2010
回路図集

麻布学園物理部無線班

はじめに

昨年の回路図集では「オクタコアCPUが現実のものとなりつつある」とありましたが、最近ではヘコア/12スレッド対応のCPUが発売されるなど、クロック面での限界を既に迎えていたCPUは、並列処理を強化する方向に、本格的にその進化の舵を切っています。

我が部でも32bitマイコンが実用化されたり、多様なマイコンが使われるようになったりなど、何とか時代に追いつこうとしている所です。また最近では、FPGAなど高性能なプログラム可能論理素子の普及によって、一般の人々でもCPUなどを使えるようになってきており、最新の技術も夢ではないのかもしれません。

この「回路図集」は、麻布学園物理部無線班の部員らが1年をかけて研究、製作したものについて詳細が記されているものです。ただし、未完成のものも含まれており、必ずしも本書の通りに作って、うまく動作する保証はありません。あらかじめご了承下さい。

H2 会計　池上 涼平

目次

Webサイト

http://btmseisaku.blog76.fc2.com/

本誌に載っていない回路図・プログラムもここで参照できます

```
        int i;
        for (i = 0; i < N_MAX_FILESYSTEM; i++) {
            if (Filesystem[i] == 0) {
                Filesystem[i] = fs;
                return 0;
            }
        }
        return -1;
}

int nFilesystemMount (
        const char* fsname,
        int diskNumber,
        const char* mountPoint
)
{
        int fs, i;
        NLogicalDisk* disk = LogicalDisk + diskNumber;

        fs = nFilesystemSearch(fsname);
        if (fs < 0) return 1;

        if (mountPoint) {
            if (strlen(mountPoint) >= N_MAX_MOUNTPOINT) return -1;
            disk->diskWorkspace = k_malloc(Filesystem[fs]->sizeOfDiskW
            if (disk->diskWorkspace == 0) return -1;
            if (Filesystem[fs]->mount_r(diskNumber, disk->diskWorkspace
                k_free(disk->diskWorkspace);
                return 1;
            } else {
                strcpy(disk->mountPoint, mountPoint);
                for (i = 0; i < N_MAX_MOUNTPOINT - 1; i++) {
                    if (disk->mountPoint[i] == 0) {
                        if (disk->mountPoint[i - 1] != '/') {
                            disk->mountPoint[i] = '/';
                            disk->mountPoint[i + 1] = 0;
                        }
                        break;
                    }
                }
                disk->filesystem = Filesystem[fs];
                return 0;
            }
        } else {
            if (Filesystem[fs]->mount_r(diskNumber, 0)) {
                return 1;
            } else {
                k_free(disk->diskWorkspace);
                return 0;
            }
        }
}

static NFile* nFileOpenReal (
        int logicalDiskNumber,
        const char* name,
        uint8_t mode
)
{
        NFilesystemResult result;
        int i;
        int fn = -1;

        for (i = 0; i < N_MAX_FILE_OPEN; i++) {
            if (OpenedFile[i].fileWorkspace == 0) {
                fn = i;
                break;
            }
```

・GENOME・

- GENOME is an Environment of Multimedia Elements -

製作：
M3 梅津
M3 川田
H2 池上

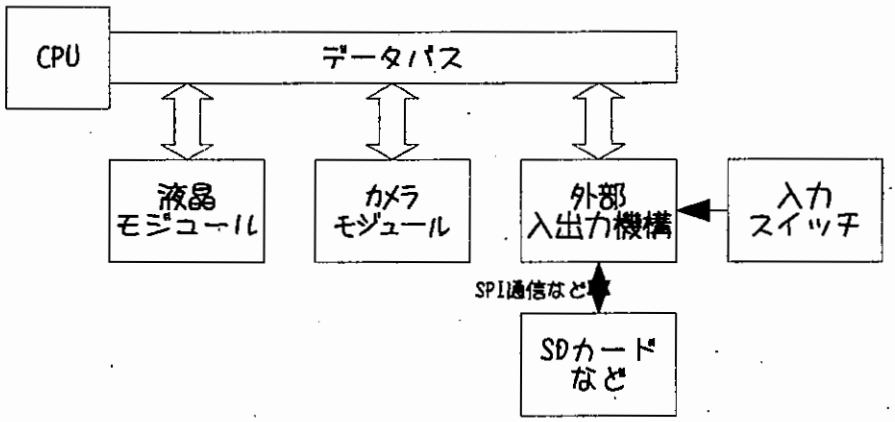
協力：
物理部の方々
インターネット

・概要

今回の作品は、「マルチメディア・ジュークボックス」一要するに、画像や音楽などを扱うこと（撮影とか再生とか）のできる簡単なコンピュータ（？）の様なものです。iPo#の様なものと考えてください。その中でも主な機能は、カメラによる写真の撮影と保存、320*240のカラー液晶に写真を保存することです。ほかにも様々な機能を搭載することもでき、割と何でもできるコンピュータを目指しています。例：ゲーム、音楽再生

・構造

簡単な構造を図で表しました。詳しい仕組みは追って説明します。



・各部分の簡単な説明

各部分の簡単な説明をします。詳しくは、後に書きます。

CPU:
中央演算処理装置。Renesasのマイクロコントローラ「SH7144」を使用します。この作品の頭脳にあたり様々な計算、処理を行います。例えば、液晶に操作画面を表示するなどです。

液晶:
320*240のカラー液晶を使い、画像などを表示します。

カメラ:
CMOSカメラモジュールというものを使用ます。これを使うと、映った画像がデータとして出てくるので、それを保存したり表示したりします。

SDカード:
写真データをここから読み出したり、ここに書き出したりします。SDカードは、電子工作でとても簡単に使うことができたりするので使っています。

• CPU

H2 池上 涼平

この作品では、例えばカラー液晶に画像や操作画面を表示したり、カメラからデータを受け取ったりするようなデータ処理を行うのですが、その処理を実際に執り行うのがCPU（中央演算処理装置）です。ここでは、そのCPU周りについて、説明をすることにします。

:: SH7144マイコンについて

今回使ったCPUは、秋葉原の「秋月電子通商」で売られている「SH7144マイコンボード」（￥2,650）です。Renesas社のSuperHマイクロコントローラ「SH7144」および、最低限の周辺回路がまとめられています。SH7144の動作クロック（動作の速さ）は50MHzとパソコンに比べれば数十分の一ですが、この値段で手に入る使いやすいマイコンとしては最強です。

このマイコンの開発は「GCC Developer Lite」（GDL）という、GCCというCコンパイラを使ったマイコンの開発環境をつかって、簡単に行うことができます。GDLに搭載されているライタソフトを使って、パソコンからRS232Cのシリアルケーブルで SHマイコンの所定のピンにつなぐと、書き込みが行え、しかも同じピンで動作中のSHマイコンとパソコンとで通信を行うことができます（もちろん電圧変換が必要。つなぎ方は回路図を参照ください）。これはH8シリーズ（H8、H8X、SHなど）のマイコンの特徴です。開発の簡単さでいっても、秋葉原で手に入るマイコンの中ではトップでしょう。今回は、USB-シリアル変換ケーブルを使って開発をしています。

:: SRAMについて

今回の作品の肝となるのが「SRAM」です。これはRAM（電気信号によって多くのデータを一時的に保存する部品。電源を切ればすべてが吹っ飛ぶ）の中では最も簡単で扱いやすい部品です。

画像を扱うとなると、画像のデータを一時的に保存するのに、大容量のRAMを必要とします。後で詳しく述べますが、例えば、カラー液晶は、常に表示するデータを右上から順番に送っていないと、光りません。常時CPUがデータ信号をを液晶に送り続ける訳には行かないので（ほかの作業ができないし、CPUはそういう信号処理には向いていない）、SRAMに一時的に保存する必要があります。そうすれば、後は液晶の表示モジュールが勝手にそのデータを送ってくれるという訳です。また、CMOSカメラも同じようなもので、常に順番にデータが送らてくるのを一時的に受け取るのにSRAMを使います。

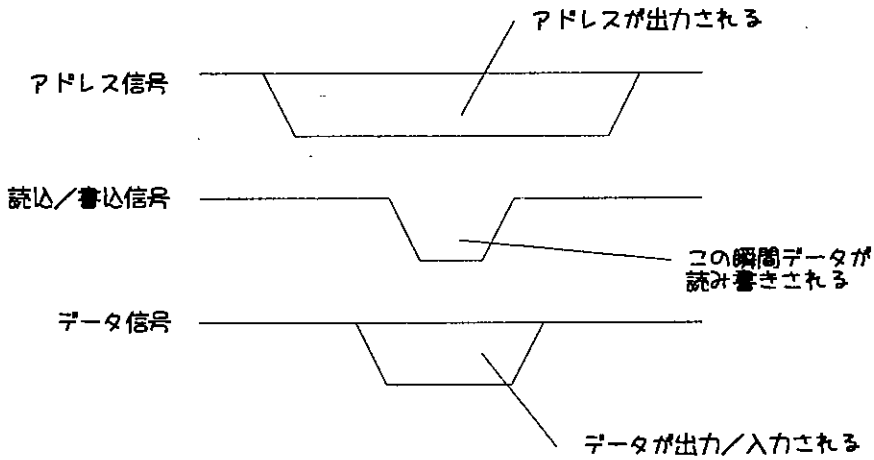
SRAMは、主に8bitや16bitなどのデータを保存できるもの（ラッチ）が、番号（アドレス）を振られてたくさん入っているものです。それに書込むときは、アドレス信号をまず出力して、書込みたいデータもデータ信号線に出力します。そして、書き込み信号をオンにする（通常負論理（0でオンになる）信号なので、0にする）と、晴れてデータが書込まれます。読込むときは、同じくアドレスを出力して、データは入力にしておきます。そして、読込み信号を0にすると、データがデータ信号線にSRAMから出力され、データを読みだすことができます。

:: 外部メモリ

また、このSRAMへのアクセスの仕方は、巷のCPUがメモリのデータにアクセスするときに使う最も基本的な方法です。このようなアクセス方式は、CPU側からの視点で「データバス」（或は、単に「バス」とも）いいます。このデータバスによってアクセスできるデータ空間のことを、「アドレス空間」と抽象的に言います。SRAMは、CPUのメインメモリとして簡単に使うことができるのです。通常のマイコンの場合、このデータバスは外へ信号として出ていないことが多いのですがSH7144は使いやすい形で信号が出ているので、SRAMを使うことにしました。今回は、液晶やカメラだけではなく、CPUのSH7144も大量のデータを扱うので、SRAM(512kB)「高速SRAM 4Mビット CY7C1041DV33 (10ns)」をコイツのメモリとして接続しました。

データ信号	アドレス信号	書込信号	読込信号

データ信号（8,16,32,64本など）：　アドレスで指定した場所のデータの受け渡しに使う
　　書込む際は、出力信号
　　読込む際は、入力信号
　　になる。
アドレス信号（たくさん）：扱うデータの番号（アドレス）を指定する
　　この本数で扱えるデータの総サイズが決定する
書込信号（1本程度）：　　この信号がオン（0）になるとデータが書込まれる
読込信号（1本程度）：　　この信号がオン（0）になるとデータを読込む



:: データバス接続とメモリマップトI/O

ちなみに、CPUと液晶、カメラなどの外部モジュールをつなげるのに使ったのも、このアクセス方式と同じバス接続です。要するに、液晶やカメラのモジュールなどを**一つのSRAMとして扱う**のです。「アドレス空間上に液晶モジュールなどを配置した」と言うこともできます。こうすれば、CPUから簡単にかつ高速にアクセスすることができるのです（ただし、配線量が異常ににに多くなるのが玉に瑕（なんと40本近く））。

あるいは、このデータバスの先に、SRAMの代わりにラッチ（書込／読込信号が入ると入力の信号を内部に保存したり、内部に保存されている信号を出力する、といったデジタル部品。実はSRAMもこれを基にして作られている）などを決められたアドレス信号のときだけに反応するように作れば、入出力ポートをつくることができます。これを「**メモリマップト**（Memory-mapped）**I/O**（入出力）」といいます。メモリに読込んだり書込んだりするのと全く同じ要領で信号の入力や出力を行うのでこう表現されます。PICマイコンなどの入出力ポートもこれでできています（因みにIntel/AMD系のCPUは少し違うらしいです）。

:: 追加入出力モジュール

また、データバスに接続され、SDカードやオーディオ再生ICなどと通信（主にSPI通信）をするための機能を持った入出力モジュールを作りました。XilinxのCPLD「XC9572XL」を使用しています。

このCPLDでは、メモリマップトI/Oの仕組みを使って、いわばまさにマイコンのSPIモジュールの外部版という感じで、ラッチとシフトレジスタを使ってSPI通信を実現しています。中にラッチを組み込むことによってレジスタをアドレス空間上に配置し、そのレジスタの読み書きでSPIを送ったり受け取ったり、という形になっています。SH7144にもSPI通信機能はついてはいるのですが、速度が遅いのでこのような形で独自に作ることにしました。

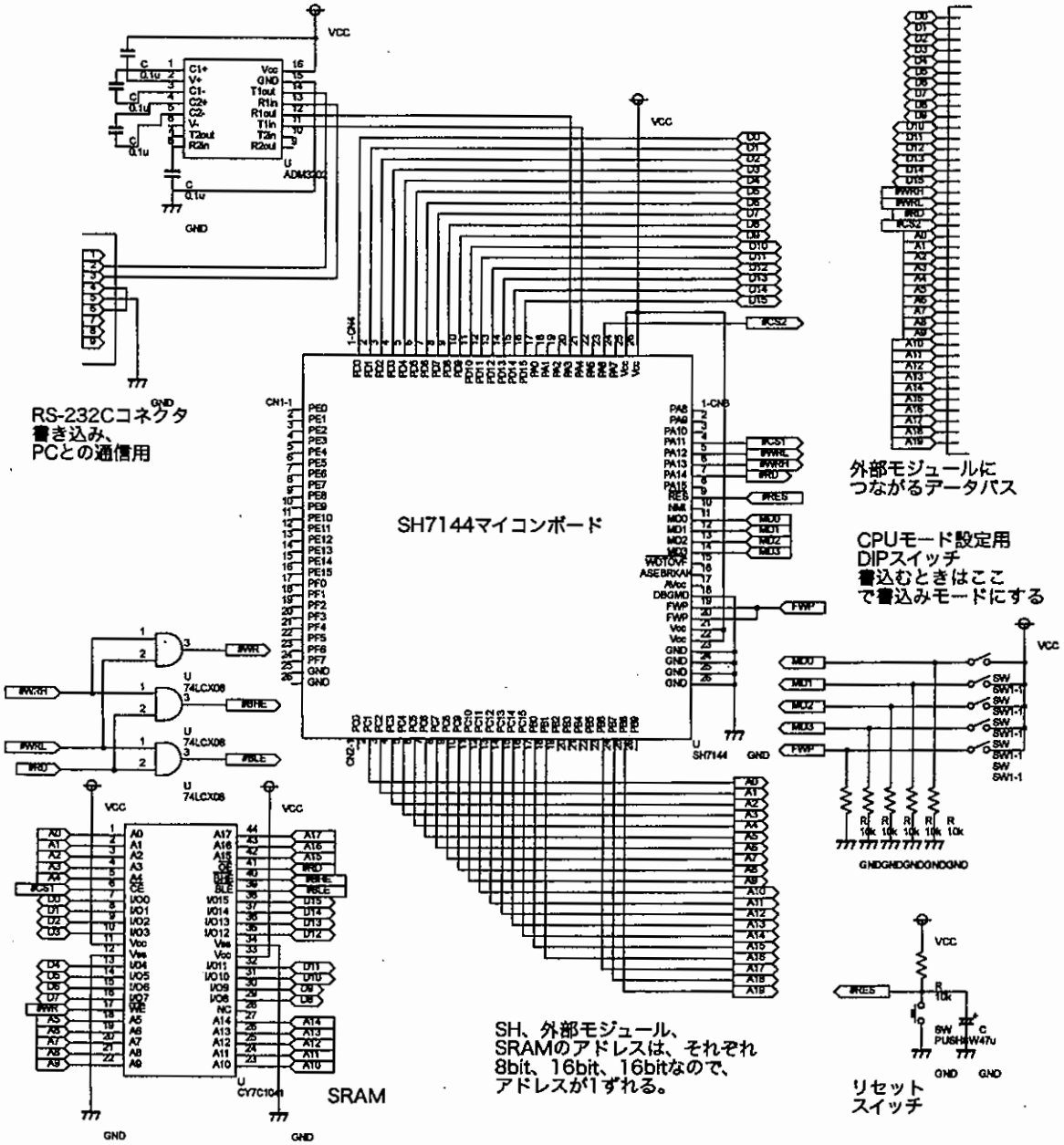
:: SDカード

SDカードは、USBメモリとともに、気軽に携帯電話やオーディオプレーヤで使える記憶媒体です。SDカードには、独自の通信規格を使うモードと、SPIモードといってSPI通信を使ってデータを読み書きするモードがありますが、後者のSPIモードの方が、電子工作の分野で一般的（ほとんどのマイコンで標準搭載している）なSPI通信を使うことができるので、今回はSPIモードを使用しました。

なお、今回SDカードを扱うプログラムで参考にしたのは、「FatFs」のサンプルプログラムです。

:: 回路図

執筆時点で正しい動作が確認されている部分のみ掲載します。全てのVccは3.3Vですので注意してください。SH7144ボードについての詳細は付属の説明書或はネットの記事がさらに詳しいです。



・ソフトウェア

H2 池上 涼平

CPUが立派なものを積んでいたとしても、その上で動くプログラムがなければ意味はありません。という訳で、この作品に搭載されているプログラムを紹介します。

このCPUの上で動くプログラムは、主に次のような機能をもっています。

・SDカード上のファイルの読み書き（改造すればハードディスクなども可能なプログラムになっています）

これによって、画像データを読込んだり、カメラからの画像データを書き出したりができます。

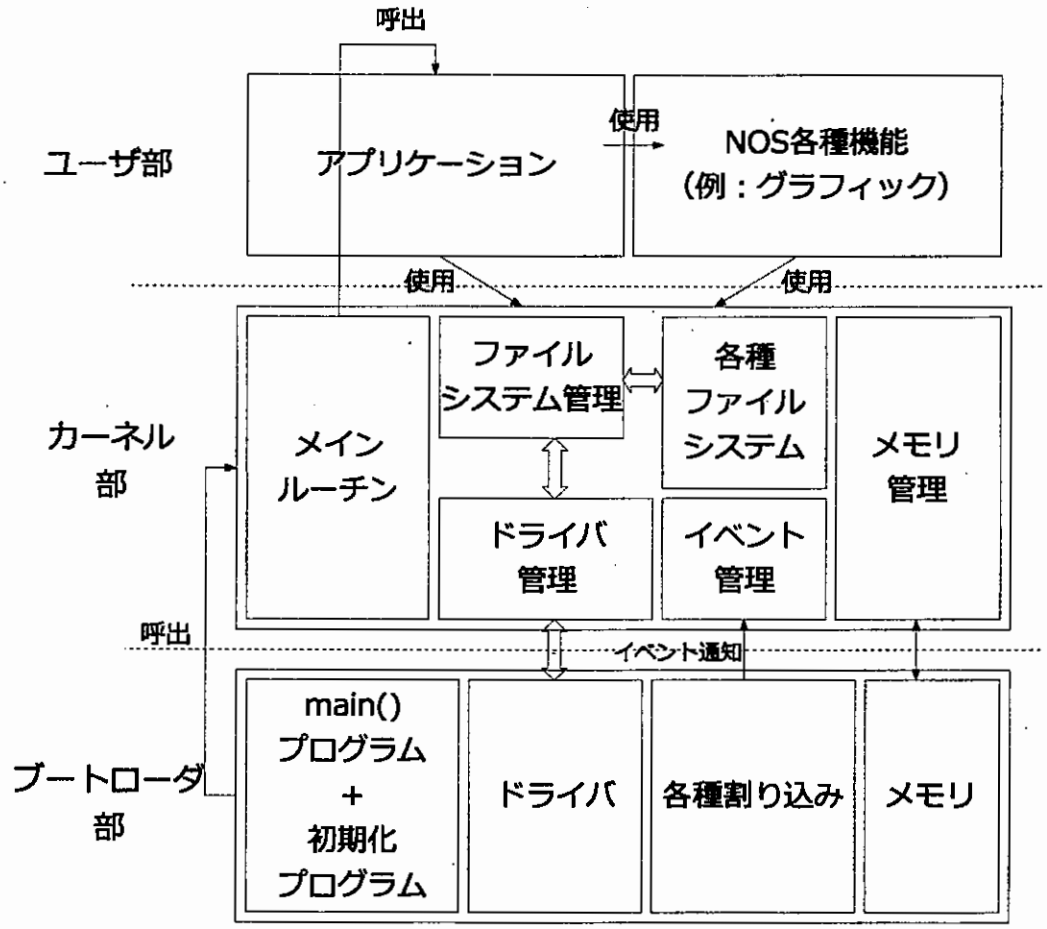
・液晶画面への表示

これによって、画像データや操作画面を映すことができます。例えば、ファイルブラウザの画面を表示して、選択した画像やテキストを見る、といったことができます。

ここでは、そのような機能がどのように搭載されているかを紹介したいと思います。

:: NOSの構造

このシステムは、「NOS」（なんちゃってOS）と勝手に僕が呼んでいます。その名の通りOSもどきのソフトです。それなりに構造化しているつもりです。



このNOSは、主に４つのモジュール（機能の集合体）に分かれています。

・ブートローダー部

よくパソコンが起動しなくなったとき「BIOSが壊れた」なんて言うことがあります。BIOSとは、パソコンに電源が入って最初に実行されるプログラムで、ICの中などに入っています。これと同じように、NOSにも「ブートローダー」と呼ばれる、最初に実行されるプログラム群があります。SHマイコンに電源が入ったときに実行されるmain()関数もありますが、それ以外にも、SDカードなどの周辺デバイスのドライバとなる関数たち（全てのドライバは、共通の規格で書かれています）、割り込みが発生したときに最初に実行される関数など、ハードウェアに直接触れるプログラム類で構成されます。すなわち、ここの部分を使うマイコンによって書き換えるだけで、NOSはほかのマイコンでも動くように設計されています。

・カーネル部

ここは、NOSのシステムの中核を担うプログラム群です。例えば、ファイルシステムを扱う部分や、割り込みなどを管理するイベント管理部や、メモリの動的割当（「malloc()」とかです）を行う部分などが含まれます。もしかしたら、文化祭当日にはマルチタスク管理機能も含まれているかもしれません。

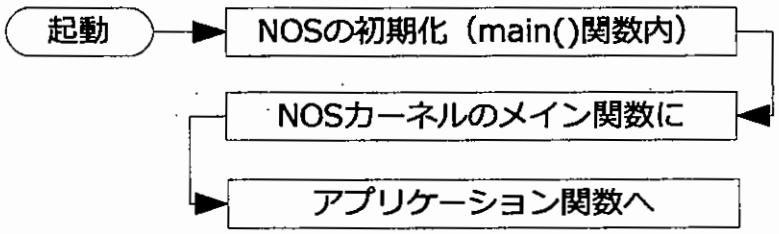
・NOS各種機能

例えば、画像ファイルを扱う機能や、グラフィックを描く機能、その他諸々のNOS機能で構成されます。これは、アプリケーション部と同列（ユーザ部）として扱われ、カーネルの機能を使うことによって成り立っています。

・アプリケーション部

ここは、やりたいことに応じていろいろなプログラムが書ける部分です。NOS各種機能とカーネルの機能を使うことができます。

処理の順番としては、次のようになります。



:: モジュール化について

基本的に、大規模なプログラムは、いくつかのモジュールに分けて作られることが必要になります。モジュールとは、機能の集合体で、複雑なデータの受け渡しはモジュールの中のみで行われ、モジュール同士の間では簡単なデータの受け渡ししかしません。これは、全体の構造を分かりやすくして、開発を楽にするためです。例えば、部品同士が盛りそばのような配線でつながれたパソコンは、故障したときなどに直すのが大変でしょう。対して、部品同士がシンプルな端子でつながれているパソコンであれば、簡単に直したり改造したりすることができます。それと同じことと言えます。

:: ファイルの読み書き

SDカードやUSBメモリ、ハードディスクといった記憶媒体というのは、一つの大きなデータのかたまりを保存できるにすぎません。なのに私たちは、いくつものファイルを記憶媒体の中に別々に保存することができます。これは、記憶媒体の中では、一定の規則に基づいてデータが並べられ、ファイルという形でデータを保存できるようになっているからです。この規則のことを、「**ファイルシステム**」と言います。

ex. ファイルシステムの種類：　FAT（USBメモリ、SDカードなど）
NTFS（Windows環境でのハードディスク）
HFS+（Mac環境でのハードディスク）
UDF（DVD）などたくさん

この作品では、SDカードを使用するので、FATファイルシステムを扱うソフトウェアが必要になります。そこで、フリーで公開されている「FatFs」というFATを扱うソフトウェアを組み込むことにしました。これは「各種ファイルシステム」にあたります。これによって、SDカードの読み書きができるようになっています。

:: 画像データ

画像をSDカードから読み出して表示するとなれば、画像ファイルを扱うことが必須になります。今回メインで使う画像ファイルは、ビットマップ形式（.BMP）です。このデータ形式は、１ドットずつ順番に色のデータが書かれているだけなので、簡単に扱うことができます。

また、一般的な画像データ形式として、JPEG（JPG）というものが有名ですね。この形式は、データを圧縮（数学的に計算をして、表される内容を極力かえずに、データの表し方をよりサイズが少ない形にすること）してサイズが縮められているので、扱いは難しいですが、これにも対応します。

:: 液晶画面への表示

液晶画面は、**VRAM方式**、すなわち液晶制御デバイスを一つのSRAMのように扱い、そのデータ（VRAM）を書き換えると、それに対応している液晶のドットの色が変わる、という方式が使われています。例えば、画像ファイルを表示したいときであれば、画像ファイルの１ドットずつのデータを順次読み出していき、VRAMに書込む、という方法になります。

• 感想

僕は物理部での数年間で、最終的に「マルチメディア」を自由に扱うことを目標にしていましたが、このように、まだ完成とは言えませんが、その形ができてきているのでその点はうれしいです。なにせコンピュータ系は、プログラミングなどの「下積み」が大変なので、形ができるまでは大変でした。共同製作の人たちには感謝したいです。とはいってもまだまだやらなければいけないことがかなり残っているので、文化祭までには完璧なものを目指していきたいですね。

願わくはこの製作物で作った画像技術がロボットの目に載せられて「自動で障害物を判定して歩く」とかですか。やってくれたら凄く感動します。

ちなみに、僕の製作の中心はプログラミングだったのですが、できるだけプログラミングは家でやってきて活動中は退屈しないようにしました。これから物無でコンピュータ系に進んでプログラミングなんかを凝ったものを作りたいという人はそんな感じにするといいかもしれません。

• 液晶制御部

M3 梅津 直弥

ここでは、画像の表示を担っている液晶と、液晶を制御する回路について解説します。今回使った主な部品は３つあります。

:: 液晶

今回はLQ035Q5DR01という液晶を使いました。￥900と割安で、かつ制御基盤付きというリーズナブルなものです。液晶パネルが分厚く場所を食うのが玉に瑕ですが。

光の三原色である、Red･Green･Blueの三色をそれぞれ6bitの64段階で制御し、262,144色間で表現できます。縦240ドット×横320ドットの液晶で、おおよそ某N社の携帯ゲーム機程の画質です。うん、荒い。バックライトを光らせるための回路は付属の制御基板についているので、今回は深く考えずとも光らせることができますが、話によれば、何千ボルトもの高圧の交流の電流が流れていて、危なっかしくてうかつに触れません。

では液晶のピン配置を説明しておきましょう。

～デジタルの世界では、物事のすべてを0と1、有と無しで表現しています。～
～ここで表記されている、HとLも、その倫理の表し方の一つです。～

・電源周り

ＶＣＣ･･･「電池でいう+のようなもの」とのこと。
ＧＮＤ･･･「電池でいう－のようなもの」とのこと。
＋１２ＶＤＣ･･･バックライト用の電源。

・信号周り

ＣＬＫ･･･「クロック（ＣＬｏｃＫ）」のこと。この信号をもとに液晶を表示しています。きっとＰＣ関連で聞いたはずです。
ＨＳ・ＶＳ･･･上で説明したとおりです。ＣＬＫぐらい重要な信号だったりもします。
Ｒ・Ｇ・Ｂ0～5･･･色データ制御に使うピンで、二進数であらわします。

:: CPLD

まず、CPLDとは、**コンプレックス・プログラマブル・ロジック・デバイス**のことで、なんと、３センチ四方の小さなICの中に、74シリーズIC（中一が使っています）20個以上　にも相当する数のICが入ってしまうのです！しかも、コンピュータを用いて何度でも（実際には一万回までの保障）書き換えることができ、気まぐれで回路を変えようとしても、いちいちICを基盤から引っこ抜いたり、導線で配線しなくても、あっという間に回路を変えてしまうことができるのです。

しかも、高速、高出力と、取説に書いてあった「高性能」の名も伊達ではありません。ただ、表面実装パッケージで、非常にICの足が細く、変換基盤に半田付けしづらいという欠点もありました。今回使ったCPLDはXilinx社製のXC95144XLというものです。

もちろん、あれしろこれしろということを、ひとりでにやってくれるはずがありません。そう、プログラムしてあげなくてはいけないのです。プログラミングには、Xilinx社製のISEというソフトを用います。プログラムをCPLDに書き込むには、LPTポート（一昔前のプリンタについていた、今となってはその座をUSBに危ぶまれている、20世紀の遺物と化しつつあるものです）とJTAGという書き込み端子を、ダウンロードケーブルを通して接続して書き込みます。ダウンロードケーブルは買うと一万円もして、部費を圧迫するので、計￥1000ほどで自作しました。回路は、ネットで調べれば出てきます。

ただ、今のパソコンで、上記のLPTポートを持つものはもはや希少価値です。というわけで、近い将来、純正のUSBダウンロードケーブルを買いたいと思っています。

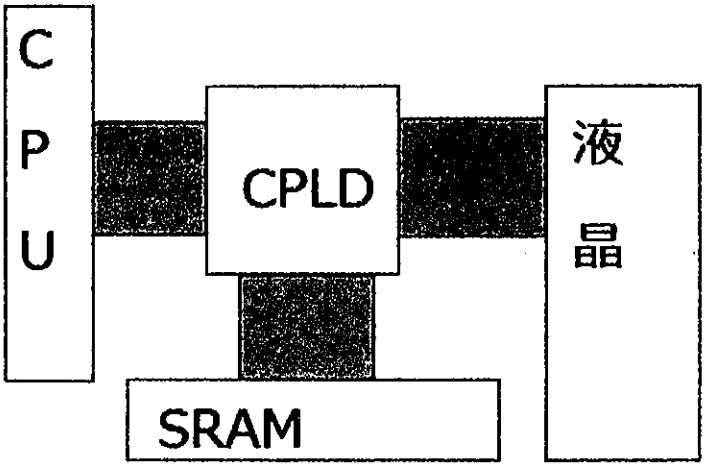
番号	記号	名称	備考
CN 1-1	GND	グラウンド	0V 供給
CN 1-2	CLK	動作クロック	Hにするたびに色データの読み込み/表示
CN 1-3	GND	グラウンド	0V 供給
CN 1-4	HS	水平同期信号	Hにするたび一行下に移る
CN 1-5	VS	垂直同期信号	Hにするたび一番上の行に移る
CN 1-6	GND	グラウンド	0V 供給
CN 1-7	R0	赤色表示信号データ（最下位）	赤色の明るさの調整
CN 1-8	R1	赤色表示信号データ	
CN 1-9	R2	赤色表示信号データ	
CN 1-10	R3	赤色表示信号データ	
CN 1-11	R4	赤色表示信号データ	
CN 1-12	R5	赤色表示信号データ（最上位）	
CN 1-13	GND	グラウンド	0V 供給
CN 1-14	G0	緑色表示信号データ（最下位）	緑色の明るさの調整
CN 1-15	G1	緑色表示信号データ	
CN 1-16	G2	緑色表示信号データ	
CN 1-17	G3	緑色表示信号データ	
CN 1-18	G4	緑色表示信号データ	
CN 1-19	G5	緑色表示信号データ（最上位）	
CN 1-20	VCC	3. 3V	3.3V 供給
CN 1-21	VCC	3. 3V	3.3V 供給
CN 1-22	B0	青色表示信号データ（最下位）	青色の明るさの調整
CN 1-23	B1	青色表示信号データ	
CN 1-24	B2	青色表示信号データ	
CN 1-25	B3	青色表示信号データ	
CN 1-26	B4	青色表示信号データ	
CN 1-27	B5	青色表示信号データ（最上位）	
CN 1-28	GND	グラウンド	0V 供給
CN 1-29	EN AB	初期位置連絡信号	一番上の行のときのみH
CN 1-30	GND	グラウンド	0V 供給
「CN2は液晶から制御基盤への信号入力のため未使用」とのこと。			
CN 3-1	GND	グラウンド	バックライト用
CN 3-2	NC	未接続	
CN 3-3	+12VDC	12V 直流電源	

↑液晶のピン配置

:: SRAM

今回は、秋葉原の秋月という電子パーツショップで売っていた、「高速ＳＲＡＭ　４Ｍビット　ＣＹ７Ｃ１０４１ＤＶ３３（１０ｎｓ）」というものを使いました。動作も速く、大容量です。ＣＰＬＤと接続し、ＣＰＵから送られてきたデータを一時的に保存し、ＣＰＬＤが読み込み、液晶に表示する、という寸法です。ちなみに揮発性のメモリで、電源を切るとあっという間に中にあったデータ全てがぶっ飛びます。

最後に、今回の製作における、僕の担当したCPLD周辺のブロック図を載せておきます。



• 感想

はじめは皆と同じようにＰＩＣマイコンを使っていましたが、こうしてＣＰＬＤを使うことになり、ちょっとプログラミング技術に引け目を感じますが、ＣＰＬＤにはマイコンには無い優れたところもあれば、逆に、ＣＰＬＤにはできないこともあるということも、使っていてなんとなくわかりました。というわけで、次は、物無で最近注目を浴びつつある、super-HやAVRといったマイコンも取り入れて行きたいと思います。

・カメラ制御部（CMOSイメージセンサー）

M3 川田 希望

CMOSイメージセンサーとは、フォトダイオード(光を当てることによって電流や電圧を発生させる部品)を使い、物を撮影できるようにした部品です。似たようなものに、CCDイメージセンサーなどがあります。

:: 今回使用した部品

・CMOSイメージセンサー
画像を取り込むためのものです。色のデータをRGB形式で出力してくれるものを探しました。

・SRAM
CMOSセンサーから出力された色のデータを一時的に保存しておくものです。これが無いとスピードが間に合わなくなります。これはピンの間隔が狭く、そのままでは使えないため、配線が面倒です。

