2009-10-13（火）
|
2010-05-31（月）

山形県立産業技術短期大学校 千秋広幸

dude-wrapng 説明書

--- Arduino IDE を修正し、hidspx 対応を追加する ---

[1] はじめに

Arduino IDE は、Windows, Linux, Mac OS 上で動作するシンプルな AVR マイコン開発環境です。多岐にわたるライブラリが用意されているため、基本的な学習から高度な応用まで適用でき、無料で利用できます。こうした優れた特長を持つ Arduino IDE から hidspx（AVR ライタ）で書込みができれば、市販ボードの利用以外に工場出荷の AVR マイコンチップを使った開発が可能になり、汎用 AVR マイコン開発環境を実現できます。

Arduino IDE では、生成した HEX ファイルを avrdude にてアップロードを行います。IDE では avrdude 以外のツールはサポートしないため、avrdude がサポートしないプログラマは利用できません。そこで私は、Arduino IDE のコードを修正し、avrdude 以外にも hidspx を利用できる仕組みを追加実装しました。このアーカイブに含まれる pde.jar というファイルです。書込みに利用する protocol に hidspx を指定すると hidspx コマンドを起動し、それ以外は従来通り avrdude を起動します。こうした改良の他にも、いくつかの改善を実施しています。そのため hidspx を利用しない方でも、dude-wrapng を導入で不具合改善と利便性向上が期待できます。

[2] 導入手順

(0) dude-wrap の利用者の方へ（dude-wrap ≠ dude-wrapng）

> dude-wrapng は、Arduino IDE 自体が hidspx をサポートし、dude-wrap のように avrdude の変更は不要です。**dude-wrap を利用していた方は、 avrdude.exe と呼ぶラッパーを削除し、avrdude_orig.exe を avrdude.exe に戻します。**
>
> dude-wrap 利用者が事前に必要な作業は、上記の前処理と dude-wrapng が提供する Arduino フォルダを指定の場所にコピーするだけなのですが、**boards.txt, programmers.txt** も大幅に書き換えを行っており、ATmega88 や mega644p のサポートを追加しています。
>
> 独自に項目を追加している場合は、利用中の Arduino フォルダを保存（設定ファイルを保護のため）し、上書きコピーを行ってください。

(1) Arduino(0018)を展開

Arduino IDE は、arduino-00**フォルダごと任意のフォルダにコピーするだけで利用可能な AVR 開発環境です。開発に必要なも基本ツール一式を含み、展開するだけで利用できます。コンパイルできるコードサイズや試用期間の制限もありません。

まずは Arduino-0018 を入手してください。展開するだけで利用できますが、展開後は 230MB ほどの巨大なサイズになります。USB メモリなどに展開する場合には空き容量に注意し、Arduino-00** を空白を含むフォルダに置くのは避けてください。

AVR マイコンの開発環境の選択に悩まれている方は、Arduino ボードに限らず、Arduino IDE の利用を検討してください。

(2) 配布アーカイブを展開し、任意の場所に置きます。

```
.¥
├──Arduino  ← (*1)
│    ├──Blink2313
│    └──hardware
│         └──MyArduino
│              ├──bootloaders
│              │    ├──atmega
│              │    ├──atmega644p
│              │    ├──atmega88
│              │    └──attiny2313
│              └──cores
│                   ├──arduino
│                   ├──arduino644p
│                   └──attiny2313
├──bin ← (*2)
│    ├──linux
│    └──windows
├──doc
├──src
│    └──arduino
└──wrap
```

(2) ファイルをコピーする
(2-1) (*1)を「C:¥Documents and Settings¥ユーザ名¥My Documents」にコピー。
(2-2) (*2)環境にあわせ、Windows環境ではwindowsフォルダ以下にあるjarファイルを
jarファイルを、Arduinoフォルダ(...¥arduino-0018¥lib)にコピー。

この作業を完了後にArduino.exeを起動すると、「ツール」「ブートローダを書き込む」に hidspxが現れます。これらの作業が完了すれば、dude-wrapngを展開したフォルダは削除しても構いませんがReadme.pdfは大切に保存し内容の理解に努めて下さい。

(3) hidspx.exeの設置
dude-wrapngを導入したArduino IDEは、必要に応じてhidspxを呼び出して書込みを行います。そのためdude-wrapngの利用にはhidspx.exeが必要です。hidspx.exe, hidspx.ini ファイルを...¥arduino-0018¥hardware¥tools¥avr¥bin￥hidspx.exeが存在すればそれを利用します。
※ dude-wrapngでは、c:¥bin をサーチしませんので、↑にコピーしておくと便利です。
Linux環境では、シンボリックリンクを作成すれば無駄がありません。

