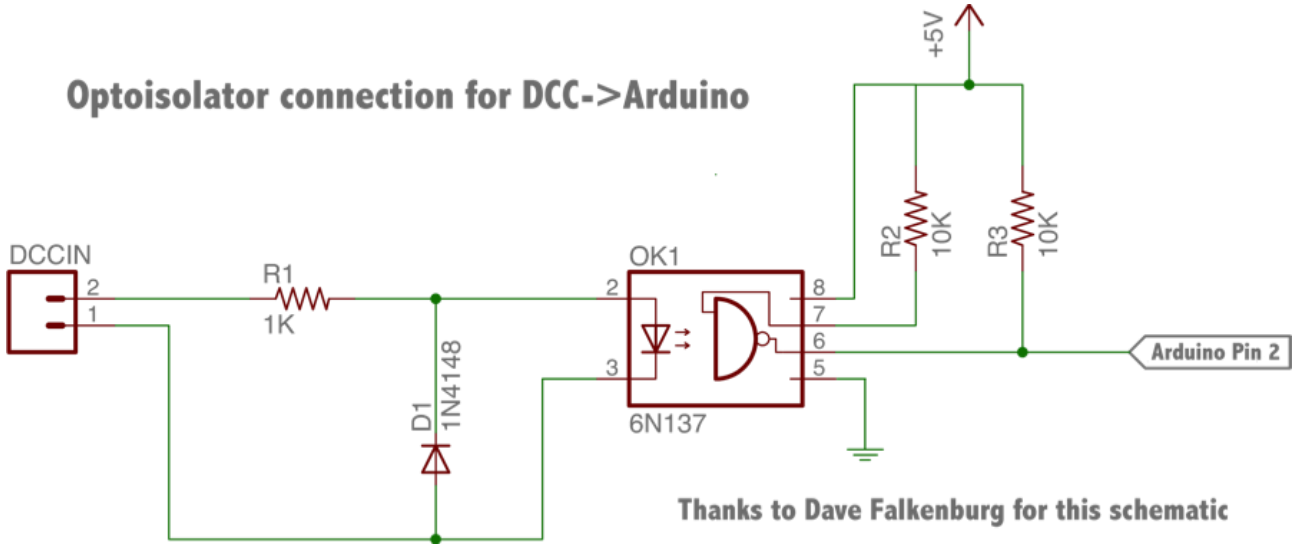
Progetto Decoder DCC universale

Prima di trasformare il nostro Arduino in un decoder DCC e' necessario costruire una semplice interfaccia che adatti i livelli di tensione fra i due sistemi.

Il circuito prevede un optoisolatore, un diodo veloce da non sostituire come modello e tre resistenze sa ¼ di Watt.

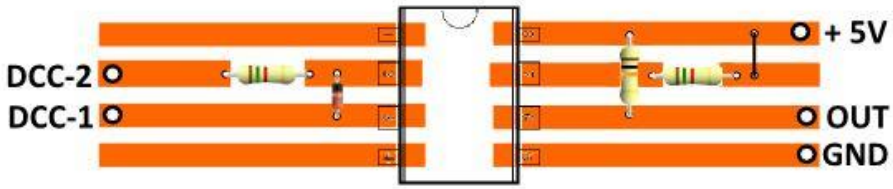


Fonte: http://www.mynabay.com/dcc_monitor/

Partendo da una piastrina ramata Veroboard a strisce (*strip board*) abbiamo tagliato le stesse con un cutter nelle posizioni indicate in nero per poi montare i componenti dal lato opposto facendo ben attenzione al verso.

Vero Board Prototyping Board 25 x 64 MM

Vista dal lato del rame: interrompere le strisce di rame nelle posizioni indicate.



