OBJECTIVE

With the increasing number of road accidents and fatalities, it is foreseen that strict enforcements are nothing but rules limited to pen and paper. Autonomous vehicle can solve the problem to some extent as a world full of autonomous vehicles which can cross talk with one another and at the same time create a safe path for us to commute to our destination. Here the objective is to:

- To create an autonomous vehicle that traces/detects the lanes of a road using a camera attached to it, processes the image captured and control the steering of the vehicle (so that the vehicle always auto-correct its path to the middle of the road).
- To stop in its path whenever it detects an obstacle and continue only when that obstacle is removed.
- To detect the end of the road and perform a U-Turn task that takes it to the other end of the road.

RASPBERRY PI 3 PIN DESCRIPTION

PIN GROUP	PIN NAME	DESCRIPTION
POWER SOURCE	+5V, +3.3V, GND and Vin	+5V -power output +3.3V -power output GND – GROUND pin
COMMUNICATION INTERFACE	UART Interface (RXD, TXD) [(GPIO15, GPIO14)]	UART (Universal Asynchronous Receiver Transmitter) used for interfacing sensors and other devices.
	SPI Interface (MOSI, MISO, CLK, CE) x 2 [SPI0-(GPIO10, GPIO9, GPIO11, GPIO8)] [SPI1--(GPIO20, GPIO19, GPIO21, GPIO7)]	SPI (Serial Peripheral Interface) used for communicating with other boards or peripherals.
	TWI Interface (SDA, SCL) x 2 , [(GPIO2, GPIO3)] [(ID_SD,ID_SC)]	TWI (Two Wire Interface) Interface can be used to connect peripherals.
INPUT OUTPUT PINS	26 I/O	Although these some pins have multiple functions they can be considered as I/O pins.
PWM	Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19	These 4 channels can provide PWM (Pulse Width Modulation) outputs. *Software PWM available on all pins
EXTERNAL INTERRUPTS	All I/O	In the board all I/O pins can be used as Interrupts

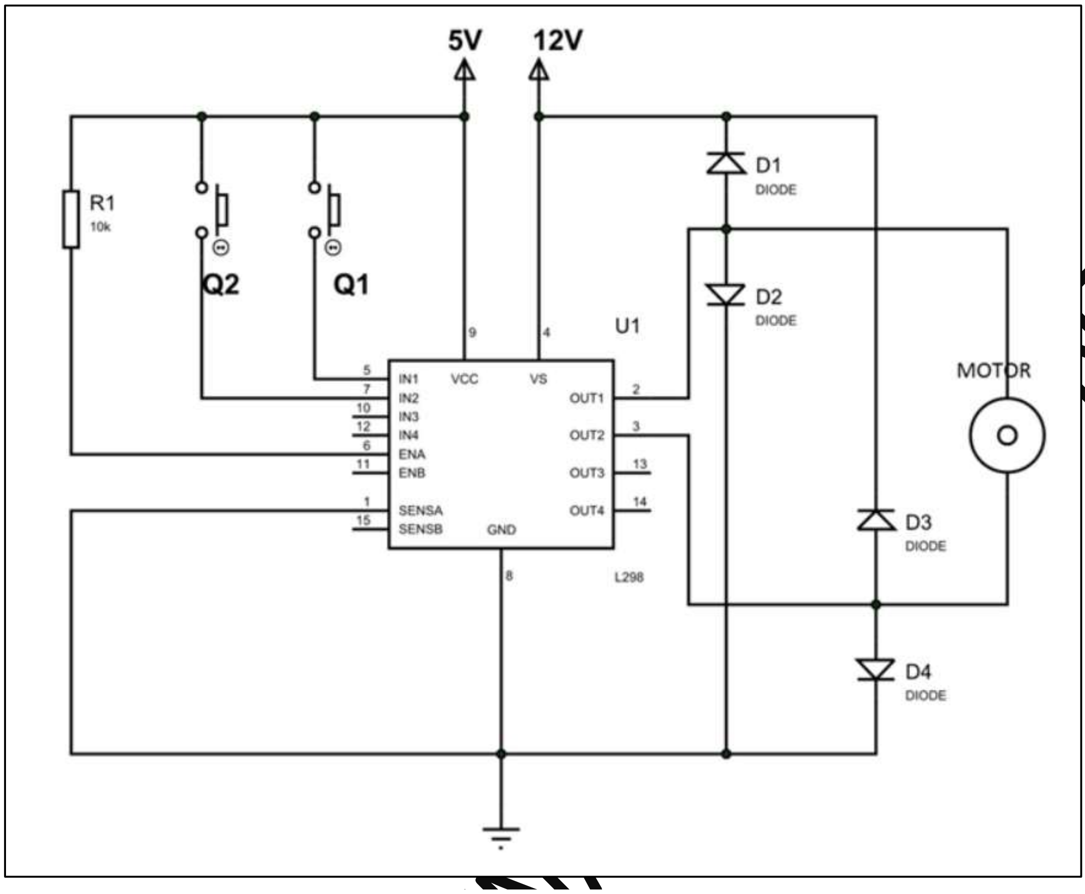
ARDUINO UNO

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits.[1] The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.

