USAR Interface
2006-2007

Kenny Darbonne
darbonne@engr.colostate.edu
Stephen Goebel
sgoebel@engr.colostate.edu
Shea Robinson
shea@engr.colostate.edu
Josh Schmitt
schmitty@engr.colostate.edu
Steve Tranby
stevemt@engr.colostate.edu
Brent Wilkins
bwilkins@engr.colostate.edu

Advised by:
Carl Kaiser
clkaiser@engr.colostate.edu

April 29, 2007

Colorado State University
Department of Mechanical Engineering
Department of Electrical Engineering
Senior Practicum Projects Program
# Table of Contents

Abstract ........................................................................................................................................... 6
Introduction/Problem Statement ........................................................................................................ 8
  Statement of Work ......................................................................................................................... 8
  Constraints ................................................................................................................................. 10
  Criteria .................................................................................................................................. 12
Preliminary Design ............................................................................................................................. 17
  UI .......................................................................................................................................... 17
    GUI ................................................................................................................................. 17
    User Input .......................................................................................................................... 18
    Linksys WRT55AG ............................................................................................................ 19
Mapping (SLAM) ............................................................................................................................. 19
  Dimensions (2D, 2.5D, 3D, 6D) ............................................................................................... 20
  Discrete Belief ........................................................................................................................ 21
  Continuous Belief .................................................................................................................... 22
Robot Controller .............................................................................................................................. 23
  Wireless ............................................................................................................................... 27
Sensors ......................................................................................................................................... 28
  LADAR ................................................................................................................................. 28
  Video ................................................................................................................................. 28
  Audio .................................................................................................................................. 30
  CO₂ .................................................................................................................................... 30
  Inertial Measurement Sensor ............................................................................................... 34
Design Details .................................................................................................................................. 36
  UI .......................................................................................................................................... 36
    GUI ................................................................................................................................. 36
    User Input .......................................................................................................................... 36
    Linksys WRT55AG ............................................................................................................ 37
Mapping (SLAM) ............................................................................................................................. 37
  Histogram Scan Matching ....................................................................................................... 37
  Occupancy Grid ...................................................................................................................... 45
Mapping ......................................................................................................................................... 45
  SDL ..................................................................................................................................... 45
  DEV CPP ............................................................................................................................. 46
  SDL into DEV CPP ............................................................................................................... 46
  Mapping Methods ................................................................................................................. 46
  Destructor ............................................................................................................................. 47
  Arrow(x, y, theta) .................................................................................................................... 47
  Arrow_pop(x/y)(x1,y1,x2,y2) .............................................................................................. 48
  Scale() ................................................................................................................................. 48
  Resize(x1,y1,x2,y2) .............................................................................................................. 48
  X and y_interp(x1,y1,x2,y2) ............................................................................................... 49
  Xy_interp(x1,y1,x2,y2) .......................................................................................................... 49
  Path_interpx, path_interpy, and path_interp(x1,y1,x2,y2) .................................................. 49
  Vector_Read ........................................................................................................................... 49
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>How the Mapping Algorithm Works</td>
<td>50</td>
</tr>
<tr>
<td>Previous Considerations</td>
<td>51</td>
</tr>
<tr>
<td>Shading</td>
<td>56</td>
</tr>
<tr>
<td>SLAM Testing</td>
<td>56</td>
</tr>
<tr>
<td>Robot Controller</td>
<td>62</td>
</tr>
<tr>
<td>CPU-1433</td>
<td>62</td>
</tr>
<tr>
<td>RS-232 De-multiplexer</td>
<td>63</td>
</tr>
<tr>
<td>Wireless</td>
<td>63</td>
</tr>
<tr>
<td>Sensors</td>
<td>64</td>
</tr>
<tr>
<td>LADAR</td>
<td>64</td>
</tr>
<tr>
<td>Video</td>
<td>65</td>
</tr>
<tr>
<td>Audio</td>
<td>68</td>
</tr>
<tr>
<td>CO2</td>
<td>68</td>
</tr>
<tr>
<td>Inertial Measurement Sensor</td>
<td>69</td>
</tr>
<tr>
<td>Conclusions and Recommendations</td>
<td>72</td>
</tr>
<tr>
<td>Technical References</td>
<td>73</td>
</tr>
<tr>
<td>Appendices</td>
<td>74</td>
</tr>
<tr>
<td>MG811 Carbon-Dioxide Sensor Datasheet</td>
<td>75</td>
</tr>
<tr>
<td>ADXL330 3-Axis Accelerometer Datasheet</td>
<td>77</td>
</tr>
<tr>
<td>IDG-300 Dual-Axis Gyro Datasheet</td>
<td>91</td>
</tr>
<tr>
<td>Linksys WRT55AG Wireless Router Datasheet</td>
<td>97</td>
</tr>
<tr>
<td>CPU-1433 Single Board Computer Datasheet</td>
<td>99</td>
</tr>
<tr>
<td>Inertial Measurement Unit Schematic</td>
<td>101</td>
</tr>
<tr>
<td>Hokuyo URG-04LX LADAR Datasheet</td>
<td>102</td>
</tr>
<tr>
<td>FLIR ThermoVision A10 Camera Datasheet</td>
<td>104</td>
</tr>
<tr>
<td>Zerolux PC209IR Camera Datasheet</td>
<td>106</td>
</tr>
<tr>
<td>Sensoray Model 314 Video/Audio Grabber Datasheet</td>
<td>107</td>
</tr>
<tr>
<td>IsTec Student Research Posters</td>
<td>121</td>
</tr>
<tr>
<td>Engineering Days Poster</td>
<td>123</td>
</tr>
<tr>
<td>Tech Reports</td>
<td>124</td>
</tr>
</tbody>
</table>
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Hardware Layout</td>
<td>7</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Software Layout</td>
<td>7</td>
</tr>
<tr>
<td>Figure 3</td>
<td>First GUI Design Concept</td>
<td>17</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Final GUI Design Concept</td>
<td>18</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Game pad Controls Concept</td>
<td>19</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Mapping Types</td>
<td>20</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Blocks</td>
<td>21</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Characteristic Points</td>
<td>21</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Elevation Blocks</td>
<td>22</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Continuous Elevation</td>
<td>23</td>
</tr>
<tr>
<td>Figure 11</td>
<td>The hardware difference between ISA (PC104) and PCI (PCI104). PC104+ Combines both bus systems.</td>
<td>24</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Images of the KoreBot controller (<a href="http://www.roadnarrowsrobotics.com/acatalog/RN-Kor-Bot_TopBot.png">http://www.roadnarrowsrobotics.com/acatalog/RN-Kor-Bot_TopBot.png</a>)</td>
<td>25</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Industrial PC’s W-LX-800 (<a href="http://www.industrialpc.com/images/sbc/epic/w-lx.jpg">http://www.industrialpc.com/images/sbc/epic/w-lx.jpg</a>)</td>
<td>26</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Eurotech’s CPU-1433 (<a href="http://65.39.65.44/photos/30224.jpg">http://65.39.65.44/photos/30224.jpg</a>)</td>
<td>27</td>
</tr>
<tr>
<td>Figure 15</td>
<td>PC Microphone</td>
<td>30</td>
</tr>
<tr>
<td>Figure 16</td>
<td>PIC16F876A Microcontroller (courtesy of sparkfun electronics)</td>
<td>31</td>
</tr>
<tr>
<td>Figure 17</td>
<td>CO2 Sensor Transducer</td>
<td>31</td>
</tr>
<tr>
<td>Figure 18</td>
<td>CO2 Transducer Break-out Board</td>
<td>31</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Serialization Circuit Board</td>
<td>32</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Sparkfun PIC-PG2 Programmer</td>
<td>32</td>
</tr>
<tr>
<td>Figure 21</td>
<td>PIC Microcontroller Flowchart</td>
<td>33</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Amplification Circuit</td>
<td>34</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Amplification Circuit Simulation</td>
<td>34</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Sparkfun Sense-5DOF Inertial Measurement Unit (courtesy of sparkfun electronics)</td>
<td>35</td>
</tr>
<tr>
<td>Figure 25</td>
<td>GUI on Startup</td>
<td>36</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Comparison of Raw Data (mm) and Angle Histogram</td>
<td>38</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Two Consecutive Scans, Angle Histogram Comparison</td>
<td>38</td>
</tr>
<tr>
<td>Figure 28</td>
<td>LADAR Data Point Definition</td>
<td>39</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Definition of Vector Angle</td>
<td>39</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Angle Histogram Chart, Two Histogram Curves are Matched</td>
<td>39</td>
</tr>
<tr>
<td>Figure 31</td>
<td>Correlation Values for Angles Between -80 and 80 Degrees</td>
<td>40</td>
</tr>
<tr>
<td>Figure 32</td>
<td>Rotational Match</td>
<td>41</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Vector Angle Deviation from Line V_Spacing = 1</td>
<td>41</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Angle Histogram Chart, vectors defined as in Fig 5, V_Spacing = 1</td>
<td>41</td>
</tr>
<tr>
<td>Figure 35</td>
<td>Vector Angle Deviation from Line V_Spacing = 5</td>
<td>42</td>
</tr>
<tr>
<td>Figure 36</td>
<td>Angle Histogram Chart, vectors defined as in Fig 7, V_Spacing = 5</td>
<td>42</td>
</tr>
<tr>
<td>Figure 37</td>
<td>Orientations for D1 and D2 matching</td>
<td>42</td>
</tr>
<tr>
<td>Figure 38</td>
<td>D1 and D2 Directions: Post Match Verification</td>
<td>43</td>
</tr>
<tr>
<td>Figure 39</td>
<td>Magnified view of original wall match before and after the second match</td>
<td>43</td>
</tr>
<tr>
<td>Figure 40</td>
<td>Diagram of Vector Operations for Correct Matching</td>
<td>43</td>
</tr>
<tr>
<td>Figure 41</td>
<td>Subsequent Scans Completely Matched</td>
<td>44</td>
</tr>
</tbody>
</table>
Figure 42 - Mismatch Flow Chart ........................................................................................................45
Figure 43 – Color Scheme Example ....................................................................................................51
Figure 44 - Line through Box .............................................................................................................52
Figure 45 - False Points Algorithm ....................................................................................................53
Figure 46 - Current Algorithm ...........................................................................................................54
Figure 47 - Next Step in Algorithm ....................................................................................................55
Figure 48 - Resultant Image ...............................................................................................................56
Figure 49 - Experimental Setup .........................................................................................................57
Figure 50 - Vector Schematic ............................................................................................................58
Figure 51 - The final Match (mm) .......................................................................................................59
Figure 52 - Angle Matching vs. Scan ..................................................................................................59
Figure 53 - X Translation Matching vs. Scan .....................................................................................59
Figure 54 - Y Translation Matching vs. Scan .....................................................................................60
Figure 55 - Hokuyo URG-04LX ........................................................................................................64
Figure 56 - Screenshot of real-time URG-04LX display (using provided software developed by Hokuyo).................................................................................................................................................65
Figure 57 - Thermovision A10 Infrared Camera (Courtesy of http://www.flirthermography.com/images/camera_images/Micron220.jpg) ............................................................................................................66
Figure 58 - Part PC209IR Low Light Camera (Courtesy of http://www.supercircuits.com/redesign_images/prodImg200/pc209ir.jpg) .................................................................................................................66
Figure 59 - Sensoray Model 314 (courtesy of http://www.sensoray.com/products/314photo.htm) ........................................................................................................................................67
Figure 60 - Serial Communication Board ..........................................................................................69
Figure 61 - CO2 Level Bar ..................................................................................................................69
Figure 62 - Robot Orientation Display ...............................................................................................70
Abstract

The goal of the Good Samaritan User Interface team was to specify, purchase, design (when necessary) and implement all the controls for the Good Samaritan Robot. This includes both the Platform (with and without the arm attached) and Mini form factors of the robot. The exception to this is the motor controllers that were designed by other teams that we had the obligation to communicate to. This project also included the development of an intuitive GUI to assist the single operator in driving the robot effectively.

The hardware on the robot consists of a PC104+ stack single board computer (SBC), a 4 input multiplexing Video capture and compression board, a low light video camera, a thermal camera, a laser based radar (LADAR), a wireless device, a CF/IDE adapter, a compact flash card, 2 PC microphones, a CO2 sensor, a five axis inertial sensor, and a custom PIC converter board. See Figure 1 for a display of how components are connected.

The majority of the controls are done in software on the robot (client) side. Figure 2 shows the two portions of the system and how they interact. All software was written in C++ except for the GUI which was written in visual C++. The majority of the software development was done in the area of SLAM. Simultaneous Localization and Mapping is method to build a map without any prior knowledge of surroundings. This software relies heavily on the LADAR. The GUI used by the robot operator contains two controls modes: one for the movement of the robot and one for the movement of the arm, and three visual modes: one for each camera and one to the display the current map.
Figure 1 - Hardware Layout

Figure 2 - Software Layout
Introduction/Problem Statement

Statement of Work

Since 2001 the National Institute of Standards and Technology (NIST) has held an annual robot competition in the Urban Search and Rescue category. The competition goal is to build fully autonomous robots capable of negotiating partially collapsed buildings, identifying trapped humans, verify the physical condition of said human, and generate maps of their location. To test these robots, NIST developed arenas to simulate the conditions of a partially collapsed building. The competition is similar to as follows: at the scene of an earthquake, a partially collapsed building is in danger of secondary collapse. Consequently, the risk to human rescuers is high and a robotic alternative is preferred. The initial response in the rescue becomes that of the role of the robot and operator, which is to present a map and data sheet to the incident commander with the location, state and situation of each victim. [3]

Using robots in this dangerous situation allows for humans to be more efficient in their rescue strategies. Typically human rescuers spend roughly 97% of their time searching for victims and only 3% abstracting them. Effective robots will limit the time wasted looking for victims. Additionally, robots are preferable because of their ability to operate in minimal light, have minimal weight, and be expendable. [6]

The long term goal of the Good Samaritan project is to develop a robotic system that can successfully navigate all arenas at the competition and receive the highest point total by detecting as many victims as possible along with their condition and state of awareness, provide a generated map of the arenas and victim locations, and do this all with as few penalties as possible, within the specified time limit. The short term priorities of the Good Samaritan robot, in order of importance, are: having a drivable robot with all sensors working, creating a server-side mapping algorithm to generate a 2.5D map or the arena, constructing a GUI that prevents cognitive overload for the driver.

The Good Samaritan User Interface team will be responsible for User, Software, and Hardware systems. Also integration of these systems with the Good Samaritan Platform, Arm, and Mini teams must be completed prior to competition.

The User system will incorporate both the interaction between the person controlling the robot and the sensory feedback from the server laptop system. A Playstation 2 controller will be utilized as the sole input to control the entire robot system. The analog control sticks will be used to operate the robot and camera movements. The right stick will control the robot, and the left stick will control the camera. The D-pad will control the direction of the IR camera. Each button will be given a single function within each single mode of operation. One button will be dedicated to provide switching capability between the different operational modes. These modes will allow a much simplified interaction based on the desired situational context that will be chosen by the driver. A Graphical User Interface (GUI) will be utilized for visual feedback. The GUI will display video feeds, sensor feedback, robot orientation, and alerts as to object collisions. The GUI will display a real-time map of the local area surrounding the robot, as well as visual representations of sensor data, and any necessary alerts of which the driver must be notified.

The software system will include both networking the client and server, as well as the autonomous assistance systems. These systems include mapping, obstacle avoidance, victim detection, and assisted autonomy for controlling the robot. The obstacle avoidance system will
help determine if a passage way is safe, and whether travel through different areas are possible. Mapping will be done via Simultaneous Localization and Mapping (SLAM)

Most of the hardware is sensory based. The robot will integrate a Laser Radar (LADAR) guidance sensor, a thermal infrared (IR) camera, a Zero-Lux video camera, a microphone, a CO2 sensor, a gyroscopic/tilt sensor and a motor position feedback controller.

Testing will be conducted for each task or sub-system in order to ensure progress is being made, as well as to provide a method for testing the entire system as a whole. The main human user who is going to control the system will be a testing tool when all the systems have been integrated together. The Playstation controller will be tested by creating a simple program to detect key presses and analog directional values. The GUI will be tested for visual integrity, and functionality of each feature. Since it can not be tested very well without the other systems, especially the User system, the GUI tests will be incorporated into each system individually.

The different software sub systems will be tested as follows: the mapping system will be tested in the RamLab using a test arena, similar to the competition arena. Obstacle avoidance will be testing by placing different objects in proximity of the robot and then running it forward and see whether it will avoid the objects. Autonomous stair control will be tested by having the robot climb the stairs and measured by how straight it climbs them. The Autonomous upright system will be tested by putting the robot in various defined immobile positions, then check if the robot can upright itself with no other physical help. The center of mass system will be tested in the Moment of Inertia test fixture designed by a separate senior design team. The robot orientation system will utilize the data from the gyroscopic sensors and will be tested by reading data values as the robot is rotated in all three axes. The networking system will be tested by attempting a client server handshake, and utilizing networking commands such as the ping utility, and remotely interacting with the client operating system.

The sensors will be tested by reading in values and comparing them to values that we have calculated. The thermal camera will be tested by reading the values within the images and comparing it to a laser thermometer (or other means) to compare the readings with the physical temperature of a test object. The video camera will be tested by visually inspecting the image feed that is received on the server laptop. The LADAR system will be tested using a test arena similar to the competition arena and it will be tested to see if it gives valid data regarding the distances to various objects in the field of view. Also, it should generate a localized map that looks similar in nature to the testing arena. The CO2 sensor will be tested by verifying readings are 2000±200 ppm when a person breathes near it and when there is nothing within a close proximity. The microphone will be tested audibly by having it receive audio from a tape player or other device, and test if we can hear the sound on the speaker of the server laptop. The decibel level should be at least 50 dB in order to receive a decent reading. The gyroscopic or tilt sensor will be tested by verifying if we can receive data regarding to the rotational position from an equilibrium point within 2.5° in each of the three axes. The motor position will be verified by having the robot run a distance of 10 feet and check to verify the readings from the feedback match within 6 in. for every 10 ft. (or 95% accuracy).

Each of these systems will be integrated in multiple generations, as the ambitions of our team are high. A first generation will be simple improvement to last years controls and interface. This will include all necessary updates of the interface to incorporate the LADAR sensor, thermal camera, visual camera, microphone, and motor control and feedback. Also, SLAM will be implemented to create simple map which the driver can input victim information on it. The GUI will be
updated slightly to better alert the driver, as well as provide the necessary video feeds on display. Integration must be present between all the teams: Platform, Arm, and Mini. A second generation robot will enhance the first generation and include more functionality and autonomy. It will incorporate more advanced mapping, simple assisted autonomy, as well as increase victim detection accuracy. Each of the added systems must be tested before integrating it into a competition ready robot and must ensure that any second generation systems must failover to first generation systems in order to provide full working functionality during the competition.

A Third generation will approach greater levels of autonomy, and will allow the driver to control the robot with ease through much simplified controls. This will include a “novice” mode enabling inexperienced users to control the robot safely. This will be accomplished using multiple sensors together to detect obstacles, victims, and landmarks. Possibly SLAM will be extended to the 3rd dimension. The mapping system will utilize landmarks detected by other sensors to create maps that are more accurate. Visual feedback will be enhanced through a second generation GUI. This will include 3D models, as well as functional alerts presented in an informative but unobtrusive manner.

The team hopes to win the competition by adhering to this document, and completing the required tasks that have been setup within the milestone section, quickly making necessary decisions, and communicating fully with the other three teams under working together on the Good Samaritan project, while having fun every step of the way.

Constraints
1. Objective
   1.1. Audio, Video, Infrared, and CO2
      1.1.1. Requirements
         1.1.1.1. Victims must be identified using three or more sensory techniques, after which the victim’s state can be assessed. [3]
         1.1.1.2. The thermal (IR) camera that will be used in competition must be able to detect a heat signature of 95° ± 15° Fahrenheit [3]
         1.1.1.3. The sound detecting system on board the robot must allow the user to identify sounds between 55 and 100 dB, apart from the loud competition environment. [3]
         1.1.1.4. The user should be able to identify the situation/body position of the victim after victim state is identified. [10]
            1.1.1.4.1. Surface: victim on open unobstructed area.
            1.1.1.4.2. Trapped: victim is partially trapped or seized by structure.
            1.1.1.4.3. Void: some of the victim is out of sight.
            1.1.1.4.4. Entombed: the entire victim is surrounded and out of sight
         1.1.1.5. Be able to detect CO2 levels of 1500 ppm or higher. [3]
         1.1.1.6. The CO2 sensor must not falsely identify victim.
      1.1.2. Goals
         1.1.2.1. User should be able to detect the state of the victim [10]
            1.1.2.1.1. Unknown – state can not be determined.
            1.1.2.1.2. Unconscious – detect sound of breathing, video of body, IR of body heat.
1.1.2.1.3. Semi-Conscious – detect some small movements, or unidentifiable sounds (like a groan).
1.1.2.1.4. Aware – detect movement of arms or yelling.

1.2. **Victim Detection**
1.2.1. **Requirements**
1.2.1.1. System power supply must drive robot operations for at least 25 minutes as per mission length and any necessary setup time. [3]
1.2.1.2. The User Interface should draw no more than 30 Watts, so that the G.S. can remain operational for 25 minutes.
1.2.1.3. Victims must be identified in a clear line of sight, without any obstruction including netting, meshing, or walls. [3]
1.2.1.4. Environmental elements, such as bricks, ramps, pallets, leaning debris, movable debris or any victims must not be moved, especially during any autonomous control.

1.2.2. **Goals**
1.2.2.1. The User Interface should draw no more than 25 Watts, so the G.S. can remain operational for 25 minutes.

1.3. **Mapping**
1.3.1. **Requirements**
1.3.1.1. Submit generated map(s) to judges within 2 minutes after the mission has been completed [3]
1.3.1.2. Create map that is partially generated by LADAR, but has victim data and other features added by the driver.

1.3.2. **Goals**
1.3.2.1. Mapping and victim placement within the virtual map will be fully automated.
1.3.2.2. Flag every victim within mapping interface with I.D. number, state (aware, semi-conscious, or unconscious), and simultaneously fill out victim data sheet with details of the victims that would be cumbersome to fit on flags.

1.4. **Graphical User Interface**
1.4.1. **Requirements**
1.4.1.1. The wireless data must transmit through the 802.11a standard. [3]

1.5. **Stair Climbing**
1.5.1. **Requirements**
1.5.1.1. The robot must climb 3 stairs ( h = 8in , w = 8in) to complete the yellow arena [3]
1.5.1.2. The G.S. will have precise and quick response torque control for the motors so that the platform does not continue off a ledge after climbing the stairs. [1]

2. **Subjective**
2.1. Audio, Video, and Infrared
   2.1.1. Requirements
   2.1.1.1. Video and IR feed should be such that the user can identify any human movement or human form that is not moving. [10]
   2.1.1.2. Video feed should allow user to identify reflective clothing, worn by simulated victims as well as I.D. cards worn by the stimulants. [10]

Criteria

3. Objective
   3.1. Audio, Video, Infrared, and CO2
      3.1.1. Requirements
         3.1.1.1. A unidirectional microphone will be mounted parallel and in the same direction to the camera on the arm, providing focused sound detection within the area of sight.
         3.1.1.2. An omni-directional microphone will be mounted on both sides of the G.S. providing stereo sound to the user through headphones.
         3.1.1.3. The IR camera and video camera will no longer toggle back and forth but rather, video will continuously feed though to the user and when on board logic detects heat between 80° and 110° Fahrenheit, the IR signal will be transmitted to the user.
         3.1.1.4. The A10 thermal camera will operate passively at 2 Hz until heat is detected, at which time the frame rate will increase to 8 Hz and the signal will begin to be wirelessly transmitted to the user.
         3.1.1.5. The video camera will operate continuously at approximately 30 fps and the un-encoded signal will be transmitted or possibly hardware encoded, to reduce latency.
         3.1.1.6. CO2 detector will have a delay of 30 seconds or less.
   3.1.2. Goals
      3.1.2.1. During blackout period, IR sensor will still mark and note hotspots.
      3.1.2.2. Use audio to help locate victims.
      3.1.2.3. CO2 detector will have a delay of 20 seconds or less.

3.2. Victim Detection
   3.2.1. Requirements
      3.2.1.1. Robot must be developed for no more than $1500, plus private donations, as dictated by the ME486 Management Team. [1]
      3.2.1.2. Obstacle avoidance such as LADAR, IR sensing, and visual feed must be implemented and very operational to avoid losing points by impacting debris in the competition area.
      3.2.1.3. The robot client system must be capable of storing 2 minutes worth of necessary data to conduct autonomous mapping, victim detection, and obstacle avoidance.
      3.2.1.4. System should be completely integrated with the rest of the GS robot and ready for testing by January 6, 2007. [1]
      3.2.1.5. Robot should be able to autonomously upright itself in any situation, or with minimal user effort.
3.2.1.6. System will be capable of reporting the center of mass to the operator.
3.2.1.7. The system will be capable of reporting the orientation of the robot at all times (possibly through a 3d model within the GUI).
3.2.1.8. CO2 sensing will be conducted using the IR camera.
3.2.2. Goals
3.2.2.1. Detection of victims with all five sensory inputs.

3.3. Mapping
3.3.1. Requirements
3.3.1.1. Navigate using assisted autonomy around the competition area such as obstacle avoidance, stair climbing mode, and IR heat detection.
3.3.1.2. Assist user in detecting victims at a distance of at least 2 feet.
3.3.1.3. Detect victim’s temperature within 15 degrees Fahrenheit.
3.3.1.4. Utilize 2D SLAM for real-time mapping, with LADAR, and place identifiers within the map that details victim’s position as well as state of awareness.
3.3.1.5. Determine relative position from starting point within 3 feet.
3.3.1.6. Generated map must contain basic features of the arena including walls, doorways, as well as victim, hazards and obstacle information.
3.3.1.7. A LADAR device will be used within the navigation and mapping sub systems.
3.3.1.8. The LADAR range finder will no be exposed to an impact load. It will be protected fully, but will not lose any capabilities.
3.3.1.9. The User must be able to control the robot so that it can articulate itself without getting stuck through the yellow arena, in order to reach the orange arena.

3.3.2. Goals
3.3.2.1. Detect victims at a distance of 1 ft.
3.3.2.2. Utilize 3D SLAM for real-time mapping.
3.3.2.3. Determine relative position from starting point within a foot.
3.3.2.4. Detect victim’s temperature within 1 °C.
3.3.2.5. Determine relative position from starting point within 4 feet.
3.3.2.6. During blackout, mapping will continue under a predefined or autonomous control.
3.3.2.7. The User must be able to control the robot so that it can articulate itself without getting stuck through the orange arena, in order to reach the red arena.

3.4. Graphical User Interface
3.4.1. Requirements
3.4.1.1. Video display will be presented at 2x the real resolution in order to optimize the operator’s visibility.
3.4.1.2. All controls will be intuitive and have a first touch to proficient learning curve time of 3 hours.
3.4.1.3. Inputs from the LADAR, video, sound, position, orientation, camera direction, and battery will be on the screen concurrently.
3.4.1.4. IR display will be dynamically controlled by either the user or the on the software.
3.4.1.5. Video streams (Video and IR) will be overlaid at same camera angle to simplify picture.
3.4.1.6. The User Interface will run on a Dell Inspiron 8600, running Windows XP professional SP2.
3.4.1.7. A full cycle latency time of less than 0.5 sec.
3.4.1.8. User interface must interface with both standard and mini GS platform.

3.4.2. **Goals**
3.4.2.1. Real time mapping on touch screen enabling the user to add features and victim locations.
3.4.2.2. A full cycle latency time of less than 0.2 sec.

3.5. **Stair Climbing**

3.5.1. **Requirements**

3.5.1.1. The user interface will display the direction of positive gradient on all surfaces and automatically align itself to within 10 degrees of that vector.
3.5.1.2. User Interface will autonomously maneuver robot up stairs to decrease user responsibilities.
3.5.1.3. The user will have the ability to toggle into and out of this automatic stair climbing mode.
3.5.1.4. Displays caster and camber while on the stairs.
3.5.1.5. Automatically recognize and adjust for reaching top of stairs.

3.5.2. **Goals**

3.5.2.1. Gives warning to user when robot center of mass is within 10 deg from rollover point.

3.6. **Obstacle Avoidance**

3.6.1. **Requirements**

3.6.1.1. Objects that are closer than 3 ft are highlighted in GUI.
3.6.1.2. Identifies vertical drops which are too large to drop off.
3.6.1.3. Accentuates in display “trouble” spots where the robot has already been, defined by the amount the robot spun out.
3.6.1.4. Accentuates in display places where the robot rolled over or came close to rolling over.
3.6.1.5. Displays a red light when camera arm is in extreme positions that will cause robot to roll.
3.6.1.6. Maintain a background process to track of the center of mass of the robot.

