Title: Development of an autonomous vehicle

Applied Electrical and Electronic Engineering: Construction Project (EEEE1002 UNUK) (FYR1 24-25): Technical Report 1

Student ID: 20673826

Full Name: Mohamed Unuse Jalloh

Due Date: 09/01/25

Abstract

Autonomous Vehicles offers the next step in the technological advancement of transportation through bypassing human intervention. This provides society with immense benefits such as 24-hour public transport and lower cost of transporting resources for businesses. This paper demonstrates my design, implementation and testing of an autonomous vehicle. The test in question would be a 10m challenge in which I see whether or not my autonomous vehicle has the capability to travel 10m linearly. My approach to succeed in my goal was to first build a H-Bridge circuit thus providing efficient bi-directional control of DC motors. This design choice was crucial as alongside the main microcontroller (the ESP32) would make designing the software easier & efficient. My initial attempt at the 10m challenge saw my vehicle unable to reach the target due to directional issues as it veered towards the left. My second attempt bared far better results as my vehicle reached 5m before veering towards the left. These differences in results came out due to software tweaks that will be explored in greater detail later on within this paper. Though the final result didn't fully meet the criteria it did demonstrate the wider project context and industrial relevance.

Table of Contents

Contents

Introduction

The operational efficiency of an autonomous vehicle heavily depends on adequate and simple control of the motors and also the provision of enough power to both motors to prevent an imbalance. Control of the motors is essential as it will provide a platform for us to integrate software to control the direction and speed of the vehicle remotely without human intervention. Power delivery is also imperative as if one motor has more power than the other it creates an imbalance on one side thus leading to the vehicles movement to be tilted to one direction at all times. This paper introduces systems in order to control the motors and efficient power delivery through various methods such as logic circuits and H-Bridge circuits

