お久しぶりです黒崎です。
SHT-75 を使ったノードを試験的に５つ量産したので長期間稼働させた時の動作を確認しました。
[実験方法]
１、Arduino Ethernet４台、MEGA１台の構成です。ソフトウェアは安場さんに作って頂いた
UARDECS02 をベースにしていますが、メモリ不足だったので不要な機能を一部削って使わせていただきました。

２、各ノードは温湿度、ファン回転数など７種類の CCM を実装しており、そのうちひとつに
WDT の動作確認用に millis()/1000 で取れる起動からの稼働秒数を直接出力するように仕込んであります。

３、NJU7291D46 で構成した外部 WDT を付加してあります。
loop()内でハートビートクロックを生成しています。２秒固まると再起動します。

４、これら５台のノードと比較用に旧 USE ベースのノードを一台温室内で稼働させてみます。

[実験結果]
１、装置間の気温の誤差は±0.3℃ぐらいです。誤差はほぼ一定ですが、
日照の影響がない時間帯でもいつも高い温度が出るもの、低い温度が出るものがあります。

２、原因は不明ですがたまに WDT が作動するのを確認しました。
Arduino Ethernet４台を稼働させているので、単純計算で稼働時間を４倍として計算してみると
約３３時間に１回の頻度で WDT が作動している計算になります。
リセットが発生する時間帯等に今のところ法則性は掴めていません。

３、安場さんが言われたとおり、MEGA は暴走すると NJU7291D46 でも再起動できませんでした。
プログラムを書き込み中にはあれほど派手に再起動してくれたのに困ったものです。
概ね問題の再現に成功したので、今後はリセット回路を少し変えて試してみます。

黒崎様、

皆様

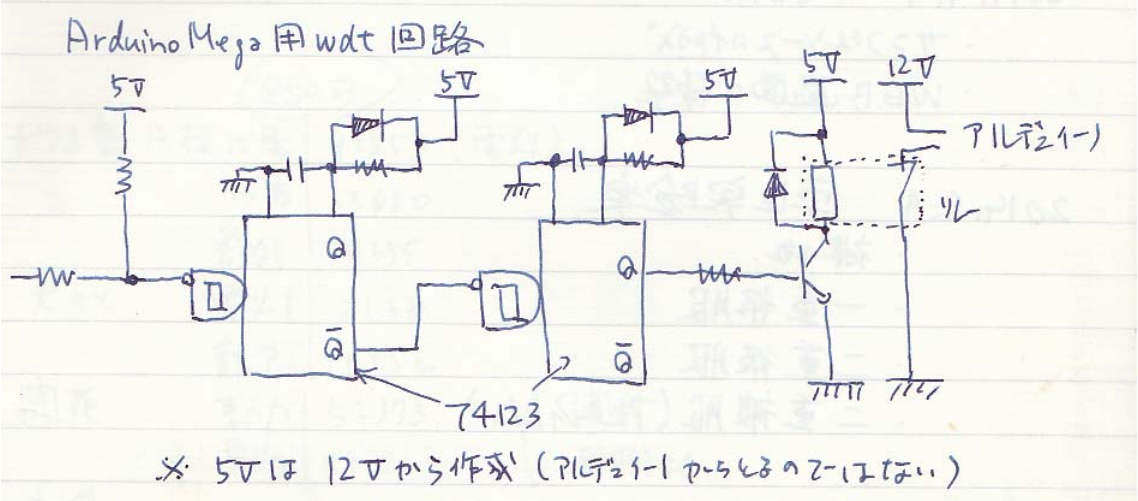
お世話になってます安場です。

MEGA の WDT の件ですが、私の場合ですが、添付ファイルのように 74123 という、マルチバイブレータ用のロジック IC を使ってリレーで強制的に OFF する WDT を使っています。抵抗とコンデンサの充放電を利用して時間を変化させています。

初段はマイコンからのトリガ入力を少し長いスパンのトリガに加工する役割をしていて、CPU の占有時間を少なくしています。後段は定期的にパルス入力が入らないとリレーが OFF されるようになっています。

面倒くさいのは確かですが、温室で動かすとなると、このくらいのものがないと怖くて使えないので仕方がないですね。

ご参考までに



黒崎です。
安場様、情報提供ありがとうございます。
とりあえず、電源遮断による強制リセットと、ブートローダー書き換えによるソフト的な対策の両面を検討してみます。
まずは秋月で売っているＡＶＲＩＳＰｍｋＩＩというツールでブートローダの書き換えが可能との情報がネット上にあったので
それを試してみるつもりです。

