MOTOR DRIVERS OPERATION AND CODING WITH PWM (FOR DIFFERENT DRIVES)

SET-2

Figure 1. A Mecanum Wheel

By

AAYUSH KUMAR (Electrical)

Omni Wheel

Omni-directional wheels are unique as they are able to roll freely in two directions. It can ether roll like a normal wheel or roll laterally using the wheels along its circumference. Omni-direction wheels allow a robot to convert from a non-holonomic to a holonomic robot. A non-holonomic robot that uses normal wheels has only 2 out of 3 controllable degrees-of-freedom which are, moving forward/backwards and rotation. Not being able to move sideways makes a robot slower and less efficient in reaching its given goal. The holonomic omni-directional wheels are able to overcome this problem, as it a highly maneuverable. Unlike normal non-holonomic robot, the holonomic omni-directional robot can move in an arbitrary direction continuously without changing the direction of the wheels. It can move back and forth, slideways and rotates at the same position.

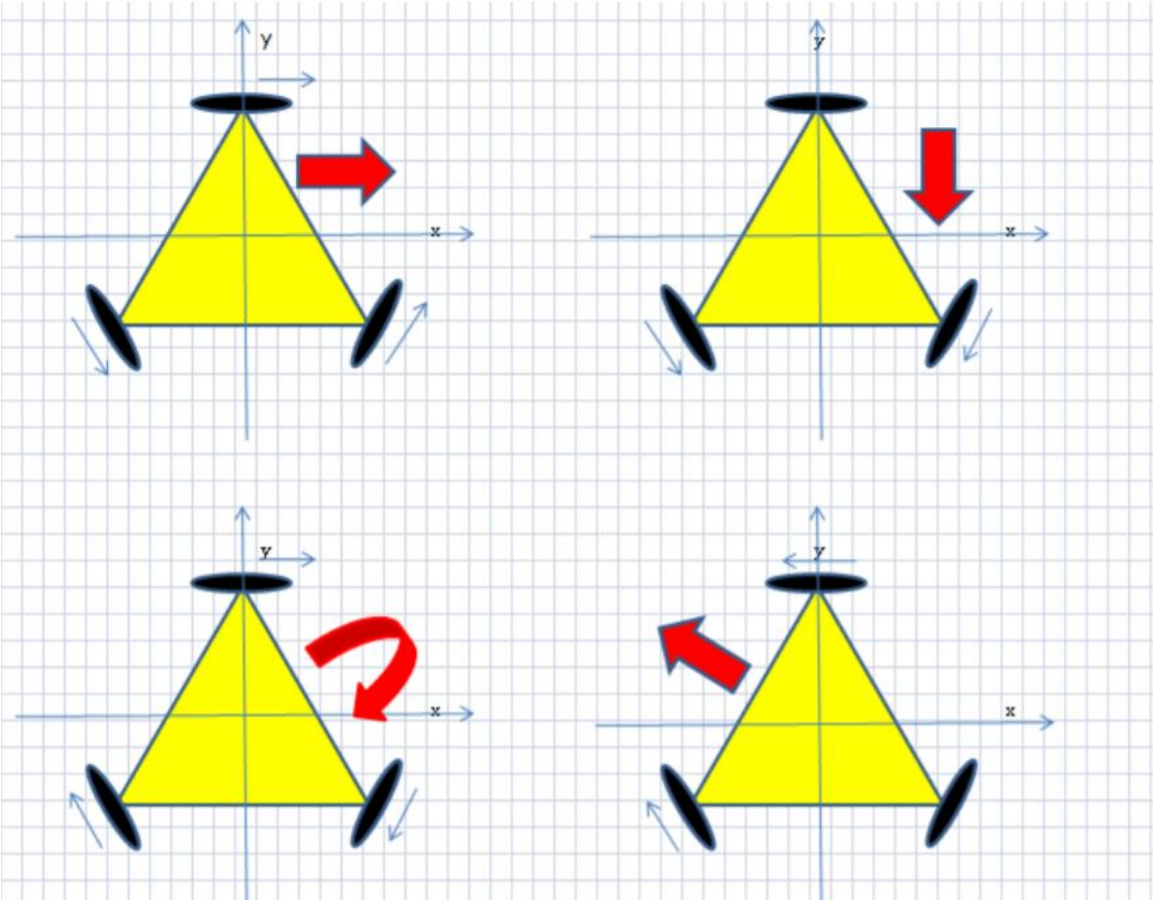

Figure 2. Working principle of 3 Wheels Omni-wheel robot

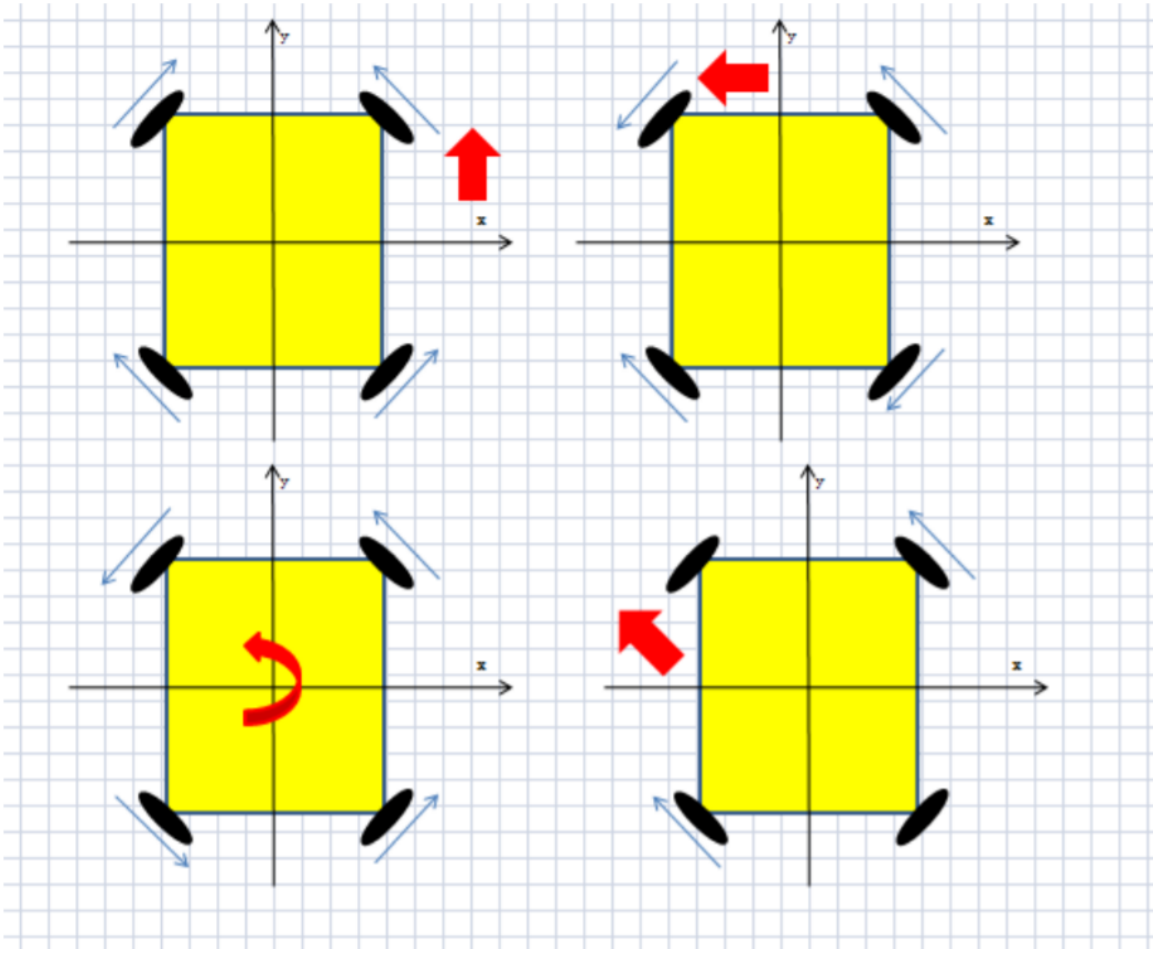

Figure 3. Working principle of 4 wheels Omni-wheel robot

THE CODE

```
void BaseMotor::moveMotor(int lf, int lb, int rf, int rb, int dlf, int drf, int dlb, int drb)
{
  analogWrite(dir_lf, dlf);
  analogWrite(pwm_lf, lf); // Left Forward


  analogWrite(dir_rf, drf);
  analogWrite(pwm_rf, rf); // Right Forward
```

```
  analogWrite(dir_lb, dlb);
