INDEX

1. INTRODUCTION 1
    a. WHY DID WE CHOOSE THIS PROJECT? 1
    b. MATERIALS USED IN THIS PROJECT 1
    c. OUTCOMES FROM THIS PROJECT 2
    d. APPLICATIONS OF THIS PROJECT 2
2. MECHANICAL CONCEPTS 3
    a. DIFFERENTIAL DRIVE 3
    b. CONTROL MECHANISM 6
3. ELECTRONIC COMPONENTS 8
    a. ARDUINO MEGA 2560 8
    b. L298N MOTOR DRIVER 11
    c. WIRELESS GAME CONTROLLER 12
4. INTERFACING ELECTRONICS 14
5. SOFTWARE CONTROL 16
    a. FORWARD AND REVERSE CONTROL 16
    b. LEFT AND RIGHT TURNING 17
    c. LEFT AND RIGHT MOVEMENT 17
    d. ABRUPT BRAKING 18

APPENDIX - ARDUINO C CODE

INTRODUCTION

Our project is a **three-wheel differential drive robot**. This robot is composed of three omni-directional wheels (two of which are arranged in a differential drive), and is supported by the Arduino Mega 2560 microcontroller. The robot is controlled by a Wireless Game Controller (Sony PlayStation DualShock 2).

WHY WE CHOSE THIS PROJECT

We believe that robotics and its allied technologies (e.g. artificial intelligence, computer vision, deep learning, etc.) is the future; a **fourth Industrial Revolution**, so to say. Factories have started to rely heavily on robots because of their better precision and efficiency with respect to humans. Recent trends in robotics show us why robots are advantageous, some of which are elicited below:

1. Robots can go in unknown spaces inaccessible to humans.
2. They don't require motivation (salary, food) to work.
3. They can perform tasks faster than humans, while also being consistent and accurate.
4. They do not usually require any human interference.
5. They are able to reduce overall cost of a product (reducing overhead and maintenance costs)

These reasons have interested us in the world of robotics and this project is the perfect opportunity to introduce ourselves in this world. If we get opportunities like this in the future, we would love to design more robots and help improve many lives.

MATERIALS USED IN THIS PROJECT

HARDWARE USED:

- Arduino Mega 2560
- L298N Motor Driver
- 12V DC Motors
- Wireless Game Controller
- Omni Wheels
- 12V Li-Po battery

SOFTWARES USED:

- Arduino IDE
- Fritzing Circuit Designer

OUTCOMES FROM THIS PROJECT

Through this project we aim to achieve a control mechanism, via a microcontroller, for a 3-wheel Differential Drive Robot. This project covers a wide range of mechatronics engineering engineering concepts ranging from mechanical concepts like Differential Drive, Forward Kinematics, Inverse Kinematics, DC Motors to electronics concepts like Pulse Width Modulation, Microcontrollers and Interfacing, Motor Control. The project also requires a basic to intermediate understanding of C programming, to be used in Arduino.

END GOALS:

- To understand the basic concepts of Differential Drive and to achieve the same with 3 wheels, ensuring the minimum possible turning radius.
- To align and orient the wheels so as to build a stable system of operation.
- To understand the basic concepts of Pulse Width Modulation and to incorporate in controlling the motors attached to the wheels of the robot.
- To formulate functions in C employing the best use of Arduino functionalities to give commands wirelessly to the robot.

APPLICATIONS OF THIS PROJECT

The motion of a differential drive robot is pretty straightforward and fairly easy to program. This is why it is used in almost all the consumer robots. It provides a very low turning radius, as opposed to a conventional four-wheeled robot, which makes it suitable for use in warehouse organisation and management. In recent times, robots are employed for use in hospitals and restaurants to pick, carry and place items from one place to a designated other place. Differential steering in such cases helps the robot to follow a path or optimise a new one.

Robots have always been associated with a lot of wires and electronic components for their operation. The ability to control a robot wirelessly, that too with a simple Game Controller, opens a lot more areas for it. Firstly, the dependence on wires is reduced to a great extent, which makes it more mobile and able to cover large distances without the operator attached with it. Secondly, wireless control through a microcontroller allows the user to program the robot in whatever way he wants, as opposed to a DPDT switch controlled robot which has a very limited scope in manipulating the control.

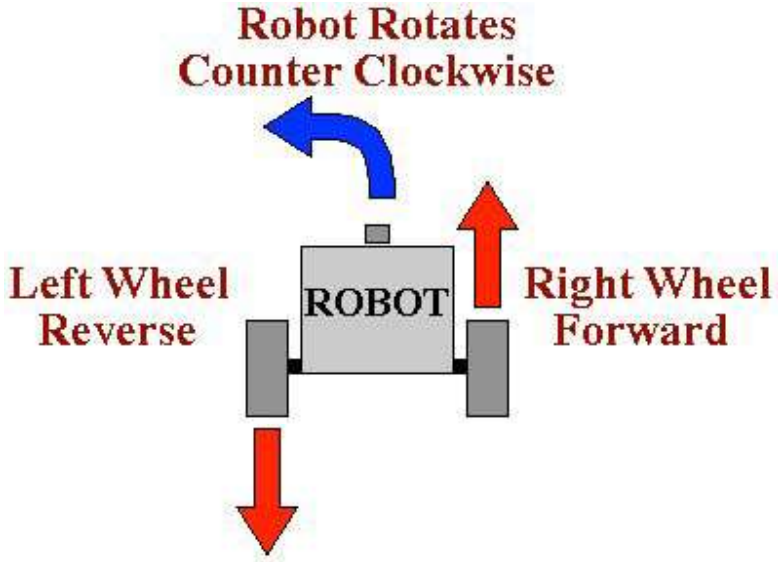
Wirelessly controlling a robot is the first step towards automation. With each and every industry employing Artificial Intelligence (AI) in its machinery, it is the need of the hour to make the robots intelligent as well. The wireless control achieved in this project is completely through software, which makes it even more convenient for an Artificial Intelligence system to take over its control. The robot conceptualised in this project can very well be automated by using Computer Vision, Motion Planning, Remote Sensing and similar AI-based technologies.

