Final Written Report

Chris Kleinknecht
John Paul Jones

EEL5666 IMDL

Dr. Schwartz
Dr. Arroyo

Adam Barnett
Mike Pridgen
Sara Keen

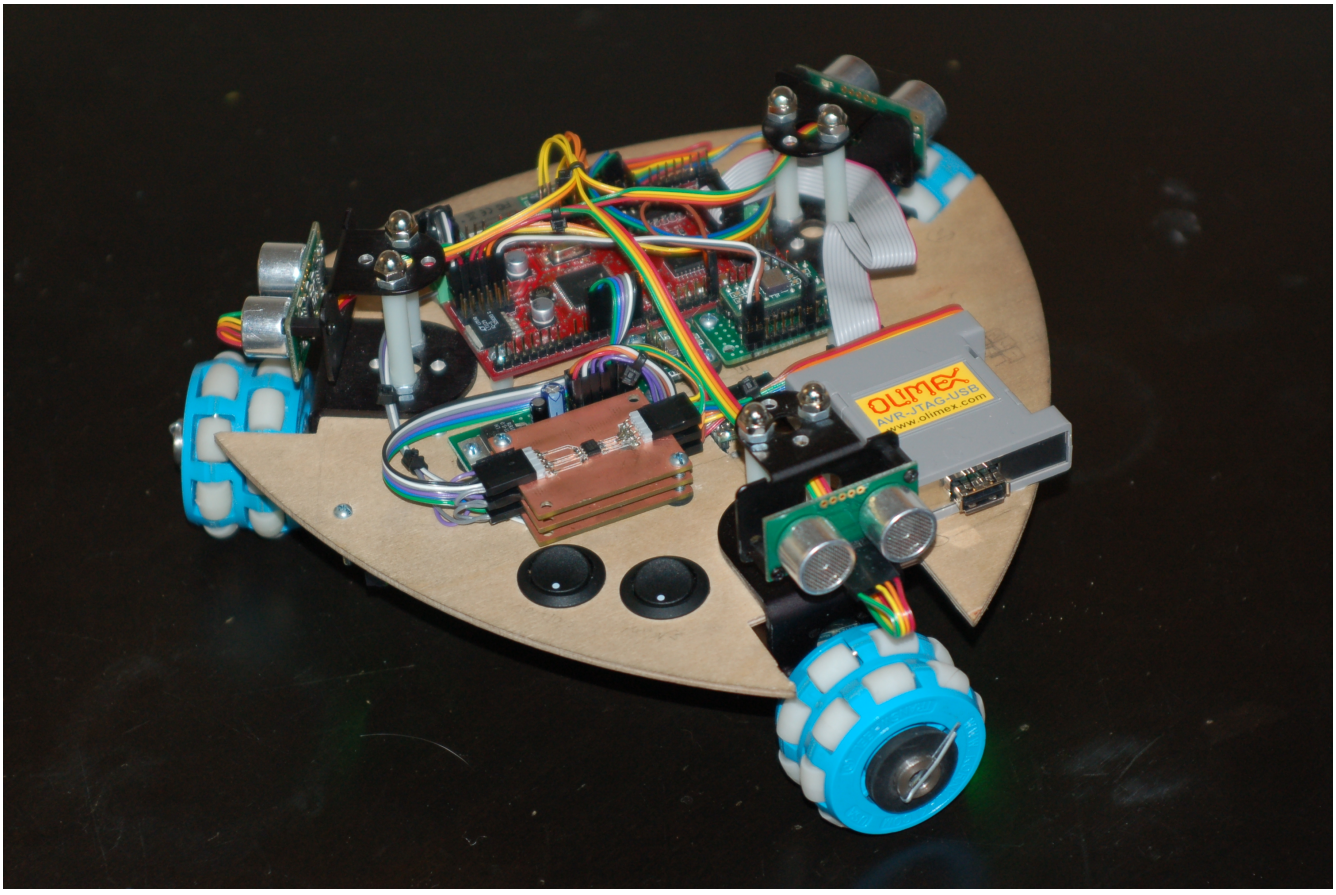


Table of Contents

Abstract

This report describes the goals and implementation of my robot, John Paul Jones.

