ユビキタスコンピューティングにおけるGUI-デバイス複合型のアプリケーション開発手法

神原 啓介　塚田 浩二

近年，ユビキタスコンピューティング環境での利用を想定した実世界指向のアプリケーション開発が盛んに行われている．こうした開発環境においても，従来のデスクトップ環境と同様にディスプレイや GUI が併用されることは多いが，様々なセンサ／デバイスなどのハードウェアと連携した操作を行う点で，従来のマウス／キーボード操作を前提とした GUI とは大きく性質が異る．そのため，従来の PC 向けの GUI 開発ツールが適用しにくく，開発プロセスが複雑化しやすかった．我々は多数のアプリケーション開発を通じ，こうした問題を解決するためのノウハウを貯め，一定の開発手法を確立してきた．我々の開発手法のポイントは，GUI とデバイス制御プログラムの間に「デバイスサーバ」と言われるミドルウェアを設けることで，両者のプログラムを分離し「疎結合」な状態にするという点である．こうすることで，GUI とデバイスというレイヤーの異なる開発を並行してスムーズに進めることができるようになった．本論文では，こうした GUI-デバイス複合型アプリケーションの開発手法について，具体的な開発方法や様々な事例，課題を交えながら紹介する．

Many research projects and applications for ubiquitous computing have appeared recently. These systems require not only various special devices (e.g., sensors and actuators), but also displays and GUI (Graphical User Interface) just like desktop environment in many cases. In these situations, users often have difficulties to develop applications since classical GUI components and GUI design tools on IDEs (Integrated Development Environment) do not support various I/O devices other than keyboards and mice, and the development process tends to be complex. We propose a novel development method for ubicomp applications through our practical experiences. The main feature of our method is reducing dependences of GUI and devices using middleware called "device server". Our approach can promote efficiency of "parallel" implementation of GUI and devices. In this paper, we introduce practical techniques for developing ubicomp applications including toolkits, frameworks, know-hows, and challenges.

1　はじめに

近年，ユビキタスコンピューティングを想定したアプリケーションの研究開発が盛んになるにつれて，その開発に適した様々なツールキットが登場してきた．たとえば，Phidgets，Gainer，Arduino などのセンサ／アクチュエータを用いたシステム開発を支援するためのツールキットが注目されている．Phidgets [1] や Gainer [4] は，USB でパソコンに接続するデバイス群 と，それらを制御するライブラリ群から構成される．Arduino [8] は，USB 経由でプログラムを書き換え可能な汎用 I/O デバイスと，Processing[†1]に統合された開発環境からなる．こうしたツールキットを活用することで，多様なセンサ／アクチュエータを PC から比較的手軽に扱えるようになってきた．しかし，実際にアプリケーションを開発する際には，単にセンサを扱うだけでなく，複数のセンサを組み合わせたロジックを記述したり，GUI (Graphical User Interface) や Web サービスと連携させたりといった，複雑なソフトウェア開発工程が必要となる．こうした状況では，多様なハードウェアとソフトウェアを組み合わせる点から，従来の PC や携帯端末向けに確立さ

Novel development method for applications integrating GUI and devices on ubicomp

Keisuke Kambara, Koji Tsukada, お茶の水女子大学 お茶大アカデミックプロダクション, Ochanomizu University, Ochadai Academic Production.

コンピュータソフトウェア, Vol.16, No.5 (1999), pp.78–83.
[ソフトウェア論文] 1999 年 8 月 3 日受付.

†1　http://processing.org/

れたマウス・キーボードを前提としたUI設計や，インタフェースビルダーのような開発ツールを単純に適用できないケースも多い．

一方，ユビキタスコンピューティングにおける開発手法としては，ユビキタスコンピューティングにおけるプログラミングモデル[14]の提案などが行われているが，これらはプログラム内部のやや抽象的なモデル化に焦点を当てており，実際のアプリケーションを開発する上での具体的／効率的な実装方法については議論されていない．

こうした中，我々は主にUI研究の立場から，多数のユビキタスコンピューティング向けアプリケーションの研究開発を行ってきた．特に，我々がこれまでに開発したアプリケーションの中では「GUIとデバイスを連携したシステム」の事例が多く，例えば以下のようなシステムを開発してきた．

- なめらカーテン [2]
- ハングル ガングル [3]
- DrawerFinder [5]
- LunchCommunicator [6]
- MouseField [7]
- インタラクティブな掃除機 [9]
- Complete Fashion Coordinator [10]
- IODisk [11]
- タグタンス [12]
- EyeWish [15]

これらのGUIとデバイスを組み合わせたシステム（以下，GUI-デバイス複合型アプリケーション）は，いずれも構成が複雑になりがちで，複数人で開発することも多い．そのため，上述したツールキットや開発手法を単に用いるだけでは，なかなか効率的に開発が進まないという状況も増えてくる．

こうした問題を改善するために，我々は何度もGUI-デバイス複合型アプリケーションの開発を繰り返す中でノウハウを貯め，実装方法を洗練させてきた．我々の開発手法のポイントは，GUIとデバイス制御プログラムの間に「デバイスサーバ」と言われるミドルウェアを設けることで，両者のプログラムを分離し「疎結合」な状態にするという点である．こうすることで，GUIとデバイスというレイヤーの異なる開発を並行してスムーズに進めることができるようになった．また，我々がこれまでに開発してきた，デバイス制御のためのソフトウェア群「MobiServer[13]」を活用することで，デバイスサーバの開発を支援する点や，GUI側の開発に「Flash」を用いることでGUIデザインの自由度を高めるといった点も重要な特徴である．本論文では，こうしたGUI-デバイス複合型アプリケーションの開発手法について，具体的・実践的な実装方法や様々な事例，課題などを交えながら紹介する．

2 ユビキタスコンピューティングにおけるUI開発の問題

ユビキタスコンピューティングにおいてUIを開発する際，大きく分けて「非WIMP方式のGUI」「GUIとデバイスの同時開発」「デバッグと分担作業」といった点が問題になる．

2.1 非WIMP方式のGUI

ユビキタスコンピューティングにおけるシステム開発においても，多くの場合において，従来のデスクトップ環境と同様に情報の表示や操作にディスプレイおよびGUIが使われる．たとえば，家電機器などにディスプレイが埋め込まれた情報アプライアンスや，リビングのテレビやモバイル端末と連携したシステム，デジタルサイネージなど，色々な場所に大小様々なディスプレイやプロジェクタが利用されている．

一方，ユビキタスコンピューティング向けのGUIは従来のPC向けのGUIとは性質が異なっている．従来のPC向けのGUI，いわゆるWIMP (Window, Icon, Menu, Pointing device) 方式のGUIは，マウスやキーボードを使った近接操作を前提に設計されている．しかしユビキタス環境では，マウスやキーボードに限らず多様なセンサやデバイスを用いてインタラクションを行うため，WIMP方式とは違ったデザインのGUIが求められる．例えば，ターンテーブル型のデバイスを使って写真や動画をコントロールするIODiskは，ディスクの回転操作にあわせて，画面上のコンテンツも円盤状に回転・アニメーションするGUIとなっている（図1）．また，ICタグのつけ

図 1　IODisk 写真ビューア

られた物を置いたり，動かしたりすることで音楽などのコンテンツを操作をする MouseField（図 2）では，例えば置いた CD ジャケットを回転する動作により音量を操作する．このような操作方法を視覚的に表現するため，MouseField では図 2 の画面内のように円弧状に変形したデザインのボリュームインジケータとなっている．さらに MouseField では CD ジャケットを上下に動かすことで再生リストを操作する．このとき画面内の CD ジャケットがアニメーションすることで視覚的なフィードバックを返す．このように独自のデバイスを使って操作をする場合，従来の GUI 部品とは異なった動きや形の GUI が必要になる．

