Vol.2019-DPS-178 No.27
Vol.2019-CSEC-84 No.27
2019/3/5

# IoT プロトタイピングのためのリモート I/O 機構の開発

林勇佑[†1]　早川栄一[†2]

**概要**：IoT のプロトタイピングではハードウェアとソフトウェアの両方をプロトタイプする必要がある．ハードウェアのプロトタイプでは，目的に応じたセンサの回路を取り付ける必要があるが，ハードウェアの組み立ての問題でセンサを取り付けることが困難であるという問題がある．ソフトウェアのプロトタイプでは、サンプルコードを容易に利用できることと、ハードウェアに応じたソフトウェアの開発が必要である。にそこで本研究では，Raspberry Pi のリモート I/O 制御を行うサーバとライブラリの開発を行う．これにより回路部分をクライアントから独立させ，新たにハードウェアを用意することなく，様々なクライアントで素早くプロトタイピングを行うことを可能とした．

**キーワード**：IoT，プロトタイピング，リモート I/O

# Development of a remote input/output mechanism for IoT prototyping

YUSUKE HAYASHI[†1]　EIICHI HAYAKAWA[†2]

***Abstract***: Prototyping IoT requires prototyping both hardware and software. In the hardware prototype, it is necessary to attach a sensor circuit according to the purpose, but there is a problem that it is difficult to attach the sensor due to the problem of hardware assembly. In software prototypes, it is necessary to easily use the sample code and to develop software according to the hardware. In this research, we develop servers and libraries that present remote I / O control of Raspberry Pi. This makes the circuit part independent from the client, and enables rapid prototyping with various clients without preparing new hardware.

***Keywords***: IoT, prototyping, remote I/O
[**] 日本語キーワード，英語アブストラクト，英語キーワードの記載はオプショナルである．

## 1. はじめに

Internet of Things（以下 IoT）のシステム開発を行う上で，ソフトウェアの開発だけでなく，ハードウェアの開発を行うことが必要である．そのため，IoT のシステム開発では完成品の作成を目指す前にプロトタイピングを行うことで，ソフトウェアとハードウェアがそれぞれ仕様通りに動作することが可能であるかの確認のすることが大切となる．このプロトタイピングを通して，システムの課題の発見や，新たなアイデアを生み出すことにも繋がっていく．そのため，このプロトタイピングの作成を素早く行っていくことで，早期に課題を発見することや，アイデアをより多く生み出すことができる．

しかし，IoT のシステム開発では，ソフトウェアの開発だけでなく，ハードウェアの開発を行う必要があるため，プロトタイプを作成する時間も長くなってしまう．ハードウェアの試作を行う際には，ブレッドボードを利用することが多く，そのケーブル数はセンサの数に応じて増加していくため，ハードウェアの取り回しが困難になるといった問題も発生する．

IoT のシステム開発を行うにあたって，安価でセンサを用いたプログラミングが簡単に行えるコンピュータとして，Raspberry Pi は様々な場所で利用されている．この Raspberry Pi は拡張基板を接続することで，様々な機能を追加することができるが，この拡張基盤によっては構造上の問題でセンサを取り付けることが困難になってしまう場合がある．例えば，図 1 の Google AIY Voice Kit は，スマートスピーカーを作成するキットとして，Raspberry Pi に拡張基板を接続する．これにより，スピーカーやマイクを接続することができるが，別のセンサを取り付ける場合は，拡張基板にはんだ付けを行う必要があるため，プロトタイピングのコストが高くなる．

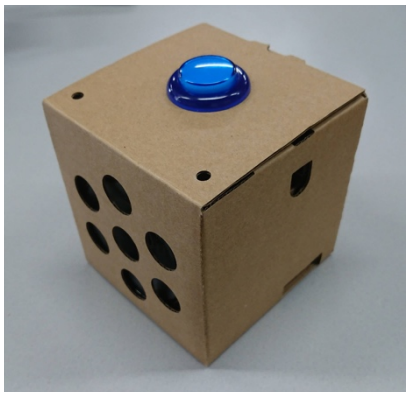

図 1　Google AIY Voice Kit

†1 拓殖大学大学院 工学研究科 情報・デザイン工学専攻
Graduate School of Takushoku University Information and Design Engineering Course

