本当の制御に使えるAndroidの今
ーAndroidとアルディーノを接続した制御ー

2012/OCT/09

竹岡尚三
(株)アックス,
OSSコンソーシアム副会長,
JASA 理事 技術高度化委員会委員長

Androidとは

Androidとは

- Googleが開発、無料で配布
  - ソースコードも開示、改変可能
    - いわゆるオープンソース
  - ライセンスは部分により異なる
- 使用も無料
- マスコット・シンボルも誰でも無料で使える
- 開発環境も無料

Androidのライセンス

- Androidのソースコード
  - アプリケーション・フレームワーク
  - 標準ライブラリ
  - ランタイム
  - カーネル
  - すべて無償で公開されている
  - 改変も自由
- ソースコードのライセンス
  - 基本的にApache 2.0
    - Androidのライセンスでは、公開する必要がない
  - WebKitのライセンスはLGPL
  - カーネルとそのライブラリなどのLinux部分のライセンスはGPL
    - Linux部分はGPL系、公開が求められる

オープンソース・ソフトウェアのライセンス

Androidのライセンス

Androidを使うだけなら、気にしなくていい
　アプリケーションを書くだけなら安全

デバイスドライバは、気をつける

- Linuxカーネルに直接リンクすると、GPLに従って、公開しなければならない

オープンソースの主なライセンス

GPL

Linux, gccなど有名なソフトウェアの多くがGPL

結構、厳しい（後述）

BSD

Apache, Mozilla などは、BSDの派生と考えられている

GPL

ソフトウェアはすべからく無料でなければならない

ソフトウェアはソースコードが見れなければならない

ソースコードは改変可能でなければならない

改変したソースコードも無料で再配布されなければならない

有償ビジネスは否定しない

- ソフトウェアの維持、品質向上
- ソフトウェアのコンサルテーション
- 再配布を有償で行う
  - フリーソフトウェアCDROMを販売など

商品にオープンソース・ソフトウェアを入れても構わない

誰にもお金を払う必要はない

誰とも契約しなくてよい

共産主義ではない

GPLとLGPL

GPLライセンスのサブルーチン

それを、リンクしただけで、すべてのソフトウェアを公開しなければならない

LGPL

サブルーチンを、広く使ってもらうためにある

LGPLのサブルーチンは、リンクしても、公開義務は発生しない

- 実は、多くのLGPLソフトウェアはダイナミック・リンクであることを要求している
  - 実時間OSでは、結構、難しい要請

LGPLのソフトウェアそのものを改変したら、それは公開しなければならない

Apache, BSDライセンス

もとのソフトウェアは無料

ソースコードは自由に改変できる

何に使ってもよい
　商品に使用しても、
　　もとの作者にお金を払ったり、
　　契約を交わしたりする必要がない

改変したソースコードは配布しなくともよい
　有償／無償で配布してもよい

BSDライセンスのソフトウェアをリンクしても、何をしても、何の義務も生じない

組込みとオープンソース・ライセンス

- BSDは、優しい
  - 何をしても、何も公開しなくてよい
- GPL, LGPL
  - 改変したソースコードも無料で再配布されなければならない
    - 頑張って、高性能化、高品質化しても、無償で公開
    - ライバルにも使用される
  - デバイス・ドライバは、公開しなくてすむ方法がある
    - 組込みは、特殊なデバイスを使用/駆動する
    - ライセンス問題の専門家に相談すべき
- GPLを回避するやり方は決まっている
  - 怖くはない
  - わざと怖がらせている人がいる。
  - 不安をあおっている人は信じなくてよい
    - パナソニック、SONY,シャープなどは、ずっとうまくやっている

Java問題

- Javaのライセンス問題は鬼門
  - 素人は近寄ってはいけない

- 1998～99年頃、私が調べていた時の記憶(誤っているかもしれない)
  - Javaの言語仕様はフリー
    - 言語仕様を不自由にすると、言語が流行しないから?
    - コンパイラなどを作るのは自由
  - 仮想機械 JavaVMの機械語の命令セットは、ライセンス対象
    - 機械語の命令セットは、x86,ARMなどはライセンスされる対象である
  - クラスライブラリの集合は、ライセンス対象
    - 他の例はあまり無いかも知れない

- 当時から、私にとっては、面倒臭かった(個人の主観的な感想です)

Java問題

- Googleは、Javaの特許は、侵害していない。と決着

しかし

- クラス・ライブラリのセットについての権利などについて、

Oracleと引き続き、係争中

Java問題

- 普通人は、無料でJavaは使えないと思った方が、いい
- Oracle (SUN)が「いい/悪い」という話ではないと思う

- 確かに、Android Dalvikは、
  - VMの機械語は、Sun(Oracle)のVMとはまったく違う
  - クラスライブラリのセットの仕様も違う
    - でも、整数とか、文字、文字列のような基本的なものは、Sun Javaと同じ
  - Java言語を使っているだけ

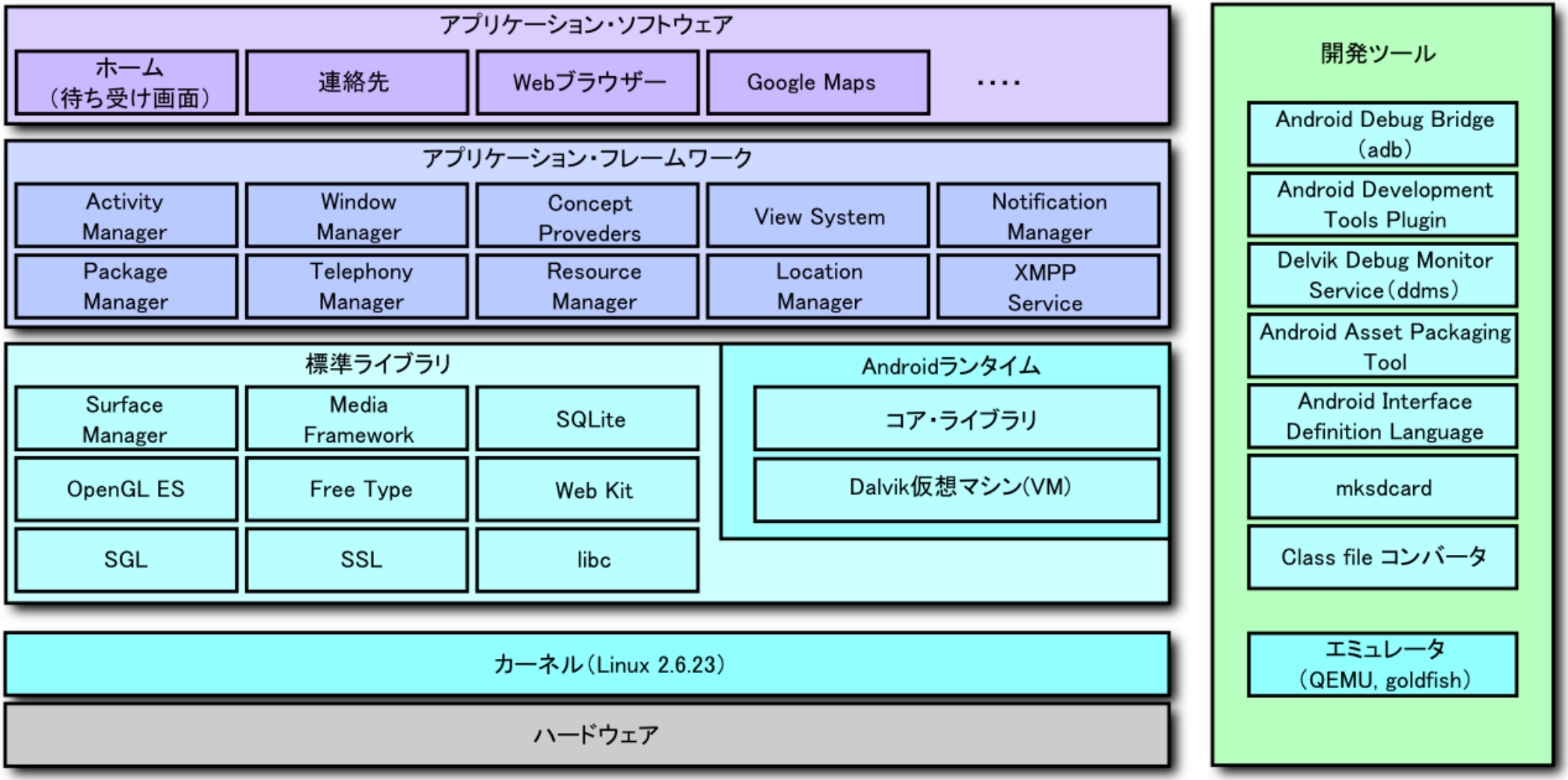
- Androidは、Javaを止めて、違う言語に乗り換えた方がいいと思う(個人の感想です)

