VIT® UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore • Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

SCHOOL OF ELECTRONICS ENGINEERING

VIT - CHENNAI CAMPUS

ECE-312 OPERATING SYSTEMS

GESTURE BASED LOCKING SYSTEM

Submitted by:
Prithvish V N – 13BEC1100
Shraddha Agrawal – 13BEC1138

Guiding Faculty:

Prof. R. K. Singh SCSE – VIT Chennai

Table of Content

ABSTRACT

This project aims to design a revolutionary security system that shall change the present day scenario of opening door locks. A mere gesture of your hand shall unlock the door on which the system is installed. The basic prototype will consist of a small cubical box with laser and LDR array. When the proper Finger gesture is produced in the box, a comparison is made based on a previous pattern recorded. A Stack and polling based concept of the microcontroller/microprocessor will come into picture while interfacing the Laser-LDR system. On circumstantial failure of the above Sub-System after a particular amount of trials, the locking system starts its own access point network through which the owner can connect via his phone. Further a custom made app will be needed to unlock the lock. In this process the lock system will start its own server and the owner’s phone shall act as client. On the door opening request of the client the server will open and close the door when properly authenticated by the correct password. The level of security in such a system is doubled in the second subsystem with a 2 level authentication by the WEP encryption based WIFI security and the App Passkey security.

INTRODUCTION

Security has been an issue ever since we have entered the modern era. Various types of security issues are introduced day after day; Cyber security is one such security which has to maintain itself up-to date from the hackers and black-hats. Our Project aims on solving a manual locking system based security issues by the means of a security standards available at the cyber security world. WIFI networks are evolving day after day and so are their security standards. The main aim is to connect a hardware device to Wireless (WIFI) platform and control its working via the previously available software applications (apps).

BRIEF DESCRIPTION

The Project is divided into 2 sections

- The LDR-LASER System
- APP Unlock System

Both the sub-systems are mounted on a single system on a chip BBB. The door unlocking procedure begins with initialization of the gesture based unlocking system through the LDR-LASER grid On conditional failure of the laser-LDR system or client exceeding the number of permitted trials the app unlock system shall instantiate by itself. A particular number of trials are given to the client or the one who wants to unlock the door. After then the server stops and the client will have to wait for couple of minutes as per coded in the system and then re-enter the passkey.

