Vol.2012-MBL-64 No.1
Vol.2012-ITS-51 No.1
2012/11/15

LISPをベースとするユビキタスコンピューティングのためのプログラム処理系の実現について

猪谷 直人[1]　藤田 直生[1]　佐野 渉二[1]　寺田 努[1,2]　塚本 昌彦[1]

概要：ユビキタス環境では，ユビキタスコンピュータ群全体で行えるタスクが重要となる．本研究では，関数型プログラミング言語であるLISPを用いたユビキタスコンピュータ群の制御手法を提案する．LISPは，実世界の情報を記号として統一的に処理を行うことができ，シンプルな動作メカニズムで複雑なプログラミングが行えることから，小リソースであるユビキタスコンピュータ上での利用に適している．本稿では，LISPの基本関数に加え，隣接デバイスにプログラムを送信，評価させる独自の関数を実装することにより，ローカルな通信に基づいてユビキタスコンピュータ群全体の制御をする手法を考案した．

A System Implementation for Processing a LISP-based Language in Ubiquitous Computing Environments

Itani Naoto[1]　Fujita Naotaka[1]　Sano Shoji[1]　Terada Tsutomu[1,2]　Tsukamoto Masahiko[1]

Abstract: In ubiquitous computing environments, there is a strong demand for controlling a lot of small computers embedded in the environments as a whole based on their topology. In this paper, we propose a method using a LISP-based system for controlling grid-connected ubiquitous computers. LISP can treat real-world information as a symbol, and can programming complex mechanism in a simple run. These features are suitable to ubiquitous computing environment. By implementing LISP processor and original LISP-based functions, we can handle all ubiquitous computers only by local communication among neighboring computers.

1. はじめに

近年，情報機器の小型化に伴い，環境内のいたるところに埋め込まれた多数のコンピュータを利用するユビキタスコンピューティング環境に注目が集まっている．ユビキタスコンピューティング環境では，一般に多くの情報を収集，処理，提示するために小型なユビキタスコンピュータが用いられる．また，このような環境においては，ユビキタスコンピュータ単体で行う処理よりもユビキタスコンピュータ群全体で何ができるかが重要となる．そのため，ユビキタスコンピュータ群全体を制御するためのメカニズムや，多数のコンピュータ同士がある目的のために協調動作することが求められる．現在提供されているユビキタスコンピュータでは，プログラム記述や処理において実世界との関連付けが行われておらず，プログラマーがその関連付けを意識する必要がある．そのため，開発者がプログラミングに熟練した者でなければ，小型コンピュータを扱いにくいといった問題がある．また，個々のコンピュータに分散，連携した動作をさせるためには，複雑かつ高度なプログラム記述が必要である．

これに対して，我々の研究グループではこれまでに関数型プログラミング言語LISPを用いたユビキタスシステムと処理系の実装を行ってきた．LISPは実世界の多様な対象を記号で表象し，雑多な対象からなる集合をリストという形式で扱えるようにする言語である．これらの特徴は，

[1] 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University
[2] 科学技術振興機構さきがけ
PRESTO, Japan Science and Technology Agency

ユビキタス環境に非常に適合しており，ユビキタス環境において重要な，コンピュータを意識せずに使用できる環境の構築が容易になるものと考えられる [1]．本稿では，LISP を用いて格子状に接続されたユビキタスコンピュータ群の分散プログラミングを行うための手法を提案する．格子状ユビキタスネットワークを対象に制御プラットフォームを構築することで，トポロジ情報をうまく利用しながら，マクロなプログラミングを行うことができる．また，LISP を用いた制御プラットフォームを構築することにより，既知の言語を用いながらコンピュータ群全体の制御や，複数コンピュータにまたがる複雑な処理が容易に行える．

