EnkelediMema
ElectronicEngineering
Hochschule Hamm-Lippstadt
enkeledi.mema@stud.hshl.de

Kristina Bala
Electronic Engineering
Hochschule Hamm-Lippstadt
kristina.bala@stud.hshl.de

Yahia Mesharaf
Electronic Engineering
Hochschule Hamm-Lippstadt
yahia.mesharaf@stud.hshl.de

Autonomous Vehicle Project

***Abstract*—Autonomous driving is one of the most trending technologies lately, and for the vehicle to work reliably under any possible condition, it is a very challenging and essential requirement. This paper describes the composition, testing and implementation of several components, that led to the development of a vehicle prototype, that acquires the basic autonomous behavioral functionalities. A system of LED signals was implemented, that makes the vehicle capable of communicating in an intelligent manner. The document portrays in details every part of the system and the way they interact with each other.**

Keywords—Line tracking sensor, ultrasonic sensor, I2C, image processing, video streaming, communication.

I. MOTIVATION AND REQUIREMENTS

Working in the Autonomous Car Project was a very interesting and useful experience. Our motivation and mindset were to embrace teamwork, as well as enjoy working with microcontrollers and enter the exciting field of automation, which was totally new to some of us. Building a functioning car before the deadline, was of course our main goal, but having a nice time together while working was also important. The objective was to build an autonomous driving car, using the line tracking sensors and ultrasonic sensors, together with an interface and a live streaming functionality, using a camera and a Raspberry Pi. All the system components should work together coherently without interfering or disrupting one another.

II. SKETCH OF APPROACH

How will we approach the system? After a discussion, it was agreed to split the overall system into smaller parts and to start working on simple tasks, rather than the complex ones. The framework we created, provided a space of freedom for each member of the team, to work with his task individually, and then discuss it with the team and suggest changes or different ideas. It was decided to work in a parallel way between hardware components and software, by constantly implementing and testing the ideas. The approach to the project, which started shortly after we received the components of the car, can be summed up in the following topics:

- Checking that all the components are available and working.
- Building the wooden car prototype.
- Building the hardware and connecting the components. (wiring)
- System design using UML diagrams.
- Implementation of the sensors
- Implementation of the Raspberry Pi and camera.
- Developing a smart LED communication.
- Testing phase
- Developing version 2.0
- Project documentation

III. ARCHITECTURE AND CONCEPT

A. Checking that all the components are available and working

Once the project box was received, we went through the checklist and verified the presence of all the components stated. Later, the functionality of each component was checked, and that proved to be important, as the tests showed that one of the line tracking sensors was broken, so it had to be replaced with a new one.

B. Building the wooden car prototype

After the list of components was checked, the process of building the wooden prototype started. With the help of the already built car model in the lab and by good teamwork, the building of the car was completed quickly and efficiently. Carefully, because the wooden pieces were very fragile, step by step, pieces were put in the right place and screwed all together. Thereafter, the sensors and other components were placed in the wooden prototype, but were let unscrewed, so

that adjustments could be easily made at later point, if necessary.

C. Hardware and connection of the components (wiring).

The Hardware consisted of:

1. Main Adruino Uno board
2. Secondary Adruino board
3. Raspberry Pi board and Raspberry Pi camera
4. 2 DC motors
5. Battery
6. Power supply (for Adruino)
7. 3 ultrasonic sensors
8. 2 infrared sensors
9. Other elementary electronic components (cables, jumpers, breadboard etc.)

The sensors and the motors were connected to the Adruino board, which is considered the brain of the car, while the camera was connected to the Raspberry Pi. To the second Adruino board, which was connected to the Raspberry Pi, multiple LED were placed as an additional feature that was implemented later on. In the wiring process, same colored wires were used for similar components, in order to make the connections clearer. Later on, when adjustments were made on the hardware level, this 'smart' wiring concept proved to be very helpful.

IV. *DESIGN*

A. Use Case Diagram

The use case diagram is a simplified representation of the interactions the user can execute with the system and the relationship it can build with different use cases.

