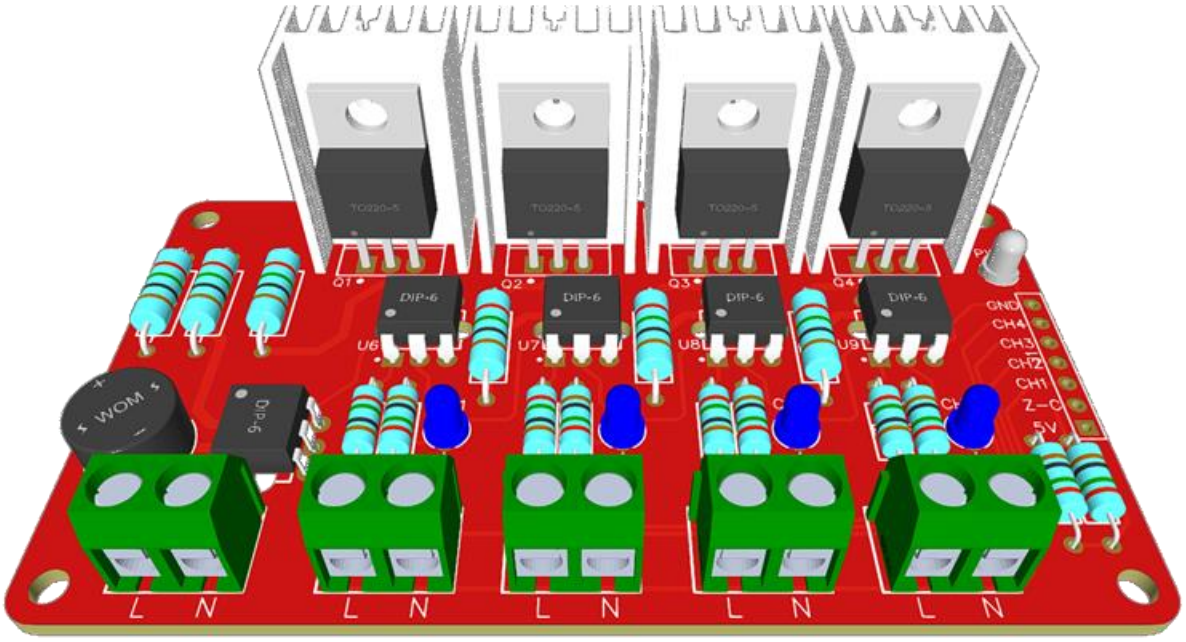


4 Channel AC Dimmer Module

User Manual

Contents

Description

The 4 Channel AC Dimmer Module can control the power of two 220V AC loads using microcontrollers.

The logical level is tolerant to 5V and 3.3V, therefore it can be connected to the microcontroller with 5V and 3.3V level logic.

In Arduino, the dimmer is controlled with RBDdimmer.h library, which uses external interrupts and process time interrupts. It simplifies the code writing and gives more processing time for main code. Which is why you can control multiple Dimmers from one microcontroller.

You can download RBDDimmer.h library and a few examples in «Documents» or on GitHub. We are constantly updating our library, so we recommend to check for the website updates or subscribe to our newsletter.

Dimmer is connected to Arduino controllers via two digital pins. First (Zero) to control the passing of Phase Null of AC, which is used to initiate the interrupt signal. Second (DIM/PWM) to control (dim) current.

