ROBOT PROJECT

ES 2 – Phase 3

Teacher: **Bakx, René R.P.M.**

Date: Monday, 18th November 2022

Student: **Sean Brokke, Andre Sanao, Covalciuc Victor**

Abstract

Approaches, conclusions, and sources in regards with the main project of the Embedded Systems 2, Semester 3, course from Fontys University of Applied Sciences is presented in the following chapters. The main project revolves around combing prior semester knowledge to achieve a practical modern application of it

Version History

Version	Date	Status	Author	Description	Remarks
1.0.0	18.11.2022	Draft	Victor Covalciuc	▪ Front page. ▪ Version History ▪ TOC + TOF + TOT ▪ Abstract ▪ Introduction ▪ Acronyms ▪ Document Formatting ▪ Document Structure ▪ Document Base for Writing ▪ Phase One (Introduction) ▪ Phase One (Procedure, Conclusion) [Writing Start]	DONE
1.1.0	01.12.2022	Draft	Sean Brokke, Andre Sanao, Victor Covalciuc	▪ Phase One (Procedure, Conclusion) ▪ [Writing Continuation, End] ▪ Corrections ▪ Conclusions per entire project ▪ Phase Two ▪ Phase Three ▪ Individual Reflection ▪ Conclusions per entire project	DONE
1.2.0	19.01.2023	Final	Sean Brokke, Andre Sanao, Victor Covalciuc	▪ Phase Three ▪	DONE

- Highlighted in green is the current version on which the document is on.

Table of Tables

Table Of Figures

Table Of Contents

❖Introduction

The project on which this document presents and keeps track of its approaches and findings, has the end goal of materializing and merging knowledge from other subjects and Embedded System lessons into one applied application, a robot.

Tasks in the assignment require the use of knowledge from other subjects in order to build a functioning robot that has its focus on the underlining the key concepts learned in the embedded systems course and the other courses, therefore the project is split into four phases, each regarding one concept, concerning one matter of each subject.

❖Acronyms

Table 1 – List of acronyms used throughout the report

Acronym	Meaning
ES	*→ Embedded Systems*
PWM	*→ Pulse Width Modulation*
FCS	*→ Feedback Control Systems*
PID	*→ Proportional Integral Derivative*
KP	*→Constant Proportional*
KI	*→Constant Integral*
KD	*→Constant Derivative*

❖Phase One – Robot Distancing

▪ Introduction

This is the starting phase of the project, and in this one it was needed to put together all the other past assignments from Embedded Systems in order to practically apply the knowledge and observe a real-world application of it i.e. put together a moving robot. Naturally, as this Is only the initial phase, only basic functionality is required and further presented in the next section.

