Indoor Obstacle Avoidance Patrolling Vehicle

ME 4405 Fundamentals of Mechatronics
Final Project Report

Zhubo Zhou
Zhengyang Weng

Introduction

Project Objective

In this project, we aim to design a system capable of avoiding collision with walls in an indoor scenario. The system will scout its surrounding environment with ultrasonic sensors to determine its position; acquiring signal feedback from the sensor, the controller will determine the operation mode of the vehicle to either propel forward or steer.

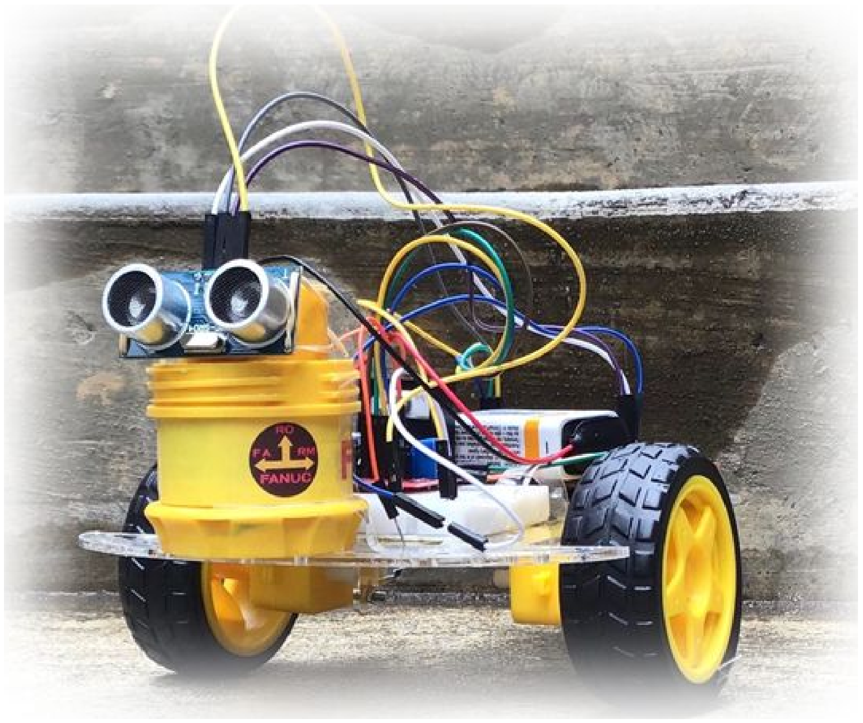


Figure 1. Project Vehicle

Motivation

Household chores are tiring and time-consuming. Many housework such as floor sweeping and mopping include repetitive ground movement in an indoor scenario. In a well defined indoor area, many of these tasks could be carried out by an autonomous vehicle patrolling the floor and performing different actions. By effectively covering ground areas, the vehicle could serve as a platform for more applications such as floor sweeper or storage delivery unit.

Performance Goals

The anticipated performance goals are listed and will be evaluated as follows:

Target Performance	Evaluation Methods & Metrics
straight line movement	eyeball, operation along smooth wall
can consistently detect & stop at 20 ± 2cm in front of wall	measure using ruler
can rotate 90° to the right after it has stopped in front of the wall	eyeball, continuing operation around
can resist disturbance	move the car back by 10 cm and see if it still wants to proceed & stop at 20cm away from wall

Table 1. Performance Goals and Evaluation Criteria

The vehicle is designed to operate consistently, therefore many of the evaluation criteria pertain to its movement. Further, the vehicle needs to be robust against disturbances to ensure its operational safety.

System Overview

In order to determine the wall-vehicle distance, the controller excites the ultrasonic sensor by sending a brief trigger signal, which will then be turned into 8 waves of pulses being sent forward. Once the sensor records an echo, the signal will be processed via gain calibration and converted to distance data. The controller determines the system operation by comparing error and outputs corresponding PWM signal, the motor outputs torque and leads to new system position, which generates a new wall-vehicle distance to be acquired by the sensor.

