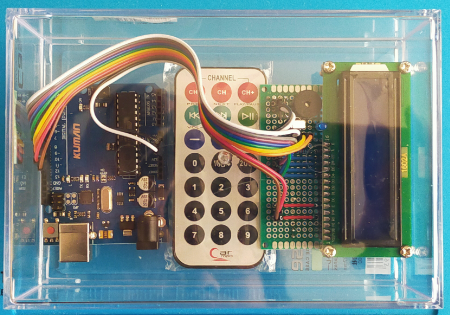
**モールス符号聴解練習機**

JJ1SLR　張弘達



# **概要**

# アマチュア無線では主に，「電話」「電信」で交信を行います。日本の国家資格のアマチュア無線技士3級以上は、電信交信のために、欧文モールスの暗記やそれによる通信の技能の習得が求められます。しかし、今日本のアマチュア無線技術者国家試験は筆記試験のみ、簡易化になって、例え1級に合格してもモールス符号による通信の実技が身に付けるとは言えません。私はこの実技ができない者の1人です。モールス符号受信を自己訓練するため、適当なツールを使って練習するのは一番大切であると思います。

# パソコンを使って練習ソフトを利用や、関連ウェブサイトやビデオ動画を聞くなど練習するのはいいですが、専用装置を使って練習するのは、もっとアマチュア無線のセンスを感じるのでしょうと思います。それで「モールス符号聴解練習機」を自作しました。

# 今回作成した練習機のソースコードは、下記のGitHubサイトでオープンソース（GPL3.0ライセンス）で掲載しました。誰でも自由に入手・利用可能です。

<https://github.com/JJ1SLR/MorseListening>

# **機能説明**

## **リモコン制御**

練習機は、ハードウェア構成の簡素化のため、赤外線リモコンを利用して、本機を制御します。もともと音楽プレーヤーの汎用品リモコンですから、各キーの機能を再定義します。そして、以下説明するキー以外は無効となります。



③

①

②

④

1. 順序練習文字間隔調整（順序練習モードのみ有効）

* 「CH-」　文字間隔を増加（遅くなる）
* 「CH+」　文字間隔を減少（速くなる）
* 「CH」　文字間隔をデフォルトに戻す

1. 速度（WPM）調整

* 「PREV」　速度減少
* 「NEXT」　速度増加
* 「PLAY/PAUSE」　速度をデフォルトに戻す

1. 音調（音の周波数）調整

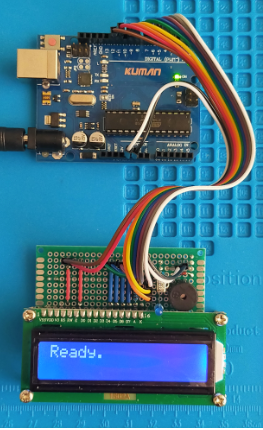
* 「VOL-」　音調を低める
* 「VOL+」　音調を高める
* 「EQ」　音調をデフォルトに戻す

1. 動作モード選択

* 「0」　初期モード
* 「1」　順序練習モード
* 「2」　ランダム練習モード１（符号全体）
* 「3」　ランダム練習モード２（英数字のみ）
* 「4」　ランダム練習モード３（英字のみ）
* 「9」　作者CQを出すモード

## **本体各部**

本体は、以下の各部分で構成します。



②

⑧

⑦

⑤

⑥

①

④

③

1. USBポート（Type-B）：PCを接続してファームウェアを更新できます。また、+5V安定化電源でこのポートから給電できます。
2. 電源ポート：7V～12V DCの電源で給電します。
3. 電源インジケータ：給電正常の場合、点灯します。
4. リモコン受信インジケータ：リモコン受信の場合、点滅します。
5. リモコン受信モジュール：リモコンの制御信号を受信します。
6. ブザー：モールス符号の音を出力します。
7. モールスインジケータ：モールス符号の光を出力します。
8. LCDディスプレイ：モールス符号、その他の情報を表示します。

## **動作モード詳細**

* + 1. **初期モード**

練習機を電源に接続して、初期モードになります。また、リモコンの「0」キーを押下の場合、初期モードになります。下記の「Ready.」画面を表示して、ユーザ入力を待機します。



* + 1. **順序練習モード**

リモコンの「1」キーを押下の場合、順序練習モードになります。

「無線局運用規則　別表第一号　モールス符号（第十二条関係）」「2　欧文」(<https://www.tele.soumu.go.jp/horei/reiki_honbun/72393000001.html>)

（以下、「モールス符号表」と略します）

表のモールス符号の順番通りモールス符号の音を出力し、LCDディスプレイで現在練習中のモールス符号の文字と「.」「-」で示す符号を2行表示します。最後の文字出力後実行を停止します。