Androidの
ソフトウェア

ソフトウェア構成

- Linux
  - ファイル・システム
  - ネットワーク
- Java技術
  - SUNの標準Javaではない
- GUI
  - 2D 派手なアニメーション付きが標準
  - 3D(三次元) 派手

Androidのアーキテクチャ



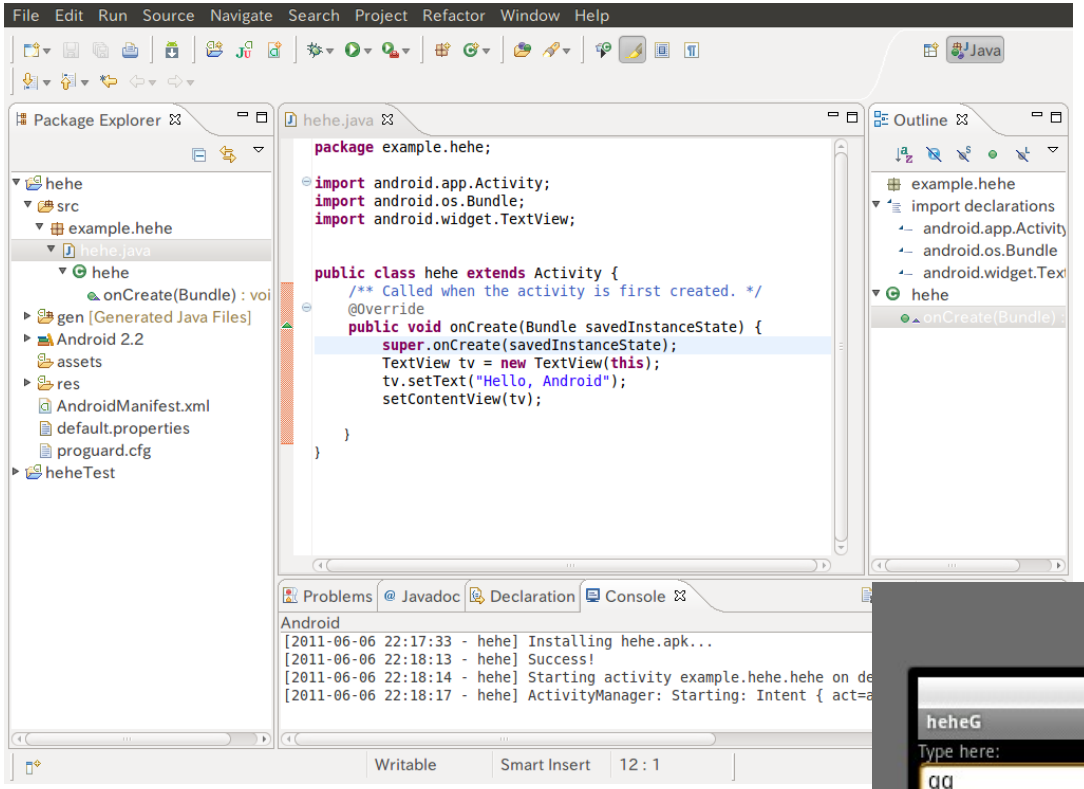
- 蓬田宏樹、他著「Androidの野望」日経エレクトロニクス 2007年12月17日号 p.47-69 より引用

ソフトウェア構成

- Linux
  - 実は、標準Cライブラリは、BSD由来で、Linuxとは異なる
    - ライセンスも異なる(緩くなっている)
  - 純粋なLinuxだと思っていると、ときどき、驚くことがある
- Java技術
  - Dalvik VM 機械語がJavaVMと異なる
  - クラス・ライブラリはまったく独自
  - 文法はJavaだが、Javaだと思っていると驚く
- GUI
  - 2D: マイナーなSkia Graphics Library
  - 3D: OpenGL/ES (業界標準)

Androidの開発環境

- 開発環境 = 統合環境 + エミュレータ



Androidの開発環境

- 統合環境 = Eclipse + AndroidSDK + NDK
  - Eclipse
    - 無料の統合環境。Google製ではない
  - Android SDK
    - Androidアプリケーション開発環境。Google製。無料
  - NDK
    - C,C++言語などの開発環境。Google製。無料
    - Javaで書ききれないハードウェアに関係する部分を書く
    - 高速性が必要な部分を書く
- エミュレータ
    - 実機とそっくりな画面と、動きを、開発PC上で実現
    - Google製。無料
    - 実機どおりにコンフィギュレーション可能
      - 特に、画面サイズを合わせられるのが便利
- 上記は、Linuxでも動く
  - すべてが無料で!!!

Androidとは

- 携帯電話向け だけではない
  - iPhoneの対抗 だけではない

- 通常の組込みにも使用できる
  - 制御パネル
  - デジタル・サイネージ(広告表示器)
  - 大画面も大丈夫
  - Cでも、プログラムできる
    - Java, Dalvikは遅い、という用途に