BLOCK DIAGRAM

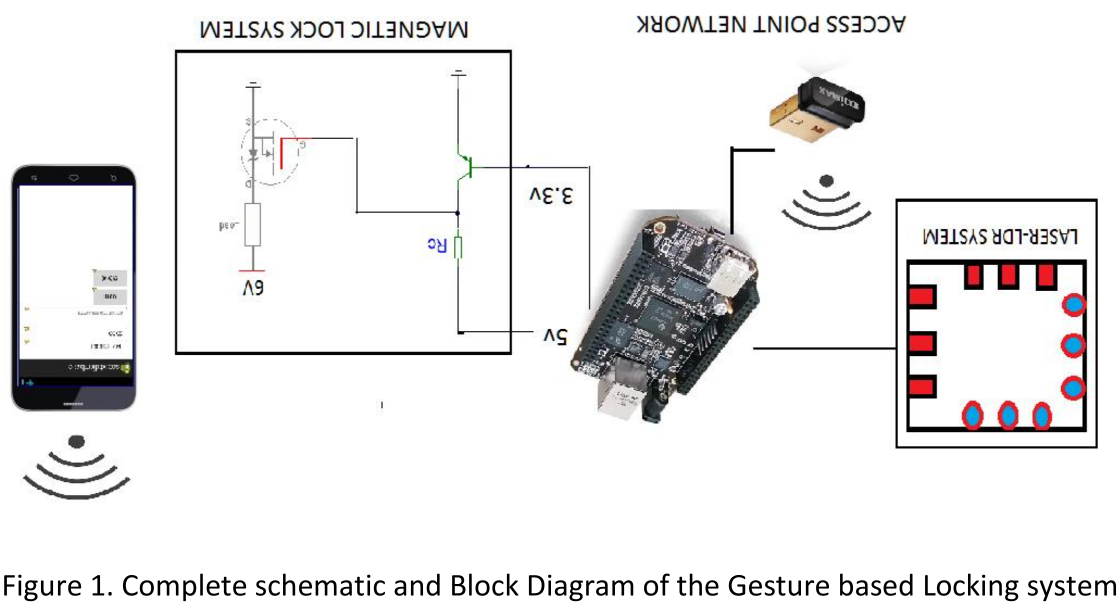


Figure 1. Complete schematic and Block Diagram of the Gesture based Locking system

COMPONENTS

BEAGLE BONE BLACK

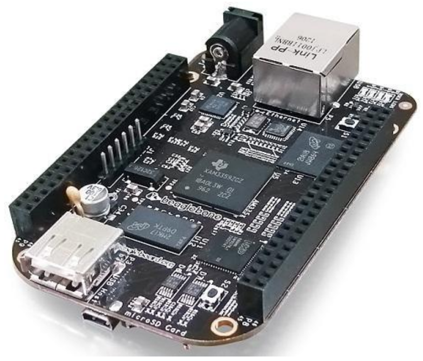


Figure 2. Beagle Bone Black

Beagle Bone Black is a low-cost, community-supported development platform with a TI processor AM335x 1GHz ARM® Cortex-A8, 512 MB DDR3 RAM, 4GB 8-bit eMMC on-board flash storage and a USB-Client which abridges the task of interfacing with the USB based WIFI Adapter and GPIOs which help with interfacing of the MAGNETIC DOOR LOCKING SYSTEM and LDR-LASER GRID. The Linux based platform entitles the user to design on their favored programming language. Its pocket sized structure provides an easy-to-fit-in feasibility. Figure above shows the Beagle Bone Black

USB WIFI ADAPTER (EDIMAX-RTL8188CUS)

Figure 3. USB WIFI Adapter EDIMAX

A 150Mbps Wireless N Nano USB Adapter N150 is a low cost, small sized USB Wi-Fi Adapter which gives connectivity over any local hosted network as well as allows you to host a local network. It supports IEEE 802.11 b/g/n standards with an operating frequency range of 2.400-2.4835GHz. It has an RTL-8188CU Real-teak chipset which supports infrastructure and Ad-hoc mode. Figure above shows Wireless N Nano USB Adapter.

ELECTRMAGNET – PLATE PAIR

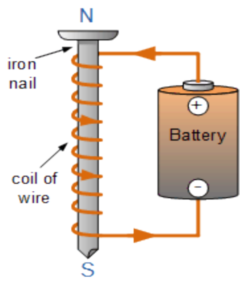


Figure 4. Electromagnet

An electromagnet is a type of magnet in which the magnetic field is produced by an electric current. The magnetic field disappears when the current is turned off. Electromagnets usually consist of a large number of closely spaced turns of wire that create the magnetic field. The wire turns are often wound around a magnetic core made from a ferromagnetic material such as iron; the magnetic core concentrates the magnetic flux and makes a more powerful magnet. In our case it acts as a powerful lock which is controlled by the controlled current passing through it.

DRIVER MOSFET

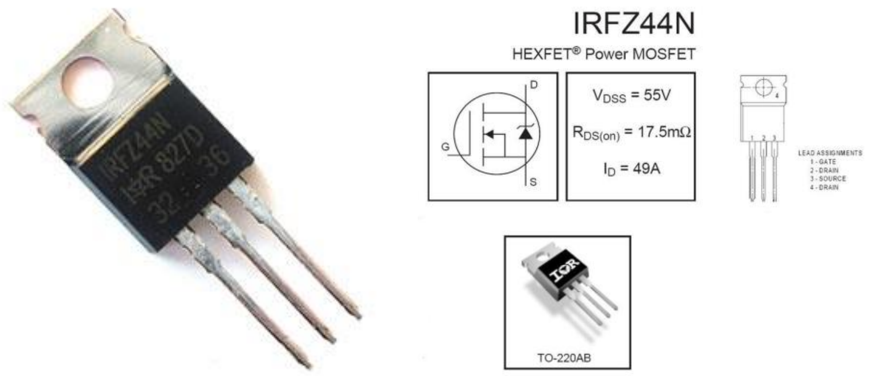


Figure 5. IRFZ44N POWER MOSFET

IRFZ44N is a power N-channel MOSFET which supports a drain source current up-to 49 A with a gate voltage of 5V which is used in our case as the driving stage for the electromagnetic load

