課題殴り書き

まず処理がどこで詰まっているかの特定から行った. 中華の互換板を昔から使っていて, いまさら買い直すのもめんどうなので, 実験でも用いたが, なかなかの曲者で 9600bps 以外のボーレートで通信すると, 正しく通信が行われなくなってしまう(シリアルモニタには文字化けした文字列が出る). つまりなにが言いたいのかと言うと, 通信速度のせいで細かい領域の出力ができない. Arduino の処理は簡単に説明すると

wav のデータ 4480 サンプルを参考に配列 A224 個を決める
配列 A224 個を参考に配列 B14個を決める
配列 B14 個を参考に実際にリレーを動作させる2回分の出力(配列 C)を決定している

B や C の出力は通信速度が足りているが, A や生の波を出力することはできない.

一つの懸念として CPU の処理速度について考えていた(解決済). 処理速度の観点, 他の実装方法の簡単さから周波数の特定にフーリエ変換は用いなかった. 簡単に言うと,

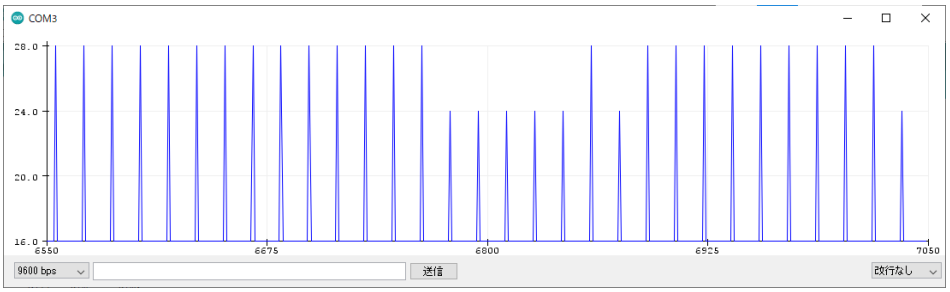
① 配列を[2][224]とる

② Wav の入力に応じたデータを[0][0]～[0][224]に代入している間に
[1][0]～[1][224]もデータを復号し ZK-80 に入力する.

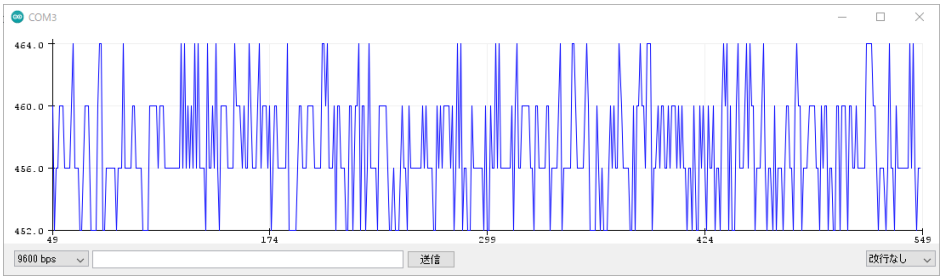
③ Wav の入力に応じたデータを[1][0]～[1][224]に代入している間に
[0][0]～[0][224]もデータを復号し ZK-80 に入力する.

④ ②に戻る.

②や③の処理は 224ms なのに対し, 割り込みは1回につき高々0.03ms
224 回繰り返しても 6.72ms



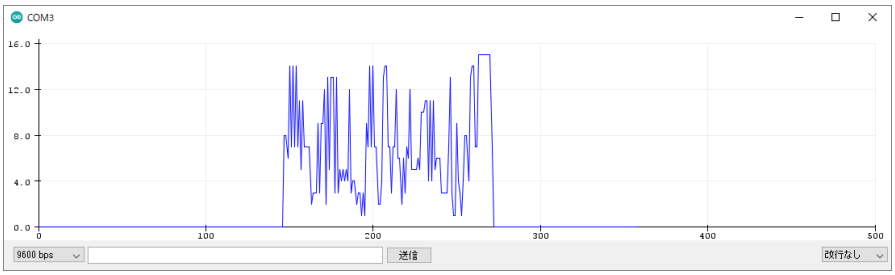
信号の複合処理は高々0.47ms



TK-80 への入力処理についてはまだ記述していないが, 処理の大半が nop であること, 複合処理のコードは 3 重 for のなかで else if を多用していることから, 入力処理よりも格段に重いので, 処理時間は気にしなくてよいと考える.
つまり 224ms の時間的猶予に対し, 処理は最悪の組み合わせでも 7ms 程度で終了することから, Arduino の処理速度が原因ではないと言える.

更新

一部解決した. Arduino 側では, wav の電圧の立ち上がりを検出していた. そして, 立ち上がりの時間差から周波数を求めていたが, つまり n 回の周波数を検出するためには n+1 回の電圧立ち上がりが必要であるが, プログラムでは n 回しか立ち上がらない wav ファイルを作成していた. テストプレイ中は wav をリピートしていたので, 一回の再生ループにつき, 電圧立ち上がりが一回ずつずれこみ. 出力に謝ったものが出てきていたと考察する. 現在"0"だけのファイル(つまり周波数がつねに 500Hz の矩形波)を入力として入れてみると, 配列 C において常に 0 になった. (これは正しい動作をしてくれている)



次に, 0 を 8 回, 1 を 8 回, 2 を 8 回, , , , F を 8 回という信号を送ったら, 上手く行かなかった.
https://github.com/nex-finger/jikken3/blob/main/Document/log.txt
たとえば, 最初の 8 回は上手く行っている. 出力は正しく 0 だが, 次の 8 回では 1 はでないし, 23456789ABCDE でも同様に間違った信号となっている. しかし, 全て信号"1"の F は上手くいっているようで, その後の 0 の連続にも期待する結果を出力している.
原因は, 割り込み処理の安直な実装かもしれない. 電圧立ち上がりの間隔には

1ms(1000Hz)と 2ms(500Hz)がある. つまり, 2ms の方では, 時間当たりの情報量が半分になるということ. よって, 配列に入れる作業を 1ms では 1 回あたり1回, 2ms では1回あたり 2 回行っている. これが悪さをしている, , , ような気がする.