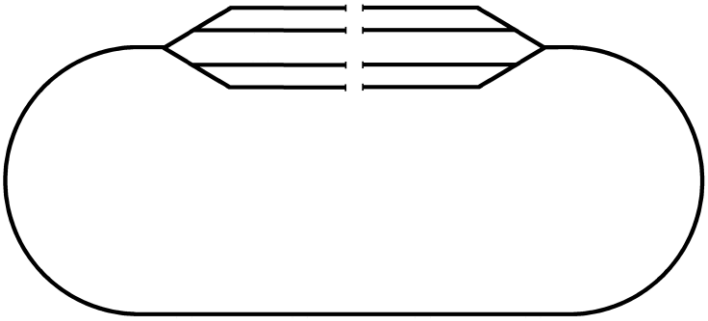
Fiddle Yard Shuffler

By Charlie Cotchin

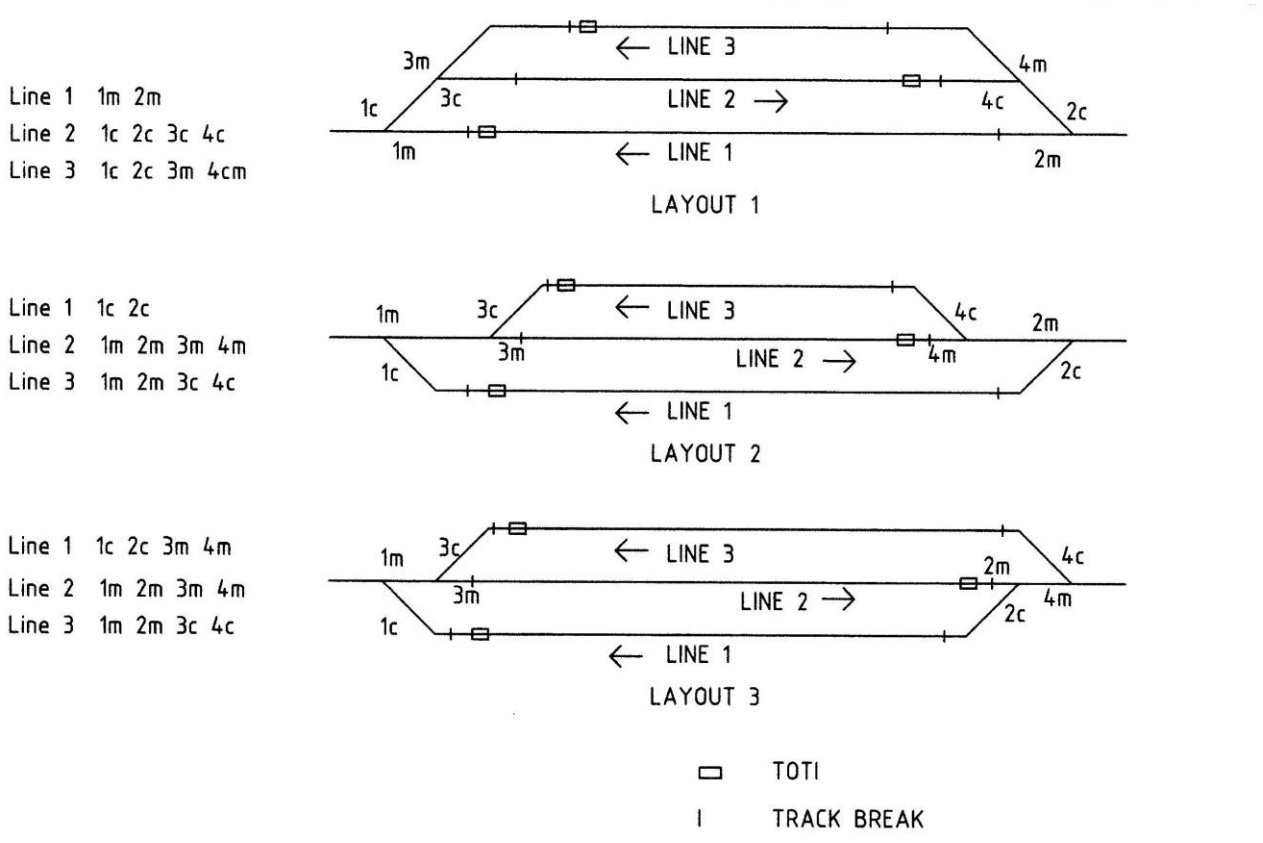
This project was inspired by the challenge set up by Davy at the end of his article on the Shuttle Kit Add-on. (September 2019 vol 53 no 3)



The challenge was to increase the number of sidings in the kit.
I thought that an automatic continuous loop connected to a fiddle yard would be of more use at an exhibition.
This is the result.

