Fall 2010

Wireless Teleoperated Robot

Ryan Kingsley
Justin Hagan
Jeremy Smoot
Jeremiah Willems

Follow this and additional works at: http://opus.ipfw.edu/etcs_seniorproj_engineering

Opus Citation
Indiana University Purdue University

Wireless Teleoperated Robot

ECE 406 Capstone Senior Design II

Fall 2010

Final Report

Group Members:
Ryan Kingsley
Justin Hagan
Jeremy Smoot
Jeremiah Willems

Advisors:
Dr. Elizabeth A. Thompson
Prof. Paul Lin P.E.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>3</td>
</tr>
<tr>
<td>Abstract</td>
<td>4</td>
</tr>
<tr>
<td>Section I: Selected Design</td>
<td>5</td>
</tr>
<tr>
<td>Section II: Building Process and Changes</td>
<td>22</td>
</tr>
<tr>
<td>Section III: Testing and Results</td>
<td>26</td>
</tr>
<tr>
<td>Section IV: Evaluation and Recommendation</td>
<td>29</td>
</tr>
<tr>
<td>Conclusion</td>
<td>31</td>
</tr>
<tr>
<td>Appendix</td>
<td>33</td>
</tr>
</tbody>
</table>
Acknowledgments

Special thanks to Dr. Elizabeth A. Thompson for being this project’s advisor and always around to help our group with everything we could possibly need. Thanks to Jim Isaacs for coming to IPFW with this project and supporting us. Thanks to Eric Harmison and Robert Carper for their successful first project between IPFW and ITT. And thanks to Jim Isaacs, Eric Harmison, Ross Sneary, and Robert Martin for guiding us and representing ITT’s will for this project.
Abstract/Summary

There is an increasing demand to put mechanized rovers into places deemed too hazardous for humans. Applications for such devices abound. The military is interested in ground-based equivalents to the air drones currently used to fly into and relay information from hostile territory. Other possible applications include inspection of unsafe buildings or inspection and exploration of hard-to-reach places or toxic environments.

The wireless controlled robot built by a 2004 IPFW Senior Design Team was a prototype system that allowed real time operation of a teleoperated robot over a wireless link using off-the-shelf commercially available wireless network cards. ITT is interested in updating and performing some advanced testing on the unit.

We propose to expand the design of the original robot to incorporate an upgraded computer, improved video capabilities, GPS functionality to introduce more autonomy, a digital compass, an open source operating system, and possibly increased terrain capability.
Section I
Original Design

This design uses the existing base with the Sabertooth motor driver. It can supply more current (5A/motor, 10A/motor, and 25A/motor available) which will be necessary for a robot. Also, the regenerative drive feature (recharges batteries whenever robot is slowing down) and the lithium cutoff mode would make this motor driver an improvement over the existing Wirz 203 motor driver. Ark-6310 has an Intel atom processor with 1 GB of ram. This configuration also has 4 USB and 3 serial ports. The Axis M1101 is a network camera; the resolution of the camera is 640x480 and up to 30fps. The camera is also able to compress the video file into many popular formats. In addition to that the Ark-6310 board eliminates the need to have the Eyebot controller board by directly connecting the Ark-6310 to the Sabertooth through the serial port. The Garmin GPS 60CSx includes a GPS and a compass which will allow for semi-autonomous features.
Physical Components:

Base:

Originally, the base was to be reused from the previous project. Jeremy decided to spend his own time and money as a personal project for himself. The design planned to use the original base keeping in mind the future base change as a possibility. The figure below shows the complete base that was redesigned for this project.
Base Station:

The plan for the base station was to include a graphic user interface, so that all of the information that the robot sends out could be seen in an easy to understand interface. All of the video and GPS information would be seen in this GUI. The base station would also be used to control the robot, the general consensus of ITT was to use some sort of game pad to control the robot instead of using a keyboard interface like the previous project. One of the most readily available gamepads was the Xbox 360 controller by Microsoft.

Circuitry:

Power Connections:

Two batteries rated at 12 Amp hours are required for this robot. One battery powers the motor driver and the motors. The other battery is to power all of the other components of the robot. There is a 12V to 19V voltage converter for the robot computer and a 12V to 5V converter for the AXIS IP camera. The components and the converters are all connected in parallel to the second battery.
Power Consumption:

12 Amp-Hour / 4.1 Amps = 2.93 Hours of operation for the motors
12 Amp-Hour / 2 Amps = 6 Hours of operation

This shows that the operation time of the robot is dependent upon the battery that is connected to the motors. The motors have the most power consumption for its time of use. This shows that the operation time meets the need of 1-2 hours operation time that ITT required.

Data Connections:

USB to SRF04

The SRF04 sensors interface to a general purpose I/O port. The ARK6310 computer does not have such a port. A pre-programmed PIC called the GPIO14, which functions as multiple I/O ports, is used to connect the sensors to the computer. The GPIO14 interfaces through I2C. Therefore, an I2C-USB converter is used to connect the GPIO14 device to a USB port of the ARK6310.

The GPIO14 is designed specifically for the SRF04 range sensors. It has an 8 MHz, internal clock which handles all timing required by the sensors. The device has (8) 8-bit internal registers. Only two of these registers are used in this application (command register and result register). The chip always performs the routine that is placed in its’ command register. The commands are sent to the chip through the I2C device. Once the device has performed the routine, it places the results in it’s’ result register. Reading the contents of the result register gives the end result of the commanded routine.

