Precision Robotics Platform

Randall M. Satterthwaite
Rob. Shockency

Project Advisors

Dr. B. Huggins

Mr. C. Mattus

May 1, 2002

Contents

I. Abstract

Many robotics applications require a low cost mobile platform that can be customized by the user. The design, construction and testing of such a platform (known as Traxx) is described in this report. Traxx consists of 3 subsystems: the microcontroller, the digital logic feedback, and motors with power electronics. An 8051 microcontroller provides the control system and memory storage for the commands. Traxx utilizes closed loop feedback via motor encoder wheels to allow for precise movement. The feedback of the system is near real time since it utilizes digital logic in the form of two complex programmable logic devices (CPLD) that were designed using VHDL. There are two DC motors that make up the electromechanical subsystem. They are controlled directly by two H-Bridge transistor arrays, which provide direction control and power amplification. The complete system was verified and calibrated using a test program.

II. Introduction

The Electrical Engineering Department purchased a robotic platform for about $3000 called Pioneer 2. This platform allows to precise control in tele-robotics applications, however the cost was quite high. The main purpose of this project was to build a precision robotics platform that cost considerably less then the Pioneer 2, and was versatile for use in future projects.

This platform is a basic close loop controlled vehicle that is capable of very precise movement (1cm and 1degree) while being able to be upgraded for future applications. The vehicle is completely self contained, and is controlled via a microprocessor and two CPLDs.

III. Functional Description

Inputs and Outputs

Inputs

Inputs to the system are obtained from a keypad via a series of key presses. The input will be in the form of a distance or turning angle, associated with the desired movement. Movements allowed are: forward, backward, turn left, and turn right. The keypad also selects the desired movement. These are described below in the user interface section

Outputs

There are two outputs: vehicle movement and LCD data. Vehicle movement is the end result of all keypad inputs. The status of each motor is displayed on the LCD, along with several user interface menus associated with specific keypad commands.

User Interface

The user programs the vehicle from the onboard keypad. The user interface is very simple, due to the nature of the coding required. To program the vehicle, the user has several choices:

- By pressing 2, the forward movement screen appears
- By pressing 8, the backward movement screen appears
- By pressing 4, the turn left screen appears
- By pressing 6, the turn right screen appears

After selecting the desired movement, a new menu is displayed, which provides further instruction to the user. The user then enters the desired distance or angle, and presses “E” to execute the command. Key “F” can be pressed to exit the current menu. The UI is very simple, but very effective. The UI has been thoroughly tested, ensuring the robot executes

the proper command entered by the user. The robot also displays the status of the motors: speed, distance and angle turned, while the robot is moving. The user can read the display to ensure the proper function of the vehicle.

Specifications

Table 1. The specification for the Precision Robotics Platform.

Vehicle

Dimensions (L x W x H): 31.4cm x 46.4cm x 21cm
Weight (approx): 28lbs
Power: Dual 12volt, 7.2Ah Lead Acid Batteries

Drive Train

Drive: 2 Pittman GM9236 Series Motors with 1:65.5 Gearing
Drive Wheels: 5cm Wide, 16cm diameter Soft Foam
Top Speed: 25 cm per second
Minimum Speed: 5 cm per second
Speed increments: 0.005cm per second
Turning: 1degree increments, 128 360degree turns per command
Distance: 1cm increments, 256 meters max per command

Control System

Steady State Error < 3%
Overshoot < 20%
Settling Time < 1 second

PWM Divisions: 1791 or .055% increments
PWM Frequency: 500 Hz
Motor Control Input Signal: 0-255 counts

IV. System level Block Diagram

To implement the functions described in the previous section, several subsystems need to be designed. These are shown in Fig. 1. The main subsystem is the microprocessor, which implements the control system, user interface, PWM, and any diagnostics required. The CPLD subsystem is used to provide feedback about the speed, distance traveled, and angle turned.

The microprocessor provides all signals required to control both motors' speed and direction based on user inputs and feedback from the CPLDs. It contains a user interface that allows the user to program in commands one at a time for immediate execution. It also continuously monitors data from the CPLD which is used as feedback to ensure that the commands are executed properly. See Fig. 2 for details of the process loops that are utilized.

The CPLDs are the interface between the motors and microprocessor. It accepts and moves data as shown in Fig. 3. First, it receives a request from the microprocessor for data from a register via 3 address lines. The registers hold a distance count (number of cm traveled), angle count (number of degrees turned) or frequency count (speed of the motors) which is derived from the motor encoder wheels. When the addressing matches the address of the register (registers are addressed using 3 address lines from the microprocessor giving them addresses 0 through 7) and bus command lines (select and read bar must be low to access the registers) are correct, the data is put onto the microprocessor bus and received by the microprocessor.

The H-bridges amplify the PWM signal for use by the motors along with controlling direction, and providing power and charging abilities for the whole platform.

