

ユビキタスコンピューティングに適したGUI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インターフェース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なる。よって，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。本論文では，我々がこれまでに取り組んできた多数の実世界指向インターフェースの開発経験を元に，GUIと多様なハードウェアを連携したプログラミング手法について，具体的な開発方法や様々なノウハウ，課題について紹介する。

Many research projects on novel user interfaces 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 GUI since classical IDEs (Integrated Development Environment) do not support various I/O devices other than keyboards and mice. In this paper, we introduce practical development techniques of GUI for ubiquitous computing 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実世界指向インターフェース（以下，ユビキタスインターフェース）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^[3]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^[4]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。一方，実際にユビキタスインターフェースを開発する際には，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れたプログラムの設計／開発手法（e.g., 構造化プログラミング手法，アジャイルソフトウェア開発）を単純に適用できないケースも多い。

本論文では，我々がこれまでに取り組んできた多数のユビキタスインターフェースの開発経験を元に，GUIと多様なハードウェアを連携したプログラミング手法

A development method of GUI suited for ubiquitous computing

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.

[研究論文] 1999年8月3日受付.

†1 <http://processing.org/>

について、具体的な開発方法や様々なノウハウ、課題について紹介する。

1.1 ユビキタスコンピューティングにおける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^{†2}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が行われるため、WIMP方式とは違ったデザインが求められる。また、「なにか別の作業をしながら利用する」「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り、求められるデザインのバリエーションが広い。

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のユーザインターフェースエディタやXcodeのインターフェースビルダー、HTMLオーサリングツールのDreamweaverといった従来のGUIデザインツールを適用することが難しい。

1.2 GUI開発とデバイス開発

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い、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。

1.3 密結合問題

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が、両者の結びつきを密接にしそうな構成・実装にすると、同時並行で開発作業を進めるのが難しくなってしまう。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。

2 ユビキタスデバイス-GUI複合開発

ここでは、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。

システムの主要な構成要素・技術として、GUIの開発には主にAdobe社の「Flash」を用い、デバイス-GUI間の仲介およびデバイス制御には「デバイス

^{†2} Window, Icon, Menu, Pointing device

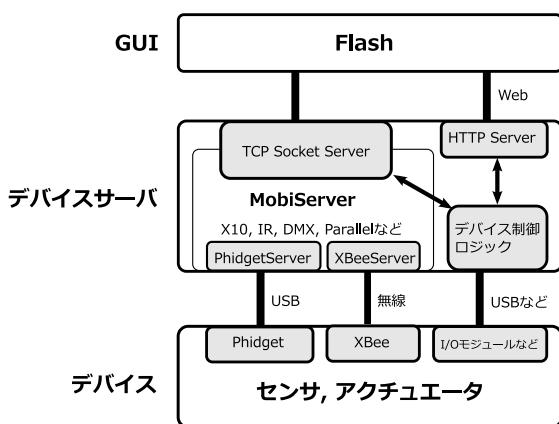


図 1 ユビキタスデバイス-GUI 複合システムの構成

「サーバ」と言われるミドルウェアを用いる(図1)。こうして GUI とデバイス制御を分離し「疎結合」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かつても連携可能にすることができる。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。

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。

2.1 Flash

Adobe Flash は主に RIA^{†3}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。

ユビキタスデバイス-GUI 複合開発に Flash を利用した理由は以下のようなものである。

- 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、一つの開発環境で、様々なシステム構成に柔軟に対応できる。

応できる。

- デザインの自由度が高く、デスクトップやモバイル、Web に限らず様々な種類の GUI を作ることができる。アニメーション用のタイムラインやグラフィックツールを内蔵しており、動きのある UI や凝った見た目の UI も作りやすい。
- 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。
- 簡易かつ強力なスクリプト言語(ActionScript)を備えており、エンジニアのみならず、多くのデザイナーにも利用されており、開発の敷居が低い。

2.2 デバイスサーバ

Flash は上述のように GUI 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^{†4}、直接外部デバイスを制御することは難しい。

よって、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。特に、API を提供する方法として、TCP Socket サーバとして動作させことが多い。これは、Flash が HTTP または TCP Socket で外部ネットワークにアクセスすることができ、HTTP に比べてより高速で双方向通信可能な TCP Socket の方がデバイスの制御に適しているためである。

