UR SMARTHOME®

(Critical Design Review & IO Demonstration)

University of Alberta

Department of Electrical & Computer Engineering

CMPE 490 – Design of Microprocessor-based Systems

Dr. Duncan Elliott

Sina Karimi (skarimi@ece.ualberta.ca)
Hector Nuñez (hnunez@ualberta.ca)
Ryan Campbell (ryan.campbell@ualberta.ca)

Preferred lab dates: Monday, Tuesday

Abstract

A ‘smart home’ is a home that allows monitoring and controlling of various systems of the house. This can include controlling lighting, monitoring temperature (and adjusting furnaces or AC systems based on the temperature, for example), video surveillance and many others.

For this project we created a ‘smart home’ system controlled by an ARM7-based micro-controller. Our system allows web-based access to monitor and control multiple subsystems. Currently this includes motion detection and video surveillance, but could be easily expanded to include temperature control, lighting, and others.

We successfully interfaced with an Ethernet controller, a UART-based camera, and motion sensor, as well as developing a HTTP server and FTP client atop a Real-time Operating System (uC/OS-ii).

Table of Contents

Declaration of Original Content

The design elements of the project and report are entirely the original work of the authors and have not been submitted for credit in any other course, except as following material:

- Atmel AT91 library
- MicroC/OS-ii RTOS
- WIZnet socket API

Hector Nuñez [illegible] April 11/2010

Sina Karimi [illegible] April 11th/2010

Ryan Campbell [illegible] April 11/10

Functional Requirements

The UR SMARTHOME® is a home Central Control System (CCS) that controls and monitors different devices contained in a house/office. These devices are a video camera, a motion control sensor, and a data storage device.

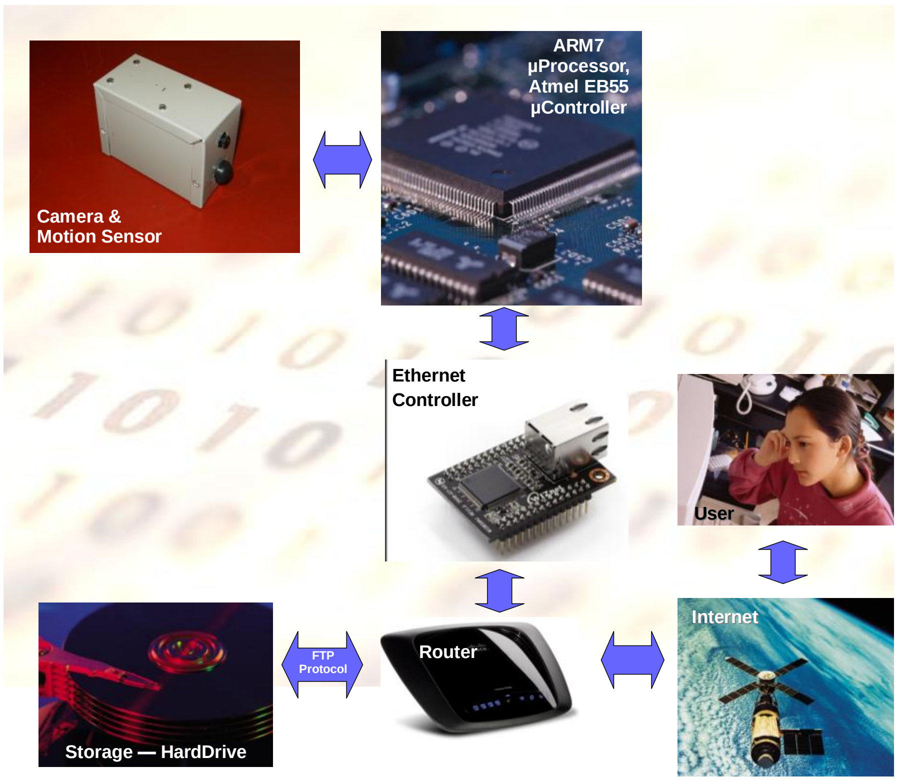
The CCS must also have a web interface, giving the opportunity of controlling and monitoring the system from a remote location via the internet.

The camera system has two modes of operation, security mode and surveillance mode. When in security mode the camera will be off by default, but as soon as movement in the room has been detected through the motion sensor, the camera will turn on and the CCS will send video data to the storage device until the motion sensor stops detecting movement for a determined period of time. This video can be later viewed through the web interface. And for the surveillance mode, the user can turn on and off the camera and receive live video through the web user interface, and have the option to record the video in storage.

All of these requirements were met. We successfully developed a basic home automation device which controls a camera and motion sensor via a web interface. For data storage, we developed our own FTP client to store and retrieve data on a PC-based FTP server.

Because of our chosen hardware, we were forced to compromise on a few things, such as the streaming video, since our camera is only able to capture images at a rate of approximately 1 frame per second. Also, we were unable to get the FLASH-based version of our software working correctly within the time allowed. This meant that the code had to be re-downloaded onto the micro-controller after every power-cycle.

Description of Operation



The above block diagram shows the various components of the system. A more in-depth overview can be found in the *Software Design* and *Schematics* sections.

Web server

As mentioned in the functional requirements, we had to interface with an Ethernet module. We used a WIZnet Ethernet module with a hardware TCP/IP stack to ease development. The manufacturer supplies a simple socket API (superficially similar to BSD socket API) that allows us to interface with the module.

We then designed a simple HTTP server (implementing the bare minimum to serve a page) which provides a web-based user-interface to allow monitoring the various modules such as the temperature and motion sensor. It allows the user to view captured images, either as a slide show or individual images. It also allows the user to turn the motion sensor on or off, as well as take a picture manually.

Given these DC characteristics (refer to I/O section), no voltage or current handling hardware is required between ARM7 and WIZnet circuitry. Please refer to the schematic section for interfacing details.

Camera

A USART-based camera will be triggered by the motion sensor to capture images to a storage device. The user can view and take images from the camera at any time from the web interface. Depending on the device we choose, we may have to store raw video, or implement an encoder ourselves. If we chose a camera with a JPEG encoder built-in, we will be able to skip that step.

The camera is connected through serial communication. There is a set of instructions sent through this serial interface that control the initialization, image request, etc. For more information on interfacing details, please refer to the schematic section.

Given the DC characteristics (refer to I/O section); there is no need to regulate voltage between the camera and the ARM7.

Storage

For the storage for SMARTHOME, we decided that hosting our files on a simple FTP server would be the most flexible approach, and design a FTP client based on the WIZnet socket API to retrieve and store images. This means we are not restricted to the resources on the micro-controller, and can store a virtually unlimited number of images.

In particular, we use a Linux-based PC with vsftpd (Very Secure FTP Daemon) as our FTP server.

Sensors

The motion sensor is configured to pull the interrupt line on IRQ1 low when motion is detected. This causes an ISR to run, which triggers the camera to begin taking pictures.

Given the DC characteristics (refer to I/O section); there is no need to interface the motion sensor with the ARM7. For more interface detail information on the motion sensor, please refer to the schematic section.

Hardware requirements

The main board (Atmel AT91EB55800A) is used as the main processor to control all other hardware in the system.

