Proceedings of the ASME 2015 Dynamic Systems and Control Conference
DSCC2015
October 28-30, 2015, Columbus, Ohio, USA

DSCC2015-9799

DEVELOPMENT OF A MINIATURIZED AUTONOMOUS VEHICLE: MODIFICATION OF A 1:18 SCALE RC CAR FOR AUTONOMOUS OPERATION

Dwarkesh Iyengar
Kettering University
Flint, MI, USA

Diane L. Peters
Kettering University
Flint, MI, USA

ABSTRACT

Autonomous vehicles are a subject of intense research interest, and are being approached by many researchers in different ways. Some of these approaches are based upon pure simulation, while others involve investigations using hardware. One possible approach, which can be useful when investigating how autonomous vehicles might interact, involves the use of physical scaled model vehicles, and the development of an appropriate vehicle is the focus of this paper. For this purpose, a commercially available 1:18 radio controlled car is remodeled and modified. An onboard microcontroller unit (MCU) is used for sensor data acquisition and preliminary signal conditioning as well as actuator control. The sensor array includes a gyroscope/accelerometer, a compass and speed encoder to find the angular and linear position of the car in a local coordinate frame as well as a range finder to detect impending obstacles in the vehicle's planned path. This information is sent over a serial communication protocol to a Master station via a 2.4 GHz wireless module. The master station consists of a National Instruments (NI) myRIO real-time FPGA module where the local coordinates are used to formulate the position of the car in global coordinates and a user defined control scheme is implemented and the appropriate actuator signal is sent back wirelessly to the MCU on the car. The main purpose of using an independent and offsite control station is to isolate the main processing and increase response speed to changing environmental factors. Furthermore, the myRIO contains the dynamic model of the car which can be modified by linking it to a personal computer station running the LabVIEW graphic user interface (GUI). This adds greater flexibility to the overall system, thus allowing the user to focus on the different control schemes to be implemented through the hardware setup. This setup will be replicated for more cars, set in an urban traffic environment, and the interactions between the cars can then be studied and optimized.

INTRODUCTION

Autonomous vehicle systems have attracted a great deal of research interest in recent years, as well as significant development efforts from industry. In 2007, DARPA has run the Urban Grand Challenge [1], with the various universities' entries documented in many different publications, e.g. [2, 3, 4]. Since 2010, Google has started developing their own self-driving car, which has been widely reported in the news [5, 6]. Although there have been significant efforts in this field, many problems still remain to be solved, these include problems of sensing, different types and levels of control, and the interaction of autonomous vehicles with their environment. It is anticipated, by many people, that partially autonomous vehicles will present certain problems as well [7]. Since in the near future vehicle autonomy is going to be an inevitable part of everyday life, the initial integration of these vehicles into the already existing road and traffic infrastructure will be of significant concern. One problem of particular interest to the authors of this paper is that of how autonomous vehicles interact with other vehicles, which may be driven by humans with varying levels of autonomy, or which may be driven autonomously using different algorithms. This can be investigated in many ways, utilizing simulation, testing on scaled models, or testing on actual vehicles.

One of the methods chosen in order to investigate these issues is that of using scaled models in a scaled model of an urban environment, fundamentally a miniaturized version of the DARPA Urban Challenge; this will allow those algorithms that appear to be promising, based on simulation, to be tested in an environment where their performance can be easily observed, but without the expense of testing on full vehicles. In order to do this, such scaled models need to be available, and that is the focus of this work: to develop a means by which a radio-controlled (RC) car can be driven autonomously from the National Instruments™ myRIO controller, utilizing suitable hardware, sensors, and appropriate physical modifications to the vehicle.

The use of RC cars for academic purposes is not a new idea; in fact, it has been done for some time at ETH Zurich, which has its ORCA Racer program. At ETH Zurich, students develop and deploy algorithms for a competition in which 1:43 scale RC cars race around a track [8]. The ORCA challenge involves the design and implementation of autonomous RC cars which can maneuver around a race track. The lap time of the car is improved by continuously varying the control strategy of the car through a wireless control station. The car's position is tracked via an overhead VICON motion tracking camera. Since our application involves simulating an urban traffic environment, having an overhead camera is both impractical and unrealistic in nature. There are other research efforts using RC cars, as well, including the use of RC cars to simulate the various behavioral attributes of a driver, as described in [9]; this work focuses on the Intelligent Traffic management systems rather than the level of autonomy or control aspects of the vehicle. Other research involving scaled vehicle model dynamics and control include the Illinois Roadway Simulator, which is a novel, mechatronic, scaled testbed used in place of full-scaled vehicle testing [10, 11].

Converting a commercially available RC car to an autonomous vehicle driven by the myRIO requires many design decisions, involving the types of sensors used, how signals are processed, and what overall architecture is used for the system. In Section 1 of this paper, the system architecture is discussed, and key design decisions are presented. In Section 2, the implementation is presented. In Section 3, the experimental validation and results are given for an open-loop and the response of a simple PID controller. In the final section, progress is summarized and future work is discussed.

