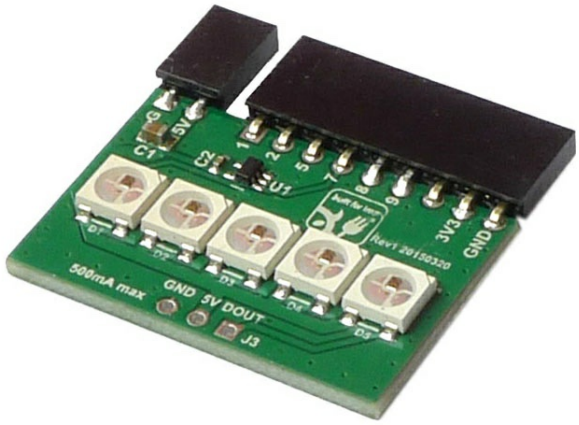
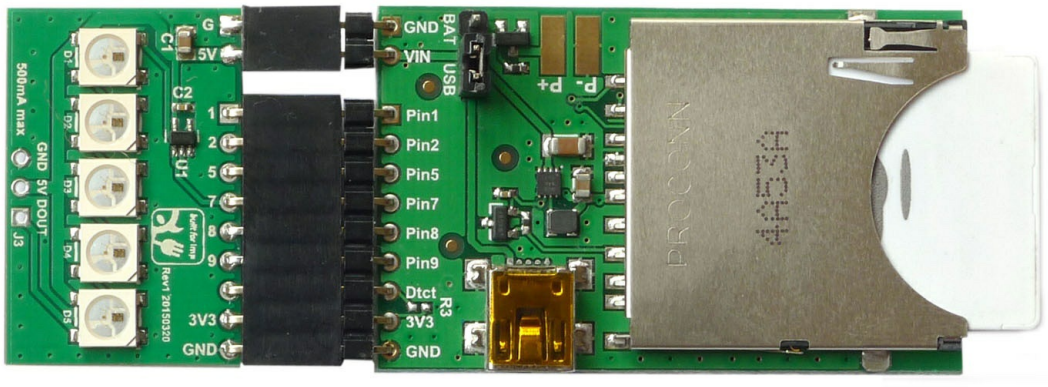
Electric Imp Tails Project: Micro Light Show

Here's a simple project to allow you try out the RGB LED Tail for the first time: a fun micro light show with four different lighting effects which you can select using a web browser.



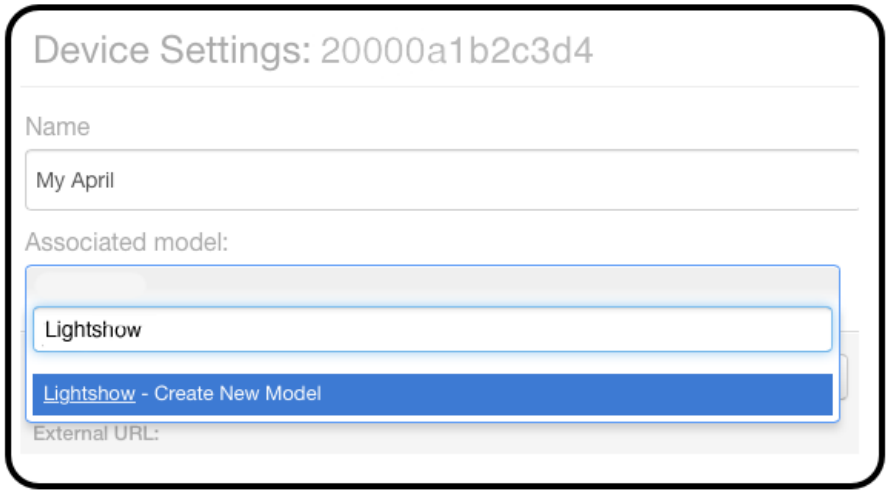
Step 1: Assemble the Hardware

If you haven't done so already, clip the RGB LED Tail onto your April dev board. Slip in the imp001 card too, and connect the mini USB cable to a power supply and then to the April.