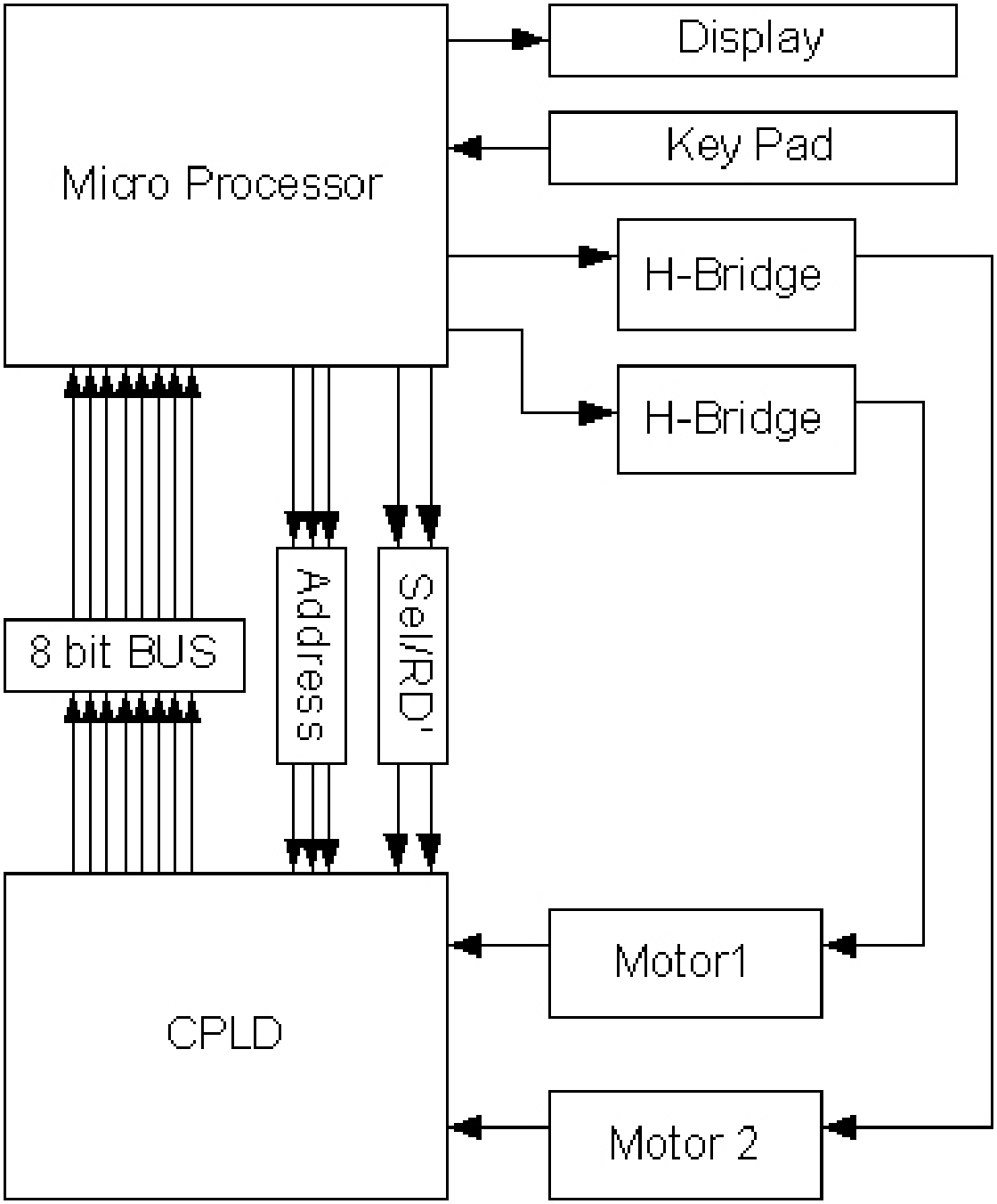

Figure 1. The complete system block diagram.

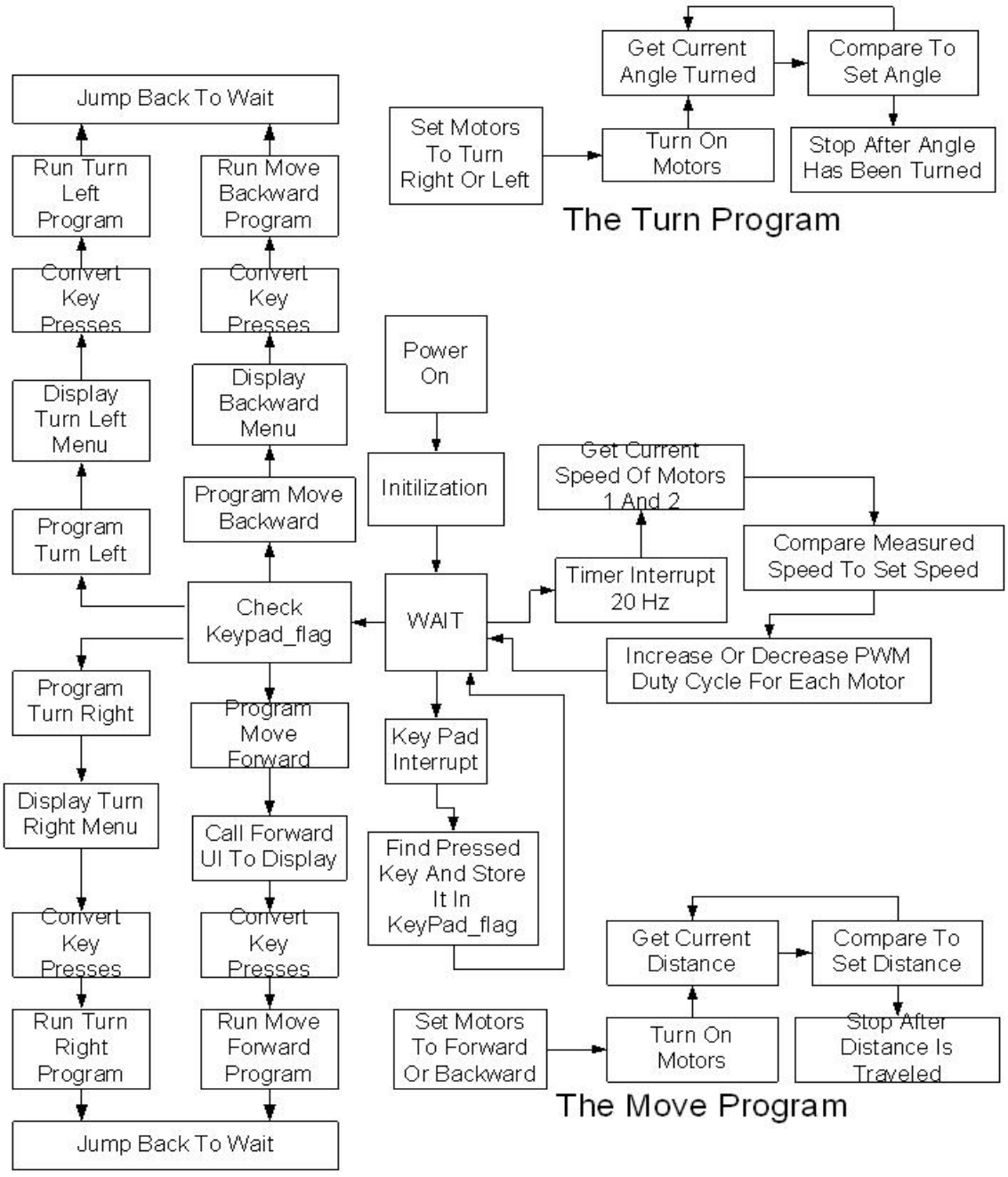

Figure 2. The Software Flowchart for the microprocessor.

