REMOTELY CONTROLLED INTELLIGENT VEHICLE TO HANDLE PUBLIC PLACES SECURITY

Abdel Latif Abu Dalhoum, Mohammed Al-Rawi, Ahmed Al-Sharieh, Ayman M. Najjar, Maen M. Najjar, S. Shayreh

Computer Science Department, King Abdullah II School of Information Technology,
The University of Jordan, Amman 11942, JORDAN

a.latif@ju.edu.jo, rawi707@yahoo.com, sharieh@ju.edu.jo, ayman@kwakeb.net, maen@kwakeb.net, s.sharieh@ju.edu.jo

Abstract: In this paper we propose a simple design and implementation of an intelligent robot that can be used to handle security breaches. Applications may include counter terrorism, handling pollution crises, monitoring for security purpose, etc. Ultrasound sensors, temperature sensors, GPS devices, and wireless cameras are mounted on the robot in order to give 360 degrees stereo vision and other environmental sensing. JPEG2000 image compression is used to enable efficient video streaming for views scanned by the cameras to the server that manages the Robot. The Robot is designed to work as a server based, and many authorized security personals can connect to the Robot in order to monitor public places such as malls, parking, etc.

Key-Words: *Robotics, Jpeg2000, stereo vision, automation, Internet remote control*

1 Introduction

Robots play an important role in every day live of humans. Basically, a robot is simply a computer with some sort of mechanical body designed to do a particular job [1, 2]. Usually, it is able to move and has one or more electronic senses. These senses are not nearly as powerful as our own senses of sight and hearing [3]. However, scientists and engineers are working hard to improve robots. They are constantly coming up with ways to make them see, hear and respond to the environment around them. The definition above describes the proposed robot that requires many electronic senses. These electronic sensors have to be connected to a computer in order to perform specific tasks. Besides the sensors, the robot has in fact a mechanical body (e.g. servo motors) that facilitates its movement and gives the robot the ability to perform its tasks.

Intellibot[1] is a robot vehicle that can make decisions by itself when it moves e.g. stop moving or change direction when an obstacle occurs. The vehicle can be controlled remotely using a computer with internet connection. The vehicle will receive control instructions through EDGE technology; a different implementation could be using Wi-Fi or Bluetooth radio. This implementation, however, restricts the possibility of controlling the vehicle from very far locations, an advantage of this approach is its reliability and low latency; control instructions will be received by the vehicle in a very less time than it would when using EDGE. Another implementation could be combining both approaches and setting priorities (Computer here does not necessarily mean the typical definition of computers that people usually would think of, however, it means any device consists of the main components of computers: CPU, Memory and storage). There will be several Wi-Fi cameras equipped with the vehicle to capture live video recordings of the scene at front and rear and transmit it to the control device (e.g. Laptop).

Besides what have been said, our robot should be easily expandable; that is, adding new modules (e.g. sensors, mechanical devices such as artificial arm) should be an easy matter and also reprogramming the robot, debugging ...etc. Hence, we have considered different approaches to implement the robot vehicle and concluded that a single board computer (SBC) is the best way to implement such robot. But choosing a specific single board computer is not an easy task; hence, we have made a study about most suitable single board computers and recommended one of them. Section 2 gives Intellibot organization, software components is the topic of section 3, Intellibot security is discussed in section 4, compression of Intellibot images is given in section 5, obstacle detection is the topic of section6, and section 7 wraps up the conclusions.

2 Intellibot Organization

Intellibot is organized into the following groups. (See Figure 1):

Mini-Computer: A computer provides CPU, Memory and Storage. The computer is used to run the Intellibot's software and store configuration files. The software runs

[1] ***This work was supported by the Dean of Scientific research of the University of Jordan under contract no. 1029***

on top of the mini-computer and controls the robot's hardware resources (such as mobility motors, sensors, camera,…etc). The computer also has a chargeable Li-ion battery which provides other hardware components with power.

Physical Interface: This interface connects between the computer and other hardware parts (e.g. motors and sensors)