As I use solenoids to operate points this project is for use with this type of point.

The software could possibly be updated in the future to operate servos instead.
There are three layout types for a Fiddle Yard that could be easily set up to keep

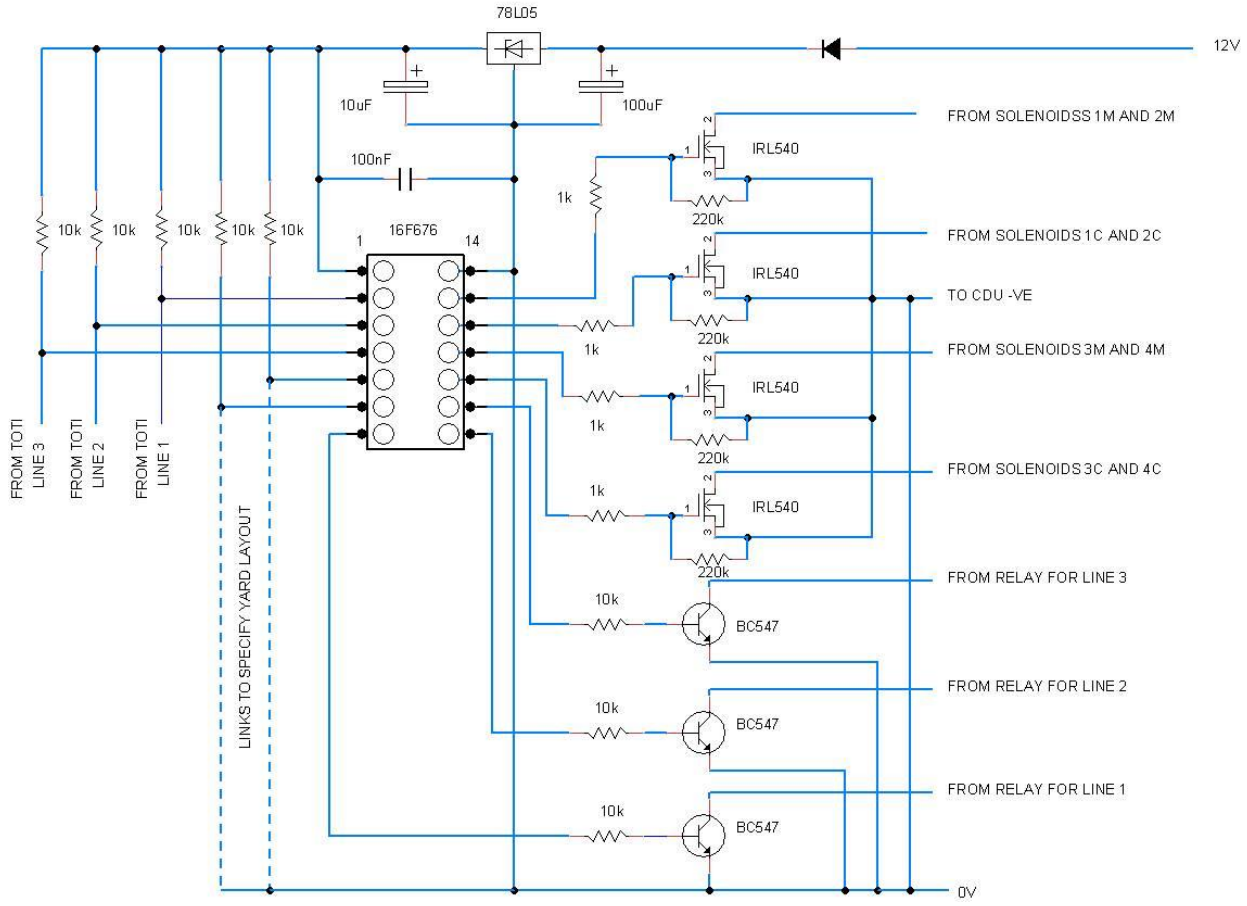


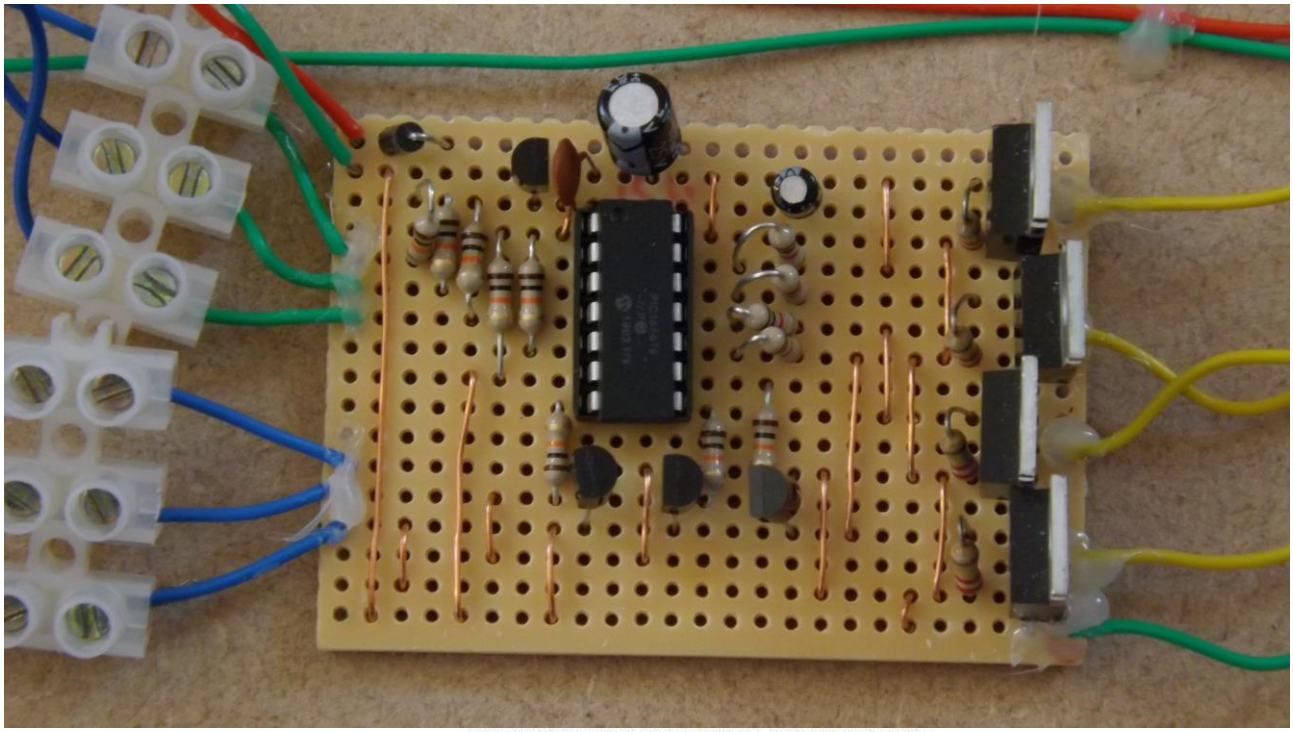
the software as simple as possible.
From this image it can be seen that the four points can be wired in pairs.

The positions of the Train on Track Indicators (TOTIs) are dependent on which way the operator wants the trains to run.

Tracks breaks are placed to fully isolate each line in the yard.
The connecting railway loop is connected to the same power supply as the lines.

Using these layouts the connections required to operate these layouts could be identified and so a schematic could be drawn.