The timing that the GPIO14 is required to handle is shown in the figure below:

The SRF04 has a Trigger Input and an Echo Pulse Output. Once the Trigger input goes to a “high” voltage state, a pulse of Sonic Bursts is sent out from the device. After the pulses are sent, the Echo Pulse Output goes to a “high” voltage state. Once the Sonic Bursts return to the SRF04, the Echo Pulse Output goes to a “low” voltage state. The width of the Echo Pulse can be
related to time, and calibrated to give the total range in inches. All of this is handled internally by the GPIO14 (pre-programmed PIC).

Router to Network Camera

The network camera is connected to the router using an Ethernet connection. Since the network camera does all the compressing of the video, the stream is able to be sent out using the wireless-N router that is connected. The network camera obtains an IP address and sends the video on that IP so that others on the network are able to obtain the video.

The base station uses the Axis Media Center API to decompress the stream and view the video. The base station plays the video using an ActiveX object of Windows Media Player that is embedded in the form.

USB to FT232R to Motor Driver

In the original design, a direct serial connection was going to be used from the serial port to the motor driver. However, the serial port outputs a negative voltage as a logic one and a positive voltage for a logic zero, also the voltage can vary between +/-3 Volts to +/- 15 Volts depending on the device. The motor controller needed a 0-5 Volt input to work correctly. To remedy this problem, our group used a USB FT232R chip to make the signal compatible with the motor controller.

The best option to implement the motor driver was to set it to accept a simplified serial command at 9600 Baud. The motor controller would accept 9-bit serial data streams to know what to do with the motors. The first 8-bits would be the command, while the final 9th bit would be the stop bit. If the motor controller was sent anything between 1-127, it would affect motor 1. Where 1 is full reverse, 64 is stop, and 127 is full forward. Motor 2 would be controlled by 128-255, with 128 being full reverse, 192 as stop, and 255 as full forward.

For the code used, an 8N1 data stream was sent out of the USB port to the motor controller. We only used stop, full forward, and full reverse. The reasoning behind this is because the motors are very high torque, but low speed, as well as the fact that the new base has a much lower center of gravity than the previous base. This allows for instantaneous full speed without the probability of the robot tipping over, as well as a very fast stop due to the high torque gear ratios.

![Dip Switch Settings](image)

Figure: This figure shows the dip switch settings used on the motor controller.
RS-232 to GPS

The Garmin GPS map 60CSx interfaces to the ARK6310 computer through an RS232 serial connection. The device is also able to connect via USB. However, the data format from a USB connection is a Garmin proprietary format, which requires their software to interpret the signal. The RS232 interface uses NMEA format which is not proprietary, and very simple to interpret. In this application, both the RS232 and USB connections are used. The RS232 connection transmits the NMEA data feed to the computer but does not supply power to the GPS device. Therefore, the USB connection is also used to supply power. The USB connection could be replaced by simply installing a pair of AA batteries into the GPS map 60CSx.

**Programming:**

**Robot Computer:**

The Robot uses OpenSUSE 11.3 which is a Linux variant. Therefore the implementation was done using Monodevelop 2.4 IDE and Monodevelop 2.6 framework. Monodevelop is developed by Novell to implement the Microsoft .Net Framework on Linux. This project uses version 3.5. On the robot any of the software that was written to control the hardware was done in C++ and using interop services those functions are called from C# code which the rest of the base station code was written in. The robot is a TCP socket server to the base station. The base station uses three threads.

The first thread checks the proximity sensor to make sure the robot is not going to run into anything. This code returns a 0, 1, or 2. A 2 means that the system timed out, 1 means that it is okay to proceed forward, and finally a 0 means that it is too close to continue. This data is stored in the distance.range variable that can be accessed by the motor threads to decide if the robot should move forward.

The second thread is the motor control thread. This uses a TCP socket connection to receive the movement data from the base station. Using that data the robot will determine what to do. The possible commands are below.

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop Movement</td>
<td>None</td>
</tr>
<tr>
<td>Forward</td>
<td>DpadUp</td>
</tr>
<tr>
<td>Backward</td>
<td>DpadDown</td>
</tr>
<tr>
<td>Right</td>
<td>DpadRight</td>
</tr>
<tr>
<td>Left</td>
<td>DpadLeft</td>
</tr>
<tr>
<td>Override Proximity Sensor</td>
<td>B+DpadUp</td>
</tr>
</tbody>
</table>
Using these commands the check function is called and it determines which function of
the motor class to call to tell the motors what direction to go.

The final thread is the GPS thread. This thread has a TCP socket connection and
when it receives the GPS command from the base station the GPS code is called and the
GPS data is stored in the file gpsXML. The connect class is then called and the GPS data
is sent to the base station. All GPS data is logged in GPS.log file which has the date and
time of each entry.

All errors can be found in the log errors.log. Each entry gives the complete
exception as well as the date and time the error occurred.

GPS

$HCHDG,101.1,,7.1,W*3C
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

The National Marine Electronics Association (NMEA) standard format is shown above. Each
line is called a NMEA sentence. The sentences are transmitted serially at a baud rate of 4800
bits/sec. Each character in the sentence is transmitted with a sequence of (8) binary bits, with
no parity bit, and one stop bit (8N1). The sentences all begin with the $ character. Following
the $ character is the data type. There are many different data types. For this application, only
two data types are used (HCHDG and GPGGA). HCHDG is the digital compass output, and
GPGGA is the fixed information. This fixed information contains latitude, longitude, altitude,
and a time stamp. At the end of each sentence is the * character followed by (2) hexadecimal
characters. This is the checksum value. If the transmission of the NMEA sentence is accurate,
the checksum is equal to the XOR of all characters between (and not including) the $ character
and the * character.