Other Hardware Components are :Mobility Motors, Sensors, GPS device, LEDs

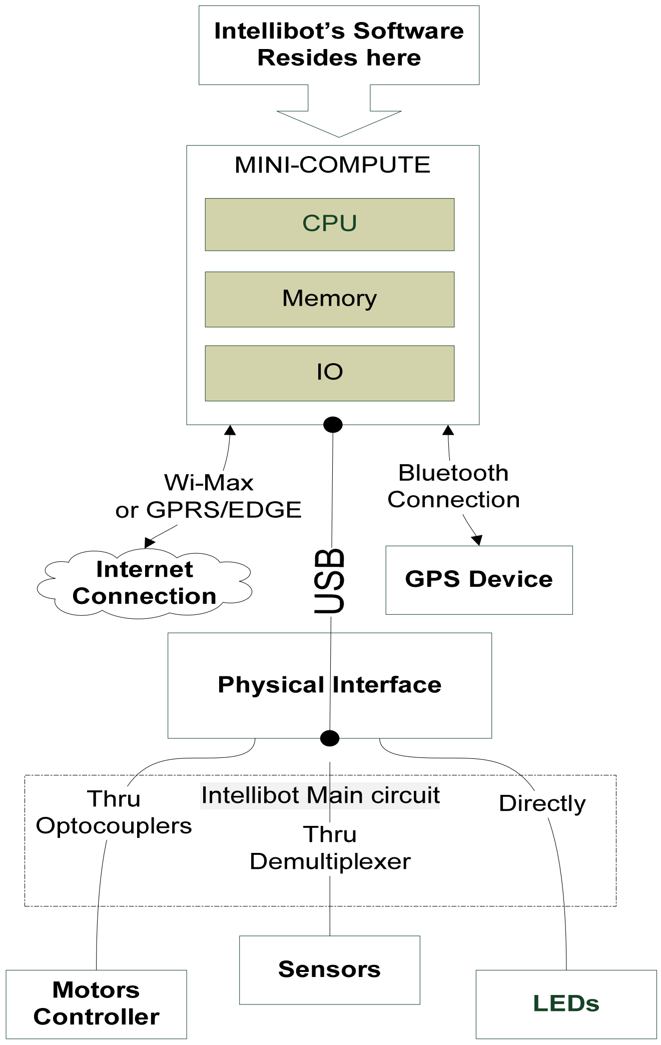


Figure 1: Intellibot Organization

2.1 *Intellibot Main Circuit*

As shown in Figure 1, the Intellibot Main Circuit connects between the Physical Interface and hardware components like Mobility Controller, Sensors and LEDs. The main circuit provides resistors to protect against excessive current coming from Mobility Controller (that is shown in Figure 2) which has its own 24V input. This approach helps protecting Physical Interface's output ports from damaging. It also multiplexes x digital outputs into $2^{(x-1)}$ outputs, this is useful when connecting sensors that cannot be operated at the same time (similar to sonar sensors which crosstalk). The main circuit also connects LEDs directly to the Physical Interface.

2.2 *Sensors*

Sonar sensors are attached through the Phidget controller to sense the surrounding objects in order to avoid obstacles. However, to minimize the crosstalk between each sensor, the sensors are fired sequentially. Figure 3 gives sensor that may be adopted for this task.

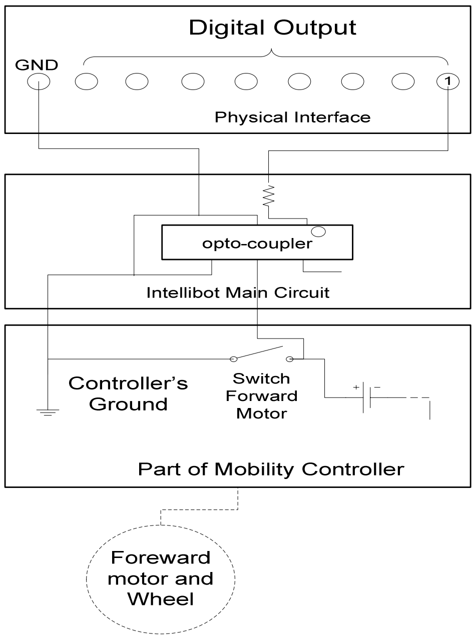


Figure 2: Mobility Controllers-Main Circuit-Physical Interface Connection

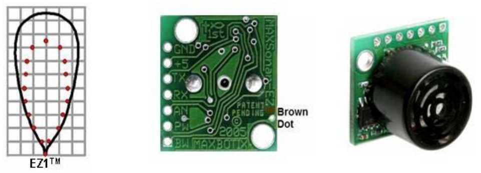


Figure 3: Showing the sensor used in the Intelibot

2.3 *GPS Device*

Internet Connection GPS device is connected to the minicomputer directly using a Bluetooth connection. Communication between the GPS device and the Intellibot software is done using NMEA protocol. The software will parse NMEA strings coming from the GPS device to track its location. The communcation between the GPS receiver and Intellibot is made over Bluetooth. Pairing with the GPS receiver requires a passkey which is hardcoded in the server. The GPS receiver sends standard NMEA strings.

2.4 Internet Connection

The internet connectivity will be available through WiMAX technology, a USB WiMAX modem is connected directly to the minicomputer. An alternative solution could be the GPRS/EDGE technology. See Figure 4 for a description of the connections where the Data-Link layer and the Physical Layer can be changed to adapt new industry standards. New Wi-Fi standards with higher ranges and data transfer rates can be adapted easily without modifying the software

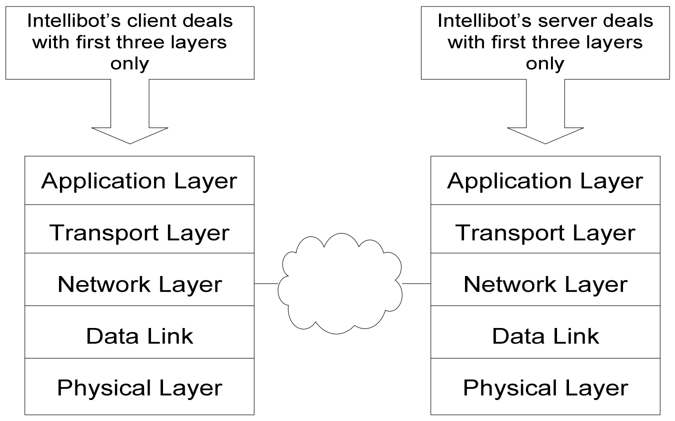


Figure 4: Data-Link layer and Physical Layer

3 Software Components

The Intellibot software is divided into several components; each component is responsible for one type of functionality. Some of the components works simultaneously, see Figure 4 for details. The software components are as given below:

3.1 Bootloader

When the system starts to boot, Bootloader component is invoked. Bootloader first asks Communication component and Phidget Interface to start a self-test and report back to Bootloader. After self-test is successful, Bootloader starts the system by invoking the core construction which initializes itself and initializes attached modules (such as Environment Module and Mobility Module), then it asks both Communication and Phidget interface to initialize themselves and their sub-components.

The bootloader uses LEDs to inform the user of its status. In case of fatal errors the bootloader boots in Failsafe mode. Fatal errors indicate the Intellibot cannot operate properly and it needs maintenance (e.g. hardware failure such as Mobility motors). See Figure 5 for an overview of the bootloading process.

