Autonomous Vehicle Project

1st Luca Brodo
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
luca.brodo@stud.hshl.de

2st Oguzkaan Tuna
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
oguz-kaan.tuna@stud.hshl.de

3st Giuseppe Scalora
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
giuseppe.scalora@stud.hshl.de

4st Gordan Konevski
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
gordan.konevski@stud.hshl.de

***Abstract*—The world is moving rapidly towards automation. Machines capable of solving tasks autonomously would make people's life less complicated and effortless. Moreover, they can also help people with disabilities, because machines can handle everyday tasks for them such as going to work or to the shop. A big topic in automation is in fact transportation. The direction in which the industry is moving is towards an always more automated driving experience. The prototype described in this paper is a first attempt of developing a vehicle capable of solving the most basic tasks of driving which are moving in different directions, following a path and avoiding obstacles**

I. Introduction

The autonomous driving vehicle is an electronic device, or miniaturized car to be more precise, which, with the aid of different electronic components all intended to work with each other, is capable of autonomously drive within an environment potentially full of obstacles and to follow a straight black path placed/taped on the floor. The main purpose of the creation is to get familiar with the use of electronic sensors and microcontrollers that allow the engineer or the projectists to obtain information from the physical world, throughout the use of the sensors, and consequently transform these information in electric signals or quantities that will be delivered by the sensors to the microcontrollers. Therefore, once the information is being received by the microcontrollers, these will do some actions that are predetermined in the code that has been created from zero by the software developer or programmer. The possible scenarios that the car has to deal with are the following: The car must be able to drive following a black path taped on the floor, without ever going off track. It must be able to avoid obstacles that may appear within the track and afterwards going back to the track. The vehicle must be able to stream a video from the small camera mounted on top of the car frame. (optional) The car must follow the instructions given by a specifically designed user interface made of buttons, sliders and indicators.

II. Motivation

A. Main Objective

The goal of the project was to develop a small autonomous vehicle. Set on the floor were some black lines (ordinary black tape), set up in a way that would resemble a simple driving track. The entire development process would revolve around such a track, as it represented the pathway which the vehicle itself had to follow autonomously. A second, equally important element was the concept of obstacle avoidance. Not only was the vehicle tasked with following a certain path, but it also had to demonstrate an ability to avoid physical obstructions, albeit in a simple manner that mostly shed light on its potentiality rather than showcased any sophisticated maneuvering procedures or complex real-time decision making. Finally, a third, just as equally important task was the implementation of a Qt-based interface, with which a user could control the vehicle manually through their device. This eventually led to an interface only limited to a computer program with a rather complex setup process - at least to users who are unfamiliar with the hardware and software at hand. To summarize, a vehicle was to be constructed that would be capable of autonomously following a set path, avoid obstacles and would also allow a user to control its movement manually, whenever requested.

B. Resources and Materials

A lack of online resources and demonstrations led to some difficulty in the end stages of the development process, the Qt interface and its connectivity to the vehicle being a notable example. Regardless, the tasks were still manageable. The actual physical materials themselves were composed of:

- A simple wooden chassis, with the front part being divided into three, and the bottom part being somewhat elevated
- A pair of low-cost IR sensors
- Three low-cost ultrasound sensors
- One pair of motors with their respected speed control board (one of the motors will prove to be faulty, though not unusable)
- One Arduino Uno board
- One Raspberry Pi board and one Raspberry Pi camera
- Elementary electronic elements form a basic electronics kit (e.g. jumper cables, breadboards)
- Two batteries

III. Limitations

The biggest limitations were mostly hardware and environmentally based. For instance, as mentioned earlier, one of the motors proved to be faulty. The problem laid in the fact that the motor could not output the speed necessary in order to keep up with its corresponding motor. Even though a software-based

solution was implemented (using the Arduino interface to manipulate the speeds), it only partially mediated the problem. Nevertheless, the interaction between the different devices and the central control unit would later produce the desired results. Moreover, the first two motors (Modelcraft RB35), which we got, were not working properly. They were, in context of power, really low and made unusual noises. For that reason, we decided to unscrew the motors, to see if there was a problem with the components. Following that, we found out that the motors were severely damaged inside, so the motors could not fulfill the requested instructions. The metal circle inside was bend abnormally and it limited other parts to execute their functions. After this discovery, we requested two new motors to continue the project. However, also the new motors had the same problem. In our third and final attempt, we managed to get two fully functional motors.

