Shenzhen Doctors of Intelligence & Technology (SZDOIT)

Arduino UNO-Compatible Motor Shield User Manual

DC motor-driven shield

Oct. 24, 2014

User Guidance

1 Introduce

UNO-Compatible Motor Shield can be compatible fully with Arduino UNO with a strong current motor-driven module, which is designed as overlapping insertion model. So, it can be inserted the Arduino UNO board directly. It has 6 ports including speed, direction, brake, and control.

The core chip is adopted by the professional motor-driven L298P chip from Stmicroelectronics company. It can drive 2-way DC motor or 1-way stepper motor. The driven current can be up to 2A. This motor-shield module is designed nationally with the protection by the high-speed Schottky diodes for the output.

By using the infrared sensor HX1838 on the module, motors can be driven by the remote control on using the IR sensor.

Especially, this module has richful outside interfaces, such as power, IIC, UART, AIO, DIO, and WiFi. So, the module is named as shield and/or motor-driven board.

The power-switch can avoid the frequent insertion for the humanized design, and provide two interfaces for each motor, which can drive 2-way and/or 4-way smart car chassis.

2 Specification

- Input voltage: 7V～12V
- Logic current Iss: ≤36mA（V_i=L）, ≤12mA（V_i=H）
- Driven current Io: ≤2A
- Max dissipation power: 25W（T=75℃）
- Two driven-ways: PWMPLL
- Control signal input voltage: high level: 2.3V≤Vin≤5V; low level: -0.3V≤Vin≤1.5V
- Working temperature: -25℃～＋85℃
- Driven model: Dual power and H bridge driver
- Arduino control ports: 3, 9, 12(A motor); 8, 11, 13(B motor)
- Weight: about 26g

3 Size

The size can be shown in the following Figure 1.

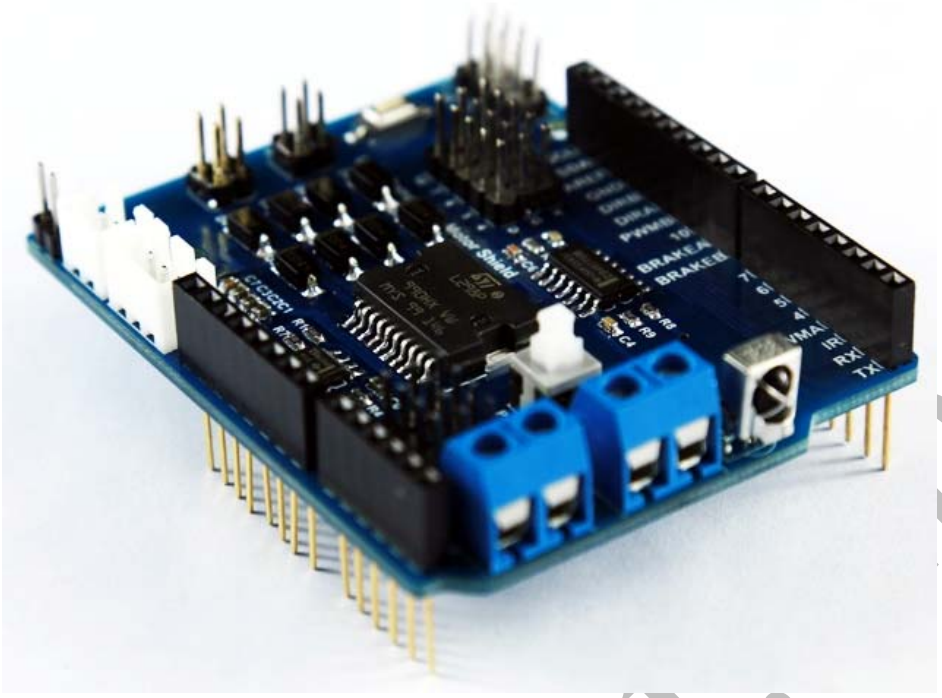

Figure 1 motor shield module

Size can be shown in the following figure.

