Vol.2014-DBS-160 No.10
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.10
2014/11/18

情報収集をするロボットプラットフォームの実現に向けて

中山 悟[†1]　柳田 将希[†1]　住谷 拓馬[†1]　中野 美由紀[†1]　菅谷みどり[†1]

近年，安価なロボットが利用可能になり，多数のロボットを利用したサービスも検討されるようになってきた.ロボットは異なる物理条件下で動作することから,その動作や開発時の記録を逐次ログすることが求められる．しかし，現状ロボットを対象として，データを収集する場合には，データ収集のための API の設計及びプラットフォームが十分整備されておらず情報収集がしづらい問題がある．本研究では，汎用ロボットの基礎的なロギングのためのプラットフォーム RLS(Robot Logging System)を設計，実装し，その仕組みを利用してデータ収集を行った内容，および，今後のロボットの情報収集のプラットフォームについて議論した内容を報告する．

Toward to develop a platform for collecting logs generated on robot

SATORU NAKAYAMA[†1]　MASAKI YANAGIDA[†1]　TAKUMA SUMIYA[†1]
MIYUKI NAKANO[†1]　MIDORI SUGYAYA[†1]

In recent years, inexpensive robot becomes available, since services using a large number of robots have also come to be discussed. Since the robots operate at different physical conditions, it is required to sequentially log the operation and record during development. However, at present, in the case of collecting the data from the robots, since the platform for data collection has not been fully developed, there is a problem information collection is difficult. In this paper, we conducted design and implementation of the platform for basic logging of general purpose robots. We report contents of performing data collection by using the mechanism, and what has been discussed about future of platform of information collection of robots.

1. はじめに

近年，安価なロボットが利用可能になり，多数のロボットを利用したサービスも検討されるようになってきた．特に，災害時のレスキューや燃料補給などは緊急性を必要とされるため複数台で行う事が求められる[1].また，Pepper など人間とのコミュニケーションを目的としたロボットが提供されている[2].こうしたロボットは，フロア案内をするなどにより，人的資源が限られた店舗において，待ち行列の緩和などが期待できる．このように，複数台のロボットを用いたサービスは様々な場面での利用が期待される．

複数台のロボットを利用したサービスにおいては，サービス提供者側は，個々のロボットを管理するために，その情報収集が必要となる．ロボットは，自律制御がある，ないに関わらず，移動や会話など，動的に変化する環境への変化が求められることから，その目的の達成についてデータを収集して確認するという要求は，システム構成上自然な要求である．

例えば，エンコーダ情報による移動距離，GPS による位置情報などの情報で位置推定，衝突回避などの場合には，障害物の検知情報などは，一般的にロボットを開発する際に最初に取得される情報である．また，ロボットと人間のインタラクションを前提とする場合には，まずは，人がどのようにロボットとインタラクションをするのか，など，未だロボットが十分普及していない段階では，こうした情報の収集が重要である．このように，ロボットから情報を収集する動機は様々であるが,その要求は確実に存在する．

我々は，現在ロボットを大学のプログラミングの授業に導入し，プログラミングの動機付けや意欲向上に役立てようとしている[3].ロボットを動かすために学生がどのような取り組みを行ったかを明確にする事で，学習の実態や，その効果を計ることができる．そのため，ロボットからのログをいかに収集するかは重要な課題である．

しかし，ロボットを対象として，データを収集する場合には，データ収集のためのプラットフォームにおける問題点がいくつかある．ロボットは計算機のように，ハードウエアプラットフォームや，ソフトウェアプラットフォームが統一されていない．そのため，統一的な形でログを収集することが困難である．また，オペレーティングシステムや，ミドルウエアなどが統一されていないことから，その機能を利用したログ取得の汎用的な仕組みが十分提供されていない問題がある．

また，ロボットは，手や足などの複数のアクチュエータを同時に動かす必要があることから，プログラムで動作させる場合には，これらのプログラムの同期などを操作する必要があり，そのプラットフォームとして，RT ミドルウエア[4]や ROS (Robot Operating System)[5]が提供されている．しかし，こうしたミドルウエアはロボット同士の通信や，ログをサーバに転送する機能などが十分に提供されていない．特に，我々が利用している iRobot Create では，

†1 芝浦工業大学　情報工学科
Shibaura Institute of Technology, Information Science and Engineering

Vol.2014-DBS-160 No.10
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.10
2014/11/18

