PICを用いたマインドストームの遠隔操作装置の開発

上村　梨紗*・森岡　弘・落合　積**・岡　正人**・白濱　弘幸***・秋本　泰宏****

Development of a Remote Control System of Mindstorms with using a PIC

KAMIMURA Risa , MORIOKA Hiroshi, OCHIAI Tsumoru, OKA Masato,
SHIRAHAMA Hiroyuki and AKIMOTO Yasuhiro
(Received July 30, 2004)

キーワード：マイクロコントローラ、PIC、遠隔操作装置、赤外線、マインドストーム、ロボットコンテスト

1．はじめに

中学校学習指導要領[1]は、ゆとりの中で生徒に豊かな人間性や自ら学び自ら考える力などの生きる力の育成を基本的なねらいとして平成10年に改訂された。

この新しい中学校学習指導要領（以下「新学習指導要領」という。）は平成12年度からの移行期間を経て平成14年度より完全実施されている[2]。各学校の技術科においては、この移行期間にさまざまな題材の開発や授業研究が行われた。

その中でも「ロボットコンテスト」は問題解決型の題材として非常に注目を集めている。ロボット製作では主体的・体験的な活動ができ、創意・工夫する場面もたくさんあることから、問題解決能力の育成につながることが期待される[3]。

筆者らはこれまでに、ロボット製作に期待されている問題解決型学習の充実のために、LEGO MindStorms Robotics Invention System（以下「マインドストーム」という。）[4)~6)]を技術科の題材として利用することを検討してきた[7)~10)]。

その中では、マインドストームを利用した授業が新学習指導要領のどこに位置付けられるかについて検討を行い[7),8)]、さらに、授業実践を通してその導入方法や授業展開について検討し、技術科の選択授業の題材として利用できることを示した[9),10)]。

授業実践を通して、マインドストームを導入した授業への生徒達の取り組みは大変積極的であることが確認できた。マインドストームは複雑な機械加工を行うことなく高度な自律型ロボットの製作が可能である。しかしながら、中学校の技術分野におけるロボコン向けの題材としては、マインドストームのような自律型のロボットより、むしろ外部から操縦可能な操縦型のロボットの方が導入しやすい場合があり、実際に授業に参加した生徒か

* (元)山口大学大学院教育学研究科技術教育専修　大学院生
** 宇部工業高等専門学校
*** 愛媛大学教育学部
**** 山口大学教育学部附属光中学校

らマインドストームを外部から制御するコントローラの開発要望が多くあった。

本研究では、これらのことをふまえて開発したマインドストームの赤外線信号の送受信機能を利用した遠隔操作（リモコン）装置について報告する。

第2章では、開発したリモコンに使用したマイクロコンピュータの一種であるPIC（Peripheral Interface Controller）について概説する[11]。第3章では、家電機器の遠隔操作に用いられている一般的な赤外線リモコンの通信方式を説明する。第4章では、マインドストーム用に開発したリモコンの製作方法を詳細に述べる。最後に第5章では、まとめを行う。

2．PIC[11)~13)]

PICとはアメリカのマイクロチップ社が開発したワンチップマイクロコンピュータのことである。もともとは、コンピュータの周辺機器の接続部分を制御するために開発されたマイクロコントローラと呼ばれる領域のICである。

プログラムで機能を実現するが、最もよく使われているPICではその命令の種類はわずか35個という簡素な構造になっているため、使いやすく1個数百円と安価である。また、開発用のソフトがフリーソフトとして入手可能であり、Windows環境で開発環境を整えることができる。非常に簡単な構成要素でコンピュータとして動作することに加え、プログラムの開発もほとんどパソコン上で行えることから、教育用教材としても注目を集めている[14),15)]。

2．1．PICファミリー

マイクロチップ社がリリースしているPICには大きく分けての表1のような3シリーズがある。

表1　PICマイコンの種類[13)]

シリーズ名	命令長	プログラムメモリ	おもなPICマイコン	命令数
ベースライン	12ビット	最大2kワード	12C509a、16C54c等	33
ミッドレンジ	14ビット	最大8kワード	16F84a、16F877等	35
ハイエンド	16ビット	最大64kワード	17C44、17C756a等	58

ベースラインシリーズは命令が12ビット幅で最初に開発されたシリーズである。ミッドレンジンシリーズは命令が14ビット幅で現在最もよく使われているシリーズである。A/D変換機能やシリアルポートなどの機能を内蔵するものもあり、種類も豊富である。今回使用するフラッシュメモリタイプのPIC16F84a[16)]もこのシリーズに属している。

ハイエンドシリーズは命令が16ビット幅で、命令数も58個あり、高機能なシリーズである。

2．2．開発環境

図1にPICプログラムの開発環境を示す。パソコンにあらかじめ開発環境ソフトのMPLABとライタソフトをインストールしておくと、プログラムの作成とデバック、プログラ

ムの書き込みをパソコン側で行うことができる。

MPLABはプログラムを作成するために必要な全てが統合された開発用ソフトでマイクロチップ社のホームページよりフリーソフトとして手に入る。

開発時に使用できる言語としてはアセンブラとC言語等とがある。今回開発したリモコンはC言語[17]を使用している。PIC用のCコンパイラはいくつかの会社から発売されているが、今回はCCS社のCコンパイラPCMを使用した。PCMはPICのミッドレンジ用のコンパイラでMPLABと統合させることが可能である。