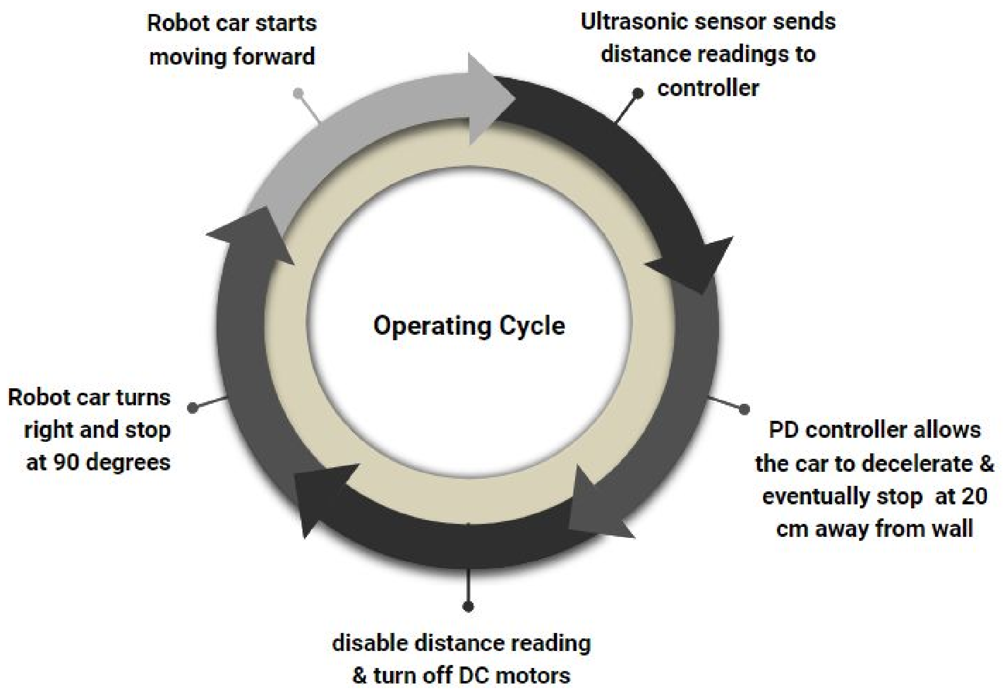

Robot car starts moving forward
Ultrasonic sensor sends distance readings to controller
Operating Cycle
PD controller allows the car to decelerate & eventually stop at 20 cm away from wall
disable distance reading & turn off DC motors
Robot car turns right and stop at 90 degrees

Figure 2. System Overview

Mechanical Design

<u>Physical System Model</u>

General System Architecture

- Chassis and Mechanisms
    - Acrylic chassis
    - Motor mount
    - Sensor mount
    - Electronics mount
    - Passive universal wheel
- Primary Actuators
    - Propelling brushed DC motors
      Gear reduction: 1:48 gear ratio
- Sensors/Transducers
    - HC-SR04 Ultrasonic sensor

Dimensions

- 25cm x 16cm x 10cm

Mass and Inertia

- 1.2 kg+0.3kg external battery
- Moment of Inertia from Center of Mass:
    - Ixx = 0.0263 kg*m^2
    - Iyy = 0.0539 kg*m^2
    - Izz = 0.0734 kg*m^2

System performance:

- Forward motion
    - Achieved through synced PWM signal
    - Propelled through gear reduction transmitted DC brushed motor
        - 1:48 gear ratio
- Rotational motion
    - Achieved through reversed PWM signal
    - Rotation about the center of axle

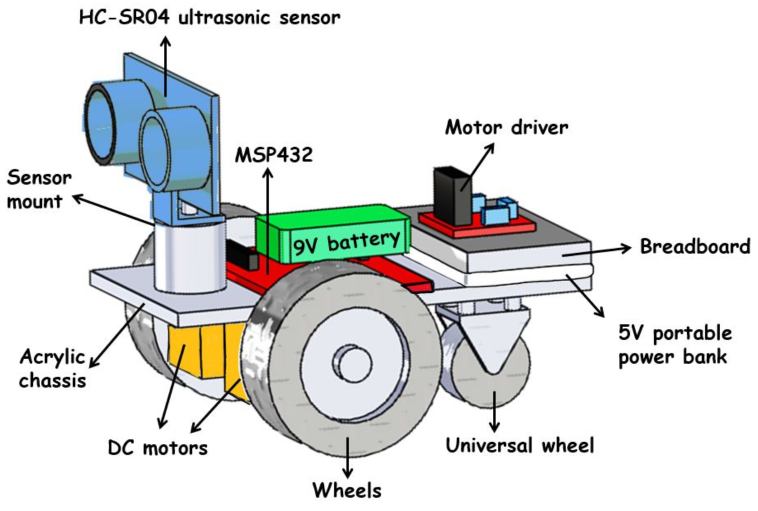


Figure 3. Physical System Model

Mechanical Model and Analysis

Free Body Diagram

In order to account for the torque required during its operation, a free body diagram of the vehicle is created and analyzed.

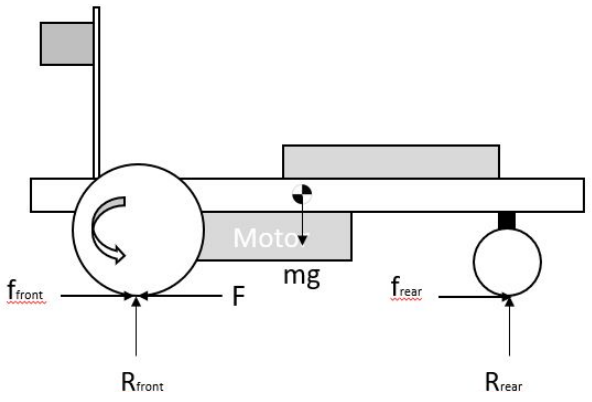


Figure 4. Free Body Diagram of the Physical System Model

Dynamic simulation:

To find out the torque required for the vehicle to maintain its operation speed, the following simulation is used:

$$\sum F_x = F + \sum f = 0$$
$$F = f = N * C_{rr}$$
$$\tau_{axle} = F/2 * R \quad (2 \; motors \; running \; in \; pairs)$$
$$\tau_{motor} = \tau_{axle}/GR$$

$$C_{rr} = 0.01 \; estimated \; from \; data$$
$$N = 15N, \; R = 0.03m$$

$$\tau_{motor} = 0.047mNm$$

However, rolling resistance C_{rr} tend to change due to different surface contact condition, therefore a worst case scenario of $C_{rr} = 0.05$ is used, resulting in:

$$\tau_{motor} = 0.235mNm$$

This is the required torque for system operation.