The robot features a full holonomic drive system, with an inertial navigation system and three transwheels. The inertial navigation system includes one 2-axis accelerometer and a gyroscope. These sensors allow for independent control of (simultaneous) rotation and movement by continuously adjusting speed to each wheel based on the robot's direction and speed of movement, and direction and speed of rotation.

Three sonic range finders allow JPJ to perform basic obstacle avoidance, and evasion of (slow moving) predators. Each rangefinder provides 45° of coverage, the robot emulates 360° of detection by rotating about his central axis and recording sensor outputs on a timer interrupt, in effect making 3 full 360° sensor sweeps each rotation.

Executive Summary

John Paul Jones is capable of fully holonomic motion, and is able to rotate at any speed (hardware supporting) while driving in any direction. The robot uses multiple sonar modules to record information about the environment, and processes this data to execute a variety of behaviors.

The mobile platform consists of two identical radially-symmetric balsa boards. These boards sandwich most of the components of the robot. The batteries, motor driver boards, motors, and wheels are attached to the underside of the bottom board. Three transwheels allow for holonomic motion.

The only moving parts on the robot are the transwheels. By performing a simple trig lookup, and compensating for the current angle of the body, JPJ is capable of moving in any direction. By adding some scalar to each wheel speed, the robot achieves rotation.

John Paul Jones makes use of three types of sensors; 3 Devantech SRF05 sonar modules, 1 Analog Devices ADXRS300 gyroscope, and 1 Analog Devices ADXLS203 Accelerometer. The sonars ping continuously, and record and store the distance of objects from the robot on given angle intervals. The gyro is used to determine angular rate and, by integrating, angle of orientation. This information is used in the drive system and in the sonar loop. The accelerometer is not currently used in any behavior, but I am working to include it. Additionally, I am using a SparkFun BlueSMiRF Bluetooth module to communicate with my computer. While this use may not qualify the module as a sensor, future uses of the module may prove more sensor-like.

The robot is capable of (slow-moving) predator evasion, roaming and obstacle avoidance, complex predetermined motion, and user-assisted driving. The platform and software are open to expansion, and I will add more behaviors in the future.

Integrated System

Drivetrain

User or behavior sets 3 external global variables, a direction, speed, and rotation. The system compares the current direction to the command direction and converts the command into a direction relative to the coordinate frame of the robot. In the future, the accelerometer will give direction of momentum, and will act to check gyro drift in a closed loop control system.

The system then converts the relative direction command into wheel speeds via a trig lookup table, and scales them based on command speed.

Finally, the system compares the command angular rate to that reported by the gyro, and adjusts the rotation scalar.

