PROGETTO 3 ESPOSIMETRO

Il progetto digitale qui proposto è un "Esposimetro analogico low-cost". La motivazione per farlo è che gli esposimetri professionali sono piuttosto costosi, ma sapere come funzionano è molto utile per usare in modo consapevole apparecchiature fotografiche. Anche quelle digitali!

Per realizzare questo progetto si userà un sensore di luminosità in cui la digitalizzazione è già integrata nel modulo con il sensore (ce ne sono 2 disponibili, in realtà)

I problemi da risolvere sono legati alla lettura del dato e alla registrazione del dato in un file con l'informazione aggiuntiva di data-ora. Queste registrazioni possono risultare utili per rilevazioni ambientali in vari contesti.

Prosecuzione: si tratta di tradurre la luminosità in coppie tempo-di-esposizione/diaframma.

Con gli studenti è importante arrivare a registrare i dati e capire il legame tra luminosità e la coppie di dati tempo-di-esposizione/diaframma.

Condividere l'apprendimento e i progetti è sempre molto importante, prima di tutto con gli studenti. Potrebbe anche essere utile creare un blog per descrivere anche questo progetto.

Useremo 2 diversi componenti:
- BH1750FVI (GY-30)
- TSL2561 Luminosity Sensor (GY-2561)

ATTENZIONE:

ABILITARE I2C (da menu→Preferences→Raspberry Pi Configuration → Interfaces)

Le seguenti informazioni sono state tratte dal blog (2018):

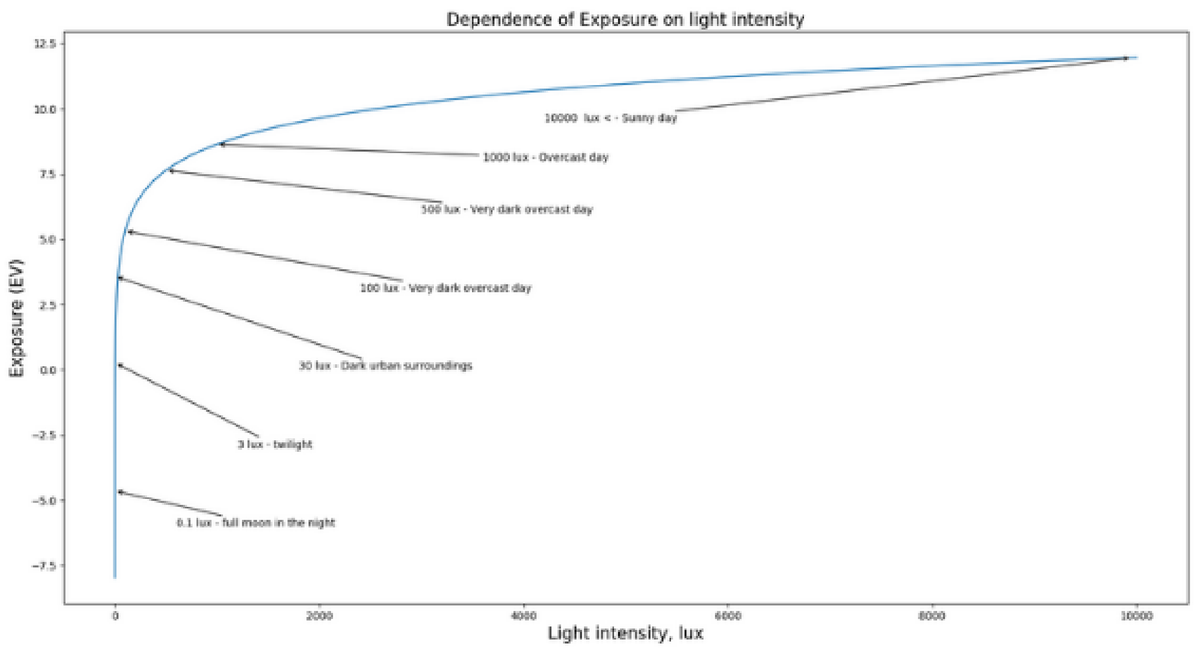
https://dzyla-photo.com/raspberry-pi-based-home-made-light-meter-for-analogue-camera/

di Dawid Zyla, il cui articolo spiega come realizzare un esposimetro a partire da economici (ma qualitativamente buoni) componenti a basso prezzo presenti sul mercato

…
The Internet is the biggest invention indeed, and everything can be found there. After short research (this time the proper one) I have learned some things which I will try to explain here. Sensor which I bought was GY-30 (BH1750FVI) light intensity sensor for ~3$. It is connected through I2C bus to controller, which in this case is Raspberry Pi Zero W. There is publicly available python script which enables to measure the light intensity in lux. According to Wikipedia lux is an SI derived unit of illuminance and luminous emittance, measuring luminous flux per unit area, which means that it is amount of light per area. If the area is a camera sensor and we would sum the light over time (exposure time) we will get a picture. However, lux itself is not telling anything about the exposure (EV) which is required to calculate the ratio of exposure time and aperture.