One notable issue was linked to the hardware used, i.e. the hardware limitations of the sensors themselves meant that the vehicle could only work in specific environments. The ultrasound sensors, as an example, could only reflect signals from certain materials, and the IR sensors could not function properly on reflective surfaces (changing its sensitivity could only help so much).

IV. Assembling the frame

This specific paper will discuss about the overall construction process of the car, the assembling, the hardware management and the importance of the creation of a user manual. With that being said, to get deeper into the topic let us begin step by step by describing the processes just mentioned. The main part to initially deal with is obviously the construction part. Wooden parts of the frame were given and they needed to be connected and assembled together in order to create the so called "skeleton" or frame of the car. The parts are several and they are listed in the following segment: a main part that is the flat wooden base, in which every other part must be mounted and assembled on and it is also the base where the batteries and the microcontrollers will be laid on. Following there are different wooden "walls" that are needed as barriers and delimiters for the components that belong to the car. Then there are the motor supports and the sensor supports, such as the ultrasound sensor supports. The last wooden part is a wooden arm that holds up the Raspberry Pi camera. The beginning of the assembly starts with the placement of the small rotating sphere, used to allow the car to change direction in a 360° range, on the bottom part of the wooden base, and the I/O switch fixed on the wooden rear part of the car, once both are in place it is possible to continue the assembling process with the other wooden parts. Once these 2 small elements are in place, there are 2 specific parts that need to be pre-assembled before being mounted on the main car frame, and these are the motors supports. The motors need to be screwed first to these supports with 3 to 4 bolts, much awareness is required in this step in order to avoid possible cracks or fractures in the wooden frame. After the motors have been placed it is possible to proceed by mounting the pre-assembled supports onto the base. During this part one of the motors, unfortunately, resulted to be in bad conditions and had to be swapped with a new one. The assembling procedure goes on as the ultrasound sensors supports must be screwed in the front top of the wooden base, these supports will serve the purpose of keeping the ultrasound sensors stuck in position to fulfill their task (explained in the next section). Lastly it is necessary to build in the last 3 wooden parts which will be placed in the middle of the car and as a cover for the motors, but after an attentive analysis it has been decided it was better to keep these parts for the last steps of the construction as they would result in occupying useful space needed in the next steps.

A. Hardware and Cable Management

At this point the most sensitive part of the construction process begins. First of all it is good practice to list the required hardware components and equipment. Beginning with the power suppliers, only two batteries are required: a 12 volts DC battery to supply energy the motors and a 5 volts DC battery or powerbank to supply energy to the microcontrollers. Moving on to the motors, a pair of 12 volts DC motors were provided, with a couple of rubber wheels that were pre-adapted for the motors with two 3D printed supports. Next, considering the Arduino cannot deliver a high enough current to the motors to make them run, an L298N module was needed, which has the capability of delivering up to 2 A for each output and can pilot up to 2 motors at the same time. Following up a pair of infrared sensors was given alongside 3 ultrasound sensors. The infrared sensors with the purpose of being mounted on the bottom of the frame to detect the black path taped on the floor and the ultrasound sensors serving as obstacle detectors. The number of ultrasound sensors and infrared sensors is not random but is based on the concept of having 2 infrared line detecting sensors on the bottom to keep the car always centered on the black path, not allowing it to go off track. The 3 ultrasound sensors, on the other hand, are placed on 3 specific positions in the car, exactly at the top front of the car, the middle one is perfectly horizontal meanwhile the other 2 are placed on the sides with an angle of around 30° in opposite directions. The concept here is to make the car able to detect obstacles that are not only in front but also on the sides and act by consequence. Moving on to the construction part, it is important to point out that there is no specific placement of the hardware and components so everything is up to the constructor. First of all, as already said, the DC motors were placed in the rear sides of the car, then the 3 ultrasound sensors were stuck in their apposite supports with the aid of hot glue, in order to keep them fixed. A similar process was applied to the infrared line detecting sensors, they have been stuck in the bottom part of the frame base, here it is necessary to state that the team required different trials before finding the perfect placement and width of these 2 sensors as the black path is not always the same and it might differ in width and reflectiveness. With that said, after many tries, a decent spot was found for both sensors, now being able to really keep the

