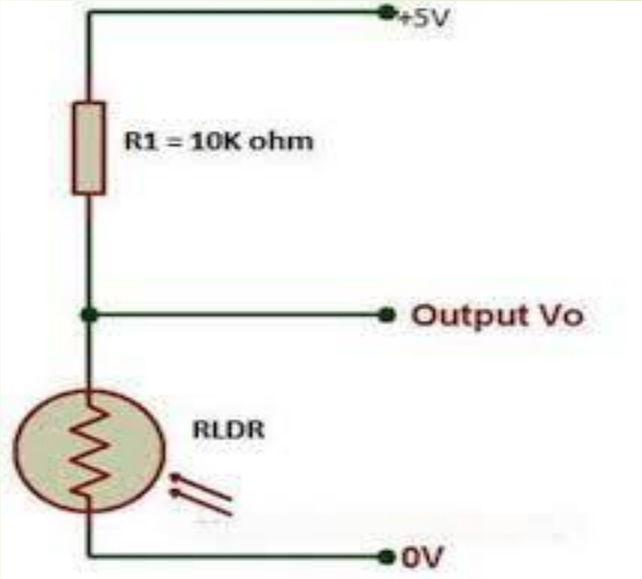
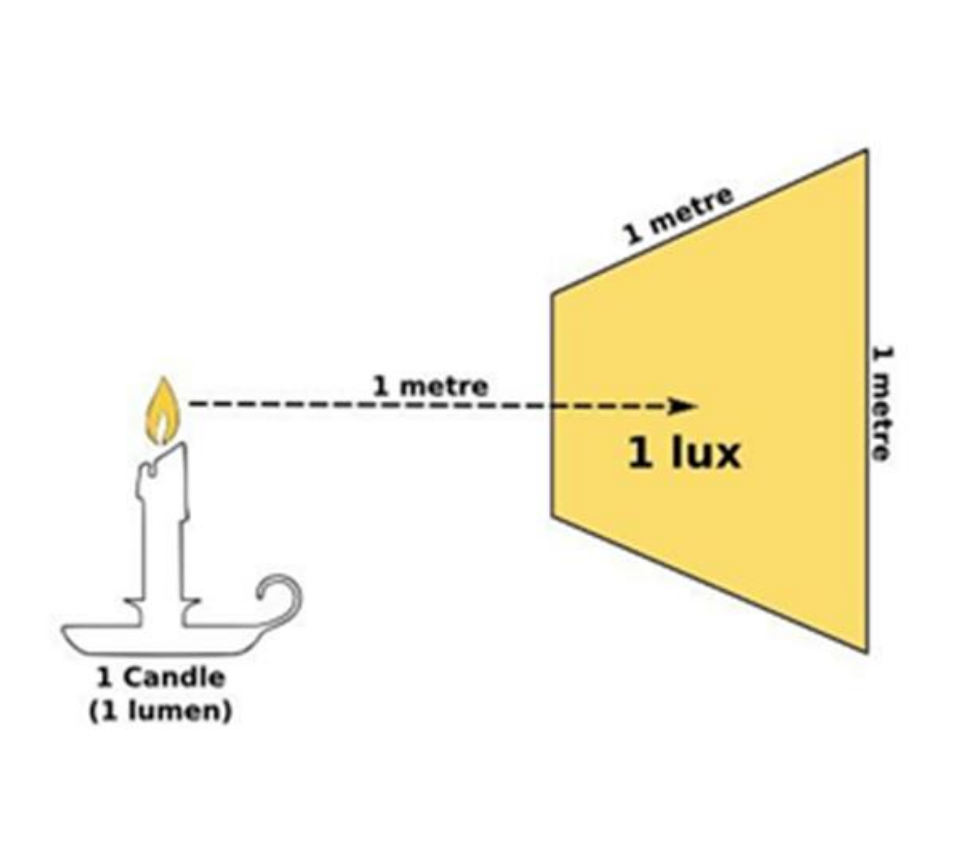
Light Intensity (NSL 19M51)

The light intensity sensor is a device that works like a speedometer. That is, it works by sensing light. But light isn't straightforward. So, the light intensity sensor measures light based on a collector's size. A few examples of collectors are solar phone chargers, landfill solar arrays, etc.



