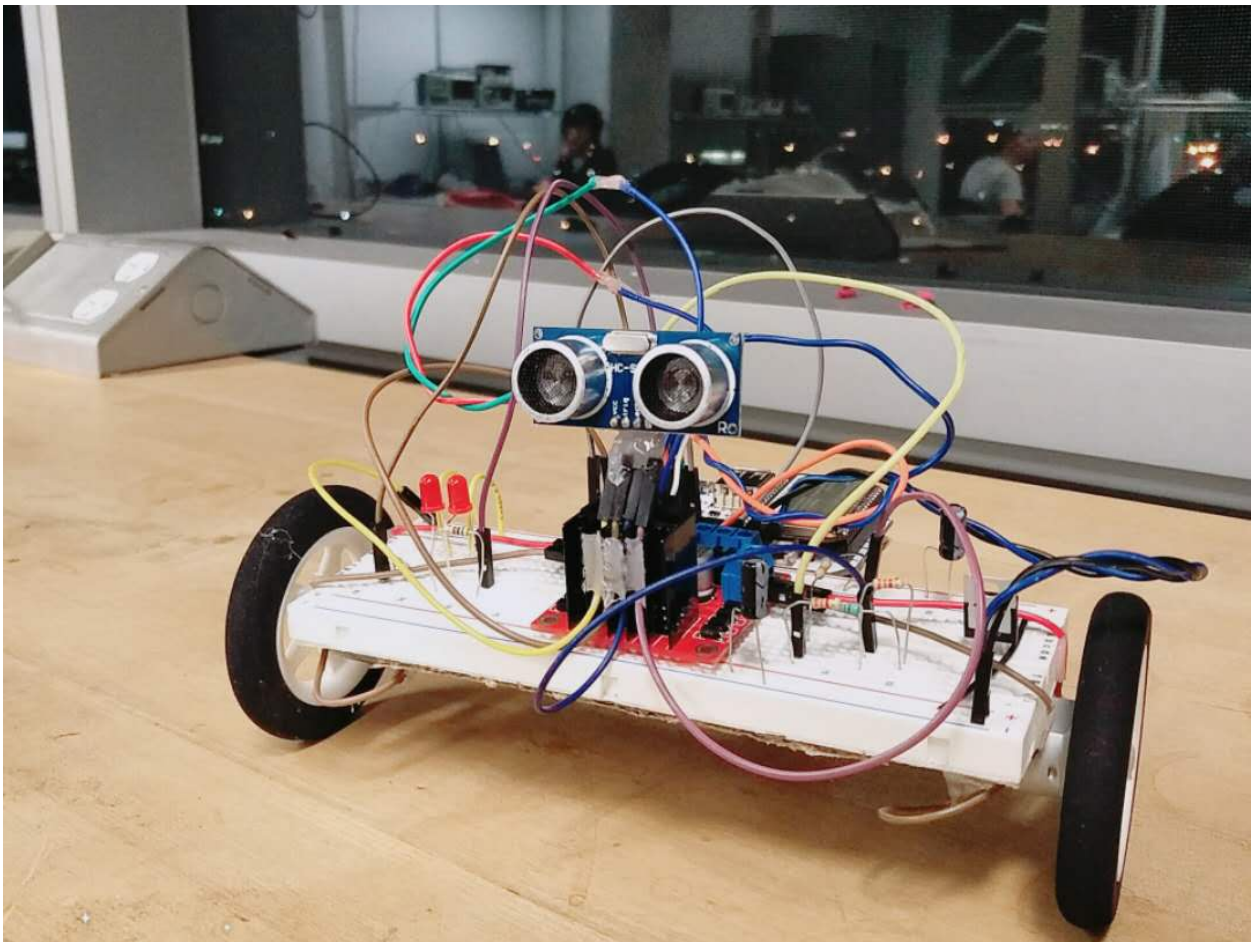
Project Report



Created by: Jiaxi Wang

Sept 2017-Dec 2017

Project Overview

What is The Project About?

The project revolves around the idea of utilizing the Omega2 to control an ultrasonic-sensor guided robot.

What Would You Have Liked it to Do (If it Could Do Everything)?

Idealistically, the robot would have been able to traverse a maze in which the robot would sense nearby obstacles and perform both right and left maneuvers to navigate a path. As well, the robot would also have wireless speed control in which case it could maneuver at different speeds at the users discretion.

What Subset Does It Do?

Maneuver: The robot travels in a straight line when turned on. This is done through the simultaneous operation of both motors in the same direction.

