Embedded Systems: Robot Project

Johnson Domacasse(4471709)
Harm Nieuwland(4926749)
Date:
Teacher: Suzana Andova

List of Figures

List of Tables

Document history

Version	Date	Status	Author	Description
0.1	2023-11-06	Draft	Johnson	Document creation
0.2	2023-14-11	Draft	Johnson	Phase 1 documentation

Terms, Abbreviations

Abbreviation	Description
IR sensor	Infrared red sensor
US sensor	Ultrasonic sensor
MCU	Micro controller unit
ARR	Auto-reload register
PSC	Pre scaler
FCS	Feedback Control Systems

1. Introduction

The robot platooning project is one of three projects that were assigned to each project group to work on. The primary goal is to make a robot that can smartly navigate a course of black tape on the floor and in later parts of the project, apply FCS for a smarter robot.

2. System Description

This section of the report is dedicated to give a brief description of what our team did to accomplish the robot project. We give a brief description of what the project entails and how we solved it.

The robot project has 3 main components that function together to form one entire system. These components are the ultrasonic sensor, the 2 Servo motors, and the infrared sensors. Each one of these components code, were compiled and analyzed to then be combined into one final machine. Below is a small description of each sensor and their purpose:

1. ***Ultrasonic sensor:*** The ultrasonic sensor acts as our feedback "giver" later on in the project when we implement feedback control systems as well. In the first design of the robot, it will simply act as a stop mechanism for the robot. Additional details will be given in the implementation phase 1 section.
2. ***Infrared sensors:*** The infrared sensors act as the steering wheel for the robot. Based on if the sensor is detecting a black line both straight or curved, it will alter send this information back to the program so the robot can steer accordingly based on the results.
3. ***Servo motors:*** The servo motors act as the wheels for the robot. It will start driving at a specific speed and based on the information it gets from both the ultrasonic – and the infrared sensors. More information on their calibration can be found in the implementation section.

Each sensor is connected on specific pins that are exclusive to the code that will be provided along wit this document. See table 1 for PIN configuration of the robot project. Be advised, some pins have the wire color attached to them because the wires are attached to the robot itself. **[5]**

	Actuator	Pin Name	Wire Color
1.	Ultrasonic Trigger	PB5	-
2.	Ultrasonic Echo	PB6	-
3.	Right-side Servo(clockwise rotation)	PA0	-
4.	Left-side Servo(counter clockwise rotation)	PA1	-
5.	IR OUT1 (leftmost)	PA3	White
6.	IR OUT2	PA2	Orange
7.	IR OUT3	PA10	Green
8.	IR OUT4	PB3	Orange
9.	IR OUT5 (rightmost)	PB4	Green

Table 1. Pin configuration

3. Design Phase 1

This section of the report will be dedicated to how the design of the robot is structured. For this project, we will only be using a state machine to show what is happening through out the robot processes. See figure 1 below for complete state machine.