PICライタとライタソフトはパソコンで作成したプログラムをPICに書き込むためのものである。今回は秋月電子のAKI-PICプログラマーキットVer.3を用いた。

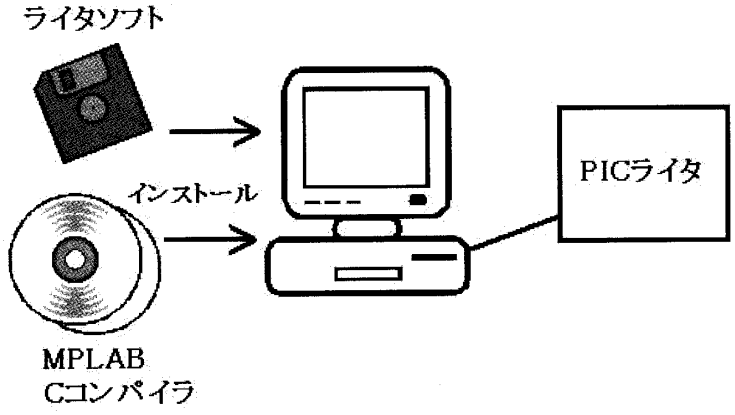

図1　PICプログラムの開発環境

3．一般的な赤外線リモコンについて

データ通信には、パラレル通信とシリアル通信の二種類がある。パラレル通信とは複数の通信回線を用いて、データを一度に複数ビット送る方法をいう。パラレル通信を行うには、ビット列をまとめて一度に送受信するための処理が必要になる。シリアル通信より高速な転送速度が得られる。

シリアル通信とは1つの通信回線を用いて1ビットずつ伝送する方法である。複数ビットのデータを解析し、1本のデータ線に0と1の繰り返しでデータを送る。パラレル伝送方式に比べて回路数が少なく低コストである。パソコンの周辺機器等で用いられるUSBにはシリアル通信が用いられている[18]。

日頃よく利用しているテレビやビデオ等のリモコンはシリアル通信であり、信号の媒体としては赤外線を使用している。家電機器の遠隔操作に用いられる赤外線データのほとんどは日本電気フォーマットまたは家電製品協会フォーマットのものである。

日本電気フォーマットは図2に示したように、リーダ・コード部、カスタム・コード部、データ・コード部で構成されており[19),20)]、これらの信号を図3に示したPPM（Pulse Position Modulation）方式により送信している。パルス部分は38kHzキャリア周波数で変調する。

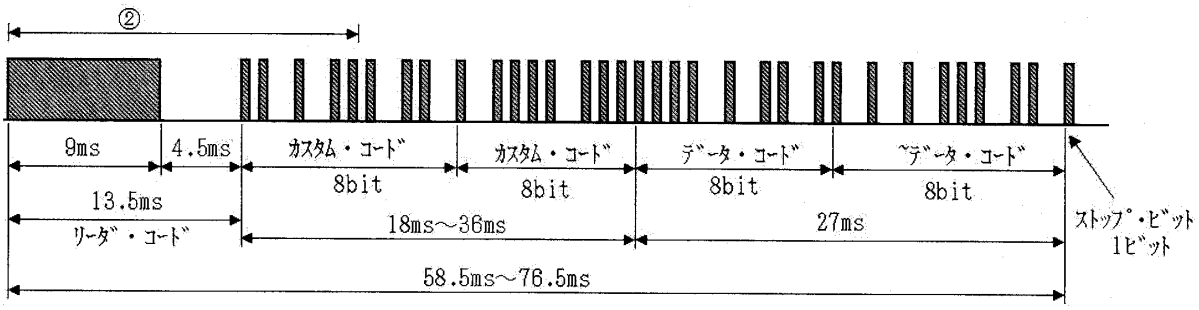

図２　日本電気フォーマットの信号

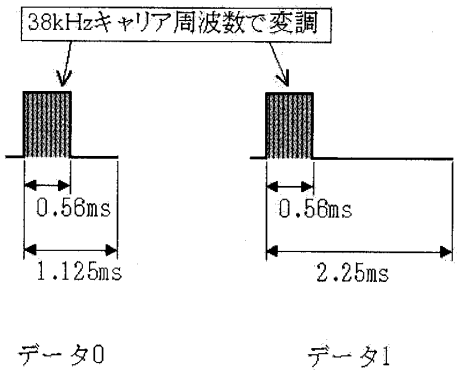

図３　PPM方式

４．マインドストームのリモコンの製作

マインドストームにはRCXのアウトプットポートの出力を制御し、モーターの回転を直接操作することのできるリモコンが販売されている。しかしこのリモコンでは、通信可能な範囲にある全てのRCXが反応してしまう。また、このリモコンを用いると、プログラミングにおける創意工夫の余地がなくなってしまう。

そこで、RCXの通信機能を利用したリモコンを製作した。RCXには赤外線通信機能があり、パソコンに接続したIRタワーからプログラムをダウンロードしたり、直接コマンドを送ったりするだけでなく、複数のRCX間で簡単な通信をすることもできる。このRCX間の通信は、１つのRCXから別のRCXに1～255の数値をメッセージとして送信するもので、受信側は受けた数値によって異なる動作をするプログラムを作成することができる。1～255の数値を細かく区切り、班ごとに使用する値を割り当てておけば、図４に示したように複数のロボットが同じ場所にあっても別々に操作することが可能になる。