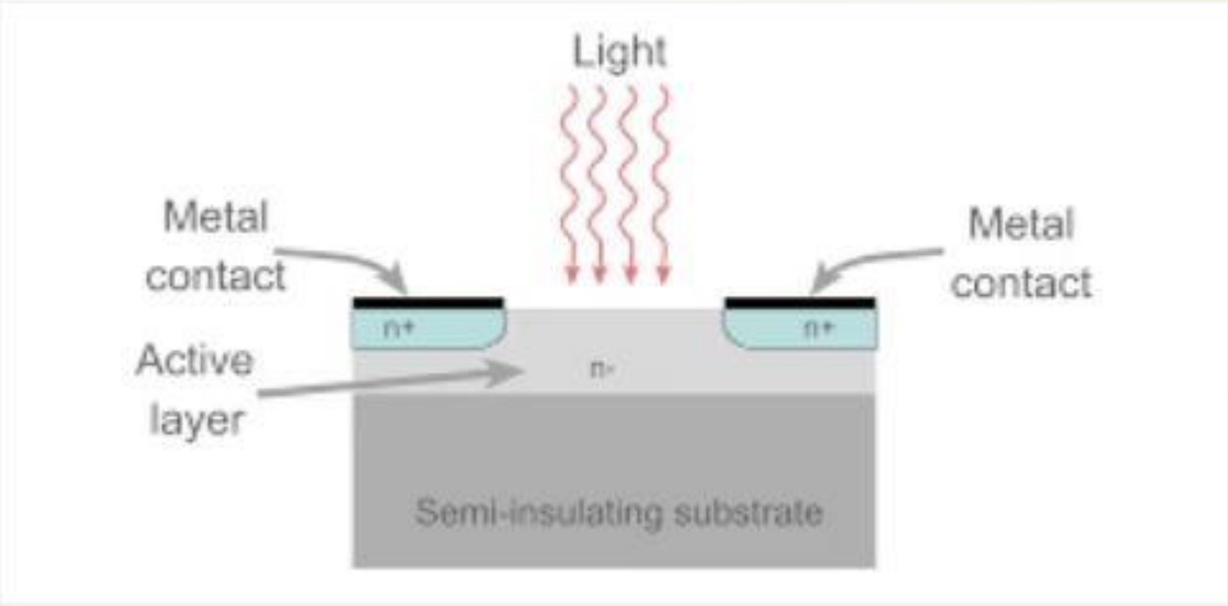
The Light Intensity Sensor Units

Lux is a standardised unit of measurement of light level intensity, which is commonly referred to as "illuminance" or "illumination". So what is exactly 1 lux? A measurement of 1 lux is equal to the illumination of a one metre square surface that is one metre away from a single candle.

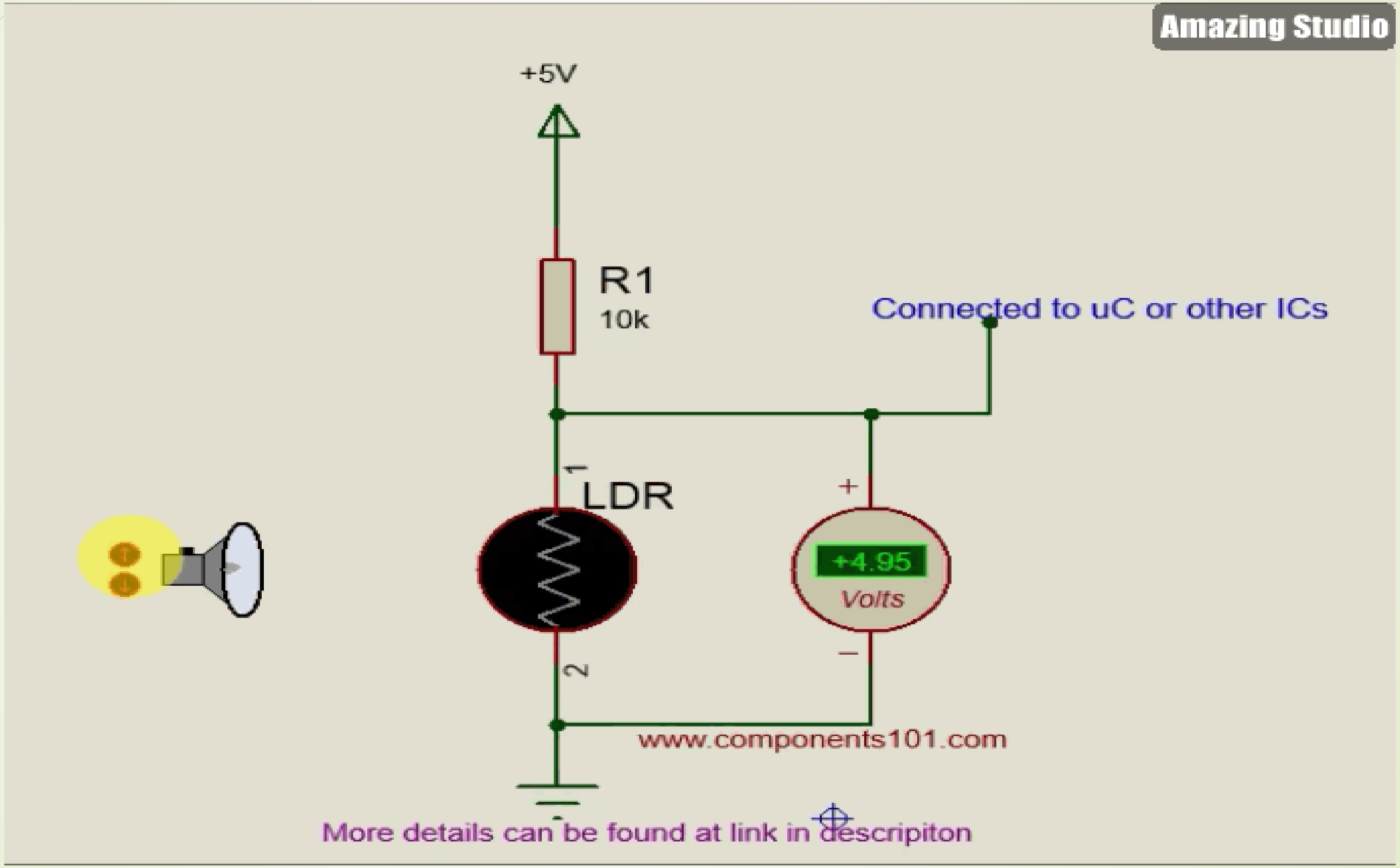


Working Principle of LDR:

- It is nothing more than the fact that when light strikes its surface, the material's conductivity decreases and the electrons in the device's valence band are stimulated to the conduction band. These incident light photons must have energy larger than the semiconductor material's band gap.

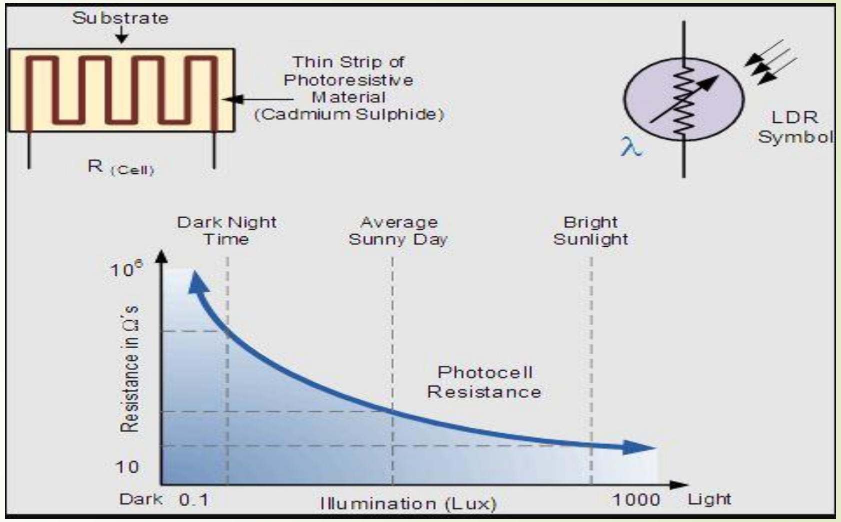


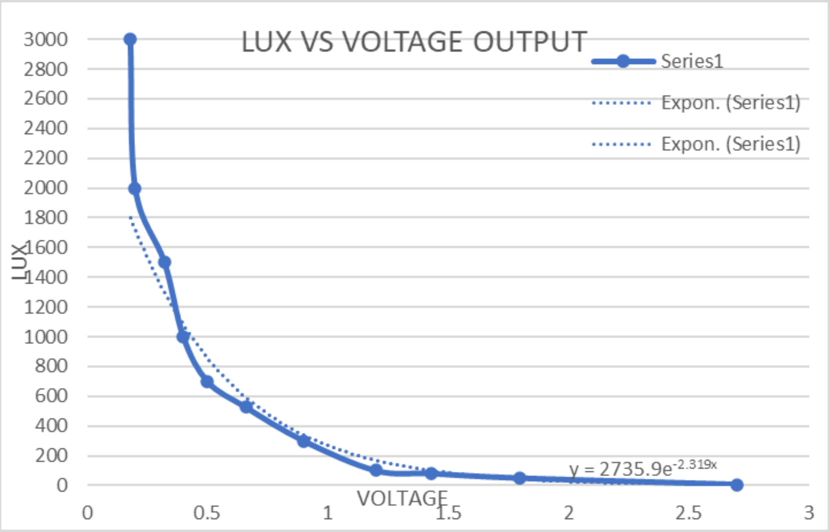
LDR Working Simulation



Resistance VS Lux Ratio

- When an LDR is exposed to light, the resistance increases; this is known as dark resistance. Conversely, when the resistor is exposed to darkness, the resistance decreases. Any device that absorbs light will have significantly less resistance. The light intensity will increase, and the current flow will begin to increase if a stable voltage is applied. Therefore, the characteristics between resistance and illumination for a particular LDR are shown in the diagram below. Since LDRs are not linear devices, the wavelength of the light that strikes them causes a change in their sensitivity. Because it depends on the material employed, some types of photocells are not at all sensitive to a particular range of wavelengths. When light enters a photocell, the resistance changes within 8 milliseconds from 8 to 12 and takes a few extra seconds to return to its initial value after the light has been turned off. Therefore, this is referred to as a resistance recovery rate. This attribute applies to audio compressors.





