Janus - Obstacle Avoidance Detection

Final Report Project

Cesar Galan

December 14, 2016

Nomenclature

γ	Specific heat ratio
ω	Angular velocity
τ	Torque
a	Speed of sound
d	Distance
E	Error
I	Current
P	Power
R	Gas constant
r	Radius
T	Period
t	Time
T_0	Temperature of the atmosphere
U	Velocity
V	Voltage

1 Overview

1.1 Background

Autonomous car will be the future of driving. However, this technology has a long way to go in order to be perfected. An infamous problem for this type of car is not being able to detect stationary objects at high speeds. An scaled down version of this problem was designed, build and tested for this project.

1.2 Project Goal

The final project for this class will be a car detection avoidance, which will measure the distance away from the wall and change its speed based on it. The reference distance away from the wall will be 10 cm, which means that the car will try to stop before 10 cm from the wall. In order, to perform this a proportional controller will be implemented and test it. If need it a derivative control, and/or a integral control, will be added. The code will be written to have all three different types of controller. The car also needs to go at full speed when there are not obstacles in order to simulate high speed cars.

The strategy used in order to solve this problem is to find the distance from the obstacle and will read from an ultrasonic sensor, and the output to the outside world will be a PWM signal to a DC motor driver which will control the DC motor.

2 Method

2.1 Block Diagram

A high level diagram is shown below in Fig. 1, which represent the idea of how this problem will be solve. First, the distance will be determined from the ultrasonic sensor. Then inside the microcontroller it would compute a control algorithm which will determinate the output power that it needs. Once the power is known it will be send to both the LCD screen and the motor driver which will control the speed of the DC motor. In the LCD screen it would show both the distance away from the wall or obstacle and the power that it is being send to the DC motor driver. This is done in order to have a visual output as to what Janus is seeing.

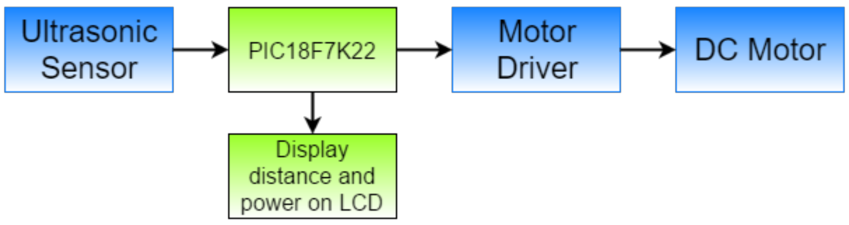


Figure 1: Functional block diagram of the project

2.2 Parts

A table with all the parts number and the cost of each is shown in table below, Table 1. The DC motor was used from Trudy's workshop, the acrylic was found in the laser cutting room, this is why the cost is $0.

Table 1: Part Numbers and their cost

Item	Part Number	Power requirements	Interface type	Cost [$]	Quantity	Total [$]
DC motor[1]	Pololu #1162 29:1 Metal Gearmotor	1.5 -19.8 W	Cable Banana plug	0	1	0
Ultrasonic sensor[2]	HC-SR04	0.01 W	Pin	1.75	1	1.75
DC motor driver [3]	DRV 8838 Single brushed	11 V	Cable or Pin	3.49	1	3.49
Pillow block [4] bearing	8600N1	56000 RPM	axel	16.28	2	32.56
Phone Charger	-	-	-	0	1	0
9V Battery	-	-	-	0	1	0
Acrylic	-	-	-	0	1	0
Gears	-	-	-	0	2	0
Wheels/ Axel	-	-	-	2.5	1	2.5
					Total Cost	40.3

2.3 Circuit

The wiring for this project was rather simple, as display in Fig. 2, first a phone charger was used to power the PIC board. Then the ultrasonic sensor was power by the 5 Volts from the board on port 9. RE1 was set up to trigger the sensor, and RD4 to wait for the response signal. A voltage divider was used in order to step down the voltage from 5 volts to 3.3 volts, in order not to damage the PIC18 microcontroller. The values for the resistors were $2k\Omega$ and $1k\Omega$ for R_2 and R_1, respectively. The DC motor driver has two powers one for the logic power and the other one is for the DC motor power.