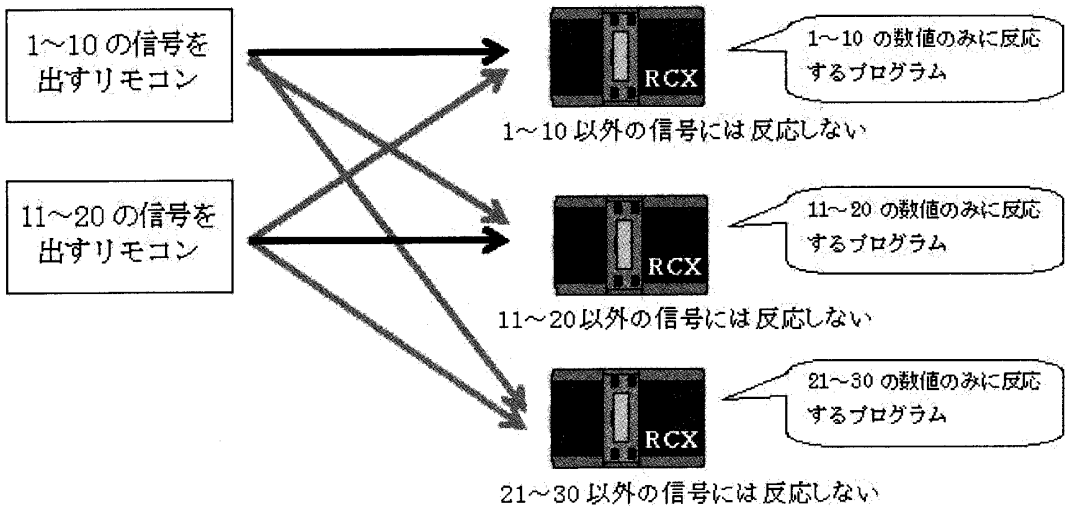

図 4　IR通信機能

4．1．RCXのデータフォーマット

RCXの赤外線は一般のリモコンと同様に38kHzに変調され、通信速度は2400bpsで使用されている。

RCXの赤外線通信の受信コードは図 5 のようになっている[21]。

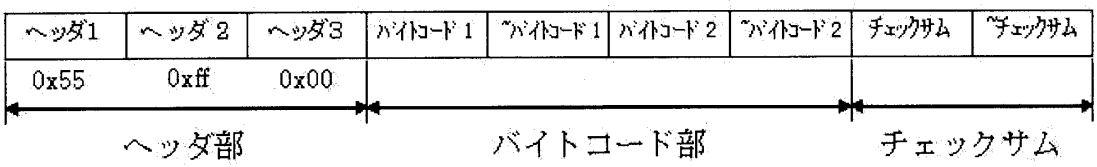

図 5　RCXの通信コード

ヘッダ部

ヘッダ 3 バイトはRCXの信号であることを示すIDコードであり、その値は0x55、0xff、0x00である[21]。

バイトコード部

RCXの通信にはコマンドに対応したバイトコードがやり取りされる。バイトコードはWeb[22]等で公開されている。今回はRCXに送られてきたメッセージを設定させるコマンド "set message" を用いる。この場合のバイトコード1は0xf7であり、バイトコード 2 には実際に引数として送る数値を入れる。

チェックサム

チェックサムは（バイトコード1＋バイトコード2）の値を送る。

これらのコードをどのように送信しているか、Web[23]上の公開された方式や、書籍[6]の情報だけでは思うように動かなかった。そこで、ロジックアナライザを使用し、実際にどのように送信されているのかを調べた。

図 6 はRCXから 6 を引数として "IRメッセージの送信" を行ったときに、受信機が受信したデータを時系列で示したものである。

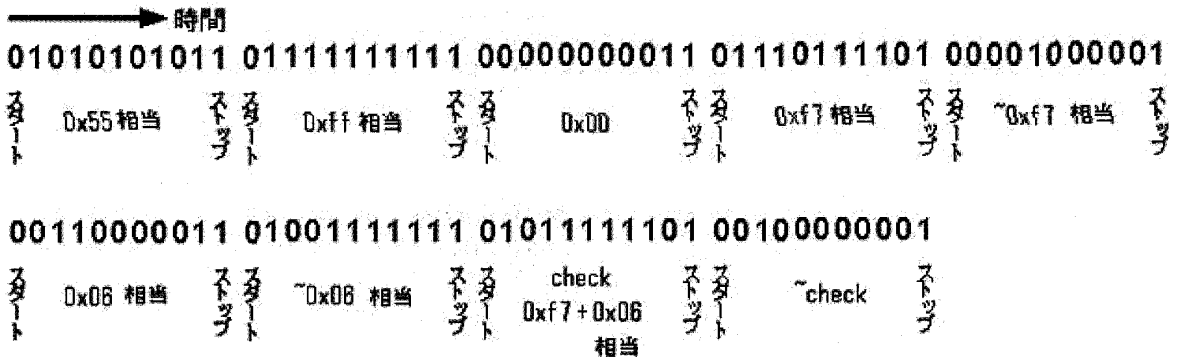

図6　数値6をメッセージとして受信した場合

その結果、RCXからの信号は一般的なリモコンとは異なり、調歩同期方式のシリアル通信フォーマットの信号で、誤り制御として垂直パリティチェック方式の奇数パリティチェックを行っていることがわかった。

