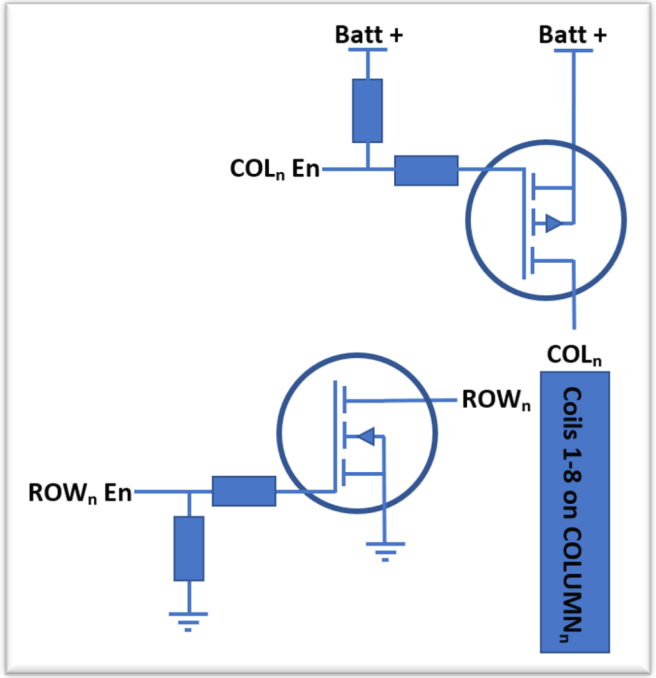
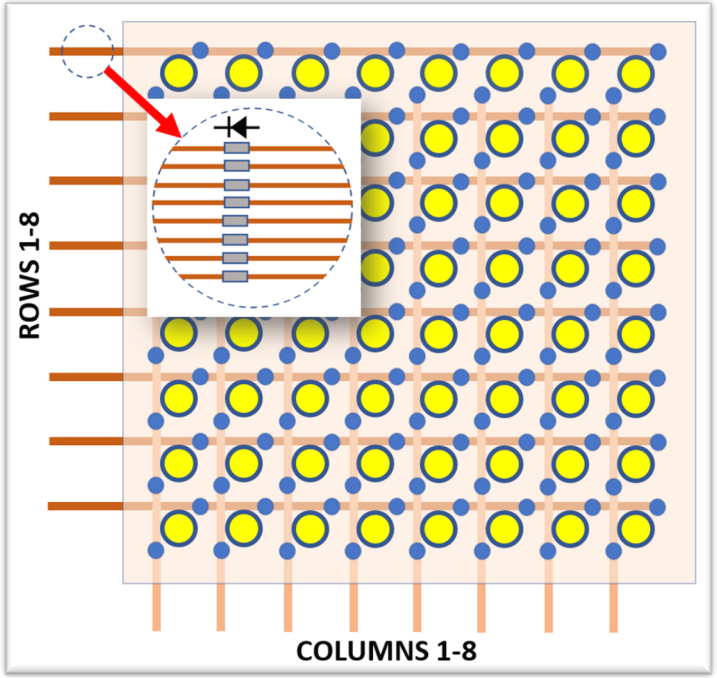
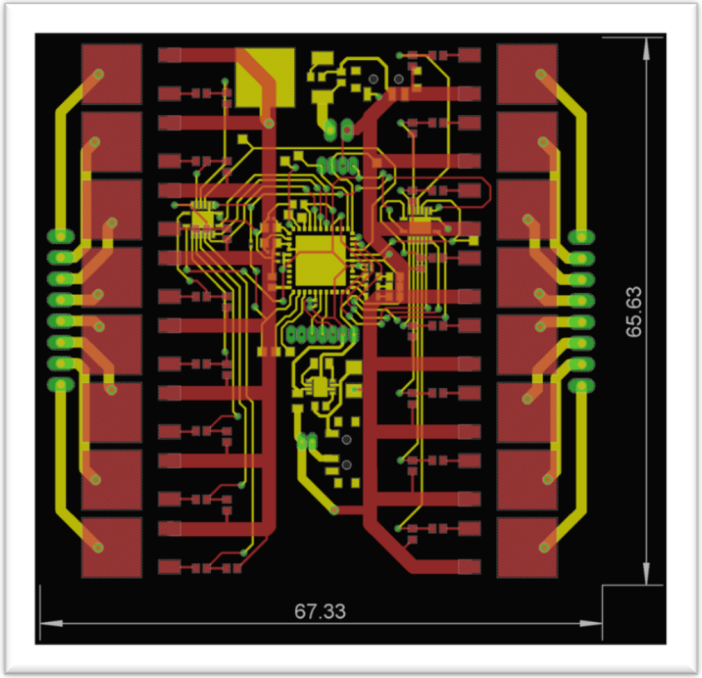
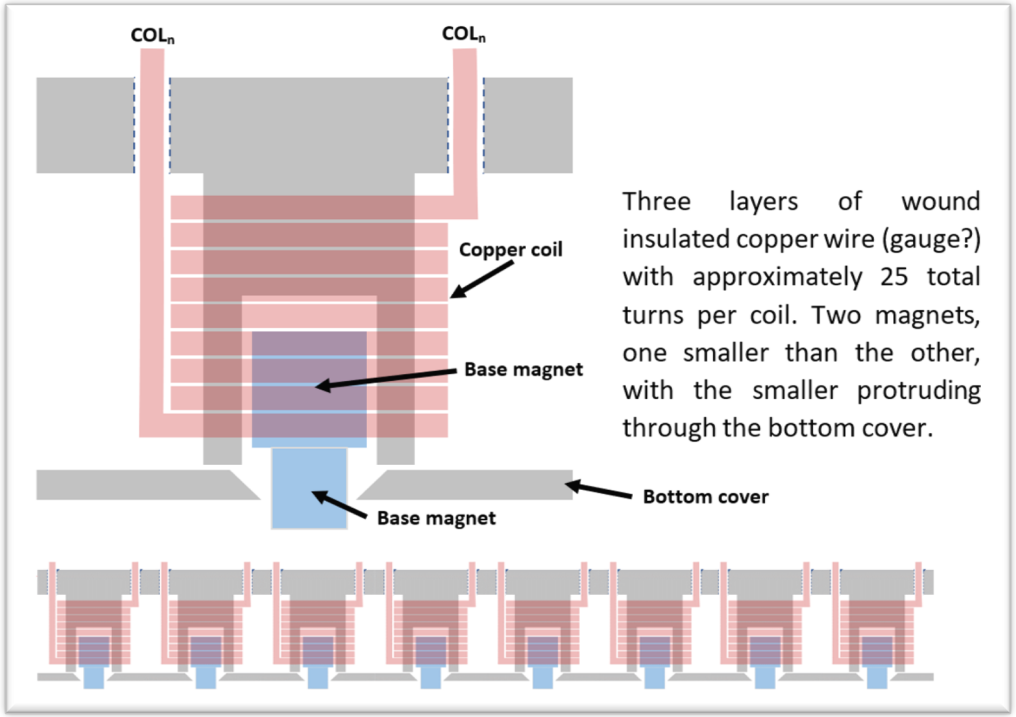
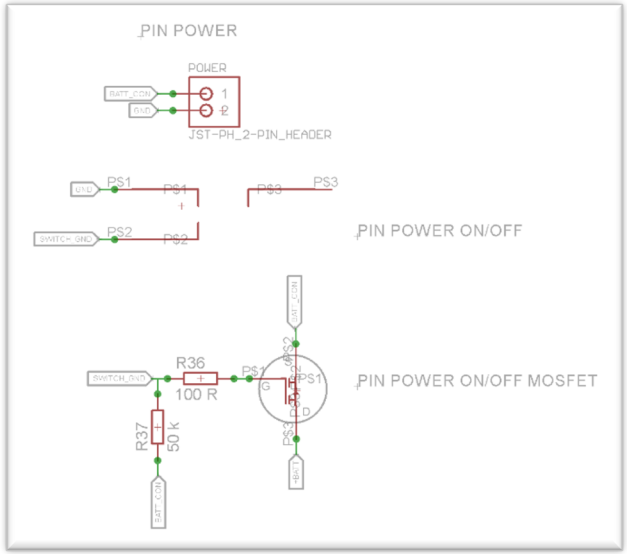