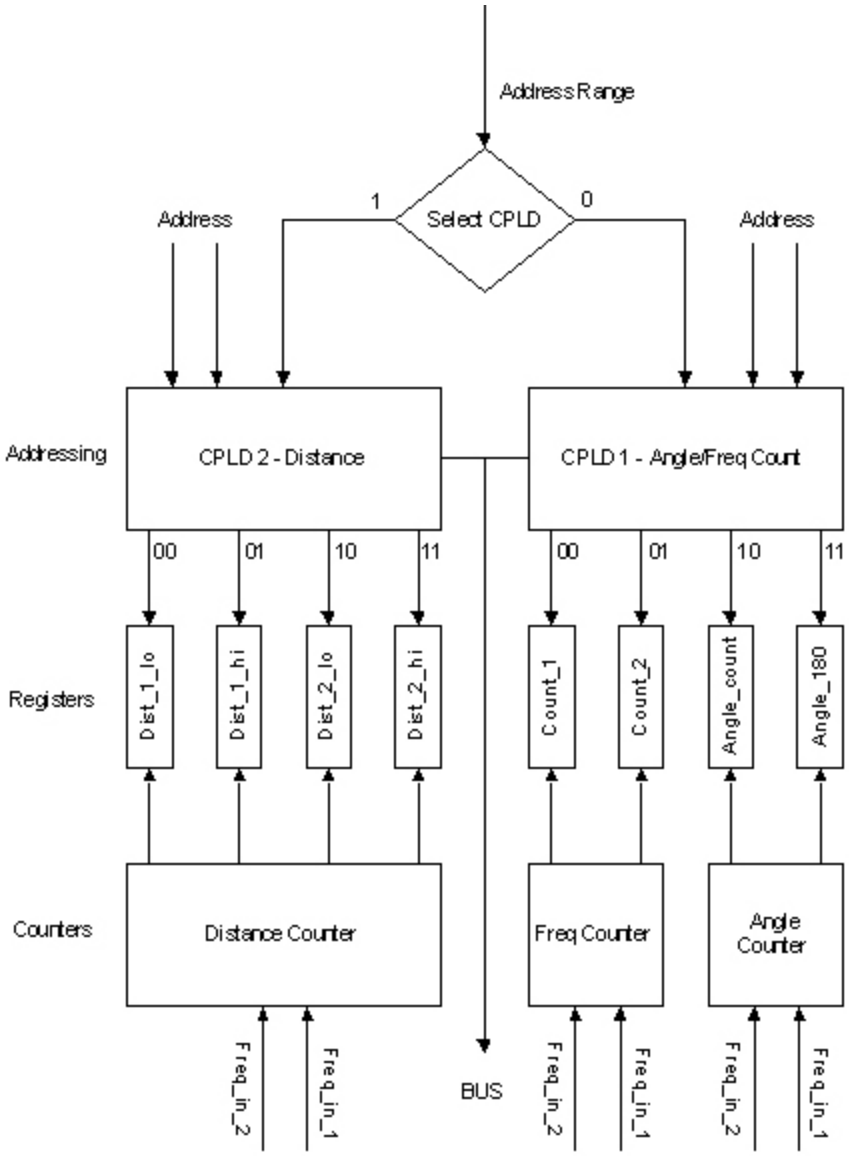

Figure 3. The logic flowchart for the CPLDs.

V. Subsystems

Microprocessor

The assembly code used in the Traxx project was written by Rob Shockency. The code is modular, so it is easy to use and debug. It also consists of several modules.

Initialization

The startup program runs several initialization routines for the 8051 based EMAC controller. This is shown in Fig. 2. First the LCD and keypad are initialized. Several other important functions are enabled such as the timers. "Timer 0" is set up to interrupt ever 1/20 of a second or at the rate of 20 Hz. The "timer 0" interrupt provides the timing for the control system and LCD refresh rate. The control system is used to control the motors by checking the measured speed and comparing it to the set speed.

"Timer 2" is also set in the startup routine. "Timer 2" is used to create the PWM signals, which are used to control the motors. "Timer 2" can be set up to automatically create three PWM signals by some internal hardware, and the outputs appear on pins 1.1, 1.2, and 1.3. This mode of operation is very useful because it requires no continuous program for PWM generation. The PWM used has a 500 Hz frequency and a duty cycle range of 0% to 100%. The PWM was initialized by the following internal variables:

- Registers: CRCH and CRCL
    - PWM period
    - F900 (Hex) sets the period to 500 Hz
- Registers: CCH1 and CCL1
    - PWM 1 complement point
    - FFFF (Hex) = 0% duty cycle
    - F900 (Hex) = 100% duty cycle

- Registers: CCH2 and CCL2
    - PWM 2 complement point
    - FFFF (Hex) = 0% duty cycle
    - F900 (Hex) = 100% duty cycle

Startup also sets all flags to their initial values, along with setting up the memory, and defining the starting point for the stack. Once all the important features are set up, the program jumps to the wait loop located in the "Main" module.

Main

The module "Main" (as shown in Fig. 2 by the wait block) contains the infinite wait loop, where the program waits for the next interrupt to occur. The wait loop is also where the interrupts return after being completed. Main also has the initial coding used to start the user interface. The wait loop continuously checks "keypad_flag." When a key is pressed, the keypad flag is set, and the wait loop responds to the pressed key. If the key pressed corresponds to a programming button, the appropriate function is called. Such function include: Forward programming, backward programming, turn left programming, and turn right programming.