Androidの開発プラットフォーム

- アットマークテクノ社のアルマジロシリーズが大人気
- Armadillo-500 FX, Armadillo-440

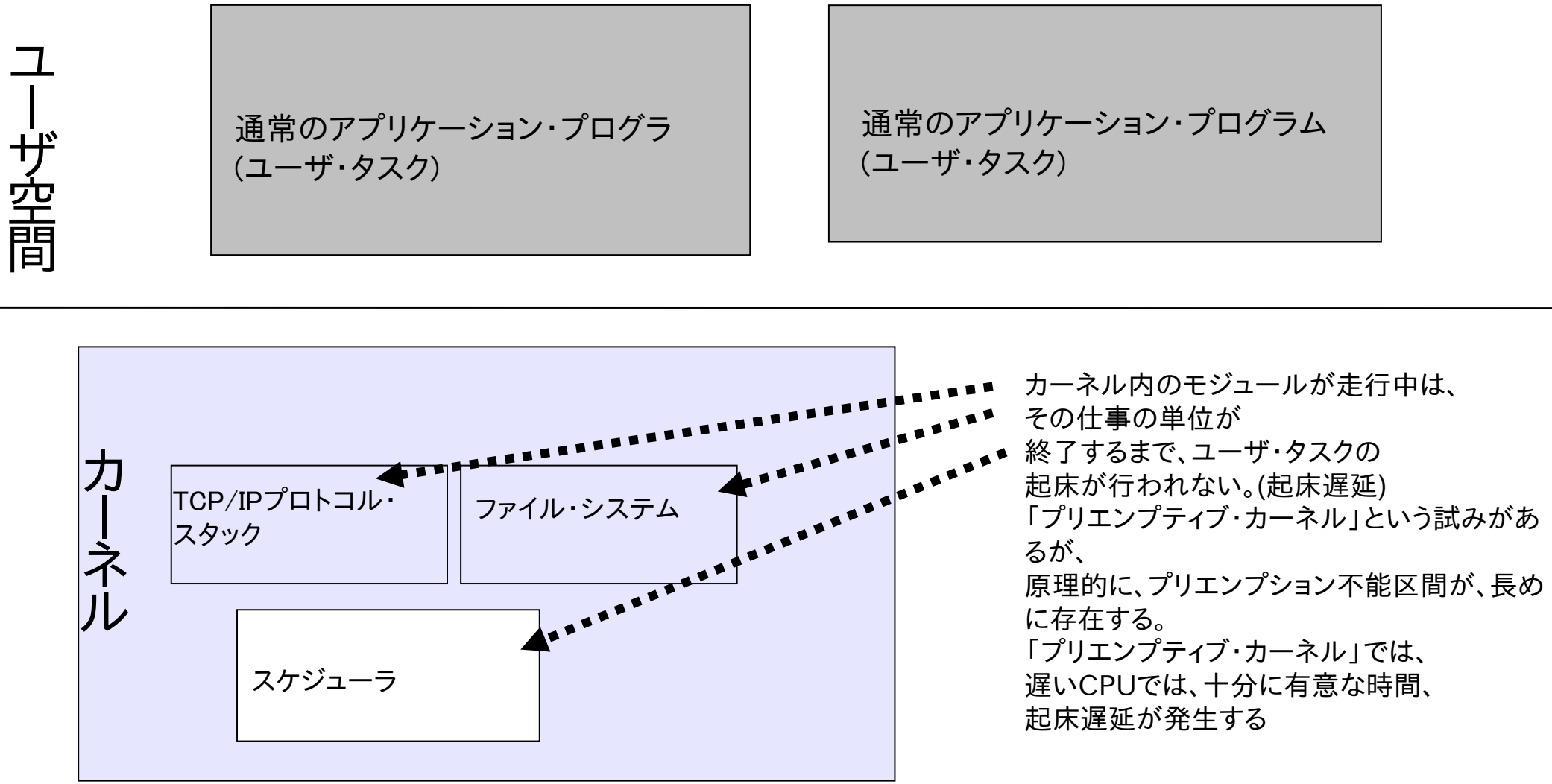
  `http://armadillo.atmark-techno.com/android`

  - ARMコア
  - タッチパネル液晶インターフェース
  - シリアル、LAN、USB 2.0、microSD スロット、GPIOなど
  - 開発に使用できる
  - 用途によっては、そのまま組み込める

実時間性について

- 実時間性とは…
  - 最悪時間保証である
  - 反応がいい、というのは実時間性とは、ほぼ関係ない
    - すごく遅いマイコンが対象でも「実時間性」の議論はできる
  - 平均の応答性能がいくら良くても、最悪時間を保証できなければ *実時間性は無い*
  - 「ソフト・リアルタイム」という言葉も概念も無い(間違っている)
    - 設計された最悪値を、ときどき越える、というのは、実時間ではない
- 通常の版のLinux, Windowsで、実時間性を実現するのは不可能
  - Androidも同様
- ダマされて夢を見てはいけない

