LTU Thinkers II – CogitoBot II

2003 IGVC team from
Lawrence Technological University

Members: David Chamulak

I Tseng

Maurice Tedder

Table of Contents

1. Introduction 2

2. Design Process 2

2.1 Design Philosophies 2

2.2 Team Organization 3

2.3 Design Planning Process 3

3. Hardware Design 4

3.1 Body 4

3.2 Drive Train 4

3.3 Gearing and Motors 5

3.4 Motor Control 5

4. Electrical Systems 5

4.1 Sensors 5

4.2 Handy Board Microcontroller and Sensor Data Processing 6

4.3 Vision Sensors 6

4.4 Power System 7

4.5 GPS Navigation 8

4.6 Computational Hardware 9

5. Software Design 9

5.1 Software Design 9

5.2 Image Processing 9

5.3 Control Architecture 10

5.4 Educating the Artificial Brain 11

5.5 Strengths and Weaknesses 12

6. Vehicle Cost Summary 13

7. Conclusion 13

1. Introduction

Building and designing autonomous robots requires a diverse set of technical and non-technical skills that encompass mechanical engineering, electrical engineering, computer technology, and software design. The availability of low cost hardware for building robots aided by the steady decline in cost for computational hardware has increased the number of attempts to solve the autonomous robot problem. Computational power and hardware has finally caught up to the level necessary to achieve the concepts of robot designers. However, with the same hardware technology available to all, the distinguishing factor in most cases becomes the software design. The software for autonomous robots must intelligently control the hardware so that it functions in unstructured, dynamic, and uncertain environments while maintaining an autonomous adaptability. Therefore, the focus of our design innovation is in the development of software, which utilizes Fuzzy Evolutionary Artificial Neural Network hybrid software technologies to solve the autonomous robot problem. The unique innovative aspect of our Fuzzy Evolutionary Artificial Neural Network hybrid software design is that the robot has an artificial brain to be trained using evolutionary computation rather than a control program that is directly coded.

2. Design Process

2.1 Design Philosophies

CogitoBot II (based on the Latin word cogito for "I think") is the second robot competing at the IGVC 2003 from the Lawrence Technological University Math and Computer Science department. Originally both teams shared the CogitoBot I robot platform to test different software strategies but as the software designs began to differentiate themselves we built a clone to CogitoBot I called CogitoBot II for more efficient testing. Except for the number of cameras, the two robots share the same hardware design; there are however significant differences at the software level. Our CogitoBot II teams primary programming language is Java with the Java Native Interface (JNI) and we are using a Fuzzy Evolutionary Artificial Neural Network software strategy. The sister robot, CogitoBot I, is using Visual C++ with MFC as the primary programming language. CogitoBot I is mainly using a Fuzzy logic software strategy.

In addition to designing two robots to test different software strategies our design philosophy was based on the need to select an on-board laptop computer as the main robot control processor, an independent sensor module based on a separate microcontroller processor with a RS-232 serial interface to the on-board laptop computer, and an independent motor controller that interfaces to the on-board laptop computer with a parallel port interface. We also wanted a modular robot platform that can be used for computer science and engineering education using low cost off the shelf components with the purpose of being mass-produced. The CogitoBot II robot platform was also required to introduce adaptable software systems such as Artificial Neural Networks, Evolutionary computing, Fuzzy, and hybrid software architectures.

2.2 Team Organization

Team Member	Person Hours	Duties	Major
David Chamulak	330	Software Design	BS Physics and Computer Science
I Tseng	300	Software Design	MS Computer Science
Maurice Tedder	350	Hardware Design and Low-Level Software Driver Implementation	MS Computer Science

2.3 Design Planning Process

Our first goal was to develop a prototype robot using a laptop with a camera to follow a line. Our second objective was to build a prototype robot to follow a curved dashed white line. Our third goal was to develop another prototype that detects obstacles on the course by using the camera as well as introducing sensors such as an infrared (IR) distance sensor. Also at the third iteration, we introduced a Pulse Width Modulation (PWM) motor control. Now we are in the fourth stage - optimizing the prototype toward the final version for the competition.