Movement Programming

A programming function is called when the user presses the appropriate programming key, such as forward as displayed in Fig. 2. Each function has a separate LCD screen corresponding to the desired movement and required input:

- When the "Program Forward" button is pressed, the display changes to: **"Move forward? (CM.)"**
- When the "Program Backward" button is pressed, the display changes to: **"Move backward? (CM.)"**
- When the "Program Right" button is pressed, the display changes to: **"Turn Right? (Degrees) Enter the # of 180's"** then it asks the user: **"Turn Right? (Degrees) Enter the # of 1's"**
- When the "Program Left" button is pressed, the display changes to: **"Turn Left? (Degrees) Enter the # of 180's"** then it asks the user: **"Turn Left? (Degrees) Enter the # of 1's"**

After the LCD information is presented, the program waits for user inputs. Each key press is counted, and the result is pushed onto the stack. After the data is entered, the user presses “E” to finish the process. The program then pops all the data off the stack, calls a conversion function to convert the key presses into an actual value stored in memory. Then, the appropriate function is called to actually move the vehicle. The user can also press “F” to exit the programming screen.

Movement Program

Each type of movement is performed by a specific function. However, forward and backward movement is contained in one module, while turning left and right are contained in another. The only difference between the “Move Forward” and “Move Backward” functions are the setting associated with the direction as seen in Fig. 2. Direction is controlled by setting or clearing P4.1 and P4.2, motor 1 and motor 2 respectively. The same is true for turning. To turn right, the left motor must move forward, while the right motor spins backward. To turn left, the opposite must occur. The move forward and backward programs use two eight bit numbers to represent distance traveled. The variables below are used in the program:

- Go_Move_Low: Low order distance (Range: 0 - 99 centimeters).
- Go_Move_High: high order distance (Range: 0 - 255 meters).
- Turn_180: The number of 180’s to turn (Range for the number of 180’s possible: 0 - 255 = 0 to 45900 degrees).
- Turn_1: The number of 1 degree increments to turn. (Range for the number of 1’s possible: 1 - 179)
    - To turn 190 degrees, Turn_180 = 1 and Turn_1 = 10.
        - 180 + 10 = 190

Control System

The speed of each motor is controlled by an automatic PWM adjustment using closed loop control as shown in Fig. 4. The CPLD provides the measured speed of each motor at a refresh rate of 10 Hz. The speed is calculated by taking a frequency count of the encoder of each motor. The CPLD converts the encoder frequency to an 8 bit representation as a value between 0 and 255: 0 corresponds to 0 Hz from the encoder while 255 corresponds to 42 KHz. The program compares the current speed measurement to the desired speed. If the motor is running too slow, the

PWM duty cycle is increased. If the motor is running too fast, the PWM duty cycle is decreased.

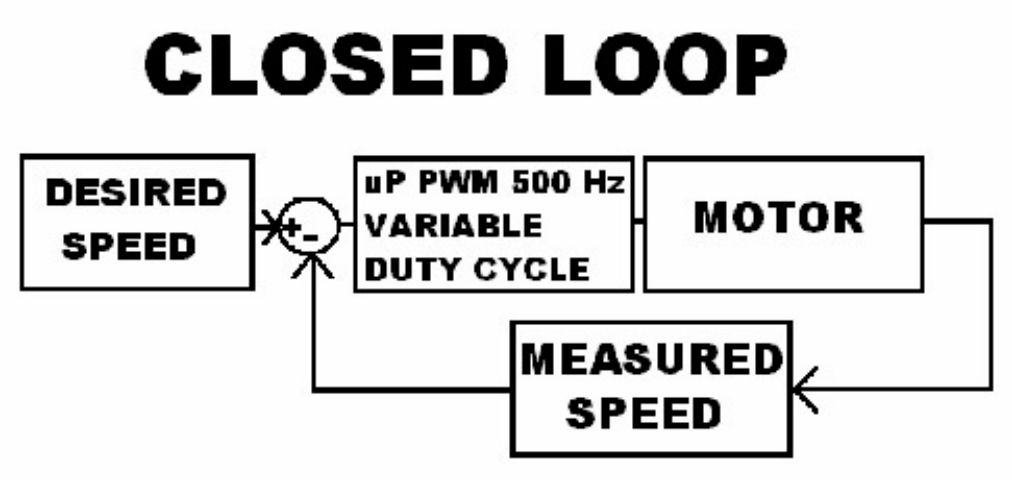

Figure 4. A block diagram of the closed loop system.

The control program is called every "timer 0 interrupt", which occurs ever 1/20 of a second. Since the control system adjusts the PWM 20 times per second, and the PWM is divided into 1791 parts, it takes 89.55 seconds for the PWM to go from 0% to 100%. To make the response time faster, the program checks for "Spans" between the measured speed and desired speed. The span is the difference between the measured value and desired value. The amount of the PWM increment/decrement depends on the span. The relationship is as follows:

- If (span > 30)
    - PWM is incremented/decremented by 50.
    - Response time = 1.791 seconds to go from 0% to 100% duty cycle.
- If(span > 15)
    - PWM is incremented/decremented by 5.
    - Response time = 5.97 seconds to go from 0% to 100% duty cycle.
- If(span > 2)
    - PWM is incremented/decremented by 1.

- Response time = 89.55 seconds to go from 0% to 100% duty cycle.

By adding the span checking, the PWM response time is greatly increased, yet, it can still precisely adjust the PWM to the desired speed. However, this method for controlling the motors does not provide for 0% steady state error and leads to slight oscillation. The control system is also interrupt driven, so the microprocessor can do multiple tasks, not just speed control of the two motors.

Once the user enters the distance to travel, the program turns on both motors to control speed 50. The vehicle begins to move, and the program continuously monitors the CPLD's calculated distance. The CPLD provides real time feedback, and the microprocessor can check the data 1000s of times per second. When the calculated distance is within 50 cm, the vehicle slows down to control speed 40, and it slows down to control speed 30 when the vehicle is within 15 cm of the desired distance. The turning code works the same, but the vehicle does not vary its speed, it only travels at control speed 50.

