Bluefield State College

ENIAC 2001

Ground Robotic Vehicle Competition

Faculty Advisor:

Dr. Bob Riggins

Team:

Paul Richards, Arthur Ball
Omar Abdulla, Joe Fink
Joshua Fowler, Tom Lambert
Ganesh Sakshi, Jeremy Moore
Saurabh S., Nick Killen
Garry Stutts, James Heath
Brian Toler, Anthony Crews
Jason Vest, Jerry Pugalee
Joe Presley

Table of Contents

Section 1 Introduction

Ideas for the design, analysis, and construction of ENIAC 2001 were born from the lessons learned in last summer's competition. Last year, the team did not have time to do a complete analysis on all the systems, and some of these systems failed in the competition such as the motor control circuitry and the steering motor. Also, the robot body itself had structural problems. This year, the team started all over, using an old go-cart frame as the beginning point. The major change this year was a shift in focus from a "let's just make it work" attitude to an analytical attitude. A few of the ideas from last year carry over to this year; for example, we

will still use diffused visible sensors to detect the road edges rather than cameras, since most other teams use the cameras, but no one else used our approach. Most of the new systems, however, are different from last year. We use a different steering system, we have totally re-designed our computer to motor interface, we have written a completely new code, and all the structure is new. A new method of implementing the road-edge detectors will sharply increase the reliability of these sensors by introducing fault tolerance through redundancy. If this robot were to be duplicated, the total cost would be about $6000. However, this cost for us was even lower since many items were donated by local sponsors. Sponsors include Pemco, Advanced Auto, Charlatte, Avalon, Pocahontas Coal, and the Elk's Lodge. Estimated person-hours on this project is about 500 hours total.

Section 2 Overview of ENIAC 2001

Several different systems had to be incorporated into one self-contained unit for ENIAC 2001. In this section, we will focus on the systems that changed the most from last year's robot. The first system of discussion in this section will be the onboard computer, which is the brain of the robotic vehicle. This computer, consisting of a 733 MHz processor, a 10 Gbyte hard drive, and a 256 Mbyte RAM chip, houses the program that commands the vehicle. Furthermore, a memory management program was installed to free up RAM and allocate it back to the operating system. This is all powered with a computer backup battery system.

The program that operates the vehicle is written in Microsoft Visual Basic. This program allows the vehicle to take input data from the onboard sensors, analyze it through its many lines of code, and then make a decision on what to do with the data. In this case, the vehicle will avoid an obstacle or try to remain on the course. This is accomplished through an input/output card installed on the motherboard of the computer. This card controls a bank of relays, which controls the circuitry that controls the drive train. Another system onboard the robot is its detection device.

Presently, there are two different detection devices on the vehicle. The first is a mini-beam photoelectric switch (diffused visible) that was designed to respond to color, and sonar sensors that were designed to determine how far an object was away from the vehicle. These sensors can be considered the eyes and ears of this vehicle. To further these vehicle capabilities, a global positioning system (GPS)., will be installed to help in the navigation process. Future systems to be incorporated on this vehicle are video cameras and battery charging systems (next year). Since this vehicle will be autonomous, an emergency stop will be installed on the vehicle in case of a malfunction.

The last system to be discussed in this section is the steering (or turning) motor. This motor is a stepper motor designed to turn in or out a software-specified number of revolutions. If the program sees an object with one of its

sensors, it will command the stepper motor to turn to avoid a collision or remain on the course. With all of these systems integrated into one application, the configuration and design had several pre-existing problems and some new ones as well.

One major problem we had to overcome was in the control of the drive train. After the initial start of the system we would be in control, but once the system was engaged, our control was lost. After many tests, it was determined that the relays that controlled the motor contactors were latching closed and that prevented the computer from having control. Using some Darlington pair transistors in place of the control relays rectified this. See Section 5 for more details.

One big task the team had to face was installing and understanding how the steering motor operated. Once we understood its operation, we still did not have an idea how to interface it with our control program in Visual Basic. After several calls to the manufacturer (and plain ingenuity), the problem was solved, mostly with a change in the code.

