Arduinoでの組込みOS自作体験

坂井弘亮
(KOZOSプロジェクト)
TwitterID:kozossakai

まず最初に連絡

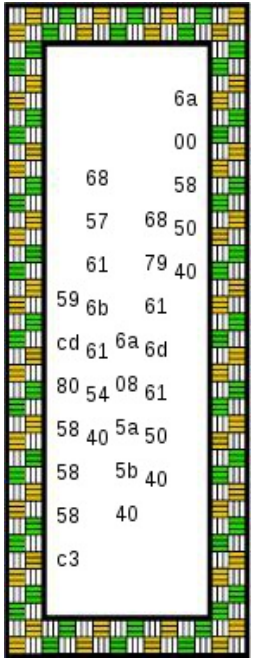
- 事務局側で用意している**USB**メモリがあります
- 「**FreeBSD-avr-kozos.ova**」という**OVA**ファイルが入っているので, これからの説明時間中に自分の**PC**にコピーしてください
  (1GB程度あるので, コピーに時間がかかります**)**
- ネットからダウンロード済みのひとは不要です
- **VM**はインストールしてありますか？ **(**そうでないひとは演習の合間にインストールしてください**)**

自己紹介

- 趣味で組込みOSを作っています **(KOZOS)**
- セキュリティキャンプの講師をやっています
- オープンソースカンファレンスなどのイベントに出展しています
- たまに勉強会とかもやっています
- 本を何冊か書いています **(12**ステップで作る組込み**OS**自作入門**)**

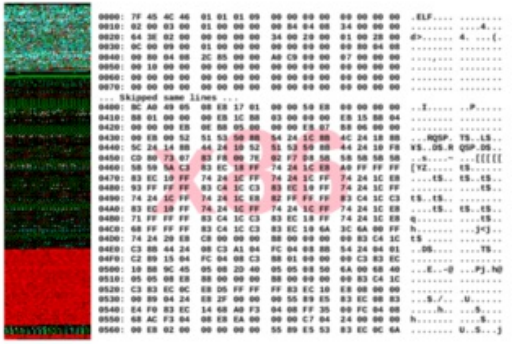
他にも最近は，こんなのを作ったりしてます

バイナリかるたとアセンブラ短歌



他にも最近は,こんなのを作ったりしてます

バイナリカレンダー2014



October 0x7DE

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			01 00001	02 00010	03 00011	04 00100
05 00101	06 00110	07 00111	08 01000	09 01001	0A 01010	0B 01011
0C 01100	0D 01101	0E 01110	0F 01111	10 10000	11 10001	12 10010
13 10011	14 10100	15 10101	16 10110	17 10111	18 11000	19 11001
1A 11010	1B 11011	1C 11100	1D 11101	1E 11110	1F 11111	

本日やること

- 「**Arduino**」というマイコンボードを対象にして
- マイコンのプログラミング体験をしてみましょう
- 「組込みOS」というものに触れてみましょう

「組込み機器」とは何か

こういうのを「組込み機器」と言います

「組込みシステム」とか言います

- 英語だと「**Embedded System**」です
- カタカナで「エンベデッド」とも書きます
- 内部で動いているソフトウェアは「組込みソフトウェア」と言います
- 内部で動いている**OS**は「組込み**OS**」と言います **(**無い場合もある**)**