また，ユビキタス環境では「なにか別の作業をしながら利用する」「立ったり歩いているとき使う」「テレビのように機器と離れた状態で利用する」「あまり操作せず閲覧が中心」といった生活の中での多様な利用状況に応じた GUI をデザインする必要がある．そのため「机の前で椅子に座り，キーボードやマウスで比較的集中して操作する」という限定された状況を前提にデザインされた既存の GUI に比べて，求められるデザインのバリエーションが広い．PC 用の GUI と携帯端末の GUI，テレビ向けの GUI がそれぞれ大きく異なるように，画面のサイズや画面からの距離，利用時間，場所に応じて，GUI 部品の大きさや画面内の情報量，レイアウト，画面遷移の仕方が大きく変わる．

このように，ユビキタスコンピューティング向けの GUI では，PC 向けの GUI とは異なる多様なデザインが求められるため，VisualStudio のユーザインタフェースエディタや Xcode のインタフェースビルダー，HTML オーサリングツールの Dreamweaver といった，WIMP に最適化された従来の GUI デザインツールを単純に適用することが難しい．上で述べた IODisk や MouseField のように，従来の GUI 部品とは動きや形が大きく異なる GUI は，GUI デザインツール上でボタンなどの既存の GUI 部品を配置するだけでは作ることができない．

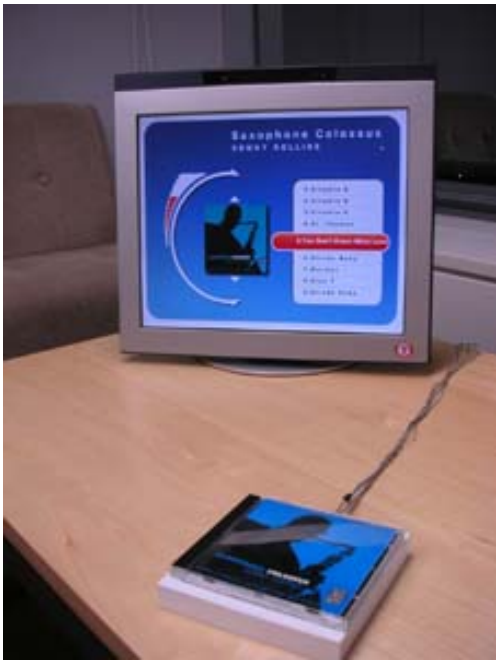

図 2　MouseField

2.2　GUI とデバイスの同時開発

従来のマウスやキーボードを前提とした GUI との大きな違いとして，様々なセンサなどのデバイスを用いた「多様な操作方法」が挙げられる．そのためシステムを作る際，これまでのように画面の中だけでなく，ハードウェアの設計や実装，制御まで考える必要がある．

研究開発初期のプロトタイピングの段階では，まだシステムの仕様がはっきりしていないことが多く，GUI とデバイスをお互いにすり合わせながら実装することになる．この場合，種類の違う 2 つの開発を調整しながら同時並行に進めるという難しさがある．

また，GUI とデバイス制御は開発のレイヤーが大きく離れている点も問題となる．デバイス制御はハードウェア寄りの低いレイヤーなのに対し，GUI はユーザー寄りの比較的高いレイヤーになる．レイヤー同士が近い開発であれば，使用する言語や開発環境が似ていることが多く，同時にまとめて開発しやすいが，レイヤーの異なる GUI とデバイス制御ではそれらが大きく異なるため同時に開発しにくい．

2.3 デバッグと分担作業

前述のように，GUI とデバイスを組み合わせたシステムでは両者の連携が重要となる．しかし，両者のプログラムをまとめて 1 つにしたり，両者の依存関係が強く，単体では動かせない「密結合」した状態になると，デバッグや分担といった開発作業が難しくなる．

システムが密結合になると，例えば「片方を直すともう片方も動かなくなってしまう」といった事態が起こりやすくなる．その結果，GUI とデバイスの双方をまとめて直していくことになり,「どちらかに集中して開発しにくくなる」「問題の切り分けがしにくくなるためデバッグが複雑になる」といったことにつながる．

さらに密結合しすぎると複数人での分担作業が難しくなる．GUI 担当とハードウェア担当に分かれて作業する場合，密結合していると「片方を直している間，もう片方の担当者が作業できない」ということになり効率が悪い．

3 GUI-デバイス複合型開発

ここでは，ユビキタスコンピューティングのためのデバイスと GUI を組み合わせた開発における様々な問題を考慮した，我々の開発手法を紹介する．

本開発手法のポイントは，デバイス側と GUI 側のプログラム・開発作業を分離することにある．そのために，まず図 3 のように，デバイスと GUI の間には「デバイスサーバ」と言われるミドルウェアを設け，GUI 側のプログラムが無い状態でもデバイスの制御・開発・デバッグをできるようにする．また，GUI 側の開発には主に Adobe 社の「Flash」を用いることで，GUI デザインの自由度を高める．さらに，マウス・キーボードによるエミュレーション機能などを用意することで， GUI 部分だけを切りだして開発・デバッグできるようにする．

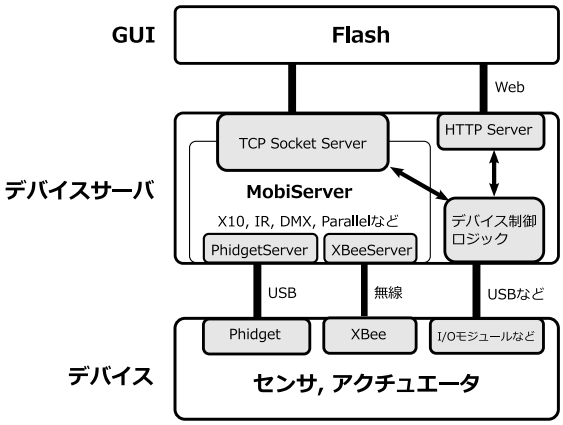

図 3 GUI-デバイス複合システムの構成

こうして GUI とデバイス制御を別々のプログラムに分離し「疎結合」な状態にすることで，レイヤーの異なる開発作業をうまく切り分けながら両者を連携できるようにした．その結果，GUI とデバイスのどちらかに集中して開発したり，問題の発生箇所を切り分けてデバッグしたり，複数の担当者が関わる作業をスムーズなものにすることができる．また，両プログラム間は TCP Socket や HTTP といったプロトコルで通信することで，LAN やインターネット経由で結合テストできるようになり，プログラムやハードウェアを別の PC に度々移し替える手間も減る．

以下では，我々の手法の特徴である「Flash」「デバイスサーバ」「疎結合」についてそれぞれ詳しく述べる．

3.1 Flash

Adobe Flash は主に RIA (Rich Internet Application) と言われる高度な Web アプリケーションの UI 作成や Web 上でのアニメーション表現，動画の再生などに広く用いられている技術である．最近では Adobe AIR という技術によって，Flash を用いたデスクトップアプリケーションの作成も可能になって

いる．

3.1.1 Flash を用いる理由

Flash は一般的に Web サイトやアニメーション作成向けのツールとされているが，GUI-デバイス複合型開発に適した特徴を備えている．

Flash が GUI-デバイス複合開発に向いている最大の理由は，他の GUI 開発ツールに比べて「非 WIMP 方式の GUI を開発しやすい」ためである．2.1 節で述べたように，非 WIMP 方式の GUI では「マウスやキーボード以外の多様なセンサやデバイスを用いたインタラクション」や「従来の GUI 部品とは異なった動きや形の GUI」といった多様なデザインが求められる．WIMP に最適化された従来の GUI デザインツールはデザインの自由度が低く，多様なデザインの実現が難しいのに対し，Flash はデザインの自由度が高い．Flash はグラフィックツールやアニメーション用のタイムラインを内蔵しており，独自の形状・動きを持つ GUI 部品を作りやすく，それらを柔軟に配置し，動かすことができるためである．また，単なるグラフィックツールとも異なり，プログラミング言語（ActionScript）や開発環境を備えているため，複雑な機能の UI も作ることができる．

