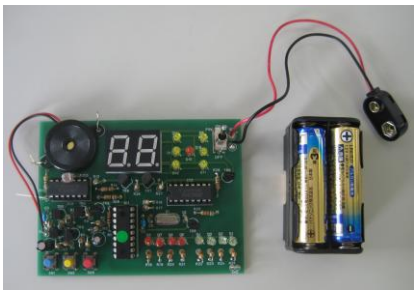
１．はじめに

東日本大震災以来，電力不足が社会問題となり本校でも節電に対する取り組みが強化されている。そのような中，節電グッズが何か作れないかと考え，待機電力をカットする装置を製作することにした。

リモコンを使用する電気製品は，常に待機電力を消費しており，家庭で消費される電力の約１０％を占めるとも言われている。機械式のスイッチを使用していない機器は，コンセントからプラグを抜くことで，待機電力を完全にカットできるが，朝の出勤前に複数の家電製品のプラグを抜くのは面倒である。実際，テレビやエアコンのコンセントを毎朝抜いているという話は，ほとんど聞いた事が無い。そこで，実習で使用している PIC１６F８４A マイコンを利用して，実用性のあるものを作れないかと考え，複数の電気製品を１回の操作で「待機電力ゼロ」の状態にする装置を製作することにした。



図－1　実習ボード

２．製作

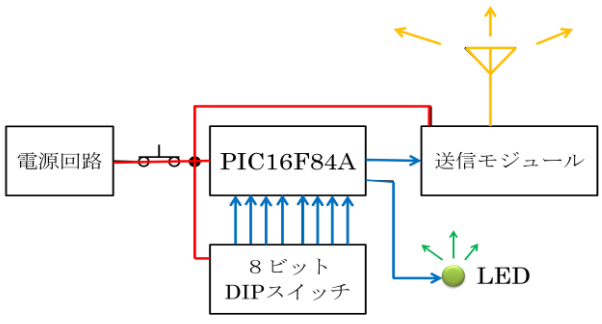
２－１　基本構想

家電製品のプラグとコンセントの間に設置した延長ケーブルの間に，リレー（a 接点）を入れる。家電製品を使用する時はリレーをＯＮにし，使用しない時（出勤前など）にはリレーを切る事で待機電力をカットする。このリレーをＰＩＣマイコンで制御し，待機電力カットの指示は無線のリモコンから行う。また，複数台の電力カット装置を１回のリモコン操作で作動させるようにする。更に，住宅密集地などでの混信を防ぐために，８ビットのディップスイッチから識別信号を作り，この識別信号自体を電力カットの指示をする信号とする。以後，この装置を「カット装置」と呼ぶ。

図－２　リモコンのイメージ

２－２　リモコン送信機

プッシュスイッチを押している間，PIC 及び FM 送信モジュールに電力が供給され，DIP スイッチによって設定された８ビットのデータが FM モジュールから送信される。

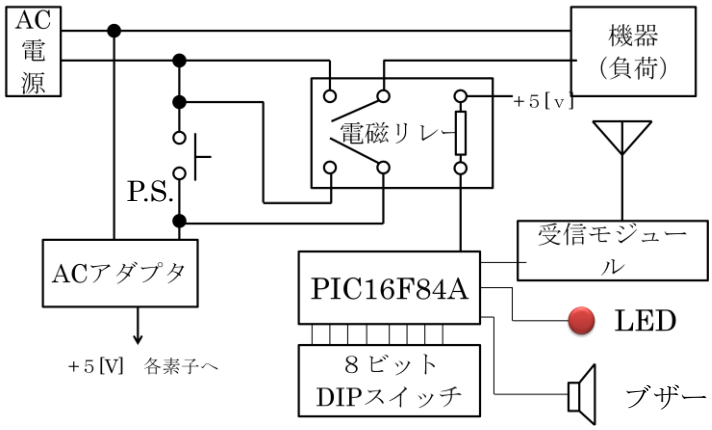


図－３　リモコン送信機（原理図）

２－３　カット装置

プッシュスイッチが押されると AC アダプタに電力が供給され，PIC が作動。PIC は，

すぐに電磁リレーに通電し，自己保持回路によって，AC アダプタへの通電を維持する。このため，プッシュスイッチは，一度押すだけで電力供給状態が保持される。この時が，接続された家電製品を使用できる状態。