1 SYSTEM ARCHITECTURE

One of the early obstacles faced during the design of a feasible and repeatable setup was the limitation of space. Since the scaled urban road environment required a rather small RC car, yet it was imperative to choose the car big enough for the on-board electronics to fit effectively. A 1:18 scale model car is chosen to accommodate these requirements and also to create a fairly small and controllable road environment. However, this size poses yet another obstacle since the myRIO is relatively large and cannot be mounted onto the model chassis. In order to tackle this, a master-slave relationship is created between the myRIO as a stationary master controller and the car as a slave with a Microcontroller unit (MCU). Another key advantage of having the myRIO as a stationary base is that it allows interfacing with a personal computer running LabVIEW graphic user interface (GUI), expanding its capabilities and features which may not be available on a standalone setup. Further, having a microcontroller on the car enables multi-sensor data acquisition as well as preliminary signal conditioning which can free the myRIO to do higher level processing and implement control algorithms. This section of the paper first introduces the different system hierarchy that is considered for the application. Next, it investigates the requirements, possible choices and decisions made for wireless communication, sensors and microcontrollers for such a system.

1.1 SYSTEM HIERARCHY

One possible solution to the problem is a single-master-multi-slave (SMMS) type hierarchy as shown in Figure 1. This type of setup provides a rather simple approach to controlling number of slave car units with only one master control. The myRIO can be a fixed control station connected to a personal computer configured to communicate wirelessly with the slave units.

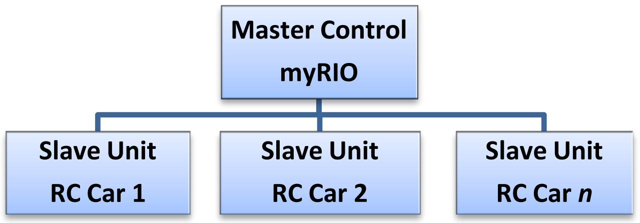


Fig. 1 Single-Master-Multi-Slave Hierarchy

This system, although provides the benefits of simplicity in designing and ease of implementation, possesses number of obvious drawbacks, such as the possibility of communication interference and noise, the high number of communication lines would mean lower baud rate and lower response time, inflexible control model, and limited number of car units.

Another viable alternative is a single-master-single-slave (SMSS) approach as shown in Figure 2. This type of setup solves the problems faced by the previous setup, and also provides the flexibility of adding any number of master-slave pairs to the network. This allows a faster rate of data transfer, hence a more robust implementation of control strategies.

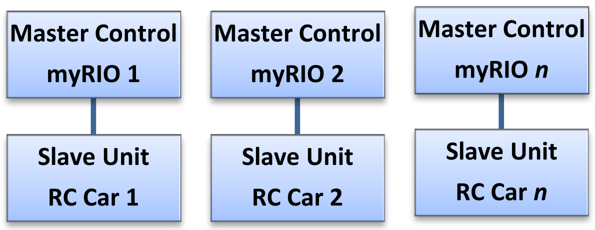


Fig. 2 Multi-Master-Multi-Slave Hierarchy

There is also less possibility of interference between communication and therefore consequently more reliable. The main drawbacks to this are power and space consumption due to the increase in number of independent master control units required. Since the benefits outweigh the drawbacks the latter approach is chosen as the system hierarchy for this setup.

