D.Frederick

10/30/2016 11:10:00 AM

df20160910\_Notes\_for\_LEDStrip\_Programming\_v04.docx

**Programming an LED (Light Emitting Diode) strip with a Raspberry Pi and JAVA**

**Purpose:**

This document explains how to build a JAVA project to control a programable LED strip with a Raspberry Pi using JAVA.

**Prerequisites:**

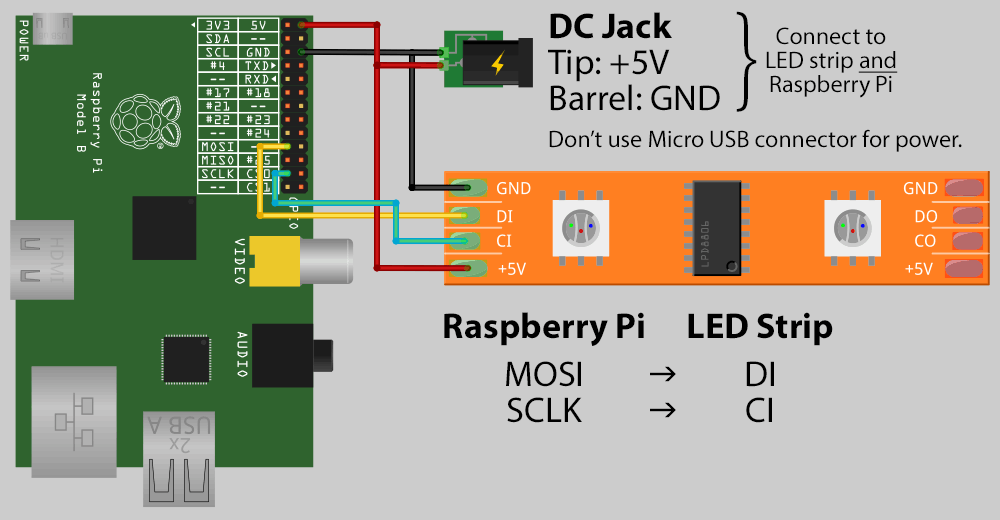
* AdaFruit LED Strip based on LPD8806 (See: <https://www.adafruit.com/products/306>)
* Raspberry Pi (Power Suppy, SD Card, …)
* Eclipse and JAVA on your workstation
* JAVA I/O Library for the Raspberry Pi (http://pi4j.com)
* LEDStrip source code (at the end of this document)
* PUTTY – SSH (Secure Shell) Client on your laptop (See: http://www.putty.org/)

**Background:**

The AdaFruit LED strip consists a chain of programmable tri-color LEDs. The color of each tri-color LED is define by writing three integer values to each LED. The three integer values are desired brightness of the red, green, blue LEDs within the tri-color LED. The values range from 0 to 255. By selecting a blend of numbers, any color can be created. Motion or color shifting is created by updating the colors written to the LEDS over time. The diagram below shows the JAVA variable array LEDdataArray [color][index] and how this maps to the physical LED strip when the LedStrip.update() method is called.



The LED Strip only has 4 wires: + 5 Volts, Ground, Data (MOSI), and Clock (SCLK). See the diagram below for the wiring.



The Raspberry Pi writes the values to the LED Strip over a serial interface called SPI (Serial Peripheral Interface).

The “LEDStrip project” is a JAVA project use to program the LED Strip. This project consists of two main classes. The LEDstrip class represents the physical LED strip. A variety of methods within the class control the LEDs. The second class, LEDstripDriver, contains the main class and allows the Programmer to write code to control the LEDstrip by calling methods.

The code is complied into a JAVA jar file on the laptop and transferred to the Raspberry Pi using SSH. Eclipse uses the "build.xml" file to guide this process. The location of the eclipse project folders and IP address of the Raspberry Pi is configured within this file. This file will need to be update to match your IP address. Review and update the configuration around line 80. You will need to update the path to the PUTTY files, Raspberry Pi IP address, and pi account credentials.

The "JAVA I/O Library for the Raspberry Pi" at <http://pi4j.com> makes accessing the real, physical inputs and outputs possible within JAVA code. The library really consists of a single file (pi4j-core.jar) that must be present in eclipse (lib folder) and on the raspberry pi (/home/[user]/dist/lib folder).

Install the library on the Raspberry pi. Copy (scp) the file (pi4j-core.jar) from the Raspberry Pi to your workstation. Once the LedStrip project is built, create a folder called lib and place the file (pi4j-core.jar) in this folder.