図－４　カット装置（原理図）

次に，送信機からの停止信号を受信すると，DIP スイッチの設定と照合し，同じであればブザーを鳴らした後，電磁リレーを停止させて回路を切断する。それに伴い PIC や無線モジュールへの電力も完全にカットされる。

２－４　PIC のプログラム

プログラム開発には，CCS 社の PCW を使用し，C 言語で作成した。

機器停止信号の通信には，シリアルデータ通信規格の RS-232C（調歩同期式）を利用し，エラー対策として，反転２連送照合方式を用いた。ＰＩＣ１６Ｆ８４Ａには USART 機能が無い為に PCW 内に用意された rs232 関数を使用しシリアル通信を行った。

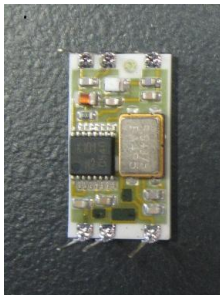
送信機は，ディップスイッチから８bit のデータを B ポートより読み取り，読み取ったデータと反転したデータをシリアルに無線モジュールから識別信号として送信する。

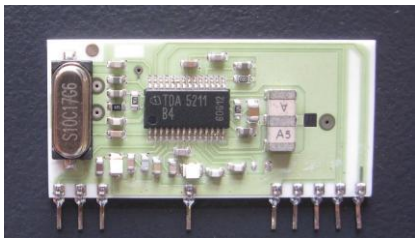
図－５　反転２連送照合方式

カット装置は８ビットのデータを受信し，初めに受信したものと，次に送られたものを反転したものとを比較し，同じであれば正しく受信できたと判断し，電磁リレーの電源を切る。

２－５　無線モジュール

送信機 ― 受信機間の通信には，RF Solution 社の FM 無線モジュール FMRTFQ1-315 及び FMRRFQ1-315 を使用した。この無線モジュールは電波を使って“０”か“１”のディジタルデータを送受信するだけの単純なもので，微弱電波を出力するため電波法にも適合している。このモジュールの購入には Digi-Key Corporation.の通信販売を利用した。注文には日本語を使用できるが，届け先には日本語が使えず，住所や氏名等をローマ字で入力しなければならない。また，使用目的やどの国で使うか？などの入力も求められる。


図－６　FMRTFQ1-315


図－７　FMRRFQ1-315

料金は銀行振り込みが可能で，１ 回７,５００円以上の注文で送料が無料になる。ちなみに７，５００円未満の場合，送料は２，０００円かかってしまう。

２－６　電源回路

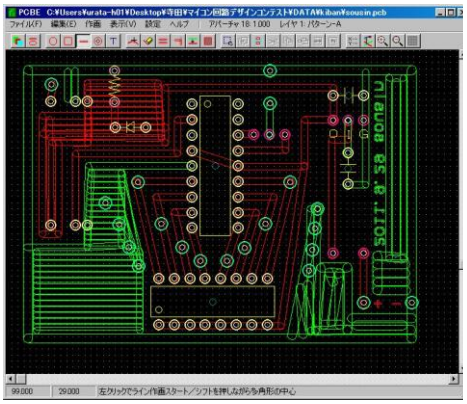
図－８　再利用のＡＣアダプタ

リモコン送信機は，FM 送信モジュールが３．３[V]動作であるため乾電池直列３本の４．５[V]から，３端子レギュレータにより３．３[V]を回路全体に供給している。

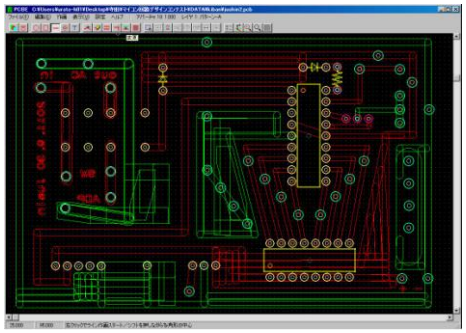
カット装置は，PIC 及び FM 受信モジュールに５[V]の電源が必要なため，AC１００[V]の電源から，ＡＣアダプタにより電力を供給した。このＡＣアダプタは，不要となったゲーム機などの付属品の中から５[V]の物を選んで再利用した。そのため，今回製作した３台の受信機に使用されているＡＣアダプタは，それぞれ形や大きさが異なる。

