社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－DPS－124（6）
2005－ GN － 57（6）
2005／9／21

イベント駆動型入出力制御デバイスのためのアプリケーション開発環境

相良亮平 †　岸野泰恵 †　寺田 努 †　義久智樹 ‡　塚本昌彦 *　西尾章治郎 †

† 大阪大学大学院情報科学研究科
‡ 京都大学学術情報メディアセンター
* 神戸大学工学部

あらまし： 本稿ではユビキタスチップのためのアプリケーション開発環境を提案する．ユビキタスチップとはユビキタスコンピューティング環境構築のために筆者らが提案しているイベント駆動型入出力制御デバイスである．提案する開発環境は複数のユビキタスチップの動作をシミュレートし，ユーザが容易にその動作を記述できるようにする．さらに，提案する開発環境は，開発環境内の仮想的なユビキタスチップと実空間に配置されたユビキタスチップを統合的に扱う機能をもち，実空間のセンサやスイッチなどのデバイスを用いたアプリケーション開発が行える．提案システムを用いることで，プログラマだけでなく一般のユーザもユビキタスコンピューティング環境におけるアプリケーションを開発できるようになる．

An Application Development Environment for Rule-based I/O Control Devices

Ryohei Sagara†, Yasue Kishino†, Tsutomu Terada†, Tomoki Yoshihisa‡, Masahiko Tsukamoto*, and Shojiro Nishio†

† Graduate School of Information Science and Technology, Osaka University
‡ Academic Center for Computing and Media Studies, Kyoto University
* Faculty of Engineering, Kobe University

Abstract : In this paper, we propose an application development environment for ubiquitous chip, which is a rule-based event-driven input/output (I/O) control devices for constructing ubiquitous computing environments. The proposed development environment simulates behaviors of multiple ubiquitous chips and helps users to create rules. Moreover, it has a function to develop applications by cooperation between virtual ubiquitous chips and real ubiquitous chips. The application environment enables not only programmers but also general users to develop and customize applications for ubiquitous computing environments.

1　はじめに

近年，コンピュータやマイクロチップ，センサ，無線モジュールといった機器の小型化，低価格化により，ユビキタスコンピューティング環境が実現しつつある [4, 9]．筆者らの研究グループではこれまでに，イベント駆動型ルールで動作する入出力制御デバイスを用いた新しいユビキタスコンピューティングを提案している [8]．このデバイスをユビキタスチップと呼び，将来は家具や家電製品，壁や床などあらゆるものに埋め込むことで，人々の日常生活をサポートするさまざまなサービスを実現することを想定している．

ユビキタスチップの動作はイベント駆動型のルールで記述し，ルールの入替えによって動作を動的に変更できる．さらに，ユビキタスチップにはさまざまな入出力機器を接続でき，これらを組み合わせてアプリケーションを構築する．日常生活と深く関連したサービスはユーザ自身が生活パターンに合わせてカスタマイズすることが必要であり，そのためにはプログラマだけでなく一般のユーザでも直観的にアプリケーションの開発やカスタマイズが可能なアプリケーション開発環境が求められている．そこで本稿では，ユビキタスチップのためのアプリケーション開発環境を提案する．提案する開発環境は，複数のユビキタスチップの動作をシミュレートでき，ユーザのルール作成や実機への書き込みを支援する．さらに，開発環境上の仮想的なユビキタスチップと実際のユビキタスチップを連携させる機能をもつ．

以下，2 章ではユビキタスチップの概要について述べる．次に，3 章ではアプリケーション開発環境の設計について述べ，4 章ではプロトタイプの実装について述べる．5 章で提案する開発環境に対する考察と関連研究について述べ，最後に 6 章でまとめる．

2　ユビキタスチップの概要

図 1 に示すように，ユビキタスチップは本体であるコア部と周辺機器を接続するためのコネクタと電池からなるクロス部で構成される．ユビキタスチップ（図 1）は 5 個のデジタル入力端子，1 個のアナログ入力端子，