これ以外の場所に置いたhidspxを実行したい場合は、boards.txt内に以下のような設定を行う必要があります。

#attiny2313.bootloader.hidspx=c:¥bin¥hidspx.exe→ 書き込み用ツールのフルパス名

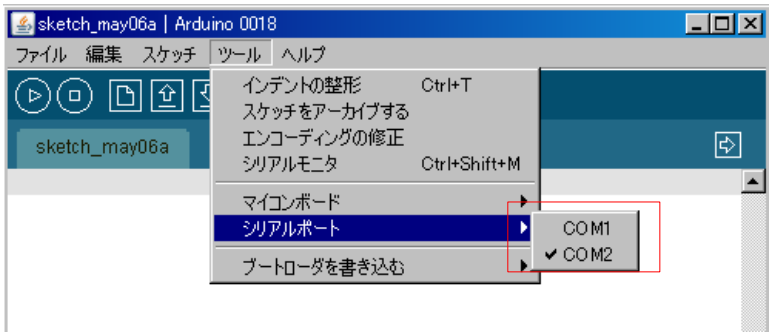
私の配布しているツールでは、c:¥binに hidspx が存在することを前提にしている場合が多いので、ツール群をこの場所にコピーしておくと便利です。一度、hidspx に含まれる Setup.bat を実行してください。

■ UpLoadボタンの注意点
Arduino IDEは、標準ではCOMポートを利用してUpLoad（AVRマイコンへの書込み）を行こなうため、事前にCOMポートの指定が必要です。これはdude-wrapには限らず、Arduino共通の基本的な注意点です。Arduino IDEはAuto resetのためにCOMポートを直接操作することから、この指定がないと内部

エラーが発生します。しかし、dude-wrapng ではこの指定は必須ではなく、この制限を回避する設定を行うことにしました。なお、実際に利用できるポートを指定するとデバックに便利に使えます。

その他にも、私の確認した不具合を修正を行ない、その結果生成された 2 つの jar ファイルを公開します。これらの jar ファイルは、Arduino IDE の日本語版ソースを元に修正し、Linux 環境で作成したものです。Linux 用と Windows 用は同名のファイルですが内容が異なりますので、適合するものをご利用ください。

dude-wrapng を利用すれば、COM ポートの指定は不要で、UpLoad ボタン操作のみで HEX ファイルを書き込みができます。またメニューも日本語で表示でき、オリジナルの Arduino ではエラーになる環境でも利用可能です。



■ dude-wrapng 利用時の注意点

dude-wrapng(hidspx による書き込み)を採用すれば、

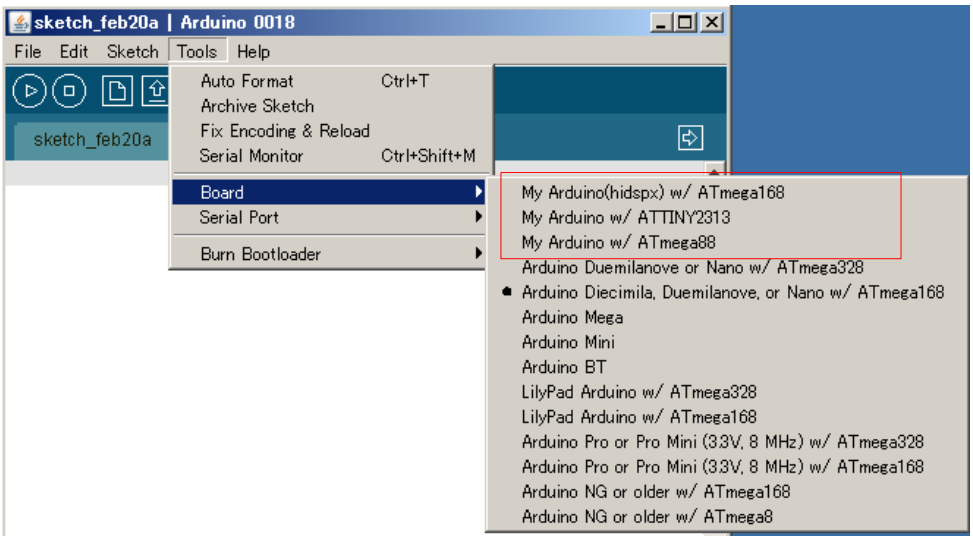
- 高速書き込み（Arduino のアップロードよりも速い）
- RESET スイッチ操作不要
- 書込み後の待ち時間がない
- ベリファイも行われる
- Bootloader の書込みも可能

などの利点があります。しかし、dude-wrapng(hidspx.exe)で書込みを行ない場合、BootLoader 領域は消去されます。dude-wrapng 以外の環境で Arduino IDE を利用する場合には、BootLoader 書込みを行ってください。

[4] 設定後の動作確認方法

■ Arduino IDE でのメニューを確認

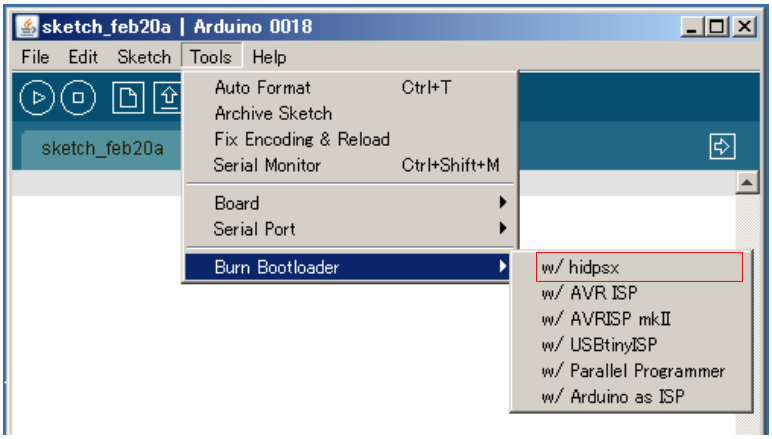
（赤枠のメニューが追加されます。赤枠内は、boards.txt の修正で変化します。）