To be able to host a web server, it is required to be interfaced with an Ethernet controller. The WIZnet® Ethernet Controller module (NM7010A-LF) is the preferred option since it has a built in TCP/IP stack required for communicating through FTP protocol. This module is interfaced through the EBI (external Bus Interface) on the Atmel board. Once this Ethernet controller is interfaced, the system, can communicate with a network and also with the file server.

To store the image data received from the camera, the capacity required on the fileserver must be enough to hold large amounts of data. Since the average image size is 8000 bytes, and the system stores an image approximately every 2 seconds, make sure there is enough space to store about one day of image data.

$$\frac{86400\ sec}{1\ day} \times \frac{1\ image}{2\ sec} \times \frac{8000\ bytes}{1\ image} = 345.6\frac{MBytes}{day}$$

The image capturing subsystem uses a serial communication camera is interfaced with the micro-controller through the UART ports. This camera has a built in JPEG encoder that can be used to reduce the image data payload. Each camera is paired with an infrared digital motion sensor that is independently connected to the interrupt inputs in the micro-processor. The motion should have a range grater than 5 meters.

A Ethernet router/switch can be used to inter-connect the WIZnet and data storage units. By using this router, it adds security to the system from anybody outside the local network.

Parts List

WIZnet Ethernet Network Module (NM7010A-LF) Rev. 2.0

Price $21.99
Supplier: Saelig
Datasheet: http://www.sparkfun.com/datasheets/DevTools/WIZnet/DEV-09472-Datasheet_V_1.1.pdf
Order status: In stock here

Description: Ethernet module with a built-in hardware TCP/IP stack and socket programming interface. Connects to the ARM board through the EBI.[1]

Relevant specs:
- Built-in TCP/IP stack

CoMedia ltd. C328-7640 JPEG Camera

Price $54.95
Supplier: Sparkfun
Datasheet: http://www.sparkfun.com/datasheets/Sensors/Imaging/C328.pdf
User manual: http://www.sparkfun.com/datasheets/Sensors/Imaging/C328_UM.pdf

Description: UART-based CMOS camera with built-in JPEG encoder which relieves us from implementing or porting our own JPEG decoder as well as offloading the ARM CPU.[2]

Relevant specs: [2]
- Configurable resolution (up to 640x480)
- configurable color depth
- configurable compression
- power consumption: 60mA * 3.3V []

Panasonic® MP Motion Sensor (Passive Infrared Type) “NaPiOn”

Price $42.53
Supplier: Digikey
Datasheet:
http://www.panasonic-electric-works.com/peweu/en/downloads/ds_61802_0002_en_napion.pdf

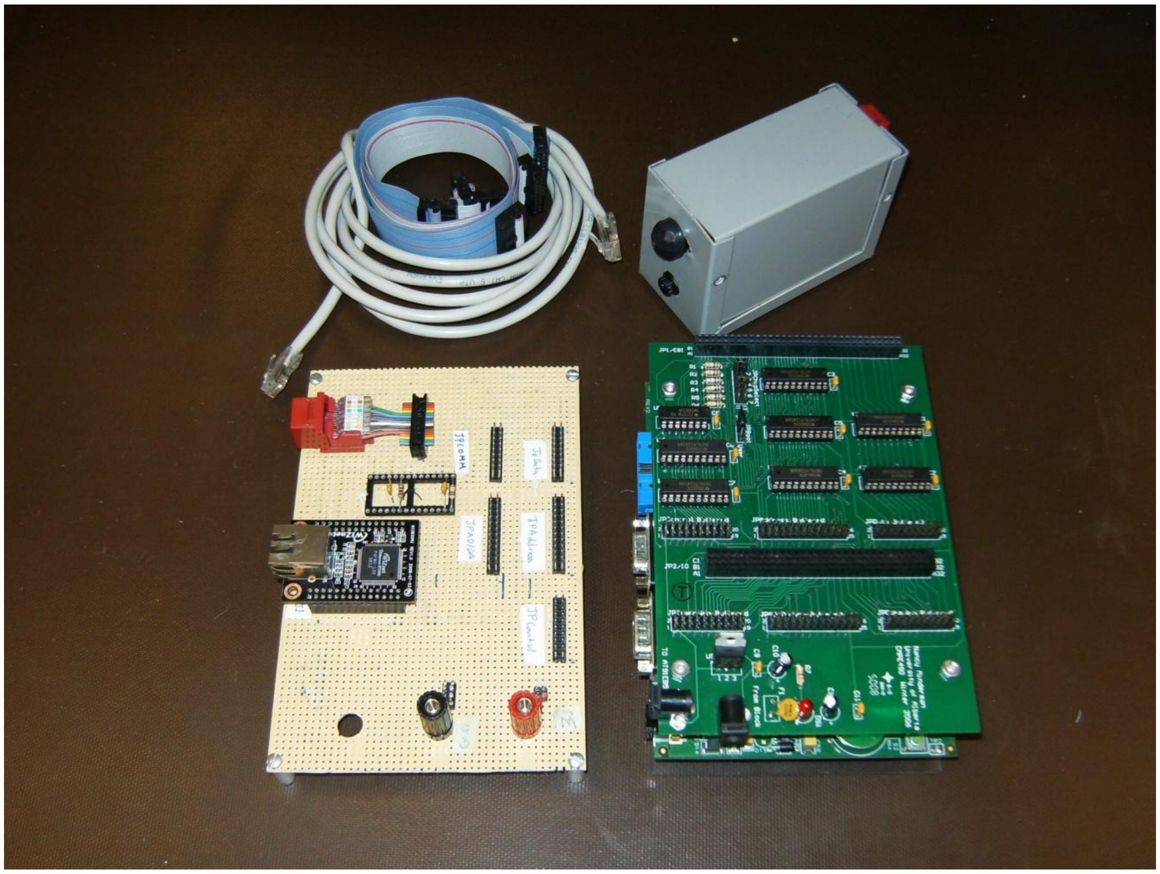
Description: Infrared digital motion sensor, it can detect up to 10 meters radius. 3.o to 6.0 V operation.

Data Storage

FTP Server
Price: $0

An old computer is used as an FTP file server to store data.

UR SMATHOME v1.0
DATA SHEET



Abstract

UR SMARTHOME is a new product that allows the user to remotely interact with you r home security system via web interface.
On this version, the user can access the web page and activate or deactivate the camera surveillance system, where on motion detection the system records images in the storage space. Also, the user can take and preview stored pictures and review them trough the web site form any internet terminal.