The program to receive the GPS data from the GPS map 60CSx device is written in C++. The
basic functionality is as follows (see source code for more detail):

1) Save the default serial port settings
2) Initialize the serial port for 4800 baud, 8N1
3) Receive a NMEA sentence and store it in an array of characters
4) Check the stored NMEA sentence for HCHDG or GPGGA data type
   a. If HCHDG is found, retrieve the first field which is the compass direction
   b. If GPGGA is found, retrieve the following fields
      i. 1 – Time
      ii. 2 – Latitude
      iii. 3 – Cardinal Latitude
      iv. 4 – Longitude
      v. 5 – Cardinal Longitude
5) Repeat Steps 3-4 until both HCHDG and GPGGA are found
6) Restore default serial port settings from Step 1

This function is called from the base station computer to provide on demand GPS information.
It could easily be utilized to provide information to the robot for autonomous navigation.
Motor Controller

The motor controller has multiple setups to change the kinds of inputs that will determine what the motors will do. The operating mode is chosen depending on the dip switch setup on the motor controller. Out of these operation modes, the simplified serial input option was chosen for the motor control.

Simplified serial is a TTL level 8N1 serial data stream, where the motors are controlled by single byte commands. Which motor and its direction are determined depending on the command. The Baud rate is also determined by the dip switch, the baud can go as high as 38400, however, 9600 was chosen for the reason that is more reliable.

To accomplish this, the code had to send out an 8-bit signal, with no parity bit, and a stop bit at 9600 Baud. The USB port was opened, then the settings was set for that port to accept an 8N1 output at 9600 Baud. The write to port command needed the port, a number, and the set buffer. The number that was used to output the USB port was a decimal digit, the write command sends the digit as its binary counterpart. To control both motors, two write commands were used back-to-back. The baud rate is fast enough that the different would be so minuscule that the human eye would not be able to tell which motor started first.

The code was designed so that the motors would go full speed forward or full speed backward. To turn the robot left or right, the motors on one side of the robot go forward and the motors on the other side go backward. This gives the robot the ability to turn on its spot and very quickly. In the forward command, we incorporated the sensors which are covered in the next section.
SRF04 Sensors

The program to receive the sensor data from the SRF04/GPIO combination is written in C++. All data is sent to the GPIO14 device through the USB port, and then through the I2C-USB device. The baud rate is 19200 bits/sec, with no parity bit, and (2) stop bits. Before calling this code, the GPIO14 chip is initialized with the SET_IN command, which tells the chip to calculate any ranging in inches. The basic functionality is as follows (see source code for more detail):

1) Open USB port
2) Send GET_S4A command to GPIO14 to start ranging of SRF04(A)
3) Wait at least 60 msec for SRF04 device to finish ranging
4) Send read command to GPIO14 (this tells the device to broadcast the contents of its’ result register back to the requesting device)
5) Read in serial data (range in inches) and store the result
6) Send GET_S4B command to GPIO14 to start ranging of SRF04(B)
7) Wait at least 60 msec for SRF04 device to finish ranging
8) Send read command to GPIO14 (this tells the device to broadcast the contents of its’ result register back to the requesting device)
9) Read in serial data (range in inches) and store the result
10) Close USB port
Robot Classes with functions and variables.
Robot Software Flow Chart

Main -> Call Threads -> Range

GPS

Connect -> Accept Connection

Motors

Wait for Update
Base Station Computer:

The base station was implemented using C# using the Microsoft 3.5 .Net Framework. The library SlimDX allows for the use of Microsoft DirectX to be used by a managed code (C#). SlimDX has the API that is used to interface with the Xbox controller. This design implements the base station as the client in the client server arrangement. The GUI gets video from the Axis m1101 network camera. The encoded stream is decoded using the Axis Media Control (AMC) library. The stream is then captured by the Windows Media Player ActiveX plug-in and displayed on the GUI. When the Start button on the GUI is pushed the base station tries connecting to the robot which is the server.

The Config class obtains all the data from the app.config file which tells the software what the connection information that it will need for the TCP socket connection. This is stored statically so when the different threads need information it is stored in one central location for all to obtain.

The acceptConnection class waits for GPS updates from the Robot. The user will specify the update time using the GUI, by putting an integer in the text box. By default if not value is put in by the user updates will occur every 5 minutes. The GPS values are displayed on the GPSForm as seen in the table below. This class also uses the log class to log all the GPS updates for later user by the user. Each entry is time stamped.

In the connect class two connections are made using sockets TCP connections. One connection is used for sending the Xbox controller updates the other connection is used for sending GPS data requests. The movements from the Xbox controller D-pad are updated every 500 mSec. If no position is selected then a “none” is sent.

The GPS connection sends the command “GPS” to the robot, the user specified update time. Then it sleeps until the next update is needed.

The Xbox class retrieves the state of the Xbox controller buttons and sends that information to the connected class to send to the robot. Below are the different commands that the robot recognizes.

<table>
<thead>
<tr>
<th>Stop Movement</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward</td>
<td>DpadUp</td>
</tr>
<tr>
<td>Backward</td>
<td>DpadDown</td>
</tr>
<tr>
<td>Right</td>
<td>DpadRight</td>
</tr>
<tr>
<td>Left</td>
<td>DpadLeft</td>
</tr>
<tr>
<td>Override Proximity Sensor</td>
<td>B+DpadUp</td>
</tr>
</tbody>
</table>
Above is the class diagram for the base station. This shows the public and private members of each class.
Flow Chart of Base Station

Program -> Form1 -> Video

Start Button

GPS

Xbox Controller

Accept Connection

Connect

GPS Form

Wait
GUI
Section II
Building Process

Base Construction:

Originally the base was going to be reused from the previous project; however, Jeremy Smoot took it upon himself to redesign the base that would provide our robot with a greater stability than the previous base. The base was designed out of thin aluminum bars, and was designed so that the motors and battery are on the same height. This makes the new base have a much lower center of gravity compared to the previous base. With the lower center of gravity, the base no longer holds the possibility of flipping that the previous base was prone to. Since the frame is made out of aluminum, it also is easier to hold the components, than the flimsy plastic frame provided from the old base.