２－７　基板製作

プリント基板エディタ PCBE で製作したプリントパターンを，Sunhayato 製クイックポジ感光基板に焼き付けた。その後，エッチングを行いオリジナルの基板を製作した。この時，校内にあった５年前製造の感光基板を使用したところ，フォトレジストに失敗し，回路を出すことができなかった。（※メーカーは製造後 １ 年以内を有効期限としている)。改めて新品を購入したところ「露光プロファイル」という説明があり，メーカーの Web サイトを確認したところ, 製造から半年間は露光時間をシビアにコントロールする必要があることが分かった，指示どおりに露光を行い，無事製作する事ができた。



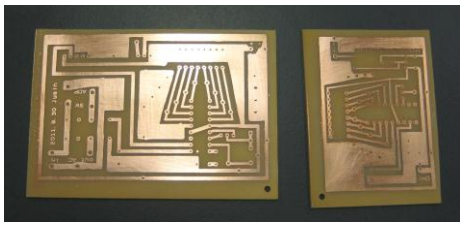
図－９　リモコン送信機（パターン）



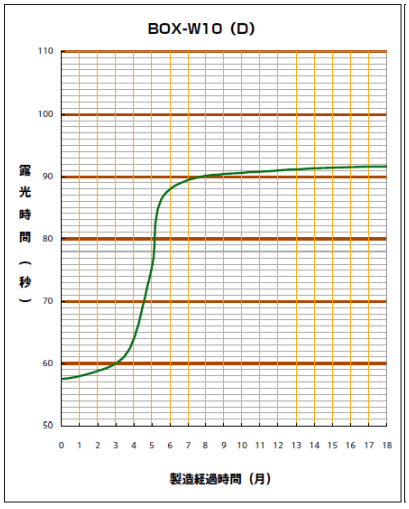
図－１０　カット装置(パターン)

穴を開けた基板にパーツを半田付けして完成。

この回路はAC１００［V］を扱うため, 安全のために１００円ショップで購入したハガキ入れに組み込んだ。

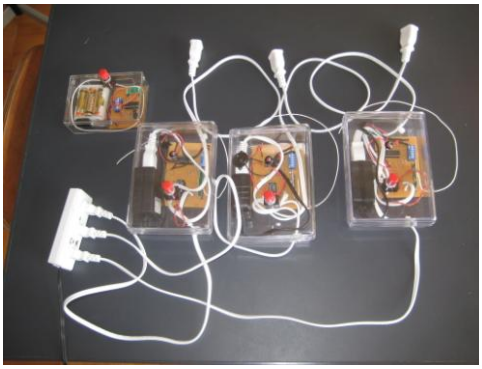


図－１１　エッチング後の基板



図－１２　露光プロファイル

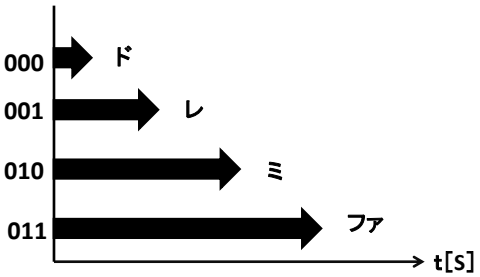
３．テストと改良



図－１３　完成した装置一式

完成した装置をテストしてみると，ほとんど問題無く作動したのだが，新たな問題点が浮上した。今回製作した３台のカット装置が一度のリモコン操作で同時に動くと，３台の装置が一斉にブザーを鳴らすため，故障した際，どれが壊れているのか直ぐに判断できない。回路を変更せずに，ソフトウェアのみの変更でこの問題を改善できないかと考え，各カット装置が別々の音を出せるように改良した。音の違いは，ブザーへの信号を FM 音源（PWM）で行った。PIC16F84A には PWM 機能が無いため，ソフトウェアでパルス幅を調整し，ドレミの音階にした。また，同時に鳴らすと聞き分けにくくなるので 1 台ごとにタイミングをずらして音が鳴るようにした。また，各カット装置を共通のプログラムで動かすために，ディップスイッチの下位 3bit で音階を設定できるようにした。