3.1.1 Failsafe Mode

As mentioned in previous sections, the bootloader tests the main component of the system when it is powered on. If there are any fatal errors that prevent the system from working properly, the bootloader will boot up in Failsafe mode. In Failsafe mode, a user with physical access only can connect to the Intellibot through Ethernet connection (crossover UTP cable with RJ45 connector). The user could then telnet to TCP port 9908 and establishes connection with CLI listener. Through CLI, the user can configure the system and make sure that all configurations are correct and see log files as well.

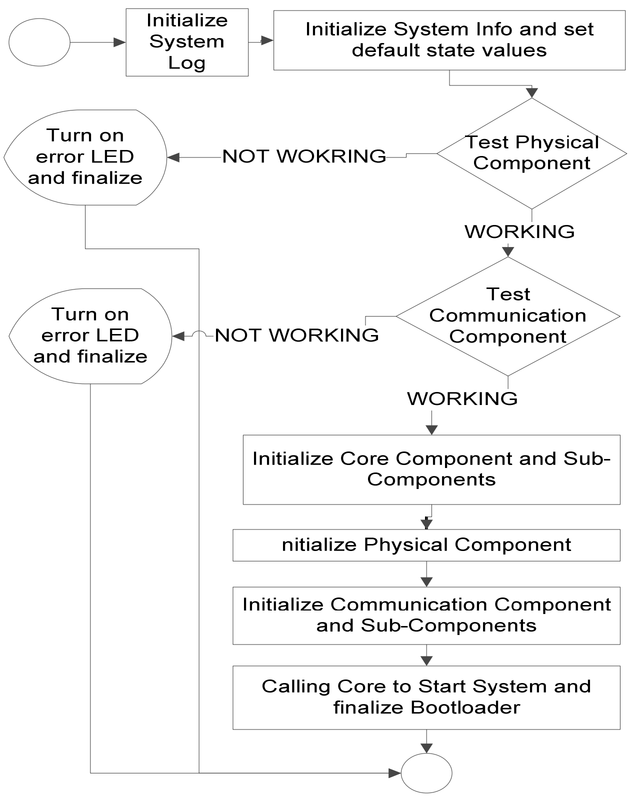

Initialize System Log
Initialize System Info and set default state values
Test Physical Component
NOT WOKRING
Turn on error LED and finalize
WORKING
Test Communication Component
NOT WORKING
Turn on error LED and finalize
WORKING
Initialize Core Component and Sub-Components
nitialize Physical Component
Initialize Communication Component and Sub-Components
Calling Core to Start System and finalize Bootloader

Figure 5: Bootloader Activities

3.1.2 LED Codes

During powering on, the bootloader performs several tests to ensure that the system can function properly. The bootloader uses LEDs to indicate the system status. There are four green LEDs that indicate the status. Refer to Table 1 in order to have an idea on the meaning of each LED (LEDs are numbered from right-to-left)

Table 1: LED Codes

LED 1	On: Sensors enabled. Blinking: Testing Sensors. Off: Sensors Failure
LED 2	**On:** Vision enabled. **Blinking:** Testing Camera. **Off:** No Vision
LED 3	**On:** GPS enabled. **Blinking:** Testing GPS. **Off:** GPS failure
LED 4	**On:** Internet Connected. **Blinking:** Testing Connectivity. **Off:** No connectivity
All LEDs	**Sequential Blinking:** Failsafe Mode. **All Blinking at same time:** Testing software integrity

3.2 Remote Access Listener

After the system is booted up, Remote Access listener starts listening on pre-defined port to accept client connections. When a client establishes a connection to the server, Remote Access listener starts the authentication process with the client by using the authentication sub-component. Figure 6 demonstrates part of Remote Access listener activity.

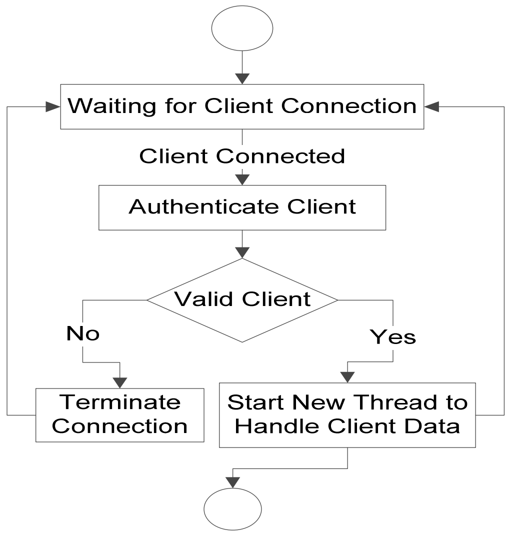


Figure 6: Remote access listener

After the connection is well- established, client can control the vehicle by sending control data defined by the network protocol. Those data are parsed by Remote Access listener which in turns asks the core for access to the required module (i.e. Mobility Module and Environment Module). Remote Access listener doesn't have any access to the physical interface (Phidget). This introduces more safety to the system in case if intruder sent modified packets to control the physical parts.

3.3 Environment Module

When the system is running, Environment Module asks the Core component for physical access. Then, each 100 millisecond the module gather Ultrasonic sensors analog inputs and stores them in the System Info sub-component. Also, it asks the temperature sensor for input each 10,000 milliseconds and stores its data in System Info. Figure 7 demonstrates Environment Module activity.

3.3 Mobility Module

Mobility Module is responsible for controlling DC motors; it provides services such as moving the vehicle to a specified direction. When such service is requested, the module asks the core for physical access and then sends the control data to it. Mobility Module service are requested from Remote Access listener when a client asks to move the vehicle.

