### Prof. Dr.-Ing. Thomas Pucklitzsch

ist am 03.07.1976 in Halle/Saale geboren. Er ist verheiratet und hat zwei Kinder. Von 1996-2003 studierte Thomas Pucklitzsch an der Technischen Universität Dresden das Fach Informatik. Anschließend arbeitete er als CIO in einem Krankenhaus und wurde an der TU Dresden am Institut für Rechnernetze promoviert. Seit dem 01.04.2016 ist Thomas Pucklitzsch Dozent an der BA Glauchau im Studiengang Wirtschaftsinformatik.

**KONTAKT:** Berufsakademie Sachsen / Staatliche Studienakademie Glauchau
Kopernikusstraße 51 / 08371 Glauchau / pucklitzsch@ba-glauchau.de

### Prof. Torsten Lehnguth

ist Jahrgang 1965, verheiratet und hat einen erwachsenen Sohn.

Er studierte von 1985 bis 1990 an der Fakultät für Elektrotechnik der TU Chemnitz, damals Karl-Marx-Stadt, die Fachrichtung Informationstechnik. Nach seiner Diplomarbeit über Bewegungskompensation bei Kodierungsverfahren der Videodatenkommunikation und seinem Abschluss als Dipl.-Ing. führte ihn der berufliche Weg in verschiedene Ingenieur- und IT-Dienstleistungsunternehmen in Sachsen, Nordrhein-Westfalen und Baden-Württemberg. In seiner leitenden Tätigkeit bei Projekten der Prozessinformatik sowie von Automatisierungssystemen für Anlagen der Energie-, Versorgungs-, Verfahrens- sowie Fertigungstechnik konnte er wertvolle Industrieerfahrungen außerhalb des Hochschulbereiches sammeln. Neben Forschungs- und Entwicklungsarbeiten über Datenbankstrukturen sowie den Einsatz von Fuzzy-Logic und Künstlichen Neuronalen Netzen bei höheren Entscheidungs- und Optimierungsfunktionen in Prozessleitsystemen wirkte Prof. Lehnguth seit 2003 als Honorardozent an verschiedenen tertiären Bildungseinrichtungen in Sachsen und Thüringen. 2015 erfolgte die Berufung zum hauptamtlichen Dozenten für Elektrotechnik am Studiengang Technische Informatik der Berufsakademie Sachsen (Staatliche Studienakademie Glauchau). Ab Herbst 2018 findet unter seiner Leitung der Aufbau eines neuen Studienangebots „Digital Engineering“ in Glauchau statt.

**KONTAKT:** Berufsakademie Sachsen / Staatliche Studienakademie Glauchau
Kopernikusstraße 51 / 08371 Glauchau / lehnguth@ba-glauchau.de

# Enterprise Application Integration auf Raspberry-PI

*Thomas Pucklitzsch / Torsten Lehnguth*

*Moderne Unternehmen brauchen immer mehr Anwendungen. Dem Trend entsprechend wird die Enterprise Application Integration (EAI) in Zukunft an Bedeutung gewinnen. Unser System mit dem Namen Oktopus ist eine Lösung für diese Herausforderung und in der Lage, verschiedene Anwendungen zu integrieren. Ein weiterer Trend ist zu beobachten: Immer mehr Geräte des täglichen Lebens werden mit dem Internet verbunden, um Daten zu liefern und über Apps gesteuert zu werden. Dies wird das „Internet der Dinge" (IoT) genannt. Zum Steuern dieser Geräte werden kleinen Computern wie der Raspberry-PI oder Rechner der Arduino-Serie benutzt. In diesem Beitrag werden neue Ideen vorgestellt, um die Welt des IoT mit der Welt der EAI-Systeme zu verbinden. Die Software verbindet dabei verschiedenen Abstraktionsebenen. Auf Schnittstellen niedrigerer Ebene werden beispielsweise Pins des Raspberry-PI ausgelesen oder angesteuert und an anderen Schnittstellen Emails versandt.*

