ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: AUTOMATION, COMPUTERS, ELECTRONICS AND MECHATRONICS

Vol. 15 (42), No. 1, 2018 ISSN 1841-0626

***Note*: The “Automation, Computers, Electronics and Mechatronics Series”** emerged from **“Electrical Engineering Series” (ISSN 1223-530X)** in 2004.

Address for correspondence: Liana STĂNESCU

University of Craiova, Faculty of Automation, Computers and Electronics

Al.I. Cuza Street, No. 13, RO-200585, Craiova, Romania

Phone: +40-251-438198, Fax: +40-251-438198

Email: stanescu@software.ucv.ro

We exchange publications with similar institutions from country and from abroad

CONTENTS

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 15 (42), No. 1, 2018

Design of Three-Dimensional Museum-Like Environment by using Virtual Reality

Andrei O. Dragomirescu, Florin Stinga

Automation and Electronics Department, University of Craiova
Craiova, Romania (e-mail: andreid93@gmail.com, florin@automation.ucv.ro)

Abstract: This paper showcases the production steps undertaken in the creation process of a three-dimensional, virtual museum-like environment, accomplished using the cross-platform Unity 3D and the aid of a third-party mixture of media production and editing tools. The consequent goal of the process is creating a commercial application available to users operating on a Microsoft Windows environment or (and) a mobile device running a version of the Android operating system.

Keywords: Virtual museum environment, Unity 3D, Virtual Reality, Windows application, Android application

1. INTRODUCTION

This paper covers the main stages of creating a museum-like environment embedded as application used by any human user on a Windows or Android operating system platform. Over the last years, the field of virtual world design is a new area of research without a strong theoretical or methodological foundation (Jakobsson, (2018)).

Virtual reality uses a set a technique that artificially creates immersive sensory experience of physical presence in places in the real or imagined world, allows the interaction between the user and that world (Ferrari and Medici, (2017)). This technique has been applied in many areas of science and visual communication, such as restoration of cultural relics, indoor decoration design, landscape design and so on, allowing access from different places and times, even that the original sites are temporal or geographical inaccessible to the human users (Ferrari and Medici, (2017), Yubin and Yufen, (2014), Higgins et al., (1996), Maietti et al.(2017)).

The showcased application is one depicting a medieval settlement, structured in such fashion to allow the final user to traverse the medium in a free manner, immersing himself in a museum-like, three-dimensional virtual representation with the extended possibility of virtually guiding said user. The application developed implies the mixed usage of several applications, used in media creation and editing processes, whose results can and are introduced in the main engine at every step in the development process.

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CCCDI–UEFISCDI, project number 78 BM.

Unity 3D is the main engine used to produce the virtual environment, the graphical and programmable medium, in order to embed the externally developed assets, and the final rendering tool used in the building process (Unity 3D, (2018)). Despite the fact that Unity is an integrated environment capable of offering all the necessary tools required to produce inside the engine itself, several other software third-packages are utilized in the presented application in this article, with the sole goal of creating a virtual medium as realistically close as possible. The current graphical capabilities of the computational hardware in today's world makes such objectives quite reachable and not that difficult to implement and run natively on several platforms, whether they seem to be a bulky tower personal computer, or a small smartphone tucked in one of your pockets. The complementary software packages used in the process of development are as follows: 3DS Max (3DS Max, (2018)), Ableton Live (Ableton Live, (2018)), Adobe Photoshop (Adobe Photoshop (2018)), FMOD Studio (FMod Studio, (2018)), SpeedTree (SpeedTree, (2018)).

An important aspect that requires mentioning, is that the programming, although implementable in various ways and languages in the Unity Engine, is done extensively in C# (C Sharp), due to its versatile way of approaching object-oriented programming and the various advantages related to the integration with the .NET Framework and its application programming interface.

This article is structured as following: In Section 2 both the medium development and the control solution utilized in the traversal of the virtual environment is summarily, for both Windows and Android operating systems, showcasing the slight variations in the input and response of the developed system-based applications, Section 3 describes the process of creation and implementation of the layers and assets used for the virtual immersion of the user in the virtual medium, in Section 4 the integration of all the previously developed elements into a built and executable, end-user application, is introduced, and Section 5 presents conclusions from the presented studies.

2. VIRTUAL MEDIUM

Both the medium's various terrain and architectural representations and its control scheme used in the traversal of said medium, are structured in the Unity project hierarchy as "Assets". These assets can be regarded as components and game objects, and can range from the three dimensional medium itself, all the way to the architectural elements and the virtual foliage representations. The terrain component can quite easily be instantiated as a sterile three-dimensional plain, devoid of any other elements that will be added at future steps in the process. Unity has an integrated set of tools that will allow an in-depth shaping of the terrain, to its utmost finest detail, offering the possibility of creating quite a realistically (or if desired, a surreal) representation of landforms. A quick visual example of the afore mentioned can be observed in the Figures 1 and 2, captured in order to showcase the intermediary and final results of the creation, shaping and texturing of the terrain-environmental elements.

Unity offers the capability of using a mixture of textured graphical elements, from both inside the included Unity packages and external sources. Several such terrain and architectural textures are introduced into the project and used after several subsequent editing steps in Adobe Photoshop, as it shown in Figure 3. The control solution implemented, just as the previous elements, is structured as a game object. More specifically, this object implies an implementation of a parent-child like relationship between two assets. The two assets are categorized as:

FPC – The First-Person Controller: In this scenario, the parent of the group, responsible for the scripting aspects of the control solution and the process of control regarding the collisions and behavioural interactions with other elements present in the virtual medium.

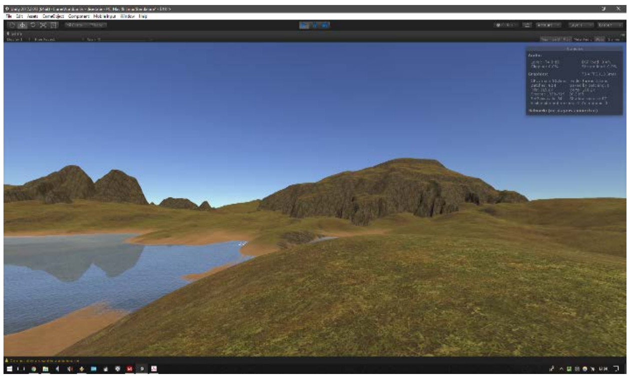


Fig.1: Running in - Engine Intermediary Terrain Preview

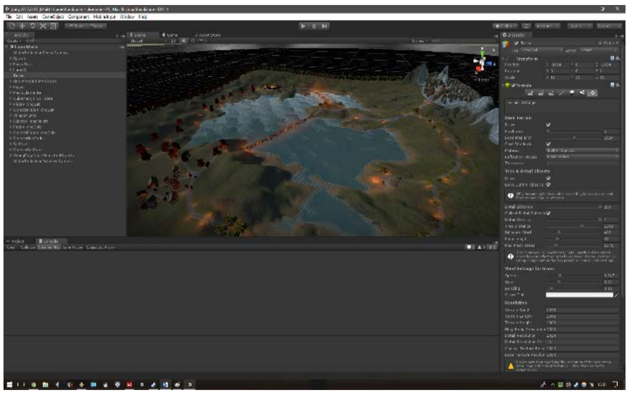


Fig. 2: Final reference of the virtual medium in Unity

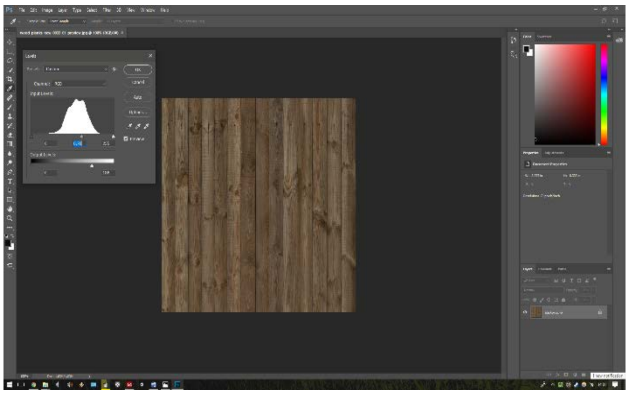


Fig. 3: Graphical elements editing done in Photoshop

Camera – The child, positioned at the "eye-level" of the control object, used as a point of view for the end user.

In Unity, programming and modification of the scripting elements is achieved mainly in the Microsoft development environment, Visual Studio Basic (Visual Studio, (2018)), bundled together with the initial installation of the Unity client (C# code extract) (Dragomirescu, (2018)):

```
//...
if (m_CharacterController.velocity.sqrMagnitude > 0 && (m_Input.x != 0 || m_Input.y != 0)){
m_StepCycle+=(m_CharacterController.velocity.magnitude+ (speed*(m_IsWalking ? 1f : m_RunstepLenghten)))* Time.fixedDeltaTime;}
if (!(m_StepCycle > m_NextStep)){
return;}
m_NextStep = m_StepCycle + m_StepInterval;
PlayFootStepAudio();}
private void PlayFootStepAudio(){
if (!m_CharacterController.isGrounded){
return;}
//…
if(m_CharacterController.velocity.magnitude>0&&m_CharacterController.isGrounded){
```

```
m_Camera.transform.localPosition=m_HeadBob.DoHeadBob
(m_CharacterController.velocity.magnitude +
(speed*(m_IsWalking ? 1f : m_RunstepLenghten)));

newCameraPosition = m_Camera.transform.localPosition;

newCameraPosition.y = m_Camera.transform.localPosition.y -
m_JumpBob.Offset();}

//…
```

Due to the variations of the input offered by the Windows platform, which implies the usage of keyboard-mouse combination or other controllers, and the compact devices running an Android operating system (like smartphones and tablets), which use a touch input, the control solution needs to possess a platform specific set of instructions, that will execute only on the specific platform the application is built for. The previously mentioned, programmable component present in the FPC, can be carefully modified to the extent of allowing and executing platform specific code in order to satisfy the needs for multi-platform application development (C# code extract) (Dragomirescu, (2018)):

```
//...

if(!FindObjectOfType<EventSystem>()){

GameObject obj = new GameObject("EventSystem");

obj.AddComponent<EventSystem>();

obj.AddComponent<StandaloneInputModule>().forceModuleAc
tive = true;}

//…

namespace
UnityStandardAssets.CrossPlatformInput.PlatformSpecific

{public class StandaloneInput : VirtualInput{

public override float GetAxis(string name, bool raw){

return raw ? Input.GetAxisRaw(name) : Input.GetAxis(name);}

public override bool GetButton(string name){

return Input.GetButton(name);}

public override bool GetButtonDown(string name){

return Input.GetButtonDown(name);}

public override bool GetButtonUp(string name){

return Input.GetButtonUp(name);}

//…
```

The Android version of the developed application has an additional graphical layer, of visual toggles, that implements equivalent controllable functions available on the keyboard and mouse version of the application.

Combining the two developed assets and packaging them in an executable fashion at this point in the development process, will allow the developer, artists, programmers and testers, to have a general idea regarding the possibilities and limits of the following developmental steps, and assure the quality of the previous and future steps that must be implemented.

3. IMMERSION

Unity allows its developers to both develop many re-usable assets inside the engine itself, or import externally developed assets, from its own Store, the Unity Asset Store or elsewhere. Game objects that are developed to be easily re-used are referred to as Prefabs (Prefabricated Objects). The prefabs are essentially templates of game objects that can be easily instantiated and quickly modified in other to create variations of the previously developed assets. Immersion is an experience in one moment in time involves a lack of awareness of time, a loss of awareness of the real world, involvement and a sense of being in the task environment (Jennete et. al, (2008)). An immersive world, even a virtual one, requires variation in its implementation of visually related elements. A quite repeatable visual pattern of the same graphical elements can detract from the experience and lower the quality desired by the developer's previously set objectives. The prefab concept can be quite the helpful solution to tackle such an issue.

Prefabricated elements can vary wildly from a complexity's standpoint in the developed application, but in itself the process of creating and modifying such an element remains fundamentally the same, whether implemented piece by piece inside Unity or in a third-party software (e.g. 3DS Max) and then imported in Unity. The Figures 4-7 shows the steps followed for one such asset inside the Unity engine.

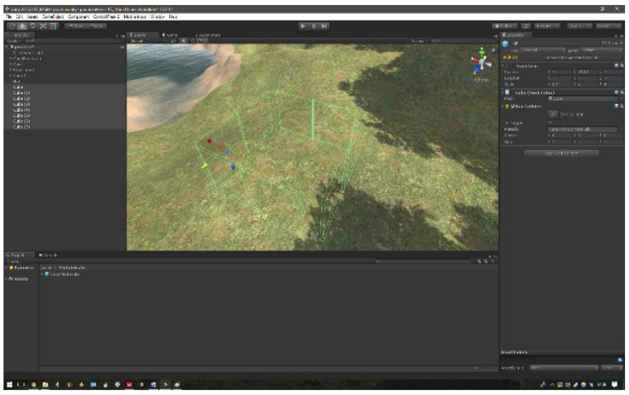


Fig 4: Prefab Step 1 – Creating the sub - components

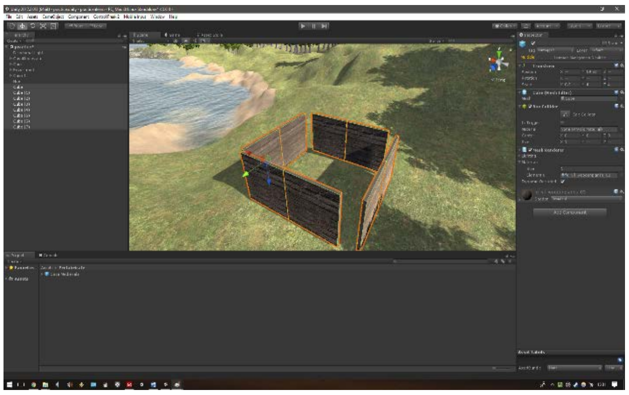


Fig 5: Prefab Step 2 – Attaching graphical meshes

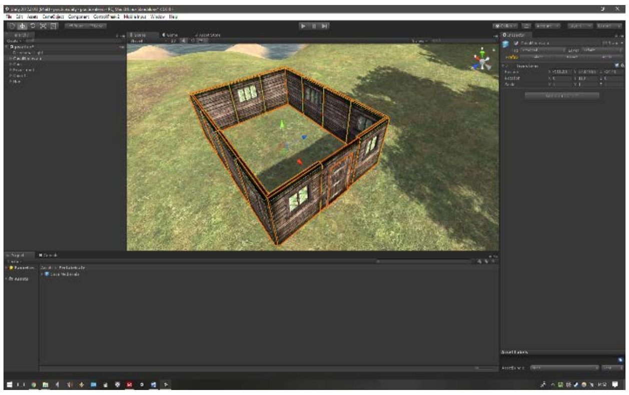


Fig 6: Prefab Step 3 – Arranging of resulted components

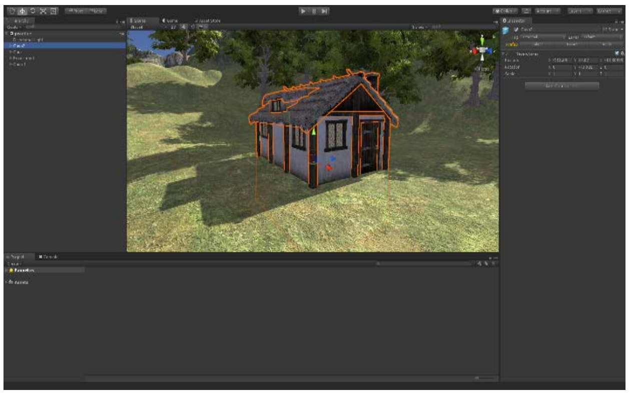


Fig 7: Prefab Step 4 – Grouping components as a Prefab

The usage of the prefabricated elements can be taken to new levels, when its implementation is brought to a smaller, detail-oriented application on objects used to populate the environment, in order to create a museum-like visual medium.

In the current application, the goal was to simulate a historical medieval setting, so the result is modelled to appear like an architecturally realistic setting that could have theoretically been populated at one point in the past as presented in Figures 8 and 9.