メインはシリアル通信となっており，これを拡張して，サーバにログを収集する仕組みとするためには，TCP/IP ベースのソケット通信を可能とする必要がある．近年では，無線のシリアル通信も可能となっているが，遠距離までの無線通信を考慮した場合，汎用の TCP/IP プラットフォームがあることが望ましい．このように，ロボットプラットフォーム実現のためには，ログを取得する仕組みのみならず，汎用通信のプラットフォームの実現が必要である．

本研究では，ICT の観点で実際の調査が少ないロボットのための共通のプラットフォーム開発に向けて，二つの試みを行った内容を報告し，今後のプラットフォームの設計の議論を行うことを目的する．

本論文の構成は，以下の通りとする． 2 節にて，既存研究を示し，3 節にて，ロボットによる情報収集の課題を示し，4 節にてロボットを用いた情報収集プラットフォームの提案，5 節で評価，6 節でまとめとした．

2. 既存研究

既存のロボットのためのプラットフォームは主に，機械/電気系などハードウエアの分野から提案として RT ミドルウェア (RTM)[4]がある．RT ミドルウエアは，複数のソフトウェアモジュールを組み合わせてロボットシステムを構築する為のソフトウェアプラットフォームの標準規格である．本規格では，コンポーネント間で情報をやり取りするためのインターフェイスが提供されており，この規格に準拠するコンポーネント間であれば通信が可能となる．しかしながら，複数のデバイスを連携させるために重要となるソフトウェア間での通信と情報共有の機能に関しては，CORBA を利用することだけが示されているものの，その下の通信の仕組みについては，具体的にされていない．

一方，OSS ミドルウエアとして ROS (Robot Operating System)[5]が提供されている．ROS の通信は，メッセージパッシングを用いたプロセス間通信を利用し，プロセス間での情報共有のために， publisher/subscriber モデルを利用する事で，複数のデバイス間で情報を共有しやすい仕組みとなっている．しかし，RT ミドルウエアと同様に，その基礎となる通信部分の実現方法については，デバイス依存となっており，汎用の TCP/IP を利用したソケット通信などを利用したい場合には，開発者が準備する必要がある．

一方，ロギングの仕組みは，こうしたソケット通信や，それをもとにしたサーバクライアントの仕組みを前提とした様々な手法が提案されている．例えば， UNIX マシンであれば，syslog-ng[6]などが利用できる．syslog-ng は，システムメッセージをネットワーク上で転送，あるいは，ファイルに記録する仕組みである，syslog の実装の 1 つである．出力するログのフィルタリングや，ローテート，出力先ディレクトリの自動作成，ログにメタ情報などを表すマクロが使用できる．

さらに，Fluentd は，インターフェイスの記述を Ruby とすることで，よりログ収集をしやすくすることを目指している[7]．オブジェクト志向での記述が可能であるため，ログを収集する際には， Input，Buffer，Output の 3 つのコンポーネントを利用したクラスで様々なログの入力(Input) や出力(Output) に対応できるものとしている．また，収集したログを JSON 形式に変換し，蓄積した後に様々な出力先にデータ出力することで，オンメモリの処理をしやすくすることができる．このようなユーザビリティの向上から，近年多くのシステムで利用されるようになっている．しかし，一方，こうしたツールは，ロボット上でログを収集する際には，ソフトウェアの規模が大きく，複雑である問題がある．本研究では，これらのツールが前提とする TCP/IP 通信を可能とし，また，今後こうしたロギングシステムとの統合を考慮した，ロボットの API ログに着目した情報収集プラットフォームを提案する．

3. ロボットによる情報収集の課題

我々は実際にログを収集する中で，ロボットのログ収集に関しての課題を次の 2 つにまとめた．それぞれの課題について述べる．

1. **ロボット構成を反映した仕組み**

一般的に，ロボットはモータなどの制動部と，制御のためのコントローラが分かれていることも多い．直進や回転などの基礎的な動作や自己位置推定を行う場合，制動部への命令，また，制動部がその結果に動作した結果などの情報を観察することで，動作が正しく行われているかを観察することができる．

ロボットの情報を収集するためには，このように，ロボットのハードウエア構成を考慮したロギングの仕組みが必要となる．

2. **TCP/IP 通信のプラットフォーム**

ロボット同士の通信制御を行う場合に，従型のシリアル接続は一般的である一方，無線を介した TCP/IP 通信は，まだロボット通信には一般的には用いられていない [8]．例えば，API が公開されている iRobot Create 社のロボットでも，シリアル通信は起動直後に行うことができるが，TCP/IP をベースとしたソケット通信の環境は提供されてない[9]．今後，情報系の開発者がロボットを用いたネットワークをより簡易に構築するための基礎的な通信プラットフォームの提供および，遅延調査が必要である．