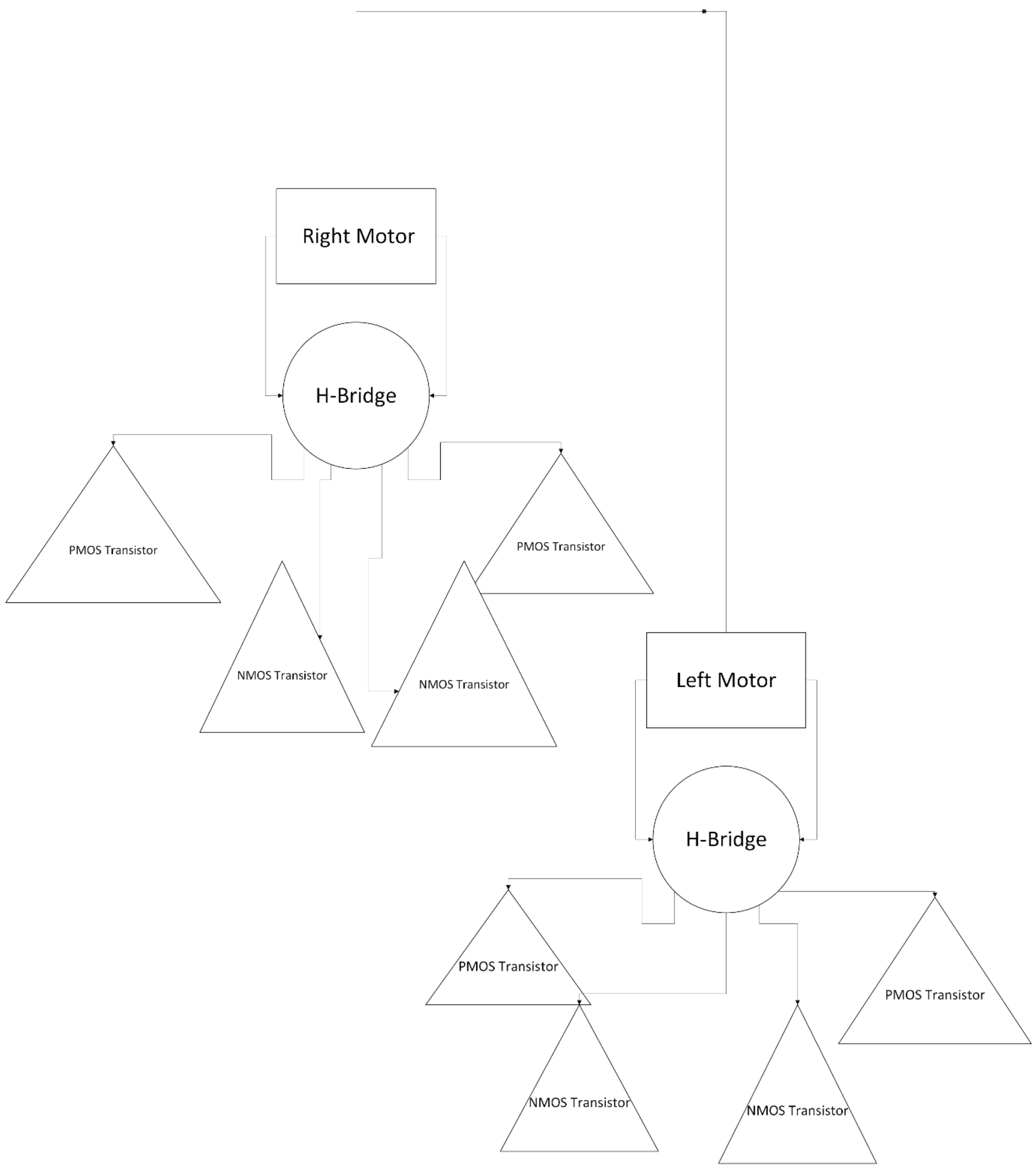


Figure 1:System Overview

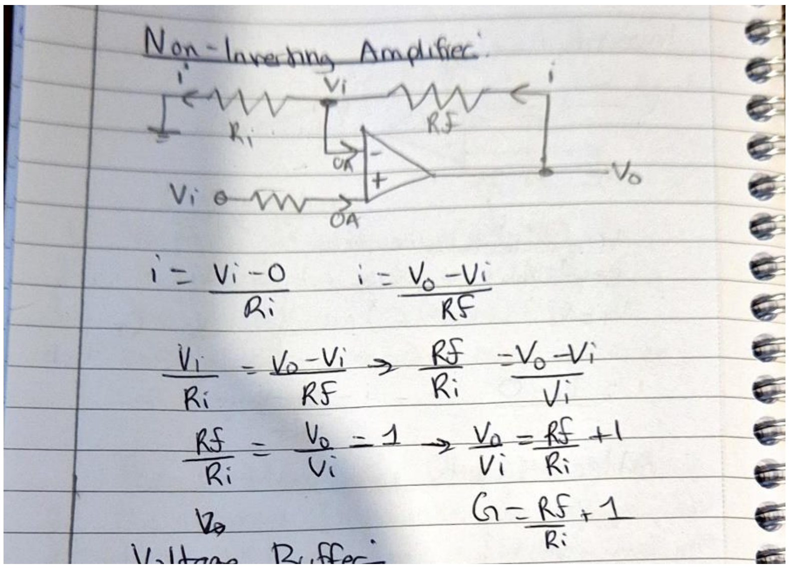
1. Implementation of an Op-Amp Circuit

This section displays my design and implementation of an operational amplifier circuit. The specific amplifier used within this design is the MCP6292 which is a non-inverting amplifier

1.1 Principle of Operation

An operational amplifier is a differential amplifier with close to ideal resistances. The characteristics of an Op-Amp are very high input resistances; Very low output resistances; very high gain and voltage gains consistent over wide range of frequencies.

The large gains of an Op-Amp can be manipulated via the input resistance and the reference resistance. This is due to the effect of the voltage divider in which voltage it split between two resistors. Therefore, by utilising a negative feedback loop by connecting the reference resistor from the input resistor to the voltage output of the Op-Amp. As a result, the gain would be determined by the ratio between the input of reference resistor.



$$i = \frac{V_i - 0}{R_i} \qquad i = \frac{V_o - V_i}{R_F}$$

$$\frac{V_i}{R_i} = \frac{V_o - V_i}{R_F} \rightarrow \frac{R_F}{R_i} = \frac{V_o - V_i}{V_i}$$

$$\frac{R_F}{R_i} = \frac{V_o}{V_i} - 1 \rightarrow \frac{V_o}{V_i} = \frac{R_F}{R_i} + 1$$

$$G = \frac{R_F}{R_i} + 1$$

Figure 2:Op-Amp Gain Derivation

As such, for a non-inverting amplifier the gain would be:

$$G = \frac{Rf}{Ri} + 1$$

1.2 Schematic Design & Simulation

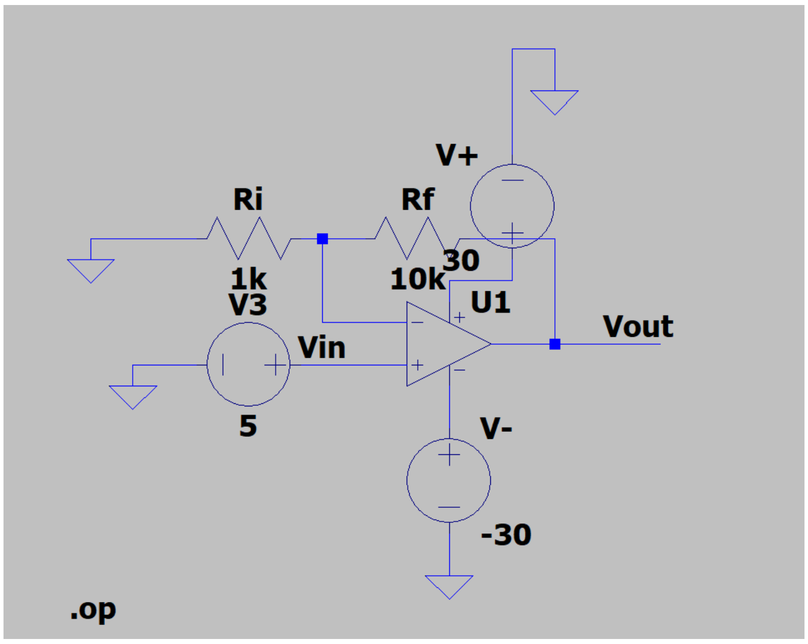


Figure 3:Op-Amp Circuit Schematic

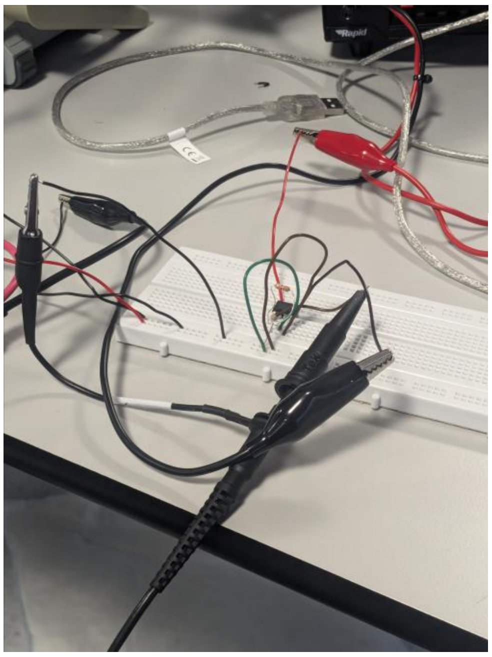
This design schematic utilises a universal amplifier therefore the real outputs in actual testing may be different. However, this schematic displays the actual design of the circuit that I will be using in for the actual Op-Amp circuit using the MCP6292. According to the Gain equation derived in the previous section a reference resistance of 10kΩ and an input resistance of 1kΩ would create a gain of 11. Running the simulation with an input voltage of 5V provides an output voltage of approximately 55V thus providing a gain of exactly 11.

```
       --- Operating Point ---

V(n001):        4.99995        voltage
V(vin):         5              voltage
V(vout):        54.9995        voltage
V(n002):        30             voltage
V(n003):        -30            voltage
I(Ri):          0.00499994     device_current
I(Rf):          0.00499996     device_current
I(V-):          6.99999e-08    device_current
I(V+):          -5.00001e-08   device_current
I(V3):          -1e-08         device_current
Ix(u1:1):       1e-08          subckt_current
Ix(u1:2):       9.99989e-09    subckt_current
Ix(u1:3):       5.00001e-08    subckt_current
Ix(u1:4):       -6.99999e-08   subckt_current
Ix(u1:5):       -0.00499995    subckt_current
```