Four main use cases that the target user can interact with, were identified. (Figure 3)

1. *Interaction with the Qt Interface*

A simple graphical user interface was created for the vehicle, using the Qt framework. The development started by doing multiple paper prototyping and testing them on different people, outside of the working team, to evaluate the user experience. After deciding on the prototype that would be used as a base for the interface, the following step was the graphical design.

The Qt interface includes: a small display for the video streaming; a joystick which is used to accelerate the vehicle, as well as choosing the direction; a button for turning the lights ON/OFF; a real time feedback box which informs the user about activities happening at a specific time; a menu bar for different settings; a speedometer and a status bar (for e.g. the wi-fi signal, current battery charge, etc.).

2. Manual driving

The Qt interface provides the option of choosing the driving mode, i.e. whether you want to drive it manually or not - in which case, alternatively, the car will drive autonomously. When in manual mode, the user would, ideally, be able to drive the car manually by using the joystick, but this has yet to be implemented in the future improvements of the car.

3. *Autonomous driving*

The user can choose the autonomous driving option hence, two main functionalities of the car are enabled: the line tracking assisting (realized by the infrared sensors) and the collision detection (realized by the ultrasound sensors). Therefore, the vehicle is capable of not only following the path and keeping track of the black lines, but also detecting obstacles and preventing crashing into them.

4. *Camera usage*

As presented on the Raspberry Pi and the Camera Module V2 section, there are several interactions accessible by the user, like capturing pictures, recording videos or streaming a live video from the vehicle to the Qt interface.

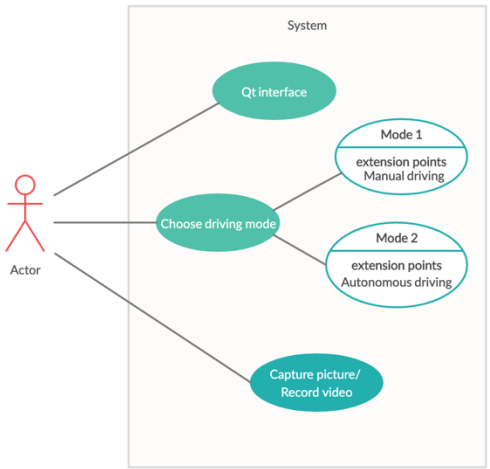


Figure 3
(Use Case Diagram)

B. State Machine Diagram

The purpose of the state machine diagram is to present the behavior of the system through different state transitions. While developing, it is very important to know the current

state of the system and what kind of activity is exactly happening there.

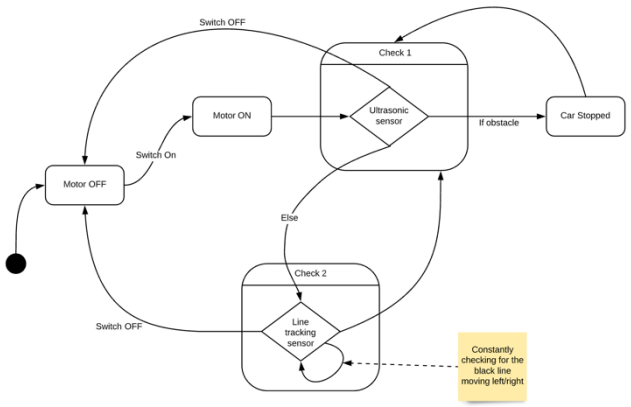
The diagram (*Figure 4*), starts with the first state *'Motor OFF'*, and after switching it on, it proceeds to the '*Motor On*' state, where the car is ready to execute its functionalities.

Initially, the vehicle starts checking the surroundings ('*Check 1*' state), using the ultrasonic sensors, in order to prevent collisions to any obstacle. In this state, three decisions are available:

1. If an obstacle is detected, the car will go to the '*Car Stopped*' state, wherefrom, it continuously jumps to the previous state, while checking for a change that will initiate another decision;
2. If there is no obstacle detected, the car will proceed to the '*Check 2*' state (explained below);
3. At any time, the user can choose to switch the car off, thus, the car would go back to '*Motor OFF*' state.

