# EV3 Arduino Extensions

## Getting Ready

Use Tools/Block Import to load up MHP2196b.ev3b. This provides access to the I2C blocks. This only needs to be done one and persists across projects.

## Create a new project.

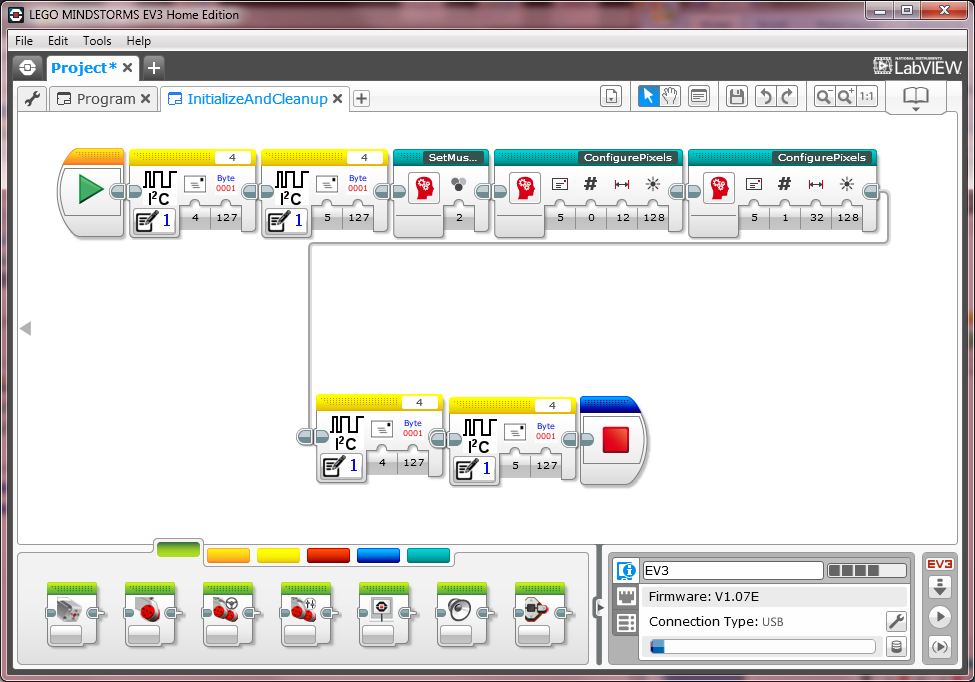
Under project properties, Programs tab, press the Import button. Select EvVArduinoExtFunctions.ev3s. This needs to be done for each new project created.

Start and end you project with two I2C blocks that write a 127 to each Arduino (make sure the port is set to ‘4’). Command ‘127’ is the ‘restore default state’ command and ensure that the Arduinos are in a known state when your program starts. If your program has an end, then it is a good idea to send them again before your program exits. If it never exits (unless you turn it off), then that is fine.

If you are going to use the sound capabilities, include the SetMusicMode block, passing ‘1’ for MIDI synthesizer mode and ‘2’ for CODEC mode (to play .mp3 or .mid files from the SD card).

If you are going to use the NeoPixels, include the ConfigureNeoPixels block for each of the NeoPixel strips that you want to use. The ‘strip’ parameter is either ‘0’ or ‘1’ to identify the strip you want to configure, the ‘length’ parameter is the number of LEDs in the strip, and the last parameter is a ‘brightness’ setting that is applied to every SetPixel\* command. Starting at ‘128’ puts it in the middle and will give you the ability to raise or lower it as needed later.

In this example, it is using CODEC mode and two strips of NeoPixels, one with 12 lights and one with 32 lights.



## Playing Music Files

The hardest part about playing music files is getting the files on the SD card.