Step 2: Program the Project

Open the Electric Imp IDE in a web browser. You'll see your device listed on the left-hand side under 'Unassigned Devices'. Click on the gearwheel icon to the right of this to display the 'Device Settings' window. Here you can give the device a more friendly name, such as 'My April'. Click on the pop-up menu under 'Associated Model:' and in the empty space that appears, type in 'Lightshow' (without the single quotes). When you've done, click on 'Save Changes'.



Your device should now disappear from 'Unassigned Devices' and reappear under 'Lightshow' in the 'Active Models' section. If you can't see your device, just click on the disclosure triangle to the left of 'Lightshow' to reveal it. Can't see 'Lightshow'? Click on the disclosure triangle to the left of 'Active Models'.

Click on 'My April' and you'll see 'Agent', 'Device' and 'Device Logs' panels appear in the space on the right-side of the screen. This is where you enter your programs: one for the device, another for its online agent. Both blocks of code together comprise a model – Electric Imp terminology for an Internet of Things app.

The code you need is listed below; copy and paste it into the IDE's agent and device code panels. Make sure you paste it correctly. The agent code's first line should read:

```
    // Agent Code
```

Agent Code

```
// Agent Code

function requestHandler(request, response) {
  try {
    if ("glow" in request.query) {
      device.send("seteffect", 0);
      response.send(200, "Glow effect on");
      return;
    }

    if ("random" in request.query) {
      device.send("seteffect", 1);
      response.send(200, "Random effect on");
      return;
    }

    if ("looper" in request.query) {
      device.send("seteffect", 2);
      response.send(200, "Looper effect on");
      return;
    }

    if ("larson" in request.query) {
      device.send("seteffect", 3);
      response.send(200, "Larson effect on");
      return;
    }

    if ("setcolor" in request.query) {
      device.send("setcolor", request.query.setcolor);
      response.send(200, "Color set");
      return;
    }

    response.send(200, "Waiting for a command");

  } catch (error) {
    server.log("Error: " + error);
  }
}

// Reqister the handler to deal with incoming requests

http.onrequest(requestHandler);
```