*Ein einfaches Beispiel könnte ein http-Request sein, der von Oktopus als Ergebnis interpretiert wird dessen Antwort z. B. das Steuern eines Ports des Raspberry-PI zum Einschalten einer Diode sein könnte. Um dieses Ziel zu erreichen, haben wir eine neue Schnittstelle für die Oktopus-Software entwickelt, um die Temperatur zu messen und das Ergebnis auf einem anderen Oktopus-Interface zu visualisieren.*

*Modern companies need more and more applications. In accordance with this trend enterprise application integration (EAI ) will become more important in the future. Our system named Oktopus is a solution to face this challenge and is able to integrate various applications. Another modern development can be seen in our society: Connecting different devices like lamps, heaters or toasters with the internet with the aim of controlling and collecting data. This is known as the term "Internet of Things"(IoT). Processing of this kind of data is supported today by small computers like the Raspberry-PI or the Arduino-Series. This paper presents new ideas for connecting the world of IoT with the world of EAI systems. For this reason data from different devices are collected, stored and transmitted to other systems. Incoming messages are processed in order to control other application by generating user-defined events. This is possible on different levels of abstraction: at lower level reading and writing a pin, at higher level sending a message to other systems.*

*A simple example could be an E-Mail received from Oktopus as a result of a special event. One kind of answering is to fire further events (e.g. controlling a port of the Raspberry-PI to switch a diode on). To reach this goal we have developed a new interface for the Oktopus-software and used it to measure the temperature and visualize the result on another Oktopus-Interface.*

## 1 Einleitung

Moderne Anwendungsintegrationsansätze, wie das Konzept des Enterprise Service BUS, kapseln Anwendungsschnittstellen in Webservices, um sie so in die Serviceorientierte Architektur eines Unternehmens einzubinden. Oktopus bietet den Nutzern einen anderen Ansatz, bei dem die Kommunikation mittels unabhängiger Nachrichten abgebildet wird. Dies bietet vor allem dann mehr Flexibilität, wenn sehr unterschiedliche Systeme, deren Schnittstellen stark differieren, miteinander verknüpft werden sollen. Wenn man den Gedanken der Integration fortführt und nach neuen Bereichen im Unternehmen sucht, die Informationen liefern und mit dem Gesamtsystem verbunden werden sollen, dann stößt man bald auf den derzeit stark wachsenden Bereich des Internets der Dinge (IoT). Darunter versteht man die vielen Geräte, die mehr und mehr an das Internet gekoppelt werden und Daten an bestimmte Portale senden oder Befehle empfangen. Dieser Trend wurde unterstützt durch die Verfügbarkeit einer Generation von kleinen energiesparenden Rechnern, die überall zu geringen Kosten eingebaut werden können. Die bekanntesten sind der Raspberry-PI oder auch die Arduino-Serie. Während Arduinos Mikrocontroller sind, auf denen kein Betriebssystem läuft, ist der Raspberry-PI ein kompletter PC, auf dem BSD oder ein bestimmtes Linux namens Raspian (der Begriff setzt sich aus der Linux-Distribution Debian und Raspberry zusammen) betrieben werden kann. Dennoch besitzt der Raspberry eine ganze Reihe digitaler Anschlüsse, die sowohl als Eingabe, als auch als Ausgabe verwendet werden können. Darüber hinaus existieren für den Raspberry-PI diverse Python-Bibliotheken, um diese Pins anzusteuern. Das PI im Namen steht im Übrigen für "Python-Interpreter", da der Raspberry-PI zunächst als Mikrocontroller mit einem Python-Interpreter konzipiert wurde. Da Oktopus eine plattformunabhängige Software ist und darüber hinaus auch sehr ressourcenschonend arbeitet, ist es problemlos möglich, die Software auf einem Raspberry-PI zu starten.

