DEVELOPMENT OF AN IPAQ-BASED ROBOTIC VEHICLE WITH WIRELESS CONTROL AND VIDEO FEEDBACK

Wayne T. Padgett Adam J. Thomas
Electrical and Computer Engineering Dept.
Rose-Hulman Institute of Technology
Terre Haute, IN 47803

ABSTRACT

The iPAQ™ computer makes an excellent hardware platform for the development of a robotic vehicle with advanced video and wireless network features. Because the hardware and accessories are off-the-shelf mass market products, a reliable vehicle can be created with very little hardware development, and the cost is minimized. This paper describes the development of the vehicle hardware and software by four undergraduate students at Rose-Hulman as part of an independent-study project course. Potential uses as part of a robotics / image understanding course are discussed.

INTRODUCTION

This project began as a prototype design for a ground vehicle for the International Aerial Robotics Competition [1]. The current rules of the competition require a small vehicle that can move and return video to a controlling computer several kilometers away. Because reliability is difficult to achieve in custom built hardware and because custom software tools are usually rudimentary, this experiment was designed to test the idea of using commercial hardware. Since the iPAQ is capable of supporting a wireless network card and a video-capable digital camera [2] it became the target CPU for the robotic vehicle. This project was only a proof-of-concept so a simple toy firetruck was chosen as the vehicle to be controlled with the iPAQ system. The firetruck has a number of accessories to control just as the competition vehicle would have. A picture of the firetruck is shown in Figure 1.

Although there remain a few software issues to resolve, the resulting prototype allows the user to control the vehicle from a remote computer over the wireless network, and watch video of the scene ahead of the vehicle.

TEAM

Four Rose-Hulman undergraduate students worked on this project. Ryan Bechtloff, a senior computer science major, and Adam Thomas, a freshman computer science student, were both in charge of figuring out the NexiCam™ API, developing the server/client software for the system, and creating code that displays pictures from the NexiCam. Thomas Perme, a computer science freshman, and Chris Dupin, an electrical engineering senior, headed up the efforts to control the fire truck with a PIC microcontroller and to get the iPAQ to talk to the new control circuit via RS-232 connection. In total, the team worked about 420 hours over the course of a ten week term.

HARDWARE

The iPAQ H3970, which requires a wireless network card accessory, was obtained through a grant from Hewlett-Packard to encourage educational use of the handheld computer. Current iPAQ models such as the H4155 include built-in wireless LAN and are in the $500 range. The iPAQ has a “sleeve” interface for supporting various accessories, such as extra batteries, extra card slots, and other types of interfaces. One such accessory is the NexiCam [2] camera sleeve which costs about $120. The NexiCam allows the iPAQ to operate as a

Figure 1. The iPAQ controlled firetruck. The custom cradle on top holds the iPAQ so that the NexiCam camera can view the forward path. The black cable is a serial link to an internal PIC processor that does the motor switching. The iPAQ is not installed because it was used to take the picture.

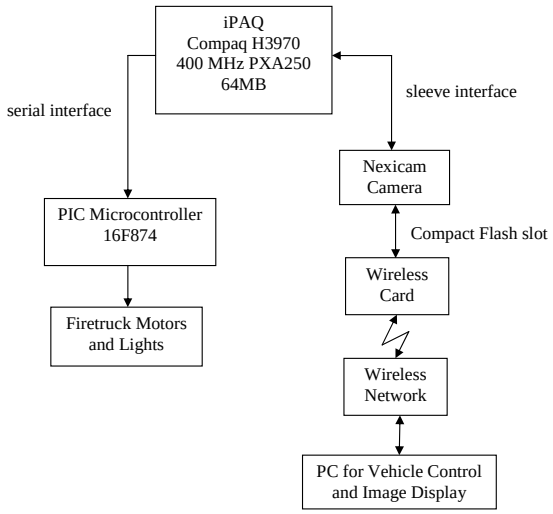

Figure 2. The Hardware. The iPAQ has a NexiCam camera sleeve, which includes a Compact Flash slot for the wireless card. The motor and light control is done through the serial port.