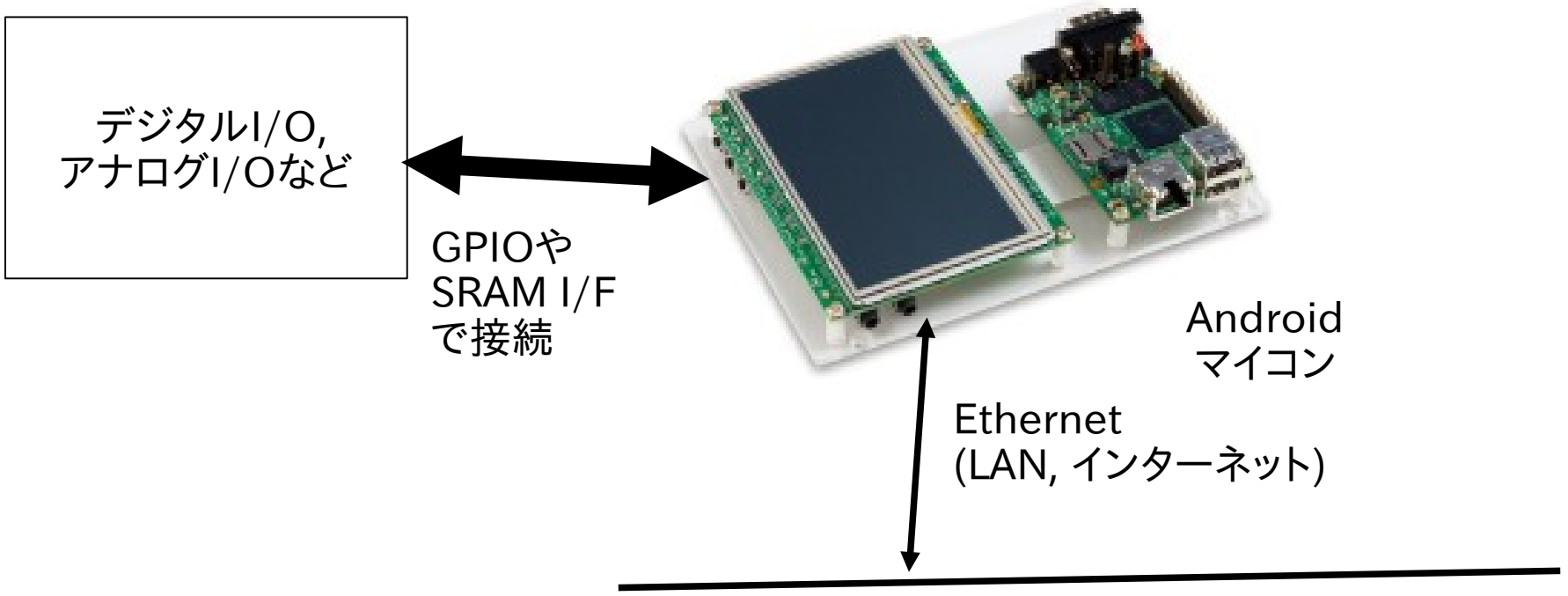
一般的にLinux, Windowsなどで実時間性を阻害する要因



現実の組込み
と
Android

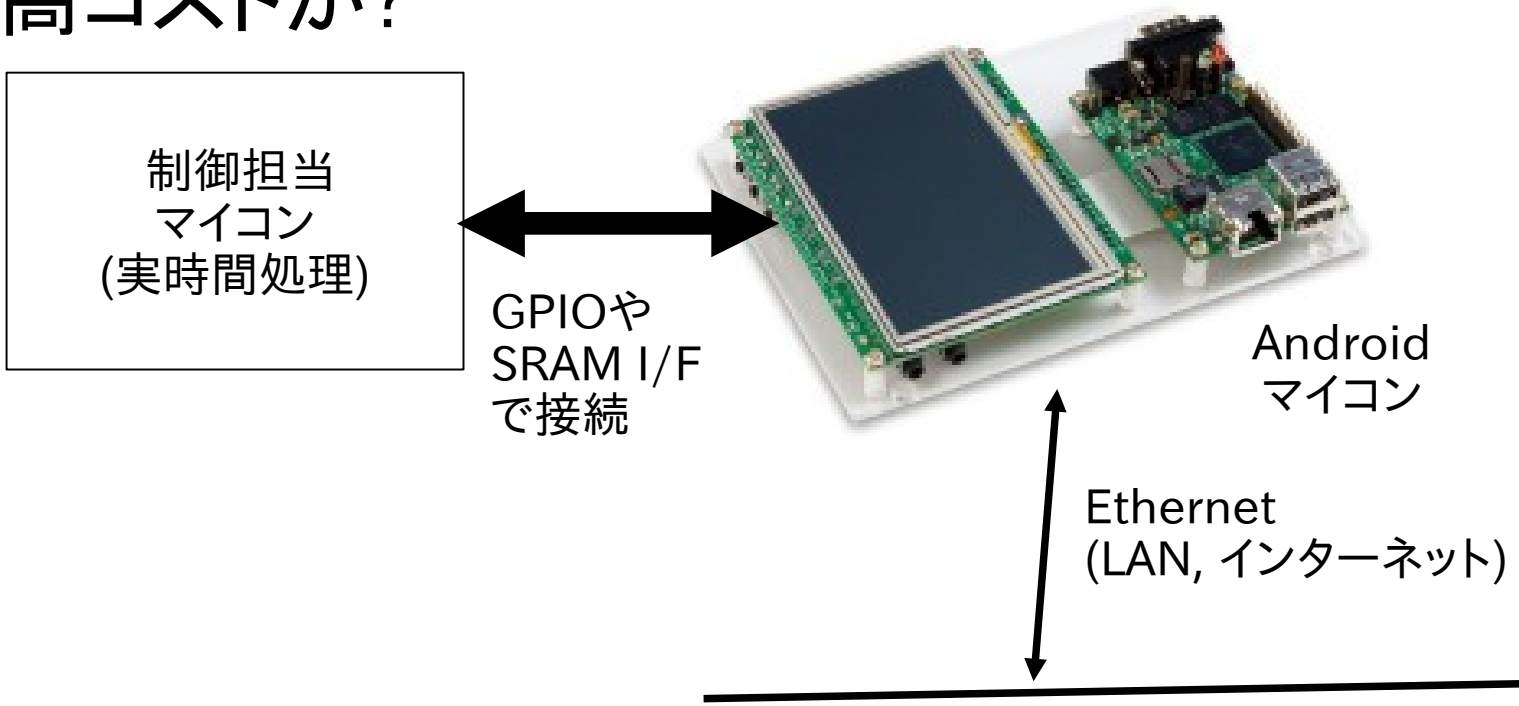
Androidマイコン単独

- Androidマイコンは、32bitの高速なCPU。メモリもかなり持っている
- 時間制約が緩ければ、プログラムはJavaで書ける
- C言語でプログラムするのも良し
- ネットワークには簡単につながる。
- GUIは派手
- 最悪時間保証が必要なものは、無理。Linuxだから



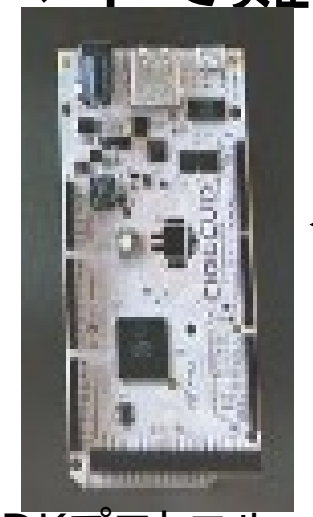
Androidマイコン+マイコン

- Androidマイコンで、GUIとネットワーク
- 実時間処理は、専用マイコンで！
- ネットワークには簡単につながる
- GUIは派手
- 最悪時間保証が必要なものは、専用マイコンで
- 高コストか?



Androidマイコン+ADK Arduinoマイコン

- Androidマイコンで、GUIとネットワーク
- 実時間処理は、専用マイコンArudinoで！
- ネットワークには簡単につながる
- GUIは派手
- 最悪時間保証が必要なものは、専用マイコンで
- ADKは、Android 2.3.3より新しいものでしか動かない(通常スマホで難しい)



ADKプロトコル
Arduinoマイコン
(実時間処理)

USB接続

Android
マイコン

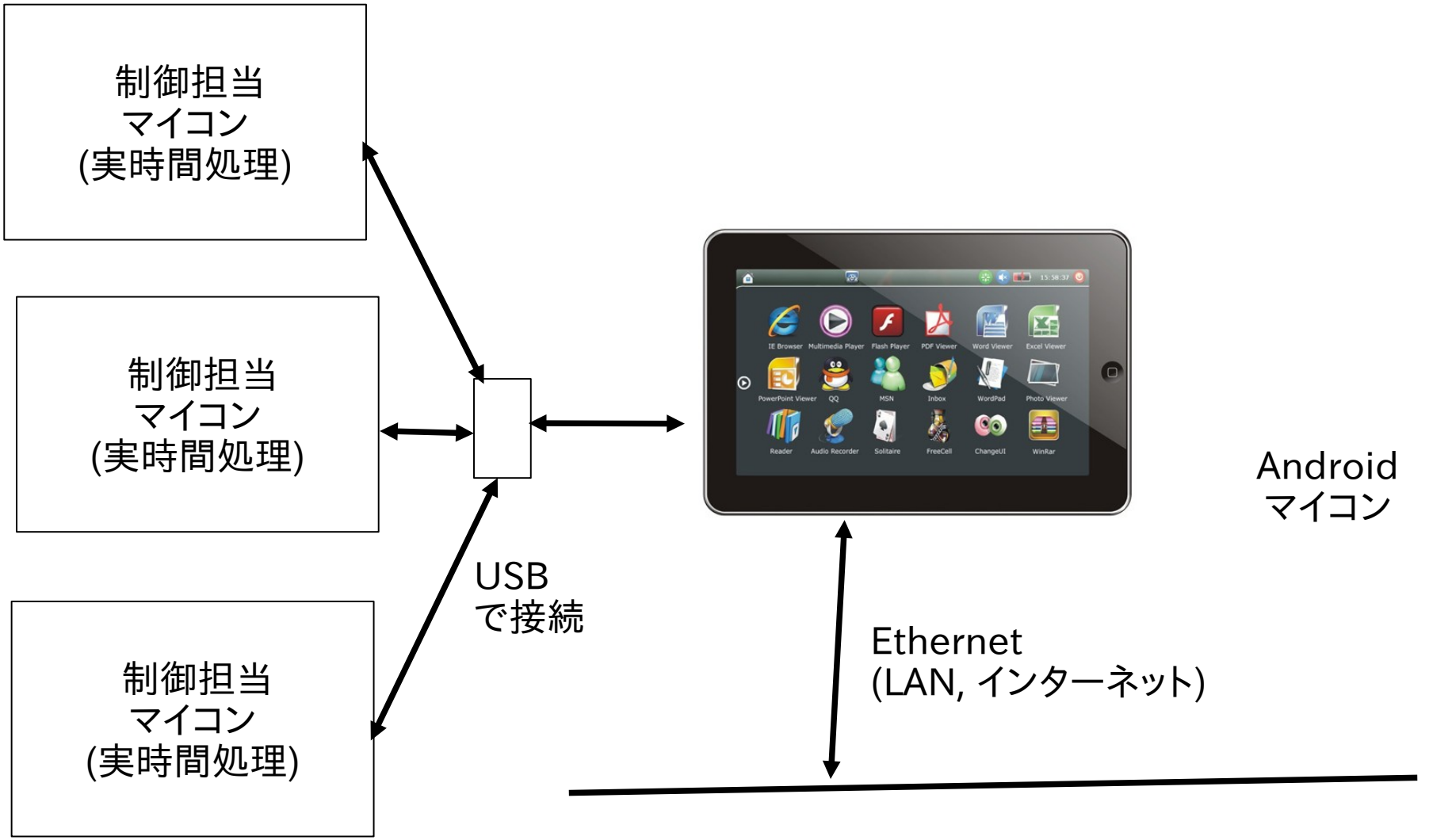
Ethernet
(LAN, インターネット)

