CPSC 483: Senior Design

Implementation Notes

Wireless Robot Control with a Personal Digital Assistant

Matt Dean

Ta-Chang Chao

Brady Wied

Quang Dang

May 5, 2003

Implementation Notes

During working on our projects we have get these experienced notes below that would be very helpful for engineers to understand, modify and reproduce our product easily and effectively.

Aria Port

Problems:

1.In ArkeyHandler.cpp, the –getch() function is used but not available in windows CE, so replaced with getchar().

2.In ArRobot.cpp, ArRobot::addActuib used std::pair<int, ArAction*>, but it should be changed to <int const, ArAction*> to complile.

3. In ArRobot.cpp, dynamic-cast is used, but RTTI is unsupported in Visual C++ 3.0.

Compilation Problems:

1. Complains about missing header files even though they are inside ifdef linux directives
2. Have to include “stdafc.h” at top of all .cpp files for pocket console.
3. ariaTypedefs.h includes winsockz.h, which has not been ported to Windows CE. Will need to change to Winsock.
4. ariaUtil.h uses the function timeGetTime(), which is apparency unsupported in CE, also in ariaUtil.cpp

5. The ArJoyHandler uncludes members of type JOYNIFO and JOYCAPS, which are included in normal Windows but not CE.

   Provided an implementation of these STRUCTS in ariaTypedefs.h
6. peror is used throughout the program but it is not directly supported.

   Created an ErrorPrinter class with a static method perror.
7. Problems with registry management in ariaUtil.cpp
   a. HKEY-CURRENT-CONFIG is not supported in Windows CE so romoved case.
   b. Prolems using CHAR* as param, so converting to wide chars.
8. In ArkeyHandler.cpp ->operator is undefined for map iterations, so replacing it ->second with (*it).second. Also-getch() is undefined, so using getchar();.
9. In Armodes.cpp, dynamic-cast is used but unsupported, so save type information in object and use static casts.
10. In Armoduleloader, have to change STATUS-SUCCES to STATUS-SUCCESSFUL and fix Unicode error.
11. In ARNetServer.cpp, replace -> with (*).
12. ArSerialConnection_WIN.cpp, fix Unicode and add perror support.
13. ArSocket_WIN.cpp, add perror support and defined SD_BOTH.
14. ArSyncTask.cpp, change std::pair<int, ArSyncTask *> to std:: pair<int const, ArSyncTask *>.
15. ArThread_WIN.cpp, add hasReturnValue to ArFactor.h.
16. ArSick.cpp use dynamic-cast to see if a ArDeviceConnection is a serial connection, so add isSerialConnection to ArDeviceConnection.

17. ArRobot.cpp use dynamic-cast to determine connection type, so add isLogFileConnection to ArDriveConnection.

PDA User Interface

GapiDraw, a library and API intentionally similar to DirectDraw, is good and efficient software tool to create the PDA user interface.

1. Ease of graphics development
2. Can develop interface on desktop without having to make changes for Pocket PC
3. Efficient (fast) implementation
4. Small size of required DLL
5. Easy full-screen mode

CPSC 483: Senior Design

User Manual

Wireless Robot Control with a Personal Digital Assistant

Matt Dean

Ta-Chang Chao

Brady Wied

Quang Dang

May 5, 2003

Introduction

As we have known, the topic of this project is "Wireless Robot Control with a Personal Digital Assistant". The main goal is to allow a user with a handheld PDA to control a robot and also at the same time provide a framework for developing useful applications for this technology. The interface will allow the user to control the robot vehicle with a number of basic motor primitives, receive information from the sensors on-board the robot, zoom in and out on the map display to adjust its size and lastly do the corrective position function.

The total hardware and system software are consist of the following: AmigoBot, radio modem, Pocket PC with a serial cradle, Pocket PC 2002 OS, eMbedded Visual Tools 3.0, Pocket PC 2002 SDK, GapiDraw SDK 1.04, ARIA and STL for eMbedded Visual C++.