2. 関連研究

ユビキタスコンピューティング環境を実現するための小型コンピュータについては，これまでにも数多くの研究がされている．藤田らは，記号処理言語 LISP の処理系を備えた小型ユビキタスコンピュータ「ULIP」を提案し，LISP 言語を使用するプラットフォームを設計，実装した [1]．小型性や省電力性，通信機能などユビキタスコンピュータに必要な機能を備えており，単一ユビキタスコンピュータに対し LISP を用いたプログラミングを可能としているが，ユビキタスネットワークを構築する上で任意のユビキタスデバイスにアクセスするマルチホップを通信を行うためのプロトコルが組み込まれていない．寺田らは，ルールに基づきユビキタスコンピュータの入出力を制御するユビキタスプラットフォームを提案し，設計，実装を行った [2]．センサまたは単一の制御アーキテクチャに基づいて，アクチュエータを使用してルールの追加，変更をすることにより，さまざまなユビキタス環境を柔軟に構築できる．スマートダストプロジェクト [4] により開発された MOTE[3] は，自発的なアドホックネットワークの形成機能や，マルチホップ機能を備えた小型デバイスであり，MOTE で稼働するアプリケーションは，nesC と呼ばれる C 言語の拡張言語を用いて記述される．しかし，これらのデバイスは，複数のデバイスにまたがる処理を記述する場合，個々のデバイス同士で整合の取れた制御プログラムを各デバイスに対して作成する必要があり，複雑かつ高度なプログラミングが求められる．

多数のユビキタスコンピュータを制御するための研究として，複数のユビキタスコンピュータに対する制御プログラムを一つのプログラムで記述するマクロプログラミングがある．Kairos[5] や Regement[6] では，複数のコンピュータに及ぶ処理やコンピュータのトポロジを用いた処理を一つのプログラムで記述可能にするシステムである．RuleCaster [7] は，各コンピュータの動作をルール形式で記述する．複数コンピュータにまたがる処理も 1 つのルールで定義でき，複数ルールを用いることで一度に多くの処理を行える．マクロプログラミングは，一般にユビキタスコンピュータネットワーク全体の振る舞いを記述した言語を，処理系によって個々のノードに配備するプログラムに変換する．しかし，これらシステムは個々のデバイスに対するプログラミングの延長であり，プログラムを変更する度に各デバイスへプログラムの再配備が必要であることから，状況に応じてシームレスに処理を変更することは困難である．

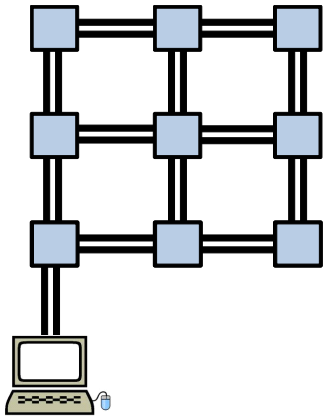

図 1 格子状ユビキタスネットワーク例

3. 研究内容

3.1 ユビキタス環境と LISP

LISP は, John McCarthy により提唱された 関数型プログラミング言語の一種である．これまで LISP は処理能力やメモリ空間が多く必要であるとされ，発表された当時は LISP 専用のコンピュータの開発なども行われた．

LISP は，処理を行う上で情報をすべて記号化してあるかうことができる特徴をもっている．ユビキタスコンピューティング環境において外部の環境情報を記号化し抽象化された情報を処理することにより，数値をメタ情報と共に処理することができ，実世界との融合を果たすことが容易となると考えられる．さらに，シンプルな動作メカニズムで複雑なプログラミングが可能である．ラピッドプロトタイピングやアドホックプログラミングに適しており，LISP はユビキタス環境を構築するための言語として適していると考えられる．

3.2 想定環境

本研究では，図 1 のように有線で接続されたユビキタスコンピュータ群の制御を想定する．図 1 では 9 個の四角形がノードを表し，マイコンやセンサ，LED などを搭載したユビキタスコンピュータを意味する．個々のユビキタスコンピュータがどのような接続関係にあり，ユビキタスコンピュータ群全体のなかでどのような位置にあるかを把握し，それらトポロジ情報を利用したプログラミングを行うことが重要となる．格子状ネットワークにおいて，ホップ