Initial Setup Process Overview:

1. Initialize the Raspberry Pi and configured networking (<https://www.raspberrypi.org/>)
2. Install JAVA then Eclipse.
3. Create the LED Project in Eclipse.
4. Install "JAVA I/O Library for the Raspberry Pi".
5. Copy the pi4j-core.jar file to Eclipse.
6. Compile and download the code.
7. Update code as desired.

Initial Setup Process details:

1. Initialize the Raspberry Pi and configured networking (<https://www.raspberrypi.org/>)

This process can take some time. Follow the instructions on many of the sources on the internet. Recommend setting a static IP address on the Raspberry Pi to allow Eclipse to download code.

1. Install JAVA then Eclipse.

Download and install JAVA on the workstation.

Download the Eclipse from (<http://www.eclipse.org/downloads/>). JAVA must already be installed on the workstation to support Eclipse.

1. Create the LED Project in Eclipse.
2. In Eclipse, select **File** => **New** => **JAVA Project.**
3. Set the Project Name to "LedStrip". (Note: Capitalization counts un linux and JAVA)
4. Set the project location (e.g., "C:\\_Robotics\Raspberry\_Pi\\_Eclipse\_workspace\LedStrip").
5. Select **Finish**.
6. In Eclipse, expand out the LedStrip project and select the "src" folder.
7. Right click and select **New** => **Class.**
8. Set the name of the Class to "LEDstripDriver" and select **Finish**.
9. Copy the contents of the LEDstripDriver.java file listed below and paste it into the new class LEDstripDriver, replacing the complete content.

Note: Ignore the errors at this time. They will resolved once all the source code and libraries are installed

1. In Eclipse, select the "src" folder.
2. Right click and select **New** => **Class**.
3. Set the name of the Class to "LedStrip" and select **Finish**.
4. Copy the contents of the LedStrip.java file listed below and paste it into the new class Ledstrip, replacing the complete content.
5. In the Windows File Explorer, browse to the project folder and create a file called build.xml and place the contents of the build.xml file from the end of this document into the file.
6. In the Windows File Explorer, browse to the project folder and create a folder called "lib" below the main folder ("….\LedStrip\lib "). This is used later.
7. Install "JAVA I/O Library for the Raspberry Pi".

Follow the installation guidance at (<http://pi4j.com/install.html#EasyPreferred>)

Example command to transfer the Pi4J package to the Raspberry Pi using PUTTY's scp copy command. Update this command to match your paths.

C:\\_Robotics\Raspberry\_Pi\PUTTY\pscp -pw pas1word C:\\_Robotics\Raspberry\_Pi\Pi4J\pi4j-1.1.deb pi@10.18.95.100:/tmp/pi4j-1.1.deb

1. Copy the pi4j-core.jar file to Eclipse.

Example command to transfer the Pi4J library Jar file from the Pi using PUTTY's scp copy command. Update this command to match your paths.

C:\\_Robotics\Raspberry\_Pi\PUTTY\pscp -pw pas1word pi@10.18.95.100:/opt/pi4j/lib/pi4j-core.jar C:\\_Robotics\Raspberry\_Pi\\_Eclipse\_workspace\LedStrip\lib

1. Add the Pi4J library to the build path
2. Right click on the project and select **Properties**.
3. Select "Java Build Path".
4. Select the libraries tab.
5. Select "Add External JARs and browse to the file (pi4j-core.jar) in the project lib folder.
6. Compile and download the code.

Sometimes the imports get messed up so resolve this using the following sequence within Eclipse. On the Eclipse menu, select **Source** => **Organize Imports**.

1. Update code as desired to get cool lighting . The Class LEDstripDriver has the code to control the lights. There are a number of different examples. Some are commented out.

=================================

Process to update and download the code:

1. Connect the Raspberry Pi to the LED Strip per the diagram above
2. Power on the Raspberry Pi.
3. Wait about 30 seconds for the Raspberry Pi to completely boot
4. Start PUTTY and connect to the Raspberry Pi.
5. Logon to the Raspberry Pi with the name “pi” and password
6. Change directory to /home/pi/dist using the command : cd /home/pi/dist
7. List your current working directory using the following command: pwd
8. List the files within the directory using the command: ls -ltr
9. Start Eclipse and expand the project LedStrip.
10. Expand the project tree structure source code (src) to see the source code classes.
11. Double click on source code “LEDstripDriver”.
12. Update the code as desired.

The LEDstripDriver class provides a number of examples of how to program the LED Strip.