調歩同期方式

調歩同期とは一定の長さのデータの前後に、データの始まりを示すスタートビット「0」と終わりを示すストップビット「1」とをつけて送信する方法をいう[24]。

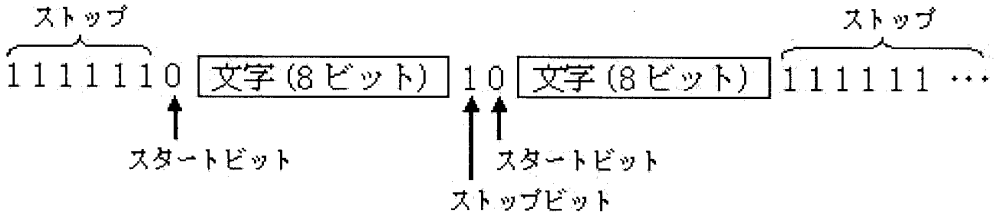

図7　非同期方式

受信側はスタートビットの「0」を検出したときにデータのサンプリングを始め、決められた数だけ（図7の場合は8ビット）サンプリングした後、最後のビットがストップビットであることを確認している。受信側がスタートビット「0」をいつでも検出できるよう無通信状態では常にストップビットの「1」を送り続けている。

誤り制御

データ通信では周波数の乱れ、外部の電磁波等の干渉、伝送路の減衰等によってデータが正しく受信側に送信されないことがある。こうした誤りはどのような対策をしても完全になくすことはできない。このため誤りをチェックし、データが正しくなければ訂正する方法をあらかじめ用意しておかなければならない。この方法を誤り制御という。誤り制御には表 2のような種類がある[18]。

表２　誤り検出方法

	再送訂正方式	誤り訂正方式
キャラクタチェック方式	**垂直パリティチェック方式** 定マーク符号チェック方式	ハミング符号チェック方式
ブロックチェック方式	水平パリティチェック方式 群計数チェック方式	CRC方式

誤り制御は検出と訂正の２つの機能が必要である。誤り検出の方法として１文字ごとにチェックするキャラクタチェック方式と、ブロックごとにチェックするブロックチェック方式とがある。また誤りを訂正する方法は送信先にデータを再送信させる再送訂正方式と、データに付加されたチェック用データをもとに誤りを訂正する誤り訂正方式とがある。RCXの信号は検出方法がキャラクタチェック方式、訂正法が再送訂正方式の垂直パリティチェック方式を用いている。

垂直パリティチェック方式とは、送信する文字ごとに検証用に1ビット付加して送信するものである。このときに付加される１ビット分をパリティビットと呼ぶ。データ１文字分にパリティビットを含めた９ビットについて「1」の数を偶数（または奇数）にしておき、受信側でその数をチェックし、偶数（または奇数）になっていなければそのデータは誤りだとする方法である。「1」の数を偶数にして送信する方法を偶数パリティチェック、奇数にして送信する方法を奇数パリティチェックという。

RCXの信号（図６）はスタートビットとストップビットの間に９ビットのデータがある。１文字は８ビットなのでストップビットの直前の１ビットはパリティビットということになる。このパリティビットをあわせた９ビットのうち「1」の数は常に奇数になるようパリティビットで調節されている。よってRCXの誤り検出方法は奇数パリティチェックである。

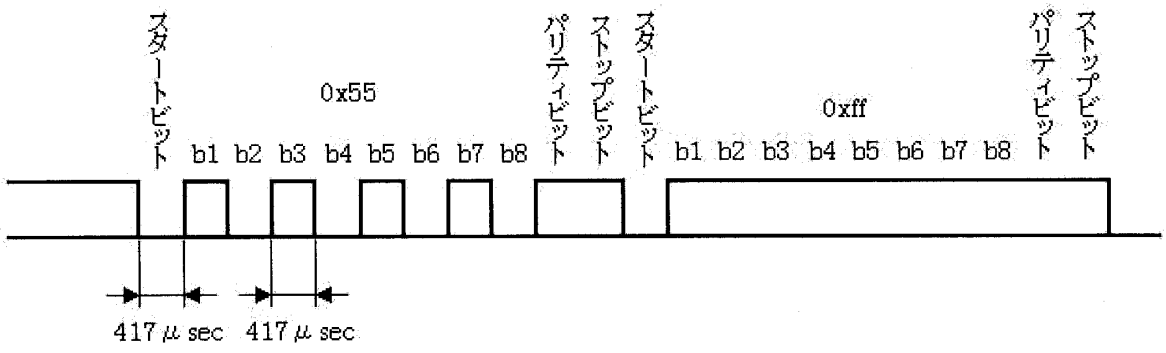

図８　RCX通信の受信側の信号

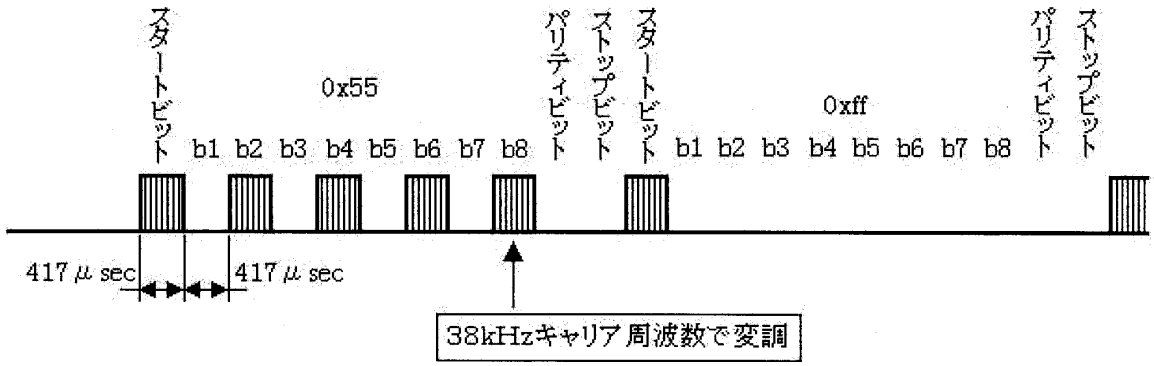

図9　RCX通信の送信側の信号

図8は受信した信号である。RCXの通信速度は2400bps、つまり1秒間に2400ビットの速さであるから、1ビット送るのにかかる時間は1÷2400=0.000416666…≒417μsecとなる。