By using off the shelf components with proven durability and reliability we achieved total durability and reliability for the whole integrated system. We also created a simple robot electrical and mechanical design that is more robust than a more complex system. The main difficulty encountered in our design was the selection of a motor driver system. Through experimentation, we determined that the Vantec motor board

with a parallel port interface gave the most reliable performance with more than adequate electrical safety margins to meet our design requirements.

3. Hardware Design

3.1 Body

The robot body was designed and built from off-the-shelf components. The majority of the parts were purchased at a local hardware store. The robot is 3 feet long and 16 inches wide in the middle. The main body was constructed from 0.75 inch thick medium density fiber (MDF) board. The upper body was constructed from 0.25 inch thick MDF board.

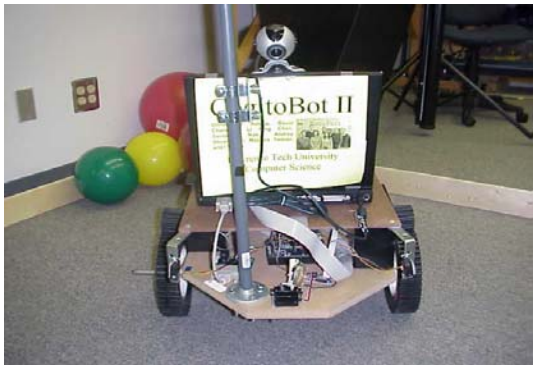

Figure 1- Front View

Figure 2- Rear View

3.2 Drive Train

The drive train was designed to be front wheel drive to allow the robot to climb a ramp easily. The wheel axle was constructed out of a hollow steel rod 22 inches in length. The axle was placed slightly ahead of the center of gravity and a caster wheel was added in the back for additional support and balancing. The main wheels used on the robot were 8-inch lawn mower wheels placed directly on the wheel axle. Each wheel has 51 teeth. This serves a dual purpose. First, it provides better traction on different kinds of surfaces. Second, it allows a direct coupling to a gear mounted on the motor shaft, which is a viable alternative to creating an intricate system of gears and pulleys for the drive system. The axle itself is stationary, while the wheels are allowed to freely rotate on the axle. The maximum incline our vehicle is capable of traversing is highly dependent on the surface of the incline. However, we have tested our robot and found it capable of climbing 18-degree inclines in most cases.

3.3 Gearing and Motors

We used windshield wiper motors as our drive motors. Each motor allows individual speed and direction control. As a result, the robot is able to execute a zero-point turn radius. The motors are worm gear high-torque motors capable of up to 24V DC operation. These motors have a maximum current draw of 4.5-5 Amps. The motor shaft rotates at 188 RPM. We added additional external gearing to the shafts of the motors for greater torque and reduced speed. The 13-tooth gear is coupled directly to the motor shaft and to the wheel on the axle. The resulting 4:1 gear ratio allows for approximately 47 RPM for our drive train. Thus the maximum speed of our robot is approximately 1 mph.

3.4 Motor Control

The motors are controlled using a Vantec CDFR-21 dual-channel driver board. The motor board can handle up to 14 Amps of current per channel, which makes it an ideal component for this particular robot. This driver board provides Pulse Width Modulation (PWM) speed and directional control. The board interfaces to the laptop through the parallel port. Commands are sent through the parallel port to the motor driver board, which in turn interprets these commands and generates a PWM signal that is sent to the motors.

We wrote low-level drivers in the Java Native Interface for accessing the parallel port and sending commands to the motor driver board. These drivers present the programmer with a high-level functional interface to the board. The programmer makes a call to functions such as forward() or reverse(), and the driver parses these commands and sets the desired bits on the parallel port, which in turn forces the motor driver board to send the desired PWM signal out to the motors.

