Wireless Motion and Flood Security System for Boats

ECE4007 Senior Design Project

Section L03/04, SecurityCam

Project Advisor, Dr. Koblasz

Rohit Sahay

Ramsey Baker

Kunal Parbadia

David Salazar

Radford Parker

Submitted

December 16, 2010

Table of Contents

Executive Summary

The SecurityCam is a remote intrusion and disaster detection system that allows its user to monitor their boat and receive alerts upon motion seen through a web camera or a flood detected inside the boat's hull. The purpose of this security system is to provide our customers, boat owners with an iPhones, a convenient way to monitor against any disturbance of their property. The device utilizes a laptop, a webcam, Arduino board for the water sensor, server database, and iPhone. The system allows the user to call local emergency services or ignore the notification altogether. SecurityCam's main goals are to provide current iPhone customers with the convenient ability to remotely monitor their property and the ability to store images of the disaster on a server for continuous retrieval. The SecurityCam group developed a new innovative product that will include a more interactive user interface, limit user setup and minimize system management. This product would be a more cost-effective alternative to buying more expensive systems and services from companies like ADT or Broadview. The hardware and software components include a webcam, a Linux laptop, a prototype C code for running average background subtraction detection, an online Linux server with a preconfigured Perl environment and MySQL database, Apple's Software Development Kit (SDK) for the iPhone, and objective C programs applicable to the application design. The only cost for developing this project was the purchase of the iPhone application development kit. The application retrieved and animated any of the previously stored images and allowed the user to also view the most recent detected event. Future work should include the implementation of the push notification system and the ability to stream live video from a remote location.

1. Introduction

The SecurityCam is a remote intrusion and disaster detection system that allows its user to monitor an area and receive alerts upon flooding or motion seen through a web camera. Our inspiration was to provide both a cost effective alternative to detect any disturbance of an owner's boat and a convenient way for the user to view the motion and respond to the threat using an iPhone. The SecurityCam team spent a total of $100.00 on all components required for the system.

1.1 Objective

The objective of the SecurityCam was to allow remote security monitoring of any boat under surveillance through an iPhone application. The components of the security system include a webcam, water sensor, and computer connected to the internet. The user's computer was connected to an offsite server that communicated with a user's iPhone. This security system sends an email alert and displays intrusion and disaster information when the device detects the presence of motion or water. When the email is pushed to the device, the user is able to enter the application to view a picture animation of the event and call a pre-programmed emergency phone number. The application also shows a table of the history of events on the server for the user to select and view.

1.2 Motivation

SecurityCam's main goals were to provide current iPhone customers that own a boat, the convenient ability to remotely monitor their property and view stored pictures on the server from a particular event. This product is a more cost-effective alternative to buying more expensive systems and services from companies such as ADT or Broadview. Some companies have designed similar products, but have raised their prices because their devices contain extra technology that is unnecessary when solely detecting motion. The SecurityCam team was able to improve the accuracy of the detection algorithm by creating a design that ignored safe motion, such as the motion changing shadows and lights or the natural rocking of a boat. Our product has provided a solution that is economical with different features than other products currently on the market.

2. Project Description and Goals

SecurityCam's original concept was to provide remote security to home owners away from their property. After discovering this part of the market had cheaper alternatives that provided security, the SecurityCam group decided to change its target market from home owners to boat owners whose boats include cabins. The SecurityCam product will provide remote security to the cabin of a boat when the owner is away. The SecurityCam team added a water sensor feature to provide boat flood detection in the engine room of the boat. The SecurityCam will communicate from the customer's boat to the customer's iPhone, alerting the user that an intrusion and/or leak have been detected. The images of the intrusion will be stored at an offsite server maintained by the SecurityCam group.

The SecurityCam product originally offered video evidence and video storage of any intrusions that would be detected. Due to the complexity of writing a program to stream video through the internet and the time constraint, this feature was removed from our product. We also could not find a solution that would effectively utilizes Apple's HTTP Live streaming service by way of the server. The infrastructure for streaming to be accomplished was not compatible with the MySQL database. Instead, a slide show was designed to show images of the intrusion through PHP scripting. The final product can flip through a reel of images taken, depicting the intrusion. This video streaming feature would become available given enough time to develop and understand the concepts of streaming video data over the internet.

SecurityCam also originally featured push notification alerts. The notification system needed to have an APNs server set up with the certificates that needed to be obtain for the application from Apple. The certificates were loaded on a PHP script that checked the MySQL database for a flag to trigger a specific notification. We were not able to execute the script necessary for this application due to the restraints on the server as administrator privileges were not accessible. The request for rights to run a PHP script using Apple's certificates was not given in time for this feature to be implemented. Instead, code was written on the laptop to email the customer whenever a flood or intrusion was detected. The iPhone's built-in push e-mail system will notify the customer of a new e-mail.

SecurityCam's final product detects an intrusion or flood within milliseconds, sends and e-mail to the customer alerting them of flood or intrusion within milliseconds, stores and array of images of each intrusion on the server and also allows the customer to input and emergency number to call in case the intrusion is not expected. Because of the iPhone's configuration, the customer will receive a push notification for new e-mails every 15 minutes, which will include an e-mail of intrusion or flood.

Product Features:

- Ability to detect motion
- May be set anywhere in the boat's cabin where a power socket is available
- Alerts customer of motion detection through iPhone application
- Stores images of intrusion on remote server
- Provides the option for customers to contact emergency services or ignore alert
- Permits customers to review images at a later time

Product Pricing:

- Total price: $300
    - Laptop with Webcam - $500
    - Server -$20/month for labor and bandwidth
    - iPhone Application – Free
    - Arduino Mega- $64.50
    - Marketing Costs

3. Technical Specifications

The SecurityCam will require the customer to own a laptop with built in webcam, wireless card and USB port. The laptop will also require the Ubuntu operating system. The water sensor for flood detection is assembled using a Aruduino mictrocrontroller. To assemble the full product, the customer will need to establish an internet connection with the internet provided by the marina as well as connect the Arduino board to the laptop with a USB connection. The Arduino board requires solid wires that are connected to the digital and ground pins reach the engine room. The laptop will require a power supply of 100 V. As the key building block of the device, the laptop unit will require enough processing power to perform image processing and data transfer. For image processing, the laptop unit will split the video feed into 30 frames per minute. The code running on the laptop will compare two different images within 0.5 seconds. When images differ, the customer will be alerted in roughly 3-5 minutes on the iPhone application. Images will be transferred from the laptop to the offsite server at speeds of three megabits per second with a resolution of 640x480 pixels. In the engine room, if a leak occurs and the water flowing in the boat triggers the water sensor, the laptop will be notified of the boat flooding. Table 1 shows the specifications of each component of the system outlining the performance, size, and price.

