Autonomous Exploring Robot with Mapping

Group 6

Scott Hewson

Braedan Jongerius

Michael Jun

Summary:

Autonomous robot implementing obstacle navigation and 2D mapping.

Table Of Contents

1. Abstract

The goal of this design project is to build an autonomous robot that will avoid obstacles and build a 2D map of its environment. This functionality is achieved by mounting the DE0 Nano FPGA board on the Rover 5 robot platform and interfacing with a motor driver board, a digital compass, and four ultrasonic rangefinders. The Rover 5 platform contains four motors in two rubber tracks and each motor has a quadrature encoder measuring wheel rotation. The robot moves by sending pulse width modulation signals to the motors through the motor driver board. The quadrature encoders provide interrupts to determine the position of the robot. Range finders measure the distance to obstacles in front of and beside the robot. If something is in front of the robot it will turn to a clear direction and continue. The digital compass provides an accurate heading for the robot and facilitates accurate turning. The positions of the robot and obstacles are stored as data points which are output to a computer and plotted to produce a map of the environment. A switch is used to start and stop the robot and a button is used to trigger the robot to output the mapping data.

2. Functional Requirements

2.1 Requirements

- The robot will explore its environment autonomously.
    - The robot will continue driving until switched off or otherwise disabled.

- The robot will avoid obstacles.
    - When an obstacle is 10 inches of the front of the robot it will turn to a clear direction and proceed.

- The robot will produce a 2D map of its environment.
    - This is achieved by taking the raw data from the encoders, digital compass and ultrasonic range finders to convert the raw (scalar data) into vector (x ,y) data points through a series of translations
    - When connected to a computer the robot will output the mapping data points which will be used to plot a map.

2.2 Results

- The robot drives autonomously and continues until disabled

- The robot correctly avoid obstacles, turning left or right when appropriate without any collisions.

- The robot generates a map however there are some inaccuracies in the data. This can be due to range finder data fluctuation and interference, compass interference, or encoder discrepancies. In the future, this may be minimized by further calibration and code optimization.

3. Design and Description of Operation

3.1 Description of Operation

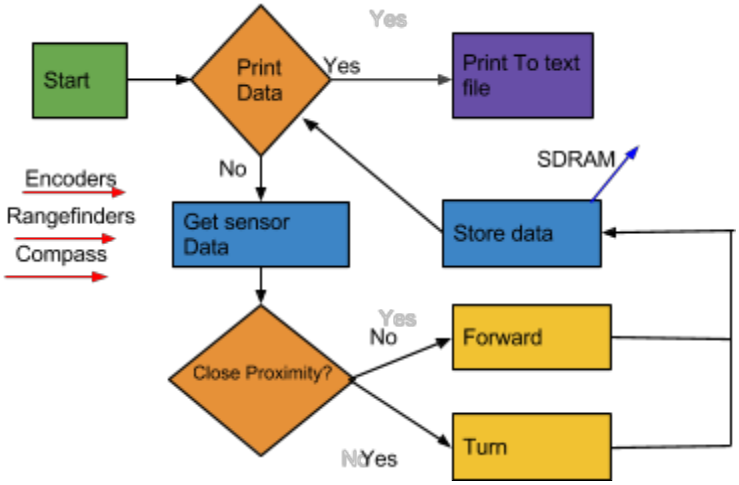

Fig. 1. Operation Flowchart

Sensor Data

Rangefinders->Report an float value indicating the distance from the object
Encoders->Reports the number of times the wheels rotated
Compass->reports a value indicating heading

The program constantly reads theses programs under a short interval (300 ms for our project). The rangefinder data is monitored to make sure that the values reported are all higher than our minimum distance (10 inches). If values are reported below that the program will then send a command out to the PWM, controlling the motors to turn the robot in a direction to avoid the object.
ie: front values<10=>turn left or right

The encoder values are converted and taken as an average to report the distance traveled. This value then used with the compass heading to update the position of the rover. The left and right range finder data is then also translated from the current position to represent the X,Y data point locations of the object in the way.

Specified address are referenced as a double pointer, holding the pointer for a malloc call of where the data points will be stored. This data is then freed once the data output is initialized.

Vectors are translated x,y data points based off the current position and the distance to obstacles.

vector.x = cos((angle) * PI / 180) * distance + position.x;

vector.y = sin((angle) * PI / 180) * distance + position.y;

The robot is set to continuously loop through the process above until the control switch is turned to the off position. In the off position the program will poll the button. The button enables the extraction of the data output into a txt file but must be run in debug mode; Nios will not allow otherwise.