4. Electrical Systems

4.1 Sensors

Our design innovation for additional obstacle detection consists of using two different types of sensors as a means of detecting obstacles. A rotating Polaroid sonar sensor and two Sony GD2D02 infrared (IR) distance sensors provide obstacle detection. We chose these sensors because they are a proven technology, have high reliability and

we have previous experience using them. The IR sensors are very robust at detecting objects less than two feet away and remain in a fixed mounted position on the front of the robot. There is one sensor for the left side and one for the right side. This arrangement provides continuous feedback for the left and right sides of the robot to identify an immediate turn, if necessary, to avoid an obstacle. The IR sensors are primarily a backup detection method for the Polaroid sonar sensor module. The sonar provides long-range directional obstacle detection up to 35 feet (7 feet reliably) for a 180-degree scan zone capable of detecting objects three inches apart. Rotating the sonar gives the robot range and angular position data for a detected obstacle. Since our robot only has one sonar sensor, we use a servomotor to rotate it and scan at variable angles.

4.2 Handy Board Microcontroller and Sensor Data Processing

As part of our design requirement, we did not want the laptop CPU involved in any low-level sensor processing because it would require extra CPU time that we needed for vision processing. The IR and Polaroid sonar sensors do not provide data output directly but must instead be driven by a separate microcontroller for this function. The microcontroller handles the lower level sensor processing functions without requiring any processing power from the laptop's CPU. In addition, the laptop is not equipped with sensor drivers and digital/analog I/O circuitry necessary for handling IR and Sonar sensor data. We chose the Handy Board as our micro controller because we have previous experience using it and it meets all the needs stated above. The laptop computer receives IR and sonar sensor data from the Handy Board via our custom ASCII data packet protocol transmitted though the RS-232 serial port connection. This ASCII serial port format has the advantage of computer hardware and software independence because as long as the laptop has a serial port and the programming language supports serial port I/O then any arrangement of computer (PC or Mac), operating system, or language (Java or C++) can use these sensors for obstacle detection. The Handy Board can also support up to four IR sensors, four servomotors, and additional I/O as our future needs expand.

4.3 Vision Sensors

To reduce the complexity and cost of providing computer vision hardware for our robot design, we selected camera equipment already available in our lab and free

software available from the Java API to interface with the camera and process image data. Detection of objects is provided by a Logitech 160 X 120 color CCD camera capable of supplying 30 frames per second using a USB interface. No other hardware is needed except for a laptop with a USB port. Capturing and processing image data is accomplished using the Java Media Framework that is provided as a part of the Java API. This library provides functions for handling camera data and provides software interface functions for capturing, processing, and displaying camera data. The main functions we use are the camera device driver function and the image frames capture functions, which give the programmer access to image pixel data.

4.4 Power System

One 12-volt 7amp hour battery (or two in series for 24-volts) is used to provide electrical power to the drive motors, Vantec motor driver board, and Handy Board external power. Based on experimental test data we found that our battery can last for 1 hour before being replaced by a newly charged battery. The GPS unit and the laptop computer both have their own rechargeable batteries separate from the main robot power system. Although this design requires the maintenance of separate batteries, it isolates and protects the laptop and GPS from the unlikely event of a dangerous electrical surge in the robot power system. A fifteen-amp fuse protects the entire robot electrical system and Vantec motor board from over currents. An additional 500mA fuse protects the Handy Board microcontroller from over currents. For safety reasons and as part of the IGVC qualification requirements, manual and remote electronic emergency stop switches are wired in series with the battery connection. Thus, both switches must be closed for electrical power to the main system power bus. Any one of these switches can be opened to cut off all electrical power and stop the robot in the event of a dangerous loss of control. The remote e-stop consists of an automotive keyless entry remote switch unit with a range of 50 feet and a 30-amp relay in series with the battery connection to the main power bus. One design innovation that we implemented is a logical electrical system layout that is made self-explanatory by proper labeling and aids in on the spot troubleshooting. These safeguards designed into our electrical system also protect against electrical failures due to human error. Figure 6 shows the robot's electrical block diagram.