The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases.[4] The ATmega328 on the board comes preprogrammed with a bootloader that allows uploading new code to it without the use of an external hardware programmer.

The 14 digital input/output pins can be used as input or output pins by using pinMode(), digitalRead() and digitalWrite() functions in arduino programming. Each pin operate at 5V and can provide or receive a maximum of 40mA current, and has an internal pull-up resistor of 20-50 KOhms which are disconnected by default. Out of these 14 pins, some pins have specific functions as listed below:

- **Serial Pins 0 (Rx) and 1 (Tx):** Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serial chip.
- **External Interrupt Pins 2 and 3:** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM Pins 3, 5, 6, 9 and 11:** These pins provide an 8-bit PWM output by using analogWrite() function.
- **SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK):** These pins are used for SPI communication.
- **In-built LED Pin 13:** This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.

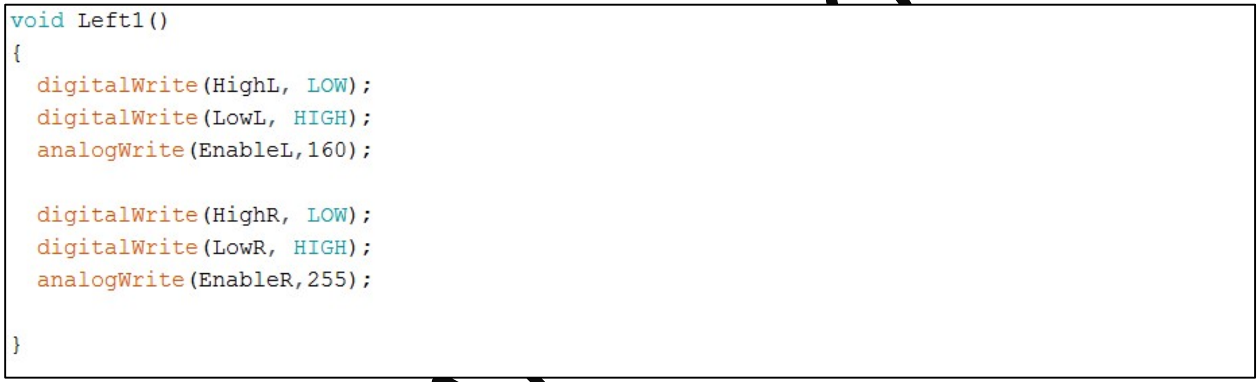


L298 CIRCUIT DIAGRAM

INPUTS	**FUNCTION**
Q1=HIGH, Q2=LOW	Forward current
Q1=LOW,Q2=HIGH	Reverse current
Q1=Q2	Fast MOTOR stop

Right Motor High	Pin 5 (Input 5 Pin)	9
Right Motor Low	Pin 6 (Left Enable Pin B)	10

Thus, by following the above table we can successfully interface the L298N Motor Driver with the Arduino Uno. In the Arduino IDE we have set three speed preferences, i.e. Level 1 Right turn, Level 2 Right turn and Level 3 Right turn. Similarly, for the simulation of the Left turn we use three levels of left turn i.e. Level 1 Left turn, Level 2 Left turn and Level 3 Left turn. A U-Turn system code is also written in the Arduino IDE to facilitate the U-Turn when the autonomous bot reaches the lane end. The following are the code snippets of the code for the above:

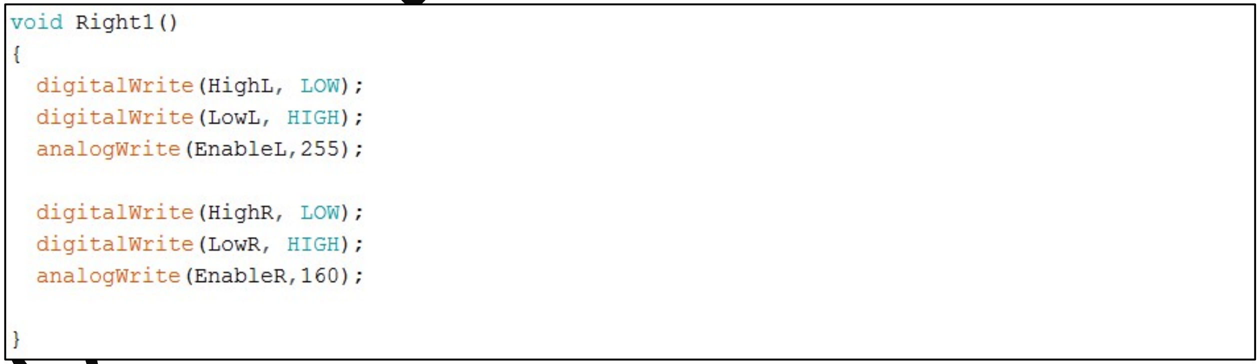


```
void Left1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,160);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,255);

}
```

The Above is the Code for Left1 turn



```
void Right1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,160);

}
```

The Above is the code for Right1 turn

Figure: Matplotlib Graph

Now using the Matplotlib Function we are going to see what is the area we are interested in for the processing of the image. The horizontal is considered as the width of the image while the vertical is called the length of the image.

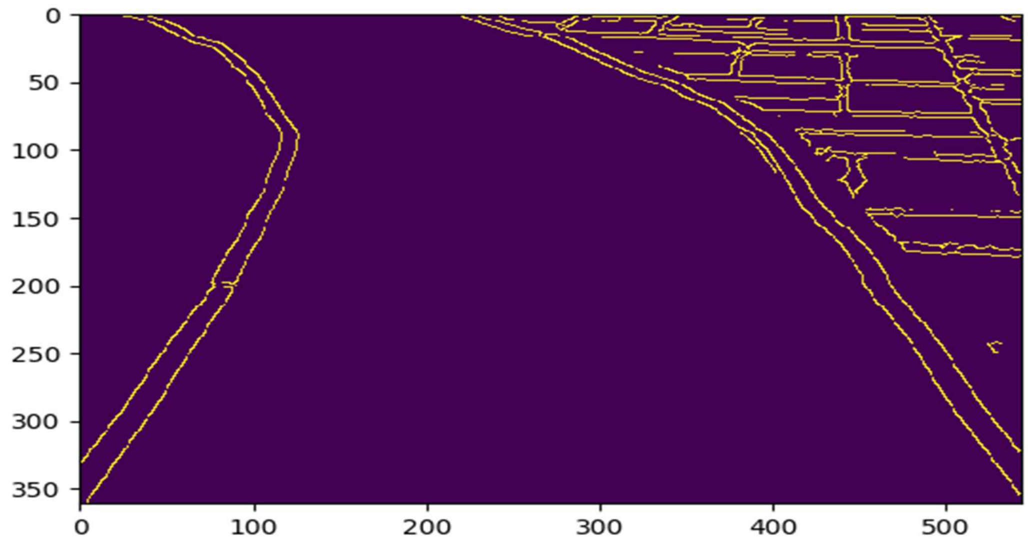


Figure: Matplotlib Graph

We will consider four points on the graph where our region of interest will be there as it is shown in the image below.

Canny Edge Detection

The **Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

The Canny edge detection algorithm is composed of 5 steps:

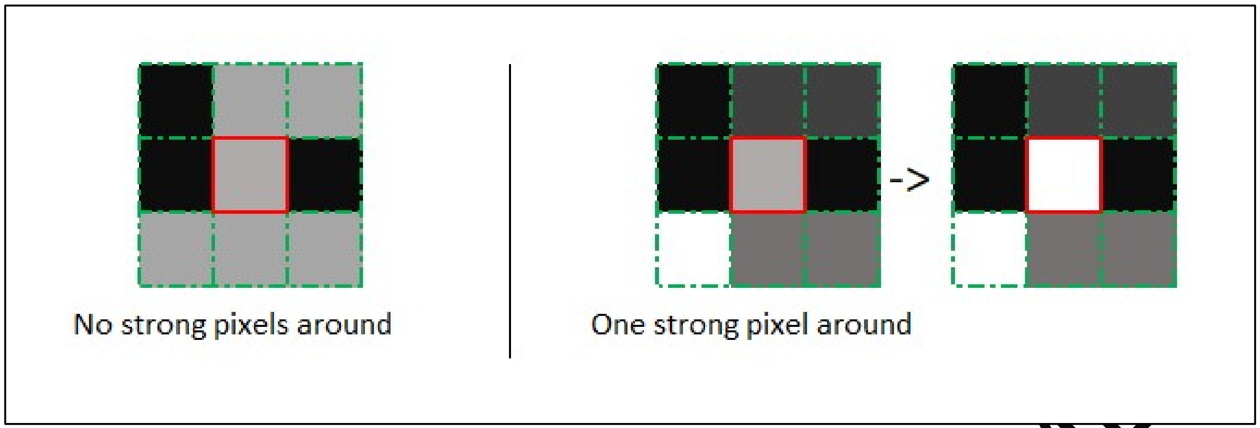
1. Noise reduction;
2. Gradient calculation;
3. Non-maximum suppression;
4. Double threshold;
5. Edge Tracking by Hysteresis.

