Design and Implementation of Remote Surveillance System using Microcontroller

Submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

In

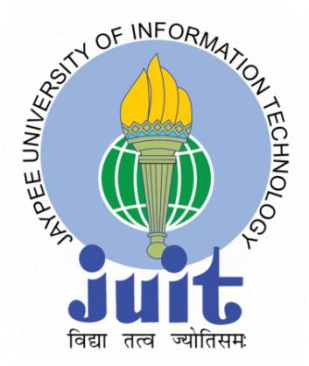
Computer Science Engineering

Under the Supervision of

Ms. Reema Aswani

By

Harshit Tiwari (101289)



May 2014

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

Certificate

This is to certify that the work titled **— "Design and Implementation of Remote Surveillance System using Microcontroller**", submitted by **Harshit Tiwari** partial fulfilment for the award of degree of Bachelor of Technology in Computer Science Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor

Name of Supervisor Ms. Reema Aswani

Designation Assistant Professor

Date 23/05/2014

Acknowledgement

As I conclude our project with the God's grace, I have many people to thank; for all the help, guidance and support they lent me, throughout the course of my endeavor.

First and foremost, I am sincerely thankful to Ms. Reema Aswani, my Project Guide, who has always encouraged me to put in my best efforts and deliver a quality and professional output. Her methodology of making the system strong from inside has taught me that output is not the END of project. I really thank her for her time & efforts.

I am deeply indebted to all those who provided reviews and suggestions for improving the materials and topics covered in my project, and I extend my apologies to anyone I may have failed to mention.

Enrollment number: 101289

Name : Harshit Tiwari

Signature

Table of Content

Chapter - 2

Design & Implementation

Chapter - 3

Future Directions

List of Figures

ABSTRACT

The video surveillance system based on computer technology, embedded technology, and network technology can carry on all-day automatic real-time monitoring to the important department or place of the various professions, and it has already become the important component of peaceful guard domain.

With the unceasing enhancement of life's quality, the family safety protection has been paid an increasing attention, this has led to an increase in demand for reliable and cheap surveillance systems.

Video surveillance with connection to E-mail account and microcontroller chip will make it possible to detect intrusion remotely and help with taking timely action.
An image of any object coming inside the specified range of the Ultrasonic Sensor will be taken and immediately sent to the authorized E-mail Id.

I hope that this project will be a great help to elderly, children and their guardians who want to ensure the safety of their family and loved ones.
Also this project might act as a stepping stone for a more high end surveillance system with more complex features and functionality.

CHAPTER 1

LITERATURE SURVEY

1.1 INTRODUCTION TO ARDUINO UNO MICROCONTROLLER

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

Arduino is a single-board microcontroller, intended to make the application of interactive objects or environments more accessible. The hardware consists of an open-source hardware board designed around an 8-bit Atmel AVR microcontroller, or a 32-bit AtmelARM. Current models feature a USB interface, 6 analog input pins, as well as 14 digital I/O pins which allows the user to attach various extension boards.

Introduced in 2005, it was designed to give students an inexpensive and easy way to program interactive objects. It comes with a simple integrated development environment (IDE) that runs on regular personal computers and allows to write programs for Arduino using C or C++.

1.2 BENEFITS OF ARDUINO

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- **Cross-platform** - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- **Simple, clear programming environment** - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino
- **Open source and extensible software**- The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- **Open source and extensible hardware** - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