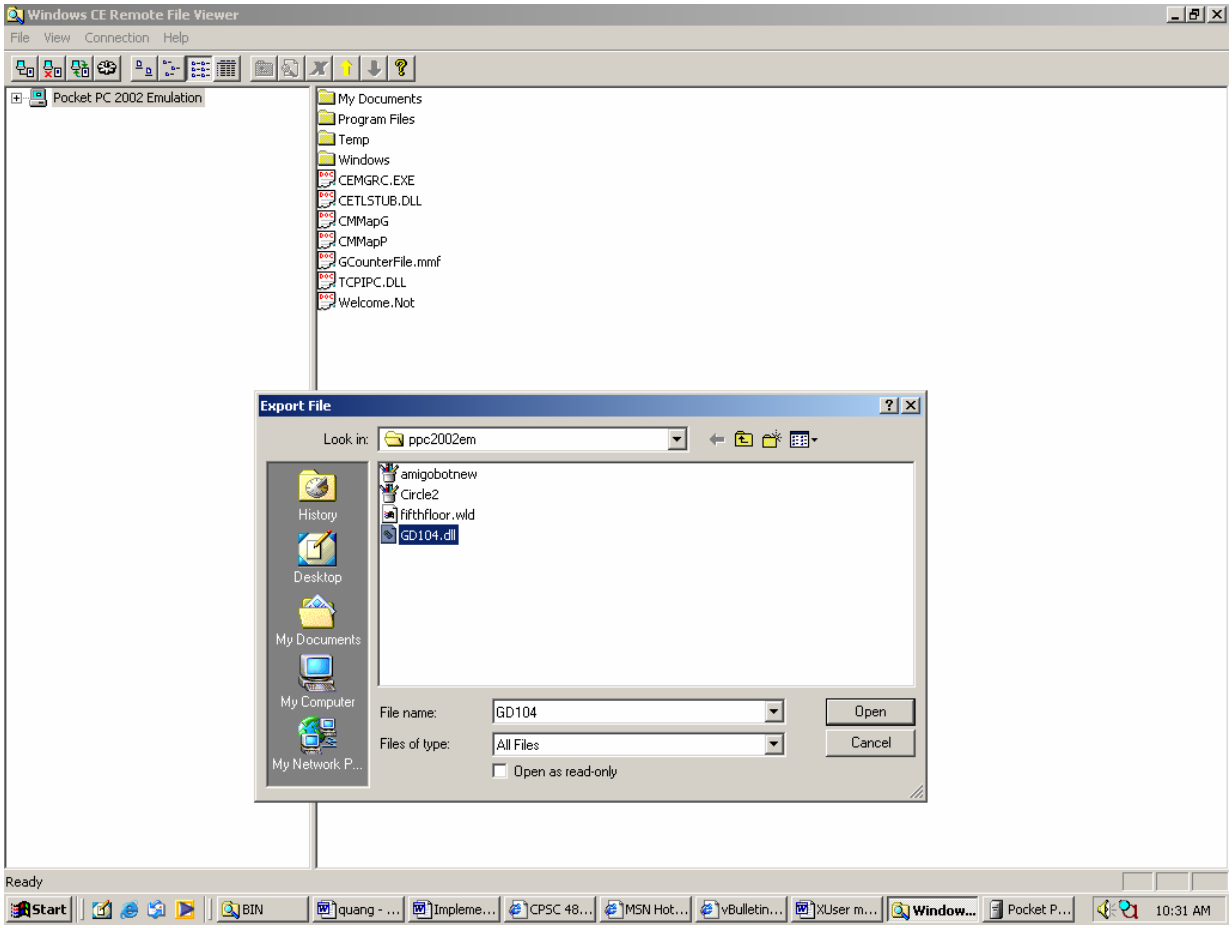
Installation

There are several things need to be installed before the robot application can be run and use to control the robot. The sequences of steps are as follow:

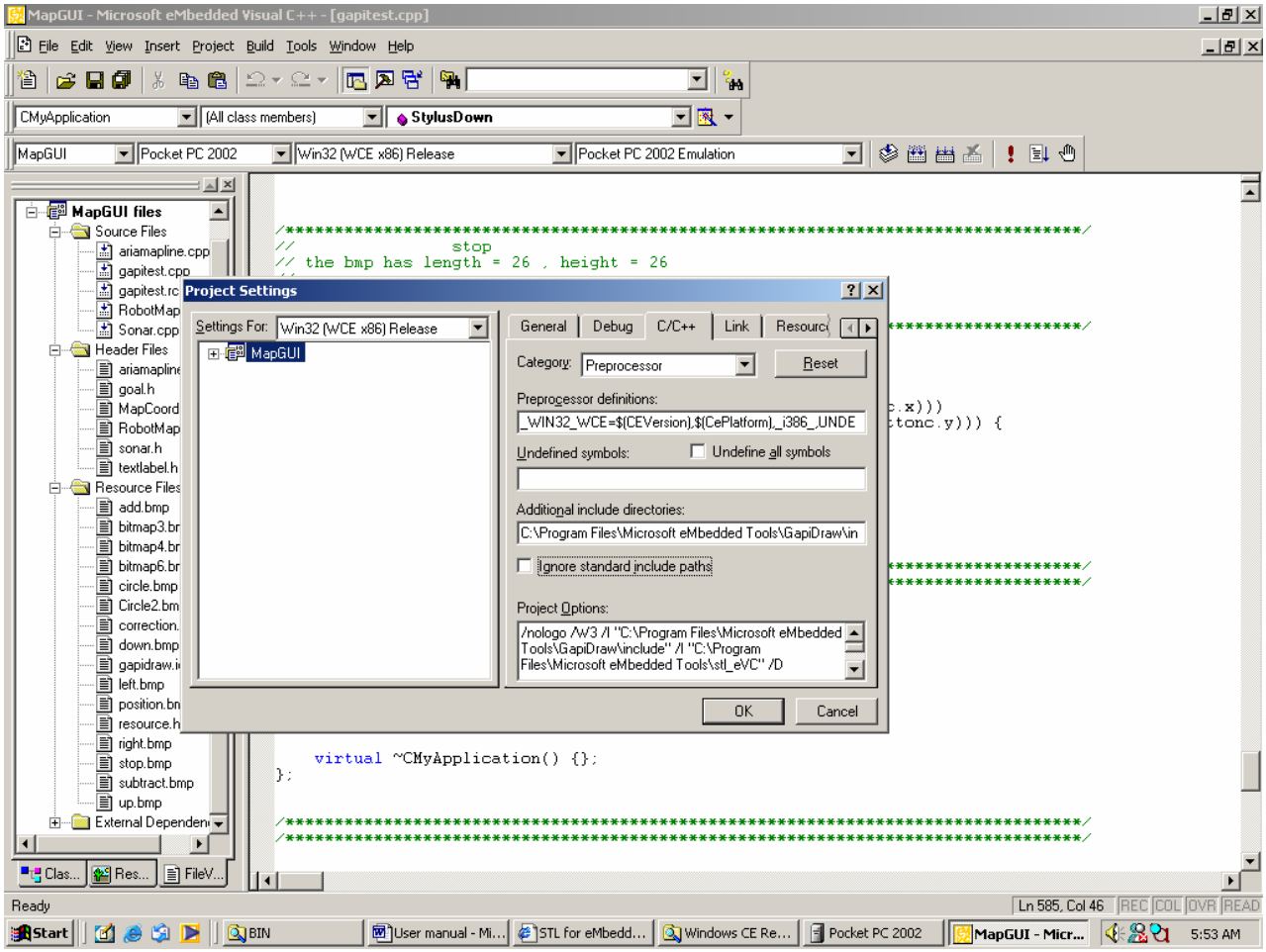
- Go to http://www.microsoft.com/mobile/developer/downloads/default.asp to download and install eMbedded Visual Tools 3.0 and Pocket PC 2002 SDK. (Make sure install the eMbedded Visual Tools 3.0 before install the Pocket PC 2002 SDK to avoid installation issue)
- Go to http://www.gapidraw.com to download and install GapiDraw SDK 1.04.
- Go to http://www.syncdata.it/stlce/stldownload.html to download and install stl_eVC.zip.

- Go to http://www.symbolictools.de/public/pocketconsole/developers/portsdk.htm to download and install PortSDK.msi.

Before you want to run the application code, you need to do the environment setup. First, use the Windows CE Remote File Viewer to import the aria.dll and GD104.dll located in the GapiDraw\dll\ppc2002em directory.

