論　　文　　要　　旨

申請者氏名　プラウィーン　アモーンタマウット

申 請 学 位　博士（工学）

主論文題目　IoTを指向したシステム開発環境の研究

主論文要旨（邦文は４，０００字以内 / 外国語は２，０００語以内）

本論文では,モノのインターネット化（以下 IoT）を指向したシステムに関する開発環境およびそのシステムを用いたIoT学習支援への適用に関する研究について述べる.

ドイツにおけるIndustrie 4.0などを始めとして，従来の組込みシステムを備えたモノをインターネットに接続して，より多くのサービスを提供するIoTは，情報産業だけではなく従来の産業にも大きな影響を与えつつある．その中で，プログラミングを容易にする環境の構築や，組込みシステムおよびネットワークシステムの両面に精通したエンジニアの育成は，大学および企業においても重要な課題となっている.

このようなIoT開発においては，扱う機器が多種多様であり，プログラマやシステム管理者にとって負担となっている．例えば，各機器でのセンシングやアクチュエーションを実行するプログラミングインタフェースは，機器の性質に応じた形で定義されていることが多い．また，各機器類の操作インタフェースと，ネットワークの接続インタフェースとは異なるモデルで実装されていることから，開発者にとって難しい課題となっている．また，IoTでは各機器が備えたセンサからのサイズの大きなデータを扱うが，これらを収集，格納，閲覧する一貫した枠組みが存在しない.

また，IoTはその一部に組込みシステムを含んでいることから，システムのオーバヘッドを減らし，

実時間性を向上させる必要がある．また，ネットワークを構成要素に持つことから，システムの負荷を計測し，システム設計に反映させる必要がある．これに対して，従来のシステムでは組込みシステムやネットワークシステムの状態や負荷，通信遅延などを取得，保存し，閲覧するための一貫した枠組みがない．このことから，開発や教育現場において使い勝手の異なる複雑なツールに頼らざるを得ないという問題がある．

さらに，プログラム開発において機器ごとに開発環境が異なり，セットアップの手間のコストが高いことも問題として挙げられる．開発環境は開発マシンやOSに依存していることから，演習などのように扱う台数が増えた場合に，管理コストが増大してしまう．

これらの問題に対して，本研究では組込み機器，およびネットワーク管理，プログラミング環境まで一貫した管理機構を提供することで，問題を解決する．本システムでは，システムを組込み機器（以下，組込みノード），ネットワークとアプリケーション実行を管理するゲートウェイ，プログラミング環境を提供するブラウザの3つの構成要素によって構築する．組込みノードの多種多様性に対応するために，ノードは統一したインタフェースを備えた形で仮想化し，システムからは仮想的なノードとして扱えるようにする．これによって，多種多様な組込みノードを統一して管理，プログラミングすることが可能になる．

また，管理機構自体にロギングのメカニズムを備えて，組込み機器として必要となるプロセスのロギング，およびノード状態のロンギング，さらに，ネットワークの遅延監視機構を用意した．ネットワークは，ネットワークおよびアプリケーションの実行を可能にするように，ゲートウェイを用意し，その中でだけノードにアクセスできるように限定している．さらに，開発環境，管理ツール，可視化ツールを，多くのPC上で動作するブラウザ上で動作させることで，管理コストを低減させつつ機能を提供可能にした．そして，多様性を確保するために，異種プロトコルを持つシステムに対しても相互アクセスを可能にさせ，接続性を向上する機構を用意した．

まず，IoTの構成要素である組込みシステムにおいて，リアルタイム性に影響を与えるプロセス状態について，プロセス状態のロギングオーバヘッドを低減させる手法を提案した．本手法は，Linuxカーネルのコンテキストスイッチへの影響を低く抑えることと，通信負荷を低減させるように，ログの性質に応じた圧縮手法を設計して，従来のFtrace機構に追加した．また，プロセスの動作への影響を低減させるために，カーネル空間で動作し，対象マシン外に転送するトレースドライバを実装した．本手法を実際に実装して評価した結果，ログサイズが平均40%低減し，割込みについても60%低減した．これによって，システムへの負荷を減らしつつ，長期間のロギングが可能になった．