3.2 Hardware Design

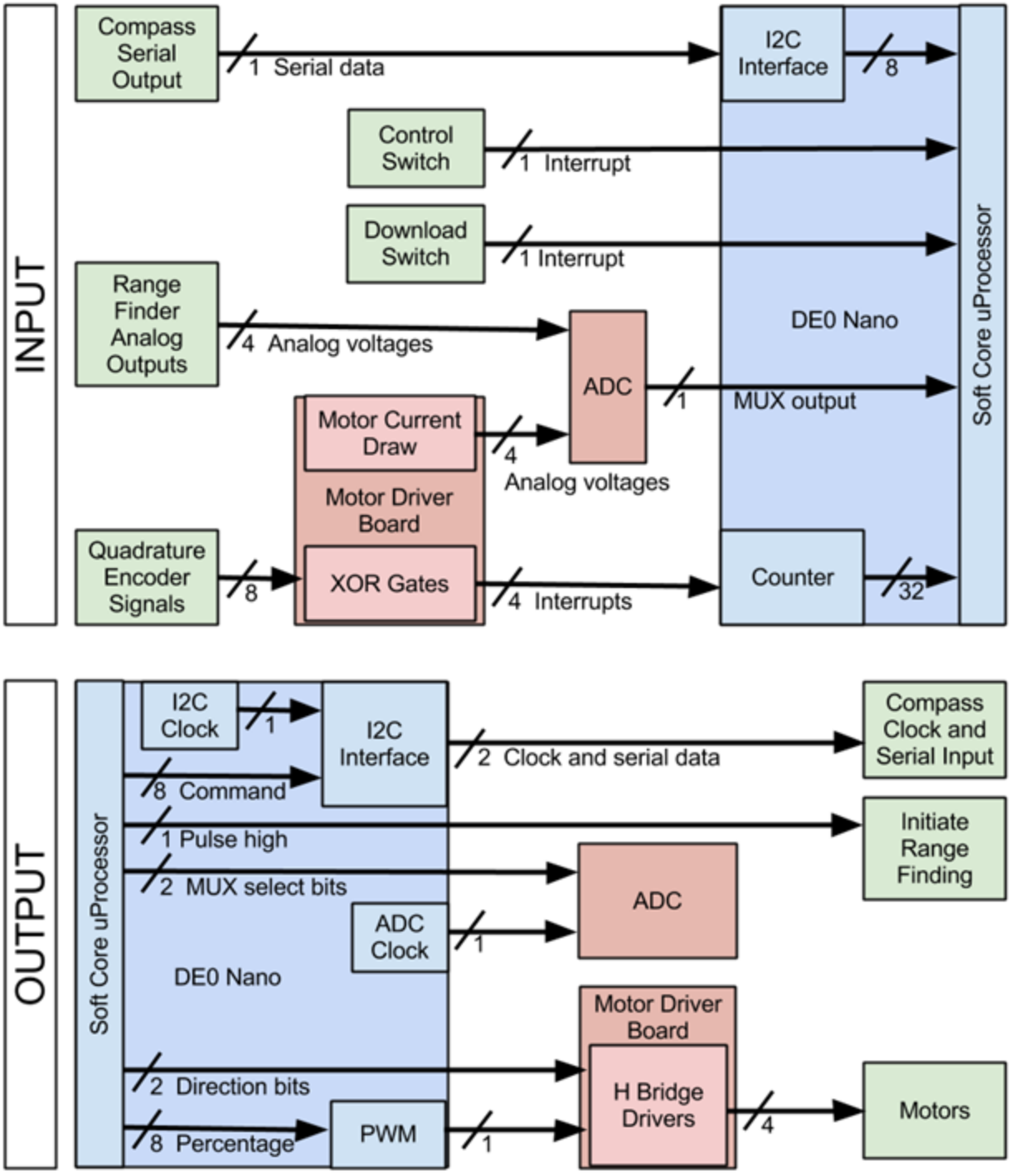

Fig. 2. I/O Diagram

Hardware is laid out on two perf boards stacked vertically on top of the robot. 2x20 pin and 2x13 pin ribbon cables connect to the GPIO and ADC on the DE0 Nano, respectively. A 2x10 pin ribbon cable connects the two perf boards together.

Motors are controlled by one PWM signal and two direction signals. The PWM module generates the PWM signal with the desired duty cycle. This signal is sent to the PWM input pin for each motor on the motor driver board. The direction pins are set by the processor and sent to the direction input pins on the motor driver board. The motors on each side of the robot receive the same direction signal. The motor driver board uses the direction and PWM signals to move the motors in the desired direction and speed.

The encoders send their quadrature signals to the motor driver board which XOR each pair together to produce a single interrupt signal with twice the resolution. The four interrupts each pass through a voltage divider to drop the voltage from 5V to 3.3V, the max signal voltage for the DE0 Nano. The voltage dividers use 20kΩ and 10kΩ resistors.

The current draw of each motor is output from the motor driver board as an analog voltage. The voltage is limited to a maximum of 3.3V before sending into the ADC on the DE0 Nano to protect the chip from damage. We initially used 3.3V voltage regulators, but it is not recommended as the regulator behaviour is not linear below the output voltage. We planned to use clipping diodes with limiting resistors but did not have time to implement them.