Range Sensors:

There was an oversight involved with the SRF04 range sensors. The original plan was to interface the sensors directly to the computer. However, the sensors only connect to an I/O port which the computer does not have. Therefore, in order to correct this problem, a GPIO14 device and I2C-USB device (both from Devantech) were integrated into the system. After much debugging, the sensors still only work some of the time.

Wireless:

The robot computer was going to use a USB wireless card, and the network would be set as an Ad-Hoc between the two USB wireless cards. Ryan Kingsley proposed to ITT that the group uses a wireless router instead, and connect it to the Ethernet port. Ryan’s reasoning for this change is because the Ethernet port did not need any drivers to set up. Since ITT’s communication devices used Ethernet as well, they would not have to do any installation or work to get their product to be compatible with our robot. Whereas, if our group decided to make the network Ad-Hoc ITT would have to install drivers and work on the code for it themselves. This made the solution easy for us and ITT to implement.

Power:

For the power, we are using two batteries to power our components. One battery is singularly set out to supply the motors with power. The other battery is connected to a voltage regulator block, which connects to a power strip. The computer, web camera, and router are all powered by connecting to the power strip.
Programming:

The group chose the operating system OpenSUSE, this is an operating system that is in the public domain and free for anyone to use. The group started out using version 11.2, however, when 11.3 came out it was decided to make the change because the newer version fixed some of the remote desktop errors that were encountered. OpenSUSE 11.3 is based off Linux Kernel 2.6.34. The Linux kernel is one of the most stable operating system kernels available. Novell is one of the key contributors of OpenSUSE project.

All of the programming on the robot computer was done using the Monodevelop 2.4 IDE and Monodevelop 2.6 framework. Monodevelop is developed by Novell to implement the Microsoft .Net Framework on Linux. The software developed on the robot computer was written in C++ and the base station used interop services developed from C# code.

GUI:

The base station code was all developed using Microsoft Visual Studio 2008 Integrated Development Environment (IDE) and using the Microsoft 3.5 .Net Framework. The GUI was build using Windows Forms which is included in the .NET Framework.

The communication between the robot and the base station were done using the Sockets class which is included in the System.Net.Sockets as part of the .NET Framework. By using this, a TCP connection was established and a client-server configuration was established.

SlimDX is a free open source framework that takes the Microsoft DirectX framework and wraps it in a library so that managed languages for example C# are able to use the DirectX library. DirectX was developed to for gaming and graphics. The Xbox controller was interfaced using the SlimDX library. Thus, it made the Xbox controller very easy to use.

GPS:

The GPS device was interfaced to the computer as planned. The RS232 serial connection was used for data transmission, while the USB connection was used to provide power to the GPS device.
**Motor Controller:**

The motor controller was originally going to be controlled by a serial port, using simple serial commands out of its output pin. It was under the idea that the serial port sent out bits at a TTL level. However, once it was found out that it was not the case, we had to use a TTL level converter. Ryan Kingsley had bought one earlier, however it was for a USB connection. The robot computer had more than enough USB ports available for use, so the port changed from serial to USB. To implement this change in the code, the port had to be redefined to a USB port. Other than changing the port that was called, no other changes were needed to be able to implement the code.
Section III
Testing

Many tests are needed to ensure, and show that this robot met all of ITT’s requirements for this project. The following are the tests that are to be preformed and the procedures that will be followed to gather the results and show that the robot is up to ITT’s expectations.

1. The robot must be able to move with an approximately 3 lb. radio on it.

   In order to test this, a 5 lb weight will be added on top of the robot to simulate the radio. The 5 lb. weight should be more than an approximately 3 lbs. and it will sufficiently show that the robot is capable of handling the weight of any of ITT’s small radios.

   The testing showed that the robot can easily handle more than 5 lbs. of weight. This means that the robot will be able to hold any of the small radios that ITT has planned to test on this platform.

2. Sensors are to be used for obstacle avoidance. (Null due to problems with sensors)

   The sensors on the robot are situated differently than sensors on the previous robot, and do not have the same area coverage. To test this area, we will try approaching a wall at various angles to see if the robot will stop before bumping into the wall and the distance from the wall when it stops.

3. The robot should have a travel speed of 2-5 MPH and should be able to be used on different surfaces.

   To test the ground speed of the robot, we will time the robot traveling at full speed across the distance of one meter. We will then use equivalence equations to change it from meters per second to miles per hour. We will test this on a flat and favorable surface to attain max travel speeds. To test the robot on outside surfaces, we will run the robot through different environments, and note the differences as compared to a favorable indoor tile surface.

   Since ITT decided to use the existing motors for this project, we have no control of increasing the speeds. We were donated 4 motors that are exactly like our previous motors, however, they have twice the RPM and less torque. We measured the speed of these motors to be 0.65 MPH, which is still slow but twice as fast as the previous motors.
4. Wireless range of robot is to have a 200 meter range in line of sight at minimum, as well as the distances of the wireless-g network and wireless-n network are to be compared.

To test this aspect of the robot, a long flat area is needed with no obstacles down a straight path. We will run the robot straight down that path and measure the distance it travels on both networks when the robot goes out of distance from our wireless signal and stops.

Due to weather conditions, we were unable to completely ascertain the full range of our robot. The robot was not build with the idea of traversing through a wet and salty environment. We do have a test recorded of the robot going 180 meters, however this test had line of sight issues as a building was in-between the base station and the robot. For this test, the base station was located at the entrance of the ET building and the connection was lost at the entrance of the Science building at IPFW. By looking at a map one can see that about half of Neff Hall is in-between those two points. From this observation, we can safely conclude that the robot will traverse farther than the requirement of 200 meters if it remains in line of sight.