利用に先立ち、自分が利用するボードをこの中から選択します。dude-wrap を利用する場合には、My～となっているボードを選択します（それ以外は、ブートローダを利用するボードです）。

○ BootLoader の書き込み

BootLoader の利用に関わらず、このメニューの選択で、AVR マイコンの FUSE 設定やブートローダの書込みを行うことができます。Arduino では工場出荷時とは異なる設定で利用することから、dude-wrap を利用する場合でも一度はこのメニューで FUSE 設定や BootLoader の書込みを行ってください。




```
#=> hidspx -d4 -ph --new-mode -lOx3F -fXOx00 -fHOxdd -fLOxff
Detected device is ATmega168.
Fuse Low byte was programmed (0xFF).
Fuse High byte was programmed (0xDD).
Fuse Extend byte was programmed (0x00).
Lock bits programm error. (3F -> FF)
#=> hidspx -d4 -ph --new-mode C:¥Programs¥arduino-0018¥hardware.....¥ATmegaBOOT_168_ng.hex -lOxCF
Detected device is ATmega168.
Erase Flash memory.
Flash memory...
Writing   [##################################################] 16038, 1.00s
Verifying [##################################################] 16038, 9.22s
Passed.
Lock bits are programmed (0xCF).
Total read/write size = 32076 B /10.47 s (2.99 kB/s)
```

※ #=> の行は、hidspx に対するコマンドを意味しています。Arduino の内部で行われている指示を確認できます。

以上の設定を行えば、通常の Arduino と同様に、スケッチをコンパイルし、書き込むことが可能になります。

○ Upload ボタンでの書き込み（Ctrl+U でも書込み可能）

```
#=> hidspx -d4 -ph --new-mode C:¥Programs¥arduino-0018....ial¥applet¥AnalogInOutSerial.cpp.hex
Detected device is ATmega168.
Erase Flash memory.
Flash memory...
Writing   [##################################################] 3152, 1.81s
Verifying [##################################################] 3152, 1.86s
Passed.
Total read/write size = 6304 B / 3.89 s (1.58 kB/s)
```

コンパイルが必要な場合には、コンパイル後に AVR マイコンに書込みが行われます。なお、繰り返し

ますが、dude-wrap(hidspx)で Upload を行うとブートローダは消去されます。通常のブートローダの利用を行うには、メニューを使って Bootloader を書き込んでください。

[5] 利用例

ATtiny2313 を Arduino IDE から利用する方法を紹介します。

■ 回路図(WSN216 の場合の接続例、ISP コネクタの配線時に参考のこと)

```
                  ATtiny2313
                   ___   ___
RESET             [1  |__| 20] Vcc
PD0(RxD)          [2       19] PB7(SCK)
PD1(TxD)          [3       18] PB6(MISO)
XTAL2             [4       17] PB5(MOSI)
XTAL1             [5       16] PB4
PD2(BOOT JUMPER)  [6       15] PB3(BUSY  LED)
PD3(D+)           [7       14] PB2(READY LED)
PD4(D-)           [8       13] PB1
PD5(USB pullup)   [9       12] PB0
GND               [10      11] PD6
                   ~~~~~~~~~~~
```

※ Vcc-GND 間に 0.1uF のパスコンを実装し、RESET ピンを 10kΩ 程度でプルアップします。
RC 発振で動作させるので、XTAL1,2(PA0,1)は I/O として利用できます。

このアーカイブでは、以下の URL で公開されているファイル一式を同梱しました。Arduino-0018 の仕様に合わせるため、一部のファイル名を変更(main.cxx → main.cpp)しています。I/O ピンの割り当てや FUSE 設定は以下のとおりです。

ATtiny2313 用ライブラリ (attiny2313_core.zip)
http://hci.rwth-aachen.de/tiki-download_wiki_attachment.php?attId=769&page=luminet

■ Fuse 設定 (Bootloader 書き込みで、この設定を反映できます)

```
Low: 11100100 (0xe4)
     ||||++++-- CKSEL[3:0] システムクロック選択
     ||++-- SUT[1:0] 起動時間
     |+-- CKOUT (0:PD2 にシステムクロックを出力)
     +-- CKDIV8 クロック分周初期値 (1:1/1, 0:1/8)

High:10-11101 (0x9d)
     |||||||+-- RSTDISBL (RESET ピン 1:有効, 0:無効(PA2))
     ||||+++-- BODLEVEL[2:0] (111:Off, 110:1.8, 101:2.7, 100:4.3)
     |||+-- WDTON (WDT 0:常時 ON, 1:通常)
     ||+-- SPIEN (1:ISP 禁止, 0:ISP 許可) ※Parallel 時のみ
     |+-- EESAVE (消去で EEPROM を 1:消去, 0:保持)
     +-- DWEN (On-Chip デバッグ 1:無効, 0:有効)

Ext: -------1 (0xff)
            +-- SPMEN (SPM 命令 1:無効, 0:有効)
```