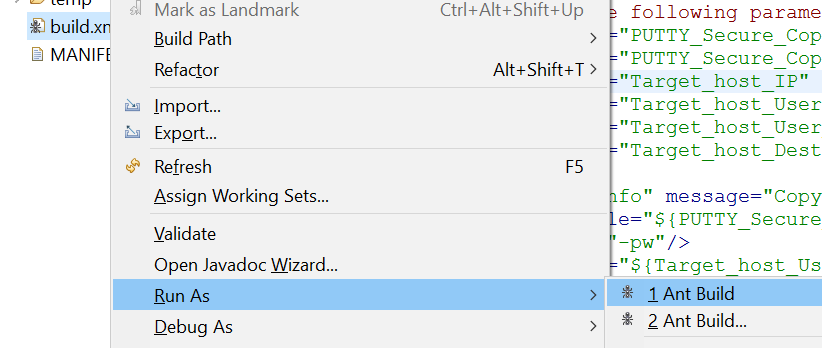
Starting around line 200, there are 4 sections:

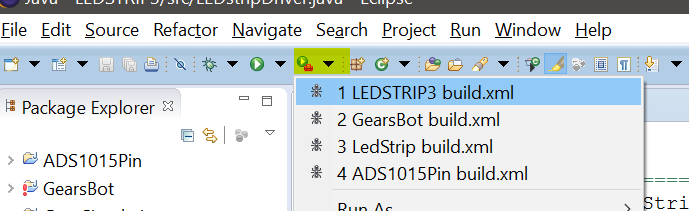
1. Read values from Text file and write to LED string
2. Example of teststrip method
3. Example of bouncing LED
4. Example of Color shifting

Review the example and make some simple modifications until you get an understanding of how the code operates.

1. Enable monitoring of the compile and deployment by opening the “Console” windows if not already open. Select **Window** => **Show View** => **Console**.
2. Compile and deploy the java code by selecting the file “build.xml”. Right click and select **Run As** then   
   **1 Ant Build**.   
     
   After the first time, subsequent compile/deployments can be implemented by selecting the tool bar button with the green arrow and toolbox followed by the option LEDStrip Build.xml

See the figures below.





1. Observe the compiling and deployment within the console window. A good deployment with have the display: BUILD SUCCESSFUL Total time: x seconds
2. If all goes well, the code will be deployed to the Raspberry Pi.
3. Run the JAVA application on the Raspberry Pi by running the following command within the PUTTY windows: sudo java -jar LEDstripDriver.jar
4. Enter control-C to stop the code.
5. When done, shutdownn the Raspberry Pi using the command: sudo shutdown -h 0

Files:

LEDstripDriver.java

LedStrip.java

build.xml

===========================================================================================================

===(LEDstripDriver.java)===================================================================================

/\*

\* Purpose: This code drives the ADAFruit LED Strip.

\*

\* Version: 1.1

\* Author: D.Frederick

\* Date: August 8, 2015

\*

\* Description: The LED strip is driven by an array of data. If the LED Strip is 32 LEDs long,

\* then an array of 32 elements should be created. Each element within the array

\* consists of 3 bytes; one byte for Red, one byte for Green, and one byte for Blue.

\* Each byte has a value range of 0 to 255 where 255 is the brightest.

\*

\*/

import com.pi4j.wiringpi.Spi;

import java.io.BufferedReader;

import java.io.\*;

public class LEDstripDriver {

// =========================================================================================================

/\*

Methods:

LEDstripDriver()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Constructor

GetLEDdataArrayValue()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Get single color in single element (location, color, color value) where color 1=red, 2=green, 3=blue)

paintComponent\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ LED Simulator

SetLEDdataArrayValue(int, byte, byte)\_\_ Set single color in single element (location, color, color value) where color 1=red, 2=green, 3=blue)

\*/

// =========================================================================================================

// =========================================================================================================

// Initialize a set of global variables

int debugLevel = 5;

byte Red = 0;

byte Green = 0;

byte Blue = 0;

static int numberLEDs = 32; // Number of LEDs

byte[][] LEDdataArray; // Two dimensional Array to move data from LEDStrip object to Jpanel [NumberLEDs][Color]

static final LedStrip ledStrip = new LedStrip(numberLEDs, 0.5F);

// Constructor to instantiate the two dimensional array

public LEDstripDriver ()

{

LEDdataArray = new byte[numberLEDs][3]; // Instantiate the two dimensional Array

// First dimension is the array of LEDs

// Second dimension is the color where 0=red, 1=green, 2=blue

}

// Method to place data into globally accessible two dimensional array

public void SetLEDdataArrayValue (int index, byte color, byte value)

{

//LEDdataArray[index][color] = (byte) (0x7F & value); // strip the leading bit [ & is bit-wise AND operator ]

LEDdataArray[index][color] = value;

}

// Method to place data into globally accessible two dimensional array

public byte GetLEDdataArrayValue (int index, byte color)

{

return LEDdataArray[index][color];

}

/\*

\*

\* --------------------------------------------------------------------------------------------------------

\* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | ...

