

ABSTRACT



Street Lights have become an essential part of our lives as they are an important source of light at evening and night time. The main advantage of street lights is that they increase safety and prevents accidents and collisions.

Auto Intensity Control of Street Lights is a simple project where the intensity of the street lights is automatically controlled based on the sunlight conditions. Generally, street lights are turned on during evening time and will continue to glow till morning.

This might result is unnecessary usage of power as the lights will be glowing at full intensity all the times. But using the Auto Intensity Control of Street Lights using Arduino, you can control the intensity based on the ambient lighting conditions. As an additional power saving feature, We have used LEDs for street lights.



`

content

* Introduction
* Components
* CIRCUIT DIAGRAM
* CRCUIT DESIGN
* WORKING
* code
* reference

INTRODUCTION:

Street lights are the major requirement in today’s life of transportation for safety purposes and avoiding accidents during night. Despite that in today’s busy life no one bothers to switch it off/on when not required. The project introduced here gives solution to this by eliminating manpower and reducing power consumption. This requires three basic components i.e. LDR, Sensors and microcontroller

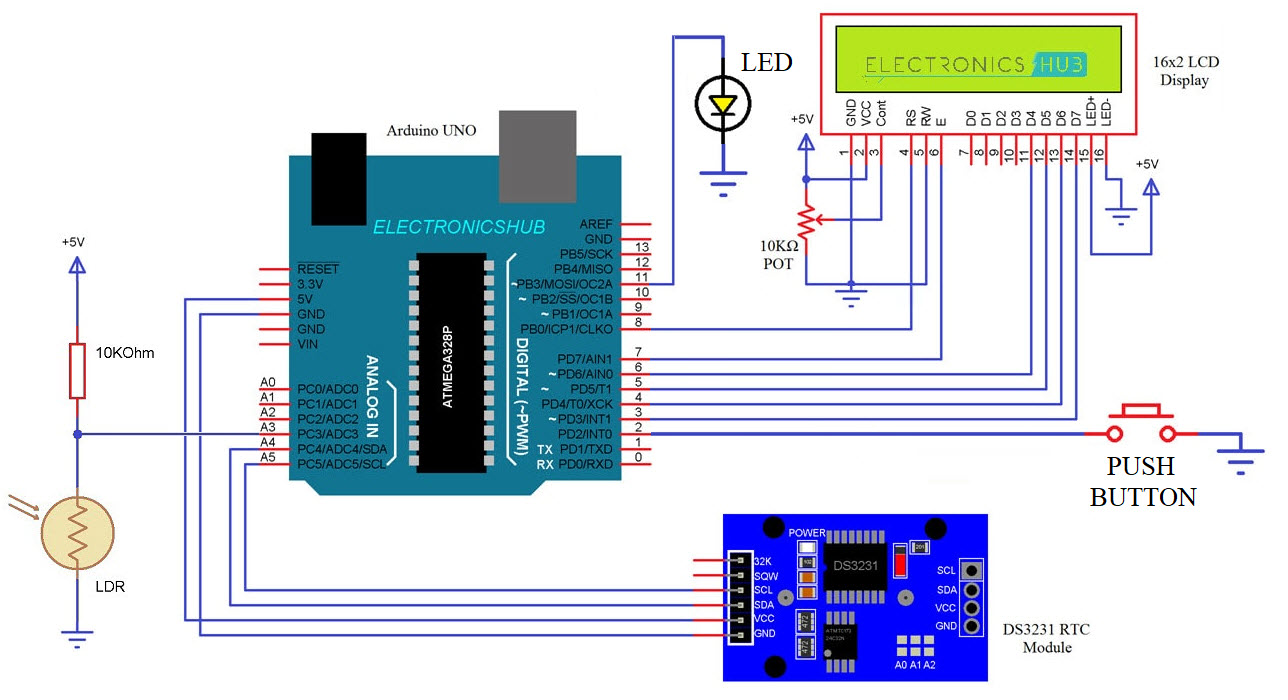
During daytime there is no requirement of street lights so the LDR keeps the street light off until the light level is low or the frequency of light is low the resistance of the LDR is high. Thus the street lights do not glow.

As soon as the light level goes high or if light falling on the device is of high enough frequency the street lights start to glow.