†2 拓殖大学 工学部 情報工学科
Takushoku University Department of Computer Science

Vol.2019-DPS-178 No.27
Vol.2019-CSEC-84 No.27
2019/3/5

IoT のシステム開発ではセンサを含めたハードウェアを新規に作成することが多いことから，プログラムのエラーや想定しないセンサデータが取得された場合に，プログラムの間違いだけでなく，回路の配線ミスやセンサの破損などといったハードウェアが原因となる箇所が増えるためデバッグも困難なものになっていく．

そこで本研究では，ネットワークを用いてセンサを利用することのできる環境を開発することを目的とする．これにより，様々なクライアントからセンサを利用したプログラミングを行えるようにする．また，同一センサを共有することで，回路の作成時間を削減し，素早くプロトタイピングを行うことを可能とする．

また，従来の直接センサを接続した場合のプログラムから，大きく変更することなく本システムを利用できるようにするライブラリの開発を行う．

## 2. 問題分析

### 2.1 IoT プログラミングの問題点と関連手法・研究

IoT のシステム開発におけるプロトタイピングでは，前章でも述べたように，実際にセンサデータを取得し，プロトタイプを開発することで，ソフトウェアとハードウェアがそれぞれ仕様通りに動作するかを確認する必要がある．そのため，プロトタイプを作成する毎にセンサを準備することや，回路を組む必要がある．このように IoT のプロトタイピングではソフトウェアとハードウェアを開発することになるため，開発に要する時間は長くなってしまう．また，利用するセンサ数が増えることで，ケーブル数も増加し取り回しがしにくくなるという問題もある．

本研究に関連する手法として三つの分類に分けて紹介する．

(1) **リモートコントロール**

一つ目の分類として, Raspberry Pi をリモートコントロールする手法を挙げる．この手法の例として，SSH(Secure Shell)や，VNC(Virtual Network Computing)を挙げる．これらの利点としては，他のコンピューやから Raspberry Pi の操作を行うことで, Raspberry Pi に接続されているケーブル数を削減できることである．一方で欠点としては，SSH では CUI での操作が多くなること, VNC では画面表示や操作にラグが生じるため操作がしにくくなることが挙げられる．

(2) **ファイルの転送と実行**

二つ目の分類として, プログラムファイルを Raspberry Pi へ転送し，実行する手法である．これは，JetBrains 社の PythonIDE である PyCharm[1]を利用する．この手法の利点も接続するケーブル数を削減することができることである．しかし, この手法は利用を始めるまでの手順が多いことや, GPIO(General Purpose Input Output)を利用する際には root 権限を取得し再実行するコードを記述する必要があることである．

(3) **ハードウェアの接続**

三つ目の分類として，クライアントにセンサを利用することのできるハードウェアを接続する手法である. これは, Raspberry Pi Zero をクライアントに USB で接続し，GPIO Expander[2]を利用することで, クライアントから Raspberry Pi Zero の GPIO を利用することができるものである．この手法の利点としては，接続先のコンピュータで全ての処理を行うため, Raspberry Pi では難しかった機械学習のような負荷の高い処理を含むプログラムを実行することが可能となる点である．この手法の欠点としては，クライアントと物理接続をすることから，他の手法と比べて回路の取り回しが難しいことである．

以上三つの分類に分けて四つの手法を紹介した．これらの手法はケーブル数を削減することができるが，ハードウェアの構造上の問題でセンサを取り付けられないといった問題や，センサデータを共有するといった目的を満たすことはできない．

また，ネットワークを用いてセンサの制御を行う関連研究として, Raspberry Pi に接続されたロボットアームをネットワーク経由で操作する研究[3]を挙げる．この研究は，Raspberry Pi をサーバとしたウェブページにアクセスし，GUI を用いてロボットアームを制御することができる．