car on the track. Once all the sensors were in place all the final components have been placed on the car, components such as the Arduino, the raspberry pi, the small breadboard for the cable connections, the battery and the 12 V LiPo battery. All these lastly mentioned components were only taped to the car with the idea of being able to easily remove them if needed. At this point the only thing missing was the cable connection. To approach this part it was first necessary to carefully analyze all the possible free space that could be used to let the cable freely move inside the car to the position they were supposed to be connected. 4 cables are originated from each ultrasound sensors (Rx, Tx, 5 V, GND) that makes a total of 12 cables that were stuck together in groups of 4 to make them look cleaner and visually better. From the line detecting sensors 3 cables are originating (Signal, 5 V, GND) and similarly to the ultrasound sensors, the 3 cables have been grouped together to save space and avoid possible interferences with other cables. One extra cable connection has been made between the camera and the Raspberry Pi to serve the purpose of streaming a real-time video from the car point of view. At this point not many cables were left, beside the connections to the L298N module which included the 4 cables coming from the motors, respectively 2 for each motor (5 V and GND) and the enabling connections with the Arduino. At the very end the microcontrollers have been connected to the whole system. Since both Arduino and the Raspberry Pi need a tension of 5 V to be able to function and there was only a single battery with a single output, it has been decided to give power only to the raspberry pi and from it a 5 V output was taken and connected to the Arduino input in order to supply power to it. After this long process that took several steps, the car was ready to be tested and drive around in an artificial environment.

V. DC Motors

Our prototype requires 2 DC Motors, in order to accomplish the given tasks. Electrical DC motor is an element, which converts electrical power into mechanical energy. Its action is created by the behavior of electromagnetism. Unlike other electronic components, the DC Motors needs, E.g. High Voltage, are beyond the capabilities of our microcontrollers, which can only provide the rotation or acceleration instructions. Therefore, an external module is needed, to produce the needed power for the motors.

A. Motor Control

The L298 is a high voltage, high current dual full bridge driver, that can drive the DC Motors. It drives the motors with 3 input for each, which are; EnA (Enable A) In1 and In2 (Inputs 1/2) and EnB (Enable B) In3 ans In4 (Input 3/4), where EnA or EnB allow us to enable and disable the motors independent of the input signals. This means in order to start the motor, we first need to bring the Enable pins to a 'High' state. To archive this, we used the PWM (Pulse Width Modulation) pins on the "Arduino Uno". PWM pins allow us a better speed stability, reasoned by the fact, that the switching transistor has a much reduced power dissipation, giving it a linear type of control. For the inputs, we used the regular Analog pins on the microcontroller, so we can control and also set specific limits on the speed of the motors.

B. H-Bridge

An H-Bridge is a circuit, that allows a voltage to be applied across a load in either direction. H-Bridges are built with four switches, where the states of the switches decide the rotation of the motor. For instance, if the switches S1 and S4 (Fig n. 1) are closed, the current can travel from left lead of the motor to the ground, which makes the motor start spinning. On the other hand, if the switches S3 and S2 are closed the current travels from right lead of the motor to the ground, which makes the motor start spinning in the reverse direction. However, the H-Bridge must be used with caution, since if the switches S1 and S2 or S3 and S4 are closed at the same time, a short circuit is created, which could damage the electronic parts in the circuit. This situation is called "shoot-through ".

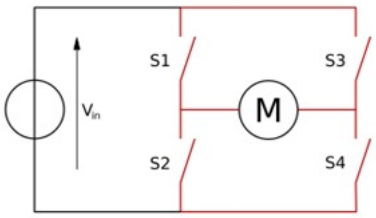

Fig. 1. Diagram of the H bridge.