digital camera, and the camera lens is on a rotating arm so that it can be used for videoconferencing as well. The rotating arm allows the iPAQ to be mounted in a variety of positions and still have the camera aimed toward the vehicle's forward path. The NexiCam also includes a Compact Flash slot, primarily for image storage, but in this project it is used for the D-Link Wireless LAN card which cost about $100. Getting digital I/O from the iPAQ is a challenge. In keeping with the "off-the-shelf" approach, the original plan was to use a USB-based I/O box, but the iPAQ has a slave USB interface and so does the I/O box, so there was no bus master. The serial port was used instead, which is slower, but simpler to work with, both in hardware and software.

A PIC 16F874 was used to receive the serial control commands and control the L298 H-bridges and IRF540 MOSFETs that switch the motors and lights on the firetruck. The firetruck can even pump water under iPAQ control. The entire vehicle is controlled from a PC and accessed through the internet, using the Rose-Hulman wireless network. Figure 2 shows a block diagram of the hardware configuration.

The vehicle hardware design was simplified by choosing a firetruck that used a 6V supply, which was easy to regulate down to 5V for the PIC, and the accessories did not require level conversion for operation with the PIC's logic levels. Serial communication with the iPAQ did require level shifting to +/-12V, which was accomplished using a MAX232 chip.

COST

The cost of the entire project was well under $1000. Because the iPAQ was donated by HP, the out of pocket expense for this prototype was under $350. Fortunately, the cost of iPAQ hardware is dropping, and the iPAQ can also be used for other projects, since it is not modified. Table 1 shows a breakdown of project costs.

Clearly, the main costs are the iPAQ and the NexiCam. The costs of the controlling PC and the wireless hub are not included here because they are part of the infrastructure at Rose-Hulman, and many other schools. The cost of manufacturing a small PC board can vary widely. At Rose-Hulman, we have a milling machine for manufacturing circuit board

Part	Approximate Cost
iPAQ (with wireless)	$500
NexiCam	$120
Remote Control Vehicle (including battery and charger)	$45
PIC 16F874	$10
iPAQ Serial Cable	$25
PCB fabrication (RHIT prototype milling process)	$30
Circuit Components	$10
Totals	$740

Table 1. Itemized Expenses. The hardware cost is quite low. The costs shown above assume the purchase of a newer iPAQ which would include a built-in wireless card, and minimal PCB fabrication costs for the PIC board.

prototypes at minimal cost, however, numerous services such as ExpressPCB [3] can quickly manufacture small boards for as little as 3 boards for $60. It is also quite practical to build a PIC system on a prototyping board (hand-wired) or even on a solderless breadboard. Note that the solderless breadboards are not very reliable, but worked well enough that after two attempts at building a circuit board were thwarted by hole sizing errors, the student team left their PIC implementation on a breadboard and moved on.

SOFTWARE

The purpose of the software was to allow a person to control the vehicle from a remote computer via the iPAQ and be able to drive the vehicle around using real-time streaming images sent over the wireless network from the iPAQ. This requires two programs, a server running on the PC side as a user interface, and a client, running on the iPAQ side to control the vehicle and the NexiCam. All of these features have been successfully implemented, however the streaming images were not reliable because they caused the server program to lock up periodically thus preventing control of the vehicle in real-time. The vehicle could however be controlled by placing a delay between images and running the vehicle for only a few seconds at a time. Finally a compromise was reached where two separate client/server programs were set up, one to control the vehicle and one to take pictures, as shown in Figure 3. The PictureServer/Client software was designed so that if the server (PC side) locked up, it could be shut down, restarted, and then the client (iPAQ side) would automatically timeout its previous connection and attempt to establish a new one.