To further increase the quality and immersion of the virtual world, several other operations such as adding elements of foliage and the implementation of auditory elements are introduced.

Auditory elements are introduced into the Unity project with the combined help of the following software packages: Ableton Live 9 and FMOD Studio (see Figures 11 and 12).

Foliage is implemented through SpeedTree (see Figure 10). In itself the process of creating the foliage elements is quite easy to grasp and implement. The structure of each element is usually an inter-linked list of generated constructive elements, textured appropriately to resemble a realistic (or if desired, a surreal), equivalent of a real-life foliage element. The composition and editing of the used auditory elements are done entirely inside the digital audio workstation, Live 9, and subsequently imported into FMOD Studio as high-fidelity, sequenced audio files.

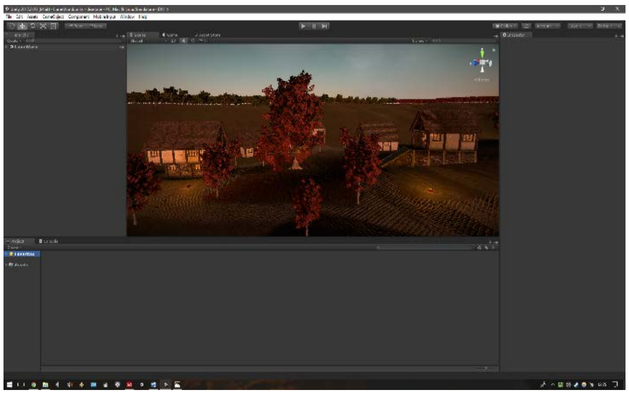


Fig. 8: Virtual Medium – Outdoors Medieval Setting

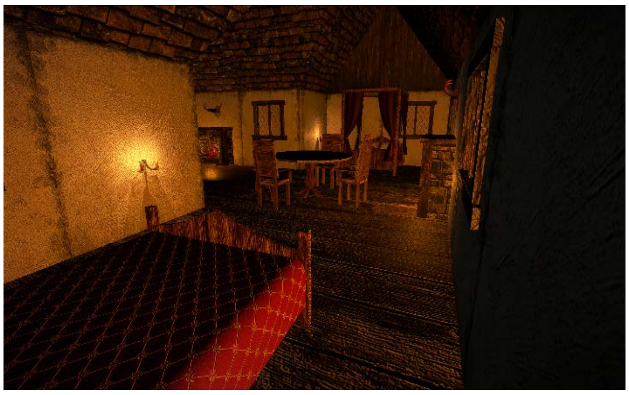


Fig. 9: Virtual Medium – Indoors Medieval Setting

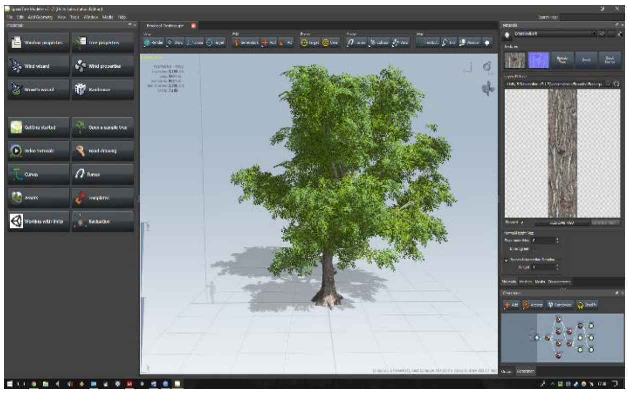


Fig. 10: Foliage elements done in SpeedTree

Then, these files are programmed via C# scripts inside the Unity project itself. FMOD acts as an intermediary, which allows high quality conditioned playback of the audio files, according to the rules and sets of instructions set in the afferent Unity project.

4. BUILDING

In the Unity environment, the final step that needs to be undertaken in order to finalize the project and obtain a structured installer file (in the case of an Android build), or an executable file (in the case of a Windows build).

Building options vary from platform to platform. Each and every build implements some lesser or higher quality form of texture compression, based on Unity texture compression algorithms. The lack of applied compression is also an available option, should this be the desired outcome. In Figure 13 is presented the Build Menu inside Unity Engine.

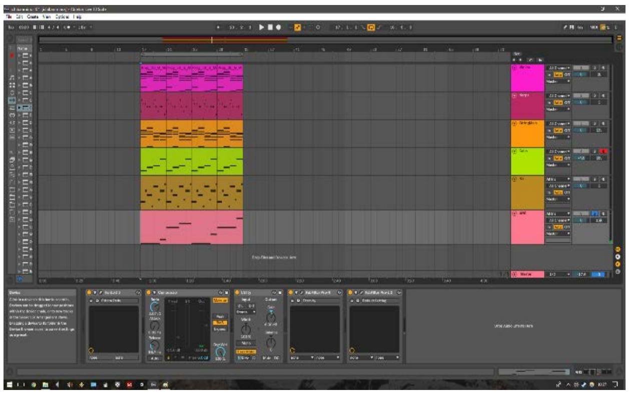


Fig 11: The Ableton Live environment – the musical score composition and the signal processing done on one of its generated tracks.

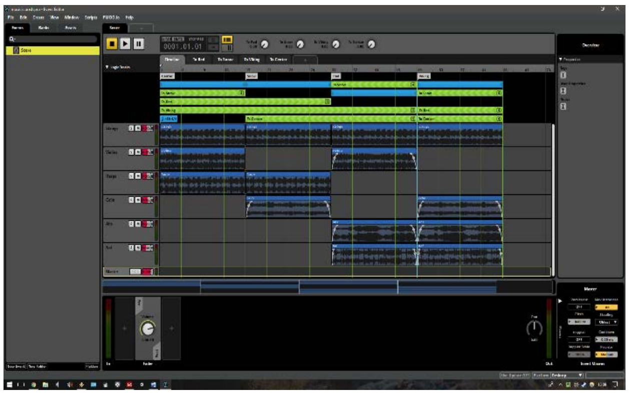


Fig 12: FMOD Interface – High fidelity audio files are structured and sequenced according to the desired programmable structure

Whilst the building process of a Windows version is quite straightforward, requiring only the selection of the Build option, the Android built will require an automated conversion of the used assets, using the Android SDK (Software Development Kit), alongside its afferent app-developing software package, Android Studio.

The engine offers building for many platforms, not just Windows/Android. Unity allows building for commercial hardware platforms such as iOS, Mac, Linux, Sony PlayStation, Xbox One, Oculus Rift, Steam VR and even TV's powered by the Android and Apple operating systems alongside many other more.

The developed application for the Android operating system can be installed and ran not just smartphones but on Android TV's just as effectively, but this implementation would most likely require an additional set of control instructions, specific to the TV operating system environment.

As previously mentioned the application described is built for both the Windows and Android operating systems.

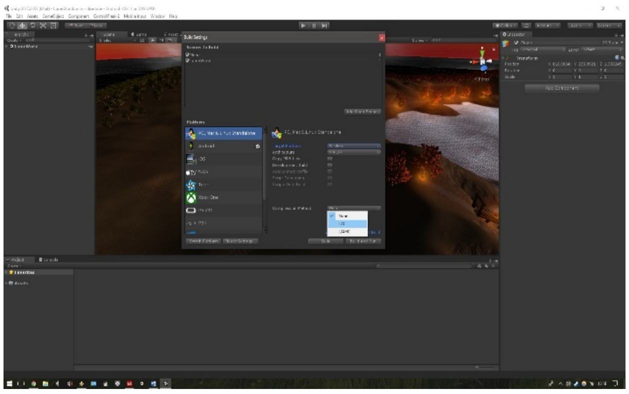


Fig 13: The Build Menu inside Unity Engine

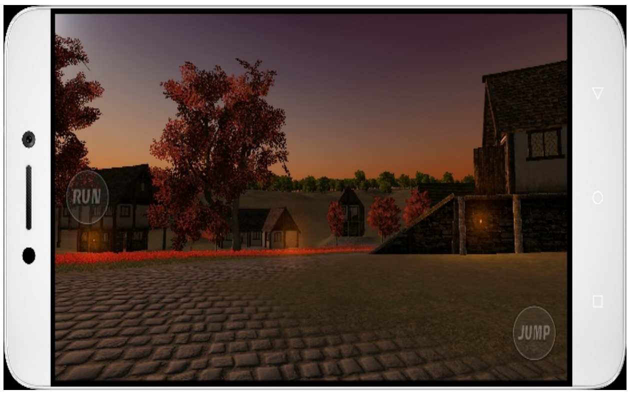


Fig. 14: The final Android interface implemented on a mobile device

In Figure 14 is presented the final Android interface of the developed application.

While the Windows version is not concerned with trivial aspects such as texture compression due to higher powered systems that usually run Windows as an operating system, the Android version has applied on itself a form of light compression that will maintain a very close high-fidelity look, just as its Windows equivalent. This is introduced due to subjective observations, regarding the implementation of a more optimized high-frame per second functionality on mobile devices and may be detrimental to users whom value the visual aspects more than an optimal performance and a lower battery consumption on their mobile Android devices.

5. CONCLUSIONS

In this paper were presented the main stages of creating a medieval settlement, structured in such fashion to allow the final user to traverse the medium in a free manner, immersing himself in a museum-like, three-dimensional virtual representation, embedded as an application usable by any human user on a Windows or Android operating system platforms.

Virtual reality is a technique used in presentation of a heritage application allowing access from different times, even that the original sites are temporal inaccessible to the human users.

Despite the rather complicated approach used in the development of this application, using Unity 3D and several third-party software packages, a virtual medium as realistically close as possible is created.

6. REFERENCES

3DS Max, Autodesk Inc., available on-line at https://www.autodesk.com/products/3ds-max/overview, 2018.

Ableton Live, Ableton AG, available on-line at https://www.ableton.com/en, 2018.

Adobe Photoshop, Adobe Inc., available on-line at https://www.adobe.com/ro/products/photoshop.html, 2018.

Dragomirescu A., Aplicație in Unity 3D, Graduated Thesis, *Faculty of Automation, Computers and Electronics, University of Craiova*, 2018.

Ferrari F. and Medici M., The Virtual Experience for Cultural Heritage: Methods and Tools Comparison for Geguti Palace in Kutaisi, Georgia, *Proceedings of International and Interdisciplinary Conference IMMAGINI?* Brixen, Italy, 2017.

FMod Studio, Firelight Technologies Pty Ltd., available on-line at https://www.fmod.com, 2018.

Higgins T., Main P., Lang J., Imaging the Past: Electronic Imaging and Comouter Graphics in Museums and Archaeology, *British Museum, UK*, 1996.

Jakobsson M., A virtual realist primer to virtual world design, available on-line at http://www8.informatik.umu.se/~mjson/files/primer.pdf, 2018.

Jennete C., Cox A., Cairns P., Dhoparee A., Epps A., Tijs T. and Walton A., Measuring and Defining the Experience of Immersion in Games, *International Journal of Human-Computer Studies* 66(9):641-661, 2008.

Maietti F., Di Giuliom R., Balzani M., Piaia E., Medici M., Ferrari F., Digital Memory and Integrated Data Capturing: Innovations for an Inclusive Cultural Heritage in Europe through 3D Semantic Modelling, In Mixed Reality and Gamification for Cultural Heritage, *Springer, Berlin*, pp. 225-244, 2017.

SpeedTree, IDV Inc., available on-line at https://store.speedtree.com, 2018.

Unity Technologies, Unity 3D, available on-line at https://unity3d.com, 2018.

Visual Studio, Microsoft Corporation, available on-line at https://visualstudio.microsoft.com, 2018.

Yubin L. and Yufen F., The virtual reality tehnique of lanscape arhitecture reconstruction, *BioTechnology Journal*, 10(11), pp. 5226-5233, 2014.

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 15 (42), No. 1, 2018

Automotive Application for Collision Avoidance

Florina Luminița Besnea*, Ștefan-Irinel Cismaru**

** Department of Mechatronics and Robotics, University of Craiova
Craiova, Romania (e-mail: besnea.florina@yahoo.com)
** Department of Automatic Control and Electronics, University of Craiova
Craiova, Romania (e-mail:cisstefan@gmail.com)*

Abstract: Car transport is crucial for society and has grown in terms of both number and complexity of the vehicles, their interaction and the traffic situation. The emergence of the technology focuses mainly on reducing human effort in every field. This paper proposes the development of an automated, easy-to-integrate and cost-effective electronic system model that will help reduce collisions. The developed system can be considered as a step towards minimization of mental and physical efforts made by the driver to control the vehicle safely. Collision avoidance is a research area that has become more relevant in recent years as the vehicles have been improved by adding driving assistant systems.

Keywords: automotive, car, electronics, sensorics, collision.

1. INTRODUCTION

Advanced driver assistance systems are developed to automate, adapt and improve vehicle systems for safety and a better driving manner. The automotive safety system's journey begins in 1901 with the Oldsmobile Curved-Dash, which offered a foot brake by direct pressure of a rail on the transmission shaft.

The vehicle has been used as a means of transport over 100 years. A few decades ago, electronic technologies and sensors were introduced to vehicles as intelligent steering assistance systems. This kind of system helps drivers in guidance for safety and comfort. Smart systems offer great potential for future mobility. Several sources indicate that the benefits of intelligent driver assistance systems are significant. The sensors monitor the driving circumstances and detect dangerous events, filling the sense of vision, distance and orientation of the human being.

A large part of the total vehicle accidents is due to the lack or decrease in the concentration of the drivers during the operation of the vehicles. Some drivers tend to handle distracting activities such as radio tuning, eating, talking to passengers or, the most common, making and taking phone calls. Other drivers find it difficult to focus only on driving, for example because of fatigue and health problems. Elderly drivers may experience difficulties in personal mobility, making it more difficult to constantly monitor the vehicle's perimeter. They can also develop other conditions that have a negative impact on their ability to focus on the road.

Failure to identify a vehicle on the side of the car, as known as the blind-spot, is still a cause of the increasing number of accidents. For some drivers, the simple solution is to place an additional side mirror. However, this is not the best solution because these additional side mirrors do not give an accurate picture of the actual or estimated distance to an object or other vehicle.

Today's vehicles are in a much better position from the point of view of safety, but unfortunately today drivers have at their disposal different devices that distract them when handling a car, causing more and more challenges in terms of avoiding possible collisions in traffic.

2. RELATED STUDIES

The main trends of today are oriented towards the effort for efficient transport combined with managing the high demands of society in terms of transport safety. Official accident statistics show that the number of vehicles and accidents has increased over the last 50 years. Automotive safety systems and applications play an important role, with the predisposition for modern cars to become 5% safer each year.

Road safety statistics for 2016, published by the European Commission, show a 2% decrease in the number of deaths registered in the EU last year. In 2017, 25,500 people were killed in road accidents in the EU, 600 fewer than in 2015 and 6,000 fewer than in 2010. Other 135,000 people suffered serious injuries, according to Commission estimates. This statistic shows the increase in population awareness for automotive applications and systems that are based on detecting or preventing an accident.

However, Romania is on the penultimate place in terms of the number of people killed in road accidents per one million inhabitants. Thus, Romania recorded 97 deaths per million, being overtaken only by Bulgaria, with 99 deaths per million.

From the annual publication of the National Institute of Statistics on road traffic accidents involving persons under the influence of alcohol in 2017, it appears that in Romania there have been 3801 accidents that have resulted in death or injury to one or more persons. Of this number, the number of accidents involving directly car drivers was 1688.

The study shows that the most frequent road accidents involving vehicles are produced in rural areas in 55%, in urban areas 37%, and on motorways they cover an 8% pricing.

3. COLLISION OVERVIEW

Collision is an event in which two or more bodies exert their forces on each other within a relatively short period of time. A traffic collision is called a collision with a motor vehicle collision (MVC) and occurs when a vehicle collides with another vehicle, pedestrian, animal, roadside remnants or other stationary objects such as a tree or a pillar (Jonas Jansson 2005). These collisions often result in injuries, deaths, and material damage.