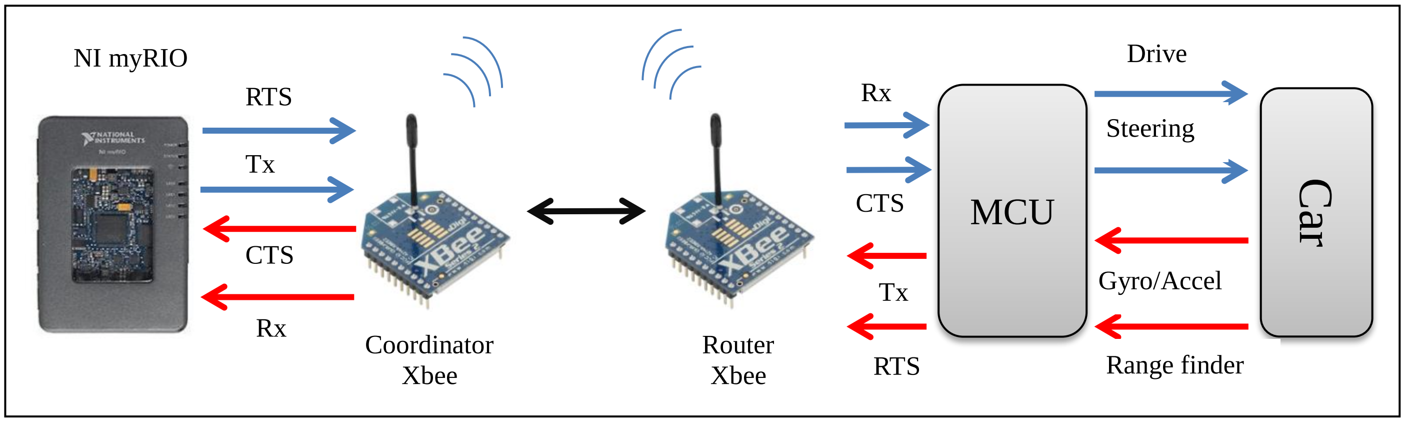


Fig. 3 NI myRIO as Maser communicates with a slave MCU through wireless Serial UART and RTS/CTS handshake

1.2 WIRELESS COMMUNICATION

In order to accommodate the chosen system setup, a robust and reliable long range method of wireless communication is required. To achieve this, a commonly available yet powerful long range wireless communication microprocessor is chosen. A frequency range of 2.4 GHz (802.15.4) is the most widely used for wireless sensor network and control applications. The Digi Xbee wireless module is used for this application for its long range (300ft), multiport sensor input, flexible AT transparent communication protocol and multi-slave address integration.

The wireless communication involves the use of serial UART protocol with RTS/CTS handshake which is used by the myRIO and the MCU to send and receive data over one common line. The serial communication requires the data buffering which is done automatically by the Xbee module which receives the data from the myRIO or the MCU in the form of data packets. The packets contain the identification of the information being sent and also a checksum for the purpose of detecting errors or noise which may have been introduced during transmission.

1.3 SENSOR SELECTION

Having established a viable mode of communication, it is essential, then to fulfill the sensor requirements of the car. Since the car unit has to be completely independent and autonomous, there are various physical data that have to be acquired to make an effective closed loop control system.

1.3.1 POSITION SENSOR

One of the most important information necessary for a completely autonomous system is position. The RC car has to be able to update and send its current position information in a localized frame of reference to the master controller which can then be used to formulate its globalized reference position (Fig.7). The longitudinal translation of the car can be captured by the use of an encoder wheel as shown in Figure 4. A proper encoder has to be chosen in order to get the rpm of the driving wheel and also programmed to negate any error caused by slip. The encoder information is carried through to the micro-controller in the form of digital pulses. Since the encoder wheel consists of a set of "teeth", the number of pulses can be used to calculate the rotation per minute by using a simple equation:

$$Vehicle\ Speed\ \ V\ (mps) = \frac{\mathrm{R}_{wheel} * \mathrm{N}_{wheel} * 2\pi}{60{,}000} \qquad (1)$$

Table 1 Summary of sensor selection

Sensor	**Photo-Reflector**	**Gyroscope**	**Compass**	**Infrared Range finder**
Part Name	GP2S60	L3G4200D	HMC5883L	GP2Y0A 21YK0F
Sensor Parameter	20µs response time	±250/±500/ ±2000 dps	±8 Gauss, ±1°, 12-bit	10cm to 80cm
Interface	Analog	I2C	I2C	Analog
Current Consumption	24mA	6.11mA	100µA	30mA

Where, R_{wheel} is the radius of the wheel in mm and N_{wheel} is the angular velocity of the driven wheel in rpm. By using the rpm, the car speed can be calculated and thus integrating it over a known step time will produce the position of the car. In choosing the encoder it is important to keep in mind the space restriction of the car. For this application an off-the shelf *Sharp* GP2S60 photo-reflective encoder is used to pick up on the angular velocity of the wheel via a digital pulse. From initial testing it was apparent that the encoder sensor in particular had a significant level of noise. To avoid this, a comparator circuit with hysteresis is used as signal conditioning and also a running average filter to smoothen the data (Fig.5). This provides an effective method to reduce noise in the signal and also the lag caused by the signal conditioning will be very small or negligible.