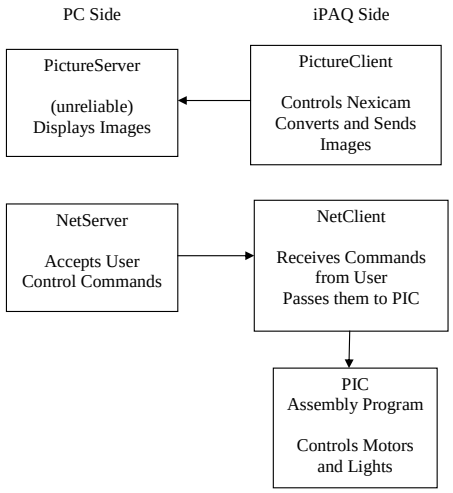

Figure 3. The Software. The client and server programs were divided into two parts so that problems with image display would not sever control of the vehicle, and so that the PictureServer could be reset. Note there are no two way communication paths, and vehicle sensors would require a NetClient return path.

USING THE IPAQ TO TAKE PICTURES

The first step in implementation was to learn how to program for the Windows CE based iPAQ. Embedded Visual C++ was used as the compiler and, after a lengthy trial and error compiling configuration process, the standard "Hello World!" program successfully ran on the iPAQ. The next step was to attempt to take a picture. Navicom (Nexian is Navicom's U.S. distributor) graciously provided the NexiCam API and SDK manual, but it was not up to date so the camera interface took extra work. Eventually the included writejpeg program was

modified to take a picture by carefully picking through the camera.h file for commands that seemed likely to work. Since the task of taking pictures was completed early in the quarter, it was decided that the project should be expanded so it would take the full ten weeks. The student team was divided into two major focus areas: the first consisted of Adam and Ryan who worked on making connections over the wireless network and eventually sending pictures and commands; the second consisted of Chris and Tom who worked on using a serial cable connected to the iPAQ to provide input to a PIC microcontroller which controlled the vehicle.

CREATING A WIRELESS CONNECTION

Creating a wireless connection to send data between the iPAQ and a host computer was the second step after learning how to write basic programs for the iPAQ. After reading several articles on the web, the decision was made to use WinSockets to establish network connections and send data, because they were the most easily ported to the iPAQ. First, MyNet, a simple server/client program was written, that could be used between two computers running Windows. MyNet allowed two computers that were connected to the internet to send text messages between them. Next, MyNet was modified and converted over to the iPAQ which runs Windows CE. This proved to be more difficult that originally anticipated. First of all Windows CE does not support all of the functions that Windows does and furthermore the iPAQ itself only supported certain Winsock 2.0 functions so an older version had to be used, Winsock 1.1. Even with Winsock 1.1, a few functions would not on work the iPAQ such as GetHostByName(). This prevented setting the iPAQ up as a server that any computer could connect to because the port number and IP address had to be entered in before a connection is initiated. So the iPAQ software had to be implemented as a client that would connect to a server.

The second stage was to take the pictures on the iPAQ and send them over the network to the server. This was harder than sending text because first the send and recv functions Winsock used only accepted the char data type so everything had to be type cast char before sending and then type cast back to BYTE after it was received so it could be written to a file. There was a problem with the recv function only receiving a limited amount of data before it returned so only half or a quarter of a picture would arrive at the display PC. This problem was resolved by sending the size of the picture first and then looping the recv function until it had received the full file. There was also a stack overflow problem that required reading the received picture into a buffer on the heap instead of the stack. The code was later modified to pass this buffer directly to the LoadPictureBuffer function instead of writing it to a file.

The final stage of software implementation was to control the PIC microcontroller through the serial port from the (iPAQ) client which was taking commands from the (PC) server program. Chris and Tom developed a program to send commands from the iPAQ over the serial port. A series of modifications of existing code resulted in a system to send text from the PC control server to the iPAQ which then converted them to PIC commands and relayed to them to the PIC.

SERIAL COMMUNICATIONS AND PIC MICROCONTROLLERS

Serial communication begins with the iPAQ sending an ASCII character out its serial port. A cable runs from the iPAQ to a Max RS232 chip, which changes the RS-232 signal to a TTL/CMOS level, and then to the USART input on a PIC16F874. Once the built-in USART in the PIC has received the 8 bit ASCII character and stored it, an interrupt vector is executed that processes each serial input and changes the output state of the corresponding ports on the PIC. PORT B has four pins that control the fire

truck's driving motions. PORT A has four pins that control the tower's motion, and PORT D has two pins to control the water pump and the headlights. The four pins for the driving motors and the four pins for the tower's motors each have an L298N H-bridge to control the direction of the motion and allow for a higher current than the PIC can output.