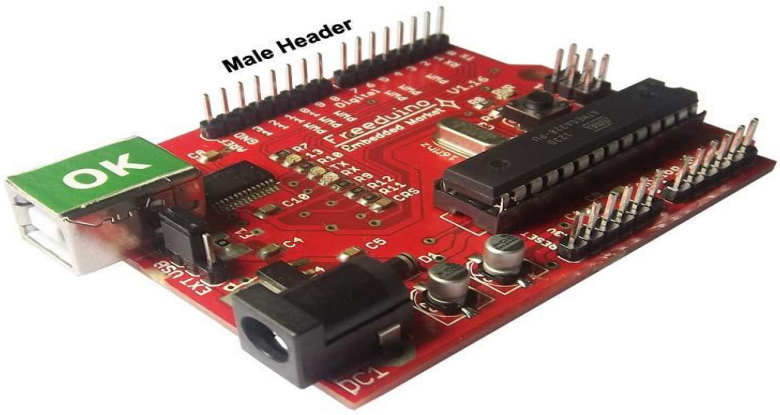


Figure 1.1 Arduino Uno

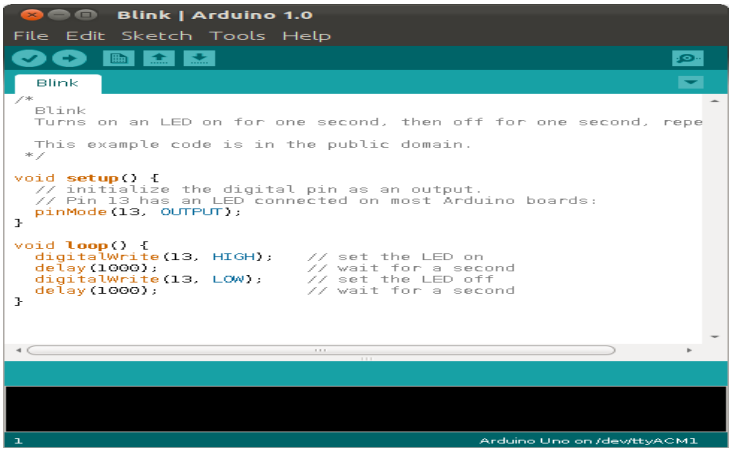


Figure 1.2 A screenshot of the Arduino IDE showing the "Blink" program, a simple beginner program

1.3 SOFTWARE USED IN PROGRAMMING

The **Arduino integrated development environment (IDE)** is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a "sketch".

Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnable cyclic executive program:

- setup(): a function run once at the start of a program that can initialize settings
- loop(): a function called repeatedly until the board powers off

A typical first program for a microcontroller simply blinks an LED on and off. In the Arduino environment, the user might write a program like this:

```
#define LED_PIN 13

void setup () {
  pinMode (LED_PIN, OUTPUT); // Enable pin 13 for digital output
}

void loop () {
  digitalWrite (LED_PIN, HIGH); // Turn on the LED
  delay (1000); // Wait one second (1000 milliseconds)
  digitalWrite (LED_PIN, LOW); // Turn off the LED
  delay (1000); // Wait one second
}
```

It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground; a convenient feature for many simple tests. The previous code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple main() function at the bottom, to make it a valid C++ program.

The Arduino IDE uses the GNU toolchain and AVR Libc to compile programs, and uses avrdude to upload programs to the board.

As the Arduino platform uses Atmel microcontrollers, Atmel's development environment, AVR Studio or the newer Atmel Studio, may also be used to develop software for the Arduino.

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

1.4 FACE RECOGNITION BASICS

A facial recognition system is a computer application for automatically identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a facial database**.**

Some facial recognition algorithms identify facial features by extracting landmarks, or features, from an image of the subject's face. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. Other algorithms normalize a gallery of face images and then compress the face data, only saving the data in the image that is useful for face recognition. A probe image is then compared with the face data. One of the earliest successful systems is based on template matching techniques applied to a set of salient facial features, providing a sort of compressed face representation.

Recognition algorithms can be divided into two main approaches, geometric, which looks at distinguishing features, or photometric, which is a statistical approach that distills an image into values and compares the values with templates to eliminate variances.

Popular recognition algorithms include Principal Component Analysis using eigen faces, Linear Discriminate Analysis, Elastic Bunch Graph Matching using the Fisher face algorithm, the Hidden Markov model, the Multilinear Subspace Learning using tensor representation, and the neuronal motivated dynamic link matching

1.5 FACIAL DETECTION USING MATLAB (VIOLA JONES ALGORITHM)

For facial recognition many different algorithms can be used.

The process of face recognition comprises of face detection, feature extraction and classification.

For Facial Detection the Viola Jones Algorithm is used.

PSEUDO CODE:-

```
clear all

clc

%Detect objects using Viola-Jones Algorithm


%To detect Face

FDetect = vision.CascadeObjectDetector;


%Read the input image

I = imread('HarryPotter.jpg');


%Returns Bounding Box values based on number of objects

BB = step(FDetect,I);
```

```
figure,
imshow(I); hold on
for i = 1:size(BB,1)
   rectangle('Position',BB(i,:),'LineWidth',5,'LineStyle','-','EdgeColor','r');
end
title('Face Detection');
hold off;
```

The step(Detector,I) returns Bounding Box value that contains [x,y,Height,Width] of the objects of interest.

```
BB =

    52    38    73    73
   379    84    71    71
   198    57    72    72
```

FOR NOSE DETECTION :-

```
%To detect Nose

NoseDetect = vision.CascadeObjectDetector('Nose','MergeThreshold',16);

BB=step(NoseDetect,I);

figure,

imshow(I); hold on

for i = 1:size(BB,1)

   rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-','EdgeColor','b');

end

title('Nose Detection');

hold off
```

EXPLANATION:-

To denote the object of interest as 'nose', the argument 'Nose' is passed.

vision.CascadeObjectDetector('Nose','MergeThreshold',16);

The default syntax for Nose detection :

vision.CascadeObjectDetector('Nose');

Based on the input image, we can modify the default values of the parameters passed to vision.CascaseObjectDetector. Here the default value for 'MergeThreshold' is 4.

