卒業論文 2006 年度 (平成 18 年度)

アプリケーション適応型センサノード配置条件抽象化機構

指導教員

慶應義塾大学環境情報学部

徳田　英幸

村井　純

楠本　博之

中村　修

高汐　一紀

湧川　隆次

慶應義塾大学環境情報学部

今枝 卓也

che@sfc.wide.ad.jp

Abstract of Bachelor's Thesis

アプリケーション適応型センサノード配置機構

近年，ユビキタスコンピューティング環境の実現を目指す研究が盛んになりつつある．ユビキタスコンピューティング環境では，あらゆる物がネットワークにつながり，仮想空間から現実空間の情報が参照できるようになる．現実空間の情報を仮想空間上で取得できるようになると，ユーザや現実空間上の物の状況に応じた様々なサービスやアプリケーションの実現が可能になる．

現実空間の情報を取得する方法として，複数のセンサノードを用いてセンサネットワークを構築する手法が注目されている．センサネットワークから得られる情報を利用することで，室内の温度や光量を自動調節するといった，現実空間の状態に応じたサービス提供が可能になる．

センサネットワークを構築するには，センサノードを環境内に配置する．しかし，センサノードの配置にはセンサノードに対する専門的な知識が必要になる．システム構築者は使用するセンサノードの特性に合わせて設置場所を選ぶ必要がある．そのため，システム構築者は事前にセンサノードに関する知識を入手しなくてはならず，専門家以外の人間がセンサネットワークを構築するのは困難である．

本研究の目的は，アプリケーション製作者がアプリケーションに最適なセンサノード配置方法を指定可能にする事である．

今枝卓也
慶應義塾大学 環境情報学部

Abstract of Bachelor's Thesis

A Framework for a Sensor Node Deployment Based on Application Feature

It is possible to acquire information of real space as digital data using sensor networks that consist sensor nodes. Sensor networks enable applications to provide service according to the state of real space. It is too difficult for persons who are not specialists to construct sensor networks. Besides, the condition of deployment of sensor nodes is different by the application. The purpose of this research is to enable to construct sensor networks and make the sensor networks adjust to the application without specialists. In this research we propose a ROASEN, a system that consists of a robot that deploys sensor nodes. Also the system provides a middleware that change behavior of the robot based on each application. Application developers can specify the condition that deploy sensor nodes by using the ROASEN middleware. The ROASEN robot satisfies this condition when the robot deploys sensor nodes.

Takuya Imaeda
Faculty of Environmental Information Keio University

目 次

図目次

表 目 次

第1章　序論

1.1 背景

近年，ユビキタスコンピューティング環境の実現を目指す研究が盛んになりつつある．ユビキタスコンピューティング環境では，あらゆる物がネットワークにつながり，仮想空間から現実空間の情報が参照できるようになる．現実空間の情報を仮想空間上で取得できるようになると，ユーザや現実空間上の物の状況に応じた様々なアプリケーションの実現が可能になる．

現実空間の情報を取得する方法として，従来は環境内にセンサを埋め込む手法が用いられてきた．最近では，センサノードを用いて環境情報を取得する手法が注目されている．センサノードとは，センサを搭載した小型の計算機で，無線通信によりネットワークを構築できる．比較的安価に入手でき，屋内だけでなく屋外の広い範囲にも適用できる．センサノードには，カリフォルニア大学を中心に開発が進んでいる Mote[1] や，カールスルーエ大学 TecO 研究所の uPart[2] 等がある．センサノードは光センサや温度センサ，加速度センサ等のセンサを備えている．センサノードから得られる情報を利用することで，室内の温度や光量を自動調節するといった，現実空間の状態に応じたアプリケーションが可能になる．

1.2 問題意識

センサノードを用いるアプリケーションを使うには，予めセンサノードを環境内に配置しておく必要がある．しかし，センサノードの配置にはセンサノードに関する専門的な知識が必要になる．例えば，センサノードの搭載センサや無線特性に合わせて設置場所を選ぶ必要がある．そのため，システム構築者は事前にセンサノードに関する知識を入手しなくてはならず，専門家以外の人間がセンサネットワークを構築するのは困難である．

また，利用したいアプリケーションによってセンサノードの配置条件を変える必要がある．アプリケーションの種類によって，センサノードの設置場所はランダムになったり特定の場所になったりする．例えば，植物を監視するアプリケーションの場合はセンサノードを植物の近くに置く必要がある．そのため，利用するアプリケーションに関して詳細な知識を持っていない人間は，センサネットワークを構築しても有効に活用できない．本稿では，アプリケーションのセンサノード配置条件に合わせて動的にセンサノード配置方法を変更する事を，アプリケーション適応的であるとする．また，単にアプリケーションと言った場合，センサノードを利用するアプリケーションを指す．

1.3 目的

本研究の目的は，アプリケーション開発者がより簡単にセンサノード配置条件を指定可能にする事である．アプリケーションにとって最適なセンサノード配置方法はアプリケーション開発者が熟知していると考えられる．アプリケーション開発者が予めアプリケーションに最適なセンサノード配置方法を指定しておく事で，専門家でなくてもセンサネッ

トワークの構築が可能になる．本研究では，アプリケーション開発者によるセンサノード配置方法指定の手段として ROASEN フレームワークを構築し，有効に機能するか検証する．センサネットワークを利用するアプリケーションは，ROASEN フレームワークを利用することでアプリケーションに適したセンサノードの配置を指定できる．

1.4 本論文の構成

本論文では，第 2 章でセンサノードを用いたユビキタスアプリケーションの概観について述べる．第 3 章では，本研究で構築する ROASEN フレームワークとその手法について述べる．そして，第 4 章で ROASEN フレームワークの実装を述べる．第 5 章では ROASEN フレームワーク上のアプリケーションとセンサノード配置機構について述べる．第 6 章で ROASEN フレームワークを評価し，第 7 章で本論文をまとめる．

第2章　センサノードを用いたユビキタスサービス

本章では，まず，センサノードについて述べ，次にセンサノードを利用して実現できるアプリケーションについて述べる．最後に，それらのアプリケーションの分類を行う．

2.1 センサノード

センサノードを配置する際にはセンサノードの特性を考慮する必要がある．本節では，まず，センサノードに搭載されるセンサ，無線について，特に MOTE と uPart に言及しながら述べる．次に，特殊なセンサノードとして，自律移動型センサノードについて述べる．

2.1.1 センサモジュール

センサノードに搭載されるセンサはセンサノードの種類によって異なる．Mote は端末基板とセンサ基板が別個に用意されており，様々なセンサを取り付け可能である．端末基板としては MICAz，MICA2，DOT があり，それぞれ搭載できるセンサ基板が違う．MICAz と MICA2 の主な違いは通信規格で，MICAz は 2.4GHz 帯で通信し，MICA2 は 315MHz 帯で通信する．DOT は MICAz や MICA2 に比べ小型で，大きな物体の取り付けに障害のある環境で使われる．主なセンサとしては光，音，温度，2 軸加速度に対するセンサがある．

uPart は MOTE に比べ小型なのが特徴であるが，センサが端末基板に取り付けられており，センサの種類は変更できない．搭載センサは光，温度，振動センサである．

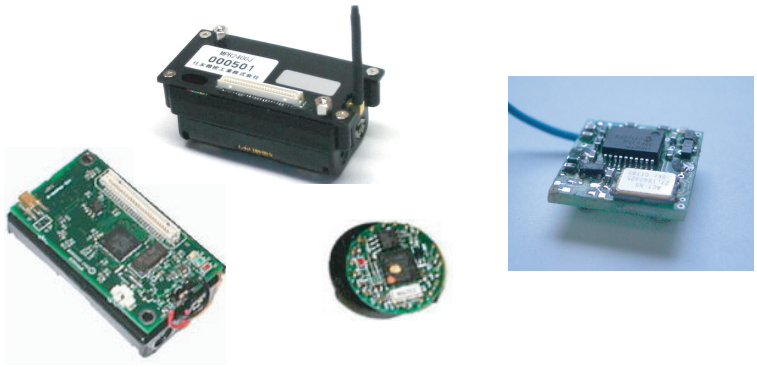
2.1.2 無線通信モジュール

センサノードの通信用に使われる無線帯域は 2.4GHz 帯か 315MHz 帯である．Mote の MICAz は 2.4GHz 帯を使用した Zigbee IEEE 802.15.4 無線通信規格を用いる．その他の Mote 基板と uPart は 315MHz 帯で無線通信する．2.4GHz 帯は無線 LAN や家電ネットワークで使われる無線通信帯域で，315MHz 帯に比べ到達距離が長く通信速度も速いが，電力消費量は高くなる．また，Zigbee だけでなく無線 LAN の IEEE 802.11b や Bluetooth 等の他の無線通信規格が同じ帯域を使用しているため，無線干渉が起こる可能性がある．MICAz で採用されている Zigbee は 2.4GHz 帯を使った無線通信規格であるが，他の規格に比べ到達距離が短く低速になるが，電力消費が低く低コストなのが特徴である．315MHz 帯は微弱電波を用いる無線通信帯域である．消費電力を抑えられるが，通信距離は短く通信速度も遅い．しかし，2.4GHz 帯に比べ直進性が弱く回り込み特性があるため，障害物を回避した通信を期待できる．また，Mote ではノード同士がアドホックネットワークを構築し，マルチホップで通信を行う．uPart はノード間でのマルチホップは行わず，直接基地局と通信する．

図 2.1 に Mote（a）と uPart（b）をに示す．また，Mote と uPart の特性について表 2.1 にまとめた．

2.2 ユビキタスサービス

センサノードを利用する事で，現実世界の状態に即したユビキタスサービスが可能になる．本節では，センサノードを用いたユビキタスサービスについて例を示しながら述べ，



(a) (b)

図 2.1: Mote（a）と uPart（b）

表 2.1: Mote 及び uPart の特性

センサノード		搭載センサ	センサ拡張性	無線周波数	通信形態
Mote	MICAz	光，音， 温度， 2 軸加速度等	有り	2.4GHz	マルチホップ
	MICA2			315MHz	
	DOT				
uPart		光，温度，振動	無し	315MHz	ワンホップ

その分類を行う．

2.2.1 屋内アプリケーション

センサノードを利用したアプリケーションとしては，まず単純な情報収集アプリケーションが考えられる．学会や商品展示会等のイベントで，来場者やブースの状態を監視し，記録しておくとイベントの運用に役立つ．実際に，2005 年に東京で行われた Ubicomp2005[3] では，来場者の入場タグに uPart を入れておき，来場者の状態をロギングしていた．

環境から得られた情報を基にして，人間にサービス提供するアプリケーションも可能である．例えば，室内の温度情報から空調の設定を変更したり，光量からカーテン等の遮光装置を動かす等のアプリケーションが考えられる．人間に対するサービス提供以外のアプリケーションも考えられる．Intel Research Laboratory の PlantCare プロジェクト [4] では，植物やその周辺の状態監視にセンサネットワークを用いている．植物を育成する際にこの情報を活用すれば，環境を植物に最適な状態に保てるようになる．

また，センサノードを災害や事件・事故対策に用いるアプリケーションも考えられる．例えば，環境内の振動情報を取得する事で，地震を検知できる．これを利用して，地震時にデバイスが異常動作しないように機能を停止させるアプリケーションも可能である．災

害が起こった後の救助活動にセンサノードを利用する例も考えられる．屋内でガス漏れが起こると，人が進入できなくなり，中の状態がわからないため救助が遅れる事がある．そこで，前節で述べた自律移動型センサノードを災害現場に進入させれば屋内の情報収集が可能になるため，後の救助活動に役立つ．事件を未然に防ぐための警備アプリケーションにもセンサノードは適用可能である．近年のロボット技術に進歩により，警備用の自律移動ロボットの開発が進んでいる．綜合警備保障の警備ロボット「Ｄ１」は既に実用化されている [5]．しかし，大きなビルのような空間で運用するにはロボット１台に搭載されたセンサだけでは異常検知が難しい．この種類のロボットはまだ高価である事から複数台の導入は難しいため，環境内にセンサノードを配置してロボットと連携する事で運用効率の向上が望める．