5. When testing the range of the robot on both wireless networks, we need to check the wireless data link as well. This is needed to check to see how many errors we get from the different networks as well as check to make sure the data rate is above 600 k bits/second.

We will use a program called Wireshark to check the amount of errors we get from our wireless transmission as well as the data rate while we do the testing of the range of both of the wireless networks. This will give a more complete comparison between the two networks.

We were unable to conduct the test of this due to the outside conditions. The robot was not made to be weather proof, so we thought that it would be prudent of us not to test this with the safety of the robot in mind.

6. Robot to base station video lag time. The expected lag time was 2-3 seconds by ITT.

This test is to show how long it takes to video to be captured on the robot and shown on the base station computer. We will perform an action or movement in front of the video camera, and time the difference with different distances between the robot and the base station.

We ran many tests, but the ranges were limited due to our testing restrictions. The lag times ranged anywhere from 1.4 seconds to 1.6 seconds, this shows that the video lag is well within the expectations of ITT for this project.
Section IV
Evaluation and Recommendations

This project has met the expectations of ITT and fulfills their requirements. The group would recommend upgrading the sensors to SRF05. These sensors have a direct I2C connection that would be compatible with our computer. Another recommendation would be to add semiautonomous navigation to the robot. Image processing, such as using edge detection, would be a good addition to the capabilities of this robot. A servo for the camera would be added to allow pan and tilt to better view the surroundings of the robot. This robot has many different ways that it could be expandable to increase its capabilities greatly.
Conclusion
In conclusion, this project was a success and it meets IPFW and ITTs expectations for this project. It works very well with the only problems being the sensors, which ITT decided were not necessary to fulfill their needs for this robot. This project allowed us to show that we could apply the concepts and logic we learned at IPFW to complete a successful senior design project.
The MAX 97 Mobile Base Platform

Technical Details:
- Each deck base is 12 in x 12 in diameter (30cm x 30cm).
- The base has dual 12 volt 20 in lb torque drive motors (The Max speed is 39 feet per minute under full load), a gear change can be requested which will double the speed.
- The drive wheels are six inches (15 cm) in diameter.
- The caster wheel is three inches (7.5 cm) in diameter.
- The base is balanced with a single rear caster.
- A motor driver kit is included with the base (Wirz #203)
- Two free optics are included and can be used for simple pulse encoders.
- The maximum recommended payload is 35 lbs. (13.6Kgs.).
Garmin GPS 60CSx

**Technical Details:**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit dimensions, WxHxD</td>
<td>2.4&quot; x 6.1&quot; x 1.3&quot; (6.1 x 15.5 x 3.3 cm)</td>
</tr>
<tr>
<td>Display size, WxH</td>
<td>1.5&quot; x 2.2&quot; (3.8 x 5.6 cm)</td>
</tr>
<tr>
<td>Display resolution, WxH</td>
<td>160 x 240 pixels</td>
</tr>
<tr>
<td>Display type</td>
<td>256 level color TFT</td>
</tr>
<tr>
<td>Weight</td>
<td>7.5 oz (213 g) with batteries</td>
</tr>
<tr>
<td>Battery</td>
<td>2 AA batteries (not included)</td>
</tr>
<tr>
<td>Battery life</td>
<td>18 hours, typical</td>
</tr>
<tr>
<td>Waterproof</td>
<td>yes (IPX7)</td>
</tr>
<tr>
<td>Floats</td>
<td>no</td>
</tr>
<tr>
<td>High-sensitivity receiver</td>
<td>yes</td>
</tr>
<tr>
<td>Interface</td>
<td>serial and USB</td>
</tr>
<tr>
<td>RoHS version available</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Maps & Memory:**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basemap</td>
<td>yes</td>
</tr>
<tr>
<td>Ability to add maps</td>
<td>yes</td>
</tr>
<tr>
<td>Built-in memory</td>
<td>no</td>
</tr>
<tr>
<td>Accepts data cards</td>
<td>64 MB microSD™ card (included)</td>
</tr>
<tr>
<td>Waypoints/favorites/locations</td>
<td>1000</td>
</tr>
<tr>
<td>Routes:</td>
<td>50</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Track log:</td>
<td>10,000 points, 20 saved tracks</td>
</tr>
</tbody>
</table>

**Features:**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Yes/No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic routing (turn by turn routing on roads):</td>
<td>yes</td>
</tr>
<tr>
<td>Electronic compass:</td>
<td>yes</td>
</tr>
<tr>
<td>Touchscreen:</td>
<td>no</td>
</tr>
<tr>
<td>Barometric altimeter:</td>
<td>yes</td>
</tr>
<tr>
<td>Camera:</td>
<td>no</td>
</tr>
<tr>
<td>Geocaching-friendly:</td>
<td>yes</td>
</tr>
<tr>
<td>Custom maps compatible:</td>
<td>no</td>
</tr>
<tr>
<td>Photo navigation (navigate to geotagged photos):</td>
<td>no</td>
</tr>
<tr>
<td>Outdoor GPS games:</td>
<td>yes</td>
</tr>
<tr>
<td>Hunt/fish calendar:</td>
<td>yes</td>
</tr>
<tr>
<td>Sun and moon information:</td>
<td>yes</td>
</tr>
<tr>
<td>Tide tables:</td>
<td>no</td>
</tr>
<tr>
<td>Area calculation:</td>
<td>yes</td>
</tr>
<tr>
<td>Custom POIs (ability to add additional points of interest):</td>
<td>yes</td>
</tr>
<tr>
<td>Unit-to-unit transfer (shares data wirelessly with similar units):</td>
<td>no</td>
</tr>
<tr>
<td>Picture viewer:</td>
<td>no</td>
</tr>
<tr>
<td>Garmin Connect™ compatible (online community where you analyze, categorize and share data):</td>
<td>no</td>
</tr>
</tbody>
</table>
Linksys WUSB300N