次に，IoT向きのウェブベース開発環境の構築を行った．本開発環境は，Linuxベースの組込みノー

ド，およびゲートウェイ，および利用者サイドのWebブラウザの三つのコンポーネントが連携して動作する．システムは，単一のインタフェースを持つ仮想的なノードとして管理している．これによって，多種のノードであっても，統一したAPIを提供している．さらに，各コンポーネント間の通信には，HTTPを基本としたREST APIベースのプロトコルを提供している．これによって，多くの組織で標準的なプロキシを含んだ構成であっても接続性を維持することができる．さらに，ブラウザ上ではプログラミングエディタ，プログラム実行，コンソール，通信プロトコルのシーケンス表示，およびノード状態の表示を可能としている．プログラミングエディタとしては，プロトタイピング向きのコマンドラインベーススクリプトおよびJavaScriptをサポートして，利用状況に応じた使い分けを可能とした．さらに，ブラウザ内で作成したアプリケーションを実行することで，実行環境を用意せずに動作検証することを可能としている．実行環境はゲートウェイ上に移設することができるので，ネットワークのRTTに応じたシステム設計を検討，実装することが容易である．さらに，通信プロトコルのシーケンスおよびノード状態については，グラフィカルに表示する可視化機構も用意することで，ノード数が増えた場合の閲覧性を改善した．

　このシステムを開発し，次の3点で評価を行った：(1)システムオーバヘッド，(2)開発コードサイズと時間，(3)可視化機構を用いた改良事例．(1)については，ゲートウェイが2m秒，ノードが8m秒と低オーバヘッドで実行でき，デバイス操作やデータ取得への影響は低いことが確認できた．また，(2)については，APIを高機能化することで90%以上のコード削減を可能とした．(3)については，リモート環境のRTTを可視化し，アプリケーションをノードの近くに移動させることでRTTが低減する開発プロセスについて示し，本システムの有効性を明らかにした．

　そして，本開発環境を用いたIoTプログラミング学習支援環境を立ち上げて，実際のIoTの学習を行い，教育への適用可能性について検証した．本実験は，JavaScriptの言語経験はあるがJavaScriptでIoTの開発経験がない本学工学部4年生7名を対象として，IoTの基本的なアプリケーションの開発課題を与えた場合のコード行数および，開発時間の測定を行い，その後アンケートにおいて被験者に対する使いやすさやIoTの理解度を取ることで，その有効性を測定した．具体的には，(1)LED点灯，(2)PWMおよび(3)複数のセンサ状態を取得して，その中から対象を特定する課題である．

　実験結果は，(1)ではすべての学生が8分以内に課題を終了している．(2)については最長でも26分であった．また，(3)については，IoTにおけるセンサ情報取得と処理という基本的な題材であるが，最大でも30分以内に終了している．コードサイズも最大で50行程度と短い行数であり，使いやすさ，理解度についてはいずれも3以上である．このことから，本システムは教育現場においても有効であることが明らかになった．また，実験に用いた20台のマシンも短時間でセットアップが可能であり，

多くの台数を整える必要がある環境においても有効であることが明らかになった．

最後に，多様なマシンとの接続性を向上させるために，ROSをターゲットとしてプロトコル変換および接続機構を用意した．ROSと本システムが提供しているプロトコルとは，基本となるモデルが異なる．しかし，IoTにおいて多様性は必須であることから，より一般的なシステムであるROSを，本システムで対応できるようにすることで，より多くのデバイスを扱うことができる．接続性を向上させるために，ゲートウェイにおいてプロトコルの変換機構を用意した．相互変換することで，双方向にシステムを呼び出す事が可能になる．また，基本的なデータ通信以外に，ノードの状態についても取得することにより，異機種が混在したマシン環境であっても，本システムの可視化環境を利用したシステム監視を可能とした．実際にシステムを実装し，可視化および双方向の呼出し実験を行い，有効性を確認した．また，プロトコル変換を含むオーバヘッドは平均20m秒であるが，ロボットを含むデバイス操作を対象した場合には，十分な低オーバヘッドであると考えることができる．

これら４つの研究を通して，IoTを指向したシステム開発環境の研究を行い，IoTに向いているシステム構成について明らかにした．IoTは単一なシステムではなく，組込みシステム，ネットワーク，アプリケーションやソフトウェアの開発，そしてプログラミング手法において，相互に依存関係がある．システムを統合するには，各システム自体のオーバヘッドの検討が不可欠であるが，異なるシステムを一つに統合する際にオーバヘッドのトレードオフは避けられない．このようなオーバヘッドのトレードオフ問題を解決するには，可能な限りオーバヘッドを押さえたシステム構成を採用する必要があるが，オーバヘッドはなくすことは不可能である．そこで，システム全体においてオーバヘッドの計測機構を用意し，システム間で調整することによって，その影響を十分に抑えることが可能であることが明らかになった．このような計測機構を用いて，開発者や学習者にオーバヘッドの存在を示し，それを用意に検証できるシステム構成が，IoTのシステム開発環境には不可欠であることが，本研究を通して明らかになった．