Presents

Armadillo

2007 Intelligent Ground Vehicle Competition

Team Members

Brandon Bell, Shawn Ellison, Jeremy Gray, and Philip Munie

Faculty Advisor Statement

I certify that the design and development of the Armadillo autonomous vehicle has been significant and each team member has earned credit hours for their work.

Dr. CJ Chung [1]
Department of Math and Computer Science
Lawrence Technological University
chung@ltu.edu (248) 204-3504

Date

[1] Co-Advisors: Dr. Lisa Anneberg, Dr. Peter Csaszar, Professor Moriconi, Professor Shih, and Professor Maurice Tedder

1. Introduction

The Autonomous Vehicle Team at Lawrence Technological University (LTU) is proud to present Armadillo, a brand new autonomous vehicle designed for the 2007 Intelligent Ground Vehicle Competition (IGVC). Armadillo's unique two-component, four wheel design features an electric power system, a complex dual-channel motor controller, dual-camera optical sensors, and state-of-the-art software written in Microsoft's C# programming language. It represents Lawrence Technological University's latest venture in providing a safe, reliable, and cost-effective autonomous vehicle that is rich in cutting-edge technologies.

2. Design Process

Team Armadillo followed the traditional software engineering process model of planning, designing, implementing, and testing the Armadillo autonomous vehicle. This model was applied to both the hardware and software design, since it stresses a "plan before execute" approach. As the project was planned out, team members were each assigned responsibilities for the project.

2.1 Project Planning Process

As *Figure 1* indicates, the Armadillo project's tasks were broken down into pieces, and a timeline was planned for the completion of these individual tasks. Due to starting this project late in the IGVC season, much more time was allocated for getting the hardware designed and implemented than for the software. The primary plan was to get Armadillo up and running this year, and to optimize the software for the following year's competition.

ID	*Task Name*	*Start*	*Finish*	*Duration*
1	Design and Implement Mechanical Components	1/10/2007	5/4/2007	115d
2	Design and Implement Electrical Components	4/15/2007	6/1/2007	48d
3	Design Autonomous Navigation	1/15/2007	5/25/2007	131d
4	Implement Autonomous Navigation	2/15/2007	6/8/2007	114d
5	Test Autonomous Navigation	5/21/2007	6/8/2007	19d
6	Design JAUS Module	5/12/2007	5/19/2007	8d
7	Implement JAUS Module	5/15/2007	5/22/2007	8d
8	Test JAUS Module	5/21/2007	6/8/2007	19d
9	System Integration and Testing	5/15/2007	6/8/2007	25d
10	Final Design Report	5/10/2007	5/18/2007	9d
11	Presentation Report	5/22/2007	6/3/2007	13d

Figure 1 – Project Task Timeline

2.2 Development Process

Figure 2 - Low-Cost Laptop Robot

The development process involved both the design and implementation of Armadillo. During this process, both the hardware and software components were constructed in small modules that were then integrated into the final product. This allowed for each team member to work independently in the early stages of development. However, due to the project plan focusing on getting the hardware completed first, software development did not begin until shortly before the completed construction of Armadillo. Fortunately, Team Armadillo was able to use LTU's 2005 Think-Tank vehicle as a development resource while Armadillo was being finished. In addition, the software team was also able to design the image processing algorithms by using L2Bots (Low-Cost Laptop Robots). The L2Bots are very simple, small robots that have a webcam and two low powered wheels that are all controlled by a laptop (see *Figure 2*). These robots allowed the team to test their image processing and control algorithms outside of the lab.

2.3 Testing Methodology

Each hardware and software module was first tested individually before the pieces were integrated with one another. The integrated modules were then tested to make sure they interacted properly with one another. Finally, the whole system was tested to ensure that the design met all of the requirements of IGVC. During the early stages of testing, the 2005 Think-Tank robot was used as a platform for software module testing. The testing was then moved to Armadillo once the mechanical and electrical construction was completed.