Technical Details:

- Standards IEEE 802.11b, IEEE 802.11g, Draft IEEE 802.11n, USB 1.1, USB 2.0
- Ports USB Port
- LEDs Power, Link/Act
- Modulations 802.11b: CCK, QPSK, BPSK
- 802.11g: OFDM
- Wireless-N: BPSK, QPSK, 16-QAM, 64-QAM
- RF Pwr (EIRP) in dBm 802.11b: 14±1dBm (Typical)
- 802.11g: 14±1dBm (Typical)
- Wireless-N: 14±1dBm (Typical)
- Receive Sensitivity in dBm 11Mbps @ -86dBm (Typical)
- 54Mbps @ -68dBm (Typical)
- Wireless-N @ -62dBm (Typical)
- Power Consumption TX: <480mA (Maximum)
- RX: <390mA (Maximum)
- Security features WEP, WPA and WPA2 Encryption Security
- Security key bits Up to 256-bit encryption
- Dimensions 2.24” x 0.39” x 3.98” (57mm x 10mm x 101mm)
- Certification FCC, Wi-Fi (802.11b/g)
- Operating Temp. 32 to 131°F (0 to 55°C)
- Storage Temp. -4 to 176°F (-20 to 80°C)
- Storage Humidity 5 to 90% Noncondensing
- Operating Humidity 10 to 85% Noncondensing

Minimum Requirements
- 600 MHz or Faster PC
- 256 MB of RAM Memory
Sabertooth Dual 10A Motor Driver

Technical Details:

- Up to 18V in: 10A continuous, 15A peak per channel.
- 24V in: 8A continuous, 10A continuous with additional heatsinking/airflow, 15A peak per channel.
- Synchronous regenerative drive
- Ultra-sonic switching frequency
- Thermal and overcurrent protection
- Lithium protection mode
Axis M1101 Network Camera

Technical Details:

- Video Input Camera Type: Network camera Color or B&W:
- Color Digital Video Format: H.264, MJPEG, MPEG-4
- Features: Backlight compensation, Brightness control, Color control, Contrast control, Digital image rotation, Sharpness control, White balance
- Min Focal Length: 4.4 mm Shutter Speed (Max): 1/4 sec Shutter Speed (Min): 1/5000 sec
- Interface Required Connector Type: RJ-45
- Interface: Ethernet 10Base-T/100Base-TX Total Qty: 1
- Type: Network
- Optical Sensor Optical Sensor Size: 1/4" Optical Sensor Size (metric): 6.4 mm (1/4"") Optical Sensor Type: CMOS
- Environmental Parameters Humidity Range Operating: 20 - 80% Max Operating Temperature: 122 °F Min Operating Temperature: 32 °F
- Header Compatibility: PC
- Power Device Form Factor: External Type: Power adapter
- Video Capture Frame Rate: 30 fps Image Resolution: 640 x 480
- Dimensions & Weight Depth: 1.3 in Height: 3.7 in Weight: 3.3 oz Width: 2.3 in
- Flash Memory Installed Size: 32 MB
- Power Low Voltage Power: DC 5 V
- RAM Installed Size: 64 MB
Technical Details:

- Processor System: CPU Intel Atom Processor N270 1.60 GHz
- Front Side Bus: 533 MHz
- L2 Cache: 512 KB
- Chipset: Intel 945GSE + ICH7M
- Memory Technology: 533 SDRAM
- Max. Capacity: 200-pin SODIMM x1 up to 2 GHz
- Graphics Chipset: Intel 945GSE GMCH integrated Graphic Media Accelerator 950
- Display Memory: Shared system memory up to 224 MB SDRAM
- LVDS: Single channel 18-bit/Dual channel 36-bit LVDS
- Ethernet
- Ethernet Interface: 10/100/1000 Mbps
- Controller Dual Realtek RTL8111C Gigabit LAN Connector 2 x RJ-45 connectors
- Watchdog Timer Output System reset Interval Programmable 1~255 sec.
- Storage Interface: 2.5" HDD Bay 1
- Compact Flash 1 x Type 1/II Socket (on the solder side of the motherboard)
- I/O Interface:
  - USB 4
  - PS/2 2
  - COM 3
  - RJ-45 2
  - Audio 2 (Mic-In, Line-Out)
  - VGA 1
- Extra I/O cutouts 1 x LVDS, 1 x COM
- Miscellaneous Overheating Protection
- 3 x LED of Power, HDD and Temperature Control
- 1 x ATX Power On/Off switch button
- Software Compatibility
- Operating System: Microsoft Widows 2000, Windows XP and Windows XP Embedded
- Power Supply Output Rating:
  - 75W
- **Input Voltage**: 14 VDC ~ 24 VDC @ 6.4 A ~ 3.75 A (Max.); 19 VDC @ 4.74 A (Max.)
- **Output Voltage**: +5 V with 5 VSB @ 6 A (+5 VSB @ 2A Max.); +3.3 V @ 5 A; +12 V @ 2 A, -12 V @ 0.3 A; -5 V @ 0.3 A
- **Environment Operating Temp.**: 0 ~ 40° C (32 ~ 104° F)
- **Operating Humidity**: 10 ~ 85% @ 40° C, non-condensing
- **Vibration Loading**
  - Duration Vibration: 1 Grms, IEC 60068-2-64, random, 5 ~ 500 Hz, 1 Oct./min, 1 hr/axis
- **Altitude**: 0 to 3,048 m (0 ~ 10,000 ft)
- **Physical Characteristics**
  - Dimensions (W x H x D): 232 x 65 x 232 mm (9.1” x 2.6” x 9.1”)
  - Net Weight: 2.74 Kg (6.03 lb)
FT232R Breakout