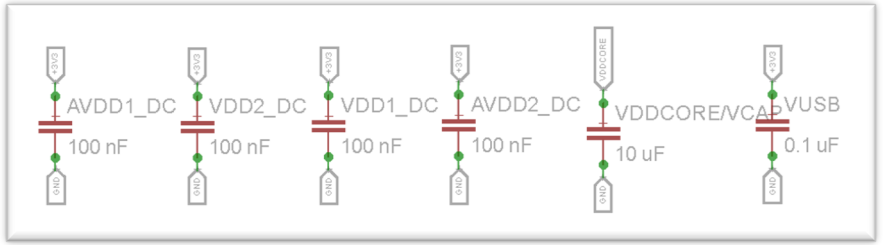


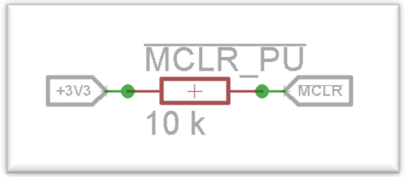


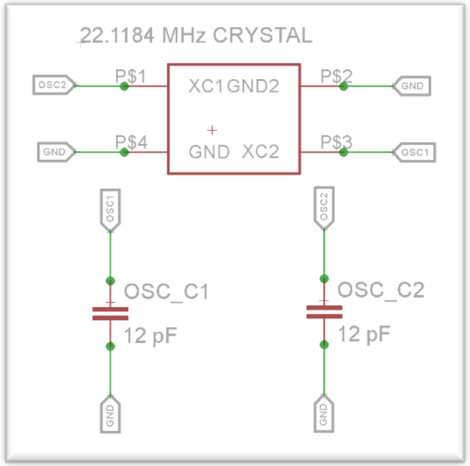


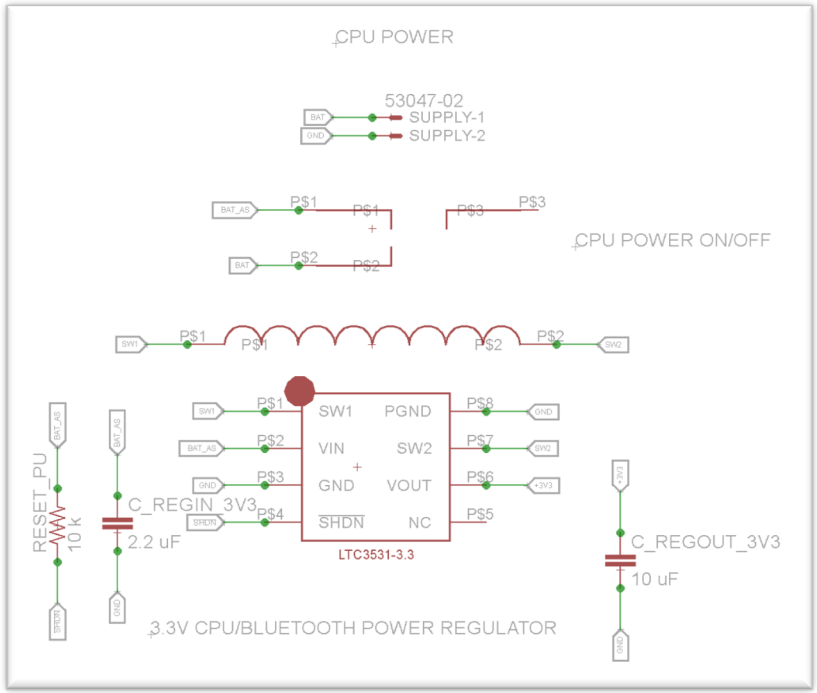
Three layers of wound insulated copper wire (gauge?) with approximately 25 total turns per coil. Two magnets, one smaller than the other, with the smaller protruding through the bottom cover.

