Copyright , disclaimer, help

In case of questions, please feel free to contact us. If we can find the time we'll try to get back to you.
- Ruud Boer - ruudboer@gmail.com - for the Arduino sketches
- Erik Holewijn - doerak_balder@hotmail.com - for the PC sound software

Required hardware

To feed the DCC signal to the Arduino, a fast opto-coupler circuit is required, to transfer 14 - 20 V DCC to 5V Arduino. Information on such circuit can be found on Rudys Model Railway blog: https://rudysmodelrailway.wordpress.com/software/

Arduino DCC Function / Servo / Sound decoder with Delay sketch

With the software included in this zip archive an Arduino can be turned into a combined DCC Function / Servo / Sound decoder. Based upon reception of a DCC accessory address the Arduino can:

- Control digital outputs (max 16)
- Control servo motors (max 12)
- Control the playback of sounds on a PC that is connected via USB

Start / stop of any function can be delayed for a specified time (on-delay, off-delay).

DCC Function decoder

When a DCC address is enabled, one or two output pins of the Arduino will react.
Each function can operate in one of three different modes:

1. Continuous. Output is HIGH as long as the DCC address is enabled.
2. One shot. Output goes HIGH when the DCC address is enabled. The output goes LOW after a specified time. If the DCC address is disabled before this time has elapsed, the output goes LOW immediately.
3. Flasher. Output alternates between HIGH and LOW, as long as the DCC address is enabled. If the address is disabled, the output goes LOW immediately. The HIGH and LOW times can be specified independently. A flasher has the option to configure a second output pin, which will have its HIGH and LOW states exactly opposite. This enables easy creation of alternating flashing lights.

DCC servo decoder

A function can have a servo output 'attached' to it. This makes it possible to mechanically drive accessories or to switch junctions. When the function is enabled by the DCC address, the servo that is attached to this function will rotate to its 'enabled' angle. When the DCC address is disabled again, the servo rotates to its 'disabled' angle.

The servo angles that correspond to the DCC enabled / disabled states can be configured individually. The servo rotation speed is also configurable, be it not individually, the speed setting is for all servos.

DCC Sound decoder

Besides having a servo attached, a function can also trigger the playback of a sound file (wav or mp3) on a PC. If the 'playsound' parameter is set, the function will send the corresponding DCC address via

USB to the PC. On the PC a program called DCCSound.exe is running. This program has a config file that lists what sound file to play upon receipt of what DCC address.

When the DCC address is disabled again, a second command is sent to the PC to stop the playback of the sound.

In addition to this DCC trigger, playback of sounds can also be triggered by time (hh:mm), or by (random) repeat, whereby a sound is played every so-many seconds.

Delay

With every function, an 'on-delay' and an 'off-delay' time can be specified. This delays the reaction of the function relative to the receipt of its DCC address. This can come in handy with e.g. the creation of a railway crossing with lights, sound and a barrier, where the barrier should go down only some time after the lights and the sound went on.

Installation

1. Build the needed opto-coupler circuit to connect your DCC to your Arduino. See: https://rudysmodelrailway.wordpress.com/software
2. Install the development environment for Arduino on your PC: http://arduino.cc/en/Main/Software.
3. In the menu 'file - preferences' you can specify the location of your sketches (which is the Arduino word for programs)
4. From the Arduino/libraries folder, copy the DCC_Decoder folder with its contents to your sketches /libraries folder
5. From the Arduino folder, copy the DCC_Decoder_Function_Servo_Sound_Delay folder with its contents to your sketches folder
6. Install the DCCSound program on your PC via the setup program. More info, see below.

Configuration of Functions

In order for the Arduino to know what DCC addresses to react to, and what to do if that address is enabled, accessory functions must be configured in the Arduino code.

The parameters that can be configured per function are:

```
accessory[2].address = 1; // The DCC address to react to
accessory[2].mode = 3; // Flasher: HIGH for ontime ms, LOW for offtime ms, repeats till DCC off
accessory[2].outputPin = 13; // Arduino Pin where the hardware is connected to
accessory[2].outputPin2 = 12; // Every Flasher has 2 outputs with opposing states
accessory[2].invert = 0; // 0=output HIGH if DCC enabled, 1=output LOW if DCC enabled
accessory[2].playsound = 1; // 1=plays a sound via DCCSound.exe, which sound is configured there
accessory[2].ontime = 300; // Time in ms that outputPin is HIGH and outputPin2 is LOW
accessory[2].offtime = 700; // Time in ms that outputPin is LOW and outputPin2 is HIGH
accessory[2].ondelay = 2000; // Delay in ms before the function switches on after DCC enabled
accessory[2].offdelay = 4000; // Delay in ms before the function switches off after DCC disabled
```

If a parameter is omitted its value will be 0.