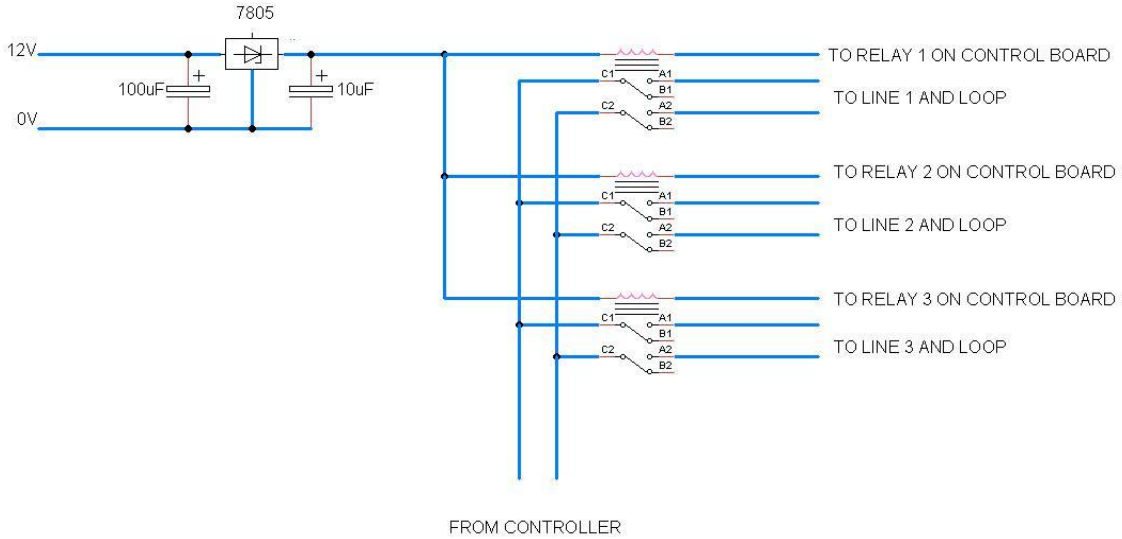




Control Board built

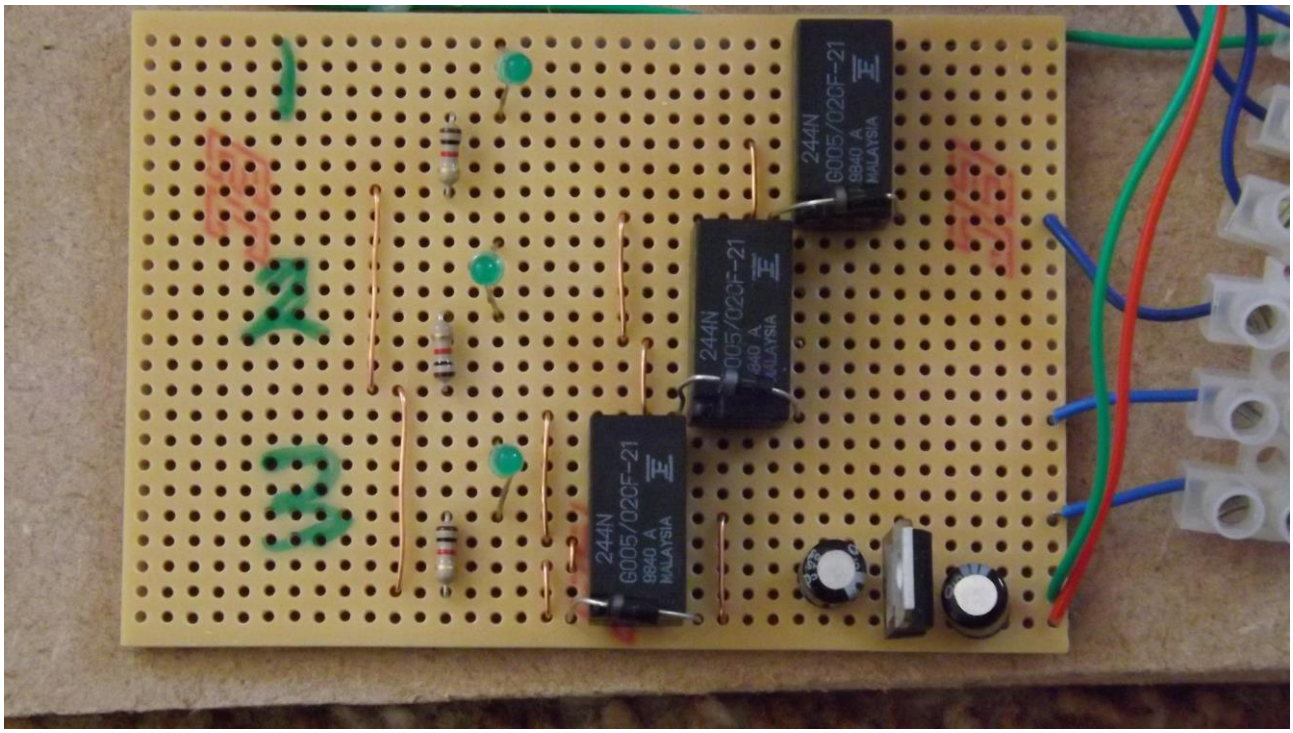
Relay Board

Here is a suggested schematic for the relay board.



As I am not using track power in this example power was taken from the 12v supply instead of from a controller.