2.2.2 屋外アプリケーション

2.3 センサノードの配置条件

2.3.1 無線到達性

2.3.2 サービス毎の用件

監視対象の指定

アプリケーションによって監視対象は異なってくる．まず特定の物を監視対象とするアプリケーションが考えられる．植物監視アプリケーションがこれに該当する．次に特定の場所を監視対象にするアプリケーションが考えられる．例えばビル警備アプリケーションで，窓からの侵入者がいないか監視する場合これにあたる．特定の物を監視するアプリケーションとの違いは，監視対象が移動可能かどうかという点である．また，監視対象とする物が事前にわからない場合は空間全体が監視対象になる．屋内で室温を収集し，空調を調整するアプリケーションがこれに該当する．

動的再配置

ガス漏れ時の室内調査アプリケーションでは，室内に人がいるかどうかが重要になってくる．災害時には人がどこにいるか事前にはわからないので部屋全体が監視対象になるが，センサノードが分散してしまうので場所あたりの監視の度合いが減ってしまう．そこで，いずれかのセンサノードが人がいる兆候をセンシングしたら，確実に人がいるか確かめるために，その場所を中心にセンサノードを密に再配置するという手法が考えられる．つまり,「人がいる兆候」というイベントにより,「部屋全体が監視対象」から「特定の場所が監視対象」へアプリケーションの活動が変化する．

センサノードからの情報以外にも，時間がイベント源となりえる．例えばビル警備では，活動は夜間のみにし，昼間はセンサノードを回収しておく（活動を停止する）といった場合である．

監視環境情報

使用するセンサもアプリケーションによって異なる．これはアプリケーションの監視したい事象によって決まる．例えば地震監視アプリケーションの場合，室内の振動を監視するので振動センサを使用する．アプリケーションによっては複数のセンサを使用する場合もありえる．環境情報を記録するアプリケーションでは複数種類の情報が扱われる．

センサノード配置密度

アプリケーションの活動にとって最適なセンサノードの密度は様々である．これはアプリケーションの活動がどの程度重要なのかに起因する．一般にセンサノードの単価は低く抑えられており，通信やセンサの品質が低い場合が多い．そのため，センサノードの密度が低いと，パケットのロスが起こったり，センサ情報が不正確になったりする．そこで，重要であると考えられるアプリケーションでは，センサノードの密度を高くしてセンサネットワーク全体の信頼性を上げる．例えば災害救助に関連するアプリケーションの場合，人命に関わるのでセンサノードの数はできる限り多くして空間内でのセンサノード密度を高くしたい．しかし，日常的に使われるアプリケーションに関しては，1つのアプリケーションがセンサノードを多く使いすぎると他のアプリケーションの活動に支障をきたしてしまうので，活動に十分な密度さえあればよい．

また，地震のように環境全体に影響を与える事象の監視に対しては，センサノードの密度が低くても監視可能であると考えられる．

以上を表2.2にまとめた．

表 2.2: 各アプリケーションの特徴

アプリケーション	監視対象	イベントによる活動の変化	使用するセンサ	密度
イベント情報収集	空間全体	無し	光，音，温度	低〜中
室内環境調節	空間全体	無し	光，温度	中
植物監視	特定の物（植物)	無し	光，温度	低
地震対策	空間全体	無し	振動	低
災害救助	空間全体，特定の場所	有り（監視対象の変化）	光，音，温度，その他	高
ビル警備	空間全体，特定の場所	有り（監視対象の変化，活動の停止）	光，音，温度，その他	高

2.3.3 配置場所による分類

2.4 アプローチ

以上で述べたように，アプリケーションによってセンサノードの配置条件は違う．このため，人手でセンサノードを配置する時には，アプリケーション運用者はアプリケーションがどのような情報を期待し，どのように動作するか詳しく知っていなければならない．また，自律移動型センサノードのように自動配置が可能なセンサノードを使う場合でも，アプリケーション運用者はアプリケーションにとって最適な配置をセンサノードに伝える必要がある．しかし，必ずしもアプリケーション運用者がアプリケーションについて詳細な知識を持っているわけではない．

本研究ではアプリケーション製作者がセンサノードの配置条件を指定できるように，配置条件の抽象化を行う．そして，条件指定の方法として，アプリケーションから利用可能なミドルウェアを開発する．

センサ情報の抽象化を行っている例として，SensorML（Sensor Model Language）[6] が挙げられる．SensorML はセンサーの情報を記述するための XML ベースの言語である．SensorML を利用する事により，センサの発見や位置情報の取得が可能になる．しかし，SensorML はセンサ情報の記述を主眼に置いているので，アプリケーションにセンサを適応させるという用途には使えない．

また，センサを使ったアプリケーションの抽象化の例としてカリフォルニア大学の Ben による A Sensor Network Application Construction Kit (SNACK)[7] が挙げられる．これはセンサネットワークを利用したアプリケーションの簡易構築キットで，言語やコンパイラ，ライブラリが含まれる．これを利用する事で，センサネットワークを利用したアプリケーションの簡単な構築が可能になる．しかし，実際の利用時にどのようにセンサを配置するかという点にまでは言及していないため，アプリケーション適応的なセンサノードの配置は実現できない．

自律移動型センサノード研究の発展に伴って，センサノードの自動配置アルゴリズムが考案されるようになった．Howard らの提案するアルゴリズムは自律移動型センサノードを障害物のある未知の空間に配置する [8]．このアルゴリズムでは，直前に移動した自律移動型センサノードの情報を元に，各自律移動型センサノードの配置場所を逐次的に決定していく．各センサノードの監視範囲が重複しないように配置していくため，少ないセンサノードでもセンサネットワーク全体の監視範囲を大きくできる．しかし，この手法ではアプリケーションに適応したセンサネットワークの構築ができない．村瀬氏は自律移動型センサノードの自動配置に際して，配置場所の偏りを無くしつつ，センシング対象を複数の自律移動型センサノードで監視する手法を提案している [9]．この手法により，センシング対象を監視できない時間を短くし，誤検知やノイズが起きてもセンシング精度を維持できる．また，アプリケーションによってセンシング対象が異なる事を考慮している点が注目できる．しかし，センサノード配置条件としてはセンシング対象しか指定できず，十分にアプリケーションに適応できるとは言えない．

2.5　本章のまとめ

本章では，センサノードを使ったユビキタスサービスについて述べた．

第3章　ROASENフレームワーク

本章では，まず本研究の目的を達成するためのアプローチについて述べる．次に，そのアプローチを実現するためのセンサノード配置条件の抽象化について述べ，最後に本研究の関連研究について述べる．

3.1 ROASENフレームワークの概要

本節ではROASENフレームワークの概要について述べる．まずROASENフレームワークが想定するシナリオを挙げ，次にROASENフレームワークのユースケースを説明する．

3.1.1 シナリオ

まず，ROASENフレームワークが存在しない場合のシナリオを挙げる．

> あるソフトウェア会社の開発者Aはセンサノードを使った室内環境調整アプリケーションを開発する事になった．このアプリケーションは，センサノードから得られる環境情報を基に室内の空調やブラインドを操作する物である．アプリケーションの開発は順調に進み，社内でのテストも無事に終わり，いよいよアプリケーションを販売する事になった．会社の経営者は，様々な人にアプリケーションを使ってもらうために，アプリケーションに必要なハードウェアやソフトウェアをパッケージ化して販売し，現場への設置はアプリケーション利用者に任せる事にした．販売からしばらくすると，会社に「アプリケーションが正常に動作しない」という苦情が多数寄せられた．ユーザサポート部署からの要請でAが詳しく事情を調べた所，アプリケーション開発時の想定とは違う場所にセンサノードが配置されていた．また，センサノードが無線の届かない位置に配置されていたため，通信が上手く行えていないケースもあった．今回のアプリケーションは多数のユーザに販売しており，Aはセンサノードの位置を直すために全国各地を回らねばならなかった．

次に，ROASENフレームワークが利用可能な状況でのシナリオを挙げる．

> Aと同じくソフトウェア開発会社に勤務する開発者Bは，センサノードを使った室内環境調整アプリケーションを開発する事になった．アプリケーションの開発途中で，Bはセンサノードを用いるアプリケーションの動作はセンサノードの位置に強く依存する事に気付いた．ユーザが間違った位置にセンサノードを配置した場合，アプリケーションは誤動作を起こす可能性がある．また，配置位置によってはセンサノードの無線通信が不安定になる場合がある事も想像された．ドキュメントを用意して注意事項を明記しておく事も考えたが，実際にユーザがドキュメントを読むかは保証できず，センサノードに関する知識が無い人にも理解できるドキュメントを作成するのは難しい．．早速社内での会議でこの事を報告した所，ソフトウェアがユーザにセンサノードをどこに置けばよいか指示してくれる機能を付ける事になった．Bがこのような機能を実現するソフトウェアを探した所，ROASENフレームワークの存在を知った．ROASENフレームワークを使うと，アプリケーション開発者はセンサノード配置条件をユーザに伝えられるようになる．Bは開発中のアプリケーションにROASENフレームワークを導入し，アプリケーション無事に発

売された．販売からしばらくしても，会社にセンサノードの配置に関する問い合わせが来る事は無く，アプリケーション開発は成功裏に終了した．

3.1.2 アプリケーションに関わる人間

センサノードを使ったアプリケーションに関わる人間として，3 種類の人間が挙げられる．まず，アプリケーションを開発，配布するアプリケーション開発者である．次に，センサノードを環境内に配置し，アプリケーションを運用するアプリケーション運用者である．最後に，実際にアプリケーションからのサービスを受けるアプリケーション利用者である．アプリケーションによってはアプリケーション運用者とアプリケーション利用者は同一人物である可能性がある．これまで，センサノードを使ったアプリケーションは研究や実験目的での開発が主だったため，アプリケーション開発者とアプリケーション運用者は同一人物か近い関係の人間である事が多かった．そのため，アプリケーション開発者とアプリケーション運用者の役割の違いや関係について深く議論される事はあまり無かった．アプリケーション開発者はアプリケーションがどのような環境で運用されるか知っており，また，アプリケーション運用者もどのように運用すればアプリケーションが有効に機能するか知っていた．しかし，センサノードを使ったアプリケーションが社会に広く浸透するようになると，そのような前提は成り立たなくなる．

アプリケーション開発者は，環境内にどのようにセンサノードを配置するか自身で想定をしながらアプリケーションを開発する．アプリケーション運用者はアプリケーション開発者の想定通りにセンサノードを環境内に配置する．アプリケーション開発者の想定通りにセンサノードを配置しなかった場合，アプリケーションは有効に機能しない可能性がある．これまでは，アプリケーション開発者がアプリケーション運用者にセンサノード配置条件を伝える決められた手段が無かった．本論文では，ROASEN フレームワークを設計，実装し，アプリケーション開発者がセンサノード配置条件を指定できるようにする．図 3.1 にアプリケーションに関わる人間のそれぞれの役割と関係をまとめた．

3.1.3 ユースケース

図 3.2 に ROASEN フレームワークのユースケースを示す．ROASEN フレームワークでは，アプリケーションは ROASEN フレームワーク上で動作する．まず，アプリケーション開発者はアプリケーションにセンサノード配置条件を格納し，アプリケーションを ROASEN フレームワークに組み込む（1）．アプリケーション開発者はアプリケーションと ROASEN を一緒にアプリケーション運用者に配布する．アプリケーション運用者は，アプリケーションを運用したい空間の情報を予めマップ情報配信サービスに登録しておく（2）．ROASEN フレームワークはアプリケーションのセンサノード配置条件や空間情報を基にセンサノード配置位置を決定する（3）．決定したセンサノード配置位置はアプリケーションに伝えられる（4）．センサノードから得られる環境情報は一度 ROASEN フレームワークに集められ（5），ROASEN フレームワークからアプリケーションに渡される（6）．ROASEN フ

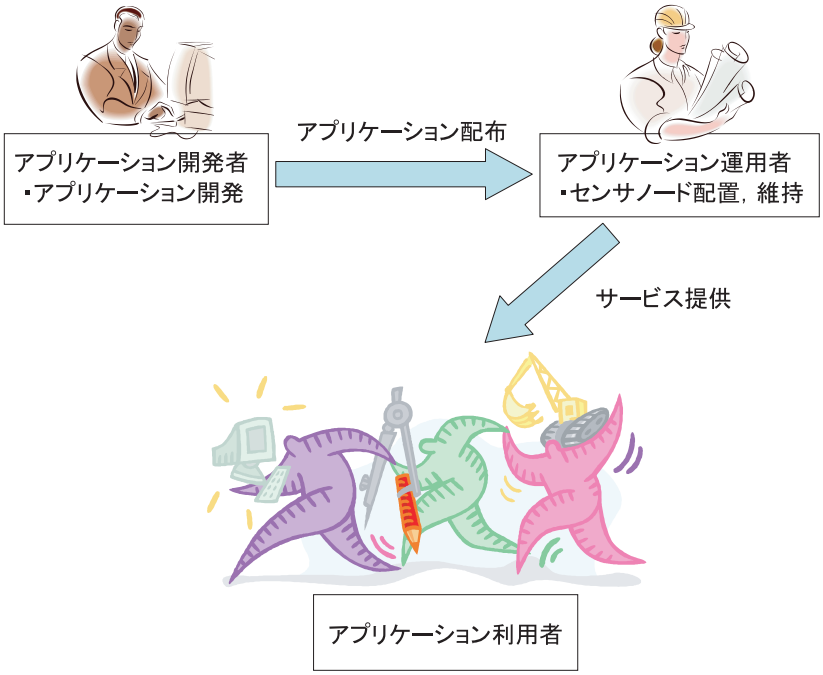


図 3.1: センサノードを使ったアプリケーションに関わる人間

レームワークから渡される環境情報を基に，アプリケーションはアプリケーション利用者にサービスを提供する（7）.