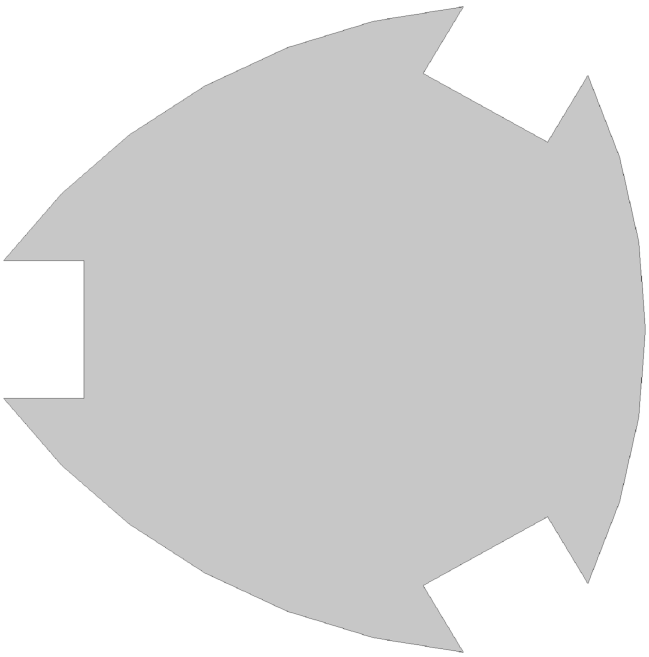
Sonar Loop

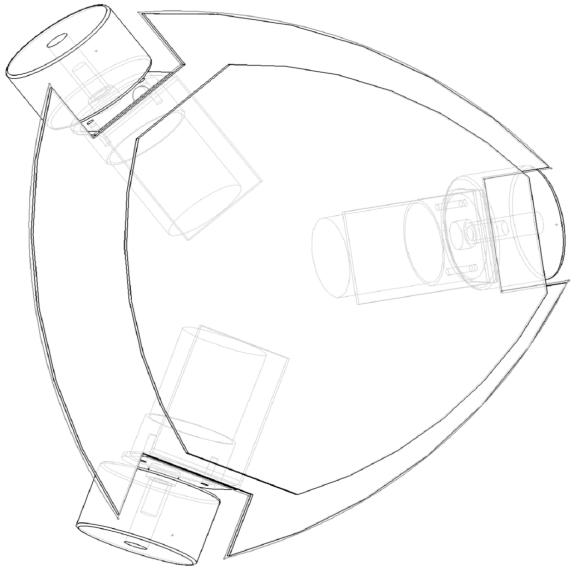
The three sonars fire sequentially, and continuously. At each recording the system checks the angle as reported by the gyro, adds the angle that the sonar is known to be fixed at, and stores the reading in an angle table at a corresponding angle interval. In the future, the accelerometer will store supplementary data, including the speed relative to the environment (not the robot) that a perceived object is moving.

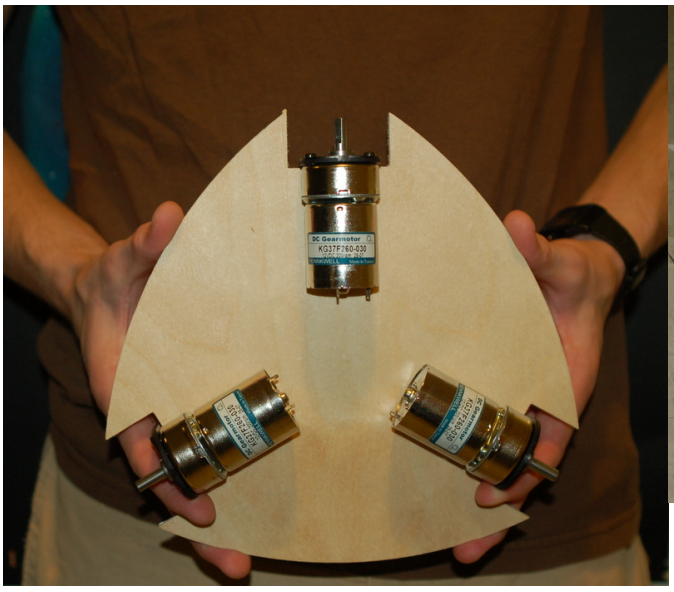
Mobile Platform

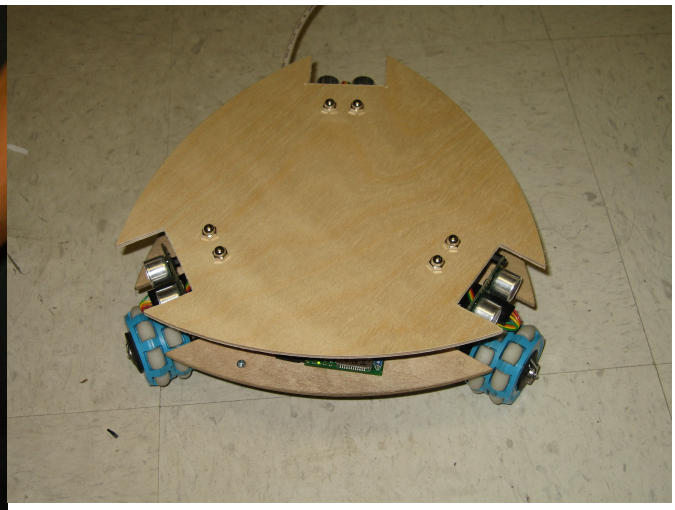
The mobile platform consists of two fixed boards. The bottom board meets the objective of holding the required hardware, and the top board meets the objective of supporting the flimsy bottom board.