Another problem the teamed had to deal with was that the sensors did not provide a fail-safe in case of a malfunction. What if one of the diffused visible minibeam sensors did not pick up the road edge? Paralleling the mini-beam sensors so that if one failed the system still had others to rely on rectified this. One of the team's great accomplishments was the design of the bumper guard that

surrounds the vehicle. This bumper will house the mini-beam sensors and provide the computer the time required to make its decisions without running off the course.

Section 3 Motor and Gear Analysis

In this section, we outline the analysis we did for determining what type and size motor to purchase, as well as how to build the sprockets and gearing we needed to meet the maximum 5-miles-per-hour requirement.

Assumptions made to determine motor size:

- 25º incline
- No friction
- 300 lb. Weight
- 3 mph velocity (later, we geared it for a maximum of 5 mph)

F = (300 lb.)(sin 25º) = 126.8 lbs. (robot weight estimated at 300 lbs)
P = FV = (126.8 lbs.)*3*(5280/3600) = 557.92 ftlbs/s.
P horsepower = 557.92 / 550 = 1.014 hp

So we used a 1 hp motor since our maximum inclination is only 15 degrees.

Assumptions to determine gear reduction:

- 12-inch wheels
- Typical velocity of 3 mph
- Maximum motor speed of 1800 rpm

3 mph = 4.4 ft/s
1 wheel revolution = 1 ft. * pi = 3.14 ft
4.4 ft/s / 3.14 ft/rev = 1.4 rps
1.4 rps * 60 = 84.1 rpm
reduction = 1800 rpm/84.1 rpm = 21.14 : 1

So, we used a 20 : 1 reduction with two sprockets.

Section 4 Sensors

We use two types of sensors for the robot: sonar for detecting objects in the path, and diffused visible minibeam (Banner) sensors for detecting the road edges and the potholes. Since the sonars are the same ones we used last year, our analysis is the same as then. We also used the banner sensors last year, but this year we did a complete analysis on these sensors, and changed how we use them. Therefore, we will focus on the banner sensors in this section. Also, we will be a little more lengthy in this section since using these sensors is not as well known as using the other more familiar methods, such as the cameras.

For the edge detection circuit, we focused on a specific switch: the color-sensing switch by Banner Engineering. We needed to determine the following: how the sensors detect color, response time, and their detection envelope and range. Through a series of tests, we showed the effectiveness of the sensor and its optimum placement in the control of an Intelligent Ground Vehicle. We shall explain in brief detail the mounting rig, the electrical circuit used in the analysis of the Banners, and the methods of analysis.

The first area of our discussion is the Test Rig used to determine the Banner sensor envelope. The rig consists of a neutral base, a mast, a power supply, and a meter stick. The mast is where the LED indicating circuit, meter stick, and power

supply are mounted. The meter stick is mounted vertically on the far edge of the mast. For our test purposes, we drilled holes on the inch marks of the meter stick, starting from 4” and continuing through 20”. We used these holes to mount the sensor to the mast. From the center of each hole, a 45° guideline is etched into the meter stick, pointing downward toward the neutral base. The meter stick is attached to the mast to provide height indication and sensor positioning. The base holds the most vital function in analyzing the sensor’s capacity. We marked the point where the base and the meter stick join together. This is where the sensor would shine if it were pointed straight down. This point is the center. Over a range of 90°, we drew 15° angles on the base. Along these lines, we placed 3 to 5 guide pins for precision angling during testing. The base and the lower half of the mast are painted flat black. We chose flat black because unlike gloss, there is minimal or no reflective shine, which could alter the true test data. Black will be the off-activated color of the sensor during test procedures. A color from the opposite side of the spectrum will be the on-activated color. The base and the mast are in the target area of the sensor, and therefore must be the same color as the off-activated color of the sensor. Due to extreme sensitivity of the sensor, care must be taken to avoid interference in the target area of the sensor.

