# DIY Car
# Build Guide using TB67H450FNG

# Reference Guide

## RD204-RGUIDE-01-E

Toshiba Electronic Devices & Storage Corporation



2020-09-14
Rev. 1.0

# Contents

# 1. Outline

This reference guide describes a reference design for operating “DIY Car” using Toshiba’s motor driver IC TB67H450FNG and Arduino Nano microcontroller board. TB67H450FNG has a wide operating voltage range (4.5V to 44V), low power consumption, and a popular pin-assignment, which helps in easy evaluation and development.

In the application example described in this guide, TB67H450FNG is used for the driving left and right wheels via DC motors. While Arduino Nano is used to generate control signals for TB67H450FNG. This guide provides an overview of the hardware and software used in this design.

# 2. Abbreviation

MCD : Motor Control Driver IC
MCU : Microcontroller Unit
BOM : Bill of Materials
DIY : Do It Yourself
EVB : Evaluation Board
RM : Reference model

# 3. About DIY Car

DIY Car is a reference model based on Toshiba’s general motor driver IC TB67H450FNG. TB67H450FNG allows the motor to be controlled in various ways which further allows the DIY car to move in various patters.

Figure 3-1 shows the DIY car build based on TB67H450FNG.

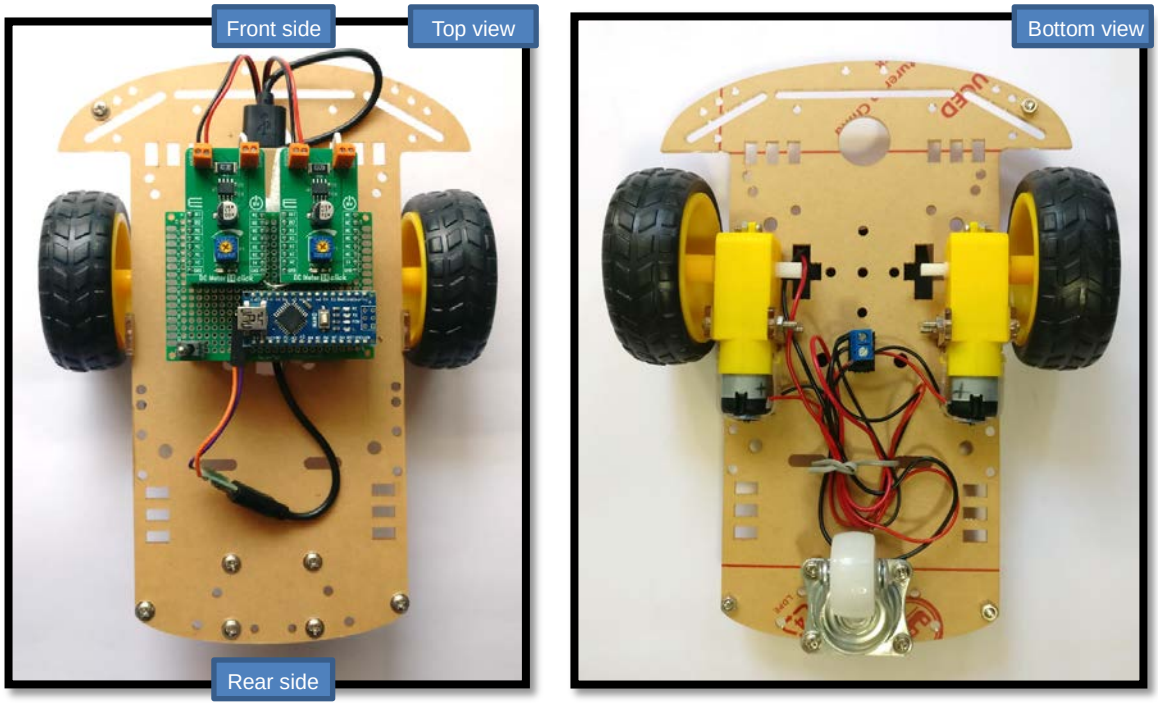


**Figure 3-1 DIY Car (with MCD) (Left：Top view, Right: Bottom view)**

# 4. Hardware Configuration

## 4.1. BOM

DIY Car is built using following components.

- Base Car Model (Chassis + Motors + Tires) (Refer section 9.1)
- Arduino Nano (Microcontroller Board)
- TB67H450FNG EVB (MIKROE-3982) (Refer to Section 5)
- Li-ion battery (5V)
- Connection Cables
- Switch

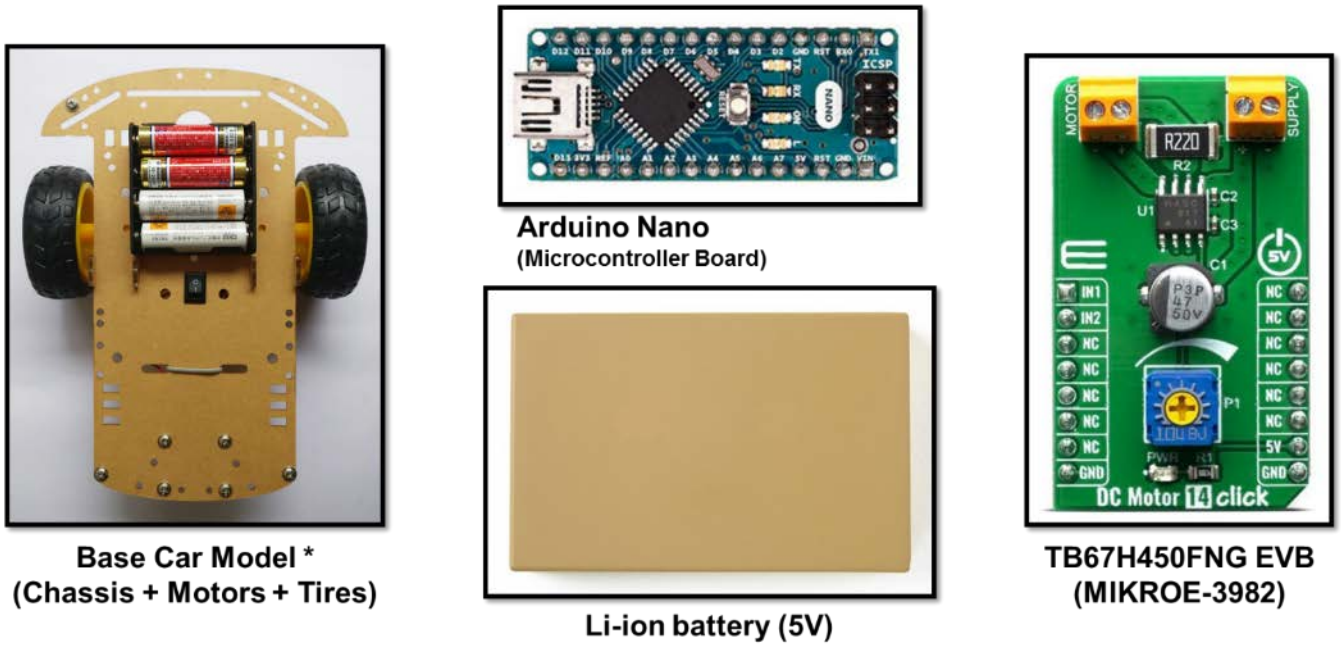


**Figure 4-1 Main Components**

* In “DIY Car”, Li-ion battery is being used instead of 4x AA batteries which comes with “Base Car Model”.

