Final Lab Project
TECH 3233
Spring 2024

Ver 1.0

Project Summary

Your team (of up to 3 people) will program the robot created for the Volkswagen Robotics Camp back in 2013, to be a collision avoidance robot.

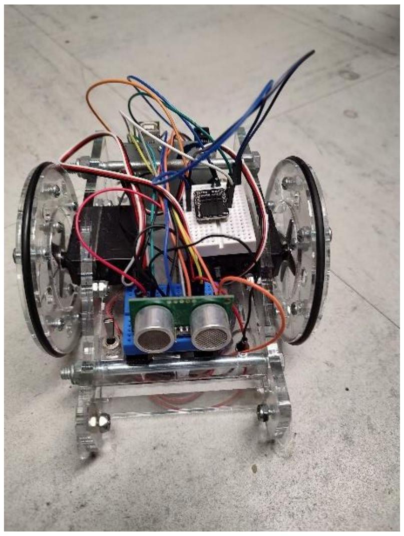


Figure 1- Simple Robot Base

You will use the onboard continuous rotation servo motors to drive the robot (using PWM to drive the robot in the desired direction).

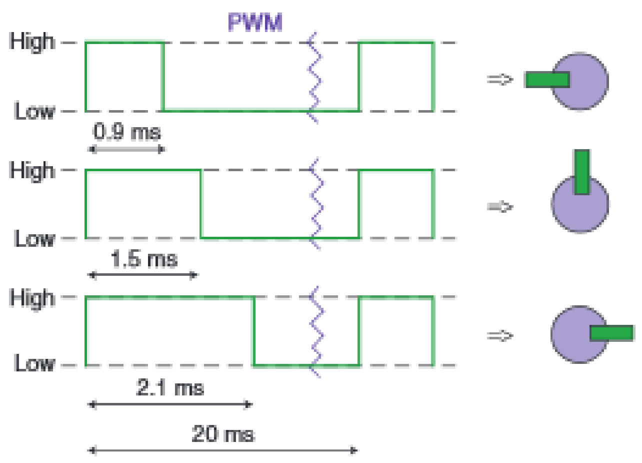
To detect objects in the robot's path, a SRF05 sonar module will be used to measure distance to an object.

Your team will write a program that will have the robot go forward until it encounters an object within 2 ft. If the robot is within that range, it will generate a random number and turn left 30 deg or right 45 deg based on the random number, then continue to go straight once again (repeating the processes indefinitely)

Project Details

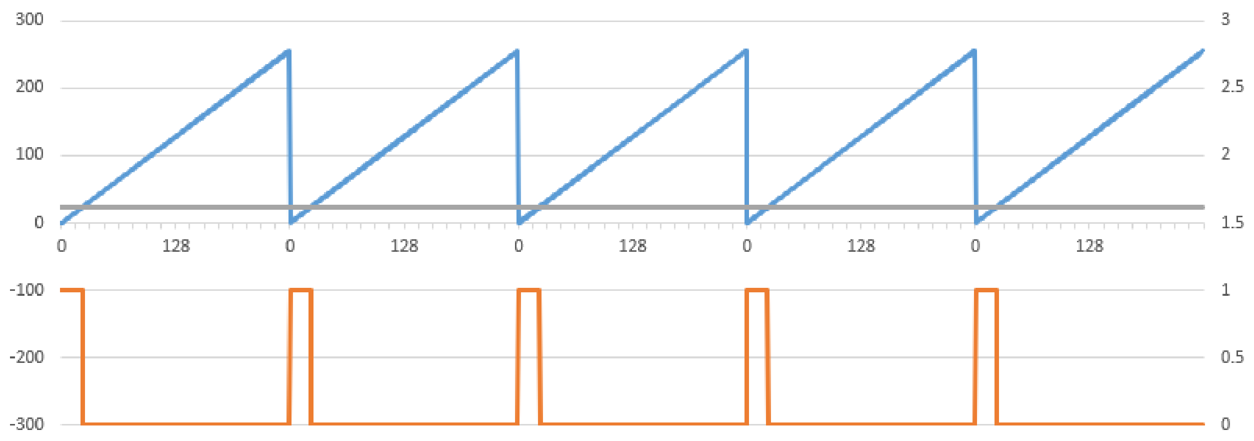
Servo Motors

To control the servomotors, you need to generate the waveforms shown below:

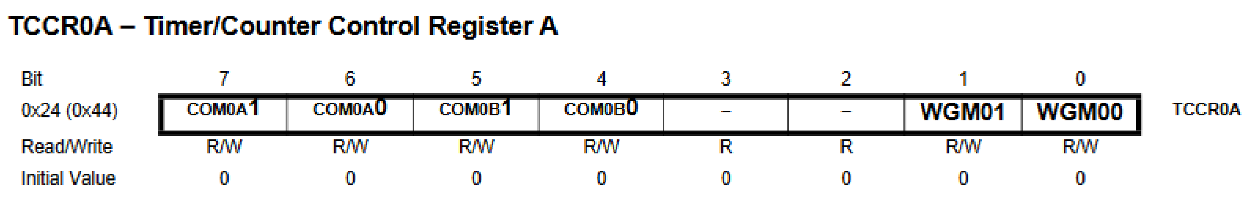


The easiest way to accomplish this is to set up Timer0 in “FAST PMW” mode. Fast PWM will use the entire count (0x00-0xff) to generate the full period of 20mSec. The OCR0x will be set up to generate the high part of the square wave (0.9mSec, 1.5mSec, or 2.1 mSec depending on desired motion).

As shown in the graph below, TCNT0 (shown in blue) is counting from 0 – 255 dec. Every time the TCNT0 counter overflows (255 -> 0) the output pin (shown in orange) goes high. It stays high until TCNT0 is equal to OCR0 (shown in grey) is reached, then the output goes low. (The image is for the 1.5mSec output)



To accomplish this, We will need to set up the following parameters in the TIMER0 control (for both A and B):



TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

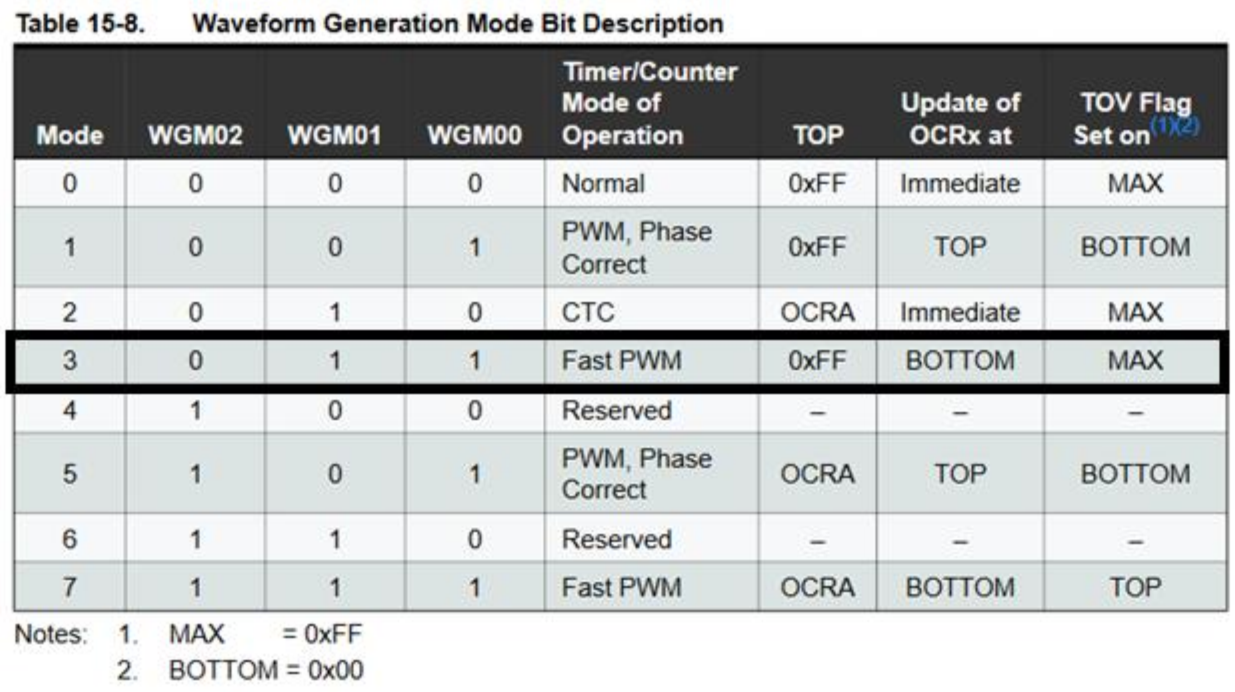


Table 15-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

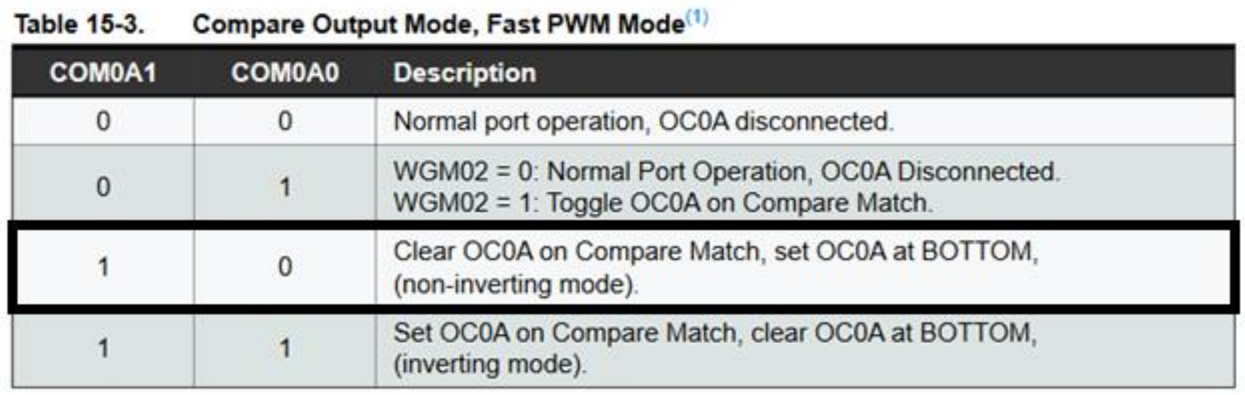
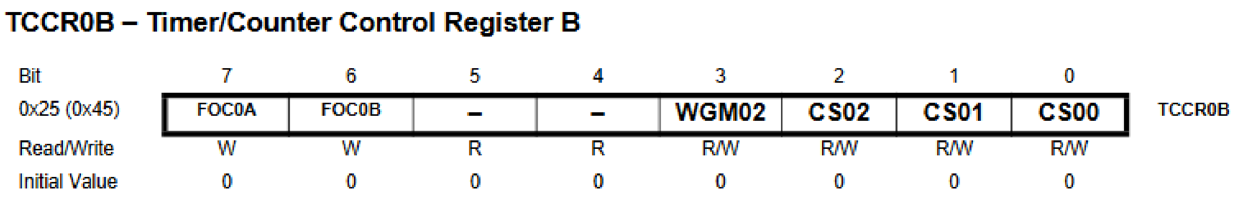


Table 15-3. Compare Output Mode, Fast PWM Mode[1]

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match, set OC0A at BOTTOM, (non-inverting mode).
1	1	Set OC0A on Compare Match, clear OC0A at BOTTOM, (inverting mode).



TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 15-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

You will set the OCR0A and OCR0B to the correct number of counts to drive the motor forward or reverse or to tell the motor to stop.

You will need to determine the divide by clock such that the TOTAL COUNT is as close to 20ms as possible as well as determining the number of counts to place into OCR0A and OCR0B registers to drive the motor forward or reverse or to tell the motor to stop. Also note that the motors are mounted facing opposite directions, so forward for one motor, is reverse for the other.

<u>Sonar Sensor</u>

The SRF05 sonar module uses sound to detect an object by sending out a pulse of sound and listening for the returning echo. You capture the timer value when you send out the pulse, and upon its return. This will give you total time the sound traveled. You can then use the speed of sound and the time of travel to calculate the distance from the object (remembering that the sound travels from the sensor to the object and back).

The Sonar will be connected as follows:

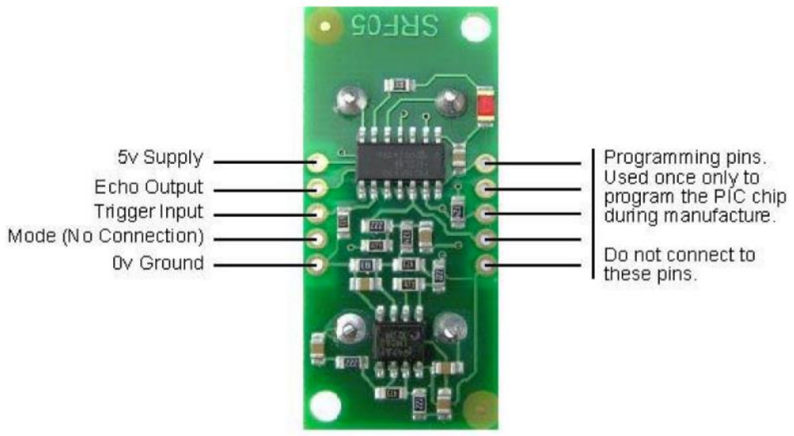

SRF05
5v Supply
Echo Output
Trigger Input
Mode (No Connection)
0v Ground
Programming pins. Used once only to program the PIC chip during manufacture.
Do not connect to these pins.

Figure 2- Sonar Wiring

To operate the sonar, you must set the Trigger Input to high for a minimum of 10uSec. The Sonar then sends out a number of pulses in succession, then puts the echo pin high (Capture first time here) until the echo is heard by the sensor, then the pin goes low (record second time here). Fig 4 shows the timing diagram of the sonar…..

