# Ultrasonic Self-Driving Car

Daisy Barrette
Supervisors: David Dove and Dr. Farhana Zulkernine
School of Computing, Queen's University

CISC499
2018

## Abstract

Autonomous cars can reduce accidents caused by driver error, and provide increased mobility for people who cannot drive or choose not to drive. This project explores the practical challenges of building a small self-driving prototype with cheap, commercially available hardware.

## Objectives

- Find off-the-shelf hardware to build the car
- Develop a machine learning model to make the car able to learn how to follow a track

## Implementation

**Hardware:**
- Controlled with an Arduino Uno microcontroller
- Propelled by two independant DC wheel motors
- Three Ultrasonic detect objects in the car's path, send data to the Arduino

**Software:**
- Data is sent serially from the Arduino to the Raspberry Pi
- The neural network runs on the Pi, classifies data as one of a set of possible turning directions
- The Pi sends a command back through the serial port, to tell the Arduino which way to turn the car

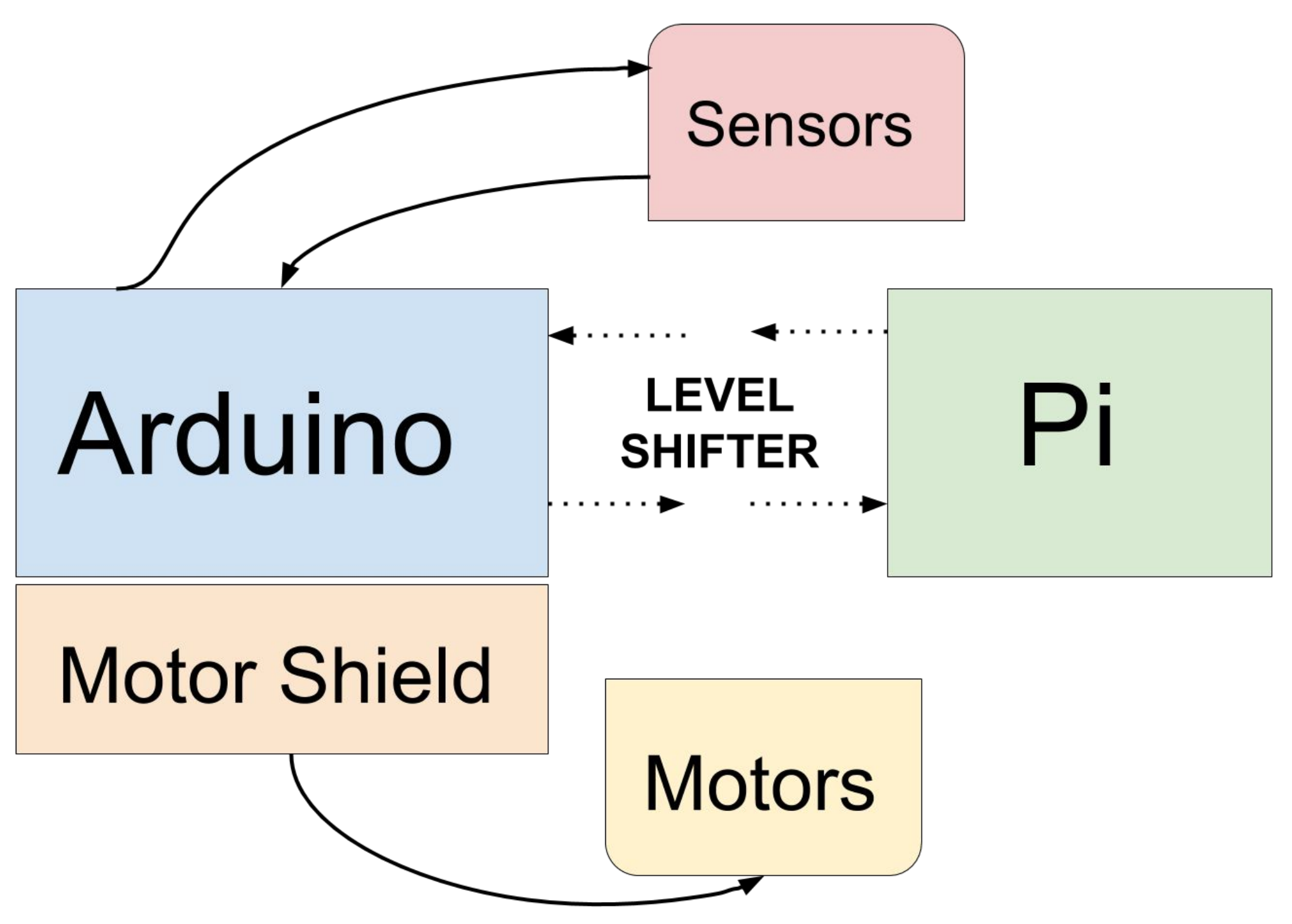

Fig. 1: Diagram showing the relationships between the main hardware components. Dashed lines indicate connection through a serial port.

- The Arduino Uno and HC-SR04 ultrasonic sensors were chosen because of their low cost and ease of use
- The speed of each motor can be controlled individually; to turn, the vehicle rotates one motor faster than the other
- Serial communication is done by connecting the TX and RX pins on the Arduino to a logic level-shifter, which is in turn connected to the Raspberry Pi's GPIO pins. This connection allows real-time communication between two devices