本研究とはネットワーク経由でデバイスを制御するという点について類似性がある．一方で大きく異なる点として，ロボットアームに特化しているものに対して，様々なセンサを利用できるという点と，GUI での操作に対してプログラムを記述して操作する点の二点を挙げる．

### 2.2 要求分析

本システムを開発するにあたっての要求は次の三点である．

(1) Raspberry Pi に接続されているケーブル数を減らすために，無線通信を利用して Raspberry Pi に接続されている回路の制御やセンサデータを利用することができるようにする
(2) 様々なクライアント上でセンサデータを用いたプログラミングを可能とする
(3) 異なる回路を利用したセンサ値の正当性の検証を簡易化する

## 3. 研究概要

### 3.1 機能

I/O 制御サーバの主な機能は，次の二つである．

(1) 無線通信を介した外部センサの制御
(2) 複数クライアントのセンサデータ同時利用

Vol.2019-DPS-178 No.27
Vol.2019-CSEC-84 No.27
2019/3/5

また，ライブラリの主な機能は，次の二つである．

(1) I/O 制御サーバとの相互通信
(2) 複数の I/O 制御サーバとの同時接続

### 3.2 構成

システム構成図を図 2 に示す．

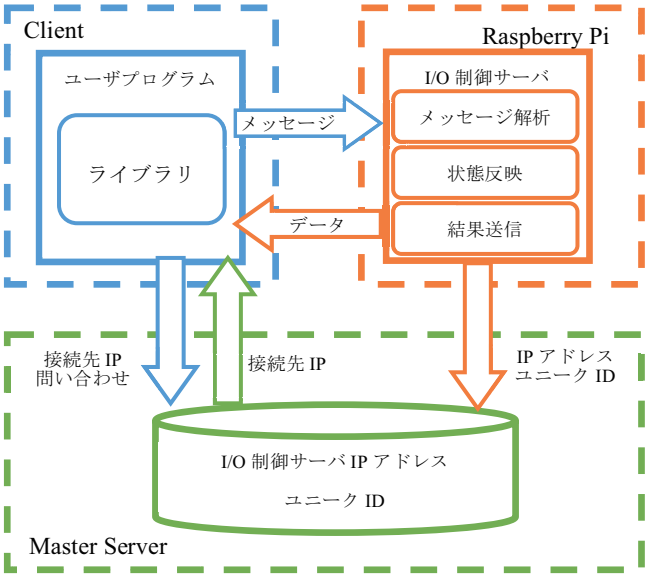


図 2 システム構成図

Raspberry Pi 上で I/O 制御サーバを動作させる．このサーバはクライアンから送信されたメッセージを解析し，センサ状態の変更や，データ取得，クライアントへのデータの送信を行う．

クライアント向けに，Raspberry Pi 上で動作する I/O 制御サーバに接続し，メッセージの送信とセンサデータの受信を行うためのライブラリを作成する．

また，マスターサーバは I/O 制御サーバとクライアント間の接続を仲介するサーバである．

クライアントと各サーバ間の接続には，物理コネクタを排除するため無線通信を利用する．この無線通信には多くのハードウェアで利用することができることから，Wi-Fi を採用することとした．これにより，クライアントはネットワークに接続することができれば，センサデータを利用することが可能となる．また，この通信には双方向通信ができることと，通信の応答速度が早いことが求められ，この要件を満たす通信規格として，WebSocket と gRPC を検討した．この二つの規格で実際に本システムの基本機能を開発し，応答速度が高速であった WebSocket を本システムの各端末間の通信に採用することにした．

### 3.3 I/O 制御サーバ

I/O 制御サーバは Raspberry Pi 上で動作を行う．この Raspberry Pi に外部センサを接続することによって，クライアントは I/O 制御サーバ上のセンサデータを利用することができる．