The SecurityCam's original concept was to have an all inclusive device that housed a single board computer, external webcam and USB Wi-Fi adapter. After switching to a laptop for more processing power, the three previously mentioned components were not needed due to the features that laptop included.

Table 1. Technical Specifications by Component.

Items Used	Specifications	Size	Price
HP Pavillion dv6000	AMD Turion Processor 2.00GB of RAM 3 USB 2.0 ports WebCam 1.3 MP Wi-Fi	Height: 1.0"“ Width: 14” Depth: 10.1”	$500
Arduino Mega Board[3]	Input Voltage 7-12 Volts 54 Digital I/O Pins 16 Analog Inputs 256k Flash Memory 16Mhs Clock Speed	Height: 0.5” Width: 4.0” Depth: 2.1”	$64.50
Remote Server[6]	Dedicated IP Address:2 Unlimited Bandwidth	-	$3.96/month with a 2 to 3

	Unlimited Disk Space		year contract
iPhone Application	Alert Option Image/Video viewing Emergency Services Call Remote Activation	-	
Items Not Implemented			
TS -7250 SBC[3]	200 MHz Processor 32MB of SRDAM 32 MB Flash Drive 2 USB 2.0 ports 2 COM ports Wi-Fi	Height: 3.8” Width: 4.5” Depth: 6”	$119.00
Logitech C-510 [4] Webcam	Still image resolution: 8.0 MP Video resolution: 1280x720	Height: 6” Width: 8.3” Depth: 3”	$59.99

4. Design Approach and Details

4.1 Design Details

The building blocks of the SecurityCam system engineered are displayed in Figure 1. The blob detection algorithm loaded on the Linux-based computer runs on the feed from the webcam attached to the laptop. Using Wi-Fi, the laptop communicates through a wireless router with the server whenever a flag is triggered from either the motion detection program or the water sensor. The system sends an e-mail to communicate the alert to the iPhone user using the e-mail push system and also loads an array of pictures for the iPhone application to download and display to the user. A PHP script runs on a cron job, an automated script calling application, to complete data transfer from a folder on the server to the MySQL database. Another PHP script fetches the table entry for the iPhone application. Finally, the iPhone displays an animation of the pictures from the event and allows the user to make an emergency phone call and view past events.

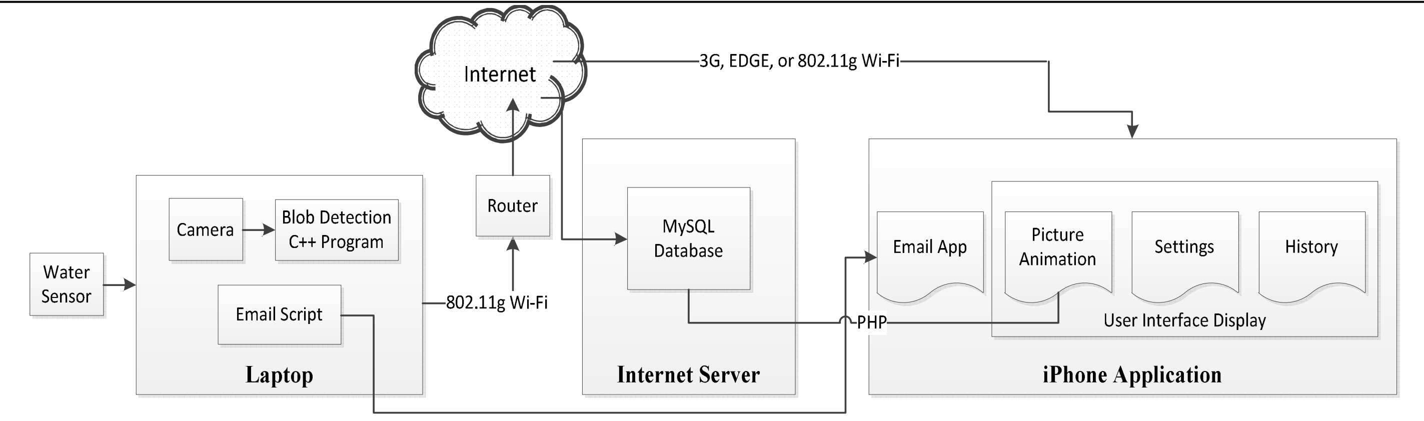

Figure 1. The building blocks of SecurityCam.

The hardware and software components include a webcam, a Linux-based laptop, Ardinuo board for the water sensor, a prototype C code for blob detection, an online Linux server with a MySQL database, PHP scripts to move the images to MySQL database and push to the application, Apple's Software Development Kit (SDK) for the iPhone, and objective C program for the iPhone application.

Detection Algorithm

The motion detection algorithm is designed to utilize the running-average background subtraction method. The algorithm stores a background image as an array and updates the background with every new frame by assigning a lower weight to the new frame and a higher weight to the current background. In order to detect motion, this algorithm subtracts any new frame from the saved running-average background. Any color that is left over after the subtraction is newly detected motion. In order to account for small changes and noise, the algorithm uses the Gaussian smoothing technique. Once all of the motion is detected, a red box is drawn around motion that is large enough to resemble a human figure. A flag is then triggered that tells the algorithm to start saving images ~every 2 seconds.

Water Sensor

The water sensor was constructed with an Arduino board uploaded with a C program to detect a flood. When water is detected, the program sends a flag to the C program running the detection algorithm. The code on the Arduino board is tasked to read a digital pin on the board. One of the wires is connected to the +5V pin and the other end is attached to a prong. Another wire is connected to the GND pin which is in series with a 100kΩ resistor. Finally, the last wire is connected to the digital pin and the prong connected to GND. When the prongs are not in contact with water, the digital pin outputs a 0. Once contact is made with water, which completes the circuit, the digital pin outputs a 1. The schematic of the water sensor is shown in Figure 2 illustrating the Arduino board's pin connections used to construct the sensor.