Detect: using an ultrasonic-sensor, the robot fires out high frequency wavelengths and records the time it takes for the wavelength to hit the obstacle and bounce back. The time is then inputted into a formula to calculate the distance of the object.

Control: The Omega2 instructs the motors to continue forward operation until an object is sensed within a 2 meter distance. When the distance of an object breaches the threshold distance, the Omega2 instructs the one motor to enter a turn phase in which it continually fires wavelengths as it turns. When the function no longer returns a value equal to or less than 2 meters, the Omega2 instructs the robot to perform the forward maneuver once more.

Turn: When an obstacle has been sensed within the threshold distance, The Omega2 instructs one motor to halt operation while the other continues. This in turn creates a pivot by which allows the robot to perform a turning motion. As this is being done, the robot continually senses any objects ahead and continues to perform the turn procedure until a clear path can be determined (no longer senses anything within the threshold).

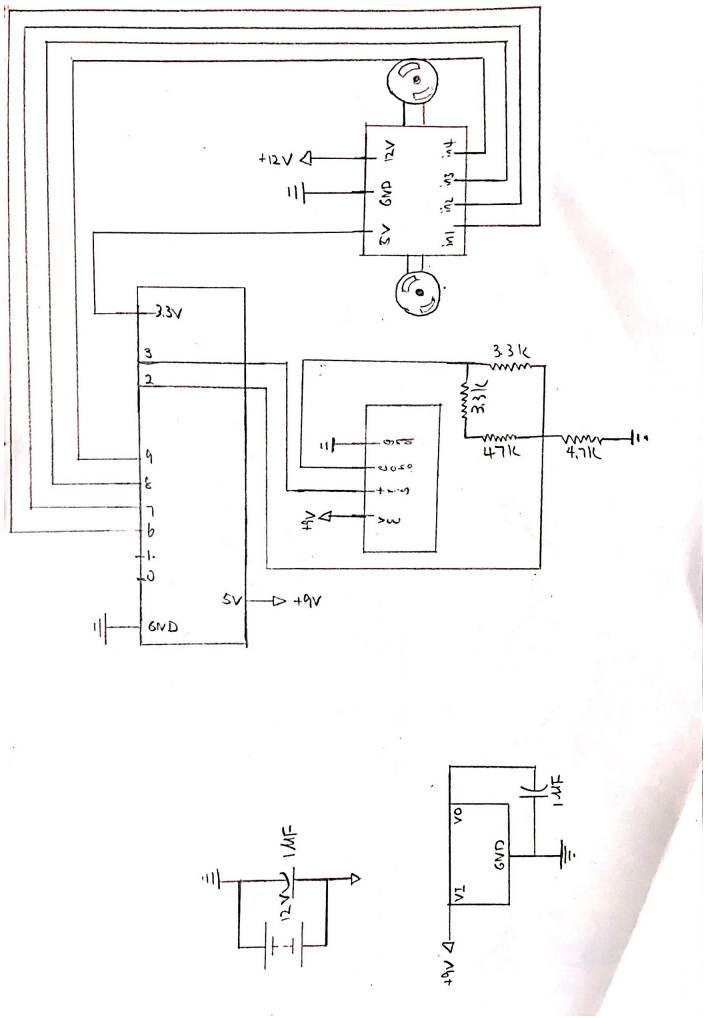
System Design

Complete System Design

Hardware Components

The design is composed of an Omega2 attached to the power dock. From here, gpio pin six and seven are connected to input pins one and two on the motor driver, respectively. Pin 0 is connected to input pin 3 and pin 1 is connected to input pin 4. Output 3 and 4 from the motor are connected to one DC motor, driving it, whilst output 1 and 2 are connected to the other motor. Gpio pin 2 is connected to the breadboard, and through a resistor pathway in which a 47k and 4.7k lie in parallel with a 3.3k while both lying is series with another 3.3k(see schematic). This is to ensure that the operating voltage of the ultrasonic sensor (5v) is made compatible. This circuit is made as there was no perfect way to convert 5v to the operation voltage of the Omega2 at 3.3v. This circuit is then connect to the sensor via the echo pin. The trig pin on the sensor is connected directly to gpio pin 3. The VCC pin(+5VDC), is connected to the positive rail of the breadboard while the ground pin is connected to the negative. From pins 18 and 19 on the power dock, jumper wires connect two LEDs situated on the breadboard. These LEDs are paired with resistors in order to limit the current and keep the LEDs operating safely. Two 1uF capacitors are connected on either rails of the breadboard in order to smooth current from the DC motors. An additional voltage regulator was placed in order to ensure smooth current throughout the board. From the H-bridge motor driver, the 5v pin is redirected to the 3.3v on the Omega2. Lastly, the ground pin from the the motor driver is attached to the negative rail, while the 12v pin is connected to the positive rail. The breadboard essentially acts as a chassis where it house the two DC motors on the bottom as well as the battery packs. The swivel-mount wheel is placed beneath the Omega2 in order to provide balance.