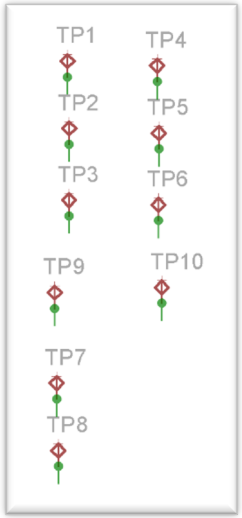

PIN POWER
POWER
JST-PH_2-PIN_HEADER
PIN POWER ON/OFF
PIN POWER ON/OFF MOSFET
R36
100 R
R37
50 k

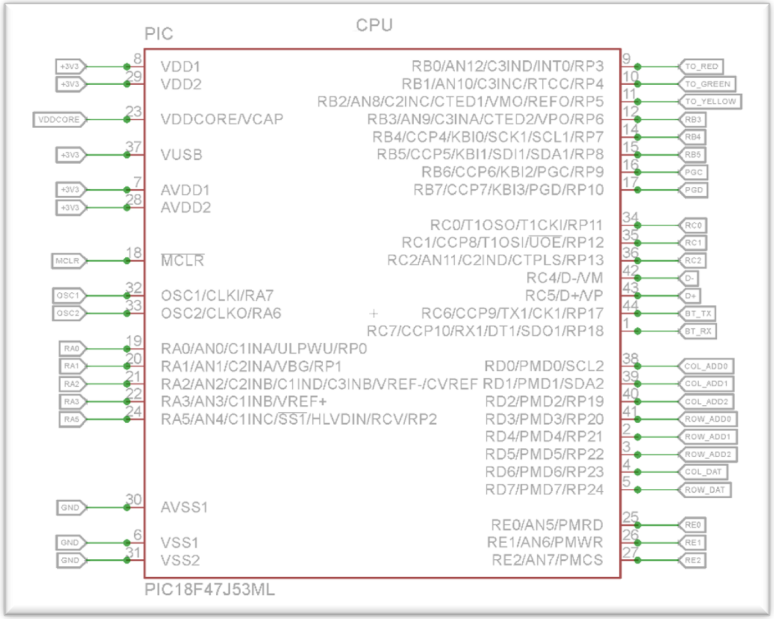

AVDD1_DC
100 nF
VDD2_DC
100 nF
VDD1_DC
100 nF
AVDD2_DC
100 nF
VDDCORE/VCAP
10 uF
VUSB
0.1 uF

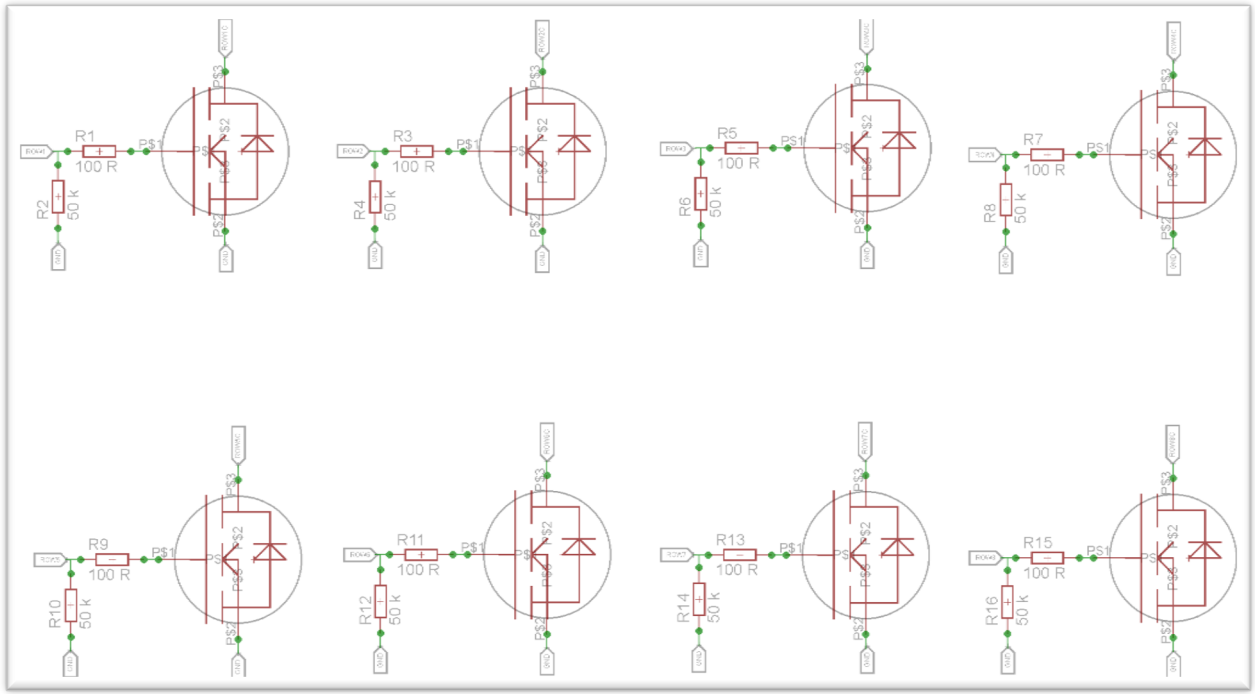

MCLR_PU
10 k

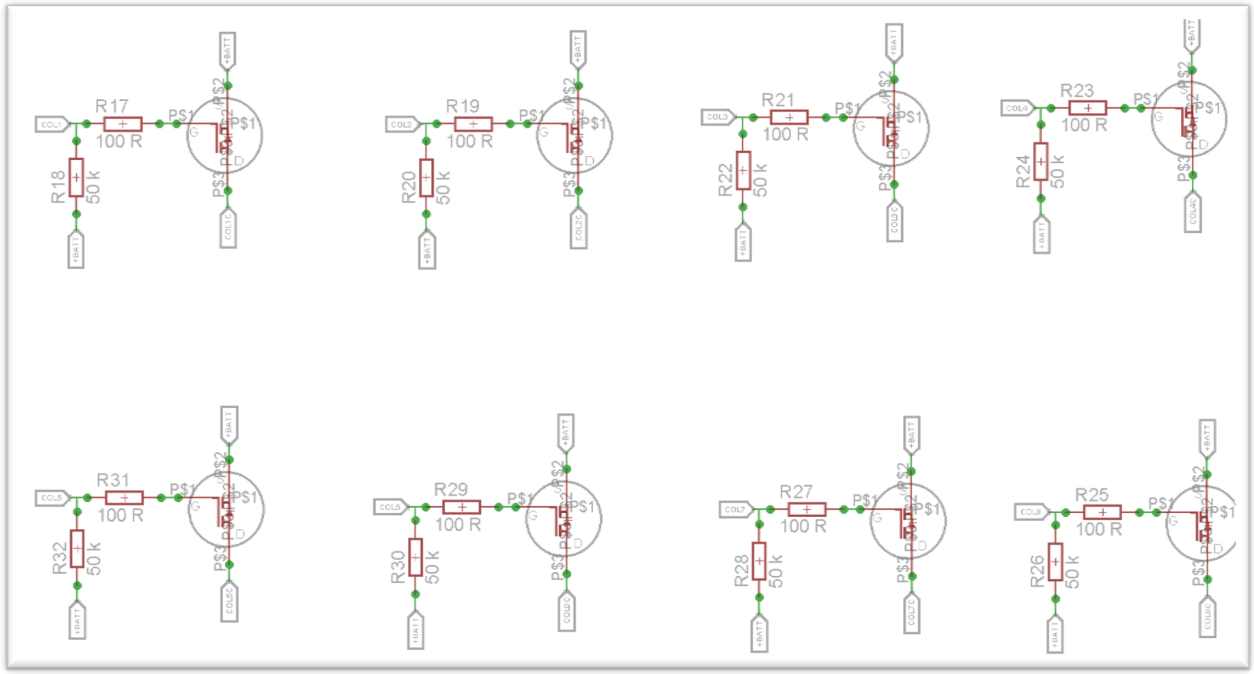

22.1184 MHz CRYSTAL
XC1 GND2
GND XC2
OSC_C1
12 pF
OSC_C2
12 pF

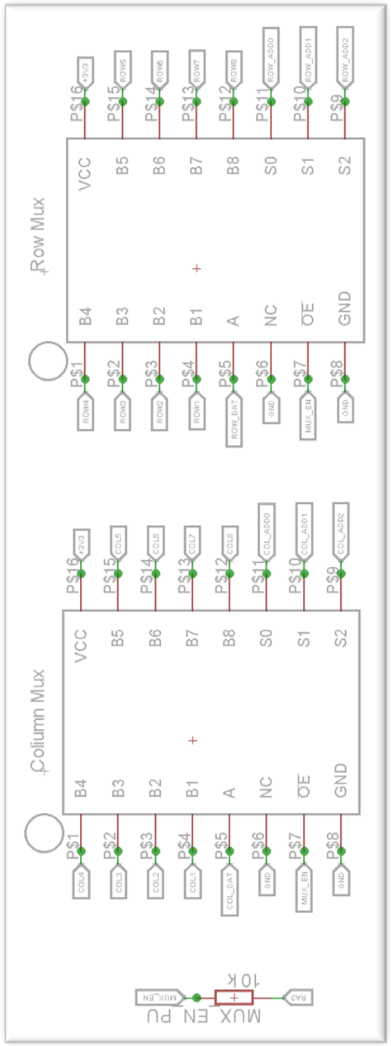

CPU POWER
53047-02
SUPPLY-1
SUPPLY-2
CPU POWER ON/OFF
LTC3531-3.3
C_REGIN_3V3
2.2 uF
