkinterfaces bereitstellt und einmalig in der obersten Modellebene eingefügt werden muss. Die Blöcke `BACnet Read` und `BACnet Write` ermöglichen das Lesen und Schreiben der Present Value Property eines der im Gateway implementierten BACnet Objekte. Um einen nicht blockierenden Betrieb zu ermöglichen, wird beim Aufruf des `BACnet Read` Blocks nur das `Read Property Request` gesendet, die Antwort steht frühstens im nächsten Abtastschritt des Simulink Modells zur Verfügung.

Um seltenere Ereignisse des BACnet Servers (z.B. Statusänderungen, manuelle Eingaben) nicht ständig abfragen zu müssen, kann mit dem `BACnet Subscribe` Block ein Objekt abonniert werden, um somit automatisch Änderungen der Present Value Property zu empfangen und diese im folgenden Abtastschritt bereit zu stellen.

5 Fazit

Im Rahmen dieses Projektes wurden bisher die folgenden vier Punkte realisiert. Die Portierung eines BACnet Stacks auf das MbedOS 5, was es ermöglicht mit sämtlichen vom Mbed Projekt unterstützten Plattformen BACnet Geräte zu entwickeln. Die Portierung des Stacks wird unter [7] veröffentlicht. Des Weiteren wurde, um die experimentelle Reglerentwicklung zu vereinfachen, mit dem vorgestellten Simulink Blockset die Möglichkeit geschaffen, direkt aus der Simulation heraus mit einem BACnet/IP Netzwerk zu kommunizieren. Der dritte Punkt ist die Implementierung einer MpBus Master Klasse, die ebenfalls auf dem MbedOS aufgesetzt. Als letzten Punkt ist die Zusammenführung der

einzelnen Komponenten auf prototypischer Hardware zu nennen, was in den weiteren Schritten des Projekts Tests in realen Reinraumumgebungen ermöglicht.

Danksagung

Dieses Projekt wurde gefördert durch das Land Mecklenburg-Vorpommern, Förderkennzeichen: TBI-V-1-135-VBW-048.

Literatur

[1] MbedOS Projekt: *https://www.mbed.com/en/*, (Zugriff 09.2017)

[2] MbedOS GitHub Projekt: *https://github.com/ARMmbed/mbed-os*, (Zugriff 09.2017)

[3] BACnet: *http://www.bacnet.org/Bibliography/AIC-97/AIC1997.htm*, (Zugriff 09.2017)

[4] BACnet Stack: *http://bacnet.sourceforge.net/*, (Zugriff 09.2017)

[5] *Belimo PP/MP Specifications*, [A91613-101, Rev26 / 05-Nov-2010]

[6] *Belimo MP Cooperation Documentation*, [A91613-100, Rev14 / 10-Dec-2012]

[7] BACnet für MbedOS *https://github.com/ATM-HSW* (Zugriff 09.2017)