Figure 4:Op-Amp Simulation

1.3 Output Data and Testing



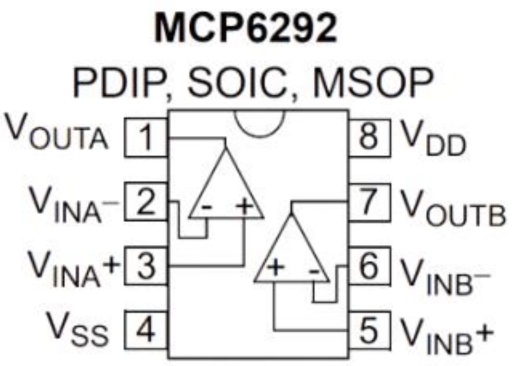
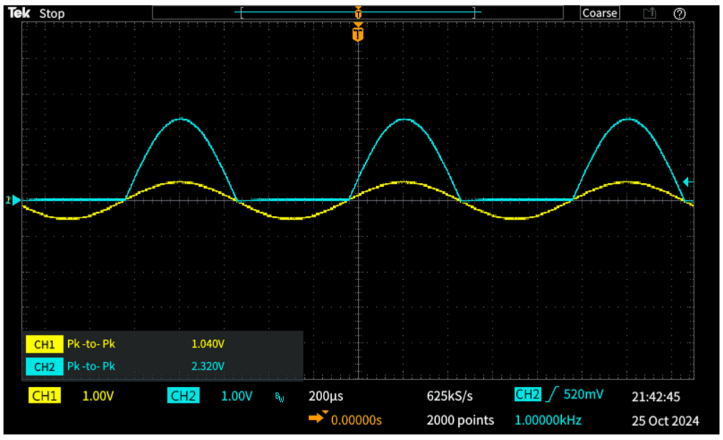


Figure 5.2: MCP6292 Schematic

Figure 5.1: Op-Amp Breadboard Circuit

This breadboard circuit uses two resistors one of which being 1kΩ and the other being 3.3kΩ. According to the gain equation this should produce a voltage gain of 4.3**.** However, the actual gain recorded by the oscilloscope was approximately 2.148V. The discrepancy in the expected and the actual output may be due to the nature of the MCP6292 as a rail-to-rail amplifier. A rail-to-rail amplifier means its maximum is near the positive DC supply voltage and the negative maximum is near the negative DC supply voltage[1]. Due to the fact that the positive DC supply in 3V the MCP6292 cannot surpass 3V voltage output therefore the expected gain of 4.3 cannot be achieved with a voltage of 1V.



```
CH1 Pk-to-Pk   1.040V
CH2 Pk-to-Pk   2.320V
CH1 1.00V   CH2 1.00V   200µs   625kS/s   CH2 ╱ 520mV   21:42:45
                        0.00000s   2000 points   1.00000kHz   25 Oct 2024
```

TBS1052C - 14:31:10 25/10/2024

Figure 6:Op-Amp Output Reading

As shown in Figure 6 the output voltage reading on the oscilloscope seen on channel 2 cannot go below 0 as the maximum negative DC voltage is 0V. In addition, the voltage output seen in the oscilloscope reading doesn't surpass 3V. After several tests I discovered that attempting to force the amplifier to output a gain that makes the voltage output surpass would cause the oscilloscope signal to truncate as the supply voltage causes a bottleneck in the system.

2. Design & Implementation of the H-Bridge Circuit

2.1 Principle of Operation

A H-bridge is a circuit in which you can control the direction of current flow via four switches. By switching the direction of current it would provide us with the ability to switch the rotational direction of the motor. When provided with a positive current the motor will spin anticlockwise and with a negative current the motor spins clockwise. If two opposite switches are turned on, then the left lead of the motor will be connected to the power supply while the other is connected to ground thus causing current to flow through the motor powering it. By pressing the other two switches the reverse situation occurs as the right lead of the motor connects to the power supply and the left lead of the motor connects to ground thus leading to the motor moving in the opposite direction [2]. The use of transistors in the H-Bridge circuit provides efficiency benefits as transistors can switch between conducting and non-conducting states very quickly thus decreasing power loss. Transistors also provide bidirectional current flow therefore allows the motor to have both forward and backward motions. In addition, the low resistance of MOFSET transistors decreases heat generation and power loss further increasing the efficiency of the system.

2.2 Design of the H-Bridge

Before I tested the full H-bridge using one DC Supply I first tested a similar circuit using split supplies in order to achieve the same bi-directional movement of the motor

2.2.1 *Bi-Directional motor with Split Supply*

By using one positive supply and one negative supply we can decrease the number of transistors and switches used in the circuit to achieve the same results.