MECHANICAL CONCEPTS

The underlying mechanical principle behind this project is of the **differential drive**. Three **omni-directional wheels** (two wheels on the axle, one front castor wheel for additional support) are a part of said drive.

DIFFERENTIAL DRIVE

Differential drive consists of 2 drive wheels mounted on a common axis, and each wheel can independently be driven either forward or backward.
The term 'differential' means that robot turning speed is determined by the speed difference between both wheels, each on either side of the robot.
The move commands sent by the motors (described in the next section) dictate the speed of each wheel in the drive at every instant of time.



Differential Drive Kinematics:

While one can vary the velocity of each wheel on the drive, for the robot to perform rolling motion, it must rotate about a point that lies along its common left and right wheel axis. The point that the robot rotates about is known as the **instantaneous center of curvature (ICC)**.
By varying the velocities of the two wheels, the trajectories that the robot takes can be varied. Because the rate of rotation ω about the ICC must be the same for both wheels, these equations follow:

$$\omega\,(R + l/2) = Vr\text{, and}$$
$$\omega\,(R - l/2) = Vl\text{, where}$$

l -> distance between the centers of the two wheels

Vr, Vl -> right, left wheel velocities along the ground

R -> signed distance from the ICC to the midpoint between the two wheels

At any instance in time R and ω can be solved as follows:

$$R = (l/2)[(Vr + Vl)/(Vr - Vl)], \text{ and}$$

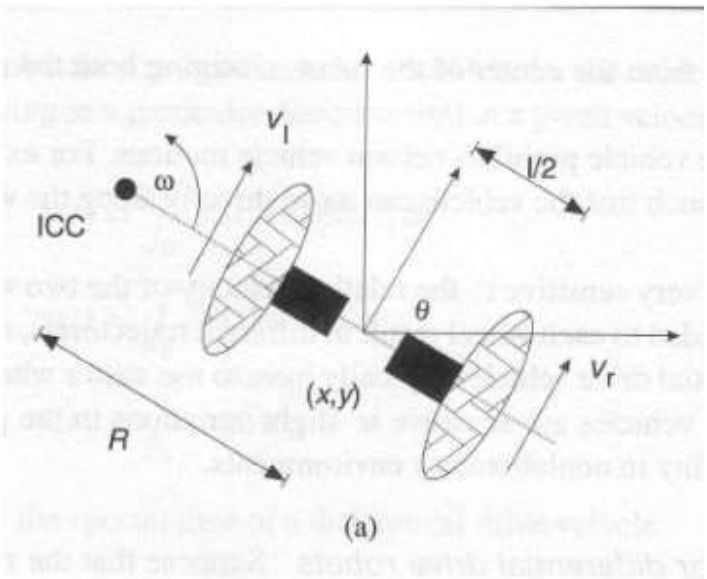
$$\omega = (Vr - Vl)/l$$

Three cases can be observed with these drives:

1. If $Vl = Vr$, the robot moves forward linearly in a straight line. R becomes infinite, and there is effectively no rotation ($\omega = 0$).
2. If $Vl = -Vr$, $R = 0$ and the robot rotates about the midpoint of the wheel axis (rotation in place).
3. If $Vl = 0$, the robot rotates about the left wheel. Here, $R = l/2$. If $Vr = 0$, the robot would rotate about the right wheel.

A robot with a differential drive has some limitations as well. It cannot move in the direction along the wheel axis (a singularity). It is very sensitive to slight changes in velocity in each of the wheels. Small errors in the relative velocities between the wheels can affect the trajectory of the robot. A two-wheel differential drive is also very sensitive to small variations in the ground plane, so require castor wheels for additional support (as is used in this project).

Forward Kinematics for Differential Drive Robots:



Assumptions:

1. Robot is at position (x, y) headed in a direction making angle θ with X axis.
2. Robot is centered at a point midway along the wheel axle.

By manipulating control parameters (Vl, Vr), the robot can move to different positions and orientations.