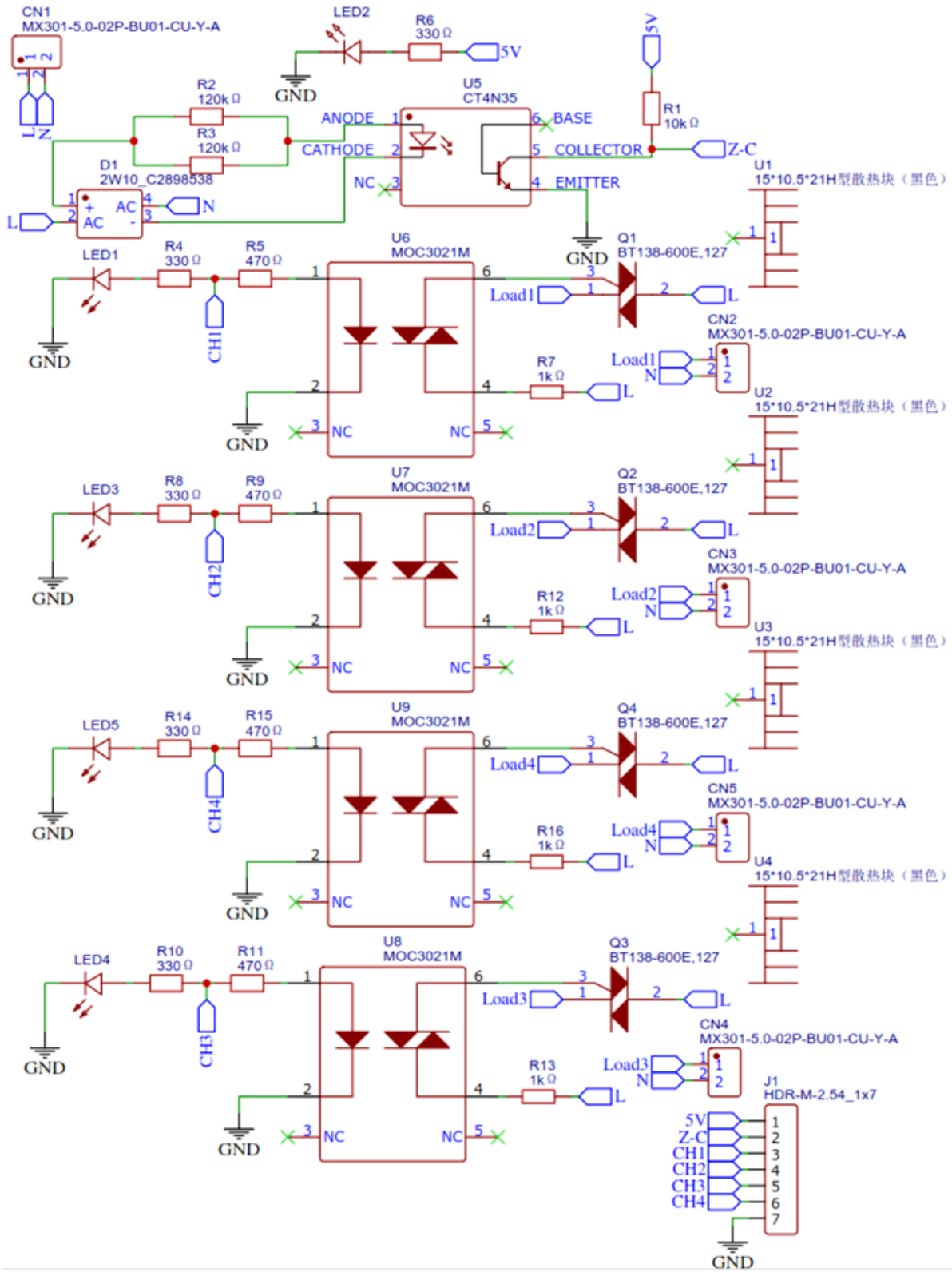
Specifications

- Channels - 4
- Operating Voltage - 5V
- Controlling Mode – Pulse Width Modulation (PWM)
- Load voltage - 230V AC
- Load Current - 5A

