Drivable & Semi-Autonomous Robot with DSPIC Microprocessor

Ajay Curam & Brian Josefowicz, CmpE 150 Spring 2017

***Abstract*—The objective for this project is to design a vehicle that navigates to a desired GPS location through autonomous control systems. This document explains the design process, materials, testing analysis, and system integration used to achieve the desired results. The purpose is to provide enough details for any developer to replicate or augment the design for use in their own project.**

I. INTRODUCTION

Autonomous vehicle control is an evolving field in robotic applications which means, "Given a desired command, the machine can perform the task free from human interaction." The user shall command the robot to execute a series of instructions, and the robot will respond by completing the process. Here, the requirement is to command a 4-wheel electric truck to navigate itself to a specified GPS location without user interaction.

Several peripheral devices are used to enhance the functionality of the vehicle adding the capabilities: coordinate commands via Bluetooth, current location and rough heading via GPS module, and bearing from true north via magnetometer.

Furthermore, an Electronic Speed Controller (ESC) drives the powertrain motors and steering servo. To interface these devices, an embedded systems microcontroller (μC) is programmed to allow full system functionality. Overall, a complex system can be encapsulated using these simple components.

II. DESIGN METHODOLOGY

Begin by understanding what the requirements of the project are: GPS, vehicle, and autonomous. Move on by performing research on each of the design constraints. Once there is enough documentation, table out the necessary pieces of hardware required to meet these criteria. During this process, it is important to ensure that your options are compatible, available, and cost-effective. Once decisions have been made on what works together, design your system by creating a pinout and overall system diagram. It is necessary to note which communication protocols will be used on each interface between the μC and the respective device. Designing the

system architecture requires an understanding of the digital circuitry which make up the system datapaths and how each register is programmed to produce the needed functionality.

This autonomous vehicle was designed such that it can receive its location from satellites via a GPS module over UART, parse the necessary values and compare them to a user input desired location. The vehicle receives this user input through a blutooth module over UART which allows the user and vehicle to exchange data. All data I/O for the vehicle is written and read through the SRAM module. Once the PIC has discovered this information, it then begins to activate its control systems.

The control systems consist of generating a PWM signal, and feeding it into the onboard Electronic Speed Controller (ESC). There are two inputs on the controller - one for the steering servo and another for drive motor control. As the GPS continues to update itself, the PIC continues to compare this data to the desired location and adjusts the steering servo left or right while telling the motors to drive fast or slow. Eventually the car reaches its destination, and the task is complete.

While the GPS module provides heading information, it proved to be inaccurate on the small scale application. To account for this, we interfaced a magnetometer with the PIC over SPI. The purpose of the magnetometer is to provide true north detection and bearing from that point. This allows the vehicle to always have a reference point so that when it is turned off, it can sense which direction it is facing. [Not implemented due to time constraints]

Overall, the system is fully autonomous, and could use additional systems like: edge or object detection, object recognition, and a wireless internet connection. These would allow the car to: steer away from impeding objects, determine what it actually can see, and transmit/update relevant data as necessary.

A. Parts List

- DSPIC33FJ128MC802 & Programmer
- Breadboard & Jumper wires
- Traaxas RC Monster Truck
- Adafruit Ultimate GPS Module & Antenna (UART)
- Microchip Bluetooth Module (UART)
- Magnetometer (SPI) [Not implemented]
- Microchip serial SRAM module (23A640)
- Header pins & Solder
- Computer with Bluetooth 2.0

- USB to TTL to RS232 Cable
- USB Battery Bank

B. Original and Derived Equations

Provided are the different equations and their calculations made to solve certain challenges faced while developing the C-code.

$$D = \frac{PW}{T} \times 100\%$$

Equation 1: Duty cycle formula used for PWM signal

$$d = r\,\Delta\sigma. \qquad \Delta\sigma = 2\arcsin\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 \cdot \cos\phi_2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)}.$$

Equation 2: Great Circle Distance Formula (Haversine Formula, where r = Radius of the Earth * Δσ, where φ = latitude λ = longitude. d = distance between 2 points.)

C. Schematics

See Appendix A for algorithmic state machine
See Appendix B for system schematic

III: TESTING PROCEDURES

1. Taking the system diagram and μC pinout, breadboard the system for testing and analysis.
2. After the hardware setup, begin analyzing the firmware requirements for each device. This will allow a proper configuration for all peripherals. The firmware development cycle should be documented along revisions.
3. Once the firmware is complete, and proper communication is established with peripherals, it is time to begin implementing the control systems software. This is done to ensure all robotic responses to certain inputs are executed as expected.
4. Test results should be analyzed and documented during this phase until the results are as expected.