C_REGOUT_3V3
10 uF
RESET_PU
10 K
3.3V CPU/BLUETOOTH POWER REGULATOR

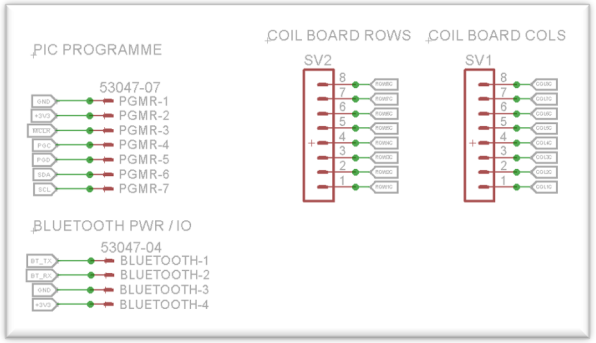

TP1
TP2
TP3
TP4
TP5
TP6
TP7
TP8
TP9
TP10

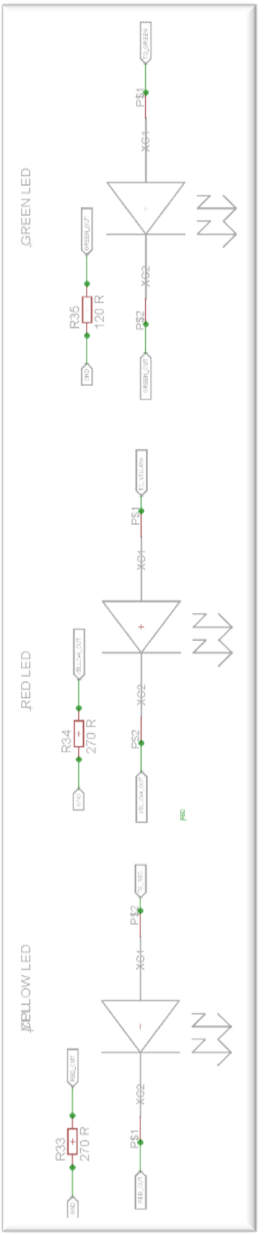

CPU
PIC
PIC18F47J53ML


R1
100 R
R2
50 k
R3
100 R
R4
50 k
R5
100 R
R6
50 k
R7
100 R
R8
50 k
R9
100 R
R10
50 k
R11
100 R
R12
50 k
R13
100 R
R14
50 k
R15
100 R
R16
50 k

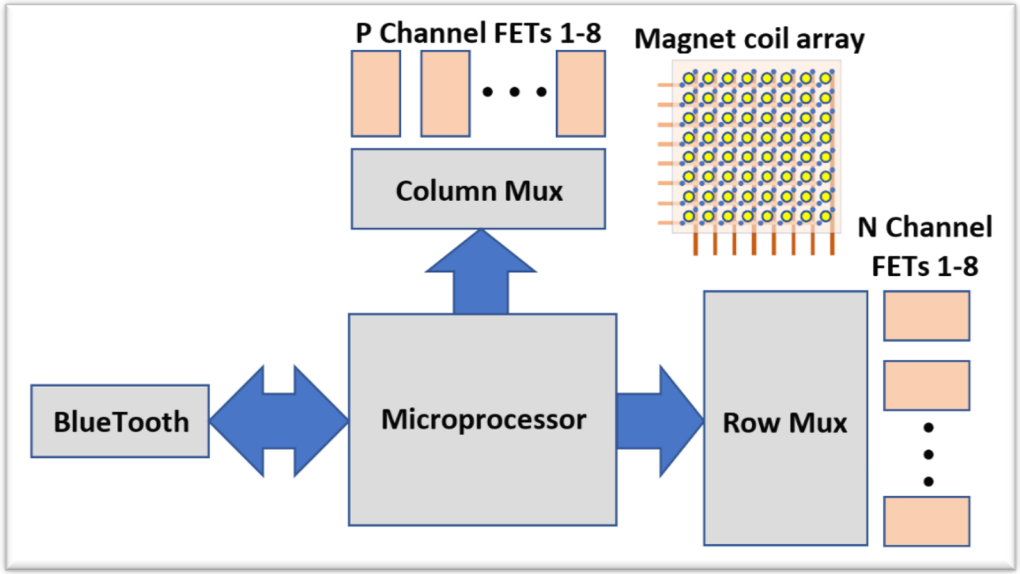

R17
100 R
R18
50 k
R19
100 R
R20
50 k
R21
100 R
R22
50 k
R23
100 R
R24
50 k
R31
100 R
R32
50 k
R29
100 R
R30
50 k
R27
100 R
R28
50 k
R25
100 R
R26
50 k