Androidマイコン+Microbridgeマイコン

- Androidマイコンで、GUIとネットワーク
- 実時間処理は、専用マイコンArudinoで！
- ネットワークには簡単につながる
- GUIは派手
- 最悪時間保証が必要なものは、専用マイコンで
- Microbridgeは、Android 2.3でも動く(スマホでOK)

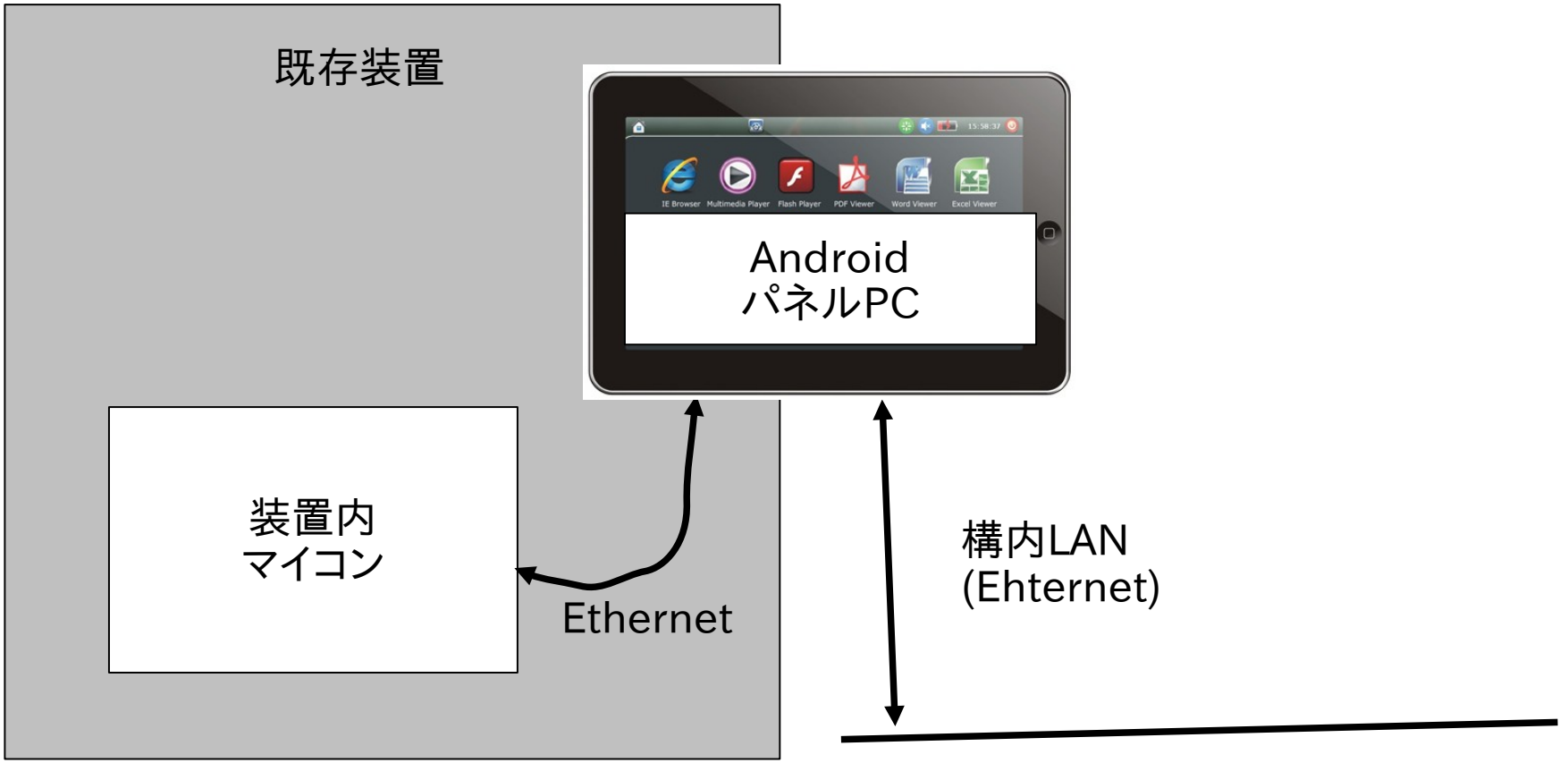
Androidマイコン+マイコン達

- 実時間処理は、専用マイコンを複数
- 複数のマイコンの面倒をみれば、まぁまぁなコスト(か?)



Androidマイコン+高速マイコン

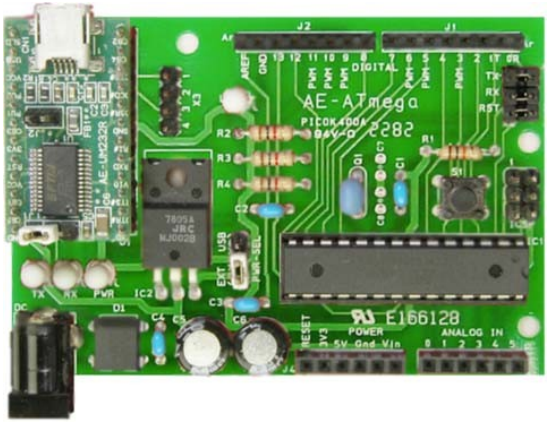
- Androidの入ったIT機能サポート用パネルPCを装置に付加
- ITパネルPCと制御マイコンは、Ethernetなどで装置内部で接続
- メールクライアント、SMBサーバなどは、ITパネルPCに行わせる
- 高付加価値なインテリジェント装置に