\* | R | R | R | R | R | R | R | R |

\* | G | G | G | G | G | G | G | G |

\* | B | B | B | B | B | B | B | B |

\* --------------------------------------------------------------------------------------------------------

\*

\*/

// =========( Main )==================================================================================================

public static void main(String[] args) throws Exception {

// setup SPI for communication with the led strip.

// DF: See http://pi4j.com/apidocs/com/pi4j/wiringpi/Spi.html

// DF: See http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/

// DF:

int current\_red = 50;

int current\_blue = 50;

int current\_green = 50;

int step\_size = 20;

int time\_delay\_value = 100;

// Initialize the hardware

int fd = Spi.wiringPiSPISetup(0, 10000000);

if (fd <= -1)

{

System.out.println("[Main] SPI initialization FAILED.");

return;

}

// System.out.println("[Main] SPI initialization SUCCEEDED.");

//======================================================================================================

//==( Manipulate the LED colors here )=====================================================================

//Read values from Text file and write to LED string

/\*

\* This section of code will cause the application to read text from a file and use the text read to drive the LEDs

\* The text must consist of 4 values per line where the values are:

\* [RED color brightness] [GREEN color brightness] [BLUE color brightness] [Duration millsec]

\* 152 250 148 100

\* 153 250 147 100

\* 155 250 145 100

\* 157 250 143 100

\* 159 250 141 100

\*/

//BufferedReader reader = new BufferedReader(new FileReader("input.txt"));

//String line = null;

//String[] parts;

//while ((line = reader.readLine()) != null)

// { // See http://docs.oracle.com/javase/tutorial/java/data/converting.html

// parts = line.split("\\s"); // Splits on whitespace

// current\_red = (Integer.valueOf(parts[0])).intValue();

// current\_green = (Integer.valueOf(parts[1])).intValue();

// current\_blue = (Integer.valueOf(parts[2])).intValue();

// time\_delay\_value = (Integer.valueOf(parts[3])).intValue();

// System.out.println("File RGB: " + current\_red + " " + current\_green + " " + current\_blue + " " + time\_delay\_value);

// ledStrip.fill(current\_red, current\_green, current\_blue);

// ledStrip.update();

// Thread.sleep(time\_delay\_value);

// }

//=========================================================================================================

// ==( Manipulate the LED colors here in a continuous loop)================================================

while (1==1) // Continuous loop

{

//==(Example of teststrip method )===========================================================================================

// ledStrip.testStrip();

// ledStrip.testStrip();

// ledStrip.testStrip();

// ledStrip.testStrip();

//==(End of Example of teststrip method )=====================================================================================

//===========================================================================================================================

//===========================================================================================================================

////==(Example of bouncing LED)================================================================================================

//

// for (int loopCounter1=1; loopCounter1 <= numberLEDs; loopCounter1++)

// {

// ledStrip.fill(0, 100, 255, loopCounter1, loopCounter1);// Red, Green, Blue, Start Position, Finish Position)

// ledStrip.update(); // Write the LED Array out to the physical LEDS

//

// try {Thread.sleep(10);}catch(InterruptedException ex){Thread.currentThread().interrupt();}//delay milliseconds.

// ledStrip.fill(0, 0, 0, loopCounter1, loopCounter1 ); // Red, Green, Blue, Start Position, Finish Position)

// ledStrip.update();

//

//

// } // End of loopCounter loop

//

//

// for (int loopCounter=numberLEDs; loopCounter >= 1; loopCounter--)

// {

// ledStrip.fill(0, 100, 255, loopCounter, loopCounter);// Red, Green, Blue, Start Position, Finish Position)

// ledStrip.update(); // Write the LED Array out to the physical LEDS

//

// try {Thread.sleep(10);}catch(InterruptedException ex){Thread.currentThread().interrupt();}//delay milliseconds.