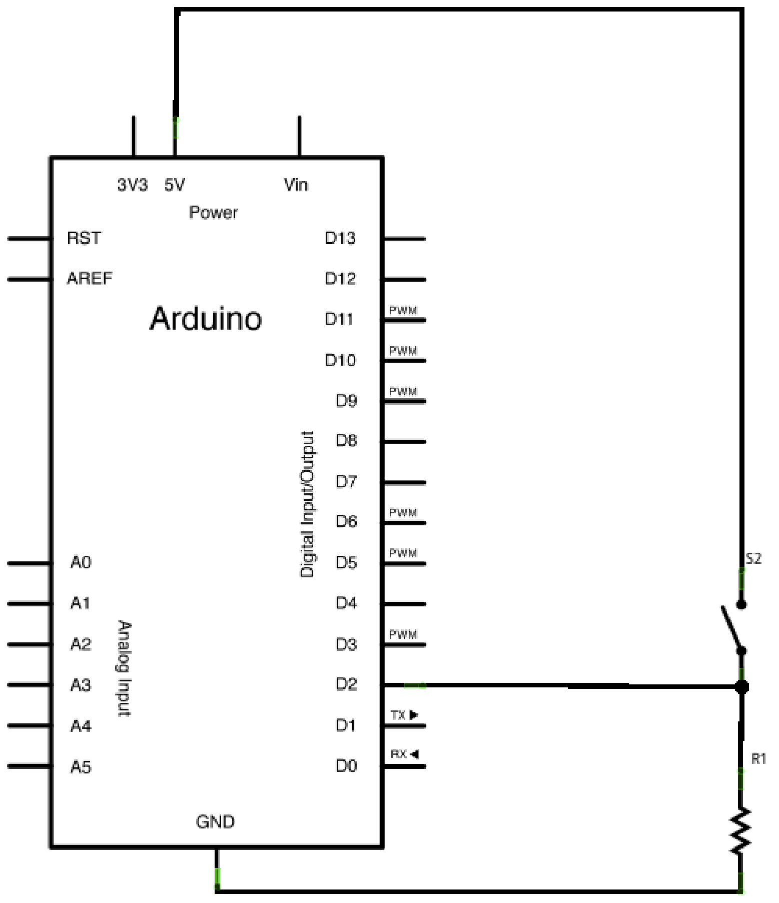

Figure 2. Schematic of the water sensor.

The detection code was written to read the output of the Arduino board from the virtual serial port. A virtual serial port, made from a USB 2.0 port on the computer, is connected to the Arduino board to the system. The detection code checks the USB port, at a 9600 baud rate, and reads in the values of either 0 or 1 from the virtual serial port.

Online Server

The online server is another key part of this design. It enables the system to retrieve pictures stored on the server's hard drive. The array of images was sent through a secure File

Transfer Protocol (FTP) connection to a folder on the server. Specifically, FTP_Auto shell script, provided in Appendix C, automatically transfers images from a folder on the laptop to the server when the blob detection algorithm detects an intrusion. The script is configured to execute on a cron job once per minute and store the images in a specified folder. A PHP script found in Appendix G is executed after this transfer to make a new table entry in the database with information about the event. The database record includes a new event ID, date and time, and the filenames of the images in the array associated with an event.

The link between the server and the iPhone application, shown in Figure 3, is established through another PHP script, found in Appendix G that fetches a database entry. The PHP script performing this action returns the filenames of the images with a specific event ID for the application to download.

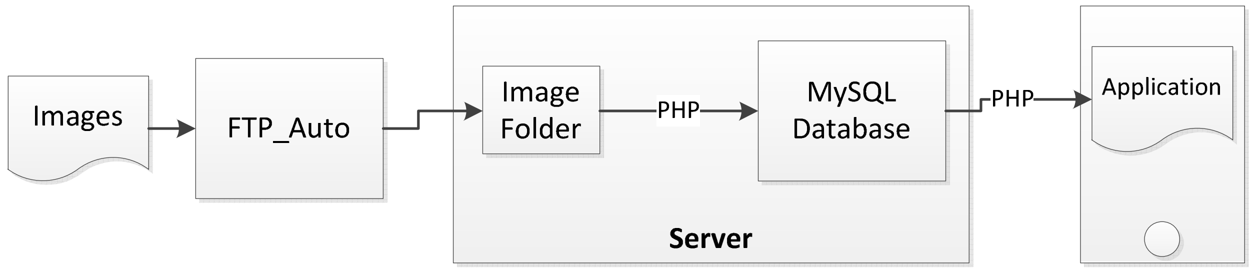

Figure 3. Diagram showing the data transfer of the system.

iPhone Application Development

The end point of the SecurityCam design approach was the development of the mobile application to be installed on an iPhone. This application was a graphical user interface (GUI) that displayed an animation of the video feed from the webcam. The screen shots of the three different views of the application are shown in Figure 4. An Apple Xcode project was set up with classes of the three different views of the application each containing objects. The main view tab includes a UIWebView object that calls the PHP script to fetch the image filenames in the database entry of the most current event. Once the filenames are retrieved, the UIImageView object on the same tab displays images running an animation method. The second view is the settings tab of the application for the user to store an emergency phone number using internal iPhone memory storage variable with the object NSUserDefault. The third view is the history of events tab that displayed a table of past detections through another UIWebView object calling a PHP script, table.php, programmed to select all of the table entries from the MySQL database. This tab includes a textbox for the user to input an event ID to view on the main tab. Finally, the system utilizes the pre-installed e-mail application, which checks a user's e-mail account every 15 minutes, for any alerts from the laptop. This is done by running a shell script to send e-mails to a pre-configured e-mail address.

4.2 Codes and Standards

The Arduino board's Universal Serial Bus 2.0 (USB 2.0) connection was used to connect to the laptop. The laptop was connected to a local area network (WLAN) IEEE 802.11g standard that allowed for a transfer of around 54 megabits per second. The programming on the laptop was accomplished in C. PHP scripts were uploaded to the server to automate store and retrieval process. MySQL database management was conducted through PHP scripting. Finally, the iPhone application was developed using purchased iOS 4 Software Development Kit (SDK). The kit contained Xcode IDE and iOS Simulator for easily development and testing of the objective-C application.

5. Schedule, Tasks, and Milestones

The SecurityCam has been designed and implemented within the past three months. Table 2 shows the major tasks involved in the product development.

Table 2. Schedule of Tasks.

Task	Duration	Start	End
C++ Programming	*40 days*	*9/27/2010*	*11/19/2010*
Server Programming	*30 days*	*10/4/2010*	*11/8/2010*
iPhone Application Development	*35 days*	*10/18/2010*	*12/3/2010*
Water Sensor Development	*15 days*	*11/15/2010*	*12/3/2010*
Demonstration	*5 days*	*12/3/2010*	*12/10/2010*

Programming efforts began solely with the linux machine, but later included server management and iPhone application development. Prior to completion of the software aspect of the design, hardware development of the water sensor commenced. Product integration was completed by 12/10/2010. Once the prescribed milestones were reached, a test demonstration was conducted to ensure the product functioned properly. See Appendix A for the complete Gantt chart.