Arduinoマイコン
と
Android

Arduino

- Arduino（アルドゥイーノ）は、AVRマイコン、入出力ポートを備えた基板
- C言語風のArduino言語と，その統合開発環境から構成されるシステム
- オープンソース・ハードウェア
  - ハードウェア設計情報のEAGLEファイルは無料で公開
- Arduinoプロジェクトは2005年にイタリアで始まる
- Wikipedia　http://ja.wikipedia.org/wiki/Arduino より抜粋
- Arduinoプロジェクト
  - http://arduino.cc/en/Main/HomePage



秋月電子のAE-ATmega ;ATmega328P; ￥2,100円

Arduino

- 統合環境と、Java風(?)の言語
  - すべて無料
  - オープンソース
  - 例題も多く、とっつきもよい

Arduino MEGA 2560

- Arduino MEGA 2560 (ATmega2560搭載)
- AtmelのATmega8U2チップにファームウェアを搭載しUSBシリアル変換を実現
  - USBシリアル変換機能は通常のCDCクラスとして動作
  - FTDIのUSBシリアル変換チップは使用しない
  - スケッチの転送速度が高速化
  - ATmega8U2チップのファームウェアを書き換えで,様々なUSBデバイスとして動作可能

- ATmega2560
  - フラッシュメモリ256KB(うち8KBはブートローダーに使用)
  - SRAM　8KB
  - EEPROM　4KB
- その他仕様:
  - デジタルI/O 54本
    - うち、PWM 14本
    - I2C含む
  - アナログ入力 16本

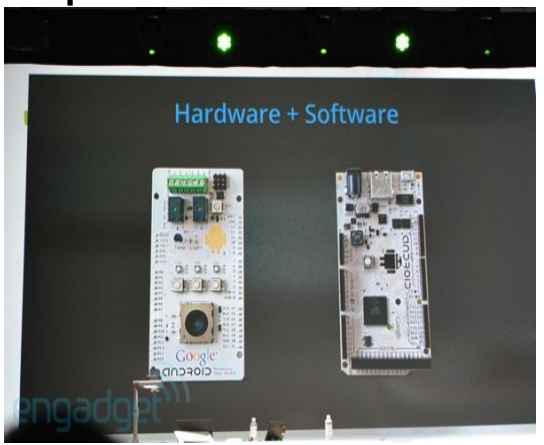
参考:販売価格(税込)： 6,495 円

http://www.switch-science.com/products/detail.php?product_id=430

http://arduino.cc/en/Main/ArduinoBoardMega2560

Android Open Accessory

- Android端末と接続するUSBハードウェアの規格
- ADKは、Android2.3.3よりも新しい版でなければ使えない
- 外部アクセサリとAndroidの接続仕様やプロトコル、ソフトウェアAPIを定めた
- Android 3.1 (Honeycomb) または 2.3.4 (Gingerbread) 以降のAndroid機器が対応
  - 外部ハードウェアがホスト
  - Android端末がUSBデバイスとして認識される
  - アクセサリ側からコネクションを開始できる
  - Android側は自動的に「アクセサリモード」に入る
  - アクセサリ側からは5V 500mAのUSB電源が給電される
- 開発用 リファレンスボード Android Open Accessory Development Kit (ADK)
  - ADK: Arduino Mega2560ベースのUSBコントローラボード
- Open Accessory、ADKはオープンソース
- ライセンス料金や守秘契約などは不要
- http://developer.android.com/guide/topics/usb/adk.html



Microbridge

Microbridgeはオープンソース

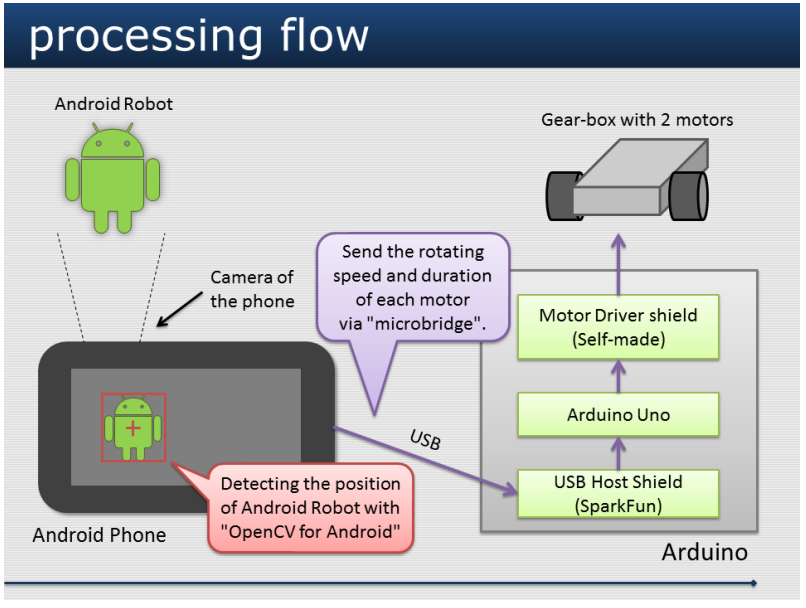
http://code.google.com/p/microbridge/

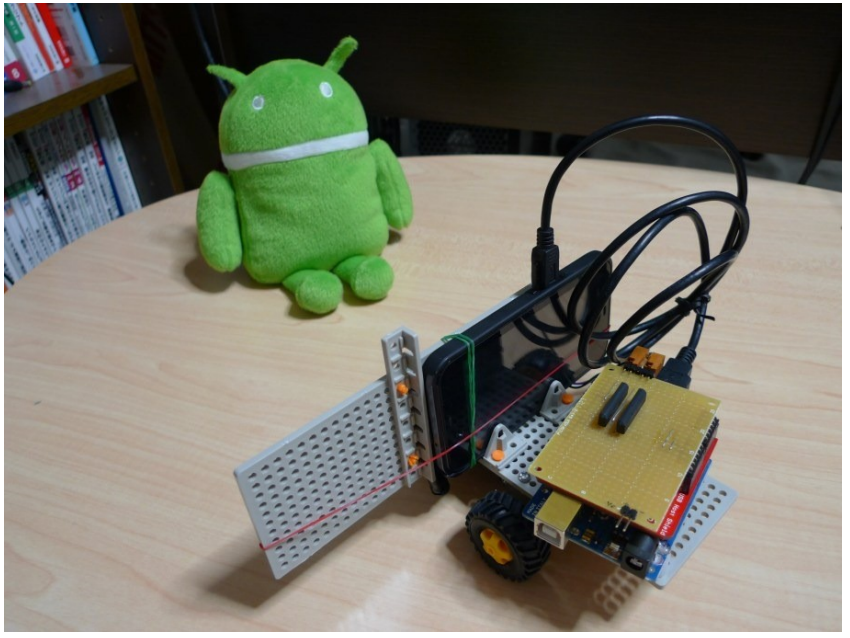
- Android開発用の Android Debug Bridge (ADB) プロトコル・スタックをマイコン用に開発
  - 設定(アプリケーション)を起動。[アプリケーション]→[開発]を選択して、「USBデバッグモード」をONにする
- Microbridgeは、古いAndroid(2.3.3以前)でも使える
- Androidを外部から操作するプロトコル (マイコンがUSBホスト)
- よく使われるマイコン
  - Arduine:若松通商、秋月などで売られているものが使用できる
  - Arduino USBシールドを使用
    - http://www.circuitsathome.com/arduino_usb_host_shield_projects
    - http://www.switch-science.com/products/detail.php?product_id=438

