Use an Arduino to make a dimmer and control the brightness of a lamp.
10

1	330-ohm resistor
2	33k resistors
1	22k resistor
1	220-ohm resistor
4	1N4508 diodes
1	1N4007 diode
1	Zener 10V.4W diode
1	2.2uF/63V capacitor
1	220nF/275V capacitor
1	Arduino
1	Optocoupler: 4N35
1	MOSFET: IRF830A
1	Lamp: 100W
1	230V supply
1	Socket
1	Solder dot board and Soldering kit

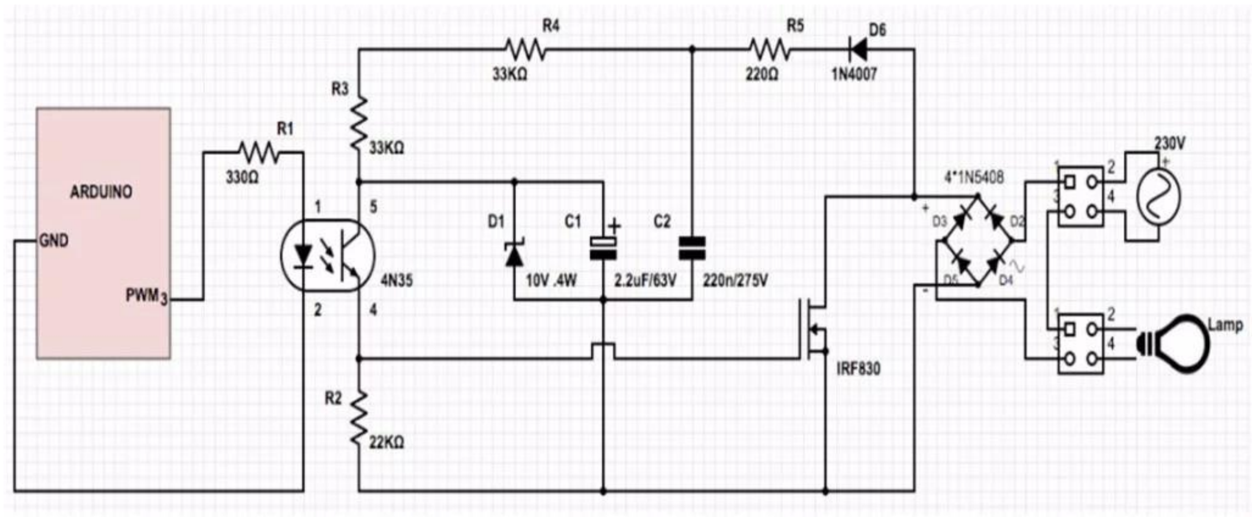
Our Arduino lamp dimmer

Ever wondered how to bring an Arduino board into your daily life? We often adjust the display brightness of our mobile phones to suit to our need. With this project, you can do that for your bedside lamps or any other lighting at home. We are going to teach you how to make an Arduino lamp dimmer. Using this project, you can control the brightness of your table lamp according to your needs and start building one!

How Does the Arduino Lamp Dimmer Work?
In this project, we are going to adjust the brightness of the lamp connected to the circuit by serial port. The brightness can be changed according to the commands we provide to the serial port. We will be using these particular commands in this Arduino project:

0 to TURN OFF
1 for 25% brightness
2 for 50% brightness
3 for 75% brightness
4 for 100% brightness
We will design a Pulse Wave Modulated (PWM) dimmer circuit which will use an IRF830A in a diode bridge which is used to control the voltage across the bulb with pulse wave modulation (PWM). The power supply voltage for driving the gate is supplied with the voltage across the Metal Oxide Semiconductor Field-Effect Transistor (MOSFET).

Circuit for the Arduino Lamp Dimmer



Lamp Dimmer Circuit

The figure above explains the positioning of the different electrical components in the circuit. Follow the circuit diagram to solder your dot board.

Arduino lamp dimmer soldering

You can use this picture as a reference when you solder your board.