To define your functions, open the sketch in the Arduino development environment (or any other code / text editor you prefer) and modify the code. When finished, upload the sketch to your Arduino.

First find 'Fill in the number of accessories and servos you want to control' and edit:

```
const byte maxservos = 2; // The number of servos you want to control with this Arduino
const byte maxaccessories = 4; // The number of functions you want to control with this Arduino
byte servotimer = 40; // Servo angle update timer: 1 degree per ## ms. Lower value -> higher speed
```

Then find 'void ConfigureDecoderFunctions()' and add and configure as many functions and servos as you need.

If needed, also add and configure servos at 'void ConfigureDecoderServos()'.

Example 1:
DCC address 17 should switch a section of street lights on/off, the hardware of which is connected to pin 12.

```
accessory[0].address = 17;
accessory[0].mode = 1; // Continuous
accessory[0].outputPin = 12;
```

Example 2:
DCC address 18 should make pin 11 HIGH for 2 seconds (it has electronics connected that simulates lightning), and should play the file Thunder1.mp3

```
accessory[1].address = 18;
accessory[1].mode = 2; // One shot
accessory[1].outputPin = 11;
accessory[1].ontime = 2000;
accessory[1].playsound = 1;
```

Which file to play is configured in the DCCSound software on the PC. The config file should contain a line like: 18,Thunder1.mp3,0,0,0

Example 3:
DCC address 1 should control a servo that is connected to pin 10

```
accessory[2].address = 1;
accessory[2].mode = 1; // Continuous
```

Notice that no output pin is configured. Since non-configured parameters default to 0, this function will use pin 0 as its output.

We need to connect a servo to accessory[2]. This is done by configuring a servo item in the code at 'void ConfigureDecoderServos()':

```
servos[0].accessorynumber=2; // CONNECTION BETWEEN FUNCTION (accessory[n]) AND SERVO
servos[0].servo.attach(10); // Arduino pin number where servo is connected to
servos[0].angle=70; // Angle of servo. Make this the same as offangle to avoid startup jitter
servos[0].offangle=70; // Setpoint angle when DCC disabled
servos[0].onangle=110; // Setpoint angle when DCC enabled
```

Example 4:
DCC address 2 should activate a railroad crossing with alternating flash lights, play the sound ding-ding.mp3 and operate a servo driven barrier. The lights are connected to pins 3 and 4, the servo to pin 5. When the DCC address is enabled, the lights start to flash and the sound plays, but the barrier should start to close only 4 seconds later. When the DCC address is disabled, first the barrier should move up and only 4 seconds later the lights and the sound may stop.

Let's first configure the alternating flash lights function, which also plays the sound:

```
accessory[3].address = 2;
accessory[3].mode = 3; // Flasher
accessory[3].outputPin = 3;
```

```
accessory[3].outputPin2 = 4;
accessory[3].ontime = 500;
accessory[3].offtime = 500;
accessory[3].playsound = 1;
accessory[3].offdelay = 4000; // 4 seconds delay before the function switches off
```

The sound that needs to be played is configured in the PC software via: 2,ding-ding.mp3,0,0,0.

We need to attach a servo. However ... it needs different on / off delay times. So, we'll use a second function. It is just a dummy, it does not have a visual output, it is only here to attach the servo to.

```
accessory[4].address = 2;
accessory[4].mode = 1; // Continuous
accessory[4].ondelay = 4000; // 4 seconds delay before the function switches on.
```

Notice that this function is coupled to the same DCC address '2'. That is perfectly OK, we can trigger as many functions as we like on one and the same DCC address.
Also notice that no output pin is configured. This function will therefore use pin 0 as its output.