On the 'Check 2' state, the infrared sensors play their role for assisting the vehicle to 'read' the black line, by constantly checking from left to right. Hence, the car stays on track and follows the path. Just like on the '*Check 1*' state, in this state, it is also available to switch the car off.

There is a constant loop that jumps from '*Check 1*' to '*Check 2*' state very fast, which makes them run almost in parallel, by making sure the car, not only stays on the line, but also prevents crashing into obstacles on the way and responds in a real-time manner.



(Figure 4)
(State Machine Diagram)

V. IMPLEMENTATION OF THE SENSORS

A. Line tracking sensors

One of the requirements states that the car should follow a black line in an autonomous behavior. To realize this function, there was a need to use two line tracking sensors also known as infrared sensors. After some discussions with group members, it was decided to put the sensor in the middle of the car body therefore, the car was enough time to response and follow the line. The implementation of the sensors was based on state machine diagram. This technique initiated an abstract idea on how the car should operate in a different states to a concrete level of operating and also helped in building a suitable code.

When both of the sensors are tracking a nonblack area, the car has to move forward with a specific speed that could be inserted buy the user .The second behavior of the car was faced when the car gets a signal from the left sensor, that now the color has change to black .In this point the left sensor is detecting a black line so the car should turn left. After that, the sensors should both recognize white color and car could be returned to the first stage and move forward. The third case was caused when the right sensor was detecting a black area. Now the car should turn right and in this point the specific function was called. So that means the black line should be in the middle between the sensors, so that the car is able to follow it .The last case was generated when both sensors detect a black area .In this point the car was programmed to stop .The reason for making this function was the fulfillment of one the tasks of the project. By implementing this function, the position of the car was possible to be tracked and the user could relate with a real physical position of the vehicle. By adding more of these cases through the black line, a remote user that has no visual information regarding the shape of the black line, now is able to create a map and have a precise information about the vehicle's location and the direction it will turn next time.

B. Ultrasonic sensors

Another important component that should be part of the vehicle are the ultrasonic sensors. There were three of them that had to be combined together, to get a better coverage of the area and detecting obstacles in different direction. The position of the sensors is in the front of the vehicle and they were shifted with nearly 90 degrees to detect obstacles in the front, left and right. By inserting the sensors in different positions the car was able to work and cooperate with the environment in a more efficient way. Before the sensor's implementation, the information was gathered on how they would operate. After some discussions with the team members it was decided to give the ultrasonic sensors the priority to control the prototype. After this point, there was a need to change the state machine diagram, therefore, new cases were added. In this point the code implementation was relevant.

The main role of the sensors is to detect if there is an obstacle in the black line, and if they detect one, the car should change the state from running to a full stop despite what values the line tracking were generating. If it is viewed on a hierarchical scale, the ultrasonic sensors are on top and after them, the line tracking sensors. The ultrasonic sensors were able to detect the distance from different physical objects, without defining the nature of the obstacle. The distance value from the object is stored and this value would help to switch from the different states. At this point, the car has two states running and they make the decision for a full stop or decreasing the speed of the motors. The car should stop even if the line tracing sensors are generating the right values because the ultrasonic sensors have a higher priority. The second case was occured when there were no obstacles and the car was safe to continue forward in the black line. In other words, if there are no objects in the predicted range, so the three sensors do not detect obstacles, the car simply follows the line. In this case, the car control is taken by the line tracking sensors as it was defined and designed in state machine diagram. The ultrasonic sensors were implemented

and programmed to check for obstacle in every moment, without any interruption on time .The car will be able to stop immediately if something falls in the black line accidentally, hence it prevents crashing. By implementing in this way, the general safety of the car is increased. Due to further integration, the ultrasonic sensors were added another functionality. By combining together the three sensors, the car was able to find a free path. In different conditions, for example a maze, the car is able to find an available path. This function provides to the car a powerful ability to operate in a better autonomous behavior and without any surveillance.This was a demonstration that the ultrasonic sensors combined together could be more beneficial than just a single one.

As a result the two components provide to the car more abilities towards the autonomous scale. Although the sensors are collaborating together, the ultrasonic sensors have a higher priority in certain condition.