// ledStrip.fill(0, 0, 0, loopCounter, loopCounter ); // Red, Green, Blue, Start Position, Finish Position)

// ledStrip.update();

//

//

// } // End of loopCounter loop

//

//==(End of Example of bouncing LED)================================================================================================

//===========================================================================================================================

//===========================================================================================================================

//==(Example of Color shifting)==============================================================================================

int tempRed = 0;

int tempBlue = 0;

int tempGreen = 0;

for (int loopCounter=0; loopCounter < 255; loopCounter++)

{

tempRed = loopCounter;

tempGreen = 0;

tempBlue = 0;

if(tempRed <= 1) tempRed = 1;

if(tempGreen <= 1) tempGreen = 1;

if(tempBlue <= 1) tempBlue = 1;

if(tempRed > 255) tempRed = 255;

if(tempGreen > 255) tempGreen = 255;

if(tempBlue > 255) tempBlue = 255;

// System.out.format ("Values RGB %x %x %x %n",tempRed, tempGreen, tempBlue);

ledStrip.fill(tempRed, tempBlue, tempGreen, 1, (numberLEDs) ); // Red, Green, Blue, Start Position, Finish Position)

ledStrip.fill(tempRed, tempBlue, tempGreen, 1, 31 ); // Red, Green, Blue, Start Position, Finish Position)

ledStrip.update(); // Write the LED Array out to the physical LEDS

try {Thread.sleep(1);}catch(InterruptedException ex){Thread.currentThread().interrupt();}//delay milliseconds.

} // End of loopCounter loop

for (int loopCounter=255; loopCounter > 0; loopCounter--)

{

tempRed = 0;

tempGreen = loopCounter;

tempBlue = 0;

if(tempRed <= 1) tempRed = 1;

if(tempGreen <= 1) tempGreen = 1;

if(tempBlue <= 1) tempBlue = 1;

if(tempRed > 255) tempRed = 255;

if(tempGreen > 255) tempGreen = 255;

if(tempBlue > 255) tempBlue = 255;

// System.out.format ("Values RGB %x %x %x %n",tempRed, tempGreen, tempBlue);

ledStrip.fill(tempRed, tempBlue, tempGreen, 1, (numberLEDs) ); // Red, Green, Blue, Start Position, Finish Position)

ledStrip.update(); // Write the LED Array out to the physical LEDS

try {Thread.sleep(1);}catch(InterruptedException ex){Thread.currentThread().interrupt();}//delay milliseconds.

/// System.out.println("Green" + " " + tempRed + " " + tempBlue + " " + tempGreen);

} // End of loopCounter loop

ledStrip.fill(0, 0, 0, 1, (numberLEDs) ); // Red, Green, Blue, Start Position, Finish Position)

ledStrip.update(); // Write the LED Array out to the physical LEDS

for (int loopCounter=0; loopCounter < 255; loopCounter++)

{

tempRed = 0;

tempGreen = 0;

tempBlue = loopCounter;

if(tempRed <= 1) tempRed = 1;

if(tempGreen <= 1) tempGreen = 1;

if(tempBlue <= 1) tempBlue = 1;

if(tempRed > 255) tempRed = 255;

if(tempGreen > 255) tempGreen = 255;

if(tempBlue > 255) tempBlue = 255;

// System.out.format ("Values RGB %x %x %x %n",tempRed, tempGreen, tempBlue);

ledStrip.fill(tempRed, tempBlue, tempGreen, 1, (numberLEDs) ); // Red, Green, Blue, Start Position, Finish Position)

ledStrip.update(); // Write the LED Array out to the physical LEDS

try {Thread.sleep(1);}catch(InterruptedException ex){Thread.currentThread().interrupt();}//delay milliseconds.

/// System.out.println("Blue" + " " + tempRed + " " + tempBlue + " " + tempGreen);

} // End of loopCounter loop

ledStrip.setLedOff(numberLEDs); // Red, Green, Blue, Start Position, Finish Position)

ledStrip.update(); // Write the LED Array out to the physical LEDS

try {Thread.sleep(3000);}catch(InterruptedException ex){Thread.currentThread().interrupt();}//delay milliseconds.

//==(End of Example of Color shifting)=======================================================================================

//===========================================================================================================================

//===========================================================================================================================

//===========================================================================================================================

} // End while loop

} // main

// =========( End of Main )==================================================================================================

} // Class LEDStrip

===========================================================================================================

===(LedStrip.java)=========================================================================================

import com.pi4j.wiringpi.Spi;

/\*\*

\* Representation of a LPD8806 based led strip.