黒崎です。
先日の SHT-75 を使った気象観測ノードでリセットがかかる原因についてさらに調

査を行いました。
結局、原因の特定はできなかったのですが、わかったのはセンサおよびDCファンはWDTの作動とは無関係ということです。

実験１
SHT-75＋30cm延長ケーブル＋DCファンで動作試験
結果：３３時間に１回の再起動

実験２
SHT-75＋30cm延長ケーブルで動作試験
結果：３５時間に１回の再起動

実験３
SHT-75をシールドに直付(延長ケーブル無し、ファン無し)で動作試験
結果：５３時間に１回の再起動

実験４
センサをソフト的に使用しない、延長ケーブル無し、ファン無しで動作試験
結果：１６時間に１回の再起動

DCファンだけでなく、センサすら使用していないノードで再起動頻度が上がるというのは
なかなか理解に苦しむところですが、センサを使用しないことによりセンサとの通信の
待ち時間が無くなってloop()の実行頻度が上がっている可能性があります。
つまり、loop()が実行されればされるほど固まりやすくなるということでしょうか？

黒崎様

安場です。

WDTの動作を調べてもらってありがとうございます。

電源のラインのノイズなんかも可能性がありますが、対策としてはフェライトコアで処理するくらいでしょうか。

あとは、私の作ったライブラリに問題がある可能性がありますが、５３時間も動

作させた後に発生するというのは、なかなか原因究明が難しい感じです。LAN コントローラとのやり取りで時間がとられているのであれば、WDT の間隔を少し長くすれば作動間隔が減るかもしれません。

Arduino は気軽に開発できる反面、ブラックボックスになっているところも多いので、問題が発生すると厄介な部分もあると感じます。

黒崎です。
とりあえず、イーサネットの通信周りが怪しいと疑っておりますが、
メモリ容量の多い MEGA でも WDT は結構な頻度で作動するので、本当にメモリ不足なのだろうか？
と考えています。（コンパイル時のスタックの量は同じなのかな？）
なお、複数台用意すれば故障率はその数だけ上がると考えられるので問題の再現が楽になります。

そういえば、ライブラリでは Arduino 内蔵 WDT も ON になっているのでしたっけ、外付け WDT がある状況では不要かもしれません。
MEGA の不具合は内蔵 WDT が作動した時、再起動した直後にそのフラグ（MCUSR）を初期化しないといけないのですが、
それを忘れると無限に再起動してしまう（という仕様らしい）というのが原因か
と考えていますが
手持ちの MEGA の数が少ないので、ある程度数を揃えてから弄ることにします。

黒崎様

安場です。

ライブラリでは WDT は clear にする文は記述してありますが、
内蔵のタイマをスタートさせるコードは自分で記述する必要があります。

LAN コントローラとのやり取りがうまくいかないとと時にリセットできるような仕様にすればひょっとしたら解決するかもしれません。今は時間的に手を付けられないのですが、少し時間がとれたら可能かどうかコードを見直してみます。

Arduino は回路図なども公開されているので安心感があるのですが、バグ取にくろうするようですと使い勝手の良い RaspberryPi にいきたくなりますね。うちの学生には Raspbery で UECS のプログラムを作らせています。バージョンやハードウェアが更新される怖さはありますが。

黒崎です。
WDT の件でプログラム書き込み中にリセットをかけてしまったら ArduinoMEGA2560 のブートローダーが破損しまして、色々と弄ってお りましたら、秋月電子で売られている AVRISPmkII というライタを使って ArduinoMEGA のブートローダー書き換えに成功したので報 告しておきます。