IV: TESTING RESULTS

Stage 1 testing consisted of interfacing with peripheral modules.

In the first series, the GPS module was connected to the PIC UART channel. After powering on the device, a logic analyzer was connected to the data line transmitting to the PIC in order to detect the frequency and data packets. Once the packets were visible as seen below, a parsing function was written in order to extract the necessary data for writing into SRAM.

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
```

Figure 1. GPS Sample data GPGGA

Second, the bluetooth module was then connected to a separate UART channel, and powered on. The module was received by the local computer, and was able to receive commands sent through a local command line prompt. The message prompts can be seen in figure 2.

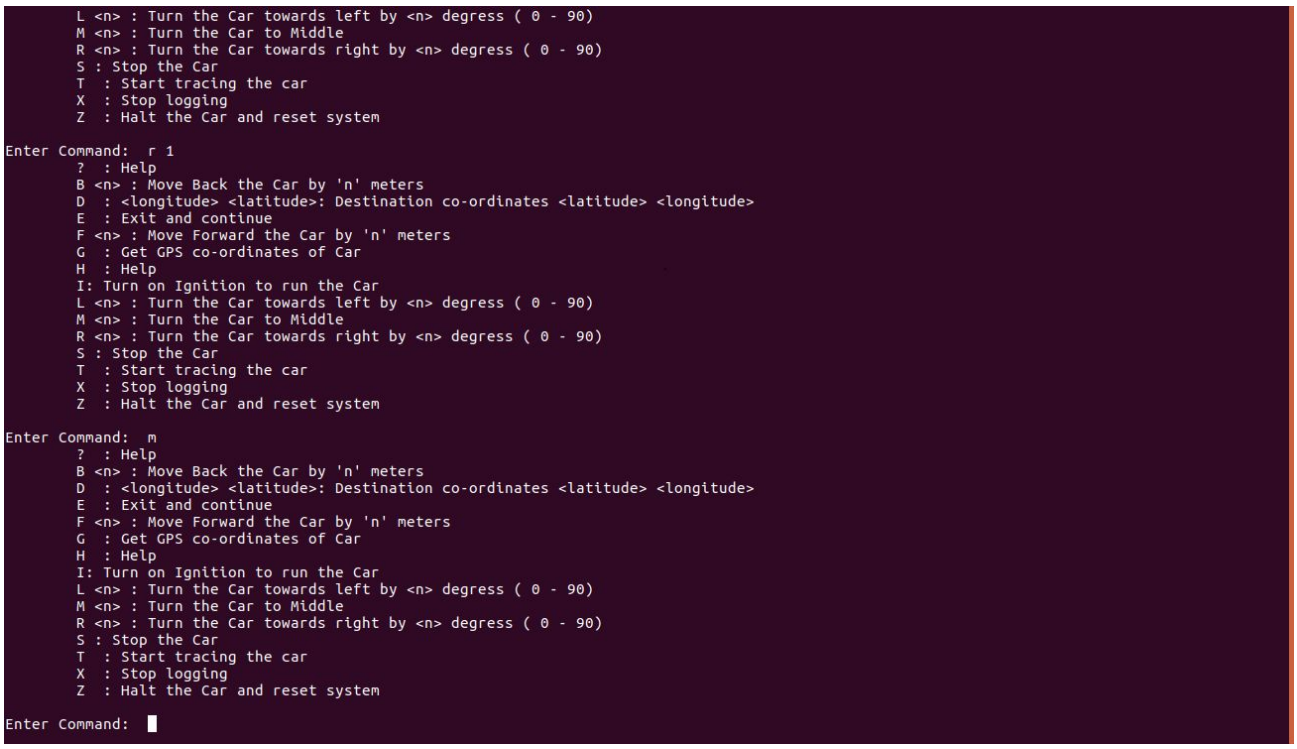


Figure 2. Bluetooth CLI Interface

Lastly, the magnetometer was connected to an SPI channel on the PIC. Attempts were made to establish communication with the device, but there were no parseable data being read back. This device was not implemented due to complications with the SPI development, so heading

and true north data were taken from the GPS module. This was not ideal, but proved to work for our application.

Stage 2 testing consisted of PWM signal control and ESC communication.

The first test required initialization of the PWM signals on the PIC. Examples of the PWM configuration can be found in the book, *Designing Mobile Robots with Microchip 16-bit DSPIC uC's chapter 7*. in order to test these signals, a logic analyzer was connected to the PWM output pins. Varying duty cycles could produce different voltage levels which were expected to be read in by the ESC in the proceeding tests and can be seen in figure 3 and 4 below.