lightshow.agent.nut hosted with ❤ by **GitHub** **view raw**

Device Code

```
// Device Code
#require "WS2812.class.nut:2.0.0"

// CONSTANTS
const NUMPIXELS = 5;
const DELAY = 0.1;
const COLORDELTA = 8;

// Instantiate the WS2812s
spi <- hardware.spi257;
spi.configure(MSB_FIRST, 7500);
pixels <- WS2812(spi, NUMPIXELS);

// Light/color data
local redVal = 0;
local greenVal = 0;
local blueVal = 0;

local redDelta = 1;
local greenDelta = 1;
```

```
local greenDelta = 1;
local blueDelta = 1;

local redOn = true;
local greenOn = false;
local blueOn = false;

local timer = null;
local pixel = 0;
local pDelta = 1;

function glowinit(dummy) {
  // All the pixels run through the range colors
  if (timer != null) imp.cancelwakeup(timer);
  redVal = 0; greenVal = 0; blueVal = 0;
  redDelta = COLORDELTA; greenDelta = COLORDELTA; blueDelta = COLORDELTA;
  redOn = true; greenOn = false; blueOn = false;

  // Call the glow effect
  glow();
}

function glow() {
  // Set the color values of the RGB LEDS
  pixels.fill([redVal, greenVal, blueVal]);

  // Write the color data to the WS2812s
  pixels.draw();

  // Adjust the color values for the next frame of the animation
  adjustColors();

  // Queue up the presentation of the next frame
  timer = imp.wakeup(DELAY, glow);
}

function randominit(dummy) {
  // A random pixel glows a random color
  if (timer != null) imp.cancelwakeup(timer);
  random();
}

function random() {
  // Clear the current color data and write it to the
  // WS2812s to turn them all off
  pixels.fill([0,0,0]);
  pixels.draw();

  // Set random color values
  redVal = ran(255); greenVal = ran(255); blueVal = ran(255);

  // Pick one of the WS2812s
  pixel = ran(NUMPIXELS);

  // Write the color data out
  pixels.set(pixel, [redVal, greenVal, blueVal]);
  pixels.draw();

  // Queue up the presentation of the next frame
  timer = imp.wakeup(DELAY * 2, random);
}

function looperinit(dummy) {
  // The pixels run through all the colors.
  // Only one pixel is illuminated at once, in order
  if (timer != null) imp.cancelwakeup(timer);
  redVal = 0; greenVal = 0; blueVal = 0;
  redDelta = COLORDELTA; greenDelta = COLORDELTA; blueDelta = COLORDELTA;
  redOn = true; greenOn = false; blueOn = false;
```

```
  redOn = true; greenOn = false; blueOn = false;
  pixel = 0;
  looper();
}

function looper() {
  // Clear all the WS2812s' colors then write the current
  // color value to the current LED and write it to the hardware
  pixels.fill([0,0,0]);
  pixels.set(pixel, [redVal, greenVal, blueVal]);
  pixels.draw();

  // Move on to the next LED, looping round to the first if necessary
  pixel++;
  if (pixel >= NUMPIXELS) pixel = 0;

  // Adjust the color value
  adjustColors();

  // Queue up the presentation of the next frame
  timer = imp.wakeup(DELAY, looper);
}

function larsoninit(dummy) {
  if (timer != null) imp.cancelwakeup(timer);
  redVal = 64; greenVal = 0; blueVal = 0;
  redDelta = COLORDELTA; redOn = true; pixel = 0; pDelta = 1;
  larson();
}

function larson() {
  // Clear all the WS2812s' color values to turn them off
  pixels.fill([0,0,0]);
  pixels.set(pixel, [redVal, 0, 0]);
  pixels.draw();

  // Get the address of the next LED to color,
  // bouncing back from the ends and the center
  pixel = pixel + pDelta;
  if (pixel == NUMPIXELS) {
    pDelta = -1;
    pixel = NUMPIXELS - 2;
  }

  if (pixel < 0) {
    pDelta = 1;
    pixel = 1;
  }

  // Adjust the color value
  redVal = redVal + redDelta;
  if (redVal > 160) {
    redVal = 160 - COLORDELTA;
    redDelta = COLORDELTA * -1;
  } else if (redVal < 64) {
    redVal = 64 + COLORDELTA;
    redDelta = COLORDELTA;
  }

  // Queue up the presentation of the next frame
  timer = imp.wakeup(DELAY, larson);
}

function ran(max) {
  // Generate a pseudorandom number between 0 and (max - 1)
  local roll = 1.0 * math.rand() / RAND_MAX;
  roll = roll * max;
  return roll.tointeger();
}
```

```
}

function adjustColors() {
  // Calculate new color values, running from red to green to blue,
  // and fading from one into the next
  if (redOn) {
    redVal = redVal + redDelta;

    if (redVal > 254) {
      redVal = 256 - COLORDELTA;
      redDelta = COLORDELTA * -1;
      greenOn = true;
    }

    if (redVal < 1) {
      redDelta = COLORDELTA;
      redOn = false;
      redVal = 0;
    }
  }

  if (greenOn) {
    greenVal = greenVal + greenDelta;

    if (greenVal > 254) {
      greenDelta = COLORDELTA * -1;
      blueOn = true;
      greenVal = 256 - COLORDELTA;
    }

    if (greenVal < 1) {
      greenDelta = COLORDELTA;
      greenOn = false;
      greenVal = 0;
    }
  }

  if (blueOn) {
    blueVal = blueVal + blueDelta;

    if (blueVal > 254) {
      blueDelta = COLORDELTA * -1;
      redOn = true;
      blueVal = 256 - COLORDELTA;
    }

    if (blueVal < 1) {
      blueDelta = COLORDELTA;
      blueOn = false;
      blueVal = 0;
    }
  }
}

function setColor(color) {
  if (timer!= null) imp.cancelwakeup(timer);
  pixels.fill([0,0,0]);

  local colors = split(color, ".");
  local red = colors[0].tointeger();
  if (red < 0) red = 0;
  if (red > 255) red = 255;

  local green = colors[1].tointeger();
  if (green < 0) green = 0;
  if (green > 255) green = 255;

  local blue = colors[2].tointeger();
  if (blue < 0) blue = 0;
```

```
  if (blue > 255) blue = 255;

  for (local i = 0 ; i < NUMPIXELS ; i++) {
    pixels.writePixel(i, [red, green, blue]);
  }

  pixels.draw();
}

function setEffect(effect) {
  switch (effect) {
    case 0:
      glowinit(true);
      break;

    case 1:
      randominit(true);
      break;

    case 2:
      looperinit(true);
      break;

    case 3:
      larsoninit(true);
  }
}

// START OF PROGRAM

// Register handlers for messages from the agent
agent.on("seteffect", setEffect);
agent.on("setcolor", setColor);

// Pick a random effect to begin with
setEffect(ran(4));
```