・AVRISPmkII の付属 CD から AVRstudio4 と USB ドライバをインストールする。AVRstudio4 のパッチも入っているので 当てておく。
(AVRstudio5 も出ているが操作がわからないので今回は使わない)
・USB ドライバは警告が出て接続しても認識されないことがあるので注意。警告が出ている場合は強引に許可する。
・ArduinoMEGA は AC アダプタを接続、AVRISPmkII は PC に USB 接続(ドライバ確認)
・AVRISPmkII のケーブルを ArduinoMEGA のリセットボタン横の 6 ピンにつなぐ。(fig1)
よく見ると白い点があるのでそこをケーブルの赤線に合わせる。
・AVRstudio4 を起動、最初に表示される画面はキャンセル
・AVRstudio4 の上の方に黒いゲジゲジで CON と書かれたアイコンを押す
・ライターの選択画面が出るので AVRISPmkII、USB を選んで connect
・AVRstudio4 の上の方に黒いゲジゲジで AVR と書かれたアイコンを押す
・表示された画面の Main タブでデバイスの種類を ATmega2560 に設定(fig2)
・接続に成功すると fig3 および fig4 が見られるようになる。
・fig3 ヒューズビット設定、fig4 ロックビット設定をメモる（データが壊れていない場合）
(たぶん MEGA のデフォルトヒューズビットは FD D0 FF、ロックビットは CF だと思います)
・Program タブより Flash の項目に書き込みたいブートローダーの Hex ファイルをセット(fig5)
なお、デフォルトの mega 用ブートローダーは以下の場所にあるらしい
C:¥Program Files (x86)¥Arduino¥hardware¥arduino¥bootloaders¥stk500v2¥stk500boot_v2_mega2560.hex
・Program ボタンでブートローダー書き込みが可能
・書き込みが完了した後、fig3 および fig4 のヒューズビット、ロックビットを確認
・ヒューズビットはたぶん初期化されないが、変化している場合は元の値に書き換えが必要
・ロックビットは初期化されてしまうので、LOCKBIT 0x??と書かれているところに元の値を入力しなおして Program ボタンで書き換える

・以上で完了、電源を切って OK

ブートローダーが壊れると、USB に応答しなくなりプログラムの書き込みができなくなるようですが、これで復活しました。

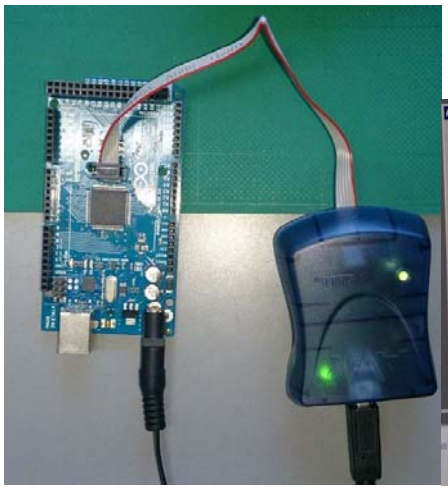


Fig.1

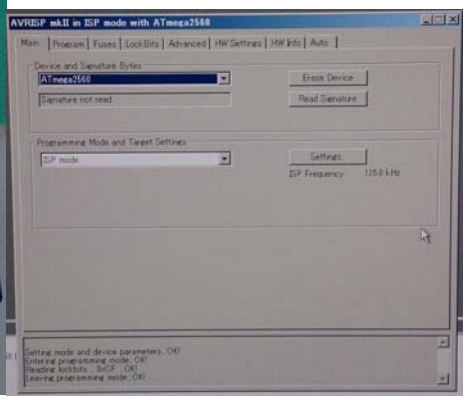


Fig. 2

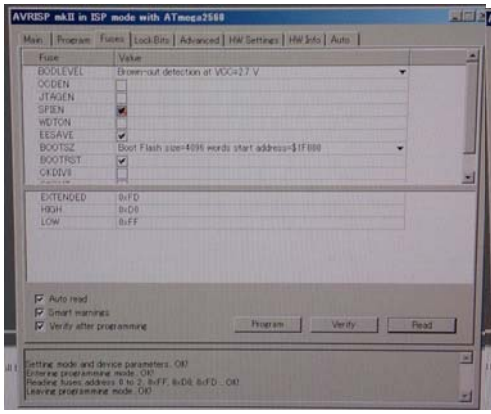


Fig. 3

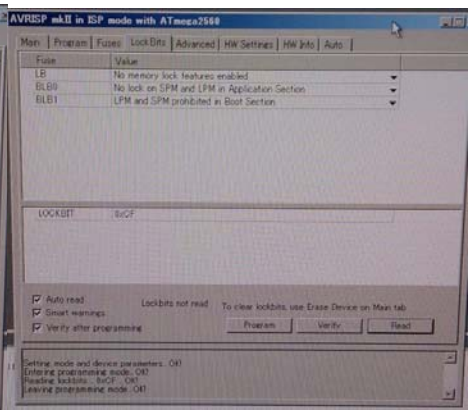


Fig. 4

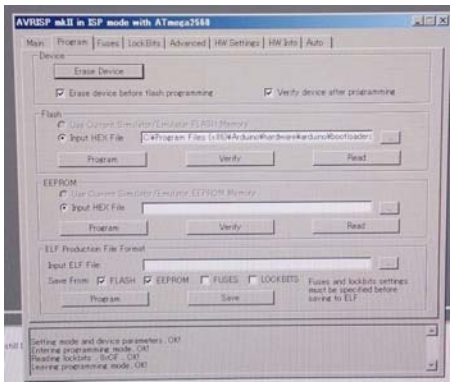


Fig. 5

黒崎です。

安場様、ラズパイの安定性にかなり興味があります。

まだ届いていませんが B+を発注しました。Arduino と比較してスペック上は高性能な上に低価格なので
正直、このまま Arduino をメインにして良いものかどうか悩んでおります。