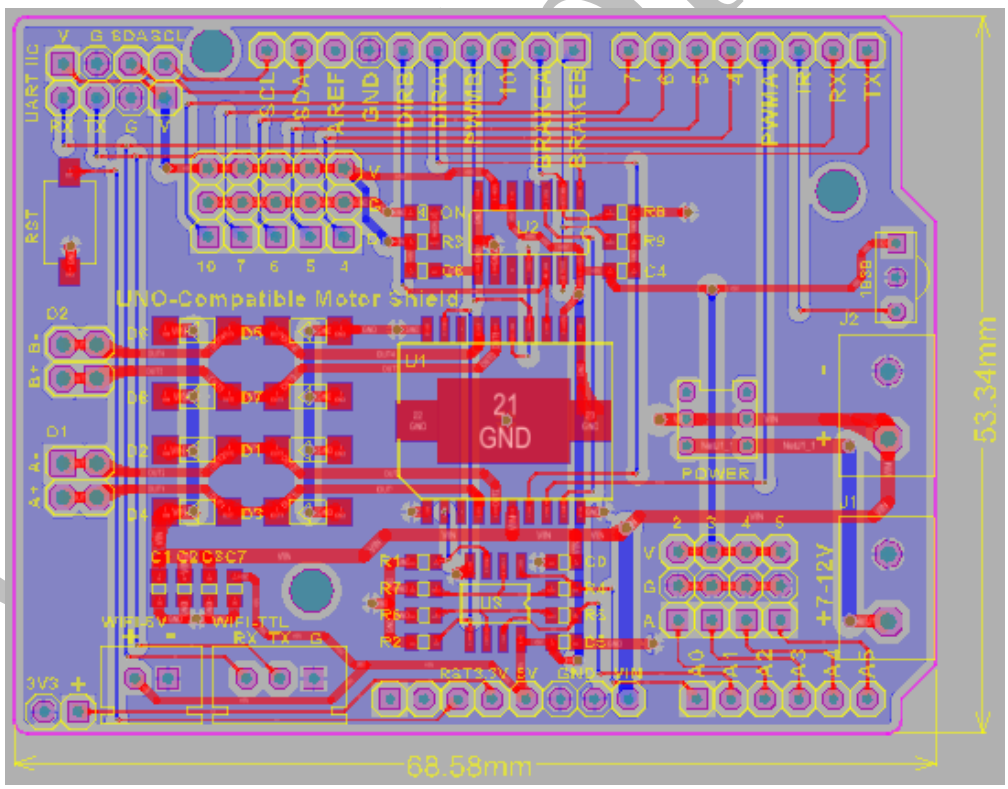

Figure 2 size of module

4 Interface

The interfaces on the motor shield module is defined as in the Table 1.

Table 1 interface definition

Item	Name	UNO interface	Function	Direction	Notation
UNO	TX	D0	Serial output（D0）	Output	connect UNOinput directly
	RX	D1	Serial input（D1）	Input	Connect UNOoutput directly
	IR	D2	IR input	Input	UNO interrupt
	PWMA	D3	Motor A adjust speed	input	Use PWM to adjust the speed of motor
	4～7,10	D4～D7,10	Digital IO	input/output	Connect UNO directly
	BRAKEB	D8	Motor B brake	input	Enable with high level
	BRAKEA	D9	Motor A brake	input	Enable with high level
	PWMB	D11	Motor B adjust speed	input	Use PWM to adjust the speed of motor
	DIRA	D12	Motor A direction	input	Adjust rotation of motor
	DIRB	D13	Motor B direction	input	Adjust rotation of motor
	AREF	AREF	extended	-	-
	SDA	SDA	IIC data	input/output	Connect analog IO A4
	SCL	SCL	IIC clock	input/output	Connect analog IO A5
	RST	RST	reset	input	Enable with lowe level
	3.3V	3.3V	Power output	output	150MA, enable with UNO
	5V	5V	Power output	output	Enable with UNO
	VIN	VIN	Power input	input	Connect power input directly
	A0	A0	Motor A sample	output	Compatible with analog inputoutput
	A1	A1	Motor B sample	output	Compatible with analog inputoutput
	A2～A5	A2～A5	Analog IO	output/input	Analog inputoutput
public	V	-	Power 5V	-	-
	G	-	GND	-	-
	A	-	Analog IO		-
	D	-	Digital IO		-
power	J1	-	Power input	J1, input	7-12V
	J2	-	Power check	J2, input	Connect voltage measure module
	3V3	-	Power output	output	3.3Voutput

	WiFi-5V	-	Power output	output	WiFi power
output	O1	-	A+， A-	output	A+, A- for motor A
	O2	-	B+， B-	output	B+, B- for motor B
other	WiFi TTL	-	RX， TX， G	Communication	Reserved WiFi interface
	POWER	-	switch	-	Power switch

5 Fast guidance

6 Revised Record

Table 2 revised record

version	Revised scope	date
1.00	DrALt Version	2014-11-26

7 Technology support

If know more, please visit our sites.