The dependency of lux and exposure (EV) is described with following equation:
(eq. 1) lux = 2.5 * 2**EV ==> EV = log2(lux/2.5), ISO=100

This tell us that exposure is in the exponent – which basically means that every double intensity of the light would change the exposure only by 1. Plotting this dependency we would get:



This plot also shows nicely that change of light intensity in lux is drastic (over 10k!), when exposure changes only slightly (from ~ -8 to 12). Obtained exposure can be already used for time and aperture calculations. There is only one variable missing: ISO – the sensitivity of the photosensitive element (film CCD or CMOS matrix). EV is usually calculated for the default ISO value of 100. Exposure at various ISO can be calculated from the following equation:

(eq. 2) EV(ISO) = EV(100) + log2(ISO/100) programma per leggere la luminosità da un GY-30:

```
# !/usr/bin/python
# bh1750.py
# Read data from a BH1750 digital light sensor.
#

import smbus
import time

# Define some constants from the datasheet

DEVICE     = 0x23 # Default device I2C address

POWER_DOWN = 0x00 # No active state
POWER_ON   = 0x01 # Power on
RESET      = 0x07 # Reset data register value

# Start measurement at 4lx resolution. Time typically 16ms.
CONTINUOUS_LOW_RES_MODE = 0x13
# Start measurement at 1lx resolution. Time typically 120ms
CONTINUOUS_HIGH_RES_MODE_1 = 0x10
# Start measurement at 0.5lx resolution. Time typically 120ms
CONTINUOUS_HIGH_RES_MODE_2 = 0x11
# Start measurement at 1lx resolution. Time typically 120ms
# Device is automatically set to Power Down after measurement.
ONE_TIME_HIGH_RES_MODE_1 = 0x20
# Start measurement at 0.5lx resolution. Time typically 120ms
# Device is automatically set to Power Down after measurement.
ONE_TIME_HIGH_RES_MODE_2 = 0x21
# Start measurement at 1lx resolution. Time typically 120ms
# Device is automatically set to Power Down after measurement.
ONE_TIME_LOW_RES_MODE = 0x23

#bus = smbus.SMBus(0) # Rev 1 Pi uses 0
bus = smbus.SMBus(1)  # Rev 2 Pi uses 1

def convertToNumber(data):
  # Simple function to convert 2 bytes of data
  # into a decimal number. Optional parameter 'decimals'
  # will round to specified number of decimal places.
  result=(data[1] + (256 * data[0])) / 1.2
  return (result)

def readLight(addr=DEVICE):
  # Read data from I2C interface
  data = bus.read_i2c_block_data(addr,ONE_TIME_HIGH_RES_MODE_1)
  return convertToNumber(data)

def main():

  while True:
    lightLevel=readLight()
    print("Light Level : " + format(lightLevel,'.2f') + " lx")
    time.sleep(2)

if __name__=="__main__":
   main()
```

Ambient Light Sensor IC Series
Digital 16bit Serial Output Type
Ambient Light Sensor IC
BH1750FVI

●Descriptions
BH1750FVI is an digital Ambient Light Sensor IC for I C bus interface. This IC is the most suitable to obtain the ambient light data for adjusting LCD and Keypad backlight power of Mobile phone. It is possible to detect wide range at High resolution.
(1 - 65535 lx).

