Infra-red Arduino Point and Route Controller

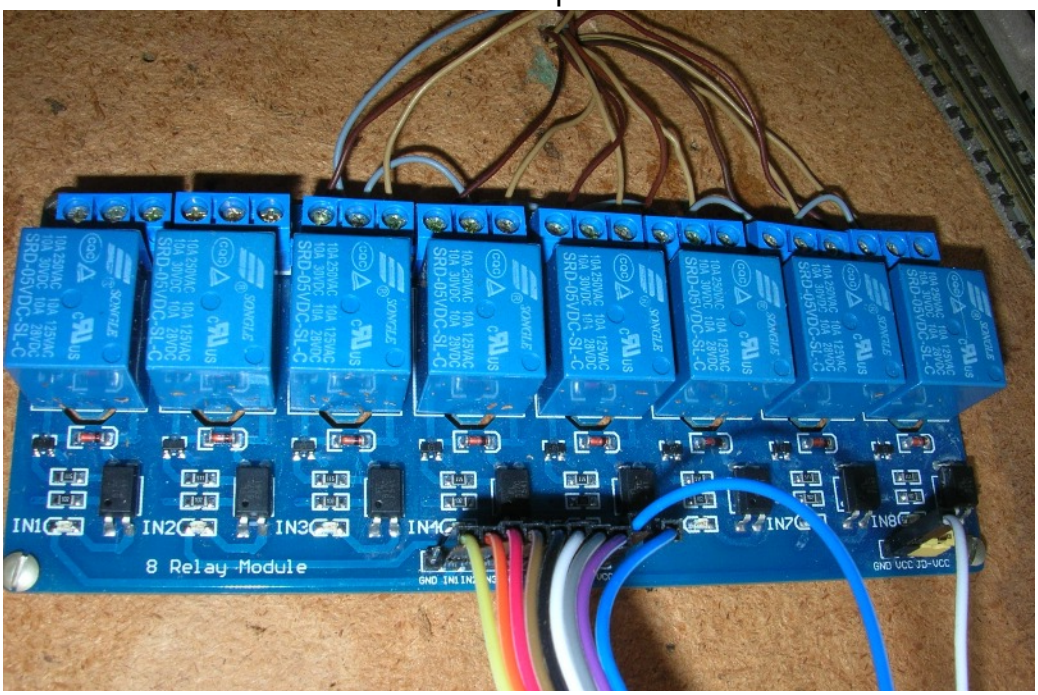
One of the weaknesses of diode matrix when dealing with points for a fiddle yard is that all the point motors will ‘fire’ at once. With Seep motors and the like drawing up to 2 amps each at the same time makes great demands on the power supply and/or Capacitor Discharge Unit. This approach allows for multiple points to be fired individually in a sequence avoiding this problem.

Using a member of the Arduino family to programmatically operate the points via a set of in-expensive relays was experimented with and found to be very effective and cost efficient.

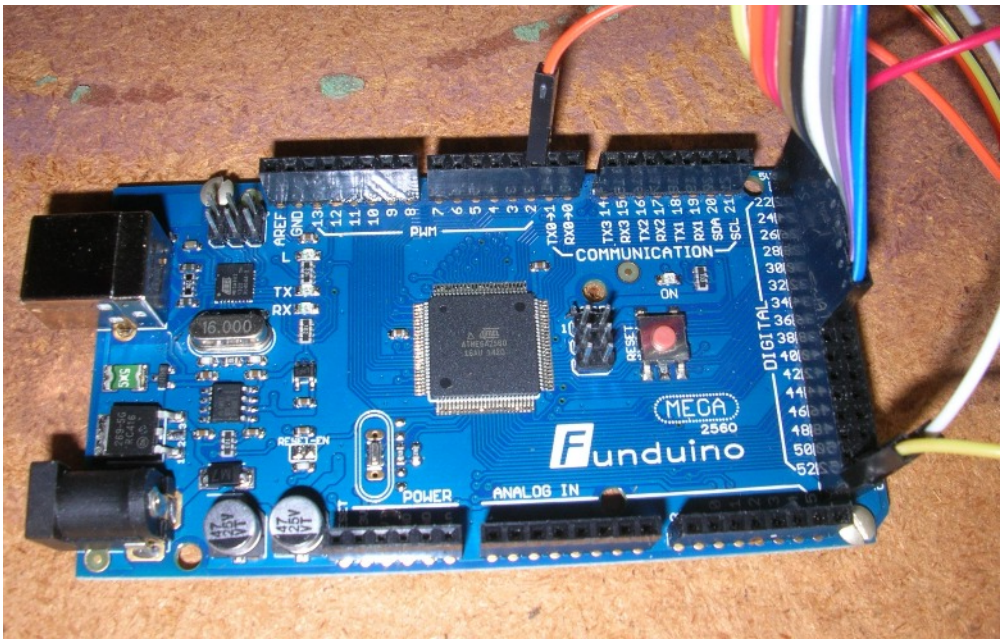
The use of an Arduino Uno would do this job but the use of the more powerful Arduino Mega 2560 would be a better place to start as it has more connections and memory.