0x06を送る場合、送信順序はまず、スタートビットとして「0」を送信する。次に0x06を1ビット目から順に送る。0x06は00000110であるから、右端のビットから、01100000と送信する。そして、この1バイトの中で「1」の数が偶数個であれば「1」を、奇数個であれば「0」をパリティビットとして送る。この場合は「1」の数が偶数であるから、パリティビットとして「1」を送る。その後にストップビット「1」を送信し、次の1バイトを送信するためのスタートビットを送る。

受信部は送信部から出力されたキャリアを検出し、キャリア部分を波形形成した信号を出力しているので、送信信号は図9のようになる。データ「0」を送信したいときは38kHzキャリア周波数に変調した信号を417μsec送り、データ「1」を送信したいときは417μsecの間何もしない。すなわち、受信側と送信側は信号が反転している。4.1に示したRCXのデータフォーマットは受信側を基準にしており、実際に信号を送信する場合には反転データを用いることに注意しなければならない。

4．2．リモコンの製作

リモコンの回路図とパーツリストを、それぞれ図10と表3に示す。PICは16F84aを用い、赤外線発光ダイオードはTLN105Bを使用した。赤外線発光ダイオードTLN105Bには100mAのパルス電流が必要だが、PICの入出力ピンは25mAの電流しか流せない。そのためRA1～4までの4個のピンを並列で利用し100Ωの抵抗を通してダイオードにつなげている。

PIC16F84aに必要な供給電源は3.0V以上であるが、クロックを20MHzにするには4.0V以上必要になる。そのため、リモコンの電源は単4の電池3個を用いて4.5Vとした。

PICは16F84aを用いた。RAポートは出力ポートとして、RBポートはスイッチからの情報を読み取る入力ポートとして使用した。入力ポートはRB4～7までの4つのピンが残っているのでスイッチを（トグルスイッチならばあと2つ）追加することも可能である。

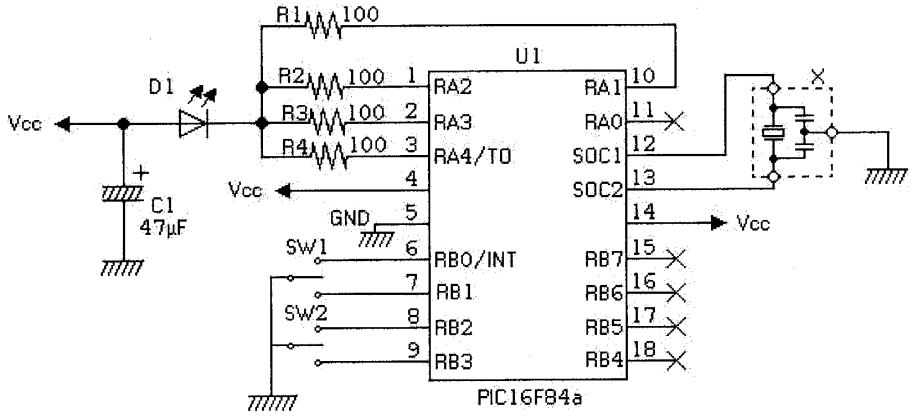

図10　リモコンの回路図

表3　パーツリスト

記号	パーツリスト	値・型名	個数	単価(目安)
U1	IC	PIC16F84a	1	350
D1	赤外線発光ダイオード	TNL105B	1	80
R1、2、3、4	抵抗	100Ω 1/4W	4	10
C1	電解コンデンサ	47μF 16V	1	55
X	セラミック振動子 (コンデンサ内蔵タイプ)	20MH	1	40
SW1、2	トグルスイッチ（はね返り）	MS-500EB	2	175
	ICソケット	18ピンDIP	1	50
	基板	ICB-93S	1	295
	電池ボックス	単4×2個用	1	90
	電池ボックス	単4×1個用	1	70
	電池	単4	3	100
	ケース	タッパー		100
	ねじ等		少々	少々
合　計				1820

トグルスイッチは図11に示すように、左右に一個ずつ上下に動くように取り付けた。左のスイッチを上に押すとRB0ポートとつながり、下に押すとRB1とつながる。そして、右のスイッチを上にするとRB2とつながり、下にするとRB3とつながるようにする。スイッチの入力にあわせたデータを送信し、1つのリモコンで表4のように8通りの数値を送信できることになる。

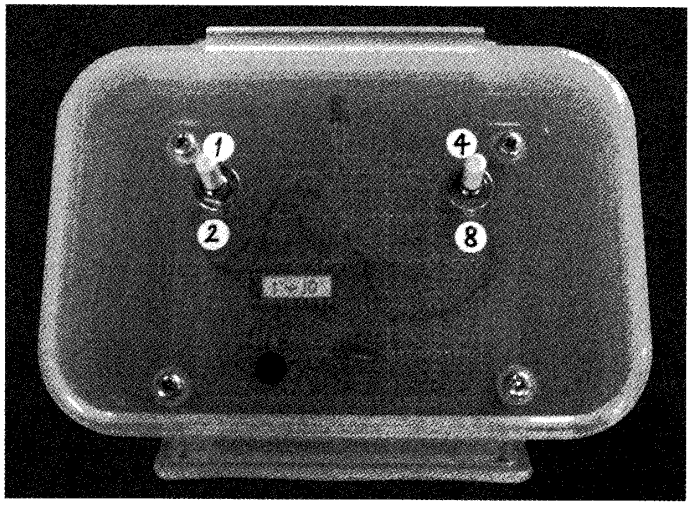

図11　リモコン完成品

表４　スイッチの状態と送信データ

		左スイッチ		
		上	中	下
右スイッチ	上	5 0101	4 0100	6 0110
	中	1 0001	送信なし	2 0010
	下	9 1001	8 1000	10 1010

