Robot Swarm - Design Document

YVON KOKKE - \_\_\_\_\_\_\_\_\_\_

Marco Cardillo - 4076923

Kristiyan Manolov - 3605310

Toma Krastanov - 3774678

Contents

[Problem 2](#_Toc88504281)

[Context Diagram 2](#_Toc88504282)

[Control Signal – TIMER 2 3](#_Toc88504283)

[Feedback Signal – TIMER 3 & 4 4](#_Toc88504284)

[Hardware 5](#_Toc88504285)

[360 Parallax Servo 5](#_Toc88504286)

[Ultrasonic sensors 5](#_Toc88504287)

[ESP32 5](#_Toc88504288)

[Performance consideration 5](#_Toc88504289)

[State Diagrams 6](#_Toc88504290)

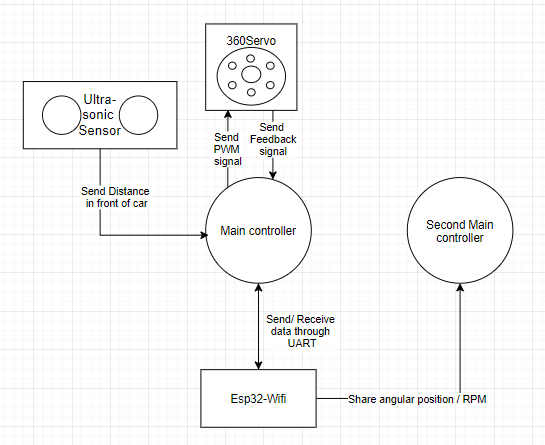
[Proof of Concepts 7](#_Toc88504291)

Design

## Problem

The use of a robot being able to move freely along a path without crashing into many anything is being used widely in our everyday lives. For example, the rumba bot is a small robot that records the path of your house and cleans it automatically. Implementing these concepts on small robot could be the start of a big project.

# Context Diagram

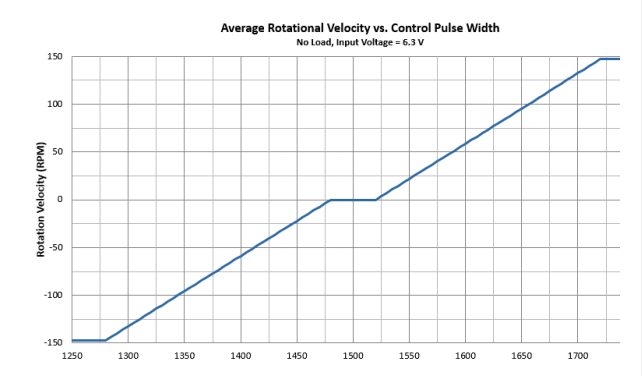


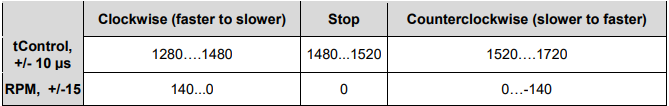
# Control Signal – TIMER 2

The Servos have a control signal sent to it that lasts 20 milliseconds. The PWM of that signal will allow It to rotate.

Timer 2 was used in this case for the output of these two servos. A prescaler of 16-1 was used to achieve results in milliseconds and the ARR value is of 50Hz ((16MHz/PRESCALER)/50), facilitating our representation in the code in comparison to the one in the datasheet.

Datasheet:





# Feedback Signal – TIMER 3 & 4

The servo sends a feedback signal at a rate of 910 MHz, this is approximately 1.1 milliseconds. We arrange Channels 1 and 2 from Timer 3/4 to capture the rising edge and falling edge. Having this we can measure the period cycle and Duty cycle. According to section ***21.3.6 PWM input mode of the Reference manual***.

The Left Servo is connected to PA6 and the Right one is connected to PA11.

We then followed the ***Parallax Feedback 360 High-Speed Servo datasheet*** and used the code from the feedback360 function. This needs to be repeated at a frequency of 50Hz, thus we use a channel on the Timer2 (Timer already setup up to 50Hz for control). With described code from the datasheet, we can obtain the Theta angle, the angle towards the wheel is rotating.

Having the theta angle, we can compare the previous theta angle to the current one to achieve Angular Velocity. As we have the function running in a 50Hz timer, the timestamp between the previous angle and the current angle is 20 milliseconds.

The data sheet mentions to not to read the angle when the time cycle is not valid. The time cycle is not valid while the total cycle of the PWM signal is not between 1000 and 1200 milliseconds. We only read the length of the PWM signal on high and not on low. Our total length of the signal is a steady 1070 milliseconds. Thus, having some small spikes on the RPM calculation. It will spike up for some intervals but most of the time has a steady value.

The calculations used for the formula is the following:

1. We find the linear velocity which is the difference between angles divided by time.

Linear velocity = ΔTheta Angle / Time = ΔTheta Angle / 20 MS

1. Having the Linear velocity, we can find the Velocity by multiplying the Linear velocity by the radius of the wheel.

Velocity = Linear Velocity \* Radius

# Hardware

## 360 Parallax Servo

Purpose: These will act as the wheels of the cars.

Reason: The micro controller will send a pwm signal to the servos in order to control the speed/direction of the car.

## Ultrasonic sensors

Purpose: To measure the distance of objects in front of the car.

Reason: The car will stop moving in order to avoid obstacles in front of it.

## ESP32

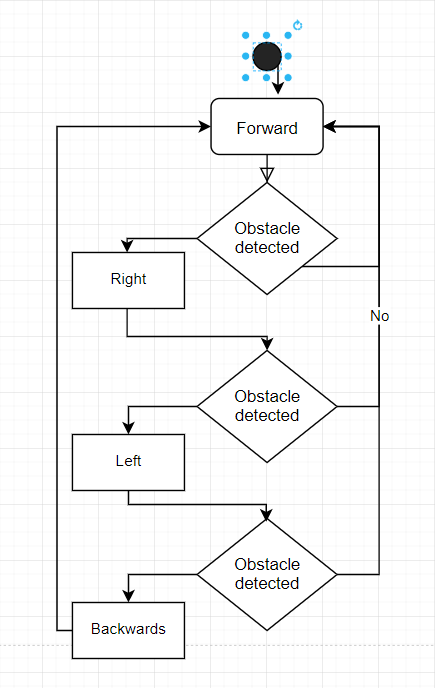
Purpose: To communicate with other cars or control the car through an external controller

Reason: We could have more than one car synchronized following each other which could act like a drone show. Moreover, we could connect a joystick to another esp32 and setup a network so that the car could be controlled externally.

# Performance consideration

In this project we will have a small car who is supposed to make decisions on its own. For example, its main function is to drive forward. The robot should avoid obstacles that it encounters and not harm itself by ramming into them. It will be able to communicate with other robots or microcontrollers. The robot will be able to control its speed and keep track of its position.

# State Diagrams



# Proof of Concepts

Phase 1 – Distancing:

<https://youtu.be/26_po33i9LQ>

Phase 3 – Feedback Control Systems:

<https://youtu.be/vLky2n3yTYU>

Phase 4 – Robot management:

<https://youtu.be/DAWVXsqnBws>

Feedback Calculations – Angular Velocity

<https://www.omnicalculator.com/physics/angular-velocity>