UR SMARTHOME Block diagram[1]

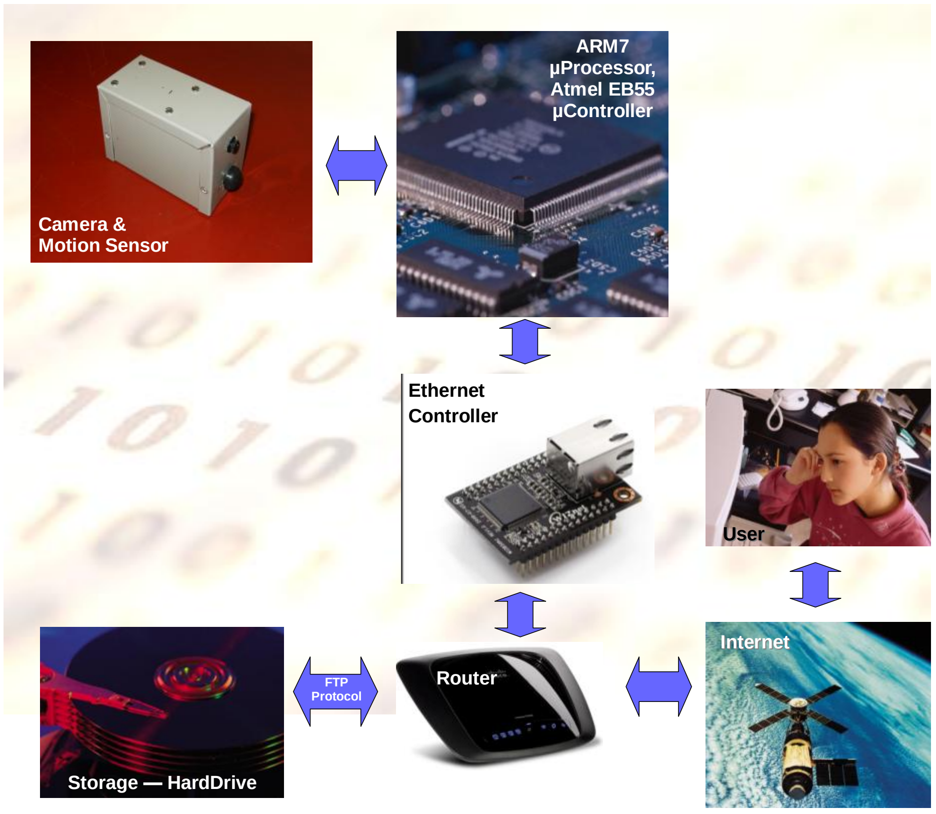


Figure 1, System Block diagram.

Components

UR SMARTHOME is composed of 2 units:

- Atmel® AT91EB55800A unit
- Interface Unit
- Camera Unit
- Storage Unit
- Router/Switch

Atmel® AT91EB55800A Unit

The Atmel evaluation board is a microcontroller unit based on the ARM7 processor family. For Detail information please refer to AT91EB55800A documentation.
http://www.ece.ualberta.ca/~CMPE490/winter2010/resources.html

Interface Unit

[1] Pictures taken from Microsoft Office XP Media Content and www.linksys.com.

This unit contains all the necessary circuitry to interface the ARM board AT91EB55800A to all other components. The interface board is powered with a DC power supply at 3.3 volts and current limited to 300 mA. Connectors JI1 and JI2 are used to mount the WIZnet Ethernet Module. JI3 contains all filtering and current limiting circuitry. JI4 is the connector used to interface to the camera unit through a CAT5 Straight through Ethernet cable. All other connectors interface to the Atmel micro-controller board.

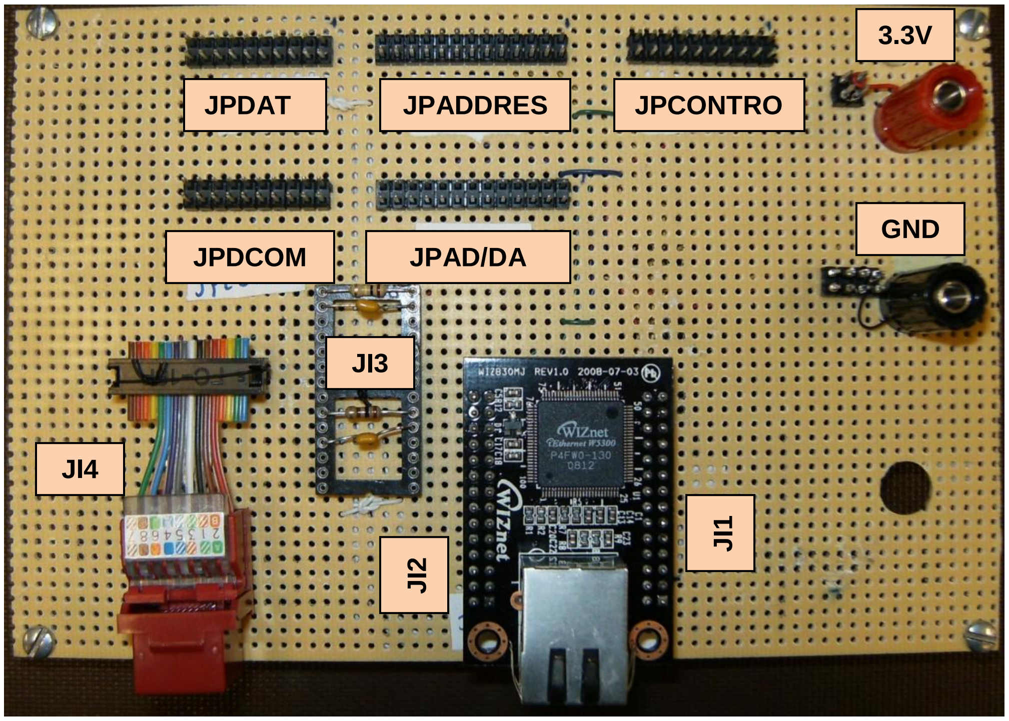


Figure 2, Interface board layout.

Camera Unit

Interface between camera and motion sensor to the interface unit. JC1 has a CAT5 type connector that attaches to the interface unit through a Straight through CAT5 Ethernet Cable. JC4 is the camera connection, and JC3 is the motion sensor connection as well as a BJT switch. And lastly JC2 contains all the filtering and current limiting circuitry. Please refer to figure 3.

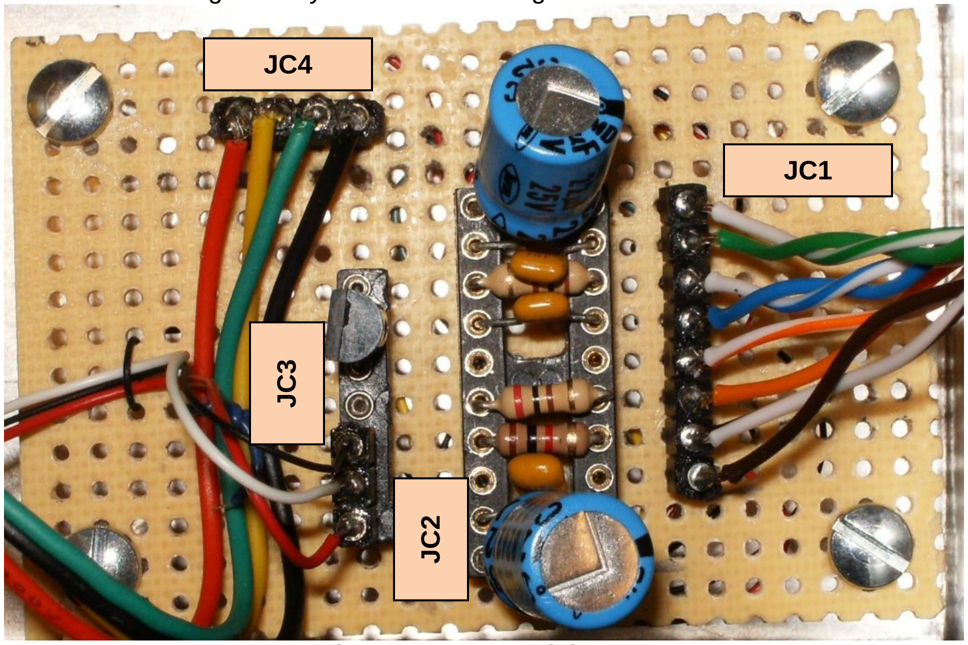


Figure 2, camera unit layout.