The ICC can be found by the following (knowing $(Vl,\ Vr)$):

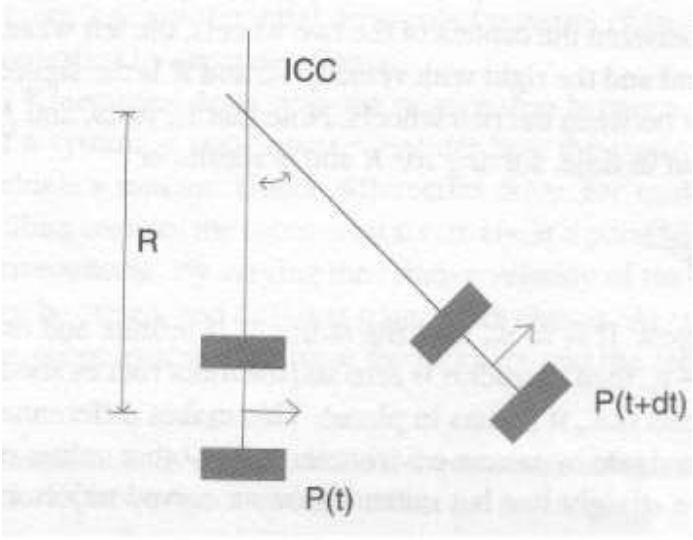
$$ICC\ =\ (x\ -\ Rsin\theta,\ y\ +\ Rcos\theta)$$

At time $t\ +\ \delta t$, the robot's pose can be described as follows:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} cos(\omega\delta t) & -sin(\omega\delta t) & 0 \\ sin(\omega\delta t) & cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix}$$

This equation describes the motion of a robot rotating a distance R about its ICC with angular velocity ω:

1. Translate ICC to the origin of the coordinate system.
2. Rotate about the origin by an angular amount $\omega\delta t$.
3. Translate back to the ICC.



Inverse Kinematics for Differential Drive Robots:

The position of a differential drive robot capable of moving in a particular direction $\Theta(t)$ at a given velocity $V(t)$ can be given by:

$$x(t)\ =\ (1/2)\int_0^t [Vr(t)\ +\ Vl(t)]cos[\theta(t)]dt$$

$$y(t)\ =\ (1/2)\int_0^t [Vr(t)\ +\ Vl(t)]sin[\theta(t)]dt$$

$$\Theta(t)\ =\ (1/l)\int_0^t [Vr(t)\ -\ Vl(t)]dt$$

Inverse kinematics deals with controlling the robot to reach a given configuration (x, y, θ). Unfortunately, a differential drive imposes non-holonomic constraints on establishing its position. For example, the robot cannot move laterally on its axle.

Equations for Special Cases:

1. $Vl = Vr = V$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + v\,cos(\theta)\delta t \\ y + v\,sin(\theta)\delta t \\ \theta \end{bmatrix}$$

2. $Vr = -Vl = V$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + 2v\delta t/l \end{bmatrix}$$

This motivates a navigational strategy of moving the robot in a straight line, then rotating for a turn in place, and then moving straight again.

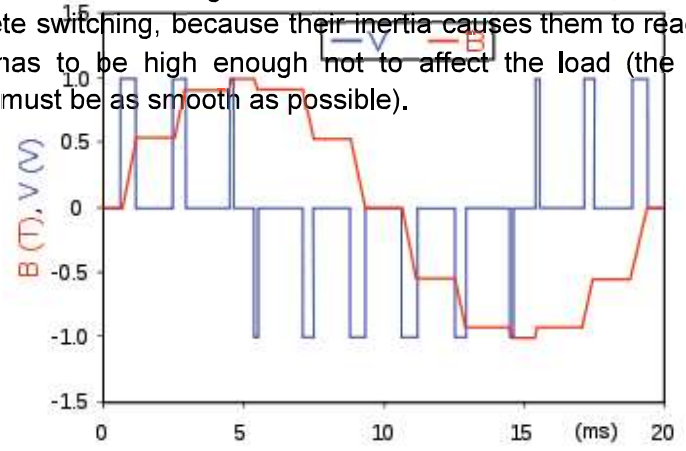
CONTROL MECHANISM

The omni-directional wheels on the axle are attached to **motors**, whose control is achieved by sending a **PWM signal**, determining the position to be achieved by their respective motor.

PWM (Pulse-Width Modulation):

Pulse-Width Modulation is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts. The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to the off periods, the higher the total power supplied to the load.
PWM is particularly suited for running inertial loads such as motors, which are not as easily affected by this discrete switching, because their inertia causes them to react slowly. The PWM switching frequency has to be high enough not to affect the load (the resultant waveform perceived by the load must be as smooth as possible).

Motor:



A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If connected directly to a battery, the motor will rotate. If the leads are switched, the motor will rotate in the opposite direction.

Its characteristics are as follows:

- Rotation is unlimited
- Rotating direction: clockwise, counter-clockwise
- Rotating position: very difficult to precisely rotate to a specific angle
- Rotating speed: easy to control how fast, but very difficult to precisely control to a specific speed value.
- Extra hardware driver: required
- Extra power supply: required
- How to control: easy (by using high-voltage PWM signal)

DC motors are generally used to control something that needs to rotate continuously but the value of speed does not need to be specified, which is ideal for this project.

Motor Control:

Motor control is achieved by sending the motor a PWM signal where either the width of the pulse or the duty cycle of a pulse train (less common today) determines the parameters to be achieved by the motor.

ELECTRONIC COMPONENTS

The electronics of this project is pretty straightforward. The control of the 3-wheel differential drive robotic system is achieved through an **Arduino microcontroller**. Each of the motors was controlled via an **L298N motor driver IC**. The whole system is made wireless through a **PS2 game controller**, the wireless receiver of which is interfaced to the microcontroller with the help of jumper wires. The whole system functions on 12V, which is supplied by a **Li-Po battery**.