※表のデータ＋リモコンナンバーが実際の送信データ
※「中」はスイッチを入力していない状態

リモコンのケースにはプラスチック製の透明なタッパーを使用した。生徒たちにリモコンや電気回路について興味を持たせることができるよう、外側から内部が見えるよう透明なケースを用いた。

４．３．リモコンのプログラム

プログラムは、図12のように、まず初期化で入出力ピンのモード設定と、ヘッダ、バイトコード1の値の設定とを行い、次にスイッチ入力のチェックをする。スイッチ入力があれば、スイッチから得られた値をバイトコード2として保存した上でヘッダから順に送信する。送信後は0.1秒待った後スイッチ入力チェックに戻る。サンプルプログラムを資料１に示す。

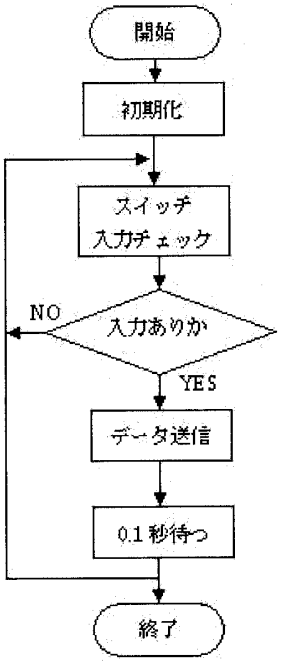

図12　プログラムの流れ

4．4．RCX側のプログラム

製作したリモコンを利用するためのRCX側のプログラムについて述べる。リモコンでのデータの送信は、スモールブロックビンの通信の中のIRメッセージの送信と同じである。よって、RCX側はセンサーのIRメッセージを使用することで、メッセージごとに異なる動きをするプログラムを簡単に作成することができる。

その一例として、はい、またはいいえコマンドを使用する方法を示す。はい、またはいいえコマンドの変数タブをクリックし、チェックするセンサーをIRメッセージに設定し、その後送られてくる数値を選ぶ。図13の例では、もしIRメッセージが11ならば「はい」の下に接続されたコマンドの動きをし、11でないならば（メッセージがなければ）「いいえ」の下に接続されたコマンドに従うようになっている。

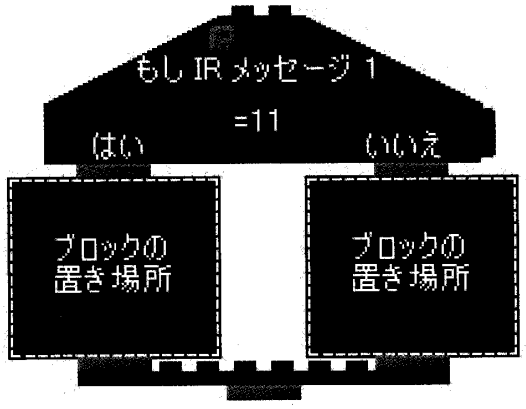

図13　はい、またはいいえコマンドの利用

5．まとめ

開発したリモコンを導入することにより、マインドストームを使用した操縦型のロボットの製作が可能となった。これにより、中学校の技術科のロボコン教材として、マインドストームを利用することが容易になった。さらに生徒のロボット製作における選択の幅が広がり、創意工夫の場面がよりいっそう増えることが予想できる。

開発したリモコンは、同じ場所で同時に使用しても混信することなく、その操作性はおおむね好評であった。今後は、リモコンからの操作のバリエーションを増やすためトグルスイッチを追加することを予定している。また同時に、指向性や到達距離の異なる赤外線発光ダイオードの使用を検討することにより、リモコンの操作性の向上を目指す予定である。

最後に本研究を進めるにあたり平成15年度山口大学教育学部、学部・附属共同研究として支援を得たことに記して謝意を表す。

参考文献

1）文部省：中学校学習指導要領（平成10年12月）解説—総則編—，1999
2）文部省：中学校学習指導要領（平成10年12月）解説—技術・家庭編—，1999
3）小川　智司，森　慎之助　他：ロボット技術を題材としたプログラミング・制御学習—マインドストームの活用とその効果—，日本産業技術教育学会第44回全国大会講演要旨集，p.13，2001
4）Paul Wallich：Mindstorms not just a Kid's Toy，IEEE Spectrum September ，pp.52-57，2001
5）S・パパート（奥村　喜世子訳），マインドストーム，未来社，1982
6）Jin Sato，白川裕記，牧瀬哲郎，倉林大輔，衛藤仁郎：LEGO MindStorms パーフェクトガイド，翔泳社，1999
7）森岡　弘，上村　梨紗，秋本　泰宏：マインドストームを利用した技術科教育方法に関する研究，日本産業技術教育学会第46回全国大会講演要旨集，p.57，2003
8）森岡　弘，上村　梨紗，秋本　泰宏，岡　正人：マインドストームを利用した技術科の教育方法に関する研究，山口大学教育学部・附属教育実践総合センター研究紀要第17号，pp.35-50,2004
9）森岡　弘，上村　梨紗，秋本　泰宏，岡　正人：マインドストームを使用した技術科の授業実践報告，山口大学教育学部・附属教育実践研究紀要第3号，pp.151-160,2004
10）森岡　弘，上村　梨紗，秋本　泰宏：マインドストームを使用した技術科の授業実践報告１(ロボコン形式の授業に対する目標準拠評価)，日本産業技術教育学会中国支部第33回大会講演要旨集，p.5，2004
11）後閑哲也：電子工作のためのPIC活用ガイドブック、株式会社技術評論社、2000
12）後閑哲也：たのしくできるPIC電子工作、東京電機大学出版局、1999
13）中尾真治：RoboBooks はじめてのPICマイコン、オーム社、2001
14）平田竜美、山本透、上田邦夫：マイクロコントローラーを用いた学習材の開発、日本産業技術教育学会中国支部第30回大会講演要旨集、p.10、2001
15）川田和男、藤澤正一郎、山本透：教材化のためのマイクロコントローラー（PIC）と

