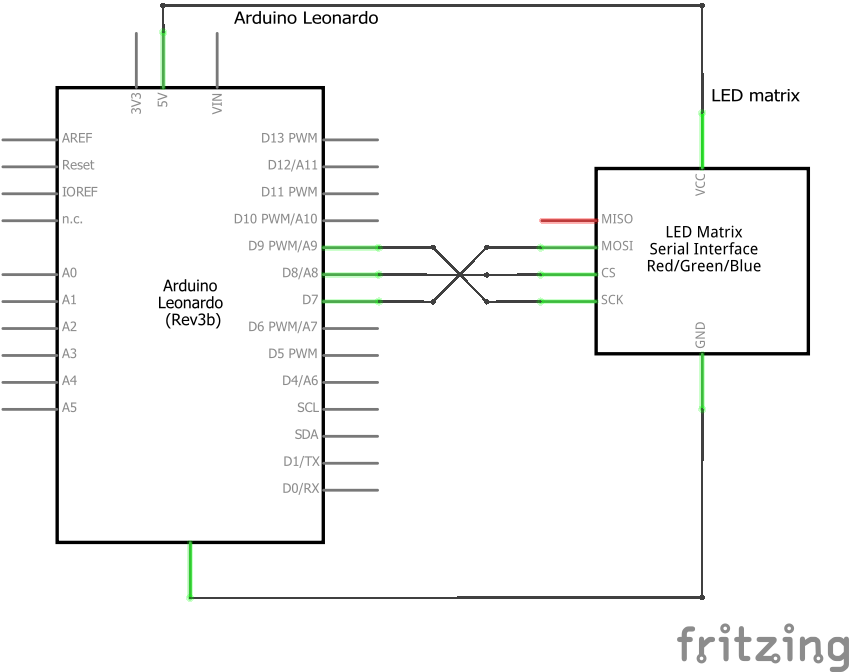
The LED matrix component features 64 red LEDS controlled by the MAX7219 chip, which uses SPI communication protocol.



The LedControl library[[1]](#footnote-1) takes care of the SPI communication, but according to the data sheet these are the registers to be set to send data:

With this line we are setting MOSI, CS and CLK as output pins

DDRB = (1<<PB4) | (1<<PB5) | (1<<PB6)

With this we are enabling SPI, setting the Master and setting the clockrate.

SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0)

This is the procedure to send a byte, named “data”.

We set the CS pin as LOW

PORTB &= ~(1<<PB5)

We write data in the appropriate register

SPDR = data

Then we are waiting for the transmission to end

while(!(SPSR & (1<<SPIF));

Finally, we write HIGH back on the CS pin

PORTB |= (1<<PB5)

The LED Matrix is used to show the new state after a state transition. The state is shown as a sequence of 4 8bit images The sequence is repeated 3 times, then the LED matrix is turned off again to limit power consumption.

The sequences are defined in this array:

byte animations[][4][8] ={

{{0x0, 0x0, 0x0, 0x0, 0x0, 0x18, 0x24, 0x0},{0x0, 0x0, 0x0, 0x3c, 0x42, 0x18, 0x24, 0x0},{0x0, 0x7e, 0x81, 0x3c, 0x42, 0x18, 0x24, 0x0},{0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0}},

{{0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0},{0x0, 0x0, 0x0, 0x0, 0x0, 0x18, 0x24, 0x0},{0x0, 0x0, 0x0, 0x3c, 0x42, 0x18, 0x24, 0x0},{0x0, 0x7e, 0x81, 0x3c, 0x42, 0x18, 0x24, 0x0}},

{{0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0},{0x0, 0x0, 0x0, 0x0, 0x0, 0x18, 0x24, 0x0},{0x0, 0x0, 0x0, 0x3c, 0x42, 0x18, 0x24, 0x0},{0x0, 0x7e, 0x81, 0x3c, 0x42, 0x18, 0x24, 0x0}},

{{0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0},{0x0, 0x0, 0x0, 0x0, 0x0, 0x18, 0x24, 0x0},{0x0, 0x0, 0x0, 0x3c, 0x42, 0x18, 0x24, 0x0},{0x0, 0x7e, 0x81, 0x3c, 0x42, 0x18, 0x24, 0x0}},

{{0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0},{0x0, 0x0, 0x0, 0x0, 0x0, 0x18, 0x24, 0x0},{0x0, 0x0, 0x0, 0x3c, 0x42, 0x18, 0x24, 0x0},{0x0, 0x7e, 0x81, 0x3c, 0x42, 0x18, 0x24, 0x0}}

};

Each element of the array is an array of 4 arrays of 8 byte values. Each byte value specifies in bit notation which LEDs on that row are to be turned on or off.

byte rep = REPS;

void writeByteArray(byte \*src){

for(int i = 0; i < MATRIXLEN; i++) //MATRXILEN{

lc.setRow(0,i,src[i]);

}

}

void startanimation(){

rep = 0;

}

void animate(byte (\*src)[8]){

static byte curr = 0;

writeByteArray(src[curr]);

curr = (++curr)%ANIMLEN;

if(curr==0){

rep++;

}

}

loop(){

. . .

static long sequencedelay = 0;

if(sequencedelay + 500 < millis() && rep < REPS){

animate(animations[IDLSANIM]);

sequencedelay = millis();

}

. . .

}

These functions define how an animation is played. The global variable rep is initialized with the macro REPS, which equals to 3. The loop calls the function animate() with the appropriate sequence to play only if more than half a second has passed and the rep variable is less than 3 (so it means that the sequence has to replayed again)

The function animate() takes in input a sequence (i.e. an element of the global array animations) and uses a static variable to keep track of the current image in a sequence. When called it will call the writeByteArray() function passing the current image to print the image to the LED matrix, increase its curr variable using modulo arithmetic (so the variable will go back to 0 after printing image 3) and if curr is equal to 0 it means one full sequence has been played and rep variable is increased by one. When rep is again equal to 3, the function won’t be called anymore, until a state transition will call startAnimation() which resets the rep variable to 0.

The writeByteArray uses a library function to write each byte of the array in input in the corrispective row of the matrix.

1. http://wayoda.github.io/LedControl/pages/software [↑](#footnote-ref-1)