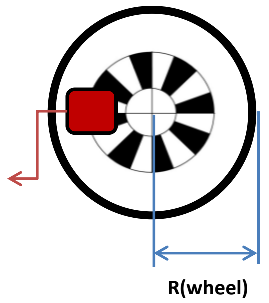
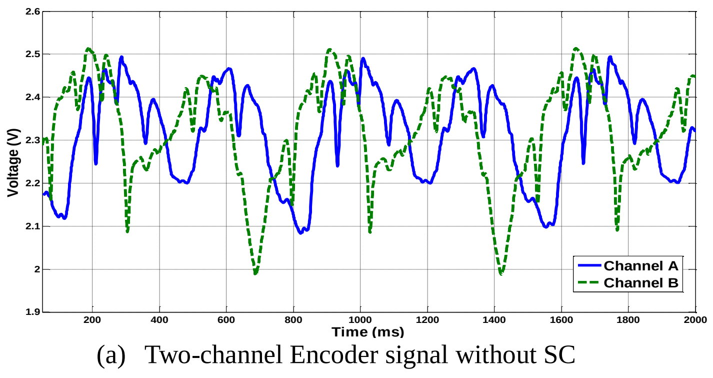
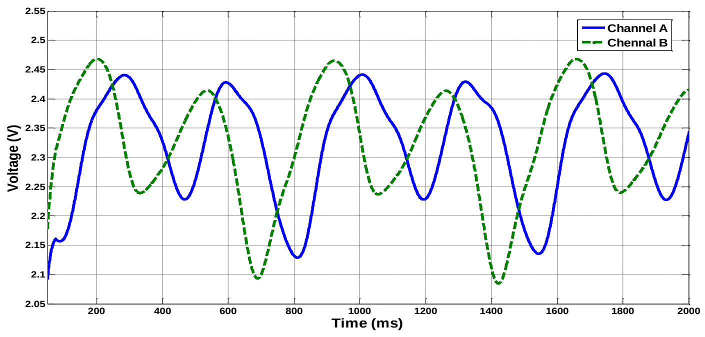


Fig. 4 Encoder wheel with photo-reflector



(a) Two-channel Encoder signal without SC



(b) Two-channel Encoder signal with SC

Fig. 5 By using a comparator circuit an Upper threshold and a Lower threshold voltage were imposed on the signal from the encoder to remove the inherent noise. (a) Encoder signal without signal conditioning (SC). (b) Encoder signal with signal conditioning in place.

The angular orientation of the car can be acquired using various sensors including but not limited to: MEMS Accelerometer, Gyro or a Compass. In this application, the localized orientation is obtained through the use of both a Gyroscope and a Compass, and accelerometer is avoided to reduce the system complexity. Two sensors are used in conjunction to reduce environmental noise and reduce drift. Drift, while using a Rate Gyroscopes, is a common source of error which has an accumulative effect on the signal acquired. To tackle this, a drift compensation equation is used in the LabVIEW GUI (Fig.6). Also, a magnetometer based compass is used to acquire the global orientation of the car with reference to the Earths' true magnetic north.

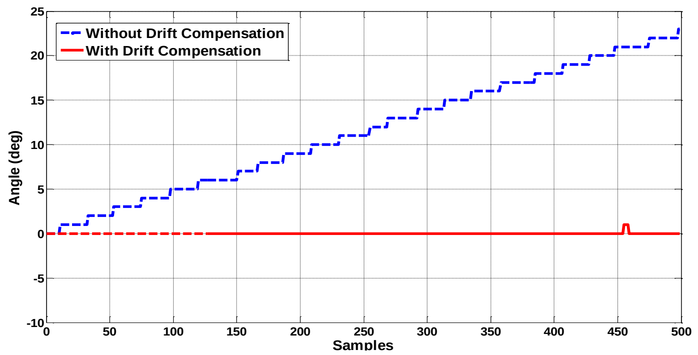


Fig. 6 Encoder wheel with photo-interrupter

Furthermore, digital sensors are preferred over their standard analog variant since they offer higher resistance to noise, higher resolution and accuracy, and ease of integration. The sensors used in this case are the *STMicroelectronics* L3G4200D rate gyro and the *Honeywell* HMC5883L triple axis precision magnetometer [12]. These two sensors can be integrated into the system via two known communication protocols; I^2C (Inter-Integrated Circuit) and SPI (Serial Peripheral Interface).

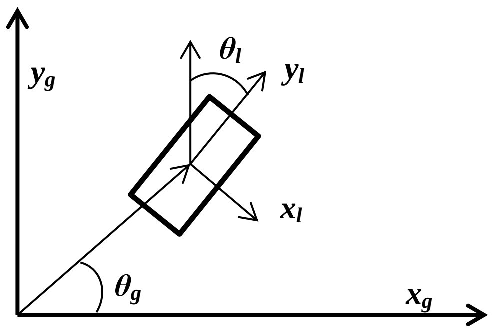


Fig. 7 Global and local positioning of the car [13]

For this application the I²C protocol is chosen since it requires only two lines for communication with a master control unit, while the SPI formally defines at least four signals or more which could otherwise be used for other sensors or actuators. Additional sensors requires more lines for the SPI, which is why the I²C proves to be ideal for this case since this will take up less lines on the on-board controller [14].