The following electronic components are used:

1. Arduino Mega 2560 Microcontroller
2. L298N Motor Driver IC
3. Wireless Game Controller
4. Wireless Receiver
5. 12V Li-Po Battery
6. Jumper Wires
7. Copper Wires

ARDUINO MEGA 2560

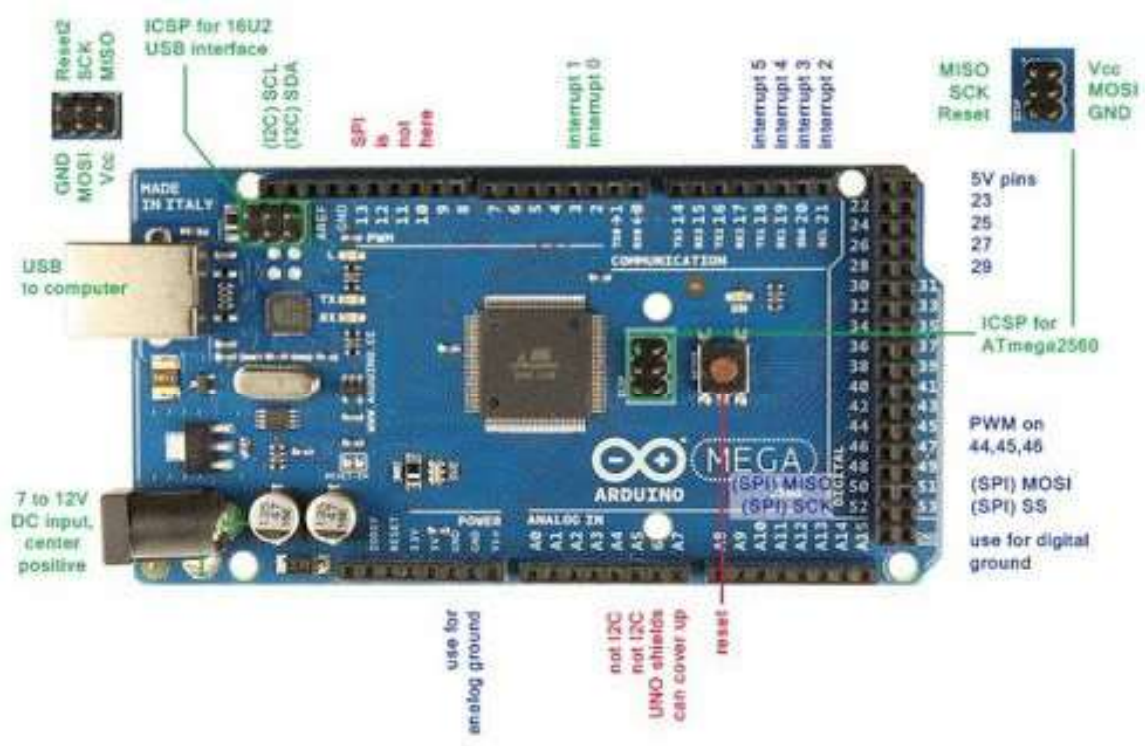


Fig - Arduino Mega 2560

Features of Arduino Mega 2560:

Arduino Mega 2560 is a Microcontroller board based on Atmega2560. It comes with more memory space and I/O pins as compared to other boards available in the market. There are 54 digital I/O pins and 16 analog pins incorporated on the board that make this device unique and stand out from others. Out of 54 digital I/O, 15 are used for PWM (pulse width modulation). A crystal oscillator of 16MHz frequency is added on the board. This board comes with a USB cable port that is used to connect and transfer code from computer to the board. A DC power jack is coupled with the board that is used to power the board. Some versions of the Arduino board lack this feature like Arduino Pro Mini doesn't come with a DC power jack. ICSP header is a remarkable addition to Arduino Mega which is used for programming the Arduino and uploading the code from the computer.

Power Regulation in Arduino:

This board comes with two voltage regulators i.e. 5V and 3.3V which provides the flexibility to regulate the voltage as per requirements as compared to Arduino Pro Mini which comes with only one voltage regulator. There are three ways to power the board. You can either use a USB cable to power the board and transfer code to the board or you can power it up using Vin of the board or through Power jack or batter. Last two sources to power the board are required once you have already built and compiled code into the board through USB cable.

Comparison with Other Arduino boards:

There is no much difference between Arduino Uno and Arduino Mega except the latter comes with more memory space, bigger size and more I/O pins. Availability of Atmega16 on the board makes it different from Arduino Pro Mini which uses USB to serial converter to program the board. There is a reset button and 4 hardware serial ports called USART which produces a maximum speed for setting up communication.

Programming the Arduino board:

Arduino software called Arduino IDE is used to program the board which is a common software used for all boards belonging to the Arduino family. Arduino Mega 2560 can be programmed using Arduino Software called IDE which supports C programming. The code you make on the software is called sketch which is burned in the software and then transferred to the board through USB cable. This board comes with a built-in bootloader which rules out the usage of an external burner for burning the code into the board. The bootloader communicates using STK500 protocol. Once you compile and burn the program on the board, you can unplug the USB cable which eventually removes the power from the board. When you intend to incorporate the board into your project, you can power it up using a power jack or Vin of the board.