A number of factors contribute to the collision risk, including vehicle design, operating speed, road design, road environment and driver qualification. Worldwide, collisions with cars lead to death and disability, as well as financial costs for both the company and the people involved in such attacks.

2.1 Collision stages

The pre-impact period is called an ante-collision stage where there is no contact between cars. It is delineated in time from the initial moment of impact and the moment of triggering the imminent danger, the main objective being to assess the possibilities of avoiding the accident.

The collision stage occurs at the time of first contact between the vehicles and the moment of their separation. In this phase of collision the deformation energy is obtained from the kinetic energy, the contact between the vehicles having two different phases: compression and restitution.

The post-collision stage is triggered when the cars are separated and lasts until they are stopped.

The blind-spot is the area around the car that cannot be directly observed by the driver during driving.

2.2 The blind-spot

Blind-spots are found in several categories of vehicles: cars, ships and aircraft. Correctly adjusting the mirrors and using technical solutions can eliminate or attenuate the blind-spot.

It has been statistically proven that many accidents occur when changing the lane when an overtaking manoeuvre is initiated, and on the relevant lane there is already another overtaken vehicle.

Checking with rear-view mirrors without checking a blind-spot is not enough. It should be noted that if a vehicle is located at a very narrow distance on the left or right, it may be in the mirror area that suffers from lack of visibility. Blind-spot checking lasts a few fractions of a second, and the prolonged viewing of the angle at this point is dangerous due to the rapid change in traffic. In order to avoid such a dangerous situation, certain collision detection and prevention systems have been developed in the dead corner.

2.3 The side collision

The blind-spot monitoring system continuously scans both sides of the vehicle with ultrasonic sensors and warns the driver when another vehicle is in the detection area. The scanned area is always larger than the blind-spot area itself. The detection area starts at 40 cm from the vehicle at the level of the rear-view mirror.

The side collision detection sensors operate on the same principle as the front collision with the indication that the measures applied in case of detection are not as severe.

4. SYSTEM DEVELOPMENT

Each ultrasonic sensor is interfaced with Arduino to determine the distance to the obstacles.

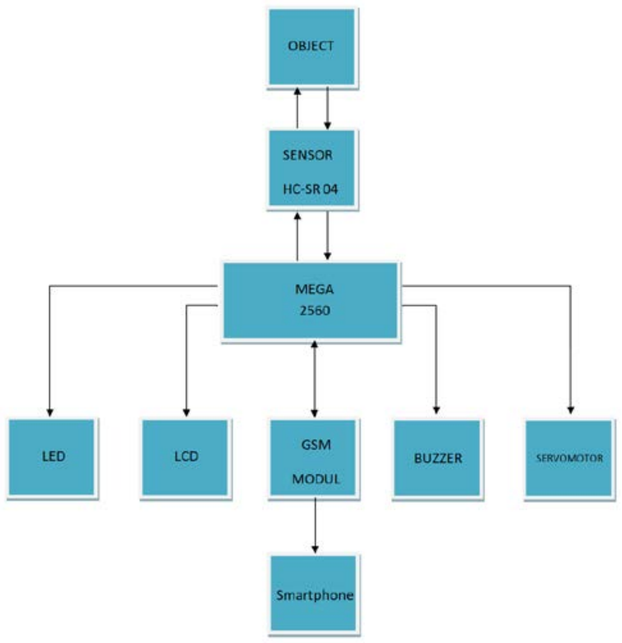


Fig.1 System block diagram

Circuit design, simulation, and block diagram are used to explain the processes involved during the project to achieve the objectives. The block diagram completely covered the process, from the ultrasonic sensor to the system.

The block diagram shows how an object is detected by the Arduino Mega 2560 by means of ultrasonic sensors and the actions taken by it are based on the detected distance. Sensor distance measured is within a predefined warning level and a complementary measure determined by the critical level is applied. The application of complementary measures is directly proportional to the hazard reported by the level at which it is framed.

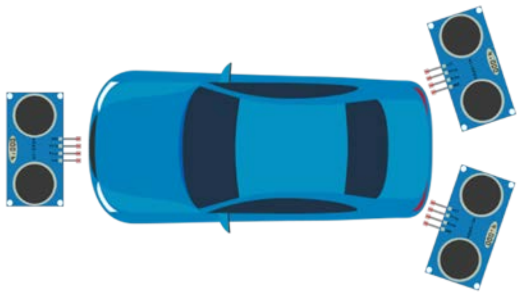


Fig. 2 Sensor placement on vehicle platform

The way the system responds to stimuli can range from a luminous or audible warning, representing the least dangerous thresholds, to braking, and even sending a message, in case of impact, to competent authorities and even close relatives.

5. CIRCUIT DEVELOPMENT

Many sensors are available on the market and can be used. Some of the most common sensors are: laser sensors, ultrasound sensors, infrared sensors and radar sensors. After performing various studies, ultrasound sensors have been selected because of their relevance to this project, easy to use and at low cost.

On the basis of the block diagram, the assembly was made through the general schema of connecting the hardware components to the Mega 2560 microcontroller:

• Ultrasonic sensors HC-SR04

• The GSM (Global System for Mobile) shield SM1500 B-D

• GSM/GPRS antenna

• I2C LCD screen

• Actuator SG 90

• LEDs

• Piezoelectric speaker (buzzer)

In order to use the GSM Shield and the Mega 2560 microcontroller, it was necessary to attach some bearers on the GSM module.

Thus, connecting the shield over the development pad allowed the wires and sensors to be connected without scattering the slot space available on the Mega 2560. The attached pins were inserted into the slots for each of them without the risk of a wrong connection that might burn the board.

The GSM shield must use an antenna, which is necessary to create both connection and transmission of SMS messages. To send SMS messages, it was used a pre-paid card with deactivated voicemail. The sim is a normal sized one, since a micro-sim or nano-sim card makes it impossible to fit into the available slot.

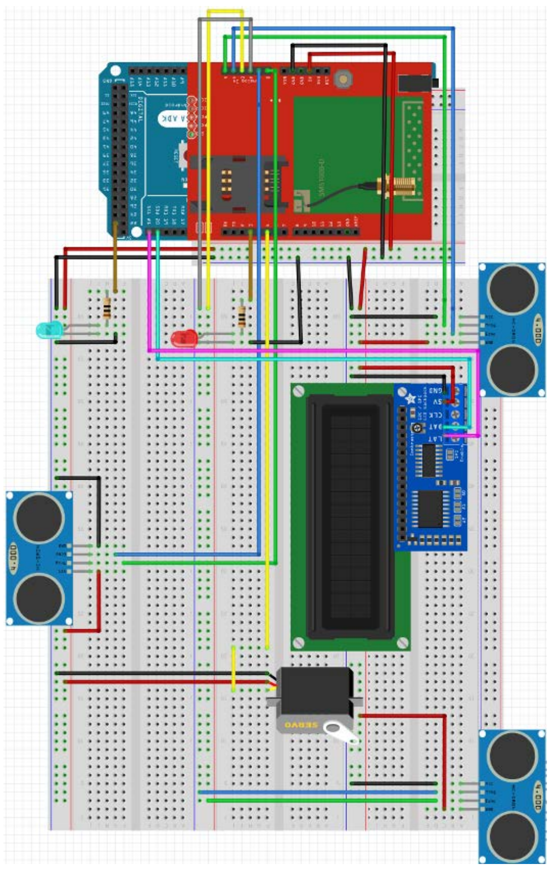


Fig.3 General scheme of the automotive system

6. METHODOLOGY AND SYSTEM OPERATION

61 Detection range

The system performs detection by means of two ultrasonic sensors placed on each side of the vehicle in front of the rear wheels. When the system recognizes a vehicle in the detection area, the driver is warned by light signals in the rear-view mirrors.

For the right-handed alert, a red LED was used, and for the left a blue LED, which changes its luminous intensity according to the distance of the other vehicle.

The ignition frequency of the LEDs is given by the following distances:

• At less than 40 cm, the LED changes its ignition frequency every 300 milliseconds;

• At less than 30 cm, the LED changes its ignition frequency every 150 milliseconds;

• At less than 20 cm, the LED changes its ignition frequency every 50 milliseconds;

• At less than 10 cm, the LED stays lit, with the collision probability most likely to occur.

The action range is set to limit the detection to a certain distance, thus excluding parked vehicles, infrastructure, vegetation, vehicles moving from the opposite direction, and does not issue warnings about them.

6.2 *Levels of collision*

The proposed system is built on a scale of 1:10 compared to everyday vehicles and has four levels of system activation to detect a collision. The system operates on 4 levels as follows:

Level 1 - over 40 cm
Level 2 - over 30 cm
Level 3 - over 20 cm
Level 4 - over 10 cm

In the case of a front collision, the sensor located at the front of the car detects Level 1 of a collision at a distance of 40 cm. Thus, at the time the driver is warned of producing a possible collision by:

• LCD display of "warning1" and the distance to the vehicle or front object;
• The sound signal transmitted by the piezoelectric speaker;
• Warning light produced by LEDs.

The same is true for Levels 2 and 3 when the distance becomes smaller, the buzzer sounding.

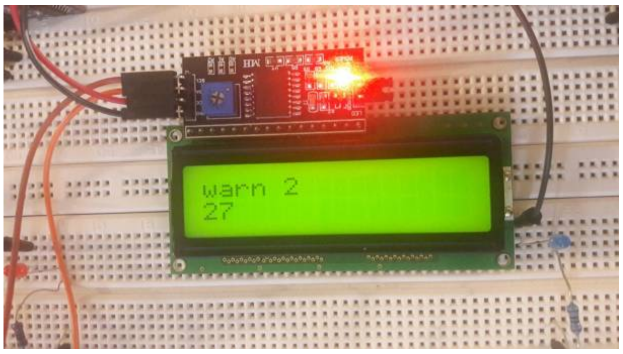


Fig.4 The message displayed on a possible Level 2 collision

The vehicle in a level 3 collision is easily broken by actuating the servomotor on the front wheels; the message on the LCD is "warn3".

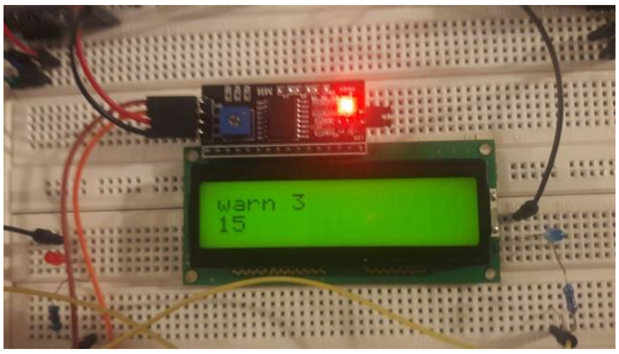


Fig.5 The message displayed at a possible Level 3 collision

Level 4 is closest to collision occurrence, often unavoidable. In this case, the message displayed on the LCD screen is "Accident" along with the distance to the accident.

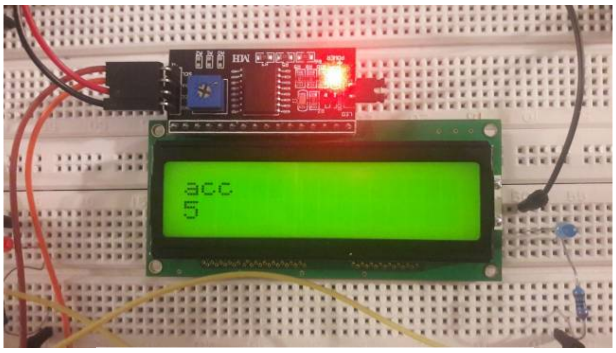


Fig.6 The collision display message - Level 4

If the critical level is reached, for example Level 4, the system automatically actuates the brake actuator, braking being stronger.

The frontal distance detection functionality of the object is obtained by constant checking of the variable [0] to fit one of the 4 risk levels. The cm [0] variable is part of a vector where the distances for each sensor running on the system are stored. Each value is updated to a 33-millisecond range, representing the ideal range for ultrasonic sensors to read, thus avoiding interfering with one another.

The side collision detection sensors operate on the same concept as the one detecting front collision, with the mention that the measures applied in case of detection are not as severe.

7. CONCLUSIONS

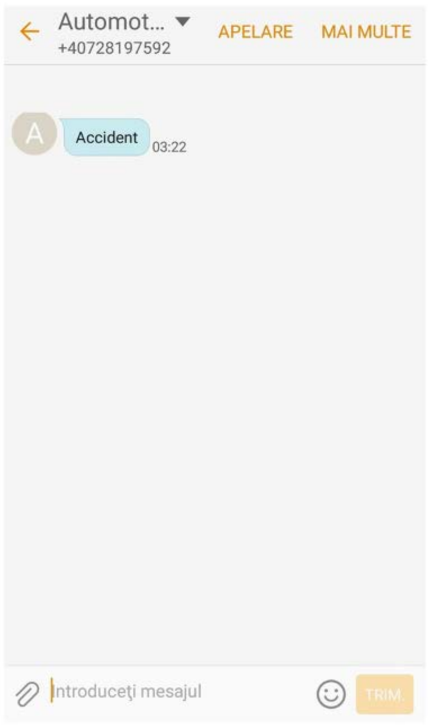
The collision prevention system is a car safety system designed to reduce the possibility and severity of an accident. These systems evolved over time from the most primitive to the most intelligent systems of the present. Modern intelligent systems are also known as pre-accident systems, pre-collision warning systems or collision abatement systems; they use electronic circuits connected to remote sensors and sometimes to detect an imminent accident.

Once detection has been made, these systems provide either a warning to the driver when an imminent Collision occurs or take measures autonomously without involving the driver in the braking maneuver.

The purpose of this work is to reduce the number of car accidents, while reducing the chances for a person to be injured in such a situation.

Future development can be guided to collect the data from the sensors and to use it as an input for defining if a person drives defensive. Also, the data will help to determine a driver’s profile which an insurance company can take into consideration if offering bonuses.

Another important improvement would be the use of a GSM shield to send a warning message "Accident" to a relative, a person close to the driver or the competent authorities.



Automot... +40728197592 APELARE MAI MULTE

Accident 03:22

Fig.7 Collision Warning Message

The collision prevention system proposed in this paper aims to implement a smart system based on a microcontroller that will use obstacle detection by using ultrasonic distance measurement sensors to detect obstacles and their distance. Requirements will continue to grow in the future, motivating ongoing research into improving new safety systems.

REFERENCES

A. Eidehall, Tracking and threat assessment for automotive collision avoidance, ISBN 91-85643-10-6, 2007

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.604.2183&rep=rep1&type=pdf

A. Goodchild, et al. Intersection Decision Support (IDS): An implementation for collision avoidance, *Proceedings of 10th ITS World Congress*, 2003

J. Jansson, Collision Avoidance Theory with Application to Automotive Collision Mitigation, ISBN 91-85299-45-6, 2005

https://www.researchgate.net/publication/324537890_Collision_Avoidance_Theory_with_Application_to_Automotive_Collision_Mitigation

K. Morita, et al., Experiments of Inter-Vehicle Communications to Prevent Crossing Collisions and a New Proposal of Communication Systems, *Proceedings of 11th ITS World Congress*, 2004

M. Dhivya, S. Kathiravan, Driver Authentication and Accident Avoidance System for Vehicles, 2015

https://nanopdf.com/download/driver-authentication-and-accident-avoidance-system-for-vehicles-t_pdf

National Institute of Statistics, Vehicule inmatriculate si accidentele de circulatie rutiera, ISSN 1841-3528, 2017

http://www.insse.ro/cms/sites/default/files/field/publicatii/vehicule_inmatriculate_in_circulatie_si_accidente_circulatie_rutiera_2017.pdf

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 15 (42), No. 1, 2018

Hand Gestures Recognition in Images

Claudia Mihaela Bînă

Computers and Information Technology Department, University of Craiova, Romania
(e-mail: claudiamihaelabina@gmail.com)

Abstract: Hand gesture recognition systems can be used as instruments to simplify the way humans interact with computers and other smart electronic devices that are bolstering our home and office comfort or security day by day. In this paper, a hand gestures recognition system has been implemented, that is able to recognize 10 static gestures from The National University of Singapore hand posture datasets. The first stage of the proposed system consists in detecting a bare hand in a plane background image and extracting the Hu invariant moments using Open Computer Vision (OpenCV) platform for Python. The second stage takes as input the characteristics file and, using A Library for Support Vector Machines (LIBSVM), trains and generate the classifier model. Finally, it provides the option to recognize hand gestures using multi-class Support Vector Machines (SVM) algorithm.