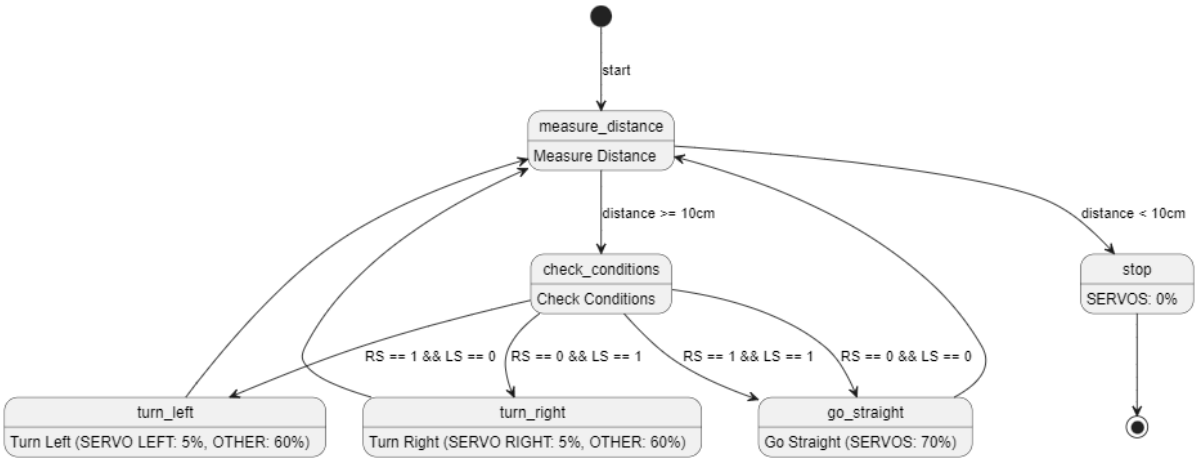

Figure 1: Phase 1 state machine.

4. Implementation Phase 1

This section of the report will be dedicated to explaining the choices made behind certain parts of the robot project implementation. These can include for example which timers were used, or why certain values for certain registers were chosen and more. Note: For the remainder of the robot project, the clock speed of each implementation will be set to 16MHz.

4.1 Servo implementation

For the servo implementation, the knowledge that was gained from the timers output assignment was applied first. Using the alternate function mapping table from the MCU datasheet**[6]** and the knowledge of setting up and using the timers**[4],** the servo implementation was then configured correctly PIN-wise.

The reasoning behind the usage of the values 32 and 45000 for the PSC and the ARR respectively was determined again using the following formula:

$$PWM\ Frequency => \frac{Clock\ Frequency}{Desired\ Frequency} = value(PSC \times ARR)$$

Since our desired frequency was a frequency of 50Hz (for a 20ms period)**[3]**, we end up with a value of 320000. We thought initially that our PSC and ARR values would be 32 and 10000 then but when read with the logic Analyzer, we noticed that these signals were not that of a 20ms period. In the end we manually calibrated it to approximately 20ms by having the ARR value be at 45000. This is all done on one timer with two separate channels.

With that being said the values that are being sent to the servo also has to be changed. Take the clockwise rotation for now. To spin the servo at 100% we need to pulse it with 1280 and 0% would be 1470. Through trial and error and using the analyser, we came to a conclusion that for the servo to spin clockwise at 100%, we would need to pulse it with the value of 2768. So this value we divided by the initial 1280. We get a constant of 2.16217.

Additionally we made another function so we can control the servo by passing the desired percentage value in the function. This is done by implementing the “map” function**[7].**

Finally, since both servos need to be pulsed by different values to act as wheels, they would need to operate both with counter clockwise and clockwise. For that two separate functions were made to control them. Next to this, we have another function so that we can stop the servos, they need to be pulsed with 1500. So both our servos will be post with that equivalent to stop them.

4.2 Ultrasonic implementation

For the ultrasonic, the knowledge that was gained from the timers input assignment was applied first. Using the alternate function mapping table from the MCU datasheet**[6]** and the knowledge of setting up and using the timers**[4]**, the ultrasonic was then configured pin wise.

The timers setup was the same as the previous assignment as well. We slow down the clock to 1MHz so we can get values of 1 microsecond. From there we can easily pulse the ultrasonic with at least 10 microseconds on the trigger pin. The output from the echo pin is then read using another timer set to input mode. From there we have an interrupt handler, constantly reading the rising and falling edges to get the final distance value. For some context, signals on the ultrasonics can be seen in the following image.