1.3.2 OBSTACLE RANGE SENSOR

Another environmental quantity required for this application is an obstacle range finder. This is a crucial piece of information for an autonomous vehicle to take into account in order to make intelligent, quick and effective decisions based on the range and nature of an impending obstacle in its immediate path. This is established by investigating two commonly used sensors based on the means of acquiring the range to an impending obstacle; Ultrasonic and Infrared. Although, the Ultrasonic sensor has the ability to capture long distances, 5 cm to 1 m, it is hampered significantly by external noise and false echo. Since this application requires a scaled urban traffic environment, the distance to be measured falls within a range of 1.5 m to 15 m, which corresponds to a distance of about 10 cm to 80 cm in 1:18 scale model [12]. This makes the Infrared sensor an ideal choice since it demonstrates higher accuracy for shorter ranges which is typically seen in the full-scale scenarios. Hence, a commonly used *Sharp* GP2Y0A21YK0F IR range sensor is used in our application.

1.4 MICROCONTROLLER SELECTION

In order for the mobile unit to effectively acquire signal and perform preliminary signal conditioning, a powerful yet easy to integrate microcontroller is required. The popular Arduino based *Fio* with the ATmega32U4 8-bit microcontroller is selected based on its various merits. The advantages of using this microcontroller includes small packaging, low power consumption, ease of programming and integration with Xbee module. It consists of 11 analog input ports, 6 Pulse Width Modulated (PWM) output ports and 18 digital input/output ports. It also allows for an independent serial communication line for the Xbee, a dedicated I^2C bus port which allows the microcontroller to perform tasks effectively for this particular application.

2 IMPLEMENTATION

For implementation and initial testing, a prototype of the required RC car is built with all the necessary sensors and actuators. The car is stripped of its pre-existent embedded module and replaced with the microcontroller. This section illustrates the various physical modifications and software implementation required to test the system. Figure 8 illustrates the physical data connections required between the microcontroller, sensors and the Xbee module.

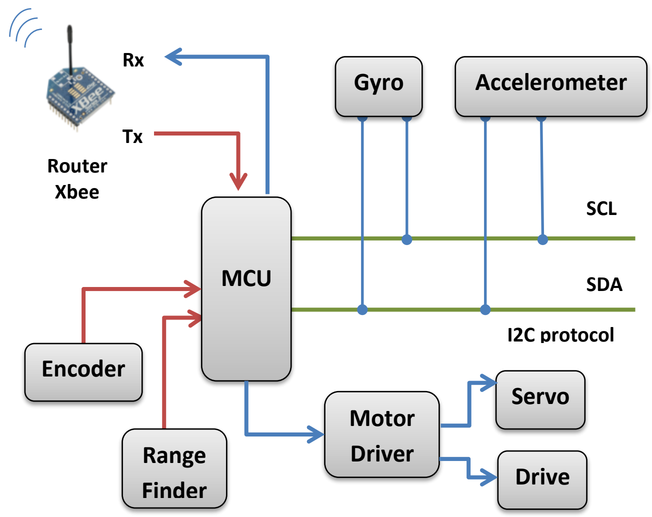


Fig. 8 Schematics of the embedded car

This section covers the physical modifications and software implementations made to the overall system. The physical modifications include MCU and sensor mounting on the car unit and the master control unit. The software aspect includes programming the various tasks assigned to the microcontroller and the myRIO unit.

2.1 PHYSICAL MODIFICATIONS

Since the purpose of the initial prototype is to test the onboard electronics and open-loop control of the car, a solderless breadboard is used in place of a conventional printed circuit board (PCB). This enables fast and easy debugging and rewiring of the circuits. The electronics from the original car is removed and replaced with a 3D printed mount for an 8.3cm x 5.5cm solderless breadboard (Fig.10). The mount also allows space for the lithium ion battery used to power the *Fio* microcontroller board. Once the circuit is tested and verified, it will be replaced with a printed circuit board, since a breadboard is more space consuming and adds considerable amount of weight to the car.