岡安様、ネットワーク周りで固まる系の報告は他からも上がっており、一番怪しいと思っております。
一度、
・UDP と TCP に分けて試す
・通信頻度が影響しないか試す
・ライブラリの方までデバッグコードを入れてどこで固まるか見てみる
ぐらいはやってみようと思っておりますが、年末になり少々慌ただしくなってきましたので、
しばらく休戦することにいたします。

On 2014/11/20 14:33, Takashi OKAYASU wrote:
> 黒崎様
>
>　貴重な情報ありがとうございます.
>　私の方での経験ですと，ネットワーク接続の辺り，もたつくことがあり，良く
> 停止したため，WDT を実装した経験があります.
>　その辺は関係ないでしょうか？
>
> 岡安崇史
>
> On 2014/11/20 11:45, hoshi@hoshi-lab.net wrote:

On 2014/11/20 14:25, Yasuba　Ken-ichiro wrote:
> 黒崎様
>
> 安場です。
>
> ライブラリでは WDT は clear にする文は記述してありますが、
> 内蔵のタイマをスタートさせるコードは自分で記述する必要があります。
>
> LAN コントローラとのやり取りがうまくいかないとと時にリセットできるような

> 仕様にすればひょっとしたら解決するかもしれません。今は時間的に手を付けら
> れないのですが、少し時間がとれたら可能かどうかコードを見直してみます。
>
> Arduino は回路図なども公開されているので安心感があるのですが、バグ取にく
> ろうするようですと使い勝手の良い RaspberryPi にいきたくなりますね。うちの
> 学生には Raspbery で UECS のプログラムを作らせています。バージョンやハード
> ウェアが更新される怖さはありますが。

黒崎です。
安場さんに作ってもらった Arduino 向けライブラリで低頻度で
WDT が動作する問題ですが、あの後、容量に余裕がある MEGA を使って
ライブラリの細部にテストコードを入れて、全ての動作をシリアル出力するように
改造し、それを PC でキャプチャしながら固まる瞬間を追跡しましたところ
なんとなく目星がついてきました。
発生頻度は概ね１日１回ぐらい、全て UDP 送信部分が呼ばれた直後に固まってい
ます。
さらに、endPacket()の中が原因である可能性が高いです。
ぱっと見た限り問題のあるコードには見えないのですが・・・
ここから先は標準ライブラリの内部になりますが現在追跡中です。

Arduino の UDP ライブラリには初期バージョンには大バグがあったようですが
1.0.1 の段階で大幅に修正されているはずなのですが、1.0.6 でも発生しますので
もしかすると未知のバグがある可能性があります。
一応、論文を出されるとのことなので速報しておきます。

黒崎です。

Arduino のライブラリに怪しい箇所を見つけました。
socket.cpp の中に
sendUDP(SOCKET s)という関数がありますが、
ここにパケットの送信ができなかった場合、while 文でタイムアウトまで待機す
る処理があります。
この while 文の脱出条件が間違っているためデッドロックに陥るのではないか
というのが私の仮説です。今度はこの辺を集中して追跡します。

MEGA の WDT の再起動できない問題ですが、この CPU は WDT を使う場合
WDT 用のフラグレジスタを再起動直後に初期化しないといけないはずなのですが、
これをブートローダーの方でやるのを忘れているようです。

再起動直後に、レジスタの値が残っていると WDT がまた作動してしまい、
短期間に再起動を繰り返すので固まったように見えるわけです。

これは内蔵 WDT を一切使わなければなんとかなる可能性があります。
安場さんのソースを見た感じではやはり内蔵 WDT が有効になっているようです。
UECSsetup()の中に wdt_enable(WDTO_2S);と書いてありましたので
ここを wdt_disable();に変更して、自前で外付け WDT だけを使用するようにすれ
ば良さそうです。
正月休みの間、この状態で動かしてみて休み明けまで無事動いていたらまた報告
します。