When default value for 'MergeThreshold' is used, the result is not correct.

MOUTH DETECTION:

```
%To detect Mouth

MouthDetect = vision.CascadeObjectDetector('Mouth','MergeThreshold',16);


BB=step(MouthDetect,I);



figure,

imshow(I); hold on

for i = 1:size(BB,1)

 rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-','EdgeColor','r');

end

title('Mouth Detection');

hold off;
```

EYE DETECTION:

```
%To detect Eyes

EyeDetect = vision.CascadeObjectDetector('EyePairBig');


%Read the input Image


BB=step(EyeDetect,I);


figure,imshow(I);

rectangle('Position',BB,'LineWidth',4,'LineStyle','-','EdgeColor','b');

title('Eyes Detection');

Eyes=imcrop(I,BB);

figure,imshow(Eyes);

I = imread('harry_potter.jpg');
```

1.6 PCA ALGORITHM

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components. Principal components are guaranteed to be independent if the data set is jointly normally distributed. PCA is sensitive to the relative scaling of the original variables.

PCA is the simplest of the true eigenvector-based multivariate analyses. Often, its operation can be thought of as revealing the internal structure of the data in a way that best explains the variance in the data. If a multivariate dataset is visualised as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional picture, a projection or "shadow" of this object when viewed from its most informative viewpoint. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced.

1.7 PCA FOR FACIAL RECOGNITION

The Eigen Faces are calculated mathematically. Once the eigen faces have been computed, several types of decision can be made depending on the application. What we call face recognition is a broad term which may be further specified to one of following tasks:

1) Identification where the labels of individuals must be obtained,

2) Recognition of a person, where it must be decided if the individual has already been seen,

3) Categorization where the face must be assigned to a certain class.

PCA computes the basis of a space which is represented by its training vectors. These basis vectors, actually eigenvectors, computed by PCA are in the direction of the largest variance of the training vectors. As it has been said earlier, we call them eigenfaces. Each eigenface can be viewed a feature. When a particular face is projected onto the face space, its vector into the face space describe the importance of each of those features in the face. The face is expressed in the face space by its eigenface coefficients (or weights). We can handle a large input vector,facial image, only by taking its small weight vector in the face space. This means that we can reconstruct the original face with some error, since the dimensionality of the image space is much larger than that of face space.

Each face in the training set is transformed into the face space and its components are stored in memory. The face space has to be populated with these known faces. An input face is given to the system, and then it is projected onto the face space. The system computes its distance from all the stored faces.

However, two issues should be carefully considered:

1. What if the image presented to the system is not a face?

2. What if the face presented to the system has not already learned, i.e., not stored as a known face?

The first defect is easily avoided since the first eigenface is a good face filter which can test whether each image is highly correlated with itself. The images with a low correlation can be rejected. Or these two issues are altogether addressed by categorizing following four different regions:

1. Near face space and near stored face =) known faces

2. Near face space but not near a known face =) unknown faces

3. Distant from face space and near a face class =) non-faces

4. Distant from face space and not near a known class =) non-faces

Since a face is well represented by the face space, its reconstruction should be similar to the original, hence the reconstruction error will be small. Non-face images will have a large reconstruction error which is larger than some threshold μr. The distance determines whether the input face is near a known face.

1.8 LIMITATIONS OF PCA ALGORITHM

Limitations of the algorithm, found from the experiments and careful considerations.

1. The face image should be normalized and frontal-view
2. The system is an auto-associative memory . It is harmful to be overfitted.
3. Training is very computationally intensive.
4. It is hard to decide suitable thresholds .
5. The suggested methods to deal with unknown faces and non-faces are not good enough to differentiate them from known faces

SOME SERIOUS QUESTIONS ABOUT THE USE OF PCA-

1) How many eigenvectors (eigenfaces) are required to recover an original face by some percent of the total?

2) Are the images of the same subject projected onto similar (near) points in the feature space?

3) Is there any idea to decide the threshold automatically?

4) Is there another tool to do similar things as PCA do?

CHAPTER 2

DESIGN AND IMPLEMENTATION

2.1 BLOCK DIAGRAM

The implementation of this project was done using Arduino IDE and MATLAB.