DISPLAYING PICTURES

Displaying a picture turned out to be relatively easy, since a jpeg image display program was available from Microsoft's website. The Microsoft example code had to be modified slightly, but it displayed the images as needed. The Microsoft example required the incoming image to be read from a file, so the images coming from the iPAQ had to be written to a file, and then read back into memory to be displayed on the screen. Displaying multiple images in rapid succession was more difficult. The display functions had no error checking so error reporting had to be added to find out what the problems were. In order to increase the speed of the display, the example code was modified to read straight from a memory buffer instead of writing to a file first. The image server often locked up entirely if it tried to paint the screen and load a new image at the same time, so a simple mutual exclusion lock was added to prevent that from occurring. The cause of an E_FAIL error which is specifically an unspecified error associated with the OLELoadPicture() function, remains a mystery. It was possible to identify that error and make the program continue each time it occurs, but not prevent the error from occurring. The picture server program locked up occasionally due to an error which was not identified during the project. The problem was later determined to be caused by incomplete jpeg files being sent to the OLELoadPicture() function, which caused it to lock up instead of throw an exception. The problem does not occur when a Sleep(2000), which stalls the program for 2 seconds, is added to the loop, so there is may be a problem in our early version of the NexiCam. The most attractive compromise solution was to design the client program so that the user could shut the picture server program down and restart it if it locked up and the picture client would automatically reconnect.

POSSIBLE IMPROVEMENTS

NEXICAM

The NexiCam had some disadvantages for this project. First, the API was in an early stage of development. Hopefully, a more mature version will be available for future work. Second, the NexiCam API is designed around a still-camera concept, which is not ideal for streaming video. Finally, the direct attachment of the NexiCam to the iPAQ requires the iPAQ to be mounted at the front of the vehicle where it is most easily damaged. A separate video camera that could be mounted away from the iPAQ, or even set up with multiple cameras (for switching views) would be nicer, but no such products seem to be available.

PROGRAMMING LANGUAGE

Visual C++ was the obvious choice for this project because that's what the NexiCam API supported for the iPAQ clients. Since it seemed impractical to write a translation of Winsockets for another language, Visual C++ was also the obvious choice for the PC servers. Further research on Winsockets seems to show that the server could have been programmed in a different language such as Java or Visual Basic which is more graphically oriented and thus, may have prevented the mysterious problem that causes the server program to lock up.

PROGRAM DESIGN

During the project it became apparent that it was necessary to use threads with the send and recv functions because they needed to loop simultaneously while waiting to send data from the user or camera and receive it on the other

end. The students had very limited experience with threaded programming, so the problem causing the server to lock up could also be a result of poor multithreaded design.

The function used to load the picture OleLoadPicture() was not the fastest option available. Many articles on the web suggested using Intel's Integrated Performance Primitives library [4] which cuts the time in half of loading pictures in a Windows program. This information became apparent during the course of the project, but it was too late to convert everything over. Pricing for this library seems quite affordable, especially with academic discounts.

PCB

During the last two weeks of the 10 week project, the students tried to make a PCB for the PIC circuit. However, the first PCB had some holes that were too small and it had to be scrapped. The second PCB had a short which the students were unable to find, rendering it useless as well. Since the solderless breadboard prototype still worked, it became the "final" implementation. It would have been desirable to have more time to complete a functioning PCB.

SENSORS

It would add significant new functionality if IR sensors and a heat sensor for the fire truck could have been added. The IR sensors would be for wall-following, so the truck could drive into a room, and then automatically follow a wall until it found a target on the camera or from the heat sensor. The heat sensor would allow simulated
firefighting, and possibly video feedback on the water stream aiming point. Another possible sensor addition would be an optical shaft encoder for the wheels so that accurate measurements of travel distance could be reported to the controller.