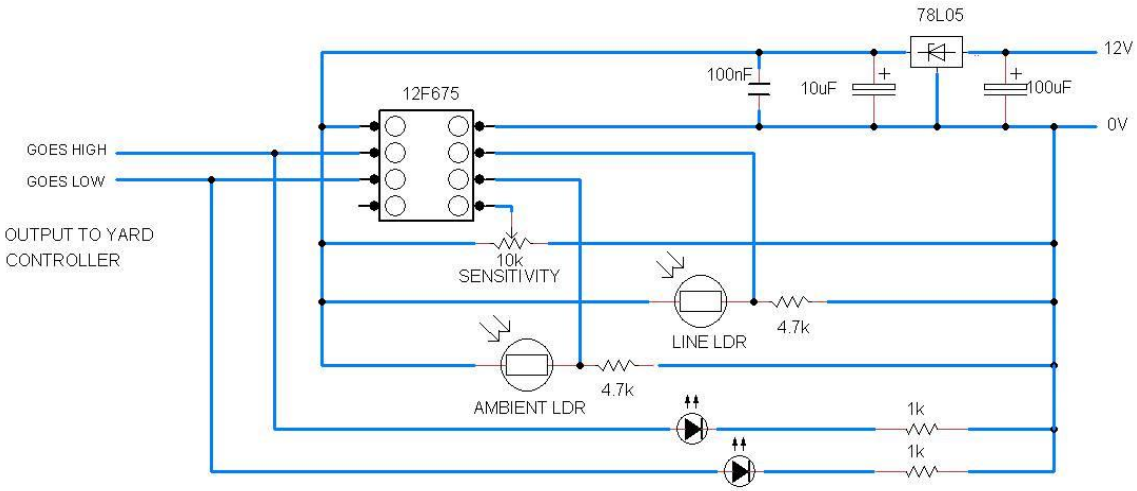
To indicate the application of power I fitted an LED and resistor across the power outputs to the separate lines.



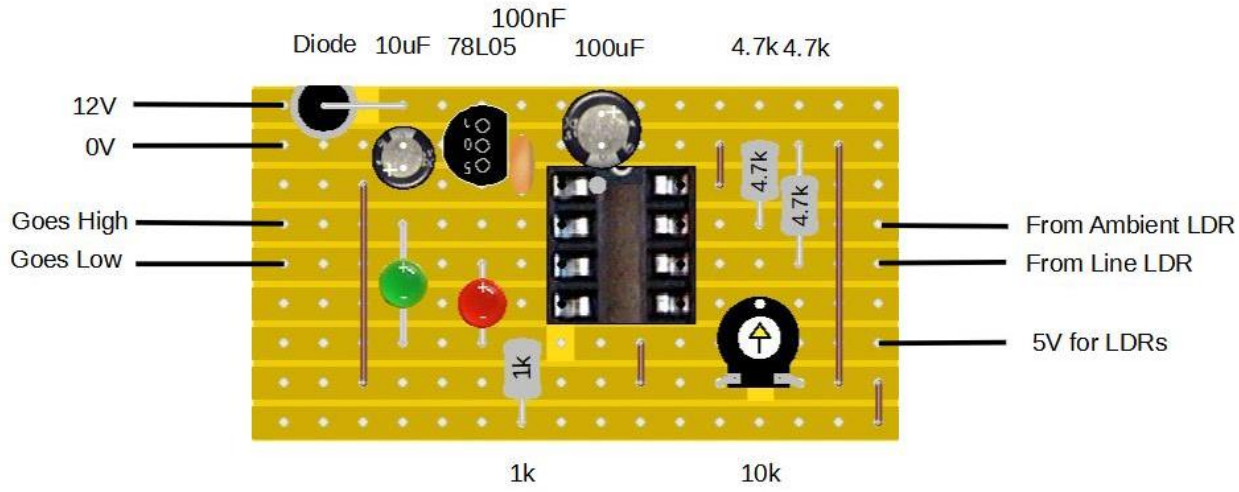
Power relays board built

Train Detectors

The type of TOTI had to be decided.
It was decided to use a new type using LDRs (light dependent resistors)
The LDR changes resistance depending on how much light falls on them.
A lot of light gives a low resistance while reducing the light level at the LDR gives a much higher resistance.
One LDR would be in a constant or ambient light area while the other would be put into shadow by a train.
Therefore comparing the output of the two LDRs could provide a method of