LEVEL SHIFTING TRANSISTER

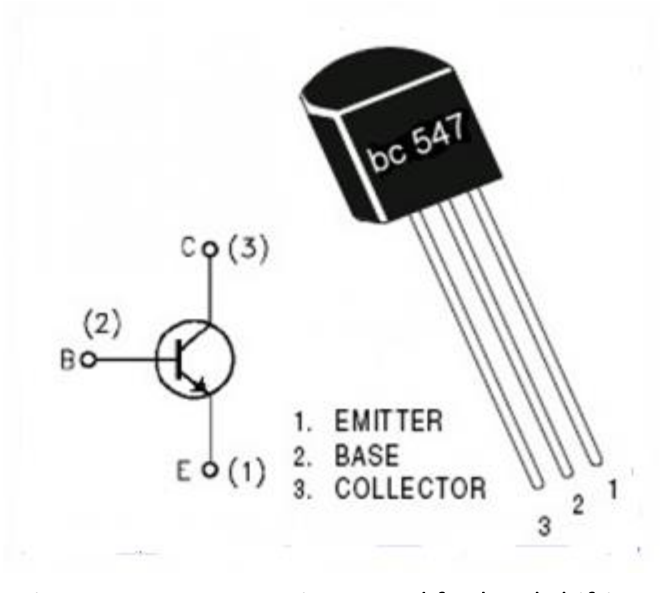


Figure 6. BC547 Transistor used for level shifting

BC547 is an ideal high gain transistor which switches at a voltage of 3.3 V sufficient enough to switch the gate of the High Power MOSFET. Basically it convert the 3.3 level high logic to 5 V high logic

LASER-LDR GRID

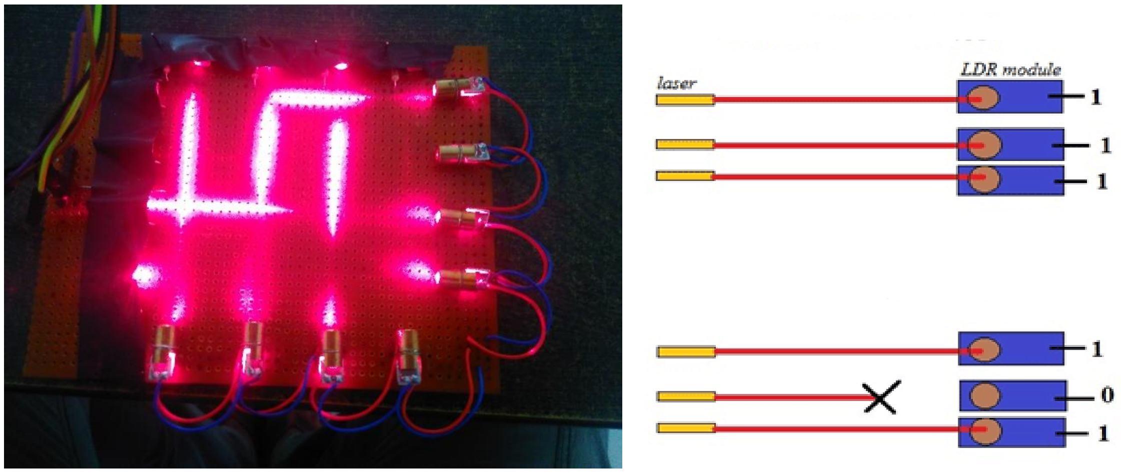


Figure 7. LASER LDR System

A laser LDR grid made of low cost LASER Diodes and low resistive range LDR’s with normal resistors forming a potential divider circuit. The LASER in the system are placed at equal spacing at the 2 adjacent edges of a square and LDR placed at the opposite edges of the laser facing the laser with a proper alignment as shown in the figure such that the laser light hits the center of the LDR reducing the resistance of the LDR to its max.