Actuator Selection

In order to find the appropriate choice of actuators, the following merchandises were investigated and compared:

Actuator Specifications					
Component Name(Cost)	Description	Performance Requirements	Performance Specifications	Power Source	Control Method
DC TT Motor ($3.50 each)	3V-6V DC TT Motor for Smart Robot Car	Speed: 15000 RPM Torque: 0.235 mNm	Speed: 16500RPM Torque: 3.125mNm	9V battery pack & motor driver L298N	PWM control using motor driver L298N
RS Pro 2389759 $9.34	DC Brushless Motor 15V	Speed: 15000 RPM Torque: 0.235 mNm	Speed: 13360 RPM Torque: 1.51mNm	15V DC 2.85A	PWM control using motor driver L298N
Anself 775 DC brushless motor	3500-9000RPM Motor with Ball Bearing	Speed: 15000 RPM Torque: 0.235 mNm	Speed: 9000rpm max Torque: 8.15mNm	12-36V DC, 0.2A	PWM control using motor driver L298N

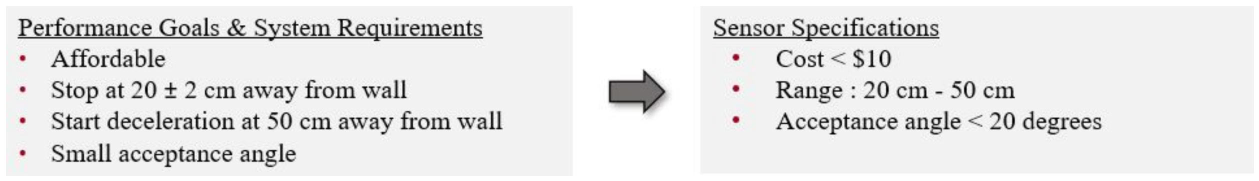
Table 2. Actuator Selection

Electrical and Circuit Design

Sensor Selection and Signal Conditioning

Sensing requirements:

In order to select a distance sensor that is both affordable and able to satisfy all the performance goals, the following criteria are used to filter out sensors fail to meet our performance goals:



Performance Goals & System Requirements
- Affordable
- Stop at 20 ± 2 cm away from wall
- Start deceleration at 50 cm away from wall
- Small acceptance angle

Sensor Specifications
- Cost < $10
- Range : 20 cm - 50 cm
- Acceptance angle < 20 degrees

It is crucial to select a sensor with a small acceptance angle. Since our robot car is only 25 centimeters tall, if the selected sensor has a large acceptance angle, chances are that we will get noisy readings from the sensor since both signals reflected by the ground and the obstacle will be received.

Using criteria been derived, the following candidates were investigated and compared:

Sensor Specifications					
Component Name	Cost (US$)	Range	Angle	Resolution	Voltage Requirements
HC-SR04 ultrasonic sensor	2.99	2cm – 4 m	15 deg	40kHz 8 Cycle burst	5V DC

Parallax's PING))) Ultrasonic Sensor	29.99	2 cm - 3 m	N/A	40 kHz for 200 µs	5V DC
Obstacle avoidance IR sensor	4.86	2 - 30 cm	35 deg	N/A	DC 3V-5V
IR proximity sensor - Sharp GP2Y0A21YK	13.95	10 - 80 cm	Smaller range than HC-SR04	3.1V at 10cm to 0.4V at 80cm	DC 0V-7V

Table 3. Sensor Selection

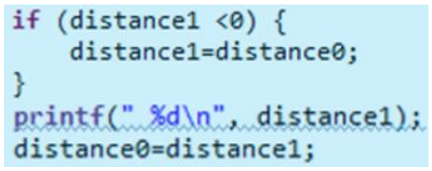
Among the above four commonly used distance sensors, HC-SR04 was selected since it is the only sensor that meets all the requirements. It has a large range, small acceptance angle, and can meet our budget.

Sources of noise

Potential sources of noises have been identified as:

- Environmental noises
  - reflection from ground - due to low elevation of the sensor
  - Jagged reflective surface
- System noises
  - data overflow (get negative values for distance reading)
    - within range - bad readings
    - exceed max range

In order to properly condition the signal, we decided to heighten the sensor position to reduce reflection from ground. Further, since data overflow always leads to negative distance readings, and that our system has a high sampling frequency and is operating at relatively low speed, noise introduced by data overflow can be corrected using the algorithm shown in Figure 5, in which current sensor reading is set to be the same as the last sensor reading, whenever a data overflow is detected.