3.6.2. **Goals**

3.6.2.1. Identifies inclines which are steeper than the rollover angle of the robot.
3.6.2.2. Identifies vertical steps which are too large to climb.
3.6.2.3. At 2 inches within obstacles, signal will be sent to the rumble pack on the PS2 controller to inform the user that an obstacle is ahead.
3.7. **Miscellaneous**

3.7.1. **Requirements**

3.7.1.1. An overall system flow chart for hardware and connections will be created with detailed notes for future students and workers and as a helpful troubleshooting tool.

3.7.1.2. An overall programming flow chart will be created as a helpful refresher, learning tool for future students, and an essential debugging tool.

3.7.1.3. The flow chart will include program file names and purposes.

3.7.1.4. In depth commentary will describe exactly what the code means, at least one line per line of code.

3.7.1.5. Once the wireless communication is activated the robot must be fully operational within 90 seconds.

3.7.1.6. A complete electrical schematic will be drawn up for troubleshooting, and debugging.

3.7.1.7. Meet our budget goal of $2000 including donations.

3.7.2. **Goals**

3.7.2.1. Meet our budget goal of $0. [1]

3.7.2.2. Once the wireless communication is activated the robot must be fully operational within 30 seconds.

4. **Subjective**

4.1. **Audio, Video, and Infrared**

4.1.1. **Requirements**

4.1.1.1. The robot must be fully operational in dark areas. [10]

4.1.1.2. Audio, Video, and IR sensing will be conveyed in an as realistic manner as possible. Enabling the user’s advanced logic and thinking power to be used in order to digest sensory information.

4.2. **Victim Detection**

4.2.1. **Requirements**

4.2.1.1. Robot must detect victims before running into them.

4.3. **Mapping Criterion**

4.3.1. **Requirements**

4.3.1.1. 2D SLAM will be interactive and real-time in order to give the user visual feedback as the robot is changing its position.

4.4. **Graphical User Interface**

4.4.1. **Requirements**

4.4.1.1. The monitor space will be managed carefully and windows will be methodically placed for an uncluttered and simplified GUI.

4.5. **Stair Climbing**

4.5.1. **Requirements**
4.5.1.1. Robot must stop movement prior to rolling over or falling off stairs.

4.6. **Obstacle Avoidance**

4.6.1. **Requirements**

4.6.1.1. Accentuates possible victims in display.
4.6.1.2. Accentuates starting point in display.
4.6.1.3. Accentuates walls in display.


**Preliminary Design**

**UI**

The interaction between the user and the robot involves both sensory feedback as well as control of the various actuators and motors. The graphical user interface is what handles both of these interactions by representing sensory information as graphical or text based elements, and by taking various input devices such as a game controller or a keyboard or a mouse and allowing them to provide the control for interactions between the user and the robot.

**GUI**

The Graphical User Interface (GUI) was designed to give the user controlling the robot only the vital information necessary to find victims and navigate the different arenas. The previous design showed all information on the screen at the same time, so our first design concept was very simple in terms of interface, shown in Figure 3. This first design was meant to give the user only the information they need in order to control the robot.

![Figure 3 - First GUI Design Concept](image-url)
The final design concept was based off the simplistic design of the first concept, but involved more realistic information and was made to look as close to what the final application would display. As can be seen in Figure 4 the final concept was designed in a way to allow all the information that the robot can sense and compute to be available on request, but not to get in the way of the vital information and create a cluttered interface that would be hard to use. This is a type of “context-sensitive” interaction where the display changes based on what the user is currently controlling. Three display modes are used to display feedback from the cameras as well as the map that is being generated from each LADAR scan. A visual camera display mode, a thermal camera display mode, as well as displaying the current status of the create map(s).

Some of the features that were planned to be implemented were more graphical articulation and orientation of the robot, a three-dimensional representation of the robot’s arm, the current scan from the LADAR, CO2 level, as well as time left in the competition run.

**User Input**

The user controlling the robot uses a game pad that enables four full analog axes of movement, a point of view control, and up to 10 active buttons as shown in Figure 5. The user also has the ability to utilize both the mouse and keyboard attached to the system. These can be used to manipulate the user interface in greater detail. However, the game pad provides all the necessary functionality that the user needs in order to fully operate the robot. There are a few intuitive features that are implemented within the GUI and software in order to make the Operator have much greater control over the robot. These include having different modes of operation, presenting alerts and notifications to the user, and displaying current relative information in a
context-sensitive manner. The different modes of operation allow the user controlling the robot to have a single control set whose functions can vary with mode. Extra features such as viewing the current state of the map will be accessed through either separate modes of operation and/or through the use of the extra input devices such as a keyboard and mouse. The goal behind this design of control is that the user has only the necessary functionality present at any given moment.

![Game pad Controls Concept](image)

**Figure 5 - Game pad Controls Concept**

**Linksys WRT55AG**

Our router that sits on the server (operator) side was donated prior to our arrival, thusly not much as included in its selection. See the detail design section for full details.

**Mapping (SLAM)**

A major constraint this year is the addition of a mapping system. Competition judges have included a considerable multiplier for mapping ability. With the ability to map, last years competition team would have won the world competition. Because winning is the overall constraint, adding mapping capabilities is a close second. A map can be generated using several different methods, GPS, Radar triangulation, video localization, inertial sensors (gyroscopes, accelerometer), and SLAM. For the USAR application, SLAM was a best fit because it functions
without outside contact of any kind. One constraint of the competition is that no prior human setup is allowed which eliminates the triangulation approach, and GPS is also disallowed.

**Dimensions (2D, 2.5D, 3D, 6D)**

The conceptualization of two a two dimensional world is simple because we, as humans living in a three dimensional world, see in two dimensions, therefore it is intuitive for humans to work write programs and do math in two dimensions. A two dimensional world can be considered as a flat plane. Putting this in terms of SLAM, a two dimensional slam problem is one in which the world is assumed to have no third dimension. One example of this is a local map of the earth. Although the earth is round, the diameter of the earth is so large that minimal errors will come from thinking of it as flat on a local scale. This means the world is approximately flat and any “un-flatness” induces very little error in calculations. Two dimensional SLAM is the easiest of the four possible considerations for SLAM. For a solid object in a plane, there are a total of three degrees of freedom. One is rotational and the others are translational.

2.5 D information is tricky because it is difficult to conceptualize a space which has less than three dimensions and more than two dimensions. 2.5 D is used when calculations cannot be made to consider all possible degrees of freedom of three dimensions, while carefully selected assumptions allow for acceptably accurate results in the third dimension. 2.5 dimensions maintains calculation requirements close to that of the two dimensional model, while giving information pertaining to the third direction. The 2.5 dimensional world can be considered a surface in three dimensions and is not necessarily flat. The 2.5 dimensional data is a two dimensional data set, each point in the plane possessing information pertaining to the third dimension (elevation). In 2.5 D SLAM, there can be either four or five degrees of freedom. A 2.5 D world can be related to a topographical map. The map is a two dimensional object, but it also contains contour lines to display the elevation of the map at each position.

3 D slam is the same as 6 D slam. The reasoning for the difference in naming convention is unknown. Both naming conventions deal with six degrees of freedom and both possess a three dimensional dataset to structure its information. This is by far the most computationally and conceptually complex of the considerations. This would be considered the “big daddy” of SLAM algorithms. The three dimensional relationship to the map in the 2 D case, and the topographical map in the 2.5 D case is a globe of the earth where, a three dimensional matrix represented a cube encompassing the earth and any point inside the sphere of the earth’s surface is occupied with dirt and everything else isn’t occupied with dirt.

---

*Figure 6 - Mapping Types*
**Discrete Belief**

In Slam, there are two ways of approaching mathematical solutions; the first is Discrete belief and the second is Continuous belief. Although both use the same basic statistical theories, the two vary in the implementation of algorithms. Both have strengths and weaknesses. If large datasets with small accuracy is required, Discrete belief is computationally, less expensive. If precision is the goal, Continuous belief is generally a better solution. With Discrete belief, a search is used to determine a most optimal solution, while with Continuous belief, an iterative approach is implemented in which a solution is formed, then, using that solution, another, hopefully more accurate solution is calculated.

When using Discrete belief, the range of data is broken into a number of regions. I will use 2.5 D SLAM in the following example. In 2.5D SLAM, an elevation corresponds to each position in the given region as shown in Figure 7. I will simplify the example by shrinking the problem into only nine discretizations as shown in Figure 7.

![Figure 7 - Blocks](image)

Each region has a characteristic point that represents the entire region as shown in Figure 8.

![Figure 8 - Characteristic Points](image)
In the case of Figure 8 and the 2.5D SLAM case, each point would possess an elevation. When the elevation of the point is calculated to be a certain value, the entire region would be the same elevation. In Figure 9, the elevation is represented as a blue scale. As you can imagine, it is pretty simple to calculate the elevation of the nine squares in this example, but as the level of precision and the number of regions goes up, the required memory goes up very quickly. In the mapping portion of the SLAM algorithm used in this years’ project, discrete belief was used.

![Elevation Blocks](image)

**Figure 9 - Elevation Blocks**

**Continuous Belief**

In the case of Continuous belief, the region is calculated as if it were continuous. The 2.5 D SLAM example is used as an example to describe Continuous belief. When calculating the elevation for the example, Elevation would be calculated as a function of x and y. Figure 10 displays a blue scale representation of a Continuous elevation where elevation=$E(x,y)=x+y$. As you can see between the two pictures, Continuous belief has the potential to be considerably more accurate without the very high memory usage in the discrete belief example.

In the two examples of Discrete belief and Continuous belief, the elevation was calculated as a function of x and y. This was just an example and we did not do any elevation calculations on our project this year. Instead of calculating elevation, we calculated the probability that the separate regions were occupied.
**Robot Controller**

The GS robot requires some form of logic in order to receive commands from the user, read data from its sensors, and to control its actuators. The robot controller discussed in this section provides this functionality.

Two common types of boards that are suitable for robotic controllers are the PC/104 and the PC/104+ form factor boards. These boards are a small form factor, roughly 3.6 x 3.8 in. They are both self stacking. This means that a collection of these form factor boards can be stacked into a nice, compact package, with out the need for backplanes or card cages.
Figure 11 The hardware difference between ISA (PC104) and PCI (PCI104). PC104+ Combines both bus systems.

The difference between the two types of boards is the type of bus connections which they have (see Figure 11). The PC/104 boards have only the 104-pin ISA bus. This bus consists of 4 rows of two different lengths. The PC/104+ adds a 120-pin PCI bus. The rows of pins on this bus are all the same length. PCI is a very common bus for video communications. There is also a third variation on the PC/104 idea named PCI-104. These boards only have PCI busses and no ISA bus. From our research they are by far the least common of the three designs. Check PC104.org for specifics on these standards.

KoreBot

The KoreBot is an XScale PXA255 board which can be used as a robot controller. We began the year with a free board that was donated to us by Road Narrows Robotics. Initially we thought that the board was an excellent solution, but as we looked into incorporating it with our wireless and video constraints we started to run into problems.

The first problem that we encountered was it’s two compact flash slots. They are only compact flash type 1. Type 1 slots are slower and out dated they use a 16 bit bus. The compact flash specification is now on revision 4.1[1]. 802.11a cards require a faster 32 bit bus speed than type 1 CF can provide. Our research didn’t turn up any 802.11a cards which were crippled to run at the lower speed and bus width.

The next major problem that we encountered with the KoreBot was its USB ports. Again just as in the case with the CF slots the USB ports are too slow. USB 1.1 only runs at a maximum of 12Mbps. This is slower than the theoretical 54Mbps for 802.11a and is also too slow to effectively capture two video streams.

**Industrial PC W-LX**

The next solution that we chose was Industrial PC’s W-LX. The W-LX is based on the AMD Geode LX-800 processor, a low power x86 compatible CPU. One nice feature of the W-LX is its integrated CF II slot for use as a hard drive. Our final solution the CPU-1433 required a simple adapter, which is not as elegant.

We were given a quote from Industrial PC for the W-LX of $313. This price didn’t include RAM. The W-LX can hold up to 1 GB. We were offered a 512 MB SO-DIMM for $124.

The W-LX is a very capable solution. It has USB 2.0 ports (x4), and RS-232 (x2). If a suitable USB video capture solution can be found this board would be a good controller for a robotics platform. It is possible to find ISA video capture boards, but they are rare and expensive.

At the time of board selection we believed that a PCI based video capture solution would be the best solution since it’s the fastest of the three options (USB, ISA, PCI). This board, not having a PCI bus (PC/104+), was ruled out.

**CPU-1433**

The final solution that we picked for our robot controller was the CPU-1433 made by Eurotech and sold by Parvus Corporation in the U.S. The CPU-1433 is powered by the AMD Geode GX466 @ 333MHz. It comes with 128 MB of RAM pre-soldered for shock and vibration resilience. This board will be further discussed in the design details section.

**Wireless**

http://ndiswrapper.sourceforge.net/joomla/  

As of 2006 the Robocup world competitions for the USAR category, wireless communications are required to be over the 802.11a protocol. Last year’s team (2005-2006) made a quick change before competition where they added a Netgear router to the robot. Our team was given the constraint to remove the router while still using the 802.11a protocol.

There are three ways that we found to communicate without a router. The first is via a hardware Ethernet bridge. This bridge would convert regular wired Ethernet signal into a wireless signal. These typically consume a lot of power and are often found in the form of a router. As power limitation was a large concern, this option was not pursued very far.

The second option for wireless communications is PCMCIA 5.0 or later. This is also known as CardBus. CardBus is effectively a 32-bit 33MHz PCI bus. This is the type of expansion slot found in most, if not all, modern laptops. There are many 802.11a cards available in this form which are supported in any modern computer operating system. Most PC/104+ boards do not come with these slots, but there are many PC/104+ stack boards that can add this functionality. Our team chose not to use this option due to the reasonably high cost of the adapter board, and the increase in the size of our board stack. It is also difficult to say if 802.11a is supported any better in this form than in the form that we chose. Cardbus cards often have a more limited range then USB adapters.
The third option for wireless, the one that we chose, was a USB 2.0 dongle. Initially location Linux compatible 802.11a wireless dongles was difficult. There are not many on the market.

One common wireless solution used in the Linux world is the NdisWrapper. Ndiswrapper is an open source API which ‘wraps’ around windows wireless device drivers in order to make them usable in linux. Many wireless device suppliers are hesitant to release opensource drivers (or even binaries) because of the potential opportunity to break FCC regulations. The NdisWrapper project appears to be highly active and supported. This is the option which we chose to work with. More details can be found in the design details section.

Another USB dongle solution is the ZyXEL AG-225H. We didn’t choose this solution due to our uncertainty about it. It claims to work well, but it doesn’t seem to have the same community support as the NdisWrapper project. Also, it’s built in hotspot finder with LCD seemed like it could cause problems, or at the very least provided functionality we didn’t need.

**Sensors**

**LADAR**


LADAR stands for LAser Detection and RAnging, this sensors was donated to the robotics project at the beginning of the year for the purpose of developing SLAM (Simultaneous Localization and Mapping) technology on the GS robot. This sensor acquires 2-D scatter plots of its surroundings. More details will be discussed in the detailed design section. For quick access to this heading please select the following link (**LADAR**).

**Video**

**Thermal Camera**

The selection of the Thermal Camera was simple. It was donated. It is extremely expensive. Our choice was made for us. We chose the Thermovision A10 Infrared Camera. See the detail design section for more information on the camera.

**Video Camera**

Many options were considered for the video solution on the Good Samaritan Robot. These can be categorized into three groups. IP cameras, traditional web cameras, and analog cameras

**IP Cameras**

These are traditionally used as surveillance cameras and often include http servers. However, very few of these function at 30 frames per second(fps) with the necessary resolution. Also they are not very versatile; you are bound to the proprietary code to access the images. This would
present a problem for the single board computer. It would have to either have a program that fetched the video and sent it to the server, or act a router and enable the server access to the camera directly. The latter option required a lot of Linux networking, of which we had little experience.

**Web Cameras**

This option was our second choice. While webcams form factor is small, they are not very sturdy. Next they typically have poor image quality. Finally, the issue of getting it to work in Linux was a potential problem. After researching and reading many forums discussing webcams’ support in Linux. We concluded that it was not a reliable solution. Webcams also limit the potential video compression. Typically the use uncompressed video or MJPEG compression. Both options were concerns for our limited bandwidth. A webcam with tested Linux drivers and MJPEG compression could be used as a last resort.

**Analog / NTSC Cameras**

This was the first choice for a couple of reasons. First, we already had analog cameras that were rated to .000 Lux. Also the additional criteria were handed down that if a camera was replaced, it must incorporate an optical zoom. However, this option necessitated the inclusion of frame grabbing hardware to digitize and compress the NTSC signal. These implications will be explored in the next section.

**Video Compression Board**

**USB frame grabber**

Preliminary tests with some USB video input devices showed a good deal of latency. Also, most usb devices merely sample the NTSC signal and rely on software compression. With our limited computing power, this was not an option.

**PC104+ frame grabber**

Price was the original deterrent for a stacked frame grabbing board. However, the versatility, Linux support, hardware compression, and low latency of these devices were preferred. Many of these devices were able to handle multiple inputs simultaneously, but a compromise was made with price to switch between video streams in software as was needed. Most frame grabbers are PC104+ and thus the board had to be capable of a PCI bus. A few different PC104+ boards were investigated and the final solution was chosen due largely to price. Many boards were able to handle multiple streams simultaneously, but were upwards of $700. Also Linux support was questionable on many options. The final choice was a Sensoray Model 314. (See the Appendix for full documentation). Support at Sensoray has been first rate for a small company. We have been able to get in contact with the original design engineers.
Audio
The user will receive stereo audio of the sound at the robot. Two small microphones (Figure 15) will be mounted on either side of the robot and will notify the user of any noises in the arena. These microphones will be connected to the Sensoray Model 314 video compression board and will be encoded along with the video.

![Figure 15 - PC Microphone](image)

CO₂
After creating a selection matrix, it was determined that the best way to add CO₂ sensing capabilities to the Good Samaritan robot was to build a sub-system in-house. This was chosen because of the lack of precision needed for high concentration detection. Most sub-systems available for purchase are very precise and therefore very expensive. Our requirements were to measure concentrations from 0 to 3,000 parts per million (ppm). Kenneth Darbonne, Electrical Engineer, was responsible for complete design of this sub-system. Using the MG811 transducer (Figure 17) we designed an amplification circuit followed by a small microcontroller (Figure 16). The microcontroller then sends out the data serially via RS232 to notify the robot’s SBC of the concentration of CO₂. An amplification circuit was needed to amplify the .03v signal from the MG811. This op-amp based circuit amplifies the signal by 100 to about 3v, this is shown in Figure 22 and Figure 23. This 3v will drop slightly in the presence of a high concentration of CO₂.

The CO₂ sensor’s data will be sampled and serialized along with the data from the IMU using a small Microchip microcontroller. This particular microcontroller was chosen because proven reliability in previous projects and the exact amount of analog-to-digital channels needed for this application. When this project began we were expecting to just set a digital output pin “high” when the concentration of carbon-dioxide was at a high level. As the design evolved it was discovered that an IMU would be needed and, to more accurately convey the data to the user, serialization was chosen to communicate the data.

We will be using a PIC16F876A microcontroller shown in Figure 16. The CO₂ sensing circuit will explained more in detail later. This is another means of determining a victim’s status. The second sensor connected to this microcontroller is an Inertial Measurement Unit (IMU). This IMU will help determine when LADAR value is valid (i.e. LADAR scanning horizontal) and will assist in Simultaneous Localization And Mapping (SLAM) by estimating the rotation of the robot.
A small break-out board for the transducer was needed. The free software PCB123 was used to make a layout (Figure 18) to be etched for this break-out board. The same software was used to create a Printed Circuit Board (PCB) for this entire serialization circuit sub-system (Figure 19).
Figure 19 - Serialization Circuit Board

Figure 20 - Sparkfun PIC-PG2 Programmer
The C programming code for the PIC microcontroller was written in the text editor SciTE, compiled for the PIC in CC5X, and written to the PIC using the Sparkfun PIC-PG2 serial port programmer (Figure 20). The flowchart for the PIC’s code is illustrated in Figure 21. This code simply does the analog to digital conversion on the voltage, sends a character representing the voltage over the RS232 line, then repeats on the next A/D channel. Some functions take care of the redundant part of the code such as commanding the A/D and sending a character. A diagnostic two-color LED is used to show the microcontroller is in operation from afar.
Inertial Measurement Sensor
The robot will need to have knowledge of its position to the horizontal. This will help determine if the LADAR data is valid (i.e. not scanning the ceiling), and will suggest a net rotation, either clockwise, or counter-clockwise. A suggested movement will ease the calculation done by the user’s computer when creating a map, as the LADAR data will be transmitted directly back to the user and use the user’s more powerful computer to do the SLAM calculations. After making a selection matrix it was chosen that because of cost and lack of precision needed we would use the Sparkfun Sense-5DOF (Figure 24). This Inertial Measurement Unit (IMU) consists of 3-
degrees of freedom accelerometer, Analog Devices ADXL330, and 2-degrees of freedom gyro meter, InvenSense IDG300, for two of the three: pitch, roll, and yaw, depending on how the sensor is oriented on the robot. Only the yaw-rate will be utilized in this design. Both of these components datasheets are available in the appendix. This sensor will produce a voltage in proportion to the force in the various directions when supplied with 3.3 volts. The inertial data will be used with a combination of if and else-if statements in C++.NET code, written using Visual Studio 2005 .NET, to display the most appropriate image of the robots orientation.

Figure 24 - Sparkfun Sense-5DOF Inertial Measurement Unit (courtesy of sparkfun electronics)
Design Details

**UI**
The User Interface incorporates the Graphical User Interface, or GUI, as well as the user input.

**GUI**
The Graphical User Interface, as seen in Figure 25 has been developed using standard C++ and managed C++, the Microsoft .Net Framework™, and both Microsoft Visual Studio™ 2005 as well as Microsoft Visual C++ Express™. It utilizes DirectX 9.0, more specifically DirectInput™ to provide game pad and video streaming capabilities. The graphical user interface is run as a separate program and communicates through with the various other subsystems such as mapping, arm kinematics through inter-process communication, as well as network communication for sending and receiving data from any given robot.

![Figure 25 - GUI on Startup](image)

**User Input**
The user input is controlled from the game pad, keyboard, and mouse. The game pad has two modes of control, drive and arm. In drive mode the left joystick is used for control of the left drive track, and the right joystick is used for control of the right drive track. In arm mode the left joystick is used to control the horizontal planar motion of the arm (parallel to the ground), while the right joystick is used to control the vertical movement of the arm (perpendicular to the ground). The point of view, or directional pad, controls the pan and tilt of the camera mount. Buttons 1-4 control set positions of articulation. Buttons 7 and 8 are currently not being utilized.
The buttons 5 and 6 control the articulation of the robot. Button 9 controls the Control Mode, and button 10 controls the Display mode.

**Linksys WRT55AG**

This is a commercial quality Router. It can handle 802.11A, B, and G protocols. It is being used with only the 802.11A network turned on. The SSID of this network is “invisibleA”, but this can be changed. In order to change any settings for this router, point a browser to 192.168.1.1 and use the password “admin” to login. Leave the username blank. For more documentation on use, contact the Linksys website. No wireless encryption is being used. It seemed that this would cause loathsome overhead, but this was not tested. The router features one internet Ethernet port and four 100 Mbps network ports. The server computer should be plugged into the one of these four ports. The server is set up using static IP with the IP address of 192.168.1.100, but this is subject to change.

**Mapping (SLAM)**

Because the Good Samaritan is designed primarily by undergraduate mechanical engineering students, a suitable SLAM algorithm must be easy to code, computationally frugal, able to deal with highly unstructured environments, and development time must be less than one academic year.

**Histogram Scan Matching**

Histogram matching is a program that matches consecutive scans to determine any change in robot pose. During development it was discovered that the program would need to be modified to more specifically cater to the USAR application. Inherent noise in the LADAR data means that pre-filtering needed to be incorporated as well as other steps for fidelity. The modified histogram matching algorithm manipulates the data into histograms to correlate and solve one degree of freedom per histogram.

**Purpose**

The purpose of scan matching is to determine change in robot pose between distinct scans, and then integrate the changes over the entire run to maintain knowledge of robot location, while developing a map of the surroundings.

**General Outline**

The histogram algorithm uses frequency histograms that are generated from LADAR data to track three degrees of freedom within a 2-D plane. Histograms are generated to determine displacement between scans in the $\Theta$, $x$, and $y$ directions. Graphical representations of how data can be manipulated are shown in Figure 26: on the left is a Cartesian chart of the surroundings and on the right is an angle histogram that is derived from the same data.
Figure 26 - Comparison of Raw Data (mm) and Angle Histogram

Note that there are five walls in the Cartesian display and five peaks in the histogram chart; this is because each wall is linear and all the point to point vectors along the wall have the same angle relative to the robot reference frame. The concept of generating an angle histogram will be explained in detail later. The same general procedure is followed for x and y translational histograms with several additional steps.

Figure 27 - Two Consecutive Scans, Angle Histogram Comparison

Matching is achieved by shifting the histograms for two consecutive scans to maximize a correlation value and is shown in Figure 27. The angle change between the two scans is measured to be -22 degrees. After $\Delta\theta$, $\Delta x$, and $\Delta y$ are calculated, a vector defining the robot movement is known. The individual scans are pasted into a global reference frame along with robot trace to form a useful map.

Creating Histograms

LADAR data consists of a 1-D string of distance measurements, each paired to an angle at which the measurement was taken. The data is in polar form so each point has the parameter shown in Figure 28:
Equation 1 - Point Definition

\[ p_i = \begin{pmatrix} r_i \\ \theta_i \end{pmatrix} \]

Figure 28 - LADAR Data Point Definition

To begin fashioning an angle histogram, a vector is drawn from one point to the next for every point in the scan, and a total of n vectors are created. Each vector angle is calculated relative to the robot reference frame as shown in Figure 29. The dark line represents a wall that the laser scanner is sensing and the small points represent actual distance measurements made by the LADAR. There is inherent noise associated with LADAR data that must be accounted for.

Figure 29 - Definition of Vector Angle

Traditionally in angle histogram matching all of the angle measurements between consecutive points are placed in an array, and a histogram chart is created. However, due to a lack of precision, this method was altered as seen below in V-Space Modification.

Matching Angle Histograms

To determine change in angular pose, one must compare subsequent histograms. To determine angular displacement the histograms are shifted until they overlap (Figure 30), the shift distance equals change in robot angular pose from one scan to the next. Due to uncertainty in LADAR data, a perfect match will never occur.

Figure 30 - Angle Histogram Chart, Two Histogram Curves are Matched
A correlation function is implemented to judge how well two histogram profiles are matched, the correlation coefficient \( r \) is found using Equation 2.

\[
Equation 2 - Correlation Coefficient
\]

\[
 r = \frac{\sum_{i=1}^{n} [(f_1 - f_{avg})(f_2 - f_{avg})]}{\sqrt{\sum_{i=1}^{n} (f_1 - f_{avg})^2 \sum_{i=1}^{n} (f_2 - f_{avg})^2}}
\]

Where “\( f \)” represents the height of an arbitrary histogram bar (number of occurrences of arbitrary angle “\( i \)”). For example the second red spike of the histogram chart above specifically says: there are 54 point to point vectors with an angle 120 <= \( \theta \) < 121. The program does not have any prior knowledge of the robot’s movement, without this knowledge the program is unaware whether the robot rotated 10 deg, or -87 deg. Therefore this function is applied to every angle between -80 and 80 degrees, as it is assumed that the robot is not capable of rotating more than 80 degrees in either direction between consecutive LADAR scans (LADAR scans taken at 10 Hz).

![Histogram Correlation](image)

**Figure 31 - Correlation Values for Angles Between -80 and 80 Degrees**

The maximum correlation value represents the best fit. In this case the robot is calculated to have rotated \( \sim -22 \) degrees (Figure 32 - Rotational Match). Applied consecutively, this process yields the current robot angle relative to the start point. Fig 11 shows the two scans before and after the rotational matching step.
Modified Algorithm

Modifications to existing algorithms allowed the scan-match technique to work optimally in the expected surroundings for the GS search and rescue robot.