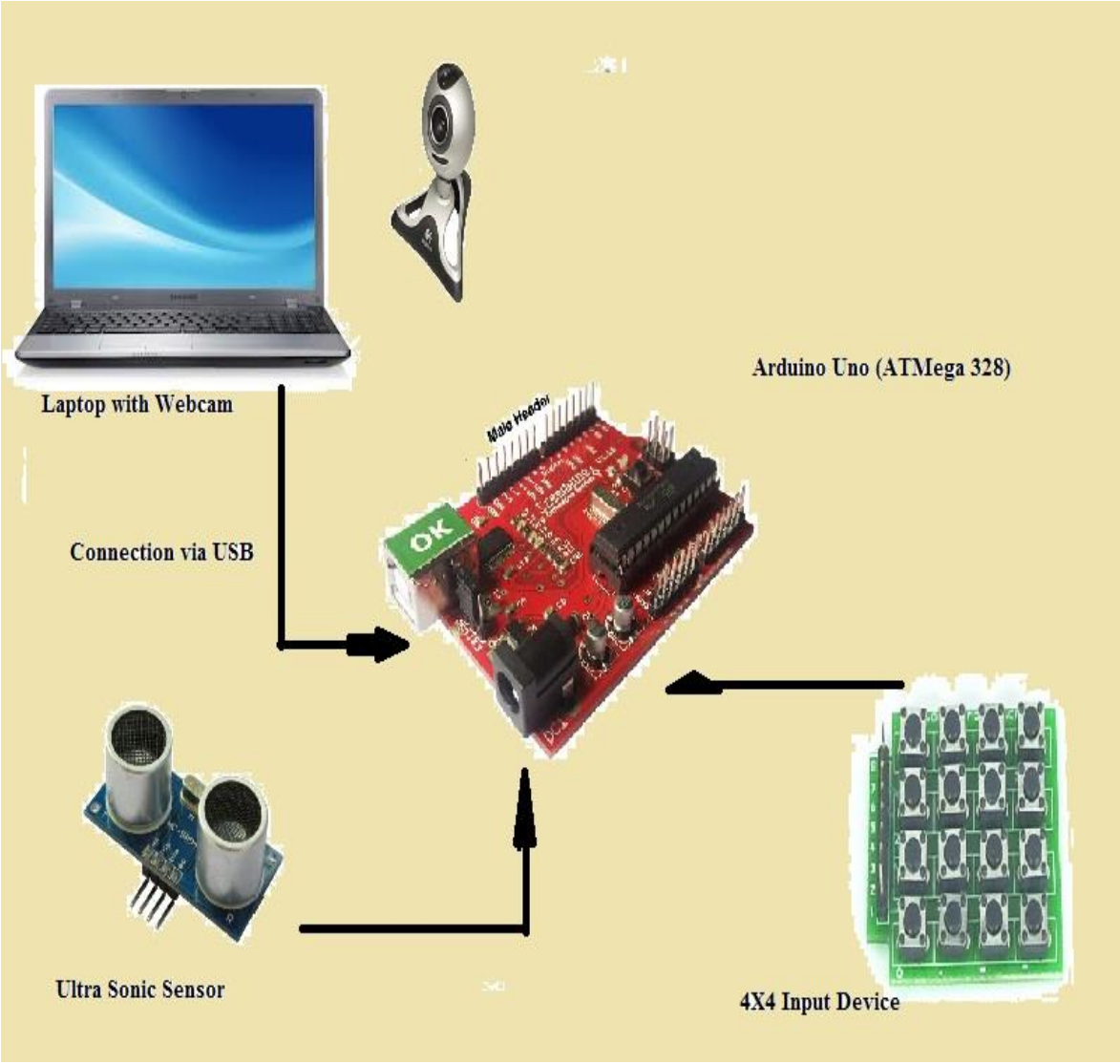


Fig 2.1 Block diagram of Project implementation

2.2 PASSIVE INFRARED SENSOR

A **passive infrared sensor** (**PIR sensor**) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors.

All objects with a temperature above absolute zero emit heat energy in the form of radiation. Usually this radiation is invisible to the human eye because it radiates at infrared wavelengths, but it can be detected by electronic devices designed for such a purpose.

The term *passive* in this instance refers to the fact that PIR devices do not generate or radiate any energy for detection purposes. They work entirely by detecting the energy given off by other objects.[1] It is important to note that PIR sensors don't detect or measure "heat" per se; instead they detect the Infrared radiation emitted from an object which is different from but often associated/correlated with the object's temperature (e.g., a detector of X-rays or gamma rays would not be considered a heat detector, though high temperatures may cause the emission of X or gamma radiation).