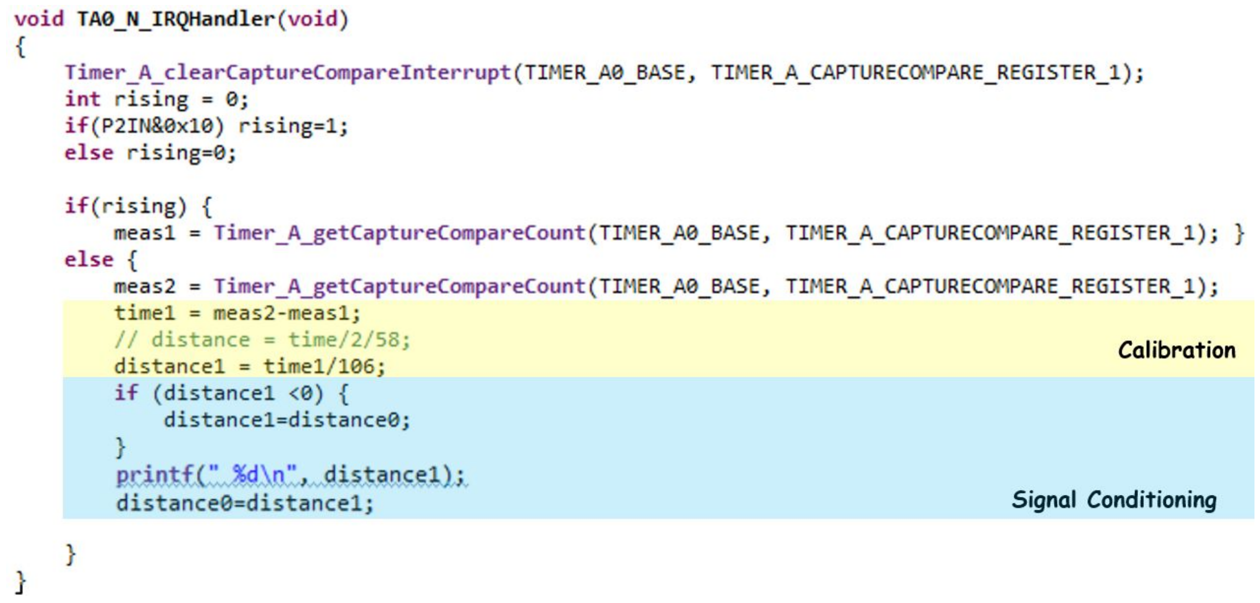
```
if (distance1 <0) {
    distance1=distance0;
}
printf(" %d\n", distance1);
distance0=distance1;
```

Figure 5. Signal Conditioning Approach

Simulate Signal Conditioning

Instead of simulating noise signals caused by the sensor in Matlab/Simulink, sensor readings were collected directly by printing out calibrated readings from the sensor using Code Composer Studio.

During testing, the following piece of code was used to read from the ultrasonic sensor using MSP 432:



```
void TA0_N_IRQHandler(void)
{
    Timer_A_clearCaptureCompareInterrupt(TIMER_A0_BASE, TIMER_A_CAPTURECOMPARE_REGISTER_1);
    int rising = 0;
    if(P2IN&0x10) rising=1;
    else rising=0;

    if(rising) {
        meas1 = Timer_A_getCaptureCompareCount(TIMER_A0_BASE, TIMER_A_CAPTURECOMPARE_REGISTER_1); }
    else {
        meas2 = Timer_A_getCaptureCompareCount(TIMER_A0_BASE, TIMER_A_CAPTURECOMPARE_REGISTER_1);
        time1 = meas2-meas1;
        // distance = time/2/58;
        distance1 = time1/106;
        if (distance1 <0) {
            distance1=distance0;
        }
        printf(" %d\n", distance1);
        distance0=distance1;

    }
}
```

Figure 6. Code used to read from HC-SR04

Initially, the data was very noisy with overflown readings. After conditioning, signal readings become a lot more continuous and smooth.

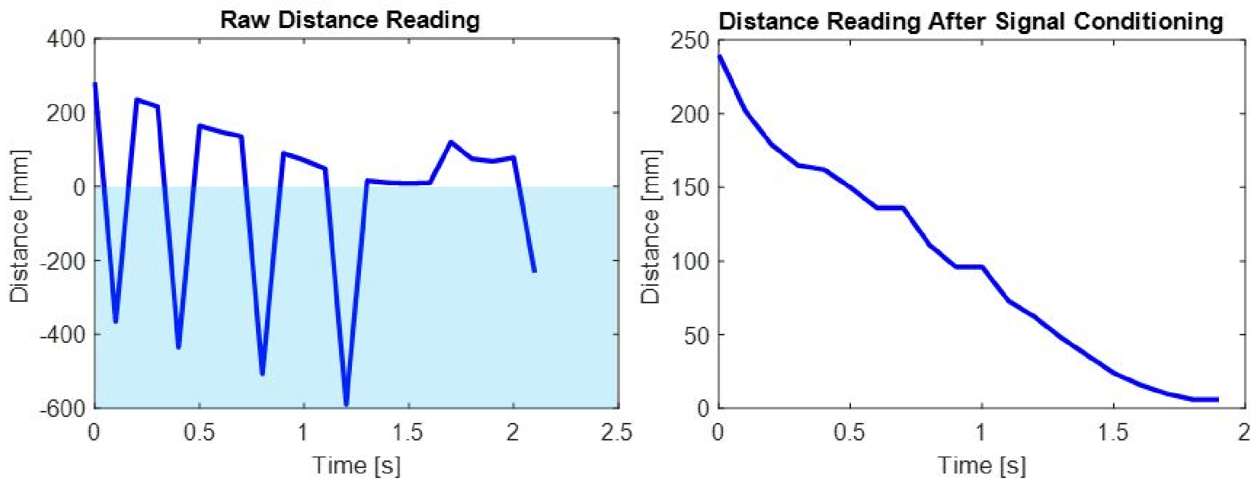


Figure 7. Signal Conditioning

Electrical Schematic

Below is a schematic layout of onboard electronics:

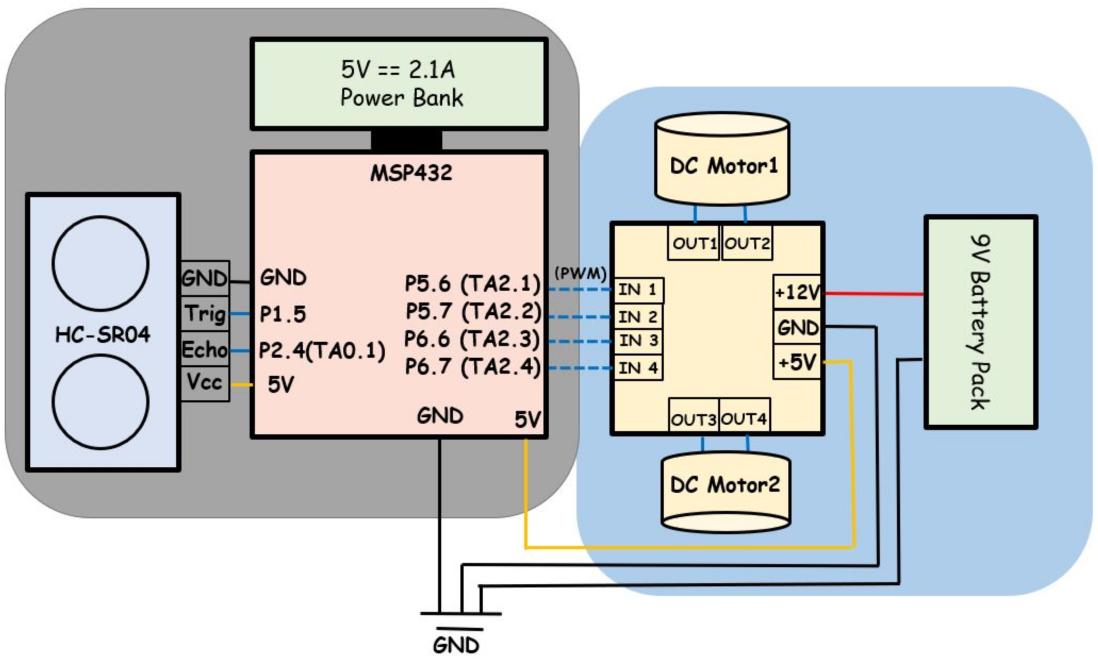
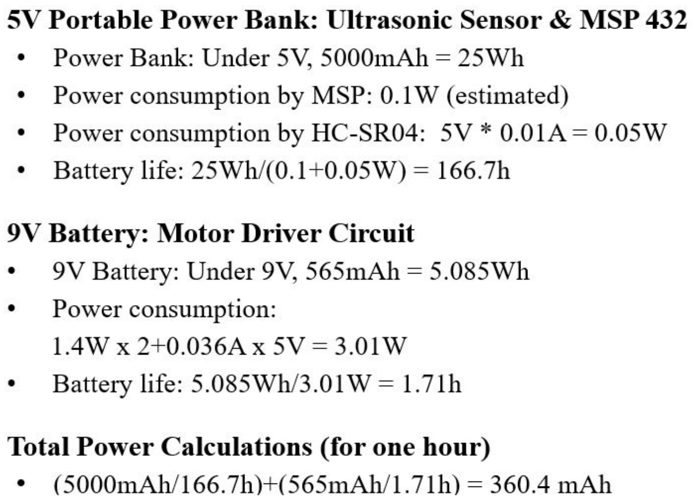


Figure 8. Electrical Schematic

Power Supply Specification

A 5V (2.1A) portable power bank was selected to power MSP432. It has a 5000mAh capacity and comes with a micro-USB cable that can be directly plugged into MSP432. Since the current draw from the ultrasonic sensor at 5V is 10mA, which is smaller than allowed current draw (25mA) can be handled by GPIO pins on MSP432, the ultrasonic sensor is powered directly using 5V port on MSP432. However, since the motor circuit requires a higher voltage level, a separate 9V battery is used to power the motor driver circuit.

In order to estimate end-of-envelope battery life for our system, the following calculations was performed:



5V Portable Power Bank: Ultrasonic Sensor & MSP 432

- Power Bank: Under 5V, 5000mAh = 25Wh
- Power consumption by MSP: 0.1W (estimated)
- Power consumption by HC-SR04: 5V * 0.01A = 0.05W
- Battery life: 25Wh/(0.1+0.05W) = 166.7h

9V Battery: Motor Driver Circuit

- 9V Battery: Under 9V, 565mAh = 5.085Wh
- Power consumption:
  1.4W x 2+0.036A x 5V = 3.01W
- Battery life: 5.085Wh/3.01W = 1.71h

Total Power Calculations (for one hour)

- (5000mAh/166.7h)+(565mAh/1.71h) = 360.4 mAh

Figure 9. Power Consumption Calculation

The result shows that our system will consume 360.4mAh per hour, and can operate for at least 1.7h without recharging.