3.4 Server Architecture

Server architecture is defined to integrate two main subsystems, the physical subsystem and the logical subsystem. Communication between those two subsystems is only possible through single interface provided by the Phidget module. Phidget module is only accessed by the core component which in turn interfaces other modules. Therefore, if a module needs to access a physical component, it requests access to the Phidgets from the core. Communication components do not have access to the Phidget interface. Instead, they can access the modules (i.e. Environment Module, Mobility Module). All components have access to the System Log components through the core in order to write events and unexpected errors. System Info is part of the core. See Figure 8 for the server state machine and Figure 9 for the server details.

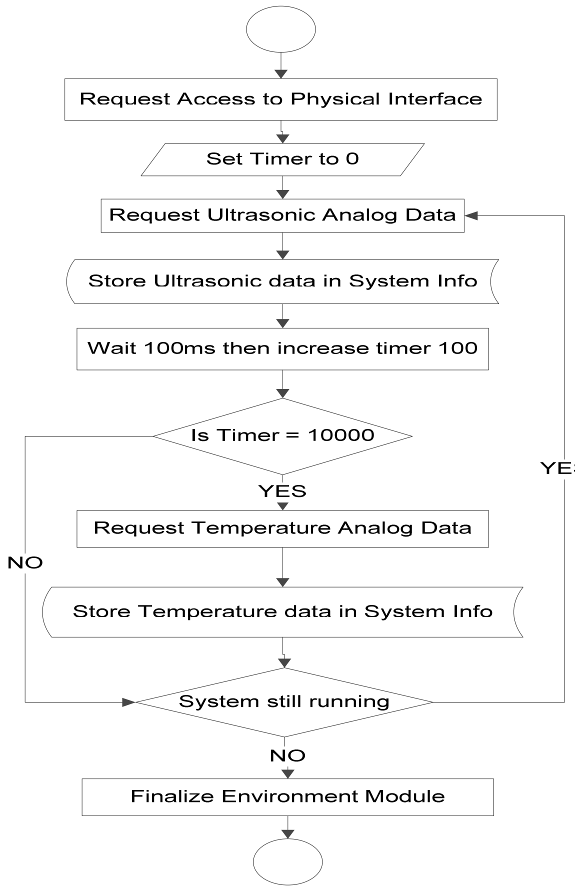

Request Access to Physical Interface
Set Timer to 0
Request Ultrasonic Analog Data
Store Ultrasonic data in System Info
Wait 100ms then increase timer 100
Is Timer = 10000
YES
NO
Request Temperature Analog Data
Store Temperature data in System Info
System still running
YES
NO
Finalize Environment Module

Figure 7: The environmental module

3.5 Client Architecture

Client software will be running in the operator computer. The software is basically a GUI application that establishes a connection to the server residing in the vehicle; each control command is sent through the connection and must conform to the specified network protocol. Figure 10 illustrates the client components and their relationships.

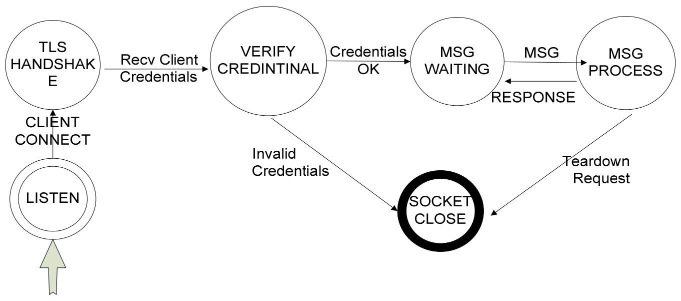


Figure 8: The server state machine

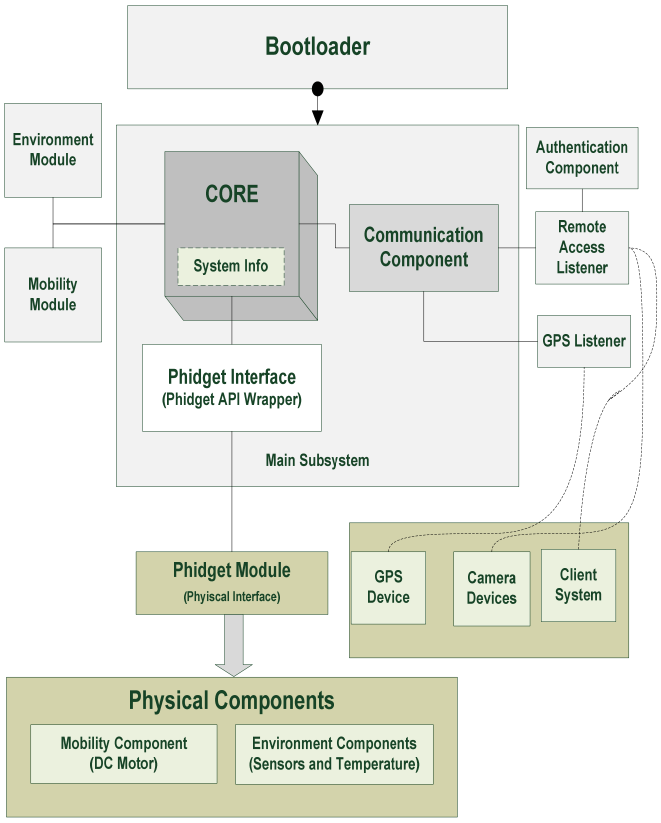


Figure 9: Server architecture

4 Intellibot Security

Since wireless connections are used to remotely control and manage the Intellibot, security is an important issue considered in our work. The security policy is divided into three main categories.