Keywords: Human Computer Interaction, Machine Learning, Hand Gestures Recognition, LIBSVM, OpenCV, SVM

1. INTRODUCTION

Gesture was the first mode of communication for the primitive cave men [Arpita et.al (2013)]. It can also be considered the most simple way of communication, if we think it is used not only to interact with another human, but also with other species, and dolphins, dogs or monkeys are reminded here. Later on, human civilization has developed the verbal communication very well. But still nonverbal communication has not lost its weightage. Such nonverbal communication is being used not only for the physically challenged people, but also for different applications in diversified areas, such as aviation, surveying, music direction etc. It is the best method to interact with a computer without using other peripheral devices [Arpita et.al (2013)].

As stated before, we also use gestures to communicate with each other. It is a natural form of interaction, it is easy to do and, most of the time, it does not require much attention. We see devices are becoming increasingly integrated into our surroundings, such as smart devices in homes, wearable devices. Here comes a need for convenient methods to control them which are always available to us, so a need for efficient ways of human interaction with the modern virtual environments. One of all these ways has predominantly came into light during the latest years: the “Hand Gesture” human-computer interaction modality.

How is the “Hand Gesture” interaction modality achieved? The presented paper proposes achieving this interaction modality by making use of machine learning in the process of hand gestures recognition. We see machine learning all around us in the products we use today. It is not always apparent that machine learning is behind it all, even though tagging people or objects inside of photos are clearly applications of it. Recommending the next video to watch is also a power of machine learning. Perhaps the biggest example of all is Google Search: every time we use Google Search, we are using a system that has many machine learning systems as its core, from understanding the text of our query to adjusting the results based on our personal interests.

1.1 System Overview

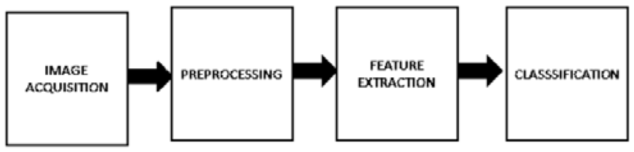


Figure 1 Block Diagram of Hand Gestures Recognition in Images System

Vision based analysis is based on the way people perceive and process information about the surroundings, yet it is probably the most difficult to implement in a satisfactory way. Several different approaches have been explored so far.

- One is to build a three-dimensional model [Matthew et. Al (1991)] of the human hand. The model is matched to images of the hand by one or more cameras, and parameters corresponding to palm orientation and joint angles are estimated. These parameters are then used to perform gesture classification.
- Second one is to capture the image using a camera, extract some features and use those features as input in a classification algorithm [Edward et. al (1991)].

This paper takes as a model the second approach to for designing the whole system. The hand gestures recognition system is a static system receiving the images as an input an not using a camera to capture them. Changing color space to grayscale, smoothing and segmentation are applied to the input image in preprocessing phase then, using contour detection and moments extraction, vector of features is obtained. The classifier is then fed with the outputed vector of features. Support Vector Machine classifier is used.

1.2 Dataset Description

In this paper, all operations are performed on colour images. The first hand posture dataset from The National University of Singapore is used. The NUS hand posture dataset I contains 10 classes of postures, 24 sample images per class, which are captured by varying the position and size of the hand within the image frame. The size of the available images is 160x120 pixels. The hand postures are selected in such a way that the inter class variation in the appearance of the postures is less, which makes the recognition task challenging. The background of the images is uniform.

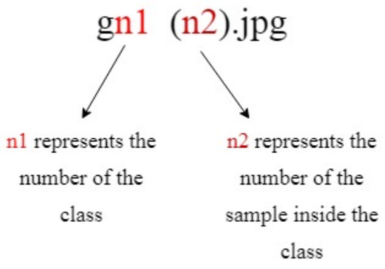

gn1 (n2).jpg
n1 represents the number of the class
n2 represents the number of the sample inside the class

Figure 2 Dataset images naming convention

From the total number of images, 190 samples are taken and used to obtain the most accurate model and 50 samples are used as totally new system testing data. The number of samples used to obtain the most accurate model are split as follows:

- 70% of them are used to actually train the SVM model;
- 30% of them are used to determine and compare the accuracy of each obtained model. In the end, the most acurate model is saved and integrated as the SVM model of the final system.

The system works as offline recognition, which means that it receives a test image as input and tells us the class of the gesture image we have given as input. The system is entirely dependent on data. Depending on the phase – Train or Test - it takes a path to a dataset or to a single image, stored on the disk, and loads the respectively data.

2. PREPROCESSING STAGE

Preprocessing is applied to images in order to facilitate the later features extraction and it consists of three steps:

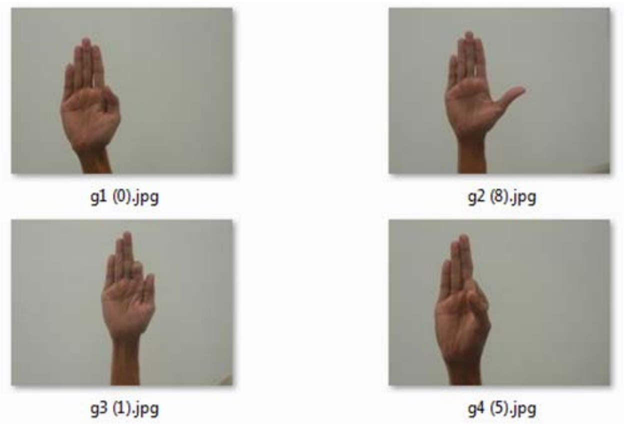


Figure 3 Samples of images from the dataset

- Colour space changing;
- Noise reduction;
- Segmentation.

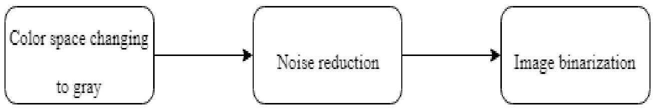


Figure 4 The pre-processing algorithm stages

2.1 Colour Space changing

When reading a colour image using cv2.imread() function, the default OpenCV colour space is BGR and not RGB. The OpenV image binarization can be applied only on grayscale images so it is a necessary step to change the BGR default color space to grayscale.

Figure 5 BGR to Gray Result

2.2 Image Smoothing

Image blurring is achieved by convolving the image with a low-pass filter kernel. It is useful for removing noises. It actually removes high frequency content (eg: noise, edges) from the image. So edges are blurred a little bit in this operation.

Some of the images from the chosen dataset may present different unwanted noise that could affect the segmentation algorithm (the algorithm may not obtain the bimodal histogram necessary for thresholding). To neutralize this noise, the already grayscale image is blurred using averaging technique. This is done by convolving image with a normalized box filter. It simply takes the average of all the pixels under kernel area and replace the central element. Our hand gestures recognition system uses a 3x3 normalized box filter that look like below:

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

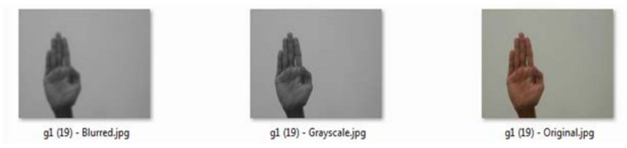


Figure 6 Image Smoothing Result

2.3 Image Tresholding

A very good segmentation is needed to select an adequate threshold of gray level for extract hand from background. In general, the selection of an appropriate segmentation algorithm depends largely on the type of images and the application areas [Sanjay et. al (2011)]. The Otsu segmentation algorithm was tested and found to give good segmentation results for the hand gestures and was, therefore, selected. It basically takes the histogram of the image into account, looks at the pixels' values, these making it a method without parameters and with no need to be supervised.

Otsu's binarization algorithm allows to quickly and automatically obtain the correct threshold value to choose between two histogram mode, so as to apply the thresholding in an optimal manner. In OpenCV, the application of the Otsu's binarization is very simple. It will be sufficient to add as parameter within the cv2.threshold () function, called cv.THRESH_OTSU. The above mentioned thresholding parameter is used together with cv2.THRESH_BINARY_INV parameter with the scope of obtaining an image with a black background and a white foreground – hand area. Such a distribution (white object on black background) is necessary for the next step of the system - computing the image contours. cv2.THRESH_BINARY parameter together with cv2.THRESH_OTSU would have outputted an image with a back object on a white background. This distribution type is not eligible for contour detection.

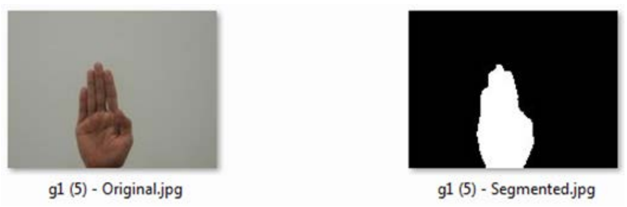


Figure 7 Otsu's Binarization Results

3. FEATURE EXTRACTION

Feature extraction is of great importance because it basically gives the input to the classifier. The first step in this process is contour detection, achieved by applying chain approximation method. The second step is computing the seven Hu invariant moments. Final part of feature extraction is processing the OpenCV output in the LIBSVM accepted data format.

3.1 Contour Detection

A contour is a list of points that represent, in one way or another, a curve in an image. This representation can be different depending on the circumstance at hand. There are many ways to represent a curve. Contours are represented in OpenCV by sequences in which every entry in the sequence encodes information about the location of the next point on the curve [Gary et. al (2008)].

Taking a more simple approach, contours can be explained as a curve joining all the continuous points (along the boundary), having same color or intensity. For a better accuracy in the contour detection, a binary image, with black background and white foreground, is used. In the presented hand gestures recognition system, the binary image was obtained in the previous step by applying Otsu's binarization algorithm.

In the presented hand gesture recognition system, the contour detection is achieved by using cv2.findContours() function, that uses the chain approximation method as a parameter. All the contours are retrieved and put in a list, being easily accessed at the moments coputation stage.

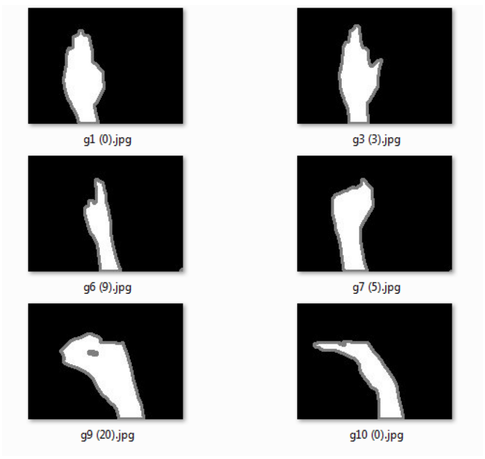


Figure 8 Contour Detection Result Samples

3.2 Moments

Regarding two dimension continuous function *f(x,y)*, the two dimension origin moment of order *(p + q)* is defined as [Ming et. al (1962)]:

$$M_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^p y^q f(x,y)\,dxdy \qquad \text{p, q} = 0,1,2... \tag{1}$$

Under the assumption that *f(x,y)* is a piecewise continuous ounded function and contains nonzero value in *x – y* plane, the according the unicity theorem, moments of each order exist and moments sequence *$\{M_{pq}\}$* is uniquely determined by *$f(x,y)$*, and vice versa. Moreover, the two dimension central moments of order *$(p + q)$* is defined as [Zou et. al (2011)]:

$$\mu_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(x-\bar{x})^p \ (y-\bar{y})^q f(x,y)dxdy \qquad (2)$$
$$p,q = 0,1,2...$$

Where:

$$\bar{x} = \frac{M_{10}}{M_{00}} \qquad \bar{y} = \frac{M_{01}}{M_{00}} \qquad (3)$$

Taking an insight in our application, we notice that the moments are extracted from the hand contours previously computed. OpenCV function that allows this computation is cv2.moments(). It takes as argument the array of contours.

3.3 Hu Invariant Moments

The conception of moment invariant was first proposed by Hu, in which a set of invariants with the characteristics of translation invariance, scale invariance and rotation invariance were obtained by a nonlinear combination of moments.

In practical applications, images can suffer different transformations of translation, scale and rotation. The origin moments or central moments features are not invariant at the same time to all the three types of transformations, so it is difficult to obtain translation, scale or rotation invariants simultaneously. To solve this problem, Hu proposed 7 famous and very useful invariants of moments by applying algebraic invariant theory to scale standardization moments in 1962. The 7 moments invariants are composed of linear combination of second order and third order central moments

In the proposed system, Hu invariant moments are computed from the previously extracted central moments and all 7 of them are used for further classification. OpenCV provides cv2.HuMoments() python routine to compute Hu invariant moments from the central moments. The extracted features are then written in a file which will be the input for the classifier. The output of cv2.HuMoments() function is processed such that the file will have the required LIBSVM file format.

4. GESTURES CLASSIFICATION

While the previous sections have shown how we extract the features from the image, now we use these features to classify the gesture the hand is showing by using a multiclass SVM algorithm.

The SVM algorithm is one of many algorithms for supervised learning. Generally, an SVM is a linear classification algorithm that maximizes the distance between the decision line (discriminator) and the closest example to it in the training set. An illustration is given in Figure 9. Generally an SVM is used to classify only two groups [Ilan et. al (2010)].

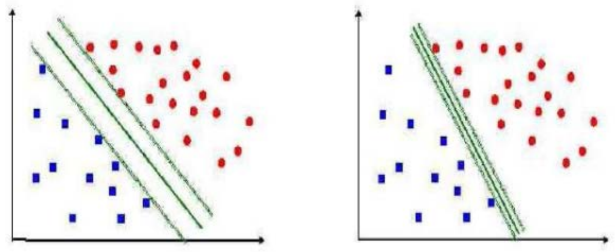


Figure 9 Linear discriminators

4.1 LIBSVM Library

For development of the proposed system, the LIBSVM library for Support Vector Machines has been used. LIBSVM package has been developing since the year 2000 and until today it has become the one of the most widely used SVM software. LIBSVM supports the following learning tasks:

1. SVC: support vector classification (two-class and multi-class)
2. SVR: support vector regression
3. One-class SVM.

A typical use of LIBSVM involves two steps: first, training a data set to obtain a model and second, using the model to predict information of a testing data set. For SVC and SVR, LIBSVM can also output probability estimates. The SVM formulations supported in LIBSVM are: C-support vector classification (C-SVC), ν-support vector classification (ν-SVC), distribution estimation (one-class SVM), ϵ - support vector regression (ϵ - SVR), and ν-support vector regression (ν-SVR). Hand Gestures Recognition in Images system uses C-support vector classification (C-SVC).

4.2 SVM in Hand Gestures Recognition System

Two routines from the LIBSVM library have been used for implementing the presented system:

- svm_train(problem, parameters)
- svm_predict(labels_set, features_set, model)

SVM parameters, C and gamma, are selected using cross-validation technique. (-8, 15) interval is considered for C parameter and (-13, 9) interval is considered for gamma parameter. All four kernels are also considered: linear, polynomial, radial basis function(rbf), sigmoid. The parameter, kernel and model selection algorithm are presented below.

For every set (C, gamma, kernel):

- a model is trained and tested;
- the parameters and corresponding accuracy are stored in a results array;
- from the final results array the algorithm selects the (C, gamma, kernel) set which has the best accuracy;
- the model for the selected set is saved and it is used in the final system.

4.3 Final LIBSVM Model File

After applying the algorithm presented in Section 4.2, the system has chosen the model which provides the best accuracy and the final LIBSVM model is generated. The most significant parameters that give the best system accuracy are:

- SVM type: C-SVC (C-Support Vector Classification)
- Kernel Type: Polynomial

5. APPLICATION STRUCTURE

The Hand Gestures Recognition System can simplify the way humans interact with computers. It has been developed as a static system which recognize gestures from the input images with a reasonable accuracy. Its implementation has culminated with the development of the graphical interface using Tkinter package from Python.