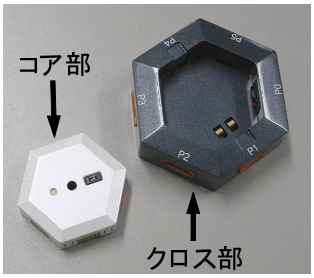

図 1: ユビキタスチップ

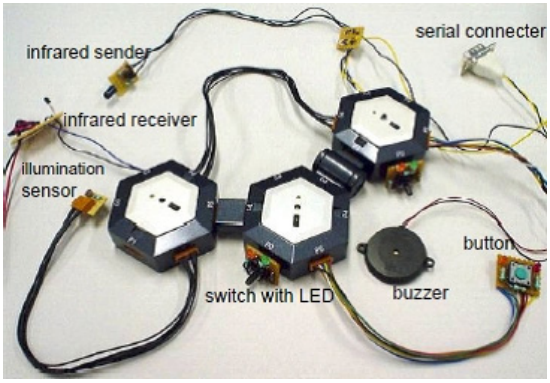

図 2: ユビキタスチップの接続例

12 個のデジタル出力端子，2 個のシリアル通信端子，1 個の汎用 LED をもつ．ユビキタスチップの各ポートには図 2 に示すように，センサ，スイッチ，LED，ブザー，モータといったさまざまな入出力機器を接続できる．これらの入出力機器を用い，複数のユビキタスチップを接続することによって，柔軟にアプリケーションを構築できる．

2.1 ECA ルール

ユビキタスチップの動作はイベント駆動型データベース分野で用いられている ECA ルールで記述する．ECA ルールは以下の 3 つの要素からなる．

イベント (E): 発生する事象
コンディション (C): 実行するための条件
アクション (A): 実行される動作

ユビキタスチップで用いられるイベント，コンディション，アクションの一覧を表 1, 2, 3 に示す．ECA ルールはユビキタスチップ内のメモリに格納される．

2.2 通信機能

ユビキタスチップはシリアル通信端子を用いて他のユビキタスチップと通信する．通信に関するアクションとしては，特定の ID をもつメッセージを送信する SEND_MESSAGE アクション，ルールで指定された数値やアナログ端子の入力電圧を 1 バイトのデータとして送信する SEND_DATA アクション，他のユビキタスチップのルールを制御するためのコマンドを送信する SEND_COMMAND アクションがある．

表 1: イベント一覧

名前	内容
TIMER_EXPIRE	タイマの発火
RECEIVE_MESSAGE	メッセージの受信
RECEIVE_DATA	1 バイトのデータの受信
NONE	イベントなし

表 2: コンディション一覧

名前	内容
INPUT	デジタル入力端子のオン/オフ状態
ANALOG_INPUT	アナログ入力端子の値の範囲
INPUT_STATE	状態変数の値
TIMER_ID	発火するタイマの ID
MESSAGE_ID	受信するメッセージの ID
DATA_RANGE	受信するデータの範囲

表 3: アクション一覧

名前	内容
OUTPUT	出力端子のオン／オフ設定
OUTPUT_STATE	状態変数の値設定
TIMER	タイマの設定／削除
SEND_MESSAGE	メッセージの送信
SEND_DATA	1 バイトデータの送信
SEND_COMMAND	コマンドの送信
HW_CONTROL	ハードウェア制御

表 4: コマンド一覧

名前	内容
ADD_ECA	ECA ルールの追加
DELETE_ECA	ECA ルールの削除
ENABLE_ECA	ECA ルールの有効化
DISABLE_ECA	ECA ルールの無効化
DEMAND_DATA	EEPROM 内のデータの送信要求
REPLY_DATA	EEPROM 内のデータの送信 (DEMAND_DATA に対する応答)

表 4 に SEND_COMMAND アクションで送信されるコマンドの一覧を示す．DEMAND_DATA コマンドはユビキタスチップのメモリの特定のアドレスのデータを要求する．ユビキタスチップが DEMAND_DATA コマンドを受信すると，要求されたデータを REPLY_DATA コマンドとして送信する．