6. Results and Acceptance Testing

Acceptance testing involved numerous trials that tested each component of the SecurityCam system. In order to test the motion detection, the device was set-up in a room with constant lighting and a human walked across the screen. Each time, motion was detected by the program. The next test involved a variable lighting situation. Variable lighting is an important design consideration to take into account, especially for boats. Throughout the day, the light and shadows in the cabin of the boat change according to the position of the sun with respect to the boat. This means that it is important for the algorithm to constantly update the background to account for slight modifications over time. In order to test this specification, we set the device in a room with an electronically controlled curtain. There were no lights turned on initially, so as the curtain was raised, light slowly filled the room. The device did not flag any motion while this was occurring, so the specification was met. The next important test involved making sure the flood sensor functioned properly. In order to test this part, the sensor was placed in a plastic cup and water was eventually added. The flood sensor triggered within 2 seconds, which proved that it was working correctly.

The SecurityCam design was originally designed to be a single-modular unit that did not exceed 8.3 inches in width, 10 inches in height, and 6 inches in depth. After redesigning from a completely embedded system to a proof of concept with a fully-functioning computer, this size specification was unable to be met. The laptop computer that is being used to demonstrate the design exceeds the previously stated dimensions without even incorporating the flood sensor microcontroller. Even though the design did not meet the original size specifications, it still qualifies as a single-modular unit in the sense that the only physical attachment leaving the device is the 12V power supply for the laptop computer.

The proposed design included certain specifications for the motion detection algorithm that were easily met by the final product. The original SecurityCam design projected that the algorithm would be able to process 30 frames per minute. The final product is capable of handling up to 750 frames per minute. The algorithm was also supposed to output its decision on whether motion was detected within 0.5 seconds. The final product is capable of running the entire algorithm at a fraction of the previously projected time.

The most important specification was that the entire process (from the actual physical motion/flood to when the user is alerted) should take place in 3-5 minutes. After redesigning the product to use a Push e-mail system instead of a Push Notification system, the product loses its ability to have a completely designer-specified processing time. This is because iPhone users are allowed to set the rate at which the phone checks for new e-mails. The fastest rate at which an iPhone can be set to check for new e-mails is every 15 minutes. The design certainly meets the original time constraint of 3-5 minutes 100 percent of the time for the stages from the actual physical motion to the time when the images of the motion are uploaded to the server. If the user's iPhone is set for the fastest Push time, the entire process takes place within 15 minutes 100 percent of the time.

Another specification that was proposed for the SecurityCam, was that it should be able to transfer images to the server at a rate of 3 megabits per second. This original specification has ended up becoming trivial as the total processing time has increased. Data can be transferred at much slower rates and still not affect the total processing time. A more relevant specification that was proposed for the design is that each frame has a resolution of 1280x720 pixels. The combination of using the built-in webcam from the laptop and the opencv image processing environment caused each frame to be set at a default resolution of 640x480 pixels.

7. Budget and Cost Analysis

All of the necessary parts for the device were purchased for $100.00. Most of the hardware was either found in the lab or supplied by individual design team members. The iPhone iOS SDK 4 was the only part of the project that required actual funds. Table 3 lists the price of each required component.

Table 3. Component Pricing Breakdown.

Product Description	Quantity	Unit Price	Supplied by	Actual Price
HP Pavilion dv6000 Laptop Computer (with built-in Webcam, WiFi interface)	1	$800.00	Radford	$0.00
Arudino Mega Microcontroller	1	$50.00	Lab	$0.00
Wiring	1	$0.50	Lab	$0.00
Server Monthly Subscription	4	$20.00	Rohit	$0.00
iPhone 4	1	$200.00	Ramsey	$0.00
iPhone iOS SDK 4	1	$100.00	N/A	$100.00
Product Cost Total				**$100.00**

8. Conclusions and Future Work

The SecurityCam is fully operational as a proof-of-concept. The original design was meant to use a completely embedded system that did not require the use of a fully-operational computer. The intent for implementing an embedded system were stemmed in the beliefs the consumer should not have to supply their own computer and the entire device should fit in a small enclosed housing that could be secured to the corner of a ceiling. During future work, these goals should be accomplished to make the product more commercially viable.

There are a few minor improvements that should be made to the design that allow for more functionality and better commercial viability as a product. One example would be to implement HTTP live streaming of the video feed from the webcam. This would give the user the ability to log-in to a live stream of their boat from any internet-connected computer anywhere. Future work should certainly entail adding Push Notifications using the APNS server instead of Push e-mail. This way, the designer can set how quickly the user is updated once the flags have been triggered on the server side. This will allow for an increase in response time and universal control of the timing by the designer. Later designs should also include enhancing the design and layout of the iPhone application. These changes should make the user experience more intuitive and more aesthetically pleasing.

If all of these features can be implemented in a future prototype, the design would then contain enough functionality to go into production. Table 4 summarizes these points for future work.

Table 4. Points of Future Work.

Point	**Reason**
Fully Embedded System	Make it so that the user does not need to dedicate their own computer to use for permanent monitoring
Single Enclosure	Minimize size of entire device to preserve limited space in boat cabin
Live Streaming	Give user ability to constantly view boat from any computer
Push Notifications	Decrease overall processing time of device by allowing user to be notified immediately when server is flagged
iPhone Application Design/Layout Upgrades	Create a more intuitive layout that is more aesthetically pleasing
Additional Sensors	Provide a more holistic method of boat security by incorporating other sensors

9. References

[1] iCam, “Sales Literature for Features to iCam,” *SKJM*, 2010. [Online]. Available: http://skjm.com/icam/support.php. [Accessed Sept. 19, 2010].

[2] DropCam, “Sales Literature for Features to the DropCam,” *DropCam*, 2010. [Online]. Available: http://www.dropcam.com/dropcam-ip-security-camera. [Accessed Sept. 19, 2010].

[3] TS-7259, “Board Detail,” Embedded Arm, 2010. [Online]. Available: http://www.embeddedarm.com/products/board-detail.php?product=TS-7250. [Accessed Sept. 18, 2010].

[4] Logitech HD-Webcam, “Product Info,” BestBuy, 2010. [Online]. Available: http://www.bestbuy.com/site/Logitech+-+C510+Webcam+-+Black/9928381.p?id=1218197327076&skuId=9928381. [Accessed Sept. 18, 2010].

[5] Belkin -Basic N150, “Product Info,” BestBuy, 2010. [Online]. Available: http://www.bestbuy.com/site/Belkin++Basic+N150+Wireless+USB+Adapter/9769185.p?id=1218170030189&skuId=9769185. [Accessed Sept. 18, 2010].