C. UltraSonic Sensors

The prototype should be able to recognize and avoid objects. For that reason, the communication and the relationship between the motors and the ultrasonic sensors are essential. In our system if one of the sensors on the sides detects any obstacle, the car will avoid it by slightly reducing the speed of the motor on the opposite side, which is done with the help of the Analog inputs, where the value of the signal to the motor is reduced compared to the other one. If an object is in front of the vehicle, we force the motors to stop by disabling the PWM pins and depending on the availability of the environment, it drives to the left or right.

D. Infrared Sensors

Another assignment, which the prototype should fulfill, is to recognize a path and follow it. Therefore, the infrared sensors are used. As a result, if a path is recognized by the sensors, instructions are sent to the motors to move to left, that if the infrared sensor on the right side is turned on. Or the opposite way, to right, if the sensor on the left is turned on. Therefore, the car replaces itself in the center on the path.

E. QT Interface

With the created interface, the user is able to control the vehicle in desired directions. In order for the system to work efficiently, every possible direction is treated as a separate case. Therefore, we tried to operate the motors with the

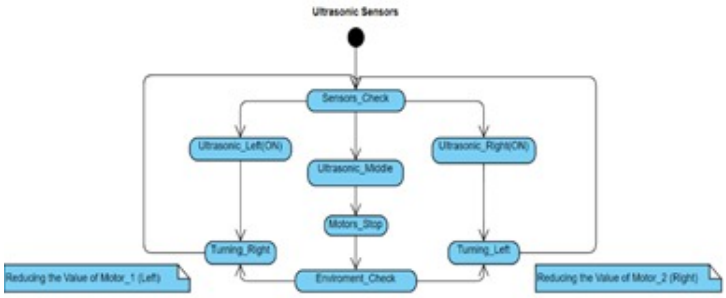

Fig. 2. Diagram of the H bridge.

Switch case structure. This enables us to bounce between the different cases really quick and in a reliable way. Every move of the "Joystick" in the interface sends a variable to the microcontroller, which then directs it to the 'Switch' case, where it is checked for the state in which the variable equals. So, the motors can adjust themselves to the instructed rotations and speed.

VI. Line Detection Capabilities

A. Objective

One of the essential goals of the vehicle in question is to allow the central unit (itself composed of an Arduino and a Raspberry Pi) to govern the behaviors of the vehicle autonomously. Two essential elements are required for such a task, one being a specified 'task' that the vehicle would be ordered to complete and the other being 'the direction' to follow. Both elements were already predetermined. The direction was outlined by a path marked with black tape, in the form of a loop, while the task itself was as simple as driving the vehicle forwards, avoiding obstacles and following the black tape in the meantime.

B. Infrared Sensors

In order to incorporate the line-following capability, a pair of small infrared sensors were glued to the sides in parallel and connected to the Arduino board. The behavior of the motors was closely linked to the signals provided by the IR sensors, so whenever one sensor would detect a change, i.e. detect the line, the motors would change their course of action correspondingly. A notable issue was the inability for the sensors to function normally even with the slightest reflectivity on any surface. The code for the sensors was written using the Arduino IDE.

C. The function of the motors in the context of the IR sensors

The vehicle was designed to always move forwards by default. Only with change in the environment would a different direction be taken. The angle in which the vehicle was situated was not taken into account, rather the change of directions was simply done by changing the speeds of the motors with a series of 'if' statements and corresponding functions. As an example, if the left sensor were to detect the line, the left motor would decrease its speed, allowing the right motor to change the direction of movement. A unique function was also implemented, where if both sensors were to detect the line, the vehicle would turn in alternating directions, allowing it to position itself on the line. This function was a bit difficult to implement due the failure of one of the motors to supply enough power, thus constantly skewing the vehicle to the side. Regardless, the vehicle was still capable of following the black tape line successfully.

D. Interaction Between the sensors and other elements

The sensors were designed to only function when no other behavior altering functions were in use. Thus, if the ultrasound sensors were to detect an object, the signals of the IR sensors would be ignored, until the action taken by the. If manual control was activated via the Qt interface, the signals would, again, be ignored. Its only interaction with the other elements was to deactivate.

Fig. 3. Activity diagram showcasing the interaction between one sensor and its corresponding motor.