4. 提案

4.1 研究の目的

3 節で示した 2 つの課題をふまえて，本研究では，次の設計方針により，情報収集を行うための仕組みをロボット上に設計，実装するものとした．

Vol.2014-DBS-160 No.10
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.10
2014/11/18

1. ロボットの構成を考慮した仕組み
2. 汎用の TCP/IP 通信プラットフォーム

以上の機能を実現するロボット向けロギングシステムを Robot Logging System (RLS) とする.

1 を実現するために，ロボットの制動部に対して命令を行うロボットライブラリから直接ログを収集する仕組みを設計する. これにより, ロボットを動かすための命令(API) の情報を収集することができるものとする.

2 を実現するために, TCP/IP 通信を Raspberry Pie, ODROID といった組込みシステムでも動作可能である簡易なソケット通信をもとにした，マルチタスクサーバを提供する．以上の方針に従い，設計，実装するものとした.

4.1 システム構成

本 RLS の全体システム構成を図 1 に示した. ロボットは, 制動部の制御の計算を計算機と接続して行うことが一般的である. そこで, 今回はまず, ロボットを計算機と接続し，計算機側で情報を収集し，処理する役割を担わせ，そこから命令を発してロボットを動作させる構成とした．また，この際に，ロボットは主に計算機とシリアル接続していることを前提とした．データの送受信は，ロボットのログを集約するサーバーを想定し，そのデータをサーバに集約するものとした.

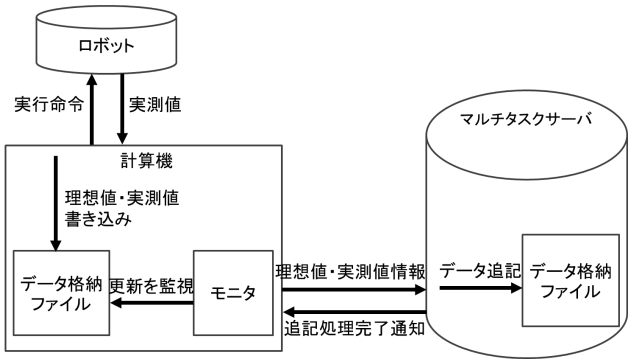

図 1　RLS のシステム構成

Figure 1　System Overview of RLS

今回，ロボットとして iRobot 社の iRobot Create を用いた.iRobot Create は, 開発可能なライブラリと簡易な移動体ハードウエアを提供しており，プラットフォーム開発に適していると考え，採用した．システム構成においては，計算機とシリアル通信を行うものとした.

通信では，計算機側から，実行命令，また，ロボット側からは実測値を得るものとした．ロボットは，計算機から送られた命令実行後に，その実行時間（日時）とともに，実行命令内の理想値と，実測値を計算機内のログファイルに保存する．ここで，理想値とは，実行命令である API の引数であり，実測値とは，ロボットが実際に命令実行した後に返されるセンサ値である．次に，ログの更新を監視するモニタアプリケーションは，更新されたログファイルの日時と内容を監視し，更新され次第，書き込まれたデータを TCP/IP 通信でサーバに送信する．複数台からのデータを受け取ったサーバは，それぞれでサーバ内のファイルにデータを保存する.

4.2 API 設計と実装

ロボットの動作を観察するためのログの収集にあたり，iRobot Create のオペコードを C 言語で隠蔽するライブラリである COIL（Create Open Interface Library）を使用することで，POSIX 準拠の OS において C 言語でプログラムを記述しロボットを動作させるものとした[10].

ログ出力の際には，API のみで物理的な動作ができるだけ明確にわかることが重要である．COIL で提供されている API は，汎用性が高い一方，初心者が動作させる場合には，物理的な動作がイメージしづらいものであった．そのため，本研究では，ログに取得することも考慮して，API の拡張を行った．拡張した結果は，表 2 にまとめた.