■ ピン割り当て（「D 0」はデジタルピン0を意味します）

```
// ATMEL ATTINY2313(このマイコンにはA/D入力ピンはありません)
//                  +-¥/-+
// RESET      PA2  1|    |20  VCC
// RX   (D 0) PD0  2|    |19  PB7 (D 16)(SCK)
// TX   (D 1) PD1  3|    |18  PB6 (D 15)(MISO)
//      (D 2) PA1  4|    |17  PB5 (D 14)(MOSI)
//      (D 3) PA0  5|    |16  PB4 (D 13)
// INT0 (D 4) PD2  6|    |15  PB3 (D 12)(BUSY  LED)
// INT1 (D 5) PD3  7|    |14  PB2 (D 11)(READY LED)
//      (D 6) PD4  8|    |13  PB1 (D 10)
//     *(D 7) PD5  9|    |12  PB0 (D 9)
//            GND 10|    |11  PD6 (D 8)
//                  +----+
```

■ サンプルプログラム

```
// LED (PB1)の点滅
int ledOPin = 10;                 // LED ATtiny2313 PB1

void setup()                     // run once, when the sketch starts
{
  pinMode(ledOPin, OUTPUT);      // sets the digital pin as output
}

void loop()                      // run over and over again
{
  digitalWrite(ledOPin, HIGH);   // sets the LED on
  delay(500);                    // waits for a second
  digitalWrite(ledOPin, LOW);    // sets the LED off
  delay(500);                    // waits for a second
}
```

[6] ライセンス

dude-wrapは、Kimio Kosaka 氏の「Arduino IDE から FTDI Bitbang method を実行する」をヒントにしました。大幅に書き換えを行っていますが、GPLでの公開となります。

http://www.geocities.jp/arduino_diecimila/bootloader/bitbang_w_ide.html

[7] 注意点

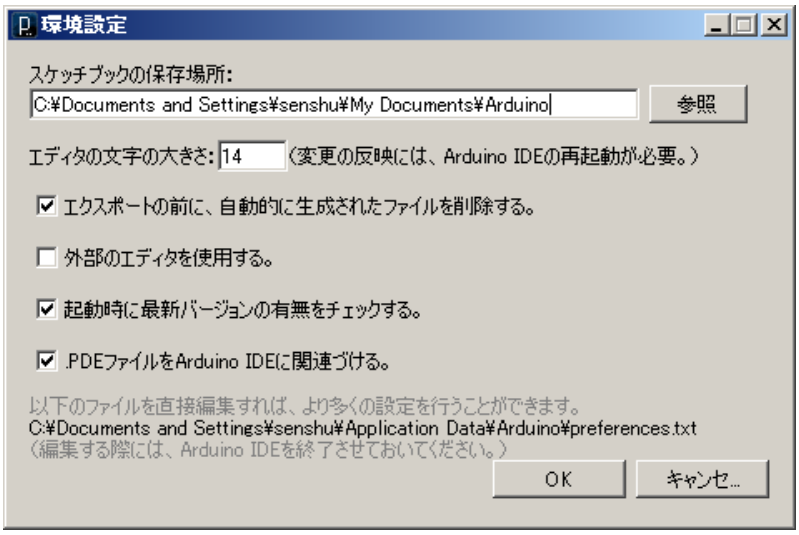
> COMポートが存在しないPCでは、プログラムのUpload時にエラーになることがあります。これは、COMポートの制御ピンを操作し、ターゲットマイコンをリセットするために発生します。「ダミーのCOMポートを割り当てるツール」を利用すればエラー回避が可能ですが、本来のデバッグには役立ちません。多少のコストが必要ですが、USB-シリアル変換モジュールを導入をお勧めします。これにより、シリアルポートを利用したデバッグなども可能になります。

この問題を改善する為に初期の段階ではcore.jar, pde.jarを修正しましたが、boards.txtへの設定項目の追加でこの問題を回避できることがわかり、hidspx利用時はその設定を適用しています。このアーカイブに含まれるboards.txtを参照ください。

[8] 利用上のヒント

Arduino の使い方は、Help メニューで各項目を選択すれば Web ブラウザで解説ページを読むことができます。付属するものは英文ですが、google toolbar をインストールしておけば、日本語に翻訳して読むことが出来ます。また、多くの関連の解説が Web 上で公開されていますので参考にしてください。5/25 以降の版から、武蔵野電波さん公開の日本語リファレンスへのリンクも追加しています。