The next topic of discussion is the electrical circuit. The test circuit is comprised of a 12 Vdc source provided by the power supply, the color sensor

switch, a current limiting resistor, and a Light Emitting Diode (LED). The sensor has two states, on-activated and off-activated. The sensor sends out an infrared beam. This beam is reflected back to the sensor by colors. Light travels in waves, and light reflects off of objects around us. Every object has a color pattern that reflects light in a range of frequencies. Some frequencies are close enough together that they can be confused with each other. This makes the sensor very sensitive to colors that have close frequencies. The sensor is pointed separately to two different colors. One is programmed for off-activation and the other is programmed for on-activation. The infrared beam reflects off of an object at a specific frequency and back to the sensor. If the frequency matches that of the on-activated color, the switch will close and the LED will illuminate. The reflected frequency is determined by the natural make-up of the object that reflects the infrared beam. Once the object moves out of the target area, the sensor will return to the off-activated state. The Visual Basic program was organized so that the required white/green contrasts and the white/black contrasts could be detected with the banner sensors. The results of the testing are tabulated below.

Through the control of a simple LED circuit and a homemade test rig, we were able to acquire essential data about this sensor. We set out to prove the effectiveness of the color sensor and what would be the optimum tactic of mounting the sensor on the IGVCS robot. We first found that the sensor can detect

a 1” square from 13” away. However, this was unreliable because the beam had to be reflected directly back at the sensor and is not very practical. This means that the robot could move into a danger area without notification from the sensor. We tested the sensor over a range of 16” to see where the sensor appeared most reliable. Any distance greater than 10” required a direct reflection and any distance less than 6” would put the robot too close to a danger zone with little time to recover. We presumed that the infrared beam emitted by the sensor would create an “eye” or target area. After our first portion of testing, we concluded that the sensors target area is very narrow and extremely precise. No data could be

Sensor Mounted 6" above BASE

Angles of Measurement	Base Measurements*	Distance from Sensor Eye to Target*
30°	3.50	6.95
35°	4.20	7.32
40°	5.00	7.81
45°	6.00	6.00
50°	7.20	9.37
55°	8.50	10.40
60°	10.40	12.01

*All measurements are in inches

attained for this theory. Next, we began testing on the AOA or angle of approach. We pointed the sensor toward the base, and took measurements over a 30° range at 5° increments.

Considering the maximum speed that the robot will be traveling (5 mph), we calculated that if the sensor picked up a danger zone 8.5" away, this would give the robot one tenth of a second to respond or stop. We also considered that the sensor would also be mounted on a bumper, which would give the robot wheels approximately 6 more inches. We feel that an angle of 55° from the deck is the optimum angle to mount the color sensors on the robot. This should give the robot ample time to respond and recover from boundary encroachment.

Section 5 Computer-to-Motor Interface and Control

For interfacing the computer to the propulsion motor, we use an input/output (I/O) card that increases the number of parallel ports. Also, we use a bank of 8 relays attached to the I/O card. These relays are small computer relays that cannot take too much current through its contacts, so we cannot just simply connect the relays directly to the propulsion motor contactors (they take about 1 amp of coil current).

In last year's competition, we used another set of relays as the "go-between". However, we had loads of problems then, and so this year the team analyzed this problem and found that the coils of these "go-between" relays had an inductance that was tripping the propulsion motor contactors. Therefore, we had to design and build a control circuit consisting of darlington pair transistors. These

transistors are able to drive a large load with a small input signal. We designed three such circuits, one for the main contactor that turns the propulsion motor on and off, one for commanding forward motion, and one for commanding reverse motion.

Section 6 Integration of GPS

GPS integration into the GRV had three main obstacles in the development of the GPS integration program. The first was to design an algorithm that would be able to calculate target distance, target location relative to current vehicle location, vehicle drive direction, and correction of the target locations using a pseudo-differential positioning system. The latter was possible because the starting point, as well as the target locations, would be given before running the course. (Note, this is not differential GPS.)