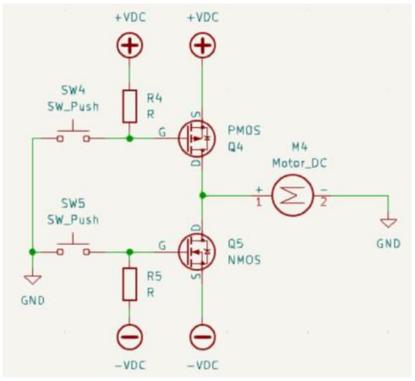


Figure 7:Half-Bridge Schematic

By connecting positive current to the PMOS transistor and negative current to the NMOS transistor allows the motors direction to be controlled via 2 switches.

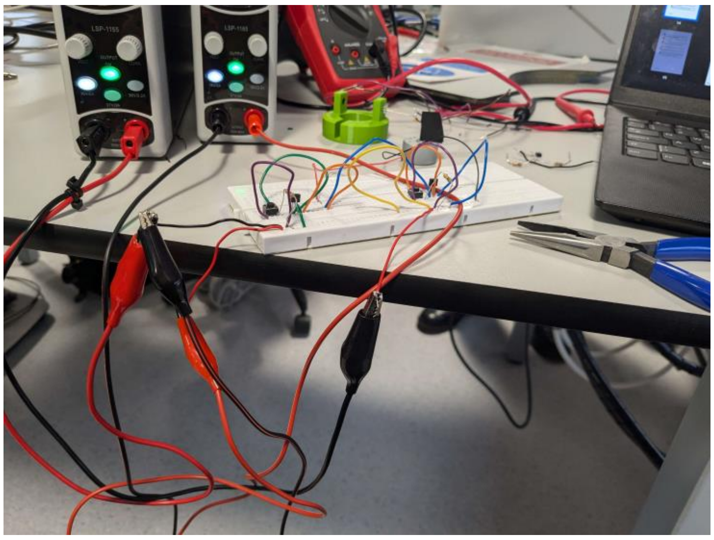


Figure 8:Half-Bridge Circuit

In my breadboard design here, I utilised a split supply. A split supply is a way of connecting power supplies in series with one another. By connecting the positive side of one supply to the negative side of the other the two-power supply will be in series leading to one only outputting positive current while the other only outputting negative current.

2.2.2 Full H-Bridge Design

The full H-bridge circuit is more suited to applications such as an electric vehicle that rely on a single DC source. In addition, using two power supplies provides size constraints and is also inefficient as other components within the system need power from the supply as well.

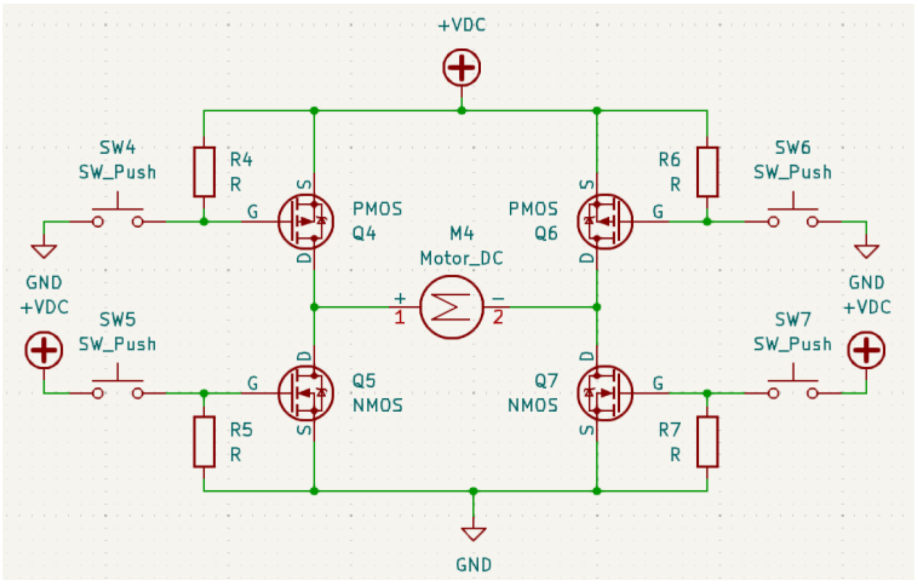


Figure 9:H-Bridge Schematic

As states prior pressing the opposite switches would cause the motor to rotate in one direction. Referring to figure 9, Pressing the SW4 and SW7 switches would produce a negative current though moving the motor clockwise. Pressing SW5 and SW6 would produce a negative current thus moving the motor anticlockwise.