Feedback Control and Simulation

Mechatronic System Block Diagram

Figure 7 shows the simulink block diagram that represents our physical system. When the wall is far away, the robot car will go straight at constant speed. When the vehicle is 50 cm away from the wall, the PD controller will allow the vehicle to decelerate and eventually stop at 20 cm away from the wall. Once it stops, the DC motors will be controlled to turn in opposite directions to allow for a 90 degrees counterclockwise turn.

For the PD controller, reference distance is set to 20 cm since we want it to stop at 20 cm away from the wall. Two step functions are used to simulate a pulse disturbance that starts at 30s and ends at 40s. The DC motors are modeled using a transfer function, for which control signal from the PD controller will be its input and angular velocity of the motors will be its output. By taking integral of the angular velocity over time, and then subtracting it from initial distance, real-time wall-vehicle distance can be represented, which will be measured by the ultrasonic sensor . Error can therefore be calculated by subtracting wall-vehicle distance measured by the ultrasonic sensor from the reference distance.

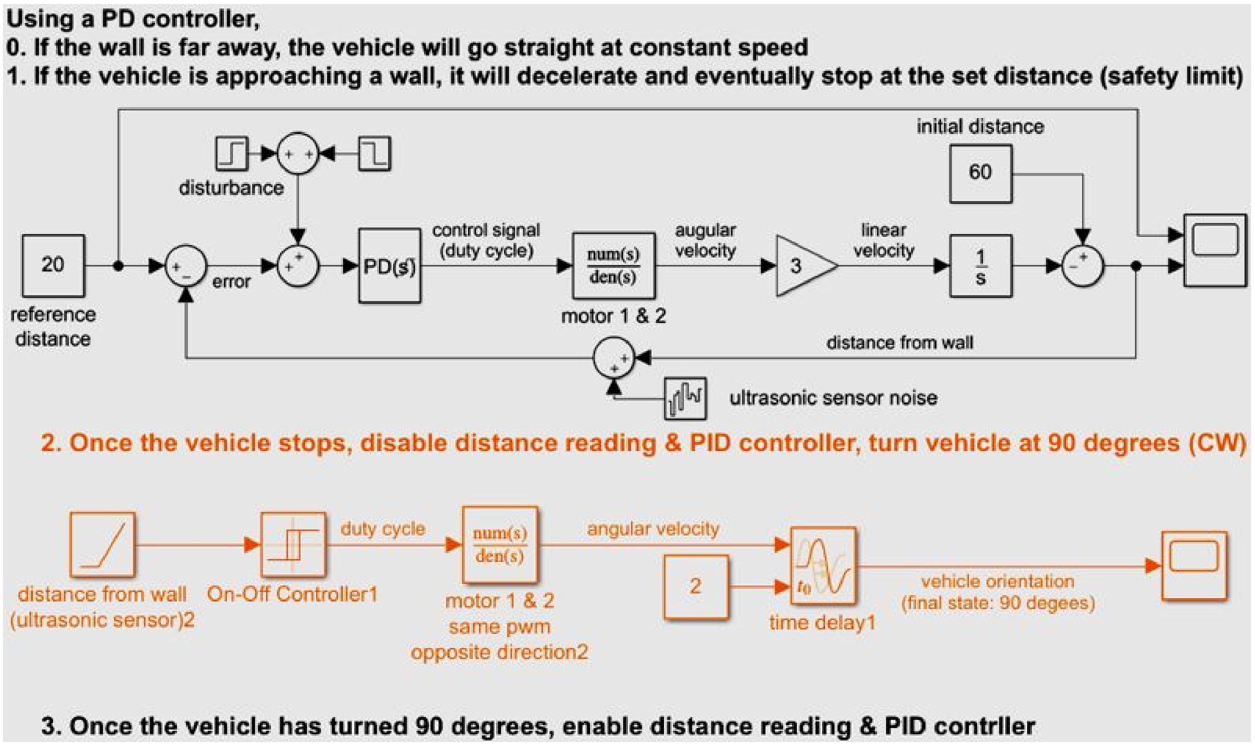

Using a PD controller,
0. If the wall is far away, the vehicle will go straight at constant speed
1. If the vehicle is approaching a wall, it will decelerate and eventually stop at the set distance (safety limit)
initial distance
60
disturbance
control signal (duty cycle)
augular velocity
linear velocity
20
reference distance
error
PD(s)
num(s)/den(s)
motor 1 & 2
3
1/s
distance from wall
ultrasonic sensor noise
2. Once the vehicle stops, disable distance reading & PID controller, turn vehicle at 90 degrees (CW)
duty cycle
angular velocity
distance from wall (ultrasonic sensor)2
On-Off Controller1
motor 1 & 2 same pwm opposite direction2
2
time delay1
vehicle orientation (final state: 90 degees)
3. Once the vehicle has turned 90 degrees, enable distance reading & PID contrller

Figure 10. Mechatronics Block Diagram

PD Controller Design

A proportional derivative (PD) controller is used to control the DC motors to stop the vehicle at a set distance from the wall. The proportional component helps to drive the system toward its safety limit distance, and the derivative component can help reduce overshoot and oscillation. Although in theory a PID controller can allow for faster system response and eliminate steady state error, an integral controller was not implemented because it might introduce large overshoot & longer settling time if not carefully tuned. Since our goal is to ensure that the vehicle does not bump into the wall, as long as the vehicle is at a safe distance away from the wall, precise position control (e.g. eliminate steady state error) is not needed. In addition, a controller saturation limit will be set to allow the robot car to go straight at a reasonable fixed speed if the obstacle is far away.