その制御用プログラム、日本産業技術教育学会誌第45巻3号、pp.157～167、2003
16) http://www.microchip.co.jp/30430c-j2.pdf
17) 後閑哲也：C言語によるPICプログラム入門
18) 竹上慶、藤澤一郎：図解入門　よくわかる　最新データ通信の基本と仕組み、株式会社秀和システム、2002
19) トランジスタ技術11、株式会社アバールデータ、1996
20) http://www.necel.com/nesdis/image/U14380JJ3V0DS00.pdf
21) http://pitecan.com/UnixMagazine/PDF/if9812.pdf
22) http://graphics.stanford.edu/~kekoa/RCX/opcodes.html#set_message
23) http://graphics.stanford.edu/~kekoa/rcx/
24) 朝日新聞社：パソコン・データ通信　プロトコル・ハンドブック、朝日新聞社、1989
25) 上村　梨紗：LEGO Mindstorms　を使用した技術科教育方法に関する研究、山口大学大学院教育学研究科修士論文、2004

資料１　リモコンのサンプルプログラム

```
1   #include <16f84a.h>                          // 標準ヘッダファイルのインクルード
2   #fuses HS,nowdt,noprotect     // HS, ウォッチドタイマなし, プロテクトなし
3   #use delay(clock=20000000)    // CPU clock : 20MHz
4   #use standard_io(a)
5   #use standard_io(b)
6
7   // 各I/Oポートのアドレス
8   #byte port_a=0x05                            // レジスタへの割りつけ
9   #byte port_b=0x06                            // レジスタへの割りつけ
10
11  #define DATA_BYTE 9                          // 出力データ
12
13  void init_port(void);                        // ポートの初期化
14  void mod_wave(void);                         // 38kHz変調
15  void send_data(void);                        // スイッチの状態を赤外線信号で発信
16
17  int key_data[DATA_BYTE];   // 送信データ保存用配列
18
19  void main(void)
20  {
21      int dat,check,pb_tmp,hed1,hed2,hed3,com;
22      init_port();                 // 各ポートの初期化
23      hed1=0x55;                               // ヘッダ3バイト
24      hed2=0xff;                               // ----------
25      hed3=0x00;                               // ----------
26      com=0xf7;                                // "Send message"コマンド
27
28      while(1)                                 // 無限ループ
29      {
30          pb_tmp=port_b;                       // ポートBの状態取り込み
31          dat=~pb_tmp;                         // データ反転
32          if(dat != 0)             // 入力があるまで送信待ち
33          {
34              dat=dat+10;                      // 班毎にデータ調整
35              check=dat + com;// チェックサム
36              key_data[0]=hed1;// ヘッダ1バイト目
37              key_data[1]=hed2;// ヘッダ2バイト目
38              key_data[2]=hed3;// ヘッダ3バイト目
39              key_data[3]=com; // Send messageコマンド
40              key_data[4]=~com;                // Send messageの反転
41              key_data[5]=dat;  // メッセージ本体
42              key_data[6]=~dat; // メッセージの反転
43              key_data[7]=check;               // チェックサム
44              key_data[8]=~check;              // チェックサムの反転
45
46              send_data();         // 赤外線データ発信
47              delay_ms(100);       // 送信後100msec待ち
48          }
49      }
50  }
51
52  //------------------
53  // 各ポートの初期化
54  //------------------
55
56  void init_port(void)
57  {
58      set_tris_a(0b100001);        // pin_a1～a4 を出力モードに設定する
59      output_a(0b011110);                      // pin_a1～a4に1の値を出力する
60
61      set_tris_b(0b11111111);  // port_b : 全ビット入力モードに設定する
62      port_b_pullups(TRUE);    // port_bのプルアップ抵抗をONにする
```

```
63  }
64
65  //--------------------------------
66  // リモコン操作する装置へ変調発信
67  //--------------------------------
68
69  void send_data(void)
70  {
71      int i,j,pal;                    // palはパリティビット用
72      for(i=0; i<DATA_BYTE; i++)
73      {
74        pal=0;                                    // パリティ用カウントの初期化
75        mod_wave();                               // スタートビット
76        for(j=0; j<8; j++)
77        {
78            if((key_data[i] & (0x01<<j))== 0)              // 往復データが0のとき
79            {
80                mod_wave();                       // 変調データのHを発信
81            }
82            else                    // 往復データが1のとき
83            {
84                delay_us(417);                    // データLを発信
85                pal=pal+1;          // パリティ；「1」の数のカウント
86            }
87      }
88      pal = pal % 2;                              // palを2で割ったものの余りを計算
89      if (pal != 0)               // データ中の1の数が奇数のとき
90      {
91          mod_wave();                             // パリティビットとして変調データのHを発信
92      }
93      else                                        // データ中の1の数が偶数のとき
94      {
95          delay_us(417);          // パリティビットとしてデータのLを発信
96      }
97      delay_us(417);                              // ストップビット
98    }
99  }
100
101 //-----------------------------
102 // 38kHz変調波を0.417msec発信
103 //-----------------------------
104
105 void mod_wave(void)
106 {
107     int i;
108     for(i=0; i<16; i++)                         // 約38kHzの変調波
109     {
110       output_a(0b000000);     // pin_a1～a4に0の値を出力
111       delay_us(9);
112       output_a(0b011110);     // pin_a1～a4に1の値を出力
113       delay_us(13);
114     }
115 }
```