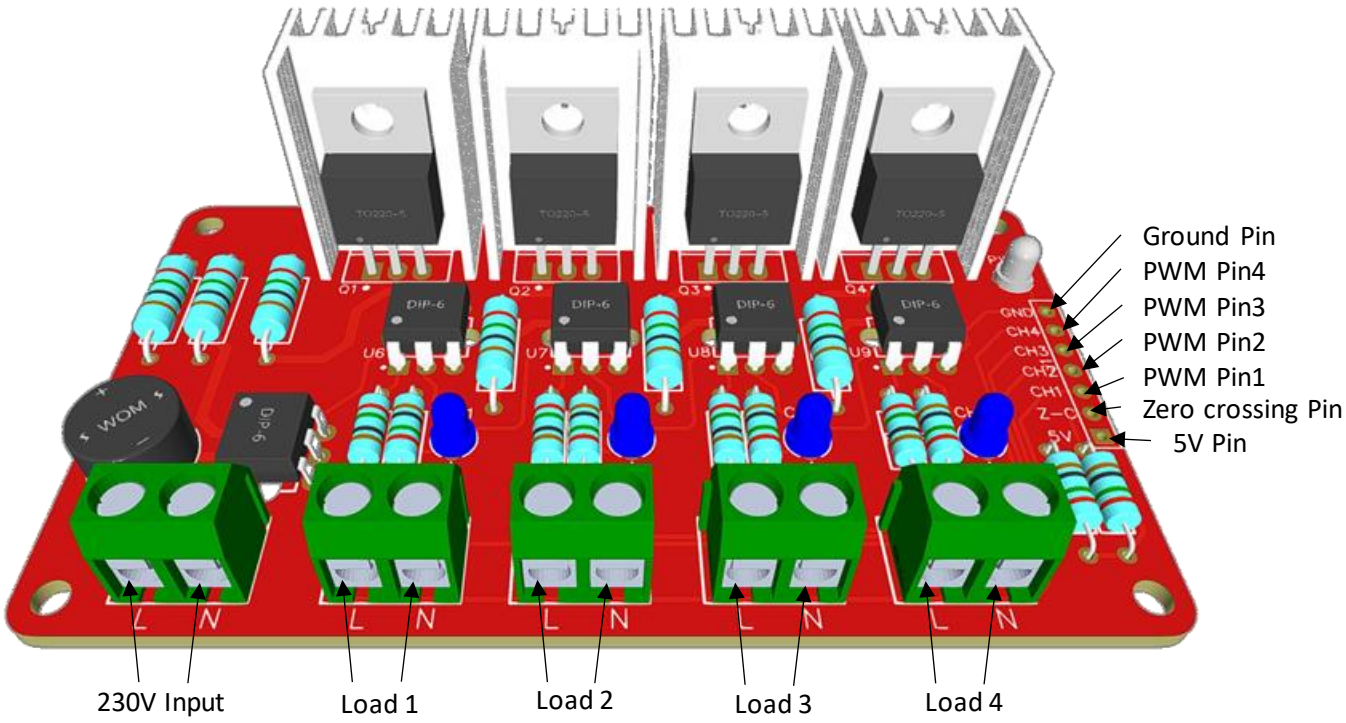
NOTE:

- Use this module with caution to avoid getting electric shock
- Do not handle this module without adult supervision
- Do not touch the heat sink or the bottom of the PCB when the main supply is connected

Circuit Diagram



Pin Configuration



- Supply 5V to the 5V pin and GND pin
- Do not connect the main power supply to the Load side, if not the module will burn
- When using the module find the operating region of the module with the connected load by fine tuning the power percentage from the code