## 2 Das Konzept von Oktopus

Diese Plattformunabhängigkeit ermöglicht Oktopus einen sehr flexiblen Einsatz überall dort im Unternehmen, wo heterogene Systeme

in die Systemlandschaft integriert werden müssen. Oktopus wird im Krankenhausumfeld produktiv eingesetzt und dort dazu verwendet, um die Kommunikation mittels HL7-Schnittstellen zu koordinieren und die einzelnen HL7-Dialekte untereinander zu übersetzen. Die Software ist jedoch generisch und kann beliebige Nachrichtenformate verarbeiten. Dazu wird für jede Gruppe von Nachrichtentypen, welche von Oktopus verarbeitet werden soll, ein Template im XML-Format angefertigt. Dies kann automatisch durch dafür zur Verfügung stehende Werkzeuge anhand einer Beispielnachricht erzeugt werden. Auf diese Art und Weise kann ein Unternehmen beliebige neue Nachrichtenformate in das System integrieren, ohne die Software anpassen zu müssen. Mit Hilfe dieses Templates wird eine ankommende Nachricht in eine Normalform konvertiert. Dabei handelt es sich um ein mehrdimensionales Array, welches aus Attribut-Wert-Paaren besteht, die hierarchisch in Sektionen gruppiert sein können. Diese Gruppierung ermöglicht eine Wiederholung beliebiger Sektionen nach klar definierten Regeln. Die Nachrichten können dann abhängig von ihrem Inhalt beliebig geroutet werden. Hierzu können Regeln definiert werden, die gemäß einer aussagelogischen Formel in disjunktiver Normalform ausgewertet werden. Auf diese Weise kann eine beliebige Aussage anhand einer Nachricht auf ihren Wahrheitsgehalt überprüft werden.

Darüber hinaus bietet Oktopus die Möglichkeit, über virtuelle Schnittstellen sein eigenes Verhalten dynamisch, abhängig von eingehenden Nachrichten anzupassen. So kann beispielsweise eine beliebige externe Nachricht ein Interface stoppen, starten oder eine neue Route hinzufügen. Oktopus unterstützt außerdem sogenannte Systemschnittstellen. Diese dienen nicht zur Kommunikation mit Anwendungssystemen, sondern ermöglichen es, Nachrichten in einer eigenen dokumentbasierten Datenbank zu archivieren, sequenziell empfangene Nachrichten zu einer Nachricht zu verbinden oder die Reihenfolge von Nachrichten zu verändern. Für die Anbindung von Anwendungssystemen stehen eine TCP/IP-Socketschnittstelle, eine HTTP-Schnittstelle, eine E-Mail-Schnittstelle, eine File-Schnittstelle, eine XML-RPC-Schnittstelle, diverse Datenbankschnittstellen und viele mehr zur Verfügung. Wird auf einer dieser Schnittstellen eine Nachricht empfangen, so wird diese anhand des XML-Templates in ihre Normalform gebracht. Bevor diese an ihre Zielschnittstelle ausgegeben wird, kann die Nachricht noch mit Informationen aus anderen Nachrichten angereichert werden. Dies geschieht mit Hilfe einer Suche in der Datenbank. So kann beispielsweise im Krankenhausumfeld die Statusmeldung eines Subsystems mit Informationen aus der Auftragsnachricht oder der Stammdatennachricht aufgefüllt werden. Dies ist ein integraler Bestandteil von Enterprise Application Integration, da dadurch neben der syntaktischen Integration auch eine semantische Integration stattfinden kann, wenn ein Subsystem weniger Informationen liefert als ein anderes erwartet. Die Intension von Oktopus ist, Systeme auf ganz unterschiedlichen Ebenen anzubinden. Beispielsweise wurde eine Viewer-Schnittstelle entwickelt, welche die Ausgabe einer beliebigen Nachricht in Form von Symbolen auf einem Display realisiert. Die Viewer-Schnittstelle verfolgt wie eine Sprach-Schnittstelle das Ziel, Applikationen nicht nur mit anderen Applikationen, sondern auch mit menschlichen Akteuren über eine Mensch-Maschine-Schnittstelle zu verbinden. Mit einer solchen Schnittstelle ist es möglich, Warnungen oder visuelle Hinweise aus diversen Ausgaben unterschiedlicher Applikationen zu erzeugen.