(1) エディタのフォントサイズを変更する
エディタ窓のフォントサイズは、FILE => 環境設定　で変更できます。



(2) コンソール窓のフォントサイズや色の変更
コンソール窓の表示スタイルなどのデザインは「arduino-0018¥lib¥theme¥theme.txt」の修正で変更可能です。Arduino IDE では起動時に設定ファイルを読み込み、その内容を反映します。そのため、変更内容を反映させるには、Arduino IDE の再起動が必要です。以下の修正で、console 窓のフォントを 14 ポイント、フォント色を黄色に変更できます。

```
# GUI - CONSOLE
#console.font = Monospaced,plain,11
console.font = Monospaced,plain,14
console.font.macosx = Monaco,plain,10
console.color = #000000
console.output.color = #cccccc
#console.error.color = #ff3000
console.error.color = #cc9900
```

↑の図は変更後のものです。文字が大きくなり、色も赤から黄色に変わっているのがわかります。#cc9900 は#rrggbb を意味し、それぞれ 256 段階(00～FF)で R,G,B の明るさをしています。R と G で穏やかな黄色を指定しましたが、好みに応じて値を変えてください。

(3) programmers.txt の内容

Arduino IDE では、内部で avrdude コマンドを呼び出し、AVR マイコンに書込みを行います。複数のプログラマに対応していますが、すべて avrdude がサポートするプログラマ（書込み器）が対象です。そのため、通常では avrdude がサポートしないプログラマは利用できません。この制約を回避するため、IDE 内部のコードを修正したのが dude-wrapng です。dude-wrapng が提供する jar ファイルに置き換えることで、programmers.txt に記述した「protocol=hidspx」で、hidspx の利用を指示できます。

```
hidspx.name=hidpsx
hidspx.protocol=hidspx
```

(4) boards.txt の内容

Arduino IDE では boards.txt ファイルに、

- 先頭の文字が'#'の行はコメント行
- ボード名称
- ブートローダ用の HEX ファイル名と所在
- FUSE の値
- 書込み方法
- 動作クロック
- 基本ライブラリのフォルダ名

などを記述します。同一のチップでも動作条件に違いがあるときには、左辺の定義名をユニークな名前に変え、全ての項目を組にして追加登録してください。Arduino IDE では、この記述に重複があると混乱しますので、ボードごとにユニークなものにします。発振方式・動作周波数・チップ名などは、採用する回路により FUSE 設定は異なりますので、熟考の上、記述してください。

ブートローダや生成した HEX ファイルの書込みの際に、この情報を参考に書込みを行いますので、このファイルは非常に重要なファイルです。誤った情報を書き込みは許されません。自作ボード用に登録を行うの場合には、十分の検討の上、適切な内容を設定してください。登録データをメニューで選択できる仕組みはありませんので、誤設定を行う可能性は極めて低いと考えます。

boards.txt を変更する場合には、MyArduino 以下の boards.txt を変更してください。AVR マイコンを使った経験と登録済みの内容を参考にすれば、自分が必要とする設定は可能だと思います。

```
##############################################################
# ☆は、dude-wrapng による拡張
attiny2313.name=My Arduino(hidspx only) / ATTINY2313((8MHz/RC) → ボード名称
attiny2313.upload.protocol=hidspx                          → ☆アップロード方法（通常は avrdude -cXXX で指定）
attiny2313.upload.delay=-d1                                → ☆オプション引数
attiny2313.upload.maximum_size=2048                        → Flash サイズ（ブートローダ領域を除く）
attiny2313.upload.speed=19200                              → ブートローダ利用時の通信速度
attiny2313.upload.disable_flushing=true                    → COM ポートの DTR によるリセットを無効化
#attiny2313.upload.using=hidspx                            → avrdude を使う場合のプログラマ名
#attiny2313.upload.hidspx=c:¥bin¥hidspx.exe                → ☆hidspx.exe のフルパス名
#attiny2313.bootloader.delay=-d10                          → ☆書き込み用ツールへのオプション引数
#attiny2313.upload.bootloadHID=c:¥bin¥bootloadHID.exe      → ☆bootloadHID.exe のフルパス名
attiny2313.bootloader.low_fuses=0xe4                       → FUSE Low
attiny2313.bootloader.high_fuses=0x9d                      → FUSE High
attiny2313.bootloader.extended_fuses=0xff                  → FUSE Extend
attiny2313.bootloader.path=attiny2313                      → ブートローダを置いたフォルダ名（省略化）
attiny2313.bootloader.unlock_bits=0xFF                     → ロック解除
attiny2313.bootloader.lock_bits=0xCF                       → ロックによる保護
attiny2313.bootloader.programmer=hidspx                    → ☆Bootloader 書込みツールの指定（省略時は hidspx）
attiny2313.build.mcu=attiny2313                            → MCU 名称（avr-gcc が認識する MCU 名）
attiny2313.build.f_cpu=8000000L                            → 動作クロック（単位 Hz）
attiny2313.build.core=attiny2313                           → ライブラリを置いたフォルダ名
##############################################################
```