表 1 ：設計した API とその役割

関数名	動作
connect/disconnect()	iRobot Create との通信を確立，または，切断する.
forward/backward(short velocity)	一定の速度(velocity)で前進, または，後退する.
wait(short time)	指定した時間(time)だけスリープする. 停止目標を設定していないコマンドの次に実行する.
forwardDistance/backwardDistance(short velocity , int distance)	一定の速度(velocity)で指定した距離(distance)前進，または，後退した後に停止する.
turnAngle(short velocity, int angle)	一定の速度(velocity)で指定した角度(angle)回転した後に停止する.
turn(short velocity, short rad, int angle)	一定の速度(velocity)と回転半径(rad)で指定した角度(angle)回転した後に停止する.

ここで，velocity は 0-500 の値の範囲を取る．これは，もとの COIL の API の値の範囲では, -500-500 であったが，負数を取らないようにする代わりに，前進と後退それぞれ別々の API を用意することで正数のみを取るものとした. を改変せずに利用するものとした. 同様に, また, distance , angle, rad は，もとの COIL の API の値の範囲を改変せずに利用するものとした．取るものとした.

実装した API と，元にした COIL の API との相違点は下記の通りである.

Vol.2014-DBS-160 No.10
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.10
2014/11/18

表２：元の COILAPI との比較

実装した API	元の COILAPI	変更点
connect/disconnect()	startOI_MT/stopOI_MT(const char* serial)	引数をそれぞれの計算機のシリアルデバイスファイル名で指定する形式であったが，計算機ごとに固定値を設定し，ユーザーはファイル名を気にせずアクセスできるようにした.
forward/backward(short velocity)	drive (short velocity, short rad)	直進のみを扱うようにした．また，前進と後退で別々の API にした.
wait(short time)	waitTime (double time)	今回の競技では，長時間の指定が不要であるため，単純化のために short 型を引数にとるようにした.
forwardDistance/backwardDistance(short velocity, int distance)	driveDistance (short velocity, short rad, int distance, int interrupt)	直進のみを扱うようにした. 前進と後退で別々の API にした. ロボットを持ち上げた際に API の実行を停止するよう固定した.
turnAngle(short velocity, int angle)	－	directDrive(short velocity, short velocity)を使って新たに定義した．左右のホイールを同じ回転数にした.
turn(short velocity, short rad, int angle)	turn (short velocity, short rad, int angle, int interrupt)	回転する半径と角度により，進行方向を変えるようにした. ロボットを持ち上げた際に API の実行を停止するよう固定した.

このように，COIL 本来の API 設計を修正することで，開発時の内容の詳細が把握しやすく，また，本 API を利用する側もより書き易くすることを目指した.

4.3 API のログ収集の仕組

4.2 節で示した API のログを収集するにあたり，さらにロボットが実行した API を収集するために，下記のログの API を設計，実装した.

・logFunc(const char* funcName, const int argNum, ...)
・logSensor()

logFunc()は，funcName で呼び出された関数名，argNum で引数の数，その後に引数として指定された実際の値を出力する.API 名の取得にはマクロ__func__を使用することで開発者は,呼び出す API を記載しなくても良いようにした. また，動作指示の命令を与える API を実行した結果，ロボットが動作した場合，その実測値を得るために logSensor()を設計した.. logSensor()は，物理的な動作を行う命令の実行前と，実行後に呼び出すことで，その物理的な変化を実測値として出力する．iRobot Create には様々なセンサが搭載されており，現状はバッテリ残量，距離，角度，速度，回転の半径，過電流，バンパーとホイールの状態の値を取得できる．また，それぞれのログ API 内で，タイムスタンプを使い，動作指示の命令が実行された日時を取得する.

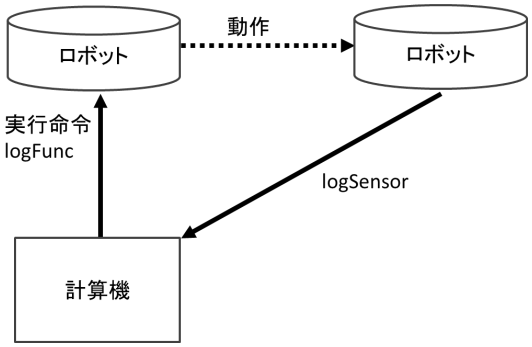

図 2　ログ関数
Figure 2　Logging functions

4.4 マルチタスクサーバの設計

マルチタスクサーバの設計，実装に先立ち，ソケット通信プログラムを作成し，複数台からの処理要求を想定した実験を行った．実験では，複数のデータ送付元が同時に，サーバにデータを送付する実験を行った．サーバは，先に到着した処理要求に対する処理が終了するまで，その後に到着したクライアントからの要求に応じないことが確認された．このブロック時間は，ログの大きさが大きくなるほど長くなるという一般的な現象を確認することができたため，サーバ側の処理を，その都度プロセスを生成するマルチタスクにすることでそれぞれの要求の同時処理を可能とするものとした．尚，これらは実行環境として Linux OS である，Ubuntu 12.04 を使用した.