リモコン送信機の変更は全く行わず，カット装置側は受信機が受信した８bit の信号から，マスク処理により上位 5 ビットを識別信号として使い，ディップスイッチ上位５bit と比較して，同じであれば電源カットの動作に進む。ディップスイッチ下位３bit は，’000’の時はすぐに‘ド’の音を出して電源カット。’001’時は約１秒後に‘レ’の音を出して電源カット。といった具合に８通りの動作が選べるようにした。これにより，送信機では８Bit，２５６通りの識別信号を送信できるが，受信側では５bit，３２通りしか識別できなくなってしまった。



図－１４　設定と作動方法

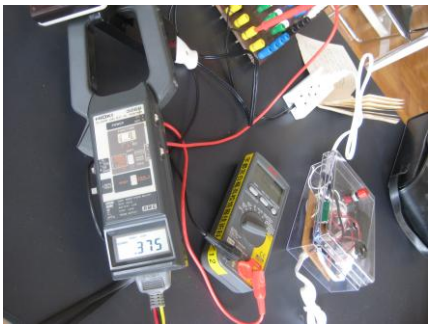
表 - 1　音階と周波数

ド（低い）	261.6Hz	基本となる周波数
レ	293.7Hz	基本となる周波数の 1.122 倍
ミ	329.6Hz	基本となる周波数の 1.260 倍
ファ	349.2Hz	基本となる周波数の 1.335 倍
ソ	392.0Hz	基本となる周波数の 1.498 倍
ラ	440.0Hz	基本となる周波数の 1.682 倍
シ	493.9Hz	基本となる周波数の 1.888 倍
ド（高い）	523.3Hz	基本となる周波数の 2.000 倍

４．考察

今回製作した装置がどれくらい節電できるかを考察した。規格表等による理論値は，PIC マイコン５０〔mW〕，FM 受信モジュール２８．５〔mW〕，電磁リレー１５０〔mW〕，LED２０〔mW〕。これを合計すると０．２４８５〔W〕となり約０．２５〔W〕消費することになる。しかし，AC アダプタに関しては，資料が無いため，これ以上理論値を出すこ

とができなかった。そこで，測定によって実測値を求める事にした。クランプメータで測定したところ，電圧値１００〔V〕，電流値１２〔mA〕，力率０．４であった。計算すると０．４８〔W〕となりカット装置を取り付けると約０．５〔W〕余計に電力を消費することになる。しかし，一般的な家電製品の待機電力は，モデム６．６〔W〕，エアコン２．４〔W〕，DVD プレーヤー１．２〔W〕と言われている。



図－１５　クランプメータによる実測

DVD プレーヤーが１日に消費する待機電力は
１．２〔W〕×２４〔h〕＝２８．８〔Wh〕
カット装置を装着すると
スタンバイ状態　１．２〔W〕＋０．５〔W〕＝１．７〔W〕
電力カット時　０〔W〕
カット装置未装着時と同じだけの電力を消費するには，
２８．８〔Wh〕÷１．７〔W〕＝１６．９４〔h〕　　約１７時間となる。
増加分を補うためには，
２４〔h〕―１７〔h〕＝７〔h〕
つまり，7 時間以上待機電力をカット出来れば節電になる。