[6] Expertplan, “Hosting Plans,” Ixwebhosting, 2010. [Online]. Available: http://www.ixwebhosting.com/index.php/-v2/pages.dspmain?sc_cid=aff-ix-top10web [Accessed Sept. 18, 2010].

APPENDIX

Appendix A: Gannt Chart

Task	**Duration**	**Start**	**End**	**Names**
Proposal	**5 days**	**9/16/2010**	**9/22/2010**	**Group**
Turn in Proposal	*0 days*	*9/22/2010*	*9/22/2010*	*Group*
Finalize inventory	**5 days**	**9/20/2010**	**9/24/2010**	**Group**
Choose operating system	*0 days*	*9/21/2010*	*9/21/2010*	*Parker*
Choose camera	*0 days*	*9/22/2010*	*9/22/2010*	*Baker*
Establish server	*0 days*	*9/23/2010*	*9/23/2010*	*Sahay*
Linux Programming	**40 days**	**9/27/2010**	**11/19/2010**	**Group**
Background subtraction algorithm complete	*0 days*	*11/1/2010*	*11/1/2010*	*Parker*
Successful blob detection	*0 days*	*11/15/2010*	*11/15/2010*	*Parker*
Communication between device and server	*0 days*	*11/22/2010*	*10/22/2010*	*Salazar*
Server Programming	**20 days**	**10/4/2010**	**10/29/2010**	**Group**
Database accessibility	*0 days*	*10/13/2010*	*10/22/2010*	*Sahay*
Successful storage/retrieval of videos	*0 days*	*10/22/2010*	*10/29/2010*	*Sahay*
iPhone Application Development	**35 days**	**10/18/2010**	**12/3/2010**	**Group**
Review tutorials	*0 days*	*10/29/2010*	*10/29/2010*	*Parbadia*
Create user interface	*0 days*	*11/22/2010*	*11/22/2010*	*Parbadia*
Communication between server and iPhone	*0 days*	*12/3/2010*	*12/3/2010*	*Sahay*
Water Sensor Development	**15 days**	**11/15/2010**	**12/3/2010**	**Group**
Hardware design	*0 days*	*11/22/2010*	*11/22/2010*	*Baker*
Communication with flag system	*0 days*	*12/3/2010*	*12/3/2010*	*Salazar*
Demonstration	**5 days**	**12/3/2010**	**12/10/2010**	**Group**
Demonstration	*0 days*	*12/10/2010*	*12/10/2010*	*Group*
Final Presentation	**0 days**	**12/16/2010**	**12/16/2010**	**Group**

Appendix B: Motion and Flood Detection Algorithm

```
//run using: g++ -ggdb -I/usr/include/opencv -lhighgui webcam.c -o webcam
//then: ./webcam
#include "highgui.h"
#include "cv.h"
#include "iostream"
#include "stdlib.h"
#include "stdio.h"
#include "string.h"  /* String function definitions */
#include "unistd.h"  /* UNIX standard function definitions */
#include "fcntl.h"   /* File control definitions */
#include "errno.h"   /* Error number definitions */
#include "termios.h" /* POSIX terminal control definitions */
using namespace std;
int main( int argc, char** argv ) {
        cvNamedWindow( "NewFrame",0);
        cvResizeWindow("NewFrame",400,400);
        cvNamedWindow( "MotionDetection",0);
        cvResizeWindow("MotionDetection",400,400);
        cvNamedWindow( "Background",0);
        cvResizeWindow("Background",400,400);
        cvMoveWindow("Background",700,0);
        cvMoveWindow("MotionDetection",350,400);
        CvCapture* capture = cvCreateCameraCapture(0) ;
        CvRect bndRect = cvRect(0,0,0,0);
        CvPoint pt1, pt2;
        CvFont font;
        IplImage*newframe,*background,*grayImage,*temp,*smallnewframe;
        int first=0,prevX = 0,numPeople = 0,avgX = 0,closestToLeft = 0,closestToRight = 320;
        CvMat* inFrame,*backFrame;
        int fd,get=5,c,count=0,motioncount=0,motionflag,oldmotion=0,floodcount=0;
        char buf[5];
        char file[256];
        FILE *in = fopen("InputFiles/BoatFlood.jpg","r");
        FILE *out = fopen("BoatFlood.jpg","w");
        fd = open("/dev/ttyUSB0", O_RDWR);
        system("stty -F /dev/ttyUSB0 9600 cs8 -cstopb -parity -icanon min 1 time 1");
        newframe = cvQueryFrame(capture);
        smallnewframe = cvCreateImage(cvSize(100,75),newframe->depth,newframe->nChannels);
        temp = cvCreateImage(cvGetSize(newframe), IPL_DEPTH_8U, 3);
        background = cvCreateImage(cvGetSize(newframe), IPL_DEPTH_8U, 3);
        grayImage = cvCreateImage(cvGetSize(newframe), IPL_DEPTH_8U, 1);
        cvConvertScale(newframe, background, 1.0, 0.0);
        while(1) {
                motionflag=0;
        if(first=0){
                first=1;
                newframe = cvQueryFrame(capture);
                cvSmooth(newframe,newframe,CV_GAUSSIAN,7,7);
                cvConvertScale(newframe, background, 1.0, 0.0);
                cvCopy(newframe,background);
                sleep(5);
        }else{
                newframe = cvQueryFrame(capture);
        }
        if(!newframe) break;
        cvSmooth(newframe,newframe,CV_GAUSSIAN,7,7);
        cvShowImage("NewFrame",newframe);
        inFrame = cvCreateMat(newframe->height, newframe->width, CV_8UC3);
        cvCopy(newframe, inFrame, 0);
        backFrame = cvCreateMat(newframe->height, newframe->width, CV_8UC3);
        cvCopy(background, backFrame, 0);
        cvConvertScale(inFrame,inFrame,0.1,0);
        cvConvertScale(backFrame,backFrame,0.9,0);
        cvAdd(inFrame,backFrame,backFrame);
        cvCopy(backFrame,background,0);
        cvReleaseMat(&inFrame);
        cvReleaseMat(&backFrame);
        cvAbsDiff(newframe,background,temp);
        cvThreshold(temp,temp,50,255,CV_THRESH_BINARY);
        cvCvtColor(temp,grayImage,CV_RGB2GRAY);
    cvErode(grayImage,grayImage);
        cvDilate(grayImage,grayImage);
        //Find the contours of the moving images in the frame.
        CvMemStorage* storage = cvCreateMemStorage(0);
        CvSeq* contour = 0;
        cvFindContours( grayImage, storage, &contour, sizeof(CvContour), CV_RETR_CCOMP,
CV_CHAIN_APPROX_SIMPLE );
        //Process each moving contour in the current frame...
        for( ; contour != 0; contour = contour->h_next )
        {

                //Get a bounding rectangle around the moving object.
                bndRect = cvBoundingRect(contour, 0);
                if(bndRect.width<25||bndRect.height<25){
                        continue;
                }
                else{
                        motionflag=1;
                }
                pt1.x = bndRect.x;
```

```
		pt1.y = bndRect.y;
		pt2.x = bndRect.x + bndRect.width;
		pt2.y = bndRect.y + bndRect.height;
		//Get an average X position of the moving contour.
		avgX = (pt1.x + pt2.x) / 2;
		//If the the previous contour was within 2 of the left boundary...
		if(closestToLeft >= 88 && closestToLeft <= 90)
		{
			//If the current X position is greater than the previous...
			if(avgX > prevX)
			{
				//Increase the number of people.
				numPeople++;
				//Reset the closest object to the left indicator.
				closestToLeft = 0;
			}
		}
		//else if the previous contour was within 2 of the right boundary...
		else if(closestToRight >= 250 && closestToRight <= 252)
		{
			//If the current X position is less than the previous...
			if(avgX < prevX)
			{
			//Increase the number of people.
			numPeople++;
			//Reset the closest object to the right counter.
			closestToRight = 320;
			}
		}
		//Draw the bounding rectangle around the moving object.
		cvRectangle(newframe, pt1, pt2, CV_RGB(255,0,0), 1);
		//If the current object is closer to the left boundary but still not across
		//it, then change the closest to the left counter to this value.
		if(avgX > closestToLeft && avgX <= 90)
		{
			closestToLeft = avgX;
		}
		//If the current object is closer to the right boundary but still not across
		//it, then change the closest to the right counter to this value.
		if(avgX < closestToRight && avgX >= 250)
		{
			closestToRight = avgX;
		}
		//Save the current X value to use as the previous in the next iteration.
		prevX = avgX;
	}
```

```
if((motioncount==0)&&(motionflag==1)&&(count>50)){
        system("/home/radford/Desktop/SeniorDesignProject/send2.sh");
        printf("motion detection email has been sent\n");
}
if((motionflag==1)||(oldmotion==1)){
        motioncount++;
        if((motioncount%25==0)&&(motioncount<1250)){//one frame every 2 seconds, only save 50 frames total
                sprintf(file,"frame-%04d.jpg",motioncount/25);
                printf("frame-%04d.jpg written\n",motioncount/25);
                cvResize(newframe,smallnewframe);
                cvSaveImage(file,smallnewframe);
        }
}else{
        motioncount=0;
}
oldmotion=motionflag;
cvShowImage("Background",background);
cvShowImage("MotionDetection",newframe);
count++;
//Begin Flood Sensor Code//
if(count%100==0){
        if (fd == -1){
                //Could not open the port.
                perror("open_port: Unable to open /dev/ttyS0 - ");
        }else{
                struct termios options;
                //Get the current options for the port...
                tcgetattr(fd, &options);
                //Set the baud rates to 19200...
                cfsetispeed(&options, B9600);
                cfsetospeed(&options, B9600);
                //Enable the receiver and set local mode...
                options.c_cflag |= (CLOCAL | CREAD);
                //Set the new options for the port...
                tcsetattr(fd, TCSANOW, &options);
                get = read(fd, buf, 1);
                if(buf[0]-48==1){

                        while((c=getc(in))!=EOF){putc(c,out);}
                        printf("flood detected\n");
                        if(floodcount==0){
                                system("/home/radford/Desktop/SeniorDesignProject/send.sh");
                                printf("flood email has been sent\n");
                                floodcount=1;
                        }
                }
```

```
                    }
                }
                char c = cvWaitKey(33);
                if( c == 27 ) break;
        }
        fclose(in);
        fclose(out);
        cvReleaseImage(&temp);
        cvReleaseImage(&newframe);
        cvReleaseImage(&grayImage);
        cvReleaseImage(&background);
        cvReleaseCapture( &capture );
        cvDestroyWindow( "NewFrame" );
        cvDestroyWindow( "Background" );
        cvDestroyWindow( "MotionDetection" );
}
```