また，I/O 制御サーバは複数のクライアントにセンサデータを提供できるため同一センサを利用する場合は回路の作成時間を削減することができる．

このサーバではクライアントから受信したメッセージを解析し，そのメッセージに応じて GPIO の状態の変化や，クライアントへのデータの送信を行う．

実装する命令として，LED やスイッチ等を利用するための GPIO ピンの制御や，サーボモータ等で利用する PWM 制御が必要となる．また，Raspberry Pi でアナログデータの入力を利用するためには A/D コンバータを必要とし，これには SPI 通信が必要となる．これらの命令を利用することで，様々な種類のセンサを利用することが可能となるため，これらの命令を I/O 制御サーバでも利用可能とした．

ただし，SPI 通信は利用するデバイスによって異なることから，本システムの SPI 通信は A/D コンバータである「MCP3008」専用の命令となっている．

以上から I/O 制御サーバに実装される命令は表 1 の 11 つである．

表 1 I/O 制御サーバの実装命令

| | | |
|---|---|---|
| 1 | GPIO | ピンモードの設定 |
| 2 | | ピン毎の入出力設定 |
| 3 | | Output 命令 |
| 4 | | Input 命令 |
| 5 | PWM | 初期設定 |
| 6 | | 周波数変更 |
| 7 | | デューティ比変更 |
| 8 | SPI | バスのオープン |
| 9 | | A/D コンバータからのデータ取得 |
| 10 | GPIO クリーンアップ | |
| 11 | 命令文実行 | |

また，I/O 制御サーバは複数のクライアントとの同時接続によってセンサデータの共有を可能としている．しかし，全てのクライアントが GPIO の状態を変更できてしまうと，利用者の意図しない動作が発生する可能性がある．そのため，状態の変更を行う命令は，最初に I/O 制御サーバに接続したクライアントからのみ受け付けることとし，二番目以降に接続したクライアントからはセンサデータを取得する命令だけを受け付けることとした．

### 3.4 ライブラリ

ライブラリは I/O 制御サーバを用いるクライアントで利用する．このライブラリを利用することで，I/O 制御サーバ

Vol.2019-DPS-178 No.27
Vol.2019-CSEC-84 No.27
2019/3/5

に接続されているセンサを利用することが可能となる．

このライブラリは，I/O 制御サーバに対してメッセージを送信し，その結果を受信する．この結果には二種類あり，「GPIO input 命令」と「A/D コンバータからのデータ取得」を実行した場合には，そのセンサデータを受信する．その他の命令を実行した場合には，その処理が正常に完了したことを知らせるメッセージを受信する．この時，エラーが発生した場合には，そのエラーメッセージを受信する．

このライブラリを利用することで，複数の I/O 制御サーバに同時に接続することが可能となっており，それぞれの I/O 制御サーバからセンサデータをそれぞれ取得することができる．これにより，複数のクライアントでセンサデータを共有している I/O 制御サーバに接続を行うことでセンサデータを取得し，そのセンサデータを元に別の I/O 制御サーバの回路を動作させることができる．

例として図 3 を挙げる．このように複数の I/O 制御サーバとの接続が可能なことにより，複数のクライアントで共有している I/O 制御サーバから温度データを取得し，このデータを元に別の I/O 制御サーバの LED を制御するといったことを可能としている．

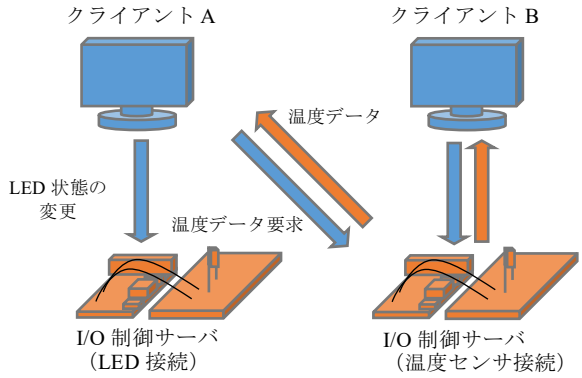


図 3 複数の I/O 制御サーバへの接続

このライブラリを用いてプログラムを記述する時，Raspberry Pi 標準の GPIO ライブラリである「RPi.GPIO」と同等に記述できるようにする．これにより，既存のプログラムを大きく修正することなく再利用することが可能となる．