Vol.2014-DBS-160 No.10
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.10
2014/11/18

5. 評価

5.1 ロボット API のログ収集の有効性の評価

5.1.1 概要

評価にあたり，我々は複数台のロボットから，ロボット API を利用して収集したログの有効性を調査する方法を検討した. 一般的に, ロボットを動作させるための開発では，基本的な動作を実現するための API を組み合わせて実行する．一般的な PC 上でのプログラミングとロボット上でのプログラミングの差異は，ロボットの個体差や物理条件に合わせてチューニングを行うことである．

そこで，我々はロボット開発者が，どのようにチューニングを行うのか，API のログを解析することにより，明らかにすることを目的とした．そのためには，ある程度の開発者が同時に開発する環境でログを取得することが望ましく，それを実現できるロボット演習の実践授業への適用を行うものとした．

5.1.2 実施方法

まずは，ロボットプラットフォーム上で，開発者（ここでは授業受講者）に，API を利用した開発を行ってもらうために，

(1)共通の課題を作成し，その課題をもとに開発してもらう
(2)課題で出した内容をどのように開発で実施したかを確認するために，API のログをプラットフォームから収集する．
(3)収集した結果を解析し，有効性について議論する

以上の３つの手順で，ログの有効性の評価を行うものとした．以降の節にて，それぞれ(1),(2),(3) の実施内容について述べる．

5.1.3 課題内容

実施方法の１）に示したように，共通の課題を設定する必要がある．短時間で成果を得るために，開発を競技形式で進めるものとした．対象が，熟練の開発者ではないことから，比較的単純な問題設定とし，短時間で効果を得ることを目的とし，チキンレースは，前進（forwardDistance 関数）のみを使って，可能な限り 2m に近い位置でロボットを停止させる課題である．

ロボットの動作の中でも，直進は，最も単純でありながら，地面との摩擦や，重量といった環境は個体差の影響があるため，指示した通りのパラメータで正確に動作させることが難しい．

開発者は，正確な値を設定できるように試行錯誤する必要があるが，こうした過程において，どのような API を提供することが望ましいのかを詳細に観察することができるように, 拡張 API を提供し, 差分を調査できるようにした．また，壁との距離を最小まで縮めるためには，発進直後に徐々に加速し，停止直前に徐々に減速するようにプログラミングすることが望ましい．これはロボットが前進する際に最大スピードで最大距離進んだ場合，停止した際の誤差が大きくなる問題を回避するために必要となる実装である(図３).

しかし，対象となる学生は，必ずしもロボットプログラミングに慣れているとは限らない．そのため，ヒント無しで，こうした減速と加速を含むプログラムを書く事は難しい．そのため，競技として提示したチキンレースにおいては，図 4 に示すサンプルプログラムを示し，学生がチューニングの参考にできるようにした．

サンプルプログラム（図 4） では，まず，少しずつ加速を行うために，while ループ内で 初期速度 200 から最高速度 300 になるまで速度を 50 ずつ増やしながら，距離 100 ずつ前進する．ここで，初期速度と最高速度は，実測し，誤差が小さく，かつ，滑らかに走行する速度を選んだ．次に，停止前に少しずつ減速を行うために，while ループ内で最高速度 300 から速度 0 になり停止するまで速度を 50 ずつ減らしながら，距離 100 ずつ前進する．

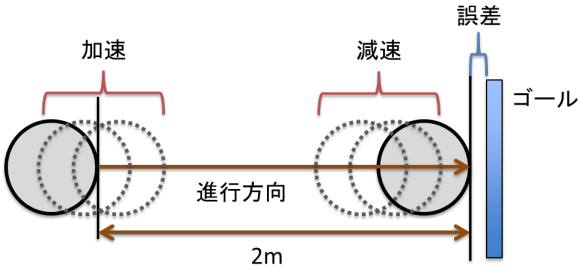

図 3　チキンレースの概要
Figure 3　Description of Chicken-race

```
#include <createoi.h>
#include <stdio.h>
void int main() {
  int velocity = 0; //速度
  connect(); //通信開始
  // 加速処理
  while (velocity < 300)
  //速度がある値になるまでループ
  {
    if (velocity == 0) {
      velocity = 200;   //初期速度を設定
    } else {
      velocity += 50; //2回目以降, 加速
    }
    forwardDistance(velocity, 100); //前進
  }
  // 前進
  forwardDistance(velocity, 1400); //前進
  // 減速処理
  while (velocity > 0) //速度が0(止まる)まで
ループ
  {
    velocity -= 50;
    forwardDistance(velocity, 100);
  }
  disconnect(); //通信終了
}
```