ALGORITHM

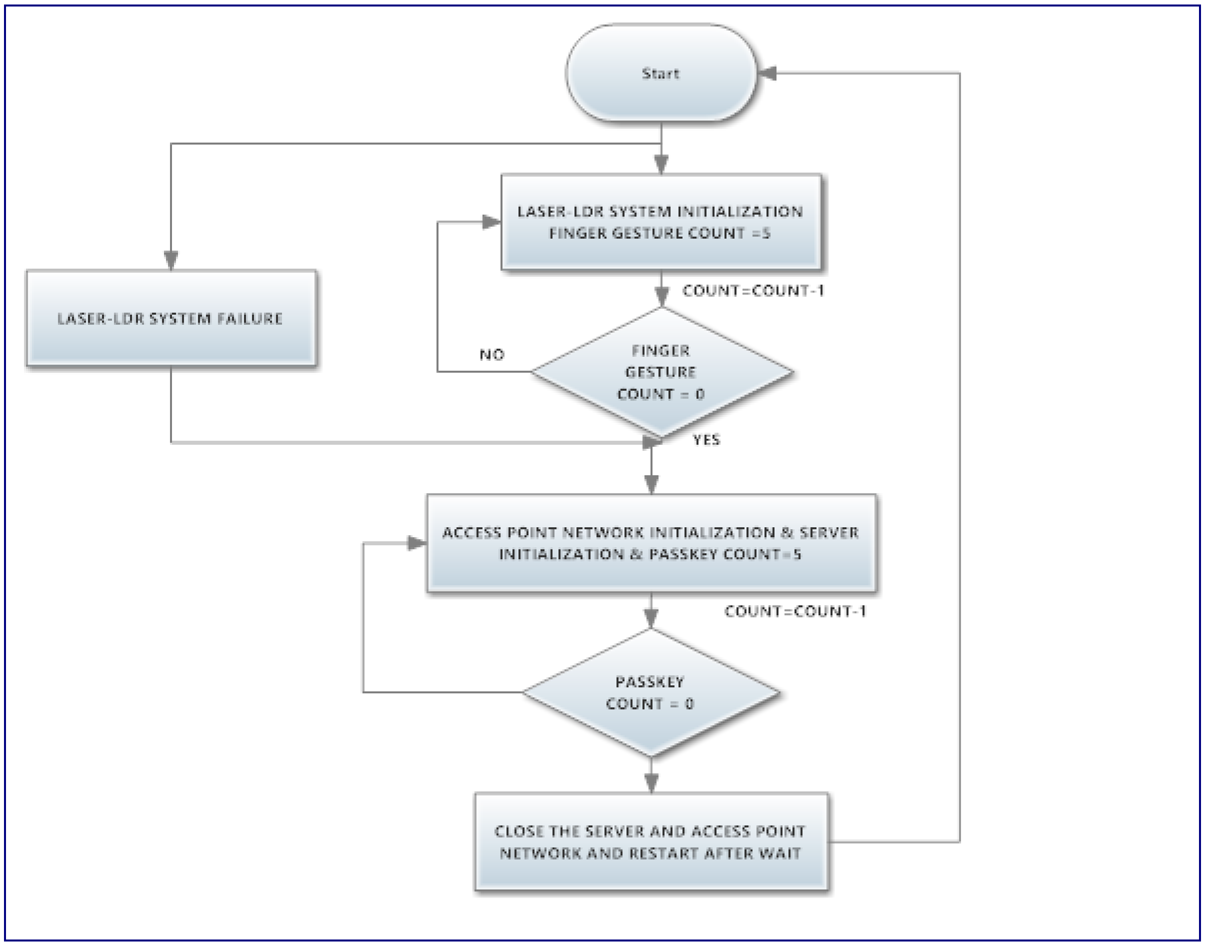

Start
LASER-LDR SYSTEM INITIALIZATION
FINGER GESTURE COUNT =5
LASER-LDR SYSTEM FAILURE
COUNT=COUNT-1
NO
FINGER GESTURE COUNT = 0
YES
ACCESS POINT NETWORK INITIALIZATION & SERVER INITIALIZATION & PASSKEY COUNT=5