http://www.doit.am

contact:

skype: yichone

Email: **service@smartarduino.com**

Arduino code

```
#include <avr/wdt.h>
/*
Function pins per Ch. A pins per Ch. B
Direction       D12           D13
PWM             D3            D11
Brake           D9            D8
Current Sensing A0             A1
    pinMode(dir1, OUTPUT);
    pinMode(dir2, OUTPUT);
    analogWrite(pwm1, value);   //PWM Speed Control
*/
//left side motor
int pwm1 = 3;
int dir1 = 12;
int brk1 = 9;
//right side motor
int pwm2 = 11;
int dir2 = 13;
int brk2 = 8;
int pbIn = 0;                   // define pin 2 as interrupt, i.e, D2
volatile int state = LOW;       // define the default input

enum{
    MOTOR_STOP=0,
    MOTOR_FWD,
    MOTOR_BWD
};
enum{
    MOTOR_LEFT=0,
    MOTOR_RIGHT
};
int PWMCurrentValL=0;//the current PWM value for the left motor
int PWMTargetValL=0;//the target PWM value for the left motor
int PWMCurrentValR=0;//the current PWM value for the right motor
int PWMTargetValR=0;//the target PWM value for the right motor
unsigned int PWMCurrentSpd = 127;//the current speed
unsigned long MS_LastMotorControlTime=0;//control time for the last time
unsigned long MS_TIMER=0;//system time tick
unsigned long lastMSTimer=0;//system time tick
unsigned long lastMSTimer2=0;//system time tick

long currentSpeed = 0;
```

```
unsigned char sendCurrentSpeedFlag=0;
unsigned long lastMSTimer3=0;//system time tick

String comdata = "";//uart recieve data

void SetMotorDirL(unsigned char mode)
{//set the direction for the left motor
    if(mode == MOTOR_STOP)
    {
        digitalWrite(brk1,HIGH);
            analogWrite(pwm1,0);
    }
    else if(mode == MOTOR_BWD)
    {
            digitalWrite(dir1,LOW);
    }
    else if(mode == MOTOR_FWD)
    {
           digitalWrite(dir1,HIGH);
    }
}
void SetMotorDirR(unsigned char mode)
{//set the direction for the right motor
    if(mode == MOTOR_STOP)
    {
        digitalWrite(brk2,HIGH);
            analogWrite(pwm2,0);
    }
    else if(mode == MOTOR_BWD)
    {
            digitalWrite(dir2,LOW);
    }
    else if(mode == MOTOR_FWD)
    {
            digitalWrite(dir2,HIGH);
    }
}
//left motor, assign the target speed, and speed up the speed to the target
//by the symbol of speed to confirm the direction of motor
//dead zone value
#define DeadZoneVal 10

void MotorControl(unsigned char motorType,int targetVal,int *currentVal)
{//motorType: left or right motor
```

```
int accVal = 20;//acceleration

if(targetVal>255) targetVal = 255;
if(targetVal < -255) targetVal = -255;

if(targetVal - (*currentVal)> 0)
{
    (*currentVal) = (*currentVal) + accVal;
}
else if(targetVal - (*currentVal) < 0)
{
    (*currentVal) = (*currentVal) - accVal;
}

if(fabs(targetVal - (*currentVal) )<=DeadZoneVal)
{
    (*currentVal) = targetVal;
}
if(*currentVal>255) *currentVal = 255;
if(*currentVal < -255) *currentVal = -255;

if((*currentVal)> DeadZoneVal)
{//正向
    if(motorType==MOTOR_LEFT)
        SetMotorDirL(MOTOR_FWD);
    else if(motorType==MOTOR_RIGHT)
        SetMotorDirR(MOTOR_FWD);
}
else if((*currentVal) < -DeadZoneVal)
{//反向
    if(motorType==MOTOR_LEFT)
        SetMotorDirL(MOTOR_BWD);
    else if(motorType==MOTOR_RIGHT)
        SetMotorDirR(MOTOR_BWD);
}
else
{//停止
    //currentVal = 0;
    if(motorType==MOTOR_LEFT)
    {
        SetMotorDirL(MOTOR_STOP);
    }
    else if(motorType==MOTOR_RIGHT)
    {
```

```
            SetMotorDirR(MOTOR_STOP);
        }
        return;
    }
    if(motorType==MOTOR_LEFT)
    {
        if((*currentVal)>=0)
        {
            analogWrite(pwm1,(unsigned char)(*currentVal));
        }
        else
        {
            analogWrite(pwm1,(unsigned char)(-(*currentVal)));
        }
    }
    else if(motorType==MOTOR_RIGHT)
    {
        if((*currentVal)>=0)
        {
            analogWrite(pwm2,(unsigned char)(*currentVal));
        }