Fig - Arduino IDE

Applications of Arduino Mega 2560:

Multitasking is another feature where Arduino mega comes handy. However, Arduino IDE Software doesn't support multitasking feature but you can use other operating systems like FreeRTOS and RTX to write C programs for this purpose. This gives you the flexibility of using your own custom build program using ISP connector.

Arduino Mega is specially designed for the projects requiring complex circuitry and more memory space. Most of the electronic projects can be done pretty well by other boards available in the market which make Arduino Mega uncommon for regular projects. However, there are some projects that are solely done by Arduino Mega like making 3D printers or controlling more than one motors, because of its ability to store more instructions in the code memory and a number of I/O digital and analog pins.

L298N MOTOR DRIVER

The **L298N Motor Driver Module** is a high power motor driver module for driving DC and Stepper Motors. This module consists of an **L298 motor driver IC** and a **78M05 5V regulator**. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

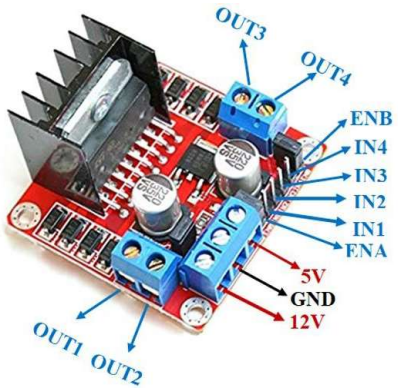


Fig - L298N Motor Driver Module Pinout Diagram

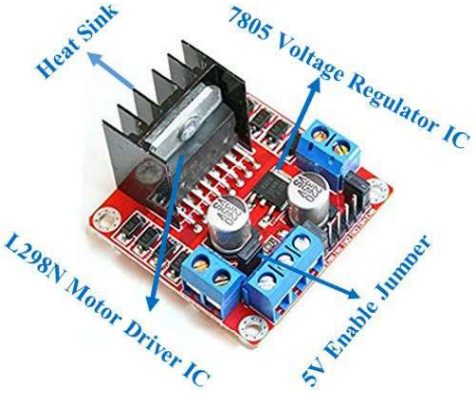


Fig - L298N Motor Driver Module Components

The 78M05 Voltage regulator will be enabled only when the jumper is placed. When the power supply is less than or equal to 12V, then the internal circuitry will be powered by the voltage regulator and the 5V pin can be used as an output pin to power the microcontroller. The jumper should not be placed when the power supply is greater than 12V and separate 5V should be given through the 5V terminal to power the internal circuitry.

ENA & ENB pins are speed control pins for Motor A and Motor B while IN1& IN2 and IN3 & IN4 are direction control pins for Motor A and Motor B.

Pin Name	Description
IN1 & IN2	Motor A input pins. Used to control the spinning direction of Motor A
IN3 & IN4	Motor B input pins. Used to control the spinning direction of Motor B
ENA	Enables PWM signal for Motor A
ENB	Enables PWM signal for Motor B
OUT1 & OUT2	Output pins of Motor A

OUT3 & OUT4	Output pins of Motor B
12V	12V input from DC power Source
5V	Supplies power for the switching logic circuitry inside L298N IC
GND	Ground pin

Table - Description of Pins of L298N Module

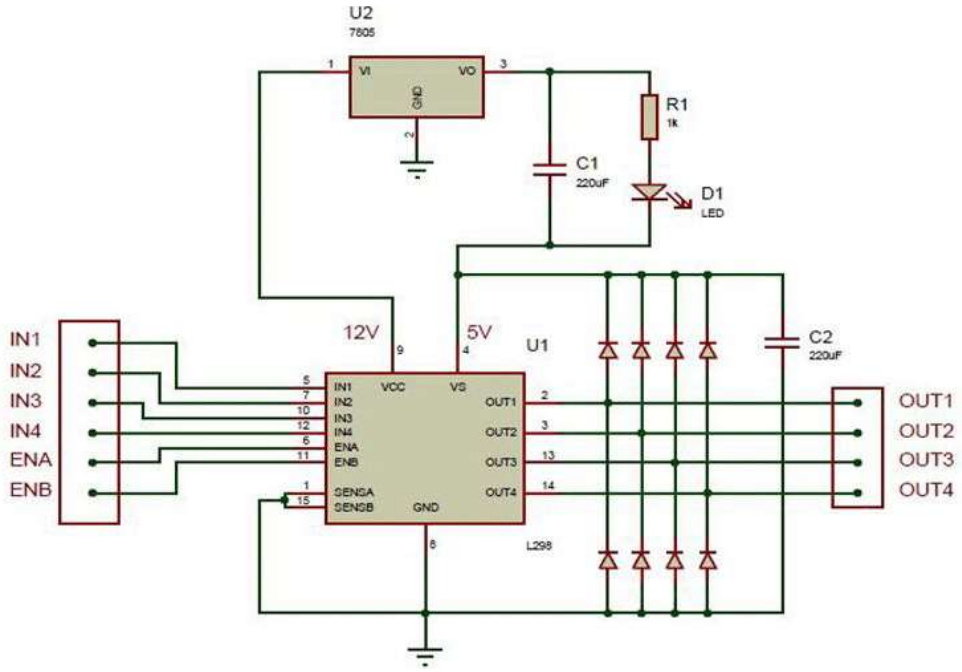


Fig - Internal Circuitry of a L298N Motor Driver

WIRELESS GAME CONTROLLER

The PS2 Game Controller is a wireless gamepad with a bunch of analog and digital inputs. It is controlled wirelessly by a Receiver which is plugged into the main console and gives the necessary signals for it to operate.