身の回りの組込み機器を考えてみよう

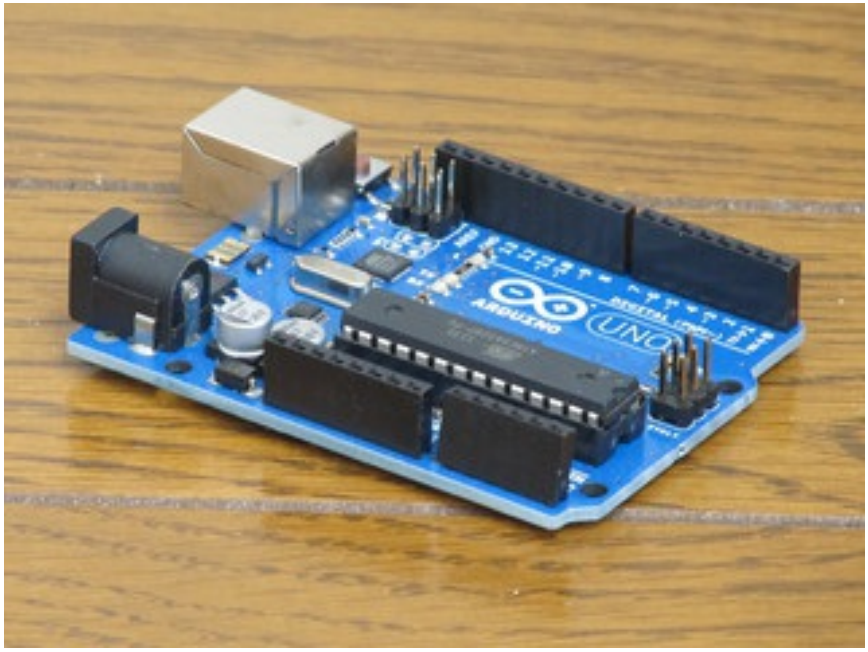
- 今日ここに来るまでに見た「組込み機器」を5つ, あげてみよう! (勘で良いです)

(ヒント)

- 「たぶんこれってマイコンが入っていて，ソフトウェア制御になっているんじゃないかなあ」というものがあれば，それは組込み機器です
- やってることは単純だとしても，たとえば動作速度をチューニングできそうなものがあれば，それはおそらくソフトウェア制御の組込み機器です
- 自分の家にあるもの，来る途中に見たもの，学校内にあるもの

Arduino

Arduinoも「組込み機器」の一種です



(おまけ) **Raspberry Pi** もそうです

本日はArduinoで動くプログラムをいじります

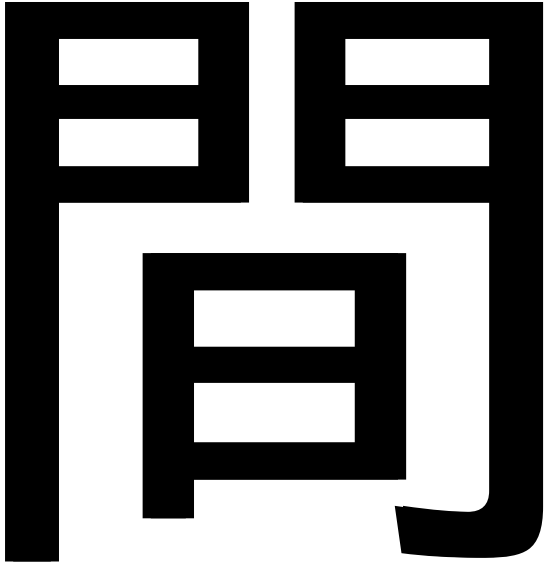
- プログラムはサンプルがあるので, それを改造します
- シミュレータ上で動作させてみます
- 実機でも動く(はず)ので, 実機で試したいひとはやってみてもOKです
- 慣れたら次は, OSを載せてみます
- OSを使うメリットは何か？ を考えてみよう
- (質問)組込み機器で, OSを使うメリットは何でしょうか？ (なぜ必要なの？)

開発環境について

- 開発には, **FreeBSD**や**Linux**系ディストリビューション**(Ubuntu**など**)**などの，いわゆる「**PC-UNIX**」が向いています
- できることならこれを機会に,「**PC-UNIX**」に慣れてほしいです **(**使っているとかっこいいです**)**
- でも, 今回の演習のためだけに**PC-UNIX**を**PC**にインストールするのはたいへんです

ということでVMを使います

- **VM**は**PC**上で動く, 仮想**PC**環境です
- **VM**環境の上に**(Ubuntu**などの**)OS**をインストールして, ひとつのウィンドウの中で ひとつの**PC**が起動しているようなイメージで使うことができます
- **VM**上に**FreeBSD**と今回の開発環境をインストールしたイメージを, すでに作成してあります
- それがさきほどコピーした「**FreeBSD-avr-kozos.ova**」です



では, VMを起動してみましょう

- **VMware Player** もしくは **VirtualBox** を起動してください
- **OVA**ファイルを「インポート」してください
- 配布資料にも説明がありますので, そっちも参考にしてください

インポートできたら, 起動してみよう

- **FreeBSD**が起動して, ログインプロンプトが出てきます
- ログインしてみよう **(**パスワード等は配布資料参照**)**

CUIに慣れよう!