徐々に加速

徐々に減速

図 4　サンプルプログラム
Figure 4　Sample program

5.1.4 実施概要

課題にあるプログラムを参考にして，実際に複数人にロボット上での開発を並行的に実施するものとした．実施にあたっては，開発ログを収集し，これらのログを比較検討することを目的とした．

複数人による同時開発を実現するために，ロボットを用いたプログラミングの授業で，5.1.3 節に示した課題を実施し，ログを収集するものとした．授業は，芝浦工業大学,

Vol.2014-DBS-160 No.10
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.10
2014/11/18

工学部，情報工学科，プログラミング演習 I の実習において 1 年生 108 人で，約 10 人ずつ 10 グループに分けて実施した．また，比較対象ために，同情報工学科，3 年生の別の選択必修の授業にて約 80 人を 10 人ずつ 8 グループに分けてそれぞれ実施した．

はじめに，ロボットを動作させるために必要な API を一通り学習させ，各グループにロボットと開発環境をもつパソコンを用意し，5.1.3 節の課題を作成させた．プログラムをコンパイルするごとにログを出力し，ファイルに追記処理した．

図 5　出力されたログ

Figure 5　Output log

5.1.5 分析結果

各グループから得られたデータの中から，直進の加速，減速を想定したチキンレースで使われた forwardDistance() 関数の速度パラメータを抽出し，コンパイル時間ごとの推移を図 6,7 に示した．

図 6 は，ある固定値で単純な試行を行ったグループの内から，また，図 7 は，図 4 のサンプルプログラムに習って，加速と減速を用いて試行を行ったグループの内から，最も典型的なものを選んだ．また，図 6 において，ロボットの速度は 0-500 の値の範囲であるが，グループによっては，速度 1000 で試行を行っているものもあった．

ログからは，forwardDistance()関数を 1 つだけ使ってプログラム作成したグループと，forwardDistance()関数を複数使って加速と減速をするプログラムを作成したグループの 2 パターンに分類する情報を得ることができた．そのうち，加速と減速をするプログラムを作成したグループは，1 年生では 10%，3 年生では 37.5%となった．このように，ログを解析することにより，開発過程に差があることが分かった．今回は，情報工学科 1 年生と 3 年生に同じ課題を出したことから，その開発にあたり，示されたサンプルを解釈実行する開発能力に差があったといえる．このログの分析により，学部の教育の有効性について議論することも可能であるといえる．

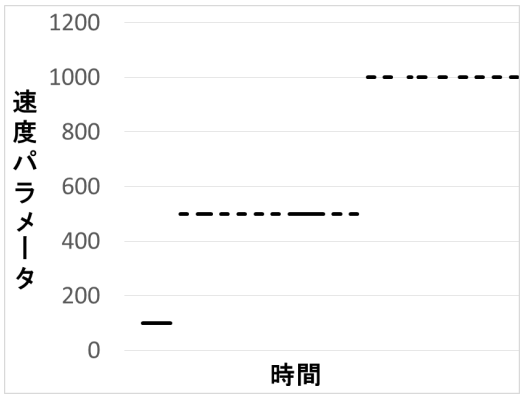

図 6　減速加速を考慮しないプログラムの開発過程

Figure 6　Simple program

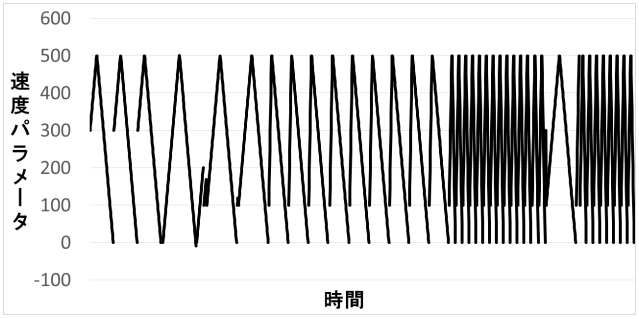

図 7　加速・減速を用いたプログラムの開発過程

Figure 7　Program using the acceleration and deceleration

5.1.6 考察

API のログを解析することで，プログラムの開発過程を知ることができることが分かった．このことから，ログを取得し，開発過程を理解することで，異なる物理条件下におけるロボットのチューニング過程を記録し，共有することができると考えられる．