Reading sensor data

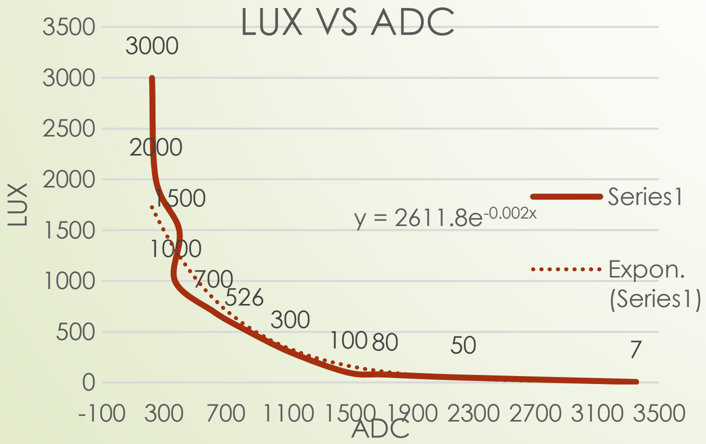
```
float read_LUX(void)
{
        //char buf[100];

        ADC1->SQR5=0;                                   //conversion sequence starts at ch0
        ADC1->CR2|=1;                                   //bit 0, ADC on/off (1=on, 0=off)
        ADC1->CR2|=0x40000000;          //start conversion

        int result = 0;
        float lux = 0;

        while(!(ADC1->SR & 2)){}        //wait for conversion complete
        result=ADC1->DR;//read conversion result

        if((result >0)&&(result < 400)){

                        lux = (-13.427*result) + 5999.8;
                        }
        else if((result >400)&&(result < 1000)){

                        lux = (-0.8761*result) + 1243.7;

                }
        else if((result > 1000) && (result <2000)){

                        lux = (-0.5371*result) + 900.01;

                }
        else if ((result >2000)&&(result<4000)){

                        lux = (-0.0385*result) + 136;
                }
        delay_Ms(100);

        //ADC1->CR2&=~1;        //bit 0, ADC on/off (1=on, 0=off)
        return lux ;
}
```

Signal processing

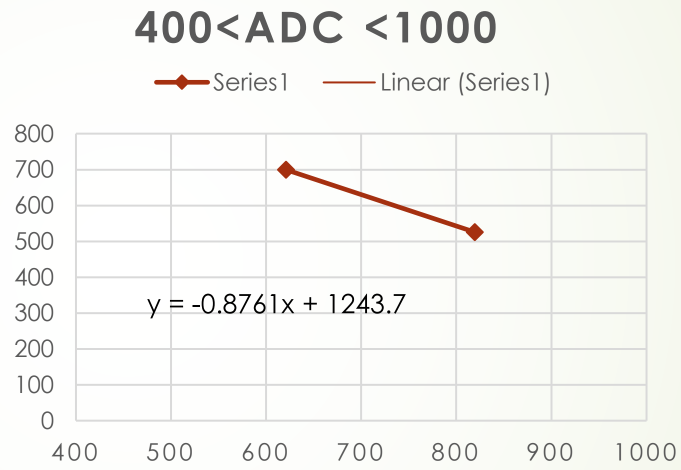
Based on the chosen components and measurements from the table bellow, the voltage range we got starts from 0.18 V to 2.7V, which is compatible with microcontrollers input channel. Between 1000 lux and 1600 lux we had an estimation that we will get some inaccurate values as it can seen from the craves there is a gap between tanged line and the curve itself.
So we decide to divide the exponential curve into 4 linear functions which can make the error ratio small as possible.

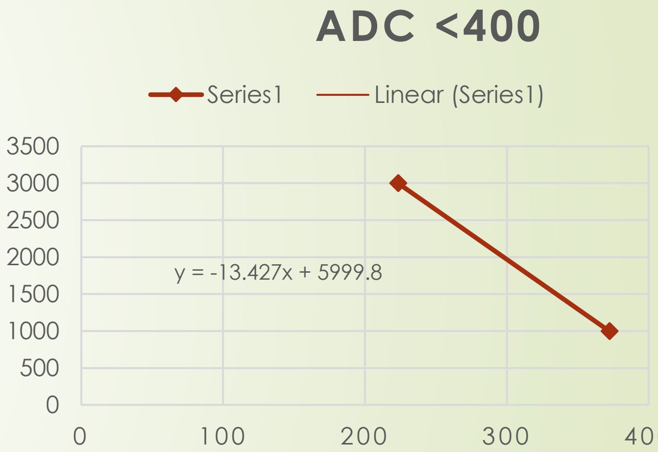
Measurements table

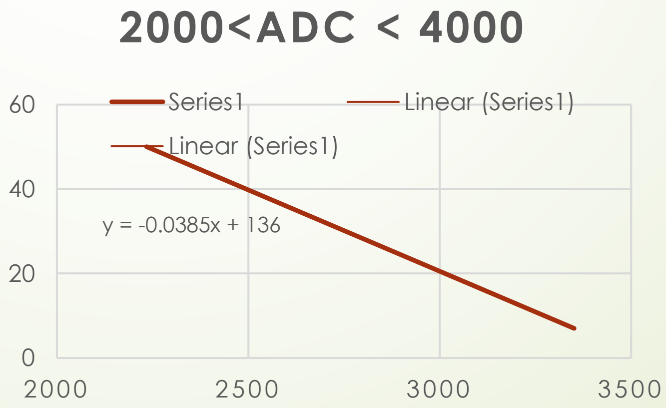
LUX	Output Votage	ADC vref 3.3
7	2.7	3351.27
50	1.8	2234.18
80	1.43	1730
100	1.2	1489.45
300	0.9	1117.09
526	0.66	819.2
700	0.5	620.6
1000	0.4	372.36
1500	0.322	399.67
2000	0.2	248.24
3000	0.18	223.41