・PIC16F886
これはCMOSセンサーとのI2C通信のために使用しました。

・CPLD
これはCMOSセンサーから出たデータをSRAMに送るために使用しました。このICは回路図を書く感覚でプログラムができるので便利です。しかし、これもピンの間隔が狭いので、変換基盤を使用します。

:: CMOSセンサー

今回画像を取り込む際に使用したものは、「TCM8240」という部品です。これはネット上の部品の通販サイトで見つけたものです。この部品は対応する変換基盤がなく、配線が面倒ですが、I2C通信で制御でき、RGB形式でも出力してくれるので便利です。

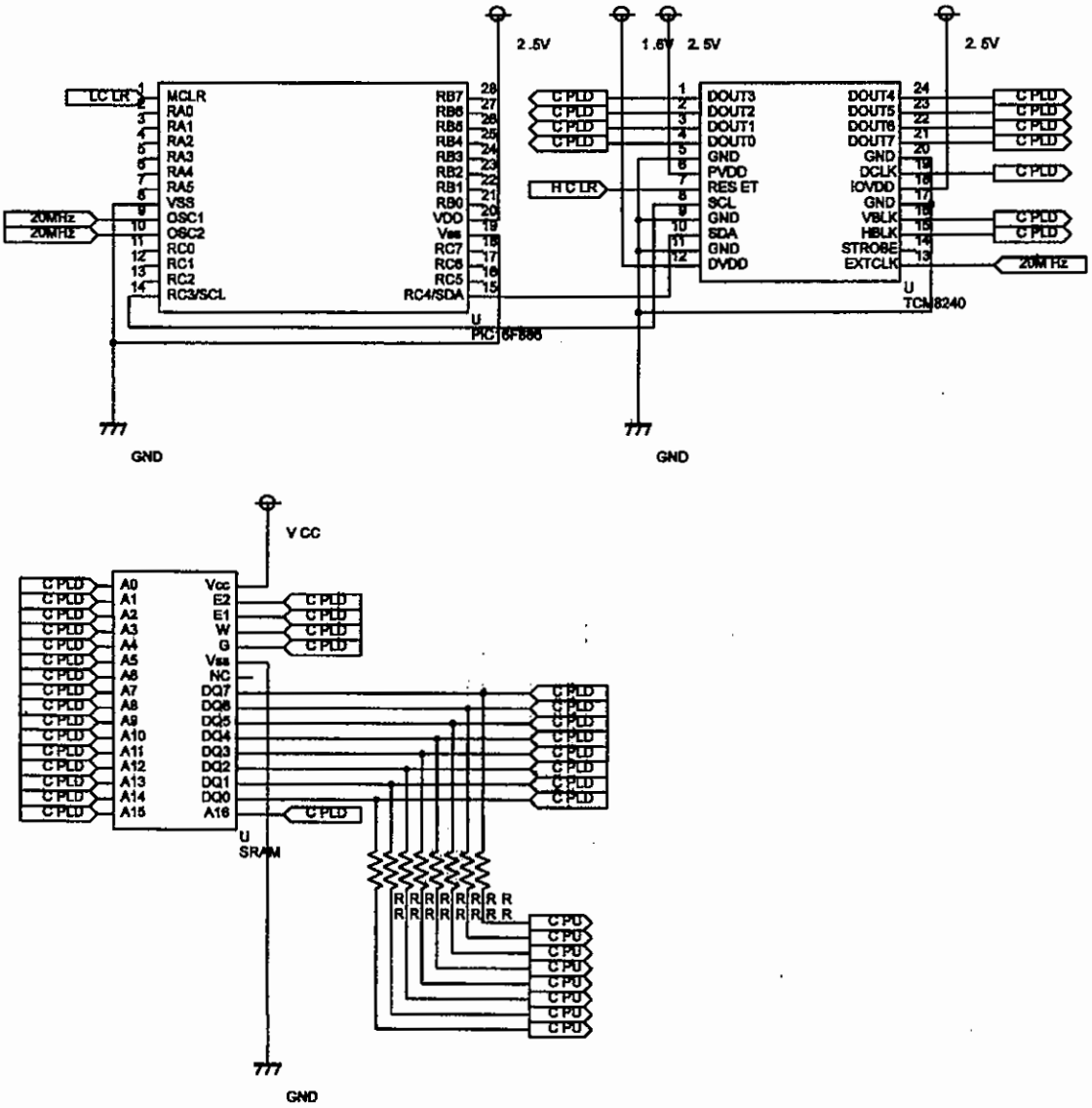
・使い方
SCL
I2C通信のCLOCKに値するものです。PICのSCLとつなぎます。

SDA
I2C通信でデータを送る際に使用します。PICのSDAとつなぎます。

DOUT0~7
色データが出力されます。I2C通信によってモードが変えられます。

EXTCLK
外部発振です。今回は20MHzにしました。

∷ 回路図



• 感想

今回の製作物は初めてプログラムを組んだりして作る製作物でしたが、それにしては難しすぎることをやっていたという気がします。使用したTCM8240も情報がほとんどなく、事前に調べておけば…と思うこともしばしばありました。また、細かい配線が多く、配線にも時間がかかってしまいました。しかし、共同制作者さんに回路などの構成を聞いたりしたことによって、今でも少しずつですが前進しています。このことから、共同制作者さんとのコミュニケーションが大事だということを学びました。
来年は今年学んだことや反省を生かして、計画的にやっていきたいです。

BS-YMZ

制作：M3　　朝倉

協力：物理部無線班の皆様

・概要

パソコンからのMIDI信号を読み込み、YMZ294から音を出そう、と言う物です。具体的には、PIC16Fシリーズに MIDIの受信をさせ、そこから別のPICにデータを送り、そのPICがYMZを制御する、という具合です。

・通信

この制作物の目玉はパソコンからのMIDIの読み込みと、発音数の増加に対応するためのマイコン間の通信です。なので、今回は二種類の通信を行います。

・パソコンと、音源ユニットの間のMIDI通信・・・USART通信

・音源ユニット内のマイコン同士の通信・・・I2C通信

それではそれぞれ説明していきます。

・USART通信

この通信方式は、信号線の数が少なく、また、中距離の通信に適していますが、高速での通信には適していません。この通信方式は、MIDIに於いては、機器間の通信の規格として採用されています。なので、パソコンからのMIDI信号はUSART通信で送られて来ます。

さて、PICには幸いなことにデフォルトでUSARTモジュールが搭載されています。これを使うことによって簡単にMIDIデータを読み込むことが出来るのです。具体的にUSARTは以下のことが出来ます。

・全二重通信

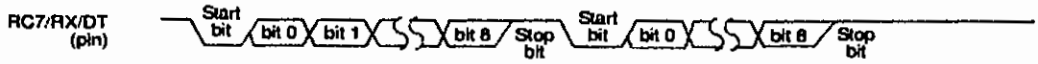
・同期or非同期通信

・送受信完了時などでの割り込み

・etc…

今回は、一方向、非同期で通信を行います。

仕組みは、スタートビットを検出すると、あらかじめ送信側と設定していた周期で、8回RXピンを監視して、そのときのRXピンの状態を読み込む、というものです。次ページの図のbit0～bit8(本当はbit7まで)のそれぞれの状態を順番に保存していきます。



・I2C 通信

この通信方式は比較的近距離(同じ基板上)で、二つ以上の機器間を二つの信号線によって高速(100kbps~1Mbps)に通信するものです。また、マスター(主)とスレーブ(従)があり、基本的にマスターが常に権限を持っています。そして、複数の機器の中から通信相手を指定するためにスレーブ側はそれぞれ個別のアドレスを持っています。

今回は 1Mbps で通信します。

さて、仕組みの方はあまり USART 通信と変わりはありません。ただし違いがあるのは、読み込み時のタイミングです。USART の時は通信する前からお互いの機器が、タイミングを打 ち合わせていましたが、I2C 通信では、マスター側からのタイミングでデータを読み込んでいきます。そのため、USART 通信よりタイミングの誤差が少なく、高速で通信できますが、二つの 信号線を用いるためそれぞれの通信線の誤差のため、長距離での通信には向いていないのです。

ちなみに I2C の「2」は実際には、二乗と同じように書かれています。

・通信規格

さて以上で説明した二つの通信方式を用いて通信を行うのですが、それはただのハードウェアの規格でしかありません。そこでソフトウェアの規格が必要になります。今回は下に挙げる二つの規格を用います。

・MIDI 規格(USART 通信)

・独自規格(I2C 通信)

それではそれぞれ解説していきます。

・MIDI 規格

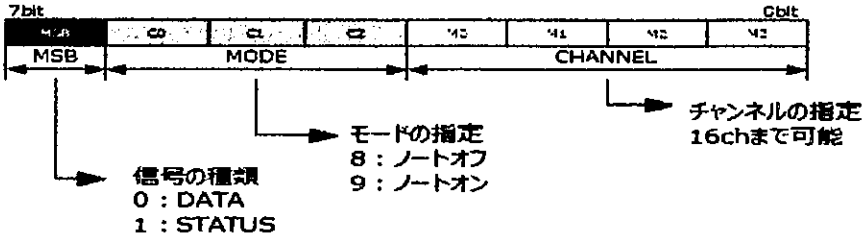
楽器メーカー各社がお互いの音源やキーボードを使えるようにすることを目的に策定された規格です。主に日本の ROLAND や YAMAHA が中心となって作ったそうです。

仕組みは簡単で、最上位ビット(MSB)でステータスバイトかデータバイトかを判断し、ステータスバイトを受信すると、その内容によってデータバイトの受信を待つだけです。
また今回は、ノートオンとノートオフのみしか使わないため、さらに単純になります。
ステータスバイトの 4~6bit の値が 8 ならばノートオフ、9 ならばノートオンとなる。そして下位 4bit は発音するチャンネルを指定します。そして発音または、消音するノートと、そのヴェロシティ(音量)の受信を待ちます。構造は次ページの図を参照してください。

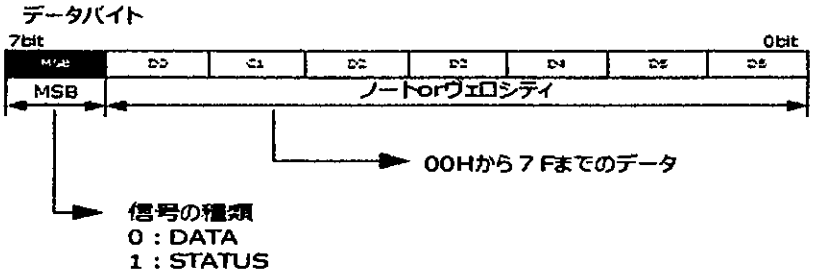
・独自規格

こちらも簡単な規格です。MIDI と同じような仕組みで、一バイト目に adr(アドレス)バイトを設け、動作を指定し、二バイト目のデータバイトで、その動作に必要なデータを送るだけです。
ただし、adr バイトの内容によっては、データバイトの長さが 16bit 長になる場合もあり、プログラムが複雑になります。

ステータスバイト

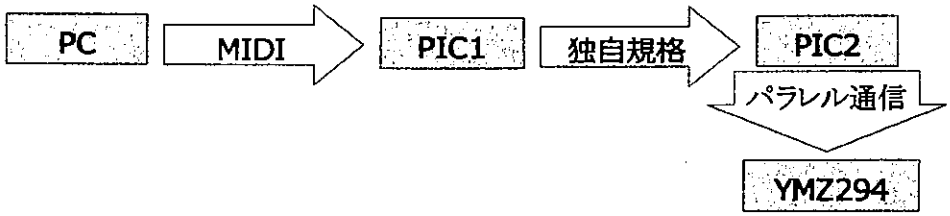


データバイト



・データフロー

これまでに説明したことを整理するために、データフローを示します。



・YMZ294の制御

大まかな流れとしては、次のようになっています。

・発音：発音する周波数を入力する　→　音量を設定する

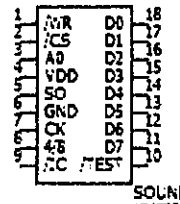
・消音：音量を0に設定する

今回は、発音消音以外行いません。

さてデータの入力の仕方は、

・データを入力する場所を指定　→　データを入力

といったものです。

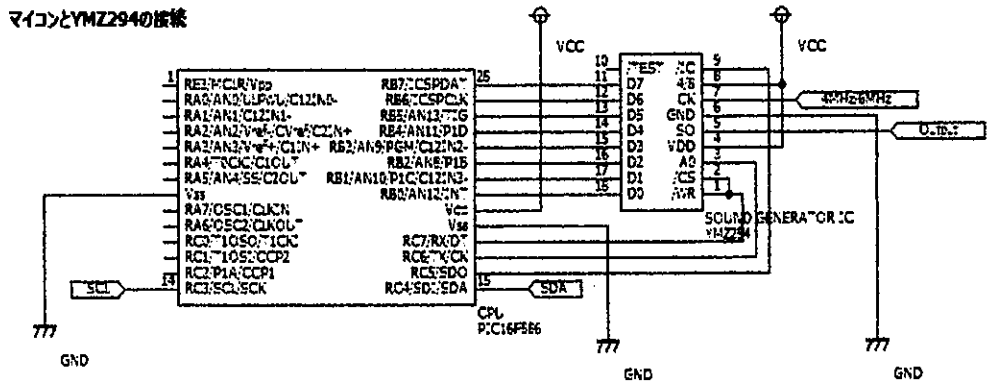


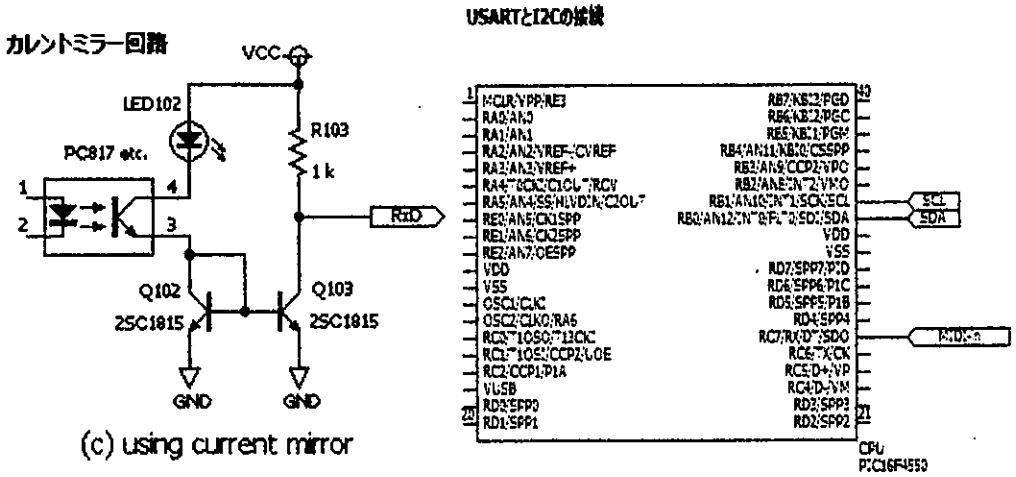
SOUND GENERATOR IC
YMZ294

具体的には、/WR,/CS,A0がL(0ボルト)の時は、D0~D7にはアドレスを入力し、/WR,/CSがL、A0がH(今回は5ボルト)ならデータを入力します。

・回路図

残りのスペースの関係上、重要な部分だけ載せておきます。大変申し訳ありませんが、この回路図は未完動ですので、これ通りに配線した場合動く保証は一切ございません。





・プログラム

大変申し訳ありませんが、現在、基盤を見直しによるパーツ交換や、プログラムの再構築のため、こちらに載せられるようなプログラムがございません。ご了承ください。

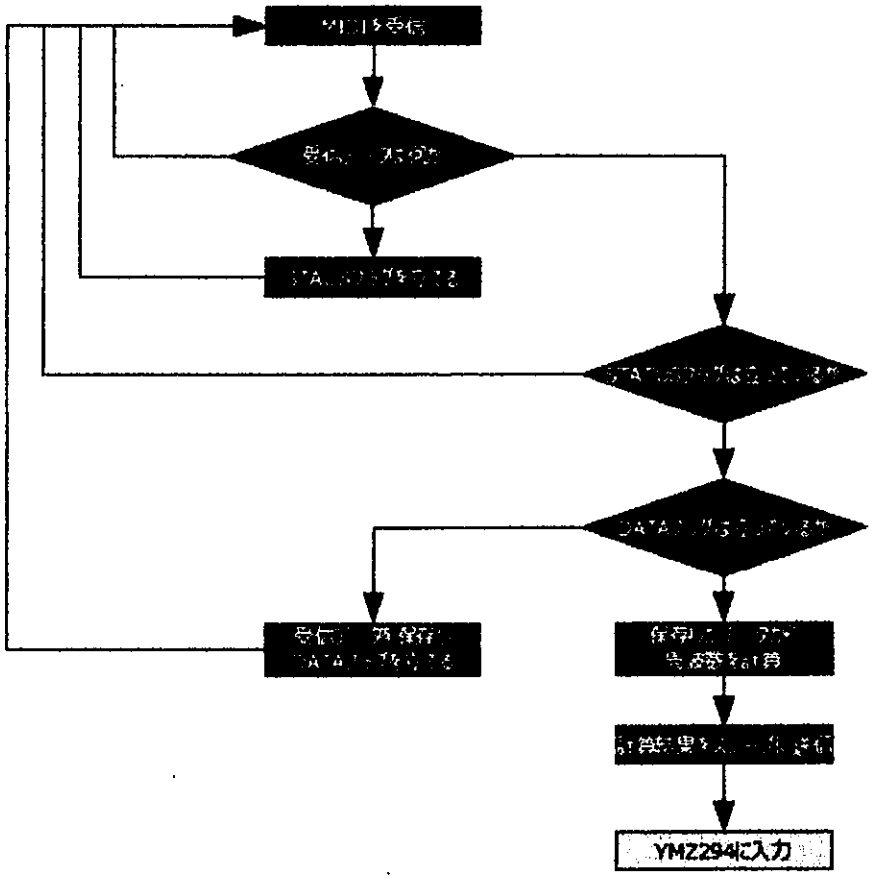
代りと言ってもなんですが、フローチャートを載せておきます。

・感想

今回は現段階で完動を見ることが出来ず大変残念です。MIDI通信以外にはあまり難しいところも無いと思って油断してかかったことが原因でよう。文化祭までの間、完動へ向けて努力は続けていきます。少なくとも、音楽を演奏するレベルにはしておきます。

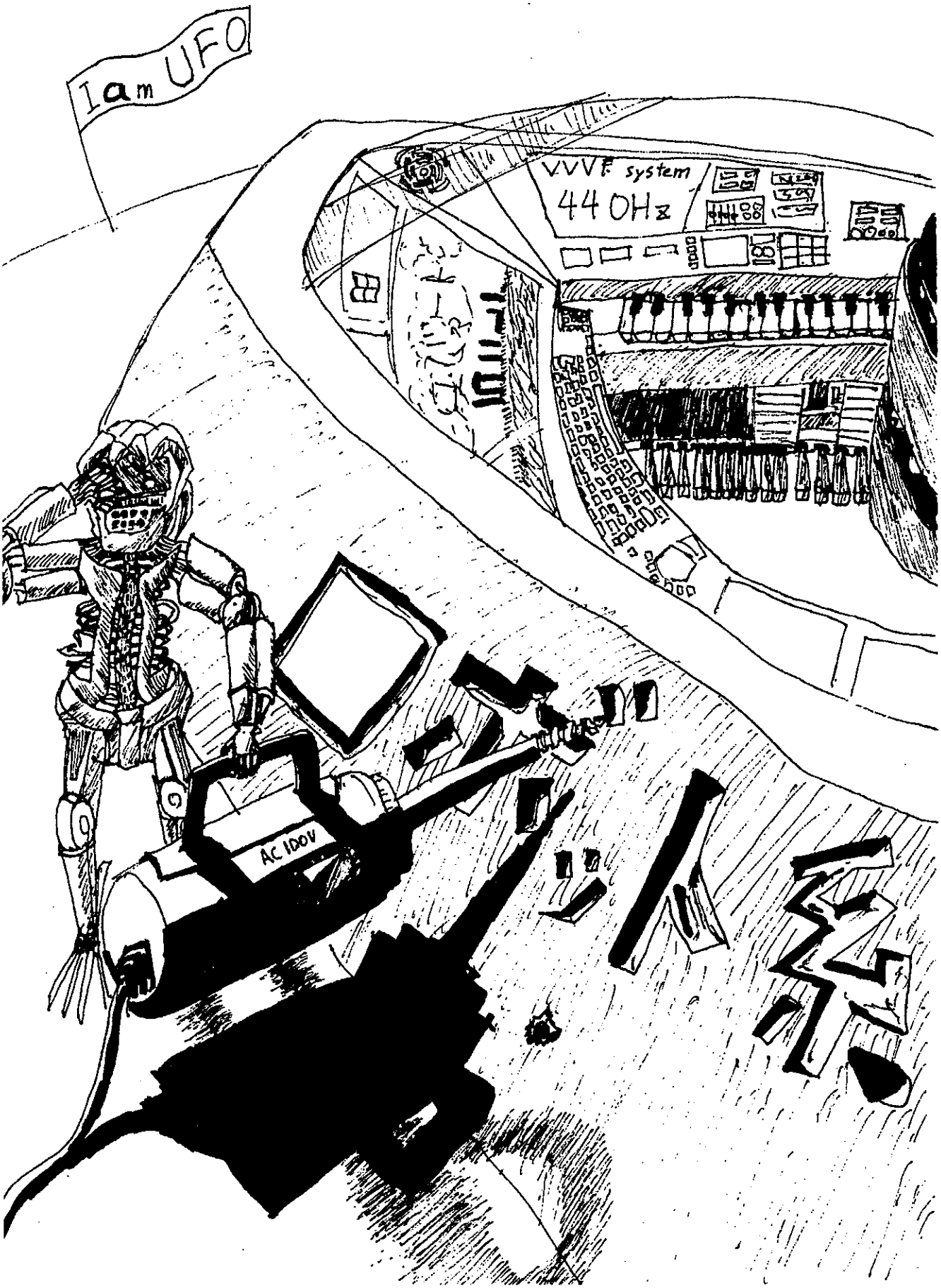
また、PICは不安定なので、次回からはH8を使っていこうと思います。出来ればUSB通信も出来るようにしたいです。

・フローチャート



・参考資料

- ・Wikipedia　http://ja.wikipedia.org/wiki/メインページ
- ・Microchip　http://www.microchip.com/
  - ・PIC18F2455/2550/4455/4550 Data Sheet
  - ・PIC16F882/883/884/886/887 Data Sheet
- ・YAMAHA　YMZ294 データシート(秋月電子通商で頒布されているもの)
- ・汎用 4 ピンフォトカプラを使った MIDI 入力
  http://www.geocities.jp/pcm1723/html/p4pcmidi.htm


I am UFO
VVVF system
440Hz
AC 1000V

MAWTIC ~Mobile Automatic Wireless Transeformable Intergent Chariot~

可動性無人無線換装知能二駆動車

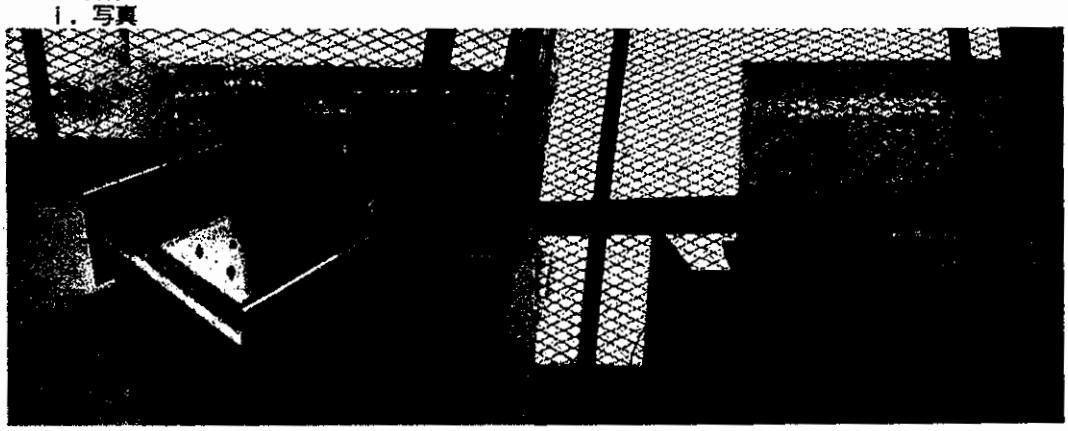
製作者 H-2：竹下 諒
M-3：栗本 郁也
協力者 物無の皆様

Ⅰ．概要

タイトルのとおり、全自走式のラジコン戦車です。
コンセプトとしては、サーボ・ステッピング・DCの三種類のモーターをドライブすることで戦車らしい動きをするものです。さらに距離センサを搭載することで周囲に障害物がないか常に見張りながら行動します。これを応用して全自動走行も行えます。
また、左右のモーターに回転計を取り付けることで、ハードウェア的な故障が生じた際に瞬間的にソフトウェアで動作を停止および液晶画面に問題を表示することでメンテナンスの簡易化を図っています。堅牢・丈夫でありながら高機能な基盤を積み、次年度以降のロボット製作にに関して良き資料（資材）になりうることを目標としています。また、制御につかうICは基本PICで二つのPICを組み合わせて使う。

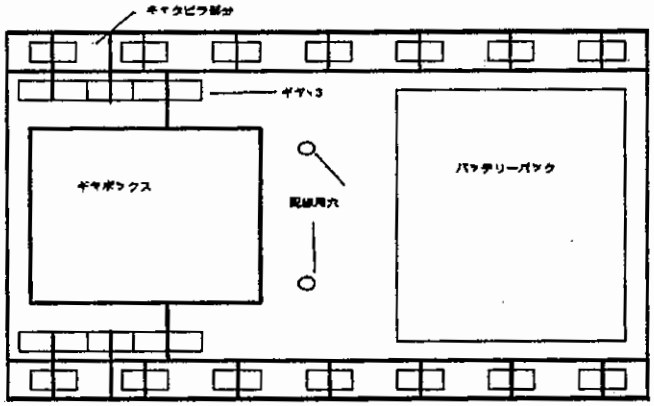
Ⅱ．外観

ⅰ．写真



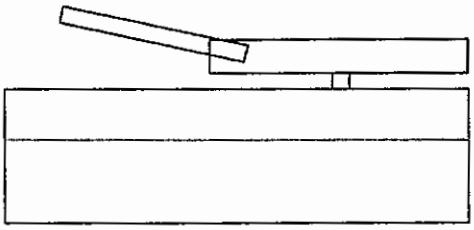
ⅱ．構造図

裏面から。ギヤボックスは同じモーターを背中合わせにしているため左右でシャフトの高さが合いませんそこで動力用シャフトとの間に一段小さいギヤを挟んでいます。また、キャタピラは外れにくいよう本体の内側にすべて収めてあります。



横から

キャタピラはほとんど見えません。一般的な装甲車の形状をしています。装甲の中ほどの位置に基盤がとりつけられておりこの部分はアクリル製です。ただ、強度に不安がありましたので、四辺をすべてアルミアングルで囲ってあります。

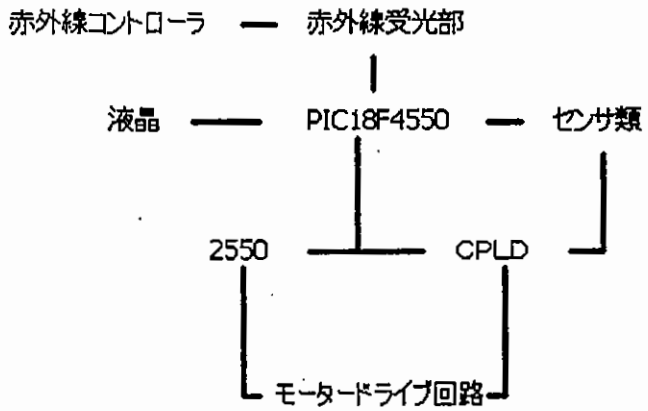


本体の強度は僕（６０Ｋｇ）が乗っても壊れません。戦車ですので。

Ⅲ．回路図

ｉ．ブロック図

わかりやすくまずブロック図にしておきます。

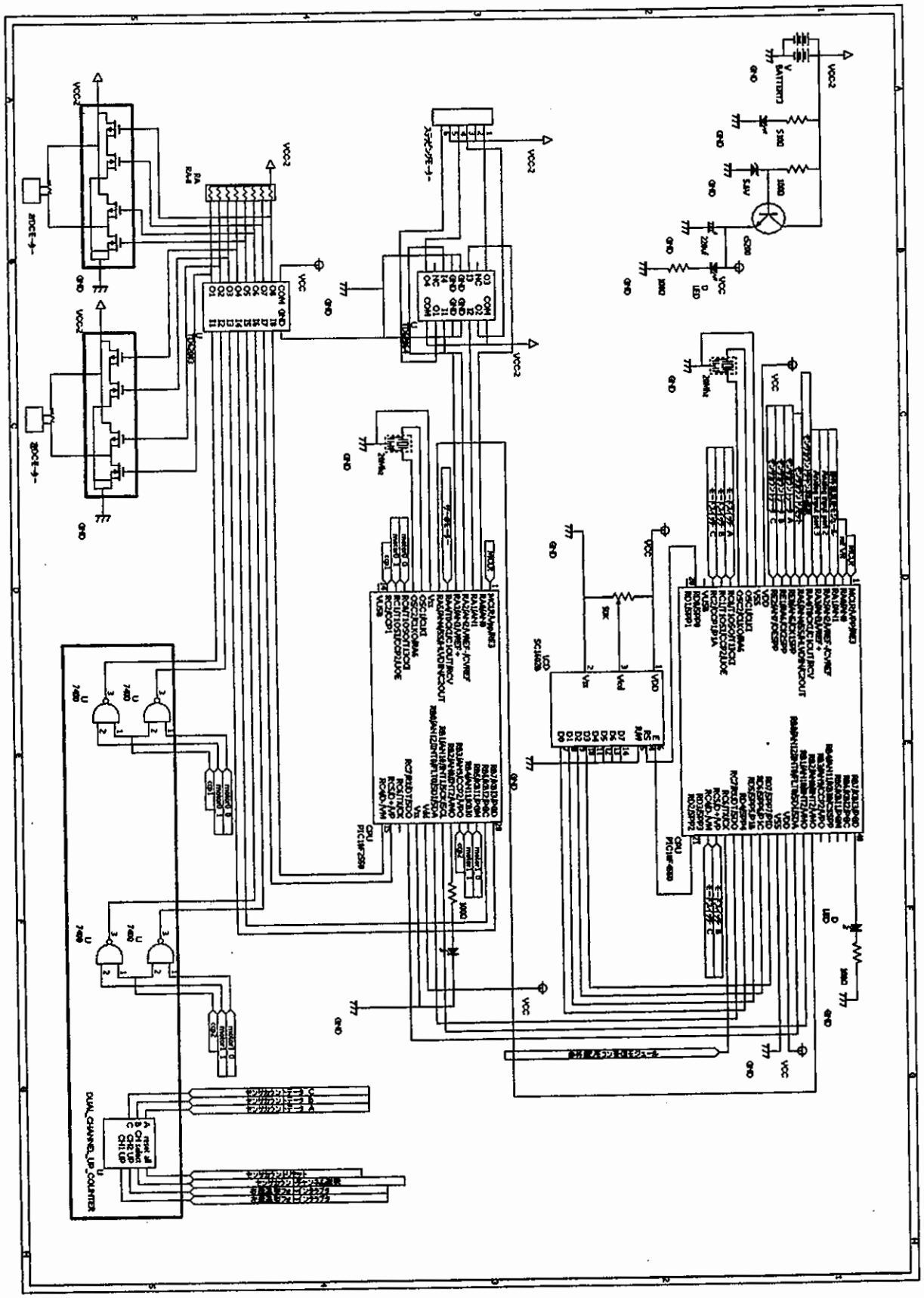


ii．動作説明

コントローラー基盤から赤外線でデータを送信し、本体の赤外線受光モジュールから PIC18F4550 がデータを読み取ります。4550 は他のセンサーからもデータを読み取り、液晶にさまざまなデータを表示します。さらに PIC18F2550 に spi 通信によってデータを送信します。2550 は３種のモーターをドライブします。CPLD はそれぞれの中継点として使われます。

iii．本体回路図

でかいので、次のページ一枚丸ごとです。

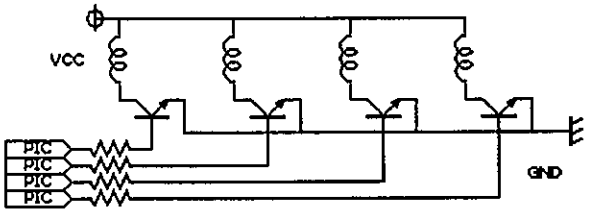


Ⅳ．電源

今回の製作物は見た目を良くするために給電線を設けず、バッテリーを搭載することにしました。バッテリーは自宅にあったラジコン用のNiCdを流用することにしました。このバッテリーが9.6Vであるためにの動作電圧である5Vに変圧してやる必要がありました。そこで最初に提案されたのが三端子レギュレーターを使う方法でしたが、500ｍＡほどの出力の素子が一般的で若干不安がありました。そこで今回採用したのが回路図左上のような回路です。これは、ツェナーダイオードの両端に電圧をためトランジスタを通して電圧を得る仕組みです。トランジスタのVbeは約0.6Vなのでツェナーダイオードは5.6Vの物にします。また2SC5200の企画は表の通りです。ここからVceが約4.6Vほどの今回の回路では15A流れても問題ないことになります。またhfeが160であることからベースには約100mA流せばよいことになるので抵抗は100Ωもあれば十分ということになります。しかしあまり限界近くで使うのもなんとなく怖いですし現実的に10Aもバッテリーが流せないので330Ωとしました。電源の平滑化のためにコンデンサを二つ並列にしてパスコンにしてあり、LEDは動作確認用に使っています。またトランジスタの発熱を抑えるためファンであおるという工夫もしました。

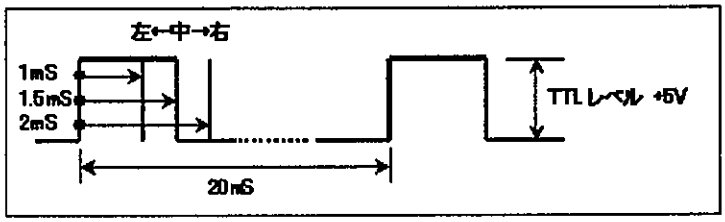
VCBO	230V	PC	150W
VCEO	230V	TJ	150℃
VEBO	5V	hfe	80～160
IC	15A	ft	30MhZ
IB	1.5A		

Ⅴ．ステッピングモーター



ステッピングモーターは、四本のコイルに順番に電流を流していくことで、回転数と速度を調整しながらまわすことができるというものです。砲台の水平方向の回転に使っています。内部の構造は鈴木と市村のページを参照してください。制御回路は上の図のとおりでTD62064にはこれが丸々内蔵されているため使いました。実際には二つずつXY→Y X'→X'Y'→Y'Xの順に通電させていきます。