2.4 Team Organization

The 2007 Armadillo team primarily consists of three graduate computer science students and one undergraduate electrical engineering student. The team members and their individual responsibilities are shown in *Table 1*. In addition to the primary team members, Lawrence Technological University's H2Bot team also contributed to this vehicle's

Team Member	Responsibilities
Jeremy Gray BSEE	• Mechanical Design • Electrical Design
Shawn Ellison MSCS	• Autonomous Challenge Team Leader • JAUS Programming • Joystick Control Interfacing
Philip Munie MSCS	• Camera Interfacing • Image Processing
Brandon Bell MSCS	• Motor Controller and Motor Interfacing • JAUS Hardware Interfacing • Electrical Design Assistance

Table 1 – Team Organization

design by acting as experienced advisors for the 2007 competition. It is estimated that the team spent approximately 850 hours on the development of the Armadillo autonomous vehicle. The hardware design took approximately 325 hours, and the software design took approximately 475 hours.

3. Hardware Design

Armadillo's hardware design is comprised of three primary parts: the mechanical design, the electrical system, and the sensors and systems integration. Team Armadillo evaluated its success on this part of the design process based on the principals of safety, reliability, and cost-effectiveness.

3.1 Mechanical Design

3.1.1 Structure

Armadillo is a four wheel vehicle with front differential drive steering and a two-part body structure that is similar to an articulated steering system. Through experimenting and research, it was found that using a two-part structure allows for a smaller turn radius and more maneuverability than a four wheel, one-part structure. The frame (shown in *Figure 3*) was constructed from angle aluminum of 3/4" width and 1/8" thickness, and aluminum sheets of both 0.125" and 0.190" thickness. The body of the vehicle is constructed from two fiberglass shells that were sculpted and formed to be easily removable and attachable to the aluminum structured frame. This fiberglass body easily supports the required payload, and it contains ventilation fans and vents to remove heat produced from the internal electronics.

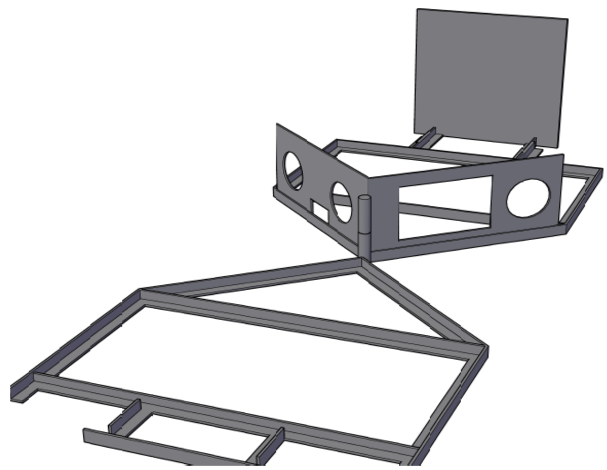

Figure 3 - CAD Model of Armadillo's Structure

3.1.2 Drive Train

Armadillo's drive train design offers low power consumption, speed control, and hardware speed limiting. The motor configuration is two 12V, 2.4HP permanent magnet motors, and these motors were selected to achieve the required torque to move Armadillo, stay within competition speed limits, and achieve the lowest power consumption relative to torque. The drive train's speed resolution is achieved by mounting encoders to the motor shaft. Each motor shaft is directly connected to each wheel using an adapter that was specially created to link the motor shaft to its wheel. Although the vehicle possesses four wheels, only the front two wheels are powered by the electric motors. The two rear wheels simply possess ball-bearing wheel housings attached to a fixed shaft bolted to the rear frame. This design allows for front wheel drive motion with minimal resistance from the rear wheels.

3.1.3 Motor Control

Figure 4 - Roboteq AX3500 Motor Controller

The speed and direction of Armadillo's motors are controlled by a Roboteq AX3500 dual-channel digital motor controller (shown in *Figure 4*). The AX3500 provides both a velocity and direction channel to govern the motion of its dual motors. It accepts simple requests and relays controller state information via a single RS-232 serial communication line. This communication is achieved through simple hexadecimal commands and responses. To verify that there are no errors in this communication, the controller echoes every command back across the serial connection. This ensures accurate and safe control of the motors through software error checking.