VI. *RASPBERRY PI 3 and CAMERA MODULE V2 - Live Streaming*

One of the requirements of the project is streaming a live video from the vehicle prototype to any given computer, and for that, a Raspberry Pi 3 with one of its accessories (Camera Module V2) was used. Below, the workflow steps that led to the fulfillment of the requirement are explained.

A. Raspberry Pi 3

Raspberry Pi 3 is a small single-board computer for which a standard keyboard and mouse, a monitor and a power supply is required in order to use it. This device operates in different programing languages, like Scratch or Python, and can execute every functionality just like a desktop computer.

B. Installing the operating system on the microSD card

Just like any other computer, Raspberry Pi needs an operating system to function. The official operating system '*Raspbian*' was used on the Raspberry Pi. After downloading the image, the next step was installing it on the card, and for that, '*balenaEtcher*' was used, as a graphical SD card writing tool.[1] Therefore, the microSD card was formatted and made ready to operate on the Raspberry Pi.

C. Configuration

On the first booting into Raspbian, after running on the Terminal the command line '*sudo raspi-config'*, a new tab with a few options available appears (*Figure 1*). The main functionality of this command is providing the most common configurations. Moving down in this menu bar, it is possible to change options related to network, booting, localization or interface.[2]

D. Camera Module V2 Configuration

The V2 Camera Module has a Sony IMX219 8-megapixel sensor. It can be used for taking pictures, live streaming as well as recording videos.

The configuration starts by inserting the ribbon cable in the slot situated between the Ethernet and HDMI ports, while making sure the silver pins are facing the HDMI port. Then, the '*sudo raspi-config'* command line can be run, and after navigating down the menu, under the '*Interfacing Options*', the camera can be enabled. Hence, a reboot of the Raspberry Pi is needed.[3]

Going back in the Terminal, it is now accessible to run a few command lines like: '*raspistill*' (for capturing pictures) or '*raspivid*' (for recording videos).

D. TCP/IP Networking

TCP/IP is a group of network protocols. To identify themselves, computers participating in this network use IP addresses. In order to configure TCP/IP across all of its network interfaces, Raspberry Pi uses '*dhcpcd*' which means assigning each interface an IP address as well setting netmasks and configuring DNS resolutions via the Name Service Switch facility. The '*ip link*' command line can be used to find the names of the interfaces that are present on the system.[4]

E. Live video streaming

The Raspberry Pi camera should be able to stream live videos into a web page that is accessible in any device that has a browser and is also connected to the same network as the Raspberry Pi.

After making sure the camera is enabled in the system preferences, the camera should be connected to the specific port. As a next step the '*if config*' command is run, which displays several information, including the IP addresses of the Raspberry Pi, that will be needed for the streaming. Thereafter, the command line 'python3 rpi_camera.py' is used to run a prewritten script on Python.[5]

Finally, once the script is running, the 'http://your-pi-address:8000/' webpage can be used to view the video streaming.

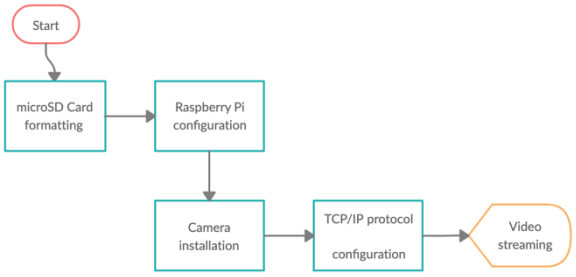


Figure 1
(Workflow Diagram)

F. I2C Communication between the Raspberry Pi and Arduino

The Inter-Integrated Circuit (I2C) is a hardware protocol which offers flexibility to interface slave integrated circuits with one or more masters. For the communication it uses two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors. Usually, the voltages used are +3.3V or 5V. [6]

It was decided to operate with this protocol because, for example, compared to other alternatives, like the SPI protocol, it has several advantages.

Firstly, I2C requires only two cables to connect to any number of slaves, whereas the SPI needs four of them for the same communication.

Secondly, I2C supports a multi-master hardware connection, which was useful considering it was worked with multiple sensors on the vehicle and therefore, they can be accessed and controlled by different master devices.