The system proposed here is capable of two big tasks:

- Training
- Testing

The Training part allows the selection of the dataset by taking the path of the folder containing the images, the creation of the input file which will feed the SVM classifier and, the most important stage, it allows the creation of the SVM model by following the algorithms presented in 4. GESTURES CLASSIFICATION.

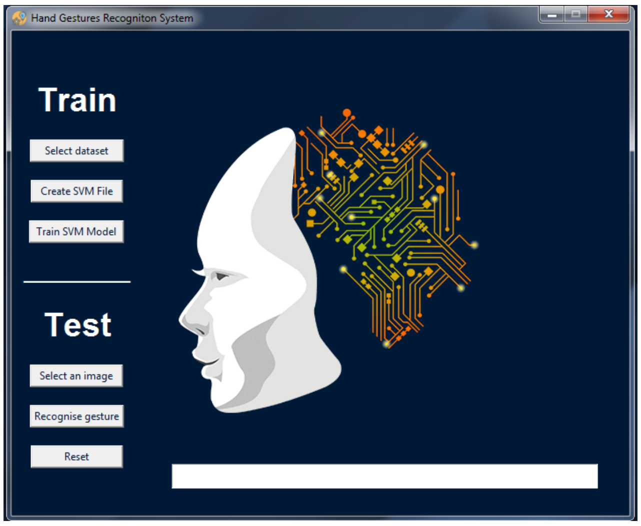


Figure 10 Application GUI

On the other side, the Testing part of the application brings the possibilities of selection the image of which the user wants to recognize a gesture, of recognizing the gesture depicted by the loaded image. The user has also the possibility of resetting the system in different situations, for instance when he has uploaded the wrong image into the system.

A flowchart of the system is outlined in Figure 11. As clearly presented in the flowchart, the Testing part is not necessary dependent on the Training part. Under the condition that the training part has been run during a previous execution, the testing part can be used during the current session. However, each stage of the two main tasks are dependent one on the previous.

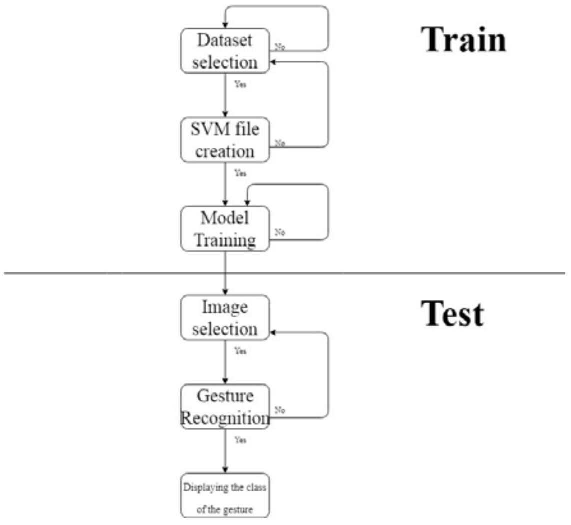


Figure 11 Flowchart of Hand Gesture Recognition System

6. EXPERIMENTAL RESULTS

The proposed method was tested with different value ranges for C and gamma SVM parameters or by applying other image preprocessing techniques, such as median filtering, until the best accuracy was found. The previous sections has shown what preprocessing techniques have been selected and Table 1 outlines the optimal chosen parameters.

Table 1 Classification rate using the final selected model

Final Model			
SVM Type	Kernel Function	Parameters of Kernel Function	Recognition Rate
C-SVC	Polynomial	degree = 3 gamma = 2^8 C = 0	52.381%

The final testing has used 5 completely new gesture postures for each of the 10 classes, resulting 50 completely new images. Table 2 depicts the results:

Table 2 Results of testing with the 50 new images set

New Images Test Result	
Class Number	Correctly classified Samples
1	19, 20, 21, 22
2	19, 20, 21, 23
3	21, 22
4	21, 22, 23
5	none
6	19, 21, 22
7	22
8	19, 20, 21, 22, 23
9	19, 22
10	19, 21, 22

CONCLUSION

A very efficient hand gesture recognition in images system requires higher dataset robustness in order to achieve a great accuracy and efficiency. In this paper, it is proposed a method, with reasonable accuracy, for classifying static hand gestures images using Otsu image binary segmentation, Hu Moments extraction, and Support Vector Machine. Even if there exist many classification algorithms, the SVM one is considered, in the speciality literature, a good choice for the hand gesture classification problem. The proposed system works well on static images with plane background. The experimental results reveal that it is able to clearly distinguish between 10 classes of gestures, depending on the posture.

REFERENCES

Arpita, R. S., Sanyal, G., Majumder, S., (2013). Hand Gesture Recognition Systems: A Survey, *International Journal of Computer Applications (0975 – 8887) Volume 71– No.15, May 2013*

Edward, J., (1991). A Users Guide to Principal Components, *Wiley Series in Probability and Mathematical Statistics*, A Wiley-Interscience Publication, 1st edition, 1991

Gary, B., Adrian, K., (2008). Learning OpenCV, O'REILLY, ISBN: 978-0-596-51613-0

Ilan, S., Tomer, M. L., Dotan, Di C., (2010). Hand Gesture Recognition in Images and Video, *Department of Electrical Engineering, Technion-Israel Institute of Technology*, CCIT Report #763

Matthew, A., Turk, and Alex, P., (1991). Face recognition using eigenfaces, *IEEE Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, Lahaina, Maui, Hawaii

Ming, K. H., (1962). Visual pattern recognition by moment invariants, *IRE Trans. Information Theory*, vol.8, no.2, pp.179-187

Sanjay, M., (2011). A Study on Hand Gesture Recognition Technique, *Master Thesis*, National Institute Of Technology, Rourkela, India

Zou, Z., (2011). Computer Human Interaction using hand gestures, *Master of Engineering Thesis*, *University of Wollongong, School of Electrical, Computer and Telecommunication Engineering*

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 15 (42), No. 1, 2018

Vehicle Rental Software System

Mihaela Ilie*, Ionuț Murarețu*, Sorin Ilie*

**Dept. of Computers and Information Technology, University of Craiova*
13 A.I. Cuza Street, Craiova, 200585, Romania
(e-mail:mpirvu@software.ucv.ro , imuraretu@software.ucv.ro, sorin.ilie@software.ucv.ro)

Abstract: This paper explores the possibility of a car renting system for private car owners. Current systems require the renter to bring back the car themselves or employ the use of still-not-perfected self driving vehicles. We propose that a car can *hitchhike* by taxing the renters that abandon it and paying drivers to take it closer to home. For this purpose we developed a mathematical model based on existing technology and approaches. We also propose a implementation architecture. We model continuous sanctions for renters that do not take steps to return the automobile to the owner at the desired time and place. We also present a game theory analysis of the utilities of both drivers and owners. The result is that longer trips are more advantageous for all involved. We also introduce a formula that can calculate the moment when the driver should start heading back in order for the system to be able to find a replacement driver, if needed. We found the system to be robust and safe to use, i.e. there is no hidden utility for the driver or the owner to act in bad faith.

Keywords: mathematical model, virtual car fleet, software architecture , game-theory

1. INTRODUCTION

The importance of private transportation for persons and merchandise to a working economy is quite self-evident. Regardless, we will enumerate some of the positive effects of readily available transport (Savelsbergh et. al. 1995): higher goods and currency exchange, better tax collection since most transactions and service payments are taxed, this in turn generates more public projects in infrastructure, housing, energy and agriculture.

Automobiles are built to be driven, inactivity causes non-lubricated metal to corrode in mechanical and in electrical systems alike. Excessive heat and cold unregulated by the car's operational systems cause plastics to crack and crumble. All these processes are greatly slowed down during normal operation. We can now state that parking a car for most of its life cycle is actually detrimental to it. We can also make the argument that parking lots occupy a lot of urban real estate that could be used productively instead of housing decaying expensive transportation machines. According to this study (Kuhnimhof et. al. 2012) conducted in Germany, in 2012, a car was used for around 2 hours each day and car ownership is on the decline. Therefore, young people are looking for flexible modes of transport.

Paper (Hansen et. al. 1997) presents statistics on roads in California concluding that increasing lane number in highways is not a solution to bad traffic. The authors found clear indication that bigger highways attract more traffic. Therefore we conclude that the answer to bad traffic is not in infrastructure but the way it is used. This paper is operating under the consensus of the scientific community that global climate change is real and has bad repercussions all over the world (Parmesan et. al. 2003). From an environmental standpoint, more cars no matter how green or clean still cause more pollution than less cars of the same type. It is self-evident that during operation, multiple cars produce more pollution than a single car but also require more fuel and maintenance cost. Even a car with zero emissions and consumables during operation, would still have emissions during its manufacturing process and prime matter mining. Therefore it is in our best interest as a species to reduce the number of automobiles on the road to a bare minimum and use the ones we have to the end of their life cycle. Public transport is created when available but you can only get it in designated point to point stations and users have to sync to their schedule. Therefore we have an ethical and economical incentive to use as few cars as possible.

1.1 State of the Art

We start by taking into consideration the existing practical application of car sharing services and their inadequacies.

Uber (UBER 2019) is a car ride sharing system. It is meant to be used is for people to pick-up hitchhikers in an urban area in order to minimize their own transportation costs. The payment is done through the Uber on-line system alongside a reputation system for drivers, who also own their cars. However, finding multiple persons with similar partial itineraries and aligning them to maximize efficiency, is not a simple problem (Savelsbergh et. al. 1995). Drivers have found that Uber is a possible income source, therefore it has become much like a freelance taxi service. However, there are cost estimations (Chen et. al. 2016) that show uber drivers

might be working at a loss when factoring in car maintenance costs and the fluctuating ride cost. The main advantage over a classical driving job is the extremely flexible working hours. The legality of being an Uber driver is not easily argued due to the fact that a public transport driver is actually required by law to have several permits, studies and driving experience. This is quite a long way away from the one click join that Uber offers. However, this problem could be easily avoided if the car would not have the car owner inside as the service provider. That is one of the reasons Uber is investing in autonomous self-driving automobiles. Even so, flawless driving robots are a long way away at the moment, and they will probably have prohibitive prices to start with, according to (Varaiya et. al. 1994). Tesla (TESLA 2019) uses their research in autonomous self-driving automobiles in order to allow their products to potentially provide revenue for their owners. Basically, their idea is that when at work or sleeping you could have your car join a fleet of cars available for rent. The autonomous car would pick up passengers and drop them off. Owners would get paid in an on-line account and return to the parking lot with time to spare. This scenario hinges on the viability of level 5 autonomous self-driving cars, i.e. zero fatality tolerance transport, the only acceptable version or robot drivers according to (Tussyadiah et. al. 2017). However, with powerful tech companies such as Google (GOGL 2019), BMW (BMW 2019), Uber (UBER 2019) working on this, there are chances we will see level 5 over the next 10 years according to this US study (Litman. al. 2017). In (Sendouda. al. 1994) a proposed system for automated car renting is proposed. In this patent a complicated, potentially expensive system is presented. In this system users would buy a subscription of a car fleet and simply pick up and deposit cars back to a designated location. Resulting in costs for the duration of their use. Car2go is an updated version of the system presented in the patent (Sendouda. al. 1994), it makes use of current smart car and phone technology. They have a fleet of smart cars. You use their app to localize the nearest available car. There have been studies conducted on the environmental impact of car sharing systems (Firnkorn et. al. 2011) and in particular Car2go. However, this is just a smarter approach to car-renting , which is not really the purpose of this paper.
Getaround (GET 2019) invites private owners to join a system to rent their cars. They take care of driver screening, insurance, system integration with smart-cars and possibly install modules to open, start, stop and monitor cars. However, this would be a system meant for owners that are leaving town or have unused automobiles since there is no actual way of constraining the renter to return the car at a given time or place. The Getaround app has a double rating system where owners can rate renters and vice-versa. Intuitively this forces users to be as nice as needed in order to keep using the system.

2. PROPOSED SOLUTION

Our proposed solution is a car sharing system that satisfies the following conditions:

- owners can capitalize on their car’s downtime.
- renters must have incentive to return of the vehicle in a timely manner. The term driver and renter will be used interchangeably in the following sections.
- if the renter decides to abandon the car, the system must provide other renters with the financial incentive to return the automobile to the owner’s desired location and on time.

In essence the system must not deprive the owner of his vehicle but also not force the driver to bring back the car. To avoid an overly complicated model in the following sections we make the following simplifying assumptions:
all cars have the same technical specs.

- rating is given solely on aspect and cleanliness. It has no impact on the mathematical model but we recognize its usefulness in helping a renter make his choice.
- the drivers do not have any special requests for the vehicle they will rent
- we concentrate on getting the driver in a working automobile and then providing enough incentive for returning the vehicle to the owner. This should be the best course of action for the driver with no reliance on good intentions.
- any liabilities will be solved via insurance that is not within the scope of this work.

For future reference, we consider a car as *being part of the virtual fleet* when it’s owner presents it as being available for rent, i.e. potential drivers will be able to find it on their mobile devices .

When it is currently used by a renter it is still part of the fleet although it is currently not available for rent.The car can become available for rent again when the driver returns or abandons said car.

2.1 Mathematical model of the system

Each car owner presents the car drop-off coordinates θ, the current coordinates l and price P per time unit. The timestamp at witch the automobile last re-joined the virtual fleet T , and $R$the current rating of the car. Additionally we also need to know at what time s the car should be back at coordinates Θ, also known as the drop-off location. In essence, we can define a car as a vector for these attributes and the fleet F as a set of cars.

$$F = \{C_j | C_j = (s_j, l, P_j, T_j, R_j, \Theta_j)\} \qquad (1)$$

where F is the set of available cars in the virtual fleet and j is an iteration index.

The set of users U of the fleet is composed of potential drivers D defined as vectors with the following attributes: the starting coordinates l the final destination coordinates d , his rating R and the maximum time he requires a vehicle.

$$U = \{D_k | D_k = (I_k, d_k, R_k, \tau)\} \qquad (2)$$

Where U is the set drivers currently looking for cars and k is an iteration index.

Next we define a function that calculates the fitness of a car for a driver's needs.

$$f(C, D) = \frac{sig(\tau - t_i - s) + 1}{|l - d|(P - \frac{P_r}{\tau})} \tag{3}$$

where :

sig is the signum function, it equals 0 when the parameter is 0, 1 if the parameter is positive, and -1 if the parameter is negative.

t_i is the current timestamp when the fitness functions are calculated

P_r is the advertised return reward, defined in this paper in equation (9)

Therefore, a driver D can be matched to a list of available cars in order of the fitness to his requirements, however, the final choice is completely his, such as in an auction (Bădică et. al. 2015). For example the driver might accept a higher price if the drop-off location is on his route. We define the history of all rentals as follows:

$$H_r = \{(D_i, C_j, t) | j \in \overline{1..|F|}, i \in \overline{1..|U|})\} \tag{4}$$

where:

D is the driver that rented the car C at time t

We define the history of all car abandonment as follows:

$$H_a = \{(C_j, t, L_a)\} \tag{5}$$

where:

C is the abandoned car

t is the timestamp at which it was abandoned

L_a is the location of abandonment

When a driver rents a car he is taxed the rate P per time unit. However, the system has ensure that the automobile can be returned at the drop-off location at the required time. For this purpose we define the following notions: estimated path duration and estimated rent time. Firstly, the estimated time to drive from point A to point B at the current time is noted $dist(A, B)$. This function should also take into account road infrastructure, road quality, instantaneous traffic and meteorological conditions resulting in a probabilistic path duration (Rao et. al. 2014) similar to Google directions (GOOG 2019). Secondly, the estimated rent time can be evaluated as follows:

$$e(C) = \frac{\sum i{=}1, j \in H`_r |F| (T_i - t)}{n} \tag{6}$$

where:

H_r is the set of rented cars in the most recent temporal segment of a given size ϵ .