The logic power needs to be at the same voltage level as the microcontroller in order not to step up or step down the voltage in between. Therefore, the logic power was powered by the 3.3V from the base board. The RE0 from the port J6 was used to turn on and turn off the DC motor, which is the sleep mode in the DC motor driver. This is shown in the logic table below, Table 2. The RC2 acted as the PWM signal that will change as needed to increase or reduce the speed of the car, this is connected to Enable pin in the DC motor driver.

Table 2: Logic table for the DC motor driver

ENABLE	SLEEP	OUT 1	OUT 2	Operating mode
PWM	1	PWM	L	Forward at speed PWM %
0	1	L	L	Brake low (outputs shorted to ground)
X	0	Z	Z	Coast (output floating disconnected)

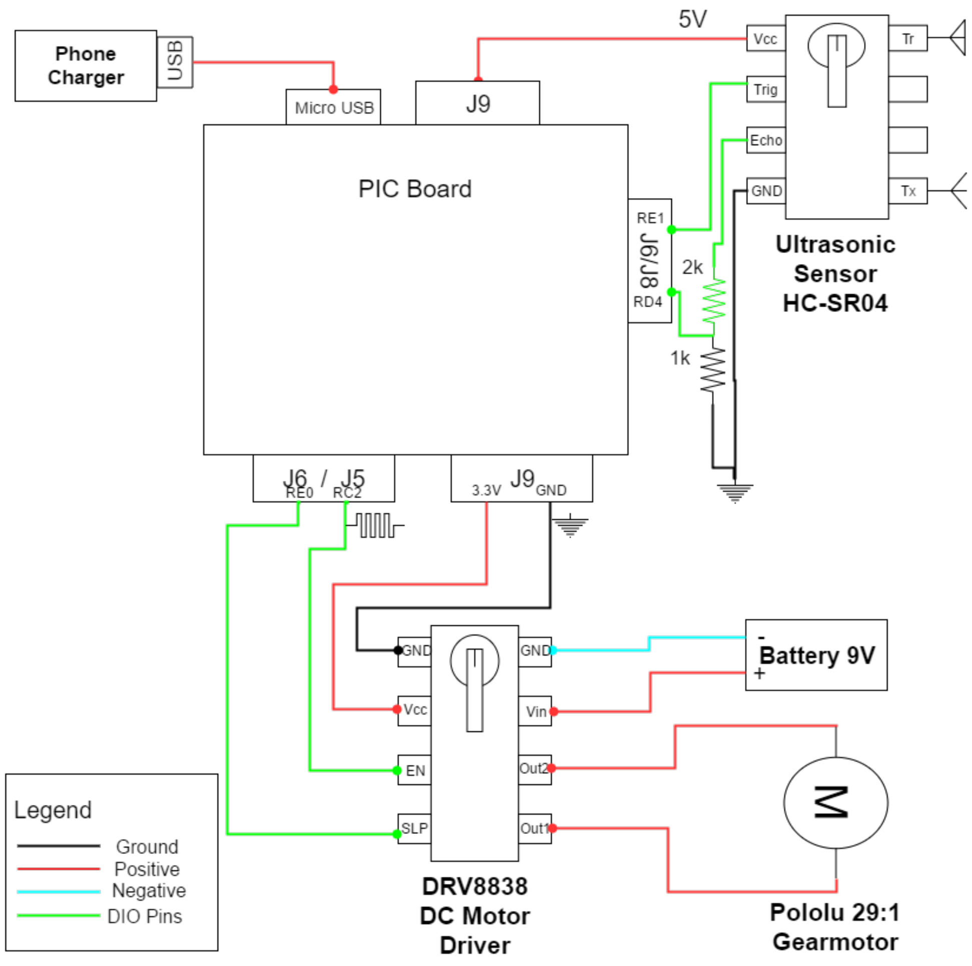


Figure 2: Wire schematic for Janus

A summarized description of all the pins and ports used in the program is shown in table 3. This table can also be use as an assembly guide.