\*

\* @author Gert Leenders

\*/

//=========================================================================================================

/\*

Methods: (Also see the JAVADOCs within the ./docs folder

LedStrip(int, float) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Constructor : pass-in the number of LEDs and the brightness

allOff() \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Set all leds off.

fill (int, int, int) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Fill all of the LEDs with the same color (R, G, B)

fill (int, int, int, float) \_\_\_\_\_\_\_\_\_\_\_ Fill all leds with a specified color and set the overall brightness.(R, G, B, brightness)

fill (int, int, int, int, int) \_\_\_\_\_\_\_\_ Fill a part of the led strip with a specified color. (R,G,B, first LED, last LED)

fill (int, int, int, int, int, float) \_ Fill a part of the led strip with a specified color and brightness. (R,G,B, first LED, last LED, brightness)

SetLed (int, int, int, int) \_\_\_\_\_\_\_\_\_\_\_ Set a specific LED (R, G, B, location)

SetLed (int, int, int, int, float) \_\_\_\_ Set a specific LED and brightness (R, G, B, location, brightness)

SetLedOff (int) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Set a specific LED off (location)

setSuspendUpdates(boolean\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Set Flag

testStrip() \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Set LEDs to Green, Blue, Red for 2 seconds each

update() \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Push the Array into the LED

GetElement() \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Get the RGB values from an array element

GetArray() \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Get the RGB values from an array

GetBlueLEDvalue (int) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Get specific array element in blue

GetGreenLEDvalue (int) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Get specific array element in green

GetRedLEDvalue (int) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Get specific array element in red

\*/

//=========================================================================================================

//=========================================================================================================

public final class LedStrip {

private static final int GAMMA\_LENGTH = 256;

private static final byte[] GAMMA = new byte[GAMMA\_LENGTH];

private int numberOfLeds;

public RGBLed[] ledBuffer;

public RGBLed element;

private float brightness;

private boolean suspendUpdates = false;

// This section creates an array used to adjust the brightness. Input 0-255 Output 128-255 exponential curve

// Not sure why this is here....

static {

for (int i = 0; i < GAMMA\_LENGTH; i++) {

int j = (int) (Math.pow(((float) i) / 255.0, 2.5) \* 127.0 + 0.5);

GAMMA[i] = (byte) (0x80 | j); // bit level Or Operator

// System.out.format("GAMMA %d %x %x %n", i, j, GAMMA[i]);

}

}

/\*\*

\* Initialize a led strip.

\*

\* @param numberOfLeds the number of leds on the strip

\* @param brightness the overall brightness of the leds

\* @throws IllegalArgumentException

\*/

public LedStrip(final int numberOfLeds, final float brightness) throws IllegalArgumentException {

if (brightness < 0 || brightness > 1.0) {

throw new IllegalArgumentException("Brightness must be between 0.0 and 1.0");

}

this.numberOfLeds = numberOfLeds;

this.ledBuffer = new RGBLed[numberOfLeds];

for (int i = 0; i < numberOfLeds; i++) {

ledBuffer[i] = new RGBLed();

}

this.brightness = brightness;

}

/\*\*

\* @param suspendUpdates if true, the trip will ignore updates

\*/

public void setSuspendUpdates(boolean suspendUpdates) {

this.suspendUpdates = suspendUpdates;

}

public RGBLed[] GetArray()

{

return ledBuffer;

}

public RGBLed GetElement(int index)

{

element = ledBuffer[index];

return element;

}

/\*\*

\* Set all leds off.

\*/

public void allOff() {

fill(0, 0, 0);

update();

}

/\*\*

\* Fill all leds with a specified color.

\*

\* @param red value between 0 and 255 for the red led

\* @param green value between 0 and 255 for the green led

\* @param blue value between 0 and 255 for the blue led

\*/

public void fill(final int red, final int green, final int blue) {

fill(red, green, blue, 1, numberOfLeds);

}

/\*\*

\* Fill all leds with a specified color and set the overall brightness.

\*

\* @param red value between 0 and 255 for the red led

\* @param green value between 0 and 255 for the green led

\* @param blue value between 0 and 255 for the blue led

\* @param brightness value between 0 and 1 for the brightness

\*/

public void fill(final int red, final int green, final int blue, final float brightness) {

fill(red, green, blue, 1, numberOfLeds, brightness);

}

/\*\*

\* Fill a part of the led strip with a specified color.

\*

\* @param red value between 0 and 255 for the red led

\* @param green value between 0 and 255 for the green led

\* @param blue value between 0 and 255 for the blue led

\* @param start the start led position in the led strip

\* @param end the end led position in the led strip

\* @throws IllegalArgumentException

\*/

public void fill(final int red, final int green, final int blue, final int start, final int end) throws IllegalArgumentException {

fill(red, green, blue, start, end, brightness);

// System.out.println("Fill 5 " + red + " " + green + " " + blue);

}

/\*\*

\* Fill a part of the led strip with a specified color and set the brightness.