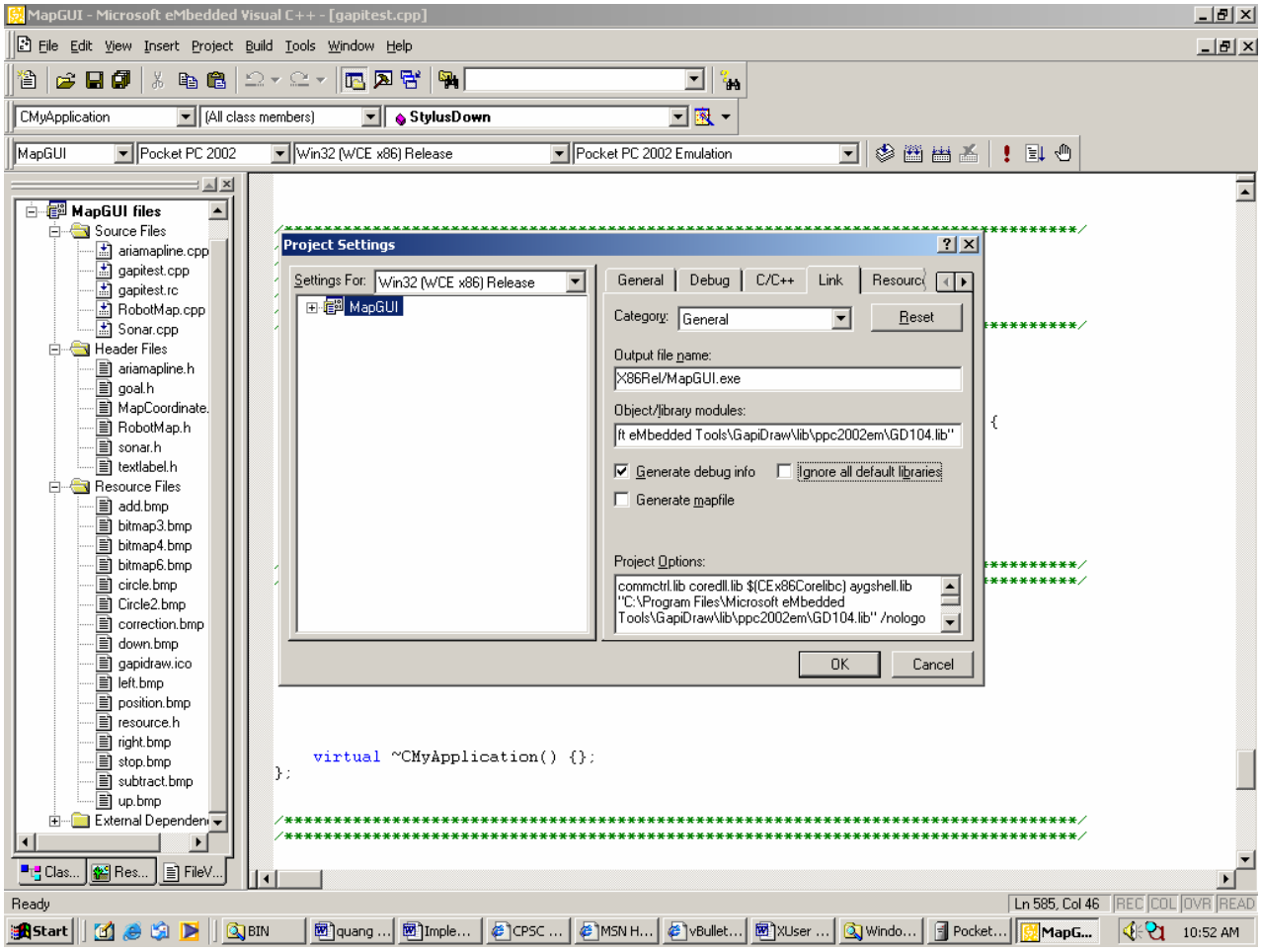


Then after you have opened the desire project Workspace, go to Project → Settings and set the Setting For: to Win32 WCE Arm Release Build. Then under the C/C++ tab, change the Category to Precompiler and set the header to off. Then change to Preprocessor and under Additional include directories: enter c:\program files\portsdk\include,../include,../stl,c:\gapidraw\include



Next, click the Link tab. Under Object/library modules: enter

commctrl.lib coredll.lib aygshell.lib Winsock.lib C:\CPSC483\Class\ariadll\aria\ARMRel\aria.lib "C:\Program Files\PortSDK\lib\ARM\Portlib.lib" "C:\GapiDraw\lib\arm\GD104.lib"



Now everything is set up and you are ready to write your application code.