▪ Procedure

High-Level Design

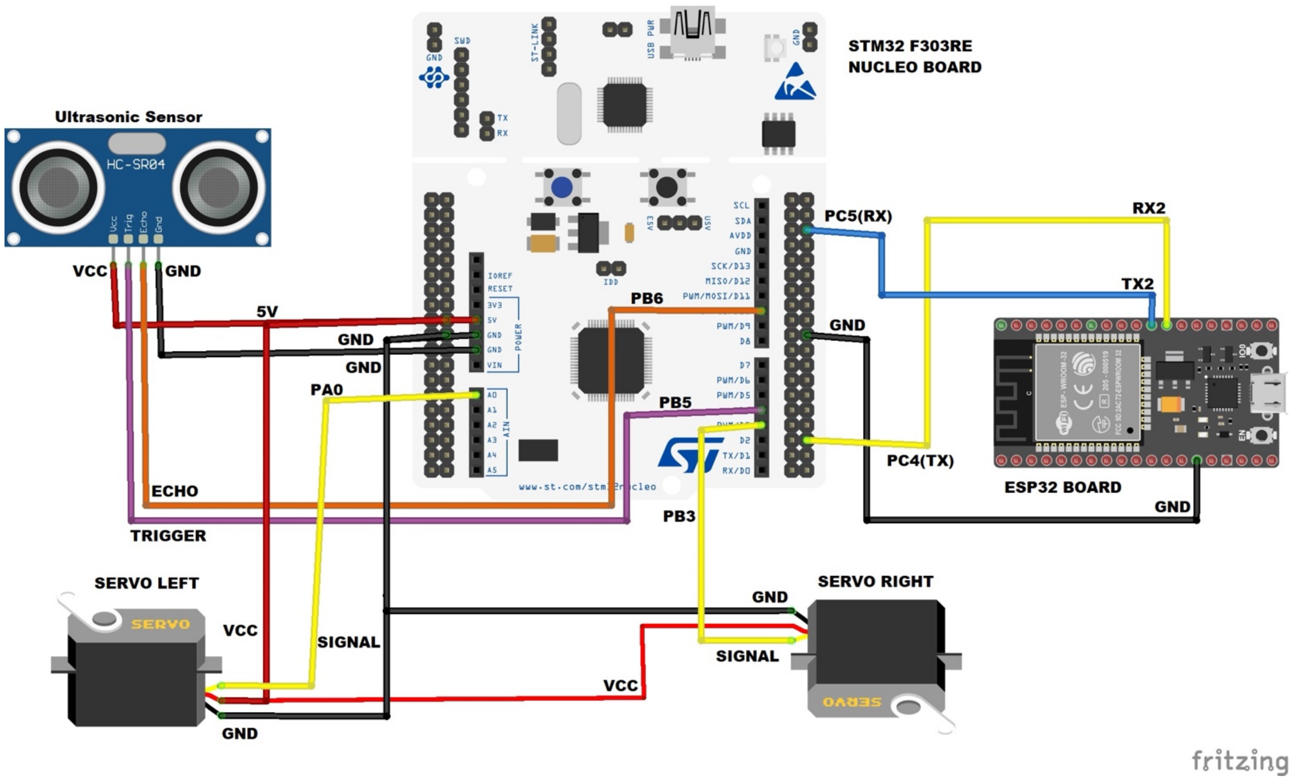


Figure 1 – High-Level graphical design of the robot system

▪ Ultrasonic Sensor

Table 2 – Connections pinout of distance sensor

Ultrasonic sensor			Nucleo STM32 F303RE	
pin	**description**	**connection**	**pin**	**description**
GND	Ground pin	->	GND	Ground
VCC	Voltage pin (input for 5V)	->	5V	5 Volt output
TRIGGER	Expects a 10 μs pulse in order to activate sensor	->	PB5	Alternate function 2: Timer(output): 3 Channel: 2
ECHO	Produces a ultrasonic pulse of 40 kHz	->	PB6	Alternate function 2: Timer(input): 4 Channel: 2 and 1

- **Servo**

Table 3 – Connections pinout of driving motors

Left Servo			Nucleo STM32F303RE	
pin	**description**	**connection**	**pin**	**description**
GND	Ground pin	->	GND	Ground
VCC	Voltage pin (input for 5V)	->	5V	5 Volt output
SIGNAL	Expects PWM	->	PB5	Alternate function 1: Timer(output): 2 Channel: 2
Right Servo			**Nucleo STM32F303RE**	
pin	**description**		**pin**	**description**
GND	Ground pin	->	GND	Ground
VCC	Voltage pin (input for 5V)	->	5V	5 Volt output
SIGNAL	Expects PWM	->	PA0	Alternate function: 2 Timer(output): 2 Channel: 1

To begin with, the required basic functionally was taken in account and the necessary ES assignments were taken out, also an extra library was created in order to attain of the functionalities.

Given so, the required basic functionality asks for the following of the robot:

1. Move forwards in a straight line until 10 cm are reached
2. Change direction to any of the following left, right and straight ahead
3. Change direction randomly when the obstacle is encountered

For achieving these the following ES assignments were needed: Timers – Input, Timers – Output, Interrupts. Also, in order to make the robot move randomly a randomiser library was created to this scope.