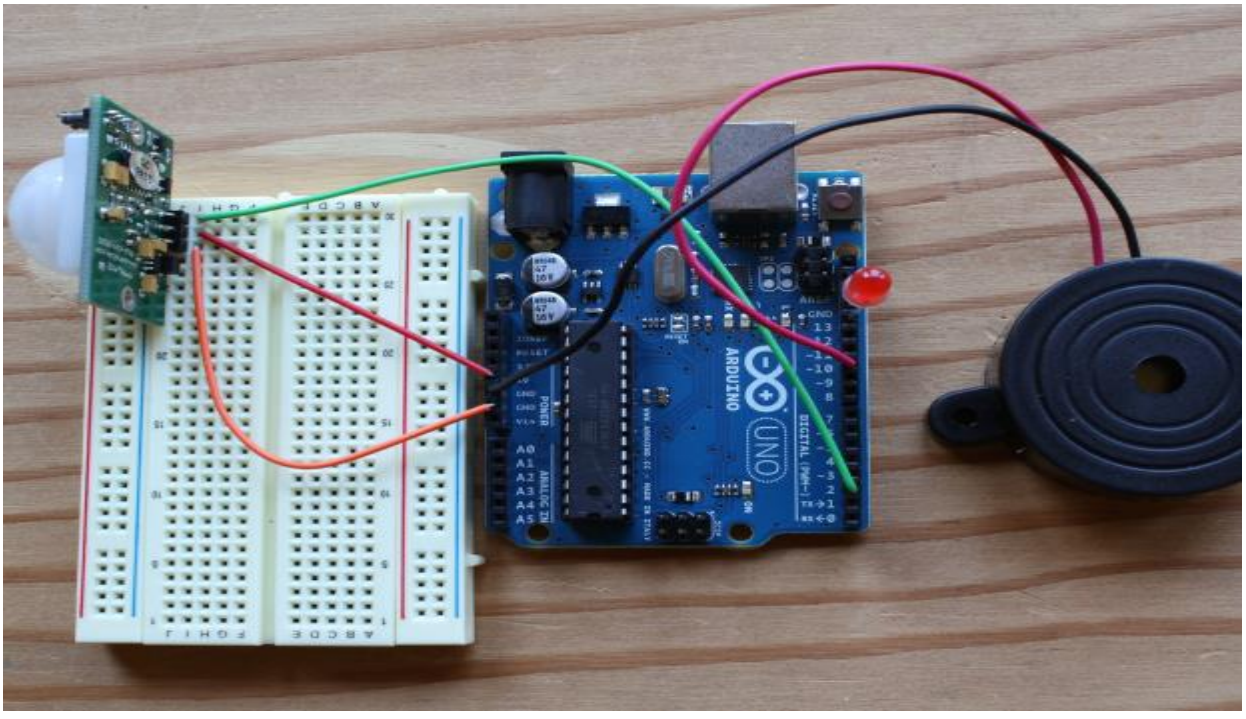


Figure 2.2 Interfacing of PIR Sensor and Buzzer alarm using Arduino

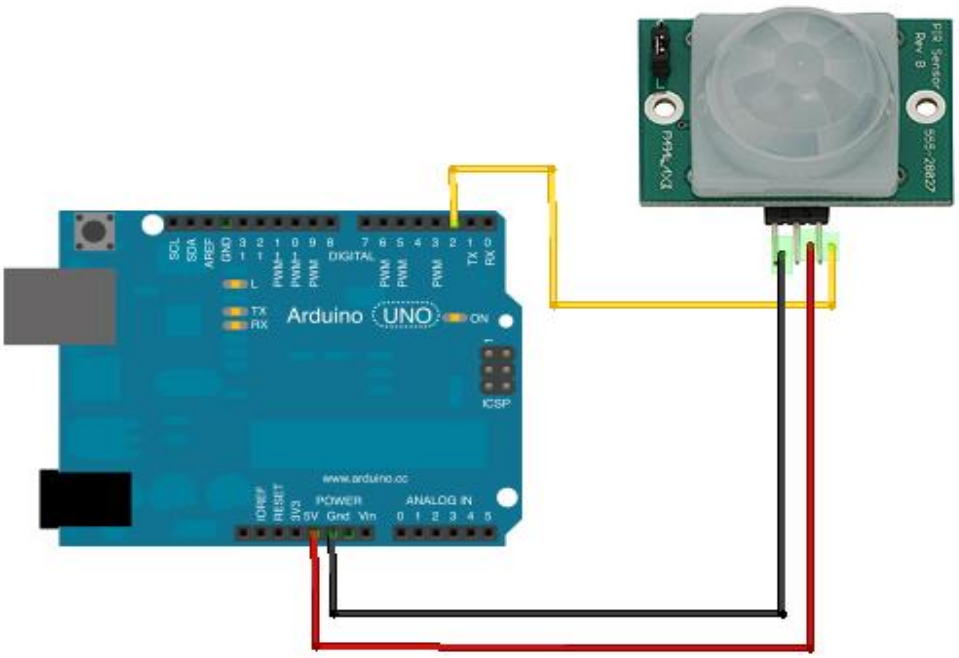


Figure 2.3 Pin Diagram of Arduino interfacing with PIR sensor.