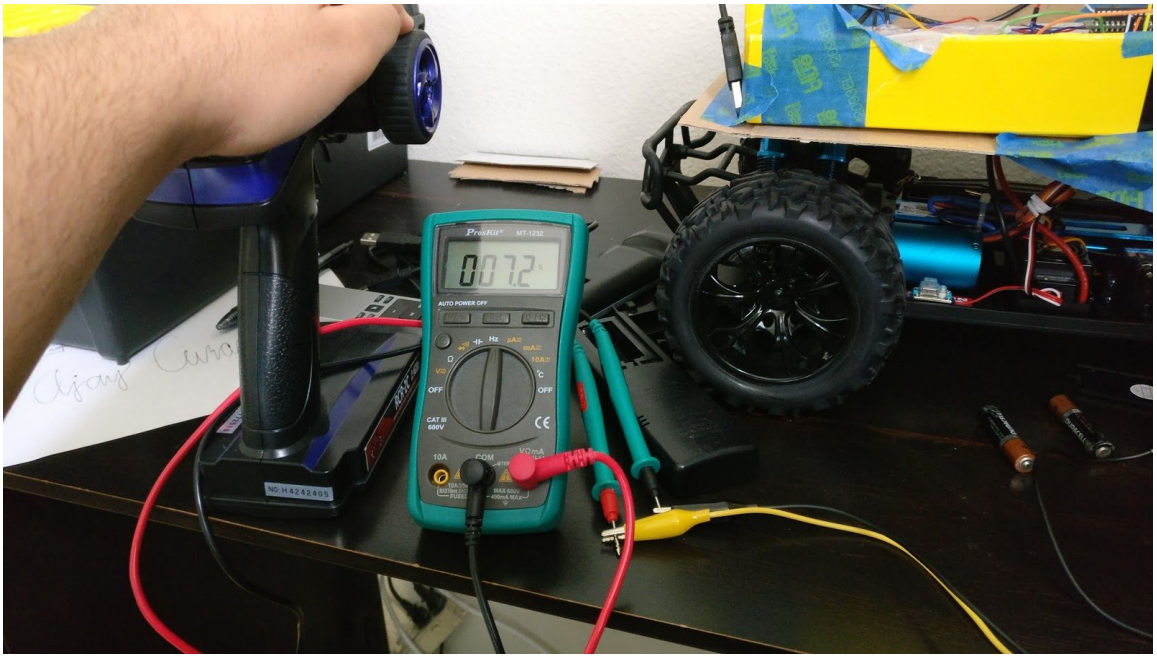


Figure 3. Using Multimeter to examine signals sent to Motors from the RC car

Figure 4. Other measurements obtained

The second test required reverse engineering of the ESC and remote controller signals. This required a signal tap on the antenna line which carried data into the ESC. After adjusting the controls in their respective directions, observations of the steering servo and wheel direction were noted. The details below provides the directions of the controls, and their corresponding effect on the steering servo and motor. Once this was established, the duty cycles were measured for different settings (steering angle and motor speed).

ESC controlled speed Motor @ 62.42Hz:
FWD = 12.3%, Neutral = 9.4%, REV = 6.1%

Steering Servo @ 62.42Hz:
Left = 11.3%, Middle = 9%, Right = 6.6%

Note: Any signals within the range of the duty cycle between FWD/Neutral, and between REV/Neutral would result in varying speeds.

Stage 3 testing consisted of full vehicle functionality testing, and proof of concept to meet minimum viable product (MVP) requirements.

The first test was to ensure the vehicle directions were being commanded properly. After assembling the entire system, commands containing desired controls were sent to the vehicle

from the local machine. Each command combination: left or right, and forward. We tried to emulate the signal to send the ESC motors in reverse but it failed to work properly, so the car can only go forward. These observations were recorded through video and are attached on the disc provided.