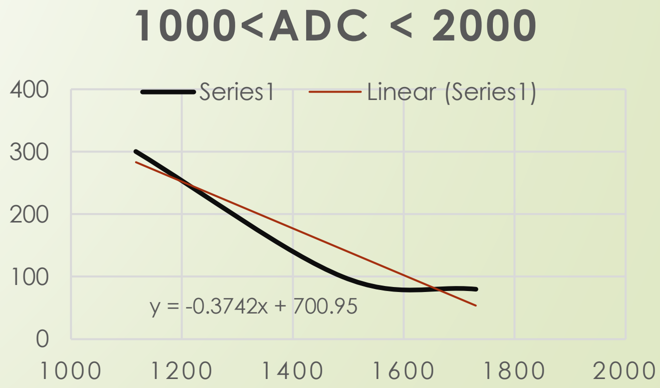


Signal division



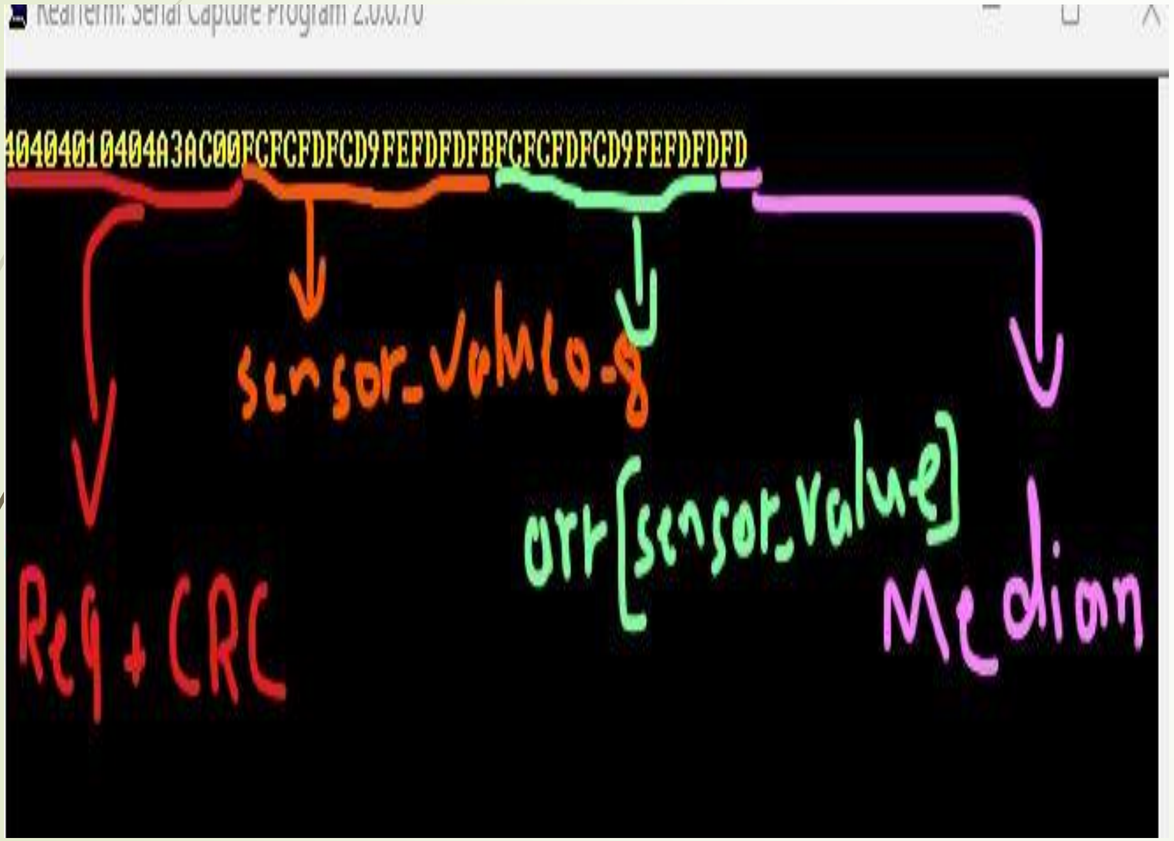






Respond frame processing

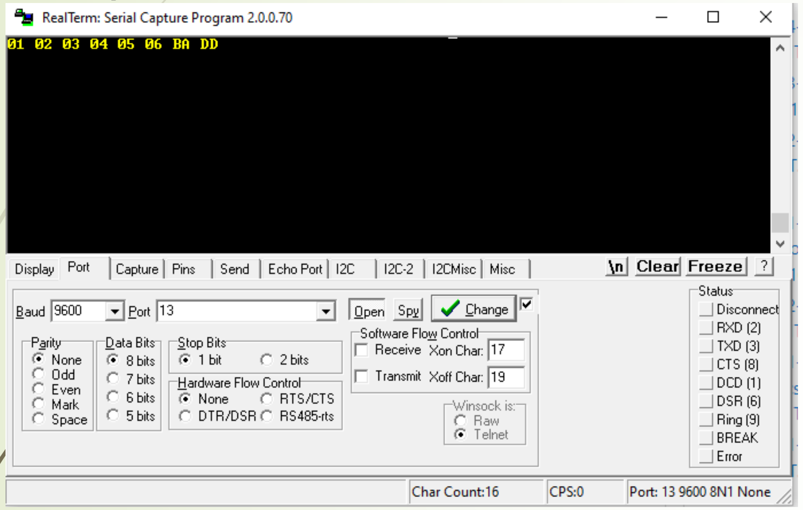
After a request received from the master, we will read the signal 9 times and then calculate the median after that we send the value (median) to respond frame.




```
void Response_frame(int sensor_median_value) {

        GPIOA->ODR |= 0x20;     // LED on while transmitting control RE/TE
        //USART1->CR1 &= ~0x04;   // disable RE pin of USART1
        //USART1->CR1 = 0x08;         // re-enable TE pin of USART1


        char response_frame[7] = {SLAVE_ADDR, 0x04, 0x02, 0, 0, 0, 0};
        char sensor_high_byte = 0;
        char sensor_low_byte = 0;
        unsigned short int calculated_CRC = 0;
        char crc_high_byte = 0;
        char crc_low_byte = 0;
        sensor_high_byte = (sensor_median_value >> 8) | sensor_high_byte;
        sensor_low_byte = sensor_median_value | sensor_low_byte;
        response_frame[3] = sensor_high_byte;
        response_frame[4] = sensor_low_byte;

        calculated_CRC = CRC16(response_frame, 5);
        crc_high_byte = (calculated_CRC >> 8) | crc_high_byte;