The range finders are mounted near the front of the board and set back from the edge to offset the 6 inch dead zone as much as possible. The range finders need to be configured to take readings in series so they do not interfere with each other. This also allows us to trigger a measurement from each sensor with only one read signal. The BW pin on each range finder is held high to put the rangefinder in chaining mode. The processor sends a high pulse out from GPIO into the RX pin on the first range finder. The range finder takes a measurement and sends a high pulse from its TX pin to the next range finder's RX pin to trigger it to read. This process repeats until all the range finders have taken a measurement. The range finders output their last measurement on an analog pin. These four analog signals are sent into the ADC.

The compass is interfaced with our I^2C module. It takes a clock signal and a bidirectional data signal from the GPIO. There is a 0.01mF capacitor between power and ground and 18kΩ pull-up resistors on the clock and data lines. The compass is configured to run in continuous mode, where it continually outputs the heading. This way we do not need to trigger every read.

Power is supplied by ten 1.2V AA Ni-MH rechargeable batteries. Six batteries supply 7.2V to the motors and 4 batteries supply 4.8V to the DE0 Nano and the motor driver board. The range finders and compass are powered by the 3.3V supply on the DE0 Nano, and the quadrature encoders are powered by the 5V supply on the motor driver board. The batteries have a runtime of over thirty minutes.

3.3 Software Design

Our design runs on the uC/OS-II operating system. Two task are created to manage our program and queue is implemented for communication of data between the two task.

Task 1:
This task is responsible for polling all external I/O: the rangefinders, compass, encoders, the switch and the button. This data is then sent into a queue and pauses for 300 ms. Externally this function can be changed to call a test function to test the I/O.
If Switch is off and button is pressed this will output the data to a txt file.
**However this must be run in debug mode or it will not work **

Task 2:
This task is responsible for receiving, processing data, controlling PWM, and storing data.
The program will pend until there is data in the queue. Once there is data, the rangefinder data is checked for minimum distances. If the front sensor report a low value (below our min threshold), it will then turn left or right (depending on which value is higher) to avoid the objects.

Turning:
Method 1 Compass) the turn command is issued recording initial angle position. This command is kept until a net difference of 90 degrees has been confirmed.
Method 2 Encoder) designed for area where the compass information will prove to be unreliable the rover will monitor the encoder distance until a specific value (25.1 inches for 90 degrees) is met.
NOTE: MUST BE HARD CODED INTO TO SWITCH FROM EACH OTHER

The data is then converted into the X,Y cartesian plane with the use of the compass and scalar value reported for each sensor. A dynamic pointer is called to hold theses data points and the address of this pointer is then stored into a user specified location.

After this process the task will then once again pend on the queue for new data.