COUNT=COUNT-1
PASSKEY COUNT = 0
CLOSE THE SERVER AND ACCESS POINT NETWORK AND RESTART AFTER WAIT

Figure 8. Working Flow of the Gesture based Locking System

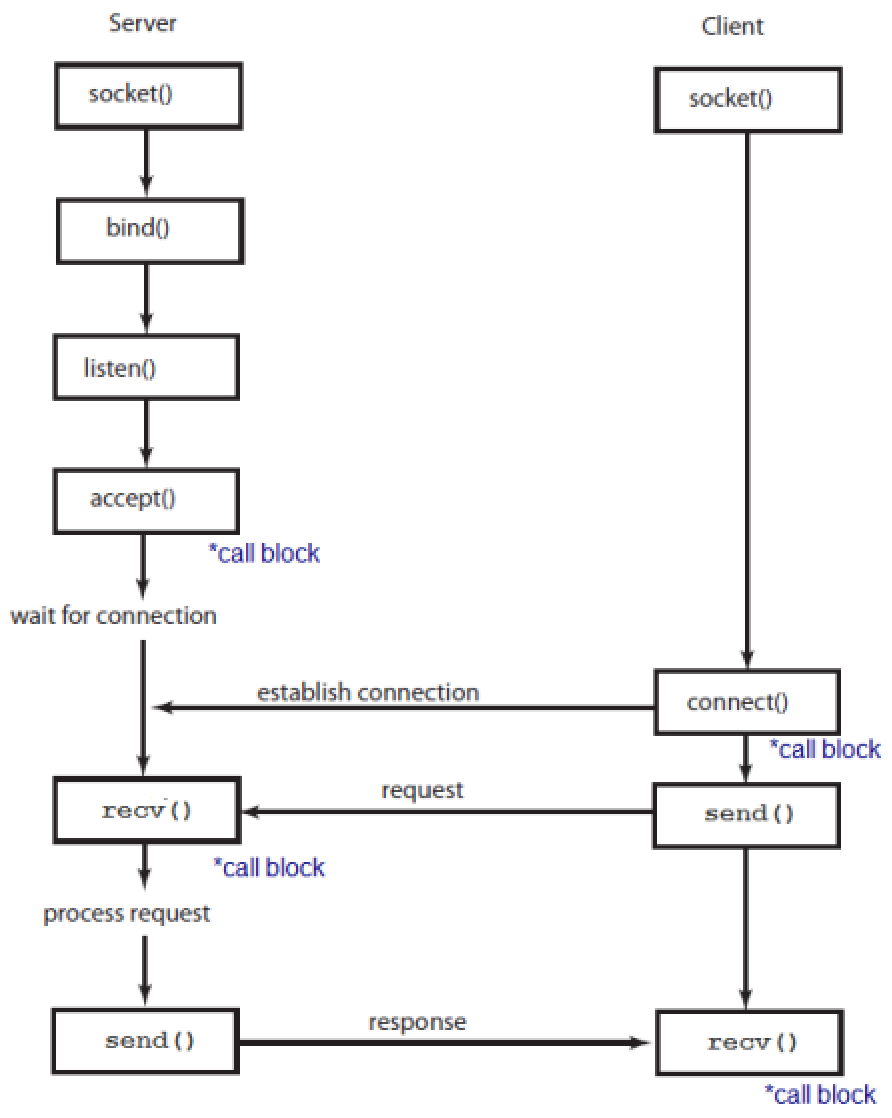