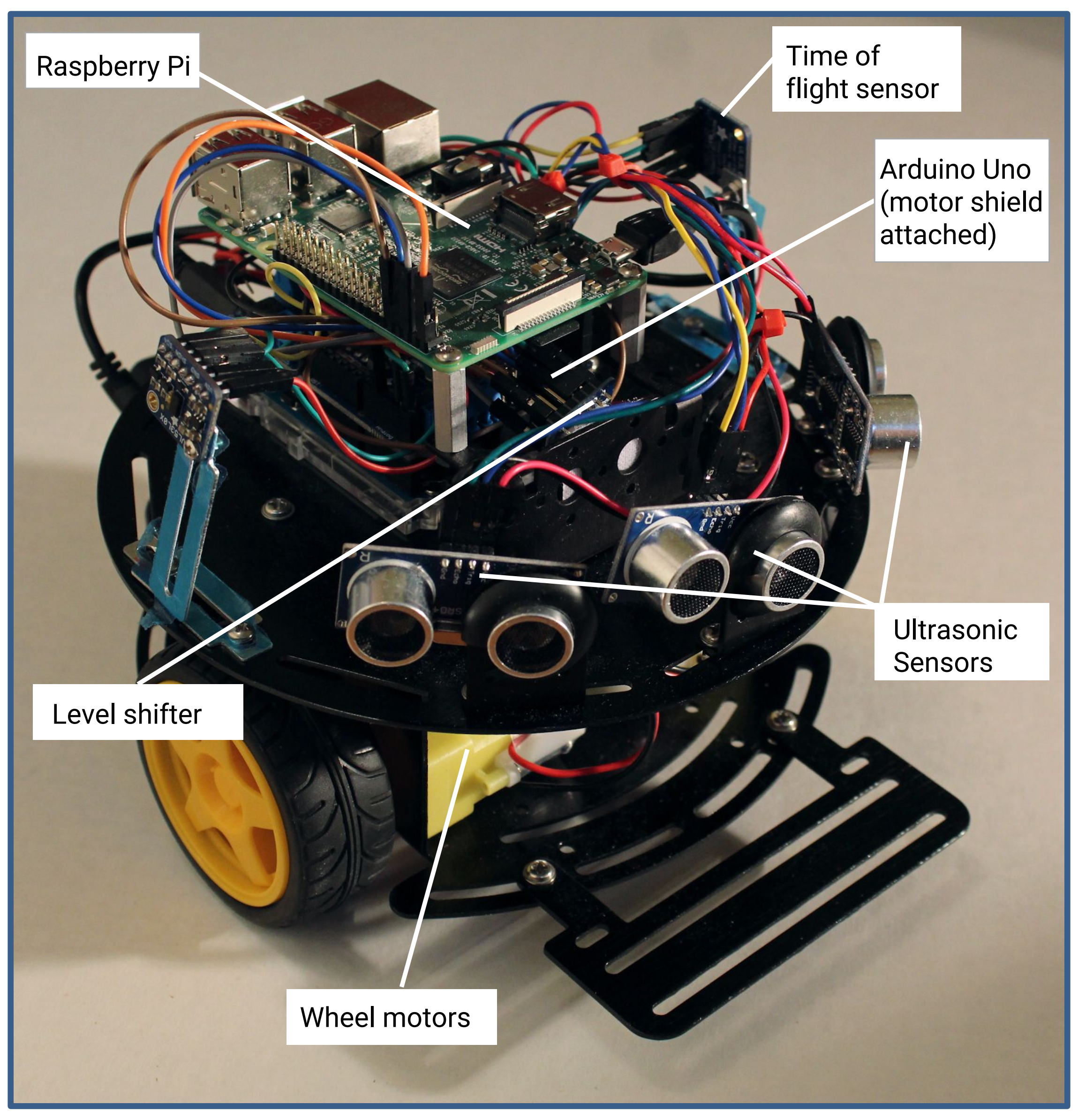

Fig. 2: The car as seen from the front

- The data from the time of flight sensors (visible in Figure 2) was less reliable than the other sensors, and so was not used in the final implementation
- Figure 3 represents the code running on the Arduino. It takes input from the sensors, outputs it to the Raspberry Pi, and then takes input from the Raspberry Pi on which direction to turn
- The car can also be driven directly by a user, by sending commands to the Raspberry Pi through SSH
- To collect data for training the network, the car was placed on a walled track; data was collected with the car in several different positions on the track, and used to train the neural network