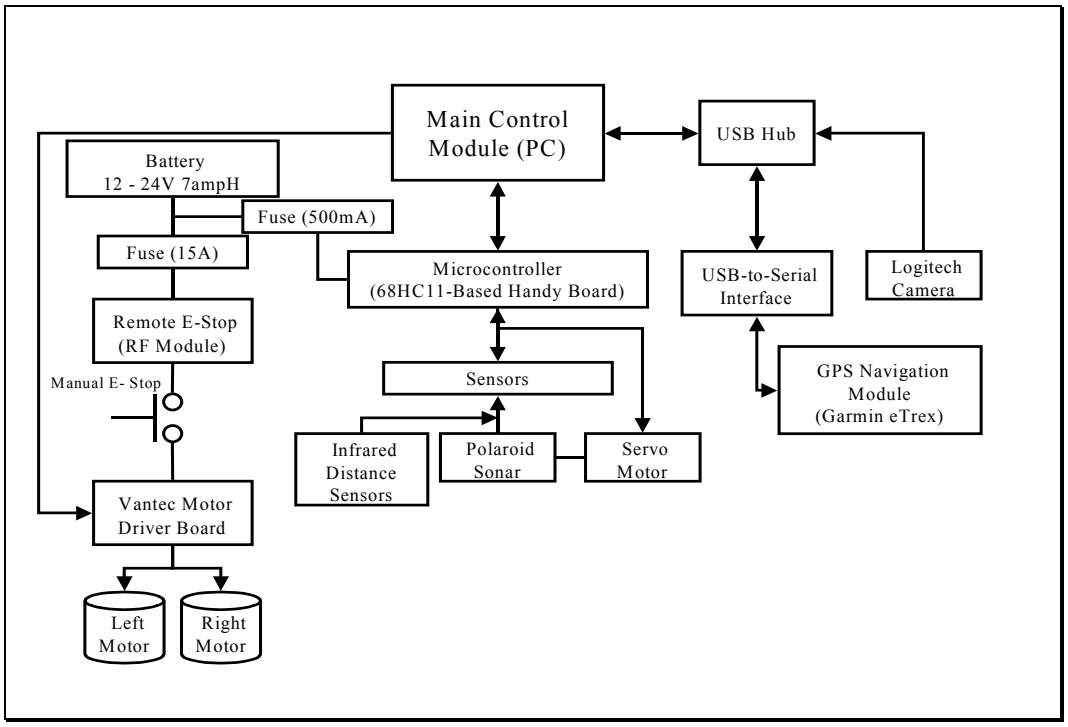

Figure 6 – Electrical Block Diagram

4.5 GPS Navigation

To effectively compete in the navigation challenge we decided to use a Global Positioning System (GPS) to navigate the course. The Garmin eTrex Venture handheld Wide Area Augmentation System (WAAS) enabled GPS unit was selected for this purpose because it provides a wide range of features at a reasonable price. The built in features of the eTrex GPS unit provides all the functions necessary to navigate the robot autonomously. It allows us to set the destination coordinates and continuously provides feedback indicating the direction and distance to the desired waypoint. This information is used in a feedback loop with the robot motion control system to guide the robot through any course. All that is needed to use the navigational features of the eTrex unit is to interface using the RS-232 serial port and the National Marine Electronics Association (NMEA) protocol for sending and receiving data from the eTrex GPS unit. If an obstacle is detected in the path of the robot then the obstacle avoidance routine assumes priority over GPS control of the robot and uses a reactive scheme to steer the robot away from obstacles. Once the obstacles are avoided, GPS navigation control of the robot resumes. The advertised accuracy of the GPS unit in WAAS mode is less than three meters, which is the actual accuracy of the robot.