VII. Ultra Sound Sensors

In parallel with the infrared sensors at the bottom, the prototype described in this paper collects data from the environment using three ultrasound sensors mounted at the front. The main purpose of these three ultrasound sensors is to detect the presence of the object and send information to the Arduino which will then elaborate the informations and react to it. The sensors used in these project are the ultrasonic rage finders HC-SR04. (Fig. 4)

A. Principle of Operation

The basic principle of these ultrasound sensors is to use sound waves and calculate the distance from an object on function of the duration for which the signal is emitted. The entire process is based on the speed of sound, which is approximately 343 m / s at 20 degrees Celsius at dry air. This specification is necessary since the speed of sound, or a wave in general, is dependent on the material which it goes through. Since sound in this context propagates through air, humidity and temperature are important factors to take into consideration. Air and humidity change the way particles are disposed in space and those are the elements that vibrate and make it possible for the sound to propagate. Although all the previous discussion is true, using the known speed of

sound would be enough since the variation in humidity and temperature will not influence it so drastically.

These sensors are essentially composed of 2 parts, a transmitter (T) and a receiver (R). The names themselves are pretty self-explanatory. The transmitter is responsible for emitting the sound waves at 40kHz. Those waves will eventually bounce back from a surface and the receiver is responsible for receiving them. More specifically, HC-SR04 sensors have a pin for the transmitter (Trig) and one for the receiver (Echo) .The other 2 will be ground and 5v.(Fig. 4) At the beginning, those pins need to be both LOW (or 0). In order to start the process, the Trig Pin needs to be set as 1 (or HIGH) from the Arduino using the function digitalWrite. This will make the Transmitter send 8 short pulses to simulate a sound wave. After a delay of 10 milliseconds (achievable with the function delayMilliseconds) the pin has to be set back to 0 again. Once those pulses are sent, the Echo Pin needs to be set as 1 and it will stay in this state until the pulses come back to the receiver, which will then set it back to 0. This can be simply done by the function PulseIn in Arduino. The duration corresponds to the amount of time during which the Echo was set as 1 (Fig. 5). Obviously, the key in this process is timing. The more precise it is, the more accurate the value of the duration will be. Having this in mind, predefined functions such as digitalWrite in the Arduino standard library might not seem perfectly suitable. They in fact hide some lines of code which make the thread a little bit slower and can be substituted by manipulating ports. In this way, the thread will run a bit faster. However, readability would be affected by that and so would be troubleshooting. This bargain might be beneficial in some other context, but not here.

The equation used to calculate the distance once the duration is calculated is the following:

$$S = v \times T$$

Where S is the distance, v is the speed of sound and T the time of the duration. However, this equation must be adjusted in this way:

$$S = v \times Tim/2 \times 10^{-6}$$

Where Tim is in microseconds. The first equation does not take in consideration that the duration outputted by the sensor is in microseconds and that the wave travels half the time for reaching the object and the other half for going back to the sensor. Finally, if we want the distance in centimeters, the final equation will result as follows:

$$Scm = 0.0017 \times Tim$$

Where Scm and Tim are respectively in centimeters and milliseconds

For the main purpose of the project, these ultrasound sensors work perfectly fine. In fact, even though they present limitations such as the angle of reflection and the material of the object, during the tests they behaved most of the times as expected, reacting quite accurate to most of the objects. A problematical situation faced more frequently than the other was with small objects which can not be detected by the sensors. It is also important to note that the range of this sensor is limited both in distance and width. In fact, they can only detect objects in a range of 2 meters and whose position is in their 30 degrees spectrum. This spectrum limitation is the reason why there are 3 of them at the front, situation that create an approximate wide 90 degrees aperture.