さらに Flash のクライアントが無い状態、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ようにする機能も開発する上で重要となる。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。

^{†4} 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} Rich Internet Application

2.2.1 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[8]。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

2.3 疎結合

ユーザーから見たときの GUI とデバイスは密接に関係していたとしても、開発・実装レベルではお互い独立性を保ち、疎結合の状態にすることが重要である。そのために、GUI 関連 (Flash) とデバイス制御関連 (デバイスサーバ) はそれぞれ別々のプログラムに分離し、独立して実行・開発・テストできるようになる。それぞれのプログラムを別々の PC で動かせるようにすることで、複数人で作業を分担しながら同時に並行で開発・テスト・デバッグ作業を行うこともできるようになる。

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。

3 システム構成の分類

ユビキタスコンピューティングにおけるシステム構成のバリエーションは広いが、ジャンルに応じていく

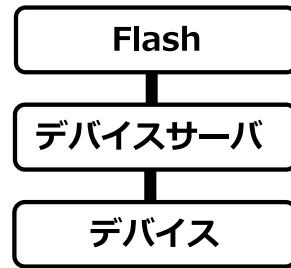


図 2 入出力デバイス系

つかの典型的なパターンに分類できる。その中でも、デバイスと GUI を組み合わせたシステムとしては以下のようないがある。

- 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、それぞれの分類について述べると共に、それに適したシステムの構成方法について述べる。

3.1 入出力デバイス系

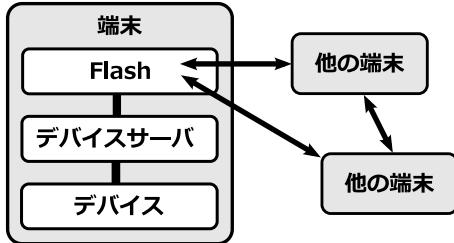
センサを用いて GUI を操作、または GUI に応じてアクチュエータで出力するシステムを指す。GUI とデバイスが 1 対 1 の関係になっており、Flash とデバイスサーバを使った最も基本構成で実装できる（図 2）。

3.2 コミュニケーションデバイス系

図 3 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^{†5}、Flash 間で P2P 通信ができる RTMFP^{†6}などがある。