数をカウントすれば座標値を得ることができるなど，トポロジ情報の把握が容易になり，それらを利用したプログラミングが行いやすい．

3.3 システム要件

ユビキタスコンピュータ群を制御するために求められる要件を検討する．ユビキタス環境においては，常時安定した電源が供給される環境下よりは，バッテリなど限られた資源での駆動が必要となる．ユビキタスコンピュータは環境に埋め込まれるため，長期間運用するためには小電力で動作することが求められる．また，ユビキタスデバイスは様々な場面で利用され組み込まれることから，容易に環境に組み込むことができるように小型である必要がある．また，ユビキタス環境においては，ユビキタスコンピュータ単体で行う処理よりも，ユビキタスコンピュータ群として全体でどのような制御が行えるかが重要となる．そのため，他のユビキタスコンピュータとの通信機能を備えている必要がある．

3.3.1 プログラム環境

ユビキタス環境におけるプログラミングでは，外部である実世界の情報をどうのようにとらえコンピュータに取り込み処理をするかが重要である．従来のプログラミングでは，複数の機器を扱う場合や外部との接続が変わった場合などに，プログラムが複雑化していく問題がある．また，プログラマーや利用者が変わった場合には，シンボルが統一的に扱われるとは限らず，コンピュータ内で処理される数字が何を表しているかが分からなくなる可能性がある．

ユビキタスコンピュータが複数連携していく中で，処理している情報が実世界とどのような関係を持っているかが容易に理解できるようなプログラム環境が必要となる．

3.3.2 動的な動作変更

ユビキタスコンピューティング環境においては，ユーザからの要求の変化，自然現象によるトラブルなど，様々な状況の変化が想定される．そのため，状況の変化に柔軟に対応するために動的な動作変更が可能であることが求められる．

3.3.3 ユビキタスコンピュータ群のスケールの変化に対応

システムを利用しながら機能や使用スペースを拡張する場合を考えた際，環境内に新たなユビキタスコンピュータを追加しなければならないことが想定される．そのため，ユビキタスコンピュータを自由に増やすことができ，コンピュータが増加した際にもプログラムの変更は最小限に抑えることができる必要がある．

3.3.4 連携動作

ユビキタスコンピュータは小型であるため，個々のコンピュータは簡単な動作しかできない．そのため，ユビキタスコンピュータ同士が連携し，複雑な動作が可能になることが望ましい．

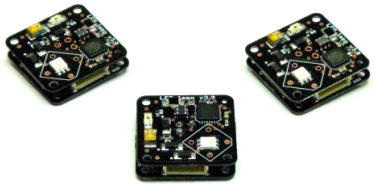

図 2 使用デバイス

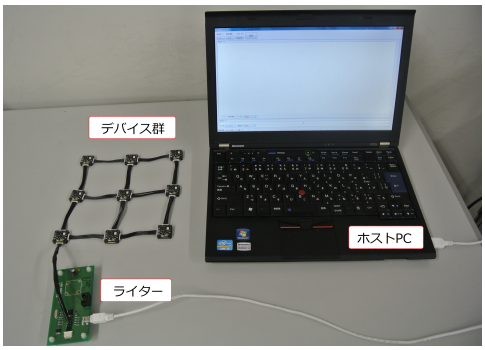

図 3 システム構成例

3.4 提案システム

LISP は，関数型プログラミング言語の一種であり，記号処理言語として集合型データ処理に特化した最小限の処理系を想定するものである．LISP を用いることで，異種のセンサデータ等も記号として統一的に扱えたり，実世界の情報を記号として表象して処理が行える．データとして数値しか取り扱えない言語と比較して，より自然にユビキタス環境における分散したリソースやデータに関わる問題を表現できるものと考えた．

本研究では，格子状に接続されたユビキタスコンピュータ群において，各デバイスに対し LISP の処理系を実装することで，各デバイスの持っているデータを統合的に扱う手法を提案する．格子状ユビキタスネットワークを想定することで，トポロジ情報を利用し，ローカルプログラミングの組み合わせによるマクロなプログラミングを実現できる．制御対象として，図 2 に示す筆者らの研究グループで開発したユビキタスコンピュータを使用する．図 3 にシステムの構成図を示す．