<u>Storage Unit</u>

A PC with Ubuntu Server 9.10 is used, make sure to install VSFTPD (Very Secure FTP Daemon).
UR SMARTHOME is configured to connect to an FTP Server with the following:

FTP Server IP:	192.168.10.101
FTP Username:	default
FTP Password:	qwe123
HTTP root directory:	/home/default/SmartHome/www/
HTTP index file:	/home/default/SmartHome/www/index.htm

Note1: Index file should be called index.htm
Note2: Any storage unit that supports FTP communication can be used instead of a PC.

<u>Router/Switch</u>

Any router/switch can be used to connect the WAN, storage unit, and WIZnet.
Note: This is an optional component. The only critical connection is the WIZnet module to the internet, the storage unit can be anywhere in the network.

Operation Modes:

UR SMARTHOME operates in two modes, User Mode and Security Mode. They are activated or deactivated through the web interface by the user.

User Mode

User Mode is active when the user requires direct control of the camera. The user can take pictures and/or review stored pictures through the web interface.
When user mode is active, motion sensor interrupt handling is disabled, camera and storage are still active waiting on user commands.

Security Mode

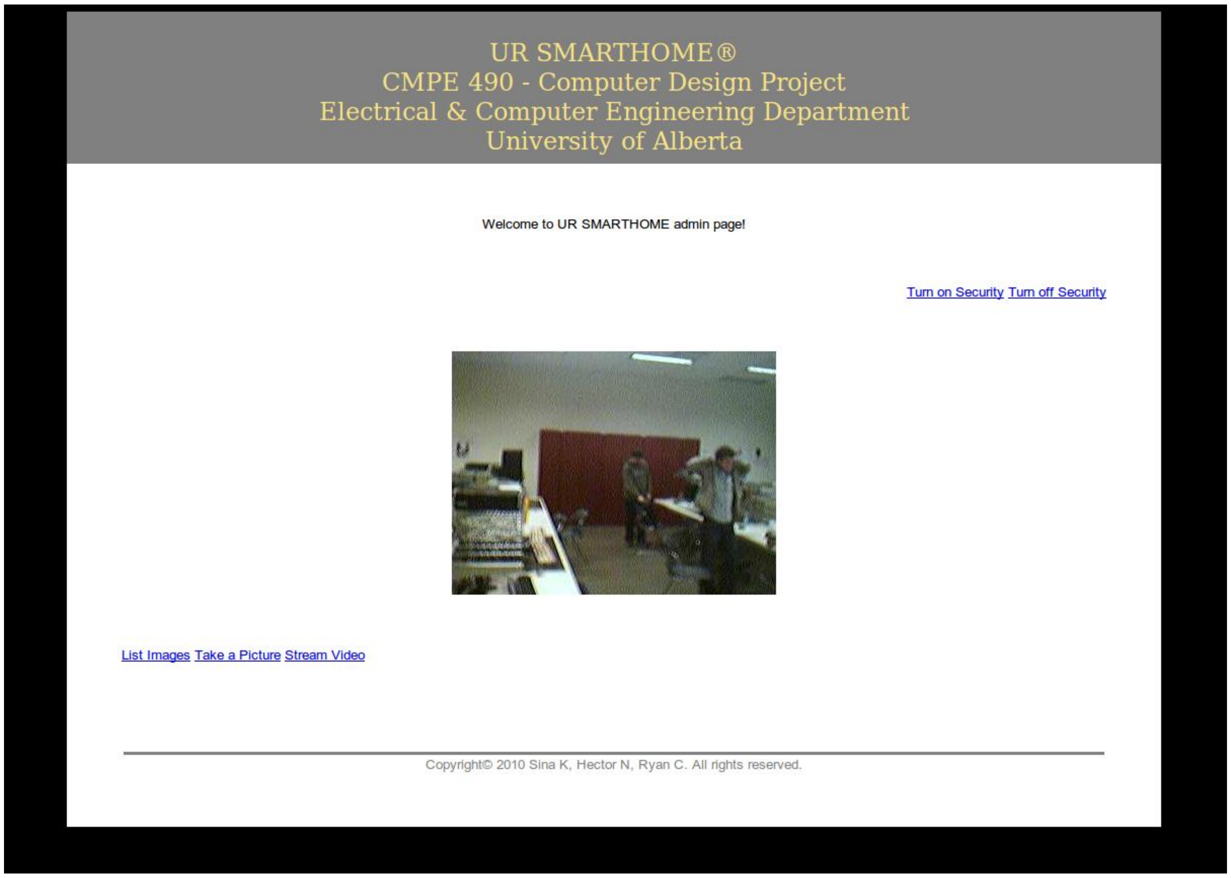
Security Mode is active when user requires automated detection of movement around the camera. When detection occurs, the camera takes continuous pictures and stores them in hard drive.
When Security mode is active, all systems are waiting for either a motion sensed interrupt or any other commands from the user via the web page.

Software Design

The main function will initialize the tasks being used, *HttpdTask* and *MotionSensorTask* and it will then initialize and start the operating system. The HTTPd task will first initialize the WIZnet sockets and the hardware and it will configure the networking options like the MAC and IP addresses and then goes to the infinite loop in which it runs the httpd server. The motion sensor task will initialize the motion sensor and the motion sensor interrupt service routine on the IRQ0 and also the motion sensor semaphore and will go to the infinite loop in which it pends on the semaphore. If the semaphore is available it will proceed to taking a picture and back to pending on the semaphore again. The semaphore will get posted in the motion sensor ISR which is called if the IRQ0 line drops low.

To run the HTTPd server, four http sockets are initialized which will be listening for any connection on port 80. When request is received, it is parsed and processed to find out the file that is being requested. And using the ftp client code the file is retrieved from the ftp server and passed back to the user. In case the requested file is of type html/text the buffer which is retrieved from the ftp server first goes through a subroutine which parses the buffer looking for any instance of text *<!--FUNCTIONCALL functionName-->* and then takes out the whole phrase out of the html and replaces it with the returned buffer from function with name *functionName*. It can be used to generate html code dynamically or perform an action in the code or both. Then the final html file is passed back to the user.

To view a slideshow ‘video’ of the images, we embedded a small amount of JavaScript into an HTML document which allows us to cycle through an array of images periodically (every 2 seconds in our case). Due to time constraints we were unable to dynamically determine the number of images to cycle through (using the above FUNCTIONCALL tag), so compromised by cycling through the first 5 images for demonstration purposes.