The goals of the platform were to hold components and to provide a body to affix the wheels to. This platform met those goals admirably.









Actuation

John Paul Jones is equipped with a full holonomic drive system. In hardware, this system includes three transwheels, mounted tangentially. These wheels are the only moving part of the robot, and are sufficient to drive it in any direction. The objective was to facilitate orientation preserving and rotation-independent translation across the plane.

Each transwheel constrains the motion of the robot at the point that it is connected to the body. For every direction the robot is capable of moving there is a certain wheel speed ratio that preserves orientation. This ratio is given by the dot product of the direction vector, V, with each wheel angle relative to the body.

Rotation involves adding the same scalar to each wheel speed. Note that summing the corresponding wheel speed vectors results in the 0 vector, and does not translate the robot, only rotate it.

algorithm.

given:
a translation direction, cmd_dir (0° 359° (-180° 180°)),
a speed, cmd_speed (0 to 1),
and a rotation component, cmd_rot (-1 to 1; full CCW rotation to full CW rotation),

Read the immediate angle of the robot relative to it's starting position. This angle is determined by integrating gyro output. Subtract from cmd_dir to get the direction, rel_dir, to move relative to the coordinate frame of the robot.

(The next three steps are not executed in hardware, but rather in the trig lookup table. Rather than doing a lot of calculation every time the robot updates direction, I generated a 360 element array of scaled wheel speeds beforehand and stored it in memory.)

Convert the angle heading into a vector, i.e., define a V := [cos(rel_dir), sin(rel_dir)]

From V, determine the wheel speed ratio required to move in the direction of V. This is accomplished by taking the dot product of V and the vector of each wheel angle. For John Paul Jones, the wheel speeds were given as:

$w_1 = V \circ [\cos(0°), \sin(0°)] = V \circ [1, 0]$
$w_2 = V \circ [\cos(120°), \sin(120°)] = V \circ [-½, \sqrt{3}/2]$
$w_3 = V \circ [\cos(240°), \sin(240°)] = V \circ [-½, -\sqrt{3}/2]$

Scale each tuple of wheel speeds such that at least one element is 1 or -1. Do this by taking $w = w * (1 / |w|_{max})$ for each wheel speed w. This results in the wheel speeds that will take the robot in the direction of V as fast as it can go while preserving the wheel speed ratio. Note that this throws off the trigonometric nicety of moving the same speed in every direction. This results in non-constant linear speed while rotating or changing directions, but ensures that the robot can go as fast as possible.

Perform the trig lookup for the given angle cmd_dir to get the (scaled up) wheel speed ratio for that direction.

Multiply each by cmd_speed to scale the directional speed of the robot. In the event that this operation results in a motor going > 1 or < -1, scale each w (before adding rotation) such that max(|w|) + w_r = 1 or -1. In this way, rotation takes precedence over directional motion, and when the robot is moving at full rotation he cannot translate.

Generate appropriate pwms for the resultant wheel speeds

Apply a correction linear equation to the pwms, as they are not exactly proportional to the resulting wheel speed. These correction values were determined by experiment.

Sensors

Devantech SRF05 Sonic Rangefinder
http://www.acroname.com/robotics/parts/R271-SRF05.html

John Paul Jones uses 3 SRF05 sonar modules, one mounted above each wheel, to determine the relative distance of objects and predators in the environment. The sonars ping sequentially, at a regular interval. Early implementations used an external interrupt on each sonar's echo line, the current version uses Steven Buss' (http://www.cise.ufl.edu/~sbuss) timer interrupt based sonar code. At each ping, store the distance reported by the sonar. The position of the sonar relative to the coordinate frame compensated for the angle of the robot relative to the starting position as reported by the gyro. Do analysis on this table to determine which direction, and at what speed to run away.

