Global Journal of Modeling and Intelligent Computing (GJMIC)

Vol. 1, No. 1, January, 2021 ISSN: 2767-1917

DEVELOPMENT OF ROBOTIC MODEL TO SUPPORT INTELLIGENT VEHICLES BEHAVIORS

Aaron R. Rababaah, American University of Kuwait/CEAS., Kuwait (haroun01@gmail.com)
Anwar M As'ad, American University of Kuwait/ CEAS., Kuwait (s00038340@auk.edu.kw)
Ahmed A Sultan, American University of Kuwait/ CEAS, Kuwait (s00038340@auk.edu.kw)
Abrar T Al-Failakawi, American University of Kuwait/ CEAS, Kuwait (s00036020@auk.edu.kw)
Khaled K Al-Arouj, American University of Kuwait/ CEAS, Kuwait (s00033904@auk.edu.kw)

ABSTRACT

Advancements of Artificial Intelligence have opened new opportunities for intelligent solutions in many areas including: industry, health care, government, education, transportation, etc. In this work, we are interested in intelligent autonomous vehicles. This paper presents development of a robotic model to support intelligent vehicles behaviors. The system is an integration of Android, Arduino, RaspberryPI and Google Machine vison/learning libraries. The hardware was design using small scale robotic kit equipped with microcontroller and mobility platform. the tested intelligent behaviors included: wireless communication, line following, find home-base, obstacle avoidance, and auto park. All the stated behaviors were tested in real world environment and results were reliable and promising to support more future inclusion of advanced behaviors.

Keywords: Mobile robotics, intelligent vehicle, Autonomous behaviors, Artificial intelligence, Machine vision.

1. INTRODUCTION

Since the development of new technologies, vehicles have become smarter and more advanced (Rababaah, 2019). Artificial intelligence is playing a big part in most of those technologies in recent years, specifically in mobile vehicles (Wang, 2016). The purpose of AI in vehicles is to minimize the tension on drivers and help them have a safer journey. Some of the goals in those cars are avoiding obstacles, keeping a safe distance, self-parking, and navigating the driver to their destination. However, to implement and release all those features, there should be a heavy testing phase done by the developers to ensure that all the features are working and integrated successfully. This testing phase has created a problem for the developers since they can't test their implementations on public roads.

Therefore, with the development of INVENT, we want to try to help developers to test and simulate their logic in a smaller version of the car to facilitate the testing phase. To be able to achieve that, we will look at different aspects of a car that is controlled by an AI and try to simulate a simple autonomous car experience. To add something new to the smart cars, we will design a software to include all the supported methods in the car itself. We have seen many projects with AI that helps people to learn and advance to better technologies in the future. The goal of INVENT is to be one of those projects. INVENT is a project that will mainly aim to simulate the experience of the real autonomous car to help with the advancement of that technology.

INVENT will consist of two main parts, an application, and a mobile robot. The application will include several programs to give inputs to the car and make it respond to commands. Each sensor will be handling different actions to analyze and read the environment and translate it into inputs for the robot. Moreover, the robot will be detecting the trail on the ground to follow it. If there was an obstacle on its way, then it will avoid it and keep moving to the final point. The software application will include some functions to test different modes of INVENT (smart car), and this application will help us to keep every functionality in one organized place.

The sensors will play a big part in the whole project by calculating the surroundings and then try to move around the track. There will be mathematical equations for the sensors' input. For example, it will calculate the distance,

rotations, speed, and collisions, which will help the robot to handle different events that occur in the environment with precise measurements. In this case, the robot will be running smoothly on the track while adjusting itself to the lane it is following. The functionalities are divided into two parts, one is handling the car's movement and steering logic through a micro-controller, and the other is giving orders to the car by using a phone application. Car movements are what basically allows the robot to drive, steer, and stop when required. However, the second part will be just giving orders/inputs from built-in methods to the robot. The built-in methods are further explained in detail in section 3.1 and 3.3. The application will be connected wirelessly using our mobile application. Besides the functionalities, there will be an easy user interface (UI) to make it easy for everyone who is using the application to navigate between the tabs and methods.