T_i is the time at which the cars have entered the fleet

And finally, we define the minimum driver timeout notification time $n = e + r$ where r is $dist(l, \Theta)$.

We can now define the instantaneous incentives to return the car to the correct drop-off location .

$$P` = \frac{[sig(t_i - s - n) + 1](t_i - s - n)}{2k} P \frac{dist(l, \Theta)}{dist(l_n, \Theta)} \tag{7}$$

where:

k is a parameter representing time increment at which the price increases by another p per time unit

t_i is the instantaneous timestamp

l_n is the location reached at time $s - n$

The number k must be adjusted depending on the local market.

At any time the driver can willingly pay an abandonment tax of:

$$P_a = (e + dist(l, \Theta))P \tag{8}$$

This tax permits the car to advertise an award of $P_r$to potential drivers that are willing to drive it to , this is awarded only when the car reaches . We define P_r as follows:

$$P_r = \frac{(sig(t_i - s) + 1}{2} P_a \tag{9}$$

2.2 System Design

We will attribute identity verification to the the system owner. Therefore, the following are not within the scope of this paper but can be mentioned here as a baseline requirement of the e-commerce application required to facilitate the described :

- an owner will have to offer proof of owning the car, up-to date car insurance, service history, send geo-located images that can be checked against the car's location, a prepaid amount or a credit card for fuel and insurance if needed.
- a driver can offer his driver license, insurance history if available, and a credit card or a deposit.
- the car must be fitted with a remote access, remote start, dash cam, driver cam, geo-location. This also has an added bonus of security when the system is on-line in the car. A possible car thief can be easily identified and tracked by the police in order to be caught in the act.
- accountability can be ensured by uploading images of the driver, keeping track of his identity and vehicle location at all times.

We can now define a finite state automaton (Wagner et. al. 2006) for the car (abr. FSA). A car can have four separate states: i) awaiting driver (initial state), ii) rented, iii) abandoned, iv) out of fleet (final state). The possible state changes are done through the following external actions:

start→i from no state to the initial state of the FSA, the owner must add the car to the virtual fleet *i*→*ii* driver rents car

ii→*iii* driver cannot return car to drop-off and abandons car somewhere else

iii→*i* automatic transition known as an eps transition

ii→*iv* driver parks car at drop-off point

i,ii→*iv* the owner decides to remove car from fleet

In figure 1 we present a schematic of the finite state machine of the car. The circles represent the states of the car, the arrows represent transitions, the text on the arrows represent the stimulus that causes the transition, the initial transition in state (i) is marked as a lightning symbol. The dotted transition is the manual override of the Owner that can remove his car from the fleet whenever it is not actively used by a driver. The double circle represents the final state of the machine.

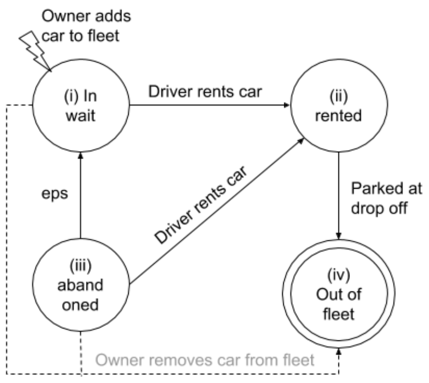


Figure 1. The automobile FSA schematic.

A driver and the owner have to agree on the transaction before renting. For this reason a protocol must be put in place. We designed a communication protocol(see figure 2 using universal modeling language(UML) sequence diagrams, in particular the agent modeling language notation. In this figure, the owner and driver are modeled as actors, the vertical lines represent the actors' time axes; arrows represent messages; the diamond shape is a decision; the rectangles on the actors' time line represent processing or user input (Becker et. al. 2000); messages are composed of the message title and the parameters, in brackets under the message arrow. At the beginning driver searches for available cars parked near himself order by the fitness defined in equation (3). The driver then chooses car and sends owner his verified profile, complete with available credit, and history of accidents . At this point, the owner can decide to deny the request or accept and grant the driver's mobile device control over the car via access token.

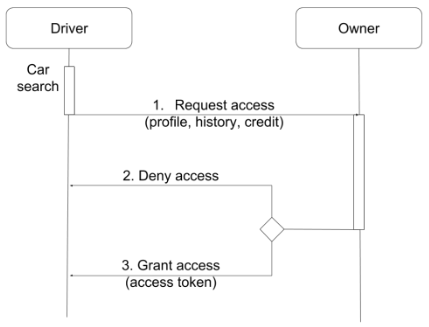

Driver
Owner
Car search
1. Request access (profile, history, credit)
2. Deny access
3. Grant access (access token)

Figure 2. The sequence diagram of the communication protocol between the driver and the car owner.

At minimum, the system must offer the driver notifications for reaching $s-n$, i.e. when additional taxes apply unless the driver starts to return the car, and when the driver's credit is running out. Also, the car owner should be able to locate his car at all times.

The system's software architecture (see figure 3) is comprised of at least four types of entities: i) many owner mobile devices, ii) many driver/renter mobile devices, iii) a system server, and iv) the cars themselves. This makes the system well suited for an implementation as a website with a web API. This is a common approach for smart home applications, for example Nest (NEST 2019), Fibaro (FIB 2019). Both smart homes and smart car fleets are a type of Internet of things (Xia et. al. 2012) application therefore it makes sense they should be implemented on similar architectures. Another reason for this suggested architecture is that Getaround (GET 2019), Tesla (TESLA 2019) and Uber (UBER 2019) already work well using similar architectures. The server would contain most of the programming, the mobiles are interfaces to the profiles in the server, and the car becomes a slave to the system.

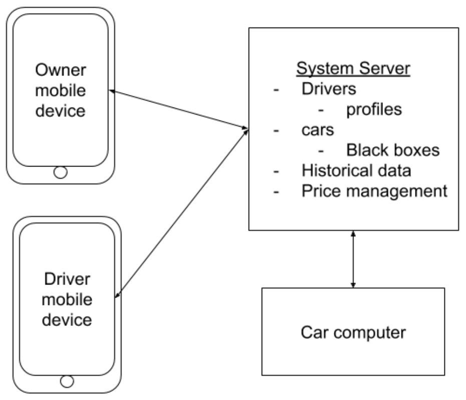


Figure 3. The block diagram of the proposed architecture for the system.

2.3 Game-theory analysis

We will now define the utility functions for each stakeholder for the game-theory analysis (Barron et. al. 2013) of the system from an economics point of view. A utility function is a quantitative mathematical model of the stakeholders' interest and its dependencies. In game theory each stakeholder would be called a \emph{player}.

For the set of car owners the utility is directly proportional to the time the car is rented. In equation (10) the coefficient of P is time. The integral in this equation is used to calculate the sum of increments of incentives the driver gets to return the car. The driver pays less per time unit when he is returning the automobile. These incentives translate into lower gains for the owner over the path of return.

$$u_o = [s - n - T + \int_{l_n}^{\Theta} \frac{dist(l_n,\Theta) - dist(l,\Theta)}{dist(l_n,\Theta)} dl]P - m \quad (10)$$

where

m is the cost of maintaining the car for that time, including wear on the car parts.

We will define the profit time t_p as the coefficient of P as follows:

$$t_p = s - n - T + \int_{l_n}^{\Theta} \frac{dist(l_n,\Theta) - dist(l,\Theta)}{dist(l_n,\Theta)} dl \quad (11)$$

The condition for the owner to profit is $u_o > 0$. Therefore the costs of maintenance m must be lower than the incoming finances $t_p P$. Maximizing u_o translates to maximizing t_p which can be done in two ways: i) longer rent times by setting the drop-off time s as far into the future as possible, and ii) reducing the notification time n. However, n is not dependent on owner actions but the driver's, i.e. the further away the driver wanders off from the drop-off location the longer notification time he will get. Therefore, the owner can only try to rent to less drivers for longer periods of time , again, by increasing s as much as possible.

For the driver the utility u_d depends on the cost of the rental which depends on his ability to return the car at the drop-off before time s expires. Therefore, the driver must choose a car with the availability larger than his need for an automobile $s \le t_d$. However, to fully benefit from the incentive to bring the automobile to the drop-off location the time must be almost identical to the car's temporal availability $t_d \simeq s$. The returning incentive will only become active after the n notification is triggered. This is also beneficial for the owner since he would lose billing time if the car changes drivers.

$$u_p = P_d - [s - n - T + \int_{l_n}^{\Theta} \frac{dist(l_n,\Theta) - dist(l,\Theta)}{dist(l_n,\Theta)} dl]P =$$

$$= P_d - t_p P$$

where:

P_d is the actual profit the driver gets externally by using a rental car measured in the same units as P, i.e. some form of currency

In order for the driver to prefer the incentive instead of paying the full rate P per time unit, the following will have to be true:

$$n > \int_{l_n}^{\Theta} \frac{dist(l_n,\Theta) - dist(l,\Theta)}{dist(l_n,\Theta)} dl$$

$$e + dist(l_0,\Theta) > \int_{l_n}^{\Theta} \frac{dist(l_n,\Theta) - dist(l,\Theta)}{dist(l_n,\Theta)} dl$$

however, for

$$l \in (l_0, \Theta]$$

and

$$l_0 \ne \Theta => \frac{dist(l_n,\Theta) - dist(l,\Theta)}{dist(l_n,\Theta)} < 1$$

$$e + dist(l_0,\Theta) > \int_{l_n}^{\Theta} \frac{dist(l_n,\Theta) - dist(l,\Theta)}{dist(l_n,\Theta)} dl \le 2$$

$$e + dist(l_0,\Theta) \le 2$$

is true if the rental cars are not in extremely high demand at the time and the distance to the drop-off Θ is not insignificant. Therefore, we can conclude that the driver is interested in renting the car for as long as possible in order to take advantage of the return incentive, but also incorporate the drop-off location Θ in his itinerary (see (Savelsbergh et. al. 1995) for rationale for path optimization). Once again, this is generally in the best interest of the owner as well.

At this point we can state that the stakeholders are playing a finite "zero-sum game" in which one player cannot win something without taking something from another player. This type of game is well known and extensively analyzed in game theory (Barron et. al. 2013). An equilibrium is a state in which the game is stable because there is no strategy the players can take to improve their winnings. This game always has an equilibrium and it is the scenario when all stakeholders have at least the minimum positive utility . This is true only if minimum maintenance cost is met by the base cost $P > m$ and the driver can profit from external sources by using the car $P_d > t_p P$. These inequalities are strict in order to account for larger than 0 net profit.

To estimate the maintenance cost m, the safest course of action is to divide the yearly cost to the duration of daily use by the owner, himself, estimated in time units. This will be the conservative estimation of use in the case the car is never rented, for some reason. In this estimation one time costs such as: road-taxes, service check ups, service consumables and insurance will be partially attributed to every time unit of use by the owner, we will note this part of the maintenance cost per time unit i. That means the total estimated yearly maintenance of the car per transaction will be $t_p i$. If we use the German study (Kuhnimhof et. al. 2012) as a guide, each day has potential of 22 rental hours, i.e. eleven times more than the driver uses the car. Therefore the owner should be paid back up to $11i$ by entering his car in the system. The second type of costs of owning a car, fuel, can be estimated more closely by time unit of use, we will note this as *g*. To sum up, the maintenance cost can be expressed as follows:

$$m = t_p i + g \quad (12)$$

One interesting thing about this system is that an owner might decide to maximize his utility by using his own car as little as possible, because the time he recalls his car he has to offer return incentives and the wait the medium renting delay e from equation (6). Therefore, rather that setting a close time to return to drop-off Θ, the owner might choose to rent another car in order to get home or maybe look for a nearby abandoned car. The cost of renting someone else's car can be offset by the income from renting his own car resulting in zero costs but also zero losses. This course of action assumes that the owner that decides to rent instead or recalling his own car can avoid abandonment taxes by finding a car that needs to be dropped off at his own destination.

2. DISCUSSION

The presented mathematical model is designed to reward good behavior but also avoid possible loopholes.

Looking into the impact of such system put into action, there are several possible consequences:

- more parking space. If the existing cars can be shared, they will spend a lot more time on the road.
- faster adoption of new transportation technologies. intense use of automobiles \item can generate enough revenue to permit owners to change them more often. One reason to do this could be cost efficiency.
- each car fitted with this system would be a deterrent to grand theft auto, due to the constant monitoring.

This system relies that there are enough people that want to rent cars and enough cars to create a never ending list or requests and offers as is the case for the taxi companies. According to the European Automobile Manufacturers' Association (ACEA 2019) reports the total number of cars in the European Union (EU) in 2017 has reached 298 million for 741 million inhabitants, i.e. approximately a car for every 2 people. Therefore, there is no shortage of cars and, implicitly, people with a driver's license, supporting the feasibility of the approach.

2. CONCLUSION

Future work involves: expanding this system to include transport planning in order to bring the car to the drop-off point using multiple drivers. We suspect that this behavior might emerge from the price reduction however experimentation is required to confirm this. Integration of automated negotiation (Luncean et. al. 2016) to guarantee the best possible scenario for all stakeholders.

ACKNOWLEDGMENT

Mihaela Ilie and Ionuț Murarețu were supported by QFORIT Programme, University of Craiova, 2018.

REFERENCES

Barron, Emmanual N. Game theory: an introduction. Vol. 2. John Wiley & Sons, 2013.

Badica, Costin, Maria Ganzha, Maciej Gawinecki, Pawel Kobzdej, and Marcin Paprzycki. "Towards trust management in an agentbased e-commerce system-initial considerations." In Proceedings of the MISSI 2006 Conference, pp. 225-236. 2006.

Chen, M. Keith, and Michael Sheldon. "Dynamic Pricing in a Labor Market: Surge Pricing and Flexible Work on the Uber Platform." EC. 2016.

Sendouda, Mitsuru. "Car rental system." U.S. Patent Application No. 09/878,052, 1994.

Litman, Todd. Autonomous vehicle implementation predictions. Victoria Transport Policy Institute, 2017.

Firnkorn, Jörg, and Martin Müller. "What will be the environmental effects of new free-floating car-sharing systems? The case of car2go in Ulm." Ecological Economics 70.8 (2011): 1519-1528.

Parmesan, Camille, and Gary Yohe. "A globally coherent fingerprint of climate change impacts across natural systems." Nature 421, no. 6918 (2003): 37.

Hansen, Mark, and Yuanlin Huang. "Road supply and traffic in California urban areas." Transportation Research Part A: Policy and Practice 31.3 (1997): 205-218.

Kuhnimhof, Tobias, et al. "Travel trends among young adults in Germany: increasing multimodality and declining car use for men." Journal of Transport Geography 24 (2012): 443-450.

Sandoval-Mercado, David Jonathán, and Roberto Rosas-Romero. "Remote car starter." Electronics, Communications and Computers, 2005. CONIELECOMP 2005. Proceedings. 15th International Conference on. IEEE, 2005.

Varaiya, Pravin. "Smart cars on smart roads: problems of control." IEEE Transactions on automatic control 38.2 (1993): 195-207.

Bădică, Costin, Sorin Ilie, Alex Muscar, Amelia Bădică, Liviu Sandu, Raluca Sbora, Maria Ganzha, and Marcin Paprzycki. "Distributed agent-based online auction system." Computing and Informatics33, no. 3 (2015): 518-552.

Becker, Jörg, Michael Rosemann, and Christoph Von Uthmann. "Guidelines of business process modeling." In Business Process Management, pp. 30-49. Springer Berlin Heidelberg, 2000.

Savelsbergh, M.W. and Sol, M., 1995. The general pickup and delivery problem. Transportation science, 29(1), pp.17-29.

Xia, F., Yang, L.T., Wang, L. and Vinel, A., 2012. Internet of things. International Journal of Communication Systems, 25(9), pp.1101-1102.

Wagner, F., Schmuki, R., Wagner, T. and Wolstenholme, P., 2006. Modeling software with finite state machines: a practical approach. Auerbach Publications.

The Automobile Manufacturers' Association (ACEA) website https://www.acea.be . Last access Jan 2019.

Uber website https://www.uber.com/ . Last access Jan 2019.