The PIR Sensor Detects motion in front of it via the input pins and then simultaneously sound the buzzer alarm sounds and informs the owner about the change in the immediate surroundings.

The code for interfacing PIR sensor and the buzzer alarm with Arduino :-

```
int ledPin = 13;                // choose the pin for the LED
int inputPin = 2;               // choose the input pin (for
PIR sensor)
int pirState = LOW;             // we start, assuming no
motion detected
int val = 0;                    // variable for reading the
pin status
int pinSpeaker = 10;           //Set up a speaker on a PWM
pin (digital 9, 10, or 11)

void setup() {
  pinMode(ledPin, OUTPUT);      // declare LED as output
  pinMode(inputPin, INPUT);     // declare sensor as input
  pinMode(pinSpeaker, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  val = digitalRead(inputPin);  // read input value
  if (val == HIGH) {            // check if the input is HIGH
    digitalWrite(ledPin, HIGH);  // turn LED ON
    playTone(300, 160);
    delay(150);


    if (pirState == LOW) {
      // we have just turned on
      Serial.println("Motion detected!");
```

```
      // We only want to print on the output change, not
state
      pirState = HIGH;
    }
  } else {
      digitalWrite(ledPin, LOW); // turn LED OFF
      playTone(0, 0);
      delay(300);
      if (pirState == HIGH){
      // we have just turned of
      Serial.println("Motion ended!");
      // We only want to print on the output change, not
state
      pirState = LOW;
    }
  }
}
// duration in mSecs, frequency in hertz
void playTone(long duration, int freq) {
    duration *= 1000;
    int period = (1.0 / freq) * 1000000;
    long elapsed_time = 0;
    while (elapsed_time < duration) {
        digitalWrite(pinSpeaker,HIGH);
        delayMicroseconds(period / 2);
        digitalWrite(pinSpeaker, LOW);
        delayMicroseconds(period / 2);
        elapsed_time += (period);
    }
}
```

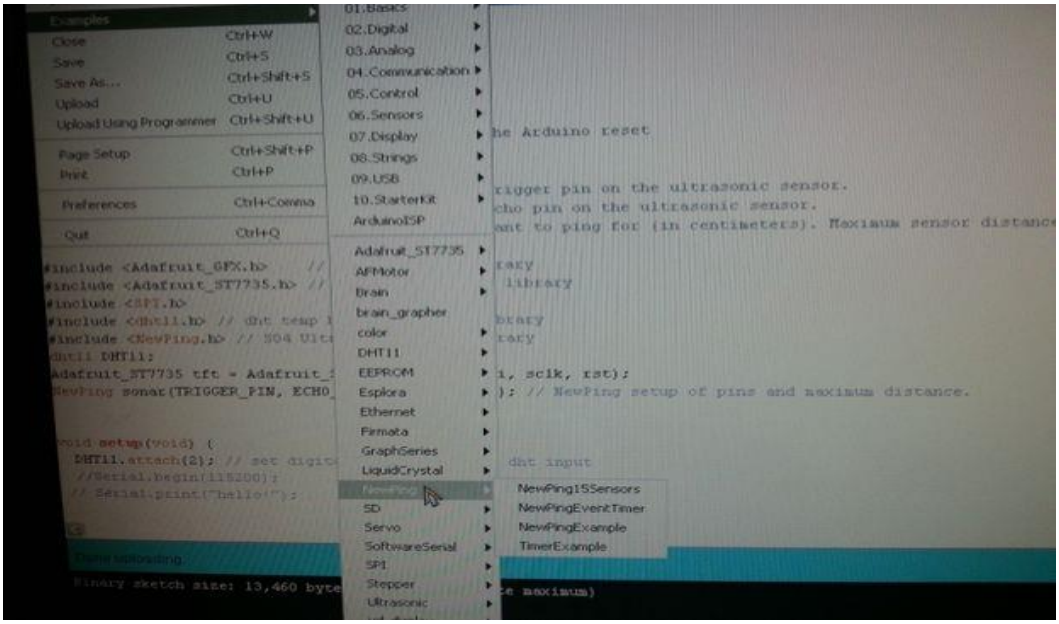


Figure 2.4 Snapshot of Arduino IDE at the time of Ping programming.

2.3 WEB CAMERA

Any web camera can be set with the Computer/ Laptop used for the project. The webcam can be integrated with the laptop or a simple webcam with USB connection option.

The webcam is interfaced with the project with the help of MATLAB and the image of any obstruction/intruder is taken and sent to the authorized email id using MATLAB.