CPLD

A CPLD was used since it is the most efficient way to do all of the counting that is required for the distance and angle calculations. It is also a much more accurate way of determining the current speed of the motors compared to frequency to voltage converters. FPGAs were not used. First, CPLDs hold their program even after power is removed while FPGAs require a boot ROM. All of this logic is implemented using VHDL and Xilinx Foundation ISE software.

The CPLDs have 3 functions: frequency counting, angle counting, and distance counting, which they perform constantly so the microprocessor can access the data from the registers. Each function is listed defined and explained below along with the address that is used to access them.

Frequency count

The frequency counting code is use to keep the motors in synchronization. This is implemented on CPLD 1 and is shown if Fig. 3. This was a replacement for the original frequency to voltage converters and A/D conversion. This was quite inaccurate and also a poor way to do it because of the multiple conversions. Since it was know that the micro processor could not compute this due to lack of resources, the CPLD was used. The code translates into a set of counters, one 4bit and one 8bit. These correspond to the two counts that are done.

A 42 Khz signal is the input and the output is an 8bit number that is proportional to the frequency. This is outputted based off of a 10hz signal. Doing the math, 42,000 divided by 256 is approximately 160Hz. That is a count of 16 per 0.1s so the first count is from 0 to 15 corresponding to this calculated number. After a complete count, the next count, an 8 bit number, is incremented. At the end of 0.1s, the output is set to the register and stored until access, and then the count is reset and starts over.

The counter is the 8bit proportional representation of the input frequency. To test this subsystem, a function generator outputted a TTL wave that was read by the CPLD. The output was then displayed by having the microprocessor read the port and output the data in decimal form. As the frequency was adjusted the count changed accordingly.

Distance

The distance code works similar to the frequency counter code. It utilizes three counters and outputs a 16bit number in two registers on the second CPLD. The low byte order register contains the distance in CM, and counts from 0 to 99 before incrementing the high order byte in the second register. This register is in meters and goes from 0 to 255 meters. This gives a maximum of 256 meters of travel that can be recorded per command.

It was determined by the circumference of the wheel that there are 663 counts per cm. This was done by taking 500 counts per motor rotation times the 65.5 gear ratio and dividing that by 49.4cm which is the circumference of the drive wheel. This equation is shown in table 2. That yields 663 counts per cm. Therefore, the first counter does 0 to 662, then increments the cm counter. That counts from 0 to 99, increments the meter counters and outputs the meter and cm count to the registers for the bus to use.

Angle

The angle code is a copy of the distance code with different values. There are three counters that are incremented and two that are outputted. The low byte has the angle in 1 degree increments, the high byte has angles in 180 degree increments.

Since the vehicle has a 39cm wheelbase the circumference of the turning circle is 122.5cm. divide this by 360 gives the number of cm/degree (about 0.34) and multiply by the number of counts per cm gives 225 counts per cm. the same principles as with distance are used, the first counter counts to 224 then increments the degree byte. This equation is listed in table 2. The degree byte counts to 179 then increments the half

turn byte. This is then put on the bus for the microprocessor via the control program on CPLD 1.

Table 2. Shows the equations used in the counters for angle and distance.

$$500\frac{counts}{rotation} \times 65.5rotations \div 49.4cm = 663\frac{counts}{cm}$$ **Distance Equation**

$$122.5cm \div 360° \times 663\frac{counts}{cm} = 225\frac{counts}{\deg ree}$$ **Angle Equation**

Control

The control code is what interfaces the microprocessor to the CPLD's. using three address lines, select and read bar the 8 registers can be selected and put on the bus. Each CPLD has 4 registers, CPLD 1 has the frequency counts and CPLD 2 has the Distance counts. The high order address lines selects the CPLD, the other two addresses select the register. Read bar and select have to both be low and the address valid before the CPLD will be brought out of high impendence and data sent to the bus. This prevents bus conflicts, and is required for the bus to operate at all.

Power Electronics

There are two parts to the power electronics, the power system and the H-bridges.

Power system

The power system consists of two 7.2Ah batteries, and a 5v DC/DC converter. The batteries are mounted on the base where they are connected to a toggle switch. The switch is double pole double throw so when the vehicle is off, the batteries are connected to a harness that can be plugged into the charger and charge the batteries. This allows for the batters to be charged without having to be removed. When the switch is in the on position all of the electronics are powered up through the DC/DC converter with the exception of the H-bridges which receive 12volts. The DC/DC converter is an 85% efficient 12volt unregulated to 5volt regulated power converter and powers the encoder wheels, CPLDs and microprocessor. The only component receiving 12volts are the H-bridges to power the motor. Using 5volts on all of the electronics eliminates the use of the on board power converters and not only conserves battery life but also reduces heat and with it reliability problems.

All of the ground on board are connected to each other through one central point near the H-bridges. This is to limit noise and clean up the signals on all ports. Also the chassis is grounded to try and eliminate high frequency noise and for safety so that any broken or loose wire will go to ground and trip the fuse. There is also a 5A inline fuse between the battery and all of the electronics to prevent the possibility of shorting the batteries.

H-bridges

The H-bridges power the motors and provides direction control. There are two parts to the H-bridge, the power transistors and the logic ports. It was found that transistor arrays are pretty robust, handling 55v, but the logic can not handle noise at all, causing a few casualties during design before the noise problem was fixed. However the H-bridges were also the constant source of headache because they not only were susceptible to noise, but produced it and amplified it from the motors. The H-bridges are also on heat sinks.

There are two input signals going to the h-bridges from the microprocessor, the PWM signal that gets amplified from 5 to 12volts, and the direction which is a TTL logic level 0 or 1. The PWM is created from the micro based on the feedback from the encoders via the CPLDs, the direction is used mainly for turning which is accomplished by turning the motors in opposite directions.

The output of the H-bridges is a differential signal that is 0 to 12volts. Neither of the lines from the motor is set to the system ground, instead they are set to a differential ground 12volts from the other line. This helps to filter out some of the noise from the motors using differential signals.

Platform

The platform is made of 1/8” aluminum plates that were custom cut, milled, and folded. It was built from scratch and fit for the electronics and motors that we were using. The design which is shown in Fig. 6, is symmetrical so that the there would be even weight distribution over the drive wheels and the center of turning would be the center of the vehicle. This made it much easier for the turning to be calculated.