Row Mux
Column Mux
VCC
GND
MUX EN PU
10K


PIC PROGRAMME
53047-07
PGMR-1
PGMR-2
PGMR-3
PGMR-4
PGMR-5
PGMR-6
PGMR-7
BLUETOOTH PWR / IO
53047-04
BLUETOOTH-1
BLUETOOTH-2
BLUETOOTH-3
BLUETOOTH-4
COIL BOARD ROWS
SV2
COIL BOARD COLS
SV1


GREEN LED
R35
120 R
RED LED
R34
270 R
BELLOW LED
R33
270 R

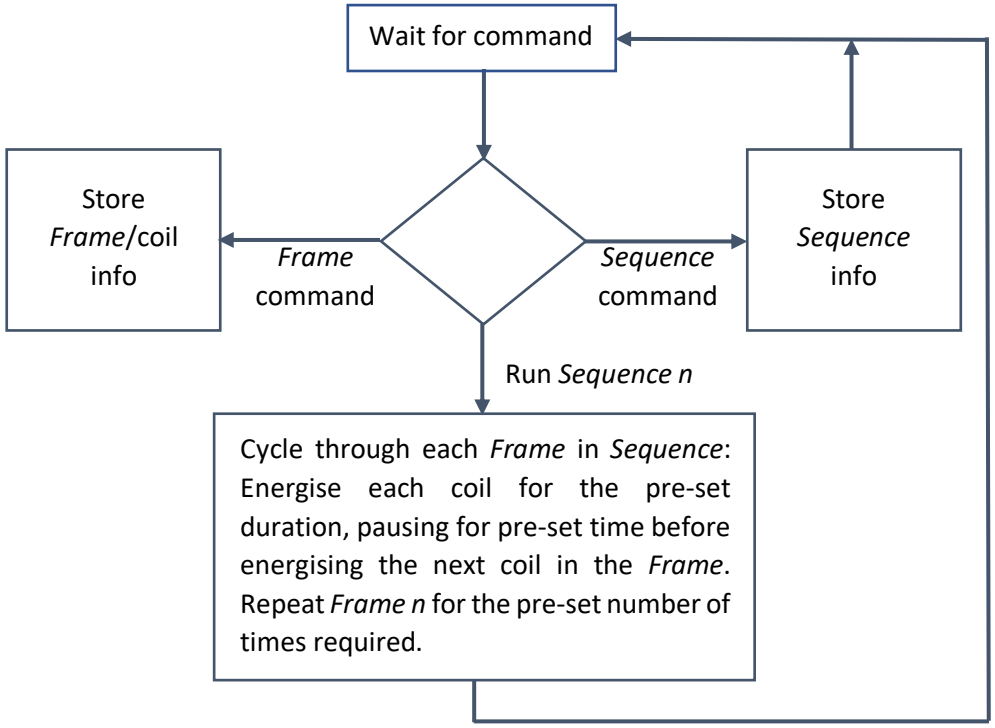
This system comprises of several units



- Bluetooth module — Communication to / from a mobile phone or tablet
- CPU — To coordinate communications with mobile phone or tablet, and to store and control the flow of signals to the 8x8 magnet array
- Column/Row Mux — These units take a 3-bit binary address each to address 1 of 8 columns or rows, such that only a single coil is energised at any one time
- Magnet coil array — 8 x 8 coils, with the “end 1” of each coil in a column connected to a P channel MOSFET (per column), but with each coil “end 2” having its own individual diode at the “row” output, prior to the N channel MOSFET, present to ensure no current can slow back up a secondary coil on the same row

The processor selected for this project is MicroChip's PIC47J53, an 8-bit microcontroller, comprising of 64 kByte of FLASH memory, primarily used for program storage, and 3.8 kByte of RAM. This device is driven by a 22.1184 MHz crystal, chosen to allow high serial communication BAUD rates, configured for 115,200 BAUD, permissible via the attached HC-05 serial Bluetooth module. Two batteries are used; one of large capacity used to deliver current through the coils, and the other for the CPU/control electronics. Two batteries are used as, if the larger battery was used to both dump current through coils, and for the control circuitry, there is the chance that its voltage may drop below that required for the voltage regulator and cause a reset. Dumping current through the coils, one at a time, is achieved by switching on 2 transistors (MOSFETS), effectively "shorting" the battery to itself for the briefest of moments.

At power on, the system clears its internal pattern buffers ready to receive data from the user. Several commands are used to assign parameters that dictate which coils will be energised in a *frame*, and in which order, how long current is allowed to flow through them, one at a time, the pause duration after one coil is switched off, before the next is switched on, and how many times an individual coil is toggled on/off, before progressing onto the next. A *sequence* can then be defined, comprising of one or more *frames*, repeated *n* times. These *frames* and *sequences* are uploaded in a matter of seconds from the user's mobile phone or tablet. The user then instructs the device to execute a *sequence* of *frames*.



The current flowing through a coil is the order of several amperes (battery capacity / charge dependent), for the briefest of moments, typically sub-millisecond. This duration is more than enough to "kick" the two magnets residing inside each coil. Also, as each coil is on for less that one millisecond, several other coils in the 8x8 array can then be energised, with the selected coils energising in a cyclic fashion at up the 40 Hz, or possibly more. This high level of control on how much current flows through a coil (timewise), and how often this occurs per coil, allows for a varying degree of sense of "strength" when the magnets are touched by the hand or fingers. With the microprocessor running at 22 MHz, this allows for a fair degree of precision regarding the timing of the coils. The pause time between energising coils is defined "per millisecond", while the coil "on-time" is "per 100 microseconds". No higher resolution was deemed necessary but is certainly possible should it be required.