5.2 ロボットにおける TCP/IP 通信の基本性能の調査

3 節に示したように，ロボットを動作させる環境では，現状主にシリアルが用いられている[8]. 内村らは，今後ロボットが自律的に動作するためには，有線を主とした通信方式ではなく，無線通信を考慮する必要があるとしている．無線通信の実現方法には，より多くの帯域幅を利用することを前提とした，TCP/IP 通信がある．TCP/IP 通信では，チャネルの確立と，データ通信というシリアル通信からするとより複雑なプロトコルにより，信頼性の高い通信を可能としている．我々も，iRobot Create を複数台動作させるにあたり， TCP/IP の無線命令でログ収集や命令を行うことを検討する必要があると考え，プラットフォームとしての設計，実装に取り組むものとした．

しかし，iRobot Create をはじめとする多くのロボットには，TCP/IP 通信を行うためのミドルウエアが十分ではない．そこで，本研究では， TCP/IP 通信を行うためのシステム基盤であるソケット通信を設計，実装し，簡易的な評価を

Vol.2014-DBS-160 No.10
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.10
2014/11/18

行うことで，その実用性について議論するものとした．

調査では，従来のシリアル通信に対して，TCP/IP 通信を行った場合の速度を比較することで，どの程度の速度比があるのかを明からにすることを目的とした．

まず初めに，ソケット通信環境をロボットに実装した．ロボットには，命令を送る側，受信した命令を実施する側がある．これをサーバ，クライアントプログラムに当てはめるためには，命令を送る側と，実行を行う側を最初に区別して，役割を割り振る必要がある．我々は，クライアントから開始する，クライアントサーバーのモデルに従い，まずは,クライアントが準備完了のパケットを送付した後，サーバ側が命令指示側として，クライアントに必要なデータを送付するものとした（図 8）．

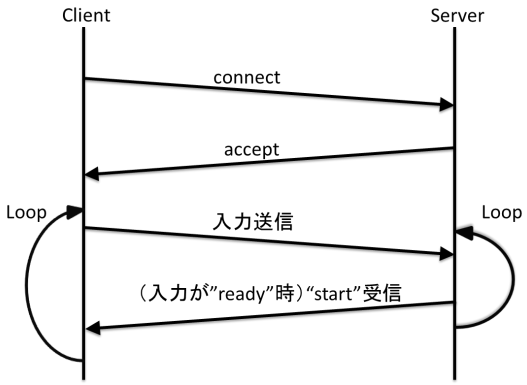

図 8　サーバクライアント通信プログラム
Figure 8　Server-Client program

これに対して，通常のシリアル通信は，特に区別しなくても,命令の送信側と受信側で役割分担がなされているため，このままの形式を用いた．実装したプログラムとシリアル通信のみを実装したプログラムを用いて，以下の実験を行った．

(1)標準入力から文字列を取得
(2)取得した文字列が”ready”の場合”start”に置換
(3)文字列が”start”ならロボットを前進

クライアント側の入力送信後からロボットが前進し始めるまでの時間をそれぞれ 10 回計測した結果を表 3 に示した．尚，本実験は Intel Corei7，2.5GHz，2GB メモリのマシンで実施した．

表 3　実験結果
Table 3　Experimental results

	シリアル通信のみ	C/S通信とシリアル通信
平均	265.5	355.9
最大	624.6	967.2
最低	32.9	42.0
標準偏差	224.7	287.7

表 3 の実験結果から，サーバクライアント通信をロボットに移植したときの平均遅延時間は 90.4msec であった．双方向の通信を行う TCP/IP のオーバーヘッドが 1/2 以下であることから，ロボットプログラムの命令実行における TCP/IP 通信が利用できるレベルにあることを示している．オーバーヘッドを値のみからだけで判断することは困難であるが，TCP/IP を用いたサーバクライアントモデルをロボットに応用することは，現実的な選択肢となりうることを示した

