```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// SOME DESCRIPTIONS HERE!
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// ADS1115 FUNCIONANDO OK >-->  DADOS PARA CALIBRACAO ONLINE >--> OLIVO
/*  MEDIDAS COM LAMPADA LED DE 20 W A 3 CENTIMETROS DE DISTANCIA
 *  >--> APRESENTARAM COMO RESULTADO ENTRE 15670 A 16573 CONTAGENS E
 *  TENSOES ENTRE 2,92 A 3,11 V
 *  DENTRO DA GAVETA AS 12 HORAS COM ILUMINACAO EXTERNA EM PENUMBRA
 *  >--> APRESENTARAM COMO RESULTADO ENTRE 13 A 50 CONTAGENS E
 *  TENSOES ENTRE 0,002435 A 0,009375 V. */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// BH1750FVI SENSOR FUNCIONANDO COM ESTE CODIGO - 31/05/2018
/* This is a simple code to test BH1750FVI Light senosr communicate
 *  using I2C Protocol, this library enable 2 slave device address
 *  Main address  0x23 |||||||||||| secondary address 0x5C           */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// INCLUDES AND DEFINES
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
#include <Wire.h>
#include<Adafruit_ADS1015.h>
//https://github.com/adafruit/Adafruit_ADS1X15
#include<BH1750FVI.h>
//https://github.com/Genotronex/BH1750FVI_Master
// ATENCAO VEM DENTRO DE UMA SUBPASTA - TEM QUE TRAZER PARA UMA ANTES >-->
OK?
#define BAUD_RATE     115200
// GPIO ESP8266 TO CONTROLL LED TO TURBIDITY MEASUREMENT
#define LED_TURBIDITY   D8  //  PAY ATTENTION HERE, TOO >--> OK?
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// LIBRARY PARAMETERS
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
BH1750FVI LightSensor;
Adafruit_ADS1115 ads(0x48);
// TEXAS INSTRUMENTS ADS1115 4 MULTIPLEXED I2C ADC ADDRESS
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// SECOND DEFINE GLOBAL VARIABLES
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
float VA0, VA1, VA2, VA3 = 0.0;
int16_t ADC0, ADC1, ADC2, ADC3;
// ADC0 = LDR1, ADC1 = LDR2, ADC2 = LDR3, ADC3 = LDR4 >---> NOT FORGET!!
int ADC_CONV_TIME = 5; // adc conversion time in miliseconds
// TO PRODUCTION SYSTEMS DO NOT FORGET TO REMOVE ALL DELAY TIMES!!! OK?
// AND TO REMOVE ALL SERIAL PRINTS THAT DECREASE I2C RELIABILITY!!! OK?
uint16_t lux = 0; //  LUX INTENSITY LONG INTEGER
// TO PRODUCTION SYSTEMS DO NOT FORGET TO REMOVE ALL DELAY TIMES!!! OK?
// AND TO REMOVE ALL SERIAL PRINTS THAT DECREASE I2C RELIABILITY!!! OK?
unsigned COUNT = 0; // just a single measurements counter here!