3 アプリケーション開発環境の設計

3.1 要件

本研究では，ユビキタスチップが家具，家電，壁，床といったあらゆるものに埋め込まれ，互いに連携しあって人々の日常生活をサポートする環境を想定する．これらのサービスはユーザの日常生活に密着しているため，ユーザ自身がサービスをカスタマイズすることが必要である．例えば，以下のような状況を想定している．

- センサとユビキタスチップが埋め込まれた新しい家具を買い，既存のアプリケーションに新しい家具を組み込む．

- 部屋の模様替えを行う際に新しい家具の配置に合わせてアプリケーションを修正する．
- ユーザの生活パターンが変化に合わせてアプリケーションをカスタマイズする．

このような環境では，ユーザが稼働中のアプリケーションをカスタマイズする必要があるため，開発環境は既に設置されているユビキタスチップのルールを収集，修正したり，実際の入出力機器を用いて動作確認を行う機能が必要となる．

3.2 設計

前節で述べた要件を満たすため，提案するアプリケーション開発環境は以下の 3 つの機能を備える．

- 仮想ユビキタスチップによるシミュレーション
- 実ユビキタスチップのルールの読み書き
- 仮想ユビキタスチップと実ユビキタスチップの連携

3.2.1 仮想ユビキタスチップによるシミュレーション

ユビキタスチップを用いたアプリケーションでは，複数のユビキタスチップが連携することによって複雑な動作を実現する．そのため，ユーザが現在のルールの実行状態を把握したり，複数のユビキタスチップの連携を考慮しながらアプリケーションを構築するのは困難な作業である．そこで，提案するアプリケーション開発環境では実際のユビキタスチップ（実ユビキタスチップ）と同様に ECA ルールを処理する仮想的なユビキタスチップ（仮想ユビキタスチップ）を用いて複数のユビキタスチップの動作をシミュレートする．提案する開発環境は，仮想ユビキタスチップに関する以下の特徴をもつ．

- 実ユビキタスチップと同様に六角形として描画され，その周囲に入出力端子，シリアル通信端子，汎用 LED が描画される．
- シミュレーション環境上には複数のユビキタスチップを設置／削除できる．また，仮想ユビキタスチップ間の入出力端子，シリアル通信端子間は自由に接続／切断できる．
- 入出力端子のオン／オフの状態は円の色によって常に表示する．
- ユーザはアプリケーションの動作中であっても状態変数の値や格納されているルールを確認できる．
- ユーザは専門知識がなくても ECA ルールを作成できる．また，ユーザは ECA ルールエディタで簡単にルールを作成できる．さらに，ユーザは格納されている ECA ルールを自然言語に近い形式で確認できる．

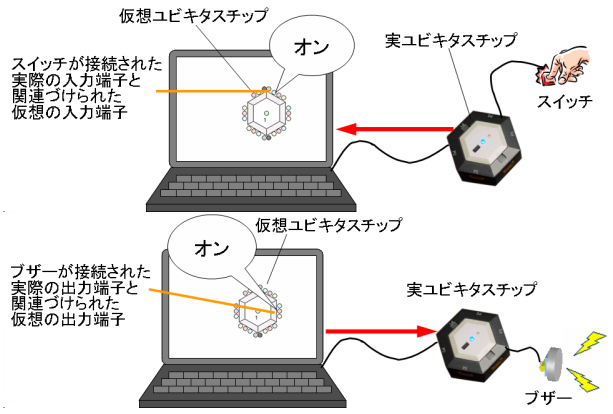

図 3: 仮想ユビキタスチップと実ユビキタスチップの連携

3.2.2 実ユビキタスチップのルールの読み書き

ユーザが既存のアプリケーションをカスタマイズする場合，環境内に存在する実ユビキタスチップから，格納されているルールを開発環境上に読み込む必要がある．そのため，提案する開発環境は実ユビキタスチップのルールを仮想ユビキタスチップに読み込む機能をもつ．