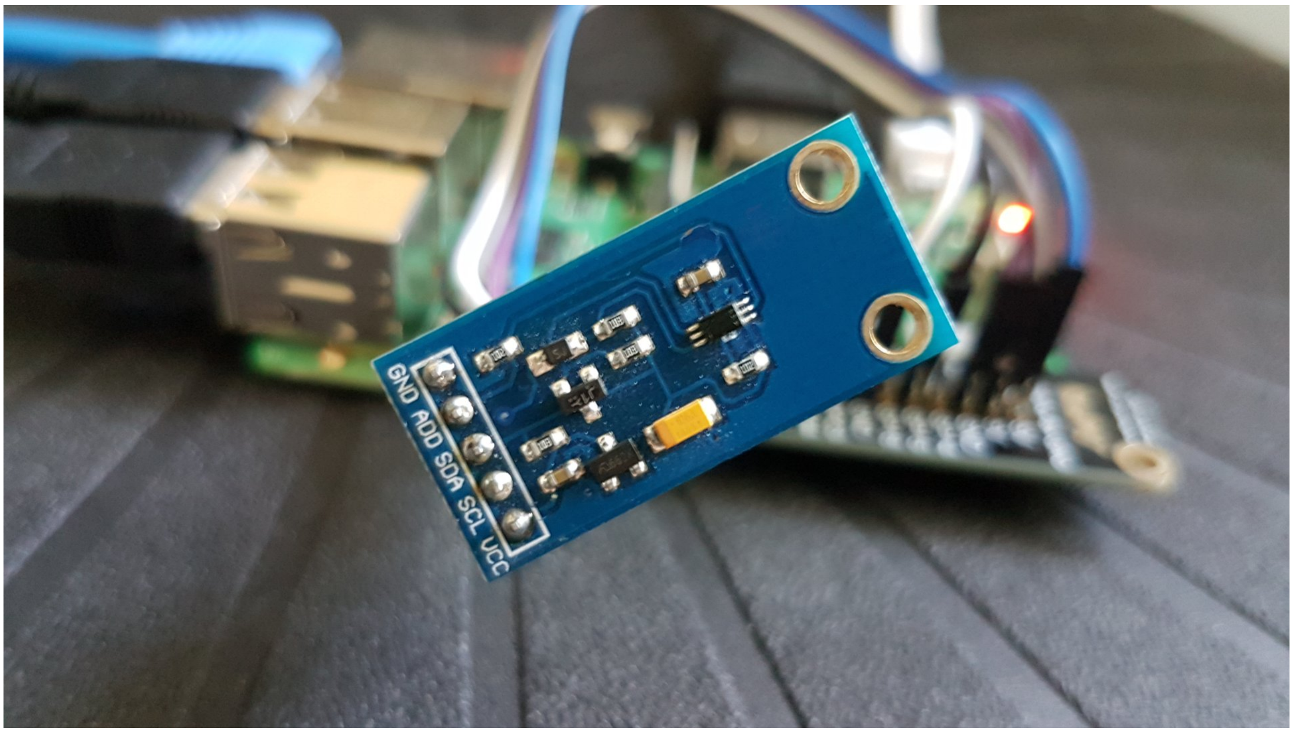
●Features
1) I C bus Interface (f / s Mode Support)
2) Spectral responsibility is approximately human eye response
3) Illuminance to Digital Converter
4) Wide range and High resolution. (1 - 65535 lx)
5) Low Current by power down function
6) 50Hz / 60Hz Light noise reject-function
7) 1.8V Logic input interface
8) No need any external parts
9) Light source dependency is little. (ex. Incandescent Lamp. Fluorescent Lamp. Halogen Lamp. White LED. Sun Light)
10) It is possible to select 2 type of I 2 C slave-address.
11) Adjustable measurement result for influence of optical window
(It is possible to detect min. 0.11 lx, max. 100000 lx by using this function.)
12) Small measurement variation (+/- 20%)
13) The influence of infrared is very small.

●Applications
Mobile phone, LCD TV, NOTE PC, Portable game machine, Digital camera, Digital video camera, PDA, LCD display

https://www.raspberrypi-spy.co.uk/2015/03/bh1750fvi-i2c-digital-light-intensity-sensor/

Using the BH1750FVI I2C Digital Light Sensor

By Matt on March 31, 2015 I2C, Sensors, Tutorials & Help



The BH1750 device is a digital light sensor which uses the I2 interface. This allows it to be connected to the Raspberry Pi with only four wires.

The module allows quick and cheap ambient light level measurement and the light level can be read from it as a digital number due to the built in 16-bit analogue-to-digital converter. The device itself is commonly used in mobile phones, LCD TVs and digital cameras.

The module I've got measures only 32 x 16mm. I soldered a five pin header onto the PCB and this allowed me to plug it onto a piece of breadboard.

The BH1750 Ambient Light Sensor IC datasheet provides all the technical details some of which are used in my example Python script.