The second test was to ensure the vehicle was taking itself to the desired destination. Commands containing desired destination were sent to the vehicle from the local machine. From these tests, results showed the vehicle was able to take it self the distance needed for reaching a destination, however it could not reach a specific destination due to failures in the heading data coming from the GPS. These results were expected given that our application requires sensitivity under the tolerances of the GPS antenna.

V. CONCLUSION

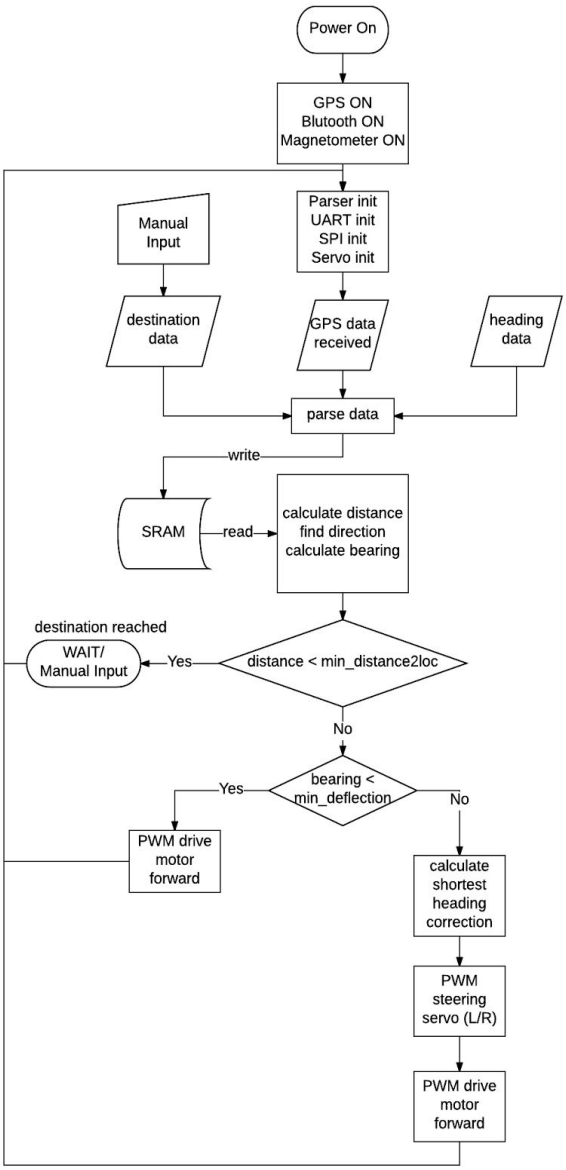
Overall, the project was a success because it was able to meet the minimum requirements initially set: autonomously drive to a desired location, after the user obtains the distance between the current location and destination coordinates specified through our built in distance function. The car cannot turn and travel straight to its destination due to time constraints and trouble we had configuring the Magnetometer via SPI. Several lessons were also learned: interfacing a PIC over UART and SPI, setting the clock oscillator for different devices, programming the PIC for PWM output controls, and interfacing several embedded system components to accomplish a single task.

There are additional testing and development features that can be added. Some would be: solve the magnetometer interface, install ping sensors for object detection, add a pixy camera for object recognition, and establish a wireless internet communication for remote data acquisition.

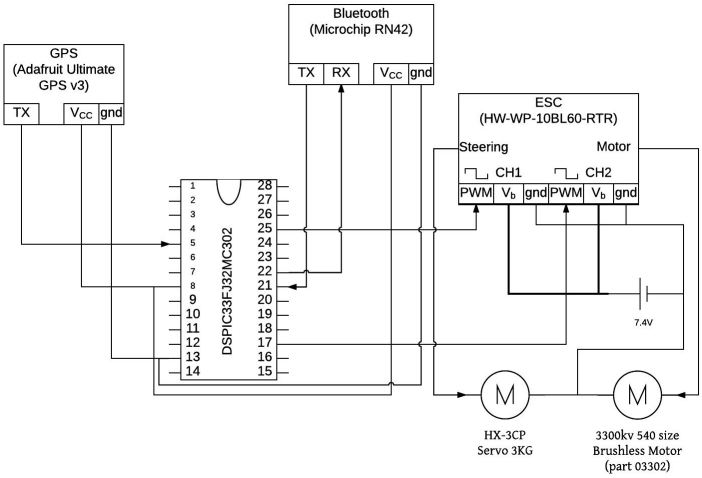
VI. REFERENCES

Bindal, A. (2017). Designing Mobile Robots with Microchip 16-bit DSPIC Microcontrollers. CA: Maple Press. Pages 253 - 280

VII. APPENDICES



Appendix A - System ASM Diagram



Appendix B - System architecture / diagram

Refer to robocar.c & robocar.h on GitHub to see source code.