【2】　コンフィギュレーションビットの指定
　　HS：4MHz以上の外部振動子
　　nowdt：ウォッチドタイマを使用しない
　　noprotect：コードをプロテクトしない
【3】　クロック速度を20MHzに指定する。
【4-5】　各ポートの入出力関数の都度、その入出力にあわせた入出力モード設定命令を追加する。
【11】　入出力データは「ヘッダ3バイト」＋「バイトコード1とその反転」＋「バイトコード2とその反転」＋「チェックサムとその反転」の9バイト
【13-15】　関数の宣言

【20-50】 本文
　【22-26】初期化
　　【22】　init_portの呼び出し
　　【57-63】init_port
　　【58-59】Aレジスタの1～4のピンを出力モードに設定し、1の値を出力する。
　　【61-62】Bレジスタの全ピンを入力モードに設定し、プルアップ抵抗を接続する。
　　【23-26】各コードの値の設定
　　hed1･･･ヘッダ1バイト目　　0x55
　　hed2･･･ヘッダ2バイト目　　0xff
　　hed3･･･ヘッダ3バイト目　　0x00
　　com････バイトコード1　0xf7
　【28-49】無限ループ：データの入力と送信
　　【30-31】スイッチの状態を取り込み、その反転をバイトコード2（図5参考)として設定する。
　　ポートBには入力のプルアップ抵抗が内蔵されており、プルアップする／しないを設定できる[12]。このプログラムでは【61-62】で初期化としてポートBを全ピン入力にし、プルアップ抵抗をONにしている。そして、RB0～3ピンにはスイッチをつなげている。
　　PICは入力モードに設定されると、TTLゲートを通って情報を読み込む。このとき値は、スイッチがOFFのとき電圧がVddでHighレベル「1」、ONのときはVssでLowとなるので「0」として入力される[17]。このため、読み込まれる値は期待していたものの反転となっているのでその値を期待値に戻すため、【31】でスイッチ状態の反転を行う。
　　【32】スイッチの入力があれば、送信を始める。
　　【34】複数のリモコンから同じデータが送信されて混信することを防ぐため、リモコンごとにデータを調整する。1つのリモコンが出すデータは1～10であるから（表4　スイッチの状態と送信データ参照)、リモコンごとに10ずつ足していけば、データが重複する心配はない。1班のリモコンならばdat + 10、2班のリモコンならばdat + 20というふうにする。
　　【36-44】送信する順にデータをkey-dataに配列する。
　　【46】　send_dataの呼び出し
　　【70-99】send_data：データの送信をする。
　　【72-98】1バイト分の送信
　　【74】　パリティのカウント用関数palを0に初期化する。
　　【75】　mod_waveの呼び出し：スタートビットとして変調データHの発信を行う。
　　【106-115】mod_wave：38kHzキャリア周波数に変調したものを417μsec送信
　　38kHzとは1秒当たり約38000回の速度でON/OFFを繰り返すという意味なので、1つのON/OFFの時間は　1÷38000＝0.0000263157894･･･≒26.3μsec　となる。
　　つまり一回のON/OFFにかける時間は26.3μsecなので、ON・OFFともに約13μsecずつかければよいことになる。
　　また、通信速度は2400bps≒417μsecなので、38kHzで1ビット送信するのにON/OFFの回数は417÷26.3＝15.8174904942965･･･≒16　である。
　　【110-111】pin_a1～a4に「0」の値を9μsec出力する。この「0」を出力する時間が13μsecでないのは、RCXの信号に近づけるための微調整である。
　　【112-113】pin_a1～a4に「1」の値を13μsec出力する。
　　この「0」と「1」の出力を16回繰り返し発信する。
　　【76-87】1バイトの送信
　　送信するデータの1ビット目から値をチェックする。0x01を左に"j" シフトしたものと、key_data[i]とでビットごとのAND演算を行う。その値が0ならば変調データHを発信するためmod_waveを呼び出す。また、値が1ならばデータLを発信し、パリティチェック用関数palをカウントする。【76】に戻りjをインクリメントする。jの値が8になるまでこれを続ける。
　　データLは「1」の値を417μsec発信する。
　　【88】　パリティが奇数か偶数かをチェックする。
　　【91】　palが奇数ならばパリティビットとして変調データHを発信する。
　　【95】　palが偶数ならばパリティビットとしてデータLを発信する。
　　【97-99】ストップビットとしてデータLの発信をした後、【72】に戻りiをインクリメントする。そして次のkey_data[i]の送信を行う。 iの値が9になるまでこれを続ける。
　　【47】　key_data[8]までの全てを送信した後、0.1秒間待つ。