* + 1. **ランダム練習モード１（符号全体）**

リモコンの「2」キーを押下の場合、ランダム練習モード１（符号全体）になります。

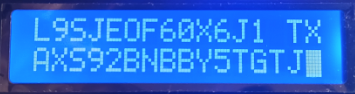
モールス符号表をランダムで出力し、32文字（空白文字も出力する場合があります）のモールス符号を出力します。最後の文字出力後実行を停止します。停止後、LCDの表示はクリアしないから、筆記したの場合、筆記文字とLCDディスプレイで表示する文字と照合ができます。

順序練習モードと違い、「.」「-」で示す符号は出力しません。そして、強化練習のため、音は先に出力してからLCDに文字を表示します。



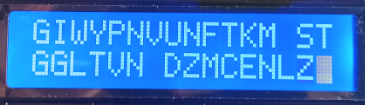
* + 1. **ランダム練習モード２（英数字のみ）**

「ランダム練習モード１（符号全体）」と同じで、出力文字の範囲は英数字のみです。



* + 1. **ランダム練習モード３（英字のみ）**

「ランダム練習モード１（符号全体）」と同じで、出力文字の範囲は英字のみです。



* + 1. **作者CQを出すモード**

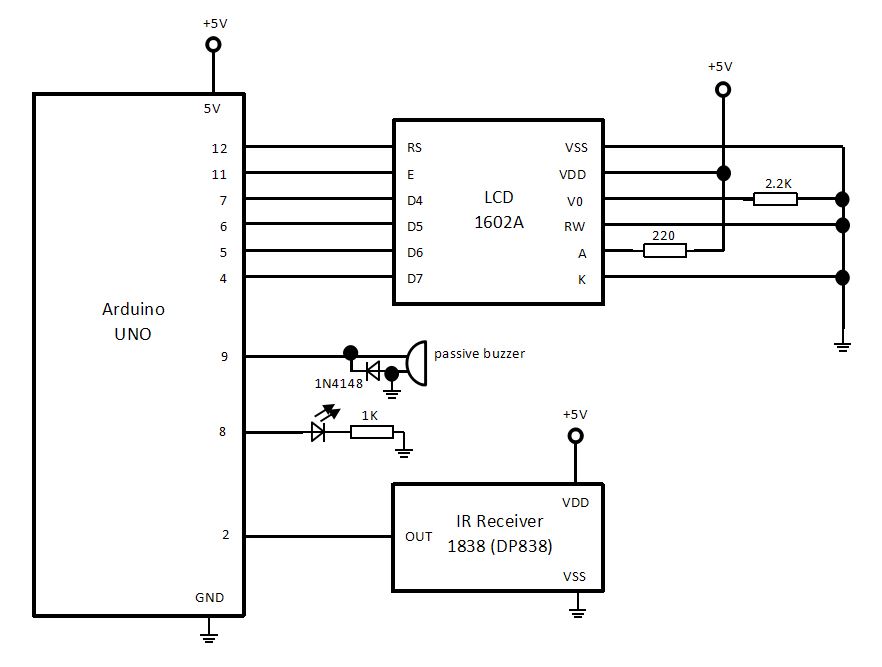
リモコンの「1」キーを押下の場合、作者CQを出すモードになります。

作者JJ1SLRのCQを出す時のモールス符号「CQ CQ DE JJ1SLR JJ1SLR K」を出力します。最後の文字出力後実行を停止します。ランダム練習モード１～３と違い、このモードでは音の出力とLCDディスプレイ文字の表示は同時に行います。



# **ハードウェア構成**

練習機の回路図は以下です。

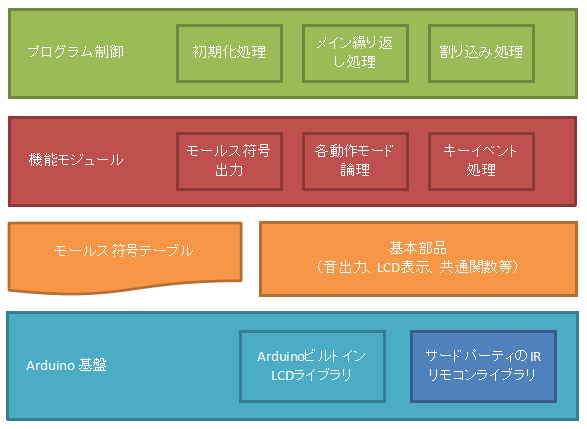


* 各抵抗器は概略計算の上、回路をテストしながら値を選定しました。
* Arduinoのピン接続はソースコード上の定義と一致しなければなりません。
* 音調を調整可能にするため、パッシブブザーを利用し、ソフトウェアで矩形波を発生して作動します。矩形波の周波数をソフトウェアで制御します。
* 今回のパッシブブザーはコイル式の部品なので、逆起電力からICを保護するため、ダイオード1N4148を追加しました。