Figure 4-2 shows the DIY Car built using above components.

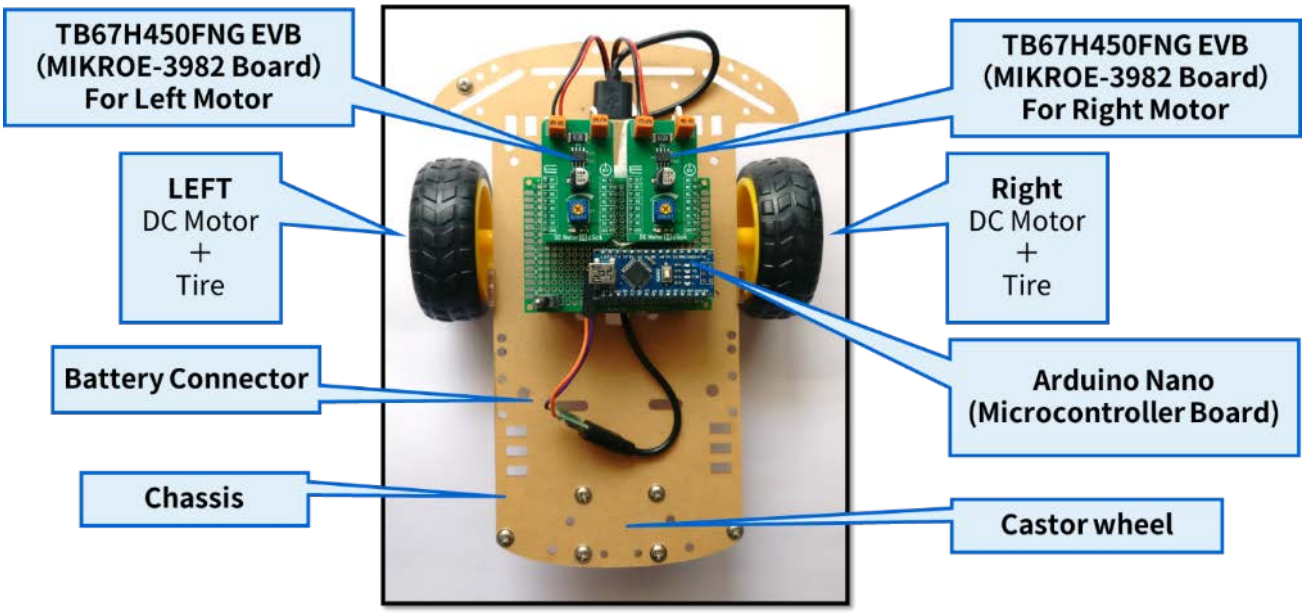


**Figure 4-2 DIY Car - Hardware Configuration**

## 4.2. Block Diagram

DIY Car's block diagram is shown in Figure 4-3. Whole system works on 5V Li-ion battery. DIY Car has 2 tires driven by 2 brushed DC motors. And these motors are driven by TB67H450FNG which are controlled using PWM signals generated by Arduino Nano microcontroller board. So overall 4 PWM signals are required.

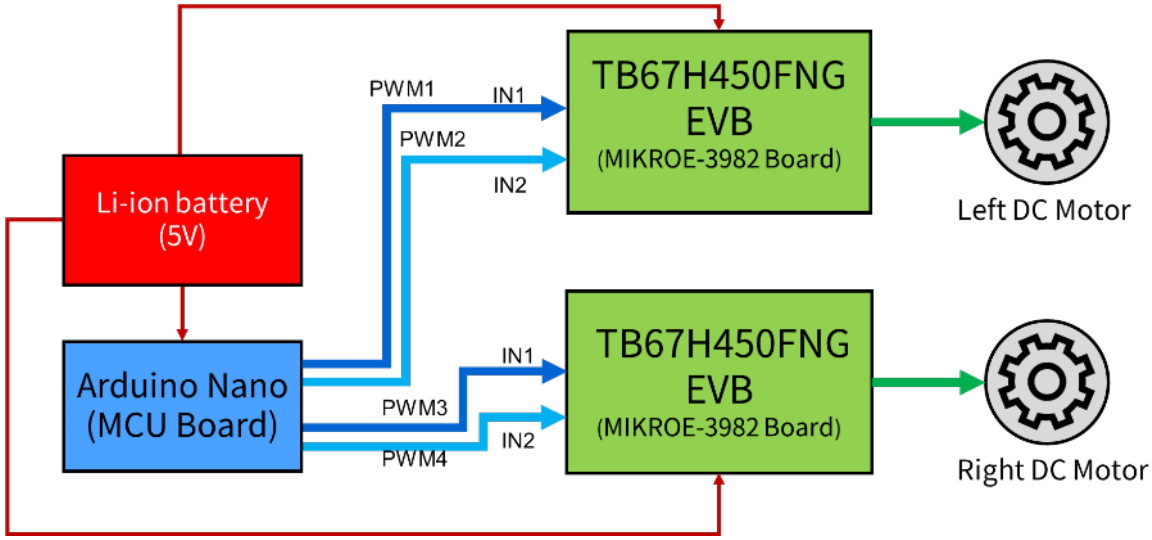


**Figure 4-3 DIY Car - Block Diagram**

## 4.3. Hardware Connection

Connection between all components of DIY Car is shown in Figure 4-4. Li-ion battery (5V) and GND are connected to Arduino Nano, both TB67H450FNG EVBs and both Motors via VM connector.

2 input pins of a motor are connected to 2 output pins of the TB67H450FNG EVB. The connection of Right side motor pins is opposite to that of left side motor pins. This has been done so that both motors take DIY car forward when forward signal is given to both TB67H450FNG EVBs.

