Development of a Vehicle for Driving with Convolutional Neural Network

Arbnor Pajaziti[1], Xhevahir Bajrami*[2], Fatjon Beqa[3]
Dept. of Mechatronics
Faculty of Mechanical Engineering
University of Prishtina, Prishtina, Kosovo

Blendi Gashi[4]
Faculty of Computer Sciences
University of Prizren
Prizren, Kosovo

***Abstract*—The aim of this paper is the design, simulation, construction and programming of the autonomous vehicle, capable of obstacle avoidance, object tracking also image and video processing. The vehicle will use a built-in camera for evaluating and navigating the terrain, a six-axis accelerometer and gyro for calculating angular velocities and accelerations, Arduino for interfacing with motors as well as with Raspberry Pi which is the main on-board computer. The design of the vehicle is performed in Autodesk Fusion 360. Most of the mechanical parts have been 3D printed. In order to control the chassis of the vehicle through the microcontrollers, the development of the PCB was required. On top of this, a camera has been added to the vehicle, in order to achieve obstacle avoidance and perform object tracking. The video processing required to achieve these goals is done by using OpenCV and Convolutional Neural Network. Among other objectives of this paper is the detection of traffic signs. The application of the Convolutional Neural Network algorithm after some of the examinations made has shown greater precision in recognizing STOP traffic sign of different positions and occlusion ratios, and finding the path for the fastest time.**

***Keywords*—Image processing; traffic sign; object tracking; autonomous vehicle; convolutional neural network**

I. Introduction

In recent years, autonomous vehicles have become of great interest to the research and industrial communities. The literature related to autonomous vehicles takes mainly two directions hardware and software developments. For the full functionality of autonomous vehicles, there is no need for more hardware. Software and testing is where much work needs to be done. Many researchers have been oriented in designing the small vehicle prototypes due to cost effectiveness and reduction of software testing with different sensors and algorithms.

The Raspberry Pi as a processing chip has been used to build a monocular vision autonomous car prototype. An HD camera along with an ultrasonic sensor has been used to provide necessary data from the real world to the car [1].

A method for autonomous control and decision making and reporting system, the types of mini robots contains self-neural schema framework for autonomous control has been proposed by [2].

The electronic design and motion planning of a robot based on decision making regarding its straight motion and precise turn using Artificial Neural Network (ANN) has been proposed in [3, 8]. The ANN helps in learning of robot so that it performs motion autonomously. The weights calculated are implemented in microcontroller [3].

Obstacle avoiding technique is very useful in real life, by changing the IR sensor by a kinetic sensor, which is on type of microwave sensor whose sensing range is very high and the output of this sensor vary in according to the object position changes [4].

The purpose of this paper is to build a vehicle that will be driven autonomously through decisions taken by Artificial Intelligence. In more detail, the objectives are:

- Design of the vehicle, and its construction;
- Design and development of the PCB board that will control the vehicle;
- Development of a communication method between the vehicle and the laptop;
- Development of the electronic circuits and program for serial communication between Arduino and Raspberry Pi;
- Creating a server for transmitting video and sensors data from Raspberry Pi to the laptop;
- Detection of traffic signs;
- Stopping the vehicle if there are any obstacles ahead;
- Building a Convolutional Neural Networks (CNN) model for predicting the movement.

Aim of this paper is to get closer in trend with today's companies that produce autonomous cars, one of which is well known, Tesla. But there are also many companies that are working to release autonomous cars on our roads.

The paper is organized in 8 sections, starting with mechanical design, electric circuits, controllers programming, training the models for the detection of traffic signs, programming and control with CNN, data processing method, and, concluding remarks with future work.

*Corresponding Author.

II. Mechanical Design and 3D Modelling by using the Autodesk Fusion 360

For modelling and designing of the parts, the Autodesk software, Fusion 360, has been used. Initially, motors and wheels have been designed by measuring the actual physical parts. Then, the assembly of the motors with wheels and the respective joints have been done. The chassis has been designed in seven different parts, because the printing area of 3D printer used was 200x200 mm, and the dimensions of the model exceed this area. Following this step, the motors and wheels have been assembled into chassis. Other parts needed for the implementation of this project have been designed, and are shown below on Fig. 1.