3.2 Electrical System

3.2.1 Power Source

Armadillo is powered by a single Power-Sonic 12V/40AH sealed lead acid battery. An onboard charger with 110V AC interface is employed to restore battery power when the vehicle is not in operation. The power supply is fed directly to a power distribution box, which dispenses power to the vehicle's various electronics and components.

3.2.2 Power Distribution

A box mounted printed-circuit board (PCB) distributes and switches power to each of the electrical components. In addition, the PCB provides connections for the wired and wireless E-stop devices, and power for the front and rear section ventilation fans. The power and communication control system schematic for Armadillo is shown in *Figure 5*.

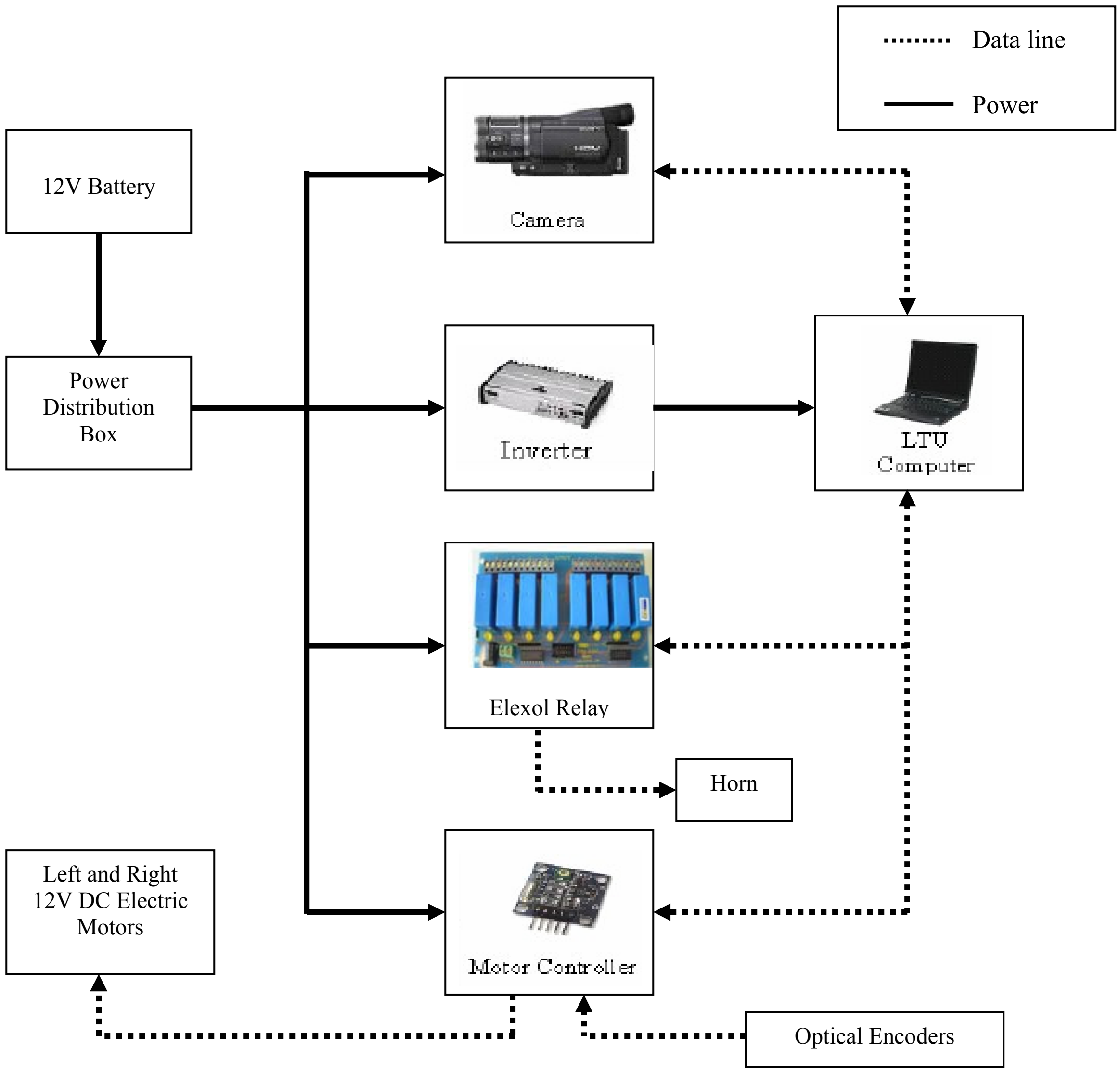

Figure 5 – Power and Control System Schematic for Armadillo

3.2.3 Safety Features

Armadillo is equipped with several important safety features that help prevent accidents or injury while the vehicle is running. The primary safety devices are both manual and wireless remote emergency stop (E-stop) systems. The manual E-stop system is activated by a large tactile button located on the back of the Armadillo's exterior. When pressed, this button stops the motion of the vehicle by cutting off all power to the motors. Similarly, the wireless remote E-stop system uses an RF remote that sends a signal to the motor controller to shut down all power to the motors. In addition to the E-stop functionality, Armadillo's fiberglass chassis prevents user contact with the active electrical system, and its sealed lead acid battery helps ensure that no hazardous material is spilled during operation.

3.3 Sensors and Systems Integration

Armadillo's sensor design uses only two cameras connected through a FireWire connection for visual interpretation. Thus, the visual information gathered from these cameras is the only decision making tool available for the Armadillo besides the very specific JAUS commands. Once the visual information is collected, it is then sent to the decision making software on a laptop computer for interpretation and motor command decisions. Alternatively, the laptop can receive JAUS commands which are then used to start or stop the autonomous mode, or are used to send a signal to the relay output device to activate the warning signal.

3.3.1 Cameras