*V-Space Modification*

If vectors are connected between consecutive points as illustrated in Figure 33, every vector angle will vary substantially reducing histogram effectiveness.

If vectors skip points, the results are improved; notice that in Figure 35 the vector angle is much closer to that of the walls (dθ in Figure 35 is small compared to dθ in Figure 33).
Spacing the vectors through several points eliminates the negative effects of LADAR error, an optimum VSpacing is ~ 30 (determined experimentally, Figure 26 uses a VSpacing = 30), but depends on the environment. As wall lengths decrease so does the optimal VSpacing value

**Un-orthogonal Bidirectional Match**

For the USAR application, walls cannot be assumed orthogonal. Therefore, two separate alignments are made for the substantial walls that are closest to perpendicular with one another. Figure 37 shows two different alignments that were made. The graphical scan representations below are rotations of Figure 32. Predominant walls are aligned normal to the x axis.

When the two histograms are created, there will be a peak at both walls that lies perpendicular to the x axis. Like before the histograms are matched and robot movement perpendicular to each of the two specified walls (∆D₁ and ∆D₂) is known. Once a displacement vector has been calculated, the original scans can be translated and the data compared to check the accuracy of the match (Figure 38).
Although, $\Delta D_1$ and $\Delta D_2$ have been validated, the two scans have not yet been completely matched as $D_1$ and $D_2$ do not form an orthogonal basis (Figure 39).

To adjust for the lack of perpendicularity, an orthogonal vector to $D_1$ will be derived, so that the two displacement vectors can be treated independently (Figure 40).

$D_{2\text{new}}$ is calculated using Equation 3.
Equation 3 - Vector Math

\[ D_{2\text{new}} = \frac{D_2}{\cos(\phi - \Delta \theta)} \]

\[ \Delta \theta = |\theta_{D1} - \theta_{D2}| \]

The final displacement vectors are now calculated and to validate the translational matching process, the second (new) scan is shifted \( D_1 + D_{2\text{new}} \) to match the first (previous) (Figure 41).

![Figure 41 - Subsequent Scans Completely Matched](image)

Mismatch Function

**Purpose**

Because SLAM success lies on the success of every individual scan match, an additional program was developed that ensures the fidelity of every scan.

**Procedure**

The mismatch function operates in the main program sequence and is follow up to the scan match program. Seven key values are passed out of the scan match function, they are used to carry out the rest of the SLAM process.

```c
localang[j] = results[0]; // angle change between the two input scans
localx[j] = results[1]; // change in y between the two input scans
localy[j] = results[2]; // change in x between the two input scans
```
corrang[j] = results[3];  // the correlation coeff. for angle match
corrd1[j] = results[4];  // the correlation coeff. for d1 match
corrd2[j] = results[5];  // the correlation coeff. for d2 match
deltawallangle[j] = results[6];  // angle between the walls that are matched

A series of if statements compare the correlation coefficients to critical values set by the user. If they do not satisfy then the scan is ignored and the program proceeds to the next scan. This is classified as a mismatch. Because robot pose in the global map is dependant upon the summation of all individual scan matches, ignoring one will throw the map into error. The mismatch function manages important data as well as reorders the scan sequence in such a way so that no data is ignored. Below is a flow chart (Figure 42) that illustrates how the SLAM program recovers from a mismatch because of the mismatch function.

![Mismatch Flow Chart](image)

**Occupancy Grid**

Scan matching is only half the battle, only localization is achieved using the histogram algorithm. A mapping program was developed to input the new robot pose estimation and LADAR data. The format for the map is a large grid that is multilayered. An imaginary line is drawn from the current robot location to each of the associated scan points. The cells through which the imaginary line runs, are registered as unoccupied, while the cells in which the point lies, is marked as occupied. Cells that are not involved remain registered as unknown. Each cell as it is hit and not hit builds a probability that it is either occupied or not, and this is what builds the overall occupancy grid, different shades can be used to represent the likelihood of occupancy for each cell.

**Mapping**

**SDL**
After many hours of searching the internet for a graphics API that was easy to use, SDL came up as the number one choice. SDL has many easy to use functions that will help out a lot. Since this paper is supposed to be designed to help the next year’s team, there will be a brief summary of how to set SDL up. Although our final product was written in Visual Studio, most of the mapping work was done in DEV CPP. Although initially OpenGL was a good choice, it became evident that OpenGL was for use with 3D graphics and stuff way above the scope of this project. The map drawn in this project was a simple 2D representation of the area. If in the future a 3D map will be used, OpenGL would be the number one suggestion.

**DEV CPP**

Dev Cpp was downloaded free from [www.bloodshedsoftware.net](http://www.bloodshedsoftware.net). Although it does not have a great debugger, it is easy to use for medium to small projects. Most of the code transfers into Visual Studio easily. For future projects, it would be advantageous to use Visual Studio from the beginning because of its superior debugging capabilities. There are free versions of Visual Studio from ENS and an express edition of Visual Studio is available free off the internet.

**SDL into DEV CPP**

Although this paragraph is about placing SDL into Dev Cpp, it is very similar to set up Visual Studio up for use with SDL. Some of this advice may seem trivial, but it may save a whole lot of time to the beginner programmer. Once Dev Cpp is installed, go to Tools<Check Updates/Packages click Check for updates. There will be a list of packages and crap, all of which can be downloaded and used somehow. Scroll down and find the SDL package and click Download Selection. Once the package is downloaded, go into Tools<Package Manager. Click the SDL package and click Install. Select the proper folders into which the dll and lib files will be placed. Once the package is installed properly, to start an SDL application, click File<New Project or Application. In the box that opens, there should be a grey bar at the top with tabs. Select the SDL tab and the SDL Template.

It would be desirable if whoever is doing this project next year already has knowledge of this, but if not, this information will prove to be useful.

**Mapping Methods**

**Constructor**

*Why the triple pointer*

In this mapping method, it was desirable to have the map be scalable. Furthermore, it was not desirable to simply make a huge matrix and only display the part of the matrix that has actual data on it because that’s a wasteful memory hog. Also, the map has a possibility of needing to be scaled over one order of magnitude between its least
resolution to its greatest resolution depending on the size of each square. The triple pointer allows freedom for the map to grow and shrink as needed to make its self large enough to store all the data.

The triple pointer

The three dimensional array used to store the data for the map has three indices. The x, y, and z direction. The x and y direction are obvious except that in SDL, the y zero is on the top of the screen, and it counts up as the pixels go downward. The z direction is set at five deep. The first layer, z=0, is the crosses layer. This layer represents the number of times that particular square has been crossed (the cross function will be described later. The second layer is the hits layer. Z=1 is the layer that represents the number of times the square indexed by x and y has been hit. The third layer or z=2 is the layer which indicates whether or not there is a victim in that square. The final layer, z=3 is the layer which represents the robots assumed path.

The triple pointer is created in three for loops. The first for loop determines the width of the matrix. The first pointee of the triple pointer points to arrays the length of the height of the map. The second pointees are the arrays stretching from top to bottom. Each of these second depth pointers points to an array containing the information in each cell. If in following years triple pointers are used, it is suggested to straight copy and paste this code. It is difficult to conceptualize and difficult to get right. When the triple pointer is set up correctly, it can be used as a standard 3 dimensional array.

Suggestion for future projects

In future projects, it would be better to create a type definition containing hits,crosses, victim information, and path information, then a two dimensional array of this type definition could be created and resized as needed. This would make it easier to read and debug, as well as make more sense to write. For knowledge about this, search for “type definition” and “struct” on the internet. There is a plenum of information about these two subjects.

Destructor

This destructor is simple because there is only one piece of dynamic memory created. When this program is ended, the map needs to be deleted. This is a fairly simple destructor. Although the program will run without deleting the dynamic memory, it creates a source for a memory leak, which can be a big pain in the butt.

Arrow( x , y , theta )
It was determined to be desirable to have an arrow in the map indicating the current pose of the robot. The arrow method takes in three parameters. The three parameters are the position of the robot in the x and y direction and the direction the robot is facing. The for loops walk over the centerline of the arrow, calling the populate functions. This function gives the functions arrowpopx and arrowpopy the correct begin and end points perpendicular to the arrow’s width.

**Arrowpop x/y (x1,y1,x2,y2)**

The name of this method is supposed to be short for arrow populate. This means it is these two functions which actually fill in the individual pixels for the arrow. These two functions are called by arrow() and are very similar to the xinterp() and yinterp() functions. These two functions are given the two points at sidemost points on the arrow. They then interpolate each of the two edge points to fill in the body of the arrow. Arrow() calls the arrowpop functions once for each width of the occupancies.

**Scale ()**

The scale () function is the function which actually draws to the pixels. The naming convention for scale() is that it is supposed to take the map and “scale” it onto the 300 pixel by 300 pixel surface designated for the map on the GUI. The idea for the scale function is that it separates the situation into two possible cases. The first case is the case where the map is larger than the screen size, and the second case is where the screen size is larger than the map. In each case, there needs to be a point on the map which maps to each pixel on the screen. In the case where the map is larger than the screen size, there are more points on the map than there are pixels on the screen; therefore the for loops will go through the pixels on the screen and pick a point out of the map which is representative of that pixel on the screen. This is done by multiplying each point on the screen by the size factor which makes the screen the size of the map, casting the scaled x and y values to integers which are the indices on the map which scale closest to the pixel on the screen. The function that scales the screen to the map is as follows:

$$\text{Point on map} = \text{Point on screen} \times \text{Width of screen} / \text{Width of map}$$

This function goes the same in the y direction if the widths are changed to heights.

This mapping scheme keeps the map square at all times, which keeps the width and height the same.

The second possibility, when the screen is bigger than the map, there are two nested for loops walking through all the points on the screen. Again, the points on the screen are scaled onto the map with the same equation and written to the SDL_Surface.

**Resize(x1,y1,x2,y2)**
This is the big daddy of the methods written in this mapping scheme. If there is any one thing I would suggest to change as little as possible, it would be this. This function changed two or three orders of magnitude in speed by changing only a few lines of code. This function does two things. First of all, this function does just as it says and resizes the map matrix if resizing is needed. This method determines if any of the four parameters are outside of its bounds. If any of them are, outside of the bounds, the method makes the array large enough hold all of the data, then changes the pointers around to where the old map is located correctly in the new memory arrays. The other thing this method does is delete the unused memory arrays. This is important to avoid memory leaks.

This method is crazy hard to imagine, because no values actually change. The only thing that happens is more memory is reserved, and the old memory is placed such that, when accessed by the indices, the correct array is called.

**X and yinterp (x1,y1,x2,y2)**

These two functions are the same function, except for one difference. In one case, each x integer between the minimum x value and the maximum x value are used to calculate an interpolated y value. The other function, yinterp (), each y integer between the two y values are used to interpolate an x value from x1 , x2, y1, and y2. These are the two functions used to count up the crosses. These functions are the ones used to increment the unoccupied part of the map.

**Xyinterp(x1,y1,x2,y2)**

This function is simply a function that calls the x and y functions, therefore doing a complete interpolation between the two points passed to it. This function takes in the four parameters, determines which which points (x1,y1) or (x2,y2) first so that when the xinterp () and yinterp () functions run through the for loops, the end condition is greater than the initial condition. This function also determines whether or not resize () needs to be called, as well as marks the point (x2,y2) with a “hit.”

**Pathinterpx, pathinterpy, and pathinterp(x1,y1,x2,y2)**

These three functions are very similar to xinterp (), yinterp (), and xyinterp (). They are the same except for one implementation difference. These functions take in the robot’s previous location, and the robot’s current location, create a line between them, and mark them. The biggest difference between the standard interpolation methods and these methods is, these methods have a width. Not only does the function mark the squares directly between the two points, but the also deviate by a #defined width, therefore making the path actually visible to the viewer.

**Vector_Read**
The Vector Read class is a class that allowed the code to be developed without having live LADAR scans. This class simply read in a formatted file containing ladar data. It converted into usable information so the map class could be created.

**How the Mapping Algorithm Works**
Above in this paper, the individual functions in the mapping algorithm were discussed. In the upcoming sections, the actual operatives of the algorithm will be discussed.

**Occupancy Grid**
The idea behind an occupancy grid is, a grid of squares is created, and each square is considered as one bulk chunk of space. What happens to one part of the chunk of space happens to the whole chunk of space. If it is determined that something is inside that chunk of space, it is assumed that something is inside the whole chunk of space. If the squares, previously called chunks of space are small enough, this is a fairly accurate assumption.

**Occupied, Unoccupied, or Unknown**
In an occupancy grid, the number of times an occupancy has been measured to divided by the total number of times it’s been measured (either hits or crosses) is the probability that occupancy is occupied. If the probability of occupancy is greater than fifty percent, the block is marked as occupied, otherwise it is marked as unoccupied. If a block hasn’t had any measurements to or through it, it is unknown whether or not it is occupied.

**Color Scheme**
In the current algorithm, occupied squares are marked red or pink depending on confidence, unoccupied squares are marked white, and unknown squares are marked black. An example of this is in Figure 43.
Previous Considerations

Three Case Algorithm

This was the first algorithm I thought of when I first tried to create a map. The algorithm is based on, if you focus on an individual square, and a line is coming into it, the line has to go out through one of the other three sides of the box. Figure 44 illustrates this. If the line exits through the top, the y value will be incremented, and the same thing will happen to the box above it. If the line exits through Side 3, x will be incremented, and the algorithm would continue untill it hit the end.

This program seemed good initially, but it was extremely buggy. It tended to screw up at random times. This algorithm was too buggy to impliment realistically.
**False Points Algorithm**

This algorithm was suggested by Stephen Goebel. This algorithm would create multiple points along the imaginary line between the robot and the measured point. It would be determined which occupancy each point lies in and “crosses” would be determined accordingly. Figure 45 illustrates this algorithm.
This algorithm never got tried because the one being used currently, worked. I think this algorithm would work fine, but I think it’s quite a bit more complex than the current method.

**Current Algorithm**

The current mapping algorithm is simple in use. There are two values used to determine the status of each occupancy in the grid. The number of times a measurement has been taken that has landed in that particular square, and the number of times a measurement has been taken through the square. The topic of discussion is how to populate all the squares from all 750 measurements in each scan efficiently enough to do it several times per second.

To do this, the idea that, every time the imaginary line connecting the robot to its measurement crosses a line in the grid, it also crosses through an occupancy. Therefore all that needs to be done is determine where the made up lines cross the grid lines, and all the squares are then located. To do this, both the horizontal lines and vertical lines must be considered.

The program uses two separate interpolation functions to populate all the squares on the grid. Xinterp () and yinterp () are the two methods called to do this. Xinterp and yinterp are the same function, except one function uses a for loop to walk through all the x values between the two points, calculating the y values, casting them to integers, and then marking them on the map.
Figure 46 shows what the map looks like after the xinterp() function has been called. As one can see, not all of the squares that the line passes through have been marked. This would be emphasized in the event where the endpoint is almost directly above the robot’s location. Because there would be no integer x values between the two floating point locations, there would be no squares marked. To fill in the unmarked squares, the yinterp() function is called. The yinterp function uses a for loop to walk through every integer y value between the endpoint and robot’s location. Figure 47 shows the resultant calculations and marks after the yinterp() function is done. As can be seen, all squares are marked. In the case above, where the endpoint of the vector is directly above the starting point of the vector, although the xinterp function wouldn’t mark any squares, the yinterp() function would mark all of them because there would be many y values between the starting point’s y value and the ending point’s y value.
Figure 47 - Next Step in Algorithm

This algorithm works well because there are very few calculations required to populate the occupancies. It uses a very elegant integer cast to eliminate needs for complex if statements. Figure 48 shows the resultant image for the above example.
Shading
It was suggested to me by teammates to create a shading algorithm which shades the occupancies based on how many times it has been crossed or hit. An algorithm like this may be created just to compare outputs. Right now, I think it wouldn’t be beneficial for two reasons. First, when a person sees a map, they want to see a clear, distinct representation of the surrounding areas. If lines were to be blurred and whites turned into grays, the map would be less clear. The way the map is, the layout is clear and understandable. If it were up to me, I would take out the pink shading as well.

SLAM Testing

Off Board (LADAR as stand alone...)
An experiment was constructed to simulate robot navigation through a test environment.

Purpose
Testing took place as the algorithm was being developed so each development step could be validated.

Setup
The robot is assumed to be equipped with a Hokuyo URG-04LX LADAR, and has no prior knowledge of either the path it will take, or the environment through which it navigates. The walls of the mock arena consist of varying materials, as LADAR accuracy depends on reflectivity and porosity of the wall. Figure 49 shows the experimental setup, the LADAR is
placed on the white poster board. An arbitrary path is scribed on the poster board and 29 static scans are acquired at discrete points along the path.

**Procedure**

For every subsequent scan acquired from the LADAR, new $\theta$, $D_1$, and $D_2$ displacements are calculated. These values relative to the previous robot pose and must be updated to a global reference frame. Since there is no prior knowledge of the map, the first robot reference frame initializes the global reference frame. To transform $\theta$ into a useful parameter one must recall its definition: the rotation from scan to the succeeding scan, so it should be more properly named $\theta_i$. The angle of the robot at any given time in the global reference frame is given in Equation 4.

**Equation 4 - Change Summation (Pose Update)**

$$\theta_{\text{global}} = \sum_{i=0}^{n} \theta_i$$

where “$n$” equals the number of previous scans. $D_1$ and $D_2$ are transformed to $x_{\text{global}}$ and $y_{\text{global}}$ with the following procedure:
Figure 50 - Vector Schematic

Figure 50 shows the vector transformations required to transpose $D_1$ and $D_{2\text{new}}$ of the robot reference frame to $X$ and $Y$ vectors of the global reference frame.

**Equation 5 - Vector Manipulation**

\[
\begin{align*}
\hat{x}_{\text{global}} &= D_1 \cdot i_{\text{global}} + D_{2\text{new}} \cdot i_{\text{global}} \\
\hat{y}_{\text{global}} &= D_1 \cdot j_{\text{global}} + D_{2\text{new}} \cdot j_{\text{global}}
\end{align*}
\]

**Dot Products**

1. $i_{\text{global}}$ = global unit vector in the $x$ direction
2. $j_{\text{global}}$ = global unit vector in the $y$ direction

**Trigonometry Equivalent**

\[
\begin{align*}
\hat{x}_{\text{global}} &= D_1 \cdot \cos(-\theta_{\text{global}} - \theta_{D1}) + D_{2\text{new}} \cdot \cos(-\theta_{\text{global}} - \theta_{D_{2\text{new}}}) \\
\hat{y}_{\text{global}} &= D_1 \cdot \sin(-\theta_{\text{global}} - \theta_{D1}) + D_{2\text{new}} \cdot \sin(-\theta_{\text{global}} - \theta_{D_{2\text{new}}})
\end{align*}
\]

**Results**

Figure 51 is a plot comparing measured location to SLAM determined location, using histogram scan matching. The blue trace is the measured trace. Each LADAR scan location was measured relative to the origin and recorded. The red trace was calculated by matching every scan to the previous scan using the histogram matching technique as described in this document. The global reference frame is defined by the first scan position and angle, and from then on computed angle and position displacement from one scan to the next are summed. Therefore any error in matching individual LADAR arrays will not be corrected in subsequent matches, but carried through the entire localization process.
Figure 51 - The final Match (mm)

Figure 52, Figure 53, and Figure 54 show each component of the Matching Process. One can more clearly see the variance between measured pose and calculated pose.
Conclusions

A benefit of this SLAM software is that it is highly modular so improvements can be made using the same platform software package. Concluding remarks are mostly going to comment and discuss future development opportunities.

Effectiveness

The effectiveness of the SLAM algorithm is yet to be rigorously tested however dry tests have shown desirable results. Because the scan match function is so pivotal to the overall SLAM functionality one can look to Off Board (LADAR as stand alone...) for general results. For comprehensive system results however are regretfully unavailable because tests have not yet been performed.

Improvements

I recommend that the first step of action to take in improving the 2006-07 SLAM development is extensive review of the tests performed in summer 2007, if that test information is available. From there a clear understanding of the newest modifications will be acquired and the next step will become clear. However from tests performed already it is clear that there are some improvements that would greatly improve the algorithm from its current state.

Currently there is no way to succintly test the final match after scan match runs. Both scans are completely modified to the algorithms best ability and should closely overlap. A test to confirm the quality of match would be a great asset to the team.

Another improvement would be changing many of the current user defined constants into variables that change as the environment changes. Here is a list of some of the user defined parameters that are habitat dependant.

This could be tied to the motor encoders, and a time function can be added to code so that this number equals “Safety Factor * Motor Speed * time between scans”

```c
#define TRAVELDIST 300  //the estimated distance that the robot can translate between scans (-x , x) (mm)
```
This could be tied to the gyroscope, and a time function can be added to code so that this number equals “Safety Factor * angular speed * time between scans”

```c
#define TRAVELANG 120 // the estimated distance that the robot can translate between scans (-x , x) (degrees)
```

This should modulate depending on average wall length in the surroundings, the longer the walls the greater VSPACE

```c
#define VSPACE 30 // the number of points that are skipped as the point to point angles are computed
```

When the surrounding are tightly packed around the robot, the angle histogram resolution can increase improving matching fidelity

```c
#define ANGHISTRESOLUTION 1 // the resolution of the angle histogram (degrees)
```

When the surrounding are tightly packed around the robot, the D1 and D2 histogram resolution can increase improving matching fidelity

```c
#define XYHISTRESOLUTION 5 // the resolution of the D1 and D2 histograms (mm)
```

```c
#define FINALSQUE 5 // not currently used (being developed)
```

Previous successful scan match processes can be logged in a data base, and experiments can be run to see what number of significant wall are sufficient depending on varying environments

```c
#define WALLSTOFIND 18 // how many maxima (from angle histogram) are acquired and passed into the D1 wall and D2 wall select algorithm
```

In tight surroundings the number of points per wall could increase, raising the standard of acceptable D1 and D2 walls

```c
#define PNTSPERWALL 5 // the critical number of lines that fall within a given wall angle category (on the angle histogram) _ classifies whether or not a wall is substantial
```

This could be modifies to more strictly limit the match quality in clean environments

```c
#define MATCHQUALITY 4 // number between -10 and 10 10 being 100% perfect match (must be the same scan to achieve this ideal)
```

This would be an important user defined constant to modify in real-time because it limits what environments can be mapped. If the walls in the range of view do not have a difference greater than MINWALLANGLE than the scan match will be thrown out regardless of the match quality

```c
#define MINWALLANGLE 25 // the critical angle between D1 and D2, if Angd1 - angd2 is less than this than the match is thrown out
```
In tricky environments when the robot is moving slow this variable could increase allowing the algorithm to search longer for a match

```
#define RESET 4  //how many times the mismatch function repeats looking for a match
```

Like RESET this parameter could increase when the robot is moving slowly to give the algorithm more opportunities to map a rough environment

```
#define SKIPCOUNT 5  //for the mismatch function, tries to match to 4 proceeding scans, then goes to a previous scan and repeats
```

This SLAM algorithm has a lot of room to be optimized so much just by changing these constants into variables, and some of these parameters are functions of one another.

**Robot Controller**

Because of the advanced controls and sensing requirements of the GS robot, relatively high computational power must be present on the platform. The 2007 user interface team tried to learn from the mistakes made in 2006, therefore rather than attempting once again to use interfaced microcontrollers as the central controls system, the user interface team of 2007 used a single board computer (SBC) as the main control unit. This provided greater computational power, industry standard hardware interface, and additional hardware solutions.

**CPU-1433**


The specific SBC in use is the Eurotec CPU-1433, with an AMD Geode 333 MHz processor. This processor is x86 compatible. When we first received the SBC we were concerned with its possibly limited abilities. We spent some time playing with Linux distributions designed solely for resource limited embedded applications. This turned out to be unnecessary. Several of the Linux installation programs indicated that it is probably 586 compatible, or at least very close.

**Hard Drive**

We had significant constraints on the hard drive we could use. It had to be steady state to be able to handle the vibrations from the robot, but had to be large enough to allow a full operating system. We decided to go with a 2GB compact flash card. This required an adapter to convert the pins to a small form factor, 44 pin IDE connection. These connectors were purchased off ebay for a minimal price. We purchased ones that support DMA, but we are not sure if this is required.

**Linux Distributions**

We tried out many different distributions of Linux over the year. The first distro which we were able to get running was Slackware (slackware.com). It was this success which verified for us that many of the mainstream distros of Linux would work on our SBC.

Now that we had Linux up and running it was on to getting all of our peripherals working. The LADAR was relatively easy to get working. It was just a matter of loading the kernel module
cdc-acm and we were able to communicate with the LADAR using the serial protocol given to us by the manufacturer.

The next major task was to get the wireless networking up and running. We were unable to get NdisWrapper working with Slackware. We tried it with several other distros and still had no luck. Finally we decided to try it with the distro which we had been told that there had been previous success, Fedora Core 4 (FC4) (http://ndiswrapper.sourceforge.net/joomla/index.php?option=com_openwiki/Itemid,29/id.list_c-f/ item #60). We tried FC4 and it worked.

Next came the time to get the Sensoray video compression board working. Because of installation problems with the Video board drivers. The documentation pointed us towards using Fedora Core 3 (FC3). See the SBC section for more on the software for this. FC3 is based around a 4K kernel stack size. Most windows drivers are written to use a 8K or 12 K stack size. See the wireless section for more information on why a stack size switch was necessary. FC3 was tried with a 16 K stack size, but this prohibited the video driver modules from loading into the kernel. When technical support at Sensoray was unable to help us, we switched to using Ubuntu 6.10. This had been recently added to the video documentation as a functional operating system. It worked well, has a 8K stack size in the kernel and boots very quickly.

Suse 9.2 was tried in parallel to FC3. It made the wireless device easy to configure, but had terrible boot times.

Currently the system is set up using ubuntu 6.10 server with the generic 2.6.17 kernel.

**RS-232 De-multiplexer**

Due to a limited number of serial ports on the single board computer, a demultiplexer is used to address the motor controllers. A PIC microcontroller serves as the demux, and relays a signal on 1 of three lines depending on the address byte received. This integrated device limits the SBC serial usage to one port and supplies signal to three separate motor controllers.

**Wireless**

A Dell 1450 is the chosen USB wireless adapter. It is capable of 802.11A/B/G communications. It weighs roughly 1.6 ounces without a cord and draws a maximum of 835mA, but typically draws about 500mA at 5V.

Ndiswrapper was used along with the windows driver files that came on the CD with the disk. In order to get the hardware to work, the standard Ndiswrapper instructions were followed. Please refer to those in order to reinstall Ndiswrapper. Versioning of Ndiswrapper has presented a significant problem. For example version 1.2 works in FC3, but version 1.8 and 1.38 work in Ubuntu. It is best practice to try the most recent version first and then jump backwards to some of the major releases.

Linux as an OS is moving toward a smaller stack size because it is more efficient, this creates a problem when trying to use a windows wireless driver that is written for an 8K or a 12K stack size. Linuxant (who provides a commercial version of software similar to Ndiswrapper) provides
a work around for 4K problems. Unfortunately, we found Linuxant to be unreliable both in their patches and drivers.

**Sensors**

**LADAR**

The Hokuyo URG-04LX LADAR is the device that the GS team is using. This 2D sensor scans angularly with an range of the 240 degrees and a radial range is 4 meters. This particular LADAR has an angular resolution of .36 degrees and a radial resolution of one millimeter. Each radial laser range measurement is an integer from 19 – 4000 mm, (19mm is the distance from the laser to the LADAR window).

![Figure 55 - Hokuyo URG-04LX](image-url)
If the closest opaque object is more than 4 meters away, then the internal software returns a 0, this is the only time the LADAR returns a zero. Every scan contains 750 points that describe a 2D representation of the environment surrounding the robot. The particular LADAR model that is used in the GS project is capable of 10Hz data acquisition rate, 7500 byte/s sampling rate. The URG-04LX comes with software that displays the LADAR output in real time with the ability to capture one second of scan data per request (10 scans). The data is saved in .csv (comma spaced values) which can easily be opened or modified in Microsoft Excel, Notepad, or WordPad.

Figure 56 - Screenshot of real-time URG-04LX display (using provided software developed by Hokuyo)

Figure 56 shows a display of the real-time scatter plot that the URG-04LX produces; this verifies that this model has a 240 degree angular range and a 4 meter radial range. To acquire data for real-time manipulation and reading for the SLAM software, a server was written for the robot controller which could be queried for LADAR scans. The LADAR was connected to the SBC via USB. The 2.6 version of the Linux kernel contains a USB serial driver named cdc-acm which provides a device named /dev/ttyACM0. Once we loaded this driver we were able to communicate with the LADAR using standard serial communication. The LADAR server provided a socket (actually many) to connect to which a user can request the data from the LADAR. Once the data is sent over the wireless network, the SLAM software maintains and utilizes the data.