Ⅵ．サーボモーター



サーボモーターはVCC・GND・Vinの三本から構成されておりVinの信号に対応してモーターの軸を回転・維持してくれます。制御信号は図のようになっています。ただこの数値はモー

ターの型番によってかなり違うので実験して調べなければなりません。この20mSごとの周期Timer1で作りました。また、pcwhにはdelay関数があり細かな波形がすぐに生成できますので非常に簡易に信号を作り出すことができます。

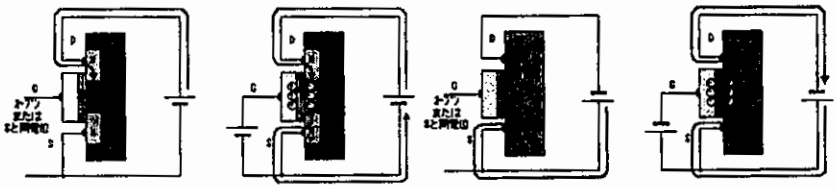
Ⅶ. DC motor

キャタピラを動かす直流モーターについての説明です。

下記のようなPchとNchのMOS-FETを組み合わせたHブリッジ回路でドライブしました。今回はMP4212(詳しくは部品解説にて)というFETアレイ(FETが複数内蔵されたもの)を使っています。

ⅰ.MOS-FET(Metal Oxide Semiconductor Field Effect Transistor)

MOSとは金属(Metal)、酸化絶縁膜(Oxide)、半導体(Semiconductor)を順番に重ね合わせた構造をしています。半導体としてはNPN構造のものとPNP構造のものがあり、NPNを使ったものをNチャネル、PNPを使ったものをPチャネルと呼んでいます。NPNまたはPNPの半導体に酸化膜を付け、その上にゲート電極として金属を配置します。NPNの場合、"N"の部分をソース電極およびドレイン電極にします。PNPの場合、"P"の部分が電極です。



↑Nch MOS-FET　　↑Pch MOS-FET

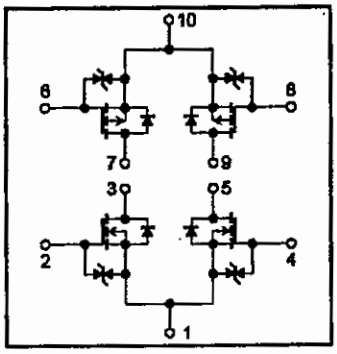
NチャネルMOS FETのゲートにプラス電圧がかかると、ソース電極およびドレイン電極のNチャネル半導体の電子がゲートに引き寄せられ、両者間のPチャネル半導体に入ります。この電子によりドレインーソース間に橋が架けられたようになり、ドレインーソース間に電流が流れるようになります。

PチャネルMOS FETの場合も電圧は逆ですが、同様な動作をします。ゲートにマイナス電圧がかかると、ソースおよびドレイン内のホールがゲートに引き寄せられ、両者間のNチャネル半導体に入ります。このホールによりドレインーソース間に橋が架けられ、ドレインーソース間に電流が流れるようになります。ゲートと半導体の間には絶縁酸化膜があるので、ゲートには電流は流れません。ゲートに加える電圧だけでドレーンーソース間の電流を制御できます。

ⅱ.Hブリッジ

等価回路

下の図がMP4212の等価回路です。一般的なHブリッジ回路を内蔵しています。モータードライブの際に必要なダイオードによる保護回路も含まれていたので外部に保護回路は設けませんでした。上の2つがPch,下の2つがNchのMOS-FETです。MOS-FETの項目の説明どおりPchはGate(6,8)に電圧をかけるとD-S間(10-7,10-9)に電流が流れなくなり、NchはGate(2,4)に電圧をかけるとD-S間(3-1,5-1)に電流が流れるようになりますね。



下の表のようにON・OFFを切り替えればモーターに電流を流せるはずです。

動作/PIN	2	4	6	8
正転	ON	OFF	OFF	ON
逆転	OFF	ON	ON	OFF
ストップ	OFF	OFF	OFF	OFF
ブレーキ	OFF	OFF	ON	ON

ⅲ．PWM

ＰＷＭ方式は、結果的には駆動電圧を変えているのと同じ効果を出しているのですが、その方法がパルス幅に依っているので、パルス幅変調法PWM(Pulse Width Modulation)と呼ばれています。具体的には、モータ駆動電源を一定周期でOn/Offするパルス状とし、そのパルス　のデューティ（On時間とOff時間の比）を変えることで実現しています。これは、Ｄｃモータが早い周波数の変化には、機械反応をしないことを利用しています。このパルス状の電圧でＤＣモータを駆動したときの、平均電力、電圧を考えれば、見かけ上、駆動電圧が変化していることになります。

Ⅷ．液晶

今回の戦車ではモーターの出力変化・マイコン間の通信を行ったため、デバッグの　利便性から考えてデーターを表示する装置が必要でした。そこでわずか７ピンで制　御でき、安価かつインタネット上に数多くの製作例が紹介されているLCD液晶を　使うことにしました。制御は後閑哲也氏が製作されたlcd_lib.cを使用しました。　後はPCWH内臓の関数で完成です。

Ⅵ．spi通信

spi通信とはモノトローラ社が提唱した通信フォーマットで多くのマイコンに内臓されています。類似の通信方式としてはsci通信などがあげられます。詳しい信号は図のとおりです。基本的にはマスタ側が生成するクロックパルスに合わせて信号線にデータを載せていくものです。またデバイスごとにイネーブルピンを設定することができるため、複数のマイコンを接続することもできます。データのやりとり を目的として作られているためノイズに弱く、最大で15cmほどの距離しか通信することができません。これよりも長い距離を通信するのでればusartやusb通信にする必要があります。スレーブ側からデータを送信する際はあらかじめスレーブの送信レジスタにデータをセットしておき、それをマスタ側からクロックが生成されたときに送り出すという動きをします。また、pcwhにはspi通信を駆動するための関数が内臓されています。以下にその使用用途を書き記して置きます。

関数名	引数	
void setup_spi()	spi_master , spi_slave	：デバイスの属性設定。
	spi_L_to_H , spi_H_to_L	：クロックのエッジ設定
	spi_clk_div_4 , spi_clk_div_16 , spi_clk_div_64	：クロック周期設定
	spi_ss_disabled	：デバイスイネーブルピンの使用有無

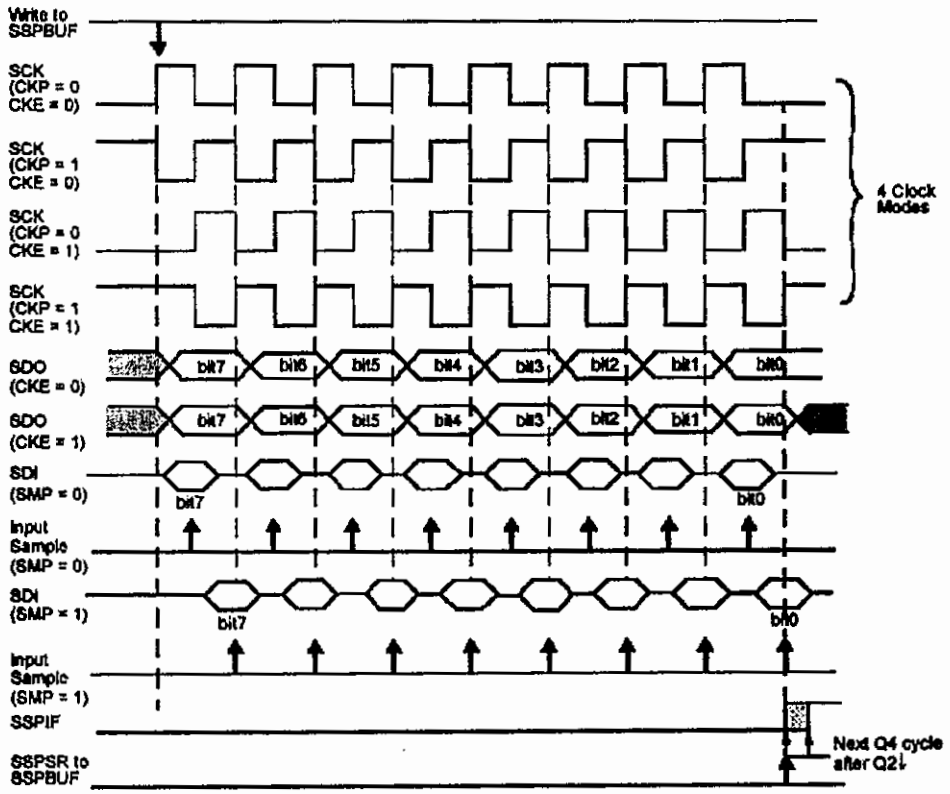
使用例：setup_spi(spi_master | spi_L_to_h | spi_clk_div_16 | spi_ss_disable)

void spi_write()	int型	：送信バッファ書き込み。マスタ側であれば通信開始

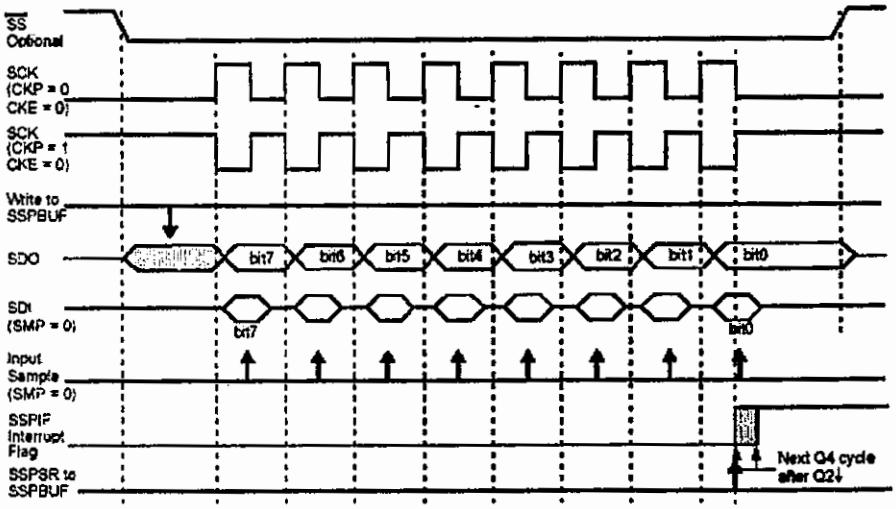
int spi_read()　　int 型　：送信ブッファ書き込み。また受信バッファを読み出す。

具体的な波形は以下のとおりです。

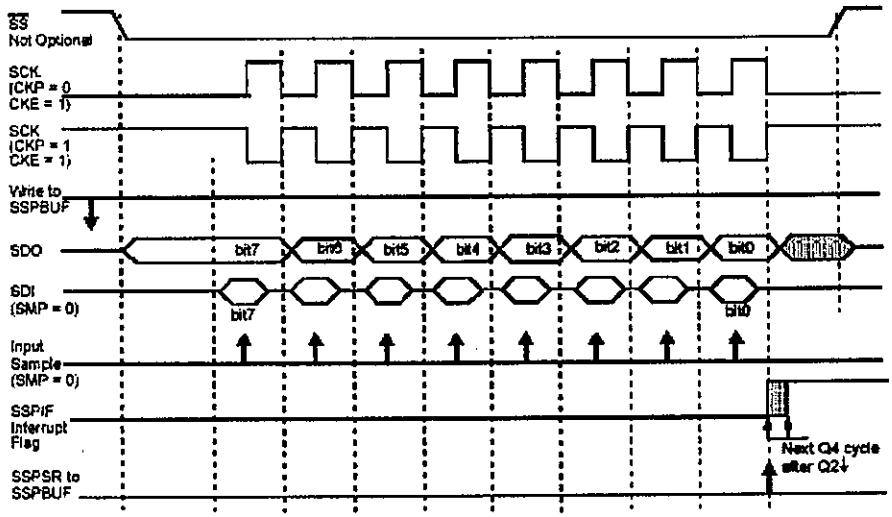
マスタ側の場合



スレーブ側エッジ立ち上がりエッジの場合



スレーブ側立下りエッジの場合



Ⅶ．赤外線リモコンについて

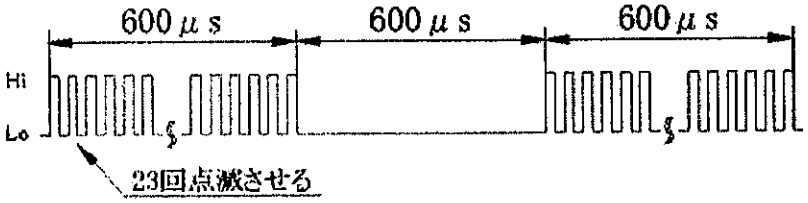
ｉ．概要

赤外線による無線通信でデータを送信します。

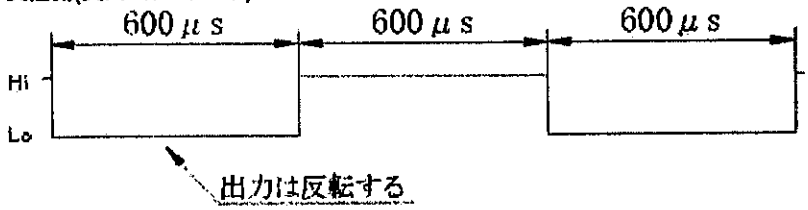
赤外線 LED と赤外線リモコン受光モジュール(PL-IRM0101)をそれぞれ PIC で制 御します。

今回はキャリア周波数 38kHz、出力パルス幅 600us のものを使います。詳しい仕組みについては新田・橋本の page を参照してください。

送信側(赤外線 LED)



受信側(受光モジュール)



ii．データ受信処理について

まずスタートビットを待ちます。何もしていない時はずっとこれを待ち続けています。スタートビットを発見したら、300μsec 待ってから再度読み込み確かにスタートビットであることを再確認します。これが間違いなくスタートビットだったら後は 600μsec 毎にデータを読み込み、 デバイスコード、

セパレータの確認、キーコードそしてストップビットシーケンスの確認と順に実行し受信したデータをメモリに格納します。引き続き２回目のデータを同様にして受信し、２つの受信データを照合します。同じであったら正常受信として次に進み、異なっていたら、何もせず最初に戻ります。正常データの受信完了により、デバイスコードがあらかじめ設定されているデータ と同じかを確認します。同じで無かったらやはり何もせず最初に戻ります。デバイスコードが一致したら自分当てのキーコードと判断してそれぞれのキーコードに従った処理を行います。実行終了後は、最初に戻って次のデータ受信を待ちます。

Ⅷ．CPLD

今回の製作でまず問題となったのはpwm制御です。PICには多くてもccpモ　ジュールが二つしかなく。PICのピンから直接繋いでしまうとPICが２個に増えて　しまう。かといってTTLICで外部回路を組むのは基盤のスペース的に無理がある。そこで、面倒な回路をすべてCPLDにやってもらおうと思い立ったわけです。また、PICには外部割込み（int割り込み）がひとつしかなく二つのセンサから回数を数えることもなかなか骨が折れます。そこで手っ取り早く、一段カウンタを挟み一定　時間ごとに読み込むこととしました。CPLDに関する詳しいことは中３梅澤のページを参照してください。

Ⅸ．4550プログラム

以下のとおりです。

```
#include"siruf.h"//いろいろインクルード
#include<stdint.h>
void main(){setup();while(1){com.all=IR_read();}}      //赤外線通信のデータを記録し続ける
void setup();//変数初期化・ポート設定・adコンバーターセット・spi通信セット・タイマセット
uint8_t ad_read(uint8_t ch);//引数に合わせてチャンネル読み込み
#INT_TIMER1//タイマ処理部
void isr();//時間計測・各身ジュールコントロール・センサデーター表示など。
void device_data_reset();//変数初期化
void spi_send(uint8_t ch,uint8_t value);//spi通信用関数
void cont_all();//各モジュールコントロール
void spi_send_all();//マイコン同士の完全同期用関数
```

Ⅹ．2550プログラム

```
//device"PIC18F2550"  step,servo,dcをspi通信で制御する。//setting
#include <18f2550.h>  #include <stdlib.h>
#fuses HS,NOLVP,NOWDT,PUT,NOPROTECT,NOBROWNOUT,NOCPD,NOWRT,MCLR,NODEBUG
#use fast_io(A)  #use fast_io(B)  #use fast_io(C)  #use fast_io(E)    #use delay(CLOCK = 20000000)
#include "original setting.h"  #include "basis.c"  #include "dc_motor-2.c"  #include "stepping-1.c"  #include
```

```
"servo_motor-1.c"
#include "spi_slave.c"  #include "something extra.c"  #byte p_lo = 0xF80
//main
void main (){
 setup_2550(224,3,192,8);          //"port"setup
 start_pic(3,300);                 //"pirotlamp"
 setup_module();                   //"ssp,ccp,timer_1,timer_2"setupt
 setup_servo();                    //"servo" setup
 servo_reset();                    //"servo" reset
 dc_reset();                       //"dc motor" reset
 step_reset();                     //"step" reset
 start_pic(5,50);                  //done
 while(1){delay_ms(1);}            //idle loop
}
//SPI受信完了割り込み
#INT_SSP
void spi_start(){s_in.all = spi_read();
 switch(s_in.data.ad){
  case 0 :control_servo(s_in.data.mode);step_start(s_in.data.duty);break;      //step_servo
  case 1 :dc_setup(right,s_in.data.mode);break;      //dc_r
  case 2 :dc_setup(left,s_in.data.mode);break;       //dc_l
  default:break;
 }
```

ＸＩ．モジュール

ｉ 赤外線測距モジュール　「GP2Y0A21YK」

秋月電子で売っているシャープ製の測距モジュールです。タイプは三種類ありそれぞれ測定できる距離が違います。どのモジュールも電源電圧は5Vで出力信号はアナログデータで出力されます。安価で、制御が簡単なので手軽に使えます。

ii フォト・IC出力方フォトマイクロセンサ(透過型)「EE-SX460-P1」

秋月電子で売っているとても安価で、制御が簡単な透過センサですオープンコレクタ出力なので複数個を並列して使うこともできます。プルアップは10kオーム程度で、そうすればTTLICに直接でも見ることができます。

iii トランジスタアレイ　「TD62083APG」

8chで1chあたり500mA流せるトランジスタアレイです。安いです。入力抵抗があらかじめついていますのでPICなどを直接つなぐことができます。

iv トランジスタアレイ　「TD62064APG」

4chで1chあたり1500mA流せるトランジスタアレイです。安いです。入力抵抗があらかじめつい

ていますのでPICなどを直接つなぐことができます。また出力が大きいの
である程度の大きさまでのステッピングモーターなら単体で動かすことができます。

v FETアレイ　「MP4212」

MOS-FETアレイです。中にPch×2 Nch×2が入ってます。また、回路保護用のダイオードも内臓されていますのでよっぽどのことがない限り死にません。千石電商で販売されています。かなりの大きさのDCモーターも駆動することができます。ただし、メーカーは生産の中止を宣言しています。

XⅡ．感想

竹下：うん死に際だがまあいいだろう

栗本：来年は機動兵器ですね

参考

http://www.google.co.jp/　万能

http://www.ys-labo.com/index.htm　spi

Drive my motor　2010年卒　竹島さん執筆

http://www.sxlist.com/images/www/hobby_elec/menu.htm　ステッピングモーター

http://www.picfun.com/　PIC関連

http://akizukidenshi.com/catalog/default.aspx　部品屋　データシートが豊富

くノー　～Kunoichi～

製作者

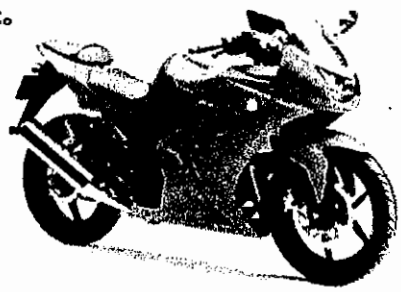
H2　棚辺教授

M3　天才森田

※　この製作物説明には厨Ⅱ病だと思われる程の表現が含まれます。

物語

私の名前は棚辺。彼はその後輩森田。二人は稀に見る普通の少年だった。
彼らはとある小さな学校の生徒。とても平和な日常をごく普通に生活していた。
事件は突然に起こるものだ。隣町のときめき学園がこの学校を征服しようと攻めて来たのである。
「彼は、たしか茂木流星。学園の番長直々にお出ましですかい」
「あちゃぁ。しかもニンジャって言うんですよね？あの男が乗ってる奴（絵A参照）」
「らしいな。あの黄緑のボディ。間違いない。」
「お手上げ～」
「お手上げにゃぁまだ早い。目には目を、忍には忍を。ってね。」
「・・・！」
――――は目覚めた。
何かが倒れる大きな音。こちらへと近づく低音質の波と共に、目の前に現れた鉄塊。
「ハハッ。お前も、落ちぶれたな棚辺さんよぉ。お、森田もいたのか。」
「・・・むかつく野郎だ。」
――――は動き出した。
「うれしいだろう？貴様等は、このニンジャの餌食になれるのさ。」
ニンジャは激しく、低く、大きく鳴いた。その流線型は私たちに向かって突っ込んできた。
そのとき、その流線型に向かって一筋の線がぶつかった。
「これが・・・俺たちの最終兵器」
くノーは倒れなかった。



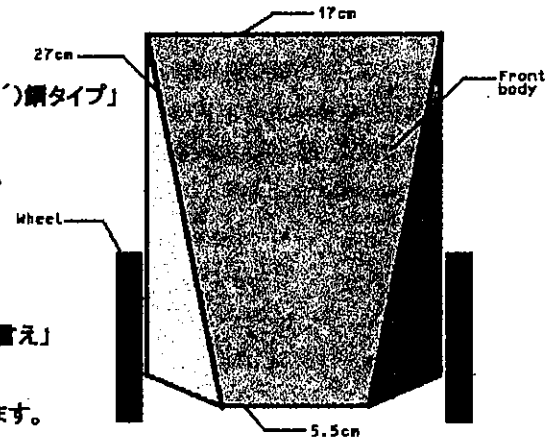
絵A

概要

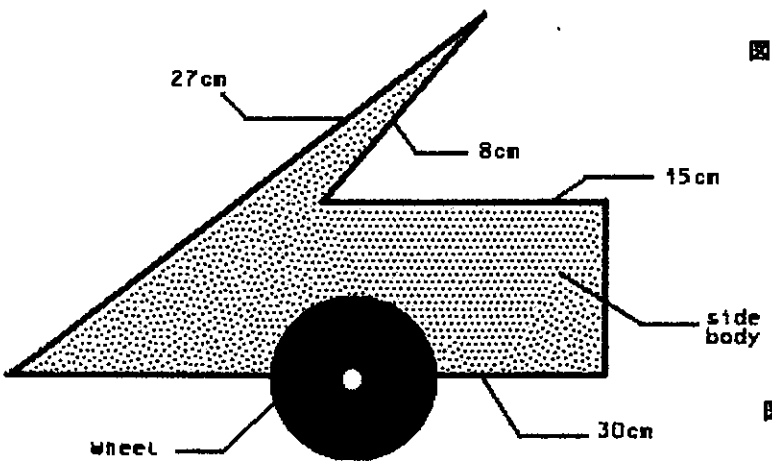
簡単に言ってしまえば、模型2輪車である。
本体の左右にタイヤが付いていて、前後に動きバランスを取りながら前に進む車輪付きロボットである。プレイステーションコントローラーが繋がれており、操作も可能となっている。いわば2輪ラジコンである。

外観

これが「くノー」の外観図。
タイヤは二つ、形は「シャキーン(`・ω・´)調タイプ」な感じになっています。矢印をイメージした、さわやかな作りになっております。
重心はちゃんと車輪の真上に来るよう設計されてますよ～
森田「さすがさわやか組です」
福辺「目の下のクマをどうにかしてから言え」
図Aは正面からのアングル
図Bは側面(左)のアングルとなっています。



図A



図B

機構

☆ジャイロセンサー

ジャイロセンサーとは、回転角加速度を測る事のできるセンサーである。このロボットに積まれているものは村田のENC－03Rであり、かの有名な「むらたくんが行くよ(・ω・`)三」で有名な「むらたくん」に積まれているものと同じである。

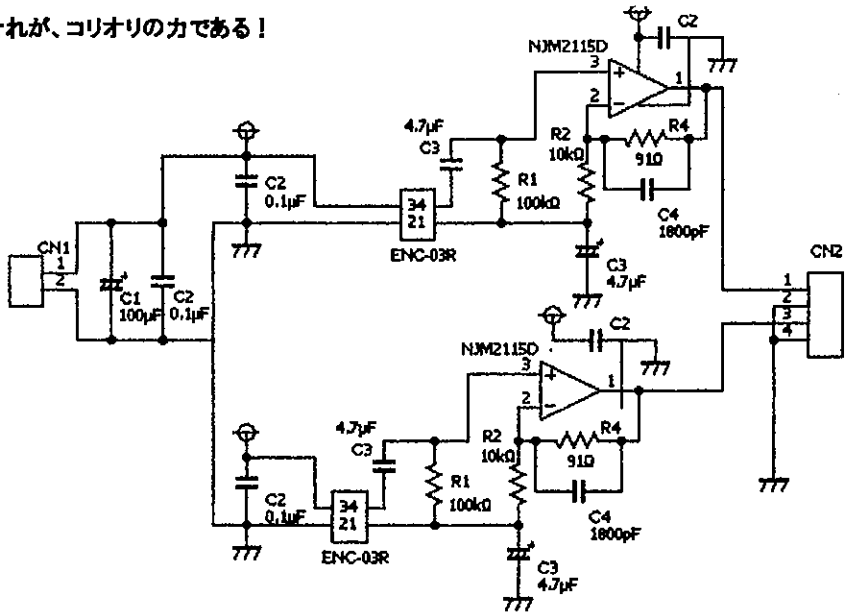
森田「どうやって回転角加速度を測っているのでしょうかね。」

その仕組みは、センサ素子に入っているセラミックバイモルフ振動子にコリオリの力のような力が与えられる事により、センサから出す電流(静止時1.35v、5～0vまで変異)を変化させるのである。

森田「コリオリの力ってなんですか？」

コリオリの力とは、地球が自転することによって、地球表面上に自転と逆向きに起こる力である。まぁ、例えば、森田君と茂木君が右回りに動いている中華料理屋さんとかにあるテーブルの上でキャッチボールをしたとすると、茂木君が投げた球は森田君から見て、左向きに曲がって見えるのである。その力が地球規模で起きている。

それが、コリオリの力である！



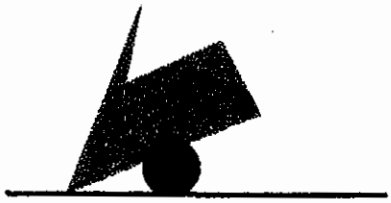
これが、ジャイロセンサーの回路図です。

CN1の1，2はVCCとGNDとなっています。

CN2は、2，4はGND。1，3は、縦、横に対応した出力となっています。

動作

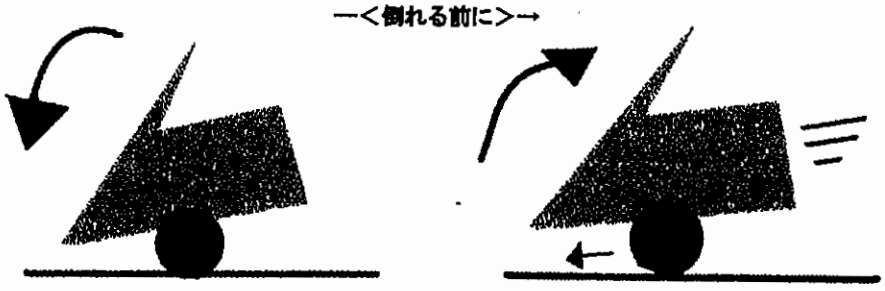
くノーはどうやって動くのか。それについて触れていこう。
森田「コイツはバランスをとりながら進まなければいけないですねー」
棚辺「そうだな。右の絵のようになってしまうな」
森田「こうなった事をジャイロで感知するんですー」
棚辺「ほうほう、そうしたらどうするんだ？」
森田「タイヤを前に回転させるんですよー」



ガタン

棚辺「そうすると、次の絵のように尻餅をつくね」
森田「そうすね。これをうまい事調整すれば」
棚辺「そうだな。とりあえず立つことができるね」
森田「そんでもってですよ。上手い具合に前のめりにさせながら前進を続ければ、前に進む事が出来ますよ」
棚辺「よくわかっているじゃないか！まるでセ○ウェイのように走るね」
森田「モーターはDCモーターを使って、モーターの制御はPWMを使えば大丈夫です」

―＜倒れる前に＞→

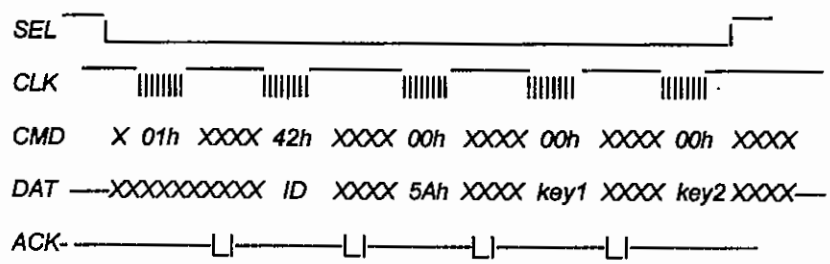


←＜スピードちょい落とす＞―

ペイントで絵を描いている私。絵で本当にすいませんf(^^;)
さて、コントローラー担当の森田君にバトンタッチしましょう

プレステコントローラー & PWM　　M3 森田晃平

プレステコントローラーとはSPIで通信をしています。
コントローラーはSELからの信号を感知した後、本体からのデータが0x01ならば応答IDを出力する
応答IDはSCPH-1010(一番最初のコントローラー)なら'A'
その後'Z'を出力し、その後データを出力する
・全体図



・キーデータ

バイト	コマンド	b7	b6	b5	b4	b3	b2	b1	b0
1バイト目	0x01		----						不定
2バイト目	0x42		0x41						'A'
3バイト目	0x00		0x5a						'Z'
4バイト目	0x00	左	下	右	上	ｽﾀｰﾄ	1	1	ｾﾚｸﾄ
5バイト目	0x00	□	×	○	△	R1	L1	R2	L2

標準電圧は3.6V
転送バイト数は(応答IDの下位4bit×2＋3)バイト
キーコードは押されたボタンが0となります
CLKのたち下がりでデータの出力
CLKの立ち上がりでデータの入力
たぶん周波数は1MHZまでは動作するはず
CMDとDATはプルアップした方がいい

僕の場合

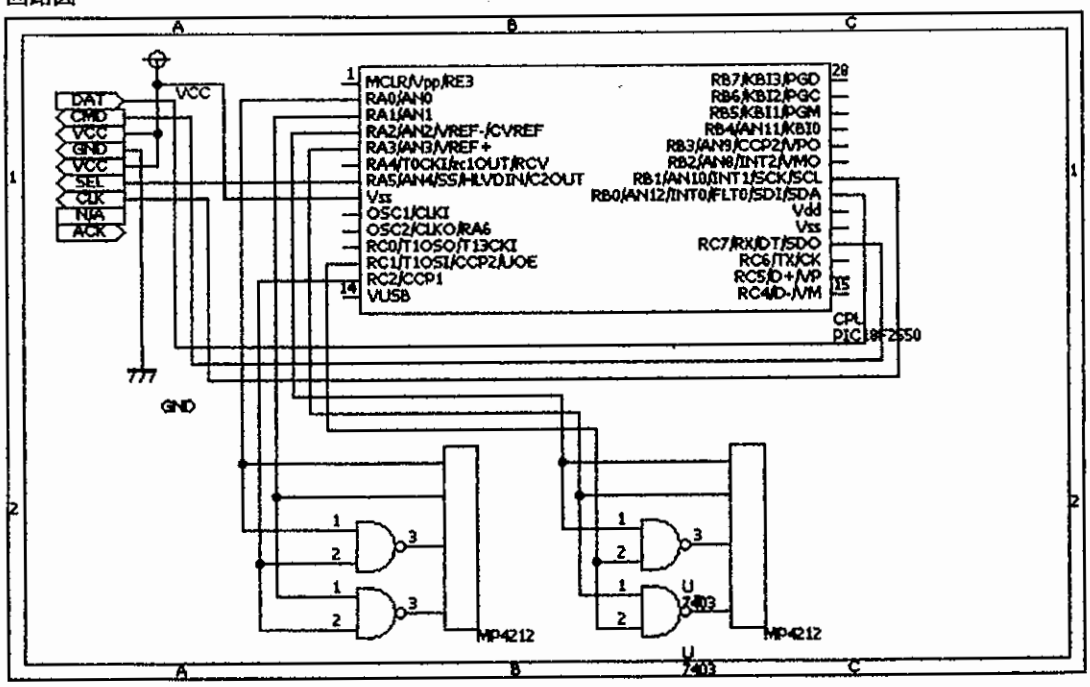
SPICON1 = 0x31　SSPSTAT = 0
PIC の SPI とプレステコントローラーでは送られてくるデータの順番が逆なので PIC で反転
DAT は 10KΩ でプルアップ
CMD は 1KΩ でプルアップ
CLK は 500KHZ
通信の間隔は 0.2MS
としています

対応コントローラー

基本は初代の PS コントローラー用ですが、一応すべて通信できるはず
僕の使っているコントローラーなら 5V でも動作しましたが、すべてするとは限りません

そして、PWM は PIC の PWM モードで動かしています
PWM と二本の制御ピンで一つのモーターを制御しています
コントローラーから得たデータで PWM を制御しています

回路図



参考文献

プレイステーション・ＰＡＤ／メモリ・インターフェースの解析
http://www.kaele.com/~kashima/games/ps_jpn.txt

森田君、ありがとうございましたー b^w^
非常にわかりやすい解説でした。フォントが違うのはご愛嬌ですm(。__。)m<ユルシテ～

感想

森田

今年の製作物では、SPI通信やモーター制御などのいろいろなことに触れることができて楽しかったです。しかし、モーターは電力とかの関係でバグがおきやすく、大変でした。
あと、共同制作者である棚辺さんとは、一緒に楽しく製作物を作ることができていると思います。

棚辺

この製作物の説明は以上です。いや一別れが惜しいですね。ここを読んだらいいことあるぞ！DCモーターの中身などは他の製作物などで解説をしていますので、きっと本誌を読破していただければ、モーターのことは分ることでしょう。この製作物が少しでも心に残ってくれる事を祈る。

自分は製作物のボディデザインにこだわりを持とうと様々なデザインを考えた結果、こんなデザインになってしまいましたが、個人的にはこの製作物の推せる点とすることができたのではないかと思います。森田君にはプログラムがんばってもらいましたね。感謝感謝。

センサー周りに重点を置いてここまで製作してきた私が、最後にあたったのがこのジャイロセンサーでしたね。ジャイロセンサーの制御、計算がイチバン難しかったですね(過去には超音波や赤外線などを扱いました！)ジャイロセンサーは、この部活にて初めて扱われました。これを今後も使っていってほしいものですね。もう只々それだけです。