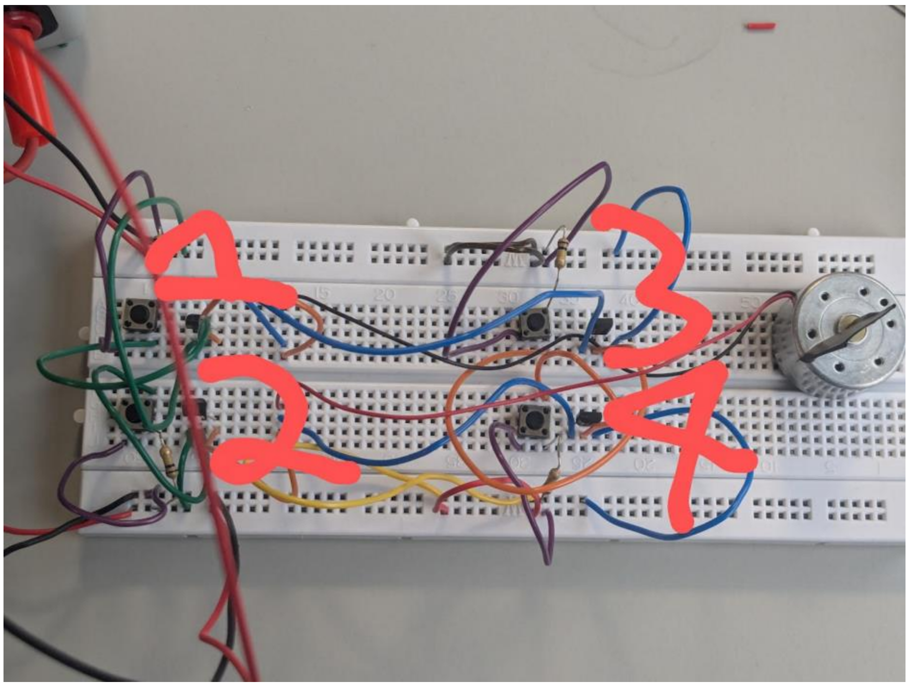


Figure 10:H-Bridge Breadboard layout

In this breadboard layout a switch combination of 1 and 4 would simulate the pressing of the SW4 and SW7 switches in Figure 9:H-Bridge Schematic and the switch combination of 2 and 3 would then simulate the combination of SW5 and SW6.

3. Design & Implementation of the H-Bridge Logic Circuit

3.1 Principle of Operation

The use of logic circuits allows the H-Bridge to be controlled by one switch. The use of multiple switches is inefficient as it could lead to certain errors such as forgetting switch combinations thus turning on transistors on the same side of the H-bridge leading to damage to the transistors. Another issue is time delay as we may not be able to press both switches at the exact same time thereby leading to a time delay which in real applications could be costly. For instance, a small delay in the ability to control a motor could result in safety risks as even a small delay in emergency stop commands could result in a severe risk of injury. In this logic circuit the two control signals will be direction and the switch state. The direction signifies the direction of the motor whether it is clockwise or anticlockwise. The switch state signifies whether the switch is one or off. When the switch is on it will output a high signal of 1 and when its off it will output a low signal of 0. When the direction of the motor is clockwise it will output a low signal of 0 and when its anticlockwise it will output a high signal of 1. Depending on the switch state and direction will dictate which transistor turns off and on thereby enabling control of the entire system with one switch.

Direction	On/Off	Q4	Q5	Q6	Q7	Motor State
0	0	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	0	0	0
1	1	0	1	1	0	1

Figure 11:Logic Truth Table

The variables Q4,Q5,Q6,Q7 are the different transistors in the H-Bridge schematic seen in the Figure 9:H-Bridge Schematic. As the truth table shows the motor state depends on whether or not the switch has been pressed the direction of the motor is purely dependant on which transistors are on. This can be displayed more simply via a K-Map between D and S (switch state)

Motor State	⌐D	D
⌐S	0	0
S	1	1

Figure 12:H-Bridge K-Map

By minimising the K-Map it can be seen that the motor state is purely dependant on the state of the switch. Changing the direction can only happen by changing which transistors are on which can be done via a logic circuit in which an output for the clockwise direction is fed into the Q4 and Q7 transistors and an output for the anticlockwise direction is fed into the Q5 and Q6 direction

3.2 Design of the H-Bridge Logic Circuit

Before connecting logic gates to the H-Bridge for control the logic circuit would have to be made and tested first. Since I was only limited to use universal NAND gates designing the logic circuit was made a lot more difficult. As I stated earlier I intend to create a design where there is two outputs one for the clockwise direction and the other for the anticlockwise direction.

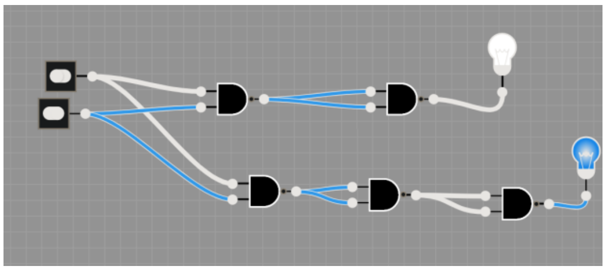


Figure 13:Clockwise Output

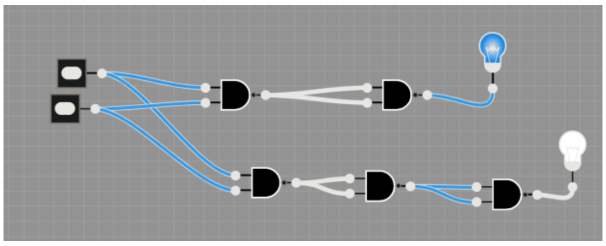


Figure 14:Anticlockwise Output

As shown in the simulations the direction in which the logic circuit moves in depends on the whether the direction input is set to 1 or 0. The switch has to always be 1 as if it's turned off then the motor will not run. As seen in the truth table in Figure 11 The motor state is dependant on the state of the switch.1