■ boards.txt 設定時の留意点

(1)「.name=My Arduino / ATTINY2313」の右辺は、自分にとってわかり易い文字列とし、記号や特殊文字は使わないでください。

(2) 「upload.protocol=hidspx」は、通常は「avrdude -cXXX」で指定できるものを記述します。dude-wrapng では、「hidspx」を指定でき、省略時は、Arduino フォルダの hardware\tools\avr\bin\hidspx.exe を指定したものと見なします。「upload.hidspx=c:¥bin¥hidspx.exe」のように指定すれば、ボードごとにアップロードに用いるツールを指定できます。

(2') 「upload.delay=-d1」は、hidspx コマンドが HEX ファイルの書込みを行う時の書込み速度を指定できます。不適切な値を設定すると書き込みエラーになります。利用を開始する場合には、一度は「ブートローダを書き込む」を選択し、FUSE 情報を設定してください。なお、「-d1=8M, -d2=4M, -d3=2M, -d4=1M, -d5=500kHz」に対応します。

(3) dude-wrapng では、「upload.protocol=bootloadHID」を指定可能で、Arduino フォルダの hardware\tools\avr\bin\bootloadHID.exe を指定したものと見なします。「upload.bootloadHID=c:¥bin¥bootloadHID.exe」のように指定すれば、ボードごとにアップロードに用いるコマンドを指定できます。なお、delay は無視されます。

(4) ブートローダは不要なら、「upload.maximum_size」を Flash メモリの最大値に設定できます。つまり、全領域をスケッチに利用できます。（FUSE 設定の変更が必要）

(5) hidspx 使用時は「.upload.speed=19200」の設定は省略可能です。

(6)「bootloader.***= ～」は、ブートローダを書き込み時に参照されます。

(7) hidspx で書き込む場合にはブートローダは不要ですが、ブートローダを利用する場合には適切なファイル名を指定します。dude-wrap では、ブートローダが不要な場合も BOOT_void.hex（空のファイル）を割り当てましたが wrapng では省略が可能になりました。

(8) FUSE 書き込みは、ブートローダの書き込み時にのみ行います。Upload では FUSE を操作することはありません。ブートローダを使わない場合も、適切な動作速度で利用するため、ブートローダの書込み操作で FUSE 設定を行ってください。

(9) 「bootloader.unlock_bits」、「bootloader.lock_bits」も hidspx が初期化を行う関係で、Tiny2313 では意識する必要はありません。

(10) 「build.f_cpu=8000000L」は実行時間の特定に使われます。FUSE 設定に応じて、的確に設定してください。

(11) upload.using には、bootloader, hidspx, bootmon の他、programmers.txt に記述したキーワード(avrisp, avrispmkii, usbtinyisp, parallel, arduinoisp)を指定します。

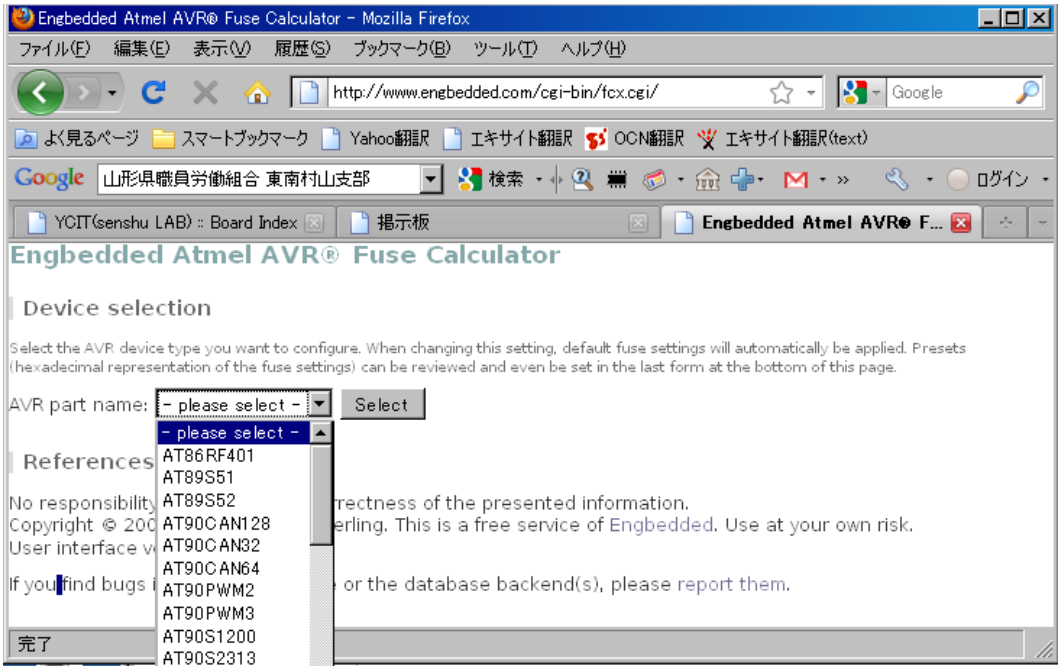
(5) FUSE 設定値の求め方

Engbedded Atmel AVR® Fuse Calculator（↓）にアクセスすれば、容易に適切な値を求めることができます。

http://www.engbedded.com/cgi-bin/fcx.cgi/

ただし、各種の用語に対する理解が必要です。AVR マイコンのデータシートや解説書を参照し、誤りの

無い値を設定してください。なお、hidspx は、FUSE 設定ミスで動作不能になるような設定には警告を発し、特別な操作無しには書込みを行わない仕様になっています。