Analog Devices ADXRS300(EB) Yaw Rate Gyro
http://www.analog.com/en/prod/0,,764_801_ADXRS300,00.html

The robot uses a single ADXRS300 on an evaluation board to determine angular rate and angle relative to starting position. This data is used in the drive system to correct rotational speed and convert directional commands relative to the environment into commands relative to the coordinate frame of the body. The gyro is also used in the sonar loop to determine what angle each sonar is facing when it pings.

The gyro gives an analog output proportional to the angular (yaw) rate of the chip. Using a modified version of Kevin Watson's (http://www.kevin.org/frc, http://www.kevin.org/frc/frc_gyro.zip) gyro interface code, the robot reads, converts, and integrates this output on a timer interrupt, giving angle of orientation relative to starting position.

Analog Devices ADXL203(EB) Dual-Axis Accelerometer
http://www.analog.com/en/prod/0,2877,ADXL203,00.html

Had I gotten it working in time, the accelerometer would have been used to complete the robot's inertial navigation system. I've succeeded in integrating accelerometer output on a timer interrupt and displaying the robot's velocity along the x and y axes, though the output is not consistent enough to use safely.

Over the summer I intend to include the accelerometer in the robot. I will use the accelerometer to compensate for gyro drift, maintain a constant speed, and take the second integral to estimate displacement for use with holding patterns and other behaviors.

Future plans include making use of the calibration and self-test pins on the gyro, and getting the accelerometer to work.

I am grateful to Analog Devices for supplying me with sample evaluation boards for the ADXL203 and ADXRS300, and recommend ADI to anyone working on a similar project.

SparkFun BlueSMiRF Bluetooth Module
http://www.sparkfun.com/commerce/product_info.php?products_id=582

I am using a SparkFun bluetooth module to handle serial communication in place of an lcd. While this is not a sensor in the typical sense, I intend to use it later to communicate with other bluetooth enabled devices, including other robots

Behaviors

Predator Evasion

Unfortunately, JPJ is not fast enough to avoid predators. The predator evasion algorithm works in theory, but the current sonar implementation is too slow to avoid moving objects.

algorithm:
- maintain a holding pattern
- ping sonars continuously
- in the event that sonars detect an object closer than some threshold:
  - drive away at a speed inversely proportional to distance of object
  - continue to drive until object is out of range
  - if another object comes within range before the first I out of range:
    - take average of escape direction vectors
    - drive in that direction

Roaming / Guided Travel

The same principles that allow for predator evasion can be adapted to allow JPJ to roam and avoid obstacles. While the code is similar to that of predator evasion, the behavior appears very different.

algorithm:
- ping sonars continuously
- determine the closest object
  - take average of current direction vector, and vector opposite closest object
  - drive in that direction

Holding Pattern

Using the gyroscope and time delays, the robot can execute complex predetermined holding patterns. This behavior could be improved with the inclusion of the accelerometer, to give distance traveled.

Command Drive (non autonomous)

The robot is able to take user commands for direction, speed, and rotation, and drive.

Future Behaviors

The hardware is not specialized, and I built the robot to be extensible. I will add more behaviors in the future. As I demonstrated on demo day, modifying JPJ to charge nearby objects instead of fleeing from them is as easy as connecting the sonars out of order.

Conclusion

Although my robot performs well, it is not complete. I have accomplished a substantial amount, but I have not completed the work I intended to.

I am not done working on the robot, and there are significant limitations of the current design. The drive system doesn't incorporate the accelerometer, and the current sonar implementation is too slow to facilitate predator evasion. The motors provide an unnecessary amount of torque, and I will gear them up in the future.

John Paul Jones' drive system exceeded my expectations. The robot can move in any direction, while preserving orientation, using a single gyro chip. I expected that the robot would drift uncontrollably without feedback from the accelerometer. I am still impressed with how well the gyro works.

The drive system can still be improved by the inclusion of the accelerometer. Also, I intend to improve existing behaviors, and add new behaviors. Once the accelerometer is working I will be able to include speed and distance-traveled specific actions, including path correcting and tracking behavior.

I am satisfied with the design of my robot, and would not change any of the specifications.