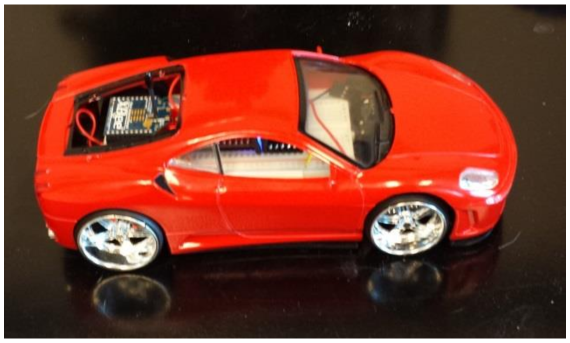


Fig. 9 Autonomous RC Car

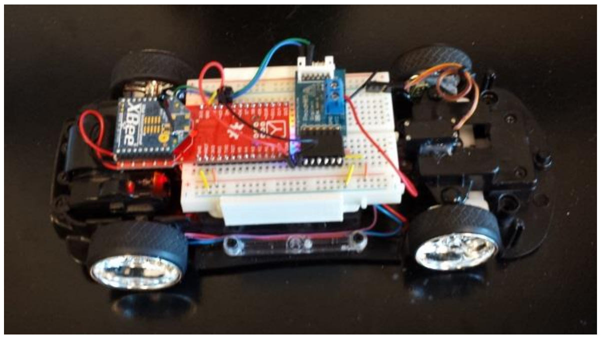


Fig. 10 Electronics for Autonomous RC Car

A commercially available RC car has a bang-bang steering control which switches between left and right. In order to achieve a more precise and discrete range of steering angle, the existing setup is replaced with a Nano-servo motor. For this application a Nano-servo, which are commonly used for model aircraft control surfaces, proves to be the right choice considering the space limitation of a 1:18 scaled model car. Since a standard servo motor has low resolution and less accuracy, a digital servo motor is a preferred choice. Additionally, unlike a standard servo, a digital servo uses a Quartz crystal controlled microprocessor with FET amplifier which offers the advantages of faster control response, constant torque throughout the servo travel, and increased holding power when stationary which are significantly necessary to simulate the steering control in a scaled model [15]. The *Turnigy* T531-BBD high speed Nano digital servo is used for this system.

The power consumption aspect of the embedded car is tackled by separating the driving DC motor which consumes the majority of power. The driving motor is powered by four 1.5V AA Alkaline *Duracell* batteries, whereas, the microcontroller, Xbee and the sensors are powered by a single 3.7V Polymer Lithium-Ion battery with a charge capacity of 850 mAh. This battery rating is chosen due to the total power demand of the microcontroller and all the accessories combined.

The physical placement of the sensors involves placing the Gyroscope and the Compass on the breadboard mount which is assumed to be perpendicular to the ground. Since, the compass is influenced by the earth's magnetic field; the declination angle of the current latitude is taken into account to calibrate the sensor for null positioning. Further a 3D printed encoder wheel is affixed along the driven shaft of the car and the photo-reflector is secured on the side of the car body. The Xbee module is also placed towards the rear of the vehicle. The plastic see-through cover is physically removed (Fig.9), for unhindered wireless communication and ease of access to the microcontroller universal serial bus (USB) port for programming.

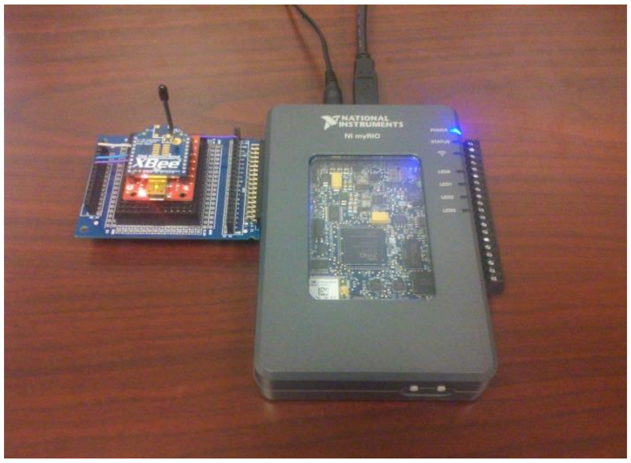


Fig. 11 myRIO connected to coordinator Xbee

The master control unit, comprising of the myRIO FPGA device is connected to the Xbee module via the Serial UART pinouts (RX and TX) and powered by the myRIO unit (Fig.11). This simple setup allows for the myRIO to be either connected to a Personal Computer or act as a standalone unit.

2.2 SOFTWARE IMPLEMENTATION

The Xbee modules are programmed through the XCTU wireless management software environment. The master and slave Xbee modules are programmed with a unique Personal Area Network (PAN) ID which enables the communication to be faster and free from interference. The two modules are also programmed to be in transparent (AT) mode of operation since in this application the Xbee modules acts only as a wireless serial UART communication between the myRIO and the MCU [16]. The master Xbee module is set to be a *coordinator* and the slave module is set to be a *router*.

The *Fio* microcontroller is programmed with the use of *Arduino* Integrated Development Environment (IDE). The program, based on C, encompasses tasks performed in a sequential order. The first task of the microcontroller entails the acquisition of digital pulse signals from the encoder. These pulses are converted to wheel speed and distance travelled. Secondly, the compass and gyroscope raw data is individually acquired through the I^2C port by calling their respective sensor addresses. These signals are filtered, error compensated and converted to angular orientation. Thirdly, the analog signals from the infrared range finder, which are converted to unsigned integers by the microcontroller Analog-to-Digital converter (ADC), are converted to distance in metric units (cm). Finally, these four sets of data are consolidated and buffered by the MCU and sent to the Serial port dedicated to the *router* Xbee via the in-built Serial communication library as a single data packet.

