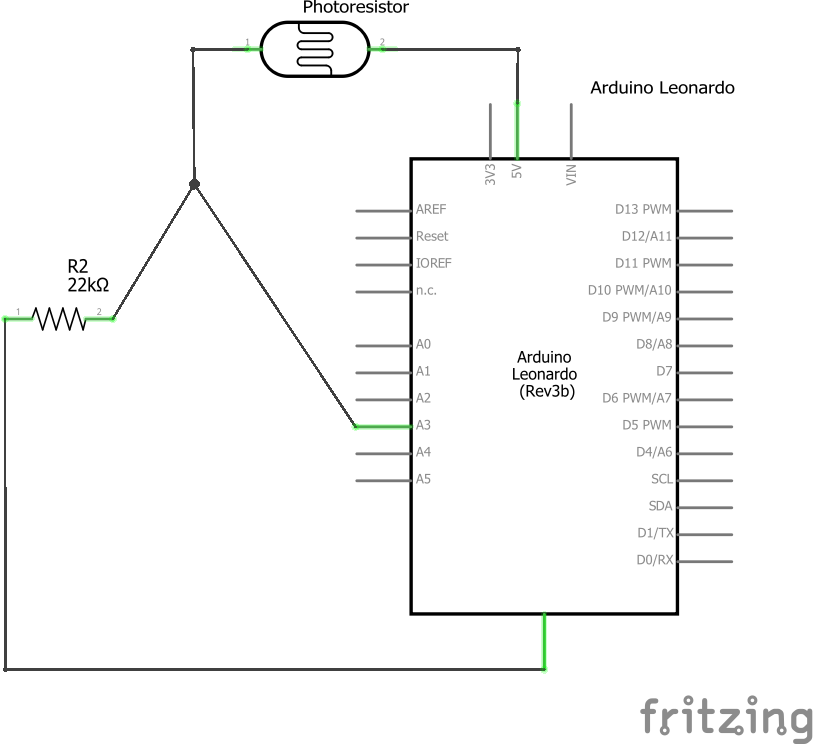
A photoresistor, or light dependent resistor, is a type of light sensitive resistor which can measure the intensity of light received. With low light their resistance is high, as high as 600k at 0.1 lux, and with light exposure their resistance drops down, even as low as 100 at 10000 lux. Photoresistors aren’t particularly accurate, in fact they are sensitive to temperature changes as well, and the relation between light intensity and resistance is nonlinear, meaning that a slight increase in light would result in a much greater decrease in resistance. Photoresistors also experience latency, with as much as a 1s latency to go from a very strong light to total darkness resistance. Nevertheless, these issues are not a concern for the purpose of our project, in which this component is used to adapt the screens brightness and not to make accurate real time measurements.

The sensor is wired up with an additional 22k resistor so that we can measure the difference in voltage for different resistance values in the photoresistor. The Arduino board in fact reads analog input by mapping the range 0 – 5V in input to 10 bit values, and does not actually read resistance differences. The value of the resistor was determined heuristically by comparing the voltage reading at different exposures of light, in order to have the widest range of values for the light settings that could be produced in the laboratory.



The code for the photoresistor functionality is contained in the loop() function:

void loop() {

. . .

static long photodelay = 0;

static long curr = 0;

curr = millis();

if(photodelay + 1000 < curr)

{

short phread = analogRead(PHOTORES);

lc.setIntensity(0,map(phread, 0, 1023, 0, 15)); //0 - 15

display.setBacklight(map(phread, 0, 1023, 10, 100)); // 0 -100

photodelay = curr;

}

. . .

}

We can see that by using the millis() function to return the current time in milliseconds and a long variable photodelay, we read the resistor value about every 1s. This value is read by performing an analog read on the pin, which can return a value from 0 to 1024. At a low level, this reading can be achieved this way by using the Arduino ADC:

    ADMUX |= (1<<MUX2);        // ADC4

    ADMUX |= (1 << REFS0);   // use AVcc as the reference

The ADMUX register is used to select the input pin by writing the appropriate value on the 5 least significant bits, to select analog pin 3 (ADC4) we just need to write 1 on the MUX2 position (00100 corresponds to ADC4). Then by writing 1 in position REFS0 we are using the reference voltage of 5V.

// 128 prescale for 8Mhz

ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

    ADCSRA |= (1 << ADEN);    // Enable the ADC

    ADCSRA |= (1 << ADSC);    // Start the ADC conversion

The ADCSRA is used firstly to set the proper prescaler, 128, by writing 1 in the 3 least significant positions. This prescaler gives the most accurate results, but each reading will take 104s.

Then we set 1 in ADEN to enable the ADC circuit and we set 1 in ADSC to start the conversion.

    while(ADCSRA & (1 << ADSC)); // waits for the ADC to finish

This loop is used to wait for the end of the conversion, by checking when the bit ADSC in ADCSRA is set to 1 by the MCU.

    int ADCval;

    ADCval = ADCL;

    ADCval = (ADCH << 8) + ADCval; // ADCH is read so ADC can be updated again

The converted value is read from ADCL and ADCH register, which contain the least significant and most significant bits respectively. ADCH only contains 2 useful bits while the rest are reserved, so we are getting a 10bit return value.

Then using the map function provided by the Arduino library, we are able to map each value in the range 0 to 1023 to the appropriate range used by the display we are regulating the brightness of. The LED matrix has 0 to 15 levels of brightness (with the lowest level being still visible), while the 7-segment displays have a range of 0 to 100, but according to the display library the lowest 10 values mean that the display will be completely off, so we are mapping only from 10 to 100.

<https://www.gammon.com.au/adc>

<http://www.resistorguide.com/photoresistor/>