Configure I2C Interface

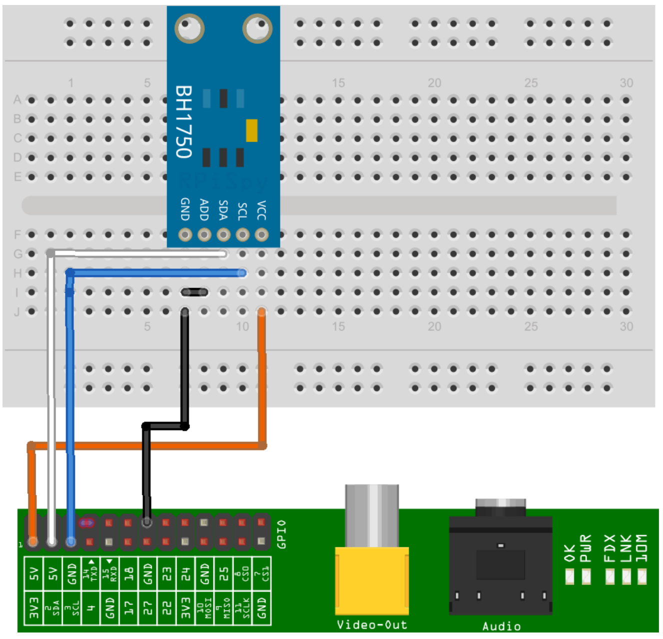
In order to use this module you must enable the I2C interface on the Raspberry Pi as it is not enabled by default. This is a fairly easy process and is described in my Enabling The I2C Interface On The Raspberry Pi tutorial.

Connecting Light Sensor Hardware to the Pi

The following table shows how I connected the module pins on the PCB to the Pi's GPIO header (P1). Please refer to my GPIO header guide for a diagram.

Module PCB	Desc	GPIO Header Pins
GND	Ground	P1-14
ADD	Address select	P1-14 non necessario
SDA	I2C SDA	P1-03
SCL	I2C SCL	P1-05
VCC	3.3V	P1-01

Here is a diagram of a breadboard setup. If you are connecting the module's five pins directly to the Pi you only need five female-female wires.



This Fritzing diagram uses a custom part I created for my module. Other modules may have a different pin arrangement so make sure you are connecting the correct pins to the Pi if yours is slightly different.

With the device connected and the Pi powered up the "i2cdetect" command should show the device with address 0x23.

Example Python Script to Show Light Level

You can download an example script to read the light level from the module and print it to the screen.

Use the following command :

```
wget https://bitbucket.org/MattHawkinsUK/rpispy-
misc/raw/master/python/bh1750.py
```

or use this link in a browser.

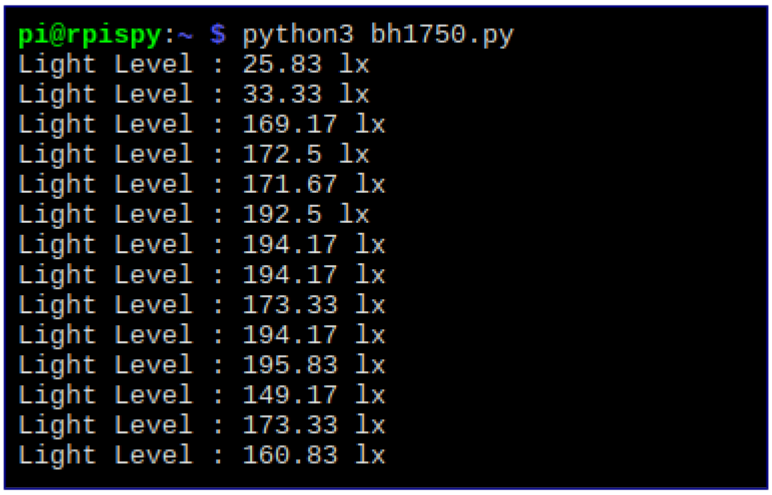
In order to run it you can use :

```
python bh1750.py
```

or for Python 3 :

```
python3 bh1750.py
```

The output should look something like :



```
pi@rpispy:~ $ python3 bh1750.py
Light Level : 25.83 lx
Light Level : 33.33 lx
Light Level : 169.17 lx
Light Level : 172.5 lx
Light Level : 171.67 lx
Light Level : 192.5 lx
Light Level : 194.17 lx
Light Level : 194.17 lx
Light Level : 173.33 lx
Light Level : 194.17 lx
Light Level : 195.83 lx
Light Level : 149.17 lx
Light Level : 173.33 lx
Light Level : 160.83 lx
```

The while loop keeps taking readings every half second until you press CTRL-C.