This starting point would be determined using the same GPS receiver as was used to find the four targets. Therefore, we would be able to use this point to determine the error difference between the point given and the location information that the GRV was receiving from its own GPS receiver. This difference would be an approximate constant for all the locations given and so can easily be subtracted out in the algorithm. The easiest way to do this was to subtract it from the target locations rather than trying to subtract it from the data stream coming in every second from the onboard receiver.

The second obstacle was being able to control the steering motor on the vehicle at the same time the GRV's position and direction were determined. This was a hindrance at first since little was known about the steering motor. Once we figured out how the Pulse Width Modulated (PWM) controller interfaced with the computer and the motor it drove, we could integrate its commands into the Visual Basic Program. This ended up being rather difficult since the initiation string for the PWM controller required carriage returns, line feeds, and motor instruction addressing. The motor is a powerful linear motion stepper motor that can deliver a peak output of 300 lb_f.

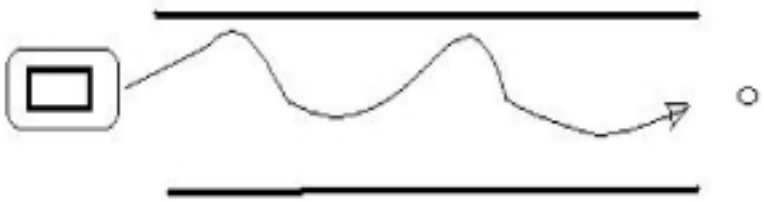
The third obstacle was a need for the GPS program to also control the GRV drive motor. This motor is activated by a series of weighted relays that in certain combinations can engage the main contactors for forward and reverse directions in two speeds - high and low. The maximum allowed speed for the vehicle in the competition is 5 mph and ours will run just under that in high speed. The drive motor requires a line contactor to be thrown before the forward or reverse directions can be selected. This requires two initiation strings, one for the line contactor and the second for forward or reverse. The program execution time was too fast for the mechanical relays to activate in the proper order and so in order to avoid a timing issue, a two-second loop was put in the program.

Integrating these controllers into one program proved to be a difficult task. Visual Basic's main functions are timer dependent. Keeping the three necessary timers sequenced properly is very difficult. In an attempt to get around this, we made drive motor subroutines with their respective timers so that they could be called out from the main program. The first portion of the main program was the information retrieval from the CMC Marconi GPS receiver. The latitude, longitude, local time converted from UTC time, altitude, and speed parameters are put into label fields in the main form and are updated once a second - the same timer interval the CMC uses.

This is followed by a few lines of code to plot out the data received so there can be a visual representation on the computer monitor of what the GPS program thinks it sees. This way, tracking of the data stream coming in can be easy. The same data is then used to calculate the difference in position of the vehicle and where it should be. The algorithm was made so upon first departure from the starting point, the vehicle would drive approximately 10 feet from the point to get a position vector. The program will use this vector to determine the direction the robot is pointed in - in case it is not in the direction of the first target. A simple subtraction then will invoke the turn-left and drive-forward motor commands until the position vector is pointed towards the vicinity of the first milk jug.

Once this happens, we programmed a tolerance for determining whether to

go straight or not and for how long. Since GPS is not incredibly precise, even with our pseudo-differential GPS code, we cannot make the direction vector a straight line for projected drive path. It will have to be a band of at about 15 feet for optimal maintenance of a straight course, allowing for the inherent deviation with GPS coordinates. This way, the vehicle will be able to tack in accordance with positioning errors to its destination target and not constantly have to rely on the algorithm to tailor a perfectly straight path – an impossibility anyway. See sketch below for view of GPS vehicle headed towards target along projected GPS path:



In order to simplify the empirical adjustments to get the turning rate and drive time correct, we have inserted some factors into the algorithm that can be changed on the fly in the program form. This way, last minute adjustments can be carried out without too much difficulty and the trajectory can be tailored to the local conditions, i.e., turning lag due to thick grass or excessive drive speed due to poor traction situations. Hopefully, thanks to these factors, there will be no need to refer to the source code to make changes.

I certify that the engineering design in the vehicle by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.