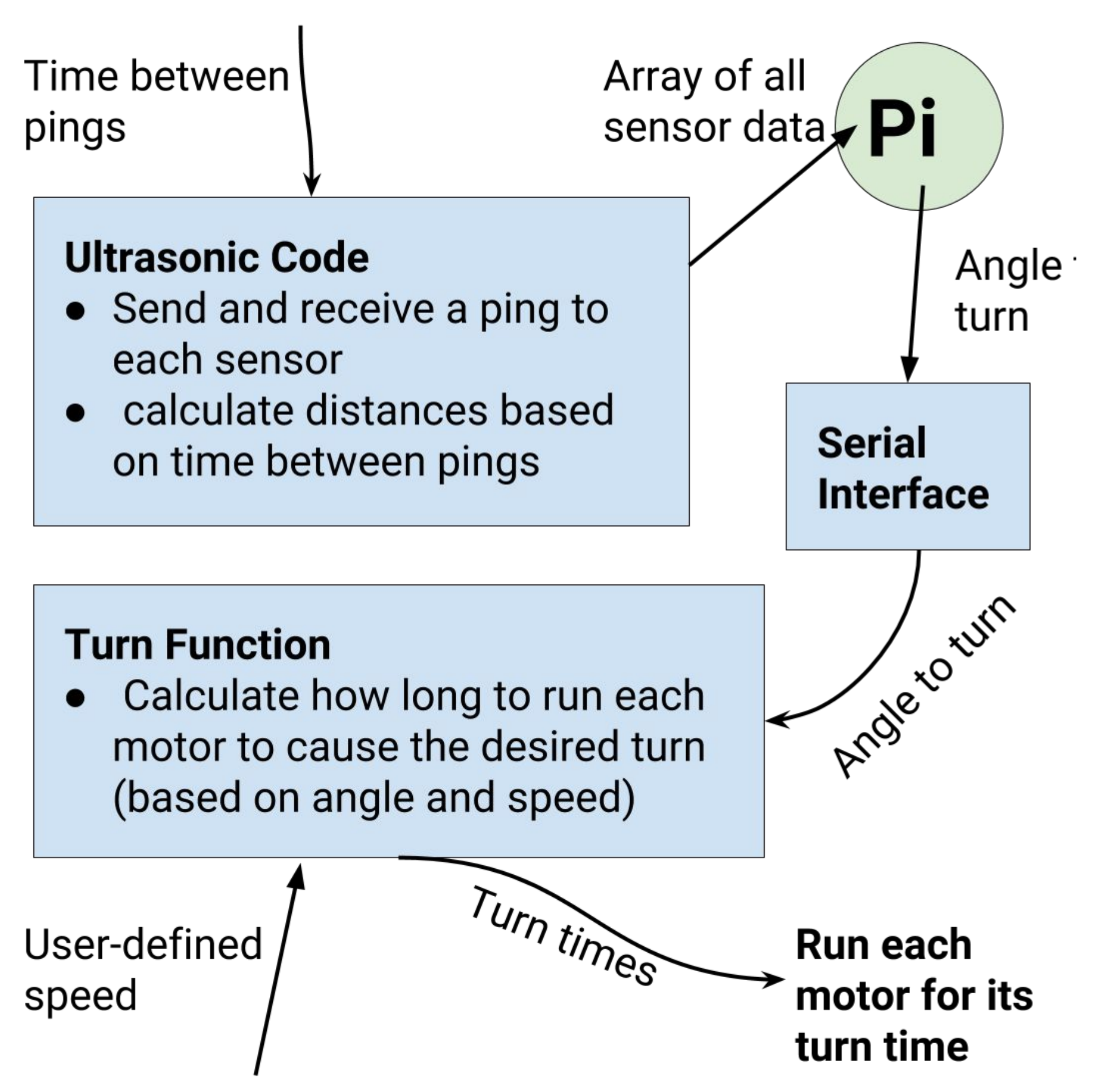
Time between pings
Array of all sensor data
Pi
Angle to turn
Ultrasonic Code
Send and receive a ping to each sensor
calculate distances based on time between pings
Serial Interface
Angle to turn
Turn Function
Calculate how long to run each motor to cause the desired turn (based on angle and speed)
User-defined speed
Turn times
Run each motor for its turn time

Fig. 3: Diagram showing the connections between the main software components

## Neural Network

- The neural network is a multilayer perceptron, implemented in Python with scikit-learn
- The network was trained using supervised learning, with data collected from the car's sensors
- Takes the data from the ultrasonic sensors as input, and classifies this input into a discrete class.
- Each class corresponds to an angle that the car should turn (including 0°, if the car should not turn at all).