  analogWrite(pwm_lb, lb); // Left Backward

  analogWrite(dir_rb, drb);
  analogWrite(pwm_rb, rb); // Right Backward
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void transferMechanism::forward() //This Function will tell the rack to move forward
{ //Used to store the current position where the assembly is currently
  Serial.println("kikikikikikikiki");
  digitalWrite(transferMechanism::rackDir, LOW);
  analogWrite(transferMechanism::rackPwm, 100);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void transferMechanism::backward() //This function will tell the rack to move backward
{
  Serial.println("hihihihihihi");
  analogWrite(transferMechanism::rackDir, 255);
  analogWrite(transferMechanism::rackPwm, 100);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void transferMechanism::halt()
```

```
{
  digitalWrite(transferMechanism::rackDir, LOW);
  analogWrite(transferMechanism::rackPwm, 0);
  Serial.flush();
}
```

Mecanum Wheels

Mecanum drive is a type of holonomic drive base; meaning that it applies the force of the wheel at a 45° angle to the robot instead of on one of its axes. By applying the force at an angle to the robot, you can vary the magnitude of the force vectors to gain translational control of the robot; In plain English, the robot can move in any direction while keeping the front of the robot in a constant compass direction. The figure below shows the motions that can be achieved for various combination of wheel rotation.