このライブラリは I/O 制御サーバに対してメッセージを送信することでセンサ状態の変更や取得を行う．このメッセージは文字列となっており，その構成は命令を示す文字列を先頭にその他の情報を「_（アンダースコア）」で結合する形になっている．

この文字列による構成により，同じ構成の文字列を送信することで，WebSocket を利用することのできるプログラミング言語において I/O 制御サーバを利用できるようにしている．また，SPI 通信を行うものや，センサの種類によっては，前述した 11 つの命令ではサポートしきれない場合がある．この構成は可読性が高いため開発者が独自に専用命令の追加をすることを容易としている．

### 3.5 マスターサーバ

マスターサーバは I/O 制御サーバとクライアントの接続を簡易化するために利用するサーバである．マスターサーバを利用した I/O 制御サーバへの接続の流れを図 4 に示す．

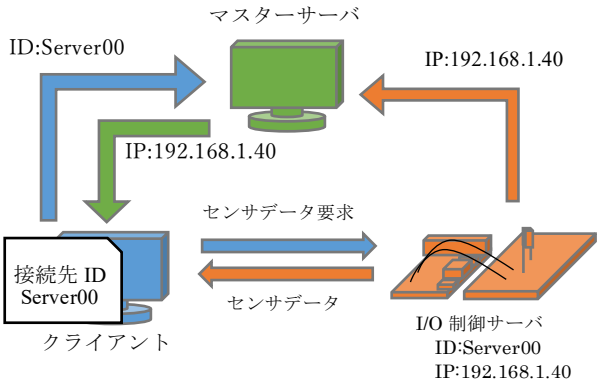


図 4 マスターサーバによる通信仲介

クライアントは I/O 制御サーバに接続する際に IP アドレスを必要とするが，プログラム内に IP アドレスを記述してしまうと，I/O 制御サーバの IP アドレスが変更になった場合に，プログラムの修正をする必要がある．

この問題を解決するために，プログラムには接続先の I/O 制御サーバの ID を記述する．これによりクライアントはマスターサーバへ ID を問い合わせることで，I/O 制御サーバの IP アドレスが取得され，接続を行うことが可能となる．

I/O 制御サーバは，起動時に自身の ID と IP アドレスをマスターサーバへ送信し，マスターサーバはこれを紐付けて保存する．この ID は JSON 形式で保存されており，ユーザが変更することもできる．

また，図 5 に示すように，現在稼働している I/O 制御サーバのリストを表示することができるため，サーバの稼働状態を確認することが可能となっている．

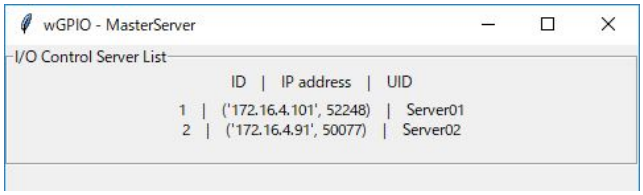


図 5 マスターサーバの表示

Vol.2019-DPS-178 No.27
Vol.2019-CSEC-84 No.27
2019/3/5

## 4. 評価

### 4.1 評価項目

本システムの評価として，次の三点を挙げる．

(1) プログラムコードの変更点
(2) 応答速度
(3) 使い勝手

### 4.2 プログラムコードの変更点

本システムのプログラムコードの変更点として，Raspberry Pi 3 上で実行させた場合と，本研究のサーバとライブラリを利用した場合で，同じ処理を行うプログラムを作成する．本実験では，Raspberry Pi と回路を利用したプログラミングの入門として利用されることの多い，スイッチを押すごとに LED を点灯, 消灯させるプログラムを作成した．この実験の回路には 26 番ピンに LED を，21 番ピンにスイッチを接続している．

