To begin with, the FastLed and AdafruitNeoPixel libraries must be installed. These can be found at the following addresses:

<https://github.com/FastLED/FastLED>

<https://github.com/adafruit/Adafruit_NeoPixel>

These will be downloaded as archives and installed in the Arduino IDE via: Sketch --> Include Library --> Add.Zip Library.

Installing FastLED on some variants of Arduino IDE gave errors. One variant that worked well is Arduino IDE 1.6.5.

At the beginning of the program, global variables are declared:

cNt: (counter) variable of type int, necessary for the operation of the Rotary Encoder routine.

n: byte variable type that describes the 24 possible states of the system.

mn: byte variable type, modulo nrLED from n . Here, nrLED=8.

Next, the hardware connections are defined, as in the wiring diagram, and the number of LEDs used in the ring (8).

The FastLED library is included and it declares the number of LEDs (CRGB leds).

Void Setup:

- The corresponding pins are initialized as outputs or as PULLUP inputs.

- FastLED is initialized with NEOPIXEL elements (mandatory and standard procedure).

-The following lines represent a small routine that initially lights a single LED in blue, the others off (initial state LED ring).

Void loop:

-Call the roTen function that describes the operation of the Rotary Encoder and assign the value of this function to the variable n. So n depends on the value of the Rotary Encoder and has values between 0 and 23.

-Calculate mn which is equal to n modulo 8, because there are 8 LEDs.

-Follow the switch (n) ... case statement.

Because n = 0 ... 23, there are 24 cases, case 0..case 23.

These define the 24 LEDring lighting cases:

Case0: LED 0 blue, all others off.

Case1: LED 0, LED1 blue, all others off.

............

Case7: all LEDs blue.

Case8: LED 0 green, all others blue.

Case9: LED0, LED1 green, all others blue.

.................

Case15: all LEDs green .

Case16: LED0 red, all others green.

Case17: LED0, LED1 red, all others green.

................

Case23: all LEDs red.

-Then the selected LEDring status is displayed with the FastLED.show statement.

-At the end of the loop, the function vOl (n) is called, which communicates to M62429 the chosen attenuation, being here also a number of 24 states (n = 0 ... 23).

Rotary Encoder routine roTen()

Returns the integer value cNt (counter) in relation to the number of pulses given by the Rotary Encoder contacts.

As long as contacts A and B of the Rotary Encoder are 1, the program flow is blocked in the while statement at the beginning of the function.

As soon as one of the contacts switches to GND it exits the while loop and tests

contact B with the switch statement.

If it is contact B that is at GND, cNt (HIGH case) is incremented.

If not, it means that A is at GND and decrements cNt (LOW case).

In both cases, the cNt value is limited in the range 0 ... 23 with the constrain function.

vOl() routine attenuation control function (volume) for M62429

It is a Void function (which does not return anything).

The function variable is aTTn of type uint8\_t.

The first part of the function builds the binary words for the M62429 command, and in the second part this word is sent serially to the M62429.

Table1 shows how binary words are constructed according to M62429 datasheet.

Such an organization is necessary because the M62429 has a special way of forming binary words (see datasheet, VOLUME CODE table).

Here, D7 and D8 contain a fine attenuation adjustment, and D2 ... D6 a coarse adjustment.

D0, D1 make a simultaneous or separate adjustment of the two channels. In this program, D0 = D1 = 0, for a simultaneous adjustment of the two channels.

In DATA INPUT FORMAT (see Data Sheet) is given the bit alignment order.

In Table 1 on the first column is the attenuation in -dB (you can choose 24 desired values), in the second column is the value of n, (n = 0 ..... n = 23) and then follow the values D10. ... D0 of the 11 bits required for programming.

On the last column is written the value in Hexa of the number thus obtained. The value appears in 0xVVVV format, 16 bits. Because we only have 11 bits to program, the first 5 will be 0, so the data will be 0x0VVV.

The data in Table 1 are used as elements of the maTn matrix, which has a line and 24 columns.

The second part of the routine is intended for the serial transmission of a selected element from the matrix, which is done according to the data sheet, according to the RELATIONSHIP BETWEEN DATA AND CLOCK section.

Thus, data and clock pulses are applied according to the datasheet, the corresponding bit is read, then binary word is rotated to the right with a bit and the operation is resumed a total of 11 times (for loop).