6. 議論

最後に，これまでの取り組みの中で，今後も引き続き，必要な項目を議論としてまとめた．

1.　API のロギングについて

PC とは違って，物理条件を反映する必要があると考え，ログを収集するプラットフォーム開発を行った．今回の場合，ログを収集した結果は，教育の現場での知見としてフィードバックすることができることが分かった．実際のロボットは様々な物理条件やサービス要求に答える必要があり，そうした場合のログを収集するには，開発 API のみならず，センサ情報を収集する必要があると分かった．発行した命令と，その結果の実測値により，より，詳細に移動ロボットの移動や運動などを把握することができ，レスキューや，自律制御に利用できることが分かった．

2. ロギング情報のリアルタイム性

サービス要求に対して，リアルタイム性が要求されることは自明である．例えば，自然な会話に期待される応答性は 1 秒程度である[12]．これに対して，ログ収集においてはリアルタイム性を阻害しないことが求められる．ロギング時にはシステムに必要なリアルタイム性を調査し，それを阻害しないような周期でのログの収集が必要である．

3. 通信とクラウドの接続ついて

従来，ロボット同士，または，ロボットからの通信はシリアル通信が主であった[8]．しかし，内藤らが述べているように，今後は TCP/IP のソケット通信が重要である．特にログの量は，高々数 kb と少なかったが，今後センサ情報や，人間とのインタラクションなど，ロボットが物理条件下で動作した記録を取り，それを収集することを考えると，より多くの帯域幅，汎用的な通信手段を備える必要がある．今回，我々はシリアル通信と，ソケット通信の比較を行い 30%程度の遅延差で実現できることが分かった．ログの量が増大するにつれて，遅延が増大することは考えられるが，汎用方式を利用した通信をプラットフォームに実現することで，より多くのログを近距離ではなくクラウドなどのサーバに送ることを検討することができる．

7. まとめ

ロボットを対象としたデータ収集のためのプラットフォ

Vol.2014-DBS-160 No.10
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.10
2014/11/18

ームの実装における汎用性の問題点を述べ，プラットフォームを開発するに当たって要求される点として，ロボットのハードウェア構成を考慮したロギングの仕組みと，TCP/IP ベースの通信制御を挙げた．ロギングの仕組みについては，ロボット演習の実践授業で実験を行い，有効性が示せた．通信制御については，シリアル通信と TCP/IP 通信の速度差から，大して遅延しないことが分かった．今後，ログの収集時間を評価し，リアルタイム性の向上を行う．また，クラウドとの連携を検討する．

参考文献

1) レスキューロボットコンテスト
http://www.rescue-robot-contest.org/
2) ソフトバンク: Pepper ロボット
http://www.softbank.jp/robot/
3) , 情報処理学会特集号,
http://www. oss-og. co. jp/service/omamori/index. html
4) Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng: ROS: an open-source Robot Operating System.
5) . Ando, T. Suehiro, K, Kitagaki, T. Kotoku, and W.-K. Yoon, “RTmiddleware: distributed component middleware for RT (robot technology),” Proc. of IEEE/RSJ Int. Conf. on robots and intelligent systems, pp. 3555-3560, 2005.
6) Official Syslog-ng site:
http://www.balabit.com/network-security/syslog-ng/.
7) ″Fluentd の仕組み -バッファ機能でログ収集漏れを防ぐ-″. Tech-Sketch.
http://tech-sketch.jp/2013/05/fluentd-buffer.html, (参照 2014-10-25).
8) 内村裕, ロボット制御の視点で見た無線通信の課題と期待. 情報通信学会. 2013
9) iRobot Create OWNER’S GUIDE. iRobot Corporation. 2006
10) iRobot Corporation. iRobot Create OPEN INTERFACE.
http://www.irobot.com/filelibrary/pdfs/hrd/create/Create%200pen%20Interface_v2.pdf, (参照 2014-10-25)
11) BalaBit IT Security Ltd.. The syslog-ng 3.0 Administrator Guide. Seventh Edition, 2009 年.
https://www.icts.uiowa.edu/confluence/download/attachments/28049747/syslog-ng-v3.0-guide-admin-en.pdf?version=1, (参照 2014-10-25)
12) Toshiyuki Shiwa, Takayuki Kanda. ″How Quickly Should Communication Robots Respond?″. HRI ’08 Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction. 2008 年, 153-160.
13) 野呂影勇, 人間工学における反応時間の測定と結果の解析, 人間工学, 21 (2), 65-70, 1985.