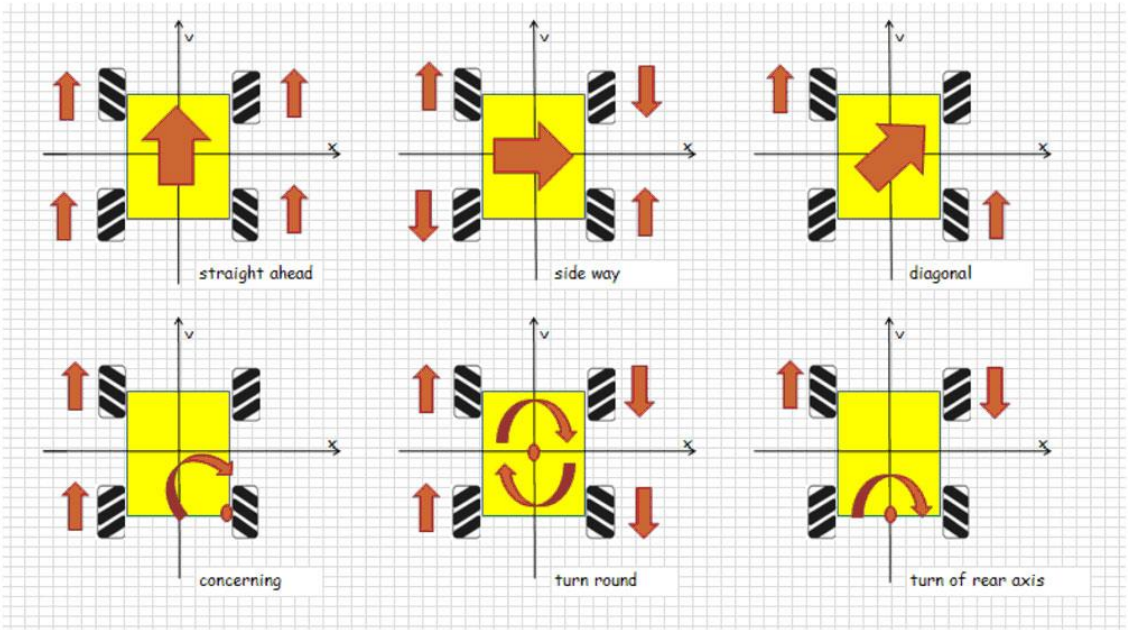

Figure 4. Working principle of 4 wheels Mecanum-wheel robot

THE CODE

```
/*

MECANUM WHEEL ROBOT - BLUETOOTH CONTROLLED v1.0

- Allows you to control a mecanum robot via bluetooth

- Tested with Arduino Mega 2560

- Android application - https://play.google.com/store/apps/details?id=pl.mobilerobots.vacuumcleanerrobot&hl=pl

- Project description - http://www.instructables.com/id/Mecanum-wheel-robot-bluetooth-controlled

Author: Adam Srebro

www: http://www.mobilerobots.pl/

Connections:

Bluetooth (e.g HC-06)-> Arduino Mega 2560

TXD - TX1 (19)

RXD - RX1 (18)
```