終章　～エピローグ～

見事に茂木流星を倒す事ができた2人。また彼らの学校に、平和が訪れる。しかし、平和な世の中もまたそれで退屈なもので・・・
棚辺「いやー。何だかんだカタチになったな。疲れた疲れたf^^;」
森田「一時はどうなる事かと思いましたけどね。C言語つかれました」
棚辺「全くだわ。お互いがんばりましたよ。部品なくなったりしたもん」
森田「そうですね。あー。これからは暇ですね」
棚辺「そうだな。暇っていうのは幸せだな。いつまでもそんな幸せやってる時間は無いな」
森田「まだなんかやるんですかぁ(°Д)ﾎﾟｹｰ」
棚辺「そりゃそうさ。この空の青い限りは、ね」
森田「お手上げ～＼(´A`)/」
――――――――――――――いまにも落ちてきそうな空の下で――――――――――――――

Fin

BT-TE DESTROY THE FORTRESS

製作者 H1 鈴木　　M3 市村
協力　物無の皆様

PROLOGUE

西暦２０XX　ヨーロッパ地方にて大きな戦乱が起こった。
国力は拮抗し、戦争は膠着状態に陥った。
東部戦線のある部隊は敵の巨大砲台を攻略できずにいた。
ある日、司令部から前線にいくつかのコンテナと一通の命令書が届いた。
そこには、「技術研究所で、革新的な兵器　六足歩行戦車 BT-TE　が開発された。これを運用して敵砲台を攻略しろ。戦果を期待する。」とあった。
これは、新兵器を運用し、奮戦する兵士たちの物語である。

はじめに

これは中一によくある設定のゲームを実際に立体化したものです。
作った動機は今まで、ロボット同士が対戦する製作物があったら面白いなというのと、戦車とか砲台とか格好良いなという軽い気持ちですね。(中高生男子ならわかるはず、この気持ち)
PROLOGUE は鈴木の映画の見過ぎの産物です。華麗にスルーしてもらえると嬉しいです。
理想↓　　現実は後述

ルール

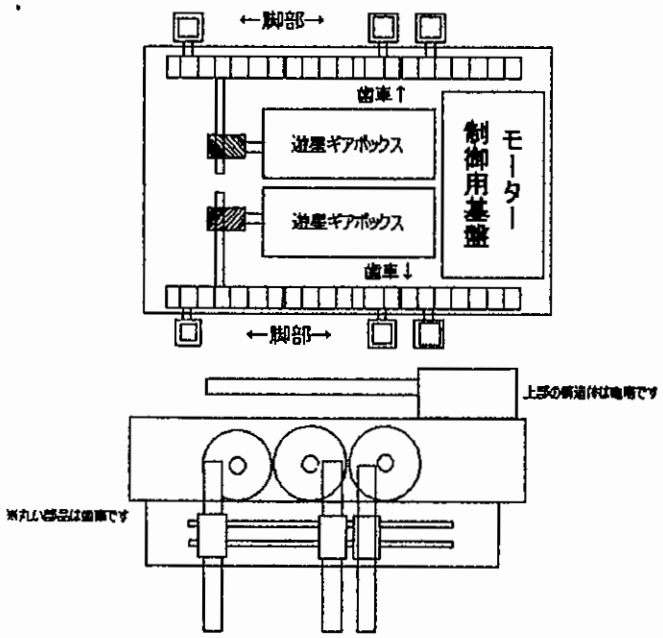
1人プレイか2人対戦ゲームかを選択できます。戦車側は砲台の上部の側面についてる弱点に回りこんで狙い撃ちましょう。砲台側は左右に旋回してちょこまかと動き回る戦車を狙いましょう。
※ 戦車は革新的な歩行方法を採用しているため、操作性が悪いかも知れません。

六足歩行戦車

この六足歩行戦車のコンセプトはズバリ「小型化」です。そのためにモーターの数を極限まで減らし、多足ロボットとしては異例の2個に抑えました※。仕様の詳細は「設計」の項で述べます。

※駆動系のみの数。砲塔制御にさらに1つ使用

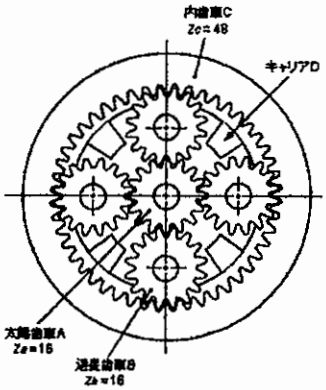
設計



上の図が戦車の大まかな図です。仕組みとしては、モーターで3枚のギアを回転させ、それぞれの端についた脚を前後上下移動させ、前進させます。
機体下部のレールは脚が常に下を向き続けるためのものです。
モーターは駆動系にはマブチ社製 DC モーターを使用し、砲塔の制御には秋月で販売している小型のステッピングモーターを使用しています。

モーター

ここでは、使用しているモーターについて解説するつもりだったが、ステッピングモーターは市村が後述しているためここでは遊星ギアについて説明します。
遊星ギアとは簡単に言うと、1つのギアから複数のギアに回転を伝達すると回転速度が遅くなり、力(トルク)が大きくなることを利用して、高トルクを得るためのギアのことです。下図参照



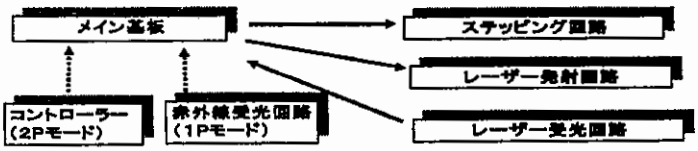
これを、何段も重ねたものを　遊星ギアボックス　といいます。
この機体では三段の物を組んで使用しています。これを使えば、貧弱で回転速度の速いマブチ社のモーター(元々はミニ四駆とかに使うそうで・・・)でも強いトルクとちょうどいい回転速度が得られます。

感想

今回は初めての共同制作だったが振り返ってみると、作っているロボットが2つということあって市村との共同作業も終盤に至るまでほとんど行わず、土壇場になっての仕様変更などが起こってしまった。また、自分の戦車の設計に想定以上に手間取り、3月30日現在砲台がほぼ完成しているにもかかわらずこっちはまだ、本体の組み上げ中という状況で「お前、なにやってるんだ」的な感じになってしまっています・・・。
まあ、よかった点としては遊星ギアの有用性に気づいたことですね。あれ以上に低コストで高トルクを実現できるギアを自分は他に知りません。24Vとかをかけると弱いマブチモーターを使ってるせいで煙を吹きながら回ったりしますが。モーターも定格電圧を守ろう!
最後にこれからロボットをやる物無員へ
これからは、もっと「小型」で「低コスト」なものを作ろう
大きいものもなかなかロマンがあるけど、「小型」も技術の結晶だから・・・
あと、回路図集用の設計図をビットマップで書くのは止めましょう。面倒で書きにくい。

巨大砲台

続いて本体の説明です。最初にブロック図を載せます

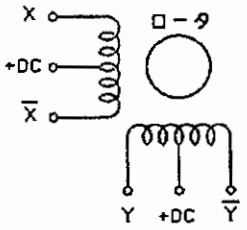


本体は１Ｐと２Ｐで、動く場所が違います（点線部など）

ステッピングモーター

１．ステッピングモーターとは

ユニポーラ型ステッピングモータ



ステッピングモーターは回す角度を調整できるモーターです

利点としては角度調整の他に、

- 強い静止トルクが得られる(止まっている時軸が回らない)
- デジタル回路向きである(パルスで回す)
- パルスを送る間隔を変えるだけで回転速度を変えられえる
- サーボモーターなどと違い何回転でもさせることができる
- こわれにくい

などがあり、とても使いやすく、物理部で最も簡単に制御できるモーターと言えます。

欠点としては

- 振動が大きい
- 回転時のトルクが小さい
- 回転速度に限界があり、送るパルスの速度が速いと脱調する。(脱調すると回らなくなる)

などがあります。重い物を回したり、ギアなどを使わずに高速回転させるのには不向きなモーターであるいえます。

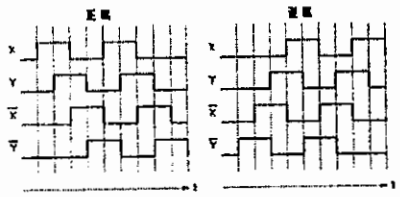
ステッピングモーターにはユニポーラ型とバイポーラ型がありますが、扱いやすいユニポーラ型を使用しました。

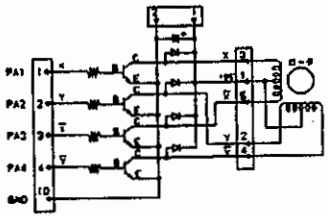
ステッピングモーターの中の回路は上の方の図の様になっていて、＋DC に VCC(+)を流し、X,'X,Y,'Y に特定の信号(＋やー)を送るとロータ(軸)を１サイクルまわすことができます。

モーターによって１サイクルの角度は決まっていて、たとえば１周 48 サイクルのステッピングモーターだったら１サイクルで 7.5 度回ります。

2．ステッピングモーターをまわすには

まわすための信号を送るための回路と、信号の波形は次のとおりです。





さきほどサイクルと呼んでいたのはこの信号のくりかえしのことです。

わかりづらい人のために表をかきます。

	STEP0	STEP1	STEP2	STEP3	STEP4	STEP1	・・・
X	L	L	H	H			・・・
Y	H	L	L	H	H		
X_BAR	H	H	L	L	H	H	
Y_BAR	H	H	H	L	L	H	・・・

Hというのは、ON,1、＋のことで、LというのはOFF,0,－のことです。

STEP0でモーターを静止させ、STEP1~4で回転させます。1>2>3>4>1>2>3>4>1・・・

先ほどの例でいう1サイクルは、1~4一回分です。

3．ステッピングモーターのプログラム

では最後にステッピングモーターのプログラムの一部を載せます。(正転のみ) 言語はアセンブラ、PICは16F819です。宣言、WAITなどは他の人参照。

(勿論ですが、これだけでは動きません様々な割り込み、分岐などをともないます。)

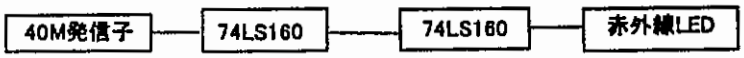
```
STEP0
        MOVLW   B'00000111'
        MOVWF   PORTA
        CALL    WAIT
        GOTO    STEP1
STEP1
        MOVLW   B'00000011'
        MOVWF   PORTA
        CALL    WAIT
        GOTO    STEP2
STEP2
        MOVLW   B'00001001'
        MOVWF   PORTA
```

```
                CALL    WAIT
                GOTO    STEP3
STEP3
                MOVLW   B'00001100'
                MOVWF   PORTA
                CALL    WAIT
                GOTO    STEP4
STEP4
                MOVLW   B'00000110'
                MOVWF   PORTA
                CALL    WAIT
                GOTO    STEP1
                END
```

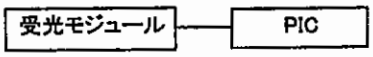
赤外線発信、受信

赤外線はデータの送信、受信に使う事ができますが、敵の位置がわかればいいだけなのでここでは使っていません。知りたい人は他の人を参照。

送信(子機に積みます)



受信(本体につみます)



レーザー

一応搭載しましたが実際には使用しない事になりました。

レーザー

正直この製作物は回路よりも物理的な壁(レーザー受光、コードが絡まる、ステッピングのはやさ)が問題になってくる製作物だとおもうのでそこを最後乗越えたいと思います。
絶対に文化祭までに完動させたいです。

SONOS~たこは生地を破る～

製作　回路担当　M３大森
　　本体担当　M３武子

０～ストーリー～

これは千葉に住む町工場勤務の男とある研究所の科学者兼アメリカ大統領による物語…
２０４０年　某月　某日。
その日浦安のあるテーマパークを中心としたＭ１１の超巨大地震が発生した。
後に「ディズニー大震災」と名づけられたこの地震は、世界各地に津波の被害をもたらし、日本の首都圏は回復不能な程のダメージを受けて浦安近辺の都市に生存者は皆無とされた。

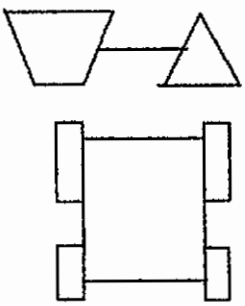
しかしそんな中でも生き延びている一人の男性がいた。
「くそ、体が動かない…」
その男性はそう言いつつ自分の体にのしかかる瓦礫をどかそうとした。
彼の名は武子。町工場の工場長をやっている男で、運良く小さい小屋で製品のテストをしており、瓦礫が少なかったので潰されなかったのだ。
「だめだ、全然動かない。大阪楽しみにしてたのにもう行けないのか…」
彼は明日から家族と大阪に行く予定だったのだ。
「もう死ぬのか。こんなことになるんだったら」
「大阪名物の串かつ、たこ焼き、うどん、お好み焼き、そばを食べておけばよかったー！」

「！…聞こえた…武子の最後の願いが」
そう言ったのは日本人初のアメリカ大統領ならびに物無研究所所長、大森であった。
彼は手元にある電話から物無研究所に電話をかけた。
「おい、今すぐ例のロボットと本場大阪の串かつ、たこ焼き、うどん、お好み焼き、そばを用意して千葉に送れ！」
「し、しかし所長、ご存知のように日本では超巨大地震が起こり千葉には生存者がいないとの事ですが…」
「いいから早くしろ！」
「わ、わかりました。すぐ手配します」
通話が終わったあと、彼は受話器を持ったまま呟いた。
「なに、心配するな武子。お前の作った車体と俺の組んだ回路ならどんな道でも進めるさ」
「串かつ、たこ焼き、うどん、お好み焼き、そば…Spit,Octopus ball, Noodle,Okonomiyaki,Soba。」
「千葉とアメリカに大阪の魂がこもった」
「SONOS ならな」

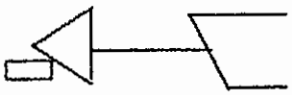
１～概要～

このロボットは動いている人を感知して、障害物を乗り越えそこまで行くことを目的としています。
人検知にはパナソニック社の NaPiOn（ナピオン）という焦電型赤外線センサーとアンプ（信号を増幅するもの。焦電型赤外線センサーはこれがないと信号が小さすぎるのです）が一緒になった物を使っています。焦電型赤外線センサーは人の体温を感知して、動くと初めて反応を示すというきわめて優秀なものです。

このロボットは前輪が正三角形、後輪が等脚台形になっています。（下図参照）
またその周りにキャタピラを巻いています。（キャタピラは単なるロマンです）



横から見た図と上から見た図です。
前輪にはステッピングモーター、後輪にはＤＣモーター（詳細は後の大森の項目に）というモーターを使っています。
これらのモーターにより、前輪を回し障害物に乗り上げて、後輪のキャタピラにより進むという形をとっています。



図がへたくそですいません。ペイントで書こうとしてひどい出来になりました。
本体については特に語ることもないのでこれくらいにして。

～回路担当に交代します～

１．　５～焦電型赤外線センサ～

実はまだ交代しません。僕も回路担当だった時代もあったので人検知に使われている「NaPiOn」についてもう少しくわしく語りたいと思います。正直僕もよくわからないので間違っていたらすいません。

まず初めに赤外線についてですが、皆さんが真っ先に思い浮かべるのはリモコンや携帯の赤外線通信などでしょう。これらはデジタルカメラなどを使えば見ることができます。

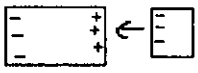
しかしそれらとはちがう種類の赤外線が、人間や他の物体から出ているのです。そしてこの赤外線は熱を帯びています。ここでは体温と同じと考えてもらってもかまいません。

赤外線センサは動作原理により、２種に大別することができます。１つは熱型赤外線センサで、もう１つは量子型赤外線センサに分けられます。

熱型赤外線センサは、感度、応答速度は低いのですが、波長帯域が広く常温で使用でき、使いやすいという特徴があります。量子型赤外線センサについては、検出感度が高く、応答速度が速いなどの特徴を持っています。

僕たちが使っているのは熱型赤外線センサの方で、仕組みとしては焦電体(微小な温度変化に応じて誘電分極する物体)が積んであり、それが人体から発されている赤外線による熱により焦電効果(微小な温度変化に応じて誘電分極する現象)を起こすことで判定するようになっています。

ちなみに誘電分極とは、絶縁体(電気を通しにくい物体)に＋か－の電気を帯びた物体を近づけると近づけた側に異種の電気が現れ反対側に同種の電気が現れることです。



簡単に言うとセンサの範囲内にある 20 度の空気から赤外線の影響を受けていて安定していた焦電体が、範囲内に 36 度の人間が入り今までとは違う種類の赤外線が来ることによる変化で見分けているのです。

このセンサの悪い点はどんな物でも出している赤外線を使っているので、人体と同じぐらいの体温を持つ小動物にも反応してしまいます。逆に良い点としては、自分は赤外線を出さず相手が出す赤外線を受けるだけなので広範囲を検知することができます。またお互い干渉しないので併設してつかう事もできます。

仕組みはこれくらいにして内部の構造に移りたいと思います。赤外線がどういうふうに信号になるかというと

レンズ→焦電体→アンプ回路→ウィンドウコンパレータ回路→出力

という感じです。

レンズはセンサの検知範囲拡大のためにつけています。そこから入った赤外線が焦電体に行き、出た信号をアンプ回路という物で増幅します。2回に分けて増幅されて、一回が 400 倍なのでここで信号が約 1600 倍になります。

そのあとのウィンドウコンパレータ回路で誤動作のチェックをしてようやく出力となります。

僕はこれらすべてを自作しようとして失敗してしまいました。ＮａＰｉＯｎすごすぎです。

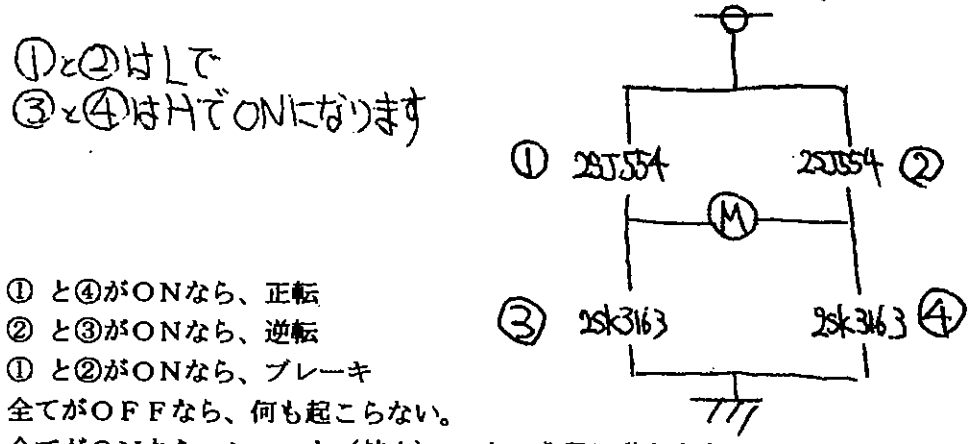
ここまで僕の文章を読んでもらって本当にありがとうございました。次こそ本当に大森君の出番です。

ここからは、回路担当の大森時生が書きます。

2、回路及びプログラム

＜後輪＞

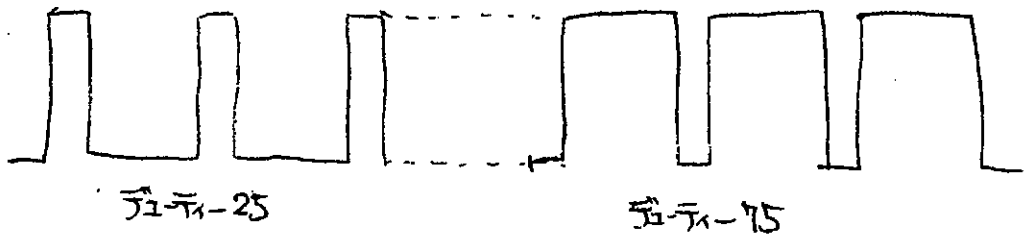
移動の基本となる後輪は、ＤＣモーターによって制御されています
ＤＣモーターとは電池につなげば回るっていうよく見るあれですよ。
みなさん、学校の授業で触ったことあるのではないでしょうか？
下の図が、ＤＣモーター制御の基本であるＨブリッジです。Ｈブリッジとは、
ＦＥＴを利用してただ正転するだけではなく、他の動作を行えます。



① と④がＯＮなら、正転
② と③がＯＮなら、逆転
① と②がＯＮなら、ブレーキ
全てがＯＦＦなら、何も起こらない。
全てがＯＮなら、ショート（禁止）　という風に動きます。
今年はＭＰ4212というＦＥＴアレイが流行っていたので僕も使いました。
これを使えば、ＤＣモーターの回路をつくるのが格段に楽になります。

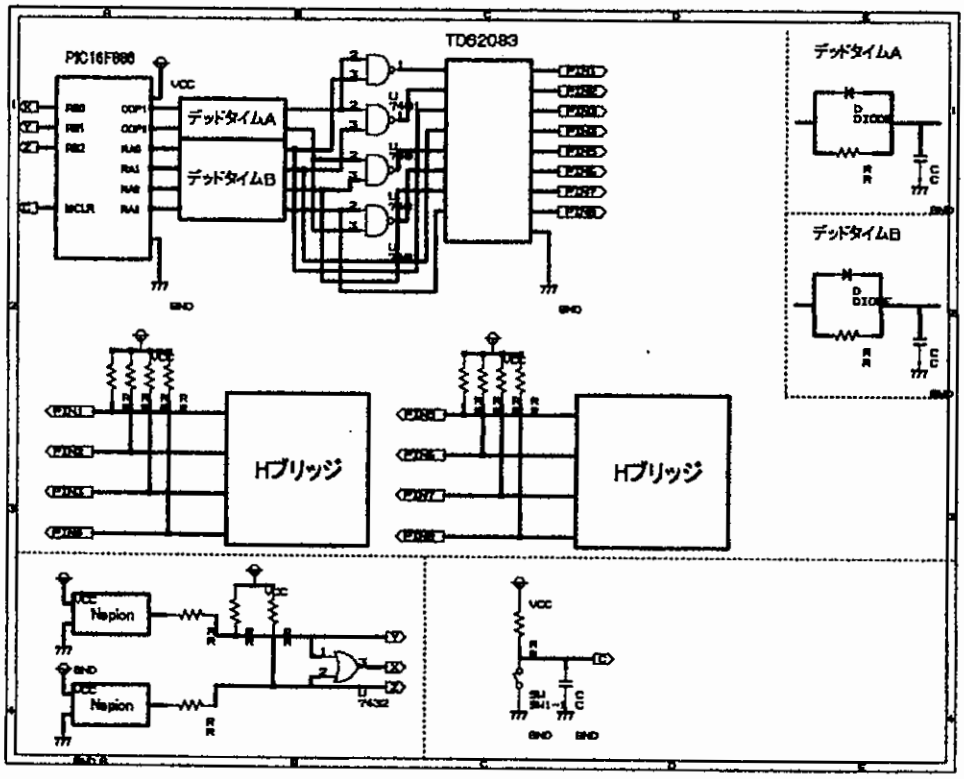
また、速く回りすぎてもキャタピラが吹っ飛んでしまうので、ＰＷＭ（図参照）というものを行っています。ＰＷＭとは、簡単にいうと次ページの図のようにＨの割合を自由に制御するものですね。Ｈの割合のことを、デューティーと言います。今回はデューティー約７５％にしています。
ＰＩＣでは、１６Ｆ８８６とかがＰＷＭモードができるピンが２つあって、１個で２個のモーターを制御できてお手頃な気がします。

図



デッドタイム回路とは何かとも説明しておきます。Hブリッジの怖いところの一つとしてスイッチング速度の差があります。この差からすべてのＦＥＴがＯＮになってしまうことがあります。すべてONになるとＶＣＣとＧＮＤが直接接触してしまいます！つまり、ショートです。ＦＥＴが火をふきます。本当に恐ろしいです。その対策としてあるのが、デッドタイム回路です。ダイオードによってONのタイミングをずらしています。ふつうはプログラムでやるらしいですよ。

後輪は、人間が手を振ると進むという機能を付けるために、「Napion」
という、赤外線センサーを使っています。使い始めて 4 カ月経ったのに読み方が未だわかりません。ナピョンですかね？？
このセンサーは、モーションセンサーなので手を振らないと反応しません。
ちなみに、手が冷たいと反応しません。
しかし、手が冷たい人は心が温かいと僕は思います。詳しい説明は、武子がします。
そして、上記のものすべてをまとめた回路図がこれです。



＜前輪＞

前輪は、武子が説明したとおり、普段は空回りして障害物にぶつかると三角形ごと回転します。
上記の動作をするのに、都合のいいモーターがあります。そうです。ステッピングモーターです。このモーター、トルクは弱いものの、動かしたい角度ぴったりに動かすのには、最適です。今回はマイクロスイッチに触れたら 120 度回転させるようにしています。回路は、市村と同じなので、省略します。
わりと大きい障害物も乗り越えられます。

3:作ってみての感想

＜大森＞

すべてにおいて、ハチャメチャに作りすぎましたね。
最初のほうは特に二人ともあたふたしていました。役割分担も決まってなければ、コンセプトも決まってなかったので、当然です。しかも、チカチカ隊長という任務もあらわれ、一時期完全にキャパ越えしてやばかったです。そして、ありとあらゆる物無員のみなさんやＯＢの方々を頼りすぎました。迷惑かけてすいませんでした。
でも、製作を本当に楽しめたのでそれはよかったと思います。
来年はもっと製作のことを勉強します。

＜武子＞

今年 1 年製作してきたわけですが、とりあえずプログラムに一切触れなかったのはミスだなと思っています。過去にいたある先輩のように金属加工技術を極めたいのですが、いかんせんセンスをあまり感じられませんね。
だから、プログラムという逃げ道も作っておきたいです。
まあ、とりあえず、文化祭後に重心移動の計算の勉強がしたいです。

ここまで、お付き合いありがとうございました

ペンと線

製作者　M3 新田 京太朗
M3 橋本 佑由太
協力者　物無・OBの皆様

1 ペンと線！？

まぁ、ペンと線です。略してなんていうんだ？小型機(おまけ)のほうは追跡者です。

2 概要

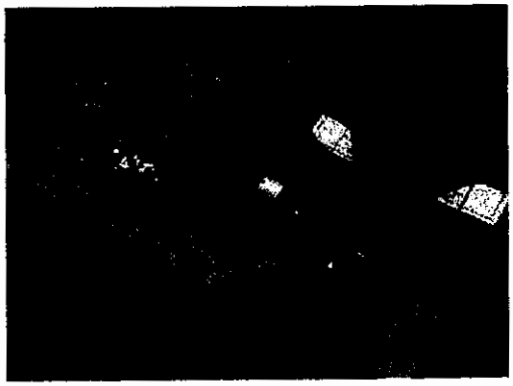
結局何をするの？と言われますと、3輪のラジコンカーがペンを持って、文字を書きます。他にも、直線を描いて、もうひとつの3輪のラジコンカーがライントレースをしたり、他にもいろいろな機能を持って自由に動いたりします。

3 概観・仕様

ペンと線

最高速度: 約25cm/s
電源　: 9.6V電源(バッテリー)
重量　: 約1500g
大きさ　: 約250×200×210mm
操縦方法: 無線リモコン
ペン幅　: 12.2mm

追跡者



最高速度: 約20cm/s
電源　: 単三電池3本
重量　: 約500g
大きさ　: 約210×100×50mm
操縦方法: オート

4 動作・機能について

ペンと線

Aラジコンカーとしての機能

前後移動、斜め前後ろへ移動、その場旋回、

B ライントレースのコース作り

ペンを下ろしながら走ればコースを作ることが出来ます

C 文字を書く

リモコンで決められたアルファベットを書くことが出来ます

普通にお絵かきすることも出来ます。

D ライントレース

一応こっちでも出来たりします。

追跡者

A ライントレース

↑が書いてコースをたどって行きます。赤外線測距モジュールを使い前に障害物があると止まれる安全走行をします。障害物が Unconfirmed writing object の場合、距離をとりながら走行します。

B 完全オート走行

A のライントレース無しのようなもので、平面を自由に走ります。勿論、障害物があったらよけます。勿論落ちそうになったら止まります。

C 文字消し

書いたら書きっぱなしもあれなので、書いたコースを消します。

5 モーター及びリモコンについて

使っているモーターは 3 種類です。DC モーター、ステッピングモーター、サーボモーター。ラジコンカーの駆動にはすべて DC モーター、ペンの上下にサーボモーター、ペンの左右移動にステッピングモーターを使います。DC モーターは　　を参照に、ステッピングモーターは　を参照に、サーボモーターは　　を参照にしてください。

ここでは、無線リモコンの仕組を説明したいと思います。

o 無線リモコン

無線とは？

→単刀直入に言ってしまえば赤外線です。赤外線の点滅によって通信をします。赤外線は目に見えません。赤・橙・黄・緑・青・藍・紫は可視光線といって目に見える光です。赤外線は赤より波長が長いものです。逆に紫より波長が短いのは紫外線って言いますね。

なぜ赤外線を使うのか？

→主に屋外で使う自動車用ドアロック・ワイヤレスリモコンは周囲の明るい光が妨害源となり赤外線通信には不向きなので電波を利用するものが多いが、強烈な光に晒されることの

ない屋内で使われる家電製品のワイヤレスリモコンは電磁ノイズの影響を受けない赤外線を利用しているものがほとんど。電波で通信する方式に比べて、信号が広がりにくく（回折を起こさず）、障害物があると通信できない欠点はあるものの、それは第三者に傍受されにくいというセキュリティ上の大きな長所でもある。携帯の通信にも使われていますね。

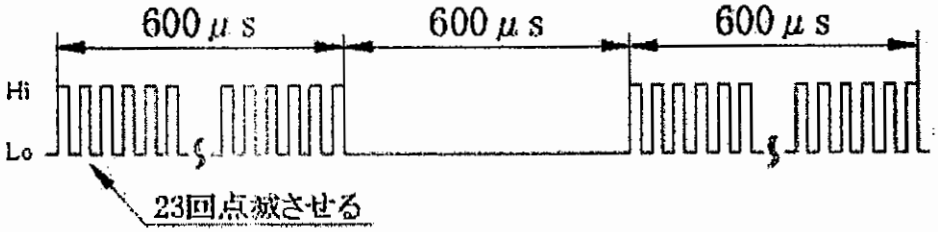
具体的にどうやってデータを送るのか？

→赤外線を出す LED(赤外線 LED)と赤外線リモコン受光モジュール(PL-IRM0101)をそれぞれ PIC で制御します。今回はテレビリモコンの通信の仕組を応用し使っています。

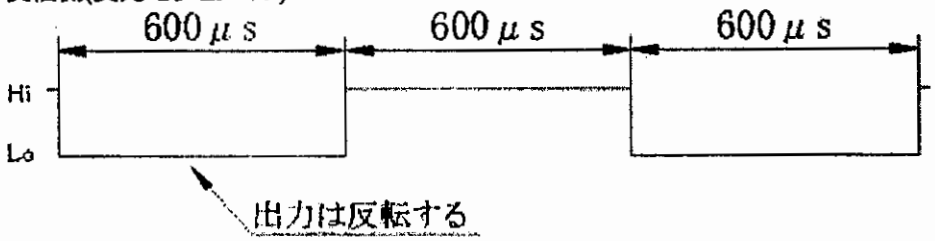
注意すべきところは"キャリア周波数"というものがありものによりますが 38kHz のものや 40kHz のものがあります。今回は 38kHz のものを使います。"キャリア周波数"って何？という人がいると思いますが、キャリア周波数とはその周波数通りの点滅でなければ受信しませんということなのです。つまり、送信側は受信させたければ、38kHz で赤外線 LED を点滅させないといけないということなのです。もう 1 つ注意すべきところがあります。"出力パルス幅"というものが存在します。今回は 600μs のものを使用します。また"出力パルス幅"ってなにという人がいますが出力パルス幅とは受光モジュールが出力する H と L の時間の単位のことです。つまり 1 周期 600μs というわけですね。

以上のことをまとめるとこんな感じになると思います。

送信側(赤外線 LED)



受信側(受光モジュール)



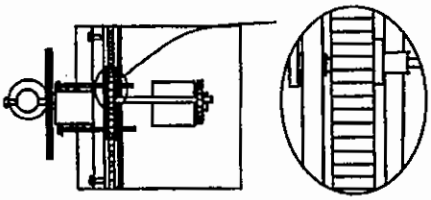
何故、23 回点滅させるかわかりますよね？38kHz で点滅させないといけないので、1 回の点灯または消灯に 1 秒÷38kHZ≒26μs.そして、点滅回数は 600μs÷26μs≒23 回となるのでそうなっています。これで、光るか光らないかでの 2 進数が出来るのでこれを PIC で読み取れば良いって話ですね。コレをもとに作っています。プログラムは 7 回路図・プログラム etc を参照してください。ちなみに、テレビリモコンには正確に受信するために、最初にリードコードというものを設け、リモコンの種類を識別するためカスタムコード、正確に受信するためデータのコードを 2 回送り(2 回目のコードは反転している)、終わったことを示すストップ

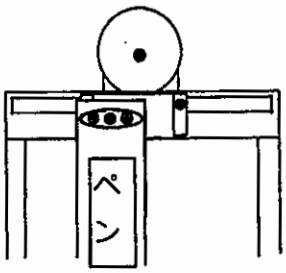
コードというものをまとめて送っています。

6 動作の仕組み

- どうやって書いているのか？

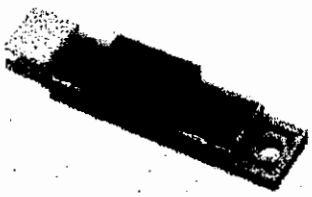
5.の最初に書いてありますが、サーボにペンをくくりつけてある角度のときに紙に接触するようにしています。左右移動は、ステッピングモーターにギアをかませその先でペンにくくり付けたサーボごと動かしています。上下移動は、ロボット自体が動きます。

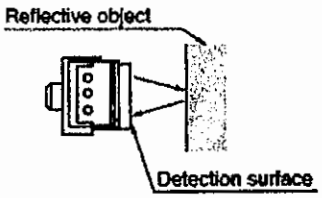




- ライントレース方法

DCモーターの前に反射型フォトインタラプタというものを2つつけています。そこからの信号をPICに入れて判断しています。反射型フォトインタラプタって何という人がいるかもしれませんが、要は、赤外線LEDとフォトダイオードがセットになっているものです。具体的には右の絵を参照にしてください。