4.1 Channel Encryption

The connection between the server and the client is made through Wireless channel (IEEE 802.11g). This poses some security threats, since the wireless channel can be easily eavesdropped, intruders can sniff packets that are flying on the air to obtain critical information. To overcome this problem, we adapted TLS standard to encrypt the communication channel between the server and the client. Man-in-the-middle attacks cannot be performed without compromising all or of some of the keys. With prober protection on the server and client machines, such attacks will be very hard.

4.2 Authorization

Clients cannot start taking control of Intellibot before logging in the server. The server adopts multi-user approach, that is, it allows more than one user to login at the same time. However, the same user cannot login at the same time from two or more different machines. Usernames and passwords can be as long as the Java Virtual Machine allows in string variables. Sun's implementation of strings uses arrays of characters, which means, usernames/passwords can be up to 2^{31}-1 characters. With properly chosen username/password, brute forcing attacks are out of the question.

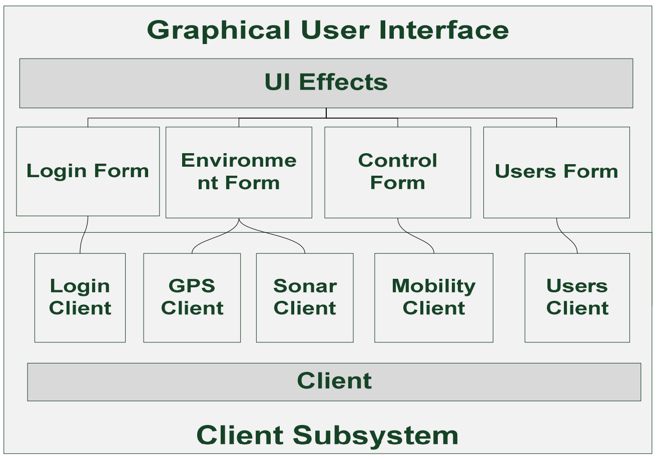


Figure 10: Client Software Subsystem

4.3 Minimal Information Disclosure

If an intruder was able to compromise a client machine, he will not be able to take control of Intellibot without logging into the system with username/password. When an attacker tries to send login requests with invalid username/password combination, the server simply closes the connections without specifying what the error was. The same applies for malformed packets that do not conform to the network protocol. Such packets will be discarded without giving any response at all.

After TLS handshaking is successful (that is verifying that both ends have valid keys). The client will send login username and password. The server will check this information and - if valid – will enter MSG WAITING state, this state waits for messages from the client side and process them according to network protocol specification. When it receives a teardown request it will close the connection.