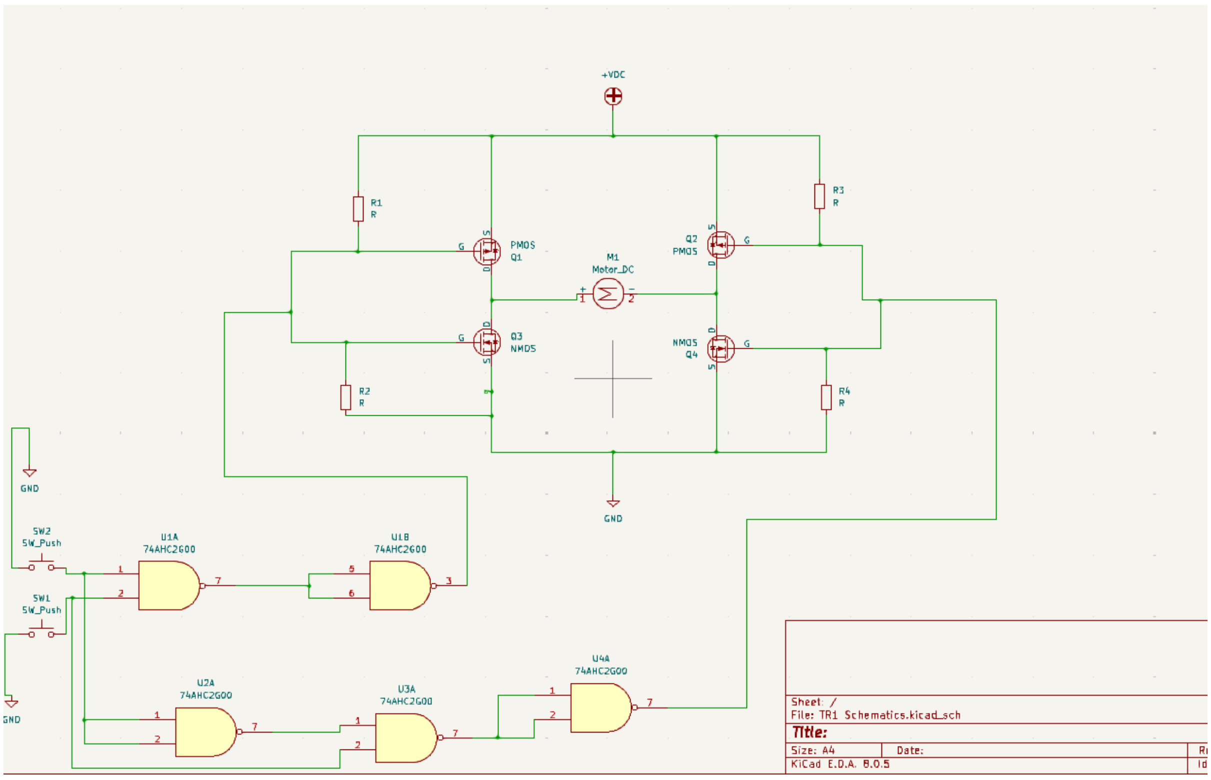


Figure 15:H-Bridge Schematic with Logic

In this schematic the direction input is displayed as a switch as to make it easier to explain and also show the effects of the logic circuit working. As seen in Figure 15 there are two outputs for the logic circuit which decide which direction the motor moves in. If the direction input is set to 0 then U1B would output 0 and U4A would output 1. As a result, an output of 0 would turn on Q1 and turn off Q3 whereas an output of 1 would turn off Q2 and turn on Q4 thus leading to Q1 and Q4 being on at the same time leading to the motor to move in a clockwise direction. If the direction input is set to 1 then U1B would output 1 and U4A would output 0. As a result, an output of 1 would turn off Q1 and turn on Q3 whereas an output of 0 would turn on Q2 and turn off Q4. Q2 and Q4 being on at the same time would cause the motor to turn anticlockwise.

The reason this system works is due to the nature of PMOS and NMOS transistors. A low signal turns on a PMOS transistor but turns off a NMOS transistor and the reverse effect occurs with a high signal. As a result, using a logic circuit in this manner makes sure that both transistors in the same side of the bridge cannot be on at the same time.

4. 10m Challenge Implementation

4.1 Hardware Design

The 10m challenge requires the use of most of the circuitry described in the previous sections including the H-Bridge circuit and using logic circuits to control the direction of the motors. The main micro-processor used for this challenge is the ESP32. It's main benefit on the hardware side is its low power consumption and its array of GPIO pins providing the ability to connect a lot of different pieces of hardware to the board.

4.1.1 Motor Circuit Schematic

There are 3 main aspects of the hardware section of this challenge which is the front and rear chassis alongside the encoders. The rear chassis houses the motors which provide enough power for the bot to move.