Following this preparatory work, the 2-servo motors and the Ultrasonic Sensor were set, configured, and implemented. The former represents the wheels, and the latter measures the distance into order to give some sense of the environment around the robot. As such, the servo needed a timer output to send PWM signals to the channel The PWM signal controls the speed and the direction of the servo in order to fit a working robot,

The Ultrasonic Sensor on the other hand requires both output and input timers. The output timer generates the 10μs pulse on trigger. This will then trigger an ultrasonic wave of 40kHz audio. The input conducts timer capture and returns the captured wave signal on echo. The result is a square wave proportional to the distance of the nearest object.

▪ Conclusion

To conclude, after researching, bringing together, mending, and creating a robot a better understanding of the practical applications of the ES assignments was achieved. This shows how theoretical knowledge can have a practical application and gave insight into the practice of linking together separate working smaller “modules” that together contributes to the grand result.

▪ Individual Reflection

Andre:

For this phase I implemented the Servo motor code and a library that provides useful functions that can be used throughout the semester. I also helped with connecting the hardware to work together and with testing the behaviour of the robot car. For this phase, it was not hard implementing the servo making everything together gave us more trouble than expected. In the end we made the robot car work as expected.

Sean:

For this phase I worked on the robot project library and contributed to Andre's library by fixing bugs and expanding on it. I was hoping to expand the robot library more but due to time constraints, I realized it would not be worth it. Furthermore, I contributed by fine-tuning the servos and developing algorithms for the servo function.

Victor:

While my personal contribution for phase was not in the essential parts, I made sure to create, organise and format the structure of our project report, so that anybody could just start writing. Besides this I also created a randomiser library which is platform independent. This was done to make the robot chose a to arbitrary direction when need be after an obstacle had been met.

❖Phase Two – Feedback Control System

▪ Introduction

In the second phase of the project, more functional requirements are to be added to create a better/smoother braking system according to feedback control system. Using phase one implementations, the robot car will start to apply an FCS PID to brake smoothly when its distance is double of the "obstacle distance".

▪ Procedure

The procedure of this phase will be an extension of the first phase with added functions for braking. The function for the PID were already done previously and only need to fill in the values. We also used Matlab to simulate and tune the robot to avoid overshooting. During simulating, we found out that the values from auto tuning are different every time, so we picked one with less overshoot. The values that we got is:

KP = 1.0001

KI = 0.1825

KD = 0.1887

With the tips coming from our teacher, we started to map the servo values using a mapping function provided by Arduino website and added the result value from the PID. To start experimenting robot behaviour, a P controller function was firstly used which only sends the difference of the distance and set point. We also used the formula for the PID with the value which only require KP for the P controller. P controller is not enough to achieve our goal for this phase but it's a start which we can build onto.

The next step is making the PD controller which require for us to extend the P controller and adding KD in the formula. With the PD controller, we had more early success than we expected and were able to break smoothly. After seeing required behaviour from the robot, we decided and started experimenting with the PID controller. The PID requires KP, KI and KD in the formula to achieve perfect brake. After implementing the PID, there were some errors that would not make the robot move from the start. After debugging and testing every value for the formula and even restructuring the formula, we concluded that it does not meet our expectation. In the end we happen to use only the PD controller as it gave us success gives us a better chance for demonstrating the robot car.

▪ Conclusion

To conclude phase 2, we have achieved the functional requirement in a way, but it requires more extensive research and knowledge on implementing the PID controller. We are satisfied with the results with what we have.

▪ Individual Reflection

Andre:

I handled the implementation, testing and debugging the PID controller. Using the guidelines given by our teacher, the steps into creating the PID controller made it easier to avoid breaking the system and gives us more chance on getting the robot to work how it supposed to. Sadly, the end product happened to be a PD controller and hopefully this one will give us more chance at the upcoming demonstration.