Table 3: Pins with their use in the program

Pin	**Ports**	**Description**
RE1	J6	Trigger the ultrasonic sensor
RD4	J8	Waits for the signal to echo back
RE0	J6	Turns on and off the DC motor
RC2	J5	Sends a PWM signal to control the motor speed
3.3V	J9	Power to the DC motor driver
5V	J9	Power to the ultrasonic sensor
GRD	J9	Have a common ground for all sensors and devices

2.4 Software Flowchart

The software is divided in 5 different functions: Initialization function, measure distance function, controls function, PWM signal function, and the delay function. It is suggested to the reader to follow Fig.3, as it contains the flow of the code. For this code only one low priority interrupt was used. It is worth mention that all variables were used as global variables in order to reduce time.

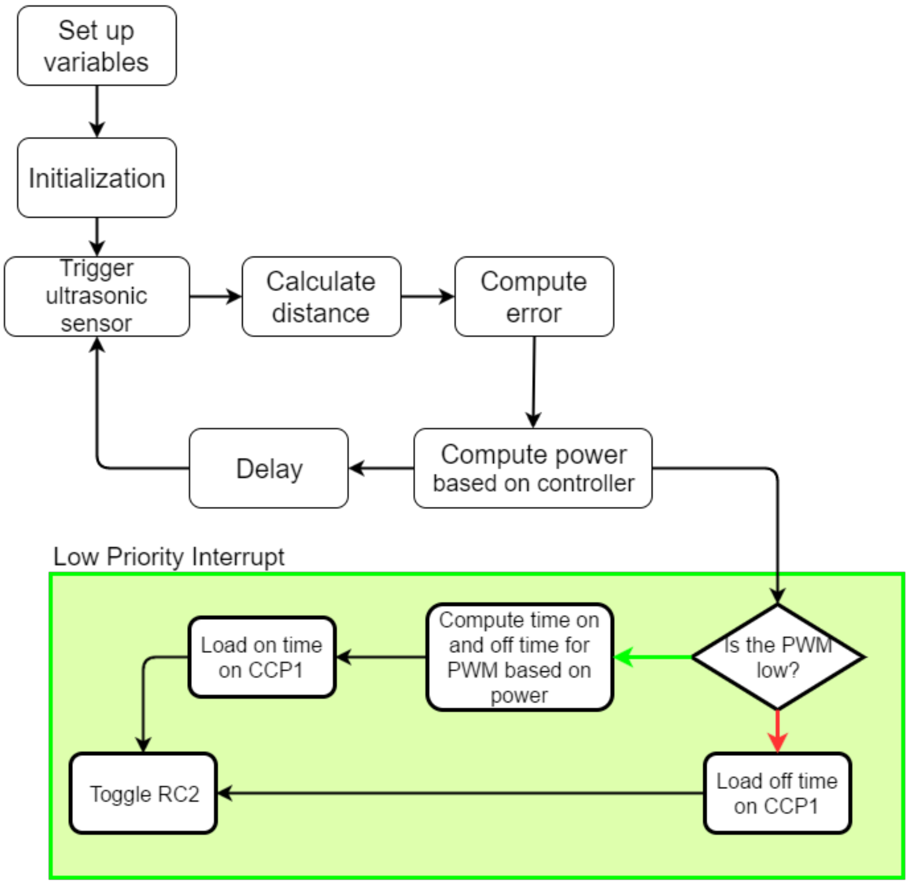


Figure 3

2.4.1 Initialization Function

The initialization function is located outside the infinite loop and it is used to initialize all ports, interrupts, and the LCD screen as well as set all dummy variables. Timers 0, 1, and 3 are also set up for the alive LED, ECCP1 and to measure time for the sonic sensor, respectively. In order to know that the code is starting or restarting a LED blinky function was created, as an indicator.