    (互換品 SparkFun社製 (千石などで買える))
  - PIC: 神戸で、Microbridgeでつながる PICボードを作っている人たちがいる

Aruduino + Microbridge + Android実例

- スマートフォンの上でOpenCVを動作させ、画像認識
  - 色と、物体の大きさを認識
- マイコン(Arduino)でモータを駆動
  - 認識された物体の見た目の大きさが、適当になるように、前後進する
    - 大きすぎ: 後退, 小さすぎ:前進
- スマートフォン(ARM)上のOpenCVで、実時間処理+制御
- http://www.kosaic.jp/wordpress/2012/02/model-car-with-opencv-is-tracking-android-robot-kof2011/





Arduinoマイコン
+
Microbridge
+
Android

Aruduino + Microbridge + Android実例

- Androidから、LEDをON/OFFを制御 & ADCの値を読む
- Arduino (マイコン)
  - LEDのON/OFF
  - ボリュームを接続したADCの値を送出 (2Bytes(0～1023),0.5sec 毎)
- Android (スマートフォン)
  - GUIボタン操作により、LED ON/OFF コマンドを送出
  - マイコンからやってきたADCの値を表示

futaba@256byte.comさんの書かれている下記のページを参考にしました。感謝します。

http://side2.jp/2011/08/android-shutter-control2/

Aruduino + Microbridge + Android実例

Arduinoマイコン+USBシールド
AndroidとUSB接続

ADBデバッガ機能で通信

Android→マイコン

- コマンド制御

Android←マイコン

- センサ入力

独立&非同期に可能
プログラムをうまく書けば、
高度な制御と
かっこいいGUIが
容易に
実現できる。

takeled, Arduino ソース 1/4

```
// Volume Control( Analog0) - LED0(13), LED1(D4)
// Arduino 0022
#include <SPI.h>
#include <Adb.h>

const int LED_CONN_PORT  = 4568; // ADB Conn port #

#define SER_BAUD 9600UL    // Serial Baudrate

const int LED0 = 13;       // LED : 13
const int LED1 = 4;        // LED : D4 takeoka

// Elapsed time for ADC sampling
long lastTime;

uint16_t sensorValue;

// Commnad from Android
#define CMD_HI_Down   0x01
#define CMD_HI_Up     0x02
#define CMD_LO_Down   0x03
#define CMD_LO_Up     0x04
#define CMD_SHOOT     0x05

// ADB connection.
Connection * connection;
```

takeled, Arduino ソース 2/4

```
// Event handler for the shell connection.
Void adbEventHandler(Connection * connection, adb_eventType event, uint16_t length, uint8_t * data){
 // Data packets contain two bytes, one for each servo, in the range of [0..180]
 if(event == ADB_CONNECTION_RECEIVE){
   if(length < 3) return;
   if(data[0] != 0xff){
            Serial.println("no sync");
            return;
    }
    uint8_t cmd = data[1]; // 2nd byte command
    uint8_t opt = data[2]; // 3rd byte parameter
```

takeled, Arduino ソース 3/4F

```
    switch(cmd){
    case CMD_HI_Down:
      digitalWrite(LED1, HIGH);   // LED off
      Serial.println("HI up");
      break;
    case CMD_HI_Up:
      Serial.println("HI down");
      break;
    case CMD_LO_Down:
      digitalWrite(LED1, LOW);   // LED on
      Serial.println("LO up");
      break;
    case CMD_LO_Up:
      Serial.println("LO down");
      break;
    default:
      Serial.println("unknown command");
      break;
    }
  }
}
```

takeled, Arduino ソース 4/4

```
void setup()
{
  // Initialise serial port
  Serial.begin(SER_BAUD);
  Serial.println("start");

  pinMode(LED0, OUTPUT); //Pin13 has an LED connected on most Arduino boards:
  pinMode(LED1, OUTPUT); //Pin4 has an LED on USB board
  // Initialise the ADB subsystem.
  ADB::init();
  // Open an ADB stream to the phone's shell. Auto-reconnect
  connection = ADB::addConnection("tcp:4568", true, adbEventHandler);
  Serial.println("OK");
}

void loop()
{
  if((millis() - lastTime) > 500){ //最後にADCを読んだ時刻から500mSec経っているか?
    sensorValue = analogRead(A0);
    connection->write(2, (uint8_t*)&sensorValue); //ADCのデータを送出
    Serial.println(String(sensorValue));
    lastTime = millis(); // 最終時刻を更新
  }
  // Poll the ADB subsystem.
  ADB::poll();
}
```

```
package org.takeoka.takeled;


import java.io.IOException;


import org.microbridge.server.AbstractServerListener;
import org.microbridge.server.Server;


import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;


import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.SeekBar.OnSeekBarChangeListener;
```

```
public class MainActivity extends Activity implements OnClickListener, OnTouchListener, OnSeekBarChangeListener
{
    // コントロールコマンド
    private static final int COMMAND_HI_Down = 0x01;
    private static final int COMMAND_HI_Up =   0x02;
    private static final int COMMAND_LO_Down = 0x03;
    private static final int COMMAND_LO_Up =   0x04;

    private static final int LED_CONN_PORT = 4568;

    private Button btn_Hi;
    private Button btn_Lo;

    private TextView sensVText;
    private int adcSensorValue=0;

    private Handler sensTextUpdateHandler;

    private int aho=0;

    private Runnable adcUpdate;

    Server server;
```

```
 /** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState){
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  btn_Hi = (Button) findViewById(R.id.button_HI);
  btn_Hi.setOnTouchListener(this);
  btn_Lo = (Button) findViewById(R.id.button_LO);
  btn_Lo.setOnTouchListener(this);

  sensVText = (TextView) findViewById(R.id.sensVText);

  sensTextUpdateHandler = new Handler();
  sensTextUpdateHandler.post(adcUpdate);

  server = new Server(LED_CONN_PORT);
  try{
    server.start();
    Log.d("microbridge", "started TCP server");
  }catch (IOException e){
    Log.e("microbridge", "Unable to start TCP server", e);
    //finish();
 }
```

```
 server.addListener(new AbstractServerListener() {
   @Override
   public void onReceive(org.microbridge.server.Client client, byte[] data)
   {
     if(data.length<2) return;
     adcSensorValue = (data[0] & 0xff) | ((data[1] & 0xff) << 8);
     sensTextUpdateHandler.post(adcUpdate);
   }
  });

 adcUpdate = new Runnable() {
    public void run() {
       sensVText.setText(""+aho+","+Integer.toString(adcSensorValue));//draw volume value
       aho++;
      // sensTextUpdateHandler.postDelayed(this, 1000);
    }
  };
}    // end of onCreate()
```

```
@Override
protected void onResume(){
   super.onResume();

  // Create TCP server
  if(!server.isRunning()){
     try{
       server.start();
       Log.d("microbridge", "started TCP server");
     }catch (IOException e){
       Log.e("microbridge", "Unable to start TCP server", e);
       //finish();
     }
  }
}

@Override
protected void onPause(){
   super.onPause();
   if(server.isRunning()){
      server.stop();
        Log.d("microbridge", "stopped TCP server");
   }
}
```

```
 @Override
 public boolean onTouch(View view, MotionEvent event){
   //sensVText.setText(Integer.toString(adcSensorValue));//draw volume value

   if(view==btn_Hi){
     switch (event.getAction()){
     case MotionEvent.ACTION_DOWN:
        sendData(CMD_HI_Down);
       //sensVText.setText(Integer.toString(10));
       break;
     case MotionEvent.ACTION_UP:
       sendData(CMD_HI_Up);
       break;
     }
   }else if(view==btn_Lo){
     switch (event.getAction()){
     case MotionEvent.ACTION_DOWN:
       sendData(CMD_LO_Down);
       //sensVText.setText(Integer.toString(20));
      break;
     case MotionEvent.ACTION_UP:
       sendData(CMD_LO_Up);
       break;
     }
   }
  return false;
}
```

```
    public void onUpdate(){
       sensVText.setText(Integer.toString(adcSensorValue));//draw volume value
    }

    /**/

    /**
     * MicroBridgeにデータを送信
     * @param data
     */
    void sendData(byte [] data){
      try{
        server.send(data);
      }catch (IOException e){
        Log.e("microbridge", "Unable to send", e);
      }
    }
} // end of MainActivity
```