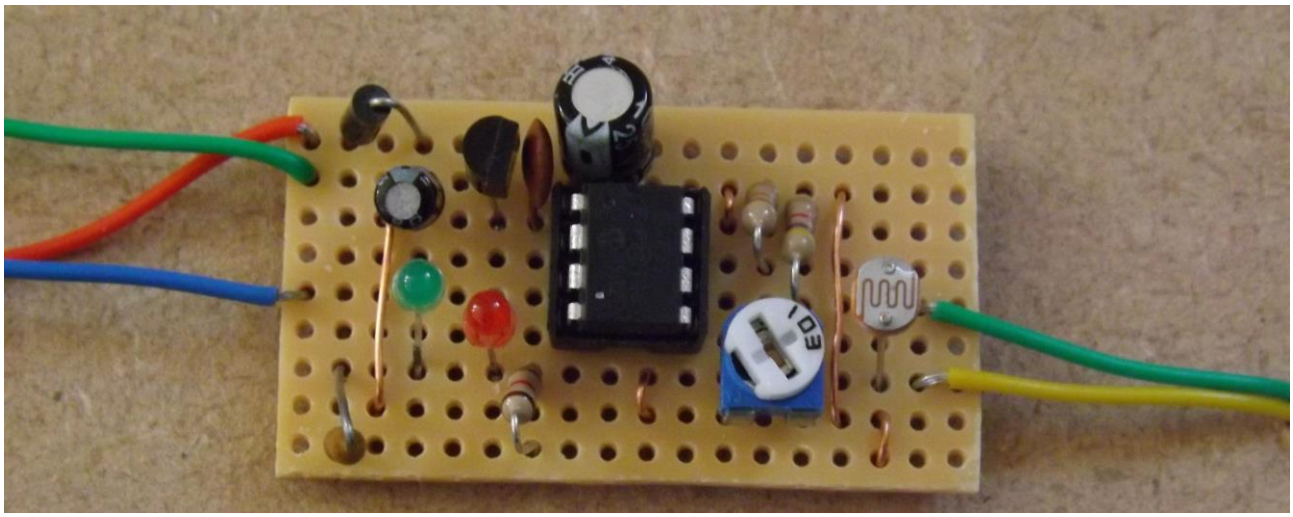


indicating the presence of a train or wagon on the track.
The sensitivity of the TOTI can be set (there are a number of TOTIs using LDRs



Board for Ambient LDR TOTI with Sensitivity

detailed in the journal.)
Other devices could be used such as a Laser TOTI, reed switch or magnets.
The output of whichever TOTI is used must go low when the line is occupied.


The red LED operates when the line is empty and the green LED operates when the line is occupied.

Ambient LDR TOTI built

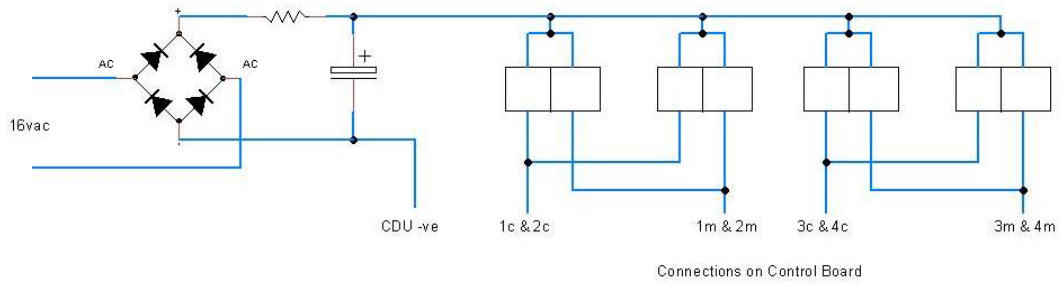
The CDU

The Capacitor Discharge Unit (CDU) is a simple unit consisting of a full wave rectifier, a resistor and a reasonably large capacitor (2200uF).

The power for the CDU is a 16v AC supply from a DC controller. The +ve side of the CDU is connected to a common connection on the solenoid that connects the two coils of the solenoid.

The separate coil connections of the solenoids are connected to the control board.

Points are connected in pairs as this reduces the number of outputs required from the PIC.



Connections on Control Board

The program code

Following on from Davy's presentation to a JALSIG and WOSAG sessions a list of requirements for the software could be made.
For the Shuffler
It is assumed, in this example, that lines 1 and 3 go anti-clockwise while line 2 goes clockwise. This sets the positions for the TOTIs to be near the exit of the line. This will result in a more interesting view at exhibitions.
PIC to be used 16F676 has 14 pins which will allow all the inputs and outputs required

Set all outputs low to ensure that the MOSFETs and transistors do not behave randomly on start up.