Using this protocol, the Raspberry Pi will behave as a single master, which means it will take the data from the user on the terminal, and then send it to the Arduino Uno, which will serve as a slave, thus, it will receive data at the I2C interface and display it on the Serial Monitor.

Below, is presented a picture (*Figure 2*) that shows how the connection was realized, by using an LED as an indicator to demonstrate the communication, by sending commands from the Raspberry Pi, to the Arduino, in order to turn the LED On and Off.

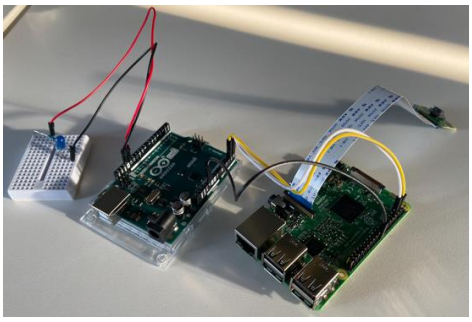


Figure 2

VII. *COMMUNICATION WITH OTHER VEHICLES BY USING LED*

A smart communication feature for the vehicle was developed. The new innovative communication consisted in LED signals. This communication was done in a simple way by using different LEDs with different colors. The LEDs were connected with the ultrasonic sensors. In the case of an object or obstacle in front of the car, the car will stop and red LED will turn on .This will indicate the presence of an obstacle and other cars that are behind, can process this image by using the camera that they had previously installed. This would make the car to stop and wait for another signal. There are no other resources used, only the existing parts. By implementing this solution the car now has three main components for operating in an autonomous way.

The second situation that occurs is when the car is moving forward and there are no obstacles and the green LED turns on. This mean that the path was clear and other cars behind can process this information and continue their operational state. If for one reason or another, the cars that are behind the vehicle would have a failure in the ultrasonic sensors, they would still be able to get signal from our prototype and will continue to operate in optimal conditions. In this case the LED signals would work as sensors for the other cars. The image processing was not fully implemented because was out of the project scope but it might be implemented at a later point.

The communication was done between Raspberry Pi and Arduino. The type of protocol that was chosen was I2C. The connection was fast and the devices were able to communicate. It was a good example to demonstrate that each device was able to send and receive information .In the Arduino was obvious that the light turns off and on but also in the Raspberry Pi, texts were sent that inform about the light state.

VIII. *TESTING PHASE*

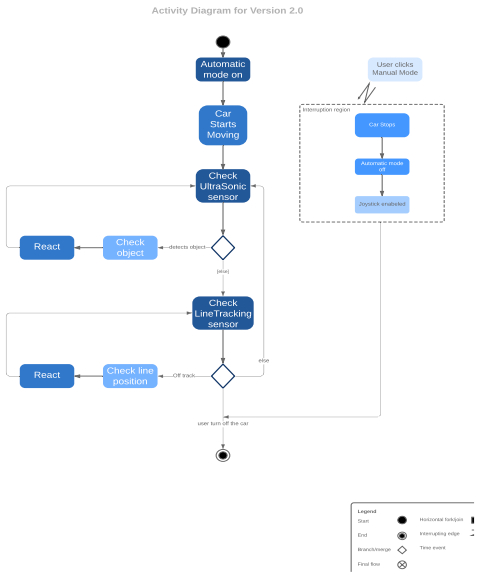
The Lab work and testing the car regularly was very important for the project. The testing phase wasn't just observing what the car was doing correctly, but instead, focusing on what the car was doing wrong and how could that be fixed, as well as trying to understand the overall car behavior. This mindset helped in eliminating errors in the system and improved the car performance and played a big role in the development of version 2.0. Having a critical eye and Inspection and notes about the tests will be illustrated more in the evaluation section.

IX. *DEVELOPING VERSION 2.0*

While monitoring our progress in the project and taking notes from the tests we were doing on the car, I started working on version 2.0. As learned from lectures, having only one version of the system is never a good idea because that decreases the system reliability and makes it not well suited for future updates and improvements. Therefore I started implementing a new version of our car, based on our current car version and on the tests we conducted throughout the project. I took the first steps in the new 2.0 version having these questions in mind:

- How can I improve our current version?
- How can I solve the problems we discovered and weren't able to fix in the first version?
- How can I prepare the system of the new 2.0 version for even more updates and improvements in following versions (version 3.0 and 4.0)?