4.6 Computational Hardware

The onboard computer used for all vehicle control, sensor interfacing, and communications is a laptop computer that uses a Pentium III 600 MHz processor running the Windows 2000 operating system. The laptop has a single USB port that is extended by a 4-port USB hub for connecting the camera and other peripheral devices, a parallel port for connecting the motor driver board, and a serial port for connecting to the microcontroller, which in turn is connected to all of the on-board sensors. The laptop is connected to a virtual private network through a wireless LAN system. This allows for remote debugging and observation by a team of programmers.

The Handy Board's features include a 2 MHz Motorola 68HC11 microprocessor, 32K static RAM, servomotor controls, digital and analog I/O, and a RS-232 serial port for communicating with the laptop computer.

5. Software Design

5.1 Software Design

For the programming of our robot we decided to use the Java programming language. Java allows us to easily add new features to our program as well as allowing our program to be platform independent. For our program we used the Java Development Kit (JDK) 1.4.1_02 as well as the Java Media Framework (JMF) and the Java Communication API add on. The JMF allowed us to collect camera images from our camera and the Communication API allows us to access the serial and parallel ports.

5.2 Image Processing

The processing procedure comprised of first converting the captured image to black and white. We first checked each pixel's RGB value in our 160x120 image and if those values were above a preset threshold value, that pixel was set white; if they were not, that pixel was set black. Next we broke up the picture into 5x5 pixel blocks and counted the number of white pixels in each block. If one third or more of the pixels were white in the 5x5 block, then the entire block was set white; otherwise the entire block was set black. Therefore, through our processing procedure our original 160x120 color image was transformed into a 32x24 black and white image. Figure 7 shows a preprocessed and

post processed picture. This processing routine allows us to detect both solid and dashed lines as well as potholes, obstacles and traps up to about 8 ft.

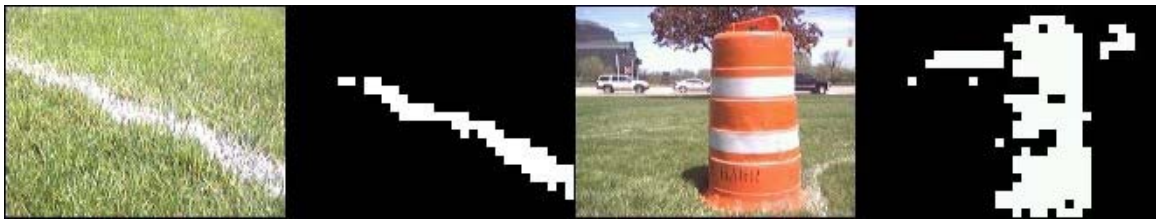

Figure 7- Preprocessed and post Processed Images

5.3 Control Architecture

With the processed image we were able to determine how the robot should move, or more precisely, the robot was able to determine how it should move itself. This is possible because for the control of our robot we programmed a Fuzzy Evolutionary Artificial Neural Network (FEANN) or, in other words, a synthetic brain. The neural net is comprised of processing units or neurons. The image data from our processed image is first fed into a series of neurons. Those neurons are networked to other neurons that process the data even further. The output from the final set of neurons is a fuzzy set or a probabilistic set of data to determine the direction in which the robot should go. After analyzing the probabilities of going in different directions, the robot will steer in the proper direction. The time it takes this for this process to complete gives our robot a reaction time of 50 ms with our current hardware. Figure 8 shows an example of our FEANN.