Estimated field strength per coil can be estimated with a well-known equation:

$$B = \mu n I$$

Where

μ is the permeability; for neodymium magnets, this is approximately 1.32×10^{-6} H/m

n is the number of turns in the coil per metre, or the "turns density" N/L; for this device, approximately 24 turns per 4 mm

I is the current flowing in the coil, A

Therefore, the magnetic field strength is of the order of 24 mT

PIC18F47J53 (PICBASIC3)

```
Include "modedefs.bas"
#CONFIG
        CONFIG XINST = OFF
        CONFIG WDTEN = ON
        CONFIG WDTPS = 512
        CONFIG ADCSEL = BIT12
        CONFIG RTCOSC = T1OSCREF
        CONFIG OSC = HSPLL
        CONFIG CLKOEC = OFF
        CONFIG CFGPLLEN = OFF
        CONFIG CP0 = ON
        CONFIG IESO = ON
        CONFIG SOSCSEL = LOW
        CONFIG CPUDIV = OSC1
#ENDCONFIG

DEFINE OSC 20 ' 20MHz oscillator (22.1184MHz)
DEFINE HSER_RCSTA 90h ' Enable serial port & continuous receive
DEFINE HSER_TXSTA 20h ' Enable transmit, BRGH = 1
DEFINE HSER_CLROERR 1 ' Clear overflow automatically
DEFINE HSER_SPBRG 11  ' 115200 Baud @ SPBRGH = 0

BAUDCON.3 = 1        ' Enable 16 bit baudrate generator

command                 var byte[4]
frame_pointer             var word[100]
frame_sequence_repeats      var byte[500]
frame_data              var byte[1500]
frame_sequence_data         var byte[1000]
frame_sequence_ptr          var word[100]
frame_sequence_f_f_pause    var byte[100]
command_ptr               var word
search_bearing            var bit
command_value               var byte
```

```
command_go                  var byte
define_frame_point_x        var byte
number_of_coils_in_frame    var byte
frame_started               var byte
coil_on_time                var word
coil_coil_delay             var byte
data_mode                   var byte
cnt                         var word
cnt2                        var byte
cnt3                        var byte
max_frame_defined           var byte
frame_frame_delay           var byte
ser_spd                     var byte
frame_number                var byte
frame_sequence_num          var byte
frame_seq_started           var byte
max_sequence_defined        var byte
frame_sequence_frame_counter  var byte
repeat_frame                var byte
coil_num                    var byte
verbose                     var byte

trisa.3                     =  0  '   !mux_en
trisd                       =  0  '   set all portd = output


mux_en                      var lata.3

col_dat                     var latd.6
row_dat                     var latd.7

col_add0                    var latd.0
col_add1                    var latd.1
col_add2                    var latd.2

row_add0                    var latd.3
```

```
row_add1                  var latd.4
row_add2                  var latd.5

data_mode                  =  0
coil_coil_delay            =  0
coil_on_time               =  0
frame_started               =  0
number_of_coils_in_frame      =  1
define_frame_point_x         =  0
command_go                  =  0
max_frame_defined           =  0
max_sequence_defined         =  0
frame_sequence_frame_counter   =  0
frame_sequence_num           =  0
verbose                   =  0

mux_en                     =  1      '   !mux_en = mux's disabled

low col_add0
low col_add1
low col_add2

low row_add0
low row_add1
low row_add2

high col_dat    ' all columns off
high row_dat    ' all rows on

low mux_en


T2CON                     = %00000000     ' Turn off timer2
INTCON.7                    = 0         ' Disable interrupts.
PR2                       = 255           ' PWM frequency.
T2CON                     = %00000100     ' Timer2 ON + 1:1 prescale
```

```
Serial_in                 var portc.7
green_led                   var portb.1
red_led                   var portb.0
yellow_led                  var portb.2
key_press                   var portb.7
Serial_out                  var portc.6

intcon                    = 0
intcon2                    = %11110101
intcon3                    = %00000000

trisb.3                   = 0
trisa.1                   = 0
trisa.2                   = 1
'trisa.3                   = 1

command_ptr                 = 0
frame_number                = 0

main:

   gosub clear_arrays

   HSerout [$d,$a,"Coil array"]

   high green_led

   command_ptr = 0
   pause 200

   high green_led

   low yellow_led
   pause 100
   low green_led
```

```
   pause 50
pp:

   HSerout [$d,$a,"Waiting for instruction",$d,$a]

   command_go = 0

   high green_led
   gosub listen_for_commands
   gosub execute_commands
   low yellow_led
   goto pp
end


listen_for_commands:

   while command_go = 0
      hserout [$d,$a,dec frame_number," > "]
      hserin [wait("$"),str command\4]
      hserout ["$",str command\4]

      command_value = (command[1] - 48) * 100 + (command[2] - 48) * 10 + (command[3] - 48)
      if command[0] = 100 then ' sequence array dump                        d
         for cnt = 0 to 50
            hserout[$d,$a,"PTRs: ",dec cnt," ",dec frame_sequence_ptr[cnt]]
         next cnt
         for cnt = 0 to 50
            hserout[$d,$a,"data: ",dec cnt," ",dec frame_sequence_data[cnt]]
         next cnt
         for cnt = 0 to 50
            hserout[$d,$a,"rpts: ",dec cnt," ",dec frame_sequence_repeats[cnt]]
         next cnt
      endif
      if command[0] = 76 then ' list stored data                           L
```

```
        HSerout [$d,$a,"List stored commands"]

        for cnt = 0 to max_frame_defined
           hserout[$d,$a,$a,"Frame #",dec cnt]
           hserout[$d,$a,"Time-on ",dec frame_data[frame_pointer[cnt]+2]]
           hserout[$d,$a,"Coil-coil time ",dec frame_data[frame_pointer[cnt]+1]]
           hserout[$d,$a,"Number of coils in frame ",dec frame_data[frame_pointer[cnt]+0]]
           for cnt2 = 0 to (frame_data[frame_pointer[cnt]+0] - 1)
              hserout[$d,$a,"Coil X:",dec cnt2," = ",dec frame_data[frame_pointer[cnt]+3 + cnt2]]
           next cnt2
           hserout[$d,$a]
        next cnt

        for cnt = 0 to (max_sequence_defined - 1)
           hserout[$d,$a,$a,"Sequence #",dec cnt]
           hserout[$d,$a,"Number of frame sets in this sequence ",dec frame_sequence_repeats[frame_sequence_ptr[cnt] + 0]]
           for cnt2 = 0 to (frame_sequence_repeats[frame_sequence_ptr[cnt] + 0] - 1)