The main objective of the project is to simulate a real autonomous car but in a minimized version. This simulation offers flexibility, fewer costings, and safer to test. The purpose of the simulation is to help understand whether the implementation of the robot requires changes or not. Besides, since the robot isn't tested on real roads, the simulation is safer, and not putting anyone at risk. Moreover, the costs to operate, fix, and implement are cheaper than a real autonomous car. Lastly, different environmental conditions can be easily tested on this simulation to check how reliable the implementation is.

Companies that are currently developing and improving on their autonomous cars can use projects like, "INVENT," that tests whether their implementations were done correctly or not. They can apply their code to the project and help them run a test on different scenarios. Doing so can to detect mistakes easily and fix, modify, or improve upon their implementations with more flexibilities.

2. LITERATURE REVIEW

Automobile Robots made a massive turnaround in the current generation (Rababaah, 2020). Mobile robots are, without a doubt the, most utilized robots to help humans in their daily life. Those robots help with lifting heavy weights, transporting the needs, and testing different features. Scientists and engineers have found many methods to control a mobile robot. Either by using a unique manual remote or an application. However, using AI for those applications is the go-to for the new generation since it's easier for the user to handle the robot and add more features to it. Since mobile robots have different types of sensors, it can be used to replace humans in preventing them from handling or going to any danger. Developers nowadays use AI in cars to compete in industries, which proves how powerful a mobile robot can be. Whether it is for entertainment, achievement, or helping in improving people's life, AI will always be used and enhanced for future needs.

Robots have invaded the technology world and made a massive impact in assisting human's life (Rababaah & Rabaai, 2017). They have different aspects that can support the workload of humans with less financial and physical risk. With the advancement of technology, driverless cars became a huge thing in the mobile robot industry. As the writer and his colleague Fortunati, Lugano & Manganelli claim that, "Self-driving cars, truck platooning, and various forms of connected and automated driving are widely promoted by policy reports, industry press releases, media stories, and online discussions." This wide promotion of self-driving cars proves that it will soon be the norm in daily life (Leopoldina, 2019).

We believe there are multiple ways to implement a driverless car system. One idea was inspired to us by (Rababaah and Kuscu, 2012). On one of their published researches, they implemented a vision-based indoor positioning system (VIPS). A local GPS system can be implemented using this technology. This approach requires the use of a camera from a high angle and program it to detect a specific color. It can be used to find the robot's position and help it stay within the track. However, there could be other objects with the same color of the car, which will cause detection issues for the camera. In addition to that, this implementation does not provide a precise positioning of the robot. Therefore, the use of VIPS cannot be implemented in this project.

Another possible implementation that we considered is the use of an infrared sensor to stay within the track. The infrared behaves differently if the reflected surface was black or white. Therefore, we can detect if the car is drifting away from the path or not. In addition to that, this implementation requires fewer resources to use. It also has a low chance of environmental distortions, which enhances the reliability of the system.

An alternative implementation could be integrated by finding a path in front of the car to follow. One attempt was made by using a camera to take pictures, and with the help of the microcontroller Raspberry Pi it will be able to identify the white color line in the road to track it.This approach requires having a high-resolution camera and light when it's dark to differentiate the colors of the path (Sumardi, 2018).

Autonomous cars can help in simulating a real car experience before it is fully implemented. A group of researchers attempted to implement a parking system in a mini car version to be able to test and identify the possible errors that could happen before implementing it in real life. This decision helps the researchers to test high-cost sensors, which requires testing and possibly fail many times to be able to produce its best possible outcome. In addition, simulating with small autonomous robots can protect people from being in danger during the testing phase, because accidents can happen. Using driverless robotic cars that can be fixed and repaired is way safer than people driving while testing (Grall et al., 2010).

Multiple robotic figures that have been implemented previously as projects. They used an autonomic robot with similar features, and diverse in terms of aspects and usage — One self-driving simulation car was implemented by using a camera to find the path. The camera would take real-time pictures then process them using Raspberry Pi to spot the white line color; then the robot will start following the trail.