また，開発したアプリケーションを実ユビキタスチップに書き込む機能をもつ．

3.2.3 仮想ユビキタスチップと実ユビキタスチップの連携

提案するアプリケーション開発環境は仮想ユビキタスチップと実ユビキタスチップを連携させてアプリケーションを開発する機能をもつ．この機能の利用形態としては以下の 2 つの状況が考えられる．

ケース 1 環境内に配置されている実ユビキタスチップを用いてのアプリケーションを開発する．

ケース 2 開発環境上でアプリケーションが正しく動作することを確認した後，実際に使用するユビキタスチップと入出力機器を用いて動作確認を行う．

連携機能は，PC に接続した実ユビキタスチップの状態を仮想ユビキタスチップとしてシミュレータ上で管理し，実ユビキタスチップと仮想ユビキタスチップの動作を一致させることで実現する．例えば，図 3 に示すように，実ユビキタスチップに接続したスイッチを押すと，対応する仮想ユビキタスチップの入力端子をオンにする．同様に，仮想ユビキタスチップの出力端子がオンになったときには，対応する実ユビキタスチップの出力端子をオンにする．

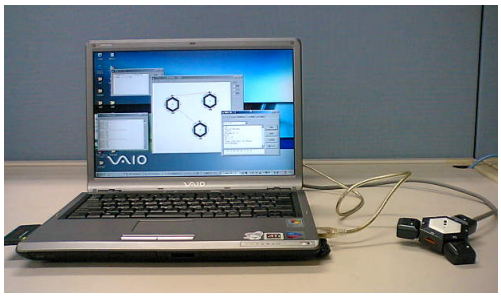

図 4: PC とユビキタスチップの接続例

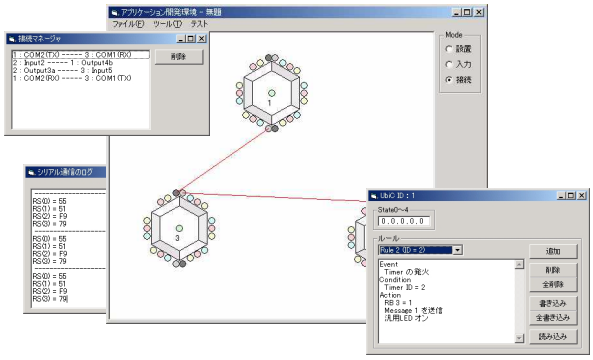

図 5: アプリケーション開発環境の画面

4 実装

提案するアプリケーション開発環境のプロトタイプを実装した．図 4 は本プロトタイプ用いて開発を行っている様子である．以降，実装の詳細について述べる．

4.1 仮想ユビキタスチップによるシミュレーション

図 5 にプロトタイプの画面を示す．中央のウィンドウ（シミュレーション領域）に示すように，表示されている仮想ユビキタスチップは六角形で表示され，周囲には入出力端子，シリアル通信端子，汎用 LED が円として描かれている．ユーザは仮想ユビキタスチップに対して以下の操作を行える．

- シミュレーション領域に複数のユビキタスチップを設置する．
- 入力端子のオン／オフを切り替える．
- 入出力端子やシリアル通信端子を接続する．
- 状態変数の値や格納されているルールを確認する．
- 図 6 に示す ECA ルールエディタを用いて ECA ルールを作成する．

ユーザはほとんどの操作をマウス操作で行える．

4.2 実ユビキタスチップのルールの読み書き

開発環境は実ユビキタスチップとコマンドの送受信を行うことによってルールを読み書きする．実ユビキタスチップからルールを読み込む際は，実ユビキタスチップに DEMAND_DATA コマンドを送信し，実ユビキタスチップが返信する REPLY_DATA コマンドを受信して実ユビキタスチップのメモリ内のデータを取得する．この操作を繰り返すことでメモリ内の全てのデータを取得する．次に，読み込んだデータを解析して仮想ユビキタスチップに自動的にルールを追加する．