        else
        {
            analogWrite(pwm2,(unsigned char)(-(*currentVal)));
        }
    }
}
////////////////////////////////////////////////////////////////////////
//////////
void DataProcess(void)
{
  //控制电机
    if(comdata.compareTo("stop")==0)
    {
        PWMTargetValL = 0;
        PWMTargetValR = 0;
    }
    else if(comdata.compareTo("fwd")==0)
    {
        PWMTargetValL = PWMCurrentSpd;
        PWMTargetValR = PWMCurrentSpd;
        MS_LastMotorControlTime = MS_TIMER;
    }
    else if(comdata.compareTo("bwd")==0)
```

```
{
    PWMTargetValL = -PWMCurrentSpd;
    PWMTargetValR = -PWMCurrentSpd;
    MS_LastMotorControlTime = MS_TIMER;
}
else if(comdata.compareTo("leftfwd")==0)
{
    //PWMTargetValL = PWMCurrentSpd-200;
    PWMTargetValL = -PWMCurrentSpd;
    PWMTargetValR = PWMCurrentSpd;
    MS_LastMotorControlTime = MS_TIMER;
}
else if(comdata.compareTo("rightfwd")==0)
{
    PWMTargetValL = PWMCurrentSpd;
    PWMTargetValR = -PWMCurrentSpd;
    MS_LastMotorControlTime = MS_TIMER;
}else if(comdata.compareTo("leftbwd")==0)
{
    PWMTargetValL = PWMCurrentSpd;
    PWMTargetValR = -PWMCurrentSpd;
    MS_LastMotorControlTime = MS_TIMER;
}
else if(comdata.compareTo("rightbwd")==0)
{
    PWMTargetValL = -PWMCurrentSpd;
    PWMTargetValR = PWMCurrentSpd;
    MS_LastMotorControlTime = MS_TIMER;
}
else if(comdata.compareTo("speedup")==0)
{
    PWMCurrentSpd = PWMCurrentSpd+10;
    if(PWMCurrentSpd>=255) PWMCurrentSpd=255;
    //Serial.println(" CurrentSpeed:%d",PWMCurrentSpd);
            Serial.print("CurrentSpeed:");
            Serial.println(PWMCurrentSpd);
}
else if(comdata.compareTo("speeddw")==0)
{
    PWMCurrentSpd = PWMCurrentSpd-10;
    if(PWMCurrentSpd<=10) PWMCurrentSpd=10;
            Serial.print("CurrentSpeed:");
            Serial.println(PWMCurrentSpd);
}
```

```
}
unsigned long UartLastRecieveTimeTick=0;
void uartTick()
{
  if (Serial.available() > 0)
    {
        comdata += char(Serial.read());
        UartLastRecieveTimeTick = MS_TIMER;
        delay(5);
    }
    if (comdata.length() > 0&& MS_TIMER - UartLastRecieveTimeTick > 100)
    {
        Serial.println(comdata);//process data
        DataProcess();
        comdata = "";
    }
}
//int cnt=0;
// the setup routine runs once when you press reset:
void setup() {
    pinMode(dir1, OUTPUT);
    pinMode(dir2, OUTPUT);
  // initialize serial communication at 9600 bits per second:
    Serial.begin(9600);
    MS_TIMER=millis(); //get the run time for the current system
    MS_LastMotorControlTime = 0;
    //watch dog
    wdt_enable(WDTO_1S);

    Serial.println("SystemStart");
  // monitor the change of the interrupt pin
  attachInterrupt(pbIn, stateChange, RISING);//FALLING  CHANGE LOW
}

// the loop routine runs over and over again forever:
void loop() {

  MS_TIMER=millis();
  wdt_reset();//feed the watch dog

  if(MS_TIMER > 4294960000)
    {
      while(1);//wait for restart
    }
```

```
  if(MS_TIMER - lastMSTimer>10)
  {
     lastMSTimer = MS_TIMER;
     // Serial.println(cnt++);//process data

     uartTick();

    if(MS_TIMER - MS_LastMotorControlTime > 500)
    {//停止电机
      PWMTargetValL = 0;
      PWMTargetValR = 0;
    }

    MotorControl(MOTOR_LEFT,PWMTargetValL,&PWMCurrentValL);
    MotorControl(MOTOR_RIGHT,PWMTargetValR,&PWMCurrentValR);
  }
  if(MS_TIMER - lastMSTimer2>300&&sendCurrentSpeedFlag==1)
   {
     lastMSTimer2 = MS_TIMER;
     sendCurrentSpeedFlag = 0;
     Serial.println("");
     Serial.print("CurrentSpeed:");
     Serial.print(currentSpeed);
     Serial.println(" rpm");
   }
}

long currentSpeedTemp = 0;
void stateChange()
{
  //currentSpeedTemp = (micros() - lastMSTimer3)/1000;
  //currentSpeed = 60000/currentSpeedTemp;
  currentSpeedTemp = (micros() - lastMSTimer3)/1000;
  currentSpeed = 60000/currentSpeedTemp;
  lastMSTimer3 = micros();
  sendCurrentSpeedFlag = 1;
}
```