## 3 Raspberry-PI Schnittstelle

Das Ziel dieser Arbeit ist es nun, eine Schnittstelle zu entwickeln, die passend zum Gesamtkonzept eine Möglichkeit bietet, Informationen, die man über die GPIO-Anschlüsse von bestimmten Geräten auslesen kann, in eine Nachricht von Oktopus zu transformieren. Umgekehrt sollte es auch möglich sein, Nachrichten, die über eine der zahlreichen Schnittstellen von Oktopus kommen, in ein Signal umzuwandeln und damit Geräte zu steuern. Leider besitzt der Raspberry-PI nur digitale Ein- und Ausgänge. Trotzdem ist es möglich, analoge Werte mittels seiner GPIO-Anschlüsse zu messen. Das ist möglich, da die Anschlüsse des Raspberry-PI dynamisch von einem Ausgang in einen Eingang und zurück umgeschaltet werden können. Man kann analoge Werte von Sensoren übertragen, indem man Zeiten misst, in denen ein GPIO-Anschluss auf Eins oder auf Null gezogen wird. Dies kann man beispielsweise mit einem Kondensator erreichen, der über den internen Pullup-Wiederstand des Raspberry-PI entladen wird. Es reicht hierzu nicht aus, lediglich das Bit eines Eingabepins auszulesen, sondern es muss möglich sein, eine generische Zeitmessung durchzuführen. Auf diese Weise kann man eine große Bandbreite physikalischer Größen wie die Helligkeit oder die Entfernung mittels eines Ultraschallsensors oder eben die Temperatur oder Luftfeuchte messen, denn die meisten passiven Sensorprinzipien beruhen auf der Änderung der elektrischen Eigenschaften eines bestimmten Materials. Ziel war es darüber hinaus auch, Schaltsequenzen der GPIO-Anschlüsse des Raspberry-PI umzusetzen. So kann man beispielsweise eine Diode einsetzen, die infrarotes Licht emittiert. Bestimmte Sequenzen, mit der diese Diode ein- und ausgeschaltet wird, können zur Fernsteuerung eines Gerätes mit Infrarotsensor verwendet werden. Auf diese Weise kann man Oktopus als eine Fernbedienung verwenden, die durch Webservices oder eine HTML-Seite gesteuert werden kann.

## 4 Temperaturmessung mittels Kondensator

Um Oktopus als Temperaturmesser in ein Netzwerk einzubinden, wurde ein Keramikkondensator verwendet, dessen Permittivität des Dielektrikums einen starken Temperaturkoeffizienten besitzt. Das Temperaturverhalten dieses Kondensators ist bezüglich seiner Kapazität nahezu linear und kann beim Herstellungsprozess präzise erzeugt werden. Die Temperaturabhängigkeit wird in der Regel durch Bezeichnungen wie beispielsweise ”N220“ angegeben.

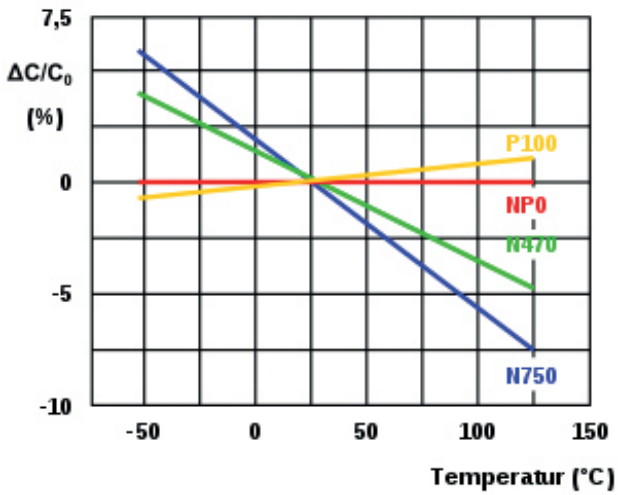


Abbildung 1: Temperaturabhängigkeit unterschiedlicher Klasse-1 Kondensatoren [2]

Verwendet man einen Kondensator mit hoher Temperaturabhängigkeit, so variiert die Kapazität dieses Kondensator stark bei einer Änderung der Temperatur. [1] Die Kapazität wiederum hat einen Einfluss auf die Entladezeit. Die Zeitkonstanten der Lade- bzw. Entladevorgänge sind abhängig von der Kapazität und dem Widerstand über den der Kondensator ge- bzw. entladen wird. Mit $\tau = R \cdot C$ ist diese Zeit linear abhängig von der Kapazität. Allerdings wird nicht erst bei 0 Volt der Wert des GPIO-Pins als Null interpretiert. Der Schwellwert liegt beim Raspberry-PI bei ca. 1,2 Volt.