Armadillo utilizes two Panasonic PV-GS320 digital camcorders for its visual recognition (shown in *Figure 6*). The effective resolution of the cameras is 1.89 megapixels, with a video resolution of 320x240 @ 6 frames per second (fps).

Figure 6– Panasonic PV-GS20

3.3.2 Laptop Computer

The laptop that runs Armadillo's software is a Dell Inspiron E1505. This system contains a 2.0 GHz Intel Core 2 Duo processor with 1 GB of DDR memory, and it was found to be an effective low power, computationally strong solution to Armadillo's processing needs. This computer also contains a GeForce Go 7300 video card with 256 MB of ram, which greatly assisted the team's video processing power.

3.3.3 Relay Output

Armadillo uses an Elexol I/O 24 Relay Output Board to handle routing power to the warning signal when the appropriate JAUS command is received. This device communicates with the laptop computer via a USB connection, and it waits for simple hexadecimal commands from the laptop. These commands tell the device to route power and cut-off power to specific electrical wires that are connected to the device.

4. Software Design

4.1 Software Strategy

The Armadillo software was developed on top of Microsoft's .NET framework in C#, using the Visual Studio 2005 software development environment. C# provides a flexible and scalable approach to multi-threaded software development, and its powerful libraries allow for simple communication with external devices via the USB and FireWire ports.

4.2 Software Architecture

Figure 7 shows a high-level diagram of Armadillo's software architecture. As this figure indicates, Team Armadillo continued to focus on using a modular design throughout the software development process. This allowed for individual team members to code and test each module separately before the pieces were integrated into the final system. After the initial calibration and setup phase, the two cameras collect raw data, which is then interpreted and combined in the "Sensor Fusion" module. Once the image data has been merged, the "Path Planning" module decides on the proper course for Armadillo, and this decision is passed to software controlling the "Path Execution" module. This module communicates with the motor controller hardware to execute the planned path. Simultaneously, the "JAUS Receiver" module waits for commands targeted for Armadillo. When a command is received, the module interprets this command, and signals either the warning indicator to activate (which produces a horn sound) or the motor controller to start or stop autonomous mode.