これまでに Flash を使ったことのあるデザイナーや UI エンジニアが多く，そのようなデザイナーとの共同作業がしやすいことも理由として挙げられる．2.2 節で述べたように GUI とデバイス制御はレイヤーが大きく異なるため，それぞれのレイヤーを得意とする人同士で作業すると効率が良い．デザイナーや UI エンジニアにとっては，新たにツールを覚えることなく，既存のノウハウを活かすことができる．

その他の理由として，Flash はデスクトップ，モバイル端末，Web ブラウザなど多くのプラットフォーム上で動作するため，様々なシステム構成に柔軟に対応できることや，C++や C#，Java といった他の高級言語に比べて動画（ストリーミング）や音声，画像などマルチメディアコンテンツを扱いやすいことが挙げられる．

3.2 デバイスサーバ

Flash は上述のように GUI-デバイス複合型開発に適した特性を備えるが，OS の機能に直接アクセスすることができないため[†2]，直接外部デバイスを制御することは難しい．そこで，我々はデバイスと Flash を仲介するためのミドルウェア（以下，デバイスサーバ）を用意した．デバイスサーバの主な役割は，PC につながった各種センサやアクチュエータを制御し，Flash から直接利用できるような API を提供することである．

Flash に対して API を提供する方法としては，TCP Socket サーバや CGI のような HTTP サーバとして動作させる．これは Flash が TCP Socket または HTTP 経由で外部ネットワークにアクセスできるためである．TCP Socket は HTTP に比べて高速で双方向通信可能なため，リアルタイムなデバイスの制御に適している．一方，HTTP はシンプルなプロトコルであり，Flash 本来の使い方として Web 上での HTTP 通信に長けていることもあって，比較的実装しやすい．

TCP Socket/HTTP サーバとデバイスを制御するプログラムがデバイス制御ロジックである．デバイス制御ロジックはアプリケーションごとに固有であり，デバイスサーバの中心的なプログラムとなる．

センサやアクチュエータは Phidget や Gainer などの I/O モジュールを利用して USB などで PC に接続して制御する．

デバイスサーバは，Flash のクライアントと接続されていない状態，すなわちデバイスサーバ単体でもデバイスの動作を確認できる機能も開発を進める上で重要となる．デバイスサーバ自体に簡単な GUI を持たせることで，手動でデバイスを制御したり，センサのパラメータや各デバイスの動作状況のログを表示する．

†2 たとえば，Windows 環境においては，Win32API を介してシリアルポートを開くことはできず，外部 DLL(Dynamic Link Library) を直接呼び出すこともできない．

3.2.1 MobiServer

デバイスサーバには多数の物理的なデバイスを制御する機能が求められる．こうした実装を支援し，汎用的なデバイスサーバとしても使えるソフトウェア群「MobiServer」を開発した[13]．

MobiServer は，以下のような多様なデバイス群を制御するためのミドルウェアの総称である．

- PhidgetServer: USB 接続のセンサ・アクチュエータ群である Phidget を制御する
- X10Server: 電灯線通信を行う X10 デバイスを用いて照明や家電を制御する
- IRServer: USB 接続の学習リモコンを用いて，エアコンやテレビなどの家電を制御する
- ParallelServer: 安価な USB パラレル変換モジュールを用いてデジタル I/O を制御する
- DMXServer: フルカラー LED 照明などを制御する
- XBeeServer: 無線通信モジュール XBee を組み合わせた無線センサシステムを制御する

これらの MobiServer はそれぞれ TCP Socket サーバやデバイス動作を確認するための GUI を備えており，単体でも 1 つのデバイスサーバとして機能する．そのため単純に 1 つのデバイス制御をするだけであれば，新たにデバイスサーバを書かなくてもよい．一方，MobiServer が対応していないデバイスを用いる場合や，複数のデバイスを組み合わせた複雑なシステムを作る場合，MobiServer と別のプログラム（デバイス制御ロジック）を組み合わせて利用することもできる．こうすることで必要最小限のプログラムを書くだけでデバイスサーバを構築することができる．MobiServer と他のプログラムの間の通信は，Flash と同様に TCP Socket を用いることが多い．

MobiServer はいずれも Web サイト[†3]からダウンロードできる．MobiServer のうち PhidgetServer については後述するが，その他の MobiServer の詳細な機能や使い方については Web サイトまたは解説論文 [13] を参照されたい．

†3 http://mobiquitous.com/mobiserver/

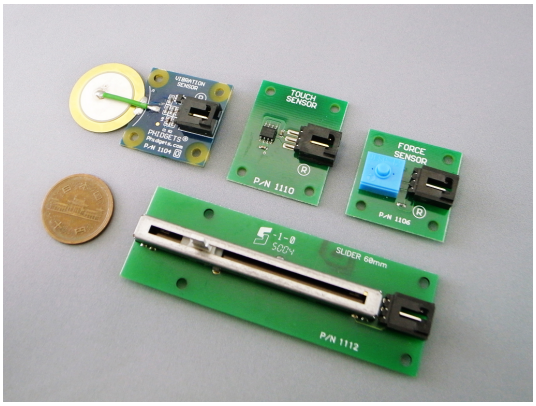

図 4 Phidgets のセンサの 1 例．上段 左:振動センサ，中央:タッチセンサ，右:圧力センサ，下段:スライダー

3.2.2 PhidgetServer

MobiServer のうち，我々の開発でも頻繁に利用している PhidgetServer を例に紹介する．Phidgets [1] は図 4 のような USB で PC に接続できるセンサやアクチュエータ（スライダー，スイッチ，LED，モーターなど），汎用 I/O ボード（アナログ入力・デジタル入出力）などのキットで，PhidgetServer ではこれら様々なデバイスを制御する．

PhidgetServer は TCP Socket サーバとして動作し，TCP クライアントからテキストのコマンドを送信することでデバイスを制御したり，センサの状態をサーバから通知することができる．送受信されるコマンドは表 1 のようなテキストとなっている．

PhidgetServer の画面は図 5 のようになっており，出力系デバイスの操作や，入力系デバイスの状態およびログを表示するための GUI を備えている．この GUI により，TCP Socket クライアントが接続されていない状態でもデバイスの動作を確認できるため，デバイス単体での開発やデバッグ作業をすすめやすくなっている．

PhidgetServer 以外の他の MobiServer も，コマンドや GUI に違いはあるものの，基本的な使い方は PhidetServer とほぼ同様である．

3.2.3 デバイスサーバの開発環境

デバイスサーバの開発環境や OS に関して原則的には制約は無いが，我々がデバイスサーバを開発する

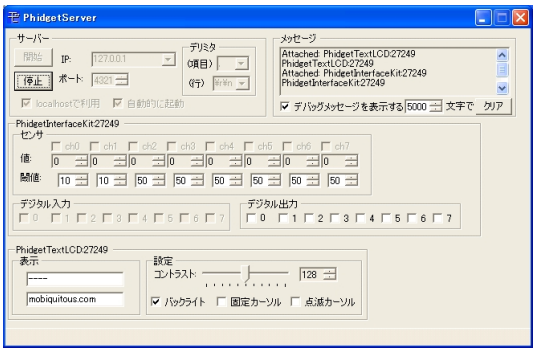

図 5　PhidgetServer

表 1　PhidgetServer のコマンド例

Phidgets,In,InterfaceKit,27249,Analog,0,512
Phidget InterfaceKit (ID:27249) のアナログ入力ポート 0 が 512 へと変化
Phidgets,Out,Servo,3633,0,230
Phidget ServoMotor (ID:3633) のポート 0 の角度を 230 に変更