Vista dal lato dei componenti

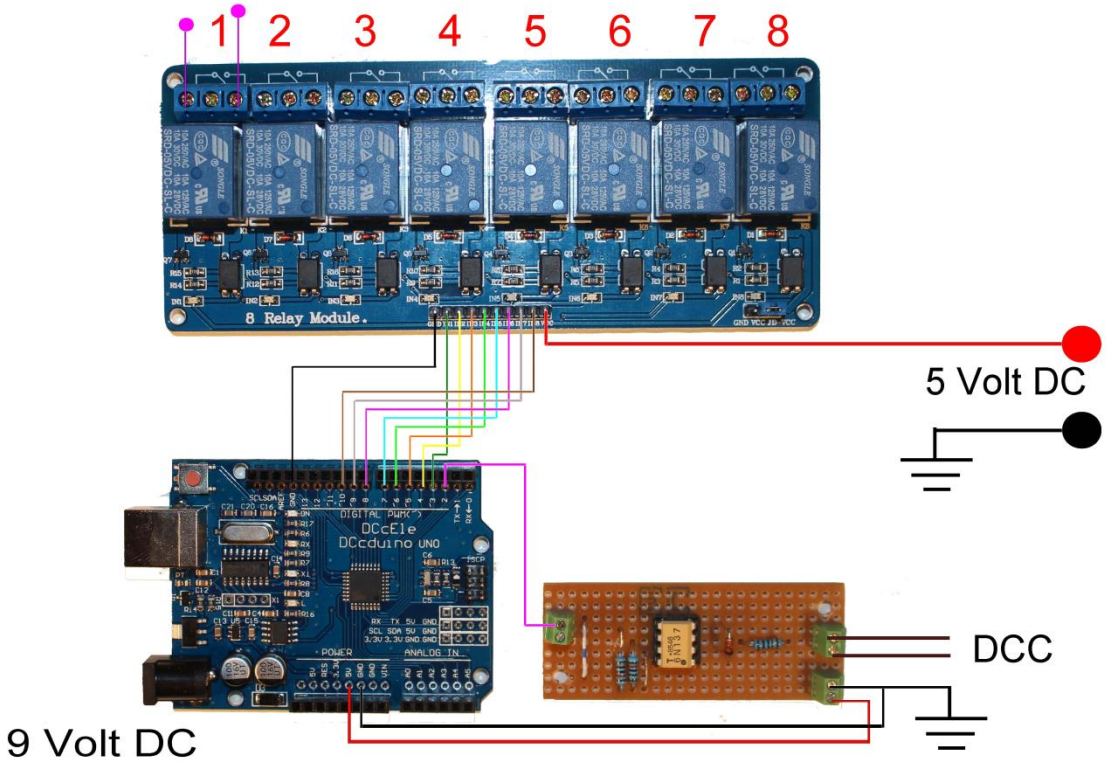
A questo punto si possono collegare fra di loro Arduino, il decoder e gli attuatori,

nel caso illustrato 8 relè che risponderanno ad 8 diversi indirizzi DCC.

