Robot swarm project

A project about making an autonomous robot

Fontys University of Applied Sciences

Stefan Teeuwen, Kevin Vissenberg and Danail Georgiev

Table of Contents

[Introduction 2](#_Toc73954819)

[Requirements 2](#_Toc73954820)

[Components 2](#_Toc73954821)

[Research 3](#_Toc73954822)

[Findings 3](#_Toc73954823)

[Implementation 3](#_Toc73954824)

[Servo motors implementation 3](#_Toc73954825)

[Ultrasonic sound implementation 3](#_Toc73954826)

[STM32Nucleo-f303re 3](#_Toc73954827)

[ESP32 as sender and receiver 3](#_Toc73954828)

[C# application 3](#_Toc73954829)

[Class diagram 4](#_Toc73954830)

[Caveats 4](#_Toc73954831)

[Testing 4](#_Toc73954832)

[Recommendations 5](#_Toc73954833)

[Role division 5](#_Toc73954834)

[Challenges 5](#_Toc73954835)

[Conclusions 5](#_Toc73954836)

# Introduction

The robot follows a pre-programmed route without human help. This document explains how we are going to achieve this.

# Requirements

The basic requirements are as follows:

* Robot will move straight forward until it reaches the end of the allowed field of the movement.
* Robot will move straight forward until it encounters an obstacle at 10 centimetres distance.
* Robot will change its movement direction randomly if it encounters an obstacle
* The robot will send its distance of the nearest obstacles in centimetres to a dashboard
* The robot will send its current speed to the dashboard

There are also optional functionalities which can be implemented when all basic requirements are fulfilled:

* Using multithreading to run the control of the ultrasonic sensor and motors in different threads
* Smoothly decrease the robot’s speed when an obstacle is near
* Keep the initial speed when the robot is more than 10 centimetres from an obstacle
* Using odometry to estimate change in position of the robot
* Enhancing the dashboard with additional data
* Integrate information of multiple robots in one dashboard

# Components

This project consists of multiple sensors, which are:

The basic requirements can be programmed with the following sensors:

* HC-SR04 ultrasonic sensor to measure distance to other objects
* 360 degree servo motors for spinning the wheels

The optional functionalities can be made with the following sensors:

* Accelerometer to measure proper acceleration
* 360 degree servo motors for spinning the wheels
* Gyroscope for measuring or maintaining orientation and angular velocity

The microcontrollers used for this project are the STM32Nucleo-f303re and ESP32. The Nucleo acts as the robot and the ESP32 as the wireless transmitter to the C# application.

# Research

The project group will use the teacher’s explanation and online research as sources. These online resources consists mostly of datasheets and websites of the manufactures about the different sensors.

# Findings

The main part of the project consist of coding the wheels. Our findings are that the wheels are not difficult to program. In addition, programming the ultrasonic sensor is quite simple. However, we found that using the STM32 Cube IDE was difficult to use, as it has a lot of buttons and functionalities. Although this is nice for advanced developers, we found that these functionalities can cause confusion.

# Implementation

## Servo motors implementation

The servo motor has a feedback pin, which was used to measure the speed of the robot. The feedback pin returns a PWM signal whose duty cycle corresponds to a wheel angle. The servo’s are also used in combination with the ultrasonic sensor to change to a random direction when an obstacle is near. Afterwards 2 PID controllers were implemented to control the speed using the measurement of the servo output (1 per wheel). The way rpm was calculated was by measuring the PWM output of the servo which is the position of the wheel and measuring it again after 100ms. We subtract the former from the latter and multiply by 10. This way we have units per second. Multiply by 60 for units per minutes and divide by 360 so that we have rpm. Then map this value accordingly and pass it as input of the PID controller. Then take the output of the PID controller and use it as input for the wheel duty cycle.

## Ultrasonic sound implementation

The ultrasonic sound measures on its own. It only requires a timed measuring pulse, but returns a wave which is easy to parse. By measuring the time the line has been kept HIGH by the sensor we can determine the distance to an object. The sensor is polled at an interval and the value is compared to a threshold value. The wheels are programmed to make a turn to a random direction when the distance becomes lower than the threshold.

## STM32Nucleo-f303re

The Nucleo serves as the robot. It has the servo motors and ultrasonic sound included.

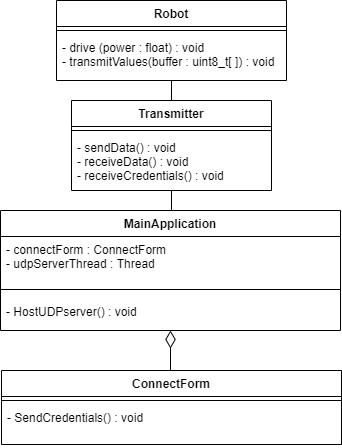
## ESP32 as sender and receiver

The ESP32 receives the credentials from the C# application. Then it connects to Wi-Fi and sends the received data from the robot (Nucleo) to the C# application using UDP. The protocol for this is ‘<:SSID:Password:HostID:port>’, where the ‘<’ and ‘>’ are used as beginning and ending characters. The strings are separated using ‘:’.

## C# application

The C# application connects to the Wi-Fi and opens a server at which the ESP32 can send packets to. It displays the values of the current speed and distance to nearest obstacle.

## Class diagram



*Picture 1: picture of class diagram*

# Caveats

The main caveat is the power supply, as the robot will stop working once the power supply is empty. Another caveat is that the robot doesn’t know where to go once an obstacle is neared. This can cause the robot to drive to the same obstacle multiple times over an extended period of time.

# Testing

Each individual component is tested before assembling. This was done by uploading and running code for each specific component and checking if it would output correct values. Then, all parts were assembled and code was combined and ran in order to test if the robot would output the correct behaviour as specified by the requirements.

# Recommendations

## For students

The easiest approach to this project is to program all components individually and then to combine everything. The robot can be powered by an external power supply.

## For teachers

The project group’s advice for this project is to not overcomplicate things. Especially coding a PID controller can be difficult on an STM32NUCLEO-F303RE. The project group would recommend to do a workshop at the end of the project to demonstrate how to make a PID controller for the robot. This way everyone can understand and learn how to use one. It would then be fun to let all robot’s loose in one room after the workshop.

# Role division

Everyone has their part in the project. Kevin worked mostly on the wheels, Danail worked on the ultrasonic sensor and Stefan worked on communication between the STM32NUCLEO-F303RE and ESP32. Stefan was also the project leader and made the planning and tasks in a scrum board on GitLab.

Graphical user interface, text, application

Description automatically generated

*Picture 2: the planning in a scrum board*

# Challenges

The biggest challenge was working together during times of COVID19, which until this day is persevering and limiting the way students can work together. Only communicating over the internet limits the possibilities of working together, as you can’t as easily reach out to fellow students for code reviews and discussing the current state of the project. However, we were able to communicate and work together using platforms such as Microsoft Teams, Discord, GitLab and Google Docs.

# Conclusions

We would say that our product meets at least all basic requirements and expectations. It took quite some time and effort to reach our desired product.