On 2014/12/25 13:37, Yasuba　Ken-ichiro wrote:
> 黒崎様
>
> 安場です.
>
> 色々と探索していただきましてありがとうございます.
> WDT タイマー作動の件，わかりました.
>
> 自分のコードも色々と確認してみたことがありますが（メモリ節約というか
> Arduino のコンパイラの問題のためかなりいりこんでいますが..），原因解明
> には至らなかった感じです.
>
> Arduino のライブラリのバグは結構あるのではと思っていますが，フリーなので
> 致し方ないところもあります．MEGA の再起動の問題もそうですが，イーサーネッ
> トコントローラとのやりとりが，どうもうまくいっていない感じです．また，開
> 発したライブラリはそれはそれとして公開しておく方が良いのではと思っていま
> す（WDT は電源ごとリセットをかける必要がありますが）.

黒崎です。
正月休みの間動かしておいた結果が出ましたので報告します。
socket.cpp の内部に sendUDP という関数がありますが以下のように
デバッグコードを仕組みました。

int sendUDP(SOCKET s)

```
{
    Serial.print("¥n*1");
    W5100.execCmdSn(s, Sock_SEND);
    Serial.print("*2");
    /* +2008.01 bj */
    while ( (W5100.readSnIR(s) & SnIR::SEND_OK) != SnIR::SEND_OK )
    {
    Serial.print("*3");
      if (W5100.readSnIR(s) & SnIR::TIMEOUT)
      {
    Serial.print("*4");
        /* +2008.01 [bj]: clear interrupt */
        W5100.writeSnIR(s, (SnIR::SEND_OK|SnIR::TIMEOUT));
    Serial.print("*5");
        return 0;
      }
    Serial.print("*6");
    }
    Serial.print("*7");

    /* +2008.01 bj */
    W5100.writeSnIR(s, SnIR::SEND_OK);
    Serial.print("*8");

    /* Sent ok */
    return 1;
}
```

その結果ですが、
正常な処理では*1*2*7*8 の順番で実行されます。
一方、固まる場合、以下のようになります
*1*2*3*6*3*6*3*6*3*6*3*6*3*6*3*6*3*6*3・・・(以下無限ループ)

if (W5100.readSnIR(s) & SnIR::TIMEOUT)の判定が機能していないのです。
送信後の W5100.readSnIR(s)でほとんどの結果は SEND_OK になるのですが、
ごくごく稀に SEND_OK が帰らない場合があってこの時、

while 文の内部に入った場合、100%固まります
処理の流れを見る限り while 文など不要で、送信に失敗しても
待たずに即退場するように書き換えたほうが良さそうです。

なお、Mega の wdt の実験は成功したようです。
外付け wdt だけを使用することで、リセットされながらも正月休みの間動いていました。

黒崎です。
例の while 文で固まる arduino のバグですが、その後
while 文を廃してすぐに戻ってくるようにしたところ、
今度はメインループは動いているのに UDP の送信が
一切できなくなるという困った事態になり、解決方法を探しているところです。
おそらく 1 回送信エラーが出た段階で W5100 がダウンしてしまい、
何らかの方法で W5100 にリセットをかけないと駄目みたいですね。

黒崎です。
例の UDP バグですが、修正できるかもしれません。
W5100 が固まってもソフト的に初期化し直すことで復活できることを確認しました。
つまり、UDP 送信の失敗を検出した場合、イーサネットを再初期化するようにライブラリを修正します。
socket.cpp と安場さんの作った EthernetManager.cpp,EthernetManager.h
を以下のように改変して動作試験を開始しました。
これで一週間ぐらい何も起きないことが確認できればバグ取りは成功かもしれません。
（SDK バージョン 1.0.6 で試しています）

１，C:¥Program Files (x86)¥Arduino¥libraries¥Ethernet¥utility¥socket.cpp
●sendUDP 関数を以下のように改変
注意：send failed に処理が回った場合、イーサネットを再初期化しない限り UDP 送信が二度とで きなくなります

```
int sendUDP(SOCKET s)
{
    W5100.execCmdSn(s, Sock_SEND);
```

```
if(W5100.readSnIR(s) & SnIR::SEND_OK)
{
/*send ok*/
    W5100.writeSnIR(s, SnIR::SEND_OK);
    return 1;
}

/*send failed*/
  /* +2008.01 [bj]: clear interrupt */
  W5100.writeSnIR(s, (SnIR::SEND_OK|SnIR::TIMEOUT));
    return 0;

}
```