I decided to take it step by step and I began with designing the new version by doing Activity Diagram for it as taught in the Microcontroller lectures and lab.



I started answering the questions above but not one by one, instead I answered them simultaneously.

That was done by building the whole system from scratch on a more organized basis, going through each part of the system trying to make it better than in the previous, remembering the problems figured out from the tests, leaving room for future updates and improvements and finally adding new features in the system that wasn't there before.

First, a pseudo code was written to make the overall system more understandable, then developing the main code started step by step.

For the line tracking sensor, I developed it in a sort of function that returns the value '1' if the car is moving in track and returns the value '0' if otherwise. This makes realization and the usage of the line tracking part inside the system much easier. Moreover, an algorithm was implemented for the car to try to find the black line if it cannot detect it anymore, and this wasn't present in the previous version. Adjusting the line tracking sensors position in the car placing it in a more forward position was also planned in order to decrease the shakiness detected in the original version.

For the Ultrasonic sensor, it was also developed in a similar fashion to the line tracking one as a function that returns either '1' or '0' depending if the car will hit or not. All three u.s. sensors functions were reimplemented in order to improve the car performance, the detection range was adjusted and made closer as well as decreasing the number of variables in the functions, that was done in a bid to increase the car sensitivity and reactivity to objects especially the ones that appear suddenly which was one of the issues faced in the previous version. Last but not least, four escaping algorithms were developed for the car to react to different situations it may face and avoid hitting objects then try to return back to its path instead of moving randomly or just stopping like in the first version.

For the car speed, it was simply set to the maximum which is something that could be improved and updated in following versions. All the car moving directions and even stopping have been implemented in the form of separate functions which facilitates controlling the car and developing complex control algorithms.

One of the new features is an external interrupt button which when pressed by the user enables the 'manual mode' and disables the 'automatic mode'. An interrupt service routine has been therefore implemented as mentioned in the lectures. The function of this button should be then mapped to the GUI to enable the user to control the car remotely which is a future improvement planned.

Another special feature is the connection established between the main Adruino and the Raspberry Pi from the Adruino side. The connection is done using the UART protocol because it is simple, efficient and most importantly full duplex as it allows data transfer in both directions which was learned this semester.

I started working on Version 2.0 before the presentation, but it was finished after the presentation of the testing phase is yet to be carried out and completed in future stages.

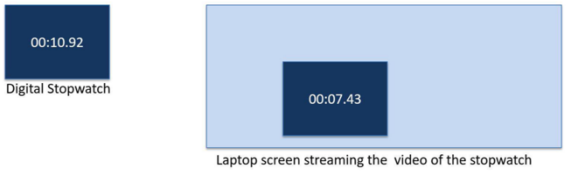
X. *CODE DOCUMENTATION*

After finishing the code implementation, the documentation of the codes was made, of both the main version and version 2.0. The idea behind that was trying to make the code as clean, organized, compressed and as easy to understand as possible.

XI. *EVALUATION*

First inspection

Several experiments were conducted to measure the latency of video streaming from the Raspberry Pi Camera to a streaming device (laptop). The experimental set up is presented on the picture below.



The latency was measured in different conditions, for example, adjusting the resolution or using internet connections with different speeds. The internet protocol used was TCP/IP and H.264 as codec.

As a result, the latency varied from 1.4 - 5.3 seconds. Therefore, it was arrived to the conclusion that the internet speed is one factor that affects the latency, meaning that the second one gets lower by increasing the speed and vice-versa. Resolution appears to also have an influence on the latency. If it is reduced, the speed of the streaming increases.

Besides the above-mentioned factors, a connection-less internet protocol UDP or more optimized algorithms for data compressing and decompressing could also result in lower latency.