Sean:

For this phase I fixed the issue we had with timers input setup. We received feedback from Rene to change timers input setup to not use ISR (Interrupt Service Routine), because the ISR interrupts the core of the Nucleo for a few milliseconds, which causes the capture compare register’s captured values to be off by a few milliseconds. This caused the Ultrasonic sensor readings to be off by a few centimetres. So, to avoid this 1 timer had to use 2 channels, instead of each timer using 1 channel.

Victor:

For this phase I was responsible of researching into PID controllers and their parameters. As such I found that our constants can be improved, and I did so by recalculating them using the Trapezoidal rule.

❖Phase Three –Robot Management

▪ Introduction

In this phase, we were to extend the phase 2 and implement more features that require knowledge from other subjects. The required knowledge for this phase is UART, Node-Red and ESP32, or in other words, the COM subject. With this knowledge, we are required to create a dashboard that would show the distance while also a button to change the current set point of the robot car.

▪ Procedure

Configuring communication between the STM32F303RE, ESP32 and Node-Red can be split into two main parts, each playing an essential role into the driving and flexibility of the robot, in total having three main nodes of the system.

- **STM32F303RE ↔ ESP32**

Communication from the STM32 MCU is done via UART1, as the UART2 is being used for the servo motors. Therefore UART1 is setup in the. ioc configuration file of the STM32's IDE project. Here is where such a configuration breaks the system, as it requires code generation of new settings, which also include changes for the system's clock configuration. These new configurations break the system as they do not allow the robot's servo motors to spin anymore. This is fixed by substituting the new generated code for the system's clock with the old one. Therefore, the problem is fixed and the system runs normally. On another note, the actual transmitting is don with the "HAL_UART_Transmit" function and the receiving is being done with the "HAL_UART_Rceive" function. Both of them use polling method and therefore block the system for a specified period of time given as a parameter. For receiving a variable is used as it suffices the use-case of out robot and for transmitting the text is inserted into a char array by the sprint(), later sent by the appropriate function.

- **ESP32 ↔ Node-Red**

Communication from the ESP32 is done in a simpler way as it represents the intermediary between the low-level STM32 which does the hardware abstraction and the high-level Node-Red that oversees the controls and statistic of the system. Communication is done via MQTT and the mosquito broker through two topics. One, "group3robot/setpoint" is subscribed to and use for receiving any new setpoint for the PID controller indicated by the user, and the other one "group3robot/distance" is used to send the data coming from the