Another robot car with a similar approach that uses a camera and line color identification system but with more features is called Timótheo. It follows the line using a mobile camera that has been connected using Bluetooth to take an image and analyze the color. The autonomous car has a flashlight to support its capability to track the path in darker environments. It also avoids obstacle using an ultrasonic sensor to prevent bumping into a bottle. When the robot avoids the bottle, it connects to another line to continue following the track. They used a flashlight for the camera to overcome the problem of weak detections in dark environments (Bordin, 2015).

The previously mentioned projects had multiple issues that they didn't deal with. The first issue is using a camera. When relying on the camera to detect the road, you are creating two problems. You need a perfect environmental light condition because the camera heavily relies on identifying the color of the track/road to help it stay within the line. If the camera had issues with detecting the path, then it creates unreliability and safety issues. In addition, relying on a camera to detect colors can create confusion for the camera if there are multiple objects with the same color within the environment. The second issue that previous projects had is the requirement of heavy processing. Using a lot of sensors to detect obstacles with the combination of constant processing to detect road track can cause slow reaction time to the robot. In other words, the implementation might work, but the time needed to process other sensors' inputs can cause a delay for the camera when detecting the line to track. Therefore, INVENT is aiming to use little processing with a powerful micro-controller to overcome this issue. In fact, INVENT won't be relying on the use of a camera to detect the road track, which will allow it to function in dark environments. Moreover, with the use of our skills that we acquired from the course, Analysis of Algorithms, we will be trying to optimize our code and implementations as much as possible. Therefore, those factors will help ensure that INVENT can function with little to no reliability issues.

3. TECHNICAL BACKGROUND

We will be using a microcontroller to program the robot. There will be a phone connected to the robot to give it the specified commands through an application. The commands can be sent by the user with the provided user interface of the application. Once the user chooses the command, a signal will be sent to the microcontroller to make the robot respond to the user's command.

Hardware: Robot DC Motors and Wheels, Microcontroller (Arduino Mega or Raspberry Pi 3 B+), Proximity Sensors (Ultrasonic and IR), Android Phone, Camera, Robot's Chassis, Motor Driver Module, Bluetooth Module HC-05

Software: Programming languages: Java, C++, Python, IDE: Android Studio, Arduino, Python, User interface design: Adobe Photoshop.

Functionalities:

- Give Wireless Commands to the Robot: This is considered as the main functionality of the project. We will develop a mobile application that connects to the robot wirelessly. This connection can be made through

Bluetooth. The wireless connection will allow the user to send commands to the robot. Then, the robot will act upon the commands that the user specifies through the application's user interface. This scenario is shown in Figure 1.

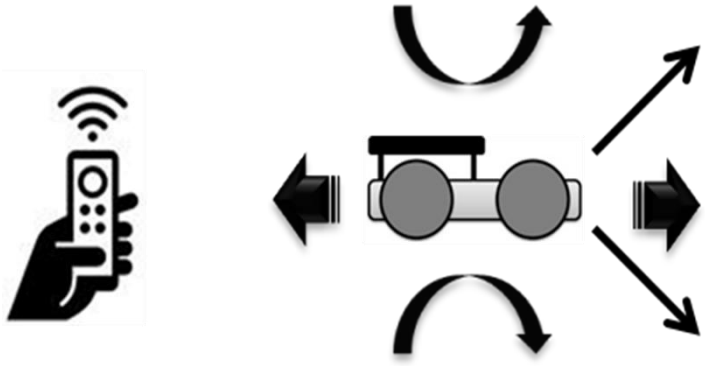

Figure 1: Wireless command scenario

- Follow the line: Following the line is one of the most important functionalities of the robot. There will be a black line (Street) underneath the robot. The robot will be using its sensors to detect the surface underneath it. If it detects that it is drifting away from the black line surface underneath it, it will try to adjust itself back into the track. This scenario is shown in Figure 2.

Figure 2: Line Follow scenario

- Stop when Required: To simulate a real autonomous car, we will be adding a traffic light to the track. We will detect the traffic light using a camera. If the traffic light is red, the car should detect it and stop. If the traffic light changes to green, the car will start moving once again. However, if the traffic light was initially green, the car will ignore it and continue driving. This scenario is depicted in Figure 3.