As soon as the Ultrasonic sensor detects the obstruction it sends a signal to the arduino board which is set at a COM PORT 9 this COM PORT's input is then inputted to the matlab code which takes appropriate action according to the code and the conditions set by the programmer .

```
vid=videoinput('winvideo',1,'YUY2_320x240');

ser=serial('COM3','BaudRate',9600);

fopen(ser);

i=getsnapshot(vid);

i1=ycbcr2rgb(i);

imview(i1);

fprintf(ser,'s');

imwrite(i1,'D:/image.png');

fread(ser); %read binary data from device

fclose(ser);
```

(MATLAB Code for image capturing and saving the image in D:/)

2.4 CONNECTION TO GMAIL

The connection to Gmail is a very important part in the project because the saved image is sent to the authorized personnel using Email only.

The Gmail connection is done via MATLAB which on receiving the information of the image via the ultrasonic sensor and Arduino takes an image of the intruder using the integrated web camera and then stores image in a particular specified location and sends the image as an attachment to the gmail id mentioned in the code along with a panic message.

The password of the specified gmail id is also specified in the code to grant access to the account.

Code for connection to Gmail-

```
%For Gmail

myaddress = 'emailid';

mypassword = 'password';

setpref('Internet','E_mail',myaddress);

setpref('Internet','SMTP_Server','smtp.gmail.com');
```

```
setpref('Internet','SMTP_Username',myaddress);

setpref('Internet','SMTP_Password',mypassword);


props = java.lang.System.getProperties;

props.setProperty('mail.smtp.auth','true');

props.setProperty('mail.smtp.socketFactory.class', ... 'javax.net.ssl.SSLSocketFactory');

props.setProperty('mail.smtp.socketFactory.port','465');

sendmail(myaddress, 'subject','body',{'D:/image.png'});
```

2.5 4X4 MATRIX KEYPAD MODULE

A 4x4 Matrix keypad module is attached to the arduino board and interfaced in such a way that if the right combination is pressed on the module the Email is not sent and the ultrasonic sensor is delayed by 10 seconds to allow an authorized entry without capturing of image and the sending of email.

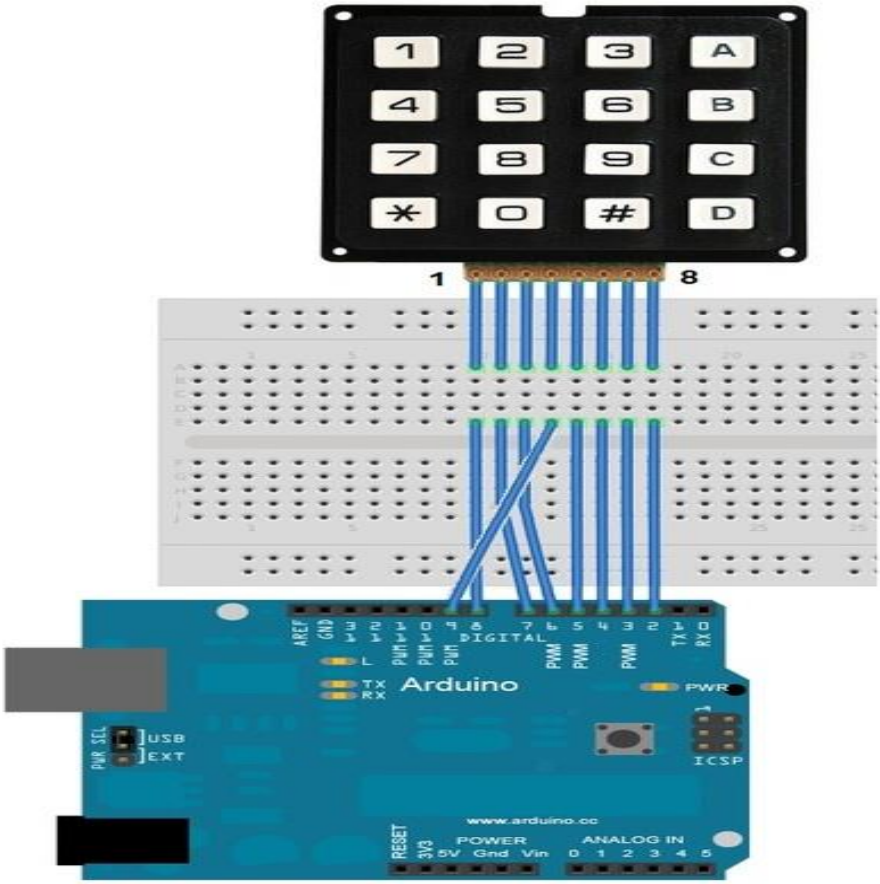


Figure 2.5 Interfacing of Arduino UNO board with a 4X4 matrix keypad module

Code for interfacing 4X4 matrix-

```
#include <Keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
char keys[ROWS][COLS] =
 {{'A','1','2','3'},
  {'B','4','5','6'},
  {'C','7','8','9'},
  {'D','#','0','*'}};
byte rowPins[ROWS] = {5, 4, 3, 2}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {  9, 8, 7, 6}; //connect to the column pinouts of the keypad
int count=0;

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup()
{
```

```
  Serial.begin(9600);
}

void loop()
{
  char key = keypad.getKey();
  if (key != NO_KEY)
  {
    Serial.print(key);
    count++;
   if (count==17)
    {
      count=0;
    }
  }
}
```