              hserout[$d,$a,"Frame set #",dec cnt2," = Frame #",dec frame_sequence_data[frame_sequence_ptr[cnt] + 1 + cnt2]," repeated #",dec
frame_sequence_repeats[frame_sequence_ptr[cnt] + 1 + cnt2]," times"]
           next cnt2
           hserout[$d,$a]
        next cnt
     endif

     if command[0] = 67 then ' clear stored data                    C
        HSerout [$d,$a,"Clear stored commands"]

        max_frame_defined = 0
        max_sequence_defined = 0
        frame_started = 0

        for cnt = 0 to 2999
           frame_data[cnt] = 0
        next cnt

        for cnt = 0 to 99
```

```
      frame_pointer[cnt] = 0
   next cnt

   for cnt = 0 to 499
      frame_sequence_data[cnt] = 0
   next cnt

   for cnt = 0 to 499
      frame_sequence_repeats[cnt] = 0
   next cnt

   for cnt = 0 to 29
      frame_sequence_ptr[cnt] = 0
   next cnt

endif

if command[0] = 77 then ' set data mode                          M
   if frame_started = 1 then
      HSerout [$d,$a,"Data mode = ",dec data_mode]
   else
      HSerout [$d,$a,"Please start a frame definition with $Fxxx"]
   endif
endif

if command[0] = 68 then ' set coil - coil delay                  D
   if frame_started = 1 then
      if verbose = 1 then HSerout [$d,$a,"Frame #",dec frame_number," coil-to-coil delay = ",dec command_value]
         coil_coil_delay = command_value
      frame_data[frame_pointer[frame_number]+1] = coil_coil_delay
   else
      HSerout [$d,$a,"Please start a frame definition with $Fxxx"]
   endif
endif

if command[0] = 86 then ' set coil - coil delay                  D
```

```
    verbose = command_value

    hserout [$d,$a, "Verbose mode = ",dec verbose]
endif

if command[0] = 80 then ' set coil on-time x100 us                    p
    if frame_started = 1 then
        if verbose = 1 then HSerout [$d,$a,"Frame #",dec frame_number," on-time = ",dec command_value]
        coil_on_time = command_value
        frame_data[frame_pointer[frame_number]+2] = coil_on_time
    else
        HSerout [$d,$a,"Please start a frame definition with $Fxxx"]
    endif
endif

if command[0] = 84 then ' terminate frame definition                  T
    if verbose = 1 then
        if frame_started = 1 then
            HSerout [$d,$a,"Frame #",dec frame_number," ended"]
        endif
    endif
    if verbose = 1 then
        if frame_seq_started = 1 then
            HSerout [$d,$a,"sequence ",dec (frame_sequence_num - 1)," ended"]
        endif
    endif
    frame_started = 0
    frame_seq_started = 0

endif

if command[0] = 70 then ' Define frame number                         F
    if frame_started = 0 then
        frame_number = command_value
        define_frame_point_x = 0
```

```
            if frame_number > max_frame_defined then max_frame_defined = frame_number

            frame_started = 1
            if verbose = 1 then HSerout [$d,$a,"Frame definition started #",dec command_value]

            if frame_number == 0 then
               frame_pointer[0] = 0
            else
               frame_pointer[frame_number] = frame_pointer[frame_number - 1] + frame_data[frame_pointer[frame_number - 1]] + 3 ' previous frame start
+ number of point + 3 vars: coils, delay and on-time
            endif
         else
            HSerout [$d,$a,"Please end current frame definition"]
         endif
      endif

      if command[0] = 78 then ' define number of coils in current frame          N
         if frame_started = 1 then
            if verbose = 1 then HSerout [$d,$a,"Frame #",dec frame_number," number of coils = ",dec command_value]
            number_of_coils_in_frame = command_value
            frame_data[frame_pointer[frame_number]] = number_of_coils_in_frame
         else
            HSerout [$d,$a,"Please start a frame definition with $Fxxx"]
         endif
      endif

      if command[0] = 83 then ' start to define a frame sequence                S
         if frame_started = 0 then
            frame_sequence_num = frame_sequence_num + 1
            frame_sequence_frame_counter = 0
            frame_seq_started = 1
            max_sequence_defined = max_sequence_defined + 1

            if verbose = 1 then HSerout [$d,$a,"frame_sequence_num: ",dec frame_sequence_num]
            if verbose = 1 then HSerout [$d,$a,"frame_sequence_ptr[frame_sequence_ptr[frame_sequence_num - 2]]: ",dec
(frame_sequence_ptr[frame_sequence_ptr[frame_sequence_num - 2]])]
```

```
            if verbose = 1 then HSerout [$d,$a,"frame_sequence_repeats[frame_sequence_ptr[frame_sequence_num - 2]]: ",dec
(frame_sequence_repeats[frame_sequence_ptr[frame_sequence_num - 2]])]

            if frame_sequence_num = 1 then
               frame_sequence_ptr[frame_sequence_num - 1] = 0
            else
               frame_sequence_ptr[frame_sequence_num - 1] = frame_sequence_ptr[frame_sequence_num - 2] +
frame_sequence_repeats[frame_sequence_ptr[frame_sequence_num - 2]] + 1
            endif
            if verbose = 1 then HSerout [$d,$a,"PTR (",dec (frame_sequence_num - 1),"): ",dec (frame_sequence_ptr[frame_sequence_num - 1])]
            if verbose = 1 then HSerout [$d,$a,"Defining sequence ",dec (frame_sequence_num - 1),". Start sending data with $nxxx for number of frame sets
in sequence, then $fxxx for frame number followed by $rxxx for repeats of that frame. $Txxx to terminate sequence once the sets have been defined"]
         else
            HSerout [$d,$a,"Please start a frame definition with $Fxxx"]
         endif
      endif

      if command[0] = 102 then ' define sequence frame                        f
         if frame_seq_started = 1 then
            frame_sequence_data[frame_sequence_ptr[frame_sequence_num - 1] + 1 + frame_sequence_frame_counter] = command_value
            if verbose = 1 then HSerout [$d,$a,"Sequence #",dec (frame_sequence_num - 1)," frame-in-sequence #",dec frame_sequence_frame_counter,"
frame #",dec command_value]
         else
            HSerout [$d,$a,"Please start frame sequence with $Sxxx"]
         endif
      endif

      if command[0] = 110 then ' define number of sets in this sequence          n
         if frame_seq_started = 1 then
            frame_sequence_repeats[frame_sequence_ptr[frame_sequence_num - 1] + 0] = command_value
            if verbose = 1 then HSerout [$d,$a,"Sequence #",dec (frame_sequence_num - 1)," has #",dec command_value," frame sets"]
         else
            HSerout [$d,$a,"Please start frame sequence with $Sxxx"]