3.4.1 初期設定関数

初期設定のため，ユビキタスコンピュータに ID を指定して記憶させる関数 idset，各ポートに接続させるＩＤを設定する関数 portset を作成した．

それぞれ書式は，`(idset 'arg)`，`(portset 'arg1 'arg2 'arg3 'arg4)` とする．

関数 idset では，関数の引数に指定した値をデバイスに ID として記憶させる．関数 portset では，引数である `arg1`，`arg2`，`arg3`，`arg4` に指定した値をデバイスの各４ポートにポート ID として記憶させる．

表 1 実装した関数一覧

コマンド	書式	処理内容
+, -, *, /	(+ 'arg1 'arg2 'arg3...)	四則演算を行う
car	(car 'list)	引数リストの第一要素を返却する
cdr	(cdr 'list)	引数リストの第一要素を除いた残りのリストを返却する
cons	(cons 'arg1 'arg2)	第一引数のシンボルを第二引数のリストの先頭要素として追加する
append	(append 'arg1 'arg2)	第一引数と第二引数のリストの要素をつなげたリストを返却
list	(list 'arg1 'arg2 'arg3...)	引数のシンボルを要素とするリストを返却
set	(set 'arg1 'arg2)	第一引数で指定した変数に第二引数のシンボルを代入
call	(call 'arg1 'arg2)	隣接デバイスにシンボルを送信して実行
light	(light 'arg)	LED を指定の色に発光させる
idset	(idset 'arg)	デバイスの ID を指定した値に設定
portset	(portset 'arg1 'arg2 'arg3 'arg4)	デバイスのポート ID を指定した値に設定

3.4.2 LISP 関数

小型のユビキタスコンピュータでは，処理能力に限界があるため，初期関数や取扱い数値には制限がある．そのため，初期状態でインタープリタが処理できる関数は LISP として最小限のものをサポートする．本研究では，記号処理言語 LISP を用いてユビキタスコンピューティング環境を構築する上で，重要度の高いと思われる LISP の基本関数や，ユビキタス環境構築のために必要となる独自の関数を実装していく．LISP をベースとするプログラム処理系において現在実装している関数を表 1 に示す．

set

書式は (set 'arg1 'arg2)．

変数にシンボルを代入する関数として，set を実装した．第一引数である arg1 に指定した変数に，第二引数である arg2 で指定したシンボルを代入する．小型のユビキタスコンピュータに LISP 処理系を実装する場合，処理能力に限界がある．そのため，本研究では，変数として指定できる文字列は 10 文字以下，シンボルとして変数に保存可能な文字列は 128 文字までとした．

3.5 独自関数

call

他デバイスのデータの参照，または連携動作を行うため，トポロジ情報に基づき，隣接するユビキタスコンピュータにシンボルを送信して実行する関数 call を作成した．送信先のユビキタスコンピュータでのシンボルの処理結果が自身のリストの評価結果となる．

書式は (call 'arg1 'arg2) とする．

第一引数である arg1 にはトポロジ情報に基づいてシンボルの送信先となる方向を指定する．図 4 に示すよう，対象デバイスの上方向へメッセージを送信する場合は N，下方向へは S，左方向へは W，右方向へは E と表し指定するものである．

第二引数である arg2 には，第一引数で指定した方向に送信するメッセージの内容を記述する．メッセージとして LISP 書式である S 式に基づくプログラムを記述することで，デバイスをまたいだ LISP でのユビキタスコンピュータ群の制御を可能にする．