3.2　センサノード配置条件抽象化

本節では，ROASEN フレームワークで用いるセンサノード配置条件の抽象化手法について述べる．まずセンサノードの無線到達性を考慮した配置について述べ，次に監視対象に応じた配置方法について述べる．

3.2.1　無線到達性を考慮した配置

センサノードにはマルチホップアドホックネットワークの構築が可能な種類がある．広い範囲にセンサネットワークを構築する場合，ネットワークの途中で無線接続が切れないように中継ノードを配置する．センサノードの無線通信の電波到達範囲は理論的には球形に広がるはずである．しかし，実際には壁や柱等の障害物や電波反射の影響で歪な形に広がり，人間が実際の電波到達範囲を把握する事は困難である．そのため人間がセンサノードを配置する際には無線到達距離の理論値に頼るしかないが，電波の届かない場所に中継ノードを配置してネットワークが寸断されてしまったり，逆に必要のない場所に中継ノードを配置してセンサノードに無駄ができてしまったりする（図 3.3）．uPart のようにマルチホップを行わないセンサノードを使う場合は，無線が到達しない場所にセンサノードを

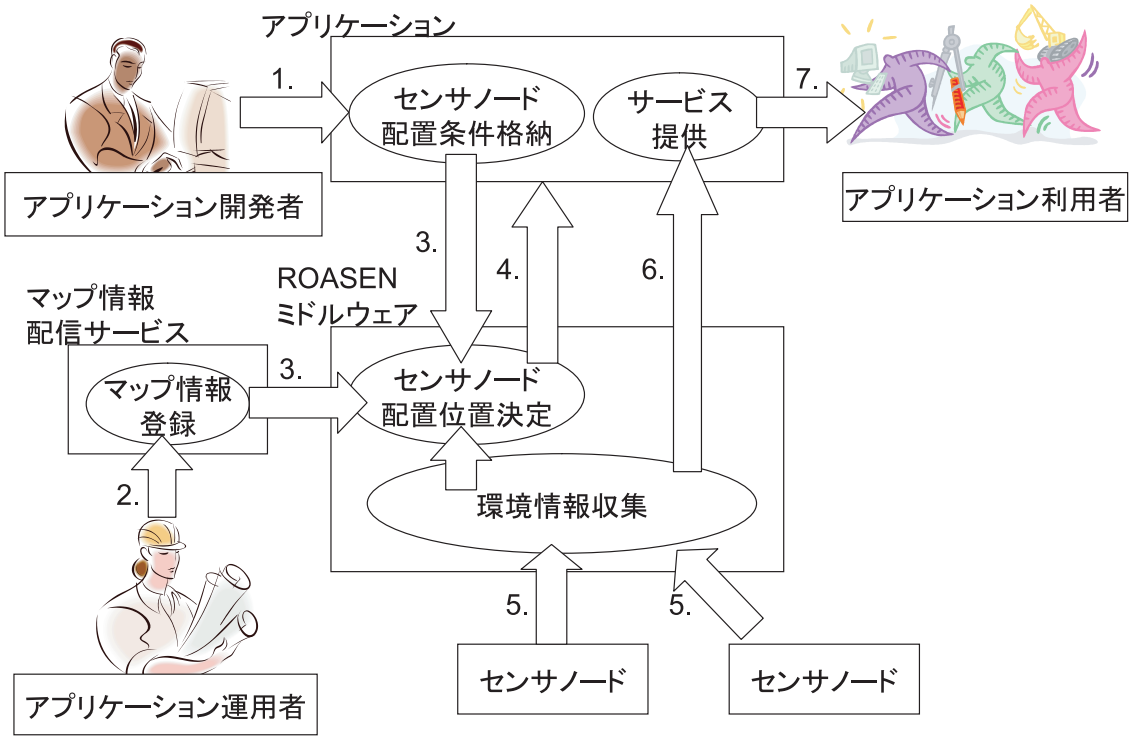


図 3.2: ユースケース

置く事でネットワークが寸断されてしまう事は無い．しかし，無線ネットワークから孤立した場所にセンサノードを置く事はセンサノードの浪費である．従来は，このような無線に関する管理はアプリケーション運用者に任されており，大きな負担になっている．

そこで，ROASEN フレームワークでは，無線到達性を考慮した配置を行う．ROASEN システムでは実際に電波が受信できるかどうかを確認しながらセンサノードを配置する．そのため電波の届く場所に限定したセンサノードの配置が可能である．また，電波到達範囲の外縁付近にセンサノードを配置していくことにより，少ないセンサノード数でセンサネットワークの無線到達範囲を最大にできる（図 3.4）．人間のように電波を吸収する物体が多数動き回る環境では，無線通信が不安定になる．そのため，センサノードを配置した時点では通信が確立していても，運用後しばらくして無線通信が行えなくなる可能性がある．そこで，ROASEN フレームワークは常時無線通信を監視し，無線が届かないと判断した場合には動的にセンサノードの再配置を行う．これにより，アプリケーション運用者が無線通信の確認を行う必要が無くなり，アプリケーション運用者の負荷が大きく軽減される．

3.2.2 監視対象

アプリケーションがセンサノードを使って環境情報を取得する時，アプリケーション開発者は情報を取得したい特定の監視対象を想定している．アプリケーション運用者がセン

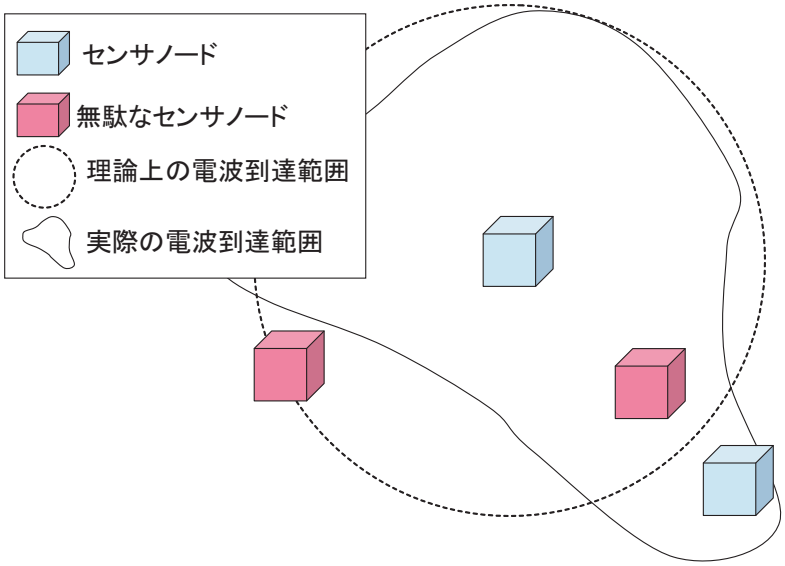


図 3.3: 人間によるセンサノード配置

サノードを配置する時には，アプリケーション開発者が想定している監視対象をセンサノードによって監視できるように配置しなければならない．しかし，従来は監視対象を抽象化する手法が無く，アプリケーション開発者がアプリケーション運用者に監視対象の情報を伝えるには口頭か文書を使って伝えるしか方法が無かった．また，実際に伝えたとしても，センサノードが監視対象を有効に監視できているか確かめるにはセンサノードやアプリケーションに関する詳細な知識が必要になる．このため，アプリケーション開発者がアプリケーション運用者に監視対象を伝える事は難しかった．

ROASEN フレームワークでは，アプリケーションがセンサノードを使って監視する対象を，空間全体，非移動体，移動体として抽象化している．空間全体は，アプリケーションが動作する環境が全て監視対象となり，アプリケーションが取得したい情報が環境内のどこで起こるかわからない場合に用いられる．非移動体は動く可能性の低い物体であり，家具や大型家電のような一度設置したらほとんど動かないと考えられる物である．移動体は，環境内を動く可能性のある物体であり，ノート PC のように人間が持ち運び可能な小型の物体がこれに当たる．ROASEN フレームワークは，これらの監視対象の種類毎に異なる配置場所決定手法が取る．

3.3 センサノード配置場所決定アルゴリズム

本システムでは，センサノードの配置場所決定手法として，空間網羅型，壁際型，移動体注目型，非移動体注目型の 4 つの方法を取れる．これらの手法は前節で述べた空間全体，移動体，非移動体という監視対象の種類に対応している．空間網羅型はセンサノードを環境全体に配置する場合の配置場所決定手法である．環境内のどこでアプリケーションが取得したい環境情報が発生するか事前にわからない場合，センサノードの配置が偏

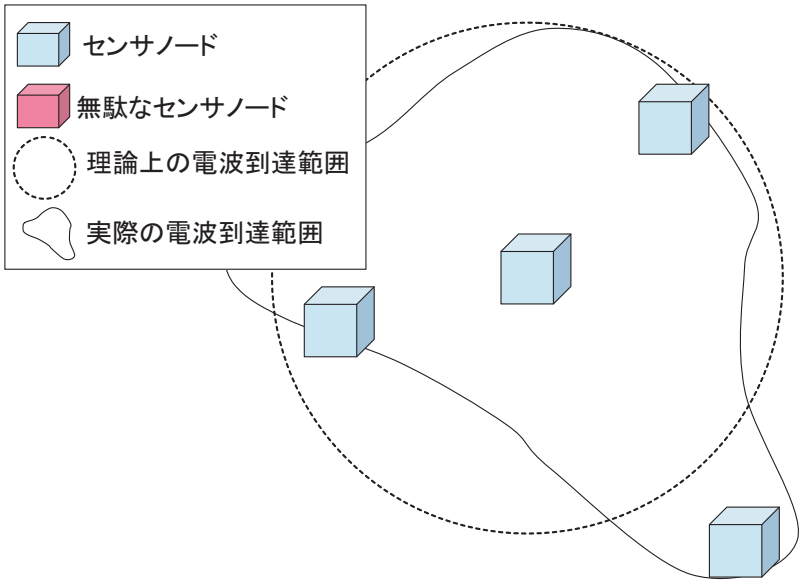


図 3.4: ROASEN によるセンサノード配置

ると情報の取得に失敗してしまう可能性が高くなる．そのため，空間網羅型はセンサノードを空間全体に均等に配置されるように位置を決定する．ここで，センサノードを空間全体に均等に配置する事は，センサノード配置位置を母点としたボロノイ図を作った時，各ボロノイ領域の面積が等しくなり，且つ隣り合う母点同士の距離が全て均等になるという事である．このようなセンサノード配置位置の決定手法として，今回はセンサノードの位置を空間内に互い違いに設定する手法を取った．この時，各ボロノイ領域は同じ面積の正六角形の形を取り，得られるドロネー図は平面内に正三角形を敷き詰めたような図になる (図 3.5)．つまり，各ボロノイ領域の面積が等しくなり，且つ隣り合う母点同士の距離が全て均等になるので要件に合致する．しかし，実際にアプリケーションを運用する空間では障害物がある可能性があり，この手法通りにセンサノードを配置できない場合がある．ROASEN フレームワークでは，障害物がある場合には理想的な配置位置から最も近い場所を配置位置とする事でこの問題に対応する．