Die Ausgangsspannung des Kondensators zum Zeitpunkt t ergibt sich aus der Gleichung: [3]

$$u_c(t) = U_0 \cdot e^{-\frac{t}{\tau}}$$

Stellt man die Gleichung nach t um, so erhält man:

$$t = -ln(\frac{u_c(t)}{U_0}) \cdot \tau$$

Da die Ladespannung und der Schwellwert konstant sind, ist die Entladezeit bis zum Schwellwert linear von der Kapazität abhängig und somit kann mittels der Zeit, die der Kondensator benötigt, bis die Spannung auf unter 1,2 V abgefallen ist, auf die Kapazität geschlussfolgert werden, die wiederum in linearem Zusammenhang zur Temperatur steht.
Als Widerstand kann der interne Pullup-Widerstand des Raspberry-PI genutzt werden, der einen Widerstand von 50 $k\Omega$ besitzt. Um die Kapazität zu messen, wird zunächst ein Anschluss als Ausgang konfiguriert und der Kondensator damit geladen. Anschließend wird der Anschluss zum Eingang gemacht und in kurzen Zeitabständen geprüft, ob eine Eins oder eine Null anliegt. Zunächst einmal liegt am Eingang eine Eins an. Sobald die Spannung auf unter 1,2 V gesunken ist, wird eine Null gelesen. Mit dieser Methode kann die Entladezeit des Kondensators gemessen werden, aus der die Kapazität und damit auch die Temperatur berechnet werden kann.

## 5 Echtzeitfähigkeit

Die einzelnen Schnittstellen von Oktopus werden durch ein einfaches Loadbalancing gesteuert.
Kommen keine Nachrichten auf einer Schnittstelle an, so wird der entsprechende Thread für eine längere Zeit in den Schlafmodus versetzt. Müssen viele Nachrichten verarbeitet werden, so wird diese Zeit schrittweise um eine Größenordnung verkürzt. Dieses Verhalten ist für die Steuerung von Geräten und das Empfangen von Signalen nicht wünschenswert. Aus diesem Grund darf die Oktopus-Schnittstelle kein Loadbalancing durchführen, sondern muss in regelmäßigen Abständen die Eingänge überprüfen und die Warteschlange auf neue Nachrichten überprüfen. Da ein Raspberry-PI mit einem bedingt echtzeitfähigen Betriebssystem betrieben wird, ist trotz dieser Maßnahme keine echte Echtzeitfähigkeit gegeben. Wenn jedoch auf dem Rechner ausschließlich Oktopus ausgeführt wird, kann man davon ausgehen, dass der Oktopusprozess genügend Ressourcen zur Verfügung gestellt bekommt, um die Zeitmessung hinreichend exakt durchzuführen.

## 6 Die Umsetzung

Für die Umsetzung ist von Bedeutung, welche Parameter im Konfigurationsfile von Oktopus konfiguriert werden können. Diese Konfiguration sollte sich möglichst einfach gestalten, um den Benutzer nicht mit den Details der Zeitmessung zu konfrontieren. Dennoch muss es möglich sein, sowohl zu prüfen, ob ein Potenzial an einem einzelnen Pin anliegt oder nicht, aber auch die Entladezeit eines Kondensators zu messen.
Eine Beispielkonfiguration für diesen Anwendungsfall ist folgende:

Listing 1: Oktopus-Konfiguration

```
INTERFACE r a s p i
TYP ra spbe r ry
FREQUENCY 10
PIN5
PIN10 LOADTIME 500000
PIN10 UNIT Degrees
PIN10 ACCURACY 5
PIN25 LOAD 24
PIN25 LOADTIME 20
PIN25 UNIT Meter
XML i n t e r f a c e s / r a s p i . xml
END r a s p i
```

Die Beispielkonfiguration veranlasst Oktopus, drei verschiedene Anschlüsse zu überwachen. Dabei wird PIN 5 nur darauf getestet, ob eine Spannung anliegt oder nicht, d.h. der Anschluss mit Eins oder Null belegt ist. Dabei wird nur dann eine Nachricht erzeugt, falls der Wert sich seit der letzten Prüfung verändert hat. Die Pins 10 und 25 dienen zum Messen einer Zeitspanne.
Zunächst einmal muss eine Ladezeit angegeben werden. Diese Zeit wird in Mikrosekunden angegeben. Oktopus benutzt PIN10 zunächst als Ausgang und setzt diesen für die Ladezeit auf den Wert Eins. Nach Ablauf der Ladezeit wird PIN10 umkonfiguriert und als Eingang verwendet. Nun testet Oktopus den Wert des PINs und misst die Zeit, bis dieser wieder auf LOW zurückgeht.
Für den Raspberry-PI gibt es bereits verschiedene Sensoren, beispielsweise solche, die mittels Ultraschall die Entfernung zu einem Objekt messen. Hierbei wird ein PIN als Eingang und einer als Ausgang konfiguriert. Wird der Ausgang kurz auf Eins und anschließend wieder auf Null gesetzt, dann sendet der Ultraschallsensor ein Signal aus und setzt den Eingang auf eins, sobald die Reflektion wieder am Sensor empfangen wird. Das Prinzip der Ansteuerung ist hier identisch mit der Temperaturmessung mit dem Unterschied, dass zwei verschiedene Anschlüsse zum Auslösen einer Messung und Empfangen des Signals verwendet werden. Aus diesem Grund darf in der Konfiguration von Oktopus zu einem zu überwachenden Anschluss noch ein anderer angegeben werden, auf dem die Ladespannung gelegt wird. Dies ist bei PIN 25 der Fall. Hier wird PIN 24 als Trigger verwendet. In der Konfiguration kann man pro Anschluss noch zwei andere Parameter angeben. Zum einen kann mit “UNIT“ eine Einheit angegeben werden. Dieser Wert ist dann in der von Oktopus erzeugten Nachricht unter dem gleichnamigen Schlüssel hinterlegt.
Hier kann man auch einen Umrechnungsfaktor angeben, denn der Wert, der gemessen wird, ist ja eine Zeitspanne, die in irgendeiner Weise proportional zur zu messenden Einheit ist. Das Schlüsselwort “FREQUENCY“ gibt an, wie oft pro Sekunde eine Messung durchgeführt werden soll. Da nur dann eine Nachricht erzeugt wird, wenn der Messwert vom zuletzt gemessenen Wert abweicht, kann man mit Hilfe des Schlüsselwortes “ACCURANCY “ einen Wert angeben, um den der Messwert mindestens abweichen muss, damit eine Nachricht erzeugt wird.

## Das Ergebnis

Mit der angegebenen Konfiguration sendet Oktopus bei jeder Kapazitätsänderung des Kondensators eine Nachricht. Diese muss nun weiterverarbeitet werden. Um die Werte zu visualisieren, wurde im Beispiel ein Viewer-Interface von Oktopus verwendet. Wird Oktopus gestartet, dann wird eine Init-Nachricht generiert. Diese wird an das Viewer-Interface geschickt und mittels eines geeigneten Templates die Abbildung eines Thermometers erzeugt. Nun können die Nachrichten der Raspberry-Schnittstelle dazu verwendet werden, um einen Y-Wert der Quecksilbersäule zu erzeugen. Hierfür können in der Konfigurationsdatei mit der Funktion “SMOD“ diverse arithmetische Operationen definiert werden.