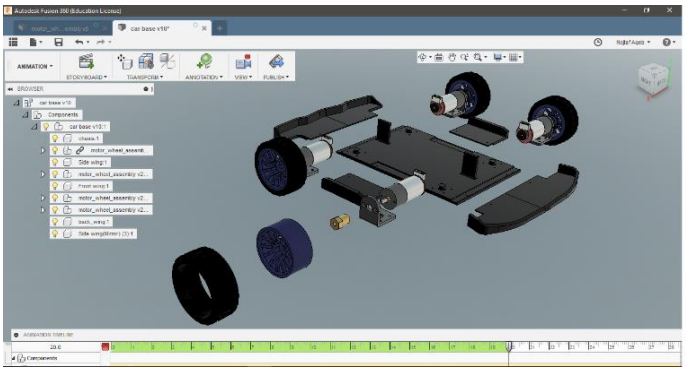

Fig. 1. A View of the Design Process by using the Fusion 360.

A. Assembly of Mechanical Parts

Initially, the motors and wheels were joint together. Then, the printed parts have been glued together and the chassis was completed. After completing the chassis, the next step was to mount the motors and wheel into chassis. This was done using M2.5x6 bolts, and also an adhesive for an additional safety factor.

III. Design of the Electric Circuits of the Vehicle

In order for the mechanic parts to be controlled through electronics, a PCB board was required. Also taking into account the numerous numbers of necessary components, PCB implementation was a good step in eliminating parasitic resistance, as well as the interconnectivity of the components being fixed so as to provide better performance.

Also, the development of PCB board significantly reduced the needed number of wires, compared to the development of the circuit in the breadboard, thus having smaller dimensions and better visual. PCB has been developed as a modular, in case if one of the components is damaged, for various reasons, it can be easily replaced by another.

For designing the circuit, Proteus 8 Professional software has been used. Since the software did not have the component libraries that were used, each component had to be manually measured and added to the program libraries, then the circuit was designed. In the PCB board, components have been placed and then soldered, Fig. 2.

The PCB board components are: 2x boost convertor-XL6009, 2xTP4056, switch, battery terminal, MPU6050, LED indicator, resistors, 3x 2N2222 transistors for RGB, boost-converter for Arduinos, 2x Arduino Pro Mini, buzzer, power amplifier for RGB lights, voltage divider, and distances for Raspberry Pi [5].

It's worth to mention that the L298N motor drivers are not placed on the board but are connected to it, due to their large dimensions.

A. Development of the Transmitter Box

For laptop and Arduino to communicate, the chosen method was through the Radio Modules, in this case through NRF24L01. For the implementation of the transmission box, Fig. 3, a 3D box has been designed and printed initially, and an Arduino Pro Mini, and a NRF24L01 radio module have been installed.

Since the Arduino Pro Mini could not be programmed directly from the laptop, it was necessary to use an FTDI.

The NRFL01+ module with antenna is capable for providing communication up to 1km in open terrain.

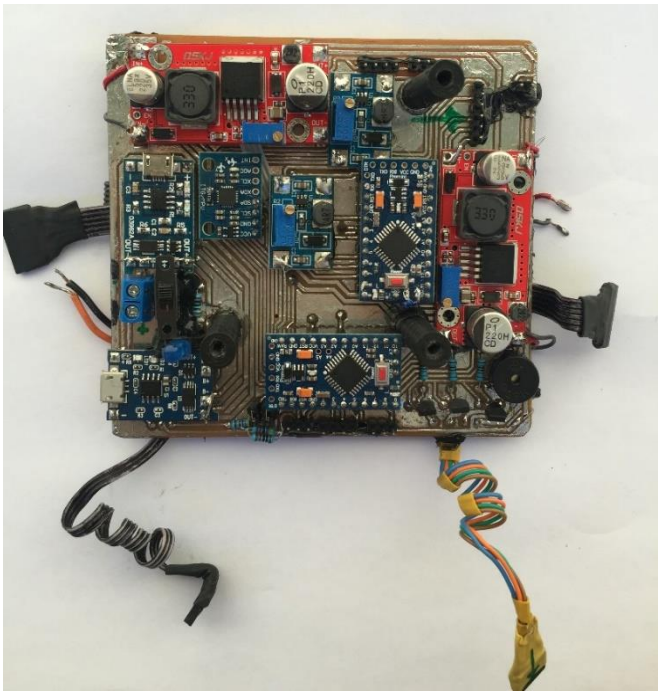

Fig. 2. View of the Completed PCB.

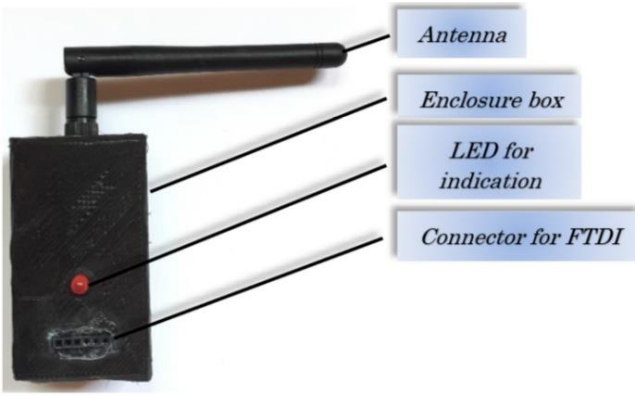

Fig. 3. View of the Transmitter Box.