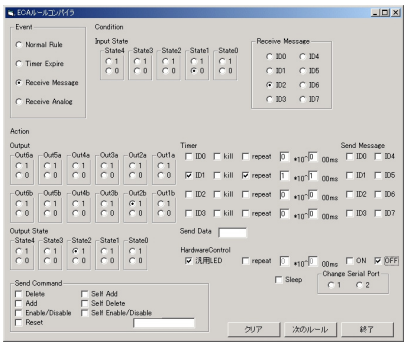

図 6: ECA ルールエディタ

実ユビキタスチップにルールを書き込む際は，ADD_ECA コマンドを用いて選択した仮想ユビキタスチップに格納されているルールを実ユビキタスチップに送信する．

4.3 仮想ユビキタスチップと実ユビキタスチップの連携

3.2 節で述べたように，連携機能の利用形態としては，ユーザが既にルールが書き込まれたユビキタスチップを用いてアプリケーションを開発するケース 1 と，ユーザが実際の入出力機器を用いて動作確認を行うケース 2 に分類される．

ケース 1 の場合，ユーザはまず実ユビキタスチップを PC に接続し，新しい仮想ユビキタスチップをシミュレーション領域に設置する．その後，連携は以下の手順で実施される．

Step 1 開発環境は実ユビキタスチップのルールを読み込み，ユーザが設置した仮想ユビキタスチップに格納する．

Step 2 開発環境は DELETE_ECA コマンドを送信し，実ユビキタスチップのルールを全て消去する．これは，後で述べる制御用ルールとの競合を避けるためである．

Step 3 開発環境は実ユビキタスチップを仮想ユビキタスチップと同様に振る舞わせるための制御用ルールを実ユビキタスチップに書き込む．

ケース 2 の場合，連携は上記の手順の Step 2 と 3 を行うことで実現する．

表 5 に制御用ルールの生成規則を示す．実ユビキタスチップの入力が変化すると，実ユビキタスチップは

表 5: 制御用ルールの生成規則

もとのルール	制御用ルール
E: (Any event) C: I(i)=0 (i=1-5) A: (Any action)	E: NONE C: I(i)=0, S(i-1)=1 A: S(i-1)=0, SEND_DATA(2i-1)
E: (Any event) C: I(i)=1 (i=1-5) A: (Any action)	E: NONE C: I(i)=1, S(i-1)=0 A: S(i-1)=1, SEND_DATA(2i)
E: (Any event) C: (Any condition) A: O(i)=0 (i=1-12)	E: RECEIVE_DATA C: RECEIVED_DATA=2i-1 A: O(i)=0
E: (Any event) C: (Any condition) A: O(i)=1 (i=1-12)	E: RECEIVE_DATA C: RECEIVED_DATA=2i A: O(i)=1
E: (Any event) C: (Any condition) A: HW_CONTROL (M_LED ON)	E: RECEIVE_DATA C: RECEIVED_DATA=25 A: HW_CONTROL (M_LED ON)
E: (Any event) C: (Any condition) A: HW_CONTROL (M_LED OFF)	E: RECEIVE_DATA C: RECEIVED_DATA=26 A: HW_CONTROL (M_LED OFF)

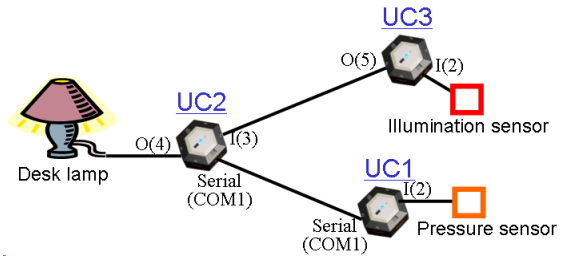

図 7: サンプルアプリケーションの構成

SEND_DATA アクションを実行し，1 バイトデータを開発環境に送信する．開発環境はデータを受信すると，実ユビキタスチップと関連づけられた仮想ユビキタスチップの入力端子を変化させる．