7 回路図・プログラム etc

- 回路図

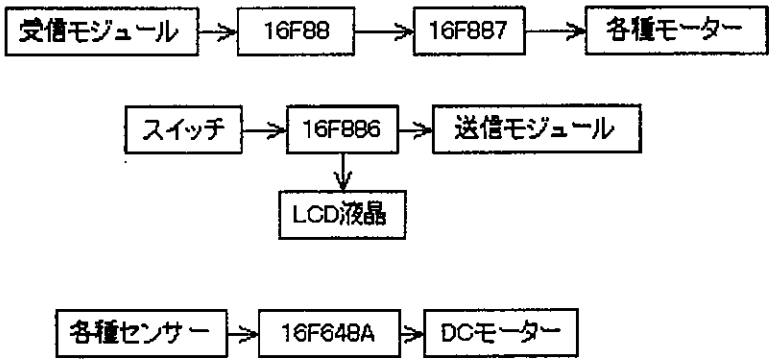
基本的に単純なのでブロック図にさせていただきます。

「ペンと線」に使用しているPICは16F887と16F88です。88は赤外線の受信にしSPI通信で転送、各種モーターをすべて887に動かさせてます。

リモコンはPIC16F886を使っています。

追跡者にはPIC16F648Aを使っています。

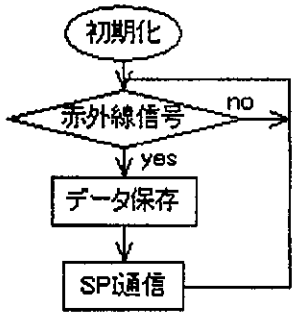
上から順番にペンと線,リモコン,追跡者になっています。

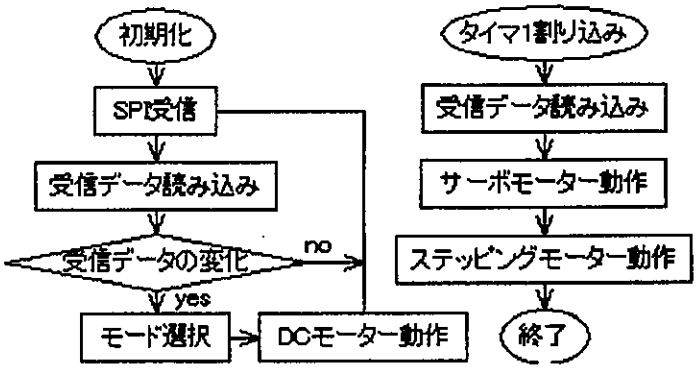


o プログラム

こちらも基本的に単純なのでフローチャートにさせていただきます。

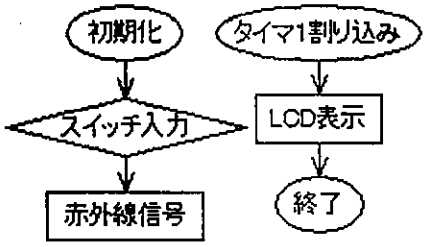
1　16F887(本体部分)



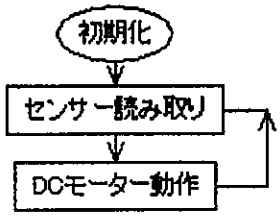


2　2　16F88(赤外線受信部分)

③ 16F886(リモコン部)



④ 16F648A(追跡者)



8 最後に。

新田

やっぱり共同制作は難しいですね。二人でコミュニケーションをとって、意思疎通はとても大事なことだと思いました。個人としては、いろいろなプログラムを書いて動かせて結構充実していたと思います。まぁ、いろいろとのんびりしているときが多すぎて、もうちょっと計画的に製作していれば今になってあせることもなかったと思います。来年はもっと、今までにないロボット

を作ってみたいです。

橋本

今回は主に本体を担当しましたが、緻密な設計をしなかったため、螺子穴が左右非対称だったり、上手く基盤がはまらなかったりと、蓋を開けたらなんとやらでした。

今回感じたのはちゃんとした計画を立てることでした。はじめから１年間の見通しを立てていればもっと効率的になったのではないかと思いました。

物無の皆様、OBの皆様、有難うございました。

CoilGun I・II

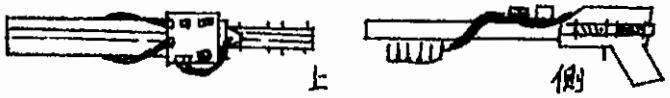
製作者：M2 岸田
回路設計者：H2 竹下さん
協力：物無の晴さん

☆概要

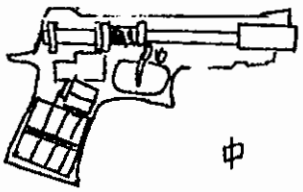
この製作物は名前の通りコイルガンです。コイルガンって何？という人はもう少し待ってください。どかの超電○砲では一切関係ありません影響も受けていません言うならばとある動画サイトでアルミ缶をコイルガンでぶち抜いている人がいたからです。因みにあれてアニメと関係を持たせると「超誘磁砲」とでもなるのでしょう。

☆外観(おおよそ)

I



II エアガンにつめてみた感じで。元の銃はトカレフと思われます。



☆多少詳しい仕様

	動作電圧(予定)	コンデンサ数	重量	大きさ	PIC	段数
I	5V	10_2	重い	大きい	有	2段
II	1.5V(単4電池)	10_3	軽い	小さい	無	1段

☆原理

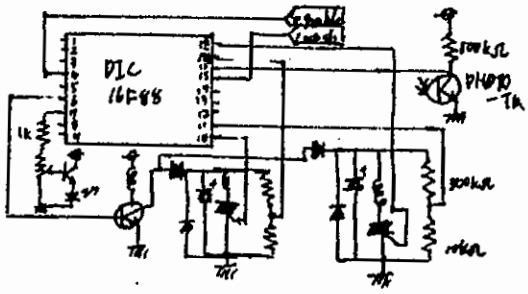
単純に言えば電磁石で引きつけた弾丸をそのまま飛ばす物です。その為弾丸は磁性体(鉄やニッケルやコバルト)である必要があります。多くの人は鉄釘の先を2cm程に切って使っているでしょう。今回もそうです。

又弾丸が電磁石を通り過ぎた後に磁力が残っていると弾丸が引き戻される為に電磁石に流す電流は一瞬である必要があります。なので一般的には放電を一瞬で行う特性のあるコンデンサという部品を使います。今回はカメラのフラッシュに使われる物を流用します。

何故かというと、さっきから電磁石と言っていますが実際は鉄芯を入れていないのでコイルというのが正しいのですが、この状態だと磁力はとてつもなく弱い為なるべく大きな電流を流す必要があるのです。そこで備えられる量が大きく安い(つか無料)のカメラのコンデンサを用いる事にしました。

※回路

I



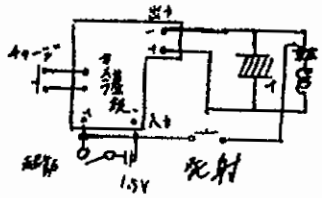
簡単な説明

の 赤 トライアック。(3)に電流を流すと(1)と(2)が導通する。非接点式スイッチ。

赤 コンデンサ

青 コイル

II カメラの昇圧回路流用



← 安あがりですよ。
どこかどこかは
ご自分で調べて下さい。

プログラムは僕にはサッパリ解らないので竹下さん人に書いていただきました。ありがとうございます。

※現在(3/1)

トライアックが動かず遅展していません。既に言えばトライアックこそ動けばほぼ完成です。トライアックをやめてサイリスタにしようかな、位人て考えてみたりもしてます。完成したらアルミ缶の片面位は貫通してもらいたいです。

※感想

トライアックなんて二度と使いたくなくなりました。不快力しが強くてそうなんですが。

個人的に、エアガンに結び込むという発想は面白いと思います。

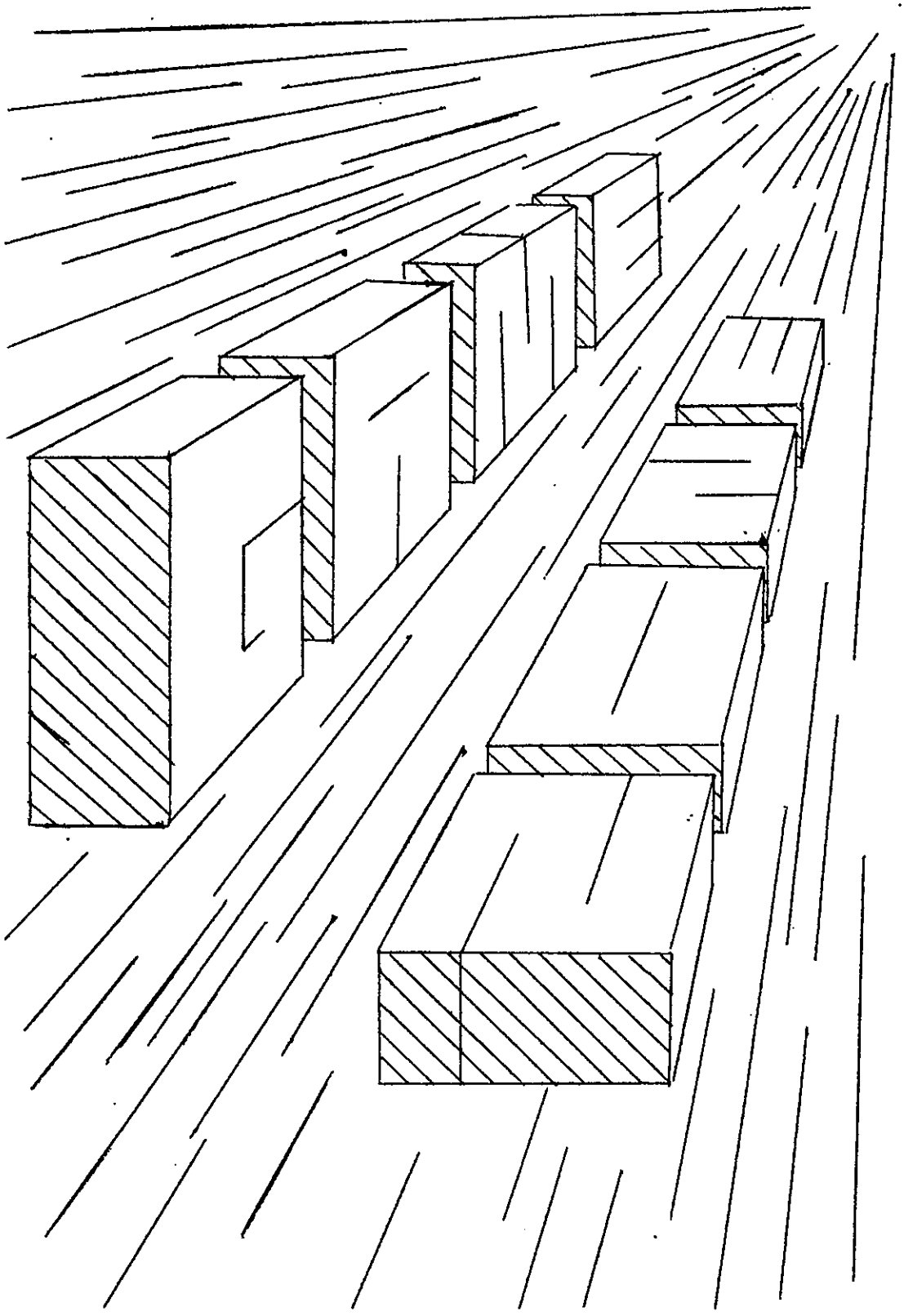
前買った物が非常に弱かったので破壊した残りを見て思いつきました。

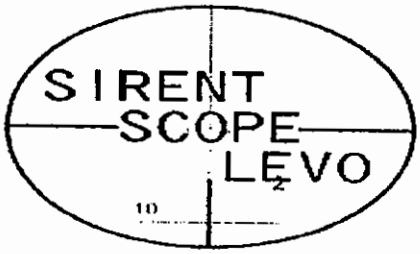
めざせ完動!! アルミ缶片面貫通!

※最後に

これを真似して起きた事故等に対する責任は一切負いません。

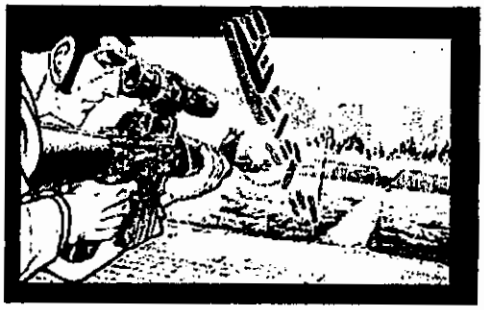
特に殺傷能力を持つ程にしたり感電したりしても一切知りません。全て自己責任で作業してください。





製作者　高二　飯田亮
協力者　物無の皆様

1.外観



これくらいやってみたいですね～(笑)→

これだけでは何が何だかわからないので説明を挟みますと、これはゲーセンにある某TIME○LISISというゲームのような、画面に向かって電子銃を撃ち、画面に表示された的を撃ち落とすというゲームです。

2.技術的なこと

～液晶～

秋月電子で売っている SG12864ASLB-HS という白黒液晶を四つ使っています。それが上の絵の「的」と書いてあるところにひとつずつくっついています。液晶一つが3cm×4cmと小さいので、上の図のように離れて置いてやらないと、画面自体の大きさというものが確保できないため、このような配置に成っています。

・制御について

この液晶には、中に制御回路や画面のデータを保存するメモリが搭載されていて、中のデータを書き換えることで画面の表示を変えることができます。ということで、非常に制御が楽です。

①ピン

この液晶のピンの説明です。

ピン数	名前	説明
1	Vss	GND。
2	Vdd	内部ロジックIC用電源。５V。
3	Vo	液晶のコントラスト。
4	D/I	液晶に表示するデータか、中のロジックICに書き込むデータの選択ピン。
5	R/W	読み出し/書き込み切り替えピン
6	E	データの読み出し/書き込みのタイミングピン
7～14	RB0～RB7	データピン
15	CS1	右半分の液晶選択
16	CS2	左半分の液晶選択
17	/RES	おなじみリセットピン

18	Vout	液晶用電源。－5V。
19	A	バックライトのLED用電源（＋）。
20	K	バックライトのLED用電源（－）。

②制御

この液晶は、64ドット×64ドットの液晶が横に二つつながっていて、128ドット×64ドットの液晶を作っています。そして、それらの液晶は、CS1ピンとCS2ピンで制御を切り替えられます。

それぞれの液晶は、（横を128ドット、縦を64ドットと見るならば）縦に8ビット×8段に分かれていて、その8ビットそれぞれにデータを書き込んで表示しています。このとき、この液晶には三つのコード(colum address, page, display start line)が用意されています。まず、「page」は、段数を指定し、「colum address」は、横何番目(左から数えて)のドットを選択するかを指定します。「display start line」は、上から何番目のドットを選択するかを指定できるようですが、よくわからなかったので今回は使っていません。「page」と「colum address」を指定したあと、「display date」でRB0～RB7に入れたデータが画面に表示されます。この操作をひたすらやっていけば、画面に絵が表示されます。

③プログラム

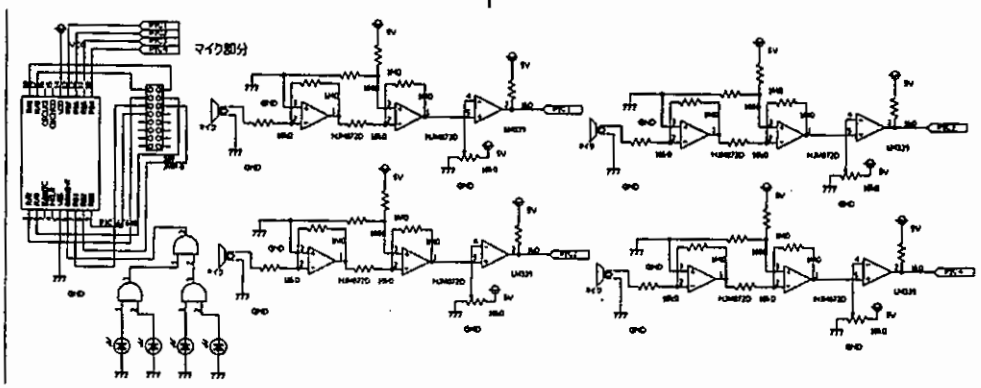
今回はマイコンにH8-3052Fを使用しています。スペースが無いので液晶への入出力のプログラムのみ載せます。

```
#define E 1
#define RW 2
#define D 4
#define IC2 8
#define IC1 16
#define RES 32
void wait() {
 unsigned char x = 1;
 while(0<x) {
  x--;
 }
}
 return ggg;
}
char reading() {
 unsigned char z;
 PADDR = 0x00;
 z = RW + RES + IC1 + IC2;
 wait();
 PBDR = z | E;
```

```
char dateread(unsigned char codes) {
unsigned char z,ggg;
PADDR = 0;
z = RW + RES + D + codes;
wait();
PBDR = z | E;
wait();
PBDR = z;
wait();
PBDR = z | E;
wait();
PBDR = z;
 ggg = PADR;
PBDR = z;
while(1) {
 z = reading();
 if(z == 0x00) {
 goto readend;
 }
}
readend : PADDR = 0xff;
```

回路図とプログラムです。プログラムは、距離測定のみを書いています。

```
#include <16f648a.h>
#fuses
#use fast_io(A)
#use fast_io(B)
#use delay(CLOCK=20000000)
    set_tris_A(0);
  datewright(k2,1);
  datewright(k3,2);
  datewright(k4,3);
  (上の while(input_pin(PIN_B0)からここ
でがもう一回続きますので省略します。)
 set_tris_B(0xf1);
 setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
 output_a(0xff);
 //reset ok!//
 while(1){
  while(input(PIN_B0)){}
  output_high(PIN_A4);
   set_timer1(0x00);
  do{
   x = input_b();
   x = rbdate & x;
   time = get_timer1();
   if((0x10 & x) == 0x10){
   rbdate &= 0xe0;
 HS,NOWDT,NOPROTECT,PUT,BROWNOUT,
NOLVP,CPD
void main(){
 unsigned char x,rbdate = 0xf0;
 unsigned long k1,k2,k3,k4,k5,k6,k7,k8,time;
     k1 = time;
   }
   if((0x20 & x) == 0x20){
    rbdate &= 0xd0;
    k2 = time;
   }
   if((0x40 & x) == 0x40){
    rbdate &= 0xb0;
    k3 = time;
   }
   if((0x80 & x) == 0x80){
    rbdate &= 0x70;
    k4 = time;
   }
  }
  while(get_timer1() < 0xbffd);
  datewright(k1,0);
  output_low(PIN_A4);
  rbdate = 0xf0;
  k1 = k2 = k3 = k4 = k5 = k6 = k7 = k8 = 0;
 }
}
```



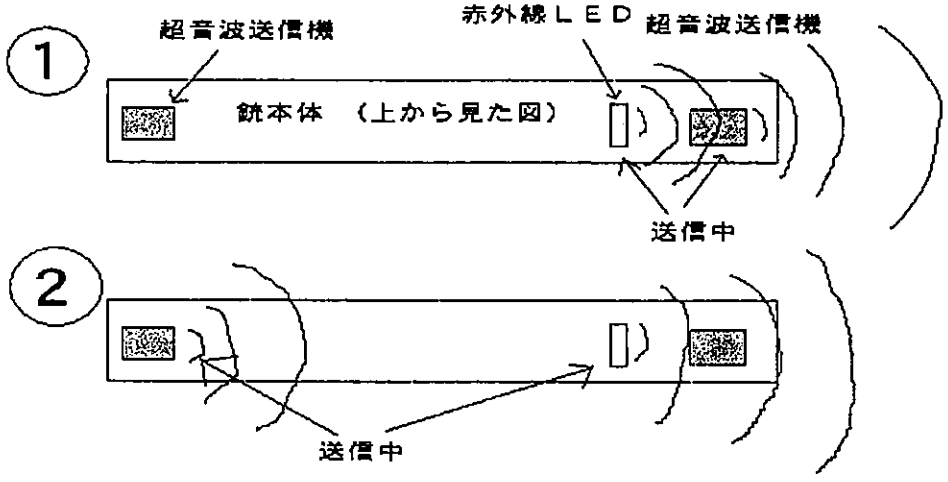
```
 wait();
 PBDR = z;
 wait();
 PBDR = z | E;
 wait();
 PBDR = z;
 z = PADR;
 return z;
}
void writing(unsigned char x,unsigned char y) {
 unsigned char z;
 PADDR = 0xff;
 PADR = y;
 z = RES + x;
 wait();
 PBDR = z;
 wait();
 PBDR = z | E;
 wait();
}
void colum(unsigned char x) {
 unsigned char codes,date;
 date = x + 0x40;
 codes = IC1 + IC2;
 writing(codes,date);
}
void page(unsigned char x) {
 unsigned char codes,date;
 date = x + 0xb8;
 codes = IC1 + IC2;
 writing(codes,date);
}
void line(unsigned char x) {
 unsigned char codes,date;
 date = x + 0xc0;
 codes = IC1 + IC2;
 writing(codes,date);
}
```

～超音波～

ここでは、超音波センサを使った距離の測定方法を説明します。超音波によって、銃との距離を測り、その距離から銃の場所（x軸、y軸、↓軸）を計算して出します。

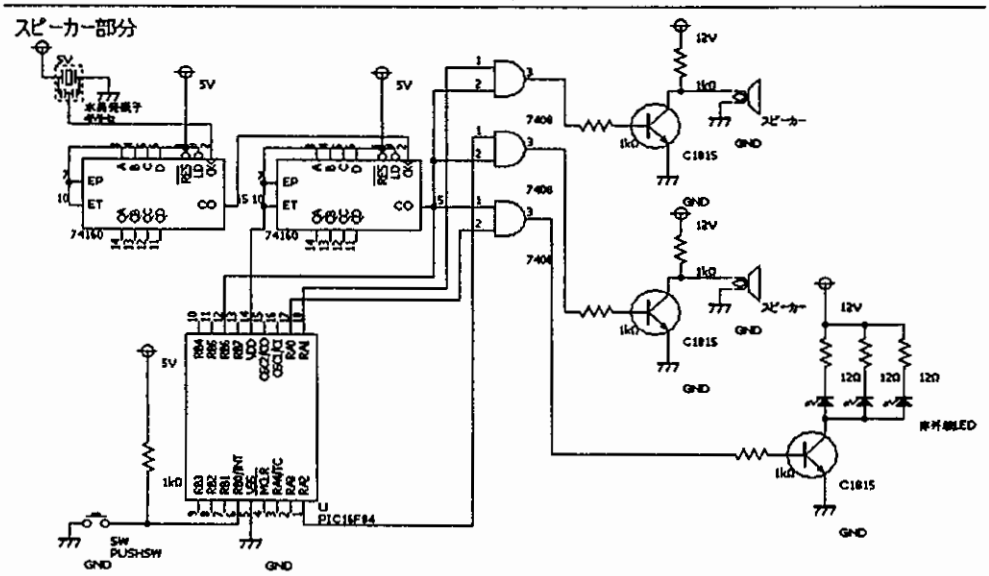
①送信部

まず、送信側（銃本体）から赤外線によって受信側に超音波を送信することを伝え（光はほぼ一瞬で届くので）、同時に超音波を送信します。その後、銃のもう一箇所に取り付けてある超音波送信機を使って同じことを行います。



回路図とプログラムです。

```
#include <16f627a.h>
#fuses
HS,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP,C
PD
#use fast_io(A)
#use fast_io(B)
#use delay(CLOCK=20000000)
#INT_EXT
void wariko() {
 output_high(PIN_A3);
 set_timer1 (0);
 output_a(0x04);
 while(get_timer1 () < 0xa) {}
  setup_timer_1 (T1_EXTERNAL_SYNC |
 T1_DIV_BY_1);
 enable_interrupts(INT_EXT);
 output_a(0x07);
 output_low(PIN_A3);
 while(get_timer1 () < 0xfe00) {}
 set_timer1 (0);
 output_a(0x02);
 while(get_timer1 () < 0xa) {}
 output_a(0x07);
}
void main() {
 set_tris_A (0);
 set_tris_B(0xff);
 output_a(0x03);
 ext_int_edge(H_TO_L);
 enable_interrupts(GLOBAL);
}
```

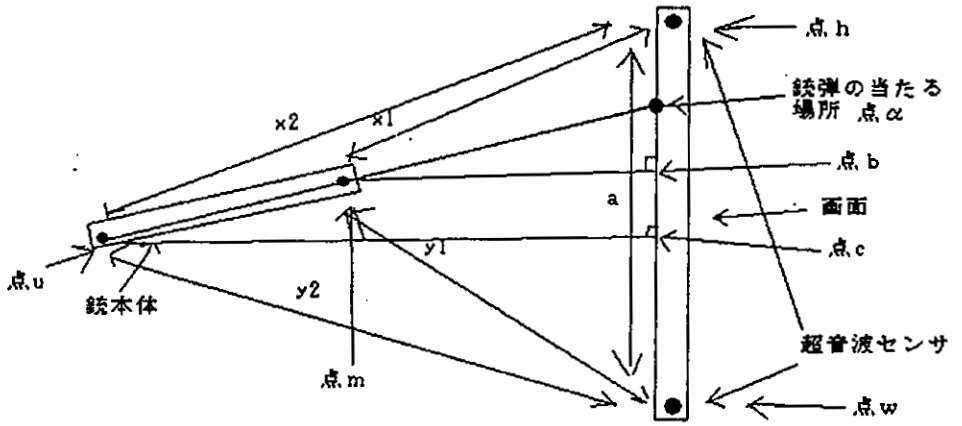


②受信部

赤外線受信モジュールに、赤外線が入ったことを確かめたら、カウントを始めて、超音波が入るまでカウントを続け、4箇所すべての超音波の入力が入ったらLCD液晶を制御しているH8-3052Fにデータを転送しています。超音波受信機の信号が低いので、OP-AMPで反転増幅して約一万倍に増幅しています。その場合、小さなノイズでも拾ってしまうので、基準電圧0vで反転増幅した方がいいです。また、ここではPICでカウントを行っていますが、LSICでカウントを行った方が正確です（たぶん）。

～場所の出し方～

たとえこれで距離が計測できても、計算できなければ意味が無いので、その仕組を説明します。



画面の幅をaと置き、銃本体の前方から画面の左はじまでの距離をx1、右はじまでの距離をy1と置き、銃本体の後方から画面の左はじまでの距離をx2、右はじまでの距離をy2と置き、銃弾の当たる位置（不明）を点aと置きます。

まず、点mから垂直に画面に線を下ろしたとき、点hからの距離 hb = a×x1/(x1+y1) となり、点uから垂直に画面に線を下ろしたとき、点hからの距離 hc = a×x2/(x2+y2) となります。点mから画面に引いた垂線の長さ mb = √(x1×x1-hb×hb) になり、同じく点uから画面に引いた垂線の長さ uc = √(x2×x2-hc×hc) になります。そして、線abの長さが、αb=(hc-hb)×mb/(uc-mb) になり、線ha = hb - αb となります。これを画面の縦、横両方行えば、銃弾の当たった場所のx軸、y軸が求まります。

～感想～

4月一日現在、超音波が一応動いて、モジュール関係の足並みが揃いました。しかし、いまいちうまく超音波が距離を測定してくれないので、CPLDを使ってロジックICでもって時間を計ろうかと思います。ゲームのプログラムとか、場所の検知とかはなんとか3,4日で動いてくれればきっと当日には動いているはず！だと信じます。今年の製作は液晶や超音波など始めてやるもの（特に超音波などのアナログ関係はほぼ始めて。それまでOP-AMPとかは全く使ったことがありませんでした。）の癖に、一人きりでこれまでコツコツ制作してきたので、まあなんとかやった方かなあと思っています。チョロッと心残りなのは、結果的にゲーム系の分類から意外のほかのコンピューター系などの分類に属するものを作れなかったことが残念です。あと、誰かと共同制作したかった！というのがあります。あとは、ここにH8-3052Fのプログラムをもっと載せたかったですね～。液晶に関する関数を色々と作ったのですが、量が膨大なのと、あのプログラムは俺の頭をフル回転させて書いたものなのでその味わいをこの部誌にかけなくて残念です。まあ、麻布生活最後の製作としてはいいモン作れたんじゃないかな？と思っております。

年刊少女マンデー

Vol.1

申し訳ございませんが

ページ数がずれました。

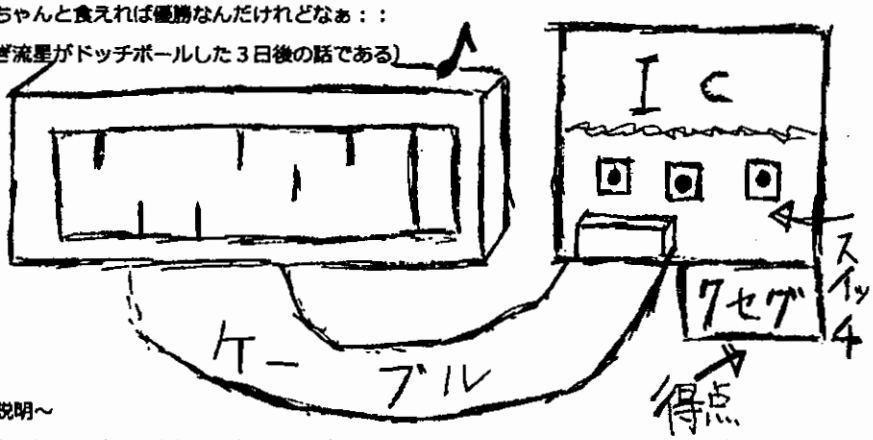
ながしそうめん

製作者　　H1　梅本 颯
協力者　　物無の樹さん（ほとんど池上さん）

～ストーリ～

2012年、もぎ流星と名乗る人物がいた。もぎ流星は、どうでもいいときだけ活躍できるやつで、これから街の「そうめん大食い大会」に出るのだった。中国産そうめんはなんと赤、緑、青の三色だ＾＾
そうめんをちゃんと食えれば優勝なんだけれどなぁ：：
（これはもぎ流星がドッチボールした３日後の話である）

～外観図～



～ゲームの説明～

外観図にもある通り、液晶の右側に黄色の縦線が入っています。そして左側から次々と３色（赤、青、緑）の棒が流れてくるので、黄色い縦線に重なったときに３つのボタンを使い分けて叩いていきましょう。というもの。成功するたびに得点が上がるよ。タイトルがあんな馬鹿みたいなのは「棒が流れる！」というコンセプトでの僕のテンションです～

～液晶について～

０）液晶表示の　キ・ホ・ン

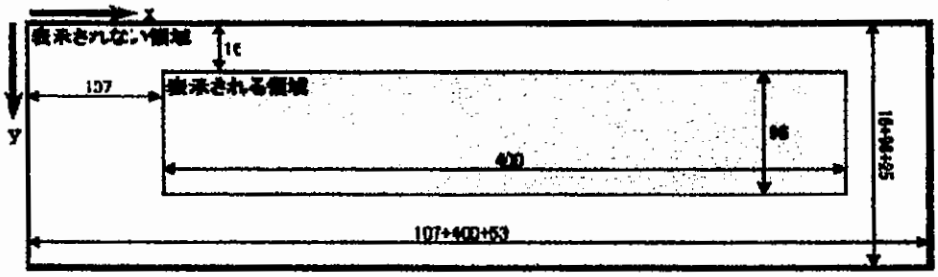
前の回路図集にもこんな項がありましたが、基礎がなってないと何事もわからないからねっ
というものの、スペースが足りないんで図を無しで説明します。すいません。
液晶は簡単にいうとデータをつっこむといったことを繰り返して表示させるものです。何を突っ込むかというと、後述に述べる色データやクロック、同期信号（あとにいうhsyncなど）です。これらを約束通りに入力していけば一応光ります。
具体的な入力の仕方は読んでいけば↓わかると思います。

そして、液晶は細かい点で出来ているのですが、各点を光らせることによって光ります。光らせる順番と言うものがあってまず左上を光らせます。クロックを１入れることで光らせる点を一個ずらすことが出来ます。それを繰り返して横一列終わったら同期信号を入れることによって、次の一列を光らせることが出

来る様になります・・・
というのをずっと繰り返して一回画面を光らせます。(大変すぎやろ・・と思った人もいるかもしれませんがクロック入れるのを繰り返すから案外楽ですよ) 一回光らせてもすぐ消えてしまうのでその作業を永遠に繰り返せばその画面はずっと光ったままです。

a) 使っている液晶

使っている液晶は「LTA042B010F」という秋月という店で 300 円で売っているものです。実は昔の OB さんが使っていたものです。「ピン配置など問い合わせないでください」っていう紙が入っているのですが、もう今ではネットで情報が載っているので安心です。↓これ



b) 液晶のピン配置

「LTA042B010F　ピン配置」とか「秋月　液晶」とかで調べたほうがきれいで見やすいはず。他はコピペしてないから許してね。

c) ピンの説明

DE――自分で液晶に表示するなら L 固定。イネーブルです。
NCLK――液晶に入れるクロックです。今回は PIC のピンを H→L→H と繰り返していくことで作りました。立下りエッジ（簡単にいうと H→L になったときですよ）で色データなどを読み込みます。他のピンの所でも説明します。
HSYNC――液晶を表示していくにあたって、横一列に 353 クロック以上入れた上でこのピンを H にすると次の一列に移動することが出来るのです。これも立下りエッジで読み込むため、
クロックを H にする→HSYNC を H にする→クロックを L にする→HSYNC を L にする
とやるんです。

VSYNC――HSYNC は横一列が終わったときに動かすのですが、このピンは縦がすべて終わり、また最初から光らせるときに使うのです。
横一列終わり→HSYNC の操作→これではまだ縦一列分しかやってないので、縦全てやる→VSYNC→最初の戻る

と使います。これも立下りエッジで読み込みます。但し、このピンをHにして153クロック以上入れないといけません。

G0～G5、B0～B5、R0～R5
これは単純に色データを入れたら、液晶にその色が出力されます。今回は3色しか使わないので、各色をまとめて、一色2bitずつにしました。これも立下りエッジです。

Vo――コントラスト用。要するに画像をきれいにするために電圧をぶち込むのですが、2～3V安定という情報があったので、半固定抵抗を使って2,2Vを作りました。

＋5V、±9V――そのままの値を入れるだけなのですが、電源周りがぐちゃぐちゃになると嫌なので、12V電源を3端子レギュレータを使って作りました。
－9Vは負電荷を楽に作ることの出来るICを使いました

ふう、36ピンもあるから説明も大変だね＾＾まあ、ピンの内容が分かってないと読んでもわかんなくなるから仕方ないんですよね～

d）プログラム
Cが分かりそうになかったので、アセンブラで書きました。（アセンブラって言うのは、低級言語の一つ。低級だからって馬鹿にしないで下さい）使ったマイコンはPIC16F887です。40ピンある物無が案外つかっているやつです。
プログラムと言っても（いや、ちゃんと書きましたよ）長すぎて到底乗せられないので簡単な指針だけを載せます。このゲームは縦棒を流すゲームです。割り込みが必要になるスイッチは3つしかないので、大事なのはちゃんと光らせることのみです。今までの僕のをちゃんと読んでいる人はわかるはず？

クロックを入れる→所々色を入れる→HSYNC→最初に戻って→VSYNC
とやっていくだけです。本当にそれだけ＾＾

e)回路図
はい、回路図というほど回路がありません ｆ＾¥＾；　ほとんどプログラムだからです。液晶本体とPIC繋いでるくらいなので、好きにピン配置決めてやってください。