B. Sensors

Similar to any autonomous vehicle, the number of sensors should be considerable so that the vehicle has sufficient information of the surrounding environment. In this project, a total of five sensors were used, two of which are ultrasonic and, on the output, provide continuous distance information, while the other three are infrared (IR) sensors and output digital information. Two sensors are mounted on the front of the vehicle, Fig. 4, while the other three on the rear, Fig. 5. The reason why there are fewer sensors on the front, is that in front there is also a camera that provides enough information even without sensors.

Also, on the PCB is a gyroscope and accelerometer (MPU6050) as well as each of the DC motors also have the encoder itself, but their information is not used in this project since there is enough information from the camera and the sensors. Using the encoders would have added unnecessary complexity to the project.

C. Batteries

Given the huge power draw from all four DC motors and the large number of electronic components, only two types of batteries were suitable for high discharge rates: Li-Po and Li-Ion. The latter are chosen since they are safer, lighter, and are cheaper and also have a smaller size. The type of batteries used is LGDBB31685, Table I. These batteries are recycled from laptop batteries. Table I shows the specifications of this type of batteries.

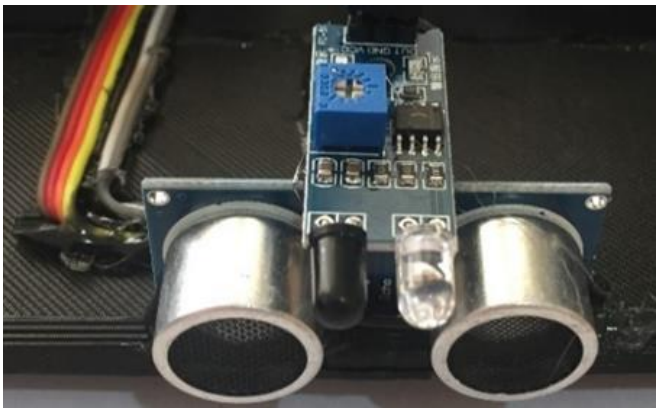

Fig. 4. Sensors on the Front Side of the Autonomous Vehicle.

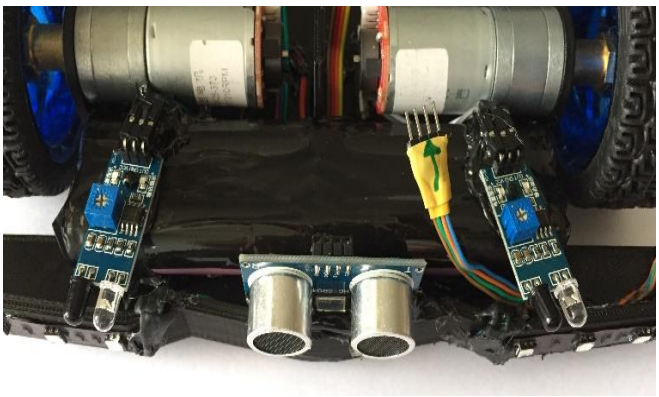

Fig. 5. Sensors on the Rear Side of the Autonomous Vehicle.

The configuration used is 4P1S, so four batteries are connected in parallel holding the nominal voltage at 3.7V, while increasing the capacity to 10400 mAh. Since batteries were recycled, their capacity is measured and the batteries have saved about 80% of their original capacity, which is more than enough for this model vehicle. Whereas, for the supply of Raspberry Pi two batteries are connected parallel with each other along with a voltage booster since Raspberry Pi works with a voltage of 5V, Fig. 6.

A final preview of the vehicle is shown in Fig. 7.

TABLE. I. BATTERY FEATURES

Capacity:	*2600 mAh* Rated
Voltage:	*3.7 V* Nominal
Charge:	*4.2V* Maximum *1250 mA* Standard *2500 mA* Maximum
Discharge:	*3.0 V* Cutoff *500 mA* Standard *3750 mA* Maximum

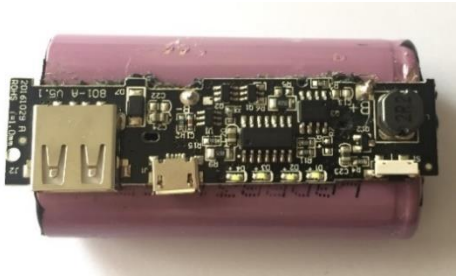

Fig. 6. Raspberry Pi Battery Power Supply.