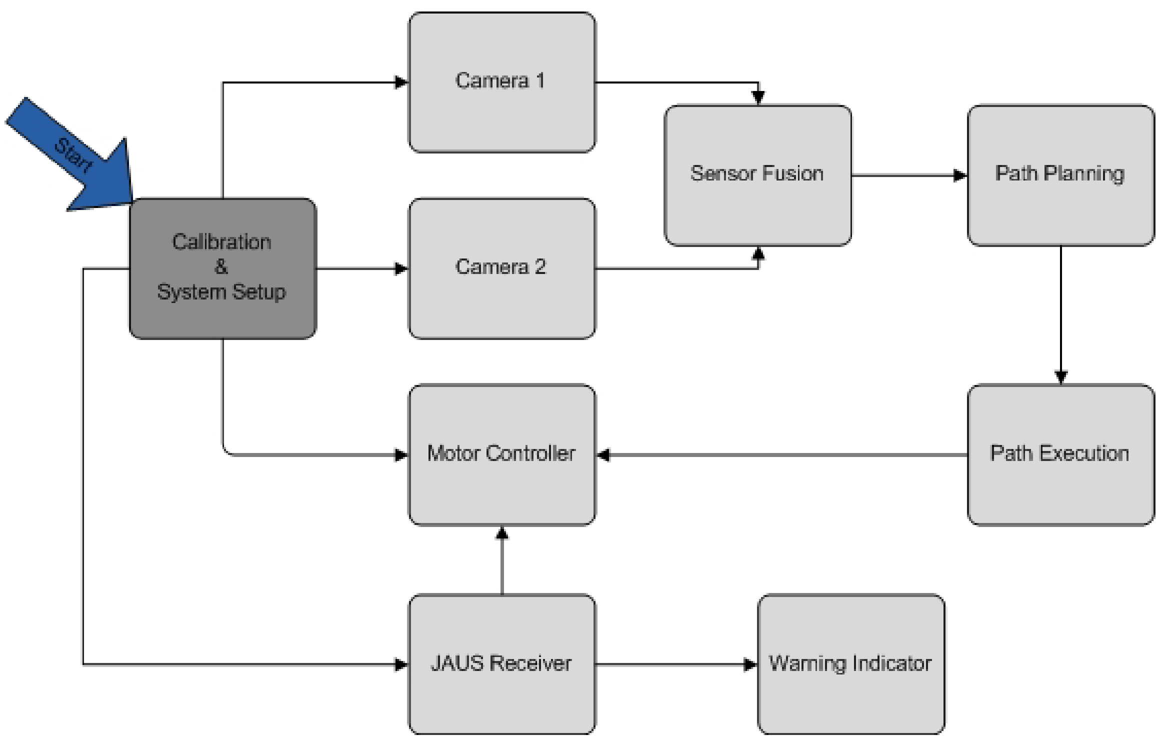

Figure 7 – High Level Software Architecture Diagram

4.3 Autonomous Challenge

Armadillo will navigate through the Autonomous Challenge with only the use of two digital cameras for its vision. Thus, all navigational logic is based on this limited knowledge, and the decision making is done exclusively through software by analyzing the raw visual information received from the cameras. Once a decision has been made, the direction and speed commands are sent to the motor controller, which carries out the decision.

4.3.1 Image Processing

To begin the image processing, each camera captures a 160x120 colored image. These colored images are then combined to create a single wide-view image. Once this single image is created, it is run through a color recognition module, which recreates the image using only a few basic colors that are in its palette. For instance, there are key colors that specify objects to avoid and lines to stay within. All other colors are ignored, and thus, they are all defaulted to the same color. After this module completes, the image is then passed off to the lane following module. *Figure 8* provides an illustration of the image fusion process.