The output side of the relays would be wired to the points as though they were normal switch controls. In the case of self-cancelling point motors, such as Fleischmann, one relay per point would suffice. With Peco, Gaugemaster and Seep motors one relay is required for each winding on the motor but this extra expense is still minor - the necessary code variation is shown later.

Wires to the points



Wires to the Arduino Mega



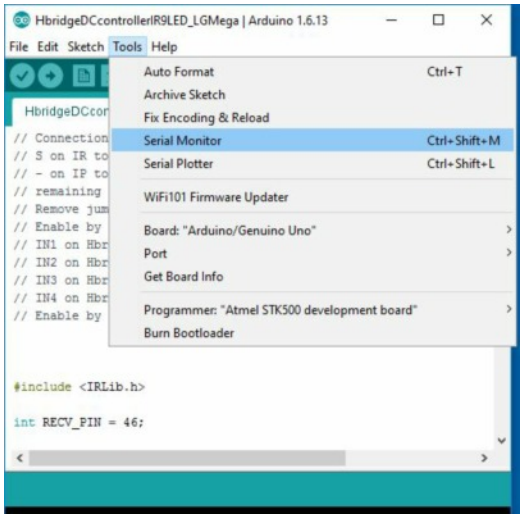
Connections
Arduino Pin 22 to Relay board In1
Arduino Pin 24 to Relay board In2
Arduino Pin 26 to Relay board In3
Arduino Pin 28 to Relay board In4
Arduino Pin 30 to Relay board In5
Arduino Pin 32 to Relay board In6
Arduino Pin 34 to Relay board In7
Arduino Pin 36 to Relay board In8
Arduino Pin 5v (next to Pin 22) to Relay board VCC
Arduino Pin GND (next to Pin 52) to Relay board GND

In this example, an infra-red remote was used to control the points through the Arduino. Whilst control buttons and/or switches could be used, the IR remote saves a lot more wiring and does not add much to the the cost of the project.

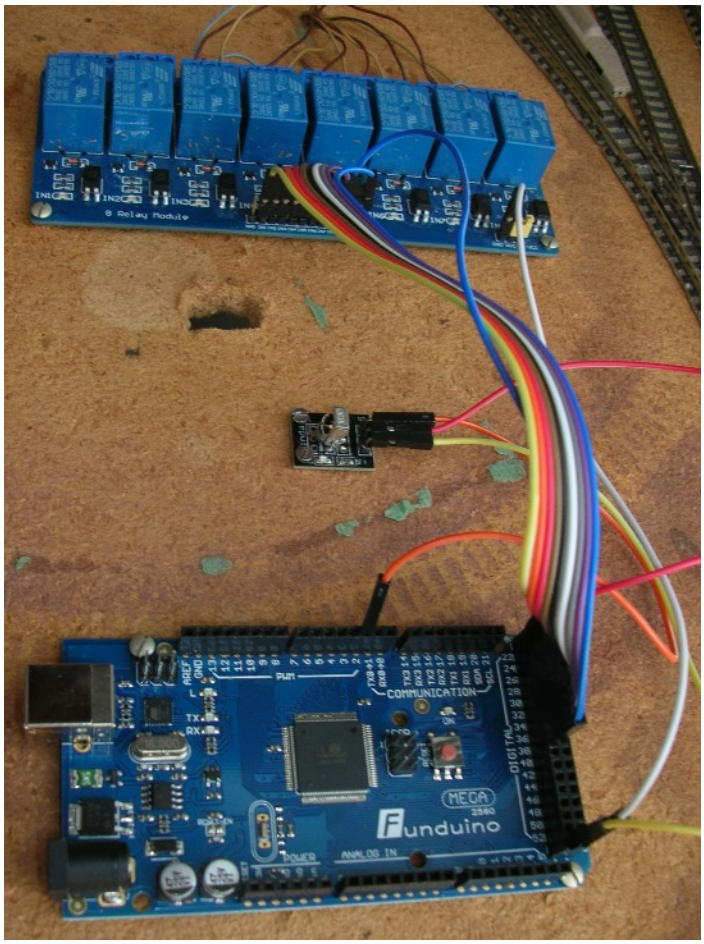
Arduino Pin 2 to IR receiver S
Arduino Pin GND (next to Pin 53) to IR receiver - (minus)
Arduino Pin 5v (Next to Pin 23) to to IR centre connection