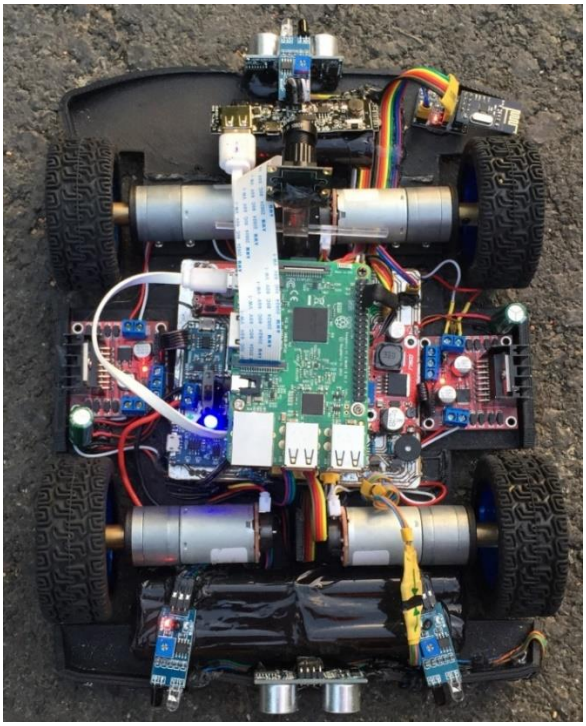

Fig. 7. Top View of the Autonomous Vehicle Model.

IV. Controllers Programming

Using only one microcontroller to process everything in the vehicle was not enough. The way this problem got resolved was by using three microcontrollers (Arduino Pro Mini) and one Raspberry Pi. Two of Arduino's are located on the vehicle's board and share the workload among themselves. First Arduino deals with the reception of radio signals and the controlling all four DC motors, while the other Arduino reads the sensor data, processes them, and transmits them to the Raspberry Pi, also it communicates with the preliminary Arduino through a single pin and controls the RGB lights at the back of the vehicle and even a buzzer. Third Arduino, is in the transmitter box and sends commands to the first Arduino for driving the vehicle. Thus, each Arduino was programmed separately and performs specific task different from one another. For programming the Arduinos, Arduino IDE software was used.

A. Arduino-Arduino Connection on PCB Board

As mentioned earlier, both Arduinos work "at their limits" to meet the requirements of this project. In order to communicate with one another, from the ready-to-use communication methods, SPI could not be used since it requires a lot of pins (such are SCLK, MISO, MOSI, and SS) and is used for communication with radio modules. The other method, the I^2C, could not be used because in the Arduino driving motors, pins A4 and A5 (SDA, SCL) were already used for directional control of the DC motors. Serial communication through the Tx/Rx pin was impossible because in the second Arduino these pins were used for communication between it and Raspberry Pi. Therefore, since there was only one free pin in the first Arduino, it was chosen to have this pin with PWM and a completely different approach was implemented, Fig. 8. Pin 3, from the first Arduino is connected to the second Arduino's A7 analogue pin. Since pin 3 is with PWM, this can output voltage in the 0-5V range, this voltage is now read by the analogue input of the second Arduino. By changing the voltage levels, various variables can be controlled. Four levels were needed, one for the front and back lights, the braking lights, and one for the buzzer. But if the pins are directly connected, this method will not work because the different voltage levels generated at pin 3 output are simply pulses with 0 and 5V values, and these will be detected by the other ADC as signals with the value 0 and 1023 (10 bits ADC) and not in-between values.

The way this problem was solved is by applying a RC filter shown in Fig. 9. Resistor and capacitor used values are: R=4.7kΩ and C=100nF on Arduino, Fig. 10.

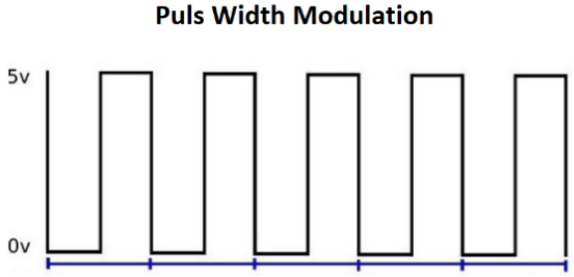

Fig. 8. Method of the PWM Function.

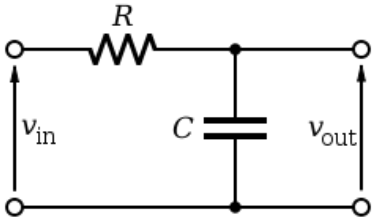

Fig. 9. Low-Pass RC Filter.

Fig. 10. RC Filter on Arduino.