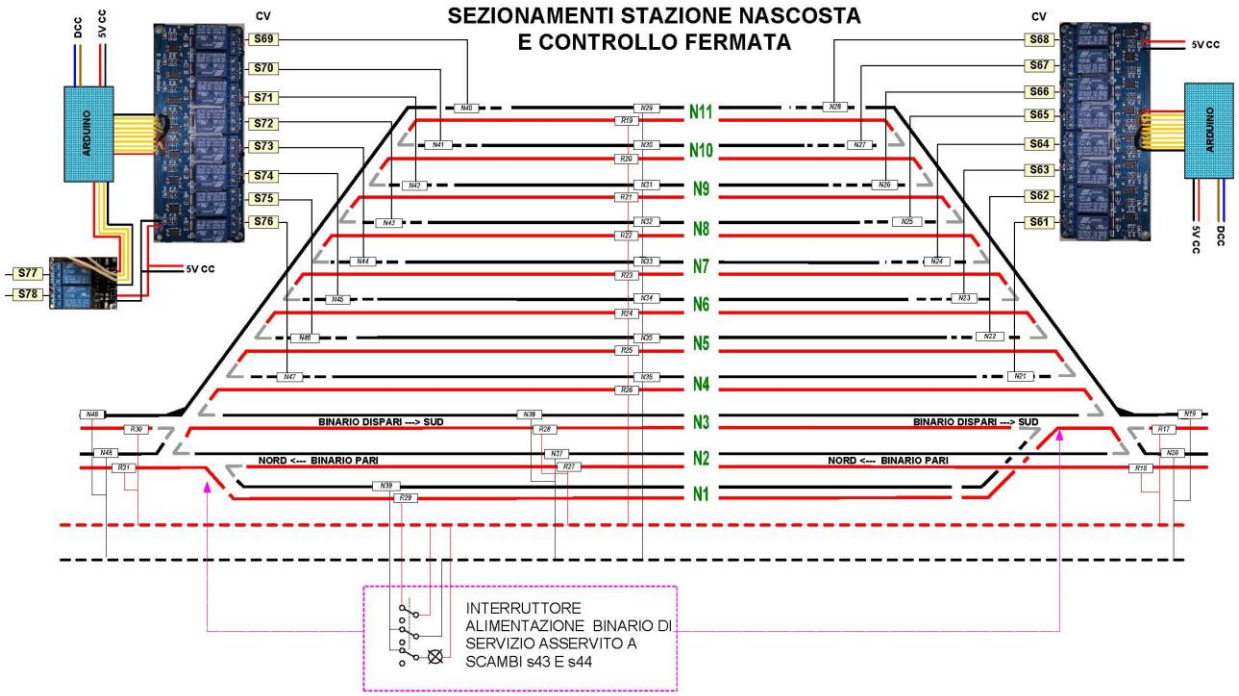
A seconda del tipo di Arduino impiegato si possono collegare fino ad una sessantina di attuatori.

Nelle figure successive riprese dall'articolo su TTM si vede come questi rele' vadano ad attivare

Altrettanti sezionamenti della stazione nascosta.



Il programma



Il programma originale da caricare su Arduino, lo *sketch* in Arduinese, puo' essere scaricato dal Sito

https://github.com/MynaBay/DCC_Decoder

insieme con le librerie necessarie e alcuni esempi e permette di assegnare ad ogni indirizzo diverse azioni quali la durata dell'impulso e il lampeggio ovvero se il segnale in uscita è analogico o digitale.

Per nostra comodità sul plastico di Fossacesia abbiamo eliminato dal codice tutte le funzionalità non necessarie.

Lo *sketch* puo' essere scaricato da

http://www.apud.it/public/DCC/dec8rel/D ... le_v02.ino.

Potete scaricare la libreria con gli esempi nella versione da noi utilizzata da qui

http://www.apud.it/public/DCC/dec8rel/d ... der.v4.zip

NB con alcuni clone di Arduino i driver presenti nel SW ufficiale non funzionano.

Quelli da noi utilizzati possono essere scaricati da qui

http://www.apud.it/public/DCC/dec8rel/CH341SER.ZIP

Per coloro che fossero interessati presentiamo il codice da noi usato commentandone le sezioni più interessanti.

```
""" Carichiamo la libreria di Minabay

#include <DCC_Decoder.h>

""" Definiamo le costanti del nostro Decoder

""" inserire fra parentesi quadre il numero massimo di relè da collegare a questo Decoder; sono 16 per una scheda Arduino Uno; oltre 50 per una Arduino Mega

DCCAccessoryAddress gAddresses(8);


""" indirizzo DCC iniziale del Decoder scelto in base al proprio sistema: sara' l'indirizzo del relè N. 1

int baseaddress = 70;

""" tempo di chiusura dei contatti del relè in millisecondi quando viene ricevuto un comando; 0 vuol dire sempre chiuso fino a nuovo comando

""" io, ad esempio ho scelto 3000 ovvero il binario rimane in tensione per 3 secondi

int mstot = 3000;

""" primo pin di Arduino cui viene collegato il pin N. 1 della scheda relè

int firstpin = 3;

""" Nella sezione seguente viene ripetuto N volte il ciclo di inizializzazione

// Decoder Init
//
void ConfigureDecoder()
{
for(int i=0; i<(int)(sizeof(gAddresses)/sizeof(gAddresses[0])); i++)
{

gAddresses[0].address = baseaddress + i;
gAddresses[0].output = 0;
gAddresses[0].outputPin = firstpin + i;
//gAddresses[0].isDigital = false;
//gAddresses[0].isFlasher = false;
//gAddresses[0].analogValue = 250;
gAddresses[0].durationMilli = mstot;
}

// Setup output pins
for(int i=0; i<(int)(sizeof(gAddresses)/sizeof(gAddresses[0])); i++)
{
if( gAddresses.outputPin )
{
pinMode( gAddresses.outputPin, OUTPUT );
}
gAddresses.onMilli = 0;
```

```
gAddresses.offMilli = 0;
}
}
```