(6) ターゲットの AVR マイコンの選定について

Arduino に限りませんが、複雑なコード（高機能なライブラリの利用を含む）の実行には多くのメモリが必要です。dude-wrap により、ATtiny2313 も Arduino IDE から利用できますが、拡張性が必要なら、ATmega168 以上のマイコン（できれば mega328p）の採用をお勧めします。

(7) ブートローダの扱い

dude-wrap を利用すると、ブートローダの領域もプログラムエリヤとして利用できるという大きな利点があります。ただし、こうした使い方は専用の AVR ライタ利用時に限られます。AVR ライタが利用できない場合にはブートローダを書き込んでおくことをお薦めします。

[9] お願い

Windows 環境の Arduino IDE の全ての版で機能すると思われますが、動作確認は Arduino-0018 で行っています。まだまだ実績が少ないので、動作テストや問題点の洗い出しにご協力ください。

[10] 同梱のファイルの入手元

このアーカイブでは、導入を容易にする為、GPL や LGPL で公開されているファイルの一部を修正して同梱しています。

・ATmega644p 用サポートファイルの入手先 → http://sanguino.cc/softwareforwindows
・ATmega88 用サポートファイル → Arduino-0018 のソースを修正して独自に作成
・シリアルポートに関する処理速度の低下を改善する方法
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1237179908
↓
http://servicios.ried.cl/arduino/temp/rxtxSerial-2.2_fixed_2009-03-17.rar --- このファイルを同梱

[11] 変更履歴

■ 2009-0925 初版公開

■ 2009-0927
(1) エラーチェックの強化（CMD バッファオーバーフロー対策）しました。
(2) hidspx へのオプション文字列の追加を可能にしました。

■ 2009-1011
「dude-wrap の更新方法」
avrdude_orig.exe に渡す引数は、操作しない仕様に修正しました。
avrdude.exe のみを差し替えてください。

(1) 説明文を見直し、プログラム関係は、src/memo.txt に移行しました。（OGURAM さんの意見）
(2) Arduino のツール群を Program Files 以下に置いた場合に発生する不具合に対処しました。ただし、GetShortPathName 関数が機能する環境に限ります。通常はこの関数は機能しますが、短い名前の生成機能を無効にした環境ではエラーになる場合があります。

■ 2009-1013
(1) 説明書の見直し（borads.txt, programers.txt の修正内容を変更）
(2) ATtiny2313 向けの修正を反映
(3) ブートローダ用のファイル指定がない場合やデバッグ用のメッセージを抑止しました。hidspx.exe を avrdude.exe のある場所に hidspx.exe が存在すれば、それを実行します。どの実行ファイルを利用するのかは、avrdude --version で確認できます。

```
C:¥Program Files¥arduino-0017¥hardware¥tools¥avr¥bin>avrdude --version
avrdude wrapper version 0.7, (Oct 13 2009)
avrdude -> C:¥Program Files¥arduino-0017¥hardware¥tools¥avr¥bin¥avrdude_orig.exe
hidspx  -> C:¥Program Files¥arduino-0017¥hardware¥tools¥avr¥bin¥hidspx.exe
```

■ 2010-0217
(1) Readme.txt に ATtiny2313 を Arduino IDE から利用する方法を追加しました。
(2) ATtiny2313 用の core ライブラリを同梱しました。

■ 2010-0223
(1) 混乱を避けるため、Arduino-0018(現行版)限定とし、以前の版に関する説明を削除しました。
(2) Arduino-0018 の日本語版は以下の URL から入手できます。スイッチサイエンスさんの関係者の尽力に敬意を表します。

http://www.switch-science.com/trac/wiki/Arduino-ja-jp

arduino-0018\lib\pde.jar, arduino-0018\lib\core.jar
英語版と日本語版は、↑の 2 つのファイルが異なるだけです。

■ 2010-0506（この版から、jar ファイルの変更版を添付）
(1) Arduino-0018 の日本語版をベースに、dude-wrap の動作の妨げになる auto reset 時に発生するエラー検出を無効にした jar ファイルを追加しました（とりあえず Window 専用です）。