Schematic of Omega2-powered Robot

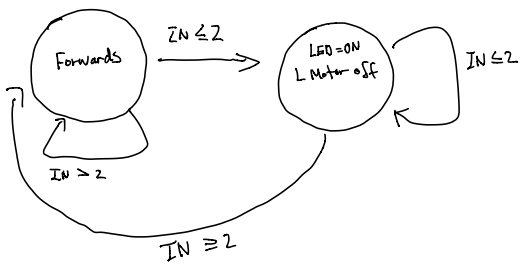


Components

- Onion Omega2
- Onion Power Dock
- MC7805 Voltage Regulator
- HC-SR04 ultrasonic sensor
- L298N Dual H-Bridge Motor Controller
- Resistors
    - 2 x 3.3k
    - 1 x 4.7k
    - 1 x 47k
- 2 x 5mm LED (red)
- 2 x 5v DC Motor
- 15 x M-M Jumper Wire
- 10 x M-F Jumper Wire
- 2 x 1uF capacitor
- Full-sized Solder-less Breadboard
- 9v Alkaline Battery
- 3.3V Li-ion Battery
- 2 x 6.5cm Diameter Wheels
- Swivel-mount wheel (275lbs)
- 2 x 10mm bolts
- 2 x machine washers

Software Components

Simple state machine diagram



Function call:
Main(trigPin, echoPin)—>sensor(distance)—>Main()

Elements:
outputSensor - sets trig pin as the output sensor
inputSensor - sets echo pin as the input sensor
setSensor - sets values for pins/fires wavelengths/turns on the sensor
time - variable for time it take for signal to be fired and returned to sensor
distance - distance of obstacle to robot. Calculated
sleep() - acts as delay. Makes calling thread inactive until a certain amount of (seconds) passes
clock() - counts number of clock ticks, used to calculate time elapsed from clock_t begin to _t end
trigPin - set according to GPIO PIN number
echoPin - set according to GPIO PIN number
time sleep - holds value for sleep()
gpioLLed - set according to GPIO PIN number. Configuration for left LED
gpioRLed - set according to GPIO PIN number. Configuration for right LED
gpioLMotor_c - Left motor clockwise direction. Sets pin from Omega2
gpioLMotor-ac - left motor counterclockwise direction. Sets pin from Omega2
gpioRMotor_c - Right motor clockwise direction. Sets pin from Omega2
gpioRMotor_ac - they motor clockwise direction. Sets pin from Omega2
valueLLed - sets value of left LED
valueRLed - set value of right LED
valueLMotor_c - sets value of left motor's clockwise rotation.
valueLMotor_ac - sets value of left motor's counter clockwise rotation.
valueRMotor_c - sets value of right motors clockwise rotation.
valueRMotor_ac - sets value of right motor's counter clockwise rotation.
outputLLed - acts as switch for left LED
outputRLED - acts as switch for right LED
outputLMotor_c - acts as switch for clockwise direction of left motor
outputLMotor_ac - acts as switch for counter clockwise direction of left motor
outputRMotor_c - acts as switch for clockwise direction of right motor
outputRMotor-ac - acts as switch for counter clockwise direction of right motor
setLLed - acts as trigger for switch for left LED
setRLed - acts as trigger for switch for right LED
setLMotor_c - acts as trigger for clockwise rotation of left motor
setLMotor_ac - acts as trigger for counter clockwise rotation of left motor
setRMotor-c - acts as trigger for clockwise rotation of right motor
setRMotor-c - acts as trigger for counterclockwise rotation of right motor

System Components:
1. Initialize direction of wheels for both motors as well as of and on for LEDs
2. While loop - while sensor can receive and transmit calculated distance
3. Continuously calculates distance through elapsed time of transmission and reception

4. If distance is over two, continues operation
5. If distance under two, switches left motor off by setting value of clockwise direction left motor to 0
6. Turns on LEDs until a two meter distance can be cleared
7. Returns value of left motor to 1, robot continues forwards motion