^{†5} Real Time Messaging Protocol

^{†6} Real Time Media Flow Protocol



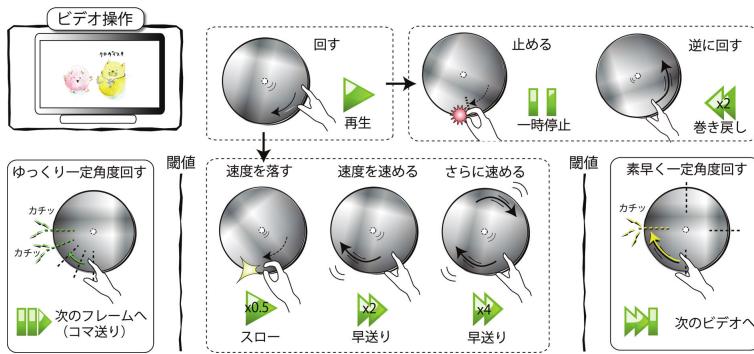


図 6 IODisk

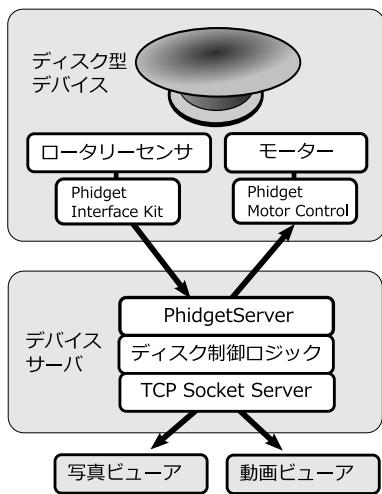


図 7 IODisk のシステム構成

ク型デバイスが手元に無い場合でも単独で開発できるように、キーボード操作で回転速度や方向、急速回転コマンドをエミュレートできるようにした（リスト1）。

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 ~ ~ ~
    }
}
```

図8のように、IODiskは主にリビングのテレビなどで利用を想定しており、画面から少し離れた位置で操作できることが求められた。そこで、手元に置けるディスク型デバイス制御用のコンピュータと、テレビ側に置く写真・動画ビューア用のコンピュータに分けることで、デバイスを画面から離れた位置に置けるようになっている。

4.2 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。



図 8 IODisk の利用シーン

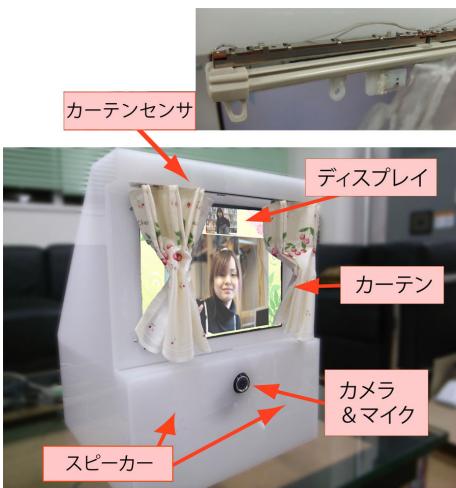


図 9 なめらカーテン端末

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ー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 9 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

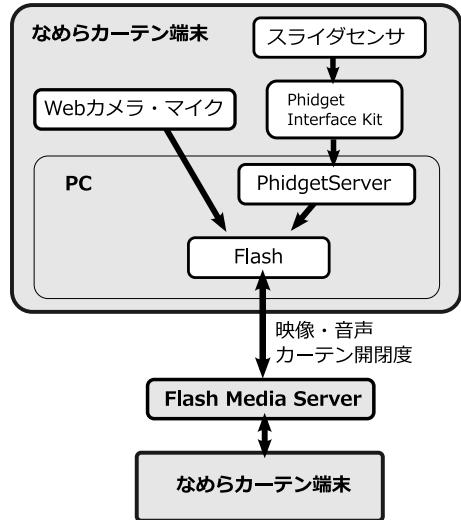


図 10 なめらカーテンのシステム構成

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、どれく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。

4.2.1 システム構成

なめらカーテンは図 10 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 FlashMediaServer 経由で相手に送る。

4.2.2 システムの特徴と開発手法

なめらカーテンの実装上のポイントは、遠隔地に置かれた2台のビデオチャット端末（中身はPC）間で常時リアルタイムに通信しあう点と、カーテン型のデバイス（スライダセンサ）を用いて操作する、という2点である。

1点目の端末間の常時リアルタイム通信では、Flash用のストリーミングサーバ（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）。

2点目のデバイスの制御では、PhidgetおよびPhidget.getServerによりスライダセンサの値をFlashで取得でき、新たにデバイス制御のためのプログラムを作る必要は無かった。

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
    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();
}
```

Flashの開発環境の一つであるFlex Builder上で基本的に全てのコーディングおよびデバッグ作業を行うことができた。ただし、実際には実運用中まれに発生するような問題もあり、この場合「デバッガを使えない」「問題箇所を特定しにくい」「再現性が低い」といった理由により、原因調査に時間がかかることが多かった。そこで、Flash側でデバイスとの通信内容/FMSとの通信内容/ストリーミング関連など様々なログを書き出すようにし、そのログを元に原因を調査した。

4.3 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[7]。図11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12のようにアウター・インナー・ボトムス用の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。

4.3.1 Last-Minute Coordinator

タグタンスを利用したアプリケーションの一つとして、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

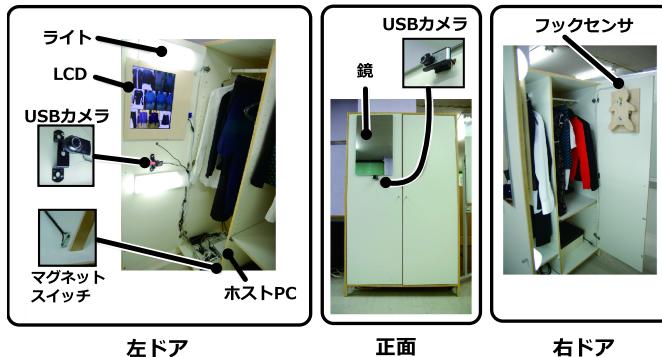


図 11 タグタンス

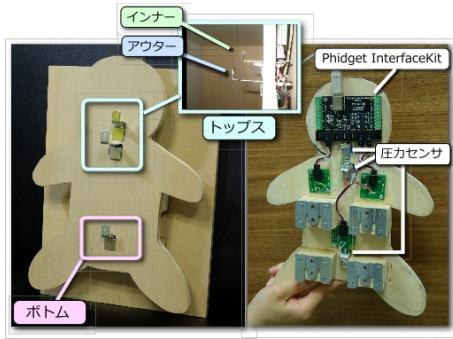


図 12 タグタンスのフックセンサ

「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 13 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