arduino-0018¥lib¥pde.jar
arduino-0018¥lib¥core.jar

オリジナルの arduino-0018 の同名のファイルを、↑の 2 つのファイルで置き換えます。日本語メニューと dude-wrap 対応が実現できます。

(2) dude-wrap を Windows, Linux, Mac OS に対応させました。

(3) ATmega88 などの BootLoader を書き込まないマイコン用に、BOOT_void.hex という空の HEX ファイルを追加しました。

■ 2010-0507

- kuman さんの提案を参考に、readme.odt の内容を見直しました。それ以外の変更はありません。

■ 2010-0517

- Windows/linux 用の jar ファイルを追加
  - preferences.txt はソート後に保存する
  - 日本語メニューの追加
    - スイッチサイエンスさんの成果を活用
  - アップロード前の COM ポート操作エラー回避
    - dude-wrap の利用に効果的
  - テンポラリフォルダに作業用ファイルが残る不具合を修正
    - Arduino IDE の起動と終了を繰り返すとテンポラリフィルダに大量のフォルダが残る不具合を改善
    - Windows / Linux の両環境用の jar ファイルを提供
  - Linux 環境で、FAQ.html が開かない不具合を修正
- readme.odt に「利用上のヒント」を追加
- src/arduino に修正差分ファイルを追加
- src/preferences.txt を追加（設定可能項目の確認用）
- Linux 用に librxtxSerial-2.1-7.so を追加
  - USB シリアルの認識機能を追加
  - 標準で提供される librxtxSerial.so に代えて利用すること

※ 私は Mac OS X の実行環境を持っていないため Mac 用のテストが出来ません。協力していただける方がいらっしゃれば協力をお願いいたします。

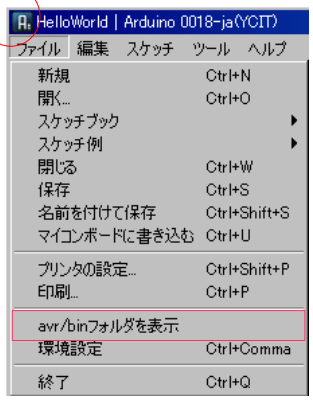
■ 2010-0519

- Windows/linux 用の jar ファイルを追加
  - アップロード前の COM ポートの操作エラー回避策修正（オリジナルに戻した）
    - java のソースではなく、boards.txt 側で対処
      - ***.upload.disable_flushing=true という設定を記述しました

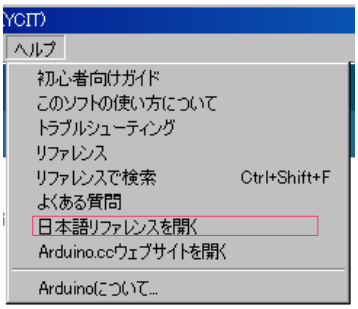
■ 2010-0528（dude-wrapng に名称変更）

- jar ファイルを変更し、以下の機能を実現した
  - hidspx の直接サポート
  - boards.txt のミスタイプ時のエラーチェック強化
  - タイトルバーへの Arduino アイコンの追加（下図 *1）
  - avr/bin フォルダへの参照機能を追加（下図 *2）
  - 日本語リファレンスへの参照機能を追加（下図 *3）
  - build フォルダへの参照機能を追加（ショートカットキー Ctrl+K）（下図 *4）
- Windows 用の rxtxSerial.dll を追加（処理が著しく遅くなる場合に利用すること）
- ATmega88, ATmega644p のサポートを追加（ただし、未テスト）

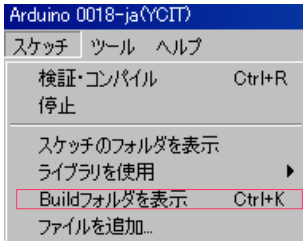
＊1



＊2



＊3



＊4

■ 2010-0531

・Bootloader 自体の書込みと、アップロード時に利用するツールを別々に指定可能にした。
・Readme.odt ファイルの見直し
・Bootloader ファイルの検証（ATmega88 を使って動作確認）

[付録]

・Arduino Duemilanove(ATmega328/16MHz)の FUSE 設定（実機で確認）

```
>hidspx -r
Detected device is ATmega328P.
Device Signature  = 1E-95-0F
Flash Memory Size = 32768 bytes
Flash Memory Page = 128 bytes x 256 pages
EEPROM Size       = 1024 bytes

>hidspx -rf
Detected device is ATmega328P.

Low: 11111111  (FF)
     ||||++++-- CKSEL[3:0] システムクロック選択
     ||++-- SUT[1:0] 起動時間
     |+-- CKOUT (0:PB0 にシステムクロックを出力)
     +-- CKDIV8 クロック分周初期値 (1:1/1, 0:1/8)

High:11-11010  (DA)
     |||||||+-- BOOTRST (1:Normal, 0:BootLoader)
     |||||++-- BOOTSZ[1:0] (11:256W, 10:512, 01:1024, 00:2048)
     ||||+-- EESAVE (消去で EEPROM を 1:消去, 0:保持)
     |||+-- WDTON (1:WDT 通常動作, 0:WDT 常時 ON)
     ||+-- SPIEN (1:ISP 禁止, 0:ISP 許可) ※Parallel 時のみ
     |+-- DWEN (On-Chip デバッグ 1:無効, 0:有効)
     +-- RSTDISBL (RESET ピン 1:有効, 0:無効(PC6))

Ext: -----101  (05)
          +++-- BODLEVEL[2:0] (111:無, 110:1.8V, 101:2.7V, 100:4.3V)

Lock:  0F
```