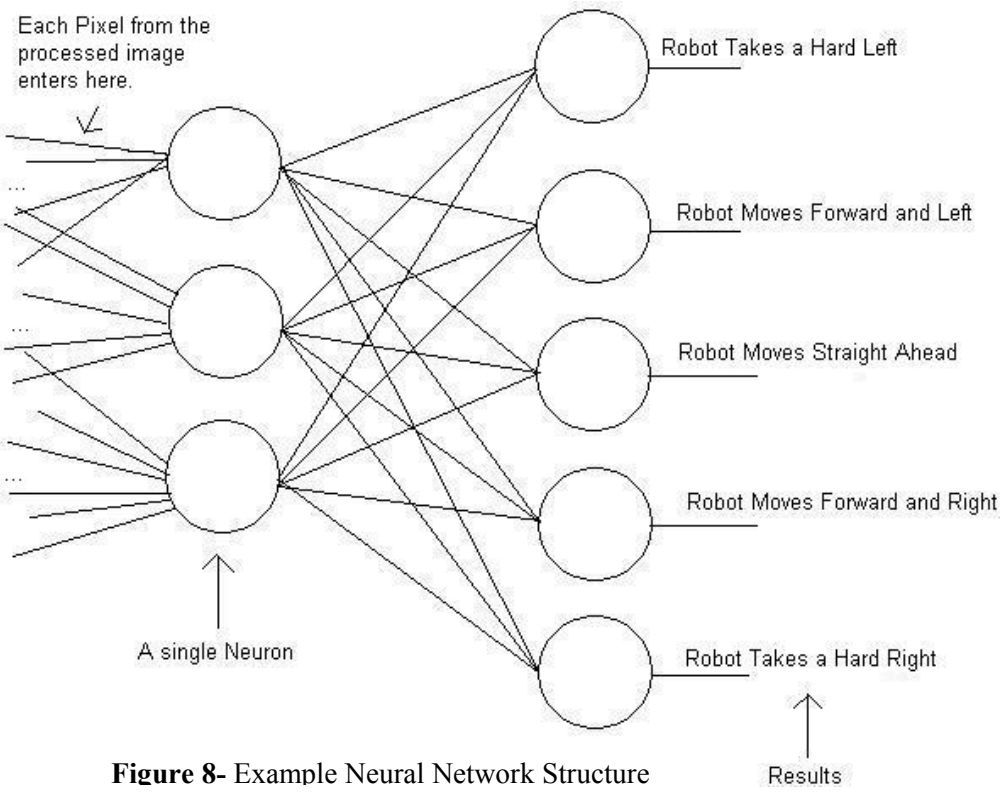

Figure 8- Example Neural Network Structure

5.4 Educating the Artificial Brain

In order for our robot to achieve this level of sophistication, it required training in how to act. Training of our robot took many steps. First, with a human at the controls, we maneuvered our robot through a practice course many times. Each time through the robot kept track of the movements it was making as well as the image it was seeing at the time of the movement. That information was then taken and, using an Evolutionary Strategy ES (1+1) with modified 1/5 rule, we evolved an artificial neural network. In other words, we created many artificial neural networks, and using the idea that the strong survive; kept the ones that best matched our sample data. Those networks were then used to create more networks that we then compared to find the best. Once we found an artificial neural network that fit all our sample data we saved it so we could use it whenever we needed it. All of this work could be thought of as educating our synthetic brain how to control the robot. The fully trained FEANN can decide how to move the robot by itself. Figure 9 and 10 show a flow chart of the training process as well as the control structure.