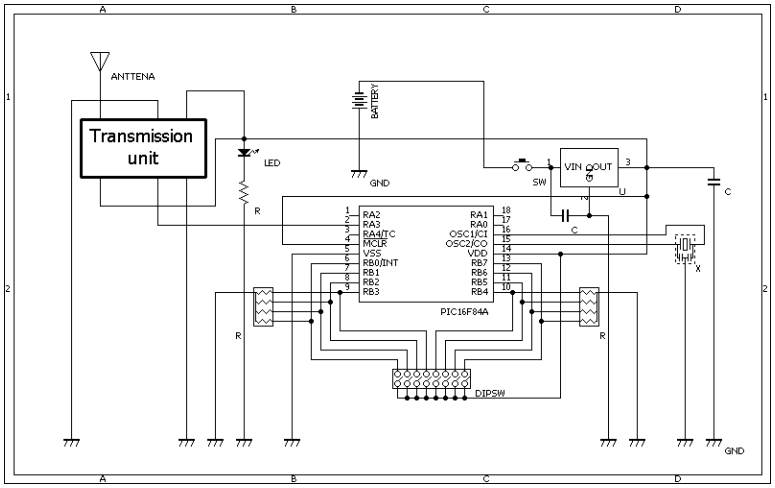
残念ながら，カット装置本体の消費電力が０．５〔W〕となり，更なる省電力化の課題が残った。ここから，カット装置の消費電力を減らすためには，「PIC のクロックを落とす」，「機械式リレーを SSR に変更する」，「PIC16F84A を nanoWatt XLP テクノロジ採用の超低消費電力 PIC に変更する」などが考えられる。

今回，PIC16F84A で制御を行ったため USART や PWM 機能を使用できなかった。しかし，短方向のシリアル通信や delay 命令を使った PWM 制御に取り組むことができた。また，参考資料も多いため，プログラムを含めてほとんど全ての作業を生徒の力だけで行うことができた。担当した生徒は，技術力や問題解決能力が身についただけでなく，節電に対する意識も向上したと思う。

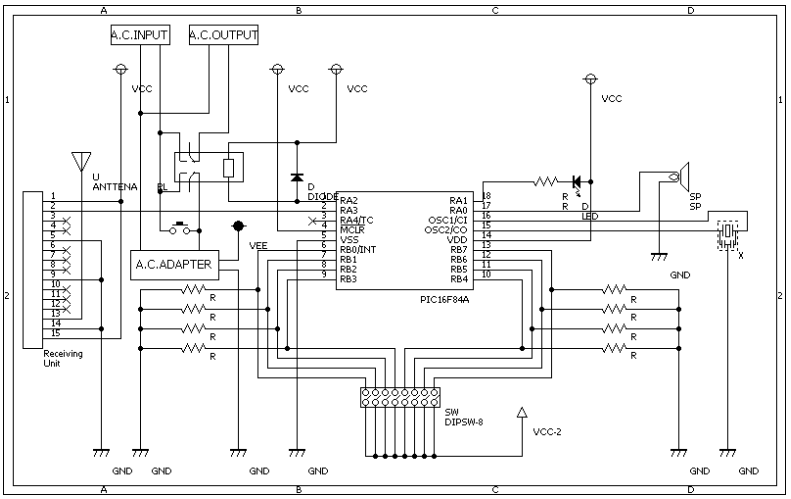
[参考文献等]

・平成 17 年度　関東甲信越地区電気教育研究会
　川崎市立川崎総合科学高等学校　白川賢一先生発表
「無線送受信モジュールを使ったシリアル通信制御」
・サンハヤト株式会社 web サイト　http://www.sunhayato.co.jp/index.php
・FMRTFQ1-315　及び　FMRRFQ1-315　データシート
・PIC16F84A　Data Sheet　http://akizukidenshi.com/download/PIC16F84A.pdf
・941 Series Relay　http://akizukidenshi.com/download/941H-2C-5D.pdf
・音楽研究所 web サイト
http://www.asahi-net.or.jp/~hb9t-ktd/music/Japan/Research/Genre/Tuning/tuning_12.html