\*

\* @param red value between 0 and 255 for the red led

\* @param green value between 0 and 255 for the green led

\* @param blue value between 0 and 255 for the blue led

\* @param start the start led position in the led strip

\* @param end the end led position in the led strip

\* @param brightness value between 0 and 1 for the brightness

\* @throws IllegalArgumentException

\*/

public void fill(final int red, final int green, final int blue, final int start, final int end,

final float brightness) throws IllegalArgumentException {

if (red < 0 || green < 0 || blue < 0 || red > 255 || green > 255 || blue > 255) {

throw new IllegalArgumentException("Red, green and blue values must be between 0 and 255.");

}

if (start < 1 || end > (numberOfLeds + 1)) {

throw new IllegalArgumentException("Led start must be greater then 0, end must be smaller then " + (numberOfLeds + 1) + ".");

}

if (end < start) {

throw new IllegalArgumentException("End must be greater then or equal as start.");

}

for (int i = start; i <= end; i++) {

setLed(i, red, green, blue, brightness);

}

}

/\*\*

\* Set the color of an individual led.

\*

\* @param number the number of the led in the led strip

\* @param red value between 0 and 255 for the red led

\* @param green value between 0 and 255 for the green led

\* @param blue value between 0 and 255 for the blue led

\*/

public void setLed(final int number, final int red, final int green, final int blue) {

setLed(number, red, green, blue, brightness);

}

/\*\*

\* Switch a led off.

\*

\* @param number the number of the led in the led strip

\*/

public void setLedOff(final int number) {

setLed(number, 0, 0, 0, 0);

}

/\*\*

\* Set the color and brightness of an individual led.

\*

\* @param number the number of the led in the led strip

\* @param red value between 0 and 255 for the red led

\* @param green value between 0 and 255 for the green led

\* @param blue value between 0 and 255 for the blue led

\* @param brightness value between 0 and 1 for the brightness

\*/

public void setLed(final int number, final int red, final int green, final int blue, final float brightness) {

if (number < 1 || number > numberOfLeds) {

throw new IllegalArgumentException("led number must be greater then 0 and smaller then " + (numberOfLeds + 1) + ".");

}

ledBuffer[number - 1].set(red, green, blue, brightness);

}

/\*\*

\* Update the strip in order to show its new settings.

\*/

public void update() {

if (suspendUpdates) {

return;

}

final byte packet[] = new byte[numberOfLeds \* 3];

for (int i = 0; i < numberOfLeds; i++) {

packet[i \* 3] = ledBuffer[i].getGreen();

packet[(i \* 3) + 1] = ledBuffer[i].getRed();

packet[(i \* 3) + 2] = ledBuffer[i].getBlue();

}

// Update the strand

Spi.wiringPiSPIDataRW(0, packet, this.numberOfLeds \* 3);

byte endPacket[] = {(byte) 0x00};

// Flush the update

Spi.wiringPiSPIDataRW(0, endPacket, 1);

}

/\*\*

\* Simple test function to test your led strip.

\*

\* @throws InterruptedException

\*/

public void testStrip() throws InterruptedException {

allOff();

fill(0, 255, 0);

update();

Thread.sleep(2000);

fill(0, 0, 255);

update();

Thread.sleep(2000);

fill(255, 0, 0);

update();

Thread.sleep(2000);

allOff();

}

public byte getRedLEDvalue (int index)

{ return ledBuffer[index].getRed(); }

public byte getGreenLEDvalue (int index)

{ return ledBuffer[index].getGreen(); }

public byte getBlueLEDvalue (int index)

{ return ledBuffer[index].getBlue(); }

/\*\*

\* RGBLed represents a 'single' led on a led strip.

\* In reality these 'single' leds consist out of 3 leds, a red, a green and a blue one.

\*

\* @author Gert Leenders

\*/

// private class RGBLed {

public class RGBLed {

private byte red;

private byte green;

private byte blue;

/\*\*

\* Initiate a single led in a led strip.

\*

\* @param red value between 0 and 255 for the red led

\* @param green value between 0 and 255 for the green led

\* @param blue value between 0 and 255 for the blue led

\* @param brightness overall brightness for the led combination

\*/

public void set(final int red, final int green, final int blue, final float brightness) {

this.red = GAMMA[(int) (red \* brightness)];

this.green = GAMMA[(int) (green \* brightness)];

this.blue = GAMMA[(int) (blue \* brightness)];

}

/\*\*

\* @return the value for the green led (between 0 and 255)

\*/

public byte getGreen() {

return green;

}

/\*\*

\* @return the value for the blue led (between 0 and 255)

\*/

public byte getBlue() {

return blue;

}

/\*\*

\* @return the value for the red led (between 0 and 255)

\*/

public byte getRed() {

return red;

}

} // RGBLed

}

===========================================================================================================

===(build.xml)=============================================================================================

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!-- WARNING: Eclipse auto-generated file.