# **ソフトウェア構成**

* 1. **ソフトウェア構成概要**

ソフトウェア構成は以下の図を参照してください。



ソースコードの所在は「1概要」を参照してください。紙幅の限りでここでは全部の説明を割愛します。重点的に一部を以下説明します。

* 1. **モールス符号テーブル**

モールス符号のデータ構造は今回のプログラムに対して一番重要なポイントであると思います。なぜなら、今回開発のプログラムはArduino UnoのATmega328Pマイコンに組み込む予定で、利用可能なROM（フラッシュメモリー）は32KB未満で、RAMは2KBのみあります。これを計画的に使わないと、特にRAMがオバーして、プログラム全体は正常に動作ができなくなります。限られた資源の有効活用は組み込み系プログラミングの特徴です。

モールス符号は一見2進数で表せるが、「・」「―」以外、空白も存在するので、実質3進数であれば適当です。しかし、コンピューターは2進数でデータを処理するので、この矛盾をどう解決するのは重要です。

* + 1. **一般的なプログラムの考え方**

一般ウェブやPCプログラミングであれば、Key-Valueペアでモールス符号テーブルを格納するのは簡単でできます。

例：

|  |  |  |
| --- | --- | --- |
| 文字 | Key | Value |
| A | ‘A’ | “.-” |
| B | ‘B’ | “-...” |
| C | ‘C’ | “-.-.” |
| ... | ... | ... |

これは一番考えやすいストレージ構造で、リスト又はハッシュテーブルに保存します。プログラムではKeyでこのテーブルを検索し、取得するValueの文字列を1文字ずつ処理しモールス符号情報出力等処理を行えます。

とてもシンプルな構造ですが、たとえ文字「B」の場合、Keyは1Byte、Valueは5Byte（最後尾‘\0’文字含む）で、少なくとも6Btyeが必要です。プログラム実行の時、このテーブル全体はRAMにロードされますから、メモリが足りない恐れがあります。

* + 1. **今回開発の考え方**

コンピュータにて文字はASCIIコードの形式で処理します。例えば文字‘A’は、メモリ上は0x41を格納し、入出力の時はデータ型によって文字‘A’と認識されます。この特徴を利用してASCII順で各文字のモールス符号情報を配列に格納すれば、指定文字のインデックスが算出できますから、Key情報の保存は必要なくなりました。

また、今回練習機の処理対象モールス符号は単独文字のみなので、一番長いものは記号の一部で音6つまでです。1Byteは8ビットで、「いくつのビットは有効であるか」の問題が解決できたら文字1つのモールス符号は1Byteで収めることは可能であると思います。

上記の観点で、以下のモールス符号とそのテーブルのデータ構造を設計しました。

■モールス符号

1Byteの「・」は0で、「―」は1で、ビット単位で左側（上位桁）から右側（下位桁）へ順序格納します。そして無効ビットの始めの桁は1を設定し、残りの全桁は0を設定します。

例：

文字‘C’の格納イメージは以下に示します。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 値 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 意味 | ― | ・ | ― | ・ | 無効ビット開始 | 無効ビット | 無効ビット | 無効ビット |

このようなデータを処理時に、まず右側ビットから左側へデータを読んで、初めて読んだ1を含めて無効ビットと認識し、無効ビット数を取得できます。そして有効ビット数も算出できます。上記の例の場合、無効ビット数は4bitで、有効ビット数は(8-4)=4Bitであることがわかりました。そして左側ビットから右側へもう一度4Bitの情報を読み込んで、「1010」つまり「―・―・」の正しい情報が取得できます。

■モールス符号テーブル

ASCIIコード表の0x20（空白文字）～0x5A（‘Z’）の範囲ではモールス符号表の全文字を含みます。長さは59のバイト配列を定義し、各アイテムは上記の「モールス符号データ」を格納すれば、モールス符号テーブルが作成できます。但し、ASCIIコード表に含み、モールス符号表に含まない文字に対して下記特別な処理が必要です。

* 空白文字：0x00で表します。
* その他モールス符号表に含まない文字（例：‘#’‘$’‘\*’等々）：0x00で表します。

文字を指定して、モールス符号情報を取得の場合、モールス符号テーブルのインデックスを算出します。計算は単に文字のASCIIコード値引く固定のオフセット値（0x20）で済みます。

例：

|  |
| --- |
| 文字‘C’のインデックス　＝‘C’のASCIIコード（0x43）―オフセット値（0x20）＝　0x23　＝　35 |

モールス符号テーブルのインデックス35の値を取得すれば、文字‘C’のモールス情報が取得できます。

* 1. **割り込み処理**
  2. **音出力とノンブロッキング遅延関数**
  3. **ロングジャンプ制御**

# **後書き**

コストの話

バージョンアップの話