         endif
      endif
```

```
    if command[0] = 112 then ' define frame-frame pause during repeats          p
      if frame_seq_started = 1 then
        frame_sequence_f_f_pause[frame_sequence_ptr[frame_sequence_num - 1] + 0] = command_value
        if verbose = 1 then HSerout [$d,$a,"Sequence #",dec (frame_sequence_num - 1)," frame-frame delay ",dec command_value," ms"]
      else
        HSerout [$d,$a,"Please start frame sequence with $Sxxx"]
      endif
    endif

    if command[0] = 114 then ' define sequence frame repeats                  r
      if frame_seq_started = 1 then
        if frame_sequence_frame_counter < frame_sequence_repeats[frame_sequence_ptr[frame_sequence_num - 1] + 0] then
          frame_sequence_repeats[frame_sequence_ptr[frame_sequence_num - 1] + 1 + frame_sequence_frame_counter] = command_value
          if verbose = 1 then HSerout [$d,$a,"Sequence #",dec (frame_sequence_num - 1)," set #",dec frame_sequence_frame_counter," frame #",dec
frame_sequence_data[frame_sequence_ptr[frame_sequence_num - 1] + 1 + frame_sequence_frame_counter]," repeats #",dec
frame_sequence_repeats[frame_sequence_ptr[frame_sequence_num - 1] + 1 + frame_sequence_frame_counter]]
          frame_sequence_frame_counter = frame_sequence_frame_counter + 1
        else
          if verbose = 1 then HSerout [$d,$a,"Maximum number of frame sets define; please terminate with $Txxx"]
        endif
      else
        HSerout [$d,$a,"Please start frame sequence with $Sxxx"]
      endif
    endif

    if command[0] = 88 then ' about to define coil in sequece no. X           X
      if frame_started = 1 then
        if command_value < number_of_coils_in_frame then
          define_frame_point_x = command_value
          HSerout [$d,$a,"Start defining coils from ",dec command_value]
        else
          HSerout [$d,$a,"Please keep coil definitions within N-1: ",dec number_of_coils_in_frame]
        endif
      else
        HSerout [$d,$a,"Please start a frame definition with $Fxxx"]
      endif
```

```
endif

if command[0] = 69 then ' sequence through all of the coils 1 to 64
   for cnt = 0 to 63 step 1

      coil_num = cnt

      for cnt2 = 0 to 25

         gosub energise
         gosub energise

     next cnt2

   next cnt
endif

if command[0] = 89 then ' point X (max N-1) = Y                    Y
   if frame_started = 1 then
      if define_frame_point_x < number_of_coils_in_frame then
         if verbose = 1 then HSerout [$d,$a,"Frame #",dec frame_number," seq no. ",dec define_frame_point_x," coil = ",dec command_value," (0-63)"]
         frame_data[frame_pointer[frame_number]+3 + define_frame_point_x] = command_value
         define_frame_point_x = define_frame_point_x + 1
      else
         HSerout [$d,$a,"Coil sequence out of range. Max coils = ",dec number_of_coils_in_frame]
      endif
   else
       HSerout [$d,$a,"Please start a frame definition with $Fxxx"]
   endif
endif

if command[0] = 71 then ' go - execute all stored commands          G
   HSerout [$d,$a,"Execute sequence #",dec command_value]
   gosub execute_commands
endif
```

```
   wend

   return

execute_commands:

   for cnt = 0 to (frame_sequence_repeats[frame_sequence_ptr[command_value] + 0] - 1) ' run through all frames in this sequence

      coil_coil_delay             =  frame_data[frame_pointer[frame_sequence_data[frame_sequence_ptr[command_value] + 1 + cnt]] + 1]
      coil_on_time                =  frame_data[frame_pointer[frame_sequence_data[frame_sequence_ptr[command_value] + 1 + cnt]] + 2]
      number_of_coils_in_frame    =  frame_data[frame_pointer[frame_sequence_data[frame_sequence_ptr[command_value] + 1 + cnt]] + 0]
      repeat_frame                =  frame_sequence_repeats[frame_sequence_ptr[command_value] + 1 + cnt]
      frame_frame_delay           =  frame_sequence_f_f_pause[frame_sequence_ptr[command_value] + 0]

      if verbose = 1 then HSerout [$d,$a,$d,$a,   "Frame #",dec frame_sequence_data[frame_sequence_ptr[command_value] + 1 + cnt]]
      if verbose = 1 then HSerout [$d,$a,        "Coil_coil_delay #",dec coil_coil_delay]
      if verbose = 1 then HSerout [$d,$a,        "Coil_on_time #",dec coil_on_time]
      if verbose = 1 then HSerout [$d,$a,        "Number_of_coils_in_frame #",dec number_of_coils_in_frame]
      if verbose = 1 then HSerout [$d,$a,        "Repeat frame #",dec repeat_frame]

      if verbose = 1 then
         for cnt2 = 0 to (frame_data[frame_pointer[frame_sequence_data[frame_sequence_ptr[command_value] + 1 + cnt]] + 0] - 1)

            HSerout [$d,$a,     "Coil #",dec frame_data[frame_pointer[frame_sequence_data[frame_sequence_ptr[command_value] + 1 + cnt]] + 3 + cnt2]]

         next cnt2
      endif

      if verbose = 1 then HSerout [$d,$a,$d,$a,   "Begin frame #",dec frame_sequence_data[frame_sequence_ptr[command_value] + 1 + cnt]," and repeat
#",dec repeat_frame," times"]

      coil_on_time = 100 * coil_on_time

      for cnt3 = 0 to (repeat_frame - 1)
```

```
        for cnt2 = 0 to (frame_data[frame_pointer[frame_sequence_data[frame_sequence_ptr[command_value] + 1 + cnt]] + 0] - 1) ' slide through each coil
in this frame (cnt)

            COIL_NUM = frame_data[frame_pointer[frame_sequence_data[frame_sequence_ptr[command_value] + 1 + cnt]] + 3 + cnt2]

            GOSUB energise

            pause coil_coil_delay ' ms

        next cnt2

        HSerout [$d,$a,   "Pause frame-frame"]

        pause frame_frame_delay

    next cnt3

    if verbose = 1 then HSerout [$d,$a,$d,$a,   "End frame #",dec frame_sequence_data[frame_sequence_ptr[command_value] + 1 + cnt]]

  next cnt

  HSerout [$d,$a,   "End of sequence"]

  return

energise:

  col_add0 = coil_num & $1
  col_add1 = (coil_num & $2) >> 1
  col_add2 = (coil_num & $4) >> 2
  row_add0 = (coil_num & $8) >> 3
  row_add1 = (coil_num & $10) >> 4
  row_add2 = (coil_num & $20) >> 5

  high green_led
```

```
    low col_dat  ' enable coil
    pauseus coil_on_time
    high col_dat ' disable coil

    low green_led

return

clear_arrays:

    for cnt = 0 to 999
      frame_data[cnt] = 0
    next cnt

    for cnt = 0 to 99
      frame_pointer[cnt] = 0
    next cnt

    for cnt = 0 to 499
      frame_sequence_data[cnt] = 0
    next cnt

    for cnt = 0 to 499
      frame_sequence_repeats[cnt] = 0
    next cnt

    for cnt = 0 to 299
      frame_sequence_ptr[cnt] = 0
    next cnt

    return
```