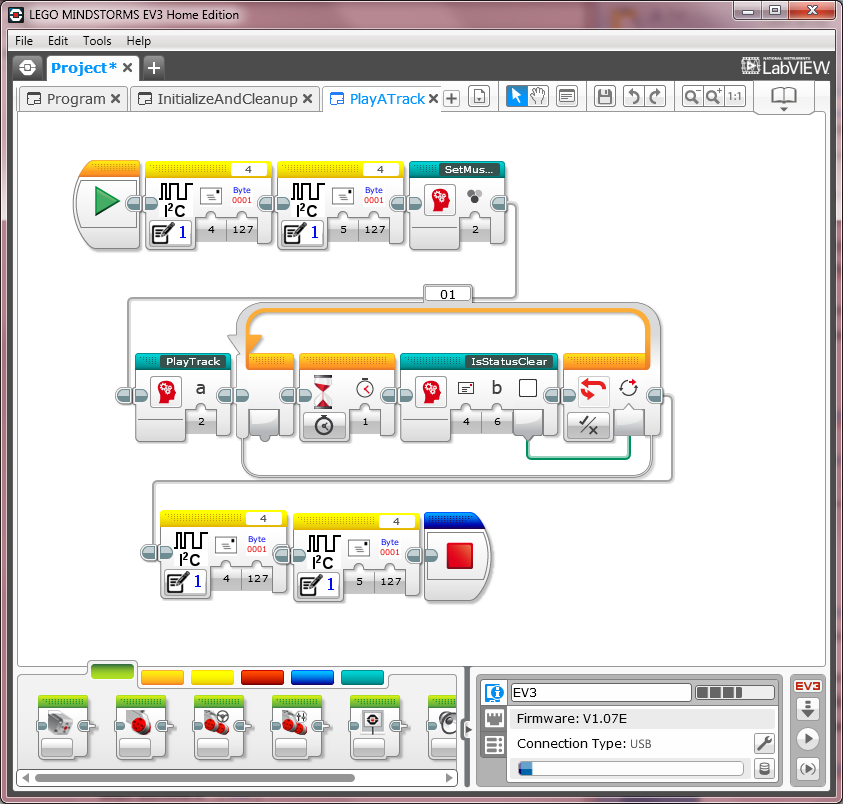
Probably the easiest to use are MP3 files. For this, simply rename the file to trackNNN.mp3 and place it on the SD card. The NNN is a decimal number padded with leading zeros. For example, track 16 would have a filename of track016.mp3.

Once the desired track is on the SD card, you can play it with the PlayTrack block. The only parameter is the desired track number. For a short track, say a sound effect, it is probably fine to just play it and forget it. If you are playing a longer track and want to know when it is finished, you can use the IsStatusClear block with ‘bit’ set to 6 to see if it is finished.

The Arduino will also play MIDI Type 0 files. Type 0 are files that contain only one track within the file. It may take some special steps to get a program like Garage Band to generate a MIDI Type 0 file. I found some MIDI files on the internet, and then had to run a conversion program on them to merge all of the tracks into a single track to make a Type 0 file. The good part about using MIDI files is that they are much smaller and take less of the Arduino’s time to play (leaving it free to do other things). But right now we don’t have much else for it to do, so it probably isn’t worth the trouble. A MIDI file should have a .mid extension on the SD card.

If the SD card has a trackNNN.mp3 and a trackNNN.mid and you tell it to play track N, it will play the .mp3. That is only because I had to pick one to look for first, and I choose .mp3.

Here is a simple program that plays track 2 and waits for it to finish. Your program will likely have more interesting stuff within the loop, like monitoring sensors or controlling motors.



## SetMusicMode

### Parameters

* mode
  + 0 – turns off music.
  + 1 – selects MIDI synthesizer mode in which you can play individual notes.
  + 2 – selects CODEC mode in which you can play entire tracks from the SD card.

### Notes

This must precede any calls to play notes or tracks.

You can change the music mode as often as you like, but it was sort of expected that any project would pick one and use it throughout. There is sometimes noise generated on the audio outputs during the transition.

## PlayTrack

### Parameters

* track – which track to play. The desired track must be stored in the root directory of the SD card and be named ‘trackNNN.xxx’. NNN is the three digit track number (with leading zeros, if necesasry). xxx is the extension, either .mp3 or .mid (for MIDI files). MIDI files must be type 1 in order to be recognized.

### Notes

If you want to know whether playback has completed, read status back from the Arduino (just do a 1-byte read and that is what you will get). Bit 6 being 1 tells you it is currently playing a track, and being 0 tells you it is not (i.e. it finished).

Starting playback of a track while one is already in progress will stop the prior playback and start the new one.

## StopTrack

### Parameters

none

### Notes

This stops the playback of any track in progress, if any.