Appendix C: FTP Auto.sh

```
#=============================

#!/bin/sh

HOST=ftp.tropicvision.com

USERNAME=cam@tropicvision.com

PASSWORD=security1

cd Desktop/SeniorDesignProject/


ftp -pin $HOST <<EOD

quote USER $USERNAME

quote PASS $PASSWORD



cd SecurityCam

binary

mput *

quit

EOD


exit 0

#================================
```

Appendix D: Flood and Motion Email

send.sh

```
#!/bin/bash
# script to send simple email
mail -s 'Flood' dsalazar3@gmail.com < /home/radford/Desktop/SeniorDesignProject/message.txt
```

send2.sh

```
#!/bin/bash
# script to send simple email
mail -s 'Intrusion' dsalazar3@gmail.com < /home/radford/Desktop/SeniorDesignProject/message2.txt
```

Appendix E: Arduino Flood Sensor

```
void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT);
  pinMode(13, OUTPUT);
}
void loop() {
  int sensorValue = digitalRead(2);
  if( sensorValue == 1){
   digitalWrite(13, HIGH);   // set the LED on
   delay(100);              // wait for a 1/2 second
   digitalWrite(13, LOW);   // set the LED on
  }
  delay(100);              // wait for a 1/2 second
  Serial.println(sensorValue, DEC);
}
```