際は，主として OS に Windows，開発環境に Visual-Studio，プログラミング言語に C#を用いている．その理由としては，まず Windows から既存のデバイスを利用するためのドライバやツールキット，情報が充実していることが挙げられる．また，最近ではネットブックと呼ばれる Windows 搭載の小型のノート PC が 3 万円前後で手に入るため，システム内に直接 PC を組み込むような場合でも，コストを押さえてコンパクトに作れるということも大きな理由となっている．そして，Windows のアプリケーションを開発するにあたっては，C++などに比べて C#が比較的容易であり，Java などと比べて低レイヤー（ハードウェア寄り）のプログラミングをしやすい．

上記のような理由から，MobiServer はいずれも C#で書かれており，Windows 上で動作する．そして我々が MobiServer を用いたデバイスサーバを作る際も，開発環境や動作環境を合わせた方が開発しやすいため，いずれのデバイスサーバも同じような構成となっている．

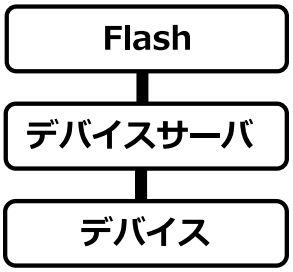

図 6　入出力デバイス系

4　システム構成の分類

図 3 では GUI-デバイス複合型開発での基本的なシステム構成を示した．一方，具体的なアプリケーションに目を向けてみると，「Flash」「デバイスサーバ」「デバイス」という構成要素は共通するものの，要素の数や通信方法などの詳細は異なる．ユビキタスコンピューティングでは多様な利用形態があり得るため，システム構成のバリエーションも広い．

実際のシステム構成はアプリケーションによって異なるものの，ジャンルに応じて以下のようにいくつかの典型的なパターンに分類できる．

- 入出力デバイス系：デバイスで GUI を操作する
- コミュニケーションデバイス系：GUI を備えた複数の機器同士で通信する
- 実世界 Web 系：Web とデバイスが連携する

以下では，それぞれの分類の特徴および適したシステムの構成方法について述べる．

4.1　入出力デバイス系

センサを用いて GUI を操作，または GUI に応じてアクチュエータで出力するシステムを指す．GUI とデバイスが 1 対 1 の関係になっており，Flash とデバイスサーバを使った最も基本的な構成で実装できる（図 6）．

我々が開発したアプリケーションの中では，IODisk [11] や MouseField [7]，ハングル ガングル [3]，インタラクティブな掃除機 [9]，EyeWish [15] が入出力デバイス系のアプリケーションである．

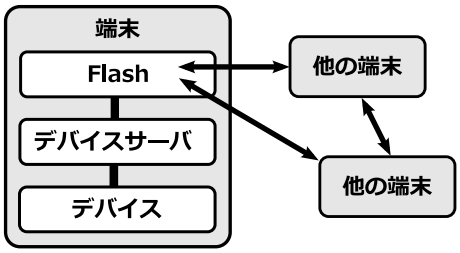

図 7 コミュニケーションデバイス系

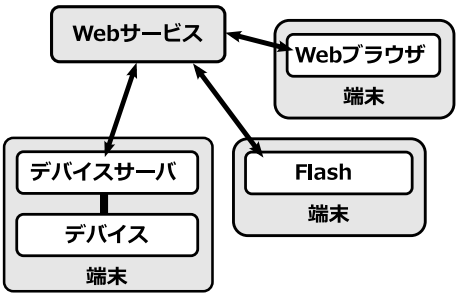

図 8 実世界 Web 系

4.2 コミュニケーションデバイス系

図 7 のようにディスプレイが組み込まれた機器（端末）を複数用いて，コミュニケーションするシステムを指す．各端末は Flash とデバイスサーバの基本構成で実装し，端末間の通信は Flash を利用してインターネット経由で通信すると良い．Flash を使った通信方法としては，TCP Socket や HTTP，ストリーミング用の RTMP[†4]，Flash 間で P2P 通信ができる RTMFP[†5]などがある．

我々が開発したアプリケーションのうち，なめらカーテン [2] と LunchCommunicator [6] がコミュニケーションデバイス系のアプリケーションと言える．

4.3 実世界 Web 系

図 8 のように実世界デバイスと Web サービスが連携したシステムを指す．Web 上の文書，写真，動画といったデータを利用するシステムや，Twitter・Facebook などのソーシャルネットワークおよび集合知を活用するシステム，またはセンサデータなどを Web 上に保存・共有するライフログ系のシステムがある．Web サービスを介して複数のアプリケーションやデバイスが連携できるのも特徴である．

デバイスサーバと Web サービス間の通信は REST (Representational State Transfer) や SOAP[†6]を用いて HTTP 上で行う．既存の Web サービスを利用する場合は Web API が提供されていることも多い．また，チャットのようなリアルタイム性の高い通信をする場合は Flash の TCP Socket 通信なども利用できる．

我々が開発したアプリケーションのうち，Complete Fashion Coordinator [10] やタグタンス [12]，DrawerFinder [5] が実世界 Web 系である．

5 開発事例

我々がこれまでに研究開発してきたシステムの中から，上述した 3 種類のシステム構成の代表的な開発事例をそれぞれ紹介する．入出力デバイス系として「IODisk」，コミュニケーションデバイス系として「なめらカーテン」，実世界 Web 系として「タグタンス」を取りあげる．

いずれのシステムの開発にもこれまでに述べた「Flash を用いた GUI や，MobiServer を利用したデバイスサーバによる構成」「複数人での分担開発，デバッグ作業」といった開発手法が取り入れられている．しかし，表 2 のように，それぞれシステムごとに疎結合の度合いやプロトコルの複雑さなどが異なっており，それによって開発の効率にも違いがあった．疎結合度が高くプロトコルがシンプルな IODisk の開発は非常にスムーズであったが，疎結合度がやや低めでプロトコルが複雑ななめらカーテンは効率さに欠ける点もあった．

ここでは，それぞれの開発事例の紹介を通して，本提案の有効性や課題について議論する．

5.1 IODisk

IODisk [11] とは DJ が用いるターンテーブル風のデバイス/操作によって，写真や動画といったデジタ

†4 http://www.adobe.com/devnet/rtmp.html
†5 http://www.adobe.com/products/flashmediaserver/rtmfp_faq/
†6 http://www.w3.org/TR/soap12-part0/

表 2　各事例の疎結合度とプロトコルの複雑さ

	疎結合度	プロトコルの複雑さ
IODisk	高	シンプル
なめらかーテン	低	複雑
タグタンス	高	やや複雑

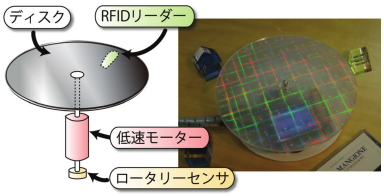

図 9　IODisk

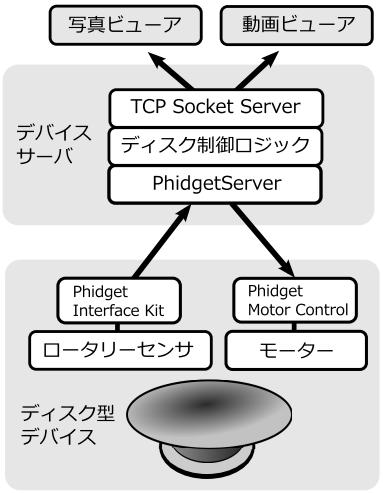

図 11　IODisk のシステム構成

ルコンテンツを閲覧するシステムである（図 9）．

図 9 のディスクを少し回すと，手を離しても内蔵されたモーターによりそのまま一定速度で回り続ける．回転しているディスクに手で摩擦をかけることで，回転速度を細かく調整できる．そして，この回転速度に応じてコンテンツの再生速度を変えるというものである．