Figure 3: Traffic light scenario

- Move Back to Base: Move back to base is a simple feature that can be called to stop the car from moving around the track and go back to its base to stop, or to charge the robot. The way how we can implement this feature is by adding a visual cue that the camera can detect. This cue can indicate the robot that it is time to back

to base. For the time being, we are planning to allow the user to manually charge the robot through a wired connection. However, a wireless charging station might be implemented if we had enough time in the implementation phase. Figure 4 shows this scenario.

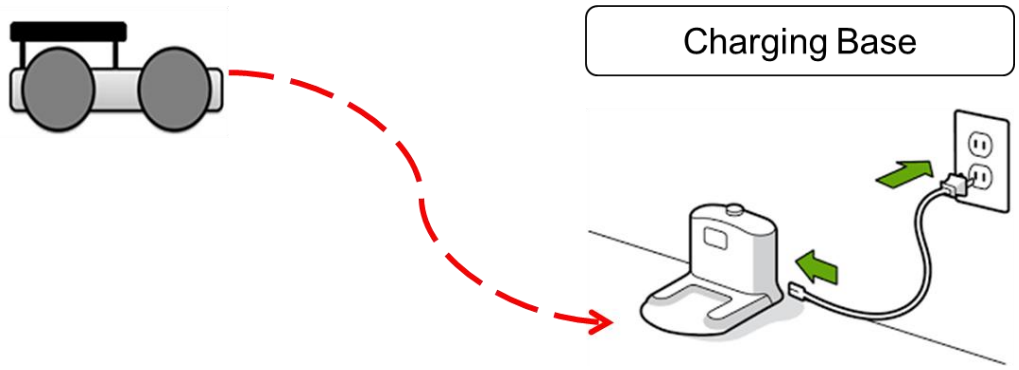

Figure 4: Return to charging base scenario

- Obstacle Avoidance: The track will include some obstacles that the robot needs to avoid. We will be using ultrasonic sensors to detect obstacles in front of the robot. If the robot detects any obstacles, it will try to avoid this obstacle by going around it. This feature helps ensure that the robot won't hit the obstacles in front of it and break. Figure 5 depict this behavior.

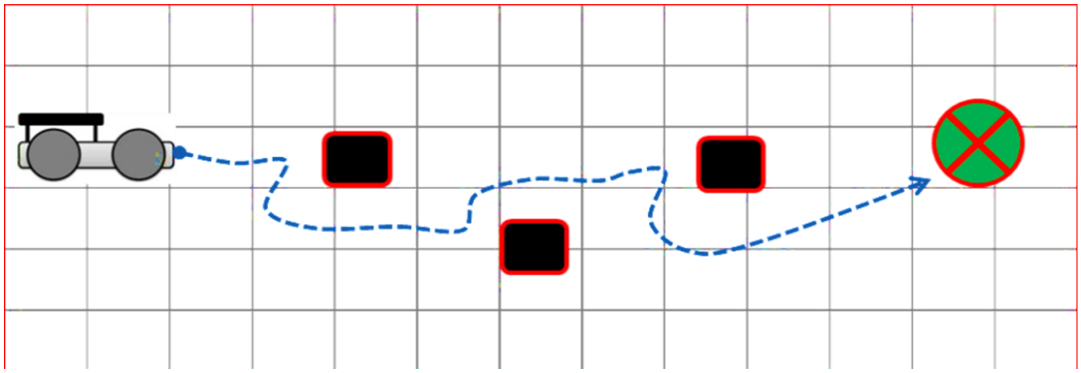

Figure 5: Obstacle avoidance scenario

- Find a Parking Place: We will add functionality to the car in which it will try to look for a parking place. We will implement this feature by allowing the camera to look for a specific visual cue. This cue will indicate the robot whether if there is a parking place or not. If the camera finds that cue, then the robot will enter the parking space as shown in Figure 6.