Video
There are two Cameras on the Good Samaritan Robot. One thermal camera and one used for basic video feed capable use in low light situations.

A-10 Thermal Imager
The Thermovision A-10 Thermal Imager outputs a video signal over RS232 at 8fps or NTSC using a converter box that is mounted to the floor of the GS robot. The A10 automatically optimizes the 14-bit image into a standard 8-bit image depending on the relative darkness (coldness) or lightness(warmness) of the objects in view. It supports a resolution of 160x128. It is currently being used with an NTSC signal. The converter box has a female BNC connector for a video signal a DB9 connector for the serial port and a rather obscure cable connector to the camera. See Figure 57 for a reference picture and the appendices for a full data sheet.
Zerolux PC209IR
The second camera is a PC209IR camera from Supercircuits (Figure 58). It can transmit video using an NTSC composite signal. The PC209IR camera has a 1/3 Super HAD CCD that is rated down to 0.000 lux. It includes 7 IR LEDs to discreetly light up dark rooms. It has a resolution of 480x350 pixels and runs on 2.5 watts. The Zerolux camera has a female BNC connector. This also has an adapter to make it a female RCA plug. See the Appendix for a full Data Sheet.

Sensoray Model 314
Both composite video signals are attached to a secondary PC-104+ board stacked on top of the SBC. The video board is a Sensoray Model 314 frame grabber (Figure 59). The 314 is capable of many compression schemes and up to 720x480 resolution at 30 frames/second. The 314 will capture video at 30 frames/s and native resolutions for the cameras. See the appendix for the data sheet.
These blocks are then compressed using normalization and discreet cosine transform and then a quantization factor. This allows complete video signals to be transmitted using 5-6 Mbps, which leaves most of 802.11a bandwidth open for other uses. The Sensoray Model 314 is powered over the PCI bus and uses roughly three additional watts when not in use and 5 watts when in use.

**Video Stream Selection**

The 314 can only compress one video signal is handled at a time. Through software, the chosen video stream is selected and compressed. This will be initiated by the robot operator at the remote computer. He will hit the space bar on the computer to switch to the second camera (the A10). The GUI will handle the input from the space bar, send a signal to the host OS on the robot, which will send a command to the video board to stop the current stream and switch devices using the same compression stream. The video will be then be transmitted and received in the same manner as the original stream.

**Hardware Compression**

The Sensoray Model 314 does all of the necessary compression on board and helps. This good as it prevents the SBC from using processing power on video compression. The selected compression scheme for the Good Samaritan Robot is MPEG1 due to its simplicity and ability to incorporate audio. MPEG1 breaks the image into 16X16 pixel Macroblocks. Each Macroblock is broken into four 8X8 pixel Luminance(Y) blocks and two 8X8 Cr and Cb blocks. Cr and Cb have half the horizontal and vertical resolution of the Y component. Therefore, each macroblock is broken into six blocks. For more information on this read the technical contribution on video compression in the appendixes section.

**Transmission**

The compressed video will be streamed back using a real-time transport protocol (RTP) unicast. The support for this protocol is built into our robot linux operating system and can be viewed by
a variety of open source video players that can be coded used in our projects, such as the GUI. To implement the RTP, we are using a modified cap-server program provided with the 314 by Sensoray.

**Audio**

The Good Samaritan Robot has two channels of audio that are transmitted back to the user. The two analog microphones positioned on the left and right side of the robot are connected to the same Sensoray Model 314 Board. Using MPEG1 Compression, the audio and video are combined into the compression scheme and streamed in the same manner. On the receiving end, a separate port is used to access the stream. The GUI will take the incoming audio stream and direct it to the computer’s main audio. The operator wears headphones and hears what is going on around the robot.

The microphone currently used in the Good Samaritan robot was taken off of an old computer. This seemed appropriate as the PC104 stack handles audio the same as a full scale PC. To replace this part, use any standard PC microphone. The microphone connects directly to the PC104+ video capture card on the SBC and uses the same software as the video. The audio plays through a set of headphones the user is be wearing.

**CO₂**

The MG811 transducer (Figure 17) is the core of the carbon-dioxide sensing. The data sheet is available in the appendices. An amplification circuit was needed to amplify the 30mV signal from the MG811. The transducer must be supplied with 6vdc for heating of the coil, to produce an electromotive force (EMF) inversely proportional to the concentration level. The op-amp based circuit amplifies the signal by 100 to about 3v. This 3v will drop slightly in the presence of a high concentration of CO₂. The PIC16F876A (Figure 16) microcontroller will record this drop and serially (RS232) communicate this concentration ultimately to the user. The serial communication board is shown here in Figure 60.
The CO2 level will be displayed to the user by means of a progress bar on the user interface display. An example of the progress bar is shown here in Figure 61.

![Figure 61 - CO2 Level Bar](image)

**Inertial Measurement Sensor**

The inertial measurement unit (IMU) (Figure 24) consists of two separate sensors, a three-axis accelerometer and a two-axis gyro meter. The accelerometer is the Analog Devices ADXL330 and the gyro is an InvenSense IDG300. The IMU will produce a voltage proportional to the force when supplied with 3.3v. This voltage is determined by the PIC microcontroller and sent to the SBC to be transmitted to the user.

The inertial data will be displayed to the user by means of a series of images of the computer model in various orientations. Microsoft Visual Studio .NET was used to create a C++.NET program to change the image in a picture box to best represent the orientation of the robot. Each view, side and back, has a resolution of 30 degrees. Shown in Figure 62 is an example of how this will be displayed on the user interface display.
Figure 62 - Robot Orientation Display
Conclusions and Recommendations

Please see the Constraints and Criteria sections for details on the overall team accomplishments relative to the goals. Personal Accomplishments are as follows:

Kenny Darbone – Designed, specked, and implemented on the CO2 Sensor as well as the IMU (Inertial Measurement Unit). This included the design of a custom PCB that mounts on top of the PC104 stack to handle the A/D conversion from the afore mentioned sensors. This board also serves to multiplex the single RS232 signal from the SBC that does motor controls for the arm, articulation, and drive. Kenny also was responsible for entering the team in a ISTec student poster contest in an attempt to fundraise as well as maintaining the Electrical Engineering required website for the project: http://www.engr.colostate.edu/ece-sr-design/AY06_07/usar/usar.htm

Stephen Goebelm – Tracked and submitted the project plan weekly. Wrote the scan matching code for the SLAM. Actively participated in group tasks such as oral slides and project deliverables. Took the lead on custom cable production.

J. Shea Robinson – Prepared the weekly meeting agenda. Took the lead on testing the wireless and video board. Assisted in building custom cables as needed. Was central in testing various Linux distributions. Lead the way in pursuing personal donations.

Josh Schmidt – Wrote the code to turn matched scans into a map, in other words, was responsible for designing and coding the occupancy grid.

Steve Tranby – Designed the full GUI, including the implementation of the PS2 controller. Worked with the Platform team to troubleshoot the motor controllers. Worked with the Arm team to finalize how to control the arm. Helped out a lot in the troubleshooting of wireless, video, and operating system evaluation.

Brent Wilkins – Did early work on the Korebot to evaluate the option. Alongside Stephen, built a large number of the custom cables. Assisted in the selection of a OS. Wrote a portion of the server code on the robot to communicate with the GUI. Wrote the hardware API’s to simplify the sensor interfaces. Handled financial logistics and purchasing.

Overall, our project was accomplished well. There were technical speed bumps along the way that were unforeseeable. Specifically, the wireless solution has been unreliable at best. Additionally, the video drivers and accompanying software have been unreliable. When the video software is run streaming an RSTP server, the wireless route tables are broken after roughly 30 seconds of heavy transfer.

Future recommendations include the replacement of the USB wireless device with an Ethernet bridge. This will maximize the range and minimize the software conflicts on the single board computer. Secondly a new video capture board such as the MPEG4000-4 or MTV4000 that can handle concurrent video streams and is geared more towards streaming than capturing.
Technical References


   http://robotarenas.nist.gov/RoboCup-AAAAI%20Rescue%20Robot%20Competition%20Rules%202004%20(v1).pdf#search=%22search%20rescue%20robot%20rules%22

[4] PC209IR - Color Day and Night Camera with IR LED’s

   \2005_2006\Projects\USAR\USAR GS Controls\Deliverables\C&C

[6] Robin Murphy, University of Southwest Florida
   http://www.cse.usf.edu/~murphy/

   \2005_2006\Projects\USAR\USAR GS Controls\Deliverables\Engineering Report

Appendices
MG811 Carbon-Dioxide Sensor Datasheet

Features
Good sensitivity and selectivity to CO2
Low humidity and temperature dependency
Long stability and reproducibility

Application
Air Quality Control
Ferment Process Control
Room Temperature CO2 concentration Detection

Structure and Testing Circuit
Sensor Structure and Testing Circuit as Figure. It composed by solid electrolyte layer
(1). Gold electrodes (2). Platinum Lead
(3). Heater (4). Porcelain Tube (5). 100n double-layer stainless net (6). Nickel and
copper plated ring (7). Bakelite (8). Nickel and copper plated pin (9).

Working Principle
Sensor adopt solid electrolyte cell Principle. It is composed by the following solid cells:

\[ \text{Air, Au\text{[NASICON] carbonate]Au, air, CO}_2 \]

When the sensor exposed to CO2, the following electrodes reaction occurs:

- Cathodic reaction: \( 2\text{Li} + \text{CO}_2 + \frac{1}{2}\text{O}_2 + 2\text{e}^- = \text{Li}_2\text{CO}_3 \)
- Anodic reaction: \( 2\text{Na}^+ + \frac{1}{2}\text{O}_2 + 2\text{e}^- = \text{Na}_2\text{O} \)
- Overall chemical reaction: \( \text{Li}_2\text{CO}_3 = 2\text{Na} + \text{Na}_2\text{O} + 2\text{Li} + \text{CO}_2 \)

The Electromotive force (EMF) result from the above electrode reaction, accord with according to Nernst's equation:

\[ \text{EMF} = \text{E}_0 - \frac{R}{2F} \ln (P_{\text{CO}_2}) \]

\[ P_{\text{CO}_2} = \text{partial Pressure} \quad \text{E}_0 = \text{Constant Volume} \quad R = \text{Gas Constant volume} \]

\[ T = \text{Absolute Temperature} \quad F = \text{Faraday constant} \]

From Figure 1B, Sensor Heating voltage supplied from other circuit, When its surface temperature is high enough, the sensor equals to a cell, its two side would output voltage signal, and its result accord with Nernst's equation. In sensor testing, the impedance of amplifier should be within 100—1000Ω. Its testing

Tel: 86 371 67169070 67169080 Fax: 86 371 67169090 E-mail: sensor@371.net

75
current should be controlled below 1 pA.

Specifications:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Name</th>
<th>Technical</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_H</td>
<td>Heating Voltage</td>
<td>6.0±0.1 V</td>
<td>AC or DC</td>
</tr>
<tr>
<td>R_H</td>
<td>Heating Resistor</td>
<td>30.0±5% Ω</td>
<td>Room Temperature</td>
</tr>
<tr>
<td>I_H</td>
<td>Heating Current</td>
<td>@200mA</td>
<td></td>
</tr>
<tr>
<td>P_H</td>
<td>Heating Power</td>
<td>@1200mW</td>
<td></td>
</tr>
<tr>
<td>Tao</td>
<td>Operating Temperature</td>
<td>-20…+50°C</td>
<td></td>
</tr>
<tr>
<td>Tas</td>
<td>Storage Temperature</td>
<td>-20…+70°C</td>
<td></td>
</tr>
<tr>
<td>EMF</td>
<td>Output</td>
<td>30—50mV</td>
<td>350—10000ppm CO2</td>
</tr>
</tbody>
</table>

Sensitivity:

Figure 2 Shows gas sensor sensitivity curve.

- Conditions:
  - Temp: 25°C
  - RH: 65%
  - Oxygen: 21%
  - EMF: sensor EMF under different gas and concentration.

Response and Resume Characteristics:

Figure 3 shows solid electrolyte sensor response and recovery characteristics.

Temperature and Humidity Dependency:
ADXL330 3-Axis Accelerometer Datasheet

FEATURES
- 3-axis sensing
- Small, low-profile package
  - 4 mm x 4 mm x 1.45 mm LFCSLP
- Low power
  - 160 µA at VDD = 1.8 V (typical)
- Single-supply operation
  - 1.8 V to 3.6 V
- 10,000 g shock survival
- Excellent temperature stability
- BW adjustment with a single capacitor per axis
- RoHS/WEEE lead-free compliant

APPLICATIONS
- Cost-sensitive, low power, motion- and tilt-sensing applications
  - Mobile devices
  - Gaming systems
  - Disk drive protection
  - Image stabilization
  - Sports and health devices

GENERAL DESCRIPTION
The ADXL330 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs, all on a single monolithic IC. The product measures acceleration with a minimum full-scale range of ±3 g. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.

The user selects the bandwidth of the accelerometer using the Cx, Cy, and Cz capacitors at the XOUT, YOUT, and ZOUT pins. Bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis.

The ADXL330 is available in a small, low profile, 4 mm x 4 mm x 1.45 mm, 16-lead, plastic lead frame chip scale package (LFCSLP_LQ).

REV. A
Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, or for any infringements of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patents of Analog Devices. Trademarks and registered trademarks are the property of their respective owners.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel:781.239.4700  www.analog.com
Fax:781.461.3113  ©2006 Analog Devices, Inc. All rights reserved.
# ADXL330

## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features</td>
<td>1</td>
</tr>
<tr>
<td>Applications</td>
<td>1</td>
</tr>
<tr>
<td>General Description</td>
<td>1</td>
</tr>
<tr>
<td>Functional Block Diagram</td>
<td>1</td>
</tr>
<tr>
<td>Revision History</td>
<td>2</td>
</tr>
<tr>
<td>Specifications</td>
<td>3</td>
</tr>
<tr>
<td>Absolute Maximum Ratings</td>
<td>4</td>
</tr>
<tr>
<td>ESD Caution</td>
<td>4</td>
</tr>
<tr>
<td>Pin Configuration and Function Descriptions</td>
<td>5</td>
</tr>
<tr>
<td>Typical Performance Characteristics</td>
<td>6</td>
</tr>
<tr>
<td>Theory of Operation</td>
<td>11</td>
</tr>
<tr>
<td>Mechanical Sensor</td>
<td>11</td>
</tr>
<tr>
<td>Performance</td>
<td>11</td>
</tr>
<tr>
<td>Applications</td>
<td>12</td>
</tr>
<tr>
<td>Power Supply Decoupling</td>
<td>12</td>
</tr>
<tr>
<td>Setting the Bandwidth Using $C_2$, $C_3$, and $C_5$</td>
<td>12</td>
</tr>
<tr>
<td>Self Test</td>
<td>12</td>
</tr>
<tr>
<td>Design Trade-Offs for Selecting Filter Characteristics: The Noise/BW Trade-Off</td>
<td>12</td>
</tr>
<tr>
<td>Use with Operating Voltages Other than 3 V</td>
<td>12</td>
</tr>
<tr>
<td>Axes of Acceleration Sensitivity</td>
<td>13</td>
</tr>
<tr>
<td>Outline Dimensions</td>
<td>14</td>
</tr>
<tr>
<td>Ordering Guide</td>
<td>14</td>
</tr>
</tbody>
</table>

## REVISION HISTORY

- 9/06—Rev. 0 to Rev. A
- Changes to Ordering Guide                                             | 14   |
- 3/06—Revision 0: Initial Version
SPECIFICATIONS

Ta = 25°C, Vcc = 3 V, Cc = Cc = 0.1 μF, acceleration = 0 g, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Conditions</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSOR INPUT</td>
<td>Each axis</td>
<td>±3</td>
<td>±3.6</td>
<td></td>
<td>g</td>
</tr>
<tr>
<td>Nonlinearity</td>
<td>% of full scale</td>
<td>±0.3</td>
<td></td>
<td></td>
<td>%</td>
</tr>
<tr>
<td>Package Alignment Error</td>
<td></td>
<td>±1</td>
<td></td>
<td></td>
<td>Degrees</td>
</tr>
<tr>
<td>Interazis Alignment Error</td>
<td></td>
<td>±0.1</td>
<td></td>
<td></td>
<td>Degrees</td>
</tr>
<tr>
<td>Cross Axis Sensitivity</td>
<td></td>
<td>±1</td>
<td></td>
<td></td>
<td>%</td>
</tr>
<tr>
<td>SENSITIVITY (RATIOMETRIC)</td>
<td>Each axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensitivity at Xout, Yout, Zout</td>
<td>Vcc = 3 V</td>
<td>270</td>
<td>300</td>
<td>330</td>
<td>mV/g</td>
</tr>
<tr>
<td>Sensitivity Change Due to Temperature</td>
<td>Vcc = 3 V</td>
<td>±0.015</td>
<td></td>
<td></td>
<td>%/°C</td>
</tr>
<tr>
<td>ZERO g BIAS LEVEL (RATIOMETRIC)</td>
<td>Each axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 g Voltage at Xout, Yout, Zout</td>
<td>Vcc = 3 V</td>
<td>1.2</td>
<td>1.5</td>
<td>1.8</td>
<td>V</td>
</tr>
<tr>
<td>0 g Offset vs. Temperature</td>
<td></td>
<td>±1</td>
<td></td>
<td></td>
<td>mV/°C</td>
</tr>
<tr>
<td>NOISE PERFORMANCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noise Density Xout, Yout</td>
<td></td>
<td>280</td>
<td></td>
<td></td>
<td>μg/√Hz rms</td>
</tr>
<tr>
<td>Noise Density Zout</td>
<td></td>
<td>350</td>
<td></td>
<td></td>
<td>μg/√Hz rms</td>
</tr>
<tr>
<td>FREQUENCY RESPONSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth Xout, Yout</td>
<td>No external filter</td>
<td>1600</td>
<td></td>
<td></td>
<td>Hz</td>
</tr>
<tr>
<td>Bandwidth Zout</td>
<td>No external filter</td>
<td>550</td>
<td></td>
<td></td>
<td>Hz</td>
</tr>
<tr>
<td>Ref Tolerance</td>
<td></td>
<td>32 ± 15%</td>
<td></td>
<td></td>
<td>kΩ</td>
</tr>
<tr>
<td>Sensor Resonant Frequency</td>
<td></td>
<td>5.5</td>
<td></td>
<td></td>
<td>kHz</td>
</tr>
<tr>
<td>SELF TEST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic Input Low</td>
<td></td>
<td>+0.6</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Logic Input High</td>
<td></td>
<td>+2.4</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>ST Actuation Current</td>
<td></td>
<td>+60</td>
<td></td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Output Change at Xout</td>
<td>Self test 0 to 1</td>
<td></td>
<td></td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td>Output Change at Yout</td>
<td>Self test 0 to 1</td>
<td></td>
<td></td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td>Output Change at Zout</td>
<td>Self test 0 to 1</td>
<td></td>
<td></td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td>OUTPUT AMPLIFIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Swing Low</td>
<td>No load</td>
<td>0.1</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Output Swing High</td>
<td>No load</td>
<td>2.8</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>POWER SUPPLY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating Voltage Range</td>
<td>Vcc = 3 V</td>
<td>1.8</td>
<td>3.6</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Supply Current</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Turn-On Time</td>
<td>No external filter</td>
<td>320</td>
<td></td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td>TEMPERATURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>°C</td>
</tr>
</tbody>
</table>
ADXL330

ABSOLUTE MAXIMUM RATINGS

Table 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration (Any Axis, Unpowered)</td>
<td>10,000 g</td>
</tr>
<tr>
<td>Acceleration (Any Axis, Powered)</td>
<td>10,000 g</td>
</tr>
<tr>
<td>Vs</td>
<td>-0.3 V to +7.0 V</td>
</tr>
<tr>
<td>All Other Pins (COM – 0.3 V) to (V± + 0.3 V)</td>
<td>Indefinite</td>
</tr>
<tr>
<td>Output Short-Circuit Duration (Any Pin to Common)</td>
<td></td>
</tr>
<tr>
<td>Temperature Range (Powered)</td>
<td>-55°C to +125°C</td>
</tr>
<tr>
<td>Temperature Range (Storage)</td>
<td>-65°C to +150°C</td>
</tr>
</tbody>
</table>

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Figure 2. Recommended Soldering Profile

Table 3. Recommended Soldering Profile

<table>
<thead>
<tr>
<th>Profile Feature</th>
<th>Sn63/Pb37</th>
<th>Pb-Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Ramp Rate (Tₚ to Tₚ)</td>
<td>3°C/s max</td>
<td>3°C/s max</td>
</tr>
<tr>
<td>Preheat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum Temperature (Tₘₚₘₚₚ)</td>
<td>100°C</td>
<td>150°C</td>
</tr>
<tr>
<td>Maximum Temperature (Tₘₚₚₚₚ)</td>
<td>150°C</td>
<td>200°C</td>
</tr>
<tr>
<td>Time (Tₘₚₚₚₚ to Tₚₚₚₚₚ)</td>
<td>60 s to 120 s</td>
<td>60 s to 180 s</td>
</tr>
<tr>
<td>Tₚₚₚₚ to Tₚₚₚₚₚ</td>
<td>Ramp-Up Rate</td>
<td>3°C/s max</td>
</tr>
<tr>
<td>Time Maintained Above Liquidus (Tₚ)</td>
<td>183°C</td>
<td>217°C</td>
</tr>
<tr>
<td>Liquidus Temperature (Tₚ)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time (tₚ)</td>
<td>60 s to 150 s</td>
<td>60 s to 150 s</td>
</tr>
<tr>
<td>Peak Temperature (Tₚ)</td>
<td>240°C + 0°C to 5°C</td>
<td>260°C + 0°C to 5°C</td>
</tr>
<tr>
<td>Time within 5°C of Actual Peak Temperature (tₚ)</td>
<td>10 s to 30 s</td>
<td>20 s to 40 s</td>
</tr>
<tr>
<td>Ramp-Down Rate</td>
<td>6°C/s max</td>
<td>6°C/s max</td>
</tr>
<tr>
<td>Time 25°C to Peak Temperature</td>
<td>6 minutes max</td>
<td>8 minutes max</td>
</tr>
</tbody>
</table>

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although this product features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.
## PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

**Figure 3. Pin configuration**

**Figure 4. Recommended PCB Layout**

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NC</td>
<td>No Connect</td>
</tr>
<tr>
<td>2</td>
<td>ST</td>
<td>Self Test</td>
</tr>
<tr>
<td>3</td>
<td>COM</td>
<td>Common</td>
</tr>
<tr>
<td>4</td>
<td>NC</td>
<td>No Connect</td>
</tr>
<tr>
<td>5</td>
<td>COM</td>
<td>Common</td>
</tr>
<tr>
<td>6</td>
<td>COM</td>
<td>Common</td>
</tr>
<tr>
<td>7</td>
<td>COM</td>
<td>Common</td>
</tr>
<tr>
<td>8</td>
<td>ZOUT</td>
<td>Z Channel Output</td>
</tr>
<tr>
<td>9</td>
<td>NC</td>
<td>No Connect</td>
</tr>
<tr>
<td>10</td>
<td>YOUT</td>
<td>Y Channel Output</td>
</tr>
<tr>
<td>11</td>
<td>NC</td>
<td>No Connect</td>
</tr>
<tr>
<td>12</td>
<td>XOUT</td>
<td>X Channel Output</td>
</tr>
<tr>
<td>13</td>
<td>NC</td>
<td>No Connect</td>
</tr>
<tr>
<td>14</td>
<td>Vs</td>
<td>Supply Voltage (1.8 V to 3.6 V)</td>
</tr>
<tr>
<td>15</td>
<td>Vs</td>
<td>Supply Voltage (1.8 V to 3.6 V)</td>
</tr>
<tr>
<td>16</td>
<td>NC</td>
<td>No Connect</td>
</tr>
</tbody>
</table>
TYPICAL PERFORMANCE CHARACTERISTICS

N > 1000 for all typical performance plots, unless otherwise noted.

Figure 5. X-Axis Zero g Bias at 25°C, Vi = 3 V

Figure 6. Y-Axis Zero g Bias at 25°C, Vi = 3 V

Figure 7. Z-Axis Zero g Bias at 25°C, Vi = 3 V

Figure 8. X-Axis Zero g Bias at 25°C, Vi = 2 V

Figure 9. Y-Axis Zero g Bias at 25°C, Vi = 2 V

Figure 10. Z-Axis Zero g Bias at 25°C, Vi = 2 V
Figure 11. X-Axis Zero g Bias Temperature Coefficient, $V_s = 3 \text{ V}$

Figure 14. X-Axis Zero g Bias vs. Temperature—4 Parts Soldered to PCB

Figure 12. Y-Axis Zero g Bias Temperature Coefficient, $V_s = 2 \text{ V}$

Figure 15. Y-Axis Zero g Bias vs. Temperature—4 Parts Soldered to PCB

Figure 13. Z-Axis Zero g Bias Temperature Coefficient, $V_s = 3 \text{ V}$

Figure 16. Z-Axis Zero g Bias vs. Temperature—4 Parts Soldered to PCB
Figure 17. X-Axis Sensitivity at 25°C, V<sub>f</sub> = 3 V

Figure 18. Y-Axis Sensitivity at 25°C, V<sub>f</sub> = 3 V

Figure 19. Z-Axis Sensitivity at 25°C, V<sub>f</sub> = 3 V

Figure 20. X-Axis Sensitivity at 25°C, V<sub>f</sub> = 2 V

Figure 21. Y-Axis Sensitivity at 25°C, V<sub>f</sub> = 2 V

Figure 22. Z-Axis Sensitivity at 25°C, V<sub>f</sub> = 2 V
Figure 29. Typical Current Consumption vs. Supply Voltage

Figure 30. Typical Turn-On Time—Cn, Cx, Cz = 0.0047 μF, Vs = 3 V
THEORY OF OPERATION

The ADXL330 is a complete 3-axis acceleration measurement system on a single monolithic IC. The ADXL330 has a measurement range of ±3 g minimum. It contains a polysilicon surface micromachined sensor and signal conditioning circuitry to implement an open-loop acceleration measurement architecture. The output signals are analog voltages that are proportional to acceleration. The accelerometer can measure the static acceleration of gravity in tilt sensing applications as well as dynamic acceleration resulting from motion, shock, or vibration.

The sensor is a polysilicon surface micromachined structure built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces. Deflection of the structure is measured using a differential capacitor that consists of independent fixed plates and plates attached to the moving mass. The fixed plates are driven by 180° out-of-phase square waves. Acceleration deflects the moving mass and unbalances the differential capacitor resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation techniques are then used to determine the magnitude and direction of the acceleration.

The demodulator output is amplified and brought off-chip through a 32 kΩ resistor. The user then sets the signal bandwidth of the device by adding a capacitor. This filtering improves measurement resolution and helps prevent aliasing.

MECHANICAL SENSOR

The ADXL330 uses a single structure for sensing the X, Y, and Z axes. As a result, the three axesense directions are highly orthogonal with little cross-axis sensitivity. Mechanical misalignment of the sensor die to the package is the chief source of cross-axis sensitivity. Mechanical misalignment can, of course, be calibrated out at the system level.

PERFORMANCE

Rather than using additional temperature compensation circuitry, innovative design techniques ensure high performance is built-in to the ADXL330. As a result, there is neither quantization error nor nonmonotonic behavior, and temperature hysteresis is very low (typically less than 3 mg over the −25°C to +70°C temperature range).

Figure 14, Figure 15, and Figure 16 show the zero g output performance of eight parts (X-, Y-, and Z-axis) soldered to a PCB over a −25°C to +70°C temperature range.

Figure 26, Figure 27, and Figure 28 demonstrate the typical sensitivity shift over temperature for supply voltages of 3 V. This is typically better than ±1% over the −25°C to +70°C temperature range.
Applications

Power Supply Decoupling

For most applications, a single 0.1 µF capacitor, Cdc, placed close to the ADXL330 supply pins adequately decouples the accelerometer from noise on the power supply. However, in applications where noise is present at the 50 Hz internal clock frequency (or any harmonic thereof), additional care in power supply bypassing is required as this noise can cause errors in acceleration measurement. If additional decoupling is needed, a 100 Ω (or smaller) resistor or ferrite bead can be inserted in the supply line. Additionally, a larger bulk bypass capacitor (1 µF or greater) can be added in parallel to Cdc. Ensure that the connection from the ADXL330 ground to the power supply ground is low impedance because noise transmitted through ground has a similar effect as noise transmitted through Vcc.