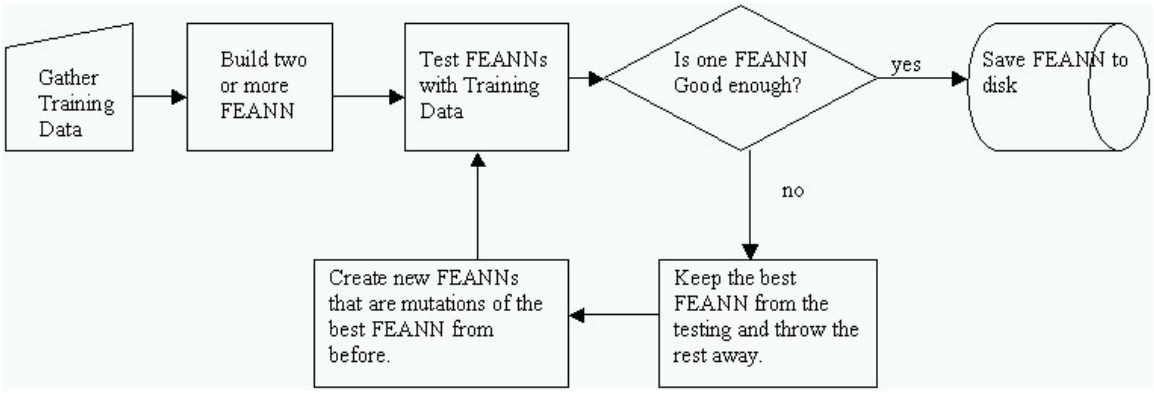

Figure 9- Flow Chart of the Training Process

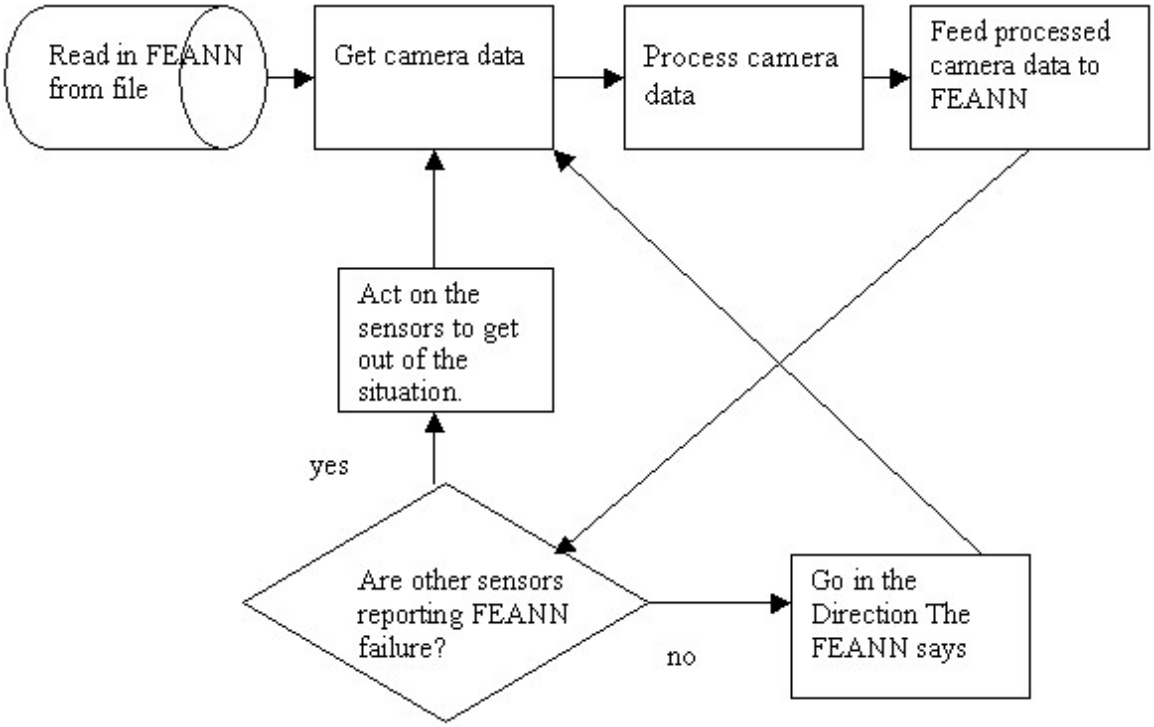

Figure 10- Flow Chart of the Control structure

5.5 Strengths and Weaknesses

The strengths of this design by far make up for any weaknesses. Since the FEANN is an artificial brain it models more closely what a human does in controlling the vehicle. Even in situations that the robot may have not seen before the FEANN can make a decision on how to act, just like a human would be able to. Also, since we save our FEANN to a file, we can train different FEANNs to do different tasks and read in the one that best fits our specific task. Theoretically, in future years, multiple FEANNs could be combined to create an even larger FEANN capable of doing much more complicated

tasks. The only weakness to the FEANN is making sure that enough training has been done so the FEANN can make sound decisions on its own. We have taken care of this by relying on the other sensors of our robot to tell us if and when our FEANN has failed and act on those sensors until the FEANN can cope again.