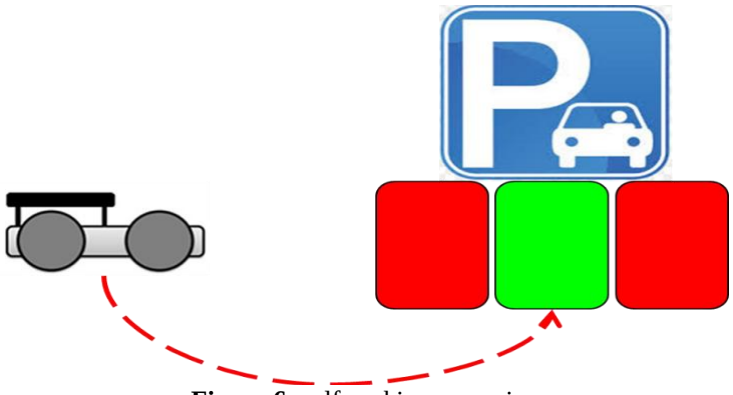

Figure 6: self-parking scenario

To accomplish the described behaviors, Figure 7 illustrates the operation feasibility requirements.

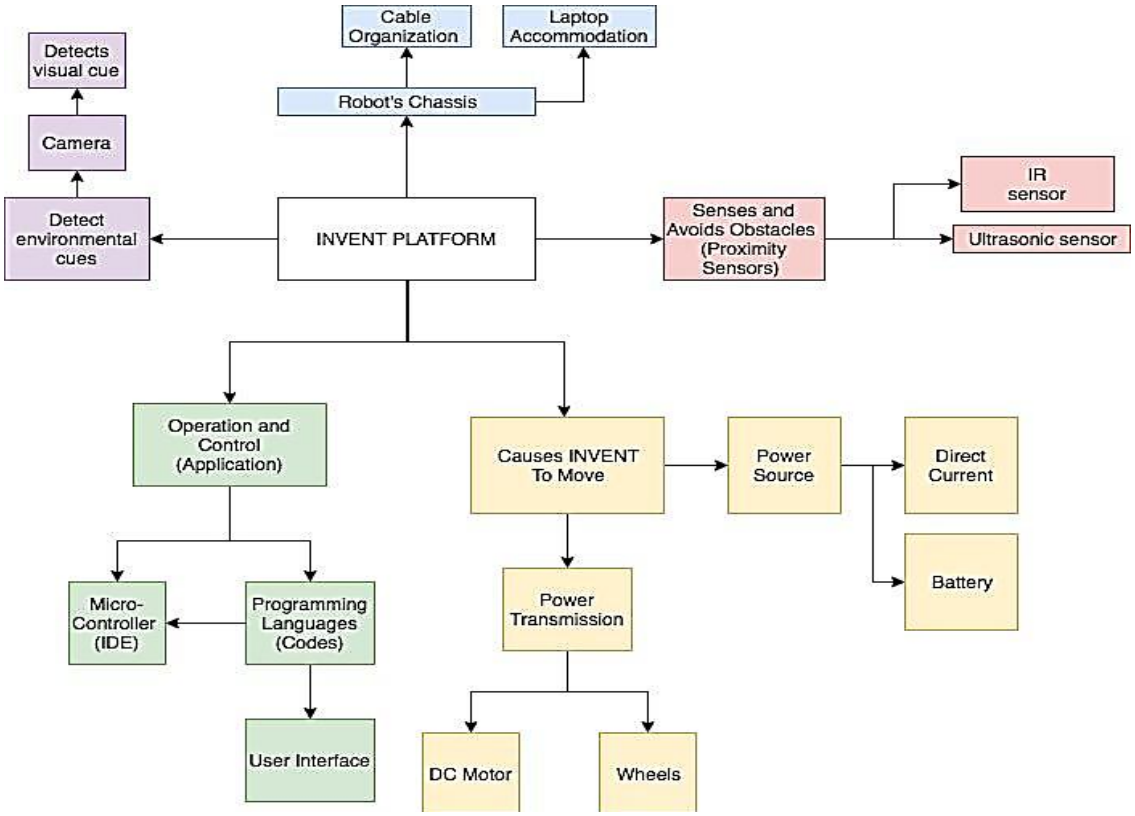

Figure 7: System operational feasibility requirements

4. EXPERIENTIAL WORK

For testing purposes, the GUI shown in Figure 8 was developed and used to run the different scenarios.