The algorithm is based on grayscale pictures. Therefore, the pre-requisite is to convert the image to grayscale before following the above-mentioned steps.

Noise Reduction

Since the mathematics involved behind the scene are mainly based on derivatives (cf. Step 2: Gradient calculation), edge detection results are highly sensitive to image noise.

One way to get rid of the noise on the image, is by applying Gaussian blur to smooth it. To do so, image convolution technique is applied with a Gaussian Kernel (3x3, 5x5, 7x7 etc…). The kernel size depends on the expected blurring effect. Basically, the smallest the kernel, the less visible is the blur. In our example, we will use a 5 by 5 Gaussian kernel.



After all these processes are executed using the OpenCV function in the Geany Programming Editor, we get Canny Edge of the Lane Image. It is shown below:

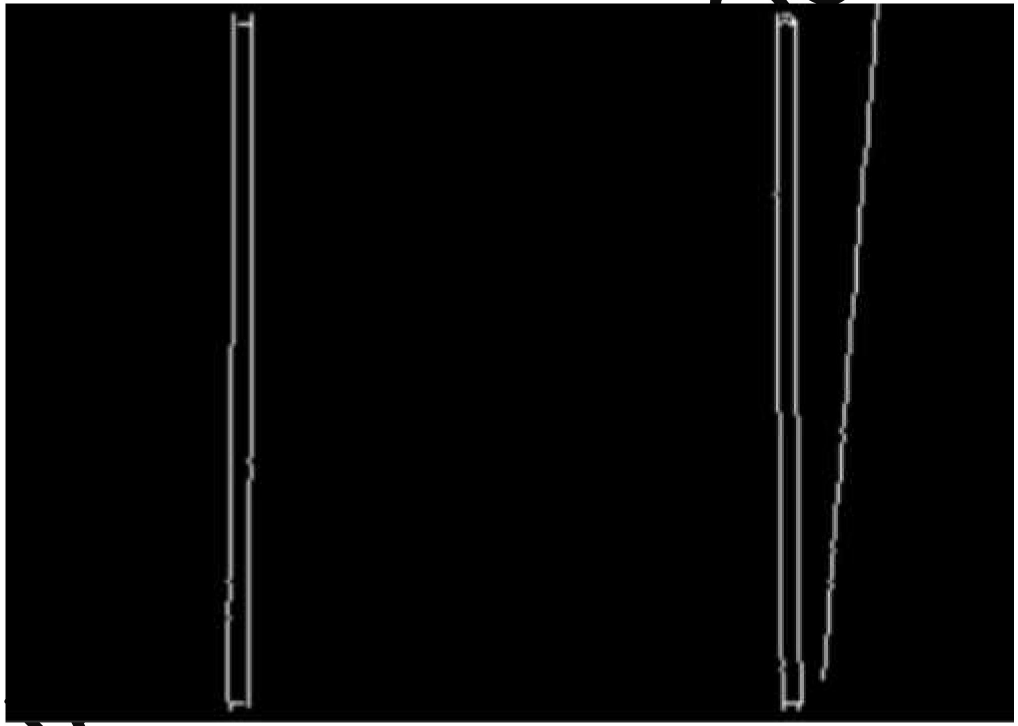


Figure: Canny Edge Detection of the Lanes

Further more the processing is done to reduce the noise which finally results in the image below:

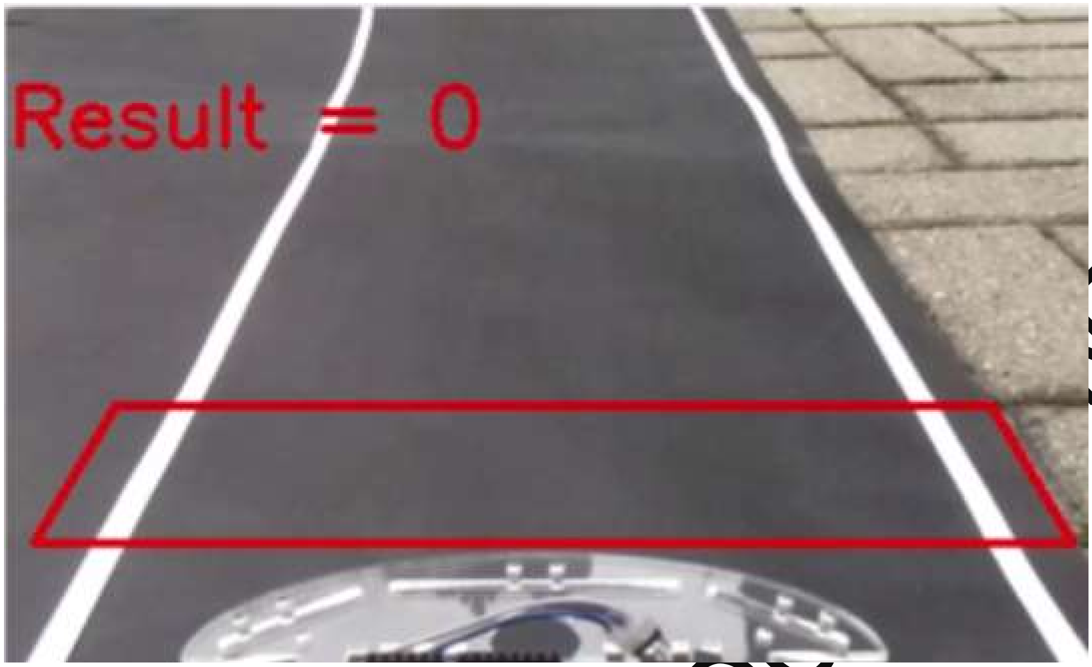


Figure: String displaying the Result Function

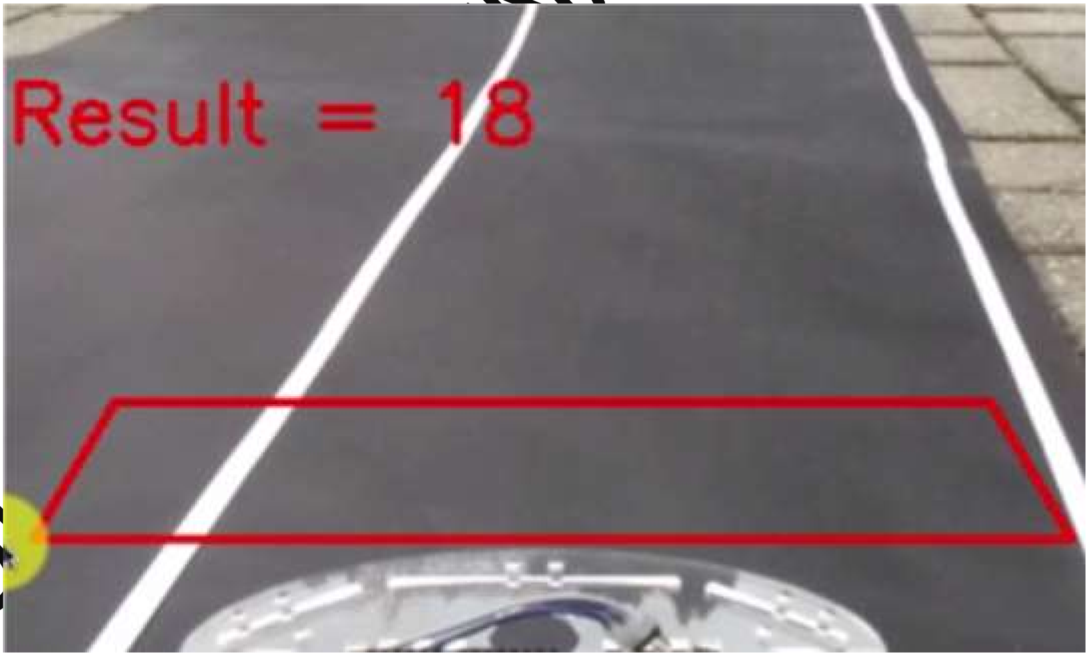


Figure: String Showing the Difference when moved left