2　　、　　C:¥Users¥user¥Documents¥Arduino¥libraries¥UARDECS02¥EthernetManager.cpp

●UDP 送信に関係する関数を以下のように改変

```
void UECSupdate16529port(UECSCCM _ccmList[], int _maxccm, char* _buffer,
char* _tempStr20, UECSTEMPCCM* _tempCCM){

    //wdt_reset();
    int packetSize = UECS_UDP16529.parsePacket();
     if(packetSize){
        _tempCCM->address = UECS_UDP16529.remoteIP();
        UECS_UDP16529.read(_buffer, BUF);
        if(UECSresNodeScan(_ccmList, _maxccm, _buffer, _tempStr20,
U_name, U_vender, U_uecsid, &U_orgAttribute)){
            UECS_UDP16529.beginPacket(_tempCCM->address, 16529);
            UECS_UDP16529.write(_buffer);
            if(UECS_UDP16529.endPacket()==0)
                {UECSresetEthernet();}//when udpsend failed,reset ethernet
status
        }
```

```
    }

}


void UECSsendCCM(char* _packetBuffer, struct UECSCCM* _ccm){
    UECS_UDP16520.beginPacket(_ccm->address, 16520);
    UECS_UDP16520.write(_packetBuffer);

    if(UECS_UDP16520.endPacket()==0)
            {UECSresetEthernet();}//when udpsend failed,reset ethernet status
}
```

●以下の関数を追加

```
void UECSresetEthernet(){

    UECS_UDP16520.stop();
    UECS_UDP16529.stop();
    UECSinitOrgAttribute();
    Ethernet.begin(U_orgAttribute.mac, U_orgAttribute.ip,
U_orgAttribute.dns, U_orgAttribute.gateway, U_orgAttribute.subnet);
    UECSlogserver.begin();
    UECS_UDP16520.begin(16520);
    UECS_UDP16529.begin(16529);
    UECStempCCM.attribute[0] = -1;
}
```

3　　、　　C:¥Users¥user¥Documents¥Arduino¥libraries¥UARDECS02¥EthernetManager.h

●以下の関数を追加

```
void UECSresetEthernet();
```

黒崎です。

前日お伝えした UDP の送信試験に成功したのでお知らせします。

Arduino Ethernet4 台と MEGA+Ethernet シールド 1 台の動作テストは

ソフトウェア改修後に 10 日間一度も WDT のお世話にならずに連続稼働しました。

この間、ノード起動からの秒数をずっと CCM 出力していましたので、

10 日間以上全てのノードでこの値がリセットされていないことを確認できました。

なお、この実験期間中、負荷として各ノードに 7400 回ずつ HTTP アクセスを行っています。
HTTP のエラー回数は 5 台全部合わせて 10 日間で 4 回のみでした。
エラーは一過性のもので全て次回アクセス時には自動復帰しました。
この時も WDT の作動はありませんでした。
これで、少なくとも HTTP の方には低頻度で発生するような致命的なエラーは無さそうです。

これで Arduino で構成されたノードの安定性がかなり保証されたものと思います。
思いつく限りでは残っているのが UDP 受信側の動作試験と、
文字列のメモリリーク等の調査ですが、UDP 受信側の動作試験は
受けた CCM を別の CCM にして返すノードを作成して既に実施中であり、
近日中に結果が出る見込みです。
メモリリークについては調査が難しく、良いテスト方法が思い浮かばないので
今後の課題とします。

黒崎です。
実はこのバグについては、最新版 SDK の 1.0.6 だけでなく、1.5.8beta でも
ソースコードを見ると治っていないようなので対処方法に気づいたのは
前例がほとんど無いのかもしれません。

On 2015/01/26 17:47, Yasuba　Ken-ichiro wrote:
> 黒崎様
>
> 安場です。
> 原因究明ご苦労様です。
>
> UECS にとってというよりも、Arduino 全体にとって有益な情報ですね。
>
> 今まで、ハードウェアで強制的にハードウェアの電源を落としていた、対処療法
> 的な解決をする必要がなくなって素晴らしいことだと思います。

先日の実験の続きで、Arduino の UDP 受信側のテストを
10 日間連続で行いましたが特にエラーらしいものは発生しませんでした。

また、イーサネットの再初期化を繰り返して副作用がないかどうか調べるために

200ms 間隔で再初期化を 10 日間連続で繰り返しましたがエラー無しです。
特にメモリを食いつぶしたりする様子はありませんでした。

これで Arduino を用いたノードの通信関係のトラブルは無くなったものと思われます。