Setting the Bandwidth Using Cc, Cc, and C2

The ADXL330 has provisions for band limiting the Xacc, Yacc, and Zacc pins. Capacitors must be added at these pins to implement low-pass filtering for anti-aliasing and noise reduction. The equation for the 3 dB bandwidth is

\[ F_{3\,db} = \frac{1}{2\pi(32 \, k\Omega) \times Cc \times \epsilon} \]

or more simply

\[ F_{3\,db} = \frac{5 \, \mu F}{Cc \times \epsilon} \]

The tolerance of the internal resistor (Racc) typically varies as much as ±15% of its nominal value (32 kΩ), and the bandwidth varies accordingly. A minimum capacitance of 0.0047 µF for Cacc, Ccc, and C2 is recommended in all cases.

Table 5. Filter Capacitor Selection, Ccc, Ccc, and C2

<table>
<thead>
<tr>
<th>Bandwidth (Hz)</th>
<th>Capacitor (µF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.07</td>
</tr>
<tr>
<td>10</td>
<td>0.087</td>
</tr>
<tr>
<td>50</td>
<td>0.10</td>
</tr>
<tr>
<td>100</td>
<td>0.025</td>
</tr>
<tr>
<td>200</td>
<td>0.027</td>
</tr>
<tr>
<td>500</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Self Test

The ST pin controls the self test feature. When this pin is set to Vcc, an electrostatic force is exerted on the accelerometer beam. The resulting movement of the beam allows the user to test if the accelerometer is functional. The typical change in output is −500 mg (corresponding to −150 mV) in the X-axis, 500 mg (or 150 mV) on the Y-axis, and −200 mg (or −60 mV) on the Z-axis. This ST pin may be left open circuit or connected to common (COM) in normal use.

Never expose the ST pin to voltages greater than Vcc + 0.3 V. If this cannot be guaranteed due to the system design (for instance, if there are multiple supply voltages), then a low Vcc clamping diode between ST and Vcc is recommended.

Design Trade-offs for Selecting Filter Characteristics: The Noise/BW Trade-off

The selected accelerometer bandwidth ultimately determines the measurement resolution (smallest detectable acceleration). Filtering can be used to lower the noise floor to improve the resolution of the accelerometer. Resolution is dependent on the analog filter bandwidth at Xacc, Yacc, and Zacc.

The output of the ADXL330 has a typical bandwidth of greater than 500 Hz. The user must filter the signal at this point to limit aliasing errors. The analog bandwidth must be no more than half the analog-to-digital sampling frequency to minimize aliasing. The analog bandwidth can be further decreased to reduce noise and improve resolution.

The ADXL330 noise has the characteristics of white Gaussian noise, which contributes equally at all frequencies and is described in terms of μg/√Hz (the noise is proportional to the square root of the accelerometer bandwidth). The user should limit bandwidth to the lowest frequency needed by the application to maximize the resolution and dynamic range of the accelerometer.

With the single-pole, roll-off characteristic, the typical noise of the ADXL330 is determined by

\[ \text{rms Noise} = \text{Noise Density} \times \sqrt{\text{BW} \times 1.5} \]

Often, the peak value of the noise is desired. Peak-to-peak noise can only be estimated by statistical methods. Table 6 is useful for estimating the probability of exceeding various peak values, given the rms value.

Table 6. Estimation of Peak-to-Peak Noise

<table>
<thead>
<tr>
<th>Peak-to-Peak Value</th>
<th>% of Time That Noise Exceeds Nominal Peak-to-Peak Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 x rms</td>
<td>32</td>
</tr>
<tr>
<td>4 x rms</td>
<td>16</td>
</tr>
<tr>
<td>6 x rms</td>
<td>27</td>
</tr>
<tr>
<td>8 x rms</td>
<td>0.006</td>
</tr>
</tbody>
</table>

Use with Operating Voltages Other Than 3 V

The ADXL330 is tested and specified at Vcc = 3 V; however, it can be powered with Vcc as low as 1.8 V or as high as 3.6 V. Note that some performance parameters change as the supply voltage is varied.
The ADXL330 output is ratiometric, therefore, the output sensitivity (or scale factor) varies proportionally to the supply voltage. At $V_S = 3.6 \, \text{V}$, the output sensitivity is typically $560 \, \text{mV/g}$. At $V_S = 2 \, \text{V}$, the output sensitivity is typically $195 \, \text{mV/g}$.

The zero $g$ bias output is also ratiometric, so the zero $g$ output is nominally equal to $V_S/2$ at all supply voltages.

The output noise is not ratiometric but is absolute in volts; therefore, the noise density decreases as the supply voltage increases. This is because the scale factor (mV/g) increases while the noise voltage remains constant. At $V_S = 3.6 \, \text{V}$, the X- and Y-axis noise density is typically $230 \, \mu\text{V}/\sqrt{\text{Hz}}$, while at $V_S = 2 \, \text{V}$, the X- and Y-axis noise density is typically $350 \, \mu\text{V}/\sqrt{\text{Hz}}$.

Self test response in $g$ is roughly proportional to the square of the supply voltage. However, when ratiometricity of sensitivity is factored in with supply voltage, the self test response in volts is roughly proportional to the cube of the supply voltage. For example, at $V_S = 3.6 \, \text{V}$, the self test response for the ADXL330 is approximately $-275 \, \text{mV}$ for the X-axis, $+275 \, \text{mV}$ for the Y-axis, and $-100 \, \text{mV}$ for the Z-axis.

At $V_S = 2 \, \text{V}$, the self test response is approximately $-60 \, \text{mV}$ for the X-axis, $+60 \, \text{mV}$ for the Y-axis, and $-25 \, \text{mV}$ for the Z-axis.

The supply current decreases as the supply voltage decreases. Typical current consumption at $V_S = 3.6 \, \text{V}$ is $375 \, \mu\text{A}$, and typical current consumption at $V_S = 2 \, \text{V}$ is $200 \, \mu\text{A}$.

**AXES OF ACCELERATION SENSITIVITY**

![Figure 31: Axes of Acceleration Sensitivity, Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis](image)

![Figure 32: Output Response vs. Orientation to Gravity](image)
ADXL330

OUTLINE DIMENSIONS

Figure 33. 16-Lead Lead Frame Chip Scale Package (LFCSLP_LQ)
4 mm × 4 mm Body, Thick Quad
(CP-16-5)
Dimensions shown in millimeters

ORDERING GUIDE

<table>
<thead>
<tr>
<th>Model</th>
<th>Measurement Range</th>
<th>Specified Voltage</th>
<th>Temperature Range</th>
<th>Package Description</th>
<th>Package Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADXL330KCPZ1</td>
<td>±3 g</td>
<td>3 V</td>
<td>−25°C to +30°C</td>
<td>16-Lead LFCSLP_LQ</td>
<td>CP-16-5</td>
</tr>
<tr>
<td>ADXL330KCPZ-R1</td>
<td>±3 g</td>
<td>3 V</td>
<td>−25°C to +30°C</td>
<td>16-Lead LFCSLP_LQ</td>
<td>CP-16-5</td>
</tr>
<tr>
<td>EVAL-ADXL33021</td>
<td>±3 g</td>
<td>3 V</td>
<td>−25°C to +30°C</td>
<td>Evaluation Board</td>
<td></td>
</tr>
</tbody>
</table>

1 Z = Pb-free part

Rev. A | Page 14 of 16
**IDG-300 Dual-Axis Gyro Datasheet**

**Integrated Dual-Axis Gyro**

**IDG-300**

### FEATURES
- Integrated X- and Y-axis gyro on a single chip
- Factory trimmed full scale range of ±500°/sec
- Integrated low-pass filters
- Superior vibration rejection over a wide frequency range
- High cross-axis isolation by design
- 3V single supply operation
- 5000 g shock tolerance
- RoHS compliant (completely lead free)
- 6 x 6 x 1.5mm QFN package

### APPLICATIONS
- Inertial measurement units (IMUs)
- Handheld GPS navigation devices
- Radio controlled helicopters
- Toys and game consoles
- Robotic and power tools
- Antenna positioning
- Remote control

### GENERAL DESCRIPTION

The IDG-300 is an integrated dual-axis angular rate sensor (gyroscope). It uses InvenSense's proprietary and patented MEMS technology with vertically driven, vibrating masses to make a functionally complete, low-cost, dual-axis angular rate sensor. All required electronics are integrated onto a single chip with the sensor.

The IDG-300 gyro uses two sensor elements with novel vibrating dual-mass bulk silicon configurations that sense the rate of rotation about the X- and Y-axis (in-plane sensing). This results in a unique, integrated dual-axis gyro with guaranteed-by-design vibration rejection and high cross-axis isolation. It is specifically designed for demanding consumer applications requiring low cost, small size, and high performance.

The IDG-300 gyro includes integrated electronics necessary for application-ready functionality. It incorporates X- and Y-axis low-pass filters and an EEPROM for on-chip factory calibration of the sensor. Factory trimmed scale factors eliminate the need for external active components and end-user calibration. This product is lead-free and RoHS compliant.

---

*See Design Notes: Section 5*
### SPECIFICATIONS

All parameters specified are @ VDD=3.0 V and T=25°C. External LPF @ 2kHz. All specifications apply to both axes.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Conditions</th>
<th>Min</th>
<th>Typical</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SENSITIVITY</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full-Scale Range</td>
<td></td>
<td>±500</td>
<td>±2.0</td>
<td>±6</td>
<td>%/s</td>
</tr>
<tr>
<td>Sensitivity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mV/°s</td>
</tr>
<tr>
<td>Initial Calibration Tolerance</td>
<td></td>
<td>-10</td>
<td>5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Over Specified Temperature</td>
<td></td>
<td>±10</td>
<td></td>
<td></td>
<td>%</td>
</tr>
<tr>
<td>Nonlinearity</td>
<td>Best Fit Straight Line</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cross-axis Sensitivity</td>
<td></td>
<td>±2</td>
<td></td>
<td></td>
<td>%</td>
</tr>
<tr>
<td><strong>ZERO-RATE OUTPUT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static Output (Bias)</td>
<td></td>
<td>±100</td>
<td>±1.5</td>
<td>+100</td>
<td>V</td>
</tr>
<tr>
<td>Initial Calibration Tolerance</td>
<td></td>
<td>±200</td>
<td></td>
<td>±500</td>
<td>mV</td>
</tr>
<tr>
<td>Over Specified Temperature</td>
<td></td>
<td>±200</td>
<td></td>
<td>±500</td>
<td>mV</td>
</tr>
<tr>
<td><strong>FREQUENCY RESPONSE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High Frequency Cutoff</td>
<td>Internal LPF -90°</td>
<td>±140</td>
<td>±5</td>
<td>±3</td>
<td>Hz</td>
</tr>
<tr>
<td>LPF Phase Delay</td>
<td>10kHz</td>
<td>±4.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MECHANICAL FREQUENCIES</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resonant Frequency</td>
<td>X-Axis Gyroscope</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>kHz</td>
</tr>
<tr>
<td>Resonant Frequency</td>
<td>Y-Axis Gyroscope</td>
<td>13</td>
<td>16</td>
<td>17</td>
<td>kHz</td>
</tr>
<tr>
<td>Frequency Separation</td>
<td>X and Y Gyroscopes</td>
<td>13</td>
<td>16</td>
<td>17</td>
<td>kHz</td>
</tr>
<tr>
<td><strong>OUTPUT DRIVE CAPABILITY</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Voltage Swing</td>
<td>Load = 100kΩ to VDD/2</td>
<td>0.05</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Capacitive Load Drive</td>
<td></td>
<td>100</td>
<td>±5</td>
<td>±5</td>
<td>pF</td>
</tr>
<tr>
<td>Output Impedance</td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td>Ω</td>
</tr>
<tr>
<td><strong>REFERENCE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voltage Value</td>
<td>Load Drive</td>
<td>1.23</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Load Drive</td>
<td>Capacitive Load Drive</td>
<td>1</td>
<td>100</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>Power Supply Rejection</td>
<td>Power Supply Rejection</td>
<td>VDD = 3.0V to 3.3V</td>
<td>1</td>
<td>1</td>
<td>pF</td>
</tr>
<tr>
<td>Over Specified Temperature</td>
<td></td>
<td>±5</td>
<td></td>
<td></td>
<td>mV/V</td>
</tr>
<tr>
<td><strong>POWER-UP RESPONSE</strong></td>
<td>Zero-rate settling time</td>
<td>±33/sec</td>
<td>200</td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NOISE PERFORMANCE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate Noise Density</td>
<td></td>
<td>0.014</td>
<td></td>
<td></td>
<td>μV/√Hz</td>
</tr>
<tr>
<td><strong>POWER SUPPLY</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating Voltage Range</td>
<td>Quescent Supply Current</td>
<td>±2</td>
<td>3.3</td>
<td>9.5</td>
<td>V/mA</td>
</tr>
<tr>
<td>Quiescent Supply Current</td>
<td>Over Specified Temperature</td>
<td>±2</td>
<td>±5</td>
<td>±5</td>
<td>mA</td>
</tr>
<tr>
<td><strong>TEMPERATURE RANGE</strong></td>
<td>Performance parameters are not applicable beyond Specified Temperature Range</td>
<td>0 to +70°C</td>
<td>0 to +70°C</td>
<td>0 to +70°C</td>
<td>°C</td>
</tr>
<tr>
<td>Specified Temperature Range</td>
<td>Extended Temperature Range</td>
<td>-20 to +85°C</td>
<td>-20 to +85°C</td>
<td>-20 to +85°C</td>
<td>°C</td>
</tr>
</tbody>
</table>
IDG-300

ABSOLUTE MAXIMUM RATINGS

Stress above those listed as "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device under these conditions is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>3.3V to 5.5V</td>
</tr>
<tr>
<td>Acceleration (Any Axis, unpowered)</td>
<td>500g for 0.3ms</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>-40 to +125°C</td>
</tr>
<tr>
<td>Storage Temperature Range</td>
<td>-40 to +125°C</td>
</tr>
</tbody>
</table>

PACKAGE DIMENSIONS (all dimensions in mm)

**TOP VIEW**

**BOTTOM VIEW**

**SIDE VIEW**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Dimensions</th>
<th>Dimensions (inch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Min</td>
<td>Nom</td>
</tr>
<tr>
<td>A1</td>
<td>9.00</td>
<td>9.02</td>
</tr>
<tr>
<td>A2</td>
<td>9.00</td>
<td>9.02</td>
</tr>
<tr>
<td>B</td>
<td>1.19</td>
<td>1.20</td>
</tr>
<tr>
<td>D</td>
<td>5.28</td>
<td>5.30</td>
</tr>
<tr>
<td>E</td>
<td>3.03</td>
<td>3.08</td>
</tr>
<tr>
<td>L</td>
<td>0.36</td>
<td>0.38</td>
</tr>
<tr>
<td>LL</td>
<td>0.30</td>
<td>0.32</td>
</tr>
<tr>
<td>EZ</td>
<td>4.50</td>
<td>4.50</td>
</tr>
</tbody>
</table>
PIN DESCRIPTION

<table>
<thead>
<tr>
<th>Number</th>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2, 22, 25, 36, 39, 40</td>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>14, 28, 34</td>
<td>VDD</td>
<td>Positive supply voltage: ±3.3V to ±3.3V</td>
</tr>
<tr>
<td>3</td>
<td>X-Rate Out</td>
<td>X-Rate Out</td>
</tr>
<tr>
<td>8</td>
<td>XAGE</td>
<td>Amplitude control filter (See Design Notes: Section 2)</td>
</tr>
<tr>
<td>17</td>
<td>COUT</td>
<td>Charge pump capacitor</td>
</tr>
<tr>
<td>20</td>
<td>YAGE</td>
<td>Amplitude control filter (See Design Notes: Section 2)</td>
</tr>
<tr>
<td>32</td>
<td>VREF</td>
<td>1.23V precision reference output</td>
</tr>
<tr>
<td>1, 9, 10, 11, 12, 13, 15, 16, 18, 22, 21, 30, 31, 33, 35, 36, 37</td>
<td>RESV</td>
<td>Reserved. Do not connect. Used for factory trimming</td>
</tr>
<tr>
<td>4, 5, 6, 7, 24, 26, 27</td>
<td>NC</td>
<td>Not internally connected: may be used for PCB routing</td>
</tr>
</tbody>
</table>

PIN CONNECTION (TOP VIEW)

This is a dual-axis rate sensing device. It produces a positive output voltage for rotation about the X- or Y-axis, as shown in the figure below.
DESIGN NOTES

1. Overview

The IDG-300 gyro is a dual-axis gyroscope consisting of two independent vibratory MEMS gyroscopes. One detects rotation about the X-axis, the other detects rotation about the Y-axis. Each structure is fabricated using InvenSense's proprietary bulk silicon technology. The structures are covered and hermetically sealed at the wafer-level. The cover shields the gyro from EMI.

The gyroscope's proof-masses are electrostatically oscillated at resonance. An internal automatic gain control circuit precisely sets the oscillation of the proof-masses. When the sensor is rotated about the X- or Y-axis, the Coriolis effect causes a vibration that can be detected by a capacitive pickoff. The resulting signal is amplified, demodulated, and filtered to produce an analog voltage that is proportional to the angular rate.

2. Amplitude Control

The scale factor of the gyroscope depends on the amplitude of the mechanical motion and the trim setting of the internal programmable gain stages. The oscillation circuit precisely controls the amplitude to maintain constant sensitivity over the temperature range. The capacitors (0.22µF, ±10%) connected to Pin E (XAXC) and Pin 23 (YAXC) are compensation capacitors for the amplitude control loop.

3. External Low-Pass Filter

An external low-pass filter is recommended to attenuate high-frequency noise. The cutoff frequency should be less than 2 kHz to attenuate tones above 10 kHz generated by the vibrating proof-masses. Recommended RC values for the 2 kHz filter are RLPX=47kΩ and CLPX=0.1µF respectively. The sensor bandwidth is limited to 140 Hz by the internal low-pass filter. Applications that require lower bandwidth should choose an external filter with a cutoff frequency less than 140Hz.

4. Scale Factor

The IDG-300 Rate-Out is not ratiometric to the supply voltage. The scale factor is calibrated at the factory and is nominally independent of supply voltage.

5. Power Supply Filtering

The IDG-300 gyro should be isolated from system power supply noise by a combination of an RC filter that attenuates high frequency noise and a Low Drop Out power supply regulator (LDO) that attenuates low frequency noise. Figure below shows a typical configuration.

The low-pass RC filter should be chosen such that it provides significant attenuation of system noise at high frequencies. The LDO should be a low noise regulator (<10µV/√Hz) that exhibits good noise rejection at low frequencies.

6. VREF

VREF is a temperature independent voltage reference that can be used as a reference for an ADC. There is offset between the zero rate output and VREF.
Linksys WRT55AG Wireless Router Datasheet

An Internet-Sharing Router and Switch, with Universal Wireless Access (A, B, and G)

The Dual-Band Wireless A+G Broadband Router is like four devices in one box! The Router function lets you securely share one high-speed Internet connection among your entire network, while the 4-port full-duplex 10/100 Switch jump-starts your wired-Ethernet network. Connect four PCs directly or daisy-chain out to more hubs and switches to create as big a network as you need.

The Dual-Band Wireless A+G Broadband Router also contains two Wireless Access Points, supporting all three wireless networking specifications. The first Access Point uses the 2.4GHz radio band, supporting both the popular and inexpensive Wireless-B (802.11b) standard at 11Mbps, and the new, almost five times faster, Wireless-G (802.11g) at 54Mbps. The second Access Point radio operates in the 5GHz band, and supports Wireless-A (802.11a) networking, also at 54Mbps. Since the two radios operate in different bands, they can work simultaneously, blanketing your wireless zone with bandwidth.

To protect your data and privacy, the Dual-Band Wireless A+G Broadband Router can encode all wireless transmissions with industrial-strength WPA encryption. The MAC Address filter lets you decide exactly who has access to your wireless network. The Router also serves as a DHCP Server, supports VPN pass-through, and can be configured to filter internal users access to the Internet. Configuration is a snap with the web browser-based configuration utility.

With the Linksys Dual-Band Wireless A+G Broadband Router at the center of your home or office network, you can share a high-speed Internet connection, files, printers, and multi-player games with the flexibility, speed, and security you need!

| Dual-band, tri-standard Access Point communicates with Wireless-A (802.11a), Wireless-B (802.11b), and Wireless-G (802.11g) wireless networks | Protect your wireless investment while preparing your infrastructure for the future | Built-in 4-port Switch jump-start your wired network, while the Router functionality securely shares your high-speed Internet connection with wired and wireless computers | Advanced Security: Stateful Packet Inspection Firewall, 152-bit Wireless data encryption (WEP), VPN pass-through, Internet access control and Wireless filers |

Dual-Band Wireless A+G Broadband Router

Product Data

Model No. WRT55AG
Dual-Band Wireless A+G
Broadband Router

Features

• Performance Investment Protection: Compatibility with Wireless-A (802.11a), Wireless-B (802.11b) and Wireless-G Standards
• Built-in 4-Port 10/100 Switch Supports Wired Ethernet Clients
• All Ethernet Ports Support Auto-MDI/MDI-X -- No Need for Crossover Cables or Uplink Ports
• Wireless Security with WEP or WPA Encryption
• Enhanced Security Management Functions: Anonymous Internet Requests, Web, MAC Address and IP Address Filtering, and NAT Technology
• Access Your Corporate Network Remotely through Virtual Private Networking (VPN) -- Supports IPSec and PPTP Pass-Thru
• Easily Configurable through a Setup Wizard or Web Browser
• Supports QoS (Quality of Service)
• Supports Static and Dynamic Routing (RIP)

Specifications

Model Number: WRT55AG

Standards: IEEE 802.11a, IEEE 802.11b, IEEE 802.11g, IEEE 802.11n

Forts: Internet, One 10/100 RJ-45 Port for Cable/DSL Modem LAN, Four 10/100 RJ-45 Switched Ports, One Power Port

Buttons: One Reset Button

Cabling Type: UTP Cat 5 Ethernet cable or better

LEDs: Power, DMZ, Internet, Ethernet (1, 2, 3, 4)

Security Features: Internet Policy, Wireless Filters, Port Filters

Encryption Key Bits: WEP and WPA, 64 and 128

Environmental

Dimensions: 7.32" x 1.89" x 6.89" (186 mm x 48 mm x 175 mm)

Unit Weight: 0.88 lbs (0.40 kg)

Power: External, 5V DC, 2.5A

Certifications: FCC

Operating Temp.: 32°F to 104°F (0°C to 40°C)

Storage Temp.: -4°F to 158°F (-20°C to 70°C)

Operating Humidity: 10% to 85% Non-Condensing

Storage Humidity: 6% to 90% Non-Condensing

Warranty: 3-Year Limited

Minimum Requirements

• One 208 MHz or Faster Processor
• 64 MB RAM
• Internet Explorer 5.5 or Netscape Navigator 6.1 or higher for web-based configuration
• CD-ROM Drive
• 802.11a, 802.11b, or 802.11g Wireless Adapter with TCP/IP Protocol installed or Network Adapter with TCP/IP Protocol installed and Category 5 Ethernet Network Cable

Package Contents

• One Dual-Band Wireless A+G Broadband Router
• One Setup CD-ROM Setup CD-ROM with Symantec Internet Security
• User Guide on CD-ROM
• One Power Adapter
• One Ethernet Network Cable
• One Quick Installation
• One Registration Card

Product Data

Model No. WRT55AG
CPU-1433 Single Board Computer Datasheet

CPU-1433

Geode GX466 333MHz PC104+ SBC with CRT/LCD, 4x USB 2.0 and Fast Ethernet

FEATURES

Architecture:
PC PCI-Architecture with ISA-bus (PCI/104-Plus)

Processor and Companion Chipset:
AMD geode GX466 333 MHz; AMD CBO58800 Chipset

RAM:
128MB SDRAM DDR (soldered for shock/vibe resilience)

Solid State Disk:
IDE Connector for DiskOnModule, Compact Flash or ATA-Flash

Audio:
AC97 v 2.3 Sound Port (CODEC Adapter Available)

OS Compatibility:
Windows CE, Windows XPe, VXWorks, Linux, QNX

Video:
VGA-LVDS CRT and LCD interface
Supports up to 1800 x 1200 x 24 bpp at 65Hz

Serial Ports:
1 RS232 and 1 software configurable RS232/422/485 port

Mouse & Keyboard:
PS/2 Interface, PC/AT Keyboard Interface

Ethernet:
Realtek RL8139DL Ethernet Controller - 10/100 Mbps with
Auto-negotiation (RJ-45 Ethernet Adapter Available)

USB:
4 USB 2.0 ports

Parallel:
Enhanced Parallel port (EPP/1.9 and ECP/EPP compliant)

IDE:
ATA-6 IDE controller (for up to 2 IDE devices)

DESCRIPTION

The CPU-1433 is a fully RoHS compliant replacement for the popular CPU-1432 PC104-Plus module. Now featuring the AMD Geode GX466 333MHz Pentium® MMX class processor, this low-power Single Board Computer offers compatibility with Windows CE, XP Embedded, VXWorks, Linux and QNX 6.x operating systems together with significant performance gains compared to the QNX without sacrificing low power consumption.

Ideal applications for the CPU-1433 include mobile computer systems, real-time industrial control, process automation, digital video and multimedia data acquisition, in-vehicle electronics networking and wireless communications. Integrated peripheral interfaces include: 4 high speed USB2.0 ports, TFT & CRT display interfaces, two serial ports, 10/100Mbit Ethernet, IDE, AC97 audio, floppy, AT-keyboard and PS/2-mouse. Onboard features include a Real Time Clock and Watch Dog timer and non-volatile Setup storage.

A wide operating temperature range is achieved with a low power design that allows structural passive heat dissipation of the system. The CPU-1433 is available in standard (0°C to +85°C) and extended (-40°C to +85°C) temperature ranges. Onboard soldered 128MB of DDR memory improves system reliability in mobile installations where severe shock and vibration is common. Mass storage can be implemented using a solid state Flash DOM (DiskOnModule) extending the usability of this module in applications requiring secure data storage and improved storage reliability in harsh environmental conditions.

Parvus Corporation
3222 South Washington Street
Salt Lake City, UT 84115 USA
Web: www.parvus.com
Email: sales@parvus.com
Phone: (801) 483 1533
Fax: (801) 483 1523

Member of Eurotech Group
SPECIFICATIONS

Dimensions:
90 x 90 mm (3.6" x 3.6"), height 15mm (0.6")

Operating Temperature:
0°C to +60°C (Standard); -40°C to +85°C (Extended)

Humidity:
Up to 95%, Non-Condensing

Power Supply:
Single +5Vdc

Power Consumption:
Less than 8 Watts

Humidity:
Up to 95%, Non-Condensing

Eurotech Embedded BIOS
- Onboard 1MB Flash-memory used for the BIOS and its extensions (part can be also used as a Read-Only Disk, to store the OS as well as user programs/data)
- SSD supports Read or R/W disk
- Watch Dog management
- Non-volatile setup configuration data storage in Flash.
- Virtual Peripheral (VP) support: allows users to redirect peripheral devices used by the module such as keyboard, display and boot device to a remote host computer.

Development Kit:
DTKs include onboard development station with power supply, hard drive, CD-ROM drive, 3.5" Floppy drive and motherbaord that includes serial, parallel, and PCI ports plus adapters to standard PCI and ISA bays. A PC/104 single board computer (SBC) module with standard utility cable set and accessory cables come full assembled on the base station to allow easy and immediate use of all of the features of the CPU, such as Ethernet and USB. Simply connect power, monitor, keyboard and mouse, and the unit is ready for use.