Listing 2: Oktopus-Konfiguration

```
ROUTING
SMOD r a s p i _ tmp=>[value ]--100000
SMOD r a s p i _ tmp=>[tmp]/ / 2 0
SMOD r a s p i _ tmp=>12000--[tmp ]
SMOD r a s p i _ tmp=>[tmp] _ _ 0 . 0 0 4 7 5
SMOD r a s p i _ Y1=>[tmp]++15
RULE r u l i n i t event=i n i t o k t o p u s
ROUTE system r u l i n i t thermo Thermo{_}
ROUTE r a s p i _ thermo
ROUTE c o n t r o l _ system
ROUTE command _ r a s p i
ROUTE l o g g ing _ l o g s y s
END ROUTING
```

Die Werte wurden mit Hilfe eines Thermometers kalibriert, indem zwei verschiedene Zeiten den passenden Ordinatenpositionen zugeordnet wurden. Aus der daraus resultierenden Geradengleichung können der statische Empfindlichkeitsfaktor und eine Offset berechnet werden. Im Beispiel wurden jedoch mehrere arithmetische Operationen hintereinander durchgeführt, um Rundungsfehler zu minimieren.

## 8 Zusammenfassung

Die Raspberry-Schnittstelle ermöglicht es, auf einfache Art und Weise mittels Oktopus unterschiedliche Nachrichten aus verschiedenen Systemen dazu zu verwenden, digitale PINs des Raspberry-PI zu steuern und mittels einer Zeitmessung auch analoge Werte diverser Sensoren zu ermitteln. Auf diese Art und Weise wird Oktopus zu einer Brücke zwischen verschiedenen Welten. Es ist damit möglich, Messwerte dazu zu benutzen, bestimmte Events auszulösen. Man könnte beispielsweise mit Oktopus eine SMS verschicken, falls die Temperatur in einem Raum zu hoch wird. Im Krankenhausumfeld könnte man hiermit Fieberthermometer entwickeln, welche ihre Messwerte direkt per HL7 in die elektronische Patientenakte des Krankenhausinformationssystems sendet.
Auf diese Art und Weise können nun Geräte ohne größeren Entwicklungsaufwand in das IT-System eines Unternehmens sinnvoll eingebunden werden.

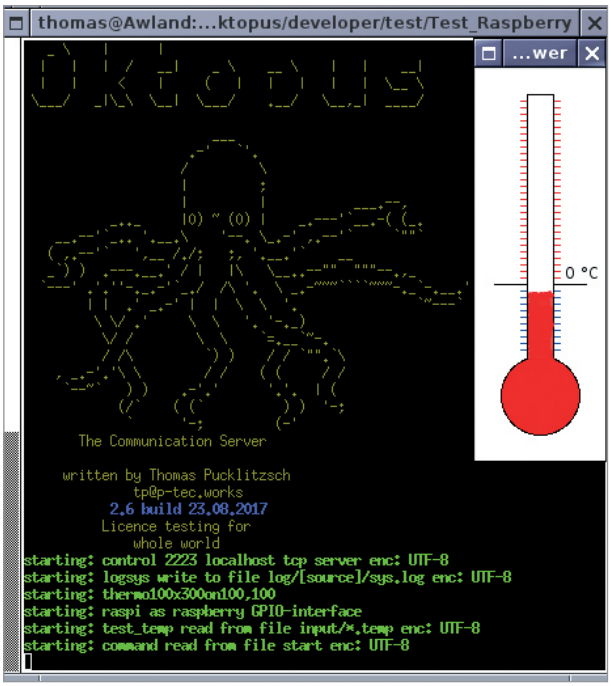


Abbildung 2: Viewer-Interface mit Thermometerkonfiguration

## Literaturverzeichnis

[1] Kainka, B.: http://www.elektronik-labor.de/Raspberry/RpiGPIO4.html
[2] Altmann, S.; Schlayer, D. (2008): Lehr und Übungsbuch Elektrotechnik Fachbuchverlag Leipzig.
[3] Möller, F. (1991): Grundlagen der Elektrotechnik. Vieweg + Teubner Verlag, Wiesbaden.