UR SMARTHOME®
CMPE 490 - Computer Design Project
Electrical & Computer Engineering Department
University of Alberta

Welcome to UR SMARTHOME admin page!

Turn on Security Turn off Security
List Images Take a Picture Stream Video

Copyright© 2010 Sina K, Hector N, Ryan C. All rights reserved.

Web Interface in Action

The ftp client starts with initiating a control connection with the ftp server on port 21, sending the username and password and passive connection type. Then from the receive buffer we calculate the port for the data connection. Then we start another connection to the ftp server using the calculated port namely the data connection on which the data is received. Then depending on the application the correct request is sent via the control connection and the data is received on the data connection.

To take a picture a subroutine is called in the camera code to sync and initialize the camera and sets the package type and then requests a picture. We initialize the camera to use the built-in JPEG encoder and the resolution and set the package size to be the maximum supported. The package size is regarding the packages in which the camera will send back the picture. Then after requesting the picture we loop on receiving a buffer until the last package is arrived. Then we send the buffer through the ftp client to be saved on the ftp server.

To send and receive buffers through the WIZnet, we are using the WIZnet API which lets us initialize sockets and send and receive on those sockets using their subroutines.

Test Plan

Software

HTTP server:

The HTTP server is mainly hardware-independent, so can be tested independently. The server will implement the minimum functionality required to serve a page.

The following functionality will be tested:

1) "GET <path>" (from browser or telnet connection)
    a. Negative tests (e.g. Invalid path)
    b. Positive tests
        i. Path to image
        ii. Path to text file (.html, .txt)
    c. Special 'paths':
        i. /getPicture
        ii. /listImages

FTP client library:

The FTP client library is also mostly hardware independent. Some code will be platform specific (WIZnet socket vs. BSD socket interface) so will need to be tested on the ARM as well.

The following functionality will be tested:

1) open remote file
2) write remote file
    a. append to file
    b. create new file
    c. overwrite existing file
3) read remote file

4) list directory

Camera library:

The software for controlling the camera is hardware dependent. Once the hardware is verified the following functionality will need to be tested:

1) capture image
2) change camera parameters (this can be verified by analyzing the resulting image):
    a. set resolution
    b. set compression

WIZnet socket interface (provided):

The code provided for the WIZnet should hopefully already be tested and working. A few simple test functions provided with the API should be run:

1) loopback_tcps() to test the TCP server functionality
2) loopback_tcpc() to test the TCP client functionality
3) HTTP/FTP code from above will further test the

Motion sensor:

The code for the motion sensor interrupt handler is hardware dependent. It will be tested by triggering the motion sensor and ensuring the ISR is run.

In addition to testing the individual components, we will have to do some system-level testing once the components have been verified.

For example:

1) Are devices properly protected with semaphores?
    a. E.g. Attempt to take a picture while in the middle of taking another
2) Can we load a web page while taking a picture?

Hardware

WIZnet Ethernet Controller:

The Ethernet controller hardware and interface was tested by a combination of oscilloscope readings and using the above software. In addition, simply pinging the WIZnet from a connected device can help verify that both the hardware and software (configuration code)

Camera:

The camera hardware and interface was tested by a combination of oscilloscope readings and using the above software. We can also analyze the captured image to ensure that they aren't garbage. This can help verify both the hardware and software components of the camera

Motion Sensor:

The motion sensor hardware and interface was tested by a combination of oscilloscope readings (does the line become active when there is motion in front of the sensor?) and using the above software

Result of experiments and characterization

The clock divisor CD that is programmed into the USART registers was calculates based on the length of one command and the ARM7 processor clock speed. Since we have that,

$$BaudRate = \frac{Clk}{16 \times CD} \Rightarrow CD = \frac{Clk}{16 \times BaudRate}$$ [2]

Thus the following values have been calculated using Clk = 33Mhz:

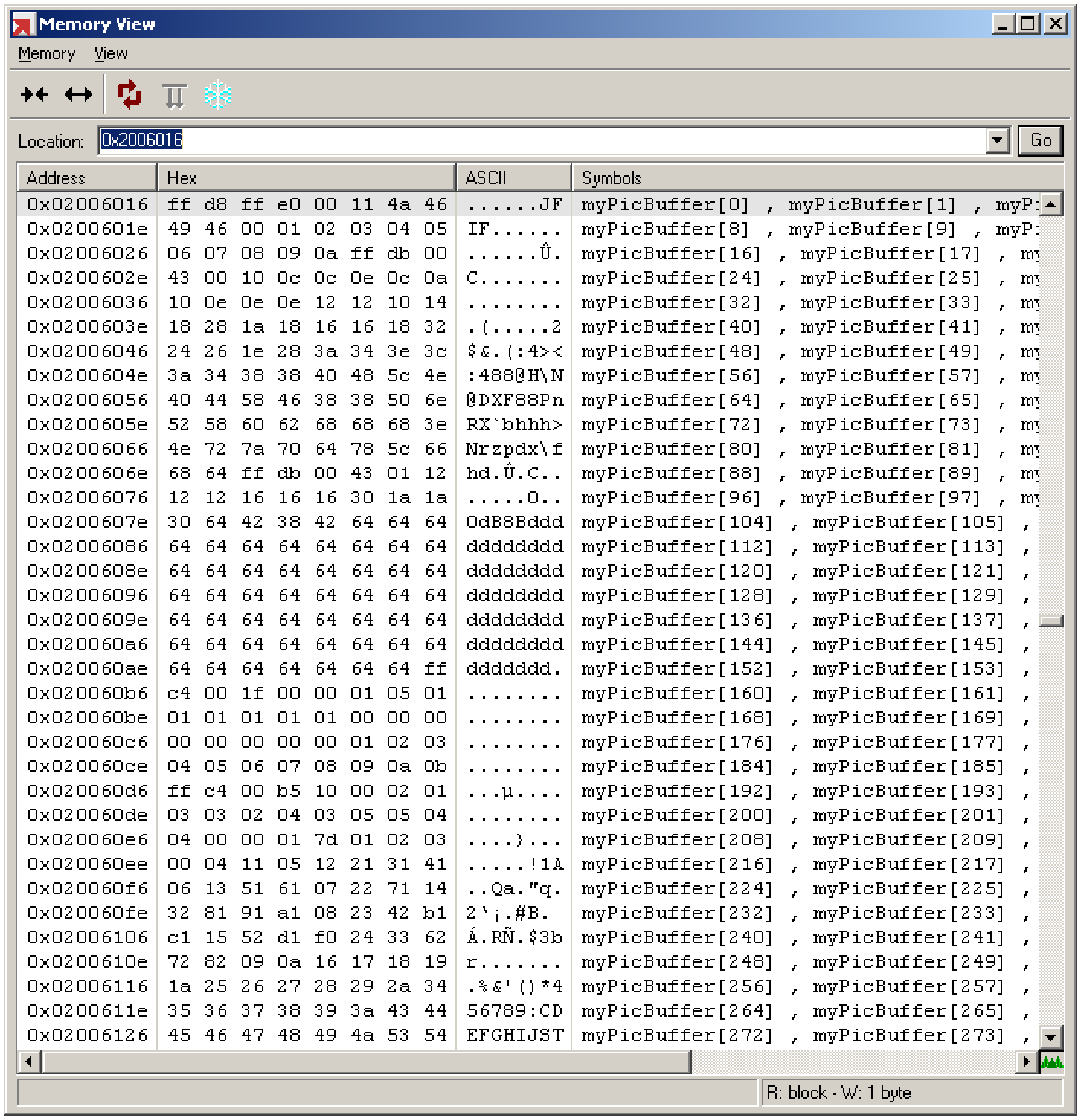
Baud Rate	CD
7200	286
9600	214
14400	143
19200	107
28800	71
38400	53
57600	36
115200	18

After this measurement on the oscilloscope for a baud rate of 14400 showed that the actual transmission was around 14490, so the CD number was changed by trial and error to 134 until it matched the 14400 bits per second measurement.