COMPONENTS:

1. Arduino UNO
2. RTC Module
3. LDR
4. 16×2 LCD Display
5. LED
6. 10KΩ Potentiometer
7. 10KΩ Resistor
8. Push Button
9. Connecting Wires
10. Breadboard

CIRCUIT DIAGRAM:

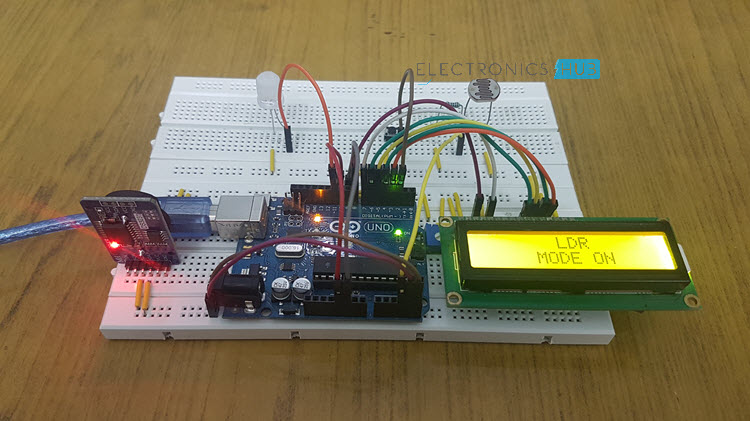
The following image shows the circuit diagram of the Auto Intensity Control of Street Lights using Arduino project.

CIRCUIT DESIGN:

First connect the SDA and SCL pins of RTC Module to A4 (SDA) and A5 (SCL) pins of Arduino. A 10KΩ Resistor and an LDR are connected in a voltage divider format and its output is given to A3 pin (of Arduino).

The data pins of 16×2 LCD Module i.e. D4 – D7 are connected to 6, 5, 4 and 3 pins of Arduino respectively. The RS and E pins are connected to pins 8 and 7.

A Push Button is connected to Pin 2 of Arduino and an LED is connected to Pin 11



***Arduino uno:***

**Arduino Uno** is a microcontroller board based on the ATmega328P ([datasheet](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.



WORKING:

After making the connections and uploading the code to Arduino, turn on the Power supply to the project. Initially, the Arduino runs in RTC Mode where there are two times set in the code: the ON TIME and the OFF TIME.

Arduino compares the ON TIME with the time from RTC Module and when they match, the LED is turned ON. After this, the Arduino waits for the OFF TIME and once the time from RTC Module reaches the OFF TIME, the LED is turned OFF.

During anytime of this operation, if the button (connected as an external interrupt to Pin 2) is pushed, the Arduino enters LDR Mode. In this mode, the Arduino reads the value of the LDR from A3 and based on the value, it adjusts the intensity of the LED.

In order to switch back to RTC Mode, all you have to do is push the button.

CODE:

|  |
| --- |
| #include <Wire.h> |
|  | #include <LiquidCrystal.h> |
|  | #include "RTClib.h" |
|  | #define ON 0 |
|  | #define OFF 1 |
|  | DateTime now; |
|  |  |
|  | RTC\_DS3231 rtc; |
|  | LiquidCrystal lcd(8, 7, 6, 5, 4, 3); // (rs, e, d4, d5, d6, d7) |
|  |  |
|  | const int buttonPin = 2; |
|  | const int led=11; |
|  | int nob = A3; |
|  | int val = 0; |
|  | int val1 = 0; |
|  | int path=1; |
|  | int a=1; |
|  | int previousState = HIGH; |
|  | unsigned int previousPress; |
|  | volatile int buttonFlag; |
|  | int buttonDebounce = 20; |
|  |  |
|  | int on\_hour=1; |
|  | int on\_minute=1; |
|  | int on\_second=10; |
|  |  |
|  | int off\_hour=23; |
|  | int off\_minute=57; |
|  | int off\_second=10; |
|  |  |
|  | int c\_hour=0; |
|  | int c\_minute=0; |
|  | int c\_second=0; |
|  |  |
|  | int onOrOffFlag = ON; |
|  |  |
|  | void showDate(void); |
|  | void showTime(void); |
|  | void showDay(void); |
|  |  |
|  | void loadHandler(int, int , int , int , int , int , int , int , int ); |
|  |  |
|  | typedef struct userTime |
|  | { |
|  | int temp\_hour; |
|  | int temp\_minute; |
|  | int temp\_second; |
|  | }userTime\_t; |
|  | unsigned char checkLessThanOrEqual(userTime\_t , userTime\_t); |
|  |  |
|  |  |
|  | void setup () |
|  | { |
|  | Serial.begin(9600); |
|  | lcd.begin(16,2); |
|  |  |
|  | pinMode(buttonPin, INPUT\_PULLUP); |
|  | pinMode(led,OUTPUT); |
|  | attachInterrupt(digitalPinToInterrupt(buttonPin), button\_ISR, CHANGE); |
|  |  |
|  | if (! rtc.begin()) |
|  | { |
|  | Serial.println("Couldn't find RTC Module"); |
|  | while (1); |
|  | } |
|  |  |
|  | if (rtc.lostPower()) |
|  | { |
|  | Serial.println("RTC lost power, lets set the time!"); |
|  | rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_))); |
|  | } |
|  | rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_))); |
|  |  |
|  | } |
|  |  |
|  | void loop () |
|  | { |
|  | if(path) |
|  | { |
|  | if(a==1) |
|  | { |
|  | lcd.setCursor(0,0); |
|  | lcd.print(" RTC "); |
|  | lcd.setCursor(0,1); |
|  | lcd.print(" MODE ON "); |
|  | delay(2000); |
|  | a=0; |
|  | } |
|  | now = rtc.now(); |
|  | showTime(); |
|  | c\_hour=now.hour(); |
|  | c\_minute=now.minute(); |
|  | c\_second=now.second(); |
|  | loadHandler( on\_hour, on\_minute, on\_second, off\_hour, off\_minute, off\_second, c\_hour, c\_minute, c\_second); |
|  | delay(1000); |
|  | } |
|  | else |
|  | { |
|  | if(a==0) |
|  | { |
|  | lcd.setCursor(0,0); |
|  | lcd.print(" LDR "); |
|  | lcd.setCursor(0,1); |
|  | lcd.print(" MODE ON "); |
|  | delay(2000); |
|  | a=1; |
|  | } |
|  | val = analogRead(nob); |
|  | if(val>300 && val<450) |
|  | { |
|  | lcd.setCursor(0,0); |
|  | lcd.print(" 30% "); |
|  | lcd.setCursor(0,1); |
|  | lcd.print(" Brightness "); |
|  | analogWrite(led, 400); |
|  | } |
|  | else if(val>450 && val<550) |
|  | { |
|  | lcd.setCursor(0,0); |
|  | lcd.print(" 60% "); |
|  | lcd.setCursor(0,1); |
|  | lcd.print(" Brightness "); |
|  | analogWrite(led, 600); |
|  | } |
|  | else if(val>550 && val<600) |
|  | { |
|  | lcd.setCursor(0,0); |
|  | lcd.print(" 100% ");; |
|  | lcd.setCursor(0,1); |
|  | lcd.print(" Brightness "); |
|  | analogWrite(led, 1023); |
|  | } |
|  | else if(val<300) |
|  | { |
|  | lcd.setCursor(0,0); |
|  | lcd.print(" 0% "); |
|  | lcd.setCursor(0,1); |
|  | lcd.print(" Brightness "); |
|  | analogWrite(led, 0); |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | void showTime() |
|  | { |
|  | lcd.setCursor(0,0); |
|  | lcd.print(" Time:"); |
|  | lcd.print(now.hour()); |
|  | lcd.print(':'); |
|  | lcd.print(now.minute()); |
|  | lcd.print(':'); |
|  | lcd.print(now.second()); |
|  | lcd.print(" "); |
|  | } |
|  |  |
|  |  |
|  | void button\_ISR() |
|  | { |
|  | buttonFlag = 1; |
|  | if((millis() - previousPress) > buttonDebounce && buttonFlag) |
|  | { |
|  | previousPress = millis(); |
|  | if(digitalRead(buttonPin) == LOW && previousState == HIGH) |
|  | { |
|  | path =! path; |
|  | previousState = LOW; |