6. Vehicle Cost Summary

The robot was designed to be low-cost and modular. Thus our expenses were kept as small as possible. Table 2 summarizes our expenses for this vehicle.

Component	Purpose	Cost Per Unit	Number of Units	Total Cost
600 Mhz Laptop Computer	Main Processing Unit	$500.00	1	$500.00
Vantec CDFR-21	Motor Driver Board	$250.00	1	$250.00
Handy Board	Microcontroller for Sensors	$250.00	1	$250.00
12 Volt 7 Ahr Battery	Main Electrical Power Source	$22.00	2	$44.00
12 Volt DC Motors	Provides Power to the Drive Train	$20.00	2	$40.00
GP2D02 IR Sensors	Infrared Distance Sensors	$15.00	2	$30.00
Polaroid 6500 Sonar Sensor	Ultrasonic Distance Sensor	$48.00	1	$48.00
0.75" Particleboard	Lower Portion of Robot Structure	$3.00	1	$3.00
0.25" Particleboard	Upper Portion of Robot Structure	$2.00	1	$2.00
8" Lawn Mower Wheels	Main Tire Assembly	$8.00	2	$16.00
Logitech Quickcam Pro 4000 Camera	Provides Visual Information	$99.95	1	$99.95
GPS- Garmin eTrex Venture	GPS Navigational Sensor	$150.00	1	$150.00
Miscellaneous	Connectors, bolts, hardware, ect.	$100.00	1	$100.00
Total				$1,532.95

Table 2 – Vehicle Component Cost Summary

7. Conclusion

This is the first year that Lawrence Technological University is participating in the IGVC. We have two teams with almost identical robot hardware. One team is using C++ and fuzzy logic methods and our team is using Java and Hybrid Neural Network methods. In addition to competing against other schools at the IGVC 2003 we will also be competing against our sister team. The purpose of this internal competition is to

conduct an experiment to determine which software methodology is the best, C++ versus Java and Fuzzy logic versus a Neural Network hybrid system. By having both CogitoBot I and II based on identical robot hardware, we can eliminate the choice of hardware variable from our experimental analysis. We believe that our software methods are unique based on the reports from previous IGVC competitions and may provide us with an innovative edge and additional data to compare competing autonomous robot software methods

In this design project we accomplished our goals of implementing unique software strategies with low cost robot platform that can follow a line while avoiding obstacles. It also has the capability to navigate autonomously using GPS navigation and correct its course based on obstacles in its path. For future work we will change the robot drive train to increase its maximum speed to 5 MPH so that we will be able to compete in the follow the leader competition. A summary of the actual robot performance specifications is listed in Table 3.

Length	3 ft
Width	1.33 ft
Height	2.5 ft
Motor RPM	188 RPM
Motor Voltage	24V
Motor Stall Current	4.5Amps
Motor Stall Torque	11 ft-lbs
Motor Power Output	0.1 hp
Max. Speed	1 MPH
Gear Ratio	4:1
Wheel diameter	8 in.
Traversable Incline	18 deg
Battery Life	1 Hour
Waypoint Accuracy	< 10 ft
Obstacle Detection Distance	8 ft
Maximum IR Sensors Distance	< 3 ft
Maximum Sonar Distance	7 ft
Reaction Time	50 ms

Table 3 - CogitoBot II Specifications

Faculty Advisor Statement

I, Dr. Chan-Jin Chung of the Department of Math and Computer Science at Lawrence Technological University, do hereby certify that the design of the vehicle, CogitoBot II, by the LTU Thinkers team, has been significant and each team member has received credit from this University for the work completed.

Signed,

Dr. Chan-Jin Chung
(248) 204-3504