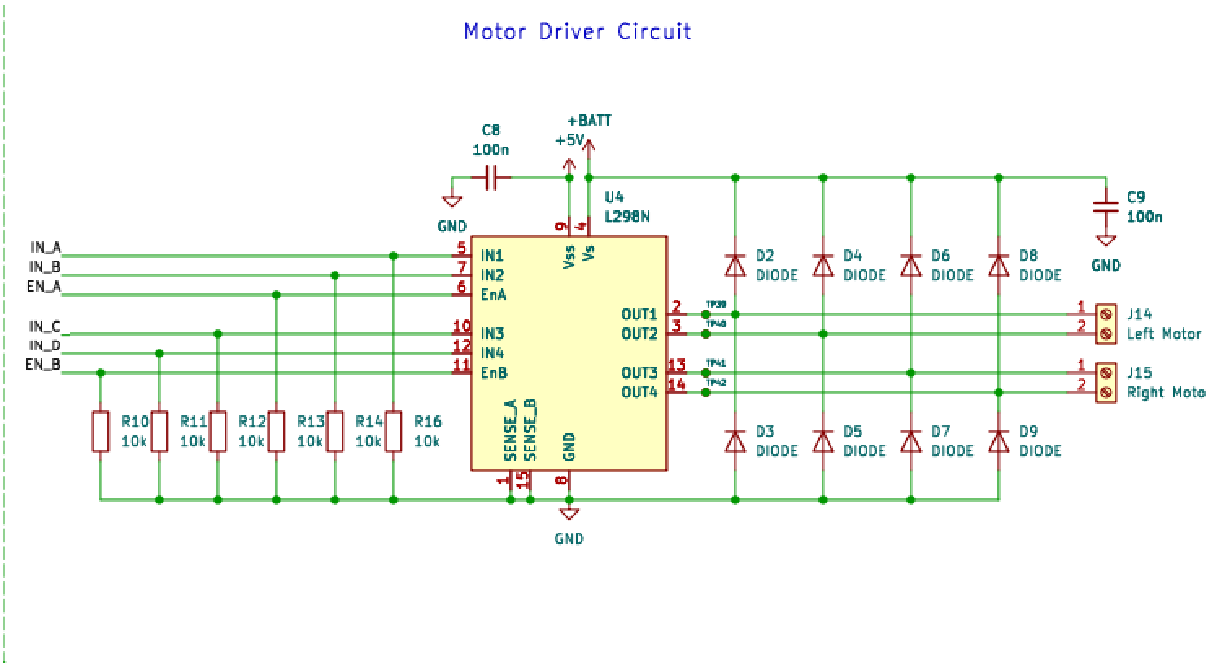


Figure 16:Motor Driver Circuit

This motor driver schematic utilises the same H-bridge circuit design in Figure 9:H-Bridge Schematic however the difference is that two motors are being controlled at once. The L298N is a dual motor driver integrated circuit that utilises both the H-Bridge and the logic circuits to control both motors.

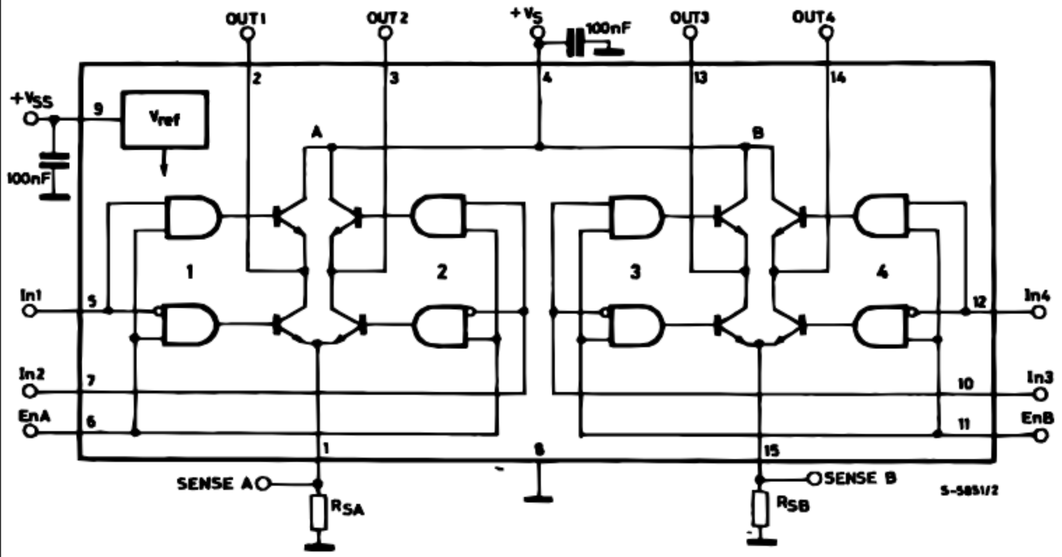


Figure 17:L298N Block Diagram

The block diagram from the L298N datasheet shows that the inputs for the logic circuit come directly from the ESP32 from the pins IN_A, IN_B, IN_C, IN_D. As a result. This allows the ESP32 to control what pin is set to high or low thus allowing for directional control of both motors. The utilisation of diodes in Figure 16 prevents current moving in the opposite direction than intended. If that was to happen, then the motor would move in the opposite direction as intended. Since diodes only allow current flow in one direction it ensures that whatever inputs the logic circuit within the L298N receives the motor will move in that direction.

The Encoders

An encoder is a piece of hardware that translates mechanical motion into electrical signals. The main use case of the encoders is to provide accurate positive feedback on the motor system. The assumption up until now is that each motor is the exact same however that is not the reality of the situation. One motor may have more power than the other for a variety of reasons. Therefore, the encoders are used to speed control and to accurately measure the difference in speed between both motors. By converting the mechanical motion to electrical signals, the differences can be seen in an oscilloscope. By understanding the power differences between each motor, we can set the speed of the motors to different values to balance out the system thus making sure one motor isn't overpowering the other thereby leading to the bot leaning to one direction

4.1.2 The Servos

A servo motor is a highly specialised motor designed for precise control of rotary or linear motion. It's a rotational motor that utilises feedback to ensure exact positioning and typically employs a control mechanism to dictate the motors movement [3].

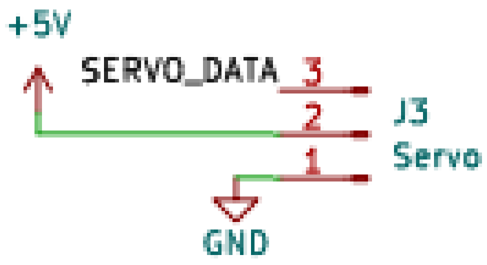


Figure 18:Servo Pin Schematic

A common mistake when connecting a servo into a PCB would be connecting the pins in the wrong orientation. A mistake like this has a real chance of frying the motor as a voltage of 5V would be passed through the wrong wire. The servo has 3 pins, The power pin (pin 2) the ground pin (pin 1) and the signal pin (pin 3). Inputting 5V into pin 3 for example would fry that pin as it isn't rated for 5V thus leading to the servo unable to communicate with the ESP32