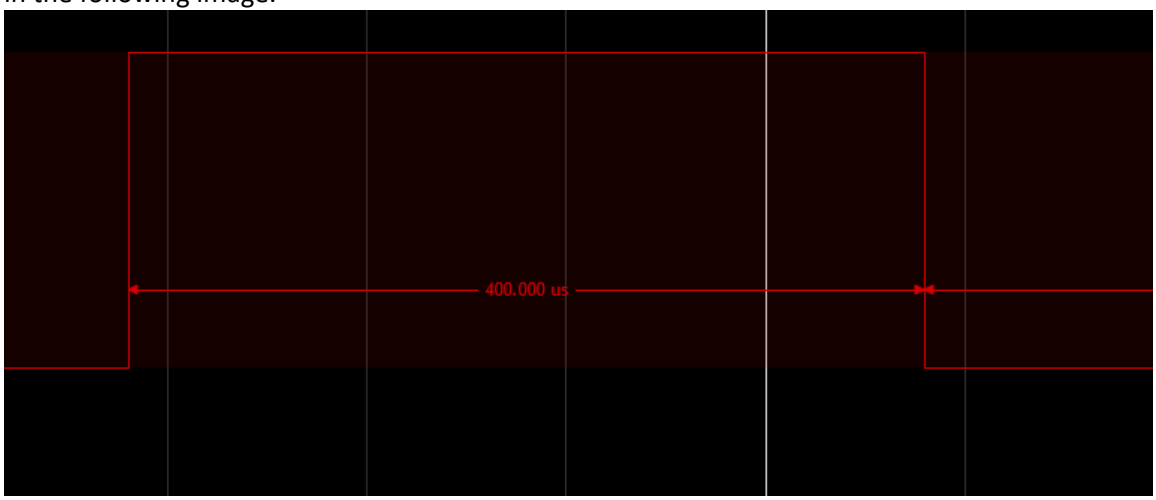

Figure 2: pulse width that is being read on the input pin

The distance is then calculated using the following formula:

$$Distance\ (in\ cm) = \frac{Time\ (microseconds)}{58}$$

4.3 infrared implementation

The infrared code is relatively the easiest one out of all of the sensors to calibrate. The general idea is when the ultrasonic is on a flat service the color sensor would “detect” something. When a black object comes across the sensor would not detect anything anymore. This is because the black object absorbs the infrared color so it cannot be reflected in order to be reached by then sensor. So with that logic, and the fact that we use black tape makes our implementation design easier. Using the knowledge we got from the pushbutton assignment, we can just use the IDR register to read the value of the pin it is connected to. We can then put in in an if-statement to confirm if the sensor isn’t detecting something. In this case it would do something.

4.4 combined implementation

The combined implementation to form the entirety of the robot was done in parts. First we combine the ultrasonic and the servo motor implementations together. You will notice an immediate change in your distance outputs. This can be recalibrated. Now that we have a servo to functions normally, and can stop at an object 10 centimeters away, it was time to work on the steering. According to the state machine design that we have, the code for the line following was implemented using the IR sensor. However there are complications with both reading the sensor and the height of the sensor on the robot.

(to be discussed during the presentation on the 16th of November.

5. Testing Phase 1

This section of the report will be dedicated to explaining the steps that were taken in order to test the completed phase 1 of the robot project.

Scenario 1:
The robot is able to follow a straight line.

Scenario 2:
The robot is able to make slight turns on bends.

Scenario 3:
The robot is able to stop when an object is detected at 10 centimeters or less from the front.

6. Design Phase 2

7. Implementation Phase 2

8. Testing Phase 2

9. Reflections

10. Bibliography

[1] – *Ultrasonic Sensor Datasheet. (n.d.).* https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf

[2] - *Reflective optical sensor with transistor output - Vishay Intertechnology. (n.d.).* https://www.vishay.com/docs/83760/tcrt5000.pdf

[3] - *Parallax feedback 360° high-speed servo (#900-00360). (n.d.).* https://cdn.sparkfun.com/assets/8/5/e/4/e/DS-16047.pdf

[4] – *STMicroelectronics Reference Manual. (n.d.).* https://www.st.com/resource/en/reference_manual/rm0316-stm32f303xbcde-stm32f303x68-stm32f328x8-stm32f358xc-stm32f398xe-advanced-armbased-mcus-stmicroelectronics.pdf

[5] - *UM1724 user manual - stmicroelectronics. (n.d.).* https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf

[6] - *ARM® cortex®-STMicrocontrollers Datasheet. (n.d.).* https://www.st.com/resource/en/datasheet/stm32f303re.pdf

[7] - *Map function. map() - Arduino Reference. (n.d.).* https://www.arduino.cc/reference/en/language/functions/math/map/