## ConfigurePixels

### Parameters

* address – the I2C address of the Arduino controlling the NeoPixels.
* strip – the NeoPixel strip to operate on, 0 or 1.
* length – the number of LEDs in the strip.
* brightness – the maximum brightness of the LEDs. All Set\* commands are scaled by this value.

### Notes

This must precede any call to the SetPixel\* functions.

The length of the pixel strip is limited by the amount of power available and the amount of available RAM. A strip requires 3-bytes per LED. The Arduino currently has about 1K of RAM available, so it should be able to handle about 300 LEDs split between the two strips.

## ShowPixels

### Parameters

* address – the I2C address of the Arduino controlling the NeoPixels.
* strip – the NeoPixel strip to operate on, 0 or 1.

### Notes

The colors of each pixel is stored in memory on the Arduino. The ‘show’ actually takes the colors and sends them to the NeoPixel strip. The SetPixel\* commands do not automatically do a show in order to allow the user to perform multiple SetPixel\* commands in order to get the strip into the desired colors before sending the colors out to the lights. The RotateAllPixels and WalkAllPixels commands always include a show.

## SetPixelRangeFade

### Parameters

* address – the I2C address of the Arduino controlling the NeoPixels.
* strip – the NeoPixel strip to operate on, 0 or 1.
* start – the start index at which to start filling pixels.
* startR, startG, startB – the color of the pixel at ‘start’.
* end – the end index (inclusive) at which to stop filling pixels.
* endR, endG, endB – the color of the pixel at ‘end’.

### Notes

Each color, red, green, and blue, are linearly interpolated for each pixel between the start and end.

For example, setting pixel 0 to (255,0,0) through pixel N-1 to (0,255,0) will generate a strip of colors fading from full red at one end to full blue at the other end.

## SetPixelBarGraph

### Parameters

* address – the I2C address of the Arduino controlling the NeoPixels.
* strip – the NeoPixel strip to operate on, 0 or 1.
* length – the length of the bar graph to create, in pixels.
* r, g, b – the color of the pixels.

### Notes

Sets the pixel strip with a bar of the specified length and color (starting at pixel 0). The rest of the strip is cleared.

## RotateAllPixels

### Parameters

* address – the I2C address of the Arduino controlling the NeoPixels.
* strip – the NeoPixel strip to operate on, 0 or 1.
* count – the number of steps to rotate the pixels.

### Notes

Positive ‘count’ shifts the pixels toward zero, and negative ‘count’ shifts the pixels away from zero. It is a ‘rotate’ in the sense that pixels that fall off the end roll back onto the other end to fill in the spaces vacated by the shift.

A show is performed upon completion.

## WalkAllPixels

### Parameters

* address – the I2C address of the Arduino controlling the NeoPixels.
* strip – the NeoPixel strip to operate on, 0 or 1.
* count – the number of steps to rotate the pixels.
* interval – the time interval between rotations of the pixels, in msec.

### Notes

See notes for RotateAllPixels.

This command sets up the Arduino to do a rotate of all the pixels by ‘count’ steps every ‘interval’ msec.

If count or interval is zero, then the command stops any walk currently in progress.

## SendCmdAndWait

### Parameters

* address – the I2C address of the Arduino.
* cmd – the command to send.
* data1 – the parameter to the command.

### Notes

This is a low level function that allows you to send commands directly to the Arduino. If available, it is better to use the functions provided specifically to do what you are trying to do.

## IsStatusClear

### Parameters

* address – the Arduino address to for which to check the status.
* bit – the bit number of interest.

Output:

* result – the output is logical true if the bit is clear, and logical false if the bit is set.

### Notes

The status bits are as follows:

* 0 – busy, set if the Arduino currently processing a command or has commands waiting in its queue.
* 1 – synth mode, set if MIDI synthesizer mode is active.
* 2 – CODEC mode, set if CODEC mode is active (whether or not it is playing).
* 3 – not used.
* 4 – MMB detected, set if the AdaFruit Music Maker Breakout board (or shield) was detected. More specifically, it is set if it detected a VS1053 on the SPI bus at the specified chip select.
* 5 – SD detected, set if an SD card was detected in the card slot.
* 6 – playing, set if the Arduino is currently playing a track in CODEC mode.
* 7 – true, always set to indicate this byte is a status readback.