例えば IODisk を用いて複数の動画を切り替えながら見る場合，ゆっくりディスクを回すと再生開始，回転を速めることで早送り，逆回転により巻き戻し，回転を止めることで一時停止といった操作をする（図 10）．また，ディスクを瞬間的に急速回転することで次のビデオに移動するといった操作もできる．そして，同様の操作方法で写真アルバムのスライドショーをコントロールする．

5.1.1　システム構成

IODisk は図 11 のような構成になっている．ディスク型デバイスは主にロータリーセンサとモーターから成り，いずれも Phidget および PhidgetServer により PC から制御する．ロータリーセンサから送られる回転角の情報は，ディスク制御ロジックによって回転方向や速度に応じて再生や停止，アルバム移動といったコマンドに解釈される．そして TCP Socket サーバを通じて，表 3 のコマンド文字列が改行区切りで写真ビューア†7・動画プレーヤー†8などのアプリケーションに送られる．

表 3　IODisk で GUI 側アプリケーションに送られるコマンド

コマンド文字列	操作内容
next	次のビデオへ移動
previous	前のビデオへ移動
pause	一時停止
stop	停止
forward	再生
ff1 〜 ff4	早送り (4 段階)
reverse	巻き戻し
fr1 〜 fr4	巻き戻し (4 段階)

5.1.2　開発作業の分割

IODisk は GUI 担当とデバイス担当の 2 人で開発した．図 11 のうち，GUI 部分である写真・動画ビューアを一人が開発し，もう一人がデバイスサーバ（主にディスク制御ロジック）とディスク型デバイスを開発した．

†7　https://github.com/tsuka-lab/IODiskPhotoViewer にてソースコードを公開

†8　https://github.com/tsuka-lab/IODiskVideoPlayer にてソースコードを公開

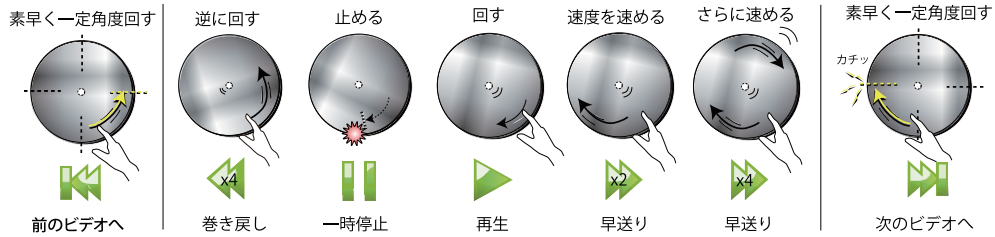

図 10 IODisk の操作

IODisk では GUI 部分をデバイス関連の処理から明確に分離することができ，開発全体を通してそれぞれほぼ単独で開発作業を進めることが可能であった．システム構成で述べたように，GUI-デバイスサーバ間の通信は「表 3 のコマンドを GUI 側に送る」というシンプルなプロトコルであり，両者のプログラムを組み合わせて動かす際も大きな問題は起こらなかった．

ディスク制御ロジックとして開発した「IODiskServer」というプログラムは，単独でディスク型デバイスを制御する機能に加えて，GUI 側へ手動でコマンドを送る機能を持たせた．この機能により，ディスク型デバイスが手元に無い状態でも GUI の動きをテストすることができた．さらに，GUI 側である写真・動画ビューアでは，キーボード操作でコマンドをエミュレートできるようにしており（リスト 1），IODiskServer が無い状態，すなわち単独のプログラムとして動作するようになっていた．

リスト 1 コマンドをキーボードでエミュレート

(ActionScript3)

```
//サーバからコマンドを受け取って実行
function onSocketData(
            e:ProgressEvent):void {
  var s:String = Socket(e.target)
                   .readUTFBytes(
                     e.bytesLoaded);
  for each (var cmdStr:String
                in s.split(/\n/)) {
    if (cmdStr.length > 0) {
      // コマンド実行
      onCommand(cmdStr);
    }
  }
}

// キーボード操作で
// コマンドをエミュレート
function onKeyboardEvent(
            e:KeyboardEvent):void {
  switch (e.keyCode) {
    case Keyboard.N:
      // nextコマンド実行
      onCommand("next");
      break;
    case Keyboard.P:
      onCommand("previous");
      break;
    ～～～中略～～～
  }
}
```

5.2 なめらカーテン

なめらカーテン[2] とは，カーテンメタファを用いることで日常空間でもプライバシーを守りながら常時利用できる遠隔ビデオチャットシステムである．

もし，ビデオチャットの接続を閉じず，遠隔地同士で常につながった状態になれば，離れた部屋にいる人の存在や様子をいつでも知ることができ，より気軽に話しかけることができるなど，遠隔地にいる人同士でのコミュニケーションがしやすくなると考えられる．しかし，家の中のような日常空間ではお互いのプライバシーが問題になる．

そこで，部屋の窓の外から中を覗かれないようにするというカーテンの機能に着目し，ビデオチャットにカーテンのメタファを取り入れることでプライバシーを直感的に制御できるようにした．

図 12 のようにビデオチャット専用端末にカーテン状の装置を取り付け，本物のカーテンと同じ感覚で開け閉めできるようにした．相手にこちらの様子を見られたくないときは，カーテンを閉じることで相手側に

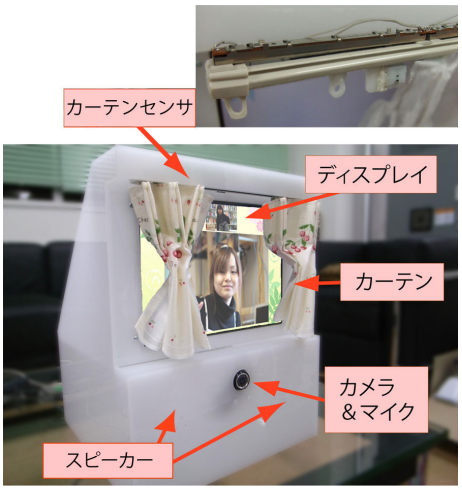

図 12　なめらカーテン端末

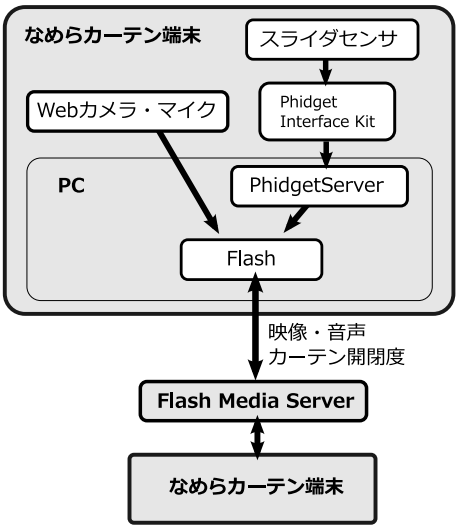

図 13　なめらカーテンのシステム構成

映る映像をぼかして見えにくくすることができる．

このカーテンの開き具合に応じてぼかし具合が変化し，どれぐらい詳細な様子を相手に伝えるかを簡単に調整できる．カーテンを半開きにしておけば「人が動いている」「話の内容は聞こえないけどにぎやかになった」といった大まかな様子だけを知る/伝えることができ，たとえば，カーテンを完全に閉じた状態であっても「相手側の部屋の明かりがついているか（部屋に人がいるかどうか）」といったことは分かるようになっている．このように詳細を省いた大まかな情報だけを常時やりとりすることで，相手の部屋で何か変化があったときにカーテンを開けて話しかけてみる，といったコミュニケーションのきっかけが生まれやすくなる．