lightshow.device.nut hosted with ❤ by **GitHub** **view raw**

Step 3: Run the Code

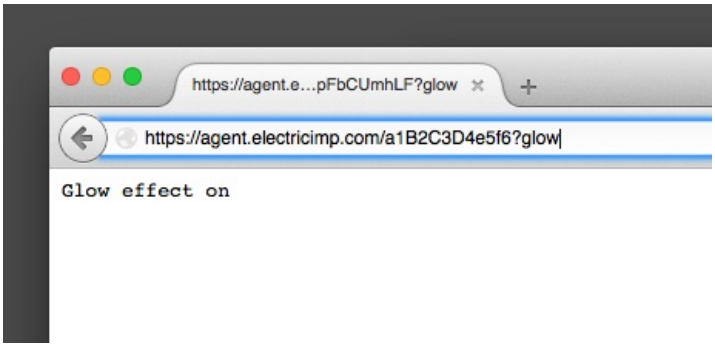
The code you pasted into the IDE is ready to run, so click on the 'Build and Run' button. Because the code initially selects one of the four pre-programmed lighting effects at random, you'll see one of these make the RGB LED Tail strut its funky stuff. To changed the effect remotely, you need to send a command to the device from your computer's web browser. To do so, look just above the Agent Code in the IDE – you'll see a line that looks a little like this:

```
https://agent.electricimp.com/a1B2C3D4e5f6
```

It will be slightly different in your case because the code at the end is unique to each device. Click on this URL and a new browser window or tab will open. Click on the URL field and move the cursor to the end of the line, making sure you don't delete the web address that's already there. Add the following text to the end of the URL:

```
?glow
```

The address should now look like this (remember your code is different):



Press the Enter, Return or Done key on your keyboard and your micro light show should now start displaying one of its pre-set patterns. There

are three others you can try – which you might want to skip straight to if the Tail is already showing the glow pattern:

```
https://agent.electricimp.com/a1B2C3D4e5f6?random
https://agent.electricimp.com/a1B2C3D4e5f6?looper
https://agent.electricimp.com/a1B2C3D4e5f6?larson
```

Step 4. What Next?

There are a number of ways you can improve on the Micro Light Show's basic design:

- Look for the extra command in the code above
  - The code you pasted in above has a fifth code you can enter alongside the address of your RGB LED Tail. It sets all the lights in the Tail to the color you specify in the form `red.green.blue`. Each value should be between zero and 255 and governs the brightness of that color.
  - You add the color data to the command you've found by adding text like this to the end of the Tail's web address: `=255.0.0`.
  - Experiment with values to see how mixing different brightnesses of red, green and blue generates other shades.
- Program some more lighting effects
  - Each effect has an initializer function which sets up the effect and calls the function which performs the effect.
  - Try changing colors and flashing the WS2812s.
  - Use the *clearFrame()* method to turn off all the RGB LEDs before you light them again.
  - Remember there are five RGB LEDs, numbered 0 through 4, and the color variables `redVal`, `greenVal` and `blueVal` take values from 0 to 255. Go beyond these limits and you'll get an error message in the log.
  - Don't forget to add a command to the agent code to allow you to trigger the effect remotely.
- Add more RGB LEDs
  - You can add extra WS2812s by wiring them up to the Tail's expansion port.
- Try some of the other Tails projects
  - Visit the RGB LED Tail page for more applications you can explore.