The design is a regular octagon that has a side length of 18cm. This size permits the motors to be butted against each other and the batteries to be centered next to the motors. This also allows for caster holes so that the casters can stick up past the bottom plate. The sides of the bottom plate are folded 90degrees with a lip to bolt the top on to it. This provides ridged ness and a place to mount the motors. Also the DC/DC converter is

mounted between the batteries. Two cable harnesses take all of the necessary connection to the top level.

Electronics are mounted on the top plate on two tiers so that they may all fit on board, H bridges and CPLD's on the lower level and Microprocessor, keypad and display on the top. The two cable harnesses can then be disconnected and the top disconnected from the bottom for service to the motors and batteries if needed.

The top also provides bracing for the outside of the drive wheels so that the amount of stress on the motor shafts is reduced. Custom made motor shaft extensions are added to the hold the wheels and nylon bearing are used opposite the motors to support the extensions. Flexing the aluminum a small amount will allow the top to be removed with relative ease.

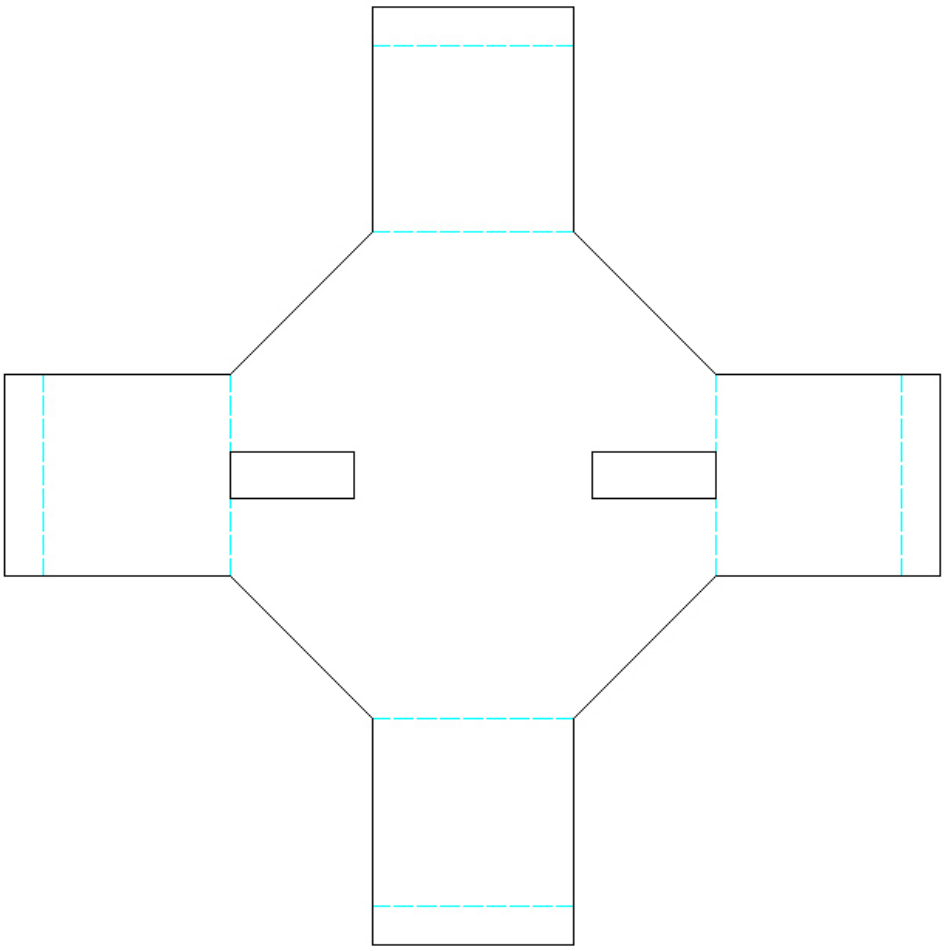

Figure 6. The drawing of the base for the platform.

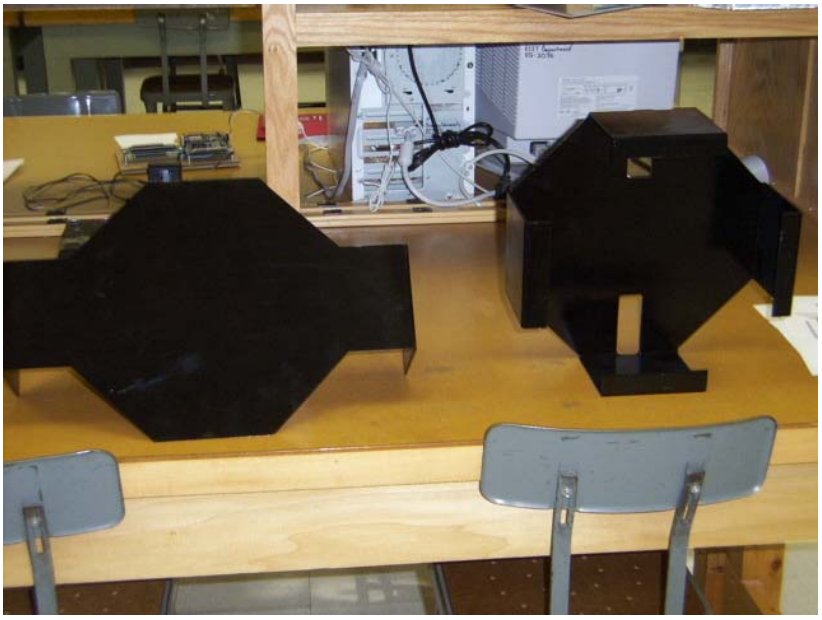

Figure 7. Two pieces of the platform after they have been cut, milled, folded, and painted.