|  | } |
|  |  |
|  | else if(digitalRead(buttonPin) == HIGH && previousState == LOW) |
|  | { |
|  | previousState = HIGH; |
|  | } |
|  | buttonFlag = 0; |
|  | } |
|  | } |
|  |  |
|  | unsigned char checkLessThanOrEqual(userTime\_t a, userTime\_t b) |
|  | { |
|  | if(a.temp\_hour < b.temp\_hour) |
|  | return true; |
|  | Else |
|  | { |
|  | if ((a.temp\_hour == b.temp\_hour) && (a.temp\_minute < b.temp\_minute)) |
|  | { |
|  | return true; |
|  | } |
|  | else |
|  | { |
|  | if(a.temp\_hour > b.temp\_hour) |
|  | return false; |
|  | else |
|  | { |
|  | if((a.temp\_minute == b.temp\_minute) && (a.temp\_second < b.temp\_second)) |
|  | { |
|  | return true; |
|  | } |
|  | else |
|  | { |
|  | if(a.temp\_minute > b.temp\_minute) |
|  | return false; |
|  | else |
|  | { |
|  | if(a.temp\_second == b.temp\_second) |
|  | { |
|  | return true; |
|  | } |
|  | else |
|  | { |
|  | return false; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void loadHandler(int onTimeHr, int onTimeMin, int onTimeSec, int offTimeHr, int offTimeMin, int offTimeSec, int rtcTimeHr, int rtcTimeMin, int rtcTimeSec) |
|  | { |
|  |  |
|  | userTime\_t in1 = {onTimeHr, onTimeMin, onTimeSec}, in2 = {offTimeHr, offTimeMin, offTimeSec}, rtc\_hr = {rtcTimeHr, rtcTimeMin, rtcTimeSec}, a = {}, b = {}; |
|  |  |
|  | if(checkLessThanOrEqual(in1, in2)) |
|  | { |
|  | onOrOffFlag = ON; |
|  | memcpy(&a, &in1, sizeof(userTime\_t)); |
|  | memcpy(&b, &in2, sizeof(userTime\_t)); |
|  | } |
|  | else |
|  | { |
|  | onOrOffFlag = OFF; |
|  | memcpy(&a, &in2, sizeof(userTime\_t)); |
|  | memcpy(&b, &in1, sizeof(userTime\_t)); |
|  |  |
|  | } |
|  |  |
|  | if((checkLessThanOrEqual(a, rtc\_hr)) && (checkLessThanOrEqual(rtc\_hr, b))) |
|  | { |
|  | if(onOrOffFlag == ON) |
|  | { |
|  | // Switch on the load |
|  | digitalWrite(led,HIGH); |
|  | lcd.setCursor(0,1); |
|  | lcd.print("OffTime:"); |
|  | lcd.print(off\_hour); |
|  | lcd.print(':'); |
|  | lcd.print(off\_minute); |
|  | lcd.print(':'); |
|  | lcd.print(off\_second); |
|  |  |
|  |  |
|  | } |
|  | Else |
|  | { |
|  | // Switch off the load |
|  | digitalWrite(led,LOW); |
|  | lcd.setCursor(0,1); |
|  | lcd.print(" OnTime:"); |
|  | lcd.print(on\_hour); |
|  | lcd.print(':'); |
|  | lcd.print(on\_minute); |
|  | lcd.print(':'); |
|  | lcd.print(on\_second); |
|  |  |
|  | } |
|  | } |
|  | else |
|  | { |
|  | if(onOrOffFlag == ON) |
|  | { |
|  | // Switch off the load |
|  | digitalWrite(led,LOW); |
|  | lcd.setCursor(0,1); |
|  | lcd.print(" OnTime:"); |
|  | lcd.print(on\_hour); |
|  | lcd.print(':'); |
|  | lcd.print(on\_minute); |
|  | lcd.print(':'); |
|  | lcd.print(on\_second); |
|  | } |
|  | else |
|  | { |
|  | // Switch on the load |
|  | digitalWrite(led,HIGH); |
|  | lcd.setCursor(0,1); |
|  | lcd.print("OffTime:"); |
|  | lcd.print(off\_hour); |
|  | lcd.print(':'); |
|  | lcd.print(off\_minute); |
|  | lcd.print(':'); |
|  | lcd.print(off\_second); |
|  | } |
|  | } |
|  | } |

CONCLUSION AND ADVANTAGES:

A simple project for saving power is implemented using Auto Intensity Control of Street Lights using Arduino. With slight modifications and enhancements, this project can be applicable for real time use.

Some of the advantages are:

* no need to control manually
* energy saving
* bunch of LEDs reduces the cost
* Lifetime of street lights can be increased
* Simple Design

REFERENCE:

[1 Electronic Devices and Circuit Theory, Boylestad.

[2] Sensors: Advancements in Modeling, Design Issues, Fabrication and Practical by Subhas Chandra

[3 <https://www.elprocus.com/auto-intensity-control-of-street-lights-circuit/>

[4] www.atmel.com,www.beyondlogic.org,www.wikipedia.org,www.howstuffwo rks.com, www.alldatasheets.com, www.wikipedia.com.

[5] Digital Design by Moris Mano (Second Addition)

[6] Hand Book of Electronics by A.K. Maini

[7] <https://en.wikipedia.org/wiki/Intelligent_street_lighting>

[8] Digital Systems Principles and Application by Ronald Ltocci (Sixth Addition).