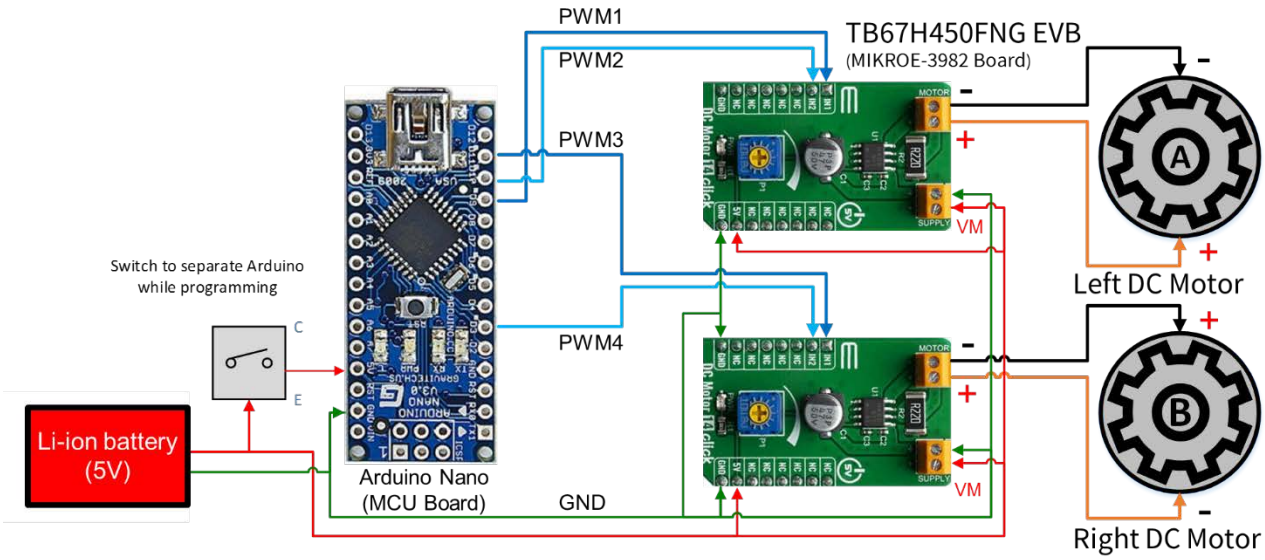


**Figure 4-4 Hardware Connection**

Circuit diagrams of TB67H450FNG EVB and Arduino Nano can be checked on their respective websites.

TB67H450FNG EVB (Mikroe-3982) Link: https://www.mikroe.com/dc-motor-14-click
Arduino Nano Link: https://store.arduino.cc/usa/arduino-nano

# 5. TB67H450FNG Operation

MIKROE-3982 is Toshiba's TB67H450FNG motor driver's evaluation board (EVB).

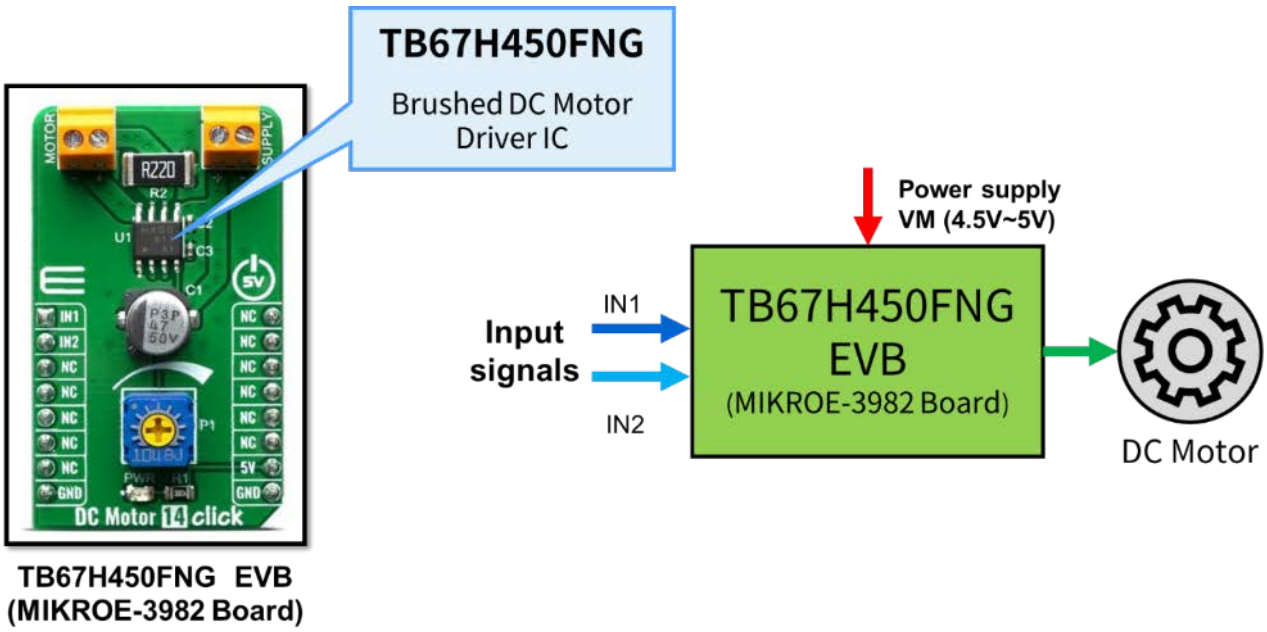


**Figure 5-1 TB67H450FNG EVB**

TB67H450FNG supports 4 operation modes for brushed DC motor, these are shown in Table 1. Motor operation mode is selected according to the configuration of 2 input signals IN1 and IN2. Speed control of motor can also be performed by sending PWM signals to IN1 and IN2.

| IN1 | IN2 | OUT1 | OUT2 | Mode |
|---|---|---|---|---|
| L | L | OFF (Hi-Z) | OFF (Hi-Z) | Stop |
| | | | | Standby mode after 1 ms |
| H | L | H | L | Forward |
| L | H | L | H | Reverse |
| H | H | L | L | Brake |

**Table 1 TB67H450FNG – DC Motor Control Modes**

Kindly refer to following links for more information.

TB67H450FNG EVB（MIKROE-3982） Link**:** https://www.mikroe.com/dc-motor-14-click
TB67H450FNG Datasheet Link**:** https://toshiba.semicon-storage.com/ap-en/semiconductor/product/motor-driver-ics/brushed-dc-motor-driver-ics/detail.TB67H450FNG.html

# 6. Software Configuration

## 6.1. Software Development Environment

- Arduino IDE (1.8.10)
- Windows 10 PC

Arduino website Link: https://www.arduino.cc

## 6.2. Two Motor Operation

Eight Operation modes for DIY car are created by configuring various operation modes for 2 motors are shown in following table and Figure 6-1.

| DIY Car Operation Mode | Left Motor Operation Mode | Right Motor Operation Mode |
|---|---|---|
| Forward | Forward rotation | Forward rotation |
| Reverse | Reverse rotation | Reverse rotation |
| Brake | Brake | Brake |
| Standby | Standby | Standby |
| Turn Right | Forward rotation | Reverse rotation |
| Turn Left | Reverse rotation | Forward rotation |
| Circle Right | Forward rotation (Fast) | Forward rotation (Slow) |
| Circle Left | Forward rotation (Slow) | Forward rotation (Fast) |

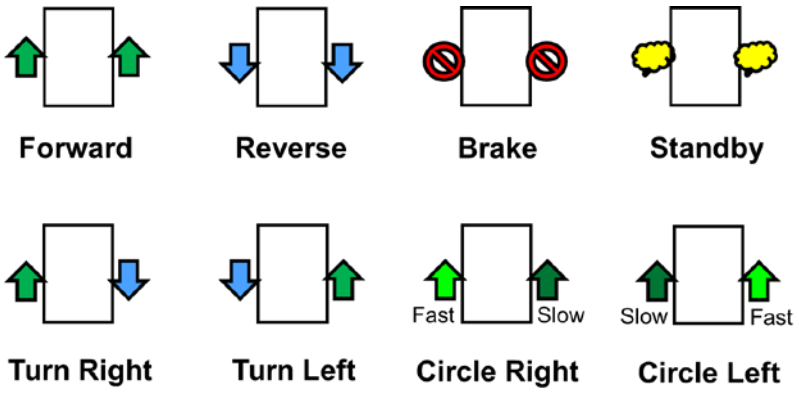


**Figure 6-1 DIY Car – Operation Modes**

### 6.3. Software Function Hierarchy

In order to move DIY car in various operation modes as described in Section 6.2, following software function hierarchy has been created.