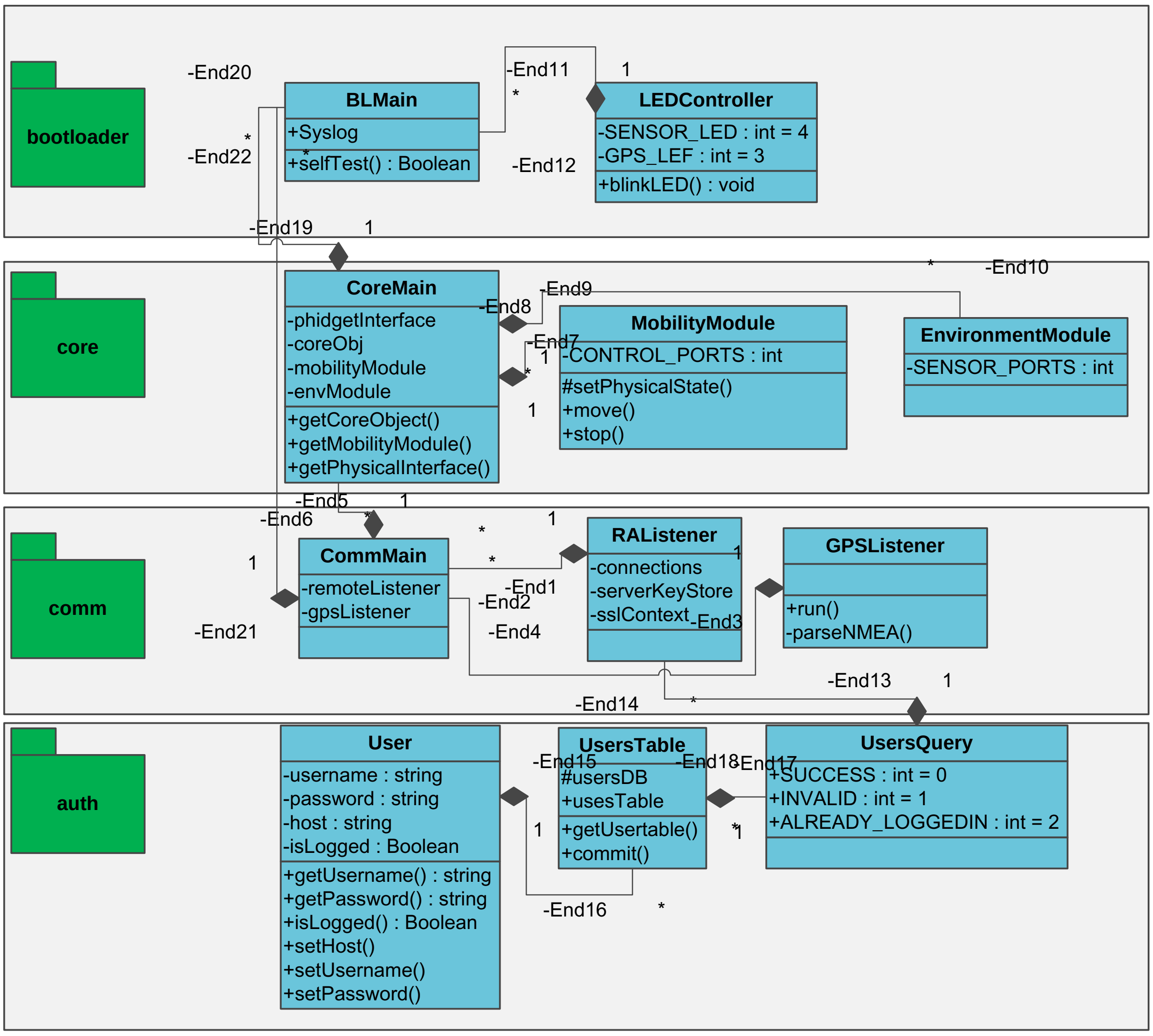


Figure 12: Server UML diagram

5 Efficient image transmission using Jpeg2000 compression

After the extensive image communication of discrete base cosine transform JPEG [4,5], JPEG-2000 still image compression have been introduced and is considered one of the most promising image compression methods [6]. Its superiority in achieving low bit rate compression, error resilience, and other features enabled it to become the tomorrow's compression standard and leads to the JPEG-2000 ISO. The JPEG-2000 ISO contains many parts. In this work, 'JPEG-2000' refers to the compression of 3-bands color still images. As referred to the JPEG abbreviation which stands for Joint Photographic Expert Group, JPEG-2000 codec is more efficient than its predecessor JPEG and overcomes its drawbacks [7]. It also offers higher flexibility compared to even many other codec, such as region of interest, high dynamic range of intensity values, multi component, lossy and lossless compression, efficient computation, compression factor control, etc. The robustness of JPEG-2000 stems from its utilization to the Discrete Wavelet Transform (DWT) in encoding the image data. The DWT exhibits high effectiveness in image compression due to its support to multiresolution representation in both time and frequency domains. In addition, the DWT supports progressive image transmission, region of interest coding, etc [8]. JPEG-2000 are enhanced for better quality and image compression [9,10].

JPEG-2000 is used in this project to enable transmission snap shots taken by the robot camera into the server. These snapshots are used by the server for analysis and detection. Moreover, video streams are also compressed by JPEG-2000.

5.1 Internet Cameras
Three internet cameras are mounted on Intellibot to provide vision to the controller. Those cameras stream two types of images: JPEG and MJPEG. To be able to obtain to the stream you need to establish a connection to fixed IP address (the IP address is configured through camera's web interface) [12]. In our network model, the cameras have fixed private class C IP addresses. Assuming there are three cameras mounted on Intellibot, their IP configurations should be as follows:

Camera1: 192.168.0.10
Camera2: 192.168.0.20
Camera1: 192.168.0.30

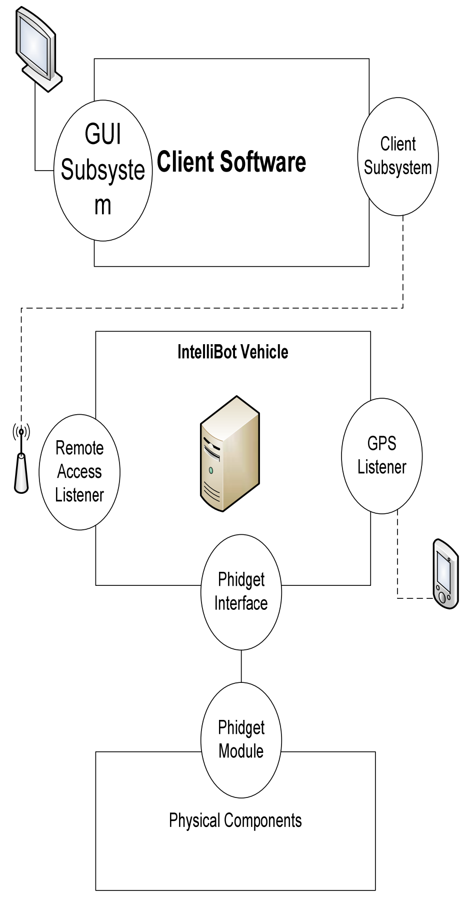
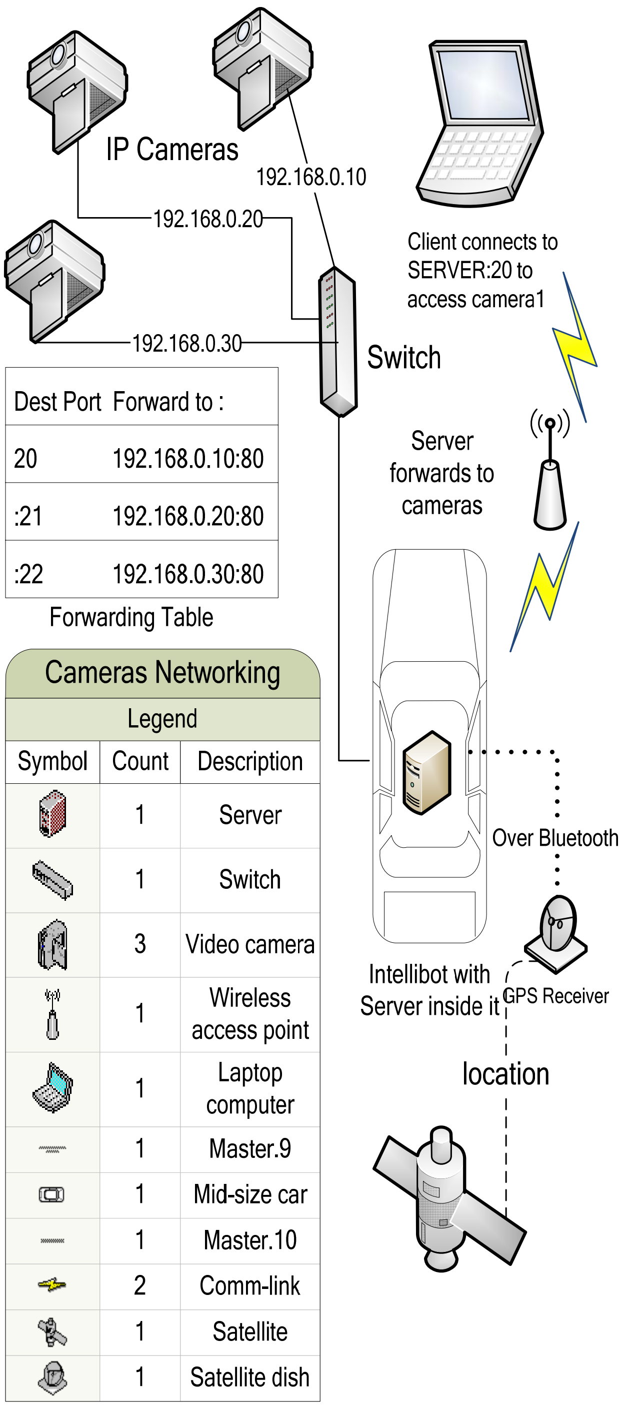