4.4 利用例

本節では実装したアプリケーション開発環境の利用例を示す．ここでは，「ユーザが椅子に座ったとき，部屋が暗ければ電気スタンドが点灯する」というアプリケーションを考える．このアプリケーションでは 3 つのユビキタスチップを用いる．それぞれ UC1, UC2, UC3 とし，3 つのユビキタスチップの接続関係を図 7 に示す．UC1 は椅子に取り付けられており，UC1 にはユーザの着座を感知するための圧力センサが接続されている．UC2 は机に取り付けられており，UC2 には制御対象である電気スタンドが接続されている．UC3 は壁に取り付けられており，UC3 には照度センサが接続されている．

ユーザは以下の手順でアプリケーションを開発する．

Step 1 シミュレーション領域に 3 つの仮想ユビキタスチップを設置する．

表 6: サンプルアプリケーション用ルール

UC1 用のルール（2 個）	
E: NONE C: I(2)=1, S(0)=0 A: S(0)=1, SEND_MESSAGE(M0)	E: NONE C: I(2)=0, S(0)=1 A: S(0)=0, SEND_MESSAGE(M1)

UC2 用のルール（5 個）		
E: RECEIVE_MESSAGE C: MESSAGE_ID=0 A: S(0)=1		E: RECEIVE_MESSAGE C: MESSAGE_ID=1 A: S(0)=0, O(4)=0
E: NONE C: I(3)=1 A: S(1)=0, O(4)=0	E: NONE C: I(3)=0 A: S(1)=1	E: NONE C: S(0)=1, S(1)=1 A: O(4)=1

UC3 用のルール（2 個）	
E: NONE C: I(2)=1 A: O(5)=1	E: NONE C: I(2)=0 A: O(5)=0

Step 2 3 つの仮想ユビキタスチップの端子を接続する．

Step 3 ECA ルールエディタを用いて表 6 に示すルールを 3 つの仮想ユビキタスチップに追加する．

Step 4 仮想ユビキタスチップの入力端子のオン／オフを切り替えて動作を確認する．バグがあればルールを修正する．

Step 5 実際の照度センサを用いて動作確認をする場合は，実ユビキタスチップを PC と接続し，UC3 を表す仮想ユビキタスチップと関連づける．この際，表 7 に示す制御用ルールが自動的に実ユビキタスチップに追加される．ユーザは部屋の明るさを変化させて動作確認を行う．

Step 6 アプリケーションが思うように動作することを確認したら，ユーザは実際にアプリケーションで用いる実ユビキタスチップにルールを書き込み，実際の家具や壁に取り付ける．

5 考察

5.1 大規模なアプリケーション開発への対応

現在のアプリケーション開発環境では，シミュレーション領域が限られているため，7, 8 個程度までのユビキタスチップしか同時に設置できず，多数のユビキタスチップを用いるアプリケーションの開発は困難である．このため，シミュレーション領域の拡大・縮小表示を行えるようにしたり，意味のある単位で複数のユビキタスチップをグループ化できるようにする予定である．

5.2 入出力機器のシミュレーション

現在の実装では，円の色によって入出力状態を表示しているが，実際の入出力機器の動作をイメージしにくいという問題がある．そのため，仮想ユビキタスチッ

表 7: サンプルアプリケーションの制御用ルール

UC3 用の制御用ルール（2 個）	
E: NONE	E: NONE
C: I(2)=0, S(2)=1	C: I(2)=1, S(2)=0
A: S(2)=0, SEND_DATA(3)	A: S(2)=1, SEND_DATA(4)

プと同様に仮想のセンサやアクチュエータといった仮想入出力機器を表示し，入出力機器の動作をシミュレートする機能を実現することで，より直感的なアプリケーション開発が可能になる．