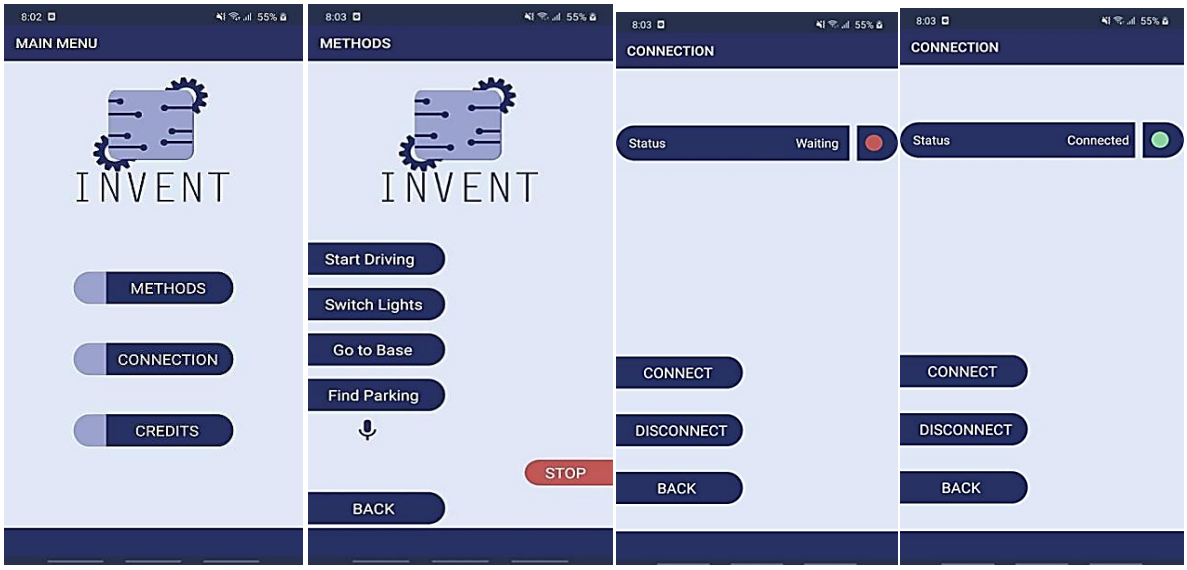

Figure 8: GUI interface of the

Verification: We have made sure to utilize minimum resources to ensure that the application runs smoothly for the user. The use of efficient algorithms will ensure that the battery lasts longer in both the robot and the android device. In addition, we made sure to utilize reliable sensors to avoid any false inputs. For example, instead of using a camera for light detection, we decided to use photocells and capacitors to ensure reliable inputs. In addition, if the robot is not receiving any inputs from the user, sensors will stop to avoid any wasteful battery drainage.

Validation: The testing phase will ensure that the user will meet the minimum to no bugs while using the project. All functionalities operate as required and retrieve the correct information from the sensors. In addition, front-end design will be going through an enhancing phase to highlight an issue and fix it.

Unit Testing: Testing each command of INVENT separately to meet all the I/O requirements because checking each function separately will help in observing errors without crossing with other features. It also supports identifying the error of whether it is a wiring or coding issue, so it can be fixed to meet its expected purpose. This test will also help in adjusting the unplanned results and repeating the test until it reaches the desired outcomes of the function. In addition, once each function is successfully tested, we will recheck them once they are integrated to ensure further that we fix as many bugs as possible. Unit testing activities are listed in Table 1.

Table 1: Software Unit testing

ID	Test Case Description	Expected Result	Actual Result	Remarks
1	Application	Application successfully starts	PASS	-
2	Connection	Phone successfully connects to the robot	PASS	Phone and Robot should be connected to the same network
3	Function: Start Driving	Robot starts driving	PASS	-
4	Function: Switch Lights	Turn on or off depending	PASS	Lights will not turn on if there is already enough light in the environment
5	Function: Stop at Red Light	Robot stops if it sees red traffic light	PASS	-
6	Function: Go to Base	Robot goes back to its base	PASS	Robot needs to see a visual cue to find its base
7	Function: Avoid Obstacles	Robot avoids objects blocking its path	PASS	-
8	Function: Find a Parking Place	Robot finds where the parking place is	PASS	-
9	Voice Commands	User records commands through microphone	PASS	-

Integration Test: When individual testing is done, the application will be tested for compatibility. We will make sure that the application works on different smartphones with an android version 7.0 or above. It also requires Wi-Fi to be able to connect to the autonomic robot to send commands. For all the compatible smartphones, the application should work with no issue, since it does not need a lot of resources and requirements.