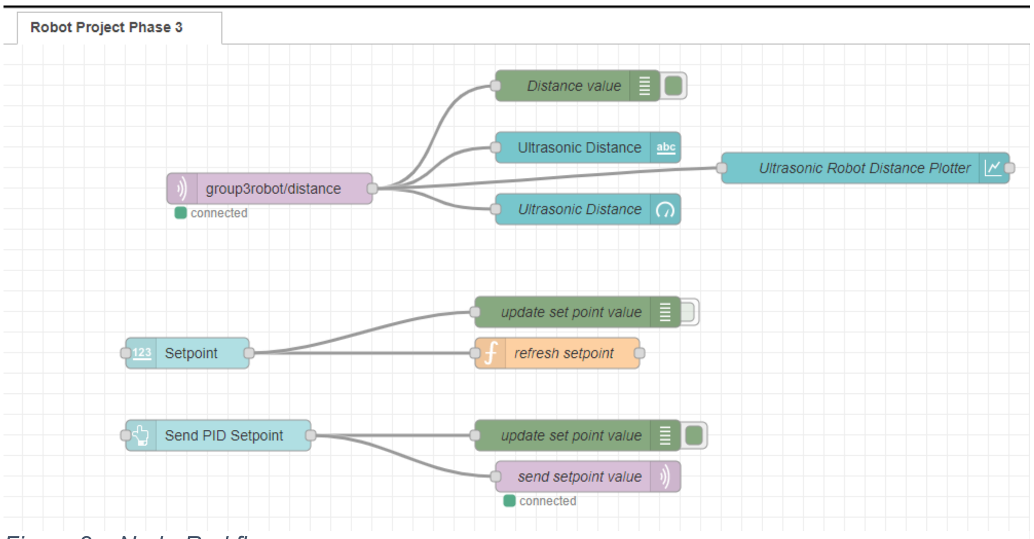
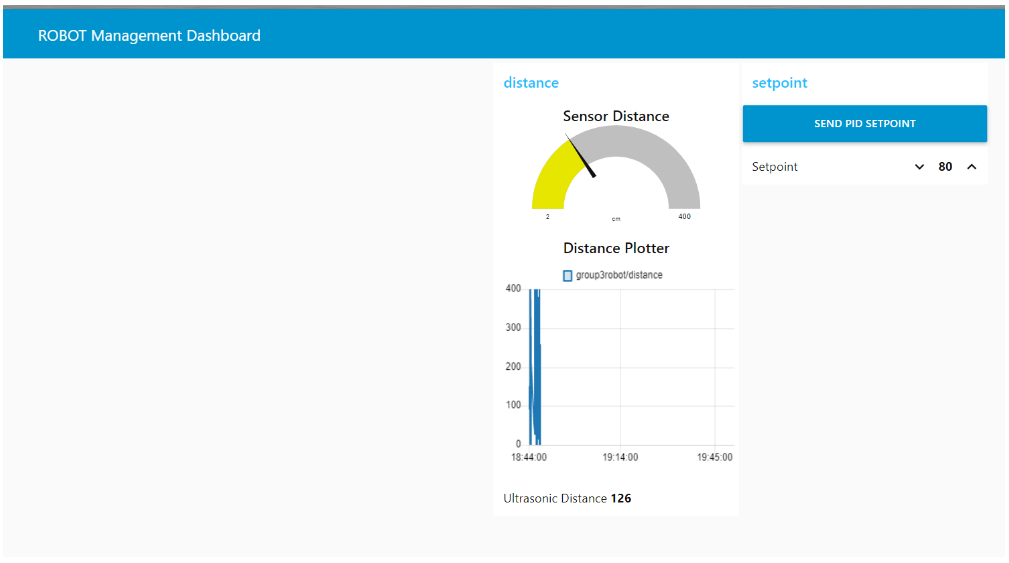


Figure 2 – Node-Red flow

STM32 further to Node-Red to be processed and displayed on the gauge and graph.



Setpoint 80

Ultrasonic Distance **126**

Figure 3 – Node-Red dashboard

- **Node-Red**

At last, on the Node-Red on the visual dashboard, depending on the user's choice a button can be pressed and a new PID's setpoint value be sent to the STM, the incoming data representing the distance can be displayed on the gauge and plotted on the graph. These can be seen in Figures 2 and 3.