From the figure it can be noticed that the signal initially comes to A1 because it was initially designed like that in circuit but later changes to A7. This filter "flattens" the PWM pulses and the output now is a constant voltage dependent on specified value of the PWM. Also, for greater precision a digital filter in the code was implemented. This filter work by taking the average of 10 consecutive measurements with a delay of 12 ms between each measurement.

B. Arduino-Raspberry Pi Connection

As noted earlier through the paper, Arduino reads the sensors and passes the data to Raspberry Pi via serial communication with the Tx/Rx pins. But connecting these two directly would damage Raspberry Pi as the latter works with 3.3V while Arduino with 5V. This problem has been solved by using a voltage divider to reduce the voltage level of Arduino from 5V to 3.3V. The pin connection method is Tx (Arduino) → Rx (Raspberry Pi), Rx (Arduino) →Tx (Raspberry Pi), Gnd →Gnd.

In Fig. 11 is shown the voltage divider scheme. The resistor value R1 is chosen arbitrarily, while the value of the resistor R2 is calculated from the following equation (1), given that the other variables are: Vin = 5V, Vout = 3.3V, R1 = 1.2 kΩ.

$$Vout = Vin * \frac{R1}{R1+R2} \quad (1)$$

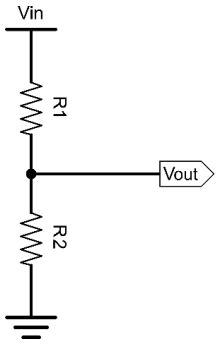

Fig. 11. Voltage Divider Scheme.

The Rx (Arduino) → Tx (Raspberry Pi) connection does not need a voltage divider, since the signal received at the Rx of Arduino is 3.3V, therefore within the working range of Arduino. Therefore, just one voltage divider is required.

C. Connection Arduino-Raspberry Pi

The Raspberry Pi-Laptop communication is implemented via a TCP server and socket. The Python library that enables this communication is called socket server. More detailed information on using this library and its implementation is available on the link in the references.

Both the Raspberry Pi and the laptop must be connected into the same Wi-Fi router to communicate. If possible, only these two without other devices for faster communication must be connected.

After obtaining devices IP addresses, these IPs are then written in Raspberry Pi as well as on the laptop:

```
h, p1, p2 = "192.168.0.100", 8000, 8002  # IP and ports h--> host.
```

Through the above line, host, two ports are defined. It should be noted that the IP can vary depending on the devices. To start communication, the program on the laptop should be initiated first, to start the server, and then in the clients (programs in Raspberry Pi). Hence the laptop is the server in this case, while the client is Raspberry Pi. The client sends data to the server.

D. Communication Laptop-Arduino

Data processed by the laptop through the serial port (USB) are sent to Arduino in the transmitter box and then conveyed to the vehicle through radio waves. In order for Python to send this data to serial port, the pyserial library is utilized:

```
import serial as s

ser = s.Serial('COM6', 115200).
```

The above lines enable communication, where 'COM6' is the port Arduino is connected, while '115200' represents baud rate.

V. Training of Models for Detection of Traffic Signs

Among other objectives of this paper is the detection of traffic signs. It is known there are many traffic signs, but for this project only the STOP sign is selected.

The selected detection mode is based on Haar features.

In order to detect a particular object, a cascade classifier should firstly be trained [6, 7]. This is done by taking positive, Fig. 12, and negative images, Fig. 13. Positive images are those images that contain the object that has to be detected, while negative images may be anything as long as they don't contain target detection object. As negative images are used the track, the road, i.e. mainly behind the scenes from moving the vehicle [9, 10], but without the traffic signs in those images. The greater the number of positive and negative images is, the longer training will take, but the classifier will be more accurate.

Fig. 12. Examples of Positive Images.

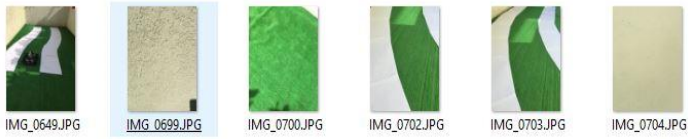

Fig. 13. Examples of Negative Images.

There are many ways to capture a large number of images, ranging from downloading from the internet to individual object photography; the method that is preferred in this paper is to capture a video with the target object for detection and then extracting images from the video frames. This speeds up the process of capturing images.

After capturing positive images, they should be cut in dimensions as close as possible to each other and should only contain the target for detection. After this step, comes the classifier training, whereby the trained classifier is stored as a xml file.

The trained classifier is then stored as *stop.xml* and used for detection. Below is given an example showing STOP sign begin detected as shown in Fig. 14.