送信先のユビキタスコンピュータでの評価結果を関数 call の評価結果とするため，関数 call を呼び出したユビキタスコンピュータは，送信先デバイスからの評価結果の返却を待機する必要がある．このとき，呼び出した関数以外のデータを受信し，評価結果としてしまう可能性が考えられる．そこで，関数 call でデータを送受信する際，処理上の ID を付与し，送信先ユビキタスコンピュータ以外からのデータはノイズとして排除する．

light

書式は (light 'arg)．

出力として，LED を指定の色に発光させる関数．赤，青，緑の三色を実装しており，それぞれ引数に r，g，b と記述することで色を指定する．

3.6 使用デバイス

制御対象のユビキタスコンピュータとしては，図 2 に示す筆者らの研究グループで開発した小型デバイスを用いる．サイズは 20mm 四方で，計算機には 8 ビットワンチップマイコンでり，8MB のフラッシュメモリを搭載している Atmel 社製 AVR を使用している．AVR マイコンは低消費電力であり，温度動作範囲が広いという特徴を持ち，ユビキタスコンピューティング環境での利用に適している．このデバイスはフルカラー LED，可視光センサ，赤外線センサが搭載されており，通信ケーブルにより隣接する 4 方向のユビキタスデバイスとシリアル通信が行える．また，通信線には隣接するノードのみと通信を行うローカル線がある．制御対象であるユビキタスコンピュータに加え，通信ケーブル，ユビキタスコンピュータの書き込み機（ライター），シリアル通信用アプリケーションは筆者らの研究グループで作成したものである．USB ケーブルとライターを

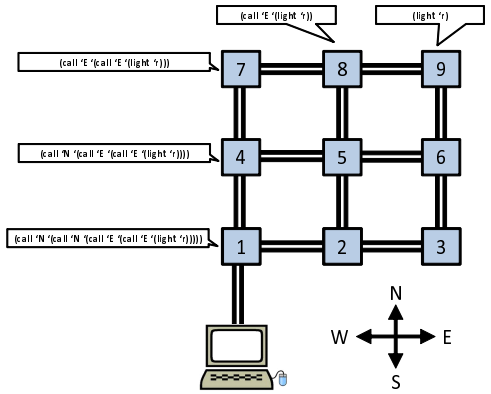

図 4 ユビキタスコンピュータ９を赤色に発光

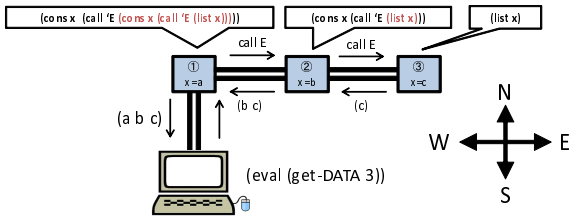

図 5 ユビキタスコンピュータのデータの収集１

接続し，ライターと格子状のユビキタスコンピュータ群を接続する．ローカル通信を用いる場合，通信が干渉しない範囲で多数のノードが同時に送受信できるため，個別の通信を各所で行う上ではスケーラビリティがあり，並列性をもつ．本研究での LISP を用いたユビキタスコンピュータの制御システムは，全てローカル線を用いての通信を行う．

3.7 ユビキタスコンピュータ群に対するプログラム例

ユビキタスコンピュータを発光させる

[例１]

```
(call 'N '(call 'N '(call 'E '(call 'E '(light 'r)))))
```

図 4 に示すユビキタスコンピュータ９の LED を発光させたい場合，上記の例のようにプログラムを作成し，ホストコンピュータに接続されたユビキタスコンピュータにプログラムを送信することで実現できる．ユビキタスコンピュータ１は受け取ったプログラムを評価し，N 方向のユビキタスコンピュータ４に，第二引数である (call 'N '(call 'E '(call 'E '(light 'r)))) を送信する．ユビキタスコンピュータ４は受け取った文字列を評価する．そこで更に call 関数によって N 方向のユビキタスコンピュータ７に (call 'E '(call 'E '(light 'r))) を送信し，評価させる．同様にプログラムの処理が続き，ユビキタスコンピュータ９は (light 'r) を評価し，自身の LED を赤色に発光させる．