Appendix F: Xcode

```
//
//  FirstViewController.h
//  SecurityCam
//
//  Created by Rohit Sahay on 11/29/10.
//  Copyright 2010 Georgia Tech. All rights reserved.
//

#import <UIKit/UIKit.h>


@interface FirstViewController : UIViewController {
	IBOutlet UIWebView *BackgroundLoader;

	IBOutlet UILabel *label;
	NSString *str;
	IBOutlet UITextField *contactfield;
	IBOutlet UILabel *contactnum;
	NSString *html;
	IBOutlet UIImageView *imageView;
	IBOutlet UIImageView *lastimage;
	IBOutlet UIButton *update;
	//NSString *score;
	NSArray *splitSourceData;
	int flag;


}
@property (nonatomic, retain) NSString *str;
@property (nonatomic,retain) NSString *html;
@property (nonatomic,retain) NSArray *splitSourceData;
@property (nonatomic,retain) UIWebView *BackgroundLoader;




-(IBAction)sqlbutton:(id)sender;
-(IBAction)savenum:(id)sender;
-(IBAction)dial:(id)sender;

@end
```

```
//
//  FirstViewController.m
//  SecurityCam
//
//  Copyright 2010 Georgia Tech. All rights reserved.
//

#import "FirstViewController.h"
//#import "dataloader.h"
```

```
@implementation FirstViewController
//@class dataloader;
@synthesize str, html, splitSourceData, BackgroundLoader;

-(IBAction)sqlbutton:(id)sender {

        //[NSTimer scheduledTimerWithTimeInterval:.1 target:self selector:@selector (viewDidLoad) userInfo:nil repeats:NO];



        ///start

        //NSString *score = [BackgroundLoader stringByEvaluatingJavaScriptFromString:@"document.documentElement.innerText"];
//      NSLog(@"score: %@", score);

//      self.splitSourceData = [score componentsSeparatedByString:@","];
        if (!flag) {
        NSString *score = [BackgroundLoader stringByEvaluatingJavaScriptFromString:@"document.documentElement.innerText"];
        NSLog(@"Score: %@", score);
          NSUserDefaults *thescore = [NSUserDefaults standardUserDefaults];
          [thescore synchronize];
          [thescore setObject:score forKey:@"returndata"];
                  self.splitSourceData = [score componentsSeparatedByString:@","];

          flag = 1;
        }
        else {
          NSUserDefaults *thescore = [NSUserDefaults standardUserDefaults];
          [thescore synchronize];
          NSString *score = [thescore stringForKey:@"returndata"];
          NSLog(@"Score: %@", score);
                  self.splitSourceData = [score componentsSeparatedByString:@","];

        }


        NSLog(@"splitsourcedata: %d", self.splitSourceData.count);


          /////////START ANIMATION////////
        //printf("button is pressed");

        int y = self.splitSourceData.count;
        NSLog(@"y: %d \n", y);
        //imageView.animationImages = [NSMutableArray arrayWithCapacity:5];
        int i;
        //printf("button is pressed");
        //NSString *docDir = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES) objectAtIndex:0];
        NSMutableArray *myImages = [[NSMutableArray alloc]init];
        for (i = 0; i < y - 1; i++) {
          printf("button is pressed");
          NSString *myArrayElement = [self.splitSourceData objectAtIndex:i];
          NSString *path = [NSString stringWithFormat:@"http://www.tropicvision.com/%@", myArrayElement];
          NSLog(@"CHECK: %@ \n",path);
          NSURL *url = [NSURL URLWithString:path];
          NSData *data = [NSData dataWithContentsOfURL:url];
          UIImage *img = [[UIImage alloc] initWithData:data];
          [myImages addObject:img];

          UIImageView *myAnimatedView = [UIImageView alloc];
          [myAnimatedView initWithFrame:CGRectMake(40, 70, 230, 200)];
          myAnimatedView.animationImages = myImages;
          myAnimatedView.animationDuration = 12; // seconds
          myAnimatedView.animationRepeatCount = 1; // 0 = loops forever

          [myAnimatedView startAnimating];
          [self.view addSubview:myAnimatedView];
          [myAnimatedView release];

          //              [img release];
          [lastimage setImage: img];

        }

        //////END ANIMATION

        ///end


}
-(IBAction)savenum:(id)sender{

        [contactnum setText:([contactfield text])];

}
-(IBAction)dial:(id)sender{
        NSUserDefaults *num = [NSUserDefaults standardUserDefaults];
        [num synchronize];
        NSString *getnum = [num stringForKey:@"phonenumber"];
        if (getnum == nil) {
          printf("this is nill");
        }
        else {
          NSLog(@"THIS IS straight %@", getnum);
          [[UIApplication sharedApplication] openURL:[NSURL URLWithString:[NSString stringWithFormat:@"tel://%@",getnum]]];
        }
}

/*
// The designated initializer. Override to perform setup that is required before the view is loaded.
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil {
        if ((self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil])) {
          // Custom initialization
        }
        return self;
}
*/

/*
// Implement loadView to create a view hierarchy programmatically, without using a nib.
- (void)loadView {
}
*/


// Implement viewDidLoad to do additional setup after loading the view, typically from a nib.
```

```
- (void)viewDidLoad {
   [super viewDidLoad];
          NSUserDefaults *theid = [NSUserDefaults standardUserDefaults];
          [theid synchronize];
          NSString *getid = [theid stringForKey:@"eventid"];
          //NSLog(str);
          if (!getid) {
             getid = @"";
          }

          flag = 0;
          NSString *Website = [NSString stringWithFormat:@"http://www.tropicvision.com/test.php?id=%@",getid];
          [BackgroundLoader loadRequest:[NSURLRequest requestWithURL:[NSURL URLWithString:Website]]];

}


/*
// Override to allow orientations other than the default portrait orientation.
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {
   // Return YES for supported orientations
   return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
*/


- (void)didReceiveMemoryWarning {
          // Releases the view if it doesn't have a superview.
   [super didReceiveMemoryWarning];

          // Release any cached data, images, etc that aren't in use.
}

- (void)viewDidUnload {
          // Release any retained subviews of the main view.
          // e.g. self.myOutlet = nil;
}


- (void)dealloc {
   [super dealloc];
}

@end
```

```
//
//  SecondViewController.h
//  SecurityCam
//
//  Created by Rohit Sahay on 12/7/10.
//  Copyright 2010 Georgia Tech. All rights reserved.
//

#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>
```

```
@interface SecondViewController : UIViewController {
	IBOutlet UITextField *contactfield;
	IBOutlet UILabel *contactnum;
}

-(IBAction)savenum:(id)sender;
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;

@property(nonatomic,retain) IBOutlet UITextField *contactfield;
@property(nonatomic,retain) IBOutlet UILabel *contactnum;
@end
```

```
//
//  SecondViewController.m
//  SecurityCam
//
//  Created by Rohit Sahay on 12/7/10.
//  Copyright 2010 Georgia Tech. All rights reserved.
//

#import "SecondViewController.h"


@implementation SecondViewController
@synthesize contactfield,contactnum;

-(IBAction)savenum:(id)sender{


	NSUserDefaults *num = [NSUserDefaults standardUserDefaults];
	[num synchronize];
	[num setObject:contactfield.text forKey:@"phonenumber"];
	[contactfield resignFirstResponder];
	contactnum.text = [num stringForKey:@"phonenumber"];

}
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event {
	[contactfield resignFirstResponder];
}


- (void)viewDidLoad {
    [super viewDidLoad];
	NSUserDefaults *num = [NSUserDefaults standardUserDefaults];
	[num synchronize];
	NSString *getnum = [num stringForKey:@"phonenumber"];
	if(getnum == nil) {
	contactnum.text = @"Enter Number";
	}
	else {
	    contactnum.text = [[NSString alloc] initWithFormat:@"%@", getnum];
	}

}
- (void)didReceiveMemoryWarning {
		// Releases the view if it doesn't have a superview.
	[super didReceiveMemoryWarning];

		// Release any cached data, images, etc that aren't in use.
}

- (void)viewDidUnload {
		// Release any retained subviews of the main view.
		// e.g. self.myOutlet = nil;
}


- (void)dealloc {
	[super dealloc];
}
@end
```

```
//
//  ThirdViewController.h
//  mysqltest
//
//  Created by Rohit Sahay on 12/12/10.
//  Copyright 2010 Georgia Tech. All rights reserved.
//

#import <UIKit/UIKit.h>


@interface ThirdViewController : UIViewController {

		IBOutlet UIWebView *table;
		IBOutlet UITextField *eventid;
		IBOutlet UILabel *currentid;
		IBOutlet UIWebView *ReturnWebsite;
		IBOutlet UIButton *update;

}

-(IBAction)saveid:(id)sender;
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
-(void)doit;
@end
```

```
//
//  ThirdViewController.m
//  mysqltest
//
//  Created by Rohit Sahay on 12/12/10.
//  Copyright 2010 Georgia Tech. All rights reserved.
//

#import "ThirdViewController.h"
#import "FirstViewController.h"

@implementation ThirdViewController
```

```
-(IBAction)saveid:(id)sender{

            NSUserDefaults *theid = [NSUserDefaults standardUserDefaults];
            [theid synchronize];
            [theid setObject:eventid.text forKey:@"eventid"];
            [eventid resignFirstResponder];
            currentid.text = [theid stringForKey:@"eventid"];
            NSString *Website = [NSString stringWithFormat:@"http://www.tropicvision.com/test.php?id=%@",[currentid text]];
            [NSTimer scheduledTimerWithTimeInterval:.1 target:self selector:@selector (doit) userInfo:nil repeats:NO];
            [ReturnWebsite loadRequest:[NSURLRequest requestWithURL:[NSURL URLWithString:Website]]];
            NSString *btntitle = [(UIButton *)sender currentTitle];
            NSLog(@"UPDATE: %@", btntitle);
            if (btntitle == @"Save") {
               [update setTitle:@"Select" forState:UIControlStateNormal];
            }
            else {
            [update setTitle:@"Save" forState:UIControlStateNormal];
            }


}
-(void) doit {

            NSString *score = [ReturnWebsite stringByEvaluatingJavaScriptFromString:@"document.documentElement.innerText"];
            NSLog(@"Score: %@", score);
            NSUserDefaults *thescore = [NSUserDefaults standardUserDefaults];
            [thescore synchronize];
            [thescore setObject:score forKey:@"returndata"];



}
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event {
            [eventid resignFirstResponder];
}

- (void)viewDidLoad {
            [super viewDidLoad];
  //        [NSTimer scheduledTimerWithTimeInterval:.1 target:self userInfo:nil repeats:NO];
            NSString *Website = [NSString stringWithFormat:@"http://www.tropicvision.com/table.php"];
            [table loadRequest:[NSURLRequest requestWithURL:[NSURL URLWithString:Website]]];
            NSUserDefaults *theid = [NSUserDefaults standardUserDefaults];
            [theid synchronize];
            NSString *getid = [theid stringForKey:@"eventid"];
            if(getid == nil) {
            currentid.text = @"Latest Event";
            }
            else {
               currentid.text = [[NSString alloc] initWithFormat:@"%@", getid];

            }

 }
- (void)dealloc {
   [super dealloc];
}
@end
```