- 開発用の様々なツール類は, **CUI(**キャラクタ・ユーザ・インターフェース**)**での操作がベースになっているものが多いです
- なので**CUI**での操作に慣れましょう
- **CUI**は覚えておいて損は無いです. あと**CUI**でガシガシ使えると, かっこいいです
- **CUI**の操作は, 配布資料を参考にしてください
- まずはファイルを作って編集したりしてみよう

マイコンで**Hello World**を動かしてみよう

- 「**Hello World**」と出力するだけのプログラムが, すでに置いてあります
- 「**avr_03-hello/os**」というディレクトリに入ってください
- **main.c**などのファイルがあるので, ソースコードを見てみよう
- 「**make**」を実行してみよう →「**kozos**」という実行モジュールが作成されます

プログラムを動かしてみよう

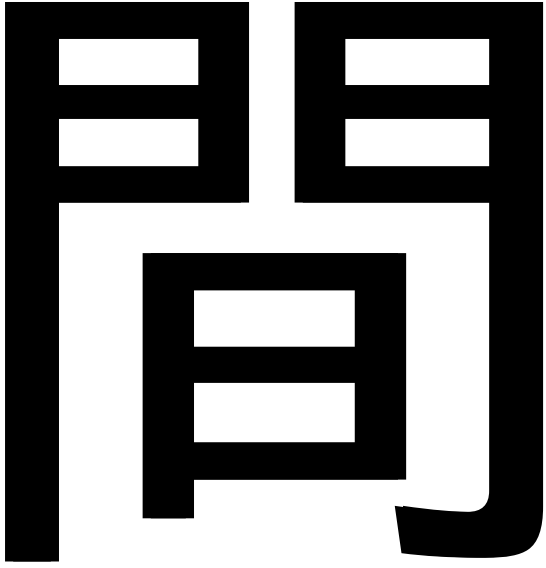
- 実行モジュールを，シミュレータで動かしてみよう
- やりかたは配布資料を参照してください
- ソースコードをいろいろいじってみよう
  色を変えてみよう
  テキストアニメーションをやってみよう

プログラムを読んでみよう

- **grep** を使って**puts()**を追ってみよう
- **gets()** を追ってみよう
- デバイスドライバを読んでみよう
- 割込み処理を読んでみよう
- **main()**に来るまでを読んでみよう
- リンカスクリプトを読んでみよう
- 読んでみたいところがあれば, そこを読んでみよう

このプログラムの問題点

- 例えば, LED点滅を考えてみよう (シミュレータだとLEDが無いので, 定期的に文字出力することで代用します)
- 2つのこと(点滅とコマンド応答)を同時にやるとしたらどうなるか？
- ダンプコマンドを考えてみよう ダンプ中に点滅はどうなるか？

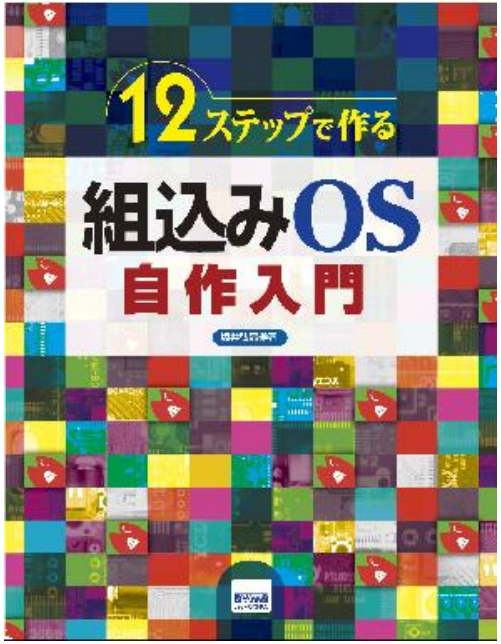


これを防ぐには

- ループ処理などがあったら，その都度そこに**LED**点滅の処理を入れる
- ダンプ処理などの時間がかかる処理の最中に，定期的にメインループに戻る
- どれもめんどくさい！