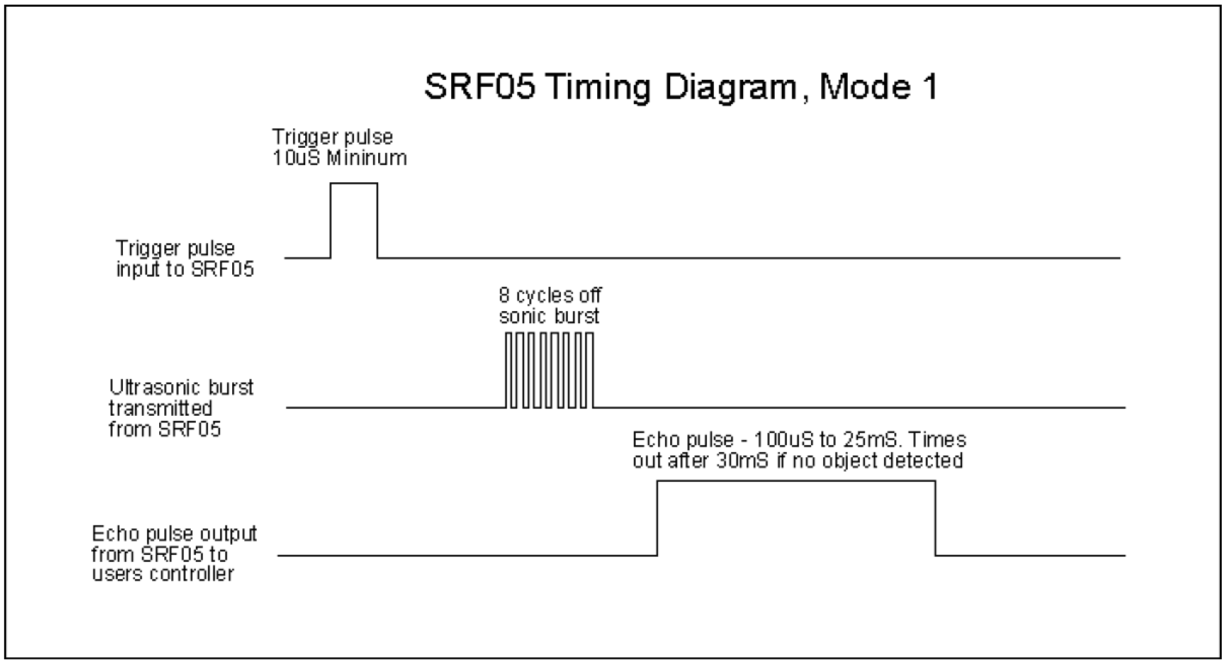

SRF05 Timing Diagram, Mode 1
Trigger pulse 10uS Mininum
Trigger pulse input to SRF05
8 cycles off sonic burst
Ultrasonic burst transmitted from SRF05
Echo pulse - 100uS to 25mS. Times out after 30mS if no object detected
Echo pulse output from SRF05 to users controller

Figure 3- Sonar Timing Diagram

Some notes:

- There is a period when the sonar is not in listening mode (prevents false readings), so there is a minimum distance that can be read.
- You do NOT have to use interrupts for this program. But you do need to be retriggering the sonar while the robot is moving to keep checking for the robot being within 2ft of an object.
- Please put in debugging statements to show the distance in inches from the object for testing purposes.
- Calculations notes:
    - Remember the speed of sound is aprox 1125 ft/sec
    - Since the sound is traveling from the robot, to the object and back again, the distance traveled is 2x as long
    - Remember if you are using TIMER Counts, you have to convert from counts to seconds for the calculation.

<u>Random Number</u>

To generate a random number, C uses the stdlib.h library's rand() function. This function generates a value between 0 and RAND_MAX (32767 for our setup). You can use the formula:

$$rand(\quad) \,\%\, 100 + 1$$

to generate a value between 1 and 100.

<u>IO Wiring</u>

Since multiple students will be using the same robot, the robot will be pre-wired as follows:

Table 1- Robot Wiring

Arduino Pin	Purpose
7	Sonar Trigger
8	Sonar Echo
6	(OC0A) Motor1
5	(OC0B) Motor 2

Assignment

Create a program that will drive the robot forward while checking for an object within 2ft distance. If and object is within range, generate a random number and turn left ~30 deg or right ~45 deg based on the random number, then continue to go straight.

The code MUST BE:

- Commented fully, including a full header block showing IO etc
- Modular: create functions for the sonar, forward, right, left, backwards, stop
- Main should contain setup, and the main loop (that should be an if structure, and a bunch of calls to the various functions).

Submit a spreadsheet with all the calculations for the PMW, the project file (with fully commented code) via online submission. Demo to the instructor and be prepare to answer questions about your code. Your ability to answer questions will be part of the final grade for the project.