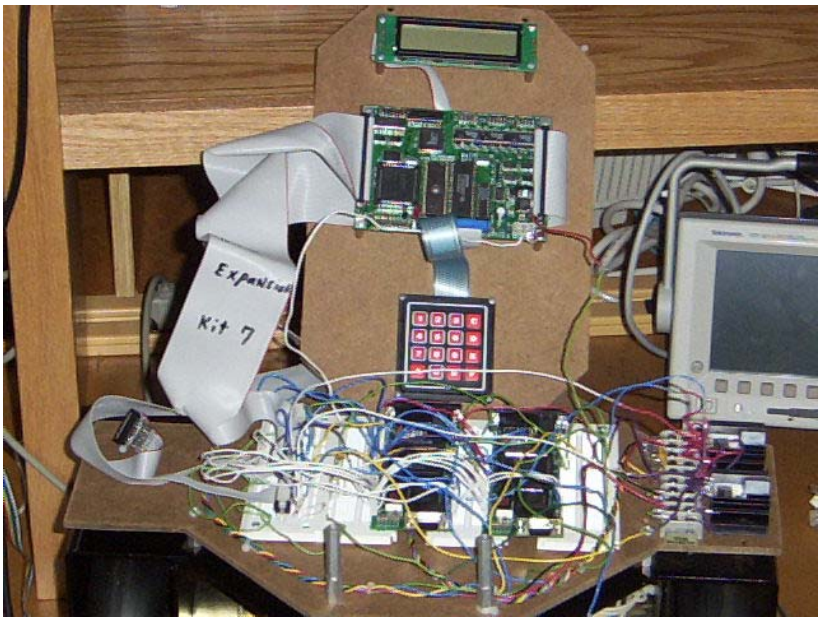

Figure 8. Assembled platform with electronics.

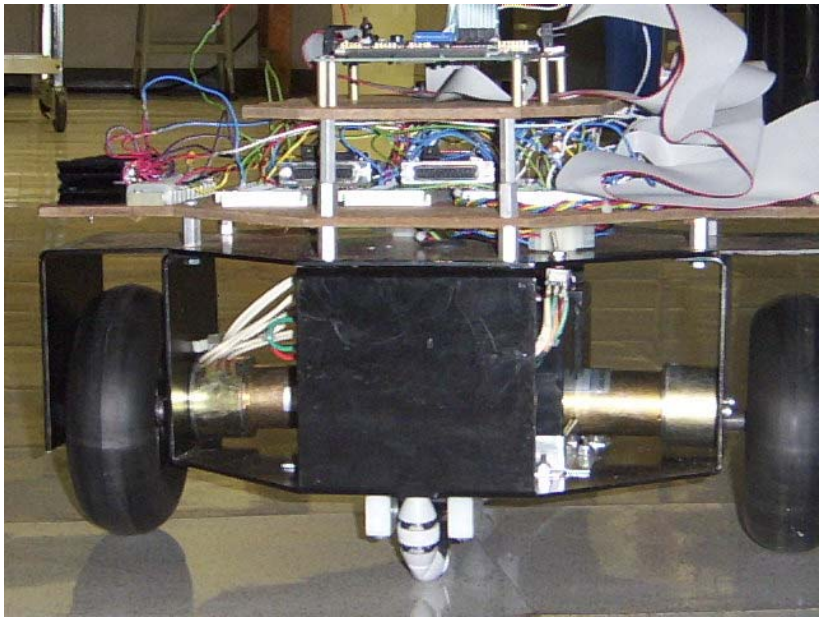

Figure 9. Assembled platform, side view..

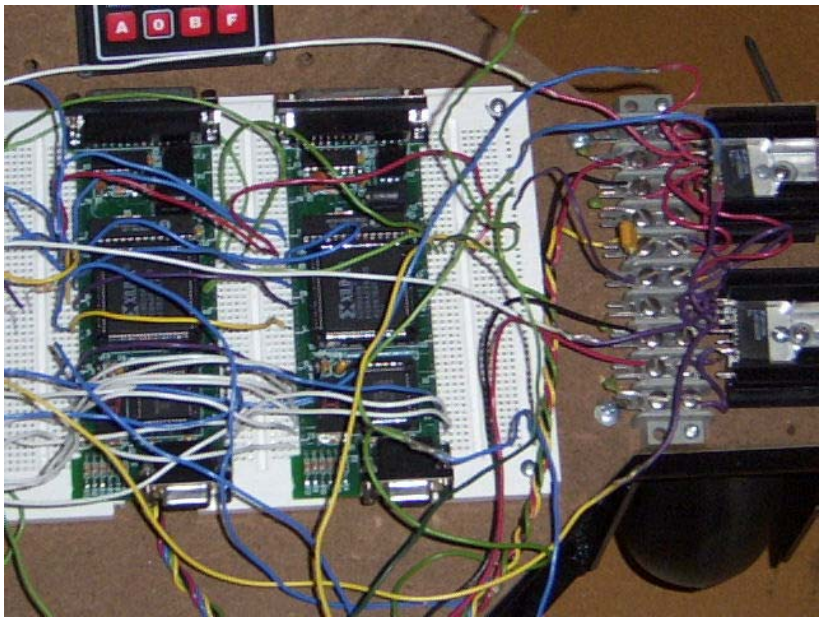

Figure 10. CPLDs on the second tier of platform.

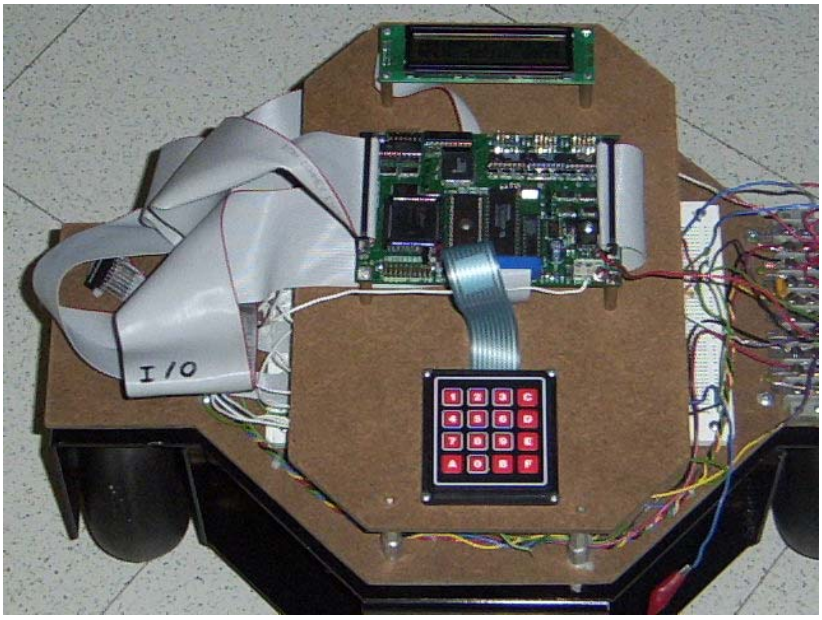

Figure 11. Complete platform with electronics.