No testing of other baud rates has been done yet, but the same procedure will be applied to each one.

And further when testing the JPEG data contained in the ARM7 coming from the camera, a halt using the Angel Debugger was done in the code after the picBuffer has been populated and a memory view shows that the buffer contains what a JPEG header should look like.

[2] AT91M55800AComplete.pdf page144



```
Address     Hex                       ASCII     Symbols
0x02006016  ff d8 ff e0 00 11 4a 46   ......JF  myPicBuffer[0] , myPicBuffer[1] , myP:
0x0200601e  49 46 00 01 02 03 04 05   IF......  myPicBuffer[8] , myPicBuffer[9] , myP:
0x02006026  06 07 08 09 0a ff db 00   ......Û.  myPicBuffer[16] , myPicBuffer[17] , my
0x0200602e  43 00 10 0c 0c 0e 0c 0a   C.......  myPicBuffer[24] , myPicBuffer[25] , my
0x02006036  10 0e 0e 0e 12 12 10 14   ........  myPicBuffer[32] , myPicBuffer[33] , my
0x0200603e  18 28 1a 18 16 16 18 32   .(.....2  myPicBuffer[40] , myPicBuffer[41] , my
0x02006046  24 26 1e 28 3a 34 3e 3c   $&.(:4><  myPicBuffer[48] , myPicBuffer[49] , my
0x0200604e  3a 34 38 38 40 48 5c 4e   :488@H\N  myPicBuffer[56] , myPicBuffer[57] , my
0x02006056  40 44 58 46 38 38 50 6e   @DXF88Pn  myPicBuffer[64] , myPicBuffer[65] , my
0x0200605e  52 58 60 62 68 68 68 3e   RX`bhhh>  myPicBuffer[72] , myPicBuffer[73] , my
0x02006066  4e 72 7a 70 64 78 5c 66   Nrzpdx\f  myPicBuffer[80] , myPicBuffer[81] , my
0x0200606e  68 64 ff db 00 43 01 12   hd.Û.C..  myPicBuffer[88] , myPicBuffer[89] , my
0x02006076  12 12 16 16 16 30 1a 1a   .....0..  myPicBuffer[96] , myPicBuffer[97] , my
0x0200607e  30 64 42 38 42 64 64 64   0dB8Bddd  myPicBuffer[104] , myPicBuffer[105] ,
0x02006086  64 64 64 64 64 64 64 64   dddddddd  myPicBuffer[112] , myPicBuffer[113] ,
0x0200608e  64 64 64 64 64 64 64 64   dddddddd  myPicBuffer[120] , myPicBuffer[121] ,
0x02006096  64 64 64 64 64 64 64 64   dddddddd  myPicBuffer[128] , myPicBuffer[129] ,
0x0200609e  64 64 64 64 64 64 64 64   dddddddd  myPicBuffer[136] , myPicBuffer[137] ,
0x020060a6  64 64 64 64 64 64 64 64   dddddddd  myPicBuffer[144] , myPicBuffer[145] ,
0x020060ae  64 64 64 64 64 64 64 ff   ddddddd.  myPicBuffer[152] , myPicBuffer[153] ,
0x020060b6  c4 00 1f 00 00 01 05 01   ........  myPicBuffer[160] , myPicBuffer[161] ,
0x020060be  01 01 01 01 01 00 00 00   ........  myPicBuffer[168] , myPicBuffer[169] ,
0x020060c6  00 00 00 00 00 01 02 03   ........  myPicBuffer[176] , myPicBuffer[177] ,
0x020060ce  04 05 06 07 08 09 0a 0b   ........  myPicBuffer[184] , myPicBuffer[185] ,
0x020060d6  ff c4 00 b5 10 00 02 01   ...µ....  myPicBuffer[192] , myPicBuffer[193] ,
0x020060de  03 03 02 04 03 05 05 04   ........  myPicBuffer[200] , myPicBuffer[201] ,
0x020060e6  04 00 00 01 7d 01 02 03   ....}...  myPicBuffer[208] , myPicBuffer[209] ,
0x020060ee  00 04 11 05 12 21 31 41   .....!1A  myPicBuffer[216] , myPicBuffer[217] ,
0x020060f6  06 13 51 61 07 22 71 14   ..Qa."q.  myPicBuffer[224] , myPicBuffer[225] ,
0x020060fe  32 81 91 a1 08 23 42 b1   2`¡.#B.   myPicBuffer[232] , myPicBuffer[233] ,
0x02006106  c1 15 52 d1 f0 24 33 62   Á.RÑ.$3b  myPicBuffer[240] , myPicBuffer[241] ,
0x0200610e  72 82 09 0a 16 17 18 19   r.......  myPicBuffer[248] , myPicBuffer[249] ,
0x02006116  1a 25 26 27 28 29 2a 34   .%&'()*4  myPicBuffer[256] , myPicBuffer[257] ,
0x0200611e  35 36 37 38 39 3a 43 44   56789:CD  myPicBuffer[264] , myPicBuffer[265] ,
0x02006126  45 46 47 48 49 4a 53 54   EFGHIJST  myPicBuffer[272] , myPicBuffer[273] ,
```

The following values[3] denote:

0xFFD8	SOI		Start of image
0xFFE0	APP0	0x00114A4643460001020304050607080 90A	Reserved for application segments
0xFDB	DQT	0043 00 10 0c 0c 0e 0c 0a…….	Define quantization table(s)
0XFFC4	DHT		Define Huffman table(s)
etc…..			