- Check line 1 TOTI
- If line occupied set points for line 1
- Switch relay on to provide power to line 1
- Train moves around layout
- Power stays on until train returns ie TOTI indicates line re-occupied
- Switch relay to turn power off
- Wait and then repeat for line 2 and then line 3

Sequence repeats for as long as required.

There is also a video clip that shows it in operation. Click to view

The code is written in JAL see below.

```
-- JAL 2.4
--  Programme to control a 3 line fiddle yard using solenoid operated points
--  powered by a CDU. Trains are powered and return to yard.
--  lines 1 and 3 go one way while line 2 goes the other depending on how the .
```

```
--  relays are connected to the track
--
--  Witten by C Cotchin  5432    08/11/20
-- ------------------------------------------------
--                    PIC16F676 pinouts
--
--                   +5v  o    o  0v
--    Line 1 TOTI ---> RA5  o    o  RA0  Solenoid 1m,2m --->
--    Line 2 TOTI ---> RA4  o    o  RA1  Solenoid 1c,2c --->
--    Line 3 TOTI ---> RA3  o    o  RA2  Solenoid 3m,4m --->
--    Layout 2    ---> RC5  o    o  RC0  Solenoid 3c,4c --->
--    Layout 3    ---> RC4  o    o  RC1  Line 3 Relay   --->
--    Line 1 Relay <---RC3  o    o  RC2  Line 2 Relay   --->
--
-- ------------------------------------------------

include 16f676

pragma target clock 4_000_000      -- oscillator frequency
pragma target OSC INTOSC_NOCLKOUT -- internal oscillator,no external outout
pragma target WDT  disabled        -- no watchdog
pragma target MCLR internal        -- reset internally

include delay
enable_digital_io()                -- disable analog I/O (if any)

-------------------------------------------------------
--  Set up the PIC pinouts
-------------------------------------------------------
alias detect1 is pin_A5   -- Line 1 TOTI
pin_A5_direction = input

alias detect2 is pin_A4   -- Line 2 TOTI
pin_A4_direction = input
```

```
alias detect3 is pin_A3   -- Line 3 TOTI
pin_A3_direction = input

alias relay1 is pin_C3    -- Line 1 power relay
pin_C3_direction = output

alias relay2 is pin_C2    -- Line 2 power relay
pin_C2_direction = output

alias relay3 is pin_C1    -- Line 3 power relay
pin_C1_direction = output

alias solen12m is pin_A0  -- Solenoids 1m,2m
pin_A0_direction = output

alias solen12c is pin_A1  -- Solenoids 1c,2c
pin_A1_direction = output

alias solen34m is pin_A2  -- Solenoids 3m,4m
pin_A2_direction = output

alias solen34c is pin_C0  -- Solenoids 3c,4c
pin_C0_direction = output

alias layout2 is pin_C5   -- goes low for Layout 2
pin_C5_direction = input

alias layout3 is pin_C4   -- goes low for Layout 3
pin_C4_direction = input

var bit  layout1
var byte cdu_pulse,  power_delay,  toti_delay,  cdu_charge
var byte between_trains,  sequence_delay
--
-- ------------------------------------------------
```

```
--                  M A I N   P R O G R A M
-- ------------------------------------------------
-- Set default outputs for Mosfet inputs used to drive solenoids
    solen12m = low
    solen12c = low
    solen34m = low
    solen34c = low
--
-- Set default outputs for transitors controlling power relays
    relay1 = low
    relay2 = low
    relay3 = low
--
--  set layout 1 to zero
    layout1 = 0
--
-- set delay times
    cdu_pulse = 50           --  time to pulse cdu ms
    power_delay = 3          --  delay to switch on power after points set s
    cdu_charge = 1           --  time for cdu to recharge s
    between_trains = 5       --  time between trains s
    sequence_delay = 5       --  time between sequence repeat s
    toti_delay = 5           --  time to let train pass TOTI

procedure power_line_1 is      --  to power up line 1
    delay_1s(power_delay)     --  wait before switching on power
    relay1 = high             --  switch on power to line 1
    delay_1s(toti_delay)      --  wait to allow train to pass TOTI

    while detect1 == high loop --  wait until train returns to yard
    end loop

    relay1 = low              --  switch off power to line 1

end procedure
```

```
procedure power_line_2 is      --  to power up line 2
    delay_1s(power_delay)     --  wait before switching on power
    relay2 = high             --  switch on power to line 2
    delay_1s(toti_delay)      --  wait to allow train to pass TOTI

    while detect2 == high loop --  wait until train returns to yard
    end loop

    relay2 = low              --  switch off power to line 2