- **Level 1:** Functions to generate PWM by configuring Hardware (Timer) settings.
- **Level 2:** Functions to control a single motor by using the hardware configuration functions of Level 1.
- **Level 3:** Functions to control DIY Car by controlling two motors via functions of Level 2.

DIY car can be moved in many complex patterns by using the combination of DIY control functions of Level 3.

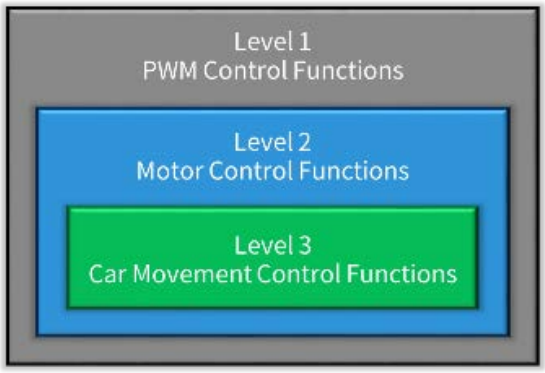


**Figure 6-2 Software Function Hierarchy (Block Diagram)**

```
//Function Definitions
//Car movement functions (Level 3)
void car_forward(int spd); // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]    spd(%):0-100
void car_reverse(int spd); // Move Car Reverse[Left_Motor-Bw, R_Motor-Bw]    spd(%):0-100
void car_brake();          // BRAKE Car movement[Left_Motor-BRAKE, R_Motor-BRAKE]
void car_standby();        // STANDBY Car motors [Left_Motor-Standby, R_Motor-Standby]
void car_turn_left(int deg);      // Turn Car Left[Left_Motor-Rw, R_Motor-Fw]     deg: rotaion angle
void car_turn_right(int deg);     // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]    deg: rotaion angle
void car_turn_circle(int L_spd, int R_spd); //Turn car in circle L_spd & R_spd are speed of left & R wheels
//Motor control functions (Level 2)
void L_forward(int spd); // Move Left motor forward[LIN1-H, LIN2-L]    spd(%):0-100
void L_reverse(int spd); // Move Left motor reverse[LIN1-L, LIN2-H]    spd(%):0-100
void L_brake();          // Brake Left motor [LIN1-H, LIN2-H]
void L_standby();        // Standby Left motor [LIN1-L, LIN2-L]
void R_forward(int spd); // Move Right motor forward[RIN1-H, RIN2-L]    spd(%):0-100
void R_reverse(int spd); // Move Right motor reverse[RIN1-L, RIN2-H]    spd(%):0-100
void R_brake();          // Brake Right motor [RIN1-H, RIN2-H]
void R_standby();        // Standby Right motor [RIN1-L, RIN2-L]
//Timer, pin functions (Level 1)
void timer_out_enDi(int pin,int enable); //pin: 9,10,11,3     enable: en:Output_Enable, di:Output_Disable
void pin_duty_set(int pin, int duty); //pin: 9,10,11,3      duty(%):0-100
```

Speed setting for Car

Turn angle setting for Car

Rotation speed setting for Motor

**Figure 6-3 Hierarchy of all Functions used**

Refer to section 9.2 for sample code.

#### 6.3.1. PWM Signal Control Functions (Level 1)

PWM [1-4] signals (shown in Figure 4-3) for TB67H450FNG are generated by using hardware timers (Timer 1 & 2) of ATmega328(*) microcontroller available on Arduino Nano board.

(*) For more information on Arduino Nano and ATmega328 microcontroller kindly refer to following links.

Arduino Nano: https://store.arduino.cc/usa/arduino-nano
ATmega328 microcontroller: https://www.microchip.com/wwwproducts/en/ATmega328

Following two functions are used to generate PWM signals:

- timer_out_enDi(pin, enable) : This function is used to Enable/Disable PWM on individual pins.
- pin_duty_set(pin, duty) : This function is used to set Duty cycle for PWM signals on individual pins.

### 6.3.2. Motor Control Functions (Level 2)

TB67H450FNG supports following four operation modes (Refer to Section 5):

- Forward (Rotation speed can be controlled)
- Reverse (Rotation speed can be controlled)
- Brake
- Standby

Following are Level 2 functions which are used to control individual motor operation.

Functions to operate individual motor are as follows: (also shown in Figure 6-4)

- **forward(x)** : Generate signals for TB67H450FNG to rotate the motor in forward direction. Input “x” can is used to set motor rotation speed.
- **reverse(x)** : Generate signals for TB67H450FNG to rotate the motor in reverse direction. Input “x” can is used to set motor rotation speed.
- **brake()** : Generate signals for TB67H450FNG to apply brake to the motor.
- **standby()** : Generate signals for TB67H450FNG to apply Hi-Z across motor terminals.


**Figure 6-4 Functions to control each motor individually**

### 6.3.3. DIY Car Control Functions (Level 3)

Operating left and right motors together, seven of the following DIY car control functions are created (Level 3). Each of these functions utilizes Level 2 functions to control individual motors.

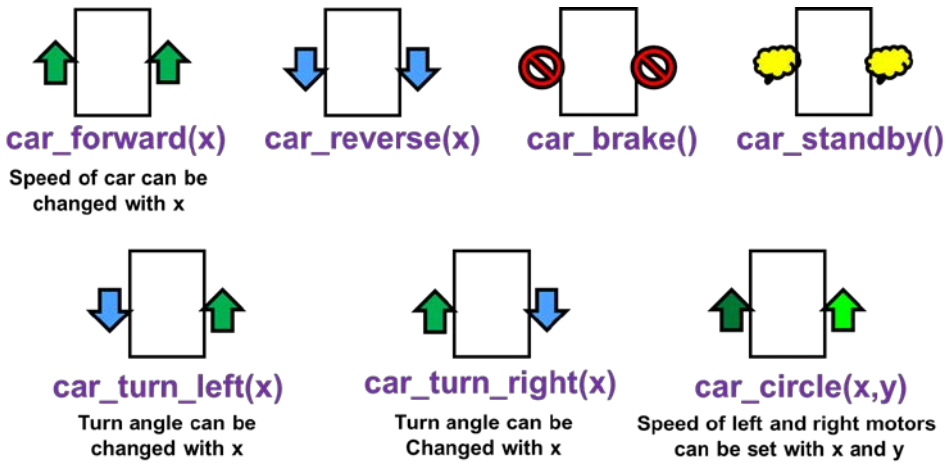


**Figure 6-5 DIY Car Control Functions**

# 7. DIY Car Operation

Various car movement patterns as shown in Figure 7-1 can be created by combining Level 3 Car control functions described in section 6.3.3.