The *coordinator* Xbee module connected to the myRIO receives the packet data sent by the *router* Xbee. A LabVIEW virtual instrument (VI) is implemented to parse the incoming data available at the myRIO serial port which is then used for obtaining the current state of the vehicle. The state variables are used to determine the control signal for the next time step by a control algorithm. Since this paper deals with open-loop control, the control signals are provided based on a predefined set points. The control signals to be sent include: PWM signal to the motor driver and the steering angle. These two signals are then buffered and the packet data is sent to the *coordinator* Xbee via the serial port. The holistic flow of the tasks can be seen in Figure 12.

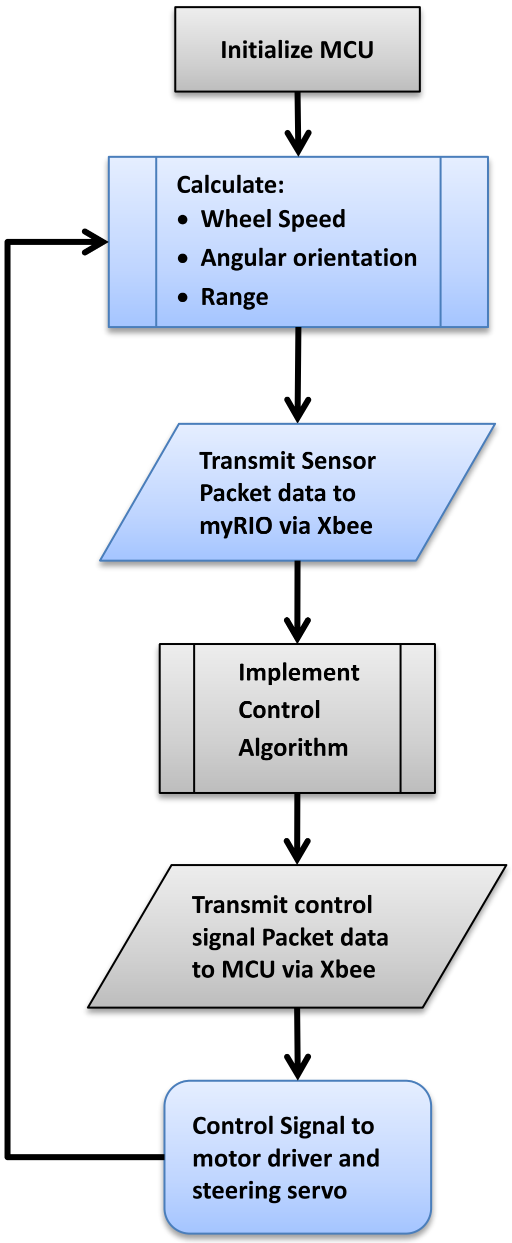

Initialize MCU
Calculate:
• Wheel Speed
• Angular orientation
• Range
Transmit Sensor Packet data to myRIO via Xbee
Implement Control Algorithm
Transmit control signal Packet data to MCU via Xbee
Control Signal to motor driver and steering servo

Fig. 12 Flowchart of allocated tasks

3 RESULTS

To demonstrate the feasibility and robustness of the implementation described in this paper, initially two separate PID controllers were used both for speed and steering control (Fig.13, Fig.14). The sensor data acquisition rate was found to be close to 400 Hz, coupled with signal conditioning results in a time delay of about 3.5 milliseconds for the microcontroller. The serial wireless transmission of the packet data causes a delay of about 104 milliseconds for each data sent and received. The operation of the car primarily lies in the region of 0.5 to 1.5 meters per second at 1:18 scale, equivalent to about 20 to 60mph in full scale, which is ideal urban traffic speeds. Hence, the setup suggested in this paper more than meets the requirement of the operating conditions and is replicable and reproducible for any 1:18 scaled urban traffic research undertakings.

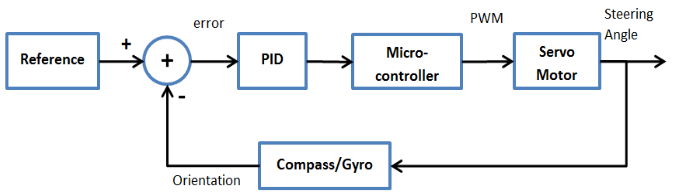


Fig.13 PID control of steering angle

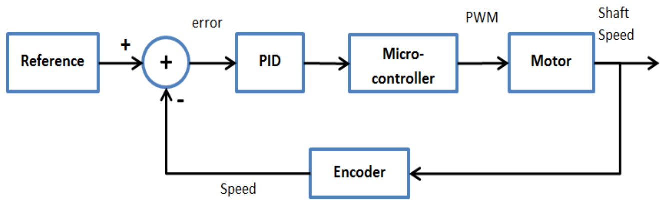


Fig.14 PID control of driving speed

By implementing the above described hardware modifications and algorithm, the sensors and actuators were successfully integrated into the system. The sensor data was effectively acquired by the microcontroller and preliminary signal conditioning performed through the *Arduino* IDE. The wireless communication was tested and successfully established between the microcontroller and the myRIO. The LabVIEW program written to parse the received serial data and send out the control signal packet data is debugged and refined accordingly. Hence, open-loop and closed-loop control of the car was successfully verified.