Figure 12: System Interfaces

An Ethernet switch is used to link these cameras together and to the Intellibot server machine. If the client needs to access the cameras, it should provide authentication details (username/password) to be able to obtain the stream. To establish a connection from the client to the IP camera, the client should connect to the Intellibot server's IP with a known port number, the server will forward the connection to the camera based on the port number. See Figure 13 for more details.

6 Obstacle detection
To perform obstacle detection we used two concepts stereo vision and ultrasound detectors. The stereo vision is accomplished by using two Day/Night cameras mounted on the Intellibot which is used to detect 3D objects far from the Intellibot. The ultrasound detectors (4 pieces) that are used to detect objects around the Intellibot can be used to sense objects up to 6 meters. Neural network are used to solve data non-linearity [11].


IP Cameras
192.168.0.10
192.168.0.20
192.168.0.30
Client connects to SERVER:20 to access camera1
Switch
Server forwards to cameras
Over Bluetooth
Intellibot with Server inside it
GPS Receiver
location

Dest Port	Forward to :
20	192.168.0.10:80
:21	192.168.0.20:80
:22	192.168.0.30:80

Forwarding Table

Cameras Networking

Legend

Symbol	Count	Description
	1	Server
	1	Switch
	3	Video camera
	1	Wireless access point
	1	Laptop computer
	1	Master.9
	1	Mid-size car
	1	Master.10
	2	Comm-link
	1	Satellite
	1	Satellite dish

Figure 13: Intellibot structure overview

To perform the stereo matching in order to exactly detect the 3D objects, image restoration and/or enhancement using Laplacian and canny edge detectors are performed in order to reduce effects of Intelibot movement as well as defocusing degradation. On the other hand, signals obtained from the ultrasound are used to train a multi-layer neural network in order to detect the ranges of nearby objects. Both the stereo vision and the ultrasound detection method are found to be successful to a rate of 97%. Moreover, image mosaic (images stitching) algorithm based on genetic algorithm estimation of affine parameters are used to generate full seen image from stereo images. The stitched image gives a full wider view of two different images on the condition that they have some overlapped region. The structure of the proposed Intellibot is further illustrated in Figure 13 and the Intellibot Server APIs are shown in Appendix A.

7 Conclusions

In this paper, we proposed an intelligent multipurpose robot called Intellibot that is manufactured with simple hardware to yield cutting edge technology performance. The Intellibot uses Jpeg2000 image compression to enable efficient image transmission, stereo matching to enable range finding of 3D objects, neural networks to predict nearby objects detected by ultrasound sensors. All the video scanned by the Intellibot is transmitted into a server that is administrated remotely via the wireless Internet or other connections.

References:

[1] Nourbakhsh I., Hamner E., Lauwers T., Bernstein D., and DiSalvo C.F., A Roadmap for Technology Literacy and a Vehicle for Getting There: Educational Robotics and the TeRK Project, Proceedings of IEEE RO-MAN 2006, September, 2006.

[2] Schreckenghost D., Fong T.W., and Milam T., Human Supervision of Robotic Site Survey, 6th Conference on Human/Robotic Technology and the Vision for Space Exploration, February, 2008.

[3] Scherer S., Singhm S., Chamberlain L.J., and Elgersma M., Flying Fast and Low Among Obstacles: Methodology and Experiments, The International Journal of Robotics Research, Vol. 27, No. 5, pp. 549-574, 2008,

[4] Skodras A., Christopoulos C., Ebrahimi T., "JPEG2000: the upcoming still image compression standard", Pattern Recognition Letters, Vol. 22 No. 12, pp. 1337-1345, 2001.

[5] Taubman D. and Marcellin M.W., “JPEG2000: standard for interactive imaging”, Proceedings of the IEEE, Vol. 90, No. 8, pp. 1336-1357, 2002.

[6] ISO/IEC 15444-1, "Information Technology- JPEG-2000 Image Coding System, Part 1, Core Coding System," 2000.

[7] Ebrahimi F, Chamik M, and Winkler S., "JPEG vs. JPEG-2000: An Objective Comparison of Image Encoding Quality". Proc SPIE Applied Digital Image Processing, Vol. 5558, pp.:300–3008, 2004.