Testing

Testing of our project is done through creating a path for the robot to traverse whilst providing obstacles that it needs to navigate past. If the system operates as it should, the robot will sense obstacles straight ahead that are within two meters and begin to rotate itself in order to steer clear of such obstacle. When not approaching obstacles, both motors should be running in the same direction, thereby creating a straight, forward moving effect. When an object or obstacle is senses within two meters of the robot, 2 red LEDs should blink on and the robot should commence the turning procedure. When it is turning, only one of the two DC motors should be turning, and once the sensor has sensed that there are no obstacles within two meters, the LEDs should turn off and the motor that was previously turned off during the turn phase should commence rotation once again. If any of these operations are compromised under scenarios in which they should be performed, the test is deemed as unsuccessful.

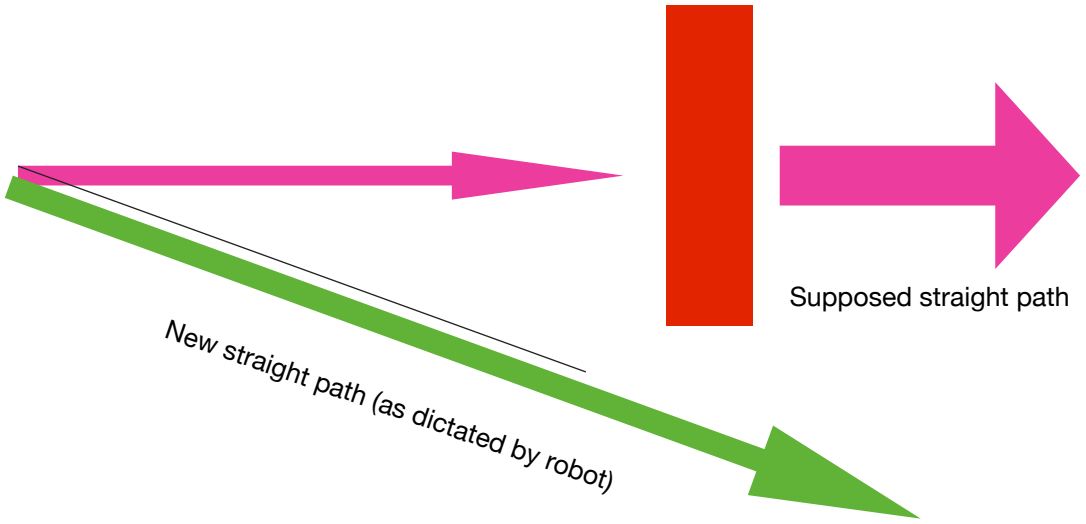
Limitations of the System

The key limitation of our robot is its inability to maneuver in both direction. For this to be plausible, we would have needed three ultrasonic sensors in order to cover the appropriate range of detection. By implementing three sensors, we would have been able to sense the optimal route. This would be done by sensing which turn would lead to an available route quicker. The sensors would be placed with two on either side of the robot, while one sensor is forward-facing. The side sensors would pick on any obstacles to the side of the robot and then turn the robot opposite to such obstacle in order to find the most efficient route. Alternatively, we could have implemented an IR sensor as its range is far greater than that of an ultrasonic sensor, and could have covered a wider area of detection.

Another limitation for our robot is its ability to change speeds. This feature could prove useful for traversing up ramps or terrain with more friction. As well, it can be expected that such a feature could be controlled wirelessly, requiring only input from a personal computer or smartphone. To accomplish this, we would have needed to utilize the servo expansion, as well as utilize the Omega2's Wi-Fi capabilities. Due to the unfortunate timing of parts, we were unable to incorporate these features.

A third limitation of our robot is its inability to remain on a targeted course. That is, once the robot encounters an obstacle, it will rotate itself until it no longer senses any foreign objects, in which it will then travel straight. However, such a course is not the original set path as the robot will have turned some vary degrees in a new direction. To the robot itself, the newly found route is considered “straight”. However from a testing standpoint, such a route deviates from the originally plot course and the robot is unable return to such a course.