Appendix G: PHP Scripts

```
------------ table.php --------------(Display table of events)

<?php


mysql_connect("localhost", "tropicv1_sec", "securitycam") or die(mysql_error());

mysql_select_db("tropicv1_cam") or die(mysql_error());

$query = mysql_query("select * from example");

$flag = 0;

echo "<Table cellpadding=2 cellspacing=2 border=1><TR><TD><B>EVENT ID</B></TD><TD><B>DATE</B></TD></TR>";

while($row = mysql_fetch_array($query)){

 if ($flag != $row['id']) {

echo "<TR><TD>". $row['id']. "</TD><TD>".$row['date']."</TD></TR>";

$flag = $row['id'];

}

}

echo "</TABLE>";

?>


-----------------test.php---------------- (Display text of Image Paths)


<?php


mysql_connect("localhost", "tropicv1_sec", "securitycam") or die(mysql_error());

mysql_select_db("tropicv1_cam") or die(mysql_error());

if ($_GET["id"]) {

$statement = "select data from example where id='".$_GET["id"]."'";

$query = mysql_query($statement);
while($row = mysql_fetch_array($query)){

                echo "cam/Saved/".$_GET["id"]."/".$row[0]. ",";

}

}

else

{

$statement = "select id from example order by id desc limit 1";

$query = mysql_query($statement);

while($row = mysql_fetch_array($query)){

                $id = $row[0];

}

 $query2 = mysql_query("select data from example where id='".$id."'");

while($row = mysql_fetch_array($query2)){

                echo "cam/Saved/".$id."/".$row[0]. ",";

}



}

?>


-------------------process.php---------------------(Crate event id, populate database with images)

<?php

mysql_connect("localhost", "tropicv1_sec", "securitycam") or die(mysql_error());

mysql_select_db("tropicv1_cam") or die(mysql_error());

$query = mysql_query("select id from example order by id desc limit 1");

while($row = mysql_fetch_array($query)){

$id=$row[0];

}


$id = $id + 1;
```

```
$findme = ".jpg";

$findme2 = "rame";

$images = array();

$i=0;

$dirflag = 0;

if ($handle = opendir('/home/tropicv1/public_html/cam/SecurityCam/')) {

    echo "Directory handle: $handle <BR>";

    echo "<B>Files:</B> <BR>";

    /* This is the correct way to loop over the directory. */

    while (false !== ($file = readdir($handle))) {

   $pos = strpos($file, $findme);

   $pos2 = strpos($file, $findme2);

    if ($pos2 && $pos){

     if (!$dirflag) {

     mkdir("/home/tropicv1/public_html/cam/Saved/".$id."/", 0755);

     $dirflag = 1;

     }

    $images[$i] = $file;

    $i++;

   // echo "$file <BR>";

        }

    } }

    sort($images)

    foreach ($images as $value){

    echo "moved $value <BR>";

   rename("/home/tropicv1/public_html/cam/SecurityCam/".$value, "/home/tropicv1/public_html/cam/Saved/".$id."/".$value);

   $query = "insert into example (id, data, date) values ($id, '$value',NOW())";

   mysql_query($query);

    }


?>
```