Wiring diagram

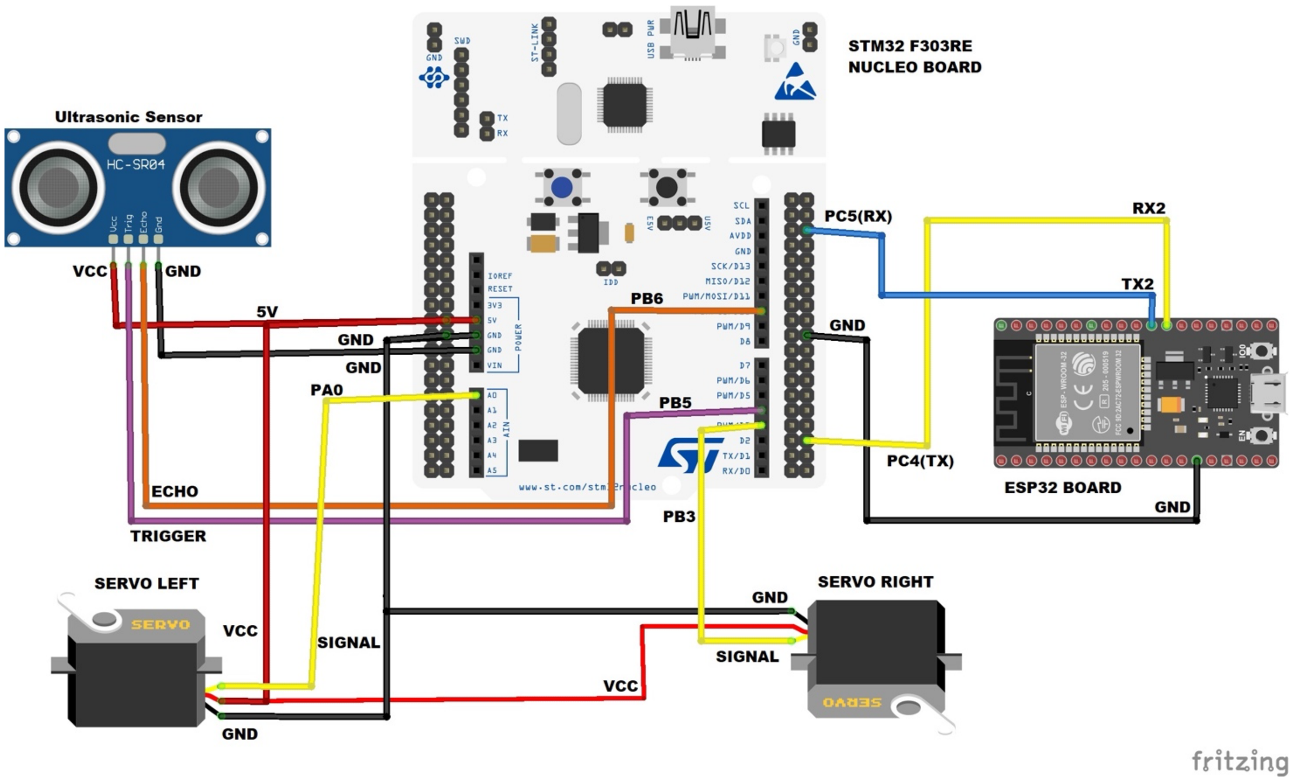


Figure 4 – Graphical wiring diagram of the robot

- **Ultrasonic Sensor**

Table 4 - Connections pinout of distance sensor

ESP32			Nucleo STM32 F303RE	
pin	**description**	**connection**	**pin**	**description**
GND	Ground pin	->	GND	Common ground
TX2	USART2 transmitter pin (GPIO_17)	->	PC4	USART1 RX
RX2	USART2 Receiver pin (GPIO_16)	->	PC5	USART1 TX

Last, Figure 5 shows an abstract block diagram of the main connections of the robot's system which together they make it possible to be communicate and control it.

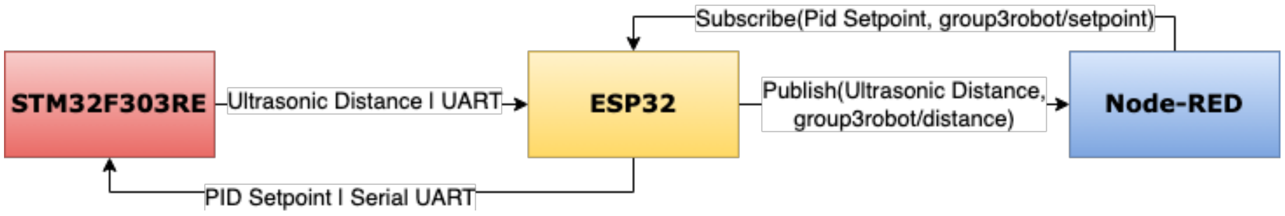

Subscribe(Pid Setpoint, group3robot/setpoint)
STM32F303RE
Ultrasonic Distance I UART
ESP32
Publish(Ultrasonic Distance, group3robot/distance)
Node-RED
PID Setpoint I Serial UART

Figure 5 – Abstract connections and communications diagram of the robot system

Conclusion

To conclude, by applying different communication approaches used for the same protocol, together with a visual programming tool able to visually interface with the robot and hardware to make it wireless, a wirelessly controlled robot was able to be build. This system highlights the ingenuity of piecing together different tools in order to simplify interfacing with complex hardware and at the same time, to alleviate workload from the main hardware driver.