Performance testing: We Tested the performance of all the commands when using the application to observe the response time of each command. Trying different ranges of how far the autonomic robot is connected and mark if it will affect its performance. If the smartphone and the robot are connected to the same network, distance should not impact the response time much. Performance testing are listed in Table 2 and plotted in Figure 9.

Table 2: Results of performance testing		
ID	Test Case	Average Response Time
1	Application Launching	1.5 Seconds
2	Connection	1 Seconds
3	Sending Commands	1 Seconds
4	Receiving Commands	0.8 Seconds
5	Commands Execution	1 to 1.5 Seconds
6	Image Processing	Around 0.3 Seconds

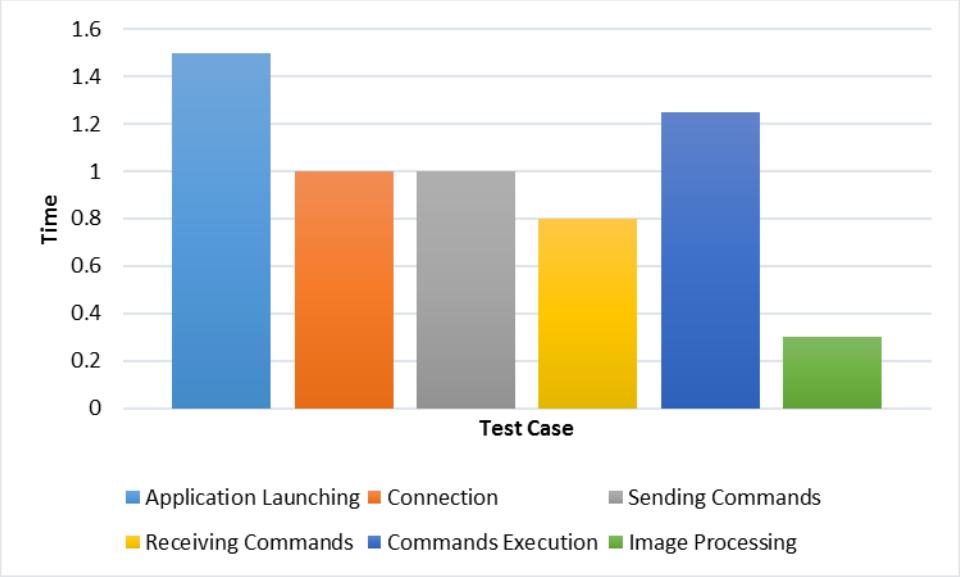

Figure 9: System performance testing

Load Testing: The load testing phase happens when the application sends and receives data frequently. Since the application will mostly rely on sending commands, we will use a buffer to ensure that all commands will arrive when the robot is available with no overloading. The application is using TCP connection, which will ensure there will be no packet loss. If the packet is corrupted or lost, the TCP will send a copy of the packet.

Stress Testing: This test will check whether the system works under abnormal conditions such as overheating or application crashing. In case of a crash, the application shall record logs to trace what exactly caused the error. In addition, we have made sure to keep the microprocessor as cold as possible, since raspberry pi can overheat in extreme conditions, which will decrease its performance. We have allowed the raspberry pi to underclocks itself when overheating to protect it from damage. In addition, we are using heatsinks to absorb the heat away from the processor also to maintain its temperature. Table 3 shows the stress testing results.