COMPETITIVE DEVICE

A Google search for robot vision will turn up another "off the shelf" solution for experimentation: Evolution Robotics' ER-1 [5]. This is a simple robot system designed to carry a laptop as its CPU. It includes a frame, stepper motors, a web cam, and a USB stepper motor controller. Clearly, the concept is similar to the iPAQ based robot described above. The main differences are in size and weight, since the laptop is a larger device than the iPAQ. The ER-1 does have the advantage of using a web cam as mentioned above, and having the same operating system at the controller and on the vehicle.

APPLICATIONS

Although this project was conceived as a proof of concept prototype for the Aerial Robotics Competition, it has become clear that the resulting system is an affordable platform for autonomous vehicle experimentation and computer vision research. Because both the control and images are available to the stationary PC, it is simple to use the images for path planning and obstacle avoidance algorithms, and test the methods on real data. The vehicle can be driven as slowly as desired, and as much computational power as you wish can be applied at the stationary PC end of the system.

Because the moving vehicle takes multiple images of roughly the same scene from different positions, algorithms similar to stereo vision can be applied. The image analysis can take advantage of substantial information about the vehicle's motion, since it controls the motion. A great deal of research has been done on the "structure from motion" problem [6], and this is an excellent topic for an upper level course. The ability to test student solutions in a real experiment should be highly motivational for the typical student, just as it was for the students who worked on this project.

CONCLUSIONS

It is both possible and practical to build an affordable, small, reliable robot for research and experimentation. The majority of the components can be obtained from commercial "off the shelf" sources, and the required software is within the reach of undergraduate students. The exercise of building the software and hardware for such a robot is an excellent learning experience, and the resulting robot makes an excellent platform for further research and experimentation. With a remote computer in control of the robot, it is simple to experiment with autonomous vehicle behavior in a real environment for testing path planning, image understanding, obstacle avoidance, and structure from motion algorithms. Many schools could benefit from the opportunity to have hands on lab work in these advanced topics, with a very small budget.

REFERENCES

1. Michelson, Robert. (2004, Feb. 19) Association for Unmanned Vehicle Systems, International's Aerial Robotics Competition. [Online]. Available: http://avdil.gtri.gatech.Edu/AUVS/IARCLaunchPoint.html

2. Nexian, Inc. (2004, Feb. 19) NexiCam Digital Camera for the HP IPAQ Pocket PC. [Online]. Available:http://www.nexian.com/product/nexicam.asp

3. Engineering Express. (2004, Feb. 19) Introduction to ExpressPCB. [Online]. Available: http://www.expresspcb.com/

4. Intel Corp. (2004, Feb. 19) Intel Integrated Performance Primitives. [Online]. Available: http://www.intel.com/software/products/ipp

5. Evolution Robotics, Inc. (2004, Feb. 19) ER1 Personal Robot System. [Online]. Available: http://www.evolution.com/er1/

6. J. Oliensis, (2004, Feb. 19) "A Critique of Structure from Motion Algorithms," NECI TR 1997. [Online]. Available: http://citeseer.nj.nec.com/oliensis00critique.html

BIOGRAPHICAL INFORMATION

Wayne T. Padgett was born in Syracuse, NY. He received the B.E.E. Degree from Auburn University, Auburn, AL, in 1989. He received the M.S. and Ph.D. degree from Georgia Institute of Technology, Atlanta, GA, in 1990 and 1994, respectively. He is currently an Associate Professor of Electrical and Computer Engineering at Rose-Hulman Institute of Technology. He has consulted with a variety of companies on digital signal and image processing. Dr. Padgett is a member of IEEEE and ASEE. He received the CoED Division's Woody Everett Award for best paper in 1999. His research interests include robotics, fixed point algorithm design, and microphone arrays.

Adam J. Thomas was born in Terre Haute, IN. He attended Northview High School in Brazil, IN. Currently he is a junior attending Rose-Hulman Institute of Technology and majoring in Computer Science with a minor in Applied Biology. His course concentrations include artificial intelligence and genetics. He plans to pursue a future career in biotechnology.