ｆ）配線
いやー番苦労したとこですよ、ここが。配線と言ってもPICと液晶本体を繋ぐ所です。
なぜ大変だったかと言うと、液晶のピンは全部で36ピンあるのですが、その1ピン1ピンの間（いわゆるピッチ）が0,5mmなんです！ピンと来ない（駄洒落じゃないよ）かもしれませんが、普通の複線や単線

じゃ無理（単線なら本当にがんばれば行ける。僕には無理でしたが）なんでポリウレタン銅線（UEW 線）を使いました。まあすごく細くてやりやすかったです。よく断線するって言われてますが、なぜか断線しませんでした。でも丁寧に扱いましょう。

様子（僕のじゃありませんが）

まあ、こんな感じです。

～感想～

元々ゲーム作る予定だったんですが、心変わりが多くて結局真面目に作り始めたのが 1 月とかな気がします。まあ、今回一番困ったことは本体と PIC の配線なわけであって、あそこで 2，3ヶ月詰まっていたと思うと物悲しいです。大事なのは最初から予定を立てておけば・・・という感じです。

ゲームとしては面白いと思って作ってきたんですが、流れるスピードと種類によってだいぶ変わってきます。プログラムも大事ですよ。

4月の最初の時点で一応盤面は流れるので、文化祭ではゲームになっていると信じて頑張っていきます。

最後に協力してくれた人に感謝します。ありがとうございました。

（余外スペース余りましたね）

バイク走2

製作者　　H1　伊藤　卓
　　M3　石井　裕太
協力者　　物無の皆様

～ゲーム説明～

このゲームは去年に作られたバイク走というゲームの進化版で、ＬＥＤに表示していたのが、ドットマトリックスというＬＥＤが 16×16 の 256 個集まったようなものに表示するようにしてあります。そしてスイッチではなくタッチセンサーを使用して操作するという仕様になっています。

上から障害物が流れてくるのでそれに当たらないように自機を移動させて進んで行くというゲームです。攻撃とかはなくただ避けるというゲームです。たださっき少し書いたように移動はタッチセンサーで行い、指をタッチセンサーの上でスライドさせると、そのスライドしたほうに自機が移動するというようになっています。

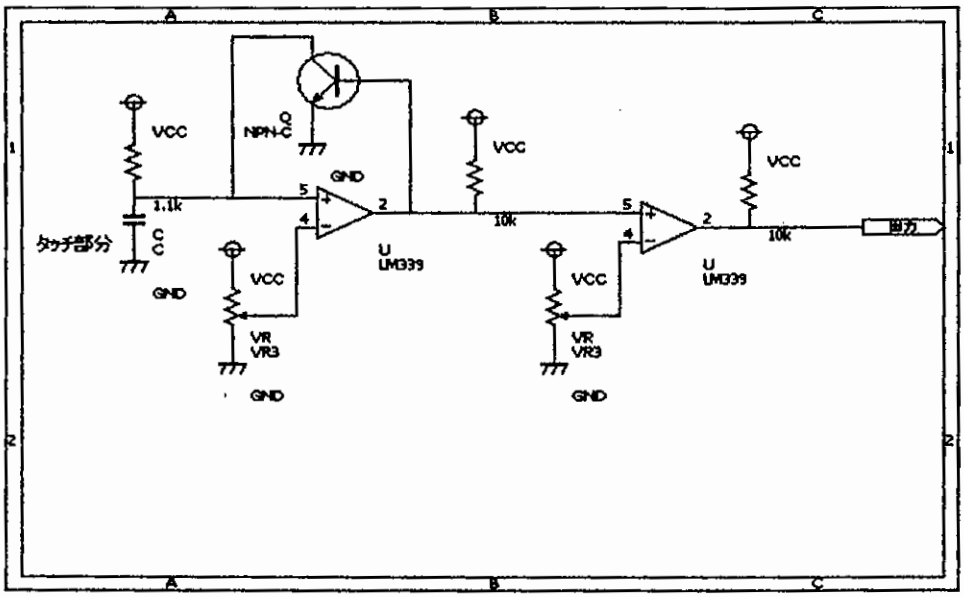
～タッチセンサー～

このゲームにはタッチセンサーが使用されています。とはいっても小さなタッチスイッチが並べられているだけなので、よくゲームや携帯などに使われているようなものではありません。

・タッチセンサーの原理

　このタッチセンサーは静電容量式という方法で作られています。静電容量式というのは、人体のようなある程度表面積があるようなものは静電容量をもっていて、その静電容量をもっているものが回路に触れるとその回路の部分の静電容量が変化します。この静電容量の変化というのを検出回路を用いて読み取るというものです。

・回路図



一応これはタッチスイッチ一つ分の回路です。

この回路の左側のコンパレータの出力のところは最初図１のように発振している状態になっています。このときにタッチ部分と書かれている場所に人体が触れると、この発振が図２のようにゆるやかになります。これはタッチした部分の静電容量が増加して、電圧が上がるスピードが遅くなったからです。そしてこの少しの発振のスピードの差を右側のもうひとつのコンパレータで読み込むことによって、タッチしているかタッチしていないかを判断します。

このタッチスイッチを大量に作って並べることによって、タッチセンサーとしてスライドなどの動作を読み込めるようにしています。

図１

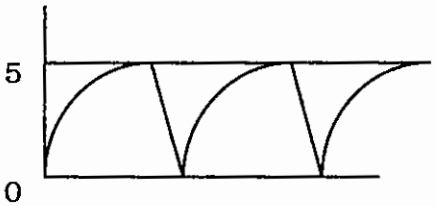
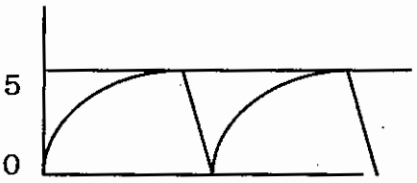


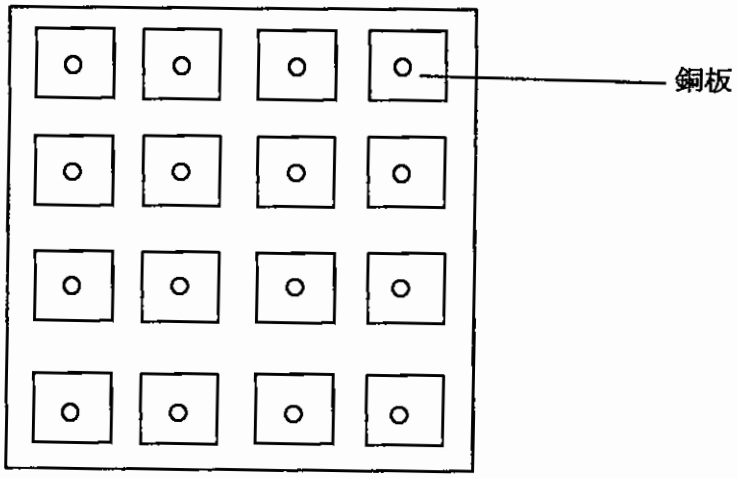
図２



・タッチセンサー部分について

このタッチセンサーはタッチスイッチを１６個作り並べたものです。

図



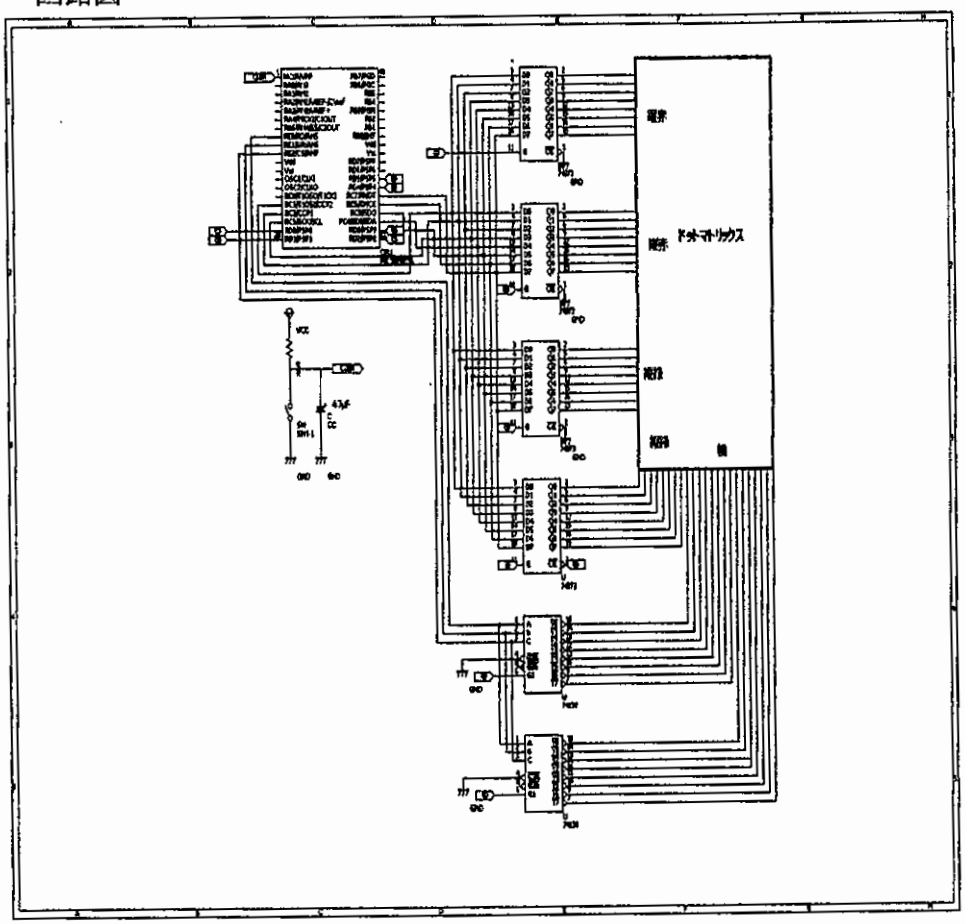
タッチ部分には銅板を使い下にテープを貼って固定しています。それぞれの銅板には真ん中に穴が開いており、そこに配線材を通して回路へとつないでいます。そして銅板と銅板の間にはホットボンドをつけて平らになるようにしています。

～ドットマトリックス部分～

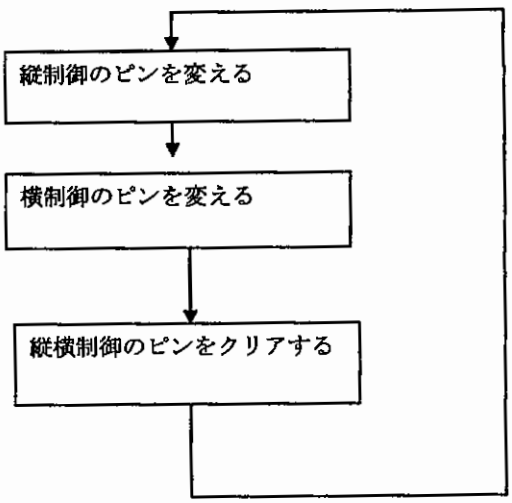
このゲームにはドットマトリックスが使われていますが、ドットマトリックスとは、横の列に対応したピンが１６、縦の列に対応したピンが３２（赤と緑を表示するため１６×２）そして一列（16 個）ごとのパターン信号で点灯し、横方向は時分割で点灯させ、それを人間の目に見えないくらい早くすることで、２５６個の LED を点灯しているように見せています。しかし、ドットマトリックスはピン数が多いので、PIC１６F８７７A というPIC のピンをＤラッチを使って増幅しています。

また、７４LS１３８のA、B、C ピンに入れる入力によって、横の光らせる場所を制御しています。

・回路図



・フローチャート



～これまでのあらすじ～

昔から「バイクでそこらへんを走りたい」と思っていたＭ君はついに念願のバイクを購入し、２ヶ月後の「WMR２０１８」という大会の第一予選に向けて、ジャンプをして障害物を乗り越えるという練習にはげんでいた・・・・・

～ストーリー～

そしてついに２ヶ月が経ち第一予選の日が来る。つらい練習を重ね障害物を避けることをついにＭ君はマスターしたが、第一予選のコースを見た彼は絶望する。一次予選は障害物などない、平坦なコースだったのだ！！ジャンプのみ極めた彼の今の実力では、到底予選通過などできない。精神的に追い詰められた彼は、あることを決意する。そう、バイクの改造である。彼は不器用ながらもバイクを改造し、バイクについているタッチセンサーをスライドさせるだけで、曲がったり速度を変えたりできるようにした。しかし、やはり貧弱な彼は、敵に触れただけで転倒し、再起不能になってしまう。華麗に敵を避け続ければ、予選通過もいけるであろう。さて、予選まではもう時間がない。彼は一度も敵に当たることなく、バイクマスターとなるための一歩である予選を通過できるのか！！

～感想～

・伊藤

今回作ったタッチセンサーはタッチスイッチの回路さえうまく動かすことができれば、あとは大量に作るだけというものだったので、うまくいけばすぐ動かすことができましたが、タッチスイッチをうごかすのに時間がかかってしまいました。ただ、PICを使わないでタッチスイッチが作ることができたのはよかったです。

・石井

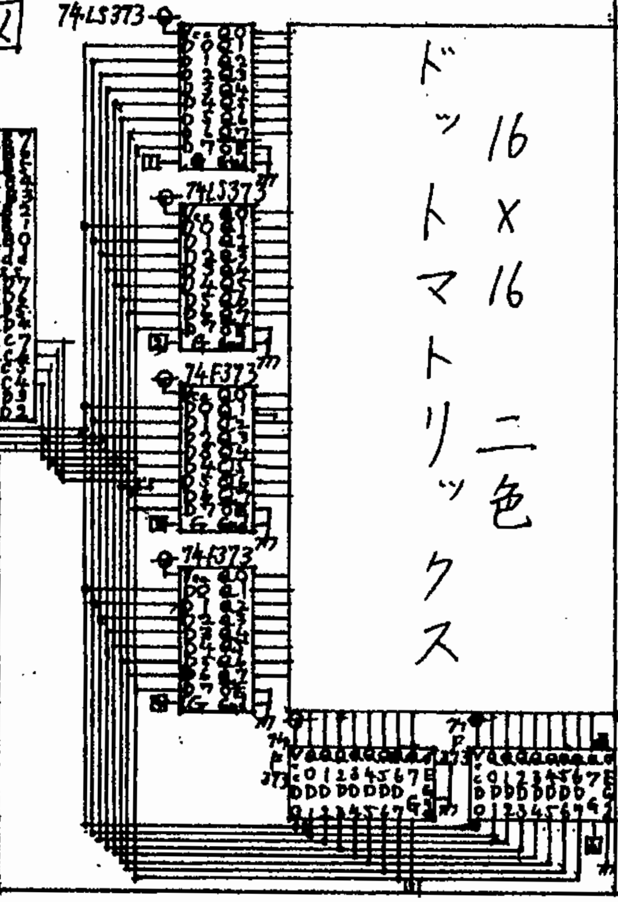
半年以上かけて制作してきたわけですが、僕の場合制作スピードがあまりにも遅すぎた気がしますね。部活にあまり来ないのが良くなかったです。そのせいで伊藤さんにも多大なる迷惑をかけたので、来年からは活動効率を上げていきたいです。

パニョを進め・・・

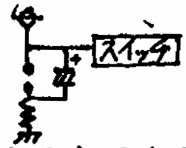
製作者　M3石原
協力者　物無の皆様

※↑題名はミスではなく省略しているのです

回路図



スイッチ類

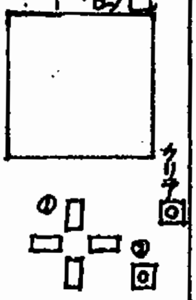


※それぞれ別です

ルール

・自機は1機、緑色の表示です。赤い流れる迷路の中を最上まで避けて進んで下さい。
・スイッチは①移動用(上下左右)自機左端と右端のすり抜けが可能になっています。
・3ステージ制になっています。
・敵の出現、ボムの使用、プレーヤーによる設定の変更やシナリオの変更などが予定されています。

外観

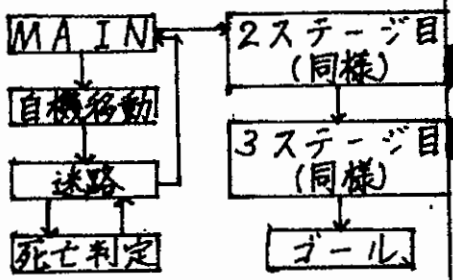


回路について

・もう少し、汎用ロジックICを使うべきだったと思います。たとえば、74164は迷路を流すのに使えましたし、あと列を選択する側の373(上の回路図で下の2つ)は、目的を考えれば74138の方が向いておりプログラムも楽になったのではないかと思います。
・ドットマトリックスにはそれぞれ縦横を制御するピンがあって、それぞれに対応する列が決まっており、ある一点において縦横両方(どちらかがVcc,Gnd)の入力の組み合わせで光るかどうかが決まります。それを人間の目に見えない速さ(参考までに発振子は20MHz)で行い、いわば活動的に(Dynamic)PICが動いて瞬時に光った後の残像を使い光っているかのように見せているのです。
※仕様は3月末現在のもので仕様変更の可能性があります。
また、この通りに作っても正常動作の保障はできません。

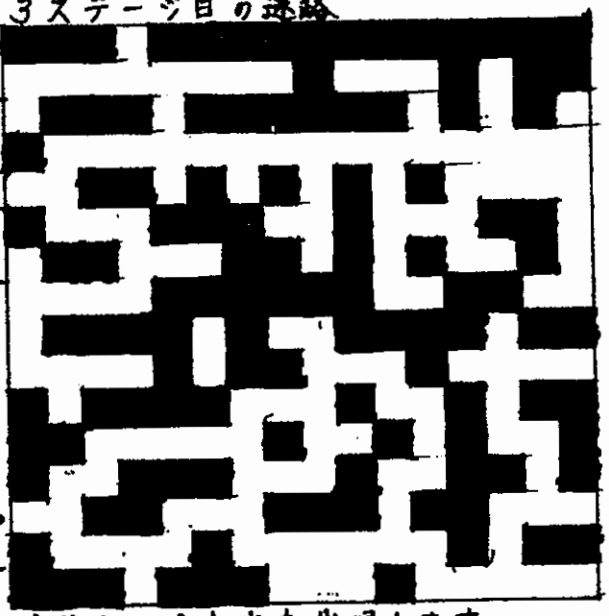
プログラム類 (変更の可能性があります)

@フローチャート



↑特記することはありません。

これを覚えてしまえば
1発クリアも夢ではありません



@3ステージ目の迷路

@プログラム ※大変長いので失礼ながら大半を省略します。

一宣言(略) マイコンは16F887, 発振は20MHzです。

・光らせる

```
void pika (unsigned int ca){
  output_a(a);
  output_c(c);
  delay_us(50);
  output_a(0);
}
void light (vc0,vc1,gn0,gn1,gn2,gn3){
 (略) ←全体を一度消します
  if (gn0!=0xff){
    pika(gn0,0x1);
  }
  if (gn1!=0xff){
    pika(gn1,0x2);
  }
  if (gn2!=0xff){
    pika(gn2,0x4);
  }
  if (gn3!=0xff){
    pika(gn3,0x8);
  }
  if (vc0!=0x0){
    pika(vc0,0x10);
  }
  if (vc1!=0x0){
    pika(vc1,0x20);
  }
```

↑回路図と照らし合わせて下さい。

・型変換

```
void kirikae(){ ←未用
  int vc0,vc1,gn0,gn1,gn2,gn3
  vc0 =(yoko&0xff);
  vc1 =((yoko&0xff00)/0x100);
  gn0 = 0xff;
  gn1 = 0xff;
  gn2 = (~tate2&0xff);
  gn3 =((~tate2&0xff00)/0x100);
  light(vc0,vc1,gn0,gn1,gn2,gn3)
}
void kirikae2(){ ←採用
 (略)
```

上記のもののgn0,gn1とgn2,gn3を交代し変数を変えました。
16ビットで保存した変数を出力用に8ビットにします。

・自機移動

```
void idou(){
  if(input(pin_b4)){
    checkertate /=2;
    delay_ms(50);
  stopper = 1;
```

自機移動は、それを少しいじっただけの物を四方向です。ルールに合わせて工夫しております。stopperは1回の入力で数回反応するのを防ぐもので、タイマで制御しています。

Ｃvsアセンブラ　～ドトマトでどちらが有利？～

実は私、11月頃まではアセンブラで書いていたのでした。しかし当時は何がおかしかったのか、自機移動さえも動かせず悩んでいた私てした。Ｃを始めるきっかけは飯田さんの言葉でした。「Ｃでやってみないか」もともと共同製作を断った僕には教えて下さらないはずでしたが、Ｃとアセンブラそれぞれでドトマトをやった飯田さんを頼りました。今その御陰で完動できたと思ってます。感謝します。ところで、Ｃで書き始めると、一旦理解すればすらすら書けるようになってアセンブラで書くとどうなるのか、と考えたことがありました。すると、案外すらすらと書けるのです。

私は気づいたのです、「アセンブラでのプログラムは複雑になりがちで自分でもその全体像が見えなくなる」と。ドトマトは表示が主に捉えられがちでアセンブラの方がいいと言われがちですが、ダイナミック点灯だけでなくそれをゲームにするとなると入力の処理やデータの管理が大変なのでフローチャートをそのまま写すように組め、16bitでもデータの管理が簡単なＣは便利だと思いました。

ただ、プログラムに馴れないうちは面倒でもアセンブラを勉強してプログラムの構造を理解することは大事だと思います。

ストーリー

見事脱走を果たした諜報部員、パニョであったがこのままでは祖国は侵攻を受け崩壊してしまう。敵の砦からの射撃は止まらないが、敵に捕まった時点で一度死んだも同然の身、パニョは祖国へと急ぐ。3つの砦を越えた先の懐しき祖国はいかに？そして突きつけられる悲しい現実とは？パニョは再び祖国のため立ち上がるのか？

感想

今回は、経験者ということで、飯田さんや伊藤さん、さらに物無の皆様には本当にお世話になりました。ありがとうございました。あと一ヶ月で文化祭ですが、このゲームをどこまで進化させられるかのスピード勝負です。いろいろ計画はあるので、1つでも多くを実現させることに時間を費したいです。

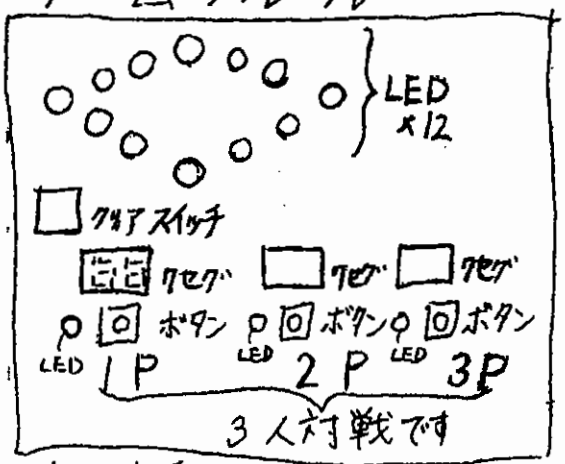
今回はドトマトをやりましたが、来年どうするかいろいろ考えています。液晶をやろうにも、話を聞くとなかなか希望者は多そうですから。私は人とは何かしら違ったものを作りたいですし、どうすればいいか迷っています。ただ、「パニョ」の3部作にすることは決めています。応援よろしくお願いします。

最後に、長文駄文　失礼致しました。
あと、手書きで読みにくくてすいません。

5963 Revolution

協力者 物無のみなさま　製作者 中2 村瀬空
回路図設計者 高1 鈴木瞬也さん

ゲームのルール



1、クリアスイッチを押してゲームをスタートします。
2、LEDが、いくつか光っていくので3、5、6、9のいずれかの個数のときボタンを押します。
3、3、5、6、9のとき1番早くボタンを押した人に1点カウントされますが、それ以外の人には点は入りません。さらに、3、5、6、9以外のときにボタンをおすと0点にもどってしまいます。
4、1番に10点をとった人の勝利です!!

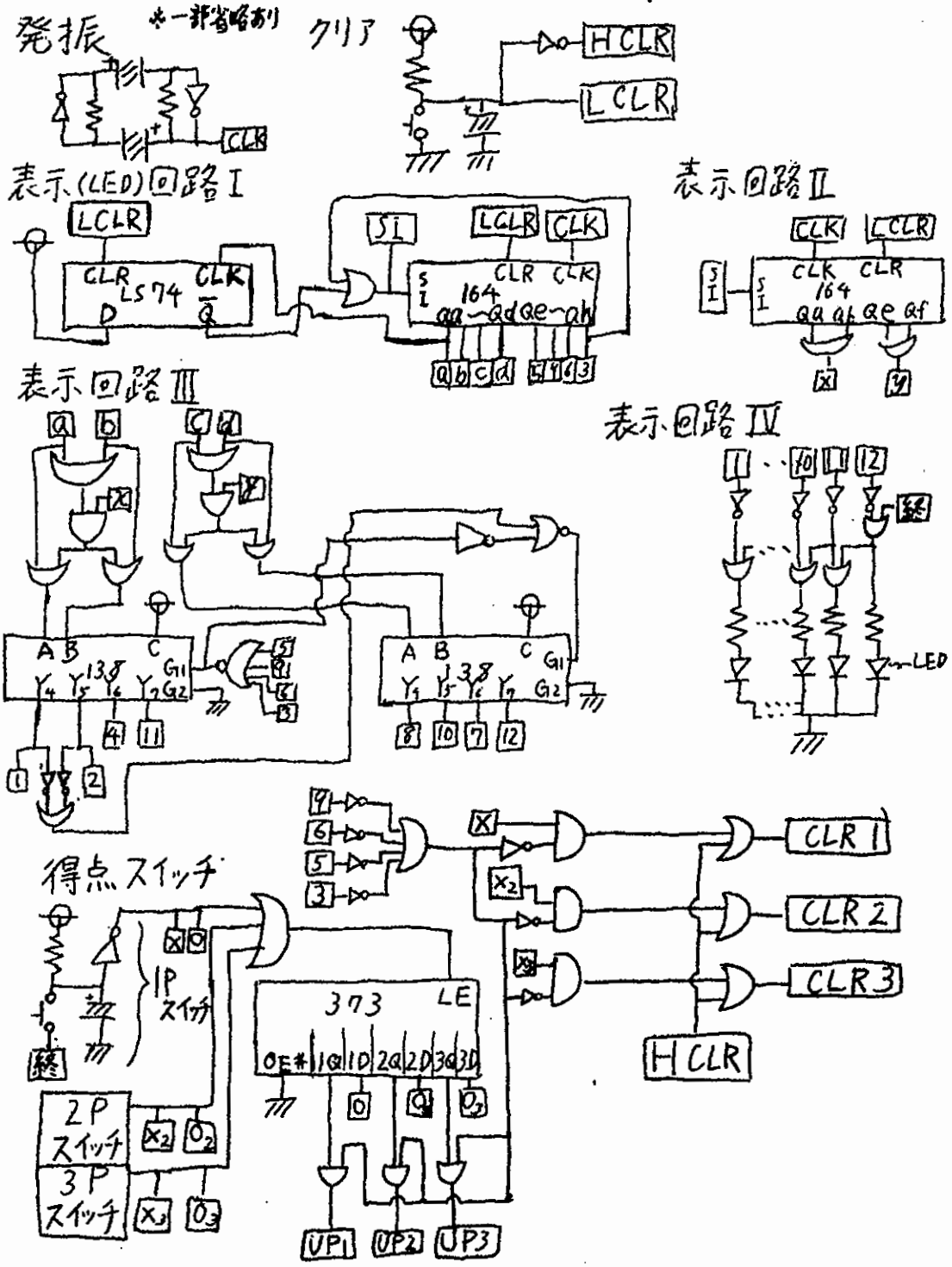
ストーリー(フィクションです)

ちょっと昔、ある町に3人の仲良し兄弟がいました。
ところが、3人はあるささいなことでケンカをしてしまいました。両親はすぐ仲なおりすると思いましたが、なかなかそうはなりません。やがて長男はだんだんさびしくなって仲なおりしようとトランプの5963(ご苦労さん)をしようとしますが、うまくいきません。そこで、彼はこのゲームをつくったのです。仲なおりできたらいいですね。

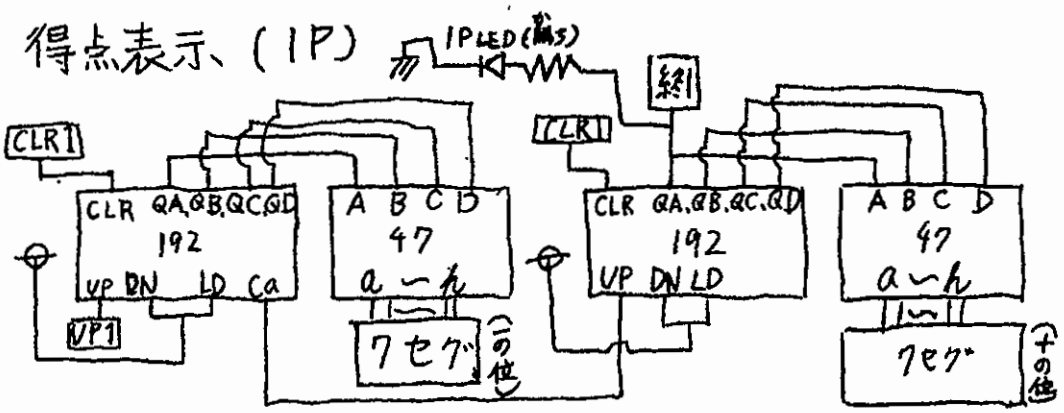
感想

バグがあったりして、そこまで楽しいゲームではないかもしれませんが、お客さんが楽しんでくれたらうれしいです。

回路図（予告なしの変更があったり、動くとは限らない 2010年3月現在 ということにご理解おねがいします。）

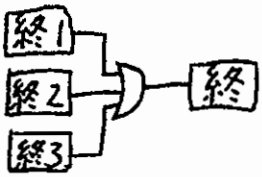


得点表示、(1P)



同様に2P、3Pもつくる。(2P … UP2、CLR2をつかう
3P … UP3、CLR3をつかう)

終わり



以上です。

反省

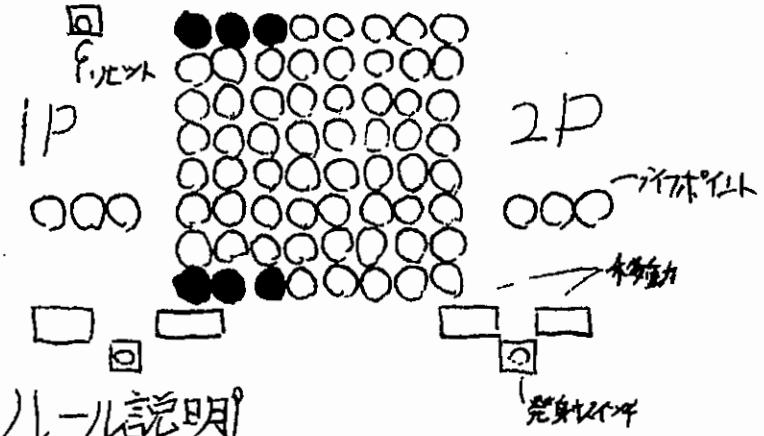
感想が少ないのでここにかきます。後書きと少し同じです。まず、よかったことはゲームがなんとか動いたことです。しかし、あまりにも配線とはんだづけがきたないので来年がんばります。しかも、この回路図通りにつくるとLEDで表示される数が8種類くらいしかないので現在、パターンを増やそうと思っていて、増やそうとがんばっています。最後に、このゲームのために協力してくださったみなさま、本当にありがとうございました。