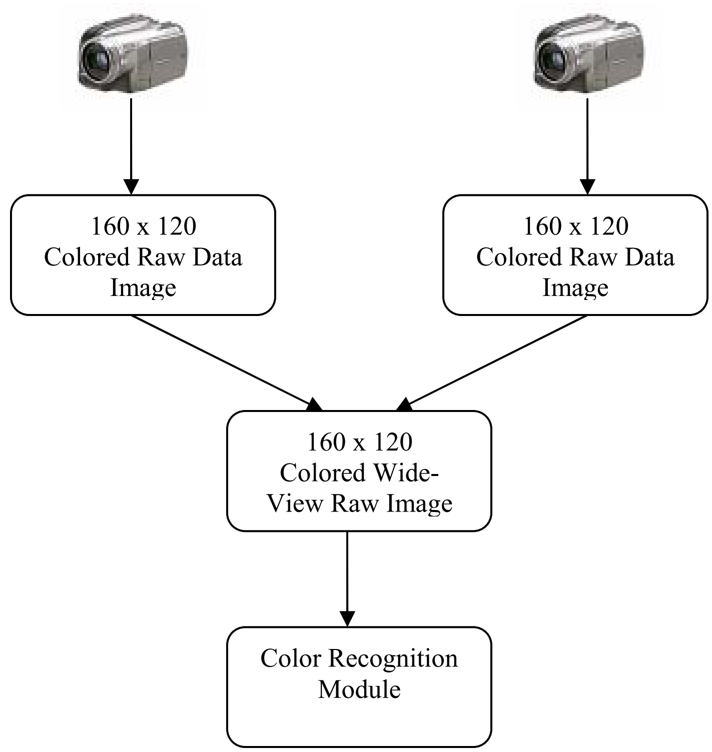

Figure 8 - Image Fusion

4.3.2 Lane Following

Armadillo uses very simple logic for its lane following decisions. Its primary goal is to navigate towards the largest available opening at all times. This opening, when looking at a 2D image, is the largest opening on the x-axis. When this opening is identified, the algorithm then calculates the center of the opening and the degrees in which the robot needs to turn in order to head in that direction. There is also a range, on the y-axis, in which this evaluation takes place. This range was implemented and then adjusted to make sure that the largest available opening was not determined from a section of the lane that was too far away. This was very important because if the range was not limited, this algorithm might make decisions too far in advance. For example, it could decide to make Armadillo turn before it even reached the beginning of a curve in the lane, and this would be very undesirable.

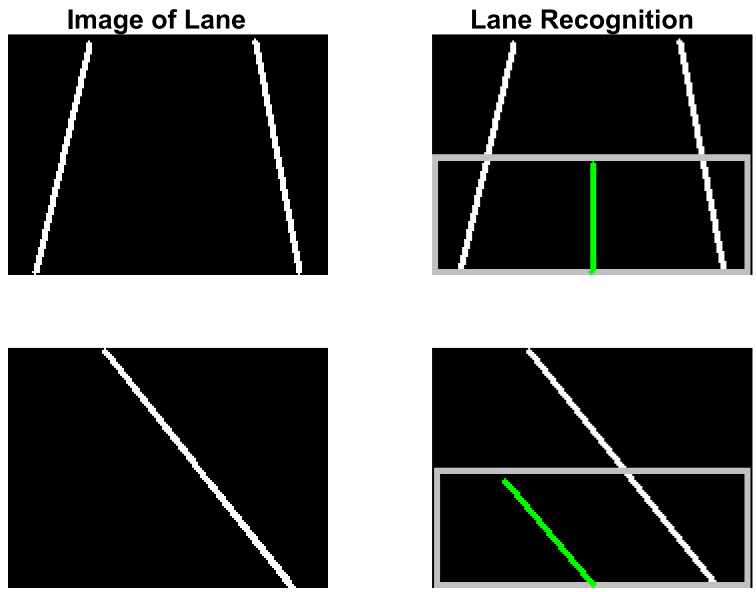

Figure 9 – Lane Following Logic

Figure 9 shows an example of two possible lane images, and the corresponding lane recognition image. Each lane image shows the view from the camera after the image has been processed, and each lane recognition image shows how the decision making logic works. The grey box in the lane recognition images demonstrates the limited area of the image that the decision making logic evaluates. As shown, the green line, which represents the angle at which Armadillo should move in order to follow the line, always points to the center of the largest available opening.