[3] http://www.digicamsoft.com/itu/itu-t81-36.html

Citations

[1] http://www.sparkfun.com/commerce/product_info.php?products_id=9472

[2] http://www.sparkfun.com/commerce/product_info.php?products_id=9334

[3] http://www.sparkfun.com/commerce/product_info.php?products_id=9587

Datasheets

Motion Detection sensor
http://www.panasonic-electric-works.com/peweu/en/downloads/ds_61802_0002_en_napion.pdf

Camera
http://www.sparkfun.com/datasheets/Sensors/Imaging/C328.pdf
http://www.sparkfun.com/datasheets/Sensors/Imaging/C328_UM.pdf

Wiznet
http://www.sparkfun.com/datasheets/DevTools/WIZnet/DEV-09472-Datasheet_V_1.1.pdf
http://www.sparkfun.com/datasheets/DevTools/WIZnet/DEV-09472-Datasheet_V_1.1.pdf
http://www.sparkfun.com/datasheets/DevTools/WIZnet/W5300%20V1%5B1%5D.1.1%20eng.pdf

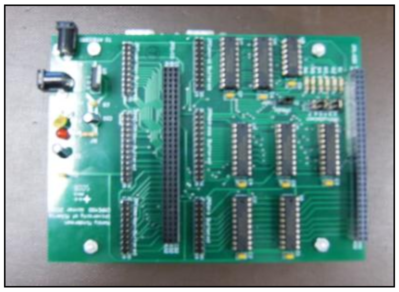
Appendix

Quick-Start Manual

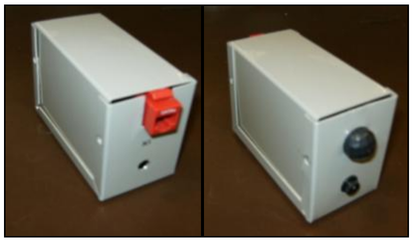
Requirements:

To set up the SMARTHOME system, you need the following components:

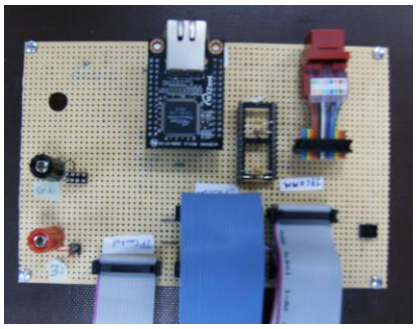
1) Atmel EB55 Micro-Controller (with power brick):



2) Camera/Motion Sensor module:



3) Interface Board (containing interface hardware and WIZnet):



4) 3.3V DC power supply
5) Router (optional)
6) FTP server
7) 2x Ethernet cable
8) Green Hills MULTI IDE
9) Green Hills Slingshot USB-JTAG cable
10) SMARTHOME code

Step 1 - Connect the Hardware:

a) Connect the ribbon cables from the Interface Board to the header with the corresponding label on the Micro-controller's buffer board.
b) Connect the WIZnet Ethernet module to the network with Ethernet cable (through router if desired).
c) Connect the Camera Module to the Interface Board using the second Ethernet cable.
d) Connect the 3.3V power supply to the Interface Board (GND to **black** connector, 3.3V to **red** connector).
e) Connect the 5V power brick to an outlet and plug in the Atmel uC.

<u>Step 2 - Network Configuration:</u>

By default, the SMARTHOME expects the following configuration that may need to be changed:

a) WIZnet configuration (in code):
    - IP: 192.168.10.100
    - NETMASK: 255.255.255.0
    - GATEWAY: 192.168.10.1
b) FTP Server configuration
    - FTP Server IP: 192.168.10.101
    - FTP Username: default
    - FTP password: qwe123
    - HTTP root directory: /home/default/SmartHome/www/
    - HTTP index file: /home/default/SmartHome/www/index.htm
c) FTP Client configuration (in code):
    - The SMARTHOME client must with the above FTP server info.

You can either duplicate these settings in your network or modify the following code:
For *Wiznet configuration*, modify the function wiznet_init() inside SmartHome/src/usr/main.c
For *FTP Client configuration*, modify the beginning of FTP_Client::ftp_open() inside SmartHome/src/usr/ftp.c.

<u>Step 3 - Build and Download to Micro-Controller:</u>

Go into the SMARTHOME code and open the smart_home.gpj project file. This will start up MULTI. From there you can build the code and download to the board as usual.

<u>Step 4 - Connect to SMARTHOME Web Interface:</u>

Open up a web browser and point it to the address you configured for the WIZnet (192.168.1.100 by default). If you are using a router and port-forwarding is enabled, you may also connect to the IP of the router.

Step 5 - Enjoy our new SMARTHOME surveillance/home-automation system!

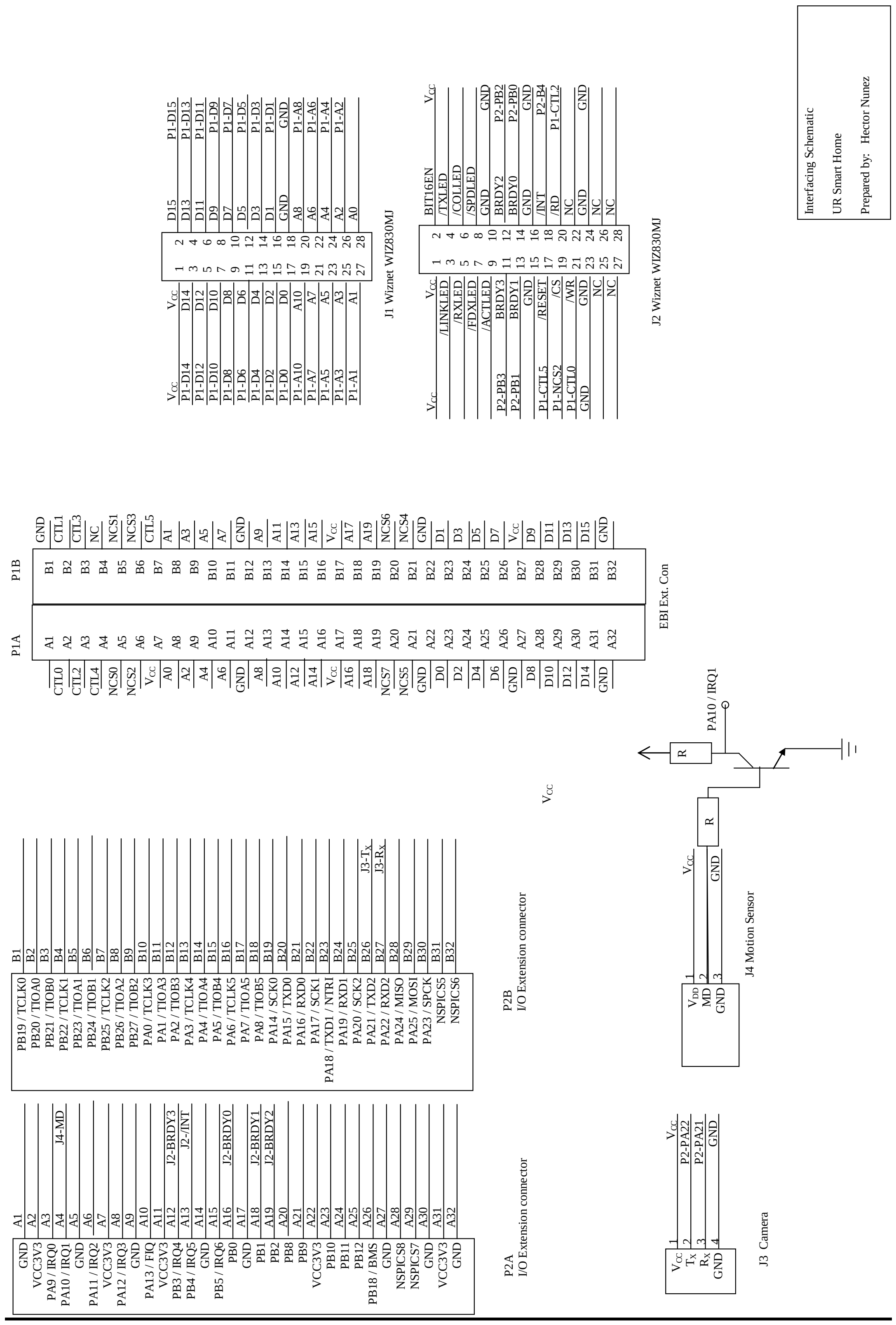
Future Work

There are a lot of possible enhancements that can be done in the future towards this project. Form the long list, improvement to the already existing code would be the first priority. The code is taking advantage of a real time operating system to be able to be easily expandable. However, the already existing code can be improved further to make these expanding hassle free. The infinite while loops with hardware time out can be exchanged with interrupts. And some refactoring can make the code run smoother and with less errors. Also adding more error handling can stop the code from halting unexpectedly. We can also make use of the LEDs more efficiently to show hardware initializations passed or failed.