Any modifications will be overwritten.

To include a user specific buildfile here, simply create one in the same

directory with the processing instruction <?eclipse.ant.import?>

as the first entry and export the buildfile again.

-->

<project basedir="." default="scp-deploy" name="LEDstripDriver">

<property name="src.dir" location="src" />

<property name="build.dir" location="bin" />

<property name="dist.dir" location="dist" />

<property name="docs.dir" location="docs" />

<property name="project-name" value="LEDstripDriver" />

<property environment="env"/>

<property name="debuglevel" value="source,lines,vars"/>

<property name="target" value="1.7"/>

<property name="source" value="1.7"/>

<path id="LEDstripDriver.classpath">

<pathelement location="bin"/>

<pathelement location="lib/pi4j-core.jar"/>

</path>

<target name="init">

<echo>"TEST 1"</echo>

<echo message="Test of variabe ${basedir} "/>

<mkdir dir="bin"/>

<copy includeemptydirs="false" todir="bin">

<fileset dir="src">

<exclude name="\*\*/\*.launch"/>

<exclude name="\*\*/\*.java"/>

</fileset>

</copy>

</target>

<target name="clean">

<delete dir="bin"/>

<echo>"TEST-2"</echo>

</target>

<target depends="clean" name="cleanall"/>

<target depends="build-subprojects,build-project" name="build"/>

<target name="build-subprojects"/>

<target depends="init" name="build-project">

<echo message="${ant.project.name}: ${ant.file}"/>

<javac debug="true" debuglevel="${debuglevel}" destdir="bin" includeantruntime="false" source="${source}" target="${target}">

<src path="src"/>

<classpath refid="LEDstripDriver.classpath"/>

</javac>

<echo>">> Compile Complete"</echo>

</target>

<target description="Build all projects which reference this project. Useful to propagate changes." name="build-refprojects"/>

<target description="copy Eclipse compiler jars to ant lib directory" name="init-eclipse-compiler">

<copy todir="${ant.library.dir}">

<fileset dir="${ECLIPSE\_HOME}/plugins" includes="org.eclipse.jdt.core\_\*.jar"/>

</copy>

<unzip dest="${ant.library.dir}">

<patternset includes="jdtCompilerAdapter.jar"/>

<fileset dir="${ECLIPSE\_HOME}/plugins" includes="org.eclipse.jdt.core\_\*.jar"/>

</unzip>

</target>

<target description="compile project with Eclipse compiler" name="build-eclipse-compiler">

<property name="build.compiler" value="org.eclipse.jdt.core.JDTCompilerAdapter"/>

<antcall target="build"/>

</target>

<target name="jar">

<echo message="Creating JAR file in: ${dist.dir} !!!"/>

<delete file = "MANIFEST.MF"/>

<manifest file = "MANIFEST.MF">

<attribute name = "Main-Class" value = "${project-name}"/>

<attribute name = "Class-Path" value = "lib/pi4j-core.jar"/>

</manifest>

<jar basedir="bin" destfile="${dist.dir}/${project-name}.jar" includes = "\*.class" manifest = "MANIFEST.MF">

</jar>

</target>

<target name="scp-deploy" depends="clean, build,jar">

<!-- Update the following parameters for copy file to target host -->

<property name="PUTTY\_Secure\_Copy\_Exe" value="pscp" />

<property name="PUTTY\_Secure\_Copy\_Exe\_Path" value="C:\\_Robotics\Raspberry\_Pi\PUTTY" />

<property name="Target\_host\_IP" value="192.168.42.1" />

<property name="Target\_host\_User" value="pi" />

<property name="Target\_host\_User\_password" value="pas1word" />

<property name="Target\_host\_Destination\_Folder" value="/home/pi" />

<echo level="info" message="Copy dist directory to Remote Pi 2"/>

<exec executable="${PUTTY\_Secure\_Copy\_Exe}" dir="${PUTTY\_Secure\_Copy\_Exe\_Path}">

<arg line="-pw"/>

<arg value="${Target\_host\_User\_password}"/>

<arg line="-r"/>

<arg value="${dist.dir}"/>

<arg value="${Target\_host\_User}@${Target\_host\_IP}:${Target\_host\_Destination\_Folder}"/>

</exec>

</target>

</project>

===========================================================================================================