IR Remote Codes

TV Infra-red remotes output a stream of numbers, different for each button pressed. The first task is to discover what each of these numbers are. To do this, a simple program (or 'sketch' to use the Arduino parlance) is required. The Arduino programming software, downloadable free and safely from https://www.arduino.cc/en/Main/Software for Windows, Linux or Mac, has a facility called the 'Serial Monitor' that will allow these codes to be viewed. From this a list of what numbers are produced by each button can be prepared. Once these codes are known then it will be possible to proceed to writing the sketch for the IR Train Controller.



There is often a repeating code (84696351 in this case) that outputs even when a button is not being pressed and these should be ignored. It will be obvious which is the correct number at the point of pressing the relevant button.



The complete setup looks like this.

In this example, it was decided to use only 6 Fleischmann points and so relays IN1 and In2 are not used.

Also, when wiring up the points for demonstration, the shortness of the point motor wiring took precedence over a logical order. Consequently, the example sketch will reflect this.

The relay board was powered from the Arduino Mega, however, in everyday use it is recommended that a separate 5v power source is provided to it. On the extreme right is a jumper that would then be removed and the VCC connection for the separate power source is revealed.

Take care not to get the +Pos and -Neg connections reversed as this will damage the relay board.

16 relay bank boards are available but have to be powered by a separate 12v regulated source instead of a 5v source.

Parts List

8 Channel Relay Board Module for Arduino

£5.50

ATmega2560-16AU CH340G MEGA 2560 R3 Board and USB Cable

£9.99

Remote Control and Sensor

£2.59

AC 100-240V to DC 12V/2A Converter UK Regulated Power Supply Adapter Transformer = 5.5 mm x 2.1 mm plug

£6.79

Sketch for reading IR Remote Codes in Serial Monitor

```
#include <IRLib.h>

int RECV_PIN = 2;

IRrecv My_Receiver(RECV_PIN);
IRdecode My_Decoder;
IRdecodeHash My_Hash_Decoder;

void setup() {
  My_Receiver.enableIRIn(); // Start the receiver
  Serial.begin(9600);
  delay(2000);while(!Serial);//delay for Leonardo
}

void loop() {
  if (My_Receiver.GetResults(&My_Decoder)) {      //Puts results in My_Decoder
    My_Hash_Decoder.copyBuf(&My_Decoder);      //copy the results to the hash decoder
    My_Decoder.decode();
    My_Hash_Decoder.decode();
    Serial.println(My_Hash_Decoder.hash, DEC);
    delay(1000);
    My_Receiver.resume();
  }
}
```