Sample Code

```
/**************
    RobotDyn
    Dimmer Library
 *  **************

   The following sketch is meant to define the dimming value through the serial
port of the controller,
   using USE_SERIAL.begin
      void printSpace() function is used for adding of space after functional
data
      void loop()  serial port evaluator, used to register and define values in
dimmer.setPower(outVal);


    ---------------------- OUTPUT & INPUT Pin table ---------------------
    +---------------+-------------------------+-------------------------+
    |   Board       | INPUT Pin               | OUTPUT Pin              |
    |               | Zero-Cross              |                         |
    +---------------+-------------------------+-------------------------+
    | Lenardo       | D7 (NOT CHANGABLE)      | D0-D6, D8-D13           |
    +---------------+-------------------------+-------------------------+
    | Mega          | D2 (NOT CHANGABLE)      | D0-D1, D3-D70           |
    +---------------+-------------------------+-------------------------+
    | Uno           | D2 (NOT CHANGABLE)      | D0-D1, D3-D20           |
    +---------------+-------------------------+-------------------------+
    | ESP8266       | D1(IO5),    D2(IO4),    | D0(IO16),   D1(IO5),    |
    |               | D5(IO14),   D6(IO12),   | D2(IO4),    D5(IO14),   |
    |               | D7(IO13),   D8(IO15),   | D6(IO12),   D7(IO13),   |
    |               |                         | D8(IO15)                |
    +---------------+-------------------------+-------------------------+
    | ESP32         | 4(GPI36),   6(GPI34),   | 8(GPO32),   9(GP033),   |
    |               | 5(GPI39),   7(GPI35),   | 10(GPIO25), 11(GPIO26), |
    |               | 8(GPO32),   9(GP033),   | 12(GPIO27), 13(GPIO14), |
    |               | 10(GPI025), 11(GPIO26), | 14(GPIO12), 16(GPIO13), |
    |               | 12(GPIO27), 13(GPIO14), | 23(GPIO15), 24(GPIO2),  |
    |               | 14(GPIO12), 16(GPIO13), | 25(GPIO0),  26(GPIO4),  |
    |               | 21(GPIO7),  23(GPIO15), | 27(GPIO16), 28(GPIO17), |
    |               | 24(GPIO2),  25(GPIO0),  | 29(GPIO5),  30(GPIO18), |
    |               | 26(GPIO4),  27(GPIO16), | 31(GPIO19), 33(GPIO21), |
    |               | 28(GPIO17), 29(GPIO5),  | 34(GPIO3),  35(GPIO1),  |
    |               | 30(GPIO18), 31(GPIO19), | 36(GPIO22), 37(GPIO23), |
    |               | 33(GPIO21), 35(GPIO1),  |                         |
    |               | 36(GPIO22), 37(GPIO23), |                         |
    +---------------+-------------------------+-------------------------+
```

```
    | Arduino M0    | D7 (NOT CHANGABLE)      | D0-D6, D8-D13           |
    | Arduino Zero  |                         |                         |
    +---------------+-------------------------+-------------------------+
    | Arduino Due   | D0-D53                  | D0-D53                  |
    +---------------+-------------------------+-------------------------+
    | STM32         | PA0-PA15,PB0-PB15       | PA0-PA15,PB0-PB15       |
    | Black Pill    | PC13-PC15               | PC13-PC15               |
    | BluePill      |                         |                         |
    | Etc...        |                         |                         |
    +---------------+-------------------------+-------------------------+
*/

#include <RBDdimmer.h>//

//#define USE_SERIAL  SerialUSB //Serial for boards whith USB serial port
#define USE_SERIAL  Serial
#define outputPin1  3
#define outputPin2  4
#define outputPin3  5
#define outputPin4  6
//#define zerocross  5 // for boards with CHANGEBLE input pins

//dimmerLamp dimmer(outputPin, zerocross); //initialase port for dimmer for
ESP8266, ESP32, Arduino due boards
dimmerLamp dimmer1(outputPin1); //initialase port for dimmer for MEGA, Leonardo,
UNO, Arduino M0, Arduino Zero
dimmerLamp dimmer2(outputPin2); //initialase port for dimmer for MEGA, Leonardo,
UNO, Arduino M0, Arduino Zero
dimmerLamp dimmer3(outputPin3); //initialase port for dimmer for MEGA, Leonardo,
UNO, Arduino M0, Arduino Zero
dimmerLamp dimmer4(outputPin4); //initialase port for dimmer for MEGA, Leonardo,
UNO, Arduino M0, Arduino Zero


int outVal = 0;

void setup() {
  USE_SERIAL.begin(9600);
  dimmer1.begin(NORMAL_MODE, ON); //dimmer initialisation: name.begin(MODE,
STATE)
  dimmer2.begin(NORMAL_MODE, ON); //dimmer initialisation: name.begin(MODE,
STATE)
  dimmer3.begin(NORMAL_MODE, ON); //dimmer initialisation: name.begin(MODE,
STATE)
  dimmer4.begin(NORMAL_MODE, ON); //dimmer initialisation: name.begin(MODE,
STATE)
  USE_SERIAL.println("Dimmer Program is starting...");
```

```
  USE_SERIAL.println("Set value");
}

void printSpace(int val)
{
  if ((val / 100) == 0) USE_SERIAL.print(" ");
  if ((val / 10) == 0) USE_SERIAL.print(" ");
}

void loop() {
  int preVal = outVal;

  if (USE_SERIAL.available())
  {
    int buf = USE_SERIAL.parseInt();
    if (buf != 0) outVal = buf;
    delay(200);
  }
  dimmer1.setPower(outVal); // setPower(0-100%);
  dimmer2.setPower(outVal); // setPower(0-100%);

  if (preVal != outVal)
  {
    USE_SERIAL.print("lampValue 1 -> ");
    printSpace(dimmer1.getPower());
    USE_SERIAL.print(dimmer1.getPower());
    USE_SERIAL.println("%");

    USE_SERIAL.print("lampValue 2 -> ");
    printSpace(dimmer2.getPower());
    USE_SERIAL.print(dimmer2.getPower());
    USE_SERIAL.println("%");

    USE_SERIAL.print("lampValue 3 -> ");
    printSpace(dimmer3.getPower());
    USE_SERIAL.print(dimmer3.getPower());
    USE_SERIAL.println("%");

    USE_SERIAL.print("lampValue 4 -> ");
    printSpace(dimmer4.getPower());
    USE_SERIAL.print(dimmer4.getPower());
    USE_SERIAL.println("%");

  }
  delay(50);

}
```

Note: Above code is a sample and it will dim all four loads in the same value user can change the code as per the requirement. Also, check the examples that come with the library.

Doc Version: 1.0.0
Product Version: 1.0.0
Report issues to: info@optimus.lk