2.4.2 Distance Measurement Function

The first step inside the infinite loop is to trigger the ultrasonic sensor, this trigger happens inside the distance measurement function. It will send a signal through RE1, wait until RD4 reads the signal. Then the code will move to the next block which is to calculate distance. This is done by multiplying the delay by the speed of sound in order to get the distance as show in Eq. 1. Once this distance is determinate, it will be converted from meters to centimeters, and output the value to the LCD. The reason why time in Eq. 1 is divided by two is because the times account for the signal to reach the obstacle and bounce back. The speed of sound will be assume to be a constant define by $a = \sqrt{\gamma R T_0}$, where the temperature of the room is 25 C, γ

for air is 1.4, and R for air is 287 J/kg/K. The expected waiting time for the signal will be from 300 microseconds at 10 cm to 12 milliseconds at 4 meters.

$$d = a\,\frac{\Delta t}{2} \tag{1}$$

2.4.3 Control Theory and Function

After the distance is computed then the code moves to the next block. The controller function uses the distance calculated and finds the error according to the reference distance. This error will be use to compute the control gain by multiplying a gain in order to get proportional gain. It can also be summed to get the integral control or divided by the delay time in order to get derivative control if necessary. This is shown below in Eq. 2, where it uses the current error for control. Since Janus works on discrete times instead of continuous times the derivative will turn into the current error minus the previews error divide by the time to calculate the derivative of the error over time, this is shown on the second term of Eq. 2. In the third term the integral of the error has to be computed by the sum of the error, since an integral is just the sum at infinitesimal time steps over a time span.

$$C_i = K_p E_i + K_d \frac{E_i - E_{i-1}}{\Delta t} + K_I \sum_{i=0}^{N} E_i \tag{2}$$

2.4.4 PWM Signal Function

The PWM signal subroutine is use as an interrupt, the ECCP compare mode together with Timer 1 will be use to provide a pulse width modulation. It will be assumed that the start of the period is the beginning of the off time. When this happen based on the power output, a value from 0 to 1, on the control function will multiply this power by the period as shown in Eq.3, the period is chosen to be 50 Hz . This value will be the on time, then to find the off time it will be the period minus the off time as shown in Eq. 4.

$$\Delta t_{on} = T * Power \tag{3}$$

$$\Delta t_{off} = T - \Delta t_{on} \tag{4}$$

It is worth mentioning that both the ECCP1 and the TMR1 have an extension byte to increase the counter. After the time on and time off were found, time off was added to the ECCP1 and the ECCP1 extension. The carrier bit was checked when the time was added to ECCP1 to account for the extra bit. The first bit of CCP1COM will be toggle to turn off the RC2. After the time off passed the interrupt handler will go into the on part, and add the on time previously calculated using the same processes. It will also toggle the first bit of CCP1COM to turn on RC2. At the end of the CCPhandler the flag will be cleared. It should be also mention that the race condition is checked everytime the code enters this function. This race condition is to make sure ECCP1 and the timer 1 extension match if not it will clear the flag and leave.

2.4.5 Timer 1

The handler for timer 1 is simple it will only check for over flow and add 1 to the extension, clear the flag and leave. Timer 1 is used as an low interrupt.

2.4.6 Delay

The reason a delay function was used is twofold. The first reason is to use the time between error computation to be able to know the Δt in the derivative term for the computation. The second reason is to prevent the code from computing the distance too quickly as it was having trouble when a delay was not there. Currently the delay is set to 50 ms.

3 Results

3.1 Motors

Three motors were found in Trudy's shop. However only one motor had the part number with it. In order to find the typical power output of the other three motor a test were performed. This test uses the current into the motor as well as the voltage difference, this way the power require to run the motor can be calculated with Eq. 5. All motors were run at 6 volts, since efficiency varies with power input. During the test some friction was added with a finger on the motor to simulate torque acting on the motor.

$$P = VI \tag{5}$$

Once the power consumption was found for all three motors it was assumed that the motor had 90% efficiency between power consumption, and power output. Then the amount of revolutions in 30 seconds were counted, by assuming constant angular speed the revolutions per minutes were found. Now that angular speed is known and the Power output the torque output can be found through Eq. 6. By knowing the torque it can be calculated which motor is able to move the car.