```

```
int TIME_BETWEEN_MEAS = 6000; //  TIME BETWEEN MEASUREMENTS
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// SETUP FUNCTION
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
void setup(void){
pinMode(LED_TURBIDITY, OUTPUT);
digitalWrite(LED_TURBIDITY, LOW);
Serial.begin(BAUD_RATE);
ads.begin();
LightSensor.begin();
/*  Set the address for this sensor you can use 2 different address
 Device_Address_H "0x5C"  >---> Device_Address_L "0x23"
 you must connect Addr pin to A3.  */
// LightSensor.SetAddress(Device_Address_H);//Address 0x5C
// To adjust the slave on other address , uncomment this line
LightSensor.SetAddress(Device_Address_L); //Address 0x23
LightSensor.SetMode(Continuous_H_resolution_Mode);}
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
/*  set the Working Mode for this sensor
    Select the following Mode:
   Continuous_H_resolution_Mode
   Continuous_H_resolution_Mode2
   Continuous_L_resolution_Mode
   OneTime_H_resolution_Mode
   OneTime_H_resolution_Mode2
   OneTime_L_resolution_Mode
    The data sheet recommanded To use Continuous_H_resolution_Mode  */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
/* illuminance is a measure of how much luminous flux is spread over
 *  a given area. One can think of luminous flux (measured in lumens) as
 *  a measure of the total "amount" of visible light present, and the
 *  illuminance as a measure of the intensity of illumination on a
 *  surface. Lumen : The unit for the quantity of light flowing from a
 *  source in any one second (the luminous power, or luminous flux) is
 *  called the lumen. In our sensor we will take a reading from it in
 *  Lux which is equal to one lumen per square metre: Lux = 1 Lm/m2
*/
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// READ ALL MULTIPLEXED ADC FUNCTION
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
float ReadAllADC(){
  ADC0 = ads.readADC_SingleEnded(0);
 delay(ADC_CONV_TIME);
   ADC1 = ads.readADC_SingleEnded(1);
   delay(ADC_CONV_TIME);
     ADC2 = ads.readADC_SingleEnded(2);
     delay(ADC_CONV_TIME);
```

```
       ADC3 = ads.readADC_SingleEnded(3);
       delay(ADC_CONV_TIME);
  VA0 = (ADC0 * 0.1875)/1000;
    VA1 = (ADC1 * 0.1875)/1000;
      VA2 = (ADC2 * 0.1875)/1000;
        VA3 = (ADC3 * 0.1875)/1000;
return(ADC0, ADC1, ADC2, ADC3, VA0, VA1, VA2, VA3);}
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// LUX READINGS FUNCTION - LUMINOUS FLUX [LUMEN PER SQUARE METER]
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
uint16_t ReadLUX(){
// FIRST OF ALL TURN ON THE WHITE LED >--> OK!
digitalWrite(LED_TURBIDITY, HIGH);
if(digitalRead(LED_TURBIDITY)){
lux = LightSensor.GetLightIntensity();}    // Get Lux value
return (lux);}
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// MAIN LOOP FUNCTION
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
void loop(void){
ReadAllADC();
ReadLUX();
Serial.println("|-------------------------------------------------------|");
Serial.println("|> LUCIANO'S WATER TURBIDITY MEASUREMENTS FIRMWARE  <|");
Serial.print("|> AIN0 >--> LDR1: "); Serial.print(ADC0);
  Serial.print("   VA0 >--> V_LDR1: "); Serial.println(VA0, 7);
   Serial.print("|> AIN1 >--> LDR2: "); Serial.print(ADC1);
   Serial.print("   VA1 >--> V_LDR2: "); Serial.println(VA1, 7);
     Serial.print("|> AIN2 >--> LDR3: "); Serial.print(ADC2);
     Serial.print("   VA2 >--> V_LDR3: "); Serial.println(VA2, 7);
       Serial.print("|> AIN3 >--> LDR4: "); Serial.print(ADC3);
       Serial.print("   VA3 >--> V_LDR4: "); Serial.println(VA3, 7);
  Serial.print("|> LUX  >--> LUMINOUS FLUX:    "); Serial.print(lux);
  Serial.println(" lux [Lumen/m2]");
Serial.print("|> LED  >--> TURBIDITY LOGIC LEVEL IS: ");
Serial.println(digitalRead(LED_TURBIDITY));
Serial.print("|> MEASUREMENT NUMBER: "); Serial.print(COUNT++);
Serial.print(" each one after: ");  Serial.print(TIME_BETWEEN_MEAS/1000);
Serial.println(" seconds");       delay(TIME_BETWEEN_MEAS);
digitalWrite(LED_TURBIDITY, LOW);
Serial.print("|> LED  >--> TURBIDITY LOGIC LEVEL IS: ");
Serial.println(digitalRead(LED_TURBIDITY));   delay(TIME_BETWEEN_MEAS/5);}
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
// END OF CODE!
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
```