Example Sketch for Point control

```
#include <IRLib.h>

int RECV_PIN = 2;

IRrecv My_Receiver(RECV_PIN);
IRdecode My_Decoder;
IRdecodeHash My_Hash_Decoder;
// ******* Define the Xinda button codes
#define RIGHT_ARROW       553536955
#define LEFT_ARROW        1386468383
#define OK                465573243
#define UP_ARROW           5316027
#define DOWN_ARROW        2747854299
#define BUTTON_0          465573243
#define BUTTON_1          3238126971
#define BUTTON_2          2538093563
#define BUTTON_3          4039382595
#define BUTTON_4          2534850111
#define BUTTON_5          1033561079
#define BUTTON_6          1635910171
#define BUTTON_7          2351064443
#define BUTTON_8          1217346747
#define BUTTON_9          71952287
#define star               851901943
#define hashkey           1053031451
#define JUNK              84696351
#define JUNK1             84696349

#define TURN  1
#define CLOSE 0
```

```
// Arduino Digital I/O pin number
#define Relay_1  22   //unused In1
#define Relay_2  24   //unused In2
#define Relay_3  26   //button 6 In3
#define Relay_4  28   //button 5 In4
#define Relay_5  30   //button 2 In5
#define Relay_6  32   //button 1 In6
#define Relay_7  34   //button 4 In7
#define Relay_8  36   //button 3 In8

int Pin02Read ;
int Pin03Read ;
int Pin04Read ;
int Pin05Read ;
int Pin06Read ;
int Pin07Read ;
int Pin08Read ;
int Pin09Read ;
int Pin10Read ;
long unsigned mvalue ;
int mdelay ;

void setup() {
  My_Receiver.enableIRIn(); // Start the receiver
  Serial.begin(9600);
  delay(2000); while (!Serial); //delay for Leonardo
  pinMode(Relay_1, OUTPUT);
  pinMode(Relay_2, OUTPUT);
  pinMode(Relay_3, OUTPUT);
  pinMode(Relay_4, OUTPUT);
  pinMode(Relay_5, OUTPUT);
  pinMode(Relay_6, OUTPUT);
  pinMode(Relay_7, OUTPUT);
  pinMode(Relay_8, OUTPUT);

  digitalWrite(Relay_1, CLOSE);
  digitalWrite(Relay_2, CLOSE);
  digitalWrite(Relay_3, CLOSE);
  digitalWrite(Relay_4, CLOSE);
  digitalWrite(Relay_5, CLOSE);
  digitalWrite(Relay_6, CLOSE);
  digitalWrite(Relay_7, CLOSE);
  digitalWrite(Relay_8, CLOSE);
  mvalue = 0 ;
  mdelay = 500 ;
}

void loop() {
  mvalue == 0 ;
  if (My_Receiver.GetResults(&My_Decoder)) {//Puts results in My_Decoder
    My_Hash_Decoder.copyBuf(&My_Decoder);//copy the results to the hash decoder
    My_Decoder.decode();
    My_Hash_Decoder.decode();
    //Serial.println(My_Hash_Decoder.hash, DEC); // Do something interesting with this value
    My_Receiver.resume();
    mvalue = My_Hash_Decoder.hash ;
  }
  if (mvalue != 0) {
    Serial.println(mvalue);
  }
```

```
switch (mvalue) {
  case 0 :
    break ;
  case JUNK :
    mvalue = 0 ;
    // Do Nothing
    break ;
  case JUNK1 :
    mvalue = 0 ;
    // Do Nothing
    break ;
  case BUTTON_0 :        // set all points to Close
    digitalWrite(Relay_1, CLOSE);
    delay(mdelay) ;
    digitalWrite(Relay_2, CLOSE);
    delay(mdelay) ;
    digitalWrite(Relay_3, CLOSE);
    delay(mdelay) ;
    digitalWrite(Relay_4, CLOSE);
    delay(mdelay) ;
    digitalWrite(Relay_5, CLOSE);
    delay(mdelay) ;
    digitalWrite(Relay_6, CLOSE);
    delay(mdelay) ;
    digitalWrite(Relay_7, CLOSE);
    delay(mdelay) ;
    digitalWrite(Relay_8, CLOSE);
    mvalue = 0 ;
    break ;
  case BUTTON_1 :
    Pin02Read = digitalRead(Relay_6);
    if (Pin02Read == TURN) {