$$P = \omega\tau \tag{6}$$

An other important variable from this test is the angular velocity, since the velocity can be back out from this parameter and the radius of the wheel. Assuming that the wheel does not slip the, calculating the circumference, which is linear distance, and knowing the revolutions per minute. This will yield the velocity in meters per minutes and dividing by 60 will yield meters per seconds. This is shown in Eq. 7, where the radius is 3 inches.

$$V = 2\pi r\omega \tag{7}$$

Finally all of this parameters are found in the table below, 4. Which shows the power consumption, RPM, torque and velocity for all three motors. As it can be seen all of the best motor found was the medium motor. However, the motor pick for this project was the pololu motor since the pololu motor had the best interface, and had the most likelihood of working for the demo.

Table 4: Parameters for all motors

Motor	**V**	**I**	**P**	**Ω [Rev/s]**	**Ω [RPM]**	**Torque [Nm]**	**Velocity [m/s]**
Big motor	6.05	0.044	0.2662	4.4	264	0.001	0.335
Medium motor	6	0.756	4.536	4.4	264	0.017	0.335
Pololu motor	6.16	0.182	1.12112	4	240	0.005	0.305

3.2 Final Control

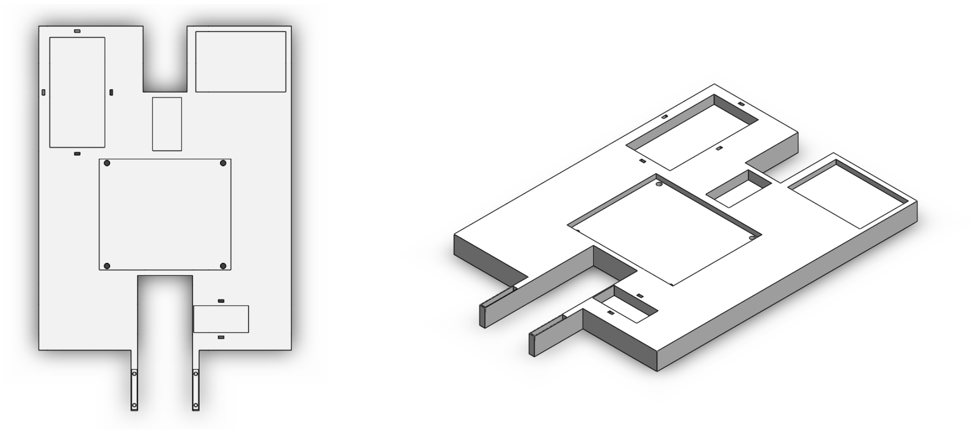
After the car was built, the car was tested in order to find the gain constant for all three control algorithm. It was found that the car does not have enough power in order to output fast RPM.

This led to the use of proportional gain only. If the DC motor were to output more power an integral gain would be used in order to reduce the steady state error. Using the brute force method the optimal value for the proportional gain is 2.5, the value for the integral and derivative control was found to be 0.

4 Lessons Learned and Obstacles

4.1 Designing a Car

Even though creating a car is not too difficult, creating a car with no budget turn out to be an creative experience. The biggest constrain was the wheel and axle, this were found on a toy in GoodWill that was taken apart. Another constrain was to be able to fit all of the parts in the car, the challenging part of this was the phone battery which was both heavy and large relatively to the other parts. Since the axle were not long enough and the gear needed to be mounted on the middle of the axle a thin strip was left with the holes for the pillow bearing block. This can be seen in the picture below 4a. Then it was originally intent to have the pic in the middle the breadboard on the top right corner, the phone charger on the top left corner and the battery in the middle. However this modification changed during the first test as explained in the next section.