[参考資料]　リモコン送信回路図



カット装置回路図



リモコン送信プログラム

```
#include <16f84.h>
#use delay(clock=10000000)              /* クロック信設定 */
#fuses HS,NOWDT,NOPROTECT,PUT           /* コンフィグレーション設定 */
#use rs232 (BAUD=4800,XMIT=PIN_A3)      /* 通信設定 */
#byte port_a=5
#byte port_b=6
#define STX   0x02                     /* STX キャラクタデータセット */
#define ETX   0x03                     /* ETX キャラクタデータセット */

void main(){
   int f_data, r_data, i;
   set_tris_a(0x00);                   /* A ポート入出力設定　全て出力 */
   set_tris_b(0xff);                   /* B ポート入出力設定　全て入力 */
   port_a=0x00;                        /* A ポートリセット */
   while(1){
      for(i=0;i<3;i++){
      putc(STX);                       /* STX 送信 */
      }
   putc('s');                          /* 's'送信 */
   f_data = (port_b);                  /* B ポートのデータを f_data に代入      */
   r_data = (~f_data);                 /* B ポートの反転データを r_data に代入 */
   putc(f_data);                       /* f_data 送信 */
   putc(r_data);                       /* r_data 送信 */
   putc(ETX);                          /* ETX 送信 */
   delay_ms(2);                        /* 待機 */
   }
}
```

カット装置プログラム

```
#include <16f84.h>
#use delay(clock=10000000)                          /* クロック信設定*/
#fuses HS, NOWDT, NOPROTECT, PUT                    /* コンフィグレーション設定*/
#use rs232 (BAUD=4800,RCV=PIN_A3)                   /* 通信設定*/
#byte port_a=5
#byte port_b=6
#define do_low 218                                  /* 低いドの周波数データ*/
#define re 193                                      /* レの周波数データ*/
#define mi 171                                      /* ミの周波数データ*/
#define fa 163                                      /* ファの周波数データ*/
#define so 142                                      /* ソの周波数データ*/
#define ra 128                                      /* ラの周波数データ*/
#define shi 113                                     /* シの周波数データ*/
#define do_high 109                                 /* 高いドの周波数データ*/
void main(){
  int f_data, r_data, mf_data, m_port_b, s_port_b;
  int i;
  long a, x, y, z;
  char start;
  set_tris_a(0x08);                                 /* Aポート入出力設定RA4のみ入力にR */
  set_tris_b(0xff);                                 /* Bポート入出力設定全て入力*/
  output_low(PIN_A2);                               /* RELAR */
  output_high(PIN_A1);                              /* LED */
  output_high(PIN_A0);                              /* SPEAKER */

  while(1){
    start = getc();                                 /* データ受信*/
    f_data = getc();
    r_data = getc();
    output_low(PIN_A1);                             /* リレーON */
    if(start =='s'){
      if(f_data == (~r_data)){
        mf_data = f_data &0x07;                     /* f_data マスク処理　識別信号取り出し*/
        m_port_b = port_b &0x07;                    /* port_b マスク処理　識別信号取り出*/
        s_port_b = port_b &0xf8;                    /* port_bot マスク処理　音階信号取り出し*/
        if(mf_data == m_port_b){                    /*受信データ, Bポート識別データ比較*/
          switch(s_port_b){                         /*Bポート音階データ比較*/
            case 0xf8:                              /* 000の時*/
            x = 0; y = 1000; z = do_low; break;
            case 0xf9:                              /* 001の時*/
            x = 2000; y = 1165; z = re; break;
            case 0xfa:                              /* 010の時*/
            x = 4000; y = 1315; z = mi; break;
            case 0xfb:                              /* 011の時*/
            x = 6000; y = 1380; z = fa; break;
            case 0xfc:                              /* 100の時*/
            x = 8000; y = 1583; z = so; break;
            case 0xfd:                              /* 101の時*/
            x = 10000; y = 1761; z = ra; break;
            case 0xfe:                              /* 110の時*/
            x = 12000; y = 1992; z = shi; break;
            case 0xff:                              /* 111の時*/
            x = 14000; y = 2062; z = do_high; break;
          }
          output_high(PIN_A1);                      /* LED オフ*/
          delay_ms(x);                              /*動作待ち時間*/
          for(a=0;a<y;a++){
            for(i=0;i<z;i++){
              output_low(PIN_A0);                   /*ブザー出力ON */
            }
            for(i=0;i<z;i++){
              output_high(PIN_A0);                  /*ブザー出力OFF */
            }
          }
            output_low(PIN_A0);                     /*ブザー出力リセット*/
            delay_ms(100);                          /*動作待ち時間*/
            output_high(PIN_A2);                    /*リレー出力OFF */
        }
      }
    }
```