れcity VS れcity

製作者
M2森 一斗
回路図設計者
H11伊藤 卓さん
協力
物無の皆様

外観

～IC～など



ルール説明

このゲームは二人で対戦するゲームで自分の自機から発射スイッチを押すことにより玉を発射し、相手の自機に当てライフポイントが先になくなった人が負けです。

ストーリー

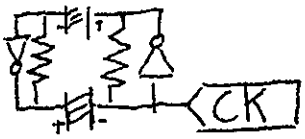
～れcityは女装いなのかもしれない～

あとれcityとkcityは実在するかもしれないし
しれないかもしれません……

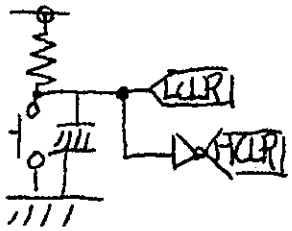
れcityは勝つことができるのか……

回路図

発振

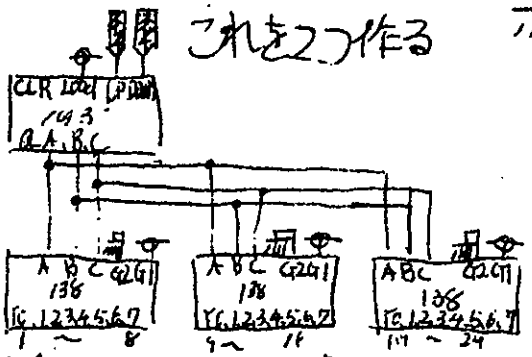


クリア回路

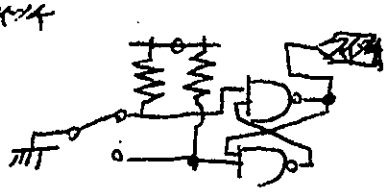


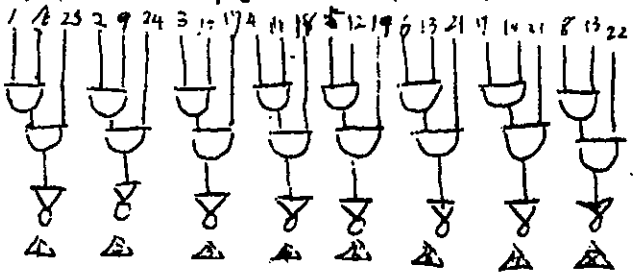
自機

これを2つ作る

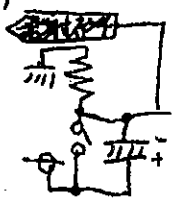


スイッチ

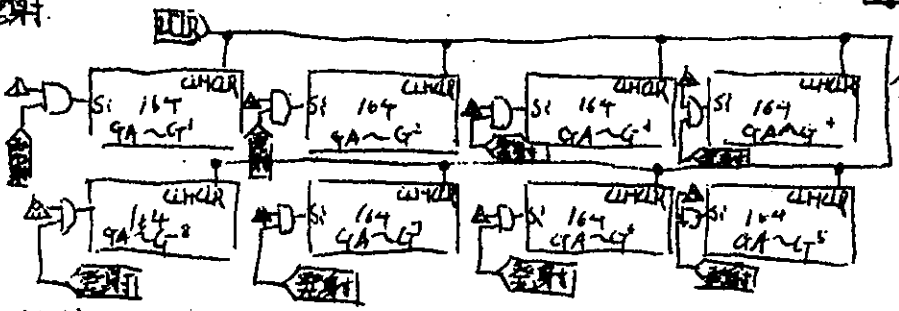




発射スイッチ

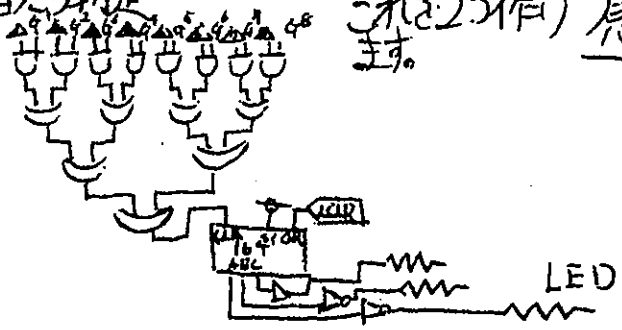


弾発射



これを2つ作る

当たり判定



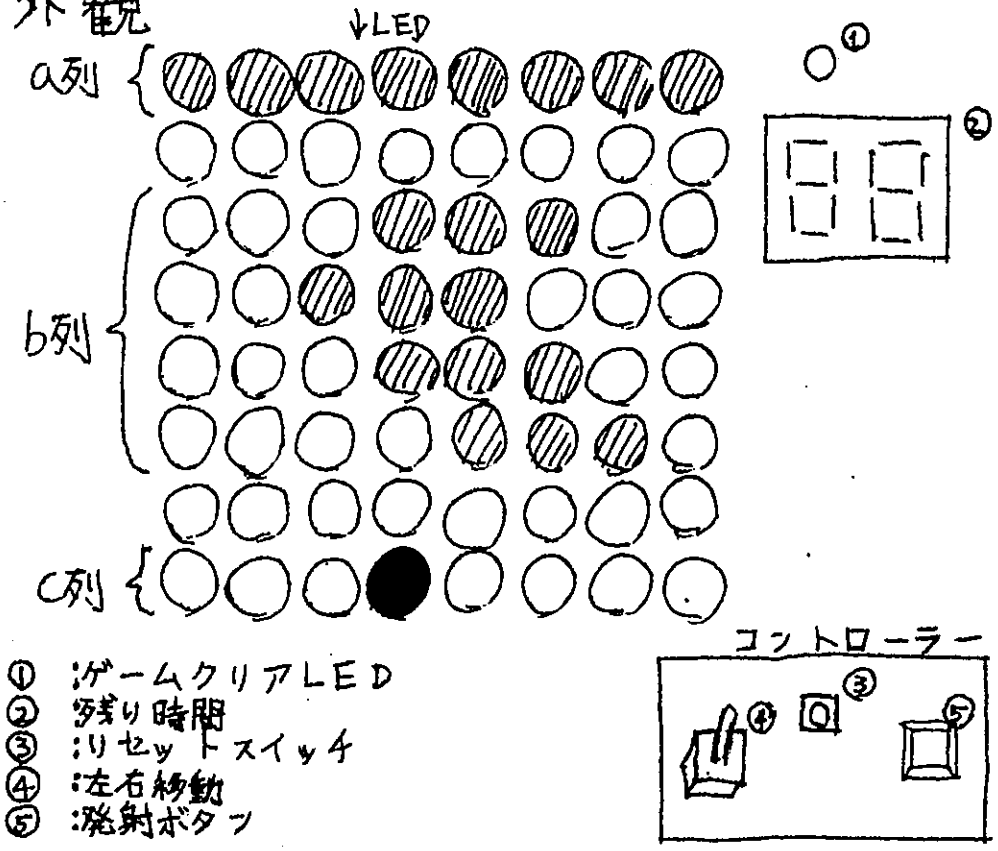
これを2つ作り感想します。

感動はしたものの、すべて配線が複雑になってしまい動いたり動かなかったりして大変でした。来年からは気をつけたいと思います。

Right Wings

製　作　者…M２　山本
回路図設備…H２　飯田さん
協　力　者…物無の皆様

1、外観

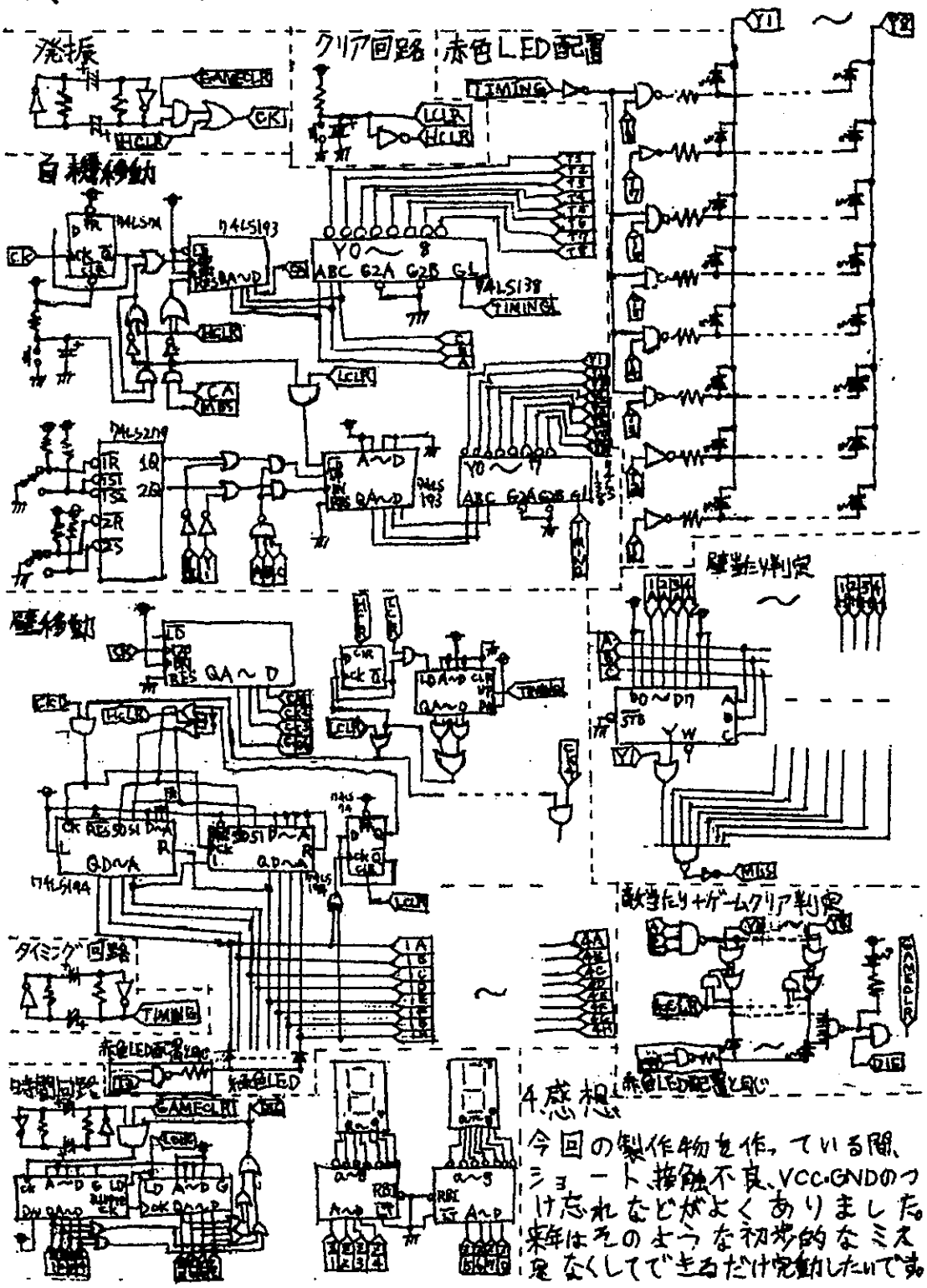


①　:ゲームクリアLED
②　:残り時間
③　:リセットスイッチ
④　:左右移動
⑤　:発射ボタン

2、ルール

発射ボタンを押すと約８秒でC列にある自機(赤)がA列に到着し、A列にある敵の戦艦(緑)に突っこみます。
その間、b列にある敵の移動するバリア(緑)に当たると自機は壊れてしまい、またC列からやりなおすことになるので、気をつけましょう。
90秒以内に敵のすべての戦艦を消せばゲームクリアです。
今、世界の平和は君の手に!!

3、回路図（この回路のICは全て74LSシリーズです。）



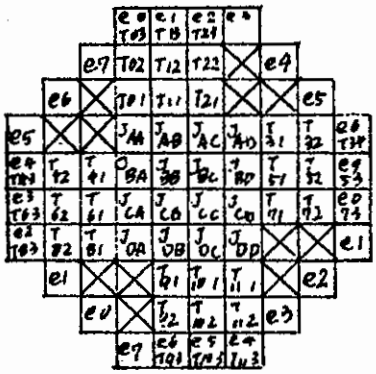
4.感想

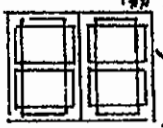
今回の製作物を作っている間、ショート、接触不良、VCC・GNDのつけ忘れなどがよくありました。来年はそのような初歩的なミスを無くしてできるだけ完動したいです。

UNKNOWN

製作：M2　岸田
回路図：H2　竹下さん
協力：　物無の皆様

☆外観





発射スイッチ

敵残機

自機移動

～LED～
J…自機
T…弾
e…敵
z…敵残

☆ストーリー

20xx年、人類は月を開拓し始めた。その時、上空から謎の飛行物体が！機体から下りてきた3体の未知の兵器は作業中の職員を捕獲しようと襲ってきた！持っていた宇宙用マシンガンで敵を破壊しろ！

☆ルール

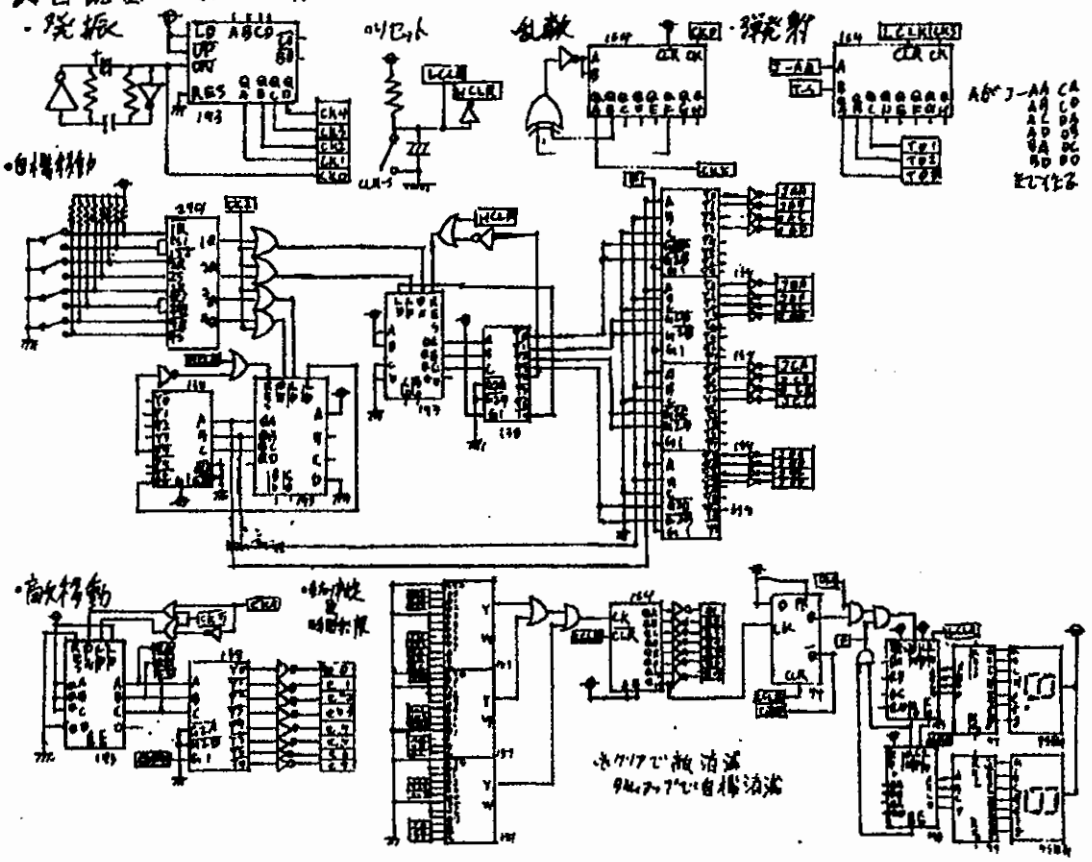
ストーリーとは予備点がありますが、気にしたら負けです。
自機(J)の範囲を動き回って周囲をランダムに回転する敵に弾を当てれば勝ちですが、残り時間に注意しながら行いましょう。
特に、表示の十の位と一の位は逆なので、仕様です。
例えば、J-AAとJ-DA間、J-BAとJ-BD間などはワープ可能です。
有効に使いましょう。

☆クリアへの近道

1. ぶっちゃけた話、3割方運です。乱数により敵が移動する為、同じ様な所を行き来する事がよくあります。そこを狙えば一瞬でEND。
2. 全体を良く見ましょう。必ずどこか狙える所に敵がいます。
   ルールの項でも言った様にワープを使うのも手です。
   自機の移動が早いので、行き過ぎに注意。
   では、健闘を祈ります。

※これを書いている時点では完動していますが、何回か回路は変更されています。よって、次ページからの回路図通りに作っても動く保証はありませんのでご注意を。
又、量が多い為にとてつもなく見にくくなっています。
飛ばしていただいても構いません

☆回路図 ～ミリ以下の世界～



☆感想

非常に見にくくてすいません。
回路が多いから出来る限り小さく書いて1枚に収めようーな人て
書文て書いたら半枚で納まりました。驚きの圧縮率です(元はA4)。
回路の方は、キレイに配線出来て良かったです。
有人まり部活に行かない不真面目な子なので、真面目に行ってれば
一番先に出来たかなぁ…と思ったりします。

ここで反省に専念すると後書きが書けなくなるのでここらで
終わります。

コイルかンの項も見てやさいね。

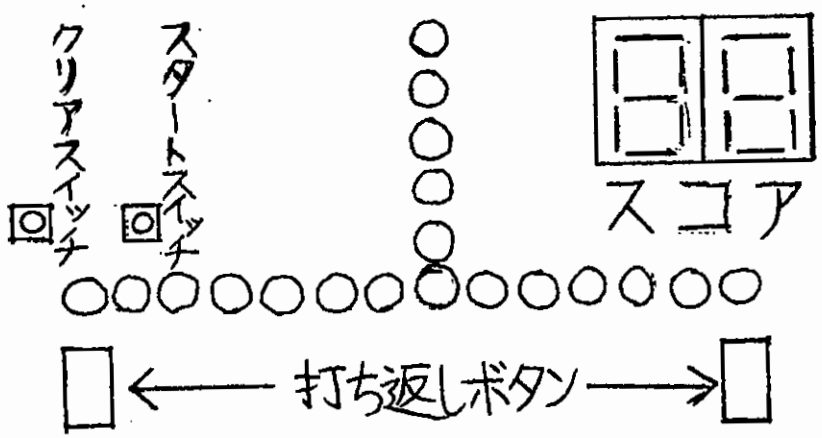
汚い字ですみません人

2002.3

ときめき
ドッジボール

製作者　M2　津田信道
回路図設計者　H1　梅本さん
協力者　物無の皆様

～外観・ルール～



クリアスイッチとスタートスイッチを押して、ゲームを始めます。ゲームを始めると上から玉が落ちてきて、一番下で左右どちらかに進みます。その玉が端まで来たら、ボタンを押してその玉を打ち返して行きます。このゲームは何秒間玉を打ち返し続けられるかというゲームです。また、玉は10秒間に1つ上から落ちて来て、時間とともに増えていきます。

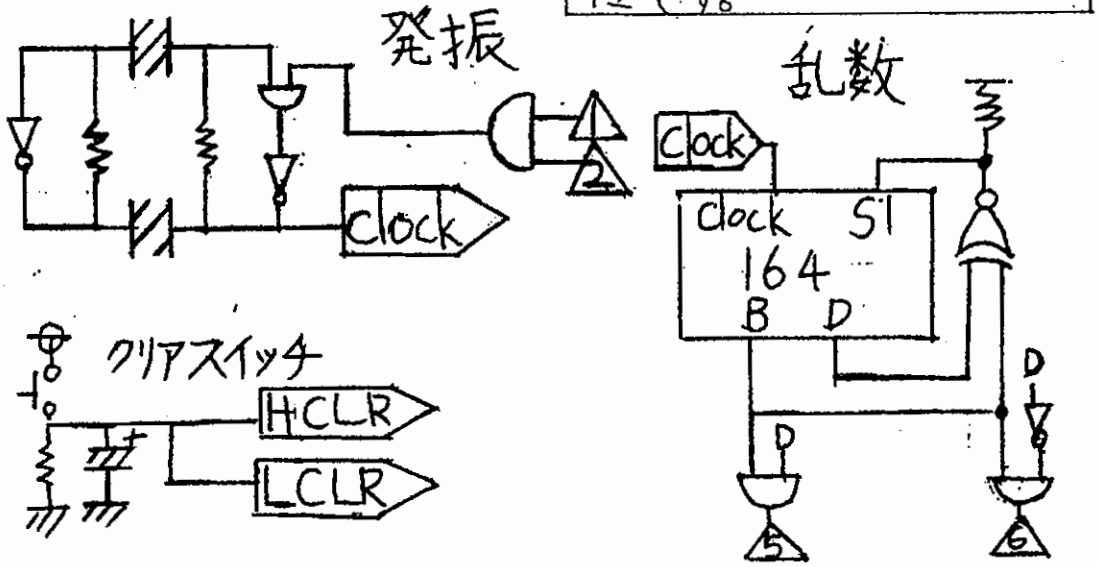
～ストーリー～

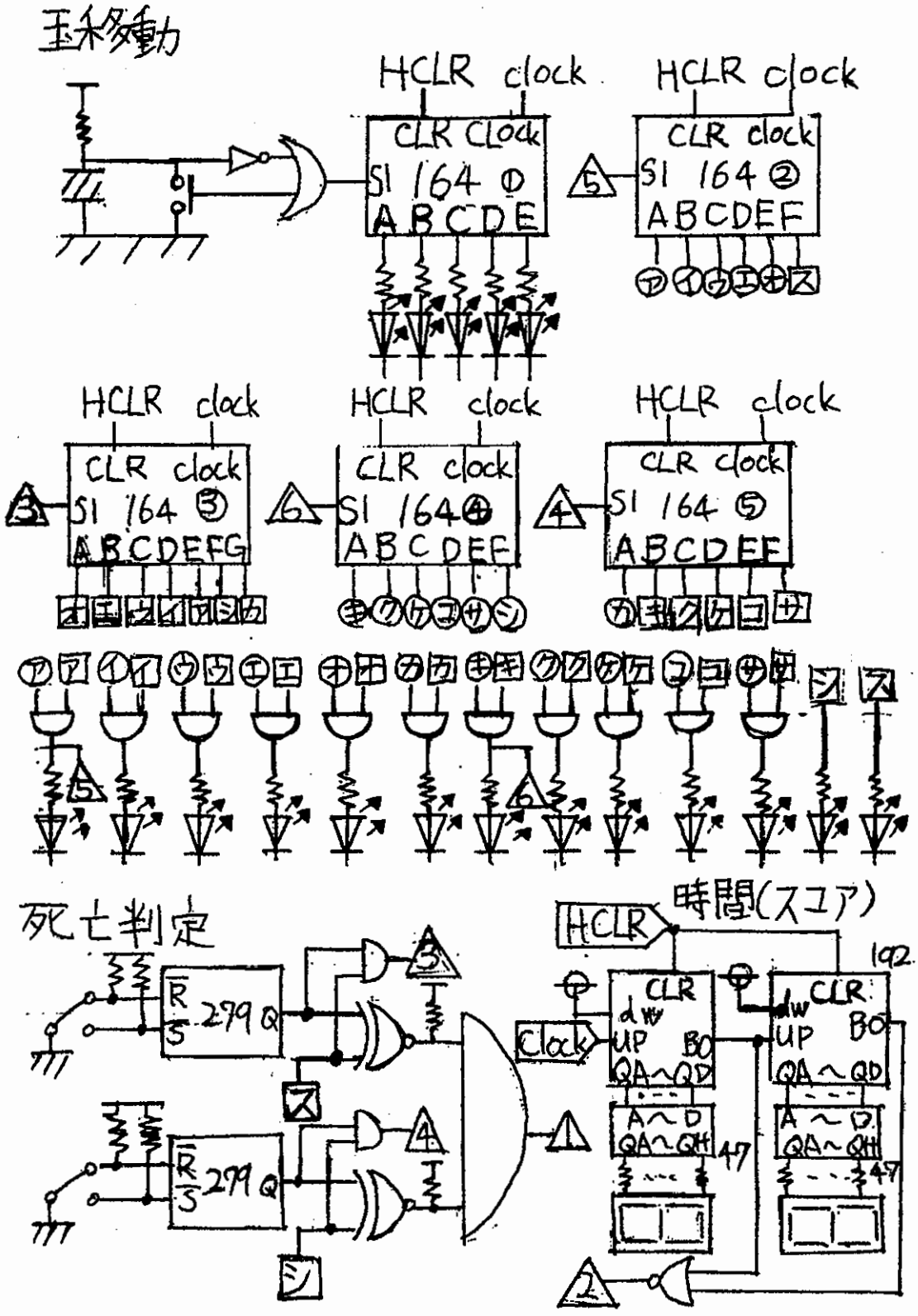
ある中学校に、茂木 流星という一人の少年がいました。その少年は、運動音痴で、いつもいじめられていました。そんなある日、その中学校でクラス対抗のドッジボール大会が行われることになったのです。その少年は、運動音痴だからどうせうまくできないとあきらめていました。しかし、少年は少しドッジボールに興味を抱き、やってみたのです。すると少年は、ものすごく速いボールを投げ、少し練習しただけで、クラスの中で一番上手になっていたのです。そしてドッジボール大会では、その少年の活躍で、クラスは優勝し、その少年はときめいたのでした。

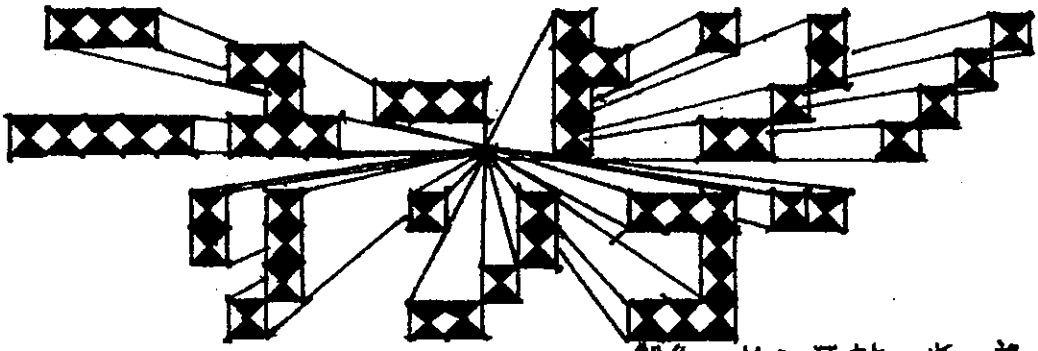
感想

はじめての電子工作は、難かしかったけど、楽しかったです。

～回路図～

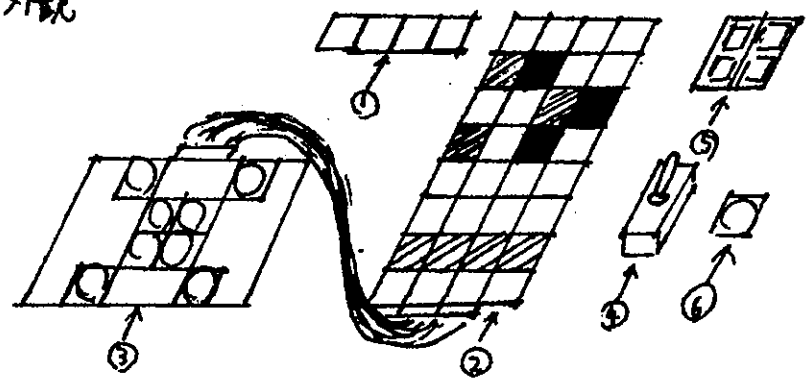



玉移動
HCLR clock
CLR CLock
SI 164 ①
A B C D E
HCLR clock
CLR clock
SI 164 ②
A B C D E F
HCLR clock
CLR clock
SI 164 ③
A B C D E F G
HCLR clock
CLR clock
SI 164 ④
A B C D E F
HCLR clock
CLR clock
SI 164 ⑤
A B C D E F
死亡判定
R
S
279
Q
時間(スコア)
HCLR
Clock
CLR
UP
DW
BO
QA~QD
A~D
QA~QH
192
47



制作　M2 河村　洋一郎
回路図設計　H1 伊藤さん
協力　物気ヶ谷様

外観

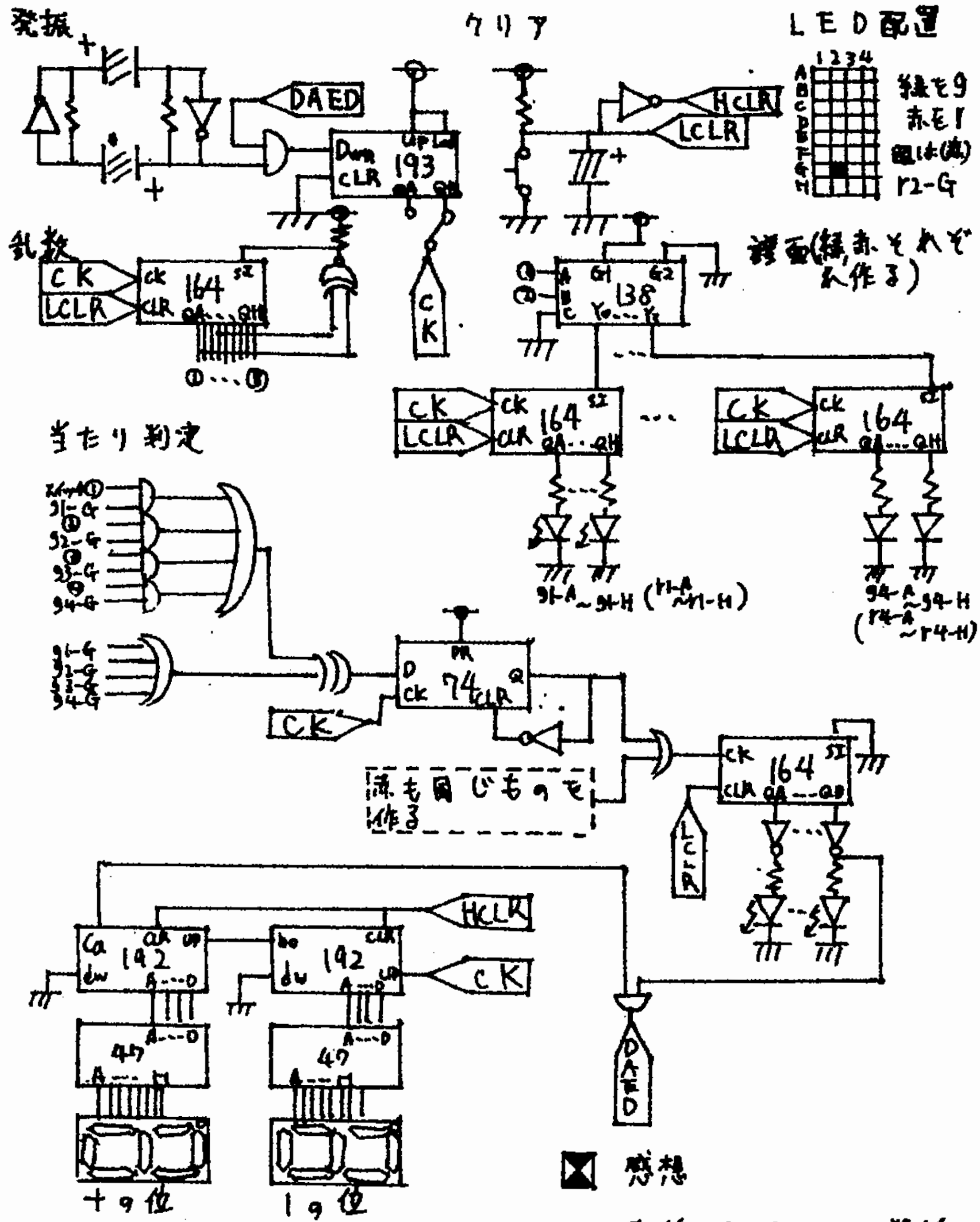


ルール

音がない音ゲーみたいなものです。②の所にリンゴが落ちて緑の玉が降ってくるので斜線の列にきたらコントローラー③の対応するボタンを押しましょう。1回ミスするとライフ⑩が1つ減ります。ライフが無くなったら終わりです。ライフが無くなるまでの時間がスコア⑨として表示されます。なお⑪でスピードが3段階に調節でき⑧でスタートします。

ストーリー

ニュートンがリンゴが落ちるのを見て万有引力を発見する話ですが今回のニュートンは少しうっかりなので何回リンゴを見ないと万有引力を発見しません。うまくリンゴを落として、万有引力を発見させましょう!!



■ 感想

配線が汚く導通や断線などが多かったですが、ICが少なかった分完動できました。来年からは正確できれいな配線を心がけたいです。

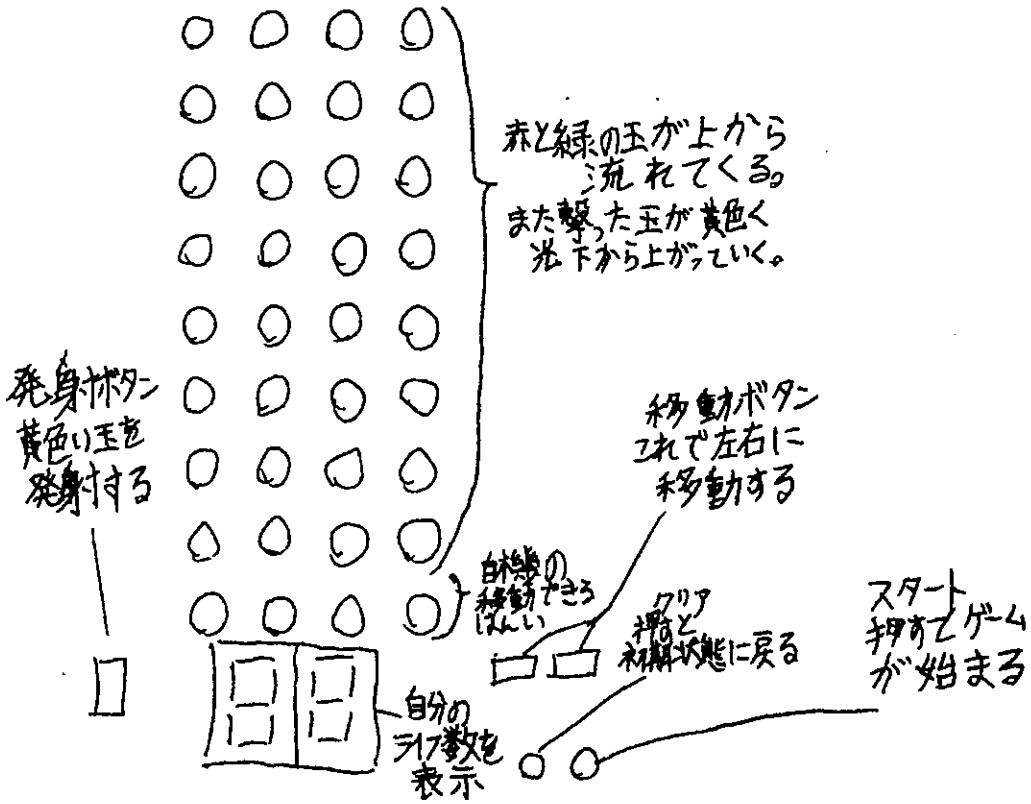
ムガル帝国Ⅲ

製作:中2水野太郎
回路図:高2池上さん
協力:物無の皆さん

ストーリー

当時は16世紀後半、イギリスの東インド会社がベンガル地方からムガル帝国に侵略を開始した。このままではインドがイギリスの植民地になってしまう。ここから逃げきるにはまず皇帝を助けださなければならない。イギリス人だけを倒し無事皇帝を助け出そう。

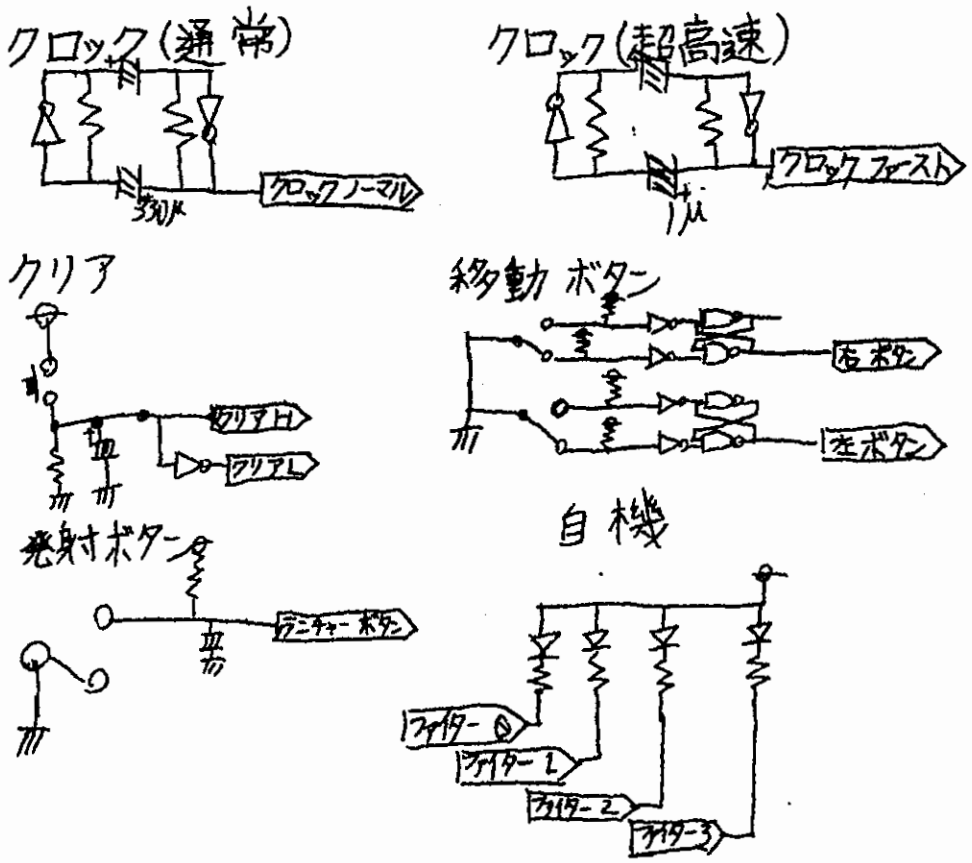
外観・操作方法



外観・操作方法(続き)