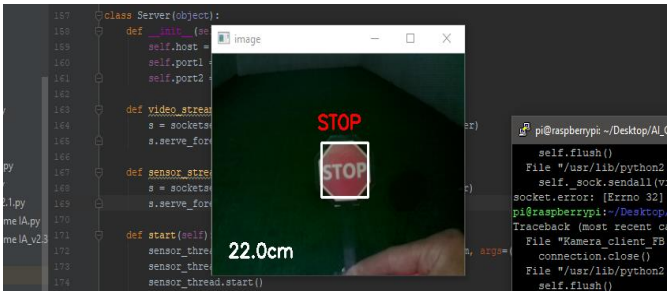

Fig. 14. Stop Sign being Detected after Training the Classifier.

VI. Programming and Control with Artificial Intelligence

This is the most challenging part of this paper. The method used for Artificial Intelligence training is CNN. CNNs are layers of perceptrons. The first being an input layer, then many hidden layers and an output layer. For example, with an image, each perceptron in the input layer would correspond to a pixel value. Then the next layer receives input from every perceptron in the previous layer, and if it passes, the value then it will be sent into the next hidden layer until the output layer is reached. The dataset was tested on different positions of the STOP traffic sign and calculated the accuracy and time required to identify those signs.

For this research paper, firstly you should drive the vehicle manually, where for each key pressed for driving the vehicle, the corresponding frame and the direction of movement are saved.

The frame obtained from the video is initially converted from RGB (colour format) to grayscale. In grayscale image formats, each of the individual pixels has a certain value that represents the brightness of that pixel.

Grayscale image is a matrix where each element of the matrix represents a pixel, and each pixel has a certain value that indicates how bright or dark that pixel is. This file, from a two-dimensional matrix, becomes a one-dimensional array of information for each pixel and is stored as a npz file.

In the input layer of the CNN as input is given the previously stored npz file (the string of pixels), while the output layer is feed with information about the button pressed for that frame.

The computer then by comparing human input with the current frame calculates backwards weights of each joint separately to match the input. So, in a way, the output responds to the input while calculating the weights of the joints, so this method is called backpropagation, and that's how neural networks they learn in this case. Repeating the process of calculating the weights for each input frame, and each current input by the human for all training data, the computer is able to generate a CNN model which for any given frame responds with an output which in this case is: Front, right, left or back.

After completing weights calculation for each frame, the neural network model is stored as *.xml, then this file is used to predict the direction of car's movement.

A. Starting the Video Transmission and Sensor Data

Since the CNN model for vehicle motion prediction is located on a laptop, the camera and sensor data must be transmitted from Raspberry Pi to the laptop. In Raspberry Pi, there are two Python scripts that enable this stream, but firstly must be executed, Fig. 15. This can be done in several ways including the Raspberry Pi connection with HDMI, but the preferable method is by accessing the RPi via SSh (Secure Shell). The way the scripts are executed in Raspberry Pi directly from the laptop is as follows: Initially we connect to Raspberry Pi through hostname and password. Then, we need to know where the Python scripts are located. After navigating to the path containing the Python scripts, we run them. Here follows the procedure of navigating and running the scripts.

During this process, both the laptop and Raspberry Pi must be connected to the same network. The video transmitted by Raspberry Pi has been reduced to 240x320px for faster transmission, this way having less lag and transmission delays.

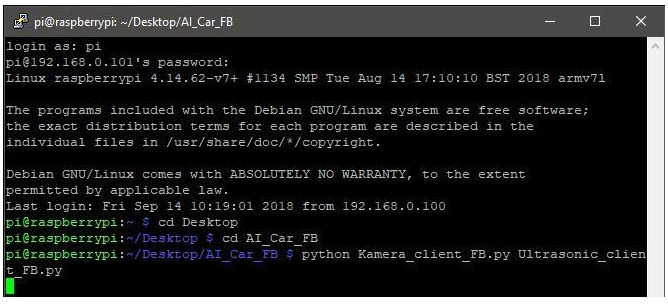

Fig. 15. Python Scripts Execution through Putty.

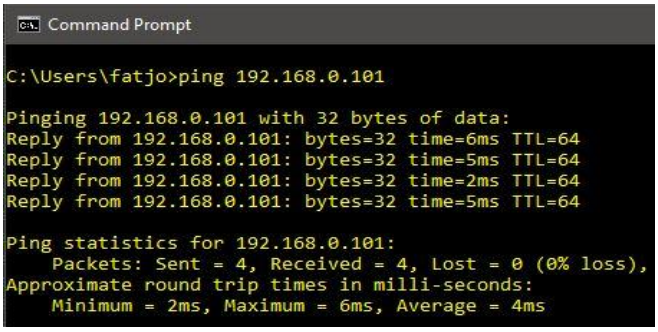

Fig. 16. Pinging Raspberry Pi.