図 6 が Raspberry Pi 3 上で実行する従来の手法によるプログラムであり，図 7 が本システムを利用した場合のプログラムである．ここでは，本研究で作成したライブラリの名前を「wGPIO」としている．また，I/O 制御サーバの ID を「Server00」としている．

```
import RPi.GPIO as GPIO
LED = 26; BTN = 21; old = 0

GPIO.setmode(GPIO.BCM)
GPIO.setup(BTN, GPIO.IN)
GPIO.setup(LED, GPIO.OUT)

while True:
   new = GPIO.input(BTN)
   if new == 1 and old == 0:
      if GPIO.input(LED) == 1 :
         GPIO.output(LED, 0)
      else:
         GPIO.output(LED, 1)
   old = new
```

図 6　RPi.GPIO のプログラムコード

```
import wGPIO as GPIO
LED = 26; BTN = 21; old = 0
gpio = GPIO.GPIO("Server00")
gpio.setmode(GPIO.BCM)
gpio.setup(BTN, GPIO.IN)
gpio.setup(LED, GPIO.OUT)

while True:
   new = gpio.input(BTN)
   if new == 1 and old == 0:
      if gpio.input(LED) == 1 :
         gpio.output(LED, 0)
      else:
         gpio.output(LED, 1)
   old = new
```

図 7　本研究のプログラムコード

本システムを利用する場合では，I/O 制御サーバに接続するコードが追加されているほか, ピン設定や output, input 命令には自分で設定したクラス変数が利用されている点が異なるが，大きな違いはない．そのため既存のプログラムコードを大きく変えること無く利用することができる．

また，図 6 の三行目に当たるプログラムが I/O 制御サーバに接続する命令となっており，この引数が I/O 制御サーバのユニーク ID を示している．この引数を変更することで，接続する I/O 制御サーバを素早く変更することを可能としている．

### 4.3 応答速度

本システムの応答速度を計測するための実験を行う．

実験の手法としては，従来の Raspberry Pi 3 上と，本システムを利用した場合それぞれで, GPIO の input 命令を 1000 回実行したときの応答速度の平均値と中央値を計測する．

本システムの実験環境は，サーバとする Raspberry Pi 3 と，クライアントは同じ Wi-Fi アクセスポイントに接続し，ローカルネットワークで実験を行った．

実験結果を表 2 に示す．本研究のサーバとライブラリを利用した場合の平均値は約 4600 μs で，Raspberry Pi 上で動作するものに比べて非常に低速である．しかし，一秒間に処理可能件数は約 220 件であり，本研究の用途であるプロトタイピングでの利用では十分な処理速度であると考える．

表 2　応答速度の実験結果

| | 平均値(μs) | 中央値(μs) | 最遅値(μs) |
|---|---|---|---|
| RPi.GPIO | 4.9 | 4.0 | 116.5 |
| 本システム | 4656.9 | 4201.6 | 100297.2 |

また，本システムは複数クライアントと同時接続を行い，センサデータを提供することができる．そこで，接続クライアント数毎の応答速度の結果を表 3 に示す．

表 3　クライアント数毎の応答速度の実験結果

| ｸﾗｲｱﾝﾄ数 | 平均値(ms) | 中央値(ms) | 最遅値(ms) |
|---|---|---|---|
| 1 | 4.657 | 4.202 | 100.297 |
| 10 | 7.188 | 6.596 | 117.752 |
| 30 | 9.002 | 7.650 | 161.259 |
| 50 | 10.385 | 8.637 | 374.760 |
| 100 | 17.211 | 11.365 | 409.805 |

複数クライアントを用いた実験では，それぞれのクライアントが連続的に input 命令を要求しているため，クライアント数の増加により応答速度が低下していることが確認できる．しかし，50 クライアントが同時に接続を行い，連続的にセンサデータの要求を行った場合でも応答速度の平均は 10ms となっており，一秒間に約 100 件処理することができるため複数クライアントとの同時接続時でもプロト