図 13 Last-Minute Coordinator

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。

4.3.2 システム構成

タグタンスおよび Last-Minute Coordinator は図 14、図 15 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 ParallelServer 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（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

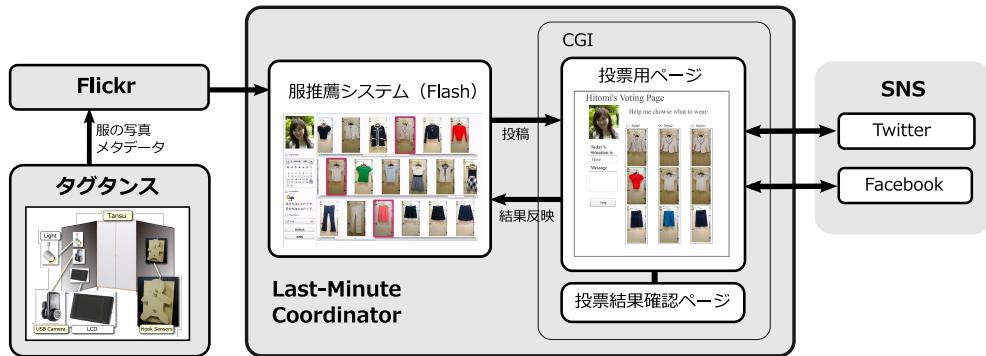


図 15 Last-Minute Coordinator のシステム構成

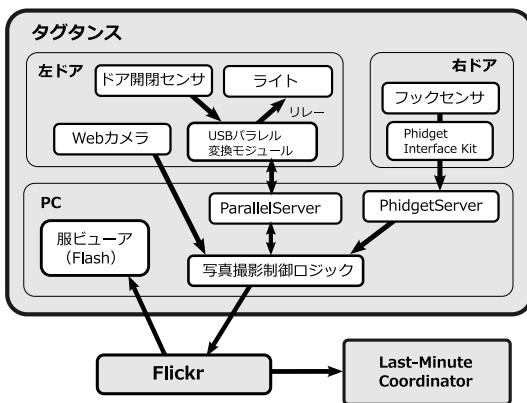


図 14 タグタンスのシステム構成

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。

4.3.3 システムの特徴と開発手法

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 通信はファイア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 課題・問題点

我々のユビキタスデバイス-GUI 複合開発には問題が起りやすいケースやいくつかの課題がある。

5.1 プロトコルが複雑な場合

单一の言語で開発する場合と異なり、我々の手法ではデバイス-GUI 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。

5.2 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。このように二種類の言語を利用するの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、单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 側の一種類で済む。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。

5.3 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 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ユビキタスインターフェースの開発手法を紹介した。

ユビキタスデバイス-GUI 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ため、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анс」「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、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。

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 Siiro, I.: SmoothCurtain: privacy controlling video communication device, Adjunct Proceedings of Ubicomp2009, 2009, pp. 186 – 187.
- [3] 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.
- [4] Mellis, D. A. Banzi, M. Cuartielles, D. and Igoe, T.: Arduino: An Open Electronics Prototyping Platform, In Alt CHI, CHI 2007, Apr. 2007.
- [5] Tsujita, H. Tsukada, K. Kambara, K. and Siiro, 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.
- [6] Tsukada, K. and Kambara, K.: IODisk: Disk-type I/O interface for browsing digital contents, Extended Abstracts of UIST2010, 2010, pp. 403 – 404.
- [7] Tsukada, K. Tsujita, H. and Siiro, I.: Tag-Tansu:A Wardrobe to Support Creating a Picture Database of Clothes, Adjunct Proceedings of Pervasive2008, 2008, pp. 49 – 52.
- [8] 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), 2010, pp. 3 – 17.