Uber self driving car reserch and development project https://www.uber.com/info/atg/car/ . Last access Jan 2019.

A description of the car sharing system proposed by Tesla https://www.tesla.com/blog/master-plan-part-deux . Last access Jan 2019.

Waymo, the Google self-driving car research and development project https://www.google.com/selfdrivingcar/ . Last access Jan 2019.

BMW self-driving automobile research and development program https://www.bmw.com/en/automotive-life/autonomous-driving.html . Last access Jan 2019.

Getaround car sharing website https://www.getaround.com/ . Last access Jan 2019.

The Nest smart home devices website https://nest.com/ . Last access Jan 2019.

The Fibaro smart home devices website http://www.fibaro.store.ro . Last access Jan 2019.

The Google directions, path finding using MAPS, API website https://developers.google.com/optimization/routing/google_direction . Last access Jan 2019.

Tussyadiah, I.P., Zach, F.J. and Wang, J., 2017. Attitudes Toward Autonomous on Demand Mobility System: The Case of Self-Driving Taxi. In Information and Communication Technologies in Tourism 2017 (pp. 755-766). Springer, Cham.

Rao, R.S., Soni, S.K., Singh, N. and Kaiwartya, O., 2014. A probabilistic analysis of path duration using routing protocol in VANETs. International Journal of Vehicular Technology, 2014.

Luncean, L., Mocanu, A. and Becheru, A.P., 2016, September. Automated Negotiation Framework for the Transport Logistics Service. In Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2016 18th International Symposium on (pp. 387-394). IEEE.

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 15 (42), No. 1, 2018

REAL CAR GAME APPLICATION - FINDING THE SIGNAL

Grigore Mihai TIMIS*. Ionut MOVILEANU.*

**Technical University "Gh.Asachi"Iasi,*
Faculty of Automatic Control and Computer Engineering (e-mail: mtimis@ tuiasi.ro).

Abstract: In this paper, the authors presents an implementation of real car game application - finding the signal. A such of intelligent prototype allows the user to control the car on a given lane, detecting as many magnets while traveling above them during the pre-established time in the microcontroller program. As the time expires, the score which represents the number of magnets detected will be sent to the application through serial communication.

Keywords: Smart car, Real car game, Robotic car, Autonomous car, Electric car, Lane Detection, Arduino Mega 2560, Visual Studio.

1. INTRODUCTION

The self-driving, autonomous vehicle has been getting lots of attention, due to significant development efforts and dramatic progress made by companies such as Google. While general use of autonomous vehicles for widespread use on public roads is likely years away, these vehicles are already being employed in "constrained" applications such as open-pit mines and farming. Among the many technologies which make autonomous vehicles possible is a combination of sensors and actuators, sophisticated algorithms, and powerful processors to execute software. The sensors and actuators in an autonomous vehicle fall into three broad categories: 1) navigation and guidance (where you are, where you want to be, how to get there); 2) driving and safety (directing the vehicle, making sure it vehicle acts properly under all circumstances, and follows the rules of the road); and 3) performance (managing the car's basic internal systems). The autonomous car must be able to see and interpret what's in front when going forward (and behind when in reverse, of course). It is also necessary to see what is on either side; in other words, it needs a 360grades view. An array of video cameras is the obvious choice, with a camera to determine where the lane is and sense objects or markers on the road. But using cameras alone presents problems. First, there are mechanical issues of setting up multiple cameras correctly and keeping them clean; second, heavy graphic processing is needed to make sense of images; third, there is a need for depth perception as well as basic imaging; and finally, conditions of lighting, shadows, and other factors make it very challenging to accurately decide what the camera is seeing.

In the present paper, the authors presents the implementation of a Real Car Game with lane detection feature named "Game Mode". It is about a 1/14 scale ”LC Racing” car controlled by an Arduino microcontroller through a Window application. Therefore, controlling was taken care of by an Arduino Mega 2560 controller, a BLDC motorisation engine and an alignment Servo type one, an ESC for engines control, Parallax sensors to determine the distance and two Hall sensors, one to measure the revolution and one to detect the magnets.

In this paper, the autonomous mode is configured and calibrated to function ideally only on a fixed route, as the calculi to determine the alignment are made according to the distance between the two walls. During the travel of the car on the route, data regarding the time passed since the start, the speed, the distance covered and the level of battery charge is being sent in real time. Communication is made through a HC-05 Bluetooth added to the car prototype.

In the manual mode ”Game Mode”, the control will be performed through the radio frequency remote control. The Bluetooth communication is active at the same time.

After opening the application on the computer, the user will have to select the ”Game Mode”. If the connection was successfully done, the user will have to introduce their name and password so they can be checked and so that the application knows whom to assign the score. Here is how the game will be played: the user controls the car on the lane given, detecting as many magnets while travelling above them during the pre-established time in the microcontroller programme; as the time expires, the score (the number of magnets detected) will be sent through serial communication to the application.

The software part was created in the programming medium made available by Arduino, using its libraries and facilities when working with the microcontroller and in the Visual Studio 2012 programming medium, for the possibility of programming in language C# with the facilities offered by Windows Forms, when it comes to the interface with the user. We also used the facilities offered by Windows operating system when talking about mating the two Bluetooth modules, the one on the car and the one in the computer for stable, optimum communication.

While implementing the prototype we considered necessary making two different types of programming:

• the medium found on the Arduino Mega 2560 microcontroller together with the restrictions and the facilities offered;

• the programming medium in Visual Studio 2012 using language C# and Windows Forms facilities for the graphic interface of the application.

We have the programme implemented on the microcontroller on one side and the control application. Between the two, stable communication had to be established. To this aim, we used the Windows facilities to choose the communication port and to mate the two Bluetooth modules, the one on the car and the one on the laptop.

2. HARDWARE COMPONENTS

The sensor is a device which, exposed to physical phenomena (temperature, travel, force etc.) produces a proportional output signal (electric, mechanic, magnetic etc.).

The term actuator is used several times as a synonym for the word sensor. Nevertheless, the sensor is a device that responds to the variation of a physical phenomenon. On the other side, the actuator is a device that converts one form of energy into another. The sensors are actuators when they have one form of energy when entering and another form when outputting.

Example: a thermocouple seizes the temperature variation (thermic energy) that it turns into a variation of the thermo-electromotor force (electric energy), therefore we will be able to name it sensor or actuator.

The linear travel or position sensors and the rotation sensors are among the most used in the mechatronic systems. Generally, the position sensors produce an electric output signal proportional to the travel.

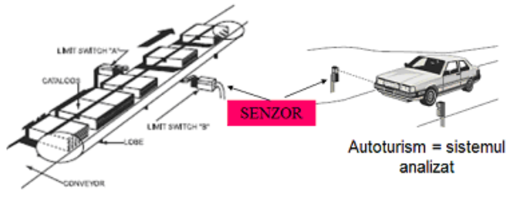


Fig. 1 – Travel Sensor

The sensor is a technical device that reacts qualitatively or quantitatively through proper measurable sizes, at certain physical or chemical features of the medium around it. As a component part of a device or detecting technical system, it can measure/register strain, humidity, magnetic field, accelerator, force, sound intensity, radiations and so forth. There are multiple classifications, one of which refers to the following types of sensors:

- Active: energy consumer such as the radar (measuring the distances through electromagnetic radiations emission).

- Passive: for instance photo-resistance that can measure the intensity of incident light.

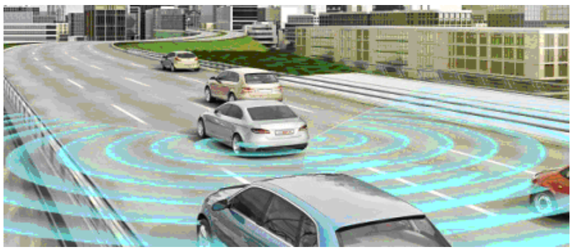


Fig. 2 – Active Sensors – Radar

Distance sensor- In order to measure the reference distance between the two walls of the of the route we used two ultrasonic sensors of the Parallax's PING))) HC-SR04 type.

Fig. 2 – The distance sensor HC-SR04

The decision to use this type of sensors was due to the fact that these cannot be influenced by certain parameters such as light, colour of the obstacle, temperature of the environment etc. that would have introduced measurement errors in the case of other sensors used in measuring distances.

As an example, let's take the infrared sensor illustrated below which, in spite of having a shorter response time and a bigger precision, it could not be used because the route's walls were matte black and the light type signal sent by the sensor was not reflected enough so it could be received.

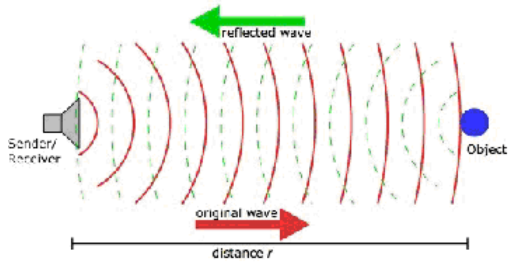


Fig. 3 – Ultrasonic waves

Arduino Mega 2560 Microcontroller: in order to connect all the components: ultrasound sensors, Hall sensor, brushless engine, servo-engine, there was used Arduino Mega environment, an open – source processing platform, based on flexible easy to use software and hardware. It has a small platform (10.16 cm / 5,8 cm – in its most used variant), built around a signal processor and it is capable to take data from the environment through a series of sensors and perform actions on the environment through the lights, the engines, the servo-engines and other types of mechanic devices. It is based on an ATmega 2560 microprocessor able to drive written code in a programming language very similar to the C++ language. It is like as a minicomputer, being able to collect information from the environment and to react to it. Arduino Mega has got 54 digital pins (entries and outputs), where 15 can be used as PWM exists (Pulse Width Modulation), 16 can be used as analogous entries, 4 are UART Asynchronous Receiver/Transmitter used as serial hardware ports), a 16 Hz crystal oscillator, a USB connection and a reset button. There is a very well developed devices ecosystem around Arduino. Regardless of the type of information we want to collect from the environment, regardless of the connections with other systems we need, it is highly likely that we find an Arduino device, capable to offer what we need. Thus, when taking information from the environment, we can have different examples of sensors: the ones that determine the level of alcohol in the air we breathe, fire sensor, GPL gas, carbon monoxide, accelerations of the moving devices, current consumed by various household devices, pushing force, degree of rotation, RFID cards, distances, illumination level, North direction, human presence, sound, temperature, human presence, sound, humidity, atmosphere or video pressure. Regarding the possibility to connect with other systems, there are Ethernet network plates for Arduino, capable to communicate information through the internet, devices capable to send data through radio connection, WI-FI network plates, GSM devices for Arduino (capable to send / receive texts (SMSs), to make voice phone calls or to send data through the 3G network) or Bluetooth connectors for Arduino connection with the mobile phone or with the laptop.

Arduino mega can be programmed with the help of the Arduino software, the Atmega 2560 microprocessor is programmed with a boot loader that allows uploading a code without using an external hardware programmer. It communicated using the original STK 500 protocol. Arduino Mega 2560 can be charged through the USB connection or with the help of an external source. The plate can function with an external source between 6 and 20 volts. Nonetheless, the producer's recommendation is to use the source ranged between 7 and 12 volts. Not following this recommendation, the plate may become unstable, the tension regulator may get overheated and may deteriorate the plate.

The RC car: In order to reach high performance, was used a LC Racing EMB-MT, 1/14 scale. It contains a metallic chassis that offers increased resistance in case of inappropriate control in case of collision with other objects.

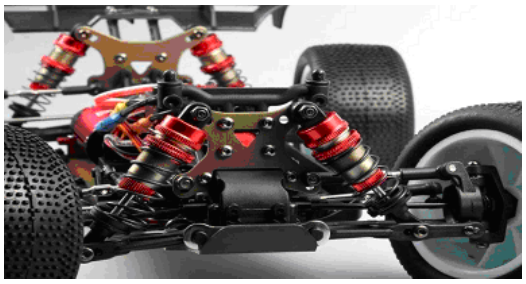


Fig. 4 – Chassis of the model motor-car

3. HARDWARE DESIGN OF THE APPLICATION

The application was divided in two modules: the autonomous module and the game module. The game module was created to train the user in spotting the magnets on the route.

In order to complete the necessary hardware part and for a user-friendly and safe use of the system, we decided to build, beside the Arduino Mega 2560 command microcontroller and the ESC of the engine, another two plates with printed-wiring. The two plates were created through the thermic imprinting process of the route and chemical corrosion.

Creating printed wiring: The first stage is designing the wiring, possibly starting from the electronic map of the assembly. The design is created in any program that helps create a PCB (Printed Circuit Board). (e.g. Eagle, OrCad etc.).

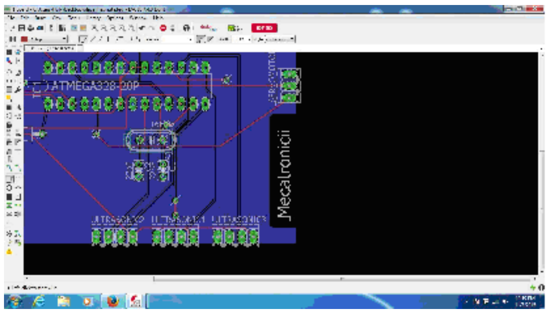


Fig. 4 – Eagle PCB

Controlling the BLDC engine: knowing the position of the knob is a quite an important factor in order to understand which transistors need to be opened. The following techniques was used: Measuring BEMF tension (Back Electromotive Force) and using Hall sensors. The ESC we use has a command circuit to control a BLDC engine using the measurement of BEMF tension as a technique to determine the position of the knob.

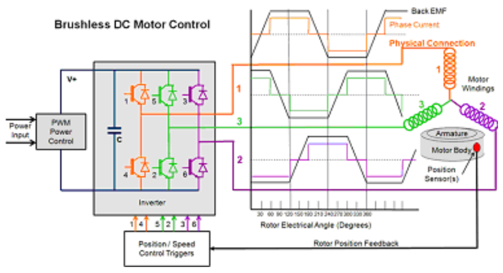


Fig. 5 – The control of the engine using BEMF

Command of the servo-engine in Arduino: the servo-engine has three wires that connect themselves to the command microcontroller. As it does not require big functioning currents and tension of only 5V, the Vcc wire connects itself to Arduino’s 5V input (5V pin). The table wire that is generally black connects itself to GND, while the third wire is the signal one and it connects itself to one of the output PWM pins of the microcontroller.

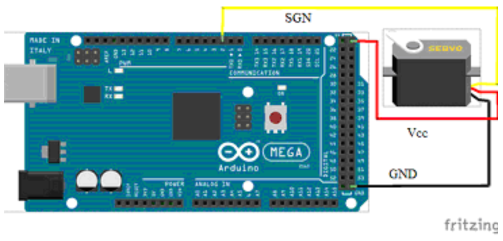


Fig. 6 – Servo – Arduino connection

Functioning mode of the HC-SR04 sensor: In order to start the measurement, the Trig pin of the sensor must receive a 3-5 V pulse for at least 10 us. Subsequently, the sensor will send 8 ultrasonic wave cycles of 40 kHz frequency, then it will set the signal pin to 5V (HIGH) and it will wait for the reflected waves. When the wave returns, the signal pin is stopped, the width of the HIGH pulse being proportional to the travelled distance.