空間全体を監視対象にしたい場合でも，イベント会場等の人の動きが激しい場所での使用を想定しているようなアプリケーションでは，環境全体にセンサノードを配置してしまう空間網羅型は使いたくない場合がある．このような時は壁際型が使える．壁際型はセンサノードを壁際に配置する場合の配置位置決定方法である．ROASEN フレームワークは隣り合うセンサノード同士の距離が等しくなるよう均等に位置を決定する．図 3.6 に壁際型配置の様子を示す．

非移動体注目型は動く事を考慮しなくてよい物体を監視する際に用いる配置場所決定手法である．非移動体を監視する際にはその物体の位置に対するセンサカバー率が重要になる．センサの効果は距離が離れるほど弱まるので，なるべく近い距離に置く事でセンサカバー率を上げられる．しかし，水を扱う物体のように監視する物体によってはセンサノードをあまり近づけたくない場合がある．また，センサノードの数を増やす事でもセンサカ

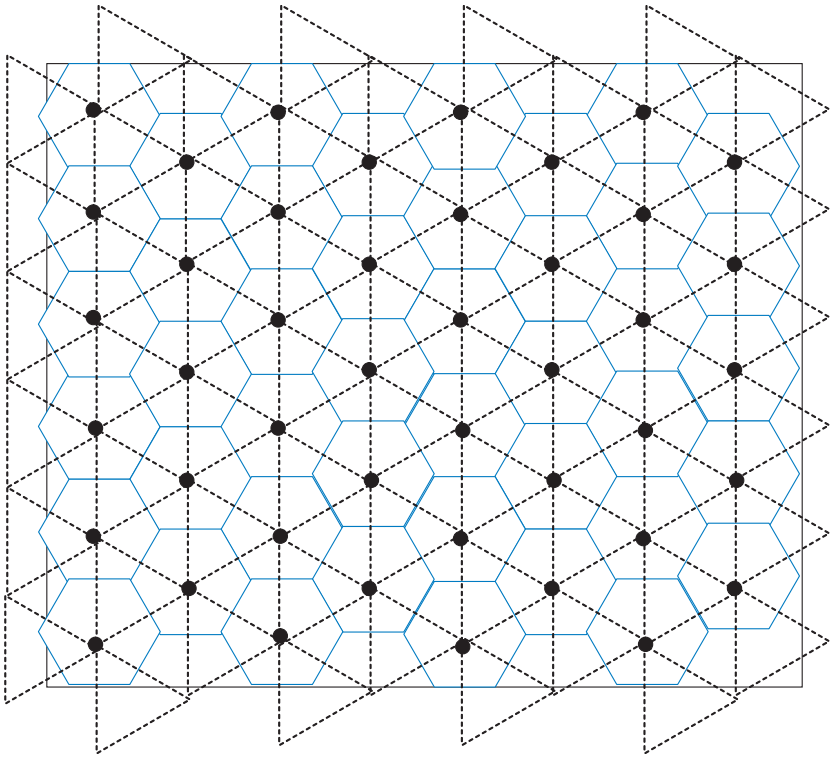


図 3.5: 空間網羅型

バー率は上昇する．しかし，あまり重要でない情報を取得するためにセンサカバー率を上げるためにセンサを最大限使用してしまう事はセンサノード資源の浪費である．ROASEN フレームワークでは，アプリケーション開発者によって監視対象とセンサノードとの距離を指定可能にしたり，アプリケーション運用者がセンサノード配置禁止領域を設定できるようにする．また，アプリケーション開発者が監視対象がどの程度重要なのか記述できるようにする事で，センサノードの数を調整可能にする．図 3.7 にオブジェクト注目型配置の様子を示す．

移動体注目型は，移動体を監視する際に用いる配置場所決定手法である．基本的な監視方法は非移動体注目型と同じだが，監視対象が移動しても引き続き監視を継続できる．移動体の移動の種類には 2 種類ある．まず移動先が決まっている場合である．例えば，ノート PC を昼は書斎の机で使い，夜は寝室の机で使うというように決めている例が考えられる．次に移動先が決まっていない場合である．これはノート PC を家の中のどこで使うか全く分からない状況に該当する．移動先が決まっている場合，移動先候補の全てにセンサノードを配置しておき，監視対象の一番近くにあるセンサノードからの情報だけを取得するという手法が使える．移動先が決まっていない場合には，移動体が移動する度にセンサノードの位置を再計算するという手法が使える．ROASEN フレームワークでは，アプリケーション開発者がセンサノードに移動体を追跡させるか決定できる．

使用するセンサによっては配置位置決定手法を調整する必要がある．例えば，センサノードは安価で製造される事が多いため，センサの品質にばらつきがあったり，センサの

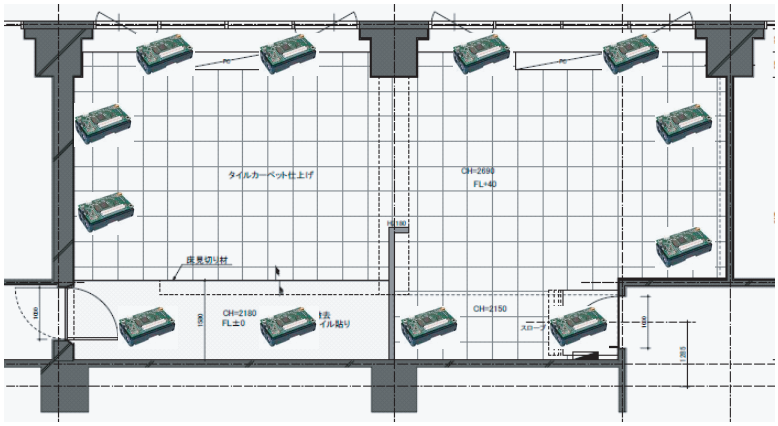


図 3.6: 壁際型配置の様子

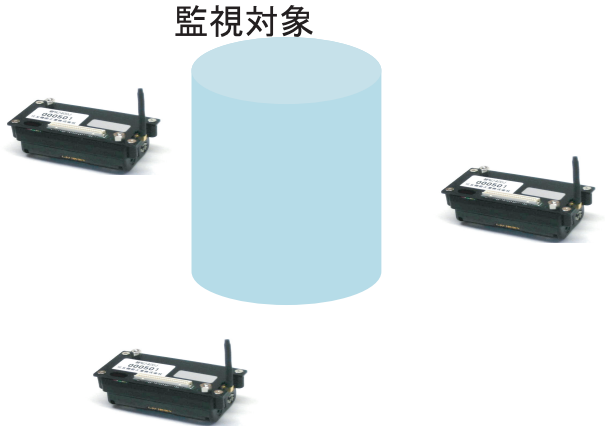


図 3.7: 非移動体注目型

性能が悪い場合がある．また，センサの種類によって監視対象とする物自体が適切な環境情報の取得を妨げる場合がある．例えば，光センサを使用する場合，監視対象周辺の光量を取得したいのに，監視対象自体が影を作ってしまい，偏ったデータしか取れない事がある（図 3.9）．これらの問題を解決するために，ROASEN フレームワークは使用するセンサによってセンサノードの配置数を調整する．図 3.10 に使用センサによって配置ノード数に補正をかける様子を示す．

3.4　本章のまとめ

本章では ROASEN フレームワークの実装について述べ，アプリケーション開発者から利用できる RoasenXML と RoasenAPI について説明した．

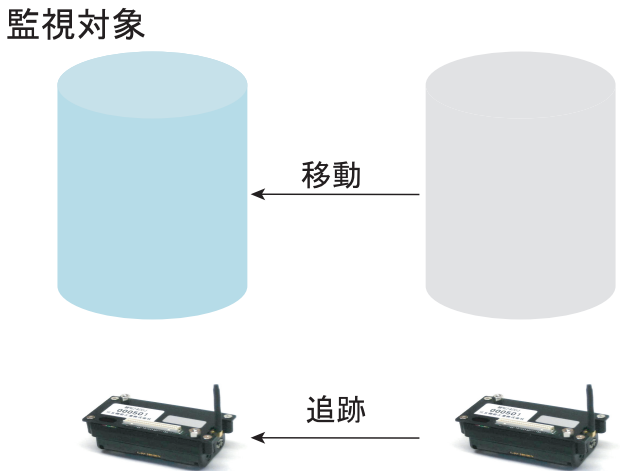


図 3.8: 移動体注目型

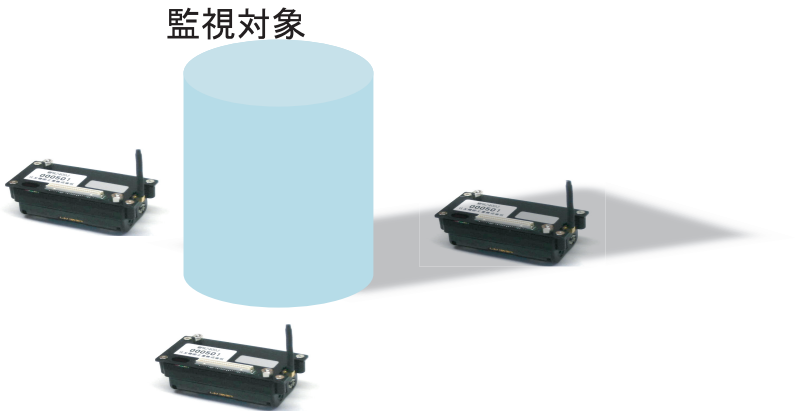


図 3.9: 監視対象自体が環境情報取得を妨げる

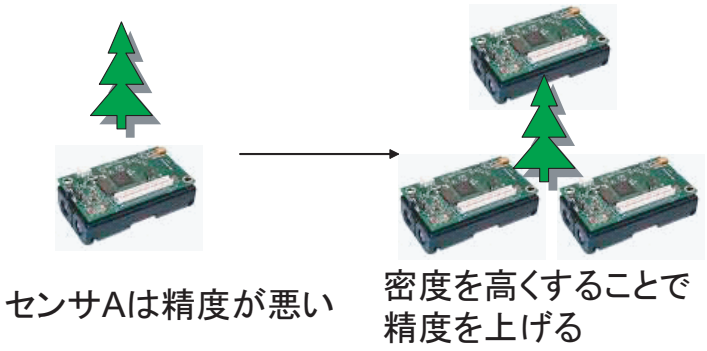


図 3.10: 配置ノード数の補正

第4章　実装

本章では，3章で述べたセンサノード配置条件抽象化機構の実現のためのROASENフレームワークの実装について述べる．まず，フレームワーク全体の概要と動作概要について述べる．次に，フレームワークを構成する各モジュールについて述べ，実装に用いた環境を紹介する．

4.1　システム概要

本節では，ROASEN フレームワークのシステム構成と動作概要について述べる．

4.1.1　システム構成

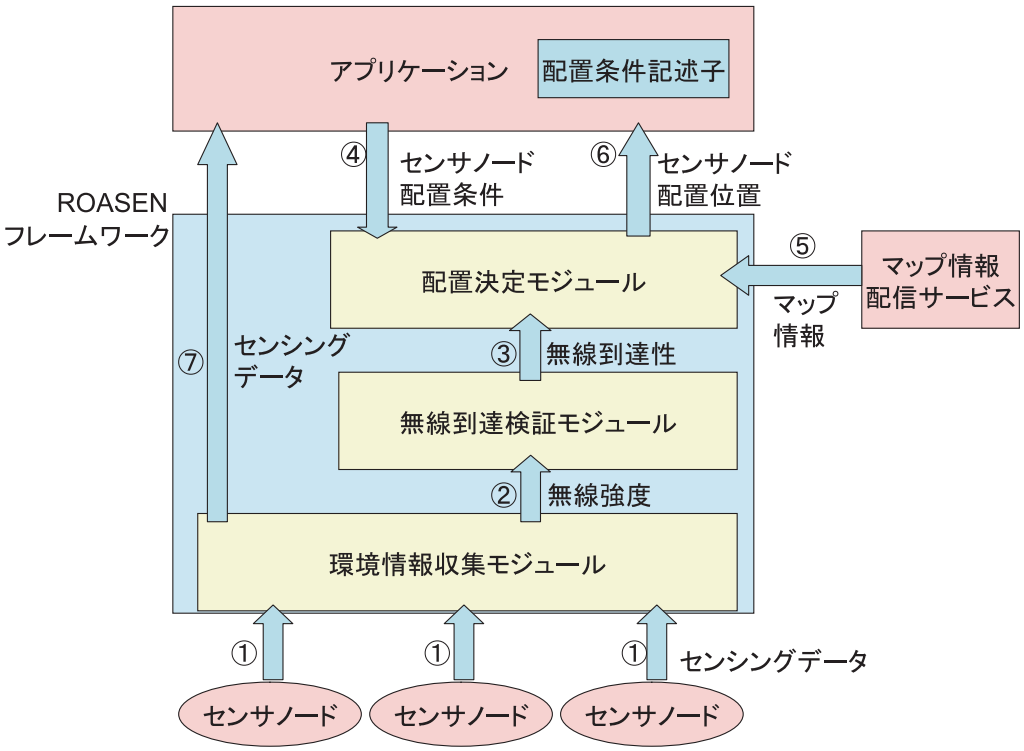


図 4.1: ROASEN システム構成図

図 4.1 に ROASEN フレームワークのシステム構成を示す．ROASEN フレームワークは環境情報収集モジュール，センサノード位置決定モジュール，無線到達検証モジュールから構成される．環境情報収集モジュールはセンサノードからの環境情報を収集し，アプリケーションに渡す．無線到達検証モジュールはセンサノードの無線強度やパケットが到達しているか確認し，無線到達性を検証する．ノード位置決定モジュールは，無線到達性とアプリケーションのセンサノード配置条件，及びマップ情報を基にセンサノードの位置を決定し，センサノード配置機構にその情報を渡す．