        crc_low_byte = calculated_CRC | crc_low_byte;

        response_frame[6]=crc_high_byte;
        response_frame[5]=crc_low_byte;

        for (int i = 0; i < 7; i++) {

                USART1_write(response_frame[i]);
        }

        for (int i = 0; i < 7; i++) {

                        USART2_write(response_frame[i]);
                }
        GPIOA->ODR&=~0x20;            // LED off, turn to receiving mode
        //USART1->CR1 &= ~0x08;   // disable TE pin of USART1
        //USART1->CR1 |= 0x04 ;       // re-enable RE pin of USART1

}
```

Request and respond frames

Req frame

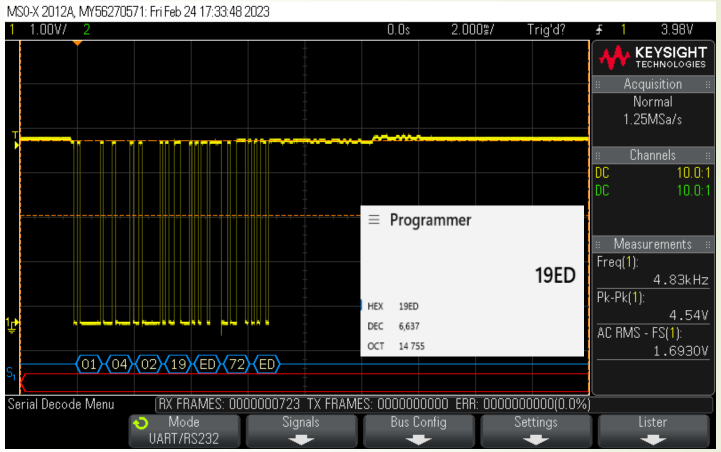


"010203040506" (hex)	
1 byte checksum	21
CRC-16	0xC6BA
CRC-16 (Modbus)	0xDDBA
CRC-16 (Sick)	0x0401
CRC-CCITT (XModem)	0xD90C
CRC-CCITT (0xFFFF)	0xD71C
CRC-CCITT (0x1D0F)	0xE832
CRC-CCITT (Kermit)	0x814F
CRC-DNP	0xF549
CRC-32	0x81F67724

010203040506 Calculate CRC

Input type:: ◯ ASCII ◉ Hex

Res frame



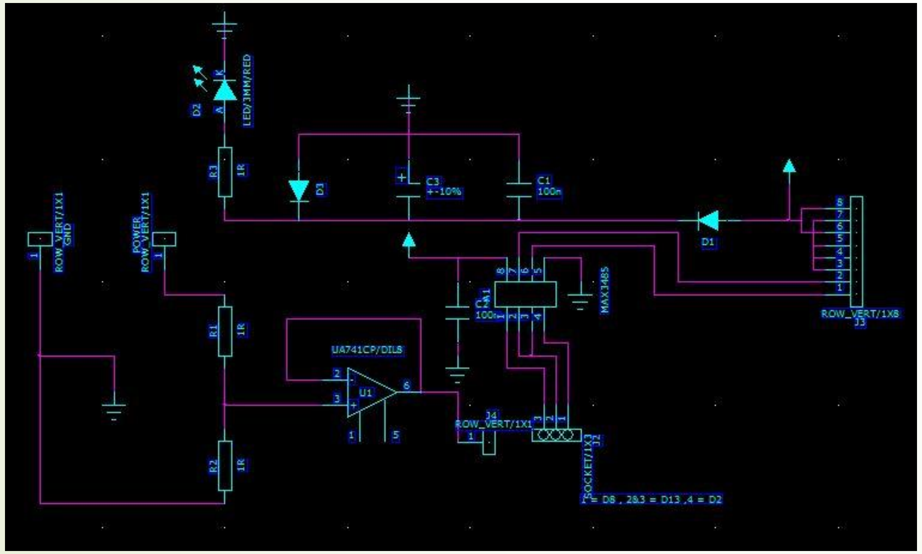
"01040219ED" (hex)	
1 byte checksum	13
CRC-16	0xED56
CRC-16 (Modbus)	0xED72
CRC-16 (Sick)	0xE705
CRC-CCITT (XModem)	0x9BA8
CRC-CCITT (0xFFFF)	0x8AA4
CRC-CCITT (0x1D0F)	0x6A66
CRC-CCITT (Kermit)	0x72B2
CRC-DNP	0xB2E4
CRC-32	0x321FAA49