5.2.1　システム構成

なめらカーテンは図 13 のような構成になっている．

カーテンレールの部分にスライダセンサが取り付けられており，Phidget および PhidgetServer によりカーテンの開き具合を取得する．そして，このカーテンの開き具合およびストリーミング映像・音声を，Flash で書かれたプログラム†9から，Flash 用のストリーミングサーバの FlashMediaServer 経由で相手に送る．Flash および FlashMediaServer を利用することで，端末間での情報のやりとりや，映像・音声のストリーミングを比較的容易に実現できた（リスト 2）．

リスト 2　ストリーミング開始部分のコード (ActionScript3)

```
// Flash Media Serverに接続
nc = new NetConnection();
nc.addEventListener(
    NetStatusEvent.NET_STATUS,
    netStatusHandler);
nc.connect(
    "rtmp://example.com/curtain");

// 接続したら映像と音声を送受信
function netStatusHandler(
      event:NetStatusEvent):void {
  switch (event.info.code) {
    case "success":
      publishLiveStream();
      connectStream();
      break;
    ～～～中略～～～
  }
}

// 自分の映像と音声を送信
function publishLiveStream():void {
  // カメラとマイクの設定
  mic = Microphone.getMicrophone();
  camera = Camera.getCamera();
  camera.setMode(640,480,8);
  // 映像と音声を送信
```

†9　https://github.com/tsuka-lab/SmoothCurtainAir にてソースコードを公開

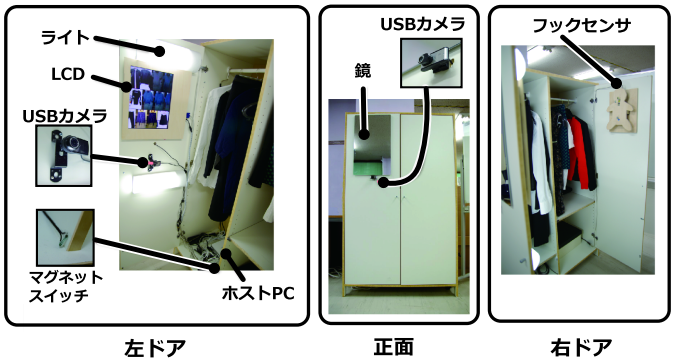

図 14 タグタンス

```
    ns = new NetStream(nc);
    ns.attachCamera(camera);
    ns.attachAudio(mic);
    ns.publish();
}

// 相手の映像と音声を受信
function connectStream():void {
    stream = new NetStream(nc);
    video = new Video();
    video.attachNetStream(stream);
    stream.play();
}
```

5.2.2 開発作業の分割

なめらカーテンは 3 人で開発した．メインの一人がシステム全体の開発に関わり，残りの二人はそれぞれ GUI 側，デバイス側を補助するという体制であった．図 13 のうち，GUI にあたる部分が「Flash」，デバイスサーバにあたるのが「PhidgetServer」，そしてデバイスにあたるのが「Web カメラ・マイク」「スライダセンサ」「Phidget Interface Kit」になる．デバイスサーバは，すでに開発済みであった PhidgetServer を単独で用いたため，新たな開発はしていない．

開発の前半では，GUI とデバイスを別々に開発し，それぞれの開発においてデバイスサーバである PhidgetServer を活用する機会が多かった．例えば GUI 側では，スライダセンサが無い状態でも，PhidgetServer の画面上で値を変えることで仮想的にスライダを動かすことができた．デバイス側では，スライダセンサが正常に動いていることを PhidgetServer の画面上に表示される値で確認することができた．

5.2.3 開発作業の問題点

なめらカーテンの開発では GUI とデバイスサーバでの処理の切り分けが不十分で，デバイスに関連する処理が GUI 側に含まれてしまい，GUI 側の処理が複雑になるという問題があった．例えば，スライダセンサから送られる値の範囲は端末によって少しずつばらつきがあり，これを原因とする分かりにくいバグの発生につながるということがあった．このスライダセンサの問題に対し，GUI 側（Flash）で対処したため，GUI とデバイスがやや密結合な状態になった．さらに，GUI 側では「ストリーミング」「PhidgetServer との通信」「Web カメラ・マイクの制御」など担当する処理が多いことも重なり，コードが複雑になってしまった．

本開発手法を厳格に適用するならば，このようなセンサ値の正規化といった処理はデバイスサーバのデバイス制御ロジックで行うべきであった．このデバイス制御ロジックは，図 13 の「Flash」と「PhidgetServer」の間に入ることになる．こうすることで疎結合な状態を保ち，テストをしやすくできると考えられる．

5.3 タグタンス

タグタンスとは家庭で利用される観音開きのタンスを利用し，手軽に服を撮影，分類して，さらに Web 上で管理・共有できるシステムである [12]．図 14 のように扉の内側にフックが取り付けられており，ここにハンガーを掛けることで反対側の扉に取り付けら

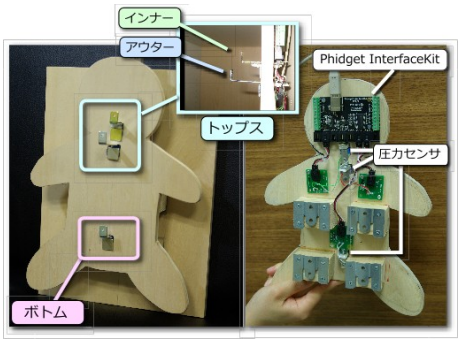

図 15　タグタンスのフックセンサ

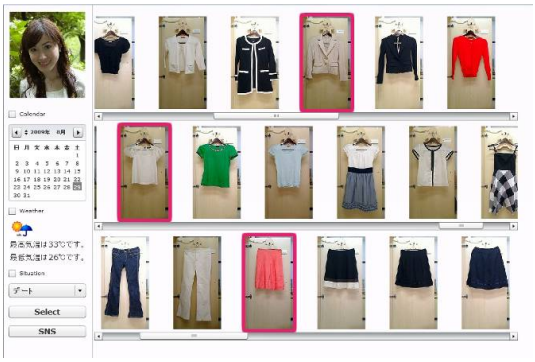

図 16　Last-Minute Coordinator

れたカメラによって服の写真が撮られる．フック部分は図 15 のようにアウター・インナー・ボトム用の 3 つのフックがついており，いずれかのフックにかけることで撮影時にタグづけ（分類）される．また，フックには圧力センサが取り付けられており服の重さも同時に記録される．そして，写真や撮影日時，タグ，重さといったデータは写真共有サービスの Flickr にアップロードされる．

5.3.1　Last-Minute Coordinator

タグタンスを利用したアプリケーションの 1 つとして，日々の服選びを支援するシステム「Last-Minute Coordinator」を開発した．Last-Minute Coordinator では「自分の服とその組み合わせを手軽に一覧できる」「これまでに着た服の組み合わせの履歴や，その日に会う人をもとに服を推薦してくれる」さらに「SNS を通じて友達に服の相談ができる」といったことを実現する．

Last-Minute Coordinator は一般的なタッチパネル付き PC 上で利用する．図 16 のようにアウター・インナー・ボトムごとに一覧でき，それぞれのリストをスクロールすることで洋服の組み合わせを確かめることができる．