The gamepad contains **two joystick inputs** which can be rotated in 360^0 in the plane. In each direction (Up, Down, Left, Right) it takes values from 0 to 255 with 0 being the position of rest, i.e. the middle. The left and right parts have **four push buttons** each, which can be used in two ways: triggering the functionality once when it is pressed, or keep the functionality running as long as it is pressed. The back side contains **two tactile switches** on each side which work in the second way as explained above.

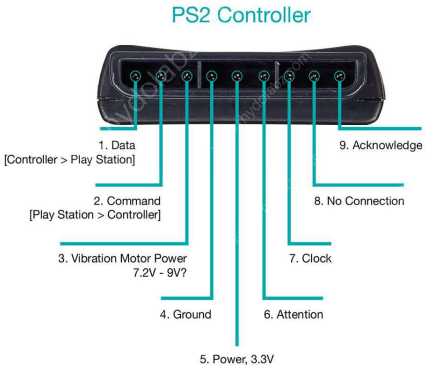


Fig - Wireless Game Controller Receiver

Pin Name	Description
Data	master line for sending data to slave (MOSI)
Command	slave line for sending data to master (MISO)
Vibration	vibration motors supply; 7.2 volts to 9 volts
Ground	circuits ground
VCC	circuits supply; 3.3 volts
Attention	CS or Chip Select pin for calling slave and preparing the connection
Clock	equivalent to SCK pin for clock
No Connection	Supplies power for the switching logic circuitry inside L298N IC
Acknowledge	acknowledge signal from the controller to PS2 receiver

Table - Description of Pins of Wireless Receiver

INTERFACING ELECTRONICS

The circuit for the proposed Differential Drive Robot with Wireless Control, designed in Fritzing, is as follows:

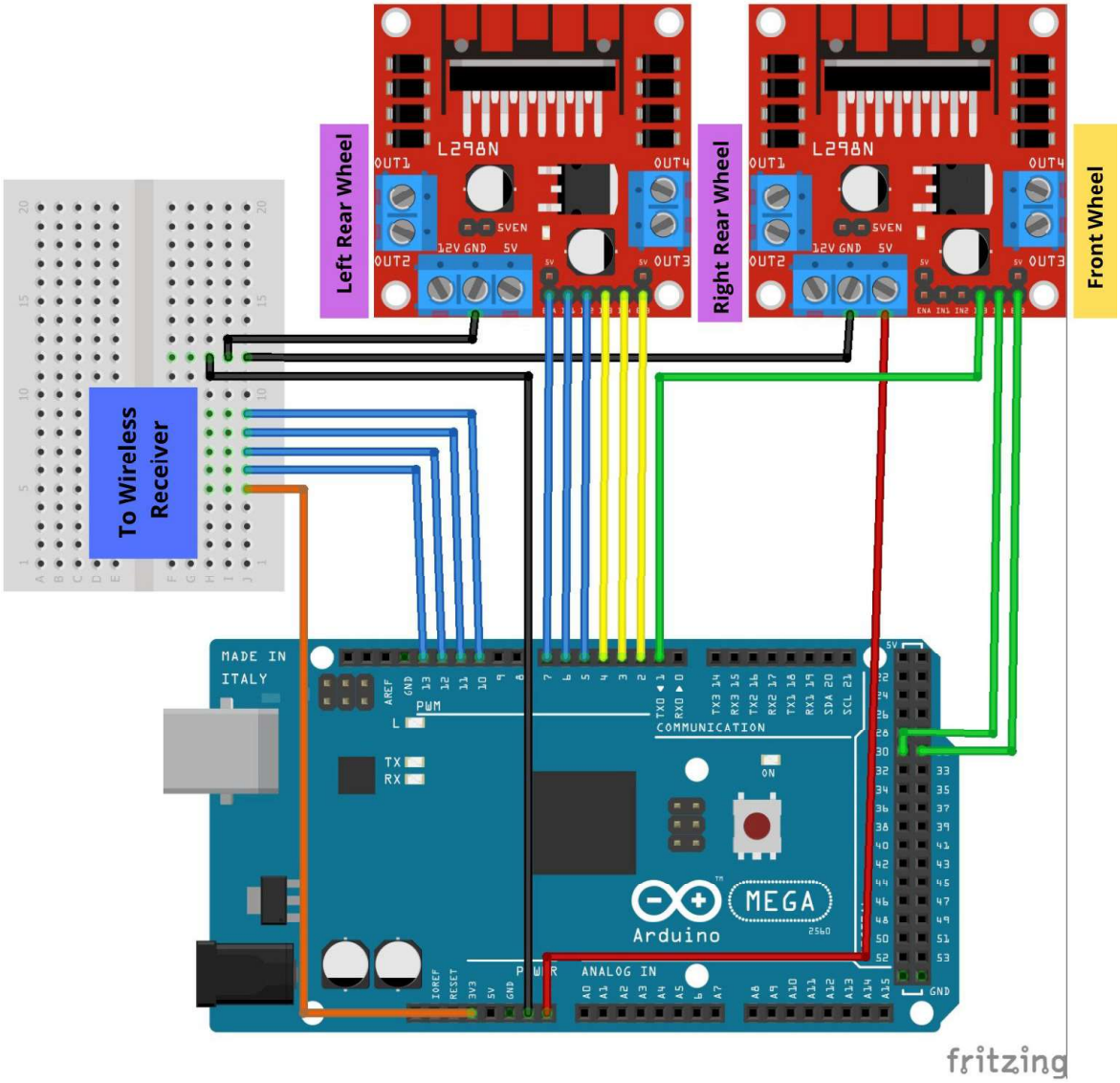


Fig - Circuit Schematic for the Differential Drive Robot

To interface the Wireless Game Controller with the Arduino microcontroller, the open-source PS2X Library, developed by Bill Porter, is used. This library maps the inputs of the Game Controller as follows:

Key	Function	Digital/Analog
PSB_SELECT	OK	Digital
PSB_START	OK	Digital
PSB_PAD_UP	UP	Analog
PSB_PAD_DOWN	DOWN	Analog
PSB_PAD_LEFT	LEFT	Analog
PSB_PAD_RIGHT	RIGHT	Analog
PSB_BLUE	X	Analog
PSB_GREEN	Triangle	Analog
PSB_PINK	Square	Analog
PSB_RED	Circle	Analog
PSB_L3	L3	Digital
PSB_R3	R3	Digital
PSB_L2	L2	Analog
PSB_R2	R2	Analog
PSB_L1	L1	Analog
PSB_R1	R1	Analog
PSB_RX	Joystick right x	Analog
PSB_RY	Joystick right y	Analog
PSB_LX	Joystick left x	Analog
PSB_LY	Joystick left y	Analog

Table - Types of Keys in the Game Controller

Two L298N Motor Driver modules are used, since one driver has the capability to drive two motors. Both the rear wheels are paired up through one of the motor drivers. The front castor wheel isn't paired up with any other motor through the motor driver.

The motor driver with two motors takes up six digital I/O pins (two PWM pins in them), and the other one takes three of the pins (one being PWM). The Wireless Receiver takes up four of the digital I/O pins of the Arduino for Data, Command, Attention and Clock. All in all, 13 digital I/O pins are used by just the receiver and the motor driver, hence the Arduino board with the maximum number of pins is used, i.e. Mega 2560.

SOFTWARE CONTROL

The main aim behind using a wireless game controller to control a differential drive robotic system is to allow a certain degree of ease of operation. The plethora of inputs available make it a lot easier to configure the system to function in any way intended. The whole functionality is dependent on the program that is downloaded on the Arduino board.

One particular set of controls which is configured in this system is as follows:

Movement	Control Input	Function in Code
Forward Movement	Left Joystick	`moveForward()`
Reverse Movement		`moveBackward()`
Left Turning	Right Joystick	`turnleft()`
Right Turning		`turnright()`
Left Movement	L2 button	`moveleft()`
RIght Movement	R2 button	`moveright()`
Abrupt Braking	R1 button	`halt()`

Table - Control Configuration

Forward and Reverse Movement -

```
void moveForward()
{
  mapped = 255 - (2*LY);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, mapped);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, mapped);
}
```

```
void moveBackward()
{
  mapped = (2*LY) - 255;
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, mapped);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enB, mapped);
}
```

For Forward and Reverse Movements, only the two rear wheels are used. in1 of both the wheels is turned HIGH for forward and LOW for reverse movements. The speed of motors is mapped by the displacement of the joystick from the mean position.

Left and Right Turning -

```
void turnleft()
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, 255);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, 255);
  analogWrite(enC, 255);
  digitalWrite(in5, HIGH);
  digitalWrite(in6, LOW);
}
```

```
void turnright()
{
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, 255);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enB, 255);
  analogWrite(enC, 255);
  digitalWrite(in5, LOW);
  digitalWrite(in6, HIGH);
}
```

For left and right turning, all the three wheels are used. To turn left, the front wheel moves in the left (which is forward in its frame of reference), the left rear wheel moves backward and the right rear wheel moves forward, giving a complete circular movement with least radius of turning. Similarly, all the wheels move in the opposite direction to achieve the right turning motion. The speed of motors is set to highest, so that a uniform turning happens in the least amount of time.

Left and Right Movement -

```
void moveleft()
{
  mapped = 255 - (2*LX);
  mapped = mapped*4  /5;
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  digitalWrite(in5, LOW);
  digitalWrite(in6, HIGH);
  analogWrite(enC, mapped);
}
```

```
void moveright()
{
  mapped = (2*LX) - 255;
  mapped = mapped*4/5;
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  digitalWrite(in5, HIGH);
  digitalWrite(in6, LOW);
  analogWrite(enC, mapped);
}
```

For left and right movement / directional alignment, only the front wheel is used. The intention behind this movement is to align the robot in a particular direction after tuning, before moving forward or backward. The speed is controlled by the displacement of the joystick from mean position.