Functions

A. ***void begin(makeKeymap(userKeymap))***

Initializes the internal keymap to be equal to userKeymap
[See File -> Examples -> Keypad -> Examples -> CustomKeypad]

B. ***char waitForKey()***

This function will wait forever until someone presses a key. Warning: It blocks all other code until a key is pressed. That means no blinking LED's, no LCD screen updates, no nothing with the exception of interrupt routines.

C. ***char getKey()***

Returns the key that is pressed, if any. This function is non-blocking.

D. ***KeyState getState()***

Returns the current state of any of the keys.
The four states are IDLE, PRESSED, RELEASED and HOLD.

E. ***boolean keyStateChanged()***

New in version 2.0: Let's you know when the key has changed from one state to another. For example, instead of just testing for a valid key you can test for when a key was pressed.

F. ***setHoldTime(unsigned int time)***

Set the amount of milliseconds the user will have to hold a button until the HOLD state is triggered.

G. ***setDebounceTime(unsigned int time)***

Set the amount of milliseconds the keypad will wait until it accepts a new keypress/keyEvent. This is the "time delay" debounce method.

H. ***addEventListener(keypadEvent)***

Trigger an event if the keypad is used. You can load an example in the Arduino IDE. [See File -> Examples -> Keypad -> Examples -> EventSerialKeypad] or see the KeypadEvent Example code.

I. ***For Now***

Here's the list of multi-keypress functions and the keylist definition. I will complete their descriptions this weekend.

- Key key[LIST_MAX]
- bool getKeys()
- bool isPressed(char keyChar)
- int findInList(char keyChar)

CHAPTER 3

FUTURE SCOPE

3.1 FUTURE SCOPE

Just like all the other college level projects this project can also be extended further to increase functionality and achieve better results and practicality.
Different safety and day to day features could be interfaced using sensors to the Arduino Board.

1) Integration with GSM module can be done so that text message is sent after intrusion.

2) Face Recognition using PCA can be implemented.

3) Buzzer alarm can also be incorporated.

4) Temperature Sensor can also be added.

5) The use of smoke detector and gas leakage can also be monitored using sensors.

REFRENCES

[1]people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2004/ch236/securitysystem.pdf

[2] http://www.instructables.com/id/Arduino-Home-Monitor-System/?ALLSTEPS

[3] https://learn.adafruit.com/wireless-security-camera-arduino-yun/introduction

[4] http://duino4projects.com/security-automation-sensors-using-arduino/

[5] http://arduino.cc/en/Main/ArduinoBoardADK

[6] http://www.ijecct.org/v3n2/(382-385)0302M22.pdf

[7] E. Yavuz, B. Hasan, I. Serkan and K. Duygu. "Safe and Secure PIC Based Remote Control Application for Intelligent Home". International Journal of Computer Science and Network Security, Vol. 7, No. 5, May 2007.

[8] Al-Ali, Member, IEEE & M. AL-Rousan,"Java-Based Home Automation System R." IEEE Transactions on Consumer Electronics, Vol. 50, No. 2, MAY 2004

[9] Muhammad Izhar Ramli, Mohd Helmy Abd Wahab, Nabihah, "TOWARDS SMART HOME: CONTROL ELECTRICAL DEVICES ONLINE" ,Nornabihah Ahmad International Conference on Science and Technology: Application in Industry and Education (2006)

[10] Rana, Jitendra Rajendra and Pawar, Sunil N., Zigbee Based Home Automation (April 10, 2010). Available at SSRN: http://ssrn.com/abstract=1587245 or http://dx.doi.org/10.2139/ssrn.1587245

[11] N. Sriskanthan and Tan Karand. "Bluetooth Based Home Automation System". Journal of Microprocessors and Microsystems, Vol. 26, 2002.

[12] R. Bhatia, L.E. Li, H. Luo, and R. Ramjee, "ICAM: Integrated Cellular and Ad Hoc Multicast," IEEE Trans. Mobile Computing, vol. 5, no. 8, pp. 1004-1015, Aug. 2006.