end procedure

procedure power_line_3 is      --  to power up line 3
    delay_1s(power_delay)     --  wait before switching on power
    relay3 = high             --  switch on power to line 3
    delay_1s(toti_delay)      --  wait to allow train to pass TOTI

    while detect3 == high loop --  wait until train returns to yard
    end loop

    relay3 = low              --  switch off power to line 3

end procedure

procedure points_1c2c is       --  to operate points 1 & 2 to curved
    solen12c = high           --  allow CDU to connect to points
    delay_1ms(cdu_pulse)      --  pulse CDU
    solen12c = low            --  disconnect CDU

end procedure

procedure points_1m2m is       --  to operate points 1& 2 to main
    solen12m = high           --  allow CDU to connect to points
    delay_1ms(cdu_pulse)      --  pulse CDU
```

```
    solen12m = low          --  disconnect CDU

end procedure

procedure points_3c4c is      --  to operate points 3,4 to curve
    solen34c = high         --  allow CDU to connect to points
    delay_1ms(cdu_pulse)     --  pulse CDU
    solen34c = low          --  disconnect CDU

end procedure

procedure points_3m4m is
    solen34m = high         --  allow CDU to connect to points
    delay_1ms(cdu_pulse)     --  pulse CDU
    solen34m = low          --  disconnect CDU

end procedure

    if layout2 == high & layout3 == high then layout1 = 1  --  set layout
    end if                                          --  being used

forever loop

--  layout 1 being used
  if layout1 == 1 then

   if detect1 == low then      --  line 1 occupied
     points_1m2m              --  set points for line 1
     power_line_1             --  train has returned to yard

   end if

  delay_1s(between_trains)      --  delay between trains

   if detect2 == low then      --  line 2 occupied
```

```
    points_1c2c              --  set points for line 2

    delay_1s(cdu_charge)     --  wait for CDU to recharge

    points_3c4c
    power_line_2             --  train has returned to yard

  end if

delay_1s(between_trains)     --  delay between trains

  if detect3 == low then     --  line 3 occupied
    points_1c2c              --  set points for line 3

    delay_1s(cdu_charge)     --  wait for CDU to recharge

    points_3m4m
    power_line_3             --  train has returned to yard

  end if

delay_1s(sequence_delay)     --  delay before returning to line 1

end if

--  layout 2 being used
 if layout2 == low then

   if detect1 == low then    --  line 1 occupied
     points_1c2c             --  set points for line 1
     power_line_1            --  train has returned yard

   end if

delay_1s(between_trains)     --  delay between trains
```

```
   if detect2 == low then       --  line 2 occupied
     points_1m2m

     delay_1s(cdu_charge)       --  wait for CDU to recharge

     points_3m4m                --  set points for line 2
     power_line_2               --  train has returned to yard

   end if

 delay_1s(between_trains)       --  delay between trains

   if detect3 == low then       --  line 3 occupied
     points_1m2m                --  set points for line 3

     delay_1s(cdu_charge)       --  wait for CDU to recharge

     points_3c4c
     power_line_3               --  train has returned to yard

   end if

 delay_1s(sequence_delay)       --  delay before returning to line 1

  end if

--  layout 3 being used
  if layout3 == low then

   if detect1 == low then       --  line 1 occupied
     points_1c2c                --  set points for line 1

     delay_1s(cdu_charge)       --  wait for CDU to recharge
```

```
      points_3m4m
      power_line_1              --  train has returned to yard

    end if

  delay_1s(between_trains)        --  delay between trains

    if detect2 == low then        --  line 2 occupied
      points_1m2m               --  set points for line 2

      delay_1s(cdu_charge)       --  wait for CDU to recharge

      points_3m4m
      power_line_2              --  train has returned to yard

    end if

  delay_1s(between_trains)        --  delay between trains

    if detect3 == low then        --  line 3 occupied
      points_1m2m               --  set points for line 2

      delay_1s(cdu_charge)       --  wait for CDU to recharge

      points_3c4c
      power_line_3              --  train has returned to yard

    end if

  delay_1s(sequence_delay)      --  delay before returning to line 1

   end if

end loop
```