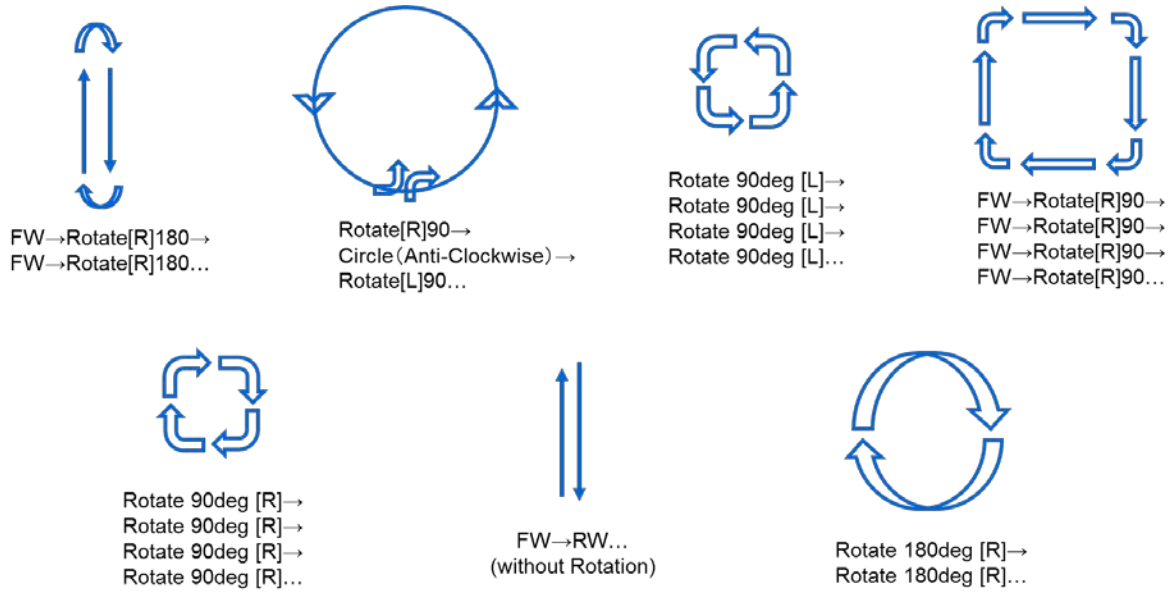


**Figure 7-1 DIY Car Operation Patter created using Motor Driver IC**

# 8. Advantages of using TB67H450FNG

Four operation modes of TB67H450FNG allows the DC motor to be operated in various patterns. Merits of using TB67H450FNG are described in following table.

| Merits of using TB67H450FNG | Base Car model (*) |
|---|---|
| ● Both Forward & Reverse operation can be performed with single battery configuration<br>● Brake Operation can be performed<br><br>● Speed regulation is possible<br>● Various complex car movements are possible:<br>➢ Forward, Reverse, Turn Left/Right, etc | ● Only one direction operation can be performed with single battery configuration<br>● Brake cannot be applied, however power supply can be disconnected so that the car stops slowly because of friction<br>● Speed regulation is not possible<br>● Complex car movements are not possible |

* For more information on Base Car Model, kindly refer to section 9.1.

# 9. Appendix

## 9.1. Base Car Model Description

Base Car Model consist of following components:

- Rubber Tires 2
- Castor Wheel 1
- DC Motor 2
- Chassis 1
- Switch 1
- Battery box for AA cells 1

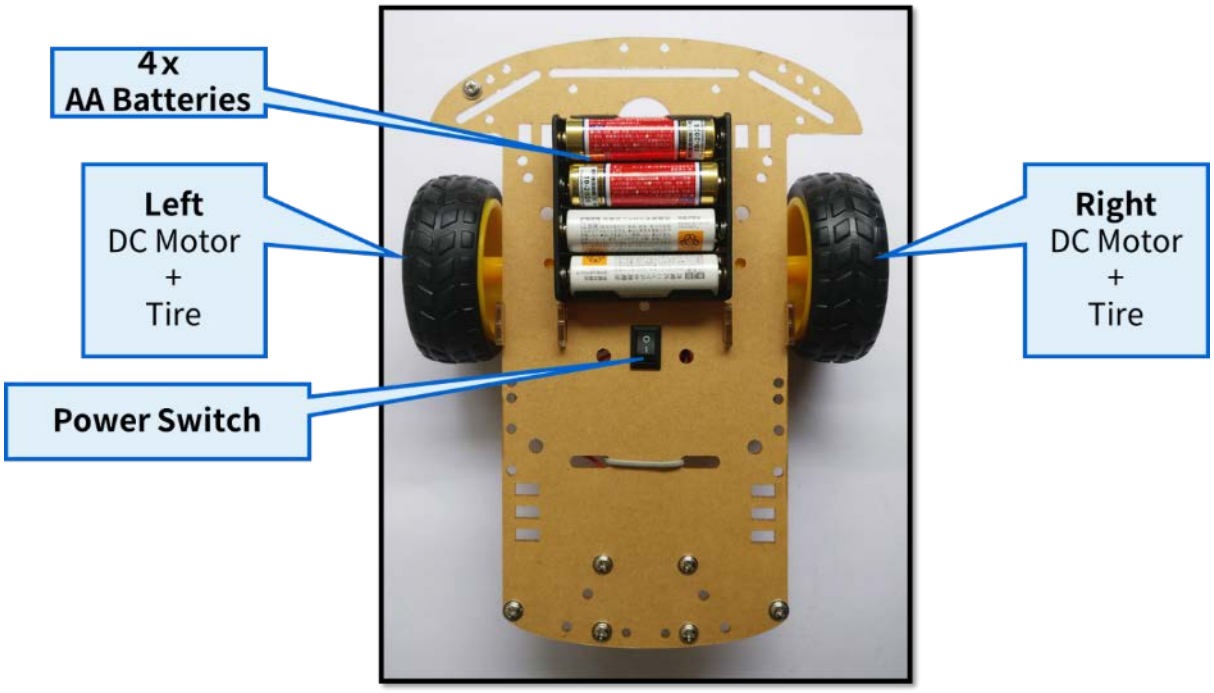


**Figure 9-1 Base Car Model**

Figure 9-2 shows the block diagram of Base Car Model.