4.1.2　動作概要

次に ROASEN フレームワークの動作概要を述べる．ROASEN フレームワークの動作は環境情報の収集，無線到達性の検証，センサノードの位置決定，センサノードの配置と再配置から成る．以下では各動作の詳細を述べる．

環境情報の収集

ROASEN フレームワークはセンサノードを管理し，環境情報を収集してアプリケーションに渡す（1）．環境情報が複数種類ある場合には，アプリケーションから指定された種類のデータを渡す．センサノードからのデータ取得周期もアプリケーションが指定できる．

無線到達性の検証

ROASEN フレームワークは環境情報の収集と同時に，センサノードとの無線到達性の検証を行う．まずセンサノードから無線強度を取得する（2）．無線強度を取得できない uPart のようなセンサノードを使う場合には，環境情報のパケットが到達しているか確認し，無線強度を計算する．この無線強度を一定時間監視し，設定された閾値以上の無線強度が保たれていた場合，ROASEN フレームワークは無線が到達していると判断する．逆に，無線強度が閾値以下になった場合，ROASEN フレームワークはセンサノードとの通信が不安定であると判断し，無線が到達していないと判断する．閾値は通常 50%と設定しているが，アプリケーションから変更可能である．また，無線強度の監視時間もアプリケーションから変更できる．

センサノードの位置決定

ROASEN フレームワークは無線到達性，アプリケーションのセンサノード配置条件，マップ情報を基にセンサノードの位置を決定する．アプリケーションのセンサノード配置条件，マップ情報は静的に決定するが，無線到達性は動的に変化する．そのため，センサノードの位置決定は ROASEN フレームワークの動作中定期的に変更される．まず無線到達性を取得し（3），次にアプリケーションからセンサノード配置条件を受け取る（4）．そして，対象空間の情報として，マップ情報を取得する（5）．環境内にはマップ情報配信サービスが動作していると想定した．センサノードの位置決定アルゴリズムは第 3 章で述べた配置アルゴリズムに従う．センサノードの位置の再計算のタイミングは，通常センサノードからのデータ取得タイミングと同期する．アプリケーションからも再計算タイミングを変更できる．

センサノードの配置と再配置

センサノードの位置が決定すると，ROASEN フレームワークはアプリケーションにセンサノード配置位置を通知する（6）．アプリケーションからセンサノード配置位置の再計算要求が送られてくると，センサノードの位置を再計算し，結果を再びアプリケーションに通知する．

環境情報の送信

ROASEN フレームワークでは，センサノードは ROASEN フレームワークが管理し，アプリケーションからは直接通信できないようになっている．そのため，センサノードから取得した環境情報は ROASEN フレームワークがアプリケーションに送信する（7）．

4.2 構成モジュール

本節では，ROASEN フレームワークを構成するモジュールについて述べる．

4.2.1 配置決定モジュール

ノード位置決定モジュールはセンサノードをどこに配置するか決定するモジュールである．入力データとしては，静的なデータとしてアプリケーションのセンサノード配置条件とマップ情報を受け取り，動的なデータとして ROASEN フレームワーク管理下の各センサノードの無線到達性を無線到達検証モジュールから受け取る．センサノード配置条件は配置条件記述子で記述され，アプリケーション内に格納されている．また，マップ情報は環境の地図や環境内にある物を示す情報で，マップ情報配信サービスから取得する．配置場所が決定すると，センサノード配置機構に配置場所を通知する．

4.2.2 環境情報収集モジュール

環境情報収集モジュールはセンサノードから送られてくる環境情報を収集する．センサノードを抽象化し，複数種類のセンサノードに対応し，アプリケーション製作者がセンサノードの種類を意識しなくてもよいようにする．本論文でははuPart と Mote に対応する．収集した環境情報はアプリケーションに送る．また，環境情報と同時にセンサノードの無線強度を取得し，無線到達検証モジュールに送る．

4.2.3 無線到達検証モジュール

無線到達検証モジュールはセンサノードとの無線通信が安定して到達しているか検証する．検証結果はノード位置決定モジュールに送る．

4.2.4 マップ情報配信サービス

ROASEN フレームワークはセンサノードの配置位置を決定するために，動作している環境のマップ情報を取得する必要がある．マップ情報とは，マップ情報は環境の形状や環境内に置かれた物の情報を含んだ情報である．環境に関する情報としては，例えば環境の大きさや壁と柱の位置等がある．また，環境内に置かれた物に関する情報として，物の

ID や種類，大きさ，位置等がある．ROASEN フレームワークはこのマップ情報から監視対象を見つけたり，センサノードを置ける場所と置けない場所を区別する．本研究では，ROASEN フレームワークがマップ情報を取得できるように，環境内にはマップ情報配信サービスが動作していると想定した．マップ情報配信サービスは環境に関する情報や環境内に置かれた物を管理し，リクエストに応じてマップ情報を配信する．

4.3 RoasenXML

本節では，センサノード配置条件を記述するための RoasenXML について述べる．RoasenXML は配置条件を記述するための XML で，アプリケーション開発者が記述する．RoasenXML は設定の細かさに応じて 2 つのレベルを持ち，それぞれ HIGH レベルと LOW レベルと呼ばれる．

4.3.1 HIGH レベル

アプリケーション開発者はどちらかを選んで配置条件を記述できる．まず，図 4.2 に HIGH レベルの RoasenXML の書式例を示す．

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE roasenXML SYSTEM "roasenxml.dtd">
<roasenXML applicationName= “exampleHigh">
  <high>
    plantObserver
  </high>
</roasenXML>
```

図 4.2: HIGH レベルの RoasenXML 書式例

HIGH レベルの RoasenXML では，アプリケーション開発者の記述量は最小で済む．ROASEN フレームワークはセンサノードを使ったアプリケーションの典型的なセンサノード配置条件例をテンプレートとして持っている．アプリケーション開発者はこれらのテンプレートの中から 1 つを選択する事で，細かく条件を設定しなくてもセンサノード配置条件を ROASEN フレームワークに伝えられる．具体的には，XML 内の high タグの中でテンプレート名を指定する．HIGH レベルの RoasenXML ではテンプレートして用意されている以上の配置条件設定はできないので，HIGH レベルでは設定しきれないような配置条件を使う場合は LOW レベルの RoasenXML を使う．

4.3.2 LOW レベル

次に，図 4.3 に LOW レベルの RoasenXML の書式例を示す．

センサノードを配置する際，搭載センサの異なるセンサノードを混ぜて配置する場合がある．例えば，光センサを搭載したセンサノードを部屋全体に配置し，加速度センサを持つセンサノードは部屋に 1 つだけ配置するといった場合である．このような状況に対応するため，LOW レベルの RoasenXML での配置条件の設定はセンサノードに搭載されたセンサの種類毎に行う．XML 内では sensorCondition 要素で条件を設定する．type 要素でセンサの種類を指定する．監視対象の指定は space，wallside，mobileObject，nonMobileObject のいずれかの要素を用いることで指定する．space 要素を指定すると，ROASEN フレームワークは空間網羅型でセンサノード配置位置を決定する．wallside 要素は壁際型でセンサノードを配置したい時に指定する．wallside 要素内では，distanceFromWall 要素を記述でき，壁からの距離を指定できる．nonMobileObject 要素はセンサノードの位置を非移動体注目型にしたい時に指定する．nonMobileObject 要素内では distanceFromObject 要素で物体からの位置を指定できる．属性として物の種類を表す typeID を指定する．物の種類はマップ情報配信サービスで使われる ID と同じ物である．mobileObject 要素を指定すると，ROASEN フレームワークはセンサノードの配置位置を移動体注目型で決定する．nonMobileObject 要素と同じように distanceFromObject 要素が使える．また，trace 要素で移動体を追跡するか指定できる．各監視対象がアプリケーションにとってどの程度重要かは importance 要素で 10 段階で表す.

図 4.4 にセンサノード配置条件 XML の DTD を示す.

4.4 RoasenAPI

本節では，アプリケーション開発者が ROASEN の機能を使うためのインターフェースである RoasenAPI について述べる．RoasenAPI はクラスライブラリの形で提供され，アプリケーション開発者は自分の目的に合ったクラスにアクセスする．以下ではアプリケーション開発者に直接関わってくるクラスについて述べる.

4.4.1 Roasen クラス

Roasen クラスは RoasenAPI の中心となるクラスである．アプリケーションが Roasen クラスのオブジェクトにアクセスする事で，センサノードの情報を取得やセンサノード配置位置の計算を要求できる．Roasen クラスのインターフェースは図 4.5 のようになっている.

4.4.2 SensorNode クラス

SensorNode クラスはセンサノードを抽象化したクラスである．ROASEN フレームワークがセンサノードから集めた環境情報はこの SensorNode クラスに格納され，アプリケーションに渡される．このクラスを利用する事で，アプリケーション開発者はセンサノードの違いに左右される事無くアプリケーションを開発できる．図 4.6 に SensorNode クラスのインターフェースを示す.

4.4.3 Application インターフェース

Application インターフェースは ROASEN フレームワーク上で動作する全てのアプリケーションが実装すべきインターフェースとして定義されている．また，ROASEN フレームワークはこのインターフェースを通してセンサノードからの環境情報が取得できた事を知らせる．

以上を図 4.8 にクラス図としてまとめた．

4.5 本章のまとめ

本章では ROASEN フレームワークの実装について述べた．まず実装の概要について述べ，次に動作概要と各モジュールについて述べた．

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE roasenXML SYSTEM "roasenxml.dtd">
<roasenXML applicationName= “exampleMiddle">
  <middle>
    <sensorCondition>
      <type>light</type>
      <mobileObject typeID="10001">
        <importance>10</importance>
        <distanceFromObject>0.1</distanceFromObject>
      </object>
    </sensorCondition>
    <sensorCondition>
      <type>oscillation</type>
      <nonMobileObject typeID="10002">
        <importance>3</importance>
        <trace>on</trace>
      </wallside>
    </sensorCondition>
    <sensorCondition>
      <type>temperature</type>
      <space>
        <importance>6</importance>
      </space>
    </sensorCondition>
    <sensorCondition>
      <type>sound</type>
      <wallside>
        <importance>6</importance>
        <distanceFromWall>1</distanceFromWall>
      </wallside>
    </sensorCondition>
  </middle>
</roasenXML>
```

図 4.3: LOW レベルの RoasenXML 書式例

```
<!ELEMENT importance (#PCDATA)>
<!ELEMENT distanceFromObject (#PCDATA)>
<!ELEMENT distanceFromWall (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT space (importance,distanceFromWall?)>
<!ELEMENT wallside (importance,distanceFromWall?)>
<!ELEMENT nonMobileObject (importance,distanceFromObject?)>
<!ATTLIST nonMobileObject typeID CDATA #REQUIRED>
<!ELEMENT mobileObject (importance,distanceFromObject?)>
<!ATTLIST mobileObject typeID CDATA #REQUIRED>
<!ELEMENT sensorCondition (type,(mobileObject | nonMobileObject | space | wallside))>
<!ELEMENT low (sensorCondition)*>
<!ELEMENT high (#PCDATA()>
<!ELEMENT roasenXML (high | low)>
<!ATTLIST roasenXML applicationName CDATA #REQUIRED>
```