Thereafter, we can improve the system by interfacing more hardware to be controlled in the household, such as more cameras and motion sensors, temperature sensors, light switches, controls for Air-conditioning and the furnace and so on. Since the memory on the board is limited, there can be some external ram interfaced to make increase the stack and heap size. Another processor can also be interfaced just to handle the image processing so we can encode images into a video.

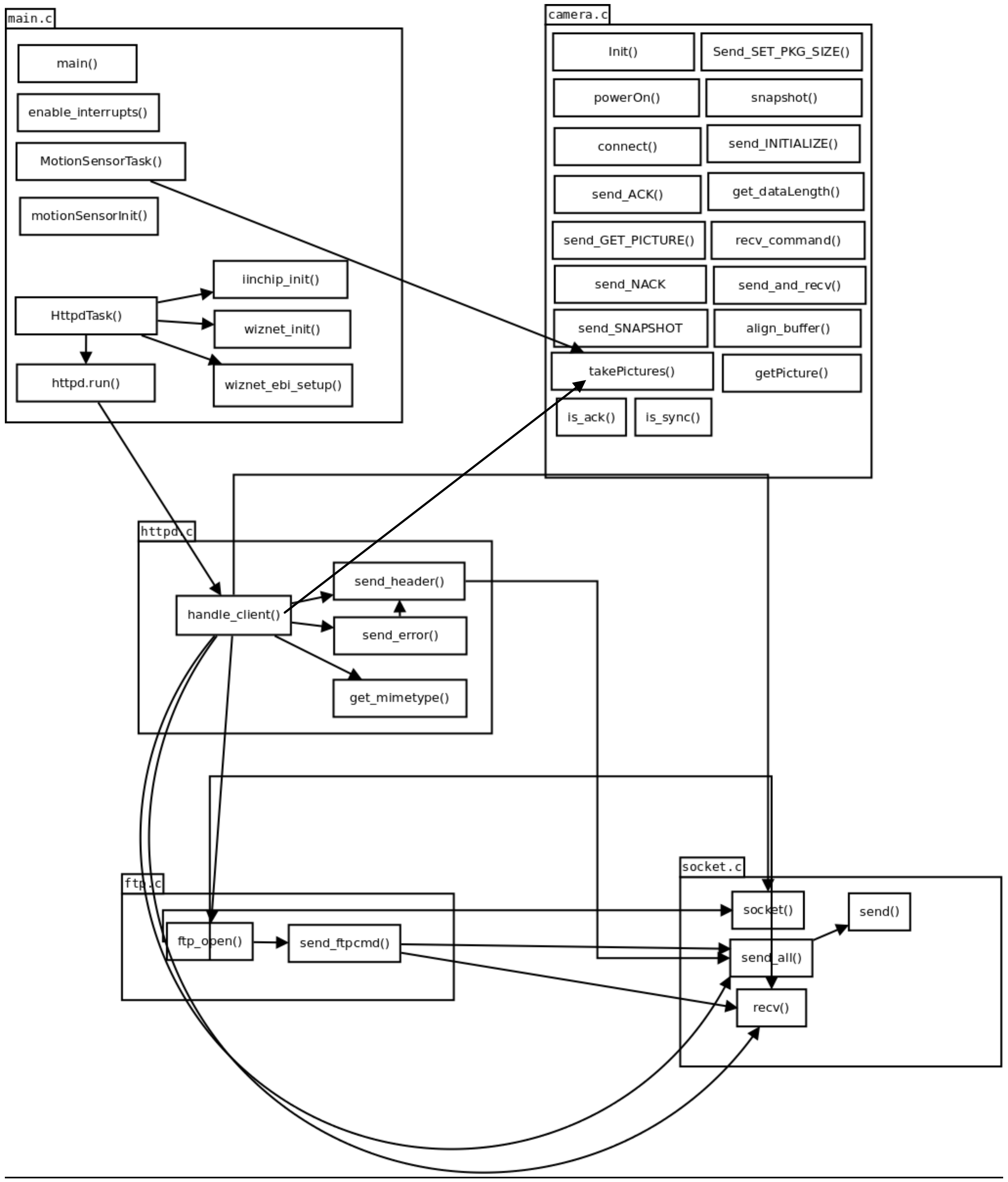
Further we could add some code that would notify a list of users through email or SMS (Standard Messaging System) about the ongoing events in the house.

Schematics



Interfacing Schematic
UR Smart Home
Prepared by: Hector Nunez
J1 Wiznet WIZ830MJ
J2 Wiznet WIZ830MJ
P1A
P1B
EBI Ext. Con
P2A
I/O Extension connector
P2B
I/O Extension connector
J4 Motion Sensor
J3 Camera
PA10 / IRQ1
R
R
V_{CC}
GND
V_{DD}
MD
T_X
R_X
P2-PA22
P2-PA21
J4-MD
J2-BRDY3
J2-/INT
J2-BRDY0
J2-BRDY1
J2-BRDY2
J3-T_X
J3-R_X

Graphical Hierarchy of source code



Index of Source Code

Directory Structure

3rd Party Code:

SmartHome/lib:

lib_drv_16/	# ATMEL-provided MCU library code
m55800_lib16/	# ATMEL-provided MCU library code
uC/	# microC/OS-II code

SmartHome/src/cmpe490:

wait_irq.arm	# provided wait interrupt handler

SmartHome/src/user/wiznet:

socket.c	# WIZnet socket API code
w5300.c	# WIZNET w5300 chipset code

SmartHome/include/wiznet/:

w5300.h	# WIZnet-provided header for w5300.c. Defines registers and basic IO functions
iinchip_conf.h	# WIZnet-provided header for w5300 compile options and MCU options
socket.h	# WIZnet socket API definitions
types.h	# WIZnet-provided typedefs

Our Code:

SmartHome/src/user:

camera.c	# camera-related code.
ftp.c	# provides FTP-related functions (reading and writing to files over FTP).
httpd.c	# provides HTTP server-related functions.
main.c	# main function.

SmartHome/include/user:

app_cfg.h	# microC related header
camera.h	# camera header
httpd.h	# header for HTTPd code
main.h	# header for main.c
ftp.h	# header for FTP client code (#defines and prototypes)