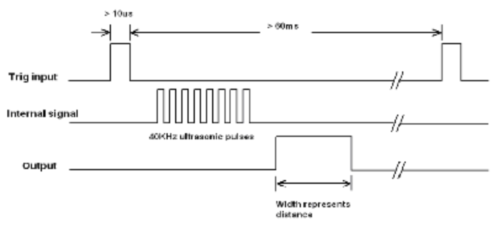
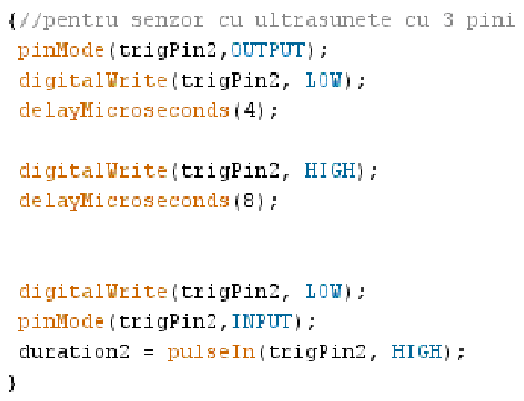


Fig. 7 – The principle of distance reading

In order to calculate the duration of the pulse, we used a sequence where the duration of the pulse is calculated with the help of the predefined function in Arduino: PulseIn(pin,HIGH).



```
{//pentru senzor cu ultrasunete cu 3 pini
 pinMode(trigPin2,OUTPUT);
 digitalWrite(trigPin2, LOW);
 delayMicroseconds(4);

 digitalWrite(trigPin2, HIGH);
 delayMicroseconds(8);


 digitalWrite(trigPin2, LOW);
 pinMode(trigPin2,INPUT);
 duration2 = pulseIn(trigPin2, HIGH);
}
```

Fig. 8 – Distance reading sequence in Arduino

Connecting the sensor to the microcontroller is made through three pins: V+,GND, SIG that can easily be connected to a breadboard or any other connection system.

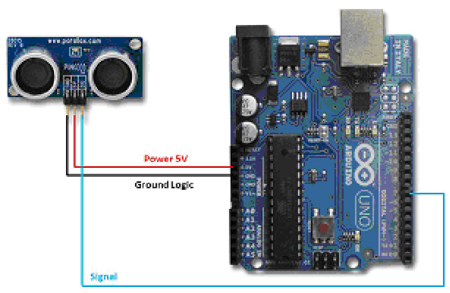


Fig. 9 – Connecting the HC-SR04 sensor

In the case of the model motor-car travelling on a route bordered by walls, in the case of curbs or other obstacles, the sensors placed at Lb distance will detect in time and will send a signal to the servo-engine in order to make the decision to swerve so that we reach efficient travel. Upon calculating and designing the bulbar, we asked ourselves whether this forward moving of the sensors does not lead to a collision when climbing up and down the ramps. This attack angle was calculated through a simple representation of the model motor-car in an as close to the ramp position as possible. Drawing a tangent line to the wheel of the auto vehicle and to the furthest point of the bulbar, the angle between this tangent and the plan of the travel route was calculated to 33.18°. Since what we needed was the bulbar not colliding with an 18° slope, the result we got matches our requests. The middle support is aimed at sustaining a sensor, so we chose to place a sensor in the middle as for the travel speed we do not have enough a couple to climb even an 18° ramp. This middle sensor has the role to detect the level differences, so that for minus level differences, the model motor-car accelerates for a few seconds so it can climb up, while for plus level differences, the model motor-car breaks so that it does not speed up when climbing down the ramps.

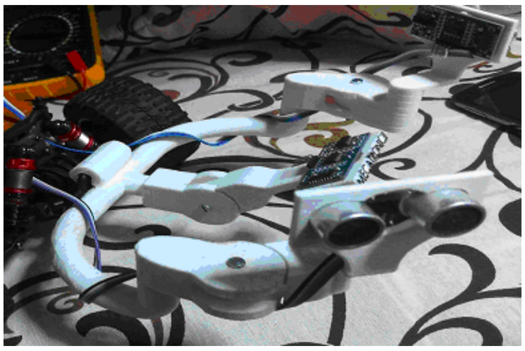


Fig. 10 – Final variant of the bulbar

4. SOFTWARE DESIGN OF THE APPLICATION

When creating the application, we used the development environment offered by Visual Studio 2012, by programming in C# language using .NET technology and Windows Forms for the creation of the interface with the user. Visual Studio includes a complete set of development instruments for generating ASP.NET applications, Web XML services, desktop and mobile applications. Visual Basic, Visual C++, Visual C# and Visual J# all use the same integrated development environment (IDE) that allows them to share the tools and facilitate the creation of solutions using multiple programming languages. These languages allow for the benefits of the .NET Framework features that offer access to key technologies that simplify the development of ASP and XML Web Services web applications with Visual Web Developer. In order to complete the application a number of four classes were created. „RealCarGame.cs" is the main class that help in calling the other classes by using the different functionalities in the interface.

Commuting between these modes is also done with the help of the physical switches set on one of the manually designed plates, but also through the software mode. To put it simply, this is a combination between hardware and software.

The functioning modes are: Autonomous, Full Radio Control (through the remote control of the car – manual control), Autonomous + Radio Control (semi-manual), Testing mode.

The Autonomous mode: from fig. 11, can be observed the route is some rather tight curves, a ramp and a slope, both on the interior and on the exterior route. That is why we considered positioning a distance sensor oriented downwards, beside the other two that read the lateral margins, which will constantly read the distance from the ground and when detecting distance shortening by meeting the ramp, the engine would accelerate in order to give the necessary speed to overtake the obstacle. In the software created in the Arduino development environment, by means of a function, we managed to also work out the speed decrease when climbing down the slope, as a dangerous curve was coming up on the route and the entry speed had to be considerably slowed down, so it stays constant and not waste a lot of time. Speed was slowed down when the car neared the walls, according to the values given by the sensors.

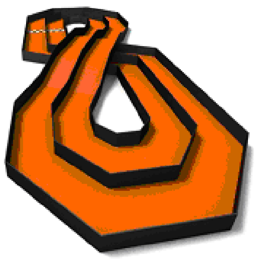


Fig. 11 – Racing Route

In order to put the autonomous mode into function, we must go through the following stages:

1. Performing the "BlueController" application;

2. Pressing the button "Connect" in order to make the connection between the application and the car through Bluetooth communication;

3. Choosing the autonomous mode by pressing the “Autonomous” button, the car will start moving;

4. Upon reaching the "Finish" line, we stop the car by pressing the button "Stop".

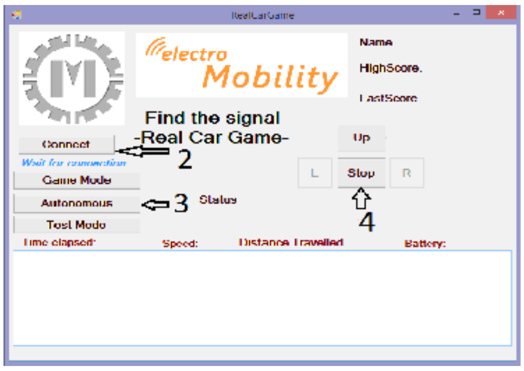


Fig. 12 – Autonomous mode stages

During the car travel, different data is being transmitted to the application through the Bluetooth module. As we can see in the application interface, we gave plenty of space to the data reception.

Here are the values: "Time elapsed" – The time passed since the ignition of the car, that is to say since we pushed the button "UP"; "Speed" – Speed during the travel; "Distance Travelled" – Covered distance; "Battery" – Percentage of battery charge.

Manual mode (Game Mode) : This mode is designed to actively engage the user in the process of car use. The user will have to control the car and detect the magnets in the pre-established space. The more magnets they will detect, the higher their score. One detected magnet score means one point. The magnets will be hidden under some boxes. There will also be some trap boxes which will be detectable with the help of the sensors Hall, set on the front side, by nearing the car to them at a convenient distance. The control is kept through the radio frequency remote control. Each user will be allotted a pre-established time period in which they will be able to gather points. When time expires, the car will stop and the score will be sent to be stocked by the application. This was designed as a game in order to bring more interaction and engagement. Upon connecting the application to the car and pressing the ”Game Mode” button in the activation process of the mode, we need to register or log in, depending on the case. Afterwards, the user will be able to handle the car as they consider best. In order to register or log in, a name and a password are needed. They will be stocked in an XML file. In case there is a user with the same name, the current user will not be able to register, as this will lead to conflicting data.

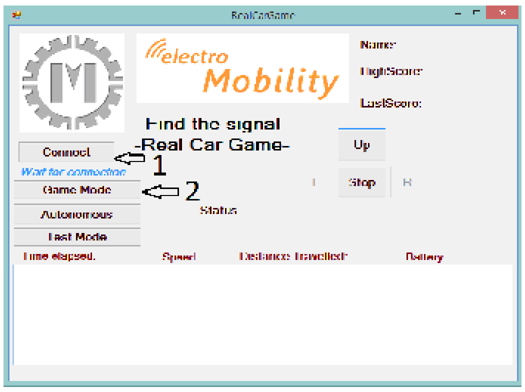


Fig. 13 – Manual selection

5. TESTING THE APPLICATION AND EXPERIMENTAL RESULTS

The main means of testing the project is achieved by the dedicated mode, ”Test Mode”, implemented in the application. This is the automated testing mode, while for manual testing we used the direction buttons observable in the interface of the application. After connection to the car, the ”Test Mode” button is pressed. A special character will immediately be sent on the serial. Once arrived at the microcontroller, it will enter a branch in the main program and it will perform different tests to the car’s components.

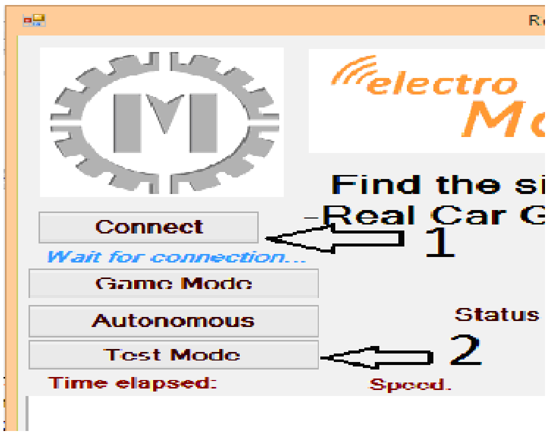


Fig. 14 – Test mode

In the following program sequence the servo and the BLDC engines were tested. Values will be sent in real time to the ”Time elapsed”, ”Speed”, ”Distance Travelled”, ”Battery” parameters, visible in the graphic interface. Also monitor the best possible or wrong functioning of information transmission, in real time. The capacity of response of the sensor Hall has been tested through the led set, created on a strip board. By nearing a magnet to the sensor, the led is lit and it is set to send 1 for detection and 0 for the times a magnet is not neared, to the serial. There is also a delay button in the program.

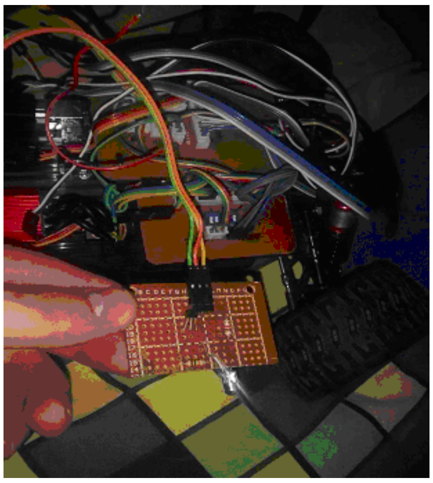
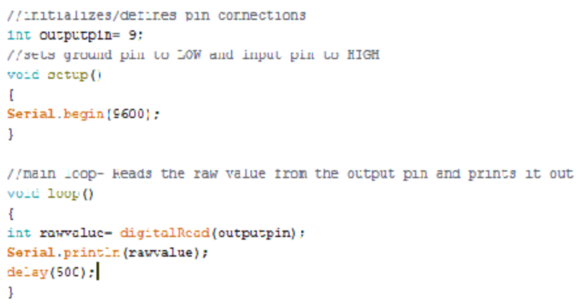


Fig. 15 – Hall sensor plate



```
//initializes/defines pin connections
int outputpin= 9;
//sets ground pin to LOW and input pin to HIGH
void setup()
{
Serial.begin(9600);
}

//main loop- Reads the raw value from the output pin and prints it out
void loop()
{
int rawvalue= digitalRead(outputpin);
Serial.println(rawvalue);
delay(500);
}
```

Fig. 16 – Test code for Hall sensor

In order to test the sensors we used the ”Autonomous-Radio” module, the semi-manual control. Forward and backward direction is controlled through the remote control and the direction is given by the sensors, based on the nearing of the car to the wall.

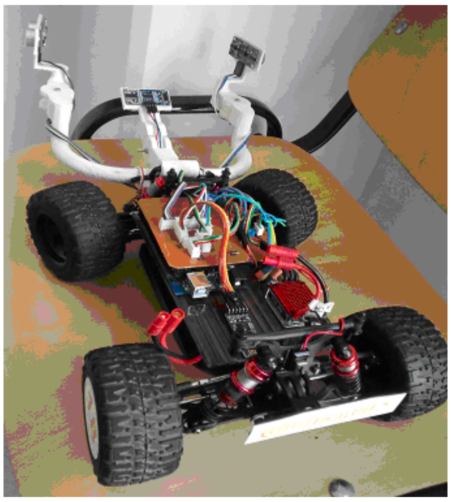


Fig. 17 – Racing car prototype

6. CONCLUSIONS

The potential benefits of autonomous cars include reduced mobility and infrastructure costs, increased safety, increased mobility, increased customer satisfaction. The self-driving, autonomous vehicle has been getting lots of attention, due to significant development efforts and dramatic progress made by companies such as Google. While general use of autonomous vehicles for widespread use on public roads is likely years away, these vehicles are already being employed in "constrained" applications such as open-pit mines and farming.

Among the many technologies which make autonomous vehicles possible is a combination of sensors and actuators, sophisticated algorithms, and powerful processors to execute software. The sensors and actuators in an autonomous vehicle fall into three broad categories: 1) navigation and guidance (where you are, where you want to be, how to get there); 2) driving and safety (directing the vehicle, making sure it vehicle acts properly under all circumstances, and follows the rules of the road); and 3) performance (managing the car's basic internal systems).

7. REFERENCES

Shahroz Tariq, Hyunsoo Choi, C.M. Wasiq, Heemin Park, Controlled parking for self-driving cars, Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference , 9-12 Oct. 2016, DOI: 10.1109/SMC.2016.7844509, Publisher: IEEE.

M V Rajasekhar, Anil Kumar Jaswal, Autonomous vehicles: The future of automobiles, Transportation Electrification Conference (ITEC), 2015 IEEE International, DOI: 10.1109/ITEC-India.2015.7386874, Publisher: IEEE.

Ovidiu Ursaru, Cristian Aghion, Mihai Lucanu, Liviu Tigaeru, Pulse width Modulation Command Systems Used for the Optimization of Three Phase Inverters, Advances in Electrical and Computer Engineering Journal. Suceava, Romania,2009, pag.22-27.

Barry Brown, The Social Life of Autonomous Cars, Published in: Computer (Volume: 50, Issue: 2, Feb. 2017), DOI: 10.1109/MC.2017.59, ISSN: 0018-9162, Publisher: IEEE.

Mike Daily, Swarup Medasani, Reinhold Behringer, Mohan Trivedi, Self-Driving Cars, Published in: Computer (Volume: 50, Issue: 12, December 2017), DOI: 10.1109/MC.2017.4451204, Print ISSN: 0018-9162, Publisher: IEEE.

T. W. Matthews; R. R. Spencer, An autonomous race car design competition, IEEE Transactions on Education, Year: 2001, Volume: 44, Issue: 2, Page: 9 pp. Publisher: IEEE.

T. W. Matthews; R. R. Spencer, An autonomous race car design competition, Proceedings Frontiers in Education 1997 27th Annual Conference. Teaching and Learning in an Era of Change, Year: 1997, Volume: 3, Pages: 1583 - 1587 vol.3, IEEE Conferences. Publisher: IEEE.

Tizar Rizano, Daniele Fontanelli, Luigi Palopoli, Lucia Pallottino, Paolo Salaris, Global path planning for competitive robotic cars, 52nd IEEE Conference on Decision and Control 2013, Pages: 4510 – 4516. Publisher: IEEE.

Thorsten Luettel, Michael Himmelsbach, Hans-Joachim Wuensche, Autonomous Ground Vehicles—Concepts and a Path to the Future, Proceedings of the IEEE (Volume: 100, Issue: Special Centennial Issue, May 13 2012), DOI: 10.1109/JPROC.2012.2189803, Page(s): 1831 - 1839. Publisher: IEEE.

AUTHOR INDEX