ORDERING INFORMATION

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU-1433-44</td>
<td>CPU-1433, PC104+ SBC, GEODE GX466@333MHz, 128MB DRAM, -40/+85°C</td>
</tr>
<tr>
<td>CPU-1433-04</td>
<td>CPU-1433, PC104+ SBC, GEODE GX466@333MHz, 128MB DRAM, 0/+60°C</td>
</tr>
<tr>
<td>ACS-6092-00</td>
<td>4x USB A Connector Adapter for CPU-1233/1433/145X/146X/1651</td>
</tr>
<tr>
<td>CBL-1433-00</td>
<td>Cable Set - CPU-1433</td>
</tr>
<tr>
<td>DTK-1433-99</td>
<td>Development Kit - CPU-1433 (includes PC104 Dev Station, CPU, Cables, Adapters)</td>
</tr>
<tr>
<td>ACS-6095-05</td>
<td>RJ-45 10/100 Ethernet Adapter for CPU Modules</td>
</tr>
<tr>
<td>ACS-6093-01</td>
<td>AC97 Audio CODEC for CPU-1433/145X/146X/1651</td>
</tr>
</tbody>
</table>
Inertial Measurement Unit Schematic
Hokuyo URG-04LX LADAR Datasheet

NEW PRODUCTS

SCANNING LASER RANGE FINDER FOR ROBOTICS

URG-04LX

The Next Generation Area Detecting Sensor

Compact Design!
L:50, W:50, H:70mm
Light Weight!
160g
Lower Power Consumption!
2.5W
High Accuracy!
±10mm
High Resolution!
0.36°
Wide Scanning Area
240°
Low Cost!

Scanning example

Specific area (A) Two poles

URG

URG is scanning the specific area (A) and is detecting two poles and it shows distance and direction of two poles.

Applications:
* Robot eyes, Security, Automatic doors, AGV (obstacle detecting sensor)
### Specifications

<table>
<thead>
<tr>
<th>Products</th>
<th>Scanning Laser Range Finder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model No.</td>
<td>URG-04LX</td>
</tr>
<tr>
<td>Light source</td>
<td>Semiconductor laser, λ=890nm Laser safety class 1 (IEC-60825-1)</td>
</tr>
<tr>
<td>Power source</td>
<td>5V DC, ±10%</td>
</tr>
<tr>
<td>Current consumption</td>
<td>500mA or less (with a current 800mA)</td>
</tr>
<tr>
<td>Detectable distance and objects</td>
<td>2mm to 4000mm, white Kast sheet 30mm x 70mm</td>
</tr>
<tr>
<td>Accuracy</td>
<td>(Offical 10 to 1000mm: ±10mm, 1000 to 4000mm: ±1% of measurement) (White Kast sheet 30mm x 70mm)</td>
</tr>
<tr>
<td>Resolution</td>
<td>1mm</td>
</tr>
<tr>
<td>Scanning angle</td>
<td>240°</td>
</tr>
<tr>
<td>Angle resolution</td>
<td>Approx. 0.05° (360°, 1024 steps)</td>
</tr>
<tr>
<td>Scanning time</td>
<td>100ms/scan</td>
</tr>
<tr>
<td>Interface</td>
<td>RS-232C/10kΩ, 5Vdc, 115.2k, 500k, 75kbps, USB, Ethernet (DC12V)</td>
</tr>
<tr>
<td>Ambient temperature/ humidity</td>
<td>-10 to 50°C, 20% RH (for operation and storage)</td>
</tr>
<tr>
<td>Protective structure</td>
<td>Optics: IP64, Case: IP40</td>
</tr>
<tr>
<td>Weight</td>
<td>Approx. 100g</td>
</tr>
<tr>
<td>Material</td>
<td>Polycarbonate</td>
</tr>
</tbody>
</table>

### Interface

<table>
<thead>
<tr>
<th>Interface</th>
<th>URG-04LX</th>
<th>Cable color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NC</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NC</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>OUTPUT/Synchronous output</td>
<td>Black</td>
</tr>
<tr>
<td>4</td>
<td>GND (5-pin, D-sub connector 2 pins)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Vcc (5-pin, D-sub connector 2 pins)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>TH (5-pin, D-sub connector 2 pins)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0V</td>
<td>Blue</td>
</tr>
<tr>
<td>8</td>
<td>5V DC</td>
<td>Brown</td>
</tr>
</tbody>
</table>

### Exception clause

* This sensor is not a safety instrument tool.
* This sensor is designed for indoor use only.
* This sensor is not for use in military application.
* Read this specifications sheet before using.

---

**PHOTO SENSOR, LASER SENSOR, MICROWAVE SENSOR, COUNTER, AUTOMATIC DOOR**

---

**HOKUYO AUTOMATIC CO., LTD.**
1-10-9 Yusa, Yodogawa-ku, Osaka
532-0031, Japan
Tel(06)694-2102 Fax(06)694-2239
[http://www.hokuyo-aut.co.jp](http://www.hokuyo-aut.co.jp)
E-mail info@hokuyo-aut.jp

---

No.FR-12
FLIR ThermoVision A10 Camera Datasheet

The ThermoVision™ A10 infrared camera is the world's smallest infrared camera, delivering high performance thermal imaging for remote integration, VxOx microbolometer detector, on-focal-plane signal processing, and unparalleled thermal resolution.

Small Size Without Compromise
The tiny A10 infrared camera delivers performance typically found in more expensive systems—ideal for stand-alone imaging applications or as an OEM core—where space, weight, and power constraints are concerns. Versatility of placement, low power consumption, low heat load, and long battery life add to the ease of integration, making the A10 the perfect solution for firefighting, civil surveillance, search and rescue, industrial processes and military applications.

SmartScene™ Video Output
The A10 provides analog video (RS-170 or CIR) output and employs a unique SmartScene feature to maximize picture quality in virtually every scene. SmartScene uses a dynamic, non-linear conversion to process the 14-bit digital image data into the 8-bit domain of analog video. This conversion algorithm is automatically adjusted, frame-by-frame, to maximize the contrast in darker (darker) parts of the frame, while avoiding “washout” of brighter (darker) objects in the image frame. The result is a continually optimized image independent of scene dynamics, all without operator intervention or concern.

Vertically Integrated Design
The world's smallest, the A10 is a direct result of FLIR’s vertical integration. No design or fabrication of critical camera components, including the ROC, chip, focal-plane array, and other processing circuitry, as well as the value-added modular accessories, is left to others. This vertical integration of technology and engineering is a result of FLIR's commitment to excellence in infrared technology.

14-bit Processing for Excellent Dynamic Range
The A10 delivers a wide dynamic range (14-bit) image at real-time video rates (30 frames per second for RS-170 and 25 fps for CIR). An auto-ranging function detects very hot scenes and automatically switches into an extended temperature range mode, allowing imaging scenes up to 400°C. The internal shutter periodically recalibrates the camera automatically or can be manually overridden, which is important for process-monitoring applications. All camera functions can be optimized through a serial interface (RS-232) command set. All settings are stored in non-volatile memory to enable pre-programmed use.

Multiple Programming Options
The A10 output can be easily leveraged to control a process when coupled with LabVIEW and FLIR’s LabVIEW Developers Toolkit. This kit allows programmers to easily access functions that can then be used to turn the A10 into a powerful machine vision tool with minimal investment in machine vision software development. Alternatively, you can work in your own programming environment with the ThermoVision™ System Developer Kit (SDK), which is based on ActiveX and Visual Basic C++.

Wide Range of Operating Temperatures
The A10 features a proprietary image optimization system that pre-processes image data and eliminates the need for temperature stabilization of the array. This enables the A10 to operate over a wide ambient temperature range without the complexity and power consumption of a thermoelectric cooler. The absence of a TE cooler also yields an “instant on” turn-on time, which can be critical in “point-and-shoot” applications such as firefighting and security.

DCAM Compatible
The ThermoVision A10 has a Vision Interface that is DCAM compatible and works with third party hardware, such as the National Instruments Compact Vision System (CVS).

High Sensitivity Detection
Real-time algorithms combine with a fully optimized 160 x 128 focal plane array to deliver image quality that is visually comparable to larger array. Moreover by using the InSb vanadium oxide (VOx) technology, the A10 delivers a much higher performance than older detector technologies. As a result, the A10 can utilize higher penetration optics, minimizing both weight and cost, yet retains such low noise levels that even in low light, the A10 delivers NETD of <0.05 mK and uses an optional 16 bit lens, this drops to a 0.01 mK. -factor of 5 better than competing technologies.
# ThermoVision™ A10 Technical Specifications

## Camera Specifications

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detector</td>
<td>Uncooled microbolometer</td>
</tr>
<tr>
<td>Acronym</td>
<td>100 Hz, 1.25 Hz [97.7 Hz display], 50 Hz, 1.25 Hz [52.0 Hz display]</td>
</tr>
<tr>
<td>Pixelate</td>
<td>32 x 32 pixels</td>
</tr>
<tr>
<td>Spectral Response</td>
<td>-5 to 15.5 μm</td>
</tr>
<tr>
<td>Temperature Specifications</td>
<td>Thermo-electric</td>
</tr>
<tr>
<td>Cooling Method</td>
<td>Conduction to camera bottom</td>
</tr>
<tr>
<td>Interface</td>
<td>1 melt ice inserts; camera base, guide pin, hole</td>
</tr>
<tr>
<td>Dimensions</td>
<td>3.85W x 2.49W x 1.010 inches</td>
</tr>
<tr>
<td>Weight</td>
<td>120 g, lens dependent</td>
</tr>
<tr>
<td>Power</td>
<td>&lt; 1.5 W maximum</td>
</tr>
<tr>
<td>Input/Output</td>
<td>30 pins accessible for video, power, communications, digital data</td>
</tr>
<tr>
<td>Software</td>
<td>RS-232 interface</td>
</tr>
<tr>
<td>Digital Video</td>
<td>Optional real-time, 1440p, greyscale and normal modes</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>-40°C to 55°C, standard; -40°C to +55°C, extended temp range available</td>
</tr>
<tr>
<td>Remote Control Operate</td>
<td>12°C to 55°C standard; optional auto-gain mode extend range to 96°C</td>
</tr>
<tr>
<td>Repeatability</td>
<td>1% non-condensing</td>
</tr>
</tbody>
</table>

*Specifications are subject to slight modification. Please contact FLIR Systems for the most current specifications.*

## Option

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optic material</td>
<td>Germanium</td>
</tr>
<tr>
<td>Frame Rate</td>
<td>14, 15 options</td>
</tr>
<tr>
<td>Lens focal length</td>
<td>11mm, 18mm, 35mm</td>
</tr>
<tr>
<td>Field of View (degrees)</td>
<td>46°×33, 28°×19, 14°×11, respectively</td>
</tr>
<tr>
<td>Field of View (millimeters)</td>
<td>454, 330, 170, respectively</td>
</tr>
</tbody>
</table>

## Accessories

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hot shoe Interface</td>
<td>Converts the A10 into a handheld infrared video for Sony camcorders</td>
</tr>
<tr>
<td>Battery module</td>
<td>Provides &gt; 2 hours of cordless camera operation</td>
</tr>
<tr>
<td>FLIRware™ (IEEE 1394) Output module</td>
<td>Allows FLIRware™ to acquire digital video from the A10 without a framegrabber</td>
</tr>
<tr>
<td>HMD@P™ attachment</td>
<td>Monitors vital parameters without real access to all camera functions</td>
</tr>
<tr>
<td>Environmental enclosure</td>
<td>Allows the A10 to be used in harsh, abusive conditions</td>
</tr>
</tbody>
</table>

*Note: accessories are always under development.*

---

1 800 464 6372  
www.flirthermography.com/A10data  
The Global Leader in Infrared Cameras  
Specifications subject to change. © Copyright 2005 FLIR Systems, Inc. All rights reserved. HS3905PL
Zerolux PC209IR Camera Datasheet

DAY-0.01 LUX COLOR/NIGHT-0.00 LUX NIGHT VISION BY 7-IR ILLUMINATOR COLOR CAMERA
MICRO BULLET WEATHER-PROOF METAL CASE TO USE INDOOR/OUTDOOR

MODEL: PC209IR

* BUILT IN CMOS SENSOR TO ON-OFF OF IR-ILLUMINATOR ABOUT 10.0 LUX AUTOMATICALLY

1. FEATURES
- 1/3" SONY SUPER-HAD COLOR CCD, 400TVL
- DAY-0.01 LUX COLOR/NIGHT-0.00 LUX NIGHT VISION BY 7-IR ILLUMINATOR
- MORE THAN 5M (15FT) CLEAN IMAGE PICK-UP DISTANCE AT NIGHT TIME.
- BUILT IN CMOS SENSOR CIRCUITS TO ON-OFF OF ILLUMINATOR ABOUT 10.0 LUX AUTOMATICALLY
- MICRO BULLET WEATHER-PROOF METAL CASE TO USE INDOOR/OUTDOOR
- LOW POWER CONSUMPTIONS: DC12V/0.2AMP.
- INCLUDED AC/DC ADAPTOR, MOUNTING BRACKET.

2. SPECIFICATIONS

<table>
<thead>
<tr>
<th>MODEL NO.</th>
<th>PC209IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE SENSOR</td>
<td>1/3&quot; SONY SUPER-HAD COLOR CCD</td>
</tr>
<tr>
<td>FILTER ON CCD</td>
<td>STANDARD</td>
</tr>
<tr>
<td>RESOLUTIONS</td>
<td>400 TVL</td>
</tr>
<tr>
<td>BUILT IN LENS</td>
<td>4.3mm</td>
</tr>
<tr>
<td>MIN. ILLUMINATION</td>
<td>0.01 LUX/DAY, 0.00 LUX/NIGHT BY 7-IR ILLUMINATOR</td>
</tr>
<tr>
<td>POWER SUPPLY</td>
<td>DC12V/0.2AMP</td>
</tr>
<tr>
<td>INSTALLATION</td>
<td>INDOOR/OUTDOOR</td>
</tr>
<tr>
<td>DIMENSIONS</td>
<td>Ø 30-97mm (1.18&quot; - 3.81&quot;)</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>90 gms</td>
</tr>
</tbody>
</table>
Sensoray Model 314 Video/Audio Grabber Datasheet

SENSORAY CO., INC.

PC/104+ MPEG Video/Audio Grabber

Model 314 Rev.A

May 8, 2006

© Sensoray 2006
7313 SW Tech. Center Dr.
Tigard, OR 97223
Phone 503.684.8005 • Fax 503.684.8164
www.sensoray.com
# Table of Contents

LIMITED WARRANTY ................................................................. 3
SPECIAL HANDLING INSTRUCTIONS ........................................... 4
INTRODUCTION ......................................................................... 5
  Feature Summary ................................................................. 5
REFERENCE ............................................................................... 6
  Connectors ........................................................................ 6
    PC/104+ Bus Connector, CON2 ........................................ 7
    PC/104 Bus Connector AB, CON1 ....................................... 8
    PC/104 Bus Connector CD, CON3 ....................................... 9
    A/V (Audio and Video) in and out Connector, J4 ................... 9
  LED .................................................................................. 10
    Power-OK indicator, D8 .................................................... 10
  Configuration DIP Switches ............................................... 10
    A/V routing DIP switch, SW1 .......................................... 10
    PCI slot # select and interrupt routing DIP switch, SW2 ....... 10
  Device Driver and SDK ...................................................... 11
    Windows ....................................................................... 11
    Linux ........................................................................... 11
    Specifications ................................................................. 11
APPENDIX A: TESTED 314 FEATURES ........................................ 13
APPENDIX B: PLAYER COMPATIBILITY AND INTEROPERABILITY .... 14
Limited warranty

Sensoray Company, Incorporated (Sensoray) warrants the hardware to be free from defects in material and workmanship and perform to applicable published Sensoray specifications for two years from the date of shipment to purchaser. Sensoray will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The warranty provided herein does not cover equipment subjected to abuse, misuse, accident, alteration, neglect, or unauthorized repair or installation. Sensoray shall have the right of final determination as to the existence and cause of defect.

As for items repaired or replaced under warranty, the warranty shall continue in effect for the remainder of the original warranty period, or for ninety days following date of shipment by Sensoray of the repaired or replaced part, whichever period is longer.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. Sensoray will pay the shipping costs of returning to the owner parts that are covered by warranty. A restocking charge of 25% of the product purchase price will be charged for returning a product to stock.

Sensoray believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, Sensoray reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult Sensoray if errors are suspected. In no event shall Sensoray be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, SENSORAY MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF SENSORAY SHALL BE LIMITED TO THE AMOUNT THEREFORE PAID BY THE CUSTOMER. SENSORAY WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEROF.

Third party brands, names and trademarks are the property of their respective owners.
Special handling instructions

The circuit board contains CMOS circuitry that is sensitive to Electrostatic Discharge (ESD).

Special care should be taken in handling, transporting, and installing circuit board to prevent ESD damage to the board. In particular:

- Do not remove the circuit board from its protective anti-static bag until you are ready to install the board into the enclosure.
- Handle the circuit board only at grounded, ESD protected stations.
- Remove power from the equipment before installing or removing the circuit board.
Introduction

Model 314 is a PC/104+ MPEG video and/or audio grabber. It captures video and/or audio from a standard NTSC/PAL analog video source optional with stereo or monochrome audio source into one of following compressed MPEG streams: MPEG1, MPEG2, MPEG4, H.263, or MJPEG. It also supports capturing raw frames from NTSC/PAL video source. The capturing frame rate is up to 30 fps for NTSC and 25 fps for PAL.

A single +5V input power supply is required to power the board, through PC/104+ connector.

Feature Summary

- PC/104+-based MPEG video/audio grabber (capturing module)
- Raw frame grabbing or previewing feature support
- Video input: 4 multiplexed input channels (2 S-Video or 4 Composite)
- Audio input: 1 pair of stereo or 1 mono
- Resolution (Max):
  - Full-D1: NTSC: 720 x 480 @ 30 fps or 720 x 240 @ 60 fps
  - PAL: 720 x 576 @ 25 fps or 720 x 288 @ 50 fps
- Other supported video resolutions:
  - D1:N: 720 x 480
  - D1:P: 720 x 576
  - D5: 480 x 352
  - SIF: 352 x 240
  - 25IF: 704 x 240
  - 4SIF: 704 x 480
  - VGA: 640 x 480
  - QVGA: 320 x 240
  - GQVGA: 160 x 112
  - CIF: 352 x 288
  - QCIF: 176 x 144
  - SQCIF: 128 x 96
  - 4CIF: 704 x 576
- Video encoding formats:
  - MPEG-4 SP@LL1, plus B-frame support, progressive and interlace,
  - Microsoft, DivX, Sigma Design compatible
  - MPEG-2 MP@ML, progressive and interlace
  - MPEG-1
  - H.263
  - MJPEG (Motion JPEG)
- Bit-rate control: CBR/VBR, 1Kbps to 10 Mbps
- OSD (On-Screen Display): 96 characters, 16x16 pixel font, multi-window supported.
- Motion detection support
- Signal loss detection support
- 1 digital input and 1 digital output: TTL signals
- Driver and SDK for Linux and Windows
PC/104+ Bus Connector, CON2.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Ground</td>
<td>B1</td>
<td>n/c</td>
<td>C1</td>
<td>+5 V</td>
<td>D1</td>
<td>AD0</td>
</tr>
<tr>
<td>A2</td>
<td>VIO*</td>
<td>B2</td>
<td>AD2</td>
<td>C2</td>
<td>AD1</td>
<td>D2</td>
<td>+5 V</td>
</tr>
<tr>
<td>A3</td>
<td>Ground</td>
<td>B3</td>
<td>Ground</td>
<td>C3</td>
<td>AD4</td>
<td>D3</td>
<td>AD3</td>
</tr>
<tr>
<td>A4</td>
<td>C/BE0#</td>
<td>B4</td>
<td>AD7</td>
<td>C4</td>
<td>Ground</td>
<td>D4</td>
<td>AD6</td>
</tr>
<tr>
<td>A5</td>
<td>Ground</td>
<td>B5</td>
<td>ADS</td>
<td>C5</td>
<td>AD8</td>
<td>D5</td>
<td>Ground</td>
</tr>
<tr>
<td>A6</td>
<td>AD11</td>
<td>B6</td>
<td>VI0*</td>
<td>C6</td>
<td>AD10</td>
<td>D6</td>
<td>M66EN*</td>
</tr>
<tr>
<td>A7</td>
<td>AD14</td>
<td>B7</td>
<td>AD13</td>
<td>C7</td>
<td>Ground</td>
<td>D7</td>
<td>AD12</td>
</tr>
<tr>
<td>A9</td>
<td>n/c</td>
<td>B9</td>
<td>C/BE1#</td>
<td>C8</td>
<td>AD15</td>
<td>D8</td>
<td>n/c</td>
</tr>
<tr>
<td>A9</td>
<td>SERR#</td>
<td>B9</td>
<td>Ground</td>
<td>C9</td>
<td>SBO*</td>
<td>D9</td>
<td>PAR</td>
</tr>
<tr>
<td>A10</td>
<td>Ground</td>
<td>B10</td>
<td>PERR#</td>
<td>C10</td>
<td>n/c</td>
<td>D19</td>
<td>SDONE*</td>
</tr>
<tr>
<td>A11</td>
<td>STOP#</td>
<td>B11</td>
<td>n/c</td>
<td>C11</td>
<td>LOCK*</td>
<td>D11</td>
<td>Ground</td>
</tr>
<tr>
<td>A12</td>
<td>n/c</td>
<td>B12</td>
<td>TRDY#</td>
<td>C12</td>
<td>Ground</td>
<td>D12</td>
<td>DEVSEL*</td>
</tr>
<tr>
<td>A13</td>
<td>FRAME#</td>
<td>B13</td>
<td>Ground</td>
<td>C13</td>
<td>IROD1#</td>
<td>D13</td>
<td>n/c</td>
</tr>
<tr>
<td>A14</td>
<td>Ground</td>
<td>B14</td>
<td>AD16</td>
<td>C14</td>
<td>n/c</td>
<td>D14</td>
<td>C/BE2#</td>
</tr>
<tr>
<td>A15</td>
<td>AD18</td>
<td>B15</td>
<td>n/c</td>
<td>C15</td>
<td>AD17</td>
<td>D15</td>
<td>Ground</td>
</tr>
<tr>
<td>A16</td>
<td>AD21</td>
<td>B16</td>
<td>AD10</td>
<td>C16</td>
<td>Ground</td>
<td>D16</td>
<td>AD19</td>
</tr>
<tr>
<td>A17</td>
<td>n/c</td>
<td>B17</td>
<td>AD23</td>
<td>C17</td>
<td>AD22</td>
<td>D17</td>
<td>n/c</td>
</tr>
<tr>
<td>A18</td>
<td>IDSEL0</td>
<td>B18</td>
<td>Ground</td>
<td>C18</td>
<td>IDSEL1</td>
<td>D18</td>
<td>IDSEL2</td>
</tr>
<tr>
<td>A19</td>
<td>AD14</td>
<td>B19</td>
<td>C/BE1#</td>
<td>C19</td>
<td>VI0*</td>
<td>D19</td>
<td>IDSEL1*</td>
</tr>
<tr>
<td>A20</td>
<td>Ground</td>
<td>B20</td>
<td>AD16</td>
<td>C20</td>
<td>AD25</td>
<td>D20</td>
<td>Ground</td>
</tr>
<tr>
<td>A21</td>
<td>AD29</td>
<td>B21</td>
<td>+5 V</td>
<td>C21</td>
<td>AD28</td>
<td>D21</td>
<td>AD27</td>
</tr>
<tr>
<td>A22</td>
<td>+5 V</td>
<td>B22</td>
<td>AD10</td>
<td>C22</td>
<td>Ground</td>
<td>D22</td>
<td>AD31</td>
</tr>
<tr>
<td>A23</td>
<td>REQ#</td>
<td>B23</td>
<td>Ground</td>
<td>C23</td>
<td>REQ1#</td>
<td>D23</td>
<td>VIO*</td>
</tr>
<tr>
<td>A24</td>
<td>Ground</td>
<td>B24</td>
<td>REQ2#</td>
<td>C24</td>
<td>+5 V</td>
<td>D24</td>
<td>GNT0#</td>
</tr>
<tr>
<td>A25</td>
<td>GNT1#</td>
<td>B25</td>
<td>VI0*</td>
<td>C25</td>
<td>GNT2#</td>
<td>D25</td>
<td>Ground</td>
</tr>
<tr>
<td>A26</td>
<td>+5 V</td>
<td>B26</td>
<td>CL10</td>
<td>C26</td>
<td>Ground</td>
<td>D26</td>
<td>CLK1</td>
</tr>
<tr>
<td>A27</td>
<td>CLK2</td>
<td>B27</td>
<td>+5 V</td>
<td>C27</td>
<td>CLK3*</td>
<td>D27</td>
<td>Ground</td>
</tr>
<tr>
<td>A28</td>
<td>Ground</td>
<td>B28</td>
<td>INTD#*</td>
<td>C28</td>
<td>+5 V</td>
<td>D28</td>
<td>RST#</td>
</tr>
<tr>
<td>A29</td>
<td>+12 V</td>
<td>B29</td>
<td>INTA#</td>
<td>C29</td>
<td>INTB#</td>
<td>D29</td>
<td>INTC#</td>
</tr>
<tr>
<td>A30</td>
<td>n/c</td>
<td>B30</td>
<td>n/c</td>
<td>C30</td>
<td>n/c</td>
<td>D30</td>
<td>Ground</td>
</tr>
</tbody>
</table>

Note:

* not connected.

n/c not connected.
**PC/104 Bus Connector AB, CON1.**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>n/c</td>
<td>B1</td>
<td>Ground ¥</td>
</tr>
<tr>
<td>A2</td>
<td>SD7</td>
<td>B2</td>
<td>RSTDRY</td>
</tr>
<tr>
<td>A3</td>
<td>SD6</td>
<td>B3</td>
<td>+5 V</td>
</tr>
<tr>
<td>A4</td>
<td>SD5</td>
<td>B4</td>
<td>IR09</td>
</tr>
<tr>
<td>A5</td>
<td>SD4</td>
<td>B5</td>
<td>-5 V</td>
</tr>
<tr>
<td>A6</td>
<td>SD3</td>
<td>B6</td>
<td>DRO2</td>
</tr>
<tr>
<td>A7</td>
<td>SD2</td>
<td>B7</td>
<td>-12 V</td>
</tr>
<tr>
<td>A8</td>
<td>SD1</td>
<td>B8</td>
<td>n/c</td>
</tr>
<tr>
<td>A9</td>
<td>SD0</td>
<td>B9</td>
<td>+12 V</td>
</tr>
<tr>
<td>A10</td>
<td>J0CHRDY</td>
<td>B10</td>
<td>n/c</td>
</tr>
<tr>
<td>A11</td>
<td>AEN</td>
<td>B11</td>
<td>n/c</td>
</tr>
<tr>
<td>A12</td>
<td>SA19</td>
<td>B12</td>
<td>n/c</td>
</tr>
<tr>
<td>A13</td>
<td>SA18</td>
<td>B13</td>
<td>IOW#</td>
</tr>
<tr>
<td>A14</td>
<td>SA17</td>
<td>B14</td>
<td>DO#</td>
</tr>
<tr>
<td>A15</td>
<td>SA16</td>
<td>B15</td>
<td>DAO3#</td>
</tr>
<tr>
<td>A16</td>
<td>SA15</td>
<td>B16</td>
<td>DR03</td>
</tr>
<tr>
<td>A17</td>
<td>SA14</td>
<td>B17</td>
<td>n/c</td>
</tr>
<tr>
<td>A18</td>
<td>SA13</td>
<td>B18</td>
<td>n/c</td>
</tr>
<tr>
<td>A19</td>
<td>SA12</td>
<td>B19</td>
<td>REFRESH#</td>
</tr>
<tr>
<td>A20</td>
<td>SA11</td>
<td>B20</td>
<td>n/c</td>
</tr>
<tr>
<td>A21</td>
<td>SA10</td>
<td>B21</td>
<td>IR07</td>
</tr>
<tr>
<td>A22</td>
<td>SA9</td>
<td>B22</td>
<td>IR06</td>
</tr>
<tr>
<td>A23</td>
<td>SA8</td>
<td>B23</td>
<td>IR05</td>
</tr>
<tr>
<td>A24</td>
<td>SA7</td>
<td>B24</td>
<td>IR04</td>
</tr>
<tr>
<td>A25</td>
<td>SA6</td>
<td>B25</td>
<td>IR03</td>
</tr>
<tr>
<td>A26</td>
<td>SA5</td>
<td>B26</td>
<td>DAO2#</td>
</tr>
<tr>
<td>A27</td>
<td>SA4</td>
<td>B27</td>
<td>TO</td>
</tr>
<tr>
<td>A28</td>
<td>SA3</td>
<td>B28</td>
<td>BALE</td>
</tr>
<tr>
<td>A29</td>
<td>SA2</td>
<td>B29</td>
<td>+5 V</td>
</tr>
<tr>
<td>A30</td>
<td>SA1</td>
<td>B30</td>
<td>OSC</td>
</tr>
<tr>
<td>A31</td>
<td>SA0</td>
<td>B31</td>
<td>Ground ¥</td>
</tr>
<tr>
<td>A32</td>
<td>Ground ¥</td>
<td>B32</td>
<td>Ground ¥</td>
</tr>
</tbody>
</table>