We can now connect a servo to accessory[4]:

```
servos[1].accessorynumber=4; // CONNECTION BETWEEN FUNCTION (accessory[n]) AND SERVO
servos[1].servo.attach(5); // Arduino pin number where servo is connected to
servos[1].angle=50; // Angle of servo. Make this the same as offangle to avoid startup jitter
servos[1].offangle=50; // Setpoint angle when DCC disabled
servos[1].onangle=140; // Setpoint angle when DCC enabled
```

Windows PC DCCSound software

Also included in the zip archive is the PC software that plays the sounds. The program works with any version of Windows from XP onwards.

With the DCCSound software it is possible to play sounds via several triggers:

- Reception of a DCC address via a serial port (where the Arduino is attached to via USB)
- A specified time (hh:mm)
- A specified repeat interval
- A random repeat interval between min-seconds and max-seconds

The program can run without a serial port allocated. In that case no DCC triggers take place, but still the timed and the interval sounds will play, making your model railway feel more 'alive', with birds chirping, dogs barking, cows booing, people chattering, station messages broadcasting, etcetera.

The program can play multiple sounds at the same time. File formats supported: .wav and .mp3.

Installation

Run the file 'setup.exe' and provide the necessary information during the setup process. That should do it.

Startup

The program can now be started via Windows 'Start' button. Find 'DCC Decoder With Sound'. You can of course create a shortcut on your desktop if you like. (Manually, the setup did not do this).

The program first reads the file DCCSounds.txt, which contains the information which sounds to play when. If a correct configuration file is found, the program asks to select the serial port where the Arduino is attached to.

If the configuration file contained errors, you're first asked to correct the file, after which it automatically opens in notepad.

Configuration

Via the menu 'File - Change sounds folder' you can select the folder where your sound files reside. Also the configuration file (DCCSounds.txt) resides in that folder. If no such file is found, an empty one will automatically be created there.

The DCCSound.txt file can be modified to play the sounds you want, triggered by the event you want. Open the file in notepad via menu 'File - Edit DCCSounds.txt', or alternatively you can use any text editor you like and open the file from there.

A line in the DCCSounds.txt file looks like any of these two (see the examples below):
1. DCC-address , file-to-play , max-duration , 0 , 0 (DCC trigger)
2. Thh:mm , file-to-play , max-duration, min-repeat-seconds , max-repeat-seconds (time trigger)

When finished editing, the file can be reloaded into the program via menu 'File - Import DCCSound.txt (after edit)', or via closing and starting the program again. The User Interface shows the table of configured sounds and triggers. **To check if the right sound are allocated to the right triggers you can double click a sound to play it.**

There are three ways in which a DCC enabled sound can stop. Which ever comes first:
1. The end of the sound file is reached
2. The max-duration is reached (see example 2)
3. The DCC address is disabled again

Example 1:
Upon reception of DCC address 17, play the sound ding-ding.mp3
17,ding-ding.mp3,0,0,0

Example 2:
Upon reception of DCC address 18, play the first 12 seconds of the sound chatter.mp3
18,chatter.mp3,12,0,0

Example 3:
At time is 13:00 play the sound bell_01.wav
T13:00,bell_01.wav,0,0,0

Example 4:
Repeat the sound train-delay-message.mp3 every 300 seconds
T00:00,train-delay-message.mp3,0,300,300

Example 5:
Random repeat the sound dog-bark.wav between 20 and 60 seconds
T00:00,dog-bark.wav,0,20,60

Where to find sounds?

These are a few sites that have a collection of sounds for free:
http://www.soundbible.com
http://www.freesound.org
http://eng.universal-soundbank.com
http://geluiden.koploperforum.nl
http://www.geluidsfragmenten.nl
http://www.geluidvannederland.nl
https://soundcloud.com

Arduino DCCSound sketch

If you do not need the DCC Function or Servo Decoder functionality, but you would like to play sounds based on DCC addresses, there is an alternative Arduino Sketch named DCC_Decoder_Sound included in the zip archive. To use this sketch, copy the DCC_Decoder_Sound folder with its contents from the zip Arduino folder to your sketches folder

This sketch does not need any configuration. It simply monitors DCC activity and sends all received DCC accessory addresses over the serial interface. Based on the DCCSound.txt configuration file, the PC software will play the associated sound file if it recognizes a DCC address. DCC addresses that it does not recognize are simply ignored.

Have fun,

Ruud Boer
Erik Holewijn