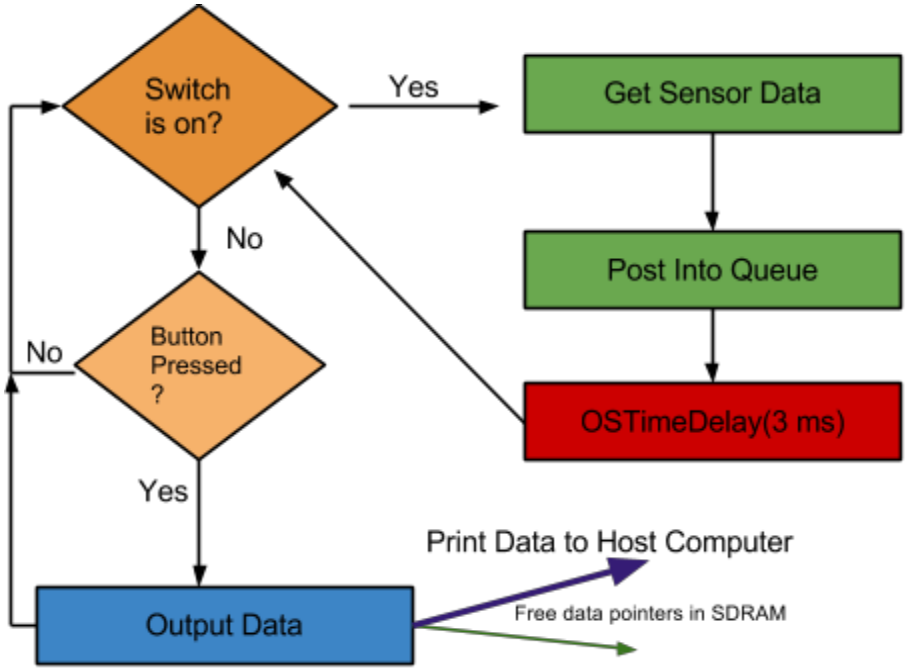

Fig. 3. Task 1 flowchart

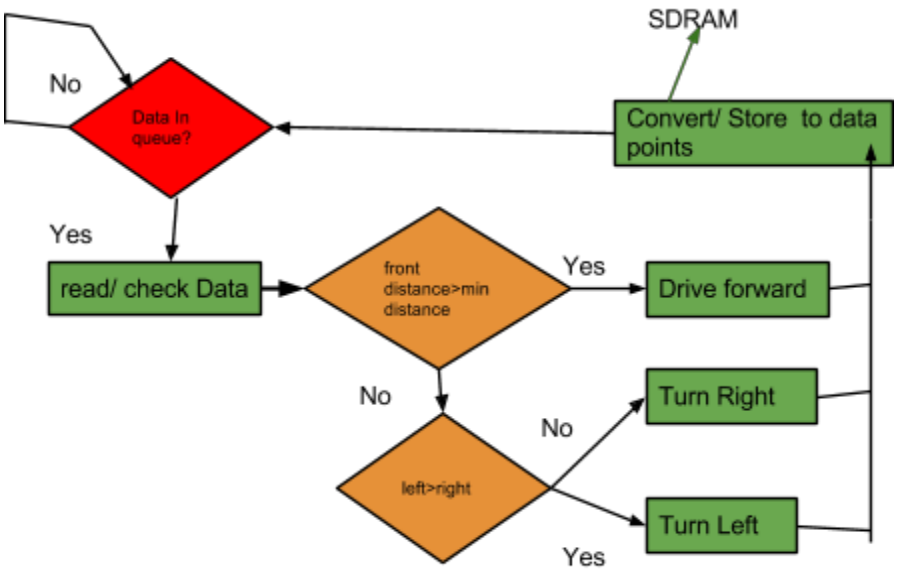

Fig. 4. Task 2 flowchart

4. Bill of Materials

Parts	Quantity	Supply	Cost	Weblink	Datasheet
Altera DE0 Nano FPGA board Platform for connections and contains the operating program	1	Lab supplied	$59.00	http://www.altera.com/education/univ/materials/boards/de0-nano/unv-de0-nano-board.html	http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=593&FID=75023fa36c9bf8639384f942e65a46f3
Rover 5 Platform Vehicle with 4 motors, each with a hall-effect quadrature encoder and gearbox.	1	Sparkfun	$59.95	https://www.sparkfun.com/products/10336	http://www.sparkfun.com/datasheets/Robotics/Rover%205%20Introduction.pdf
Rover 5 Motor Driver Board H-bridge driver board with quadrature signal mixing	1	Sparkfun	$24.95	https://www.sparkfun.com/products/11593	http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Robotics/4%20Channel%20instruction%20manual.pdf
Maxbotix LV-EZ4 Ultrasonic Range Finder Sensors that will be attached to the car. Operates from 0 to 6.54 m	4	Sparkfun	$27.95 (x4) = $111.80	https://www.sparkfun.com/products/8504	http://maxbotix.com/documents/MB1040_Datasheet.pdf
HMC6352 Compass Module Digital compass using I2C	1	Sparkfun	$34.95	https://www.sparkfun.com/products/7915	http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Magneto/HMC5883L-FDS.pdf

Right-angle Headers 1x7, 0.1” spacing	4	Mouser	$1.48 (x4) = $5.92	http://ca.mouser.com/ProductDetail/Molex/90121-0767/?qs=%2fha2pyFaduhgA%252b2vT5VnawRueORA5Z4xaQiBDRNOjy%2fq4WItMFJFrw%3d%3d	www.mouser.com/ds/2/276/0901210767_PCB_HEADERS-153061.pdf
Female Socket 16 pin	1	Mouser	$3.98	http://ca.mouser.com/ProductDetail/FCI/66951-016LF/?qs=sGAEpiMZZMs%252bGHln7q6pmxAVkKtOEC39tVCSOjPJMEU%3d	http://www.mouser.com/ds/2/154/66951-85357.pdf
Female Terminals	4	Mouser	$0.18 (x4) = $0.72	http://ca.mouser.com/ProductDetail/Molex/16-02-0097/?qs=%2fha2pyFaduj8V9nDkfZZhtw651Jb96MbGQkv6WmaPhVPl%2fuAGoju7A%3d%3d	http://www.mouser.com/ds/2/276/0016020097_CRIMP_TERMINALS-158075.pdf
Misc. Resistors and Capacitors	16	Lab supplied	$0.02 (x16) = $0.32		
Perf Board	2	Lab supplied	$7.50 (x2) = $15.00		
Standoffs metal, 1-inch	12	Lab supplied	$0.40 (x12) = $4.80		
Straight Header 1x4, 0.1” spacing	1	Lab supplied	$0.15		
Battery	3	Lab supplied	$1.95 (x3) =		