Second inspectation

The decision about where the line tracking sensors should be placed, was one of the trickiest decision done in the project because there were more than one possibility and possibility has a totally different coding approach than the other. The initial idea was to place both sensors in the middle of the car; one exactly in the middle (s1) and the other slightly to the right (s2). The concept behind that was to track the black line by the sensor s1 and then if the line makes a turn the car will detect that by the sensor s2. In this concept, there were four possibilities about the value of the sensors when the car starts running which can be represented in the following truth table

Case	S1	S2
1.	0 (detects black)	0
2.	0	1 (detects white)
3.	1	0
4.	1	1

Case 1: Car is exactly on the black line

Case 2: Black line starting to turn left

Case 3: Black line starting to turn right

Case 4: Car totally off track (line isn't detected at all)

The code implementation was almost done, but later in the next group meeting it was decided to change the line tracking code and sensors position and placing them on both sides under the car and try to contain the black line with the 2 sensors instead of tracking it because this makes our system simpler, more efficient, and decreases the car shakiness. Also during the tests it was noticed that the brightness of the black line affects the line tracking sensor efficiency; for very dark mediums or very bright mediums the sensors don't operate properly due to problems with the light reflection which affects the reading of the infrared pulses generated by the line tracking sensor to detect black.

Third inspection

During the code evaluation some problems that were produce by line tracking sensors should be managed. These difficulties consist in physical constraint. Two most significant were the thickness of the black line and the light. The thickness of the line should fit between two lines tracking sensors or the line tracking sensors should open wider. In this way the state machine functions became functional. The other problem was light, due to different light changing conditions a reflection was present to the black line .The sensors were generating wrong values and as a consequence the car was taking wrong decision and did not behave in an appropriate way as it was defined .This two problems were solved in a manual way .For the case of the thickness was solved by increasing or decreasing the space between sensors in a way to fit the line .Also the light reflection was optimized by increasing or decreasing the sensitivity of the sensor .These adjustments were done manually .To realize and discover what was the best case and values some different test with different values were needed firstly to be done . in conclusion the best values were chosen and after this point the implementation took part. Also, the changes were done in the software part not only in hardware. After some different version the most efficient one was selected. Despite this fact both hardware and software need to be optimized for better usage in future. In this point this feature was implemented in a successful way and the car was able to operate independently.

XII. *SUMMARY*

At the end of the project the team succeeded in building a fully autonomous moving car with a live streaming possibility, as well as an updated version, which opens the way to further future improvements.

Although the project's duration was not long, it was a unique experience from which we acquired a lot of useful skills and one of the most important things, we learned that the systems never work as thought from the first approach. They require a lot of patience and effort to make function and operate as required. We also learned that Google can't solve all our problems, we need to use our mind as future engineers.

XII. *ACKNOWLEDGE*

I agree that I have written the paper by myself and have not used any other sources than those listed in my bibliography. I declare that I have not plagiarized any material and that this exam paper is my own work, based on my study and / or research. I have acknowledged all material and sources used in its preparation, whether they are books, articles, reports, lecture notes and any other kind of (electronic) document. I have not copied any part of or plagiarized the work of any other student and / or person.

Enkeledi Mema

Kristina Bala

Yahia Mesharaf

Contributions :

Enleledi Mema 40%

- Line Tracking sensor code
- Ultrasonic sensor code
- Development of I2C Communication
- Smart Led communication
- Inspection no.3

Kristina Bala 30%

- *Abstract*
- *System design using UML diagrams.*
- *Use case diagram*
- *State machine diagram*
- *RASPBERRY PI 3 and CAMERA MODULE V2 - Live Streaming*
- *Installing the operating system on the microSD card*
- *Configuration*
- *Camera Module V2 Configuration*
- *TCP/IP Networking*
- *Live video streaming*
- *I2C Communication between the Raspberry Pi and Adruino*
- *Inspection 1*

Yahia Mesharaf 30%

- *Motivation and Requirements*
- *Sketch of Approach*
- *Architecture and Concept*
- *Testing Phase*
- *Develpoing version 2.0 (code, diagrams and system design)*
- *Code Documentation*
- *Inspection no.2*
- *Summary*

Group no. 5

Github link :

https://github.com/KristinaBala/Autonomous-Car