Operation

In the operation phrase, first make sure the ARIA port is stored on the PDA. Then connect the PDA to the PC using the cradle via USB port. Then build your .exe file. The file will be automatically transferred to the PDA after compilation. Lastly, just execute your program by locating it on the start menu.

After the program is being executed, a user friendly GUI will take up the whole PDA screen. To operate the robot, simply use the stylus to point at the small ball located at the center of the joystick screen. Moving the ball up indicates telling the robot to move forward and vice versa. Moving the ball to the right makes the robot to turn right and vice versa.

While the robot is running around the premises, you can clearly see from the sonar screen the relative reading that correspond to each individual sonar. If the sonar is detecting a far away object, the signal length on the sonar screen will be long and appear in green color. If the sonar signal is not so far away from the object, then the signal length will be shorter and appear in yellow color. If the sonar detects a close up object, then the signal length will be very short and appear in red color.

The first button on the top right hand corner of the PDA screen is the positioning button. It is use to move the robot to a new position on the map where the stylus is pressed. The second button is the correction button. It is use to correct robot's position. The third button is the left button and it is use for turning the robot indicator on the map to left. The fourth button is the right button and it is use for turning the robot indicator on

the map to right. The fifth button is the zoom in button and it is use to zoom in to make the map appear larger. The last button on that column is the zoom out button and it is use to zoom out to make the map appear smaller.

.

CPSC 483: Senior Design

Course Debriefing

Wireless Robot Control with a Personal Digital Assistant

Matt Dean

Ta-Chang Chao

Brady Wied

Quang Dang

May 5, 2003

Question: ***Did your group management style work? If you were to do the project again, what would you do the same, what would you do differently?***

Our group management style did work. For most part it has to do with the structure and organization of this class. Every week's team meeting, the bi-weekly report, the written proposal at the beginning of the project and the critical design review has helped our team to keep focus and stay on track the whole semester long. Also, due to the particular strong leadership and knowledge of certain individuals in the group, all the tasks have been clearly defined and distributed at the early stage of our project. All of this has lead to successful completion of the project.

If we were to do the project again, we will keep the structure and organization of the class the same. What we would do differently is we would try to work as a team instead of individual and also overlaps each other's tasks so that we all can find help from each other if we need help.

Question: ***Are there any particular safety and/or ethical concerns with your product(s)? What steps did your group take to ensure these concerns were addressed? Are there any additional steps you would have taken if you were to do the project again?***

The only safety concern regarding our project will be the robot travel out of the range of wireless transmission and thus might cause harm to others in its way. Our group ensures these concerns were addressed by implementing a shutdown function for any key on the PDA that has been pressed so that the robot will cease to function immediately

when done so. There is also a reset button (Red) on top of the robot that can be used to stop the robot.

If we were to do the project again, we would have probably implement a function that will allow the robot to "backtrack" the path where it just came from shortly before it looses wireless connection signal so that it can regain it's connection signal.

Question: ***Did you test your product(s)? Do they work as advertised? Can you think of any relevant situation in which you haven't tested your product(s)? If you were to do this project again, what additional verification and testing procedures might you add?***

Our group did test our products. In fact, we had tested it almost every time we did the partial integration to ensure the partial product and be implemented further. They usually work as it meant to work. While sometimes we suffer some little setbacks but the problems were usually solved within a short time.

Occasionally we didn't tested our products because either they had just been implemented and someone in the group is asking that for reference or because due to the time constraint that they needed to be quickly integrated with other parts to make a functional unit.

If our group were to do this project again, we might assign a designated individual to do the weekly testing to ensure we have bugs free intermediate products before they were being integrated. We would also keep a logged record of each testing phase to improve the quality of our software.