01040219ED Calculate CRC

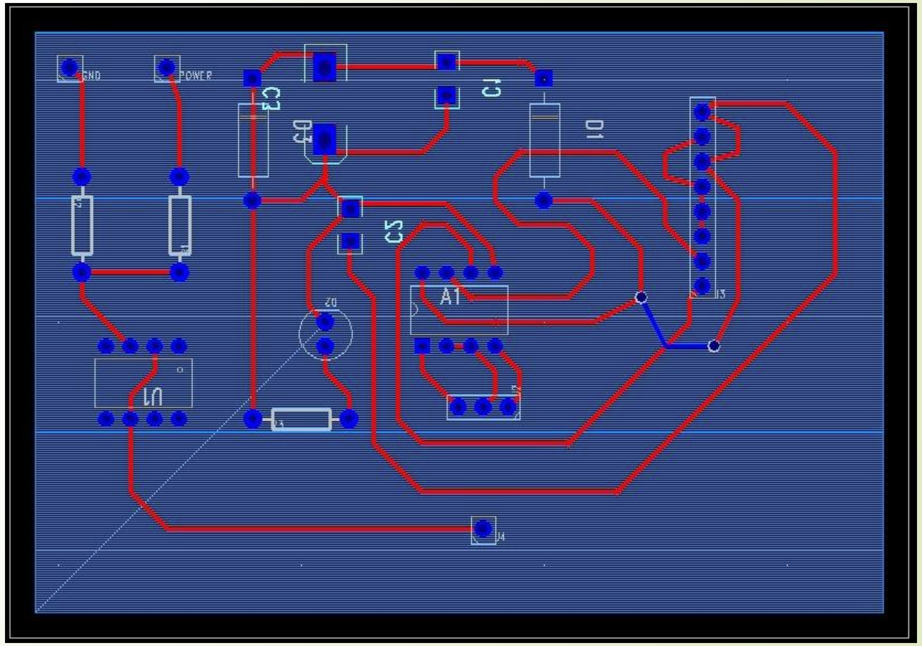
Input type:: ◯ ASCII ◉ Hex

Drawing of a circuit diagram for electronics:

Schematic design



PCB design (Top layer)



Visualize the data on cloud.