[8] Antonini, M., Barlaud, M., Mathieu, P., Daubechies, I., "Image Coding Using Wavelet Transform", IEEE Transactions on Image Processing, Vol. 1, No. 2, pp. 205-220, 1992.

[9] *Al-Rawi M. S., Abu-Dalhoum A.-L, Salah Y. Al-MobaideenW., Khoury A.* Using genetic algorithms to find new color transformations for jpeg-2000 image compression, 12th WSEAS Int. Conf. on Computers, Crete, Greece, 2008.

[10] Christopoulos C., Skodras A., and Ebrahimi T., "The JPEG2000 still image coding system: An overview", IEEE Trans. on Consumer Electronics, Vol. 46, No. 4, pp. 1103-1127, 2000.

[11] *Abu-Dalhoum A.-L, Al-Rawi M.*, Non-Linear Data for Neural Networks Training and Testing, WSEAS transactions on information science & applications, Vol. 3, No. 2, pp.410-416, 2006.

[12] Jackson J. D., Callahan D. W.; Appleby D. S., Callahan L. B. Architecture for a TCP/IP based robotic protocol, WSEAS-Transactions-on-Communications, Vol. 5, No.1, pp. 98-103, 2007.

Appendix A: Intellibot Server APIs

This appendix includes the Javadoc generated prototypes. This javadoc should be used as reference that shows the standard APIs that the Intellibot might use.

Many Methods are inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Summary

BLMain()   Initializes output objects such as Phidget interface for LEDs and Syslog for writing events

Method Summary

`void`	**close**() Called when bootloading is done
`static void`	**main**(java.lang.String[] args) Starts the bootloading operations, called by JVM
`void`	**selfTest**() Performs self test operations
`boolean`	**testGPS**() Tests if GPS is attached
`boolean`	**testIntegrity**() Tests program integrity
`boolean`	**testInternet**() Tests internet connectivity
`boolean`	**testMobility**() Tests if mobility controls are attached
`boolean`	**testSensor**() Tests if sensors are attached
`boolean`	**testVision**() Tests if camera is attached

Method Summary

`static CoreMain`	**getCoreObject**() Returns reference to core instance.
`static CoreMain`	**getCoreObject**(com.phidgets.InterfaceKitPhidget phidget) Returns reference to core instance, create one if there is no already created one.
`EnvironmentModule`	**getEnvironmentModule**() Returns reference to environment module
`MobilityModule`	**getMobility**() Returns reference to mobility module
`com.phidgets.InterfaceKitPhidget`	**getPhysicalInterface**()

Constructor Summary

EnvironmentModule()

Method Summary

`void`	**run**() Checks the environment sensors each 1000ms and stores inputs in System Variables

Constructor Summary

`MobilityModule()`

Method Summary

`boolean`	`move(java.lang.String moveDirection)` Moves the bot
`boolean`	`stop()` Stops the bot movement

Constructor Summary

`RAListener()` Create a Server that listens on the given port.

Method Summary

`static void`	`main(java.lang.String[] args)`
`void`	`run()` Starts accepting connection and pass the socket to client handler

Nested Class Summary

`static class`	`SysInfo.Mode`

Constructor Summary

`SysInfo()`

Method Summary

`static SysInfo.Mode`	`getState(java.lang.String key)` Returns system state
`static java.lang.Object`	`getVar(java.lang.String key)` Returns system variable value
`static void`	`initSystemInfo()` Initializes System Info objects
`static void`	`setState(java.lang.String key, SysInfo.Mode value)` Set system states value
`static void`	`setVar(java.lang.String key, java.lang.Object value)` Set system variable value

Enum Constant Summary

`DISABLED`
`ENABLED`

FORCEFULLY_DISABLED

Method Summary

static SysInfo.Mode	**valueOf**(java.lang.String name) Returns the enum constant of this type with the specified name.
static SysInfo.Mode[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Constructor Summary

Syslog()

Method Summary

static void	**close**() Closes and destructs all objects used in Syslog class
static void	**initSyslog**() Constructor initializes required objects to start writing on the syslog
static void	**writeEvent**(java.lang.String event) writeEvent is required to write event in the syslog
static void	**writeEvent**(java.lang.String event, java.lang.Exception ex) another version of writeEvent requires exception reference to write in the file

Constructor Summary

User()

Method Summary

java.lang.String	**getHost**() Gets the host.
java.lang.String	**getPassword**() Gets the password.
java.lang.String	**getUsername**() Gets the username.
boolean	**isLogged**() Checks if user is logged.
void	**setHost**(java.lang.String host) Sets the host.
void	**setLogged**(boolean isLogged) Sets the logged.
void	**setPassword**(java.lang.String password) Sets the password.
void	**setUsername**(java.lang.String username) Sets the username.

Field Summary

static int	**ALREADY_LOGGEDIN**
static int	**INVALID**
static int	**SUCCESS** Error codes for authUser

Constructor Summary

UsersQuery()

Method Summary

static boolean	**addUser**(java.lang.String username, java.lang.String password) Adds user to the database and returns result
static int	**authUser**(java.lang.String username, java.lang.String password, boolean checkLoggedIn) Authorizes user (or not).
static boolean	**changePassword**(java.lang.String username, java.lang.String newPassword) Changes user's login password
static void	**delUser**(java.lang.String username) Delete user
static User	**getUser**(java.lang.String username) Returns User of given username
static java.lang.String	**getUsersList**() Returns users list

Field Summary

static UsersTable	**usersTable** A static reference to users database

Method Summary

void	**commit**()
static UsersTable	**getUsersTable**()

Field Summary

static CLICommand	**SET**
static CLICommand	**SHOW**

Constructor Summary

CLICommand()

Method Summary

abstract void	**execute**(java.lang.String[] params)