Since these three processes were independently tested and confirmed to be working well, it demonstrates the feasibility of implementing a PID controller but also higher complexity closed-loop real-time control schemes such as Model Predictive Control (MPC) and Iterative Control (IC) for further development towards the final goal of this project. The speed of wireless communication used in this system, although relatively faster than other forms of communication, influences the overall response time of the system. This is significant because in order to effectively replicate, and possibly enhance, the response time of a human driver the data processing time has to be equally responsive to changing environmental conditions which meets the ultimate goal of developing an urban traffic model.

4 CONCLUDING REMARKS

This paper introduces a means of modifying a commercially available RC car into an autonomous car. This was done by implementing a wireless Master-Slave scheme of architecture using a microcontroller and an NI myRIO FPGA device. It describes the hardware and software modifications required to successfully carry out closed-loop control of the car. Further significant steps in the process of truly making the car autonomous includes developing a dynamic control model of the vehicle, implementing a robust and more complex closed-loop controller and testing and validating the control system design in a real-world setting. Once the prototype is tested, the breadboard has to be replaced by a circuit board and the design process replicated for more such cars. The interactions between multi-mode control systems can thus be studied further and used for education, research and for refining the dynamic model to predict the vehicle's performance in a real-world setting.

ACKNOWLEDGEMENTS

Partial support for this work was provided by National Instruments. This support is gratefully acknowledged.

REFERENCES

[1] McBride, J. (2007). DARPA Urban Challenge.

[2] Urmson, C., Bagnell, J. A., Baker, C. R., Hebert, M., Kelly, A., Rajkumar, R., & Team, D. U. C. (2007). Tartan racing: A multi-modal approach to the DARPA Urban Challenge.

[3] Bohren, J., Foote, T., Keller, J., Kushleyev, A., Lee, D., Stewart, A., & Satterfield, B. (2008). Little Ben: the Ben Franklin racing team's entry in the 2007 DARPA Urban Challenge. Journal of Field Robotics, 25(9), 598-614.

[4] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., & Thrun, S. (2008). Junior: The Stanford entry in the Urban Challenge. Journal of field Robotics, 25(9), 569-597.

[5] The Economist. (2013). Look, no hands. Retrieved from http://www.economist.com/news/special-report/21576224-one-day-every-car-may-come-invisible-chauffeur-look-no-hands

[6] Dockterman, E. (2015). Google's self-driving car may come with airbags on the outside. Time Magazine. Retrieved from http://time.com/3758446/googles-self-driving-car-may-come-with-airbags-on-the-outside/

[7] Ross, P.E. (2014). When 99% safe isn't safe enough. IEEE Spectrum. Retrieved from http://spectrum.ieee.org/transportation/advanced-cars/driverless-cars-optional-by-2024-mandatory-by-2044

[8] ETH Zurich (n.d.) Retrieved from https://control.ee.ethz.ch/index.cgi?page=sada;action=details;id=347 on 3/26/2015.

[9] Brennan, S., and A. Alleyne. "The Illinois Roadway Simulator: A Mechatronic Testbed for Vehicle Dynamics and Control." *IEEE/ASME Transactions on Mechatronics IEEE/ASME Trans. Mechatron.* 5.4 (2000): 349-59. Web. 20 July 2015.

[10] He, Wen, Guisheng Chen, Shuming Tang, Deyi Li, Mu Guo, Tianlei Zhang, Peng Jia, and Feng Jin. "A Scaled-down Traffic System Based on Autonomous Vehicles: A New Experimental System for ITS Research." *2012 15th International IEEE Conference on Intelligent Transportation Systems* (2012): 13-18. Web. 20 July 2015.

[11] Brennan, S., and A. Alleyne. "Robust Scalable Vehicle Control via Non-Dimensional Vehicle Dynamics." Vehicle System Dynamics 36.4-5 (2001): 255-77. Web. 20 July 2015.

[12] Doering, E. (2013). NI myRIO Project Essentials Guide. Retrieved from http://www.ni.com/academic/myrio/project-guide.pdf on 4/7/2015.

[13] Gillespie, T. (1992). Fundamentals of Vehicle Dynamics. Warrendale, PA: Society of Automotive Engineers.

[14] Eady, F. (2004). Networking and internetworking with microcontrollers. Amsterdam: Newnes.

[15] Futaba: Digital FET Servo. (n.d.). Retrieved from http://www.futaba-rc.com/servos/digitalservos.pdf on 4/15/2015.

[16] The AT Command Set. (n.d.). Retrieved from http://www.digi.com/support/kbase/kbaseresultdetl?id=2205 on 4/14/2015.