Android
と
本当の
日本の組込み

日本の組込み

- メジャー大企業の気持ち
  - 独自性
  - ブランドとしての「顔」が大事

- 中国製と同じでは困る
  - 単に、ハードウェアを作るだけでは、物価の安い国が勝つ

- Android電話機はともかく、TVはどうだろう
  - GoogleTVというものもできつつあるが…
  - WindowsをTVに入れるというのは、どうなのだろう…
  - AppleTVもどうだろう…
  - 日本のTVは、最後の起死回生の時期!!!

日本の組込み

- 日本はWindowsCEに裏切られた、と言っても過言ではないだろう
  - WindowsCEは、日本のセットメーカ、CPUメーカが乗っていた
    - 日本語版は、日本メーカしか出荷できなかった
  - ある日、Win HPCとなり、台湾メーカでも日本語版を出せるように…
  - CPUのサポートを自由に打ち切り
    - SH-3, MIPS が打ち切られていった（正確には、WinCEとして、打ちきったとは言っていないが）
    - WindowsNTもPowerPC, MIPSを順次、打ち切った
- ー私企業の資産に乗るリスク
  - WindowsなどMS社のソフトウェアは、資産化されていない
    - 開発費を損金処理しており、ソフトウェアは資産計上していない
  - いつ打ちきっても、会社のポートフォリオは、悪くならない
    - 逆に、IBMはOS/2 を資産化して、帳簿上では大きな価値を持っていた。それを打ちきった時には、莫大な特別損失として計上した

日本の組込み

逆に…

オープンソースの Linux, Android であれば

- 自分達なりに維持できる
- 安定版を、ずっと維持する
- 無料のソフトウェアは怖いなぁ…

↓

- 維持、サポートする専門会社が存在できる
  - 品質向上も独自にできる
  - 餅は餅屋。OSを自社で維持するのは、コストが高すぎる
  - リーズナブルな価格で、相当以上の大規模ソフトウェアを安心して使用できる
  - 日本連合として、サポート会社をみんなで応援すべき
- 低リスク

※ソースコード開示義務には、注意しよう

- ライバルにも、ソースコードを開示するのが嫌ならば

バージョンアップ地獄にハマるな

- スマートフォンは、DoCoMoが「最新バージョン」を強調するが、電話機のOSでそういう必要があるのか???
- 最新版がいいとは限らない
  - リリースされたばかりの版の方がバグや問題が多い
- 日本の組込みなら
  - 安定性を見極める
  - 自分で育てる
  - 安定して動いたものは、いじるな
- Androidは、バージョンアップすると、APIやフレームワークが変わっていることがあった
  - ケータイ電話に採用される前夜
  - アプリケーションが動かなくなる
  - ある種のデバイス駆動ミドルウェアが動かない

気になる? 他者

- 主にモバイル端末をめぐる動き
  - 真面目な組込みは、冷静に判断しよう
- Qt
  - OSではなく、GUIミドルウェア
  - Linux, Windows, Macで開発、使用できる
  - MeeGoなどから人気に
- Taizen (MeeGo)
  - 自動車業界で生きる(?)
- Symbian
  - ほぼ終了だが、まだ、様子見が必要か?
- HP WebOS (旧 Palm) (クローズドなケータイ用OS)
  - また復活とHPが宣言
- Black berry (クローズドなケータイ用OS)
  - 欧米では、非常に人気ある。政治的にも安定している。

気になる? 他者

- 結局
- Android
  - 現在、Taizen(MeeGo)は、かなり弱っている
- Linaro
  - ARMのためのLinux
  - Linuxの開発メインストリームは、組み込みに優しくない
    - エンタープライズ、デスクトップ向け
    - メモリ大食い
    - 高速CPU前提
  - ARM版Android用のLinuxがLinaroベースになるかも
    - 特に問題ない

スマートフォンの種類が多すぎ

- スマートフォンの全機種で動作するか?
  - 雨後の竹の子のごとく、スマートフォンが出てくる
  - Androidのバージョンもまちまち
    - 古いものから、新しいものまで
- 検証環境を、一社で整えるのは大変
- 都立産業技術研究所さんなどで、環境を整えてほしい
  - スマホの種類をたくさん備えてくれるだけでも、大助り

女子ハンダ付け会

- OSSコンソーシアム 組込み部会&女子部 主催
- Arduinoなどをハンダ付け
- AndroidとArduinoを接続して動作させる
  - IT系 女子に、ハードウェアを体験してもらう

女性エンジニアがはんだ付け勉強会
OSSコンソーシアム組込み部会女子部主催

オープンソースビジネス推進団体であるOSSコンソーシアム（会長＝渡辺剛喜・サイバーコム副社長）の組込み部会女子部主催による「ハンダ付け勉強会」が23日、若松通商6階のアキバNET館で行われた。女性エンジニアから「はんだ付けを体験し、ものづくりをしたい」という声が上がり実現したもので、今回が4回目の開催。参加者は組込み系、IT系など部会以外の一般参加もあり、女性5人、サポート男性6人。書籍付録の基板を利用し、Arduino互換ボードを作成した。

参加者からは「はんだ付けは経験しておいた方がいいと思った」「業界の人と交流できる場がうれしい」と様々な声が上がっていた。

小暮敦彦アキバNET館館長は「当館は、技術者の交流の場として新ビジネスモデルの創出を図ることも目的。こういう勉強会は歓迎する」と話す。

女子による女子のための勉強会
（アキバネット館で）

女子ハンダ付け会

- 2011年に 2回 アキバにて開催
  - 若松通商さんより絶大なバックアップをいただき
  - 都立産業技術研究所のセミナー&懇親会がきっかけ
- 2012年6月 大阪日本橋にて開催
- 2012年9月 アキバで第5回 開催

おまけ

Javaで遅いと感じたら…

C言語で開発

- JAVAから、C言語のサブルーチンを呼び出せる
  - JNI (Java Native code Interface)という仕組みがある
  - データの型変換
  - ガベージ・コレクションとの共存
- Cから、ハードウェアもたたける
- Cのサブルーチンで、画像処理、信号処理

- NDK
  - Android標準のC言語開発キット