5.3 無線通信のシミュレーション

現在の実装では，2 つのシリアル通信ポートを結びつけることで通信相手を指定しているため，有線の接続はシミュレーションできるが，無線通信をシミュレーションできない．実際にはこれまでに赤外線，微弱無線，bluetooth など，さまざまな無線通信モジュールの試作を行っているため，今後は無線通信にも対応する予定である．

5.4 関連研究

Smart-It[2], MOTE[3], U-Cube[5] はいずれもセンサネットワークやユビキタスコンピューティング環境を構築するための小型デバイスである．これらのデバイスの動作は C 言語や NesC 言語といったプログラミング言語で記述する．いずれのデバイスにおいてもアプリケーション開発用のツールやライブラリが提供されているが，プログラミング言語に対する知識が必要なため，一般のユーザがアプリケーションの開発やカスタマイズを行うことは困難である．

特定のハードウェア専用の一般ユーザ向けのアプリケーションの開発環境としては，MINDSTORMS[6] や ROBOT WORKS[1] がある．これらの開発環境は，条件や動作を記述したブロックをつなげることで簡単にプログラミングができ，動作記述が容易である．しかし，これらの開発環境にはハードウェアの動作をシミュレートする機能はなく，複数のデバイスの連携も考慮されていない．

MPLAB[7] はユビキタスチップでも用いられているマイクロプロセッサである PIC のための開発環境である．MPLAB は PIC の動作をシミュレートする機能をもち変数の値を確認しながら動作を確認できるが，複数の PIC の動作をシミュレートすることや入出力端子の状態を可視化することはできない．

6 まとめ

本稿ではユビキタスチップのためのアプリケーション開発環境を提案し，その設計と実装について述べた．提案する開発環境は複数のユビキタスチップの動作をシミュレートでき，さらに，実際のユビキタスチップのルールを読み書きし，実ユビキタスチップと連携してアプリケーション開発を行える．

今後は大規模なアプリケーション開発への対応，入出力機器のシミュレーション，無線通信への対応など，アプリケーション開発環境の機能をさらに拡張する予定である．また，アプリケーション開発環境の実運用と評価を行う予定である．

謝辞

本研究の一部は，文部科学省 21 世紀 COE プログラム「ネットワーク共生環境を築く情報技術の創出」，および文部科学省基盤研究 (A)(17200006) の研究助成によるものである．ここに記して謝意を表す．

参考文献

[1] BANDAI: “ROBOT WORKS,” http://www.roboken.channel.or.jp/borg/.

[2] M. Beigl and H. Gellersen: “Smart-Its: An Embedded Platform for Smart Objects,” *Smart Objects Conference (sOc)* (May. 2003).

[3] Crossbow Technology Inc.: “MICA,” http://www.xbow.com/products/Wireless_Sensor_Networks.htm.

[4] J. Kahn, R. Katz, and K. Pister: “Mobile Networking for Smart Dust,” in *Proc. ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom99)*, pp. 271–278 (Aug. 1999).

[5] Y. Kawahara, M. Minami, H. Morikawa, and T. Aoyama: “Design and Implementation of a Sensor Network Node for Ubiquitous Computing Environment,” in *Proc. VTC2003-Fall* (Oct. 2003).

[6] LEGO: “MINDSTORMS,” http://mindstorms.lego.com/japan/products/.

[7] Microchip Technology Inc.: “MPLAB,” http://www.microchip.com/1010/index.htm.

[8] T. Terada, M. Tsukamoto, K. Hayakawa, T. Yoshihisa, Y. Kishino, S. Nishio, and A. Kashitani: “Ubiquitous Chip: a Rule-based I/O Control Device for Ubiquitous Computing,” in *Proc. Int’l Conf. on Pervasive Computing (Pervasive 2004)*, pp.238–253 (Apr. 2004).

[9] M. Weiser: “The Computer for the Twenty-first Century,” *Scientific American*, Vol. 265, No. 3, pp. 94–104 (Sept. 1991).