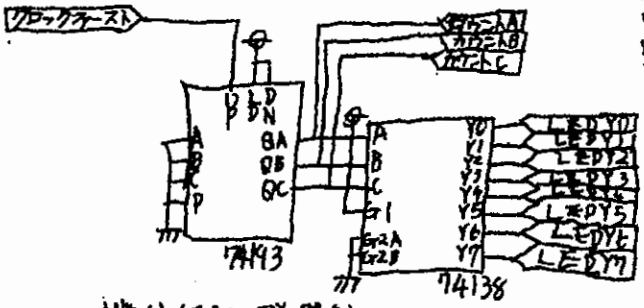
上から赤か緑の玉が流れてきます。その内赤は撃たなければなりません。逆に緑は撃ってはいけません。撃ちたい場合はその列に移動ボタンで移動し、発射ボタンで撃ってください。赤を撃つとライフが1増え、緑を撃ったり、赤を撃たなかったりするとライフが1減ります。ライフは最初10で始まり0になるとゲームオーバーになります。また10・20・30…と増えていくと玉の速さが速くなります。

回路図

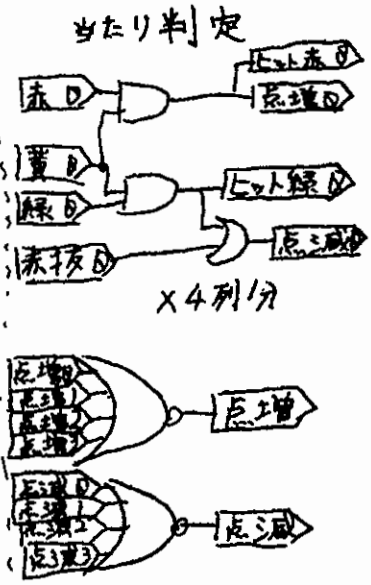


回路図（続き）

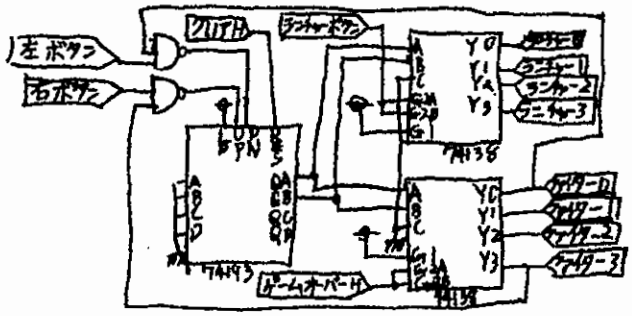
ダイナミック点灯

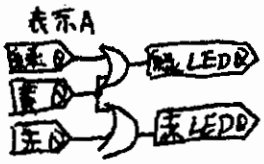


当たり判定

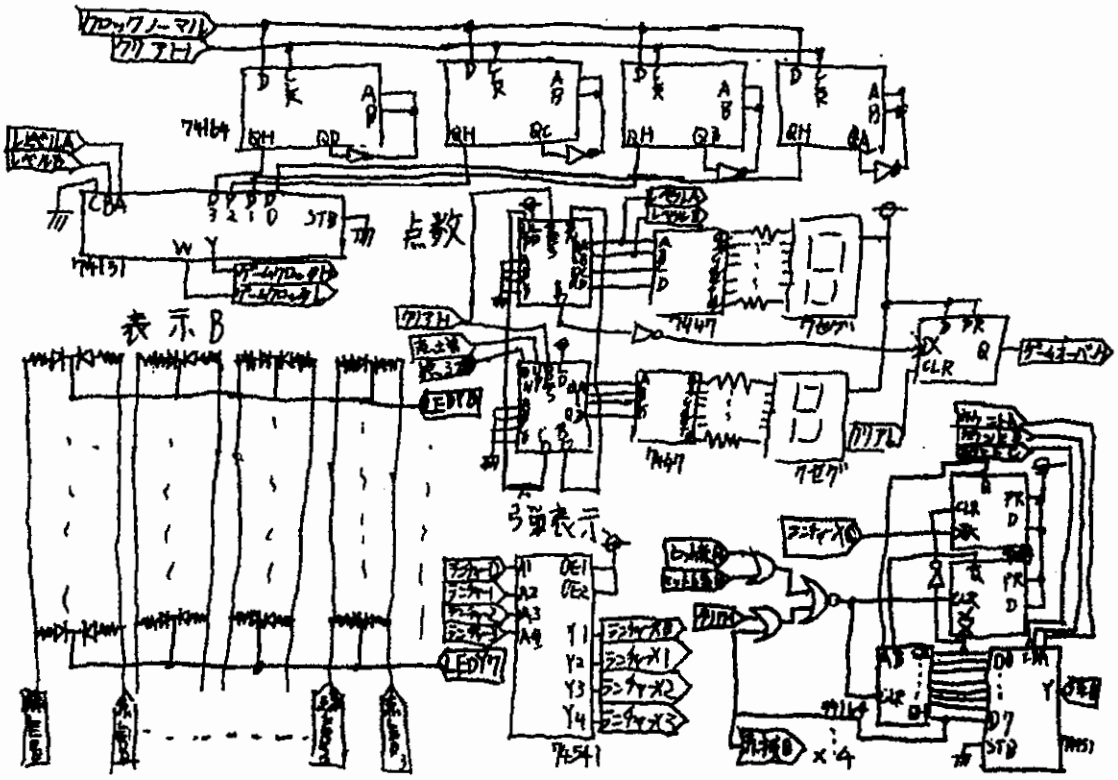


機体移動・弾発射



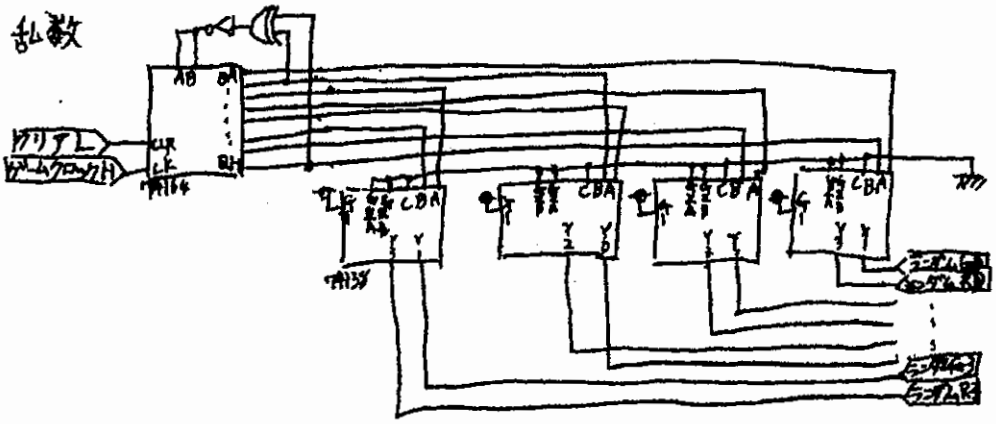


クロック速度の変更

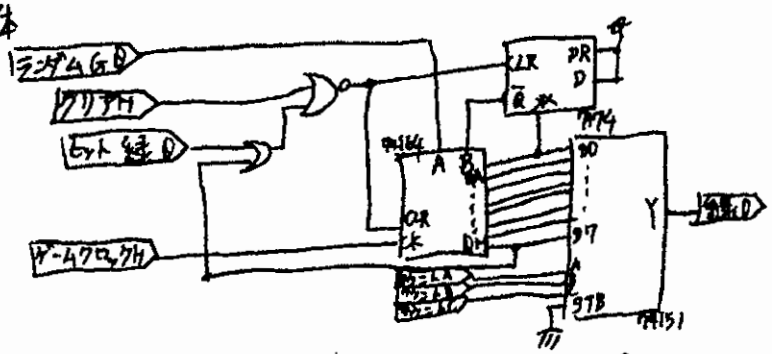


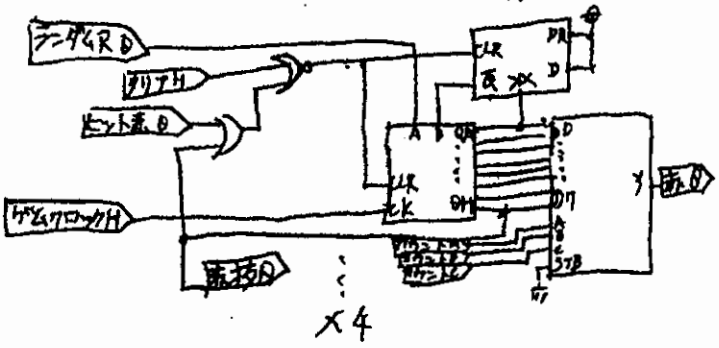
回路図(続き)

乱数



落下物体

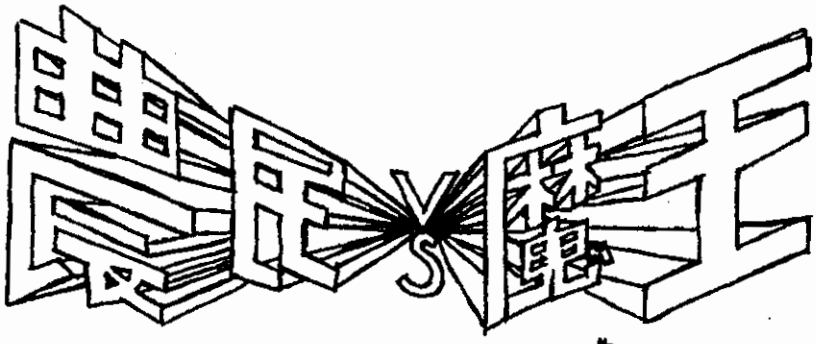




メイン

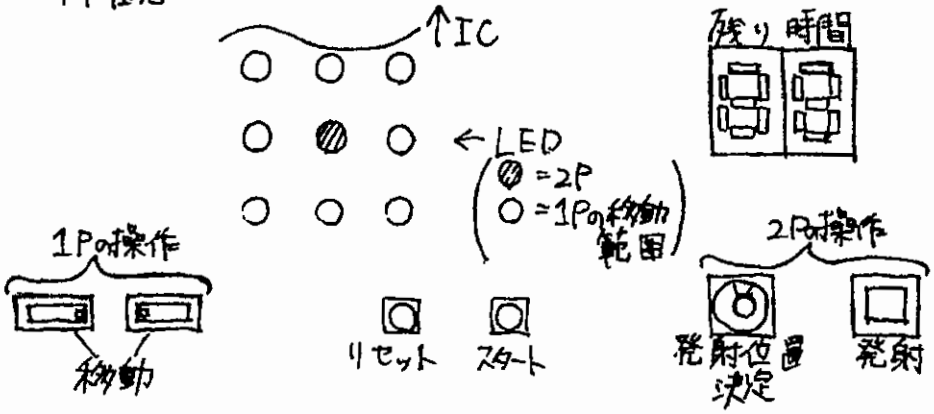
感想

回路図が雑ですいません。難かしいゲームですが楽しんでいただけたら嬉しいです。



製作・・・M2 金子
回路図設計・・・H2 棚迫さん
協力・・・物無の皆様

―外観―



―ルール―

このゲームは2Pゲーで、1P＝農民，2P＝魔王です。
2Pは左のスイッチで位置を選択し、発射ボタンを押すと、回りの8つのLEDの内、選択した一ヶ所に攻撃できます。
1Pは移動ボタンで動き、2Pの攻撃から逃げます。
残り時間がなくなるまで逃げ切れれば1Pの勝ち．
残り時間がなくなる前に1Pが攻撃を受ければ2Pの勝ちとなります。
ゲームが終わったらリセットボタンを押して、スタートボタンで再スタートです。

―感想―

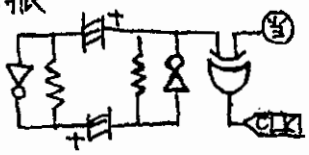
初めての製作だったのですが、先輩方に色々と教えていただき、完動することが出来ました。しかし、棚迫さんに頼りすぎてしまったと自分でも反省しているので、来年はもって自分の力で作っていけるように努力したいと思っています。

-ストーリー-

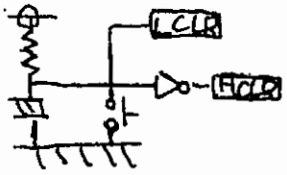
時は23世紀、世界は終焉を迎えようとしていた。
世界の始まの都市は魔王の手によって破壊され、人類は絶滅しかけていた。
もう間もなく人類は絶滅するという頃、残された人類の総力を結集した、対魔王の最終兵器が完成した。この兵器は魔王を一撃で倒せる程の威力を持っていた。
しかし、この兵器には大きな欠点があり、攻撃する直前に、99秒の準備が必要だったのだ。99秒もあれば、魔王に潰されてしまう。
全ての人が諦めかけた時、一人の男が立ち上がった。
農民だった。
その農民は、魔王の恐怖に負ける事なく、立ち向かっていった……

-回路図-

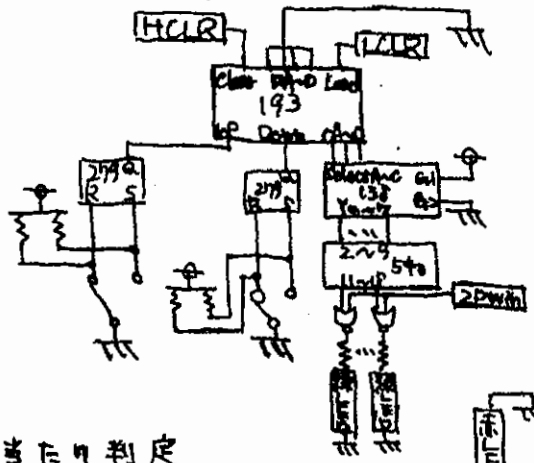
発振



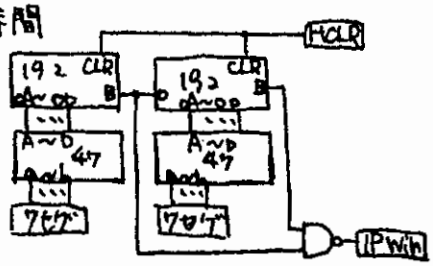
クリア



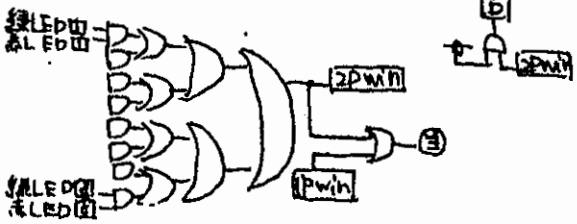
自機移動



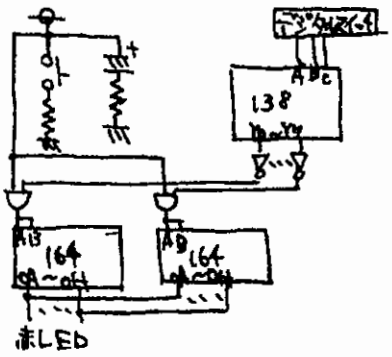
時間



当たり判定



弾発射



イライラ棒 2014

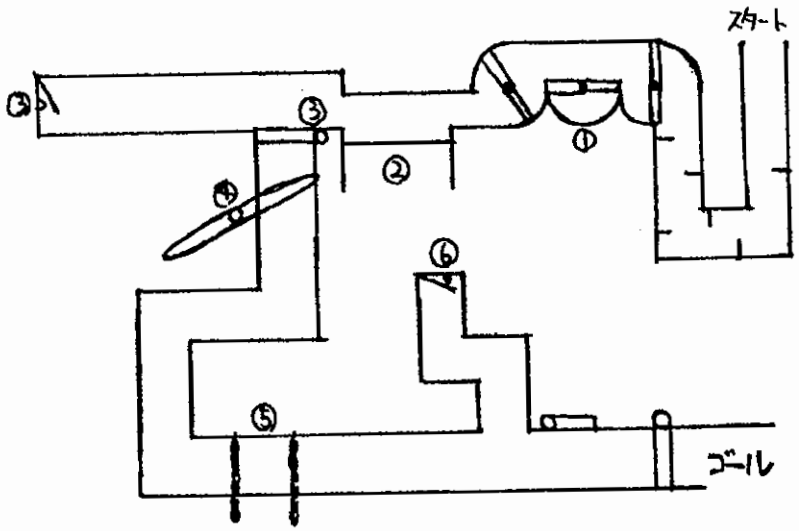
製作…回路 M2 河村
コース M2 金子

協力…物集、OBの皆様

～イライラとは～

イライラ棒とは、電極棒で触れると音と火花がでるコースやギミックをうまくよけながら、ゴールを目指すゲームです。
コースやギミックに当たったら失敗となります。

～今年のイライラ棒の外観～



今年のイライラ棒はこのようになっています。
右上のスタートから右下のゴールを目指します。

～ギミックの説明～

①直線状の金属板が三ヶ所で回転しています。
スピード、回転の向きが場所によって違ったりします。

②壁状の金属が上下しています。壁だけでなく、金属のやりのような物も上下しています。

③左のスイッチを押すと右の扉が開きます。

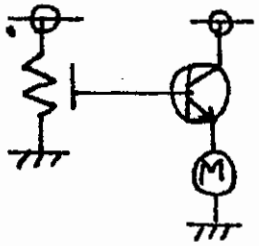
④①が巨大化した物が一つ回転しています。

⑤キャタピラについたチェーンが移動しています。
チェーンなので結構ゆれます。

⑥上のスイッチを押すと一番右の扉が開きます。
数秒経つと右から二番目の扉が開きます。

回路

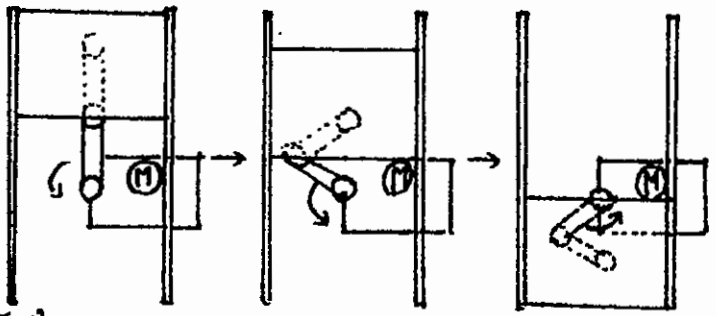
①④⑤ 回路



説明

Tアリッチ回路と呼ばれる回路です。5Vを分流して半固定抵抗で電圧を調整し,モーターの速さを変えています。

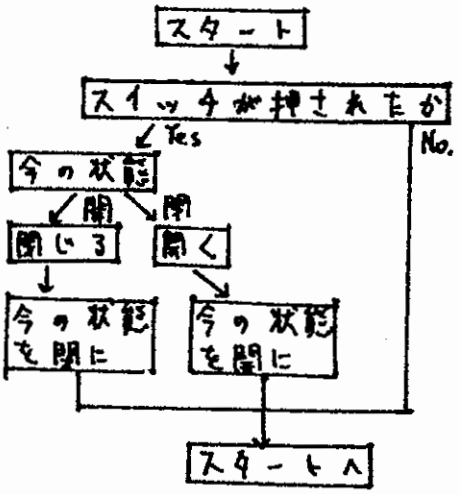
② 仕組み



説明

モーターが回るとこの字枝にはさまれた板がスライドします。

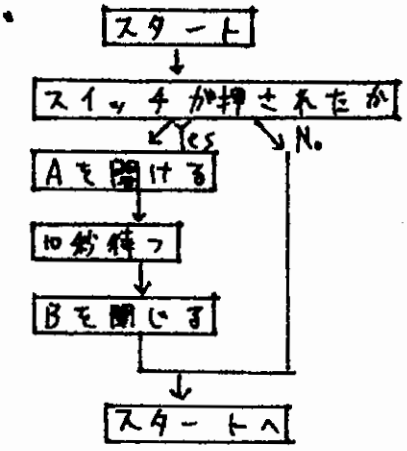
③ フローチャート



説明

PICからサーボモーターにパルスを送って扉を開閉しています。

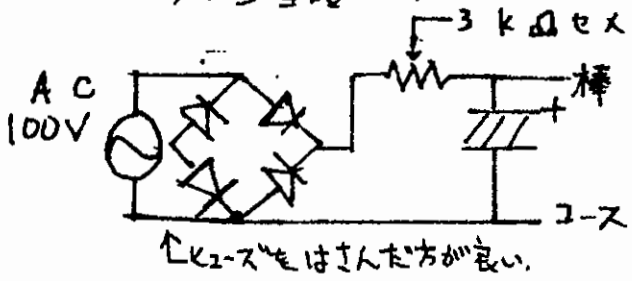
④ フローチャート



説明

PICからの出力をトランジスタを使いDCモーターにつなぎ2つのモーターを動かしています。

フラッシュ回路



説明

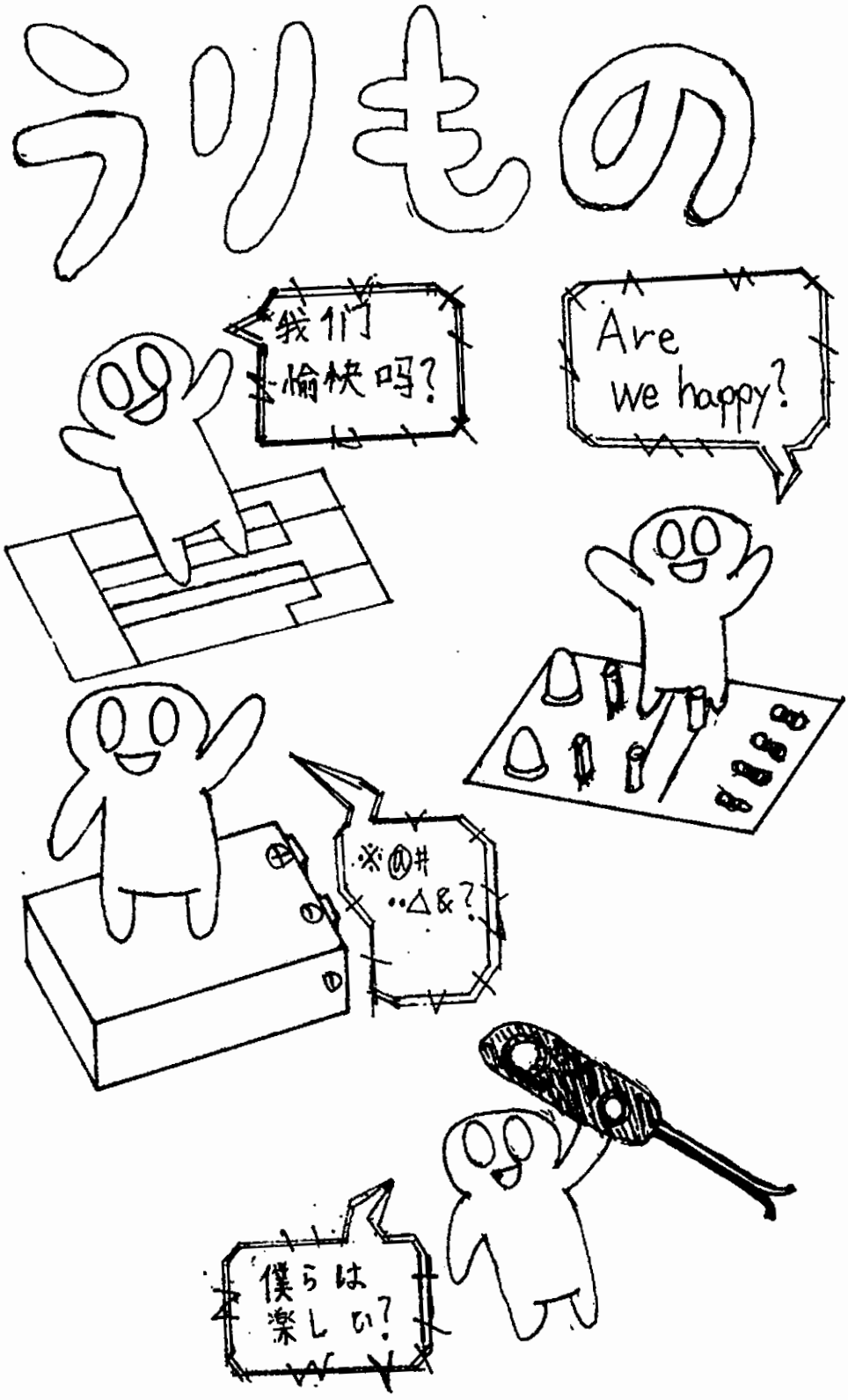
棒とコースがつなぎコンデンサーからきた電気がショートします。

感想

河村　今年のイライラ棒はPICを使ったりと、なかなか難しいですが、なんとか、だいたいのモーターは動くようになりました。最後の所はまだ実動していないので動かしたいです。

金子　河村君にはずいぶんと迷惑をかけたと思いますが、完璧までもう少しという所まで来ました。なるべく早く最後のモーターを動かしたいです。

うりもの



チカチカ～2010～

M3　大森　時生

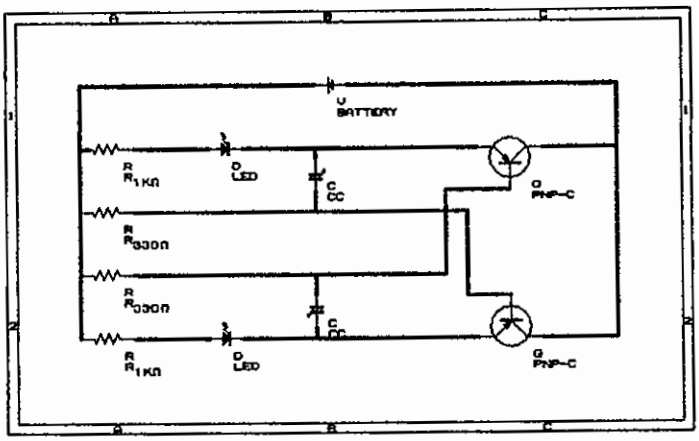
・チカチカとは？

チカチカとは、物理部無線班の展示で製作体験のできる、ＬＥＤが交互に光る製作物です。初心者でも簡単に作れることで有名です。本当の名前は「自走マルチバイブレーター」らしいです。100円で作れます。

・どんな仕組みか？

まず、電気がコンデンサーという電気がたまる部品にたまります。その後、一気にＬＥＤに放出します。それを、左右交互に繰り返すので点滅します。そのため、コンデンサーの容量を変えると点滅の速度も変わります。

・どんな回路図か？



・ストーリー

とある麻布の部活に、遠距離恋愛のカップルがいました。
その恋人は、年に３日の文化祭という場でだけ会うことができます。
しかし、実際に会ってみたものの１年のブランクは長く、心はお互いすれ違ってしまうのです・・・・・・・

超本格的ビーチバレーボウル

制作:M3　　朝倉
協力:物理部無線班の皆様

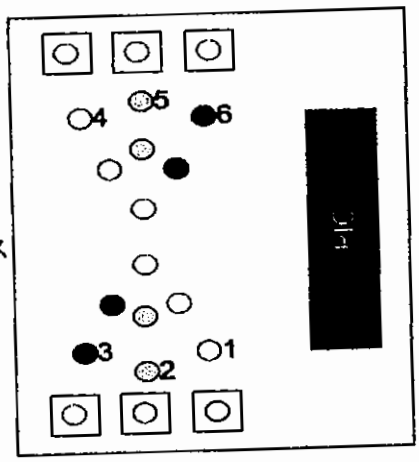
・ストーリー

”姐さん、姐さん。バレーやろうぜ、バレー!!!ｷｬﾊﾊﾊﾊﾊ”
”ヤろうヤろう!!!ｷｬﾊﾊﾊﾊ”
”食らえ、零式サーブ!!!”
”ならばこっちだって、ウォーターフール!!!”
”これは・・・、天衣無縫の極みっ!!!”
”ﾌﾀｴﾉｷﾜﾐ、ｱ――ｯ”

”はあ・・・、はあ・・・”
”ﾊｱ・・・ﾊｱ・・・”
”・・・”　”…”
”姐さん、これ、バレーじゃなくね?”
”なん・・・だとｯ!?”

END

外観



・概要

バレーゲーと見せかけて実はテニスゲーです。もちろん二人対戦です。

・遊び方

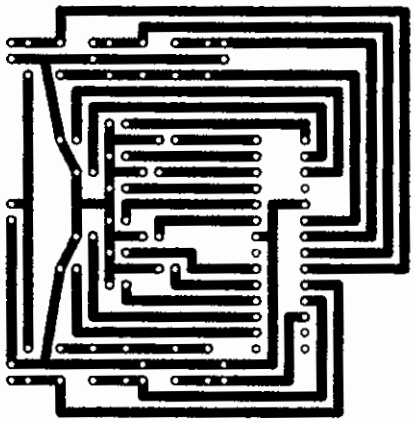
4のLEDが光ったら1,2,3どれかのボタンを押します。するとそこから玉が一発出てきますので、それをタイミングよく光ったところのボタンを押して跳ね返しましょう。

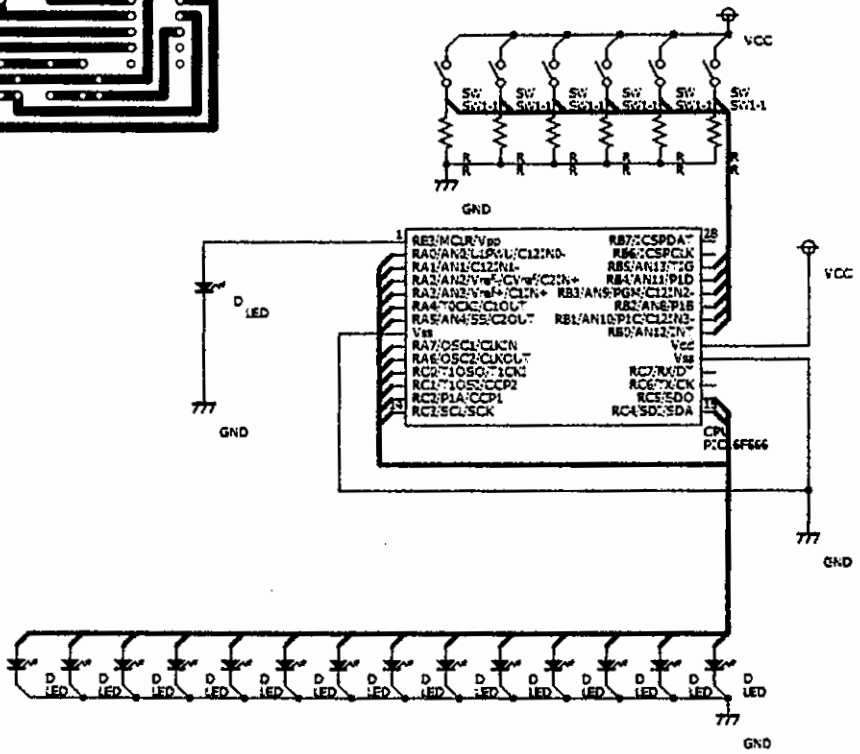
また、跳ね返した直後に曲げたい方向のボタンを押すと(例:6の方向に曲げたい…1のボタンを押す)その方向に玉が曲がります。何も押さないと直進します。

さらに、ラリーが続くと徐々に玉の速度が速くなってきます。気をつけてください。

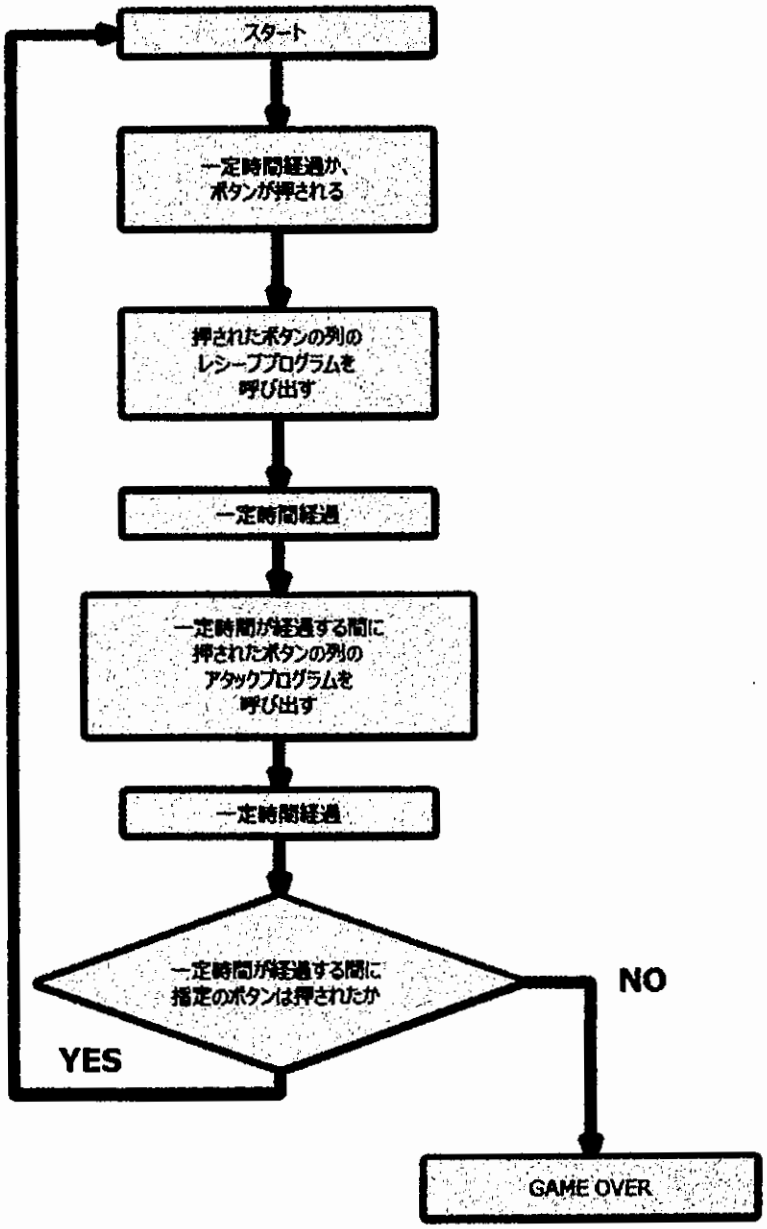
・回路図、プリントパターン

おそらく動くと思いますが、保障は出来ません。





・プログラム
長すぎるので、フローチャートを掲載します。



麻布学園物理部無線班・回路図集・2010

発行日：　2010/4/20

編集責任者：　棚辺　駿介

頒布責任者：　池上　涼平

デザイン：　池上　涼平
竹下　諒
棚辺　駿介
朝倉　哲夫
岸田　聖生
木野　太郎

＞麻布学園物理部無線班　回路図集

＞定価：100円