ユビキタスコンピュータの持つデータの収集１

[例２]

```
(defun get-DATA (m)
  (cond ((> m 1) (list 'cons 'x (list 'call ''E
          (list 'quote (get-DATA (- m 1))))))
        (t '(list x))))
```

これは，ホスト PC に接続されたユビキタスコンピュータに，E 方向にあるｍ個先までのデバイスに保存されているデータを収集するためのプログラムを生成するプログラムである．

図 5 に示すユビキタスコンピュータ群に保存されているデータ a,b,c を，リストとしてプログラムの実行されたデバイスに収集することを考える．図 5 に示すよう各ユビキタスコンピュータの x という変数の中にそれぞれデータが保持されているものとする．

E 方向３個先までのデバイスに保存されているデータを収集するプログラムはホスト PC 上で，定義した関数 get-DATA を用い，(get-DATA 3) と実行することで生成できる．生成されるプログラムは次のようになる．

```
(cos x (call 'E '(cons x (call 'E '(list x)))))
```

プログラムの処理について説明する．まず，PC から接続されたユビキタスコンピュータがプログラムを受信すると，LISP の書式に従い，関数 cons の第二引数である (call 'E '(cons x (call 'E '(list x)))) というリストが評価される．関数 call により，(cons x (call 'E '(list x))) というリストがユビキタスコンピュータ２に送られる．このとき，ユビキタスコンピュータ１は関数 call の評価結果の返却があるまで処理を待機した状態となる．同様に，ユビキタスコンピュータ２はまず (call 'E '(list x)) というリストを処理，ユビキタスコンピュータ３に (list x) というリストを送信する．ユビキタスコンピュータ３は (list x) の処理結果である (c) を，ユビキタスコンピュータ２に返却する．ここで，ユビキタスコンピュータ２において，処理結果の返却待機状態にある関数 call の評価結果は (c) となる．さらに，ユビキタスコンピュータ２で関数 cons が評価され，関数 call の評価結果である (c) に自身の保持する変数ｘの値を加え，ユビキタスコンピュータ２での処理結果は (b c) となり，ユビキタスコンピュータ１に返却される．同様にユビキタスコンピュータ１上での関数 call の評価結果は (b c) となり，最終的な評価結果は自身の持つ変数の値を加えた (a b c) となる．このようにして，各ユビキタスコンピュータの保持するデータをリストとしてプログラムの実行されたデバイス上に収集することができる．

ユビキタスコンピュータの持つデータの収集２

[例３]

```
(defun gets-DATA-of-matrix (m n)
  (cond ((> n 0) (list 'cons (get-DATA m)
           (list 'call ''N (list 'quote
           (gets-DATA-of-matrix m (- n 1))))))
```

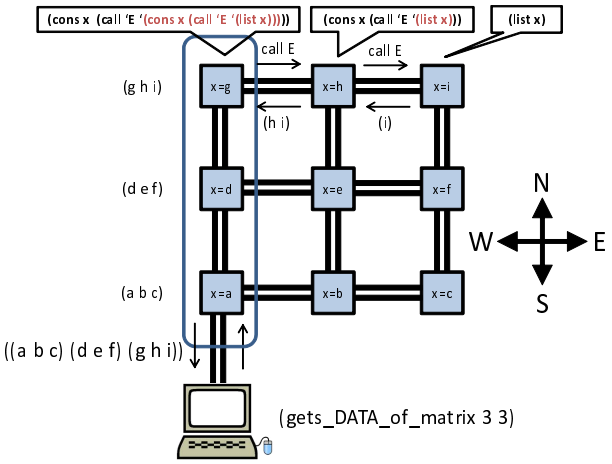

図 6 ユビキタスコンピュータの持つデータの収集 2

```
        (t nil)))
```

例 2 で述べたユビキタスコンピュータの保持するデータを収集するプログラムを，m×ｎの格子状に接続されたユビキタスコンピュータ群を対象に拡張したものである．このプログラムは例 2 で定義した関数 get-DATA を利用することで，任意のm×ｎに接続されたユビキタスコンピュータ群の保持するデータをリストとして収集する関数を生成することができる．