VI. Implementation & Evaluation

Testing

The robotic platform is very simple to test. Once the control program was created, the outputs were measured on an oscilloscope to ensure the correct functionality. The automatic PWM adjustment code works because the desired speed and measured speed are both displayed on the LCD. The measured speed oscillates around the desired speed as predicted, and the oscillation is very small. The user interface was also tested in the same manner, but the UI output was seen on the LCD. The UI was tested by simulating a user. However, the UI is not "idiot proof." If invalid data is entered, the vehicle will malfunction. However, an easy fix is possible, but a time limit stopped the progress. Once the user interface was functioning, the vehicle could be given a command. The final test phase required the vehicle to execute a command which was then measured with a ruler. To calibrate the vehicle, the desired command was programmed, and the output was measured with a meter stick, or by measuring the angle turned. The CPLD's could then be adjusted to remove the error in movement. After several iterations, the vehicle was calibrated to correctly travel and turn. The CPLDs are very easy to program, so the final calibration was simple.

Final Tests and Problems

To measure the success of the project, a set of maneuvers was programmed into the robot to maneuver a square. At first, the results were very bad. The vehicle still needed much calibration. Little can be said about the accuracy of the vehicle because it cannot even maneuver a straight line. However, we were informed of a huge mistake in our design. We had twisted all the wires together into one long cable. This was a fatal mistake because the encoder frequency and 12 volt PWM signals were coupled together because of the twisting wire cable. Once the wires were separated, the results were much better, but the vehicle would still travel in an arch. Still, the vehicle did maneuver a square, but the accuracy was undesirable. It did perform all the necessary maneuvers involved in making a square, and it stopped after the program was executed. Much work is still needed on the system, so a final test could not be done. The best advice for the future is to use a laptop and AD/DA card to control the vehicle. Assembly language is a low level programming language, and it

is very difficult to use affectively in this project. A laptop and C code would make the vehicle work flawlessly.

VII. Conclusion

With the ambitious goal of creating a complete robotic platform from scratch to compete with one that was professionally designed and cost over 3 times as much, we have had great success. With a little under 8 months of development and building our prototype is nothing less then remarkable. It operates well, and has vast amounts of expandability for future projects.

This platform was designed to be a lower cost solution than the Pioneer 2. However, unlike the pioneer 2, the expandability on this is much greater. Not only can cameras, and wireless web control be added, but different types of control systems can be added. This could be used in lab for experimentation with different types of control systems running the vehicle to see the results. It could also be used in a VHDL or VLSI setting to see the advantages of these technologies over conventional microprocessors. Even the microprocessor classes could use a faster/larger microprocessor development board and add much more functionality, friendlier user interface, or the ability to store multiple commands and execute them. No to mention RF labs could use this vehicle for work with wireless transition of data to and from the vehicle. The fields that this vehicle can be used for are quite abundant.

Like the vehicle itself, the future for it can utilize any field of study. Even before its completion upgrades and expansions for it were developed. A VHDL version of the entire control system was created, utilizing RAM for command storage and even better real time feedback by eliminating microprocessor delay. This vehicle not only benefited us, but will continue to benefit others in the years to come.

This project is not simply something that will just disappear after creation. It will also be used in future applications. This project is not a completely new, revolutionary idea. It is however, a new and improved version of an old concept that will see much use in years to come. This vehicle is using some of the latest and greatest technologies such as VHDL (Very high speed integrated circuit Hardware Descriptor Language), and high density CPLD (Complex Programmable Logic Device). It also takes some of the most commonly used devices such as microprocessors and power

transistors; combining them with the new technology to create a control system that allows for the high precision desired. When you start from scratch on an old idea, new ways of implementing that idea come to light, such as the combination of the VHDL code and Assembly code for a complete real-time control system.

This platform will also serve as a starting point for other projects to come. Another group can take this platform and add a camera and some sensors, allowing it to follow people, or they could add an arm to it for retrieving items desired by the user. The possibilities are only limited by human ingenuity.

References

Hayt, William H. and Jack E. Kemmerly. (1993). Engineering Circuit Analysis 5th Edition. New York, NY: McGraw Hill.

Sedra, Adel, S. and Kenneth C. Smith. (1998). Microelectronic Circuits 4th Edition. New York, NY: Oxford University Press.

Zionix LLC. (2002) www.zionix.com.

Koninklijke Philips N. V. (2002) www.semiconductors.philips.com

National Semiconductors. (2002) www.national.com

Pittman. (2002) www.pittmannet.com

Xilinx Inc. (2002) www.xilinx.com

Xess Corporation. (2002) www.xess.com

Xilinx University. (2002) xup.msu.edu

IEEE Standards online (2002). standards.ieee.org/reading/ieee/std/lanman/802.11b-1999.pdf

IEEE Standards Online (2002). standards.ieee.org/getieee802/

PATENTS:

US199700079438

This is a patent on remote control vehicle that uses PWM to control the motors. It also uses a transistor that automatically changes the PWM signal to allow for both forward and reverse direction of the motor. The only difference is the PWM signal is transmitted as an analog-frequency signal from the hand held transmitter. We transmit the control signal, and the micro controller creates the PWM. The whole transistor-switching network is the same as what we use because it turns the motor on and off at a rate of the PWM frequency.