Python Script Notes :

- The import statements import Python libraries used in the rest of the code including smbus which handles the I2C interface.
- All I2C devices must have an address. In this example I tied the ADDR pin to ground so the address used by the device is 0x23. The address becomes 0x5C if the ADDR pin is tied to 3.3V.
- The block of constants are listed in the datasheet and define the different modes the device can operate in. I define all of them but only use "ONE_TIME_HIGH_RES_MODE_1". The other modes are only of any real interest if you need to take high speed readings.
- The "smbus.SMBus(1)" function setups the I2C interface.
- The "read_i2c_block_data" function is used to read 2 bytes of data from the device using the ONE_TIME_HIGH_RES_MODE_1.
- The convertToNumber function is then used to convert those 2 bytes of data into a number. The 1.2 value in the calculation is defined in the datasheet

TSL2561 Luminosity Sensor (GY-2561)

The TSL2561 luminosity sensor is an advanced digital light sensor, ideal for use in a wide range of light situations.
Compared to low cost CdS cells, this sensor is more precise, allowing for exact Lux calculations and can be configured for different gain/timing ranges to detect light ranges from up to 0.1 - 40,000+ Lux on the fly. The best part of this sensor is that it contains both infrared and full spectrum diodes! That means you can seperately measure infrared, full-spectrum or human-visible light. Most sensors can only detect one or the other, which does not accurately represent what human eyes see (since we cannot perceive the IR light that is detected by most photo diodes).

The sensor has a digital (i2c) interface. You can select one of three addresses so you can have up to three sensors on one board - each with a different i2c address. The built in ADC means you can use this with any microcontroller, even if it doesn't have analog inputs. The current draw is extremely low, so its great for low power data-logging systems. about 0.5mA when actively sensing, and less than 15 uA when in powerdown mode.

Some Stats:

Approximates Human eye Response
Precisely Measures Illuminance in Diverse Lighting Conditions
Temperature range: -30 to 80 *C
Dynamic range (Lux): 0.1 to 40,000 Lux
Voltage range: 2.7-3.6V
Interface: I2C

Programma:

```
# for TSL2561
# This code is designed to work with the TSL2561_I2CS I2C Mini Module

import smbus
import time

# Get I2C bus
bus = smbus.SMBus(1)

# TSL2561 address, 0x39(57)
# Select control register, 0x00(00) with command register, 0x80(128)
#                   0x03(03)        Power ON mode
bus.write_byte_data(0x39, 0x00 | 0x80, 0x03)
# TSL2561 address, 0x39(57)
# Select timing register, 0x01(01) with command register, 0x80(128)
#                   0x02(02)        Nominal integration time = 402ms
bus.write_byte_data(0x39, 0x01 | 0x80, 0x02)

time.sleep(0.5)

# Read data back from 0x0C(12) with command register, 0x80(128), 2 bytes
# ch0 LSB, ch0 MSB
data = bus.read_i2c_block_data(0x39, 0x0C | 0x80, 2)

# Read data back from 0x0E(14) with command register, 0x80(128), 2 bytes
# ch1 LSB, ch1 MSB
data1 = bus.read_i2c_block_data(0x39, 0x0E | 0x80, 2)

# Convert the data
ch0 = data[1] * 256 + data[0]
ch1 = data1[1] * 256 + data1[0]

# Output data to screen
print "Full Spectrum(IR + Visible) :%d lux" %ch0
print "Infrared Value :%d lux" %ch1
print "Visible Value :%d lux" %(ch0 - ch1)
pi@raspberrypi:~/ESPOSIMETRO/GY-2561 $
```

Programma con loop:

```
# for TSL2561
# This code is designed to work with the TSL2561_I2CS I2C Mini Module

import smbus
import time

# Get I2C bus
bus = smbus.SMBus(1)

while True:

# TSL2561 address, 0x39(57)
# Select control register, 0x00(00) with command register, 0x80(128)
#               0x03(03)        Power ON mode
        bus.write_byte_data(0x39, 0x00 | 0x80, 0x03)
# TSL2561 address, 0x39(57)
# Select timing register, 0x01(01) with command register, 0x80(128)
#               0x02(02)        Nominal integration time = 402ms
        bus.write_byte_data(0x39, 0x01 | 0x80, 0x02)

        time.sleep(0.5)

# Read data back from 0x0C(12) with command register, 0x80(128), 2 bytes
# ch0 LSB, ch0 MSB
        data = bus.read_i2c_block_data(0x39, 0x0C | 0x80, 2)

# Read data back from 0x0E(14) with command register, 0x80(128), 2 bytes
# ch1 LSB, ch1 MSB
        data1 = bus.read_i2c_block_data(0x39, 0x0E | 0x80, 2)

# Convert the data
        ch0 = data[1] * 256 + data[0]
        ch1 = data1[1] * 256 + data1[0]

# Output data to screen
        print "Full Spectrum:%d lux - " %ch0+"Infrared :%d lux - " %ch1+" Visible:%d lux" %(ch0 - ch1)

        time.sleep(1)
```