Holders 4 AA and 2x3 AA			$5.85		
Switch SPDT	1	Lab supplied	$0.50		
Button push button	1	Lab supplied	$0.35		
Ribbon Cable 2x20, 2x13, 2x10, 0.1” spacing	3	Lab supplied	$4.95 (x3) = $14.85		
Straight Headers for Cables 2x20, 2x13, 2x10, 0.1” spacing	3	Lab supplied	$0.30 (x3) = $0.90		
Screws	9	Lab supplied	$0.02 (x9) = $0.18		
Nuts	6	Lab supplied	$0.02 (x6) = $0.12		
Misc. Consumables Wire, solder, glue, velcro, etc.		Lab supplied	$5.00		
TOTAL COST			**$374.24**		

5. Sources of Reusable Design

Hardware Component Code: 2012 Altera Corporation - From SOPC Builder

PWM Avalon glue - Application Notes from 2012's group 1

ADC - VHDL and C code - Altera DE0-Nano Demonstration CD

Additional I2C VHDL - Altera DE0-Nano Demonstration CD

6. Datasheet

6.1 Power Calculations

Standby

- DE0, driver board, encoders,range finders and compass: $5.092V * 11.57mA = 58.914mW$
- Motors: $7.2V * 0.01mA = 0.072mW$
- Total: $58.914mW + 0.072mW = 58.986mW$

Full speed

- DE0, driver board, encoders,range finders and compass: $5.092V * 11.57mA = 58.914mW$
- Motors: $7.2V * 1.194A = 8.5968W$
- Total: $58.914mW + 8.5968W = 8.655714W$

6.2 Operating Conditions

The robot can operate indoors and outdoors. When operating indoors, ensure that the robot is as far away from sources of magnetic interference as possible. The compass is sensitive to magnetic fields and can easily be interfered with from metal objects. When operating outdoors, ensure the weather is clear and dry. The robot has no shielding to the elements and is very susceptible to damage. The robot should also be used on relatively flat ground as steep angles can cause the compass to malfunction. In either situation, the rangefinder may not detect sound-absorbing materials, such as foam or heavy carpet.

6.3 I/O Signals

Type of Signal	Name(s)	Description	Voltage Requirements	Pins
Electronics	Motors	Motors that move the Rover 5	7.2 V (powered by AA batteries)	-
Electronics	Driver board	Rover 5 Motor Driver Board	5.0 V (powered by AA batteries)	-
Electronics	DE0-Nano	The DE0-Nano FPGA	3.3 V (powered by AA batteries)	-
Electronics	Ultrasonic range finders	The Maxbotix LV-EZ4 ultrasonic range finders	2.5 - 5.5 V	-
Electronics	Compass	Digital compass for heading	2.5 - 5.0 V	-
Electronics	Quadrature encoders	The encoder on each wheel used to calculate the robots position	5.0 V	-
Electronics	Control switch	Control switch to enable operation	3.3 V	-
Off-board	left_pwm right_pwm	PWM signals to Driver Board	0.0 - 3.3 V	2 GPIO (output)
Off-board	left_dir right_dir	Direction signals to Driver Board	0.0 - 3.3 V	2 GPIO (output)
Off-board to FPGA	range_left range_right range_front_left range_front_right	Range finders analog input	0.0 - 3.3 V	4 analog GPIO (input)
Off-board to FPGA	compass	GPIO pins for I2C interface with compass	3.3 V	2 GPIO (clk - output, sda - bidir)
FPGA to board	initiate_sensors	GPIO pin to initiate range sensors	3.3 V	1 GPIO (output)
Off-board to FPGA	encoder_front_left encoder_front_right encoder_rear_left encoder_rear_right	Encoder pulses from motor rotation	3.3 - 5.0 V	4 GPIO (input)
Off-board to FPGA	enable_button	Control switch	0.0 - 3.3 V	1 GPIO (input)

		signal to enable operation		
Off-board to FPGA	data_button	Control button to dump data into text file	0.0 - 3.3 V	1 GPIO (input)

7. Background Reading

Our project design and it’s components closely resemble the project in the "Feasibility study of FPGA based Real-Time controller for autonomous vehicle applications" [5] paper.

“Radio Controlled Rover 5” [6] and “Rover 5 Robot” [7] are other very similar projects that both deal with controlling the Rover 5’s motors and the latter project which heavily uses ultrasonic sensors.

8. Test Plan

The individual components are tested through the C file “test.c”.

Range finders:
Polling the sensor while stationary and measuring objects at various distance as well as value fluctuation.

Encoders:
The robot is placed in a stationary position. Motors are spun individual and in combination while monitoring the encoder values.

Motors:
Various test and simulation are ran, set up through the software of the possible movement and turning options (left, right, forward, reverse) to ensure proper functionality

Motor Speed:
The robot is driven in a straight line at varying speeds.

Avoidance Test:
The robot drives, reads sensor data, and turns to avoid obstacles.

Compass Test:
The compass heading is monitored as the rover is rotated. The compass calibration mode is called to fix any serious issues.

Turn Test:
The robot is calibrated to turn 90 degrees by monitoring compass and encoder readings.