タイピングでの利用では十分な処理速度であると考える．

### 4.4 使い勝手

著者らが所属する大学の演習講義内で，一部の学生に対して本研究のシステムを利用して開発を行ってもらった．

この講義は情報工学科三年生を対象に行う演習講義であり，Raspberry Pi と回路を用いてプログラミングを行う．また講義の後半では，Raspberry Pi と回路を用いて受講者がそれぞれ考えたサービスを開発してプレゼンテーションを行うものとなっている．本研究のシステムをこの演習講義の後半に当たる部分において利用してもらった．

この演習講義後に本研究を利用した学生に対して，従来の手法である RPi.GPIO ライブラリを利用した場合と，本研究のシステムを利用した場合で，開発の難しさについてアンケートを実施した．

演習講義内において本研究のシステムを利用した学生三名に対してアンケートを行った結果を表 4 に示す．

表 4　難しさについての回答結果

| **従来の手法（RPi.GPIO）と比較して** | | |
|---|---|---|
| 難しかった | 変わらなかった | 簡単だった |
| 0 人 | 2 人 | 1 人 |

変わらなかったと答えた理由として,「プログラムの記述方法に大きな変更が無かったため」との回答を得た．これは，プログラムの記述方法が従来の手法と変わらなかったからとのことだった．そのため，I/O 制御サーバを利用するためのライブラリは，従来の RPi.GPIO ライブラリを利用できる人にとって簡単に利用を始めることができるものであるため，本研究の目的である I/O 制御サーバを利用するためのライブラリの開発に関して達成できたと考える．

また，簡単だったと答えた理由として,「回路を作成する必要無く負担が減ったため」との回答を得た．これは，同じセンサを利用する学生間でセンサを共有することによって，回路を作成せずとも，目的のセンサデータを利用したプログラミングを行えたことによるものである．そのため素早くシステム開発を行えたものであると考える．

このように，本研究で作成したライブラリは従来の手法を理解している人にとって利用を始めやすいという点や，センサデータの共有による回路の作成時間の削減は，演習講義の受講者によるシステム開発では一定の評価を得ることができたと考える．そのため，プロトタイピングに対しても有効であるのではないかと考える．

## 5. おわりに

本研究では，ネットワークを用いたセンサ通信環境の開発を行った．ネットワークを介したセンサデータの利用を行うことで，様々なクライアントからセンサデータを利用したプログラミングを可能とした．また，I/O 制御サーバとすることで，一つのセンサから複数のクライアントに対してセンサデータを提供することにより，回路の作成時間を削減することで迅速にプロトタイピングを行えるようにした．

本システムではセンサデータのやり取りにネットワークを利用し，クライアントと回路を分離することで，Raspberry Pi 上で開発する場合と比較して，ケーブル数を削減することができ，ハードウェアの取り回しを簡単にすることができた．また，ネットワークを利用した場合でも，十分な応答速度でセンサデータを利用することが可能となった．

この I/O 制御サーバを利用する際には Raspberry Pi の標準ライブラリである RPi.GPIO ライブラリを利用したプログラムを大きく変更すること無くプログラミングを行うことが可能である．そのため，既存のプログラムや，インターネット上に存在するプログラムを再利用することができる．

## 参考文献

[1] “PyCharm: the Python IDE for Professional Developers by JetBrains”. https://www.jetbrains.com/pycharm/, (参照 2019-01-11).
[2] “GPIO expander: access a Pi's GPIO pins on your PC/Mac - Raspberry Pi “. https://www.raspberrypi.org/blog/gpio-expander/, (参照 2019-01-11).
[3] Ron Oommen Thomas, Prof. K. Rajasekaren. REMOTE CONTROL OF ROBOTIC ARM USING RASPBERRY PI. International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE), ISSN: 0976-1353 Volume 8 Issue 1 – APRIL 2014