To start, we have the diode D6, the load resistor R5, and the capacitor C2 connected. This forms our rectifier. Also, in the circuit, the positioning of the resistor R5 is such that it limits the current pulses through D6 to about 1.5A. This shows that the rectifier is not a pure peak rectifier.

Talking about the capacitor C2, the voltage across it is regulated to a maximum value of 10V by the resistors R3 and R4, capacitor C1, and diode D1.

The optocoupler and the remaining resistor R2 are used for driving the gate.

The function of R1 is to protect the LED in the optocoupler. Like R5, R1 also limits the current so that a 'hard' voltage can be applied safely. The optocoupler, CNY65, provides class-II isolation. (A class-II or double insulated electrical appliance is the one which has been designed in such a way that it does not require a safety connection to electrical earth). This is good enough to ensure safety to the regulator.

Now we need the conduction in the MOSFET as quickly as possible. For this, we connect the transistor in the optocoupler to the positive power supply. We have to make a compromise between the switching loss and the inductive voltages as we keep the rating of R2 a bit high, i.e., 22 K-ohm. This is done to reduce the switching spikes as a consequence of parasitic inductances.

One additional advantage of the MOSFET is that it helps in the conduction. It conducts for a longer duration than a PWM would do single headedly. Some interesting information is that when the voltage across the MOSFET reduces, the voltage across D1 remains equal to 10V upon a duty cycle of 88%.

A duty cycle is the percentage of one period up to which a signal is active. A period is the time it takes to complete an on-and-off cycle. The expression for the duty cycle is:

D = (T/P)*100%

Where:
D is the duty cycle.
T is the time the signal is active.
P is the total period of the signal.

A higher duty cycle results in lowering of the voltage. For an instance, at 94% duty cycle, the voltage of 4.8A proved to be just enough to cause the MOSFET to conduct sufficiently. This value is therefore considered as the maximum duty cycle. Also, at this value, the transistor conducts just about cent percent. Measuring with a 100W bulb, the voltage across a 230V mains supply is just 2.5V lower.

Note: This circuit should not be used to control inductive loads. The MOSFET is switched asynchronously and this can cause the DC current to flow.

Some tips regarding the whole Arduino project. Starting off with, we must know that electronic lamps, such as PL types, cannot be dimmed with this circuit. This is because these lamps use a rectifier and internally, they operate off DC. A few remarks about the value of R3 and R4. This is a compromise between the lowest possible current we consume (when the lamp is off) and the highest possible duty cycle that is allowed. Now as mentioned earlier, the higher the duty cycle, the less is the voltage. So, we look at a case where the duty cycle is 0%. This results in a maximum voltage across the resistors around 128V across a supply of 230V. Because (depending on the actual resistor) the voltage rating of the resistor may be less than 300 V, two resistors are connected in series. The power that each resistor dissipates amounts to a maximum of 0.5 W. With an eye on the life expectancy, it would be wise to use two 1-W rated resistors here.

Uploading the Code to Arduino
Here you can see a few more pictures of the finished product.

Arduino lamp dimmer

Arduino Script

You can upload this code to program your Arduino lamp dimmer:

```
intledPin = 3;
void setup()
{
Serial.begin(9600);
Serial.println(“Serial connection started, waiting for instructions…n0 = Offn1 = 25%n2 =50%n3 =
75%n4 = 100%”);
}

void loop ()
{
if (Serial.available()) {
char ser = Serial.read(); //read serial as a character

//NOTE because the serial is read as “char” and not “int”, the read value must be compared to
character numbers
//hence the quotes around the numbers in the case statement

switch (ser)
{
case ‘0’:
analogWrite(ledPin, 0);
break;

case ‘1’:
analogWrite(ledPin, 64);
break;

case ‘2’:
analogWrite(ledPin, 128);
break;

case ‘3’:
analogWrite(ledPin, 192);
break;

case ‘4’:
analogWrite(ledPin, 255);
break;
default:
Serial.println(“Invalid entry”);

}
}
}
```