VCC - 5V

GND - GND

TB6612FNG Dual Motor Driver -> Arduino Mega 2560

//PWM control

RightFrontMotor_PWMA - 2

LeftFrontMotor_PWMB - 3

RightRearMotor_PWMA - 4

LeftRearMotor_PWMB - 5

//Control of rotation direction

RightFrontMotor_AIN1 - 22

RightFrontMotor_AIN2 - 23

LeftFrontMotor_BIN1 - 24

LeftFrontMotor_BIN2 - 25

RightRearMotor_AIN1 - 26

RightRearMotor_AIN2 - 27

LeftRearMotor_BIN1 - 28

LeftRearMotor_BIN2 - 29

//The module and motors power supply

STBY - Vcc

VMOT - motor voltage (4.5 to 13.5 V) - 11.1V from LiPo battery

Vcc - logic voltage (2.7 to 5.5) - 5V from Arduino

GND - GND

TB6612FNG Dual Motor Driver -> DC Motors

MotorDriver1_AO1 - RightFrontMotor

MotorDriver1_AO2 - RightFrontMotor

MotorDriver1_BO1 - LeftFrontMotor

```
MotorDriver1_B02 - LeftFrontMotor

MotorDriver2_AO1 - RightRearMotor

MotorDriver2_A02 - RightRearMotor

MotorDriver2_B01 - LeftRearMotor

MotorDriver2_B02 - LeftRearMotor

*/

#include <Wire.h>

#include <math.h>

/TB6612FNG Dual Motor Driver Carrier/

const int RightFrontMotor_PWM = 2; // pwm output

const int LeftFrontMotor_PWM = 3; // pwm output

const int RightRearMotor_PWM = 4; // pwm output

const int LeftRearMotor_PWM = 5; // pwm output
```

```
//Front motors

const int RightFrontMotor_AIN1 = 22; // control Input AIN1 - right front motor

const int RightFrontMotor_AIN2 = 23; // control Input AIN2 - right front motor

const int LeftFrontMotor_BIN1 = 24; // control Input BIN1 - left front motor

const int LeftFrontMotor_BIN2 = 25; // control Input BIN2 - left front motor

//Rear motors

const int RightRearMotor_AIN1 = 26; // control Input AIN1 - right rear motor

const int RightRearMotor_AIN2 = 27; // control Input AIN2 - right rear motor

const int LeftRearMotor_BIN1 = 28; // control Input BIN1 - left rear motor

const int LeftRearMotor_BIN2 = 29; // control Input BIN2 - left rear motor


long pwmLvalue = 255;

long pwmRvalue = 255;
```

```
byte pwmChannel;

const char startOfNumberDelimiter = '<';

const char endOfNumberDelimiter = '>';


void setup(){

  Serial1.begin(9600);// HC-06 default baudrate: 9600


  //Setup RightFrontMotor

  pinMode(RightFrontMotor_AIN1, OUTPUT); //Initiates Motor Channel A1 pin

  pinMode(RightFrontMotor_AIN2, OUTPUT); //Initiates Motor Channel A2 pin


  //Setup LeftFrontMotor

  pinMode(LeftFrontMotor_BIN1, OUTPUT); //Initiates Motor Channel B1 pin

  pinMode(LeftFrontMotor_BIN2, OUTPUT); //Initiates Motor Channel B2 pin
```

```
  //Setup RightFrontMotor

  pinMode(RightRearMotor_AIN1, OUTPUT); //Initiates Motor Channel A1 pin

  pinMode(RightRearMotor_AIN2, OUTPUT); //Initiates Motor Channel A2 pin



  //Setup LeftFrontMotor