### 1.1 Hardware Features

- Single chip USB to asynchronous serial data transfer interface.
- Entire USB protocol handled on the chip - No USB-specific firmware programming required.
- UART interface support for 7 or 8 data bits, 1 or 2 stop bits and odd / even / mark / space / no parity.
- Fully assisted hardware or X-On / X-Off software handshaking.
- Data transfer rates from 300 baud to 3 Megabaud (RS422 / RS485 and at TTL levels) and 300 baud to 1 Megabaud (RS232).
- 256 byte receive buffer and 128 byte transmit buffer utilising buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free VCP and D2XX drivers eliminate the requirement for USB driver development in most cases.
- In-built support for event characters and line break condition.
- New USB FTDIChip-ID™ feature.
- New configurable CBUS I/O pins.
- Auto transmit buffer control for RS485 applications.
- Transmit and receive LED drive signals.
- New 4.5MHz, 24MHz, 12MHz, and 6MHz clock output signal Options for driving external MCU or FPGA.
- FIFO receive and transmit buffers for high data throughput.
- Adjustable receive buffer timeout.
- Synchronous and asynchronous bit bang mode interface options with RD# and WR# strobes.
- New CBUS bit bang mode option.
- Integrated 1024 Bit internal EEPROM for storing USB VID, PID, serial number and product description strings, and CBUS I/O configuration.
- Device supplied preprogrammed with unique USB serial number.
- Support for USB suspend and resume.
- Support for bus powered, self powered, and high-power bus powered USB configurations.
- Integrated 3.3V level converter for USB I/O.
- Integrated level converter on UART and CBUS for interfacing to 5V - 1.8V Logic.
- True 5V / 3.3V / 2.8V / 1.8V CMOS drive output and TTL input.
- High I/O pin output drive option.
- Integrated USB resistors.
- Integrated power-on-reset circuit.
- Fully integrated clock - no external crystal, oscillator, or resonator required.
- Fully integrated AVCC supply filtering - No separate AVCC pin and no external R-C filter required.
- UART signal inversion option.
- USB bulk transfer mode.
- 3.3V to 5.25V Single Supply Operation.
- Low operating and USB suspend current.
- Low USB bandwidth consumption.
- UHCI / OHCI / EHCI host controller compatible
- USB 2.0 Full Speed compatible.
- -40°C to 85°C extended operating temperature range.
- Available in compact Pb-free 28 Pin SSOP and QFN 32 packages (both RoHS compliant).
ASUS RT-N16 Multi-functional Wireless N Router

Technical Details:

Specifications

**Hardware**

- **Ethernet ports**: WAN x 1, LAN x 4 RJ-45 for 10/100/1000 Base-T
- **Supports Ethernet and 802.3 with max. bit rate 10/100/1000 Mbps and auto cross-over function (MDI-X)**
- **Antenna**: 3 x external antenna
- **USB**: 2 x USB port
- **Power adapter**: AC input 100V ~ 240 V; DC output 12V with max. 1.25 A current
- **Size**: 216mm x 161.9mm x 40.5 mm
- **Weight**: 470g

**Wireless LAN**

- **Operating Frequency**: 2.4~2.5 GHz
  - 802.11n Draft up to 300Mbps
  - 802.11g 6, 9, 12, 18, 24, 36, 48, 54Mbps
  - 802.11b 1, 2, 5.5, 11Mbps
  - n mode: 15.8~19.5dBm
  - g mode: 15.5~16.5dBm
  - b mode: 15.8~19.5dBm

- **Output power**
  - At 54Mbps data rate
    - ANT0 -75dBm
    - ANT1 -75dBm
    - Supports 64/128-bit WEP
    - WPA-PSK, WPA2-PSK
    - WPA-Enterprise, WPA2-Enterprise

- **Encryption/Authentication**
  - Radius with 802.1x
Software

- EZQoS (Easy Quality of Service): Allows multiple network activities (FTP, Gaming, P2P) to work smoothly at the same time
- DHCP Server: Supports up to 253 IP addresses; Changeable DHCP lease time, IP pool, domain name; Static mapped IP
- Web-based administration: Supports IE 5.5 or later, Firefox 2.0.0.1 or later; Managed from LAN and Internet; Password Setting
- System Event Log
- Firmware Upgrade: Web Interface, Bootloader

Management

- Save/Restore Configuration File

Internet connection type

- Automatic IP, Static IP, PPPoE (MPPE supported), PPTP, L2TP

- Firewall: NAT and SPI (Stateful Packet Inspection), intrusion detection including logging
- Logging: Dropped packet, security event, Syslog
- Filtering: Port, IP packet, URL keyword, MAC address

Security
Up to 2 SRF04's or 2 SRF05's controlled, including all timing.
Up to 14 general purpose Input/Output lines.
Up to 5 Analogue input channels with 10-bit A/D conversion.
1 PWM output usable as an 8-bit D/A with a simple filter.
I2C address 0x40, can be changed to allow up to 8 devices on the same I2C bus.
6 I/O lines have programmable pull-up resistors built into the chip.
Individual control of each pin for Input or Output.
Simple commands for Bit Set, Bit Clear and Bit Toggle.
Easy I2C bus control, similar protocol as popular EEPROM's such as 24C02.