4.3.3 Obstacle Avoidance

Obstacle avoidance is also handled the lane following module. This is accomplished by having the objects internally seen as part of the lane. Thus, the objects are combined with the lane to make a larger, less elegant version of the lanes. After the object and lane data is combined, the lane following module determines the largest opening and plans the appropriate route. In essence, the lane following and object avoidance are handled simultaneously, and use the same method. *Figure 10* illustrates the logic behind this computation.

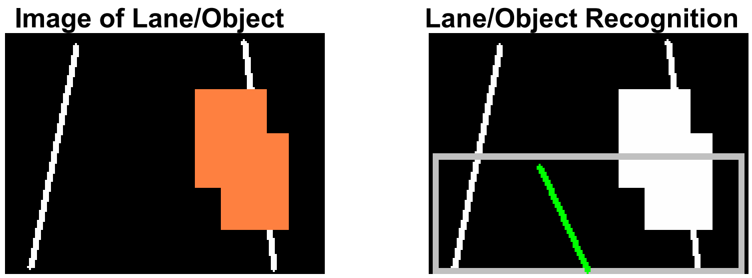

Figure 10 – Obstacle Avoidance Logic

As shown, the only difference between *Figure 9* and *Figure 10* is the addition of the object in the camera's field of view. Since the object is seen as part of the lane, the lane following logic can still be used to stay in the middle of the largest available opening.

4.4 Navigation Challenge

Due to cost constraints, Armadillo was unable to acquire the GPS device needed to participate in this section of the IGVC competition. Thus, the team's plan is to obtain the funding for this device for the 2008 competition, so that it can participate in this event. This will allow the team an opportunity to build upon the complex software already developed in the Autonomous and JAUS challenges for this year's competition.

4.5 JAUS Challenge

Armadillo is designed to handle all of the JAUS commands that are specified for this challenge. The JAUS software was seamlessly integrated into the rest of the Armadillo project, although it did provide the team with some interesting new challenges.

4.5.1 JAUS Integration

The JAUS software is designed to interpret very specific JAUS messages that are sent via a Radio Frequency (RF) data link that adheres to the 802.11g specification. The JAUS messages that Armadillo can interpret are as follows:

- Start the vehicle moving forward in the autonomous mode: resume message <Cmd Code = 0004h>
- Stop the vehicle from moving forward in the autonomous mode: standby message <Cmd Code = 0003h>
- Activate the warning device such as a horn or a light: discrete devices message <Cmd Code = 0406h>
- Start the waypoint information: query global waypoint message <Cmd Code = 240Ch>

Once the JAUS protocol was understood, and once a C# interface was developed for the protocol, another interface had to be developed for the relay output device that controlled the horn warning device. These two software modules were then linked together so that all of the JAUS commands could be properly executed.

4.5.2 Challenges Encountered

The biggest challenge faced during the JAUS integration was correctly reading and interpreting each field of the header in the JAUS messages. Once this was accomplished, the design of the software allowed the JAUS handling code to be easily integrated with the rest of the system. The only other challenge involved figuring out the correct hexadecimal commands to send to the relay output device so that the warning horn would sound. This task turned out to be very simple due to C#'s powerful USB port connection libraries.

5. Performance Analysis and Estimates

Armadillo's performance was measured based on several key areas: speed, ramp climbing ability, reaction time, battery life, and obstacle detection. These areas were assessed through both theoretical estimates and practical test case scenarios.

5.1 Vehicle Speed

The two NPC-2212 permanent magnet motors have a maximum, no-load speed of 285 rotations per minute (RPM). Thus, with 10.5 inch wheels, this means that Armadillo's theoretical maximum speed is 8.91 miles per hour (MPH). In testing, the actual maximum speed was found to be approximately 8.64 MPH. However, in accordance with the IGVC regulations, this speed is limited by the motor controller software to a maximum of 156 RPM, which is just under 5 MPH.

5.2 Ramp Climbing Ability

The driving power of the motors theoretically allows for Armadillo to easily clear the 15% grade requirement of the IGVC competition. During testing, it was found that Armadillo was able to climb inclines of approximately 23% grade, which was more than sufficient for the competition rules. However, due to Armadillo's unique two-component design, it almost did not clear the apex of the test ramps because of the limited flexibility of the central hinge. This could potentially be a problem if the ramp in the competition has a large gap in its apex.