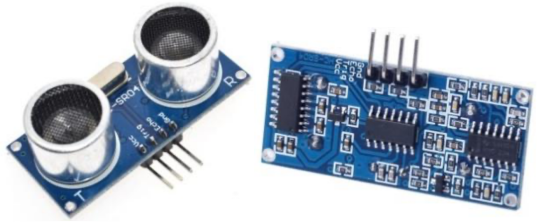

Fig. 4. Picture showing an example of HC-SR04

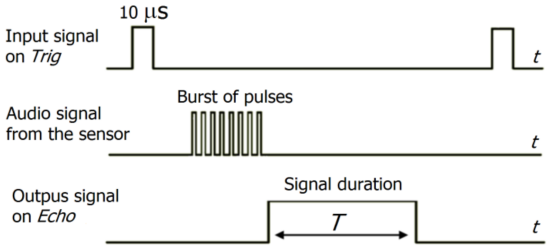

Fig. 5. Representation of how the ultrasound sensors work

B. Implementation of the sensors according to the context

The whole object detection process takes place in a protothread, different from the main one. Arduino is a mono core microprocessor, so the creation of real parallel threads is impossible. A solution to this problem is to use protothreads using the library pt.h [3]. This library provides a conditional blocking wait operation that can be used to call threads which will run until blocked or the execution is finished. This wait conditional blocks make possible to call the threads after a specific time interval. In this scenario, the protothread is called every 50 ms. In addition, the object detection functionality has been implemented having the concept of a state machine in mind. As shown in the diagram in Fig. n (11), there are 4 different states in which the prototype can stay. The central one, namely the default state, represents the state in which the car stay when no objects are detected and it is free to go forward. The other states are related to each sensor. If the distance calculated in one of the sensor is less than 20 cm, then the prototype goes into one of these states depending on which sensors calculates the 20cm distance first. In the diagram, the number ”1” symbolizes that the distance from the object is less than 20 cm, while ”0” refers to a distance inferior to 20cm. If the car finds itself in one of these states, its purposes are

to turn 180 degrees in order to avoid the obstacle in order to come back on track and send to the interface which sensor detected the obstacle. In order to do so, a 'if else' statement has been implemented where the distance is calculated for every sensor. Even though this is not as fast as a switch case (a switch case implementation will have a O(1) asymptotic growth, while this has O(n) in the worst case), it seemed a suitable option since there are not many conditions and both finding errors and trouble shooting result easier. In addition, this implementation suits the diagram thoroughly because the prototype returns to the default state once all the operations end.

VIII. User Interface

One of the most important part of the project is represented by the user interface. The main purpose of having it is to allow the user to make important decisions and impart commands to the prototype. In addition, the user interface is responsible for notifying important messages to the user. The entirety of the interface was designed having in mind one of the Dieter Rams' ten principle of design which goes as following : "Good design is as little design as possible". [1]

The interface is in fact devoid of unnecessary elements, or to quote Rams it is "not burdened with non-essentials". What the user sees is a clean and minimalistic interface where elements are disposed in an easy-to-understand way. Both the layout and the design of the elements put at ease also the less experienced user who also does not feel overwhelmed by the amount of objects. The two predominant colors are dark blue and white. The dark blue is used as background because darker color are known to tire less the eye of the viewer. On the other hand, white is used for the elements allowing them to stand out meanwhile creating a nice and elegant contrast with the background.

The first recognizable element is the picture which represents a well-known real car in the middle of the interface .(Fig. 9) The angle of view tricks the eye simulating a real 3D scenario of the car in a road. This ploy is used to display whether or not the car is crossing the street line using the two lines as signifiers. The two figures at the sides of the car which simulates the road turn red if the line is crossed either left or right. In order to avoid confusion, they do not turn red at the same time but they change color corresponding to the side in which the line has been crossed. This whole scenario makes possible even for not-well trained eye to extrapolate information from it and react to them as fast as possible. As a first approach, a 2D top view of the car was taken in consideration. This model had the same characteristics as the first one mentioned. However, it did not reflect the reality as close as the other and it resulted unpleasant to the eyes. A 2D top view has been instead chosen to display the location of a near object. An example of it can be seen in Fig. n (Fig. 7). This mechanism is triggered when one of the ultrasound sensors in the front detects an object which is closer than 20cm. The immediate change in perspective is clear and the user immediately recognize it which makes it faster for him to react. In addition, the bright red color of the icon suggests danger and the eye is captured by it. Moreover, this new 2D view of the car is a zoom of the front part of the car and this suggests the object is very close. On the left side of the interface a virtual joystick is placed and it represents the core of the interface (Fig. 8).The joystick is made of two parts. The outer circle, which is a bit darker than the other, represents the boundary while the inner one is the fundament part to give commands to the car. The inner circle is in fact a moving element and the user can directly interact with it by moving it around. The direction in which the inner circle is moved will then be the direction in which the car will go. As long as the inner circle is moved or pressed, the back and front lights of the picture of the car will light up which signifies that the car is turned on and moving. (Fig. 10) When the circle is released, it goes back to its normal position, in the center of the other, the car will immediately stop and the lights will turn off. (Fig. 9) Finally, the other elements that compose the user interface are two switch buttons, a divider, which is a design element and not a usable one, and a progress bar. The two switch buttons are positioned at the sides of the divider. The left one, when switched on, shows the camera view and if switched off, it interrupts it. On the other hand, the right one switches on and off the automatic mode, namely when the car goes around following a line and avoiding obstacles.