Using docker to run database where we can store the data from the master,

LightInt DataBase

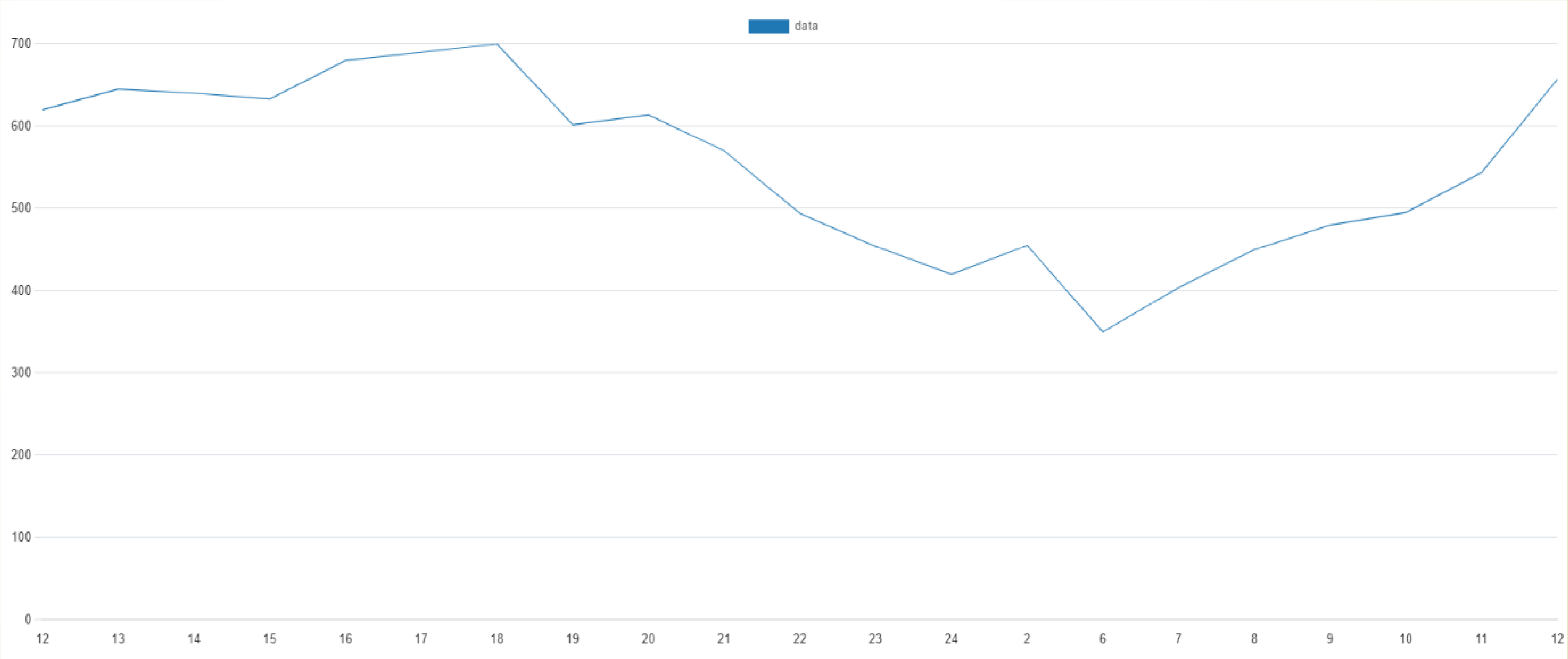
```
db=# select * from lightint;
 data | date
------+------
  620 |   12
  645 |   13
  640 |   14
  633 |   15
  680 |   16
  690 |   17
  700 |   18
  602 |   19
  614 |   20
  570 |   21
  494 |   22
  454 |   23
  420 |   24
  455 |    2
  350 |    6
  404 |    7
  450 |    8
  480 |    9
  495 |   10
  544 |   11
  657 |   12
(21 rows)

db=#
```

Python code to push data to the database

```
GNU nano 6.2                                   pushdata.c *
#include <stdio.h>
#include <curl/curl.h>

int main(void)
{
  CURL *curl;
  CURLcode res;
  char Input_data[10];
  char Input_date[10];

  //int Input_value_data;
  //int Input_value_date;
 // printf("Please Enter LUX value : \n");
  //scanf("%s", &Input_data[0]);
  //printf("Please Enter date : \n");
  //scanf("%s", &Input_date[0]);

//  sprintf(Input_data, "%d", Input_value_data);
 // sprintf(Input_date, "%d", Input_value_date);
  curl = curl_easy_init();
  if(curl) {
    struct curl_slist *headers = NULL;
    headers = curl_slist_append(headers, "Content-Type: application/json");
    char query[1000];
    //char *query = "{\"query\":\"mutation ($Input: CreateLightintInput!) { createLightint(input: $Input) { lightint { data date } } }\",\"variables\":{\"Input\":{\"lightint\":{\"data\":333,\"date\":20}}}}";
    sprintf(query, "{\"query\":\"mutation ($Input: CreateLightintInput!) { createLightint(input: $Input) { lightint { data date } } }\",\"variables\":{\"Input\":{\"lightint\":{\"data\":%s,\"date\":%s}}}}", Input_data, Input_date);
    //printf(query, "{\"query\":\"mutation ($Input: CreateLightintInput!) { createLightint(input: $Input) { lightint { data date } } }\",\"variables\":{\"Input\":{\"lightint\":{\"data\":%d,\"date\":%d}}}}", Input_data, Input_date);


//printf("%s\n", query);

    curl_easy_setopt(curl, CURLOPT_URL, "http:[redacted]);
    curl_easy_setopt(curl, CURLOPT_POST, 1);
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, query);

    res = curl_easy_perform(curl);
    if(res != CURLE_OK)
      fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));

    curl_slist_free_all(headers);
    curl_easy_cleanup(curl);
  }
  return 0;
}
```

Ip address : port number/graphql

Light intensity during 24 hours in closed environment



MODBUS RTU DEVELOPMENT

- I will not go through ModBus frame development In this presentation, but for more details you can find the source code of the whole project on git hub.

ADVANTAGES:

- Sensitivity is High
- Simple & Small devices
- Easily used
- Inexpensive
- There is no union potential.
- The light-dark resistance ratio is high.
- Its connection is simple

DISADVANTAGES:

- The spectral response is limited.
- The best materials have limited temperature stability due to the hysteresis effect.
- Its chemical reaction in stable materials.
- LDR Sensor is only used in situations when the light signal fluctuates dramatically.
- It is not a particularly responsive tool.
- As soon as the operating temperature changes, it gives the wrong results.

Applictations

- Smoke Detector Alarm, Automatic Lighting Clock
- Design of optical circuits
- Proximity switch for photos
- Security measures utilising lasers
- Solar street lighting
- Light metres for cameras
- Radio clocks
- Can be used in dynamic compressors; some compressors adjust the signal gain by connecting LDR and LED to the signal source.