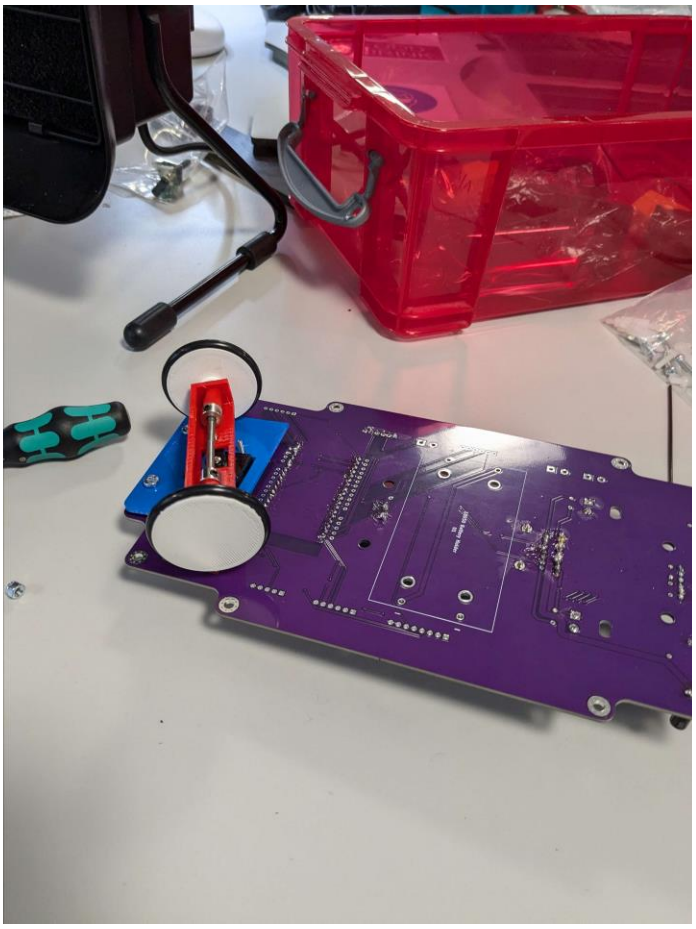


Figure 19:Front Chassis Design

The function of the servo in this case is to rotate the two smaller wheels. The servo is better suited for this purpose instead of another set of motors due to its ability to rotate up to 180° and it's low power consumption as driving a motor to move two small wheels would be very inefficient. The two smaller wheels attached to the front chassis is used for steering. Despite the 10m challenge being linear steering is also vital as it could provide support in case the bot starts moving left or right due to any imbalances in the motor system. A potential issue with the servo is that it may not be centred correctly thus leading to issues when it comes to software development. As such, a small piece of software is required to centre the servo.

```
#include <ESP32Servo.h>


Servo steeringServo;


int steeringAngle = 90;    // variable to store the servo position

int servoPin = 13;  // the servo is attached to IO_13 on the ESP32


void setup() {

	// allow allocation of all timers

	ESP32PWM::allocateTimer(0);

	ESP32PWM::allocateTimer(1);

	ESP32PWM::allocateTimer(2);

	ESP32PWM::allocateTimer(3);

	steeringServo.setPeriodHertz(50);    // standard 50Hz servo

	steeringServo.attach(servoPin, 1000, 2000);   // attaches the servo to the
pin using the default min/max pulse widths of 1000us and 2000us


  steeringServo.write(steeringAngle);

}
```

Figure 20:Servo Re-Centre Program

This program initially sets the variable “steeringAngle” to 90 which is the centre angle of the servo. Then sets the servoPin to 13 which would be the GPIO pin the servo is connected to on the ESP32. The function “steeringServo.setPeriodHertz(50)” sets the servos frequency to 50Hz which is the default frequency for the servo. Then the function “steeringServo.attach(servoPin, 1000, 2000)” attaches the servo pin to the ESP32 thus allowing the software to control it. The most important part is the function “steeringServo.write(steeringAngle)” which sets the servo position to the value of the variable “steeringAngle” which as seen in the code snippet is set to 90.

4.2 Software Development

Another major benefit of the ESP32 is its interaction with the Arduino IDE software. The integration of Arduino software into the ESP32 allows for simple software control of the two major components of the bot which are the motors and the servos.

4.2.1 Motor Control

As shown in Figure 17 the motor driver IC takes inputs from the ESP32 in order to control motor direction. The inputs can be set as HIGH or LOW depending on whether you want the motors to move forwards, backwards or rotate clockwise and anticlockwise. However, before setting up any inputs the motor would have to be initialised first. The motor initialisation is done by defining which of the GPIO pins the motors are connected to.

```
#define INa 26  direction

#define INb 27  direction

#define INc 14

#define INd 12
```

Figure 21:Motor Pin Setup

Then by using the pinMode function each GPIO pin can be initialised to receive a HIGH or LOW output which would allow us to control the direction of the motor.

```
pinMode(INa, OUTPUT);

 pinMode(INb, OUTPUT);

 pinMode(INc, OUTPUT);

 pinMode(INd, OUTPUT);
```

Figure 22:Motor Initialisation

Once the motor is setup we can then use the loop function in the Arduino IDE and set the motor to continuously move in a specific way. In the context of the 10m challenge we want the motors to only move forwards. Prior to that, the speed of the motors have to be set. As stated in the Motor Circuit Schematic each motor has to be set to different speeds due to the differences in power between them. The accurate way of doing this would be the usage of the encoders however due to time limitations I was unable to do this. My implementation of this was through trial and error which is very inaccurate.

```
int leftSpeed = 95;

 int rightSpeed = 254;
```

The speeds I chose was 95 for the left motor and 254 for the right motor. The speeds of the motor are controlled via a PWM interface in which the range of speeds is between 0 and 255. The large disparity between the speeds of the motors was due to the left motor constantly overpowering the right motor. When set to the exact same speeds the bot would move towards the right meaning the left motor was overpowering the right motor. In my first attempt at the 10m challenge the speeds were 150 and 250 for the left and right motor respectively. During the challenge the bot immediately went towards the right and only travelled about 1-2m linearly. By changing the speeds a bit I was able to achieve up to 3-4m linearly with other tweaks to the software in other areas enabling me to achieve more linear motion and get closer to the 10m goal.

Now the speeds are set the loop function that tells the motor the direction to move in has to also be set.