この服の一覧の中から条件によって服を絞り込む．たとえば，カジュアル・フォーマル・セミフォーマルといった条件を選ぶと，過去に選択した組み合わせや服の種類をもとに，適していない服は薄く，おすすめの服が強調表示される．

さらに,「どちらがいいと思う？」「この組み合わせは変じゃないか？」といったことを他の人に相談してみたいときは，Twitter や Facebook といった SNS を使って聞くことができる．Last-Minute Coordinator から迷っている服の組み合わせをいくつか選んで投稿すると，Web 上に動的に投票用ページが作られ，その URL が SNS に投稿される．そして，SNS から誰かが投票およびコメントすると集計結果が表示され，服選びの参考にすることができる．

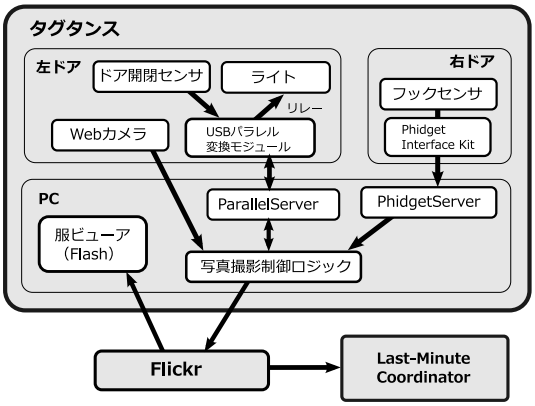

図 17　タグタンスのシステム構成

5.3.2　システム構成

タグタンスおよび Last-Minute Coordinator は図 17，図 18 のような構成になっている．

タグタンスのフック（圧力センサ）は Phidget と PhidgetServer で，ライトとドア開閉センサは USB パラレル変換モジュールと ParallellServer でそれぞれ PC から制御される．フックにハンガーが掛かると USB カメラで服が撮影され，その写真がタグ情報と

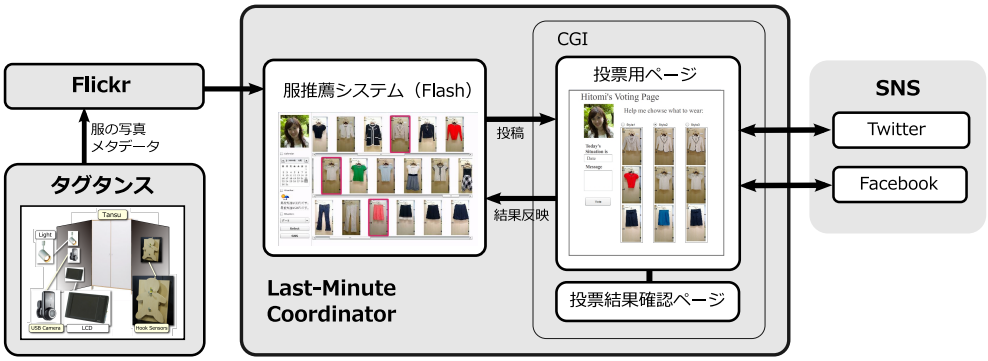

図 18 Last-Minute Coordinator のシステム構成

ともに Flickr にアップロードされる．撮影した服を一覧する服ビューア[†10]（Flash）は，Flickr の Web API を利用して写真を取得する．

Last-Minute Coordinator も服ビューアと同様に Flickr から写真を取得する．また，SNS 用に投票ページおよび集計ページを生成する CGI のサーバがあり，このサーバ経由で SNS に投稿する．

5.3.3 システムの特徴

タグタンスのシステムの特徴は，GUI とデバイスの連携に Web サービス（Flickr）を利用している点である．タグタンスは写真やメタデータを Flickr にアップロード・保存し，服ビューアは Flickr 経由でそのデータを取得している．今回は Flickr を利用したが，実際には他の写真共有サービスでも問題ない．

Flickr のような既存の Web サービスを利用した理由として「API を共通化できる」「Web API 用ライブラリにより開発しやすい」「Web ブラウザからでもアクセスできる」「ネットワーク環境からでも利用しやすい」といったことが挙げられる．タグタンスで取得したデータは複数のアプリケーションから利用されることを想定しており，どのアプリケーションからでも共通した API で扱えることが望ましい．そして Web API (REST や SOAP) のように簡単で一般的な API ほど扱いやすい．特に Flickr のようなメジャーな Web サービスでは各種言語用のライブラリが提供されており，アプリケーションを開発しやすい．また Web サービスであれば，専用アプリケーションに限らず，PC やモバイル端末の Web ブラウザからデータを閲覧・編集できるのも便利な点である．さらに，HTTP 通信はファイヤーウォールの影響を受けにくいため，セキュリティの厳しいネットワーク内でも設置・利用しやすいというメリットもある．

5.3.4 開発作業の分割

図 17 のうち，GUI にあたる部分が「服ビューア」および「Last-Minute Coordinator」，デバイスサーバーにあたるのが「写真撮影制御ロジック」「ParallelServer」「PhidgetServer」，そしてデバイスにあたるのが左右のドアに含まれるセンサやライト，Phidget Interface Kit などである．

タグタンスおよび Last-Minute Coordinator はシステムの構成要素が比較的多く複雑であったため，開発に時間はかかったものの，それぞれの構成要素がいずれも疎結合な状態であったため，効率的に開発を進めることができた．デバイスサーバの中心となる写真撮影制御ロジックは，GUI である服ビューアや Last-Minute Coordinator と外部の Web サービス（Flickr）を介してデータ（服の写真とメタデータ）をやりとりしている．そのため，デバイスと GUI のプログラムは互いに依存することなく動作し，それぞれほぼ単独で開発作業を進めることができた．実際これらのシステムの開発には 4 人ほど関わっていたが，それぞれ，タグタンスデバイス・服ビューア（Flash）・服推薦システム（Flash）・投票用ページ（CGI）というように分担，独立して作業することが可能であった．

†10 https://github.com/tsuka-lab/TansuViewer にてソースコードを公開

6 課題と展望

我々の GUI-デバイス複合開発には，問題が起こりやすいケースやいくつかの課題がある．ここでは「プロトコルが複雑な場合」「複数の開発言語を利用する場合」「限定された動作環境」「実用・商用アプリケーション開発」という 4 つの課題について，それぞれ解決方法や今後の展望について議論する．

6.1 プロトコルが複雑な場合

単一の言語で開発する場合と異なり，我々の手法では GUI-デバイス間で通信をするためのプロトコルを設計する必要がある．そのプロトコルが複雑になると，開発が難しくなるという問題がある．

IODisk やタグタンスは基本的に一方向通信の単純なプロトコルであり，通信フォーマットも単純なカンマ区切りテキストや REST，XML という一般的なテキストベースであったため，開発も比較的スムーズであった．一方，なめらカーテンはやや複雑な双方向の通信プロトコルであり，情報量も多く，バイナリフォーマットで通信内容を確認しにくい．そのため開発作業が難しいものとなった．中でも通信関連のデバッグ作業が難しくなる．

このような問題への対策としては，まずできる限りシンプルなプロトコル，通信フォーマットを選ぶということである．通信フォーマットは MobiServer で用いている単純なカンマ区切りテキストや REST，JSON，YAML などテキストベースの物が望ましい．しかし，要求仕様によってプロトコルが複雑になるのは時に避けられない問題である．なめらカーテンで行っていたように，ログ出力やエラー出力を詳細・丁寧にするといった，できるだけデバッグの手間を軽減する工夫が必要である．

6.2 複数の開発言語を利用する場合

我々の開発手法では，GUI 側は前述のように Flash を，デバイスサーバの実装には C#を利用している．このように 2 種類の言語を利用するのはメリットとデメリットがある．