5.3 Reaction Time

The reaction time that was calculated for Armadillo is based on the computational time for a single cycle of visual interpretation and motor control execution. Each component of this calculation is shown in *Table 2.* As this table indicates, the total maximum cycle time was found to be approximately 170 ms, which was sufficient for the successful execution of the image processing and lane following algorithms.

Process	Time (ms)
Image Fusion	40
Color Recognition	60
Lane / Obstacle Recognition	50
Motor Control Command/Execution	20
Total	**170**

Table 2 - Reaction Time Summary

5.3 Battery Life

Armadillo is equipped with an on-board battery charger, which is used to charge the battery whenever it is not in motion. Due to Team Armadillo's choice of low power consumption electronics, it was first estimated that Armadillo could operate for approximately 3 hours between each charging cycle. Through testing, it was found that Armadillo's average battery life lasted around 2 hours and 50 minutes, which was sufficiently close to the desired runtime.

5.4 Obstacle Detection

Armadillo attempts to handle all complex situations early before they become a serious issue. Thus, efficient path planning algorithms are Armadillo's most important tool.

5.4.1 Obstacle Detection Distance

Given the angle and height parameters of Armadillo's cameras, its calculated distance for object detection is about 5 ½ ft. However, the decision making logic limits that distance to around 3 ½ ft, so that objects are not anticipated too early. Thus, objects are detected when they are closer, but this allows the cameras to gather more information about the object since the cameras can see more of the object. This design decision was made because when the object detection distance was not limited, it was found that the software often made bad path decision choices since the object was only just starting to come into the range of the camera.

5.4.2 Traps, Potholes, and Dashed Lines

Traps are the most serious threat to Armadillo because the vehicle can not move backwards, and it has a relatively large turning radius. Therefore, the camera algorithms attempt to evaluate the incoming images as far ahead as possible so that threatening situations can be avoided.

Potholes (both real and simulated) are evaluated as objects, so normal object avoidance procedures are used to avoid them. Switchbacks are easily handled by having a larger range of view from the dual cameras, so theoretically there are not any situations were a turn is not executed because the line "disappears."

Dashed lines, or lines that somehow disappear, are reconstructed based on previous "short-term memory" used by the vehicle's software. This memory stores a limited amount of previous decisions that were made by Armadillo, and in the event that the line disappears, this data helps it figure out where the line should approximately be located.

6. Cost Analysis

Table 3 summarizes the total material cost for Armadillo. As this table indicates, the relative simplicity of the hardware kept development costs relatively low. The most expensive item was the laptop, and this was provided by a team member.

Component	**Total Cost**	**Team Cost**
(1) Dell Inspiron E1505 Laptop	$1,500	$0
(2) Panasonic PV-GS320 3 CCD MiniDV digital camcorder	$750	$750
0.45X wide-angle lens and filters	$120	$120
Electrical hardware parts	$300	$300
(1) Roboteq AX3500 Dual Channel Motor Controller	$395	$395
(2) MPC-2212 Motors, 12V DC power, 285 RPM	$162	$162
(1) Power-Sonic 12V/40AH sealed lead acid battery	$108	$108
(1) ProMite on-board charger, 12V	$88	$88
(1) Omron emergency stop switch	$40	$40
Chassis materials and fiber glass materials	$400	$400
Miscellaneous hardware (nuts, bolts, etc…)	$150	$150
(2) 10" wheels and 10.5" tires	$130	$130
Total	$4,143	$2,643

Table 3 - Cost Breakdown of Armadillo

7. Conclusion

Lawrence Technological University's Armadillo Autonomous Vehicle Team has brought together a diverse combination of undergraduate and graduate students, both of engineering and computer science backgrounds, to continue its tradition of excellence in producing high-quality autonomous vehicles. Armadillo's innovative design and state-of-the-art software technology should set it apart from other competitors in the 15th annual Intelligent Ground Vehicle Competition.