      digitalWrite(Relay_6, CLOSE);
    }
    else {
      digitalWrite(Relay_6, TURN);
    }
    mvalue = 0 ;
    delay(1000) ;
    break ;
  case BUTTON_2 :
    Pin05Read = digitalRead(Relay_5);
    if (Pin05Read == TURN) {
      digitalWrite(Relay_5, CLOSE);
    }
    else {
      digitalWrite(Relay_5, TURN);
    }
    mvalue = 0 ;
    delay(1000) ;
    break ;
  case BUTTON_3 :
    Pin04Read = digitalRead(Relay_8);
    if (Pin04Read == TURN) {
      digitalWrite(Relay_8, CLOSE);
    }
    else {
      digitalWrite(Relay_8, TURN);
    }
    mvalue = 0 ;
```

```
  delay(1000) ;
  break ;
 case BUTTON_4 :
  Pin06Read = digitalRead(Relay_7);
  if (Pin06Read == TURN) {
   digitalWrite(Relay_7, CLOSE);
  }
  else {
   digitalWrite(Relay_7, TURN);
  }
  mvalue = 0 ;
  delay(1000) ;
  break ;
 case BUTTON_5 :
  Pin06Read = digitalRead(Relay_4);
  if (Pin06Read == TURN) {
   digitalWrite(Relay_4, CLOSE);
  }
  else {
   digitalWrite(Relay_4, TURN);
  }
  mvalue = 0 ;
  delay(1000) ;
  break ;
 case BUTTON_6 :
  Pin07Read = digitalRead(Relay_3);
  if (Pin07Read == TURN) {
   digitalWrite(Relay_3, CLOSE);
  }
  else {
   digitalWrite(Relay_3, TURN);
  }
  mvalue = 0 ;
  delay(1000) ;
  break ;
 case BUTTON_7 :                 //Route 1 points 1 and 2
  Pin02Read = digitalRead(Relay_6);
  if (Pin02Read == CLOSE) {
   digitalWrite(Relay_6, TURN);
  }
  else {
   digitalWrite(Relay_6, CLOSE);
  }
  delay(mdelay) ;
  Pin05Read = digitalRead(Relay_5);
  if (Pin05Read == CLOSE) {
   digitalWrite(Relay_5, TURN);
  }
  else {
   digitalWrite(Relay_5, CLOSE);
  }
  mvalue = 0 ;
  delay(1000) ;
  break ;
 case BUTTON_8 :                 // Route 2 points 3 and 4
  Pin04Read = digitalRead(Relay_8);
  if (Pin04Read == CLOSE) {
   digitalWrite(Relay_8, TURN);
  }
  else {
```

```
      digitalWrite(Relay_8, CLOSE);
    }
    delay(mdelay) ;
    Pin06Read = digitalRead(Relay_7);
    if (Pin06Read == CLOSE) {
      digitalWrite(Relay_7, TURN);
    }
    else {
      digitalWrite(Relay_7, CLOSE);
    }
    mvalue = 0 ;
    delay(1000) ;
    break ;
  case BUTTON_9 :             // Route 3 points 1,2,3,4
    Pin02Read = digitalRead(Relay_6);
    if (Pin02Read == CLOSE) {
      digitalWrite(Relay_6, TURN);
    }
    else {
      digitalWrite(Relay_6, CLOSE);
    }
    delay(mdelay) ;
    Pin05Read = digitalRead(Relay_5);
    if (Pin05Read == CLOSE) {
      digitalWrite(Relay_5, TURN);
    }
    else {
      digitalWrite(Relay_5, CLOSE);
    }
    delay(mdelay) ;
    Pin04Read = digitalRead(Relay_8);
    if (Pin04Read == CLOSE) {
      digitalWrite(Relay_8, TURN);
    }
    else {
      digitalWrite(Relay_8, CLOSE);
    }
    delay(mdelay) ;
    Pin06Read = digitalRead(Relay_7);
    if (Pin06Read == CLOSE) {
      digitalWrite(Relay_7, TURN);
    }
    else {
      digitalWrite(Relay_7, CLOSE);
    }
    mvalue = 0 ;
    delay(1000) ;
    break ;
  }
}
```

Code variation for Peco, Gaugemaster and Seep motors

The current to each solenoid winding has to be turned ON and then OFF after a short delay.

```
digitalWrite(Relay_1, ON);
delay(CDU);
digitalWrite(Relay_1, OFF);
```

‘CDU’ is the variable containing the desired delay e.g 500 is 0.5 seconds, 250 is 0.25 seconds. The above extra code would be equal to one CLOSE or one TURN as appropriate.