""" Nella sezione seguente viene catturato il flusso delle informazioni DCC e trasformato in un formato comprensibile da noi e da Arduino

```
//
// Basic accessory packet handler
//
void BasicAccDecoderPacket_Handler(int address, boolean activate, byte data)
{
// Convert NMRA packet address format to human address
address -= 1;
address "= 4;
address += 1;
address += (data & 0x06) >> 1;

//*****************************************************************
//con il Multimaus occorre aggiungere 4 all'indirizzo trovato per far coincidere
//l'indirizzo digitato con quello calcolato: verificare nel proprio sistema

address +=4;

//*****************************************************************

boolean enable = (data & 0x01) ? 1 : 0;

for(int i=0; i<(int)(sizeof(gAddresses)/sizeof(gAddresses[0])); i++)
{
if( address == gAddresses.address )
{
Serial.print("Basic addr: ");
Serial.print(address,DEC);
Serial.print(" activate: ");
Serial.println(enable,DEC);

if( enable )
{
gAddresses.output = 1;
gAddresses.onMilli = millis();
gAddresses.offMilli = 0;
}else{
gAddresses.output = 0;
gAddresses.onMilli = 0;
gAddresses[i].offMilli = millis();
}
}
}

}
```

```
""" Il programma vero e proprio che fa scattare il relè all'indirizzo 70 + x con x compreso fra 0 e 7


//
// Setup
//
void setup()
{
Serial.begin(9600);
DCC.SetBasicAccessoryDecoderPacketHandler(BasicAccDecoderPacket_Handler, true);
ConfigureDecoder();
DCC.SetupDecoder( 0x00, 0x00, kDCC_INTERRUPT );
}


//
// Main loop
//
void loop()
{
static int addr = 0;

////////////////////////////////////////////////////////////////////
// Loop DCC library
DCC.loop();

////////////////////////////////////////////////////////////////////
// Bump to next address to test
if( ++addr >= (int)(sizeof(gAddresses)/sizeof(gAddresses[0])) )
{
addr = 0;
}

////////////////////////////////////////////////////////////////////
// Turn off output?
if( gAddresses[addr].offMilli && gAddresses[addr].offMilli<millis() )
{
// Clear off time
gAddresses[addr].offMilli = 0;

// Disable output
digitalWrite( gAddresses[addr].outputPin, LOW);

return;
}

////////////////////////////////////////////////////////////////////
// Turn on output?
if( gAddresses[addr].onMilli && gAddresses[addr].onMilli<=millis() )
{
// Clear off time
gAddresses[addr].onMilli = 0;
```

```
// Enable output

digitalWrite( gAddresses[addr].outputPin, HIGH);

// If still enabled and a flash type, set off time
if( gAddresses[addr].durationMilli )
{
gAddresses[addr].offMilli = millis() + gAddresses[addr].durationMilli;
}

return;
}

}

/
```

*Le varie righe che contengono il comando **Serial.Print** e **Serial.Println***
servono per monitorare il funzionamento attraverso il monitor seriale del SW di controllo: possono essere commentate con la doppia barra //.

utilizzandocon le porte non utilizzate si potrebbero comandare 8 semafori in contemporanea con gli 8 relè

—