**Note:**

¥ Only ground connected; all other pins are not connected.
PC/104 Bus Connector CD, CON3.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>Ground *</td>
<td>D0</td>
<td>Ground *</td>
</tr>
<tr>
<td>C1</td>
<td>SBHE#</td>
<td>D1</td>
<td>MEMCS16#</td>
</tr>
<tr>
<td>C2</td>
<td>LA23</td>
<td>D2</td>
<td>I/OCS16#</td>
</tr>
<tr>
<td>C3</td>
<td>LA22</td>
<td>D3</td>
<td>IRO10</td>
</tr>
<tr>
<td>C4</td>
<td>LA21</td>
<td>D4</td>
<td>IRO11</td>
</tr>
<tr>
<td>C5</td>
<td>LA20</td>
<td>D5</td>
<td>IRO12</td>
</tr>
<tr>
<td>C6</td>
<td>LA19</td>
<td>D6</td>
<td>IRO15</td>
</tr>
<tr>
<td>C7</td>
<td>LA18</td>
<td>D7</td>
<td>IRO14</td>
</tr>
<tr>
<td>C8</td>
<td>LA17</td>
<td>D8</td>
<td>n/c</td>
</tr>
<tr>
<td>C9</td>
<td>MEMR#</td>
<td>D9</td>
<td>n/c</td>
</tr>
<tr>
<td>C10</td>
<td>MEMW#</td>
<td>D10</td>
<td>DACK5#</td>
</tr>
<tr>
<td>C11</td>
<td>SD8</td>
<td>D11</td>
<td>DRQ5</td>
</tr>
<tr>
<td>C12</td>
<td>SD9</td>
<td>D12</td>
<td>DACK6#</td>
</tr>
<tr>
<td>C13</td>
<td>SD10</td>
<td>D13</td>
<td>DRO6</td>
</tr>
<tr>
<td>C14</td>
<td>SD11</td>
<td>D14</td>
<td>DACK7#</td>
</tr>
<tr>
<td>C15</td>
<td>SD12</td>
<td>D15</td>
<td>DRQ7</td>
</tr>
<tr>
<td>C16</td>
<td>SD13</td>
<td>D16</td>
<td>+5 V</td>
</tr>
<tr>
<td>C17</td>
<td>SD14</td>
<td>D17</td>
<td>n/c</td>
</tr>
<tr>
<td>C18</td>
<td>SD15</td>
<td>D18</td>
<td>Ground *</td>
</tr>
<tr>
<td>C19</td>
<td>n/c</td>
<td>D19</td>
<td>Ground *</td>
</tr>
</tbody>
</table>

Note:
* Only ground connected; all other pins are not connected

AV/Video in and out Connector, J4.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Digital I/O 1</td>
<td>2</td>
<td>+5 V</td>
</tr>
<tr>
<td>3</td>
<td>Digital I/O 0</td>
<td>4</td>
<td>Digital ground</td>
</tr>
<tr>
<td>5</td>
<td>Audio Ground</td>
<td>6</td>
<td>Audio in - R</td>
</tr>
<tr>
<td>7</td>
<td>Audio in - L</td>
<td>8</td>
<td>Audio out - R</td>
</tr>
<tr>
<td>9</td>
<td>Audio out - L</td>
<td>10</td>
<td>Audio Ground</td>
</tr>
<tr>
<td>11</td>
<td>Composite video out 3 / S-Video 1 - C</td>
<td>12</td>
<td>Video Ground</td>
</tr>
<tr>
<td>13</td>
<td>Composite video out 2 (for loopback)</td>
<td>14</td>
<td>Video Ground</td>
</tr>
<tr>
<td>15</td>
<td>Composite video out 1 / S-Video 1 - Y</td>
<td>16</td>
<td>Video Ground</td>
</tr>
<tr>
<td>17</td>
<td>Composite video in 4 / S-Video 2 - C</td>
<td>18</td>
<td>Video Ground</td>
</tr>
<tr>
<td>19</td>
<td>Composite video in 3 / S-Video 1 - C</td>
<td>20</td>
<td>Video Ground</td>
</tr>
<tr>
<td>21</td>
<td>Composite video in 2 / S-Video 2 - Y</td>
<td>22</td>
<td>Video Ground</td>
</tr>
<tr>
<td>23</td>
<td>Composite video in 1 / S-Video 1 - Y</td>
<td>24</td>
<td>Video Ground</td>
</tr>
</tbody>
</table>
LED

Power-OK indicator, D6.

The LED D6 is used for indicating on-board Power-OK status.

Configuration DIP Switches

A/V routing DIP switch, SW1.

The DIP switch, SW1, is used for choosing audio input routing and video direct loopback routing. Refer to the following chart to select the A/V routing that you prefer:

<table>
<thead>
<tr>
<th>DIP</th>
<th>ON (down)</th>
<th>OFF (up)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW1-5</td>
<td>J4-pin6 (AIN-R) (\Rightarrow) J1-R, and (\Rightarrow) both of Audio-in Right1 and Right2 on PCI A/V decoder</td>
<td>Audio-in Right1 (\Rightarrow) J4-pin6, and Audio-in Right2 (\Rightarrow) StereoJack J1-R</td>
</tr>
<tr>
<td>SW1-4</td>
<td>J4-pin6 (AIN-L) (\Rightarrow) J1-L, and (\Rightarrow) both of Audio-in Left1 and Left2 on PCI A/V decoder</td>
<td>Audio-in Left1 (\Rightarrow) J4-pin7, and Audio-in Left2 (\Rightarrow) StereoJack J1-L</td>
</tr>
<tr>
<td>SW1-3</td>
<td>Composite video out 1 (\Rightarrow) Composite video in 1 or S-Video out 1 (\Rightarrow) S-Video in 1 (\Rightarrow) No direct loopback (i\rightarrow v\rightarrow 1) or (s\rightarrow s\rightarrow 1)</td>
<td></td>
</tr>
<tr>
<td>SW1-2</td>
<td>Composite video out 2 (\Rightarrow) Composite video in 1 or S-Video out 1 (\Rightarrow) S-Video in 1 (\Rightarrow) No direct loopback (i\rightarrow v\rightarrow 2) or (s\rightarrow s\rightarrow 2)</td>
<td></td>
</tr>
<tr>
<td>SW1-1</td>
<td>Composite video out 3 (\Rightarrow) Composite video in 1</td>
<td>No direct loopback (i\rightarrow v\rightarrow 3)</td>
</tr>
</tbody>
</table>

Note: \(\Rightarrow\) presents connected; \(\rightarrow\) presents "take from"; \(\Rightarrow\) presents "pass to".

PCI slot # select and interrupt routing DIP switch, SW2.

The DIP switch, SW2, is used for selecting PCI slot number and interrupt routing for the Model 314.

Refer to the following tables to choose a right/preferred setting for Model 314 in a (your) PC104+ stack:

<table>
<thead>
<tr>
<th>SW2-2 (SelB)</th>
<th>SW2-1 (SelA)</th>
<th>PCI Slot #</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON (down)</td>
<td>ON (down)</td>
<td>0</td>
</tr>
<tr>
<td>ON (down)</td>
<td>OFF (up)</td>
<td>1</td>
</tr>
<tr>
<td>OFF (up)</td>
<td>ON (down)</td>
<td>2</td>
</tr>
<tr>
<td>OFF (up)</td>
<td>OFF (up)</td>
<td>3</td>
</tr>
</tbody>
</table>
### Device Driver and SDK

Device driver and SDK including demo application program are available for both Windows and Linux.

**Windows**

Sensory Co. provides 314 WDM driver and DirectX filter for Windows platform. All are packaged in 314-Win-SDK (coming soon).

**Linux**

Linux SDK package, 314-lnx-sdk, provides device drive, A/V capturing & streaming service library, and A/V capturing & streaming over IP demos, for Linux platform. Currently, kernel version 2.4.23 and/or above are supported.

To demonstrate the application, two server demos are enclosed in Linux SDK, 314-lnx-sdk:

- The server cap-server is mainly used for demonstrating how to capture video and/or audio streams from 314 and save them into files (in a variety of supported formats including MPEG1, MPEG2, MPEG4 (.avi or .divx), H.263, MPEG (. avi), and even VOB, SVCD/VCID for A/V, and WAV/MP2 for audio only).
- The server str-server is used for demonstrating how to stream the live A/V over IP. Also, a variety of supported formats listed above is supported.
## Specifications

<table>
<thead>
<tr>
<th>Dimension</th>
<th>96mm x 105mm x 23mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>70 g</td>
</tr>
<tr>
<td>Power</td>
<td>+5V, 500mA</td>
</tr>
<tr>
<td>Bus</td>
<td>PC/104+</td>
</tr>
<tr>
<td>Video inputs</td>
<td>4 multiplexed input channels: 2 S-Video or 4 Composite, 75 Ohms</td>
</tr>
<tr>
<td>Video formats</td>
<td>NTSC and PAL</td>
</tr>
<tr>
<td>Video Encoding formats</td>
<td>MPEG1, MPEG2 (MP@ML), MPEG4 (SP@L3 + B-frame support), H.263, and MJPEG (Motion JPEG)</td>
</tr>
<tr>
<td>Resolution</td>
<td>Up to Full-D1:</td>
</tr>
<tr>
<td></td>
<td>NTSC: 720x480</td>
</tr>
<tr>
<td></td>
<td>PAL: 720x576</td>
</tr>
<tr>
<td></td>
<td>Supported:</td>
</tr>
<tr>
<td></td>
<td>D1.N: 720x480</td>
</tr>
<tr>
<td></td>
<td>D1.P: 720x576</td>
</tr>
<tr>
<td></td>
<td>D5: 480x352</td>
</tr>
<tr>
<td></td>
<td>2SIF: 704x240</td>
</tr>
<tr>
<td></td>
<td>4SIF: 704x480</td>
</tr>
<tr>
<td></td>
<td>VGA: 640x480</td>
</tr>
<tr>
<td></td>
<td>QVGA: 320x240</td>
</tr>
<tr>
<td></td>
<td>QVGA: 160x112</td>
</tr>
<tr>
<td></td>
<td>CIF: 352x288</td>
</tr>
<tr>
<td></td>
<td>QCIF: 176x144</td>
</tr>
<tr>
<td></td>
<td>SQCIF: 128x96</td>
</tr>
<tr>
<td></td>
<td>4CIF: 704x576</td>
</tr>
<tr>
<td>Capture rate</td>
<td>Up to: 30 frames/sec for NTSC/RS-170/CCIR</td>
</tr>
<tr>
<td></td>
<td>25 frames/sec for PAL</td>
</tr>
<tr>
<td>Bit-rate</td>
<td>CBR/VBR, 1 Kbps to 10 Mbps</td>
</tr>
<tr>
<td>OSD (On-Screen Display)</td>
<td>96 characters, 16x16 pixel font, multi-window supported</td>
</tr>
<tr>
<td>Video outputs</td>
<td>3 channels (only for direct loopback monitoring and testing)</td>
</tr>
<tr>
<td>Audio input &amp; output</td>
<td>Stereo or monochrome line in/out from/to connector or optional Jacks (output only for loopback testing), Signal level +/- 1.0 Volt</td>
</tr>
<tr>
<td>Digital I/O</td>
<td>1 input = 1 output, TTL signals</td>
</tr>
<tr>
<td>OS Platform</td>
<td>Linux and Windows</td>
</tr>
<tr>
<td>Temperature</td>
<td>0 – 70°C</td>
</tr>
</tbody>
</table>
### Appendix A: Tested 314 Features

#### Tested 314 Features:

<table>
<thead>
<tr>
<th>Feature</th>
<th>MPEG1</th>
<th>MPEG2</th>
<th>MPEG4 (DivX)</th>
<th>H.263</th>
<th>MPEG3</th>
<th>HP2</th>
<th>WAY</th>
<th>WDV</th>
<th>SD/NC/CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capturing AV to file</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>TEB</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Streaming over IP</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OD2</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>TEB</td>
<td>V</td>
<td>NA</td>
<td>NA</td>
<td>TEB</td>
<td>TEB</td>
</tr>
<tr>
<td>OTF Change FPS</td>
<td>V</td>
<td>V</td>
<td>TEB</td>
<td>TEB</td>
<td>TEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change Bit-rate</td>
<td>V</td>
<td>TEB</td>
<td>TEB</td>
<td>TEB</td>
<td>TEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brightness/Contrast Adjustment</td>
<td>V</td>
<td>V</td>
<td>S</td>
<td>/</td>
<td>S</td>
<td>NA</td>
<td>NA</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Hue/Saturation Adjustment</td>
<td>V</td>
<td>V</td>
<td>S</td>
<td>/</td>
<td>S</td>
<td>NA</td>
<td>NA</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Change Resolution</td>
<td>V</td>
<td>V</td>
<td>S</td>
<td>/</td>
<td>V</td>
<td>NA</td>
<td>NA</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>

**Including:**

- D1: N: 720 x 480
- D1: P: 720 x 576
- D2: 480 x 360
- SIF: 352 x 240
- CIF: 234 x 176
- DIF: 352 x 240
- CIF: 234 x 176
- OF: 768 x 448
- QOF: 176 x 144
- SQOF: 176 x 144
- QVGA: 320 x 240
- QQVGA: 160 x 128
### Appendix B: Player Compatibility and Interoperability

**Player Compatibility:** for captured video & audio playback

<table>
<thead>
<tr>
<th>Player</th>
<th>MP3G2</th>
<th>MP3G3</th>
<th>MP4/64 (divx)</th>
<th>MP4/94</th>
<th>H.263</th>
<th>H.264</th>
<th>MP2</th>
<th>WAV</th>
<th>VOB</th>
<th>WMA/W3C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mplayer (Linux)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>WMP-10 (Windows)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RealPlayer (Windows)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLC 0.8.2 (Windows)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Player Compatibility:** for multimedia video & audio streaming

<table>
<thead>
<tr>
<th>Player</th>
<th>MP3G2</th>
<th>MP3G3</th>
<th>MP4/64 (divx)</th>
<th>MP4/94</th>
<th>H.263</th>
<th>H.264</th>
<th>MP2</th>
<th>WAV</th>
<th>VOB</th>
<th>WMA/W3C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mplayer (Linux)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Quicktime (Windows)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>RealPlayer (Windows)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>VLC 0.8.2 (Windows)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>

**Stream Player Interoperability:**

<table>
<thead>
<tr>
<th>Player</th>
<th>MPlayer</th>
<th>Quicktime</th>
<th>RealPlayer</th>
<th>VLC 0.8.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTF Change Resolution</td>
<td>✔</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>OTF Change FPS</td>
<td>✔</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

**Notes:** for all above

- ✔ — Tested and working well
- X — Not supported
- TBT — To be tested
- OTF — On the Fly
- 3rd — 3rd party codec
- N/A — Not Applicable
Simultaneous Localization And Mapping
S.L.A.M.

This paper presents a solution to the problem of Simultaneous Localization And Mapping (SLAM) related to mobile robots in an unknown environment. In the past, this problem has been solved with the use of various feedback mechanisms. However, counting wheel rotations is subject to accumulated error and is securely supported by odometry, thereby decreasing reliability with distance. This new solution utilizes references of angles from the front of the robot, acquired by a Laser Detection And Ranging (LADAR) device. This SLAM approach will be used on the Coad Navigation Robot to create real-time maps of the robot arena and locate the robot within this generated map. The LADAR data will be transmitted from Coad Navigation to the network for processing and creating of the map. Our version of a SLAM algorithm is based on correlating consecutive angle, x-, and y-histograms to determine the robot's translational and rotational movement.

From here we can make a histogram of the angles of consecutive points to one another. Therefore, walls will appear as a set of constant angles. After generating histograms for two consecutive scenes, a rotation of the LADAR will appear as a peak in the angle histogram.

As with the rotation, for translation we will again make a histogram, this time for D1 and D2 movement. Again, correlating consecutive histograms will show translational movement in each direction. Before generating histograms, D1 and D2 were determined by shipping the largest walls.

From here we can see the algorithm has calculated a movement of 6.0m at 20 degrees from the two LADAR scans. This is very close to the measured movement of 6.0m at 20 degrees.

Kenneth Durkee
Stephan Kettel
Shain Robinson
Josh Schmidt
Steve Enzley
Ben Whisnant
Good Samaritan Robotics User Interface Team

This poster presents the topic of the user interface team for the Urban Search And Rescue (USAR) Robot, Good Samaritan, senior design project. The USAR senior design project consists of four teams: platform, robotic arm, miniature, and user interface. The user interface team is responsible for: human interaction controls, on-robot computing, networking, localization and mapping, sensing (CO2, distance, audio, visual, thermal), and sending motor controls. The user will input controls using a Logitech gamepad into a laptop computer, which will be sent via 802.11 a wireless network to the robot's client-side AMD Geode single board computer. This computer will send the appropriate signals and receive feedback of motor positions and status of sensors. The user will be interfacing with Good Samaritan's Linux computer through C++ software, written by the team, on a Windows laptop.
Tech Reports

Common Hardware Video Compression
Real Time Systems

Common Hardware Video Compression

J. Shea Robinson
Good Samaritan Robot, User Interface – Colorado State University

Copyright © 2006 Department of Mechanical Engineering, Colorado State University

ABSTRACT

In order to effectively transmit video data and minimize bandwidth, analog video signals must be digitized and compressed. This document will address hardware video compression using the following algorithms: MJPEG, MPEG-1, and MPEG-4. This paper considers specifics related to latency, common uses, and hardware. MJPEG is the selected compression algorithm to be implemented on the Good Samaritan Robot.

INTRODUCTION

Uncompressed HDTV requires a bandwidth of over 1 Gbps. That is the equivalent of 200 DVD’s for a half hour of television. [1] Video compression is a necessity for a practical implementation of streaming video. As the emergence of HDTV and a demand for multimedia content offered over a network progresses, the need for video compression is greater than ever.

Specifically, in regards to the Colorado State University Urban Search and Rescue robot named Good Samaritan, there is a need for compression in order to transmit the video stream from the low-light camera and the thermal camera. The bandwidth of the Good Samaritan robot is limited by the protocol constraint for 802.11a. This has a maximum throughput of 54mbps and a sustained throughput of roughly 25 mbps. [2] Please see the Good Samaritan Implementation section for more specific constraints and criteria.

There will be three compression schemes detailed in this paper, namely MJPEG, MPEG-1, and MPEG-4. The reason for the three selected is their potential use in the Good Samaritan robot. There are very few usable hardware products available for any other compression schemes. A slight exception is MPEG2 which has a few products available. This will not be discussed because licensing costs for MPEG2 hardware encoders/decoders are very high and consequently the hardware compression devices are quite expensive. [3]

COMPRESSION BASICS –

GOVERNING BODIES –

There are two bodies that define the standards for video compression. The ISO (International Standards Organization) defined the standards for JPEG (and consequently MJPEG) as well as all MPEG compression schemes. The second of these, ITU (International Telecommunications Union) has been responsible for h.261, h.263, and h.264. These were originally developed for use in video conferencing over Integrated Services Digital Networks. Until recently, there have been few ITU consumer hardware products available due to the definition of video compressors and decompressors in the ITU standards. [4]

INTERLACING - Interlacing originated in analog video to increase the refresh rates to one that was unperceivable to the human eye. [5] However, interlacing is used in digital video to halve the necessary bandwidth.

Typically video is displayed progressively. The display starts in the top left, moves right across the full line then continues to the next line on the left and moves to the right to the end of the line. This process is repeated for every line in the display. However with interlacing, instead of sweeping across every line, the display will skip every second line for a full scan, and on the next one will display only the previously skipped lines. This results in half the height of the original image. [5]

Interlacing reduces the necessary bandwidth by a factor of two by effectively halving the height of the given image. On the other hand, interlacing can create problems if there is motion in the picture. Since successive interlaced frames are meant to be shown at different times, horizontal movement causes noticeable motion artifacts.

To be seen on a progressive display (most displays with the exception of older CRT are progressive) any interlaced video must be de-interlaced which is a computationally intensive process. It is also not a perfect process and typically reduces the horizontal resolution of the video. [5]
INTRAFRAME –

Intraframe compression is the most straightforward compression scheme. The term ‘intra’ refers to being entirely within one frame. As each stream is broken down into frames, each frame is compressed using schemes similar to that of a static picture. Each frame can be referred to as an I-frame and it includes all of the information necessary to rebuild the picture. If any of the frames get lost in transmission using intraframe compression, the entire image can be regenerated during the next frame. [7]

INTERFRAME –

Nearly all movement in videos happens in the foreground with a limited amount of change between frames. Therefore, if every frame is being fully sent, there is a good deal of redundancy in the encoding and transmission. This waste will be addressed with interframe compression techniques.

Take the example of a video of a person talking. There is very limited change between successive frames. Instead of sending the complete image each frame, interframe compression sends only the change in the image. There are two types of interframes. The first is called P-frames (predicted frames). They are the difference between the previous I-frame or P-frame and the current frame. The second type of interframe is the B-frame (bidirectional frame). As the name suggests, these don’t just give the change in the frame based on a previous I-frame or P-frame, but can give the difference from a future frame as well. In order to do this, future frames must be accessible when the current frame is being compressed.

This difference between frames is given in one of two ways. First, if the object or block in question existed in the referenced frame, the P-frame gives a vector of the movement. Second, if the object or block in question did not exist in a previous frame, the full contents are included, just as they would be in an I-frame. [8]

Figure 1 through Figure 5 show a simple example of interframe compression schemes. Figure 4 shows an example of a P-frame used to create figure 3 from Figure 1. As the circle in the image changes from Figure 1 to Figure 3, the difference between the two images can be summarized by a vector and a drawing of the new portion of the image. This compressed information for this frame is considerably less information than the full Figure 3.

Figure 2 is the example of a B-Frame. It has bidirectional references to previous and/or future I-frames or P-frames. The difference necessary to create Figure 2 can actually be simply a vector of movement of the circle referencing Figure 3.
MJPEG –

SPECIFICATIONS –

Common Uses – MJPEG is used in a wide variety of settings. It requires very low computational power; as such, it can be done in software in real time. Dedicated solutions include Internet video, webcams, IP-Cameras, Surveillance Systems, screen capture software, video playback software.

MJPEG is intraframe compression using the JPEG compression scheme. The Joint Photographic Experts Group (JPEG) is a subgroup of the ISO, and in 1992, defined the JPEG standard to compress digital photographs for storing purposes. [8] There was no intention of having JPEG compression be used for video.

JPEG Compression – In static image JPEG compression, each pixel is assumed to have an eight bit depth. This is ideal for monochrome images. For color pictures, either the Red (R), Green (G), Blue (B) values are each given 8 bits or the Luminance (Y), Cr (R-Y) and Cb (B-Y) are each allotted 8 bit depth. The image is broken into 8 pixel by 8 pixel blocks because this has been found to be a good balance between computational need and included information. [8] This will be demonstrated with an example. Figure 6 shows the values in one block of 64 pixels of 8 bit depth. Each number corresponds to the bit value of the pixel.

<table>
<thead>
<tr>
<th>84 93 98 102 93 96 105</th>
<th>84 93 98 102 93 96 105</th>
</tr>
</thead>
<tbody>
<tr>
<td>95 91 122 141 117 101 104</td>
<td>95 91 122 141 117 101 104</td>
</tr>
<tr>
<td>94 91 145 176 136 98 103</td>
<td>94 91 145 176 136 98 103</td>
</tr>
<tr>
<td>95 103 154 186 138 102 101</td>
<td>95 103 154 186 138 102 101</td>
</tr>
<tr>
<td>99 93 100 136 158 120 100 102</td>
<td>99 93 100 136 158 120 100 102</td>
</tr>
<tr>
<td>111 97 102 109 100 90 107</td>
<td>111 97 102 109 100 90 107</td>
</tr>
<tr>
<td>117 103 96 91 87 93 97 115</td>
<td>117 103 96 91 87 93 97 115</td>
</tr>
<tr>
<td>119 111 101 100 97 108 110 126</td>
<td>119 111 101 100 97 108 110 126</td>
</tr>
</tbody>
</table>

Figure 6 - Original Pixel Values

Each 8 bits has a level shift to take it from a 0:255 range to a range of -128:127. Figure 7 displays the adjusted values of those in Figure 6.

<table>
<thead>
<tr>
<th>44 41 35 30 26 35 32 23</th>
<th>44 41 35 30 26 35 32 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>33 37 41 6 13 11 27 24</td>
<td>33 37 41 6 13 11 27 24</td>
</tr>
<tr>
<td>34 37 28 17 48 8 30 23</td>
<td>34 37 28 17 48 8 30 23</td>
</tr>
<tr>
<td>33 38 25 26 58 10 26 27</td>
<td>33 38 25 26 58 10 26 27</td>
</tr>
<tr>
<td>-29 -35 -28 8 30 -8 -28 -26</td>
<td>-29 -35 -28 8 30 -8 -28 -26</td>
</tr>
<tr>
<td>-17 -31 -36 -26 -19 -28 -38 -21</td>
<td>-17 -31 -36 -26 -19 -28 -38 -21</td>
</tr>
</tbody>
</table>

Figure 7 - Adjusted Values

A forward discrete cosine transform is conducted on each block. This serves to decorrelate the values for each pixel, enabling a few of the values to hold most of the information necessary for the entire block. These values can then be used to recreate an image similar to the original. [8] Figure 8 (matrix D) gives the results of a discrete cosine transform done on the values in Figure 7.

<table>
<thead>
<tr>
<th>-72.2 -92.3 -89.1 -26.9 11.3 -42.1 -81.3 -86.2</th>
<th>-72.2 -92.3 -89.1 -26.9 11.3 -42.1 -81.3 -86.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>-31.4 -18.7 -5.2 25 46 14.4 -2.7 14.2</td>
<td>-31.4 -18.7 -5.2 25 46 14.4 -2.7 14.2</td>
</tr>
<tr>
<td>5.4 8.1 -5.8 -49 -77.9 -31.3 3.7 14.3</td>
<td>5.4 8.1 -5.8 -49 -77.9 -31.3 3.7 14.3</td>
</tr>
<tr>
<td>2.9 -5 -7.2 -29.04 -43.8 -31.23 -10.7 -6.4</td>
<td>2.9 -5 -7.2 -29.04 -43.8 -31.23 -10.7 -6.4</td>
</tr>
<tr>
<td>-7.1 -8.4 7.8 9.9 10.7 4.6 7.8 1.1</td>
<td>-7.1 -8.4 7.8 9.9 10.7 4.6 7.8 1.1</td>
</tr>
<tr>
<td>-2.2 -2.6 4.2 -4.4 -6.9 -4.9 -4.2 -1.1</td>
<td>-2.2 -2.6 4.2 -4.4 -6.9 -4.9 -4.2 -1.1</td>
</tr>
<tr>
<td>-1.5 -1 2.4 -1.9 -1.4 1.1 -3.9 2.1</td>
<td>-1.5 -1 2.4 -1.9 -1.4 1.1 -3.9 2.1</td>
</tr>
<tr>
<td>-2.4 -1 3.6 5.2 -4 -2 -1.7</td>
<td>-2.4 -1 3.6 5.2 -4 -2 -1.7</td>
</tr>
</tbody>
</table>

Figure 8 - DCT Results

Each of the 64 DCT coefficients is divided by a quantization parameter (Q) and rounded to an integer. A
larger Q value gives greater compression because more of the DCT coefficients round to zero. The quantization step is the only one that is not reversible when decompressing an image. Q is often an 8x8 matrix with values set that further weight the top left corner. For this example, Figure 9 illustrates a simple Q, and Figure 10 shows the result of integer division between matrix D and Q.

\[
Q = \begin{pmatrix}
7 & 14 & 21 & 28 & 35 & 42 & 49 & 56 \\
14 & 21 & 28 & 35 & 42 & 49 & 56 & 63 \\
21 & 28 & 35 & 42 & 49 & 56 & 63 & 70 \\
28 & 35 & 42 & 49 & 56 & 63 & 70 & 77 \\
35 & 42 & 49 & 56 & 63 & 70 & 77 & 84 \\
42 & 49 & 56 & 63 & 70 & 77 & 84 & 91 \\
49 & 56 & 63 & 70 & 77 & 84 & 91 & 98 \\
56 & 63 & 70 & 77 & 84 & 91 & 98 & 105
\end{pmatrix}
\]

Figure 9 - Quantization Parameter

\[
F = \begin{pmatrix}
-10 & -7 & -4 & -1 & 0 & -1 & -2 & -1 \\
-2 & -1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & -2 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

Figure 10 - Final Matrix after JPEG Encoding

The quantized values are rearranged in a zigzag order starting in the top left and zigzagging to the bottom right, such that the more important values are grouped together at the start of the array. Figure 11 demonstrates the process in which the matrix is transformed.