IX. Connection between Devices

The purpose of having a form of communication between the three devices is to allow the user to control the movement of the prototype directly via the interface. The basic idea behind it is to use the RaspberryPi as a bridge between the Arduino and the user interface taking advantage of its integrated network card. As shown in the diagram (Fig. 6), the RaspberryPi and the laptop are connected with a Wifi connection using both User Datagram Protocol (UDP) and Internet protocol suite (TCP/IP) protocol. Those protocols have been implemented for two different functionalities.

A. Implementation of the UDP protocol

A type of communication using a UDP protocol is achieved by transmitting information from client to server without verifying the state of the receiver. Packages of data are sent in datagrams which are small in size and fast since the protocol is not designed to check the order of arrival nor the state of arrival. Even though in this way congestions are prevented, the communication may result in an unreliable state since the order can not be predicted. Given these characteristics, the UDP protocol is suitable for sending directions and commands to the vehicle. Using the available class in the qt library called "QUdpServer", the laptop is able to open a socket-type communication with the Raspberry Pi. The user interface presents to the user a virtual joystick which would be used to give directions to the car. This area is divided in 8 different equally large zones (right, forward-right, forward, forward-left, left, backwards-left , backwards and backwards-right) and moving the inner circle of the joystick all the way to a specific area

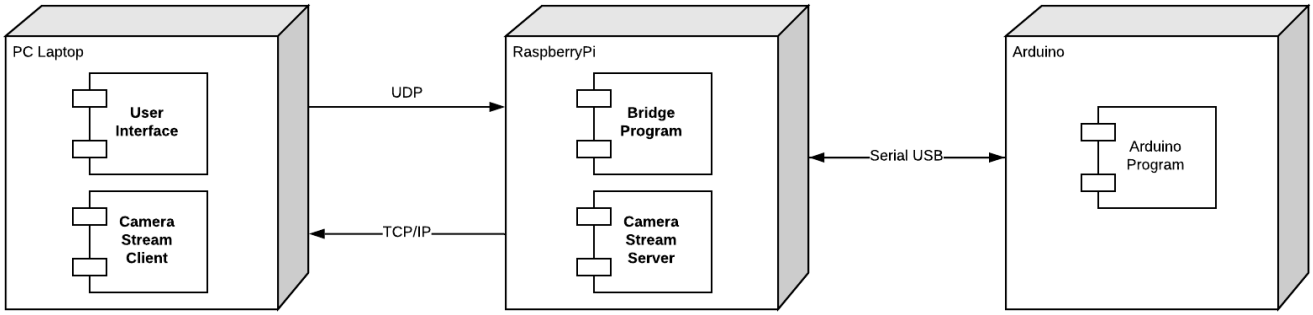

Fig. 6. Diagram displaying the architecture of the system.

results in the program sending the corresponding characters to the RaspberryPi. This program will then receive the character and transmit it to the Arduino via serial communication. The Arduino will then map the characters to the specific position using a switch case. This is very easy to implement and very fast to execute because a switch case will be compiled in a jump table. However, the way this function is implemented is an oversimplification of what it really should be. As a matter of fact, the best implementation would be to map the position of the inner circle in the joystick with a function that gives as output direction and speed. Another functionalities which uses the UDP protocol is sending data from the Arduino to the Qt program. When the Arduino detects an object with one of three ultrasound sensors in the front, it sends to the RaspberryPi the position using the three characters 'r', as in right, 'c', as in center, and 'l' as in left. The RaspberryPi will then send them to the computer, which is constantly checking those in a designed thread, and then display the position of the object in the interface.(Fig. 7)