図 4.4: センサノード配置条件指定 XML の DTD

```
class Roasen /* シングルトンで管理 */
{
public:
  const list<SensorNode*>& getSensorNodeList();
  void setSensorNodeNumber(int);
  int getMaxSensorNodeNumber();
  void calculateSensorNodePosition(const SensorDeploymentCondition&);
};
```

図 4.5: Roasen クラス

```
class SensorNode
{
public:
  const SensorPosition& getSensorPosition();
  int getSensorNodeType();
  const SensingInformation& getSensingInformation();
};
```

図 4.6: SensorNode クラス

```
class IApplication
{
public:
  virtual ~IApplication() = 0;
  virtual void receivedSensingInformation() = 0;
};
```

図 4.7: IApplication インターフェース

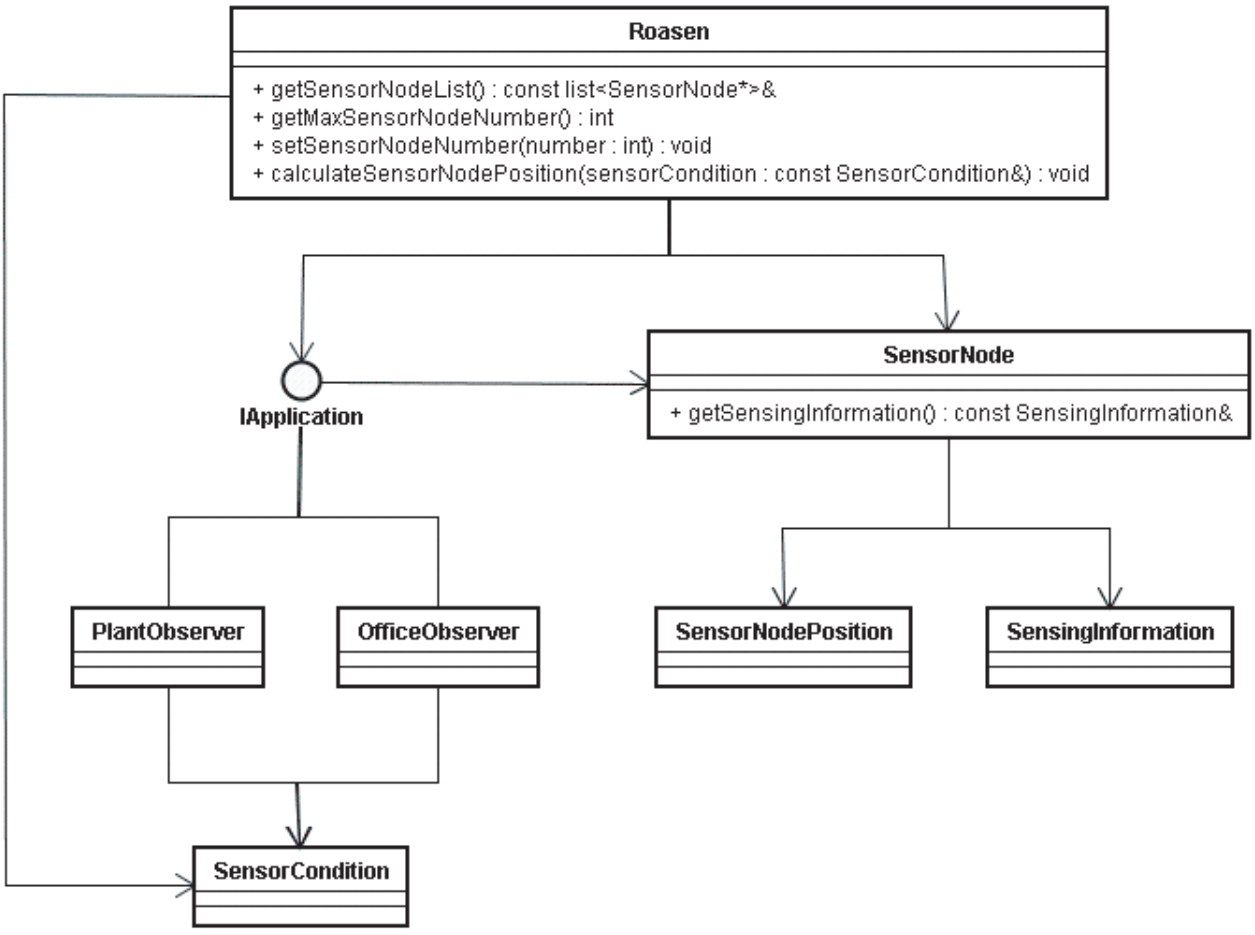


図 4.8: RoasenAPI クラス図

第5章　ケーススタディ

本章ではROASENフレームワークを利用したアプリケーションや，センサノード配置機構について述べる．

5.1　植物監視アプリケーション

本節では，ROASEN フレームワークを用いたアプリケーションとして，植物監視アプリケーションの例を述べる．

5.1.1　概要

植物監視アプリケーションはセンサノードをを使って植物とその周辺環境を監視するアプリケーションである．植物の成長はその周辺環境の状態に影響を受ける．適切な光量と温度を維持した環境下でないと植物は成長を阻害される可能性がある．そこで，植物監視アプリケーションはセンサノードに搭載されたセンサを使って植物と周辺の環境情報を取得し，空調やカーテン等の環境調節装置の設定を変更し植物の育成に最適な環境を作り出す．

植物監視アプリケーションはリビングのような室内環境での動作を想定している．また，対象とする植物は室内で育てられる観葉植物を想定している．

5.1.2　シナリオ

植物監視アプリケーションが有効な例として，次のようなシナリオが挙げられる．

> A は自宅の寝室で観葉植物をたくさん育てている．これらの植物が枯れないように，常に室内の温度や光を植物に最適になるように維持していたが，この作業は非常に面倒だった．ある時，仕事が忙しくなった A はしばらく植物の世話をあまりしなくなってしまった．すると，植物は段々と弱りだした．そこで，A はセンサノードを使って室内の環境調節を自動化する事を思いついた．早速センサノードと植物周辺環境監視アプリケーションを用意した．アプリケーションが動作を始めると，植物の周辺環境は植物に最適に調整されるようになり，観葉植物は元気を取り戻した．

5.1.3　ユースケース

図 5.1 に植物監視アプリケーションのユースケースを示す．

植物監視アプリケーションは ROASEN フレームワーク上で動作するアプリケーションとして実装した．ROASEN フレームワークから環境情報を取得する．この環境情報から，部屋の環境が植物にとって最適か判断し，最適でないと判断した場合には環境内のデバイスを操作する事で環境調節する．環境内には環境管理サービスが稼動していると想定した．環境管理サービスは空調やカーテン，ライト等の部屋の環境調節をするデバイスを管理し，LAN を介してデバイスの操作が可能である．

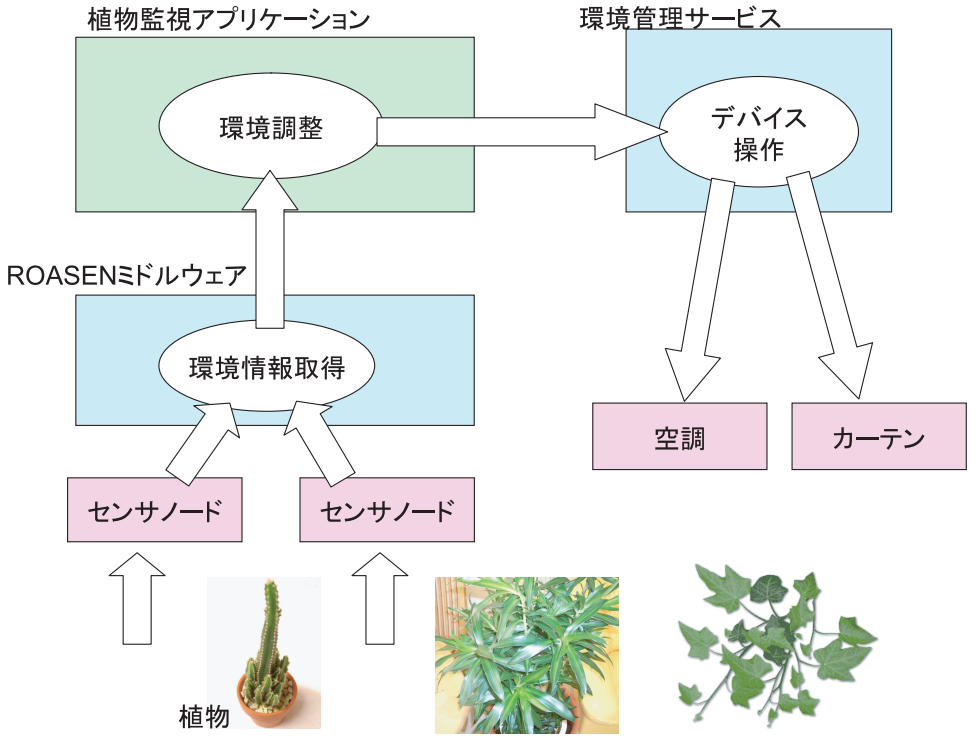


図 5.1: 植物監視アプリケーションのユースケース

5.2　センサノード配置機構

ROASEN フレームワークによってセンサノードの配置位置が決定した後は，実際にセンサノードを環境内に配置していく．センサノードを配置する方法としては，人手による配置と機械による自動配置が考えられる．ROASEN フレームワークでは，センサノードを配置すべき位置をアプリケーション層から取得できるので，これを基にセンサノード配置機構を構築できる．例えばセンサノードの配置位置をディスプレイ上に図示する簡単なビューアを作れば，人手によるセンサノード配置ガイドとして利用できる．また，センサノードを配置するような機械があれば，それにセンサノード配置位置を渡せばセンサノードの自動配置が可能になる．本研究では，センサノードの自動配置を行うための機構として，センサノード自動配置ロボットを実装した．

関連研究

従来のロボットは周辺環境の情報取得方法としてロボット自身に取り付けられたセンサに頼っていた．しかし，近年のユビキタスコンピューティング環境に関する研究の結果，環境内に遍在するセンサから環境情報を取得するロボットが提案されている．ロボットとセンサネットワークの組み合わせとしては，Intel Research Laboratory の Anthony らがセンサネットワークのメンテナンスを行うロボットを提案している [10]．このロボットは，PlantCare プロジェクトで植物を監視するセンサネットワークにおいて，センサノードの

メンテナンスを行う．実験ではセンサノードに搭載されたセンサの誤差調整をロボットを使って行い，その有効性を確かめている．この研究は人手のかかるセンサネットワークのメンテナンスをロボットに行わせるという点で本研究の先行研究と言える．しかし，センサノードの配置問題に関しては詳しく述べられておらず，どのようにアプリケーション毎にセンサネットワークの適応を行うのかという問題が残されている．

また，センサノード自身をロボットにする研究も行われている．通常のセンサノードは自律的な移動手段を持たないため，その配置は人間が行う必要がある．しかし，通常，センサノードは数十個以上という単位で運用されるため，全て人手で配置するには負担が大きい．航空機を使って空中から散布するという手法も考えられているが，風や地形等の影響を受け易く，思い通りの配置にならない場合がある．このような配置に関する困難を解決するために，センサノード自体に移動性能を持たせる研究がある．Southern California 大学の RoboMote は，現在センサノードとしては最も普及している Mote に移動性能を持たせる研究である [11]．同様の研究として，Notre Dame 大学の MICAbot がある [12]．また，MIT の Mathew らはセンサノード自体には移動性能を持たせず，人間や乗り物等の他の移動体に寄生して移動する手法を提案している [13]．これらの自律移動型センサノードは，通常のセンサノードに比べて配置に人手がかからないという利点を持つが，欠点も抱えている．まず，移動手段としてモータと車輪を持つため，サイズが大きくなってしまう．また，電力消費も大きくなり，長期的な使用には向かない．Mathew らの手法にはこれらの欠点は無いが，移動体が周囲にいないと移動できないという欠点を持つ．