  pinMode(LeftRearMotor_BIN1, OUTPUT); //Initiates Motor Channel B1 pin

  pinMode(LeftRearMotor_BIN2, OUTPUT); //Initiates Motor Channel B2 pin



  Wire.begin();

}// void setup()


void loop(){

  if (Serial1.available()) {

    processInput();
```

```
  }

}// void loop()


void motorControl(String motorStr,int mdirection, int mspeed){

  int IN1;

  int IN2;

  int motorPWM;

  if (motorStr == "rf") { //right front

    IN1 = RightFrontMotor_AIN1;

    IN2 = RightFrontMotor_AIN2;

    motorPWM = RightFrontMotor_PWM;

  }

  else if (motorStr == "lf") { //left front

    IN1 = LeftFrontMotor_BIN1;
```

```
  IN2 = LeftFrontMotor_BIN2;

  motorPWM = LeftFrontMotor_PWM;

}

else if (motorStr == "rr") {

  IN1 = RightRearMotor_AIN1;

  IN2 = RightRearMotor_AIN2;

  motorPWM = RightRearMotor_PWM;

}

else if (motorStr == "lr") {

  IN1 = LeftRearMotor_BIN1;

  IN2 = LeftRearMotor_BIN2;

  motorPWM = LeftRearMotor_PWM;

}
```

```
  if (mdirection == 1){

    digitalWrite(IN1, LOW);

    digitalWrite(IN2, HIGH);

  }

  else if (mdirection == -1){

    digitalWrite(IN1, HIGH);

    digitalWrite(IN2, LOW);

  }

  analogWrite(motorPWM, mspeed);

}



void processInput (){

  static long receivedNumber = 0;
```

```
static boolean negative = false;

byte c = Serial1.read ();


switch (c){

case endOfNumberDelimiter:

 if (negative)

   SetPWM(- receivedNumber, pwmChannel);

 else

   SetPWM(receivedNumber, pwmChannel);


 // fall through to start a new number

case startOfNumberDelimiter:

 receivedNumber = 0;

 negative = false;
```

```
  pwmChannel = 0;

  break;


case 'f': // Go FORWARD

  goForward(255);

  //Serial.println("forward");

  break;


case 'b': // Go BACK

  goBackwad(255);

  //Serial.println("backward");

  break;


case 'r':

  moveRight(255);
```

```
  break;

case 'l':
  moveLeft(255);
  break;

case 'i':
  turnRight(255);
  break;

case 'j':
  turnLeft(255);
  break;
```

```
case 'c': // Top Right
  moveRightForward(255);
  break;

case 'd': // Top Left
  moveLeftForward(255);
  break;

case 'e': // Bottom Right
  moveRightBackward(255);
  break;

case 'h': // Bottom Left
  moveLeftBackward(255);
```

```
  break;

case 's':
  hardStop();
  break;

case 'x':
  pwmChannel = 1; // RightFrontMotor_PWM
  break;
case 'y': // LeftFrontMotor_PWM
  pwmChannel = 2;
  break;

case '0' ... '9':
  receivedNumber *= 10;
```

```
    receivedNumber += c - '0';
    break;

  case '-':
    negative = true;
    break;
  } // end of switch
} // void processInput ()

void goForward(int mspeed){
  motorControl("rf", 1, mspeed);
  motorControl("lf", 1, mspeed);
  motorControl("rr", 1, mspeed);
  motorControl("lr", 1, mspeed);