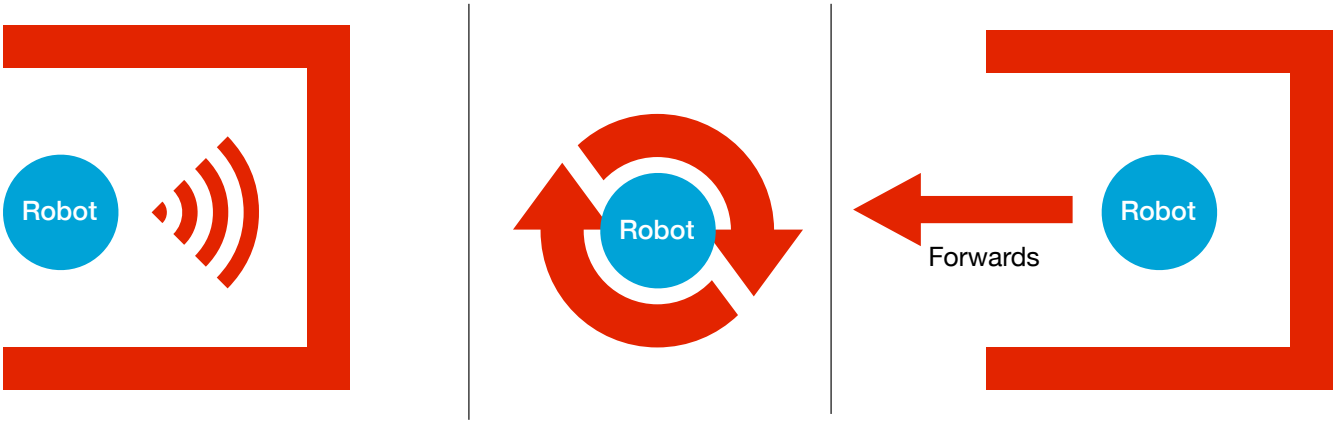
Diagram of Limitation no.3



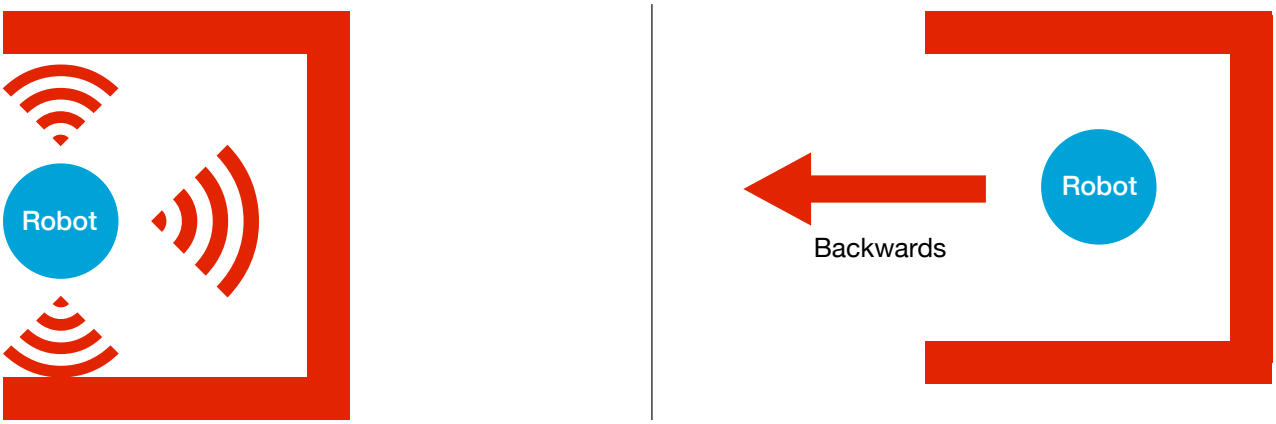
A final limitation of our robot is its inability to go backwards. We could have implemented such a feature by having both motors simultaneously rotate in the opposite direction from that of the forward motion. Doing so allows the robot to move backwards in a straight line. This function could prove useful for situations in which it would be more efficient for the robot to simply move backwards instead of rotating a full 180° and then proceeding forwards. This function was originally to be implemented had we utilized 3 sensors instead of one, due to the fact that we could assign this function to be activated when the robot is boxed in. However, since our bot only senses the area in front of it, it is impossible to tell when the backwards maneuver can be performed to a high degree of effectiveness. See example below.

Illustration of limitation no.4

Robot without backwards maneuvering capabilities



Robot with backwards moving capability



As displayed above, by implementing the backwards moving capability, the robot is much more efficient at navigating through tough obstacles such as the one pictured above. A scenario such as the one depicted is the worst case scenario, and fully exposes the shortcomings, for the robot without a backwards moving capability.