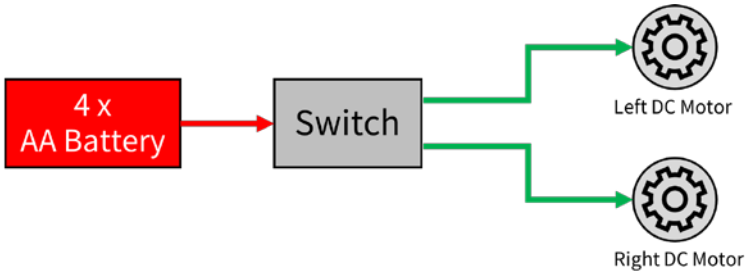


**Figure 9-2 Base Car Block Diagram**

In Base Car, power supply can be turned ON/OFF using the switch.
Car operation according to switch condition is as follows:

Switch ON: Base car moves forward.
Switch OFF: Base car stops slowly because of friction

## 9.2. Software Code

Software code for DIY Car written in Arduino IDE is as follows.

### 9.2.1. Initial Definitions

```
#define pinLin1 9  // Left motor MCD IN1
#define pinLin2 10 // Left motor MCD IN2
#define pinRin1 11 // Right motor MCD IN1
#define pinRin2 3  // Right motor MCD IN2
#define en 1
#define di 0

//Initial third wheel status (during power up)
char third_wheel='B'; //Back 'B', Front 'F', Left 'L', Right 'R'  //Status of which side the third wheel is
protruding.

//Function Definitions
//Car movement functions (Level 3)
void car_forward(int spd); // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
void car_reverse(int spd); // Move Car Reverse[Left_Motor-Bw, R_Motor-Bw]   spd(%):0-100
void car_brake();         // BRAKE Car movement[Left_Motor-BRAKE, R_Motor-BRAKE]
void car_standby();       // STANDBY Car motors [Left_Motor-Standby, R_Motor-Standby]
void car_turn_left(int deg);     // Turn Car Left[Left_Motor-Rw, R_Motor-Fw]    deg: rotaion angle
void car_turn_right(int deg);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotaion angle
void car_turn_circle(int L_spd, int R_spd); //Turn car in circle L_spd & R_spd are speed of left & R wheels
//Motor control functions (Level 2)
void L_forward(int spd); // Move Left motor forward[LIN1-H, LIN2-L]   spd(%):0-100
void L_reverse(int spd); // Move Left motor reverse[LIN1-L, LIN2-H]   spd(%):0-100
void L_brake();         // Brake Left motor [LIN1-H, LIN2-H]
void L_standby();       // Standby Left motor [LIN1-L, LIN2-L]
void R_forward(int spd); // Move Right motor forward[RIN1-H, RIN2-L]   spd(%):0-100
void R_reverse(int spd); // Move Right motor reverse[RIN1-L, RIN2-H]   spd(%):0-100
void R_brake();         // Brake Right motor [RIN1-H, RIN2-H]
void R_standby();       // Standby Right motor [RIN1-L, RIN2-L]
//Timer, pin functions (Level 1)
void timer_out_enDi(int pin,int enable); //pin: 9,10,11,3    enable: en:Output_Enable, di:Output_Disable
void pin_duty_set(int pin, int duty); //pin: 9,10,11,3     duty(%):0-100
```

### 9.2.2. Arduino Setup Function

```
////////////////////////////////////////////////////////////////////////////////////////////
// the setup function runs once when you press reset or power the board
////////////////////////////////////////////////////////////////////////////////////////////
void setup() {
  Serial.begin(9600);

  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(pinLin1, OUTPUT);
  pinMode(pinLin2, OUTPUT);
  pinMode(pinRin1, OUTPUT);
  pinMode(pinRin2, OUTPUT);

  digitalWrite(pinLin1, LOW);
  digitalWrite(pinLin2, LOW);
  digitalWrite(pinRin1, LOW);
  digitalWrite(pinRin2, LOW);


  //TIMER 1 (8bit, fixed top value-255)
  TCCR1A = _BV(WGM10) ; //Fast PWM Mode
  TCCR1B = _BV(WGM12)  | _BV(CS12);           //Prescaler 1/256 (For changing frequency) // 244Hz
  //TCCR1B = _BV(WGM12)  | _BV(CS11) | _BV(CS10); //Prescaler 1/64 (For changing frequency) // 1kHz
  //TCCR1B = _BV(WGM12)  | _BV(CS11);           //Prescaler 1/8 (For changing frequency) // 7.8kHz
  //TCCR1B = _BV(WGM12)  | _BV(CS10);           //Prescaler 1/1 (For changing frequency) // 62kHz
  OCR1A = 64; //Counter for pin9
  OCR1B = 128;  //Counter for pin10
  //TIMER 2
  TCCR2A = _BV(WGM21) | _BV(WGM20); //Fast PWM mode
  TCCR2B =  _BV(CS22) | _BV(CS21); //Prescaler 1/256 (For changing frequency) 244// Hz   40%
  //TCCR2B =  _BV(CS22); //Prescaler 1/64 (For changing frequency) // 1kHz   50%
  //TCCR2B =  _BV(CS21); //Prescaler 1/8 (For changing frequency) // 7.8kHz  70%
  //TCCR2B = _BV(CS20);  //Prescaler 1/1 (For changing frequency) // 62kHz
  OCR2A = 140;  //PWM on Pin 11 (Duty OCR2A/255)
  OCR2B = 200;  //PWM on pin 3   (Duty OCR2B/255)

  car_standby();  // STANDBY Car motors [Left_Motor-Standby, R_Motor-Standby]
  if(third_wheel=='B' or third_wheel=='F'){
    digitalWrite(LED_BUILTIN, HIGH);
  }else{
    digitalWrite(LED_BUILTIN, LOW);
  }
  delay(4000);
}
```