```

```
}// void goForward(int mspeed)


void goBackwad(int mspeed){
  motorControl("rf", -1, mspeed);
  motorControl("lf", -1, mspeed);
  motorControl("rr", -1, mspeed);
  motorControl("lr", -1, mspeed);
}// void goBackwad(int mspeed)


void moveRight(int mspeed){
  motorControl("rf", -1, mspeed);
  motorControl("lf", 1, mspeed);
  motorControl("rr", 1, mspeed);
  motorControl("lr", -1, mspeed);
```

```
}// void moveRight(int mspeed)


void moveLeft(int mspeed){
  motorControl("rf", 1, mspeed);
  motorControl("lf", -1, mspeed);
  motorControl("rr", -1, mspeed);
  motorControl("lr", 1, mspeed);
}// void moveLeft(int mspeed)


void moveRightForward(int mspeed){
  motorControl("rf", 1, 0);
  motorControl("lf", 1, mspeed);
  motorControl("rr", 1, mspeed);
  motorControl("lr", 1, 0);
```

```
}// void moveRightForward(int mspeed)


void moveRightBackward(int mspeed){
  motorControl("rf", -1, mspeed);
  motorControl("lf", 1, 0);
  motorControl("rr", 1, 0);
  motorControl("lr", -1, mspeed);
}// void moveRightBackward(int mspeed)


void moveLeftForward(int mspeed){
  motorControl("rf", 1, mspeed);
  motorControl("lf", 1, 0);
  motorControl("rr", 1, 0);
  motorControl("lr", 1, mspeed);
```

```
}// void moveLeftForward(int mspeed)


void moveLeftBackward(int mspeed){

  motorControl("rf", 1, 0);

  motorControl("lf", -1, mspeed);

  motorControl("rr", -1, mspeed);

  motorControl("lr", 1, 0);

}// void moveLeftBackward(int mspeed)


void turnRight(int mspeed){

  motorControl("rf", -1, mspeed);

  motorControl("lf", 1, mspeed);

  motorControl("rr", -1, mspeed);

  motorControl("lr", 1, mspeed);
```

```
}// void turnRight(int mspeed)


void turnLeft(int mspeed){

  motorControl("rf", 1, mspeed);

  motorControl("lf", -1, mspeed);

  motorControl("rr", 1, mspeed);

  motorControl("lr", -1, mspeed);

}// void turnRight(int mspeed)


void stopRobot(int delay_ms){

  analogWrite(RightFrontMotor_PWM, 0);

  analogWrite(LeftFrontMotor_PWM, 0);

  analogWrite(RightRearMotor_PWM, 0);

  analogWrite(LeftRearMotor_PWM, 0);
```

```
  delay(delay_ms);

}// void stopRobot(int delay_ms)


void hardStop(){

  analogWrite(RightFrontMotor_PWM, 0);

  analogWrite(LeftFrontMotor_PWM, 0);

  analogWrite(RightRearMotor_PWM, 0);

  analogWrite(LeftRearMotor_PWM, 0);

}// void stopRobot()


void SetPWM (const long pwm_num, byte pwm_channel){

  if(pwm_channel==1){ // DRIVE MOTOR

    analogWrite(RightFrontMotor_PWM, pwm_num);

    pwmRvalue = pwm_num;
```

```
  }

  else if(pwm_channel==2){ // STEERING MOTOR

    analogWrite(LeftFrontMotor_PWM, pwm_num);

    pwmLvalue = pwm_num;

  }

}// void SetPWM (const long pwm_num, byte pwm_channel)
```