Since if Kp is too small, the system will respond slowly and has low tolerance to outside disturbance, while if Kp is too large, the system might have large overshoot and tend to be unstable, the PD controller needs to be tuned carefully. Starting from setting all the gains to zero, the PD controller will be tuned by first increasing Kp until the response to a disturbance is steady oscillation, then increasing Kd until the oscillation go away, and then repeat the above two steps until increasing the Kd does not stop the oscillation.

While the system is under operation, distance perturbations are expected and will be corrected after being fed back to the control loop. The system will be tested by pushing the vehicle forward randomly on its way approaching the wall. The control scheme should compensate for the disturbance sensed. This process might involve some trial and error.

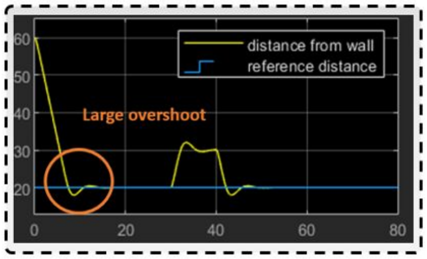
Feedback Control Simulation

Performance of the controller was simulated in Simulink using block diagram shown before using the following parameters:

- **Time Step**: 0.005s
- **Time Range**: 80s
- **Controller Saturation Limit:** ± 10
- **DC Motor Transfer Function:** $\dfrac{-kt}{R * J \cdot s + R * B + ke * kt}$

Electrical Params	Mechanical Params
R = 0.55	**B = 0.2**
K_t = 0.21	**J = 0.05**
K_e = 0.21	

To determine whether a PD controller is good enough for our application, both the performances of a PID controller and a PD controller were evaluated.



PID Controller

K_p = 3

K_i = 1

K_d = 0.5

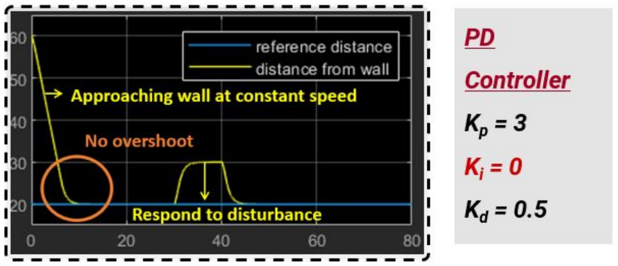


Figure 11. Controller Selection

Simulation results show that we made the correct decision. A PD controller can be tuned to achieve a system with no overshoot and negligible steady state error. However, if not properly tuned, using a PID controller not only increases complexity of the system, but may also introduce large overshoot and longer settling time. The simulation result also shows that the system is able to correct itself after disturbance has been introduced.

Writing Code to Control your System

High-level pseudocode:

The following is a pseudocode flowchart of the code. Starting from the top left corner, the system initiates the ultrasonic sensor by sending out a trigger signal. After receiving an echo feedback, an timer A will be used to determine the total length of the echo signal and apply gain calibration to the echo. After determining the actual distance from the obstacle, the controller will determine the PWM output, generating new system dynamics; the loop then repeats.