**Connection Diagram** showing GPIO14 Pin connections.

```
<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RA2/AN2</td>
</tr>
<tr>
<td>2</td>
<td>RA3/AN3</td>
</tr>
<tr>
<td>3</td>
<td>RA4/AN4</td>
</tr>
<tr>
<td>4</td>
<td>RA5</td>
</tr>
<tr>
<td>5</td>
<td>Vss (Gnd)</td>
</tr>
<tr>
<td>6</td>
<td>RB0</td>
</tr>
<tr>
<td>7</td>
<td>SDA</td>
</tr>
<tr>
<td>8</td>
<td>RB2</td>
</tr>
<tr>
<td>9</td>
<td>RB3/PWM</td>
</tr>
<tr>
<td>10</td>
<td>SCL</td>
</tr>
<tr>
<td>11</td>
<td>RB5</td>
</tr>
<tr>
<td>12</td>
<td>RB6</td>
</tr>
<tr>
<td>13</td>
<td>RB7</td>
</tr>
<tr>
<td>14</td>
<td>Vcc (+5V)</td>
</tr>
<tr>
<td>15</td>
<td>RA6</td>
</tr>
<tr>
<td>16</td>
<td>RA7</td>
</tr>
<tr>
<td>17</td>
<td>RA0/AN0</td>
</tr>
<tr>
<td>18</td>
<td>RA1/AN1</td>
</tr>
</tbody>
</table>
```

The GPIO14 requires a 5v power supply. Current consumption is very low - around 2mA. A 100n capacitor should be connected between the 5v supply and Ground close to the chip. I2C is connected to SDA (pin 7) and SCL (pin 10). You should have pull-up resistors on the SDA and SCL lines. A value of 4k7 is normally OK. We use 1k8 resistors for better noise immunity, but anything from 1k8 to 10k should be OK. You only need one pair of pull-up resistors on the whole I2C bus, not for each device. The pull-up resistors are normally located on the bus master. Our CM02 already has 1k8 resistors fitted on the module.
There are two 8-bit ports on the GPIO14. Port A and Port B. The individual bits in Port A are RA0 to RA7, and for Port B they are RB0 to RB7. Only 6 of these are available because RB1 is used for the SDA line and RB4 is used for the SCL line. You can still write anything you wish to Port B though, because the firmware will prevent the I2C lines from being overwritten. Bits RB0, RB2, RB3, RB5, RB6 and RB7 are available for general purpose I/O. Some pins can have other functions. RA0 to RA4 can be used for analogue inputs. RB3 can be used as a PWM output. Bits RB0, RB2, RB3, RB5, RB6 and RB7 can have a pull-up resistor enabled. RA5 is an input only pin, this is a limitation of the PIC16F818 used. All I/O lines default to inputs on power up.

**Internal Registers**
The GPIO14 has eight internal registers, some of which have different functions for read and write.

<table>
<thead>
<tr>
<th>Register</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Firmware Revision Number</td>
<td>Command Register</td>
</tr>
<tr>
<td>1</td>
<td>Result high byte</td>
<td>Port A Input/Output Mask</td>
</tr>
<tr>
<td>2</td>
<td>Result low byte</td>
<td>Port B Input/Output Mask</td>
</tr>
<tr>
<td>3</td>
<td>A/D Control</td>
<td>A/D Control</td>
</tr>
<tr>
<td>4</td>
<td>Port A</td>
<td>Port A</td>
</tr>
<tr>
<td>5</td>
<td>Port B</td>
<td>Port B</td>
</tr>
<tr>
<td>6</td>
<td>PWM</td>
<td>PWM</td>
</tr>
<tr>
<td>7</td>
<td>Nothing (Reads Zero)</td>
<td>I2C Address Change</td>
</tr>
<tr>
<td><strong>Voltage</strong></td>
<td>5.0V</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td><strong>Current</strong></td>
<td>30mA Typ. 50mA Max</td>
<td></td>
</tr>
<tr>
<td><strong>Frequency</strong></td>
<td>40kHz</td>
<td></td>
</tr>
<tr>
<td><strong>Max Range</strong></td>
<td>3 meters</td>
<td></td>
</tr>
<tr>
<td><strong>Min Range</strong></td>
<td>3 centimeters</td>
<td></td>
</tr>
<tr>
<td><strong>Input Trigger</strong></td>
<td>10uSec minimum, TTL level pulse</td>
<td></td>
</tr>
<tr>
<td><strong>Echo Pulse</strong></td>
<td>Positive TTL level signal, proportional to range</td>
<td></td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>0.4 oz</td>
<td></td>
</tr>
<tr>
<td><strong>Sensitivity</strong></td>
<td>Detect a 3cm diameter stick at &gt; 2 m</td>
<td></td>
</tr>
</tbody>
</table>
### I2C-USB

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C Devices</td>
<td>Any registered or unregistered I2C device.</td>
</tr>
<tr>
<td>Host support</td>
<td>Drivers available for Windows XP64, XP, 2000, ME, 98, CE.NET V4.2, Linux,</td>
</tr>
<tr>
<td></td>
<td>Mac OS8, OS9, OSX.</td>
</tr>
<tr>
<td>Voltage</td>
<td>Self powered - 5V available</td>
</tr>
<tr>
<td>Current</td>
<td>Up to 70mA available for your own circuits</td>
</tr>
<tr>
<td>On-board Pull-up resistors</td>
<td>4.7k</td>
</tr>
<tr>
<td>Digital I/O</td>
<td>1 Input and 2 I/O available when not using I2C.</td>
</tr>
<tr>
<td>Analog Input</td>
<td>2 10-bit channels available when not using I2C.</td>
</tr>
<tr>
<td>I2C clock rate</td>
<td>100 kbps</td>
</tr>
</tbody>
</table>