![Array ordering](image)

Figure 11 - Array ordering

The values are then truncated after the last non-zero value and an end of set character is added. The array of Figure 10 is illustrated here:

-10  
-7  -2  
0  -1  -4  
-1  0  0  0  
0  0  0  1  0  
-1  1  -1  0  0  0  
0  0  0  -1  -2  0  -2  
-1  0  -1  -1  [end of set]

What were originally 64 pixels of 8 bits has become a short array. The full code for this example can be seen in Appendix A.

Bandwidth – Because MJPEG is by far the simplest compression scheme, it will have a relatively high bandwidth. However, the compression ratio is also dependent on aspects such as image complexity and the amount of chosen compression.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Image Complexity</th>
<th>Compression (1-5)</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>High</td>
<td>Low(2)</td>
<td>24.4 Mbps</td>
</tr>
<tr>
<td>640x480</td>
<td>High</td>
<td>High(4)</td>
<td>17.5 Mbps</td>
</tr>
<tr>
<td>640x480</td>
<td>Medium</td>
<td>Low(2)</td>
<td>11.3 Mbps</td>
</tr>
<tr>
<td>640x480</td>
<td>Medium</td>
<td>High(4)</td>
<td>7.6 Mbps</td>
</tr>
<tr>
<td>320x240</td>
<td>Medium</td>
<td>Medium(3)</td>
<td>2.3 Mbps</td>
</tr>
<tr>
<td>160x120</td>
<td>High</td>
<td>Medium(3)</td>
<td>1.2 Mbps</td>
</tr>
<tr>
<td>160x120</td>
<td>Medium</td>
<td>Medium(3)</td>
<td>.6 Mbps</td>
</tr>
</tbody>
</table>

Table 1 was compiled using Axis Communications Design Tool. All values have a frame rate of 25 fps. Axis communications is the front runner in developing IP-Cameras for the surveillance industry. Their large range of cameras uses predominantly MJPEG and MPEG-4 compression. [9]

Advantages & Disadvantages – Compression ratios of 1:10 can still create images indistinguishable to the human eye. Compression ratios of 1:100 create noticeable artifacts in the reproduced image. [8] Typical Compression ratios for MJPEG videos are between 1:20 and 1:30. [7]

Open JPEG C++ libraries exist that allow video compression and display to be coded easily into new applications. [3]

MJPEG is advantageous because the compression ratio can be scaled based on the Quantization Factor (Q) that
is used. Depending on the software, this potentially can be done dynamically as bandwidth priorities change.

Hardware Availability — Ironically, the availability of dedicated MJPEG hardware is limited. Due to its simplicity, dedicated hardware is not often necessary. However, MJPEG is included as a feature in almost all options for compressing video; this includes digital cameras, surveillance cameras, digital video recorders, and frame grabbers.

MPEG-1 —

COMMON USES —

The MPEG standards were set by, and are named after the Motion Picture Experts Group which is a sub group of the ISO. The first standard they published was known as MPEG-1 and had the goal of enabling video and audio playback from CD-ROM at a constant bit rate of 1.4 Mbps with a quality similar to a VHS tape.

While it never gained popularity as a replacement for VHS, it did gain widespread use in CD based interactive applications as well as video playback over the internet.

SPECIFICATIONS —

Blocks — MPEG-1 has a typical resolution of either 352x288 or 352x240. It breaks the image into 16x16 pixel macroblocks. Each Macroblock is broken into four 8x8 pixel Luminance(Y) blocks and two 8x8 Cr and Cb blocks. Cb and Cr have half the horizontal and vertical resolution of the Y component. Therefore, each macroblock is broken into six blocks. [8] Figure 12 demonstrates the full hierarchy breakdown of MPEG-1 compression. The video sequence is chopped into A Group of Pictures (GOP). Each GOP consists of a set of images or frames. Each GOP contains one I-frame. The number of frames in a GOP is set by the codec designer. Pictures are further broken down into slices that cover the picture in raster order. There is no set number of slices in a picture. Slices help with transmission errors. They create synchronization points inside of the frame so that a single transmission error will not ruin an entire frame. [10]

It should also be noted that MPEG-1 makes no attempt to include interlacing. There were attempts to add it after the fact, but standardization of interlacing was not included until MPEG-2.

I, P, B Frames — MPEG-1 is designed to use I-frames, P-frames, and B-frames. However, there is a good deal of flexibility built in enabling the codec designer to balance the compression ratio, computational power, and robustness. A few potential designs are as follows:

- Only I-frames. This is very similar to MJPEG. It requires very little computational power on both ends of the process, is very robust regarding errors in the data, very straightforward and has very low compression based latency; however, this design has relatively low compression ratio.

- Only I-frames and P-frames. This design is still very straightforward, has low inherent latency, better compression ratio, and still fairly good tolerance of errors. Error tolerance is related to the frequency of the I-frames. These serve the purpose of fully refreshing the screen and extinguishing any visual error.

- Small Groups of Pictures (GOPs), using P-frames and B-frames. This is more computationally intensive, as it requires much more motion prediction in the B-frames. It also has some latency due to the B-frames that are dependent on future P-frames. On the plus side, this design has a much greater compression ratio.

- Large GOPs require significantly more computational power than do the Small GOPs design. Compression ratios are at a maximum with this design. Unfortunately any error is
MPEG-1 is designed to be an asymmetrical encoding process. Because MPEG-1 was intended to be encoded once and played back many times, the encoding process is much more computationally intensive. The motion prediction involved with creating B-frames is one of the major reasons for the asymmetry. Encoding I-frames is not much different from the process used for JPEG images. Each block is run through a discrete cosine transform, quantized, and turned into an array using a zigzag pattern. The major difference is B-frame and P-frame can use arrows to designate changes from reference frames. [8]

Frame Ordering – An additional aspect of complexity that gets affected with the inclusion of B-frames is the order in which the frames must be compressed and sent. Figure 13 shows the order of the frames as they would originally be seen. Frames 1-6 would represent a GOP. Due to the interframe dependencies of B-frames, any reference frames must be sent before the B frames. Figure 14 shows the order in which this specific example would be compressed, transmitted, and decompressed. [8]

HARDWARE AVAILABILITY –

Out of the three options that were analyzed, dedicated MPEG-1 hardware for the Good Samaritan is the rarest. Often it is a hardware encoder included as a feature on another board. MPEG-1 is an older technology and as such most of the available products are considered legacy hardware.

MPEG-4 –

COMMON USES –

MPEG-4 version 2 was officially made a standard in early 2000. It improves on the video and audio compression of the earlier MPEG-1 and MPEG-2, but includes major additions useful in games, online streaming, and interactive multimedia. MPEG-4 is very versatile and as a result, it is by far the most complicated compression algorithm discussed in this paper. If this can’t be deduced from the official overview being 67 pages long, it could be deduced from the fact that MPEG-4 can include any number of 23 different parts. [11]

SPECIFICATIONS –

Objects – The major difference between MPEG-4 and previous standards is that video stream or video session (VS) is broken into objects before it is encoded and compressed. This is a large jump from the rectangles and cubes of previous algorithms. This also enables MPEG-4 to encode different types of objects uniquely. For example, most compression schemes always have a problem with overlayed text because it is such a stark contrast to the background. As a result, there is discoloration at the edges. However, in MPEG-4 the text is removed as a separate object, compressed using ASCII code along with information for the font, size, color, etc. This produces clean edges between the text and the background using less bandwidth.

Figure 15 shows the breakdown of a video signal going through MPEG-4 compression, while Figure 16 displays the breakdown of the VS.

ADVANTAGES & DISADVANTAGES – MPEG-1 lands in the middle of nearly all categories. The compression ratio is more than MJPEG, but it still requires 5-6Mops to transmit 720X480 resolution. However, the computational need is much greater for encoding MPEG-1. Additionally, there is inherent latency because any B-frames must wait on all referenced frames before they can be encoded.
1 to be a circle object in the VS, while VO 2 would be the
surrounding portion.

Figure 16 - MPEG-4 hierarchy

Objects can be any shape. Each VO includes two parts
the shape/position which is known as key signal and the
video image, which is referred to as the texture. The
decoupling of one object from another allows the codec
designer to allocate more bandwidth for the portion of the
image that is receiving the most attention. For instance, if
the foreground object is a talking head, while the
background is only gradually changing, MPEG-4 opens
the door to encode the foreground at a greater frame
rate and a smaller quantization factor, causing a much
smoother image. This is known as scalability and is one
of the features of MPEG-4.

Another feature is a sprite. A sprite is a fully static image
that can be held constant, panned across, or stretched.[11] Its most obvious use is in video games, where the
background could be a large still image that is panned
across as the character moves around. This means that
the full image only has to be sent once and kept in
memory in the decoder. Another use of a sprite is in a
watermark. Network television or a website could use
this to mark their video feed.

The idea of a sprite is loosely related to the meshing
options that are also available in MPEG4. A single object
could be meshed using small triangles (similar to a mesh
in computational fluid dynamics) and then changes in
that object are not sent as direction vectors for the
macroblocks, but as stretching and warping of the mesh.
This meshing can be done in either two or three
dimensions.[8]

MPEG-4 does not necessarily need to be implemented
using any of the additional features. It does include a
noticeable upgrade to the straight video compression
that can be done in MPEG-1. MPEG-4 is also much
more resilient to errors. It should also be noted that
1.254 is a subpart of MPEG-4, specifically Part 10.[7]

ADVANTAGES & DISADVANTAGES – It is not
uncommon for MPEG-4 to have a compression ratio of
1:100. It is also not uncommon for MPEG-4 to transmit
720x480 video in fewer than 2 Mbps. However, it is much
more computationally intensive than any of the previous
methods. Also, due to vast number of potential options
available in the compression, it is much more difficult to
decode.

HARDWARE AVAILABILITY – While hardware decoders
are considerably more complex because of all the
additional options, simple hardware encodes do exist
and are getting progressively more available. MPEG-4 is
poised to be the major standard in IP based multimedia.
It is already standard on most IP cameras [9] and is
where the market is heading in terms of surveillance [3].
Additionally, it is the most common compression format
for a frame grabber.

GOOD SAMARITAN IMPLEMENTATION

Hardware Constraints – Video compression solutions for
the Good Samaritan must be able to handle both
cameras:

- An A10 Thermal Camera with 160x120
resolution and 14-bit depth [12]
- A Zeorlux PC209IR with 400X300 resolution and
8-bit depth [13]

Both Cameras have an NTSC (RS-170) signal output. It
must be able to handle both video feeds simultaneously
in real time.

The selected hardware for the Good Samaritan robot
was the Sensaray Model 314 Video/Audio Grabber. It is
capable of compressing video at 30fps in MJPEG,
MPEG1, or MPEG4 as well as others.[14] Therefore our
hardware does not limit the options of the compression
algorithm. Aside from the options available for
compressing video, the Sensaray Model 314 is the best
option because of the PC104+ form factor, reasonable
price, and included Linux drivers.

Latency Constraints – The Good Samaritan User
Interface team has decided on a maximum full loop
latency of 3 s. This is the maximum amount that is
cognitively unnoticed when driving a vehicle remotely.[2]
While most of the latency will be caused the wireless
communications, some is still caused by the inherent
compression algorithm and this should be minimized.
MJPEG would have the least inherent latency because it
is intraframe compression.

Bandwidth Constraints

Table 2 shows the bandwidth constraints imposed by the
additional hardware on the Good Samaritan. This
demonstrates that video compression is necessary.
Provided there is a sustained throughput of 25 Mbps
between the robot and the display device. There remains
just under 24 Mbps for video transfer. Consequently, the video compression doesn't require a high compression ratio.

### Table 2 - Bandwidth Budget courtesy of CSU – USAR User Interface Constraints and Criteria [2]

<table>
<thead>
<tr>
<th>Description</th>
<th>802.11a (25 Mbps sustained)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A10</td>
<td>8,064,000</td>
</tr>
<tr>
<td>Thermal Camera</td>
<td>bit/s</td>
</tr>
<tr>
<td>Zerolux</td>
<td>30fps 400X300</td>
</tr>
<tr>
<td>Color Video Camera</td>
<td>24,000,000</td>
</tr>
<tr>
<td>LADAR</td>
<td>(64 bit/dist * 680dist/scan * 10scans/sec)</td>
</tr>
<tr>
<td>Microphone</td>
<td>bit/s</td>
</tr>
<tr>
<td>Motor Control</td>
<td>64kbits X 2 channels</td>
</tr>
<tr>
<td>Arm Control</td>
<td>128,000 s</td>
</tr>
<tr>
<td>CO2 Detector</td>
<td>8bits/motor/signal + headers</td>
</tr>
<tr>
<td></td>
<td>8bits/motor/signal + headers</td>
</tr>
<tr>
<td></td>
<td>8bits/signal + headers</td>
</tr>
</tbody>
</table>

The three main categories on which to choose a compression solution are computational simplicity, compression ratio and minimal latency. Table 3 summarizes the three presented options. (1 is poor, 5 is good in that respective category).

### Table 3 - Summary of Compression Formats

<table>
<thead>
<tr>
<th>Format</th>
<th>Computational Simplicity</th>
<th>Compression Ratio</th>
<th>Low Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>MJPEG</td>
<td>5</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>MPEG-1</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>MPEG-4</td>
<td>1</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Based on Table 3 and on the available hardware for the Good Samaritan robot, the chosen compression scheme is MJPEG. There is a minimal need on the Good Samaritan robot for high compression ratios and due to the simplicity of implementation MJPEG is the best option. While these constraints may change down the road, the implemented hardware will minimize the time and effort needed to switch to another algorithm.

### CONCLUSION

Compressing the two video streams is a necessity (Table 2). Because the bandwidth is not an overwhelming concern, MJPEG is the preferred method for the Good Samaritan robot due to its simplicity in encoding and decoding. In the future, given ample time, a basic MPEG4 compression algorithm should be implemented in order to free up more bandwidth for additional communications.

### ACKNOWLEDGMENTS

Carl Kaiser – Instructor, Colorado State University

### REFERENCES


<table>
<thead>
<tr>
<th>CONTACT</th>
</tr>
</thead>
</table>
| J. Shea Robinson  
Mechanical Engineering Student  
Colorado State University  
Shea.Robinson@Colostate.edu  
www.srheo.com |

**DEFINITIONS, ACRONYMS, ABBREVIATIONS**

**GOP**: Group of Pictures

**ISO**: International Standards Organization

**ITU**: International Telecommunications Union

**JPEG**: Joint Picture Experts Group. Commonly used to refer to the compression standard for still images.

**MPEG**: Motion Picture Experts Group. Commonly used to refer to one of many sets of standards for digital video.

**APPENDIX A:**

Matlab code used for JPEG Compression:

```matlab
%written by shea robinson
%used to find a good 8x8 matrix to display
the JPEG compression format

format compact;
clear;

I=[52,55,61,66,70,61,64,73];
[63,59,55,90,109,85,69,72];
[62,59,68,113,144,104,66,73];
[63,58,71,122,154,106,70,69];
[67,61,68,10,4,126,88,68,70];
[79,65,60,70,77,68,58,75];
[85,71,64,59,55,61,65,83];
[87,79,69,68,65,76,78,94];
]+32

%average the values around zero
S=I-128

%take the Discrete Cosine transform of it
D=dct(S)

%set q and divide by it
Q=7*[1,2,3,4,5,6,7,8];
[2,3,4,5,6,7,8,9];
[3,4,5,6,7,8,9,10];
[4,5,6,7,8,9,10,11];
[5,6,7,8,9,10,11,12];
[6,7,8,9,10,11,12,13];
[7,8,9,10,11,12,13,14];
[8,9,10,11,12,13,14,15]

%find the final values
F=round(D./Q)
```
ABSTRACT

With this document, future robotic system teams will obtain a basic understanding of modular software design. Modular software design is used to create flexible, malleable, and robust systems. [1] The nature of robotic systems is that they are used devices and technologies. They also communicate in many different ways. This paper will focus on what software design is, how it relates to robotics, and developing systems that have high degrees of modularity.

INTRODUCTION

Building a complex system using many simple individual components is already utilized in Mechanical Engineering and can be applied to any other domain, such as software design. This process of simplifying the problem into individual parts that are each self contained can be difficult to achieve, so the best component-based solution is found that will meet all the requirements set for the given problem.[1]

SOFTWARE DESIGN

Software design can be described by the principles of design, the approaches utilized, and more.

PRINCIPLES OF SOFTWARE DESIGN – Different principles include modular, portable, malleable, that will be discussed in detail within the remainder of this paper. [2] Each principle has its purpose and most of the principles are utilized in every software engineering project, regardless of size and scope. However, in more complex systems features such as modularity, portability, and malleability are commonly given a higher priority or status. Many engineers and software developers today continue to redevelop code and systems that have been implemented before. While the world will never be perfect, [2] much advancement in code and design reuse needs to happen before software engineering can become a professional standard that the rest of the engineering fields adhere to already.

MODULAR DESIGN – Modular design is implemented by dividing a large problem set into multiple sub problems or sub-sets of the main problem. Loose internal coupling and high internal cohesion is desired when designing how the system will be broken down into sub-system components. A Modular design should hide any internal workings of each component. This allows for the component itself to be changed and modified with no concern of the rest of the system. Encapsulation, separation, and interfacing are key ideas in a modular implementation.

Develop a Design – Software design should have no regard to the type of programming whether it is a functional, object-oriented, or procedural in nature. In this respect modularity is introduced in the design which is then translated into working code. During the design process, however, a programming language can and should be selected, but only after the system and possible design solutions have been thoroughly analyzed.

Object-Oriented Programming – An object is an entity that contains data or properties, as well as functions or actions. This process of combining all relevant information into a single entity is called encapsulation. The idea of object-oriented programming allows software to be designed as a collection of objects that all interact with one another to solve a problem.

PORTABLE DESIGN – A new software system is often created for every new robotic system. This poses a problem when the solution needs to be built and developed quickly, safely, and made to last. This challenge has been solved in part by the designers of business applications, and the technologies surrounding the internet and its related applications. These applications need to work on multiple hardware configurations, as well as different operating systems and run-time platforms. Developing an application-independent framework can help with making a system more portable.[5] Portable design is one in which a system can be designed, and then utilized on multiple platforms or with different hardware. There must be an abstraction layer for portability to be achieved, as each platform
or hardware manufacture will put in new features or different protocols that allow them to be competitive within the market, as well as allowing them to be innovative. Many times there are multiple abstraction layers, or portability layers.

**Application Programming Interface (API)** – Portability is usually achieved through APIs. These are usually a defined method of interaction with an abstraction layer. Operating systems usually have at least one functional API; many applications have their own API. An API can be defined, and standardized allowing other developers to create their software using the API with the security that it will work on any system that can handle that specific API. The hardware and framework developers then need to support the API by integrating it into their current interface, layering the new API over their current API, or redesigning the foundation of their system to incorporate the new API.

**MALLEABLE DESIGN** – A system that must be changed or must adapt anytime an outside force is applied or requirement is changed, or added is not a good one. A malleable design allows a system to be robust, yet adapt to changes both internally and externally. Adapting to internal change is relatively easy, compared to external change; it can be handled during design and development phases of a project. The correctness of the system will not change with a good design. [1] Mobile Robotic systems must also adapt too many external elements in the surrounding environment. In some cases the environment is well known, but many robotic applications are mobile, and thus must be able to readily adapt. This principle should be addressed at the same time as modularity, during the design phase. Waiting to address this issue later will lead to a loss of valuable resources, mainly time and labor. A robotic system should account for a sensor failing, a sub-system giving erroneous data, as well as allowing a hardware component to be replaced without affecting the rest of the system. This type of componentized system design is even more necessary as the robotic projects move to greater degrees of autonomy.

**SOFTWARE ARCHITECTURE**

The architecture of a software system includes not just the individual software components or subsystems, but also multiple systems and the interaction between them. This interaction defines the interfacing requirements of the overall system. Each of the following models can be used alone, or together to help architect a modular system.

**DISTRIBUTIVE MODEL** – There are a few commonly used implementations of this model, Client-Server and Peer-To-Peer (P2P), and Tiered.

**Client-Server** -- This model usually has one system that is the server, or the master. Fig. 1 shows a visual diagram of the connections between each system. This system sends and receives messages to and from each of the clients. If one client needs to talk to another client, then the message is routed and handled by the server. A robotics project can utilize this method when there are a lot of individual robots that each has their own task. Design each robot to be a client, and have a central monitoring and control system that can be the server. [1, 2]

![Figure 1: Client-Server Architecture](image)

**Peer-To-Peer** – In this model each system or robot acts as both a client and a server. This allows any one system to interact with any other system that it knows about. [2] An application where this can be utilized is when creating a search and rescue team of robots that are fully autonomous. Each one can work on its own, but can also ask other robots for information about the environment or its state. One could correlate this to search team of humans looking for a body along a river, or in a disaster area. Each member of the team can search on their own, but they can communicate with other members in order to determine areas already covered or other such information. No one system or person is in charge, but rather each member works as a part of a greater whole. The peer to peer nature of the system provides a greater scalability as well as efficiency not having to communicate to the server in order to communicate with another member.
Tiered – Similar to Client-Server, Tiered architecture allows multiple levels of clients and servers to exist. A common 3-Tiered architecture includes the client tier, the server tier, and a middle man or broker tier, shown in Fig. 3. This middle tier allows clients and servers to interact without much if any knowledge about the each other. All it needs to know is how to interface with the broker. This similar to abstraction layers within a system, except applied externally to multiple systems. [2]

![Tiered Architecture](http://www.mannrt.com/peertopeer.png)

**Figure 2: Peer-to-Peer Architecture**

![Subsumption Layered Approach](http://www.evolution.com/images/product/oem/3d_diagram)

**Figure: Layers of a software system**

**Dependency Issues** – A better approach would be to make each layer a self contained component. Then interface them in a way that one component can override another component, which would give the same effects as the Subsumption design, except only the layer that failed would not be working any more. The goal of this type of design is to actually reduce the complexity that dependency issues cause. In some cases a dependency must exist, for example a motor controller might be dependent on the functional state of the motor. If the motor fails, there is no need for continued operation of that motor’s controller. This type of dependency ideally would be removed, however it is an acceptable limitation of the system as the two can be treated as a single unit. There are other reasons that a system dependency is acceptable including those that support a greater degree of modularity such as APIs and frameworks.

**Layered Model** – A Layered architecture model gives each system a layer to run in, and then separates these layers into distinct entities. In many cases each layer adds complexity to the layer below it. Layers are sometimes given more priority at the lower levels, and less priority systems at higher levels. The Subsumption design ideas, discussed by Brooks [6] allows for systems at higher levels to fail, and the lower level will continue running without any concern of the failed system. This is very useful in robotic software design as it gives the system a higher degree of malleability. In some respect layers are identical to creating abstractions; however it can be distinguished as layers are only a component of abstraction. The idea that a layer can fail and the underlying layers will take over is a nice feature for any system to have, but what about the layers above the one that failed. This is a problem of dependency.

**Framework Model** – Frameworks are a foundation on which to build applications. A framework is a set of libraries, resources, and other supporting structures that provide a base of functionality that other programs can use and program against. Examples include the standard C library, Standard Template Libraries (STL) for C++,
as well as Microsoft’s .Net framework. A framework is useful for setting up a variety of standard tasks that most every software application needs. This way it enables applications to be designed on top of the framework. The framework can then be implemented on any hardware configuration, and every application written to depend only on that framework will run on any hardware that has an implementation available. This implementation is relatively small compared to the hundreds of applications written on top of it, thus the cost of writing a framework on top of a different hardware or underlying configuration.

**CASE STUDIES**

There are two case studies that will be discussed. A software scheduling and meeting system developed in Java, along with a robot geared to compete in the RoboCup Urban Search and Rescue (USAR) competition.

**USAR ROBOT** – This robot, named Good Samaritan, is a combination of a motorized platform that can articulate sensors, an attached arm device, and a server laptop with user input controls. The robot itself has a computer onboard that can accept input and output.

**Hardware** -- There are five major sensors: Visible camera, Thermal camera, Laser Radar (LADAR), CO2, and Audio. Each sensor has a unique input interface that must be incorporated into the system. There are also controllers and actuators: driving motors, articulation motor, a fully functional arm, as well as pan and tilt servos for camera movement. Each of these hardware devices is abstracted in order to allow the main software application to run even if a device is not working, or if the device is modified, removed, or replaced.

**Robot Software Design** -- The onboard computer software design includes running a general purpose operating system with specific driver, which gives a base layer of abstraction for the hardware through the operating system’s HAL. A standard framework on top of the operating system provides algorithmic functionality as well as networking and communication protocol foundations. Many APIs are utilized within the framework in order to access the functionality within the framework itself. An API is also written for communication protocol interfacing. These protocols are the specific information about the sequence and structure of the data received or sent through each hardware device.

**Server Software Design** – A Laptop running Windows XP operating system, is used as the server. The software design itself is build on top of two major frameworks must be used for the server software. The first is for the human user interfacing, and the second, like the robot’s design, is used for developing against standard system functionality. The first framework is the .Net framework, specifically DirectInput, which is utilized to gather input from the mouse, keyboard, and Gamepad devices. The second framework is utilized for network communication, as well as standard algorithms and data structures.

**CONCLUSION**

There are many reasons to consider designing robotic systems using a modular approach, including adaptability, maintainability, as well as robustness. Software design has come a long way, but in the field of robotics it is in its infancy, as new complexity arises, and autonomy is strived for.

“Reuse is not a new concept. It was Isaac Newton who said, ‘If I have seen further it is by standing on the shoulders of giants’”. [2]

A component based modular design is essential to the success of any software application, especially in
the field of robotics. However, achieving pure modular design along with other desired properties such as malleability and portability is a difficult task. Utilizing many of the design methods and ideologies mentioned in this paper will help in designing any software system, and is geared for developing systems based on components in order to achieve as high a degree of modularity possible.

There is never one tool for every job, and this applies to software design as well. Take a look at the methods presented here, as well as others and analyze them carefully with respect to your specific application need. Some applications might require the full use of all methods discussed, or maybe just a portion of a method. Just like any design it is open to interpretation and modification.

For any applications regarding robotics, a modular approach has been found to be beneficial, as well as necessary as the field moves further in its progress toward full autonomy and other complex systems.

REFERENCES


T. Lethbridge, R. Laganiere: 2005 Object Oriented Software Engineering “Practical Software Development using UML and Java”

http://citeseer.ist.psu.edu/384720.html


REFERENCES


T. Lethbridge, R. Laganiere: 2005 Object Oriented Software Engineering “Practical Software Development using UML and Java”

ANTONIO C. DOMINGUEZ-BRITO, MAGNUS ANDERSSON AND HENRIK I. CHRISTENSEN, "A SOFTWARE ARCHITECTURE FOR PROGRAMMING ROBOTIC SYSTEMS BASED ON THE DISCRETE EVENT SYSTEM PARADIGM", TECHNICAL REPORT CVAP 244, CENTRE FOR AUTONOMOUS SYSTEMS, KTH - ROYAL INSTITUTE OF TECHNOLOGY, S-100 44 STOCKHOLM, SWEDEN, SEPTEMBER 2000. 11
http://citeseer.ist.psu.edu/384720.html


Dr. France: Computer Science and Software Development Methods, class notes. http://www.es.colostate.edu/~cs314/

CONTACT

Steve Tranby
Colorado State University
Email: stevetranby@gmail.com
Website: http://www.stevetranby.com/

ADDITIONAL SOURCES

Dr. France a Colorado State University Professor

DEFINITIONS, ACRONYMS, ABBREVIATIONS

Abstraction: Used in layered programming, or design, in order to have each layer know only the previous layer’s inputs and outputs, but not the definition or details of how the underlying layers work or how they have been created.

Encapsulation: Having a single object describe all of the functionality and properties that fully describe itself and its actions

Framework: A broad set of resources that are bundled together as a foundation for developing applications.
Functional Programming: A programming that builds complex functions by utilizing recursion and simple base functions. (definitely make better – use scheme book)

Object-Oriented Programming: Using the idea of objects that have methods or functions that they can perform as the object’s action, as well as having internal data about the object’s properties (make better)

OOP: Object-Oriented Programming (See previous entry)

Procedural Programming: Programming in a sequential order. Usually utilize methods or procedures that can call other sub-procedures. Languages such as C and Basic are examples.