Abrupt Braking -

```
void halt()
{
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  analogWrite(enC, 0);
}
```

This control is added as a safety precaution. All the motors are immediately stopped as soon as the input is provided. The motors stop at the same instant to avoid any unwanted movement.

APPENDIX

ARDUINO C CODE

```
/*
Differential Drive Robot
Arduino 2560
*/

#include <PS2X_lib.h>
#include <Servo.h>

#define PS2_DAT         12
#define PS2_CMD         13
#define PS2_SEL         11
#define PS2_CLK         10

/* Motor driver 1 has motors {enA,in1,in2} and {enC, in5, in6}
Motor driver 2 has motor {enB, in3, in4}
*/

#define enA 2   //Motor connected on driver 1
#define in1 52
#define in2 3

#define enB 6   //Motor connected on driver 2
#define in3 5
#define in4 4

#define in5 8   //Motor connected on driver 1
#define in6 9
#define enC 7


Servo myservo;

int RY, RX, LX, LY, mapped;

//#define pressures   true
#define pressures   false
//#define rumble        true
#define rumble    false
```

```
PS2X ps2x; // create PS2 Controller Class

int error = 0;
byte type = 0;
byte vibrate = 0;
int flag = 0;

void setup()
{

  Serial.begin(57600);
  myservo.attach(24);
  delay(300);
  error = ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_SEL, PS2_DAT,
pressures, rumble);

  if(error == 0)
  {
      Serial.print("Found Controller, configured successful ");
      Serial.print("pressures = ");

      if (pressures)
      Serial.println("true ");

      else
      Serial.println("false");
      Serial.print("rumble = ");

      if (rumble)
            Serial.println("true)");

      else
            Serial.println("false");
            Serial.println("Try out all the buttons, X will vibrate the
controller, faster as you press harder;");
            Serial.println("holding L1 or R1 will print out the analog
stick values.");
            Serial.println("Note: Go to www.billporter.info for updates and
to report bugs.");
  }
  else if(error == 1)
      Serial.println("No controller found, check wiring, see readme.txt to
```

```
enable debug. visit www.billporter.info for troubleshooting tips");

  else if(error == 2)
      Serial.println("Controller found but not accepting commands. see
readme.txt to enable debug. Visit www.billporter.info for troubleshooting
tips");

  else if(error == 3)
      Serial.println("Controller refusing to enter Pressures mode, may not
support it. ");

//  Serial.print(ps2x.Analog(1), HEX);

  type = ps2x.readType();

  switch(type)
  {
      case 0:
      Serial.print("Unknown Controller type found ");
      break;

      case 1:
      Serial.print("DualShock Controller found ");
      break;

      case 2:
      Serial.print("GuitarHero Controller found ");
      break;

      case 3:
      Serial.print("Wireless Sony DualShock Controller found ");
      break;
  }

}

void moveForward()
{
  mapped = 255 - (2*LY);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, mapped);
  digitalWrite(in3, LOW);
```

```
  digitalWrite(in4, HIGH);
  analogWrite(enB, mapped);
}

void moveBackward()
{
  mapped = (2*LY) - 255;
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, mapped);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enB, mapped);
}

void turnright()
{
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, 255);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enB, 255);
  analogWrite(enC, 255);
  digitalWrite(in5, LOW);
  digitalWrite(in6, HIGH);
}

void turnleft()
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, 255);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, 255);
  analogWrite(enC, 255);
  digitalWrite(in5, HIGH);
  digitalWrite(in6, LOW);
}

void halt()
{
```

```
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  analogWrite(enC, 0);
}

void moveleft()
{
  mapped = 255 - (2*LX);
  mapped = mapped*4  /5;
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  digitalWrite(in5, LOW);
  digitalWrite(in6, HIGH);
  analogWrite(enC, mapped);
}


void moveright()
{
  mapped = (2*LX) - 255;
  mapped = mapped*4/5;
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  digitalWrite(in5, HIGH);
  digitalWrite(in6, LOW);
  analogWrite(enC, mapped);
}

void loop()
{

  if(error == 1) //skip loop if no controller found
      return;

  else  //DualShock Controller
  {

      ps2x.read_gamepad(false, vibrate); //read controller and set large
motor to spin at 'vibrate' speed

      LX = ps2x.Analog(PSS_LX);
      LY = ps2x.Analog(PSS_LY);
```

```
if(LY>127)
{
moveBackward();
delay(100);
}

if(LY<127)
{
moveForward();
delay(100);
}

if(LX<127)
{
moveleft();
delay(30);
}

if(LX>127)
{
moveright();
delay(30);
}


if(ps2x.Button(PSB_R1))
{
halt();
}

if(ps2x.ButtonPressed(PSB_L2))      //turning left
{
turnleft();
delay(40);

if(ps2x.ButtonReleased(PSB_L2))
halt();
}

if(ps2x.ButtonPressed(PSB_R2))      //turning right
{
turnright();
delay(40);
```

```
      if(ps2x.ButtonReleased(PSB_R2))
      halt();
      }

      if(ps2x.ButtonPressed(PSB_CROSS))   //Servo flap in one direction
      {
      myservo.write(180);
      delay(10);
      }

      if(ps2x.ButtonPressed(PSB_TRIANGLE))      //Servo flap in other
direction
      {
      myservo.write(530);
      delay(10);
      }

  }

  delay(30);

}
```