メリットは，それぞれのレイヤーに適した言語を使えることである．そして，GUI デザインやハードウェア開発に慣れた人であれば，これまでの知識やスキルをほぼそのまま活かせる．

一方，デメリットとして，1 人で GUI とデバイスの両方を開発する場合，2 種類の言語や開発環境を使い分ける必要がある．どちらかに詳しくなければ新たに言語やツールを習得する必要がある．また，単一言語であれば必要なかった，両者の連携という手間がかかる．

もちろん，なめらカーテンのように GUI とデバイス間を MobiServer だけで連携できるような単純なシステムであれば，実際に使用する言語は GUI 側の 1 種類で済む．しかしシステムが複雑なものになるに従って，レイヤーの分離や複数の言語の使い分けが必要になる場合も多い．将来的にミドルウェアが充実し，より多くのデバイスや通信方式に対応できれば，ある程度システムが複雑になっても単一の言語だけで開発できるケースが増えると考えられる．

6.3 限定された動作環境

我々の実装するデバイスサーバは Windows 上での動作を前提とするため，現在のところデバイスの制御は PC の利用が前提となっている．最近では手のひらサイズの超小型 PC や，ネットブックのような比較的小型で安価な PC も存在するが，さらなる小型化や省電力化，低コスト化には限界がある．

ユビキタスコンピューティングでは様々な場所や機器にいくつものコンピュータやデバイスを組み込むことが多いため，小型化や省電力化，低コスト化は重要な課題である．現状ではまだ難しいものの，将来的に MobiServer のようなミドルウェアを Android や Embedded Linux，Windows Mobile といったモバイル・組み込み系の OS で動かすことができれば，大幅な小型化や省電力化が期待できる．

6.4 実用・商用アプリケーション開発

我々の開発手法は主にプロトタイプの開発および実験環境での利用を想定しており，研究段階の試行錯誤・調整を繰り返すような段階の開発には向いているが，実用段階・商用のアプリケーションを開発にあ

たっては課題が残る．実用・商用アプリケーションを開発する際は，上述した低コスト化や小型化に加えて，パッケージ化による設置・設定のしやすさを新たに考慮する必要がある．

例えばMobiServerはデバイスの細かなパラメータ調整や動作確認をするための開発者向けのGUIを備えているが，開発者以外のユーザー自身がそれらの設定項目を操作するのは難しく現実的でない．システム内部について詳しくないユーザーにとっても設定・操作しやすいGUIが別途必要になる．

また，本手法で開発したシステムはPCやデバイスといったハードウェアや，デバイスサーバ，Flashを用いたGUIなどのソフトウェアで構成されるが，これらをユーザー自身で構成し設置するのは手間がかかり難しい．できるだけこれらのシステムを単一のパッケージに収めて簡単に導入できるような工夫が求められる．

7 まとめ

本論文では，ユビキタスコンピューティングの研究開発において，GUIと多様なデバイスが連携したアプリケーションの開発手法を紹介した．

GUI-デバイス複合システムの開発では，既存のデスクトップアプリケーション用のGUIデザインツールが使いにくいことや，GUIとデバイスというレイヤーの違いから両者の適切な分離・連携が問題となる．このような問題に対して，我々はこれまでにFlashとデバイスサーバを連携した手法により「IODisk」「タグタンス」「なめらカーテン」といったシステムを開発してきた．

個別のシステムに求められる要件に応じてシステムの構成方法は変わるものの，利用目的や利用形態によっていくつかのパターンに分類でき，それぞれのパターンに適した構成・通信手段がある．また，この開発手法において問題の起こりやすいケースとその対処法，将来の課題などについて検討した．

ユビキタスコンピューティングにおける開発手法はまだ確立しておらず，多くの人がいまだ試行錯誤を繰り返している手探りな段階である．我々を含めたユビキタスコンピューティングの開発に関わる人が，それぞれのノウハウやツールといった開発手法を公開・共有し，改善してゆくことで，汎用的で効果的な開発手法を実現・確立してゆきたい．本稿がそのための一助となり，ユビキタスコンピューティングの発展につながることを期待している．

謝辞 本研究の一部は，科学技術振興機構さきがけプログラムの支援を受けた．

参考文献

[1] Greenberg, S. and Boyle, M.: Customizable Physical Interfaces for Interacting with Conventional Applications, In Proceedings of the ACM Symposium on User Interface Software and Technology (UIST2002), 2002, pp.31–40.

[2] Handa, T. Kambara, K. Tsukada, K. and Siio, I.: SmoothCurtain: privacy controlling video communication device, Adjunct Proceedings of Ubicomp2009, 2009, pp.186–187.

[3] 門村亜珠, 塚田浩二, 馬場哲晃, 串山久美子: ハングル ガングル: ハングル文字学習のためのインタラクティブ玩具, 情報処理学会 インタラクション 2011 論文集, 2011, pp.789–793.

[4] Kobayashi, S. Endo, T. Harada, K. and Oishi, S.: GAINER: a reconfigurable I/O module and software libraries for education, In Proceedings of the 2006 Conference on New Interfaces for Musical Expression (NIME 2006), 2006, pp.346–351.

[5] 小松崎瑞穂, 塚田浩二, 椎尾一郎: DrawerFinder: 収納箱に適した物探し支援システムの提案と運用, 情報処理学会第 73 回全国大会講演論文集, 2011, pp.1-603-604.

[6] 小谷尚子, 塚田浩二, 渡邊恵太, 椎尾一郎: お弁当箱を介したコミュニケーション支援システム, 情報処理学会 インタラクション 2011 論文集, 2011, pp.239–242.

[7] Masui, T. Tsukada, K. and Siio, I.: MouseField: A Simple and Versatile Input Device for Ubiquitous Computing, Proceedings of UbiComp2004, Springer LNCS3205, 2004, pp.319–328.

[8] Mellis, D, A. Banzi, M. Cuartielles, D. and Igoe, T.: Arduino: An Open Electronics Prototyping Platform, In Alt CHI, CHI 2007, Apr. 2007.

[9] 小笠原遼子, 山木妙子, 塚田浩二, 渡邊恵太, 椎尾一郎: インタラクティブな掃除機, エンタテインメントコンピューティング 2007 講演論文集, 2007, pp.71–74.

[10] Tsujita, H. Tsukada, K. Kambara, K. and Siio, I.: Complete Fashion Coordinator: A support system for capturing and selecting daily clothes with social network, Proceedings of the Working Conference on Advanced Visual Interfaces (AVI2010), 2010, pp.127–132.

[11] Tsukada, K. and Kambara, K.: IODisk: Disk-type I/O interface for browsing digital contents, Extended Abstracts of UIST2010, 2010, pp.403–404.

[12] Tsukada, K. Tsujita, H. and Siio, I.: TagTansu:A Wardrobe to Support Creating a Picture

Database of Clothes, Adjunct Proceedings of Pervasive2008, 2008, pp.49–52.
[13] 塚田浩二: 日曜ユビキタスのための手軽なミドルウェア, 日本ソフトウェア科学会論文誌 (コンピュータソフトウェア), Vol.27, No.1, 2010, pp.3–17.
[14] 鄭顕志, 中川博之, 川俣洋次郎, 吉岡信和, 深澤良彰, 本位田真一: ユビキタスコンピューティングにおけるアプリケーション開発手法に関する研究動向, 日本ソフトウェア科学会論文誌（コンピュータソフトウェア）, Vol.25, No.4, 2008, pp.121–132.
[15] 渡邊恵太, 塚田浩二: EyeWish: 目を閉じることを利用したインタラクション手法, ソフトウェア科学会 WISS2009 論文集, 2009, pp.81–84.