本研究では，センサノード配置機構としてロボットを採用する事で，実際にロボットによるセンサノード配置の実験をした．センサノードを配置する機械としては，モバイルセンサノードが使用できる．しかし，モバイルセンサノードを使う場合，アプリケーションで使用する全センサノードが環境を動き回る．人間の生活環境では，人間が環境内を動き回るため，生活の邪魔になるような多数の物体が移動を繰り返すセンサノード配置機構は避けたい．ロボットを使ったセンサノード配置機構の場合，動く物体は最低 1 個で済む．また，人間の生活空間には，既にペットロボットや家事ロボット等が登場している．情報家電や位置取得センサ等のロボットから利用可能なデバイスの増加に伴い，将来的には人間の生活空間に今まで以上の種類のロボットが増えていくことが予想される．これらのロボットがセンサネットワーク構築機能を持てば，ユビキタスコンピューティング環境の構築に役立つと考えられる．

ハードウェア

図 5.2 にセンサノード配置ロボットのハードウェア構成図を示す．また，図 5.3 にロボットの外観を示す．

ロボットの基本ハードウェアは JAPAN ROBOTECH の RoboDesigner[14] を利用して製作した．RoboDesigner は簡単に自作ロボットの製作が可能な学習用キットである．マイコンを搭載したコントローラボードから各種センサからの入力やモータの操作等を行える．本システムでは，ロボットの車体を 2 層構造とし，上層にはコントローラボードと電池を備え付けた．この部分は主にモータ制御と通信の役割を持つ．車体下層にはモータ，セン

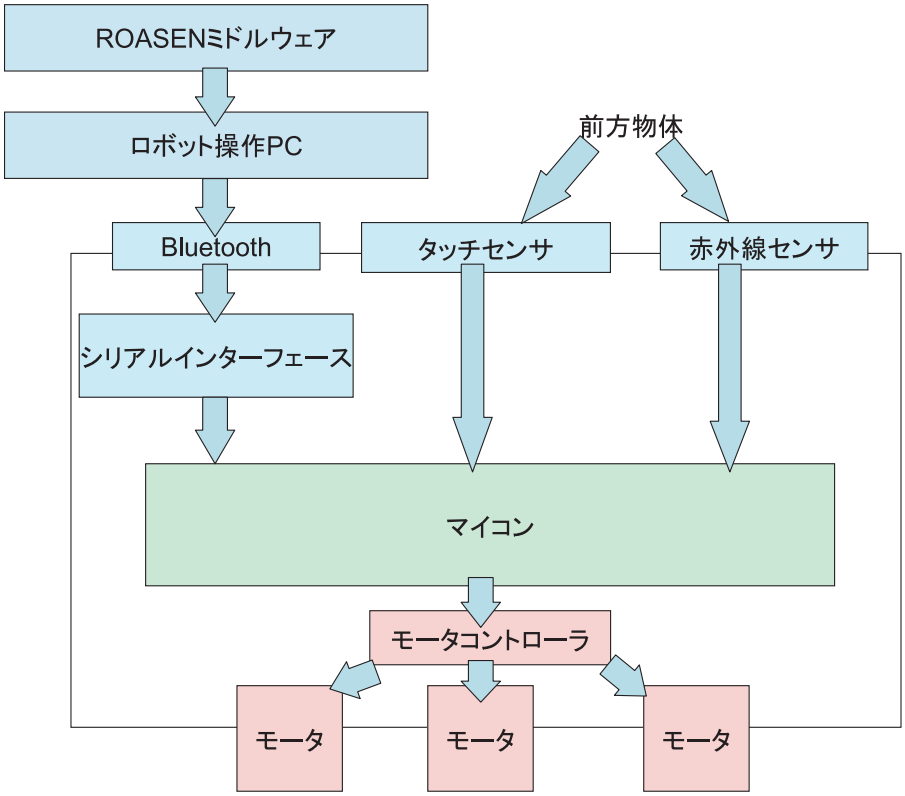


図 5.2: ロボットのハードウェア構成図

サノード配置装置，主処理用計算機を備えた．2 個のモータで車輪を回して移動し，前進，後進，回転が可能である．また，ロボットはセンサノード配置のための装置を持つ．モータによってベルトコンベアを前後に動かし，センサノードを車体から排出して環境内に配置していく．前方確認用のデバイスとして，ロボットは赤外線センサとタッチセンサを搭載する．前方に赤外線を照射し，前方物体の反射から進路上に障害物が無いか確認する．また，タッチセンサに反応があった場合，ロボットのすぐ前に障害物があると判断する．

また，ロボットの動作する環境には，環境内の物体の位置情報を取得するための位置取得サーバが動いていると想定した．今回は位置情報取得用のデバイスとして超音波位置測定センサを備え付け，位置取得サーバで環境内の超音波タグの位置を受信できるようにした．これを利用してロボットが室内での相対位置を取得できるよう，車体には超音波タグを付けた．超音波位置測定センサは産総研の ZPS を用いた [15]．この位置取得サーバは環境情報配信サーバとしても動いており，環境内にある物体の位置情報の配信にも用いられる．

実際のロボットの動きを決定するために環境内にロボット操作 PC を置く．ロボット操作 PC は位置取得サーバと LAN を介して通信し，ロボットの現在位置を取得する．また，ロボットと通信して動作命令を出す．ロボットとの通信方法は Bluetooth を採用した．ロボットが動作命令を受信すると，モータ制御用のマイコンがモータを操作してロボットを操作する．

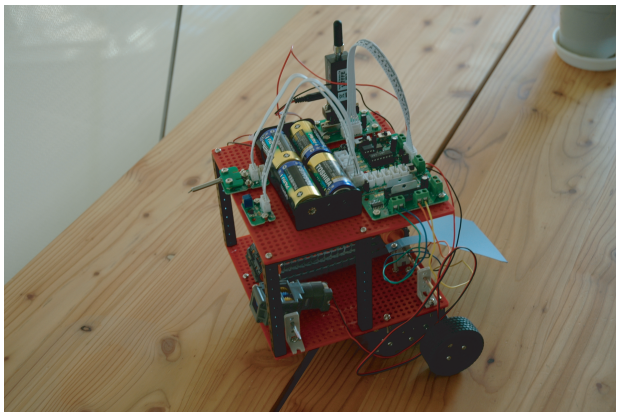


図 5.3: ロボット外観

5.2.1 ソフトウェア

センサノード配置ロボットのソフトウェア構成図を図 5.4 に示す．

センサノード配置ロボットのソフトウェアは，マイコン上で動くモータ制御機構と，ロボット操作 PC 上で動作するロボット操作ソフトウェアからなる．ロボット操作ソフトウェアが ROASEN フレームワークからセンサノード配置位置を受け取ると，ロボットに実際にセンサノードの配置を始めるよう指示する．まず位置管理モジュールが位置取得サーバからロボットの位置情報を取得する．次にセンサノード配置位置とロボットの位置を基にロボットの針路を決定する．ロボットの針路が決まると，ロボット通信機構に針路が伝えられる．ロボット通信機構はロボットと通信して進む方向を伝える．モータ制御機構はロボット通信機構から伝えられたロボットの針路に従ってモータを動作させる．ただし，ロボットの前方確認機構が前方に障害物を発見した場合，迂回動作をする．また，センサノード配置位置にロボットが到達した場合には，センサノード配置装置を動かしてセンサノードを配置する．全てのセンサノードを配置し終えると，ロボットは動作を停止し，次のセンサノード配置位置受信まで待機する．

5.3 実装環境

本節では ROASEN フレームワークとアプリケーションの実装環境について述べる．表 5.1 に ROASEN フレームワークの開発に用いた環境を示す．開発言語として C 言語と C++ を用いた．基本システム部分とユーザインタフェース部分は Microsoft Visual Studio を使って開発した．センサノードとの通信部分の実装では Cygwin と GCC を使った．

センサノード配置ロボットのマイコン上のソフトウェアは C 言語で実装した．また，ロボット操作ソフトウェアは C++で実装した．

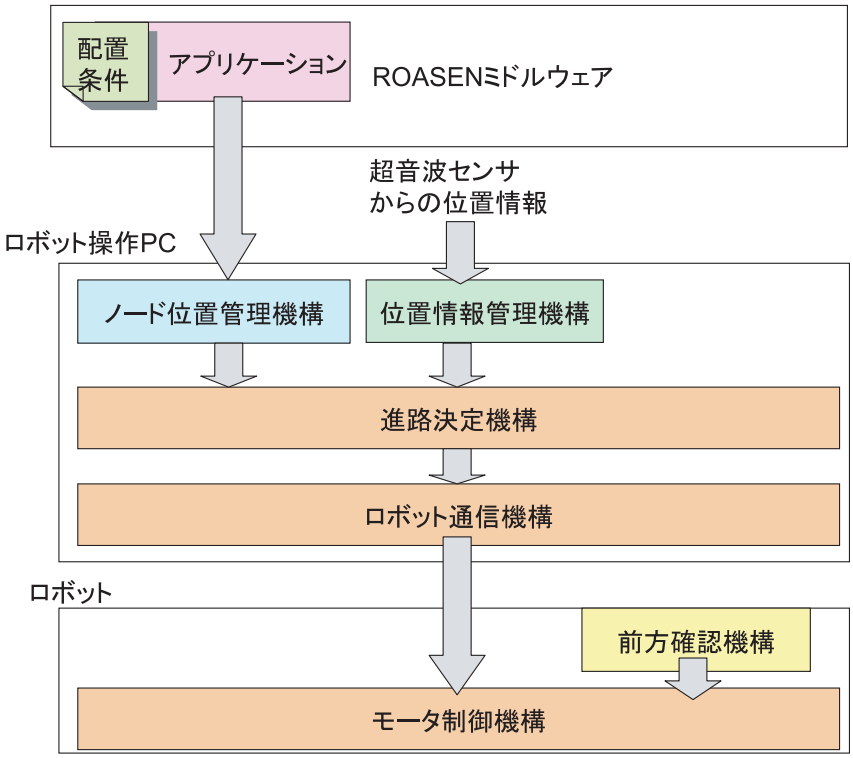


図 5.4: ロボットのソフトウェア構成図

5.4 本章のまとめ

本章では，ROASEN フレームワークを使ったアプリケーションと，ROASEN フレームワークのセンサノード配置を実現するセンサノード配置機構について述べた.

表 5.1: 実装環境

ハードウェア	Pentium M 2.4GHz
OS	Windows XP Professional SP2
言語	C，C++
開発環境	Visual Studio .NET 2003，Cygwin + GCC

第6章　検証と評価

本章では，ROASENシステムの検証と評価を行う．

6.1　定性的評価

本節では，センサノードの配置位置決定について，ROASEN フレームワークを用いた場合と ROASEN フレームワークを用いずに手動で配置する場合を比較する．比較項目として，アプリケーション適応性，無線到達性，配置柔軟性を挙げる．

6.1.1　アプリケーション適応性

ROASEN フレームワークを用いた場合，アプリケーション開発者は開発時のセンサノードに関する想定を配置条件として設定できる．ROASEN フレームワークはアプリケーションのセンサノード配置条件に沿うようにセンサノードの位置を決定する．手動でセンサノードを配置する場合，配置する人間がアプリケーションの要求仕様を全て理解していれば，アプリケーション開発者の想定通りにセンサノードを配置できる．しかし，実際には全てのユーザがアプリケーションに関する知識を持っているわけではなく，

6.1.2　無線到達性

ROASEN フレームワークでは，センサノードからの実際の無線の到達を確認しながら配置位置を決定する．また，無線が何らかの要因で不安定になった場合にも，配置位置を再計算する事で無線到達性を保てる．手動での配置の場合，人間が無線到達性を確認しながらセンサノードを配置するのは難しい．また，人間の場合，常に無線を確認するのは難しいため，通信が不安定になった場合に対応が遅れたり原因がわからなくなったりする．

6.1.3　配置柔軟性

ROASEN フレームワークを用いた場合，マップ情報やアプリケーションの配置条件，無線到達性以外の事柄を考慮に入れない．一方，手動でセンサノードを配置する場合，配置する人が自分で考えて，センサノードをなるべく人目に付かない所に配置したり，邪魔にならない場所に置く等ができ，配置に関する柔軟性が高い．