### 9.2.3. Arduino Loop Function

```
//////************************************************************************************///////
//                the loop function runs over and over again forever
//////************************************************************************************///////
void loop() {
  ////////////////////////////////////  FW-Rotate[R]180-FW-Rotate[R]180
  delay(2000);
  for(int i=0;i<2;i++){
    //FW
    car_forward(50);  // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
    delay(500);
    car_brake();
    delay(200);
    car_turn_right(180);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
    delay(500);
    //Return
    car_forward(50);  // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
    delay(500);
    car_brake();
    delay(200);
    car_turn_right(180);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
    delay(500);
  }
  ////////////////////////////////////  Rotate 90deg [L]x4
  delay(2000);
  for(int i=0;i<4;i++){
    car_turn_left(90);     // Turn Car Left[Left_Motor-Rw, R_Motor-Fw]    deg: rotaion angle
    delay(500);
  }
  ////////////////////////////////////  Rotate 90deg [R]x4
  delay(2000);
  for(int i=0;i<4;i++){
    car_turn_right(90);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
    delay(500);
  }
  ////////////////////////////////////  FW & RW (without rotatio & without acceleration)
  delay(2000);
  for(int i=0;i<4;i++){
    car_forward(50);  // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
    delay(500);
    car_brake();
    delay(200);
    car_reverse(50); // Move Car Reverse[Left_Motor-Bw, R_Motor-Bw]   spd(%):0-100
    delay(480);
    car_brake();
    delay(200);
  }
  ////////////////////////////////////  Rotate 180deg [R]x2
  delay(2000);
  for(int i=0;i<2;i++){
    car_turn_right(180);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
    delay(500);
  }
  ////////////////////////////////////  normal Square
  delay(2000);
  for(int i=0;i<1;i++){
    //FW (1st side of square)
    car_forward(50);  // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
    delay(700);
    car_brake();
    delay(200);
    for(int i=0;i<3;i++){  //(2nd, 3rd, 4th side of square)
      //Rotate_Right
      car_turn_right(90);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
      delay(200);
      //FW
      car_forward(50);  // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
      delay(700);
      car_brake();
      delay(200);
    }
    //Rotate_Right
    car_turn_right(90);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
    delay(200);
  }
  ////////////////////////////////////  Circle
  delay(2000);
  car_turn_right(90);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
  delay(200);
  car_turn_circle(40,100); //Turn car in circle L_spd & R_spd are speed of left & R wheels
  delay(2900);
  car_brake();
  delay(200);
  car_turn_left(90);     // Turn Car Left[Left_Motor-Rw, R_Motor-Fw]    deg: rotaion angle
  delay(200);
}
```

### 9.2.4. DIY Car Control Functions (Level 3)

```
////////////////////////////////////////////////////////////////////////////////////////
//////////             Car Movement control functions             ////////////////////
////////////////////////////////////////////////////////////////////////////////////////
void car_forward(int spd){ // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
  L_forward(spd); // Move Left motor forward[LIN1-H, LIN2-L]   spd(%):0-100
  R_forward(spd); // Move Right motor forward[RIN1-H, RIN2-L]   spd(%):0-100
  third_wheel='B'; //Back 'B', Front 'F', Left 'L', Right 'R'  //Status of which side the third wheel is
protruding.
  digitalWrite(LED_BUILTIN, HIGH);
}
void car_reverse(int spd){ // Move Car Reverse[Left_Motor-Bw, R_Motor-Bw]   spd(%):0-100
  L_reverse(spd); // Move Left motor reverse[LIN1-L, LIN2-H]   spd(%):0-100
  R_reverse(spd); // Move Right motor reverse[RIN1-L, RIN2-H]   spd(%):0-100
  third_wheel='F'; //Back 'B', Front 'F', Left 'L', Right 'R'  //Status of which side the third wheel is
protruding.
  digitalWrite(LED_BUILTIN, HIGH);
}
void car_brake(){        // BRAKE Car movement[Left_Motor-BRAKE, R_Motor-BRAKE]
  L_brake();         // Brake Left motor [LIN1-H, LIN2-H]
  R_brake();         // Brake Right motor [RIN1-H, RIN2-H]
}
void car_standby(){       // STANDBY Car motors [Left_Motor-Standby, R_Motor-Standby]
  L_standby();       // Standby Left motor [LIN1-L, LIN2-L]
  R_standby();       // Standby Right motor [RIN1-L, RIN2-L]
}
void car_turn_left(int deg){     // Turn Car Left[Left_Motor-Rw, R_Motor-Fw]    deg: rotation angle
  int extra=0;
  //Turn Left
  R_forward(60); // Move Right motor forward[RIN1-H, RIN2-L]   spd(%):0-100
  L_reverse(60); // Move Left motor reverse[LIN1-L, LIN2-H]   spd(%):0-100    //motor speed cant be super slow

    if(third_wheel == 'F' || third_wheel == 'B'){ //90deg different from desired direction (left)
    extra=0;
  } else if (third_wheel == 'R'){  //180deg different from desired direction (left)
    extra=14;
  }

  if(deg<120){
    delay(277*(deg/90.0)+extra);
  }
  else{
    delay(495*(deg/180.0)+extra);
  }

  car_brake();
  third_wheel='L'; //Back 'B', Front 'F', Left 'L', Right 'R'  //Status of which side the third wheel is
protruding.
  digitalWrite(LED_BUILTIN, LOW);
}
void car_turn_right(int deg){    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
  int extra=0;
  //Turn Right
  L_forward(60); // Move Left motor forward[LIN1-H, LIN2-L]   spd(%):0-100
  R_reverse(60); // Move Right motor reverse[RIN1-L, RIN2-H]   spd(%):0-100

  if(third_wheel == 'F' || third_wheel == 'B'){ //90deg different from desired direction (right)
    extra=0;
  } else if (third_wheel == 'L'){  //180deg different from desired direction (right)
    extra=14;
  }

  if(deg<120){
    delay(281*(deg/90.0)+extra);
  }
  else{
    delay(500*(deg/180.0)+extra);
  }
  car_brake();
  third_wheel='R'; //Back 'B', Front 'F', Left 'L', Right 'R'  //Status of which side the third wheel is
protruding.
  digitalWrite(LED_BUILTIN, LOW);
}
void car_turn_circle(int L_spd, int R_spd){ //Turn car in circle L_spd & R_spd are speed of left & R wheels
  L_forward(L_spd); // Move Left motor forward[LIN1-H, LIN2-L]   spd(%):0-100
  R_forward(R_spd); // Move Right motor forward[RIN1-H, RIN2-L]   spd(%):0-100
}
```

### 9.2.5. Motor Control Functions (Level 2)

```
////////////////////////////////////////////////////////////////////////////////////////
//////////                Motor control functions                 ////////////////////
////////////////////////////////////////////////////////////////////////////////////////
void L_forward(int spd){  // Move Left motor forward[LIN1-H, LIN2-L]   spd(%):0-100
  //pin L in1
  timer_out_enDi(pinLin1, en); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  pin_duty_set(pinLin1, spd);    //pin:9,10,11,3     duty(%):0-100
  //pin L in2
  timer_out_enDi(pinLin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin2, LOW);
}
void L_reverse(int spd){ // Move Left motor reverse[LIN1-L, LIN2-H]   spd(%):0-100
  //pin L in1
  timer_out_enDi(pinLin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin1, LOW);
  //pin L in2
  timer_out_enDi(pinLin2, en); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  pin_duty_set(pinLin2, spd);    //pin:9,10,11,3     duty(%):0-100
}
void L_brake(){          // Brake Left motor [LIN1-H, LIN2-H]
  //pin L in1
  timer_out_enDi(pinLin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin1, HIGH);
  //pin L in2
  timer_out_enDi(pinLin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin2, HIGH);
}
void L_standby(){       // Standby Left motor [LIN1-L, LIN2-L]
  //pin L in1
  timer_out_enDi(pinLin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin1, LOW);
  //pin L in2
  timer_out_enDi(pinLin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin2, LOW);
}

void R_forward(int spd){  // Move Right motor forward[RIN1-H, RIN2-L]   spd(%):0-100
  //pin R in1
  timer_out_enDi(pinRin1, en); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  pin_duty_set(pinRin1, spd);    //pin:9,10,11,3     duty(%):0-100
  //pin R in2
  timer_out_enDi(pinRin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin2, LOW);
}
void R_reverse(int spd){ // Move Right motor reverse[RIN1-L, RIN2-H]   spd(%):0-100
  //pin R in1
  timer_out_enDi(pinRin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin1, LOW);
  //pin R in2
  timer_out_enDi(pinRin2, en); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  pin_duty_set(pinRin2, spd);    //pin:9,10,11,3     duty(%):0-100
}
void R_brake(){          // Brake Right motor [RIN1-H, RIN2-H]
  //pin R in1
  timer_out_enDi(pinRin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin1, HIGH);
  //pin R in2
  timer_out_enDi(pinRin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin2, HIGH);
}
void R_standby(){       // Standby Right motor [RIN1-L, RIN2-L]
  //pin R in1
  timer_out_enDi(pinRin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin1, LOW);
  //pin R in2
  timer_out_enDi(pinRin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin2, LOW);

}
```