OSを使うことでこれらの問題をクリアできます

今回使っているOSは，これです



では，**OS**を使った場合を見てみましょう

- 「**avr_03-sim**」を見てみましょう
- シミュレータで動かしてみましょう
- **LED**の点滅処理を入れてみましょう

OSのソースコードを見てみよう

- **main()**を見てみる
- 各タスクの**main()**を見てみる
- **kz_XXXX()**の先を追ってみる
- タスクスケジューリングを見てみる
- タスクディスパッチを見てみる

OSの必要性

- ベースシステムにOSを使うことで，これらの2つの処理を「タスク」にして，並列に動作させることができます
- 実際にはCPUはひとつなので，適当なタイミングで処理を切替えて複数のタスクを一見並列に動いているように見せかけてくれます

OSの必要性

- なぜ, OSが必要なのか考えてみよう!
- PCでは, なぜWindowsなどのOSが必要なのだろうか?
- 組込み機器でもOSが必要な理由は何か?
- (質問)ファイルシステムは, OSの必須の機能か?

OSの必要性

- PCと組込み機器では，OSの必要性が異なります
- PCでは，アプリケーションが主役．アプリを動かすための汎用的なベースシステムが欲しい
- 組込み機器では，デバイスが主役．コスト削減のため，1つのマイコンで複数の制御を行いたい
- このためOSへの要望も異なってきますし，OSの思想も異なってきます (あと歴史も違う)
- (もう1度質問)ファイルシステムは，OSの必須の機能か？

OSの歴史の違い

- 汎用OS
  メインフレームという巨大な共用コンピュータ (研究機関)
  (→ミニコン →ワークステーションやサーバのように発展)
  → 大学や研究所で, みんなで共用利用したい
  → マルチタスク, マルチユーザが発展
- PC用OS(汎用OSの一種？)
  家電メーカーがマイコンという超安いCPUっぽいものを開発
  → これで家庭用コンピュータ作れるんじゃね？ (家庭向けPC)
  → GUIほしいよね → マルチタスクOSほしいよね(汎用OS化)
- 組込み機器
  電子機器を, マイコンでソフトウェア制御したい (家電業界)
  → 複数のデバイスを1個のマイコンで制御したい(コスト削減)
  → モニタをマルチタスク化 → これってOSだよね

じゃあOSって何なのか?

- (PCユーザの視点)アプリを使うための基盤 (システムアプリを含む)
- (ソフトウェア開発者の視点)アプリを作るための共通基盤 (システムコール, デバドラ)
- (OS開発者の視点)コンピュータの基本要素 (CPU, メモリ, I/O)を管理するもの
- (ソフトウェア階層からの視点)ソフトウェアを階層下したとき, 最下層にあるもの

共通して言えるのは

- ユーザ視点では
  自分が「ここから下層はもう知らなくても良いや」という下位層を「**OS**」と言っている **(Web**アプリ開発者にとっては, ブラウザが**OS.** でも**OS** 開発者からすれば, ブラウザは単なる1アプリ**)**
- そういう意味で, 自分の知らないところでいろいろ勝手にやってくれるものを「**OS**」と言っている
  → **OS**は「妖怪」と言える

共通して言えるのは

- **OS**開発者は
  自分が「自分ですべての世界を作ってみたい！」という下位層を，「**OS**」と言っている
- **OS**開発者としては
  (OSに限らず**)**ぜひ「使う側」から「作る側」に来てほしいです
  いろいろ新たな世界が開けます