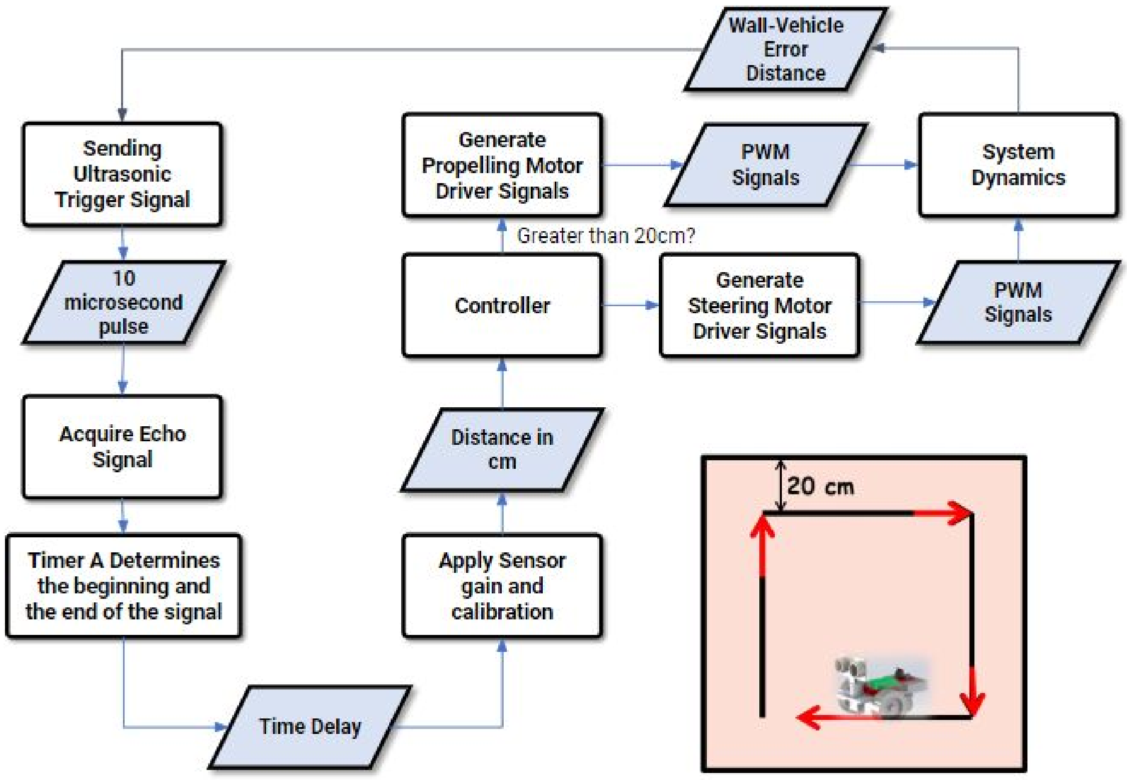

Wall-Vehicle Error Distance
Sending Ultrasonic Trigger Signal
10 microsecond pulse
Acquire Echo Signal
Timer A Determines the beginning and the end of the signal
Time Delay
Apply Sensor gain and calibration
Distance in cm
Controller
Greater than 20cm?
Generate Propelling Motor Driver Signals
PWM Signals
System Dynamics
Generate Steering Motor Driver Signals
PWM Signals
20 cm

Figure 12. Pseudocode

Process codes:

- Generating sensor trigger signal

```
GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN5);
Timer32_setCount(TIMER32_0_BASE, 24 * 10); //Using timer 32 to generate trigger
while (Timer32_getValue(TIMER32_0_BASE) > 0); //Generate a 10μS pulse under 24MHz clock
GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN5);
```

- Acquiring and determining the beginning and the end of the echo signal

```
   if(P2IN&0x10) rising=1;
  else rising=0;
  // Start
  if(rising)
{
  meas1 = Timer_A_getCaptureCompareCount(TIMER_A0_BASE,
TIMER_A_CAPTURECOMPARE_REGISTER_1);
}
else
{
  meas2 = Timer_A_getCaptureCompareCount(TIMER_A0_BASE,
 TIMER_A_CAPTURECOMPARE_REGISTER_1);
 }
```

- Apply sensor gain and calibration

```
time1 = meas2-meas1;
distance1 = time1/106;
//printf("Received: %d\n", distance1);
```

- PID controller

```
y = distance1;
error = r - y;
error_integral += error*dt;
error_derivative = (error - previous_error)/dt;
// u is a pwm ratio
u = Kp*error + Kd*error_derivative;
// limit max speed
if(u > upper) u = upper;
else if(u < -upper) u = -upper;
previous_error = error;
dutyCycle = abs(u)*base;
```

- PWM signal

```
if (forward==1 && dutyCycle~=0){
Interrupt_enableMaster();
// set bit 6 of P6SELO and clear bit 6 of P6SEL1 to enable TA2.3 functionality on P6.6
P6SEL0 |= 0x40;
P6SEL1 &=~ 0x40;
// clear bit 7 of P6SELO and set bit 7 of P6SEL1 to disable TA2.4 functionality on P6.7
P6SEL0 &=~ 0x80;
P6SEL1 |= 0x80;
// set bit 6 of P5SELO and clear bit 6 of P5SEL1 to enable TA2.1 functionality on P5.6
P5SEL0 &=~ 0x40;
```

```
P5SEL1 |= 0x40;
// clear bit 7 of P5SELO and set bit 7 of P5SEL1 to disable TA2.2 functionality on P5.7
P5SEL0 |= 0x80;
P5SEL1 &=~ 0x80;
```

Conclusion

Challenges and Solutions

Throughout this project, we have encountered many challenges in building the system, many of them turned into great experiences. Initially, we had a lot of trouble receiving correct signals from the ultrasonic sensor; we spent a long time tweaking the timer interrupt and tuning for the correct sensor gain. Combining the sensor code with motor code had also been a challenge. The system requires multiple timers running at different clock rates and interrupt handlers. We've gone through a long and tedious debugging session before getting the codes to work.

Potential Improvements

Due to the COVID-19 outbreak, many electronics supplies had become unavailable or were very slow to deliver. Our system spec was limited by the resources that we purchased before the regional lockdown. However, we do have other ideas for improvements requiring different new components and subsystems.

Though speed at which the vehicle operates can be determined from differentiating successive sensor distance readings, its accuracy could be greatly improved by adding a set of rotary encoders or velocimetry disks to the propeller wheels. Having encoder feedback would enable us to determine the revolution speed, and the difference in revolution speed of two wheels to achieve higher resolution speed and steering control.

Instead of fixing the ultrasonic sensor to a static angle, a set of servo motor could be added to the base of the sensor to allow a frontal 180 degree sweep. In case of the vehicle approaching the wall at an angle, doing so would allow us to determine the shortest distance from the wall; together with the frontal distance, the angle between vehicle path and the wall could be determined using trigonometry. It would greatly improve the freedom of trajectory planning.

Finally, the signal readings from the ultrasonic sensor could be smoothed using average readings about the data point; further increasing the sampling rate would also help to improve the continuity of data, though at a cost of increased computational power consumption.