生成されたプログラムの処理としては，例 2 で述べた E 方向m個先までのデバイスのデータを収集するプログラムを，N 方向ｎ個先のデバイスまで送る．ホスト PC から見て N 方向のデバイスに収集された E 方向のデバイスデータは，要素数ｎのリストとしてプログラムの実行されたデバイスに収集される．

図 6 に示す，ユビキタスコンピュータ群のデバイスデータ a,b,...,i を取得するには，定義した関数 gets-DATA-of-matrix を用い，ホスト PC 上で (gets-DATA-of-matrix 3 3) を実行することによって生成されたプログラムを，PC に接続されたユビキタスコンピュータ上で実行すればよい．評価結果として，((a b c) (d e f) (g h i)) のように，各デバイスの保持するデータが収集される．

4. まとめ

本研究では，格子状に接続されたユビキタスコンピュータ群の制御を目的として，LISP を用いた制御手法を提案した．各ユビキタスコンピュータに LISP をベースとしたプログラム処理系を構築し，群制御のための基本となる LISP 関数や独自の関数を用いて，ユビキタスコンピュータ群を対象とした制御手法を考案した．格子状ネットワークを想定することで，トポロジ情報を利用したローカルプログラミングにより，コンピュータ群全体を制御するプラットフォームを構築した．LISP を用いることにより，実世界の情報を記号と組み合わせ，リストとして処理を行うことが可能となり，実世界の情報をさらに利用しやすくなる．さらに，古くから使われている既知の言語を用いながらユビキタスコンピュータ群を対象として，容易性を備えたプログラミングを行える．また，関数型プログラミング言語である LISP をユビキタスコンピュータ群に適用することにより，分散環境において再帰的処理を行うような複雑な処理も可能になると考えられる．

謝辞

本研究の一部は，文部科学省科学研究費補助金基盤研究 (A)(20240009, 23240010) によるものである．ここに記して謝意を表す．

参考文献

[1] 藤田直生，塚本昌彦：ユビキタス環境のための記号処理言語 LISP を用いた小型コンピュータの設計と実装，DICOMO 2006，pp. 897–900 (2006).

[2] T. Terada, M. Tsukamoto, K. Hayakawa, T. Yoshihisa, Y. Kishino, A. Kashitani, and S. Nishio: Ubiquitous Chip: a Rule-based I/O Control Device for Ubiquitous Computing, *Proceeding of the International Conference on Pervasive Computing (Pervasive 2004)*, pp. 238–253, 2004.

[3] B. Warneke, M. Last, B. Liebowitz, and K. Pister: Smart Dust: Communicating with a Cubic-Millimeter Computer, *Proceedings of the IEEE Computer Magazine*, Vol. 34, Issue 1, pp. 44-51, 2001.

[4] J. M. Kahn, R. H. Katz and K. S. J. Pister: “Next century challenges: Mobile networking for “smart dust””, *MOBICOM*, pp. 271-278, 1999.

[5] R. Gummadi, O. Gnawali, and R. Govindan: Macroprogramming Wireless Sensor Networks Using Kairos, *Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS2005)*, pp. 126-140, 2005.

[6] R. Newton, G. Morrisett, and M. Welsh: The Regiment Macroprogramming System, *Proceedings of 6th International Conference on Informetion Processing in Sensor Networks (IPSN2007)*, pp. 489-498, 2007.

[7] B. Urs and K. Gerd: A State-Based Programming Model and System for Wireless Sensor Networks, *Proceedings of the 3rd International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS2007)*, pp. 261–266, 2007.

[8] C. Zhang and T. Herman: Localization in Wireless Sensor Grids, Computers and Their Applications, pp. 388-393, 2006.

[9] R. ton and M. Welsh: Region Streams: Functional Macroprogramming for Sensor Networks, *Proceedings of the First International Workshop on Data Management for Sensor Networks (DMSN2004)*, pp. 78-87. 2004.