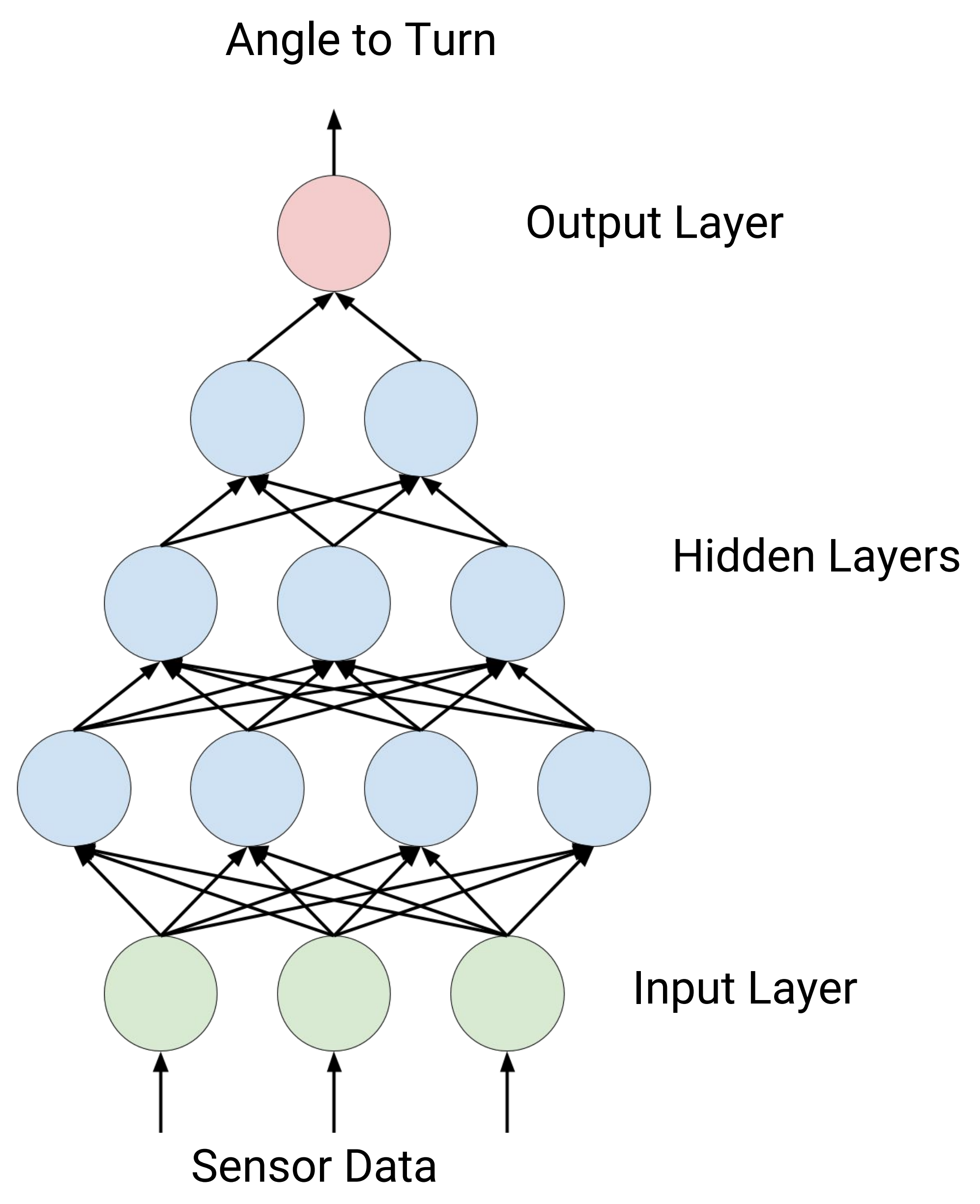

Fig. 4: The structure of the neural network

| Class | Left Sensor | Centre Sensor | Right Sensor |
|---|---|---|---|
| Turn 0° | 119cm | 172cm | 122cm |
| Turn 0° | 31cm | 135cm | 32cm |
| Turn Right | 16cm | 34cm | 62cm |
| Turn Right | 28cm | 263cm | 64cm |
| Turn Left | 160cm | 96cm | 21cm |
| Turn Left | 301cm | 35cm | 22cm |

Fig. 5: Example sensor data from each of the three classes

## Challenges

- Size of the car; the first model chosen was too small to hold all necessary sensors
- Noise in the sensor data made it difficult to train the network to predict turns accurately

## Future Work

- Add more output classes to the neural network, or switching to a more efficient model
- Add more sensors to gather more data on the car's environment, to make better turning decisions
- Apply normalization techniques to reduce noise in the data

## References

[1] A.Deepak, M.Sriramprasath, R. Lokesh, M. Sarath Kumar. "Automotive Collision Avoidance System." International Journal of Modern Engineering Research, pp. 106–109.

[2] Maksimovic et al. "Raspberry Pi as Internet of Things hardware: Performances and Constraints" (2015). ELI1.6.

[3] S.Raju, K.Sanjay, T.Sathish Kumar, B.Madhini. "Semi Autonomous Vehicle To Prevent Accident." International Journal Of Technology Enhancements And Emerging Engineering Research, vol. 2.