### 9.2.6. PWM Signal Control Functions (Level 1)

```
////////////////////////////////////////////////////////////////////////////////////////////
//////////                  Timer, pin functions                  ////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
void pin_duty_set(int pin, int duty){ //pin:9,10,11,3     duty(%):0-100
  switch(pin){
    case 9: //Pin9  - OC1A - Timer1
      OCR1A= (duty*255)/100;
      break;
    case 10://Pin10 - OC1B - Timer1
      OCR1B= (duty*255)/100;
      break;
    case 11://Pin11 - OC2A - Timer2
      OCR2A= (duty*255)/100;
      break;
    case 3: //Pin3  - OC2B - Timer2
      OCR2B= (duty*255)/100;
      break;
  }
}

void timer_out_enDi(int pin,int enable){ //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  //Serial.print(pin);
  //Serial.println(enable);
  switch (pin){
    case 9: //Pin9 - OC1A - Timer1
      if(enable==1){
        TCCR1A |=_BV(COM1A1); //Timer output on pin 9 ON
      } else if(enable==0){
        TCCR1A &=~_BV(COM1A1); //Timer output on pin 9 OFF
      }
      break;
    case 10: //Pin10 - OC1B - Timer1
      if(enable==1){
        TCCR1A |=_BV(COM1B1); //Timer output on pin 10 ON
      } else if(enable==0){
        TCCR1A &=~_BV(COM1B1); //Timer output on pin 10 OFF
      }
      break;
    case 11: //Pin11 - OC2A - Timer2
      if(enable==1){
        TCCR2A |=_BV(COM2A1); //Timer output on pin 11 ON
      } else if(enable==0){
        TCCR2A &=~_BV(COM2A1); //Timer output on pin 11 OFF
      }
      break;
    case 3: //Pin3 - OC2B - Timer2
      if(enable==1){
        TCCR2A |=_BV(COM2B1); //Timer output on pin 3 ON
      } else if(enable==0){
        TCCR2A &=~_BV(COM2B1); //Timer output on pin 3 OFF
      }
      break;
    default:
      break;
  }
}
```

# Terms of Use

This terms of use is made between Toshiba Electronic Devices and Storage Corporation (“We”) and customers who use documents and data that are consulted to design electronics applications on which our semiconductor devices are mounted (“this Reference Design”). Customers shall comply with this terms of use. Please note that it is assumed that customers agree to any and all this terms of use if customers download this Reference Design. We may, at its sole and exclusive discretion, change, alter, modify, add, and/or remove any part of this terms of use at any time without any prior notice. We may terminate this terms of use at any time and for any reason. Upon termination of this terms of use, customers shall destroy this Reference Design. In the event of any breach thereof by customers, customers shall destroy this Reference Design, and furnish us a written confirmation to prove such destruction.

1. Restrictions on usage
   1. This Reference Design is provided solely as reference data for designing electronics applications. Customers shall not use this Reference Design for any other purpose, including without limitation, verification of reliability.
   2. This Reference Design is for customer's own use and not for sale, lease or other transfer.
   3. Customers shall not use this Reference Design for evaluation in high or low temperature, high humidity, or high electromagnetic environments.
   4. This Reference Design shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable laws or regulations.

2. Limitations
   1. We reserve the right to make changes to this Reference Design without notice.
   2. This Reference Design should be treated as a reference only. We are not responsible for any incorrect or incomplete data and information.
   3. Semiconductor devices can malfunction or fail. When designing electronics applications by referring to this Reference Design, customers are responsible for complying with safety standards and for providing adequate designs and safeguards for their hardware, software and systems which minimize risk and avoid situations in which a malfunction or failure of semiconductor devices could cause loss of human life, bodily injury or damage to property, including data loss or corruption. Customers must also refer to and comply with the latest versions of all relevant our information, including without limitation, specifications, data sheets and application notes for semiconductor devices, as well as the precautions and conditions set forth in the "Semiconductor Reliability Handbook".
   4. When designing electronics applications by referring to this Reference Design, customers must evaluate the whole system adequately. Customers are solely responsible for all aspects of their own product design or applications. WE ASSUME NO LIABILITY FOR CUSTOMERS' PRODUCT DESIGN OR APPLICATIONS.
   5. No responsibility is assumed by us for any infringement of patents or any other intellectual property rights of third parties that may result from the use of this Reference Design. No license to any intellectual property right is granted by this terms of use, whether express or implied, by estoppel or otherwise.
   6. THIS REFERENCE DESIGN IS PROVIDED "AS IS". WE (a) ASSUME NO LIABILITY WHATSOEVER, INCLUDING WITHOUT LIMITATION, INDIRECT, CONSEQUENTIAL, SPECIAL, OR INCIDENTAL DAMAGES OR LOSS, INCLUDING WITHOUT LIMITATION, LOSS OF PROFITS, LOSS OF OPPORTUNITIES, BUSINESS INTERRUPTION AND LOSS OF DATA, AND (b) DISCLAIM ANY AND ALL EXPRESS OR IMPLIED WARRANTIES AND CONDITIONS RELATED TO THIS REFERENCE DESIGN, INCLUDING WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY OF INFORMATION, OR NONINFRINGEMENT.

3. Export Control
   Customers shall not use or otherwise make available this Reference Design for any military purposes, including without limitation, for the design, development, use, stockpiling or manufacturing of nuclear, chemical, or biological weapons or missile technology products (mass destruction weapons). This Reference Design may be controlled under the applicable export laws and regulations including, without limitation, the Japanese Foreign Exchange and Foreign Trade Law and the U.S. Export Administration Regulations. Export and re-export of this Reference Design are strictly prohibited except in compliance with all applicable export laws and regulations.

4. Governing Laws
   This terms of use shall be governed and construed by laws of Japan.