9. Results of Experimentation

Rangefinder distance fluctuation: +/- 0.15 inches

Encoder distance reporting: The values are very accurate when the robot is moved by hand. Deviations are seen when the robot is moved by the motors.

Motors: Due to the rover 5's wheels not being straight, the left motors must run at 81.5% to drive in a straight line.

Turning with encoder data: Moving 25.12 inches was found to give a fairly accurate turn for 90 degrees. However, the range of angle did vary between 85-95 degrees, leading high in-accuracy over several turns.

Compass: When compass is indoors or near high metal material, the compass needs to be recalibrated often for its surroundings.

Runtime: Since our motors draw the most power, they are our time limiting device. When in full speed operation, they use $8.5968W$. The motors are supplied by six 1.2V 2650mAh batteries.

$$6 * 1.2V * 2650mAh = 19.08Wh$$
$$19.08Wh / 8.5968W = 2.21943h$$

Using the following calculations, we should theoretically be able to run our robot for just over 2 hours on fresh batteries. While building and testing our robot, the longest we ran the robot before charging the batteries in preparation for the next week (not because they were dying) was approximately 45 minutes.

10. References

[1] T. Bonar. (2012, July 11). Using Multiple MaxSonar Sensors [Online]. Available: http://www.maxbotix.com/articles/031.htm

[2] Bob. (2011, April 7). Rover 5 Motor [Online]. Available: http://roboticcore.com/?p=73

[3] Group 1 (CMPE 450 2012), “iOS Device Controlled RC Car Capstone Project”, Edmonton, AB, 2012

[4] http://www.sparkfun.com/datasheets/Robotics/Rover%205%20Introduction.pdf

[5] Chan, KimChon. (2010, November 20-21), Feasibility study of FPGA based Real-Time controller for autonomous vehicle applications [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5687010&isnumber=5686981

[6] Duane Degn. (2013, February 24), Radio Controlled Rover 5 [Online]. Available: http://letsmakerobots.com/node/36276

[7] RobotBASIC. (2012, August 13) Rover 5 Robot [Online]. Available: http://letsmakerobots.com/node/33794

12. Appendix

12.1 Quick Start Manual

External Interface:
3 Battery Packs: 10 AA batteries are required to fully power the robot.
Switch: Switches between the Idle task of polling for button to output Data, to Data Polling/Driving
NOTE: *DATA RESETS WHEN SWITCH IS ON*

Button: Press to output the recorded data onto the host computer.
****MUST ENSURE DEBUG MODE, CONNECTION, SWITCH IS OFF BEFORE PRESSING****

FLASH:
The flash version is designed for obstacle avoidance, and will not record mapping information
In order to flash updated software must be used or requirement of a software patch is required order to reflash successful..

Setup:
Generate SOPC File with EPCS, set exception vector and interrupt vector to epcs
Top level: Set The following pin assignments:

1. Compile SOPC and SOF, and program the device
2. Open the project in Nios Eclipse
3. Open the flash programmer
4. Flash device as per normal

Run:

1. Set Power On
2. Set Switch to engage avoidance mode
3. Set switch to off to disengage avoidance mode

Non flash:
Similar to the previous steps with the exception that CPU exception vector should be not set to EPCS and flash programmer does not have to be used. Hit Run.

12.2 Future work

Backtracking:
When the robot makes a turn in a new location it will create a node containing the location of the turn, the direction the robot came from, and the direction the robot went. Whenever the robot makes another turn, it will check if there is a node at its location. If so, it will check the data and turn in a new direction if possible.

Save mapping data to removable storage:
The robot will write mapping data to an SD card, or other removable storage. This will allow the map to be obtained more easily.

Stream mapping data wirelessly
The robot will stream the mapping data in real time over a wireless connection to a computer.