Individual Reflection

- Andre:

For phase 3, I worked on the Node-Red Dashboard and ESP32 work together and make a connection to the STM32 board through UART. I also did the functional requirement with changing the setpoint in the dashboard to the STM32 board. This phase was the hardest because I began on working with every functionality but made it somewhat halfway with transmitting the distance. We had to debug the code and restart from phase 2 code to find out the problem last minute.

- Sean:

In this phase a lot of troubleshooting had to be done because we ran into some errors during integration with ESP32, I contributed on fixing some of the software and hardware bugs we ran into, and I also created some wiring diagrams and tables with pin descriptions, and their settings.

- Victor Covalciuc

For this phase, I helped in transmitting data from the ESP32 to the STM32F303RE upon being received from the Node-Red dashboard by gaining insight into the different way string are sent, converted, and interpreted on both sides. Further, I implemented on the Node-Red side the Distancing statistics, those being the gauge and the graph. Also, I help with transmitting data from STM32F303RE to ESP32. At last, I refactored the code where need be and eliminated the magic numbers. Moreover, I also discovered faster way of transmitting and receiving data, although unfortunately due to strange IDE configuration error those were not included in the product as they stopped, they entire robot from working.

❖Conclusions per entire project

As the first phase was done, and the robot was being put together, some difficulties were met with the Ultrasonic Sensor where it would not work properly. After looking back into the code, this was fixed and then servo motor was fine-tuned motor so that it had a prober and symmetrical working order. Moreover, great knowledge was gained into the inner working of linking together different applied features of the STM32 MCU into one practical application, a working robot with decision behaviour.

As the second phase was done, the robot’s movement was improved as now it facilitated from a better control of its movement due to the addition of a PD controller, which added the ability of predicting and reacting in a more natural way to obstacles in front.

As the third phase was done, technical data of the robot could be seen on a dashboard built with node-red and connected to the robot through an ESP32, that communicated with the help of a MQTT Broker to the node-red dashboard.

❖References

1. Robot Project - Robot Swarm Project_v2
   #001 -- https://fhict.instructure.com/courses/12566/pages/robot-swarm?module_item_id=840233
   #002 -- https://git.fhict.nl/technology/t-sem3-cb/-/blob/master/projects/Robot%20Project/Robot%20Swarm%20Project.docx

2. Parallax Feedback 360° High-Speed Servo - 900-00360-Feedback-360-HS-Servo-v1.1
   #001 -- https://git.fhict.nl/technology/t-sem3-cb/-/blob/master/projects/Robot%20Project/Toolbox/900-00360-Feedback-360-HS-Servo-v1.1.pdf

3. Ultrasonic Sensor - HCSR04_user_manual_1
   #001 -- https://git.fhict.nl/technology/t-sem3-cb/-/blob/master/projects/Robot%20Project/Toolbox/HCSR04_user_manual_1.pdf

4. Ultrasonic Sensor - hc-sr04_ultrasonic_module_user_guidejohn
   #001 -- https://git.fhict.nl/technology/t-sem3-cb/-/blob/master/projects/Robot%20Project/Toolbox/hc-sr04_ultrasonic_module_user_guidejohn.pdf

5. Map - math_map_function_arduino
   #001 -- https://www.arduino.cc/reference/en/language/functions/math/map/

6. STM32F303RE UART
   #001 -- https://controllerstech.com/uart-transmit-in-stm32/
   #002 -- https://deepbluembedded.com/stm32-debugging-with-uart-serial-print/

7. ESP32 UART Serial
   #001 -- https://github.com/G6EJD/ESP32-Using-Hardware-Serial-Ports/blob/master/ESP32_Using_Serial2.ino

8. Node-Red dashboard
   #001 -- https://flows.nodered.org/node/node-red-dashboard