If one ping Raspberry Pi from the laptop, one can get the time it takes for Laptop - Raspberry Pi communication, Fig. 16. In this case an average round trip of 4 ms was achieved.

Further, the evaluation of learning performance for the STOP traffic sign recognition with the different slant angles and occlusion ratios based on CNN has been done.

Fig. 17 and Fig. 18 show the learning performance with slang angles normal and parallel to sign, respectively. Fig. 19 shows the learning performance with occlusion ratios.

B. Manual Drive for Data Collection

The vehicle is capable of moving in eight different directions: forward, backwards, forward-right, forward-left, backwards-right, backwards-left, and also spin in place clockwise and counter clockwise.

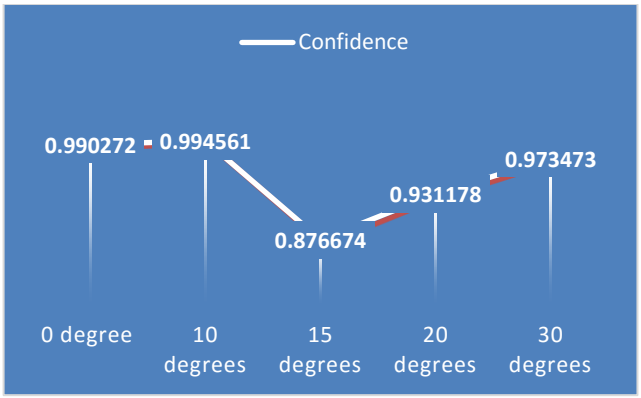

Fig. 17. Performance with Slang Angles Normal to Sign.

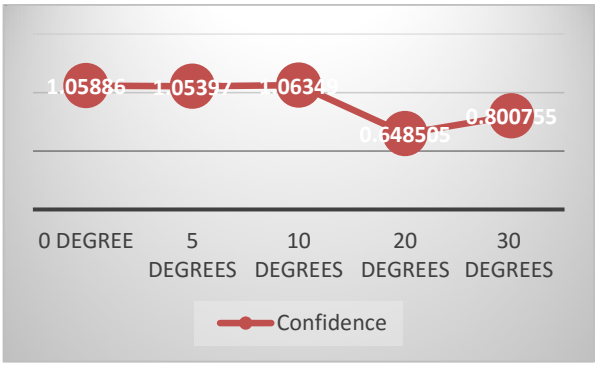

Fig. 18. Performance with Slang Angles Parallel to Sign.

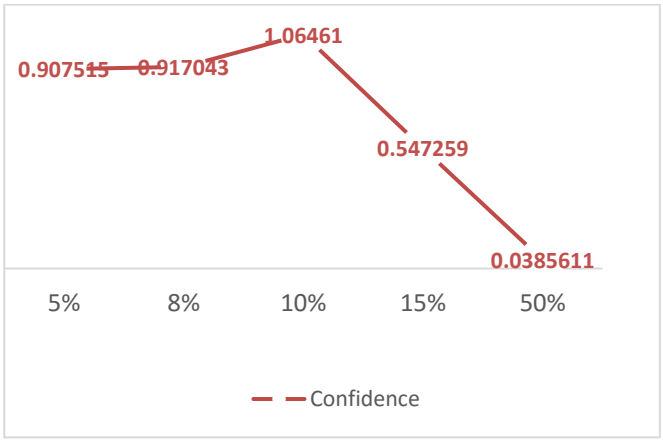

Fig. 19. Performance with Occlusion Ratios.

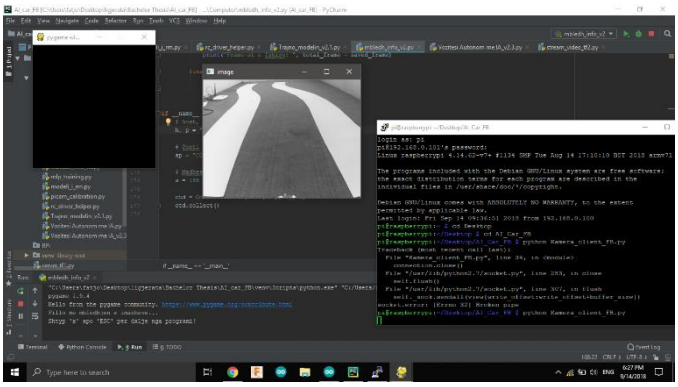

Fig. 20. Data Collection Process for CNN Training.