12.3 Hardware documentation

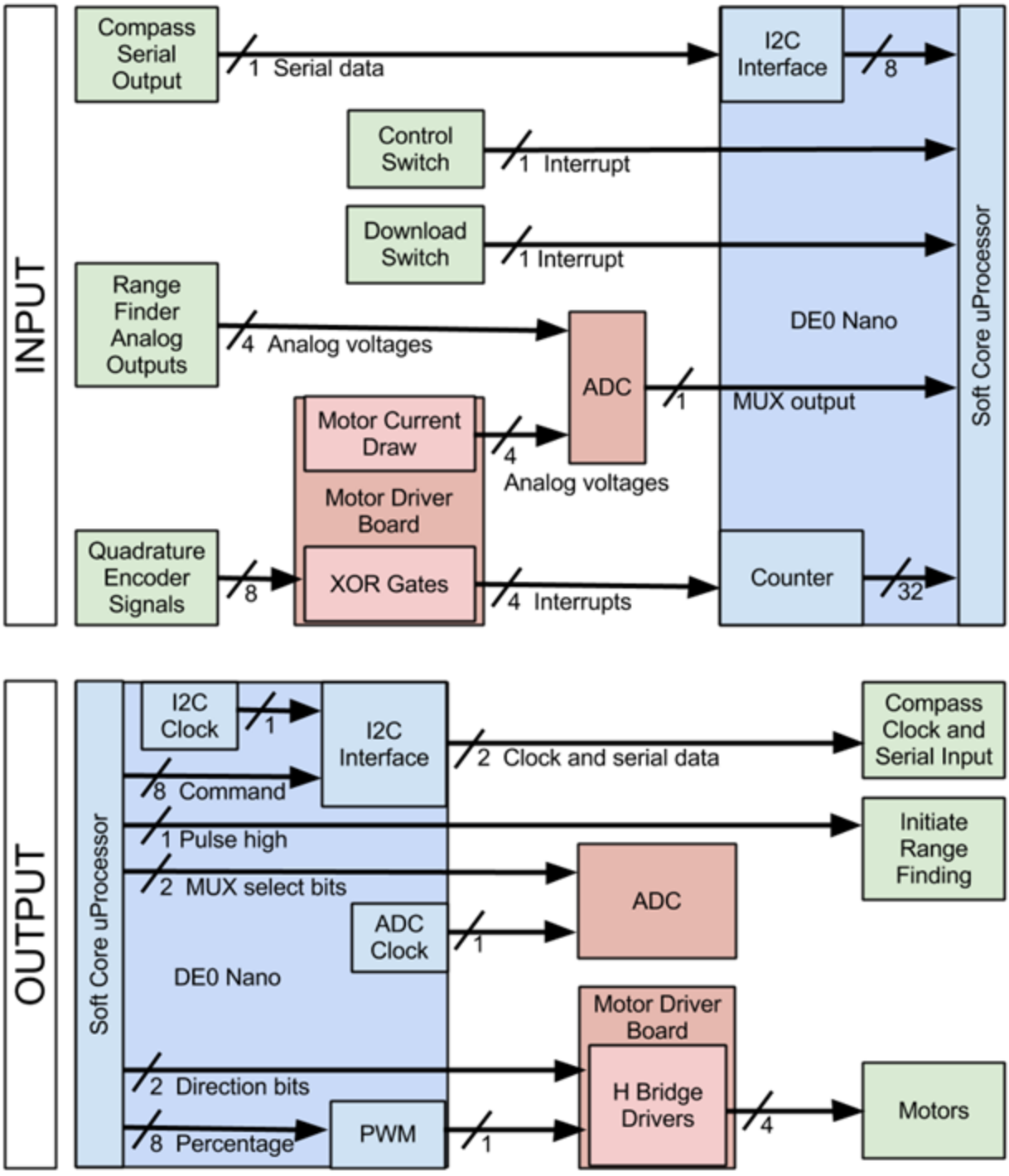

Fig. 5. I/O Diagram

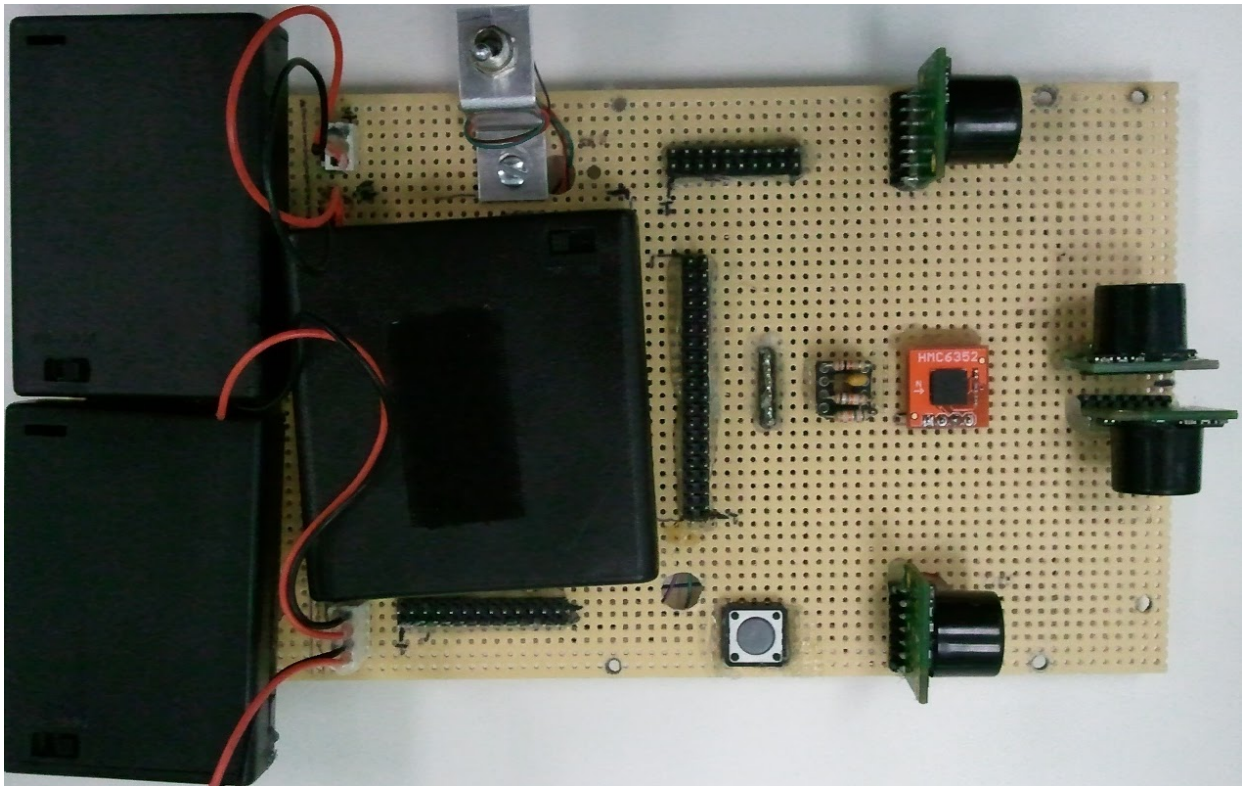

Fig. 6. Layout of upper perf board

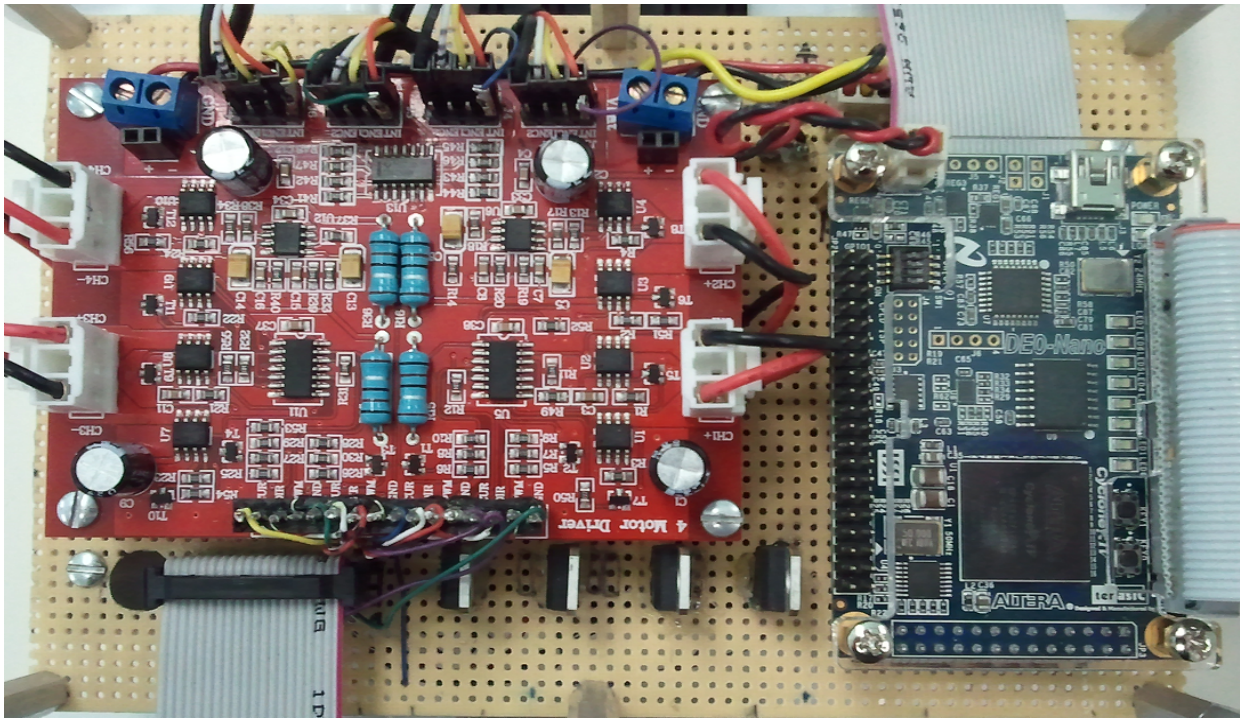

Fig. 7. Layout of lower perf board

12.4 Source Code

Source code can be found on our GitHub repository:
https://github.com/BraedanJ/AutonomousExplorer

C files:
Explorer.C - Main C file, Gathers Sensor Data, Process Data, Store Data, Output Data
Explorer.H - Header file of the functions, shared DEFINE are listed here
Sensor.C/H - This file is contain all pertaining to Rangefinders or Compass
Rover.C/H - The Rover 5 PWM and Encoder Functions
Map.C/H - Contains Processing Data and Conversions

VHDL/Verilog Files:
encoder.VHDL - Avalon interface to the input encoder data
PWM.VHDL - Avalon interface to export the pwm signal