6.2　パフォーマンス評価

本節では，Roasen フレームワークのパフォーマンス評価を行う．まず評価に用いたシミュレータについて説明し，次に評価結果を示す．

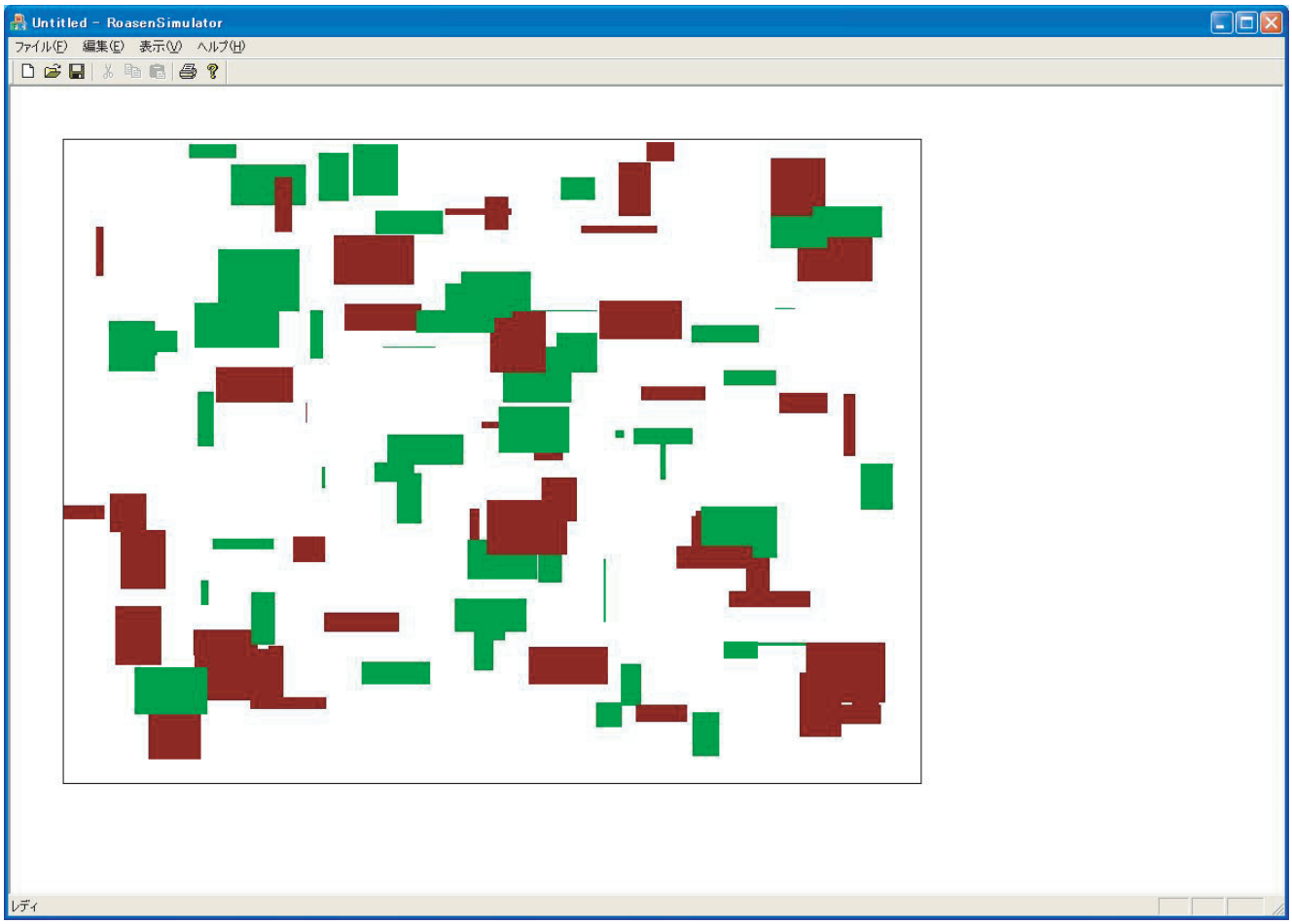


図 6.1: シミュレータ動作画面

6.2.1 シミュレータ

シミュレータの動作画面を図 6.1 に示す．画面では環境に存在する障害物や，センサノードを母点にしたボロノイ領域の表示等ができる．

シミュレータ上での実験として，環境内に障害物が多くなるとセンサカバー率がどのように変化するか計測した．

6.2.2 評価結果

評価結果のグラフを図 6.2 に示す．

6.3 本章のまとめ

本章では，ROASEN フレームワークの定量的評価とパフォーマンス評価を行った．

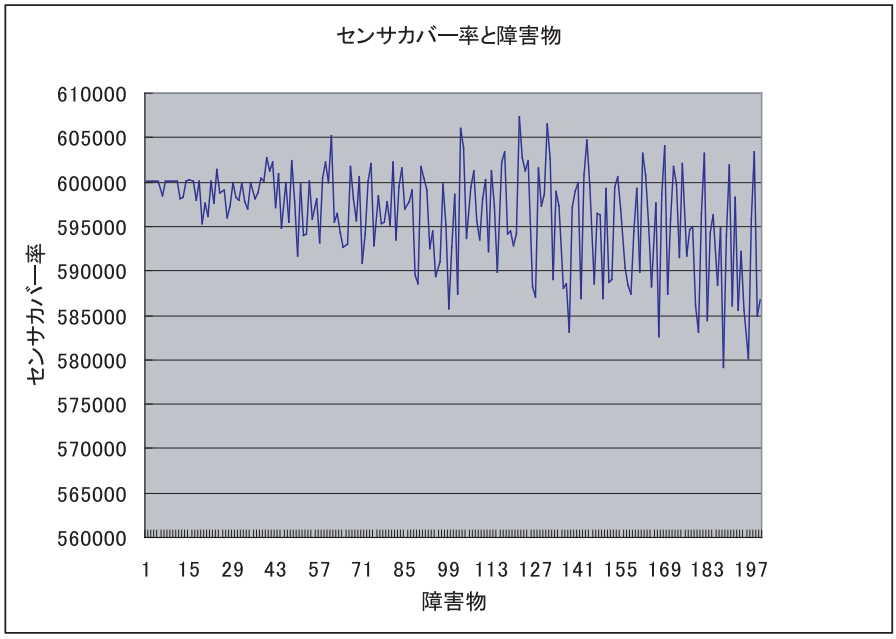

センサカバー率と障害物
610000
605000
600000
595000
590000
585000
580000
575000
570000
565000
560000
センサカバー率
1
15
29
43
57
71
85
99
113
127
141
155
169
183
197
障害物

図 6.2: 評価結果

第7章　結論

7.1 今後の展望

本節では，本研究のこれからの展望について述べる．

7.1.1 ロボットの改良

今回のロボットはセンサノードの配置のみを行うロボットとしたが，今後の展開としてセンサネットワークの展開から維持，回収までを行うロボットシステムを実装する予定である．センサノードは電池残量の低下や故障等の理由で動作を停止する場合がある．停止したセンサノードがマルチホップアドホックネットワークでの中継ノードとして機能していた場合，ネットワーク全体に影響を与える．そのためセンサネットワーク維持のために機能していないセンサノードを正常動作している物と交換する必要がある．また，学会や実験等で一時的に構築したセンサネットワークの場合，役割の終了とともにセンサノードを回収する必要がある．これらの作業は専門的な知識や多大な労力がかかり，人間が行うには負担が大きい．そこでロボットにこれらの作業を行わせることで，簡単にセンサネットワークの運営が行えるようになる．

7.2 まとめ

本稿では，アプリケーション適応的なセンサノードの配置を行うためのセンサノード配置条件抽象化機構を提案した．また，この抽象化機構を実際に運用するためのロボットとミドルウェアにを実装した．これまでの人手によるセンサノードの配置は専門家の知識を必要とし，アプリケーションに適応した配置を行う事も難しかった．ROASEN システムを使ってセンサノードの配置を行う事で，専門家でなくてもセンサネットワークの構築が可能になる．また，アプリケーションによって配置条件を変えるミドルウェアにより，アプリケーション適応的なセンサノードの配置を可能にする．アプリケーション開発者は XML でセンサノード配置条件を記述し，ROASEN システムはこの条件に合うようにセンサノードを配置する．アプリケーションの目的によって，センサノードの配置方法はオブジェクト監視型，空間網羅型，壁際型の 3 種類の方法を取れる．監視対象がどの程度重要かによってセンサノードの配置密度を指定できる．また，使用するセンサによってセンサノードの配置数を補正する．

謝辞

本研究の機会を与えてくださり，ご指導を賜りました慶応義塾大学環境情報学部教授徳田英幸博士に深く感謝いたします．また，重要な御助言を頂きました，慶應義塾大学政策・メディア研究科助教授高汐一紀博士の御指導に深く感謝致します慶応義塾大学徳田・村井・楠本・中村・南合同研究会の先輩方には折りにふれ貴重な指導と助言を頂きました．特に，徳田研究室の先生方や先輩方，ACE(Active Computing Environmets) 研究グループの方々に深く感謝いたします．また，駒木亮伯氏，大澤亮氏，青木崇行氏には絶えざる励ましや丁寧なご指導を賜りました．最後に，研究の日々を共に過ごした，鈴木慧氏，その他多くの友人に深く感謝し，謝辞と致します．

平成 19 年 1 月 24 日
今枝 卓也

References

Published Papers Related to this Thesis

- 今枝卓也, 鈴木慧, 大澤 亮, 神武 直彦, 高汐一紀, 徳田英幸
  “俺デスク:ユーザ操作に基づく参照履歴検索ソフトウェア, ”
  インタラクション 2006, Vol. 2006, pp. 219-220, May. 2005.

- 今枝卓也, 大澤亮, 高汐一紀, 徳田英幸
  “アプリケーション適応型センサノード配置ロボットの提案, ”
  情報処理学会第 12 回ユビキタスコンピューティングシステム研究会 (UBI), Vol. 12, pp. 29-30, Mar. 2005.

関連研究

[1] Crossbow. Mote. http://www.xbow.jp/motemica.html.

[2] University of Karlsruhe. upart. http://particle.teco.edu/upart/.

[3] Ubicomp 2005. http://www.ubicomp.org/ubicomp2005/.

[4] David Koizumi Matthew Lease Stefan B. Sigurdsson Kevin Sikorski Dieter Fox Gaetano Borriello Anthony LaMarca, Waylon Brunette. Plantcare: An investigation in practical ubiquitous systems. In *UbiComp 2002: Ubiquitous Computing : 4th International Conference, Goteborg, Sweden, September 29 - October 1, 2002. Proceedings*, p. 316. Springer Berlin / Heidelberg, 2002.

[5] 綜合警備保障. http://www.alsok.co.jp/.

[6] Sensor model language(sensorml). http://vast.nsstc.uah.edu/SensorML/.

[7] Ben Greenstein, Eddie Kohler, and Deborah Estrin. A sensor network application construction kit (snack). In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 69–80, New York, NY, USA, 2004. ACM Press.

[8] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Auton. Robots*, Vol. 13, No. 2, pp. 113–126, 2002.

[9] 村瀬正名, 西尾信彦, 徳田英幸. 引力・斥力モデルに基づいたセンサノードの動的再配置手法. 情報処理学会システムソフトウェアとオペレーティングシステム研究会論文集, pp. 31–38, 2003.

[10] Anthony LaMarca, Waylon Brunette, David Koizumi, Matthew Lease, Stefan B. Sigurdsson, Kevin Sikorski, Dieter Fox, and Gaetano Borriello. Making sensor networks practical with robots. In *First International Conference Pervasive 2002*, 2002.

[11] G.T. Sibley, M.H. Rahimi, and G.S. Sukhatme. Robomote: a tiny mobile robot platform for large-scale ad-hocsensor networks. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2002.

[12] M.B. McMickell and L.A. Goodwine, B. andMontestruque. Micabot: a robotic platform for large-scale distributed robotics. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, 2003.

[13] Mathew Laibowitz and Joseph A. Paradiso. Parasitic mobility for pervasive sensor networks. In *Pervasive*, pp. 255–278, 2005.

[14] JAPAN ROBOTECH LTD. Robodesigner. http://www.japan-robotech.com/index.html.

[15] 独立行政法人産業技術総合研究所, 古河機械金属 (株). Zps(zone positioning system). http://www.furukawakk.jp/products/ZPS1.html.