The first step in CNN training is manually driving the vehicle. In the Python code, the vehicle is driven by the W, A, S and D keys: W for forward, S for reverse movement, D-right, A - left, WD - right front, WA - forward left, SD-backwards right as well as SA - backwards left. During manual data collection, Python scripts for video transmission from Raspberry Pi on the laptop should be executed at the same time, so that for each key pressed, the laptop will capture the frame and the corresponding key. These frames, as mentioned earlier from a 2D matrix, are transformed into a string and stored as a *.npz file. Data collection process is very important because the accuracy of the CNN training and its prediction depends directly on the data collection process. In this case, over 440 Mbytes of data for CNN training were collected as in Fig. 20.

VII. Data Processing Method

Throughout the paper is mentioned that the video and data are transmitted from the RPi to the laptop. Knowing the fact that RPi is a single board computer, the question why the processing isn't done directly on Raspberry Pi instead of laptop might arise. This all comes down to processing power. Below is given a Table II showing key differences between these two. From the table, it's clear that laptop's processing power is far "superior" compared to Raspberry Pi's.

The way how the entire hardware "talks" to each other is shown in Fig. 21.

TABLE. II. Laptop–Raspberry Pi Comparison

	Processor model	**Processor clock speed**	**RAM**	**Internal memory**
Laptop	Intel core i3	@2.53 2.53 GHz	6 GB	240 SSD +750 GB HDD
Raspberry Pi 3B	ARM cortex A53	@1.2 GHz	1 GB	8 Gb SD card

Fig. 21. Laptop–RPI–Laptop–Arduino Communication Routes.

VIII. Conclusions

For this research paper, mechanics was applied for designing the vehicle and mounting of the entire hardware, electronics was used to develop a circuit board, and the hardware was synchronized with the electrical circuits through the programming language. The fusion of all of these has resulted into a mechatronic project.

In conclusion, a vehicle capable of moving in all the necessary directions (forward, forward right, forward left, backwards, backwards right, backwards left as well as turn and rotate in place) was "born".

The vehicle has been equipped with sensors for gathering necessary information about the surrounding environment as well as camera to collect the pictures of the traffic signs.

Based on the CNN algorithm one can conclude that the influence due to occlusion is much larger than that of slant angle. Therefore, it can identify the stop sign even if the sign is inclined 30 degrees and with 15% occlusion.

The importance of this paper can be seen in the fact that some steps have already been made to navigate the autonomous car model that can be compared with the autonomous machines of the elite companies in this field.

There is also room for improvements that will be made in future models of this vehicle using other learning methods for traffic sing recognition.

References

[1] Pannu G. S., Ansari, M. D. & Gupta P. Design and Implementation of Autonomous Cars Using Raspberry Pi, International Journal of Computer Applications (0975 – 8887), pp.222-222, Volume 113, No.9, March 2015.

[2] M.Karthikeyan, Mr. G.Sreeram, M.Tech, (Ph.D). Intelligent Exploration and Surveillance Robot In Defense Environment. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering. Vol. 3, Special Issue 1, February 2014.

[3] G. N. Tripathi and V.Rihani. Motion Planning of an Autonomous Mobile Robot using Artificial Neural Network. Mody Institute of Technology and Science, Lakshamangarh, Sikar, Rajasthan.

[4] Rakesh Chandra Kumar et al. Obstacle Avoiding Robot – A promising One. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, Issue 4, April 2013.

[5] Beqa, F. Development of autonomous vehicle driven by Artificial Intelligence, University of Prishtina, 2018.

[6] Pajaziti, A., & Bajrami Xh. & Paliqi A. Path Control of Quadruped Robot through Convolutional Neural Networks, 18th IFAC Conference on Technology, Culture and International Stability, Sept 13-15, 2018, Baku, Azerbaidschan, 2018.

[7] Bajrami, X., Gashi, B., & Murturi, I. (2018). Face recognition performance using linear discriminant analysis and deep neural networks. International Journal of Applied Pattern Recognition, 5(3), 240-250.

[8] Hwu, T., Isbell, J., Oros, N., & Krichmar, J. (2017, May). A self-driving robot using deep convolutional neural networks on neuromorphic hardware. In 2017 International Joint Conference on Neural Networks (IJCNN) (pp. 635-641). IEEE.

[9] Ishibushi, S., Taniguchi, A., Takano, T., Hagiwara, Y., & Taniguchi, T. (2015, November). Statistical localization exploiting convolutional neural network for an autonomous vehicle. In IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society (pp. 001369-001375). IEEE.

[10] Tai, L., Li, S., & Liu, M. (2017). Autonomous exploration of mobile robots through deep neural networks. International Journal of Advanced Robotic Systems, 14(4), 1729881417703571.