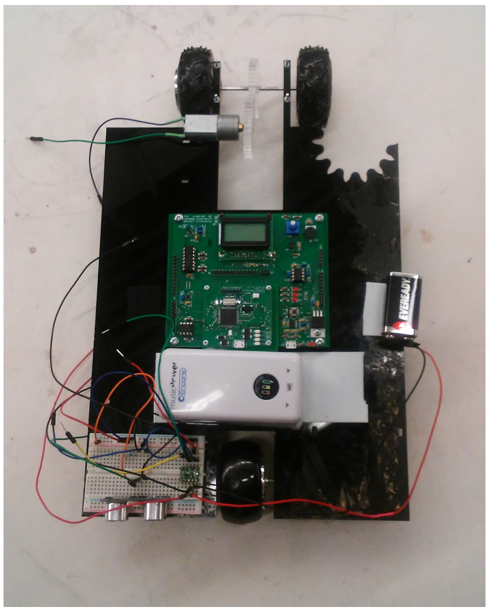


(a) Top view (b) Isometric view

Figure 4: SolidWorks model as it was designed to be made

4.2 Interface

During the first test the car did not have enough power to move for two reasons. The first one was because the power was capped to 50% to make sure the car did not go out flying. This cap was changed to 90%. The second reason was the unbalance weight distribution in the car. Since the phone charger weighted so much and was so far out from the center of gravity it was causing the car to tip over and not being able to hold the weight. The phone charger was changed to be in the middle with more weight towards the right side of the car and the 9 volt battery was placed as far out as it can in order to counter act the weight exerted by the phone charger. This arrangement seems to works as the car was able to move. A picture of the final arrangement is shown below in Fig. 5.


(a) Top view

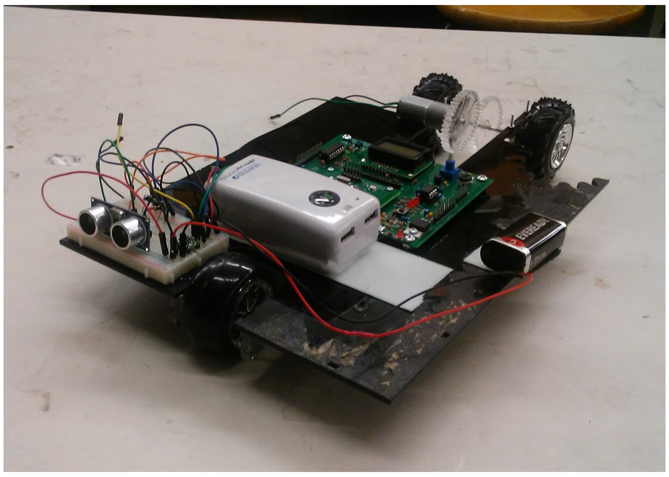

(b) Isometric view

Figure 5: Final model

4.3 From Error to Power Output

Another difficulty in this project was going from the output of the control function in error to actual power. This was difficult since in all of the control classes, everything is being simulated in Simulink and in the laplace domain. This time it needed to be written in C and in time domain. The way that this problem was solve was by multiplying the output by the ratio of maximum power divided by maximum error. However, this will only work for proportional gain since it assumes to be linear. If derivative gain or integral gain were used the scaling would have to be different. For the application the ratio made the code to work.

When thinking of this project it never came to mind that this was going to be a problem but figuring this out turned out to be more of a challenge. An there were not a lot of documents in the online about this topic.

5 Conclusion & Recommendation

If something in this project needed to be replaced or taken out, the motor would need to be replace with a more powerful motor. Instead of using the pololu motor, the motor would be the medium motor as it has 4 times the power of the pololu motor. A longer axle would be useful too as it can provide more balance to the car.

References

[1] https://www.pololu.com/product/1162/specs

[2] http://www.gearbest.com/development-boards/pp_58067.html?currency=USD&lkid=10133904&gclid=Cj
ILJI4e4nSbZ60uQLaerhyQGPv3meM_6Ebn_XUVy77wNZxoCuKnw_wcB

[3] https://www.pololu.com/product/2990

[4] https://www.mcmaster.com/#standard-plummer-block-mounted-bearings/=15giwp9