Table 3: System stress testing results		
ID	Test Case	Microprocessor Temperature
1	5 minutes of use	32 Celsius
2	15 minutes of use	46 Celsius
3	30 minutes of use	51 Celsius
4	1 Hour of use	53 Celsius
5	2 Hours of use	52 Celsius

5. CONCLUSION

There is no doubt that autonomous vehicles will start spreading around in the future. However, to test and ensure this technology's safety, we need to start with prototypes. Therefore, we introduced, "Intelligent Navigator Vehicle for Effective Transportation" (INVENT), an autonomous vehicle controlled by a micro-controller. INVENT's primary use is to test various independent features on a small car to avoid major mistakes when testing autonomous vehicles on roads. INVENT's leading development software, and hardware are Android Studio, Python, Java, and Raspberry Pi. Android studio and java were used to develop the android application part. The Raspberry Pi and Python were used to control and develop the vehicle's logic. INVENT currently supports Android devices with version 7.0 and above. Throughout the development process, we ensured that INVENT would be very easy to use and modify for the end-users. The team is dedicated to enhancing and continuing working on the project to facilitate autonomous vehicles testing.

It was not possible to implement one of the project's promised features due to the current global pandemic. Therefore, the first step that we are willing to take is to fully implement the "Follow the Line" feature into the project. In other words, this feature is at our highest priority to implement. In addition, we will make sure that the installed infra-red sensors auto-calibrate the surface underneath to help the robot work on multiple roads. The auto-calibration feature will adjust the infra-red's sensor sensitivity towards dark objects to avoid false positives while driving along the track.

To make the robot drive more accurately when driving along the track, we are planning to install wheel encoders on the motors The wheel encoders will help the robot detect if one of the motors is moving faster than the other, which will help it slow down or speed up to balance both motors. In other words, the primary use of this feature is to detect whether the robot's wheels are balanced. Another feature that we are planning to implement in the future is the use of front angled wheels. This hardware will help the robot to take turns and avoid obstacles easier. In addition to that, adding motors in the front wheels will help the robot to move faster in the track; Thus, improving its effectiveness.
The fourth idea that the team is planning to implement in the future is wireless charging. The central concept behind this idea is to allow the robot to charge wirelessly when it goes back to its base. This feature will facilitate the charging process for the user. However, since this feature does not improve the functionalities of the robot much, this feature will have the lowest priority.

REFERENCES

Rababaah, Aaron R. Kandil,A., Gharib, Y. & Ahmed, H. (2019). Visual Gesture Recognition for Drawing Applications. *Journal of Global Information Technology*, 14(1,2), 1-9.

Wang, j. (2016). Wearable sensor based human posture recognition. *IEEE International Conference on Big Data (Big Data) Washington DC,USA: IEEE*., 3432-3438

Bhavsar, P. (2017). Machine Learning in Transportation Data Analytics. *In P. Bhavsar, Data Analytics for Intelligent Transportation Systems* (pp.). *ELSEVIER*, 283-307, doi:10.1016/B978-0-12-809715-1.00013-4.

Rababaah, A.R. (2020). Angle histogram of Hough transform as shape signature for visual object classification – (AHOC). *Int. J. Computational Vision and Robotics*, 10(4), 312–336.

Leopoldina, F., Lugano, G., &Manganelli, A. M. (2019). European Perceptions of Autonomous and Robotized Cars. *International Journal of Communication*, 13, 2728-2747.

Rababaah, A. R., &Kuscu, E. (2012). Mobile Robot Localization Via Efficient. *International Journal of Science & Informatics*, 2(1), 23-32.

Sumardi, S., Taufiqurrahman, M. R., &Riyadi, M. A. (2018). Street Mark Detection Using Raspberry PI for Self-Driving System,Yogyakarta, Indonesia: Universitas Ahmad Dahlan. Retrieved November 28, 2019, 16.

Gall, R., Troester, F., &Mogan, G. (2010). On the development of an expiramental car-linke mobile robot. *12th International Conference on Optimization of Electrical and Electronic Equipment,Basov, Romania: IEEE*, 734-739.

Bordin, A. (2015). Line Follower Robot with Android and Arduino. Retrieved November 28, 2019, from Hackaday: https://hackaday.io/project/5216-line-follower-robot-with-android-and-arduino.

Rababaah, R. A., & Rabaai, A. A. (2017). Utilizing of Robotics as Contemporary Technology and an Effective Tool in Teaching Computer Programming. *MTMI Annual International conference on Globalization and Competitiveness in Business & Technology*.