B. Implementation of the TCP protocol

When it comes to streaming the camera, however, the UDP protocol is hard to implement given the the fact that the image needs to be fragmented into packages and those might get lost. A better approach would be to use the TCP protocol. This protocol allows to stream heavier packages whose order is checked every time. This makes the connection more reliable but a bit slower than the one with UDP protocol, hence more suited for a camera stream. The program utilizes the OpenCV library which requires to be installed in both the raspberryPi and the laptop. Using OpenCV, the raspberry Pi is able to capture data from the camera frame by frame, store it in a "cv::Mat" type and send the data to the TCP socket. Once the data has been entirely collected by the laptop, the picture needs to be reconstructed pixel by pixel. It's a very intricate process, which however allows a wide degree of freedom. In fact, once the picture has been reconstructed, it can be processed. For example, the program so far is able to find contours and draw them. More importantly, this implementation can be embedded with the interface since both utilize C++.

Fig. 7. Example of the interface showing where the object is.

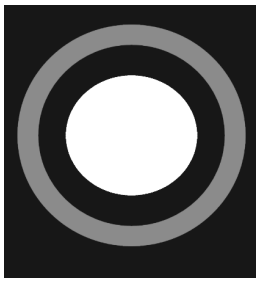

Fig. 8. Picture showing the joystick situated in the left part of the interface.

Fig. 9. Representation of the car in the interface

Fig. 10. Detail of the car when the controller is moved

X. User Manual

Within the construction process an extra task needed to be solved and after discussing how all the construction process works it is important to clarify some extra things regarding the device. As an electronic device it is necessary to deliver it alongside a user manual that contains all the necessary instructions needed to make the device function in the correct way and also all the warnings and details one has to be aware of while using it. The user consulting the manual has the possibility to find out, in a very easy manner, how everything in the car works, how to set everything up before turning on the car and how to deal with possible problems that may occur as a result of software or hardware accidents.

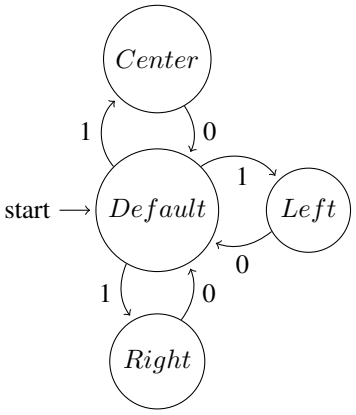

Fig. 11. Figure displaying the states in which the car can stay. "1" symbolize that the distance from the object is less than 20 cm, while "0" refers to a distance inferior to 20cm. "Left","Right" and "Center" are the states triggered when a object is detected in the corresponding position

XI. Evaluation

From the construction point of view, the final results have been very satisfying. Despite encountering some early problems with the assembling part and discovering some defects in some parts, the overall project was a success. Regarding the frame, the main structure held everything perfectly without giving troubles although at some point it was necessary to fasten the bolts and screw one more time just for safety. In terms of hardware and components everything had its own working space and the cable management made sure that there was enough space to deal with the cables at a later point in case of changes in the structure or the circuit. The car was able to keep everything fixed and behave in the correct manner doing what it was supposed to do at every moment.

XII. Conclusions

As the main purpose of the whole project, was to be able to learn and deal with the interaction between the physical world and the electronical world, the main concepts discussed in the paper are useful to understand how this interaction occur via the use of specific equipment and devices. The main points proven in this project are in fact, the capability of the car of adapting to the surrounding environment and interacting with it following instructions that are given by the creator/s.

References

[1] Norman, Donald A. The Design of Everyday Things. New York: Doubleday, 1990. Print.

[2] Zhmud, Vadim and Kondratiev, N and Kuznetsov, K and Trubin, V and Dimitrov, Lubomir,2018, Application of ultrasonic sensor for measuring distances in robotics, 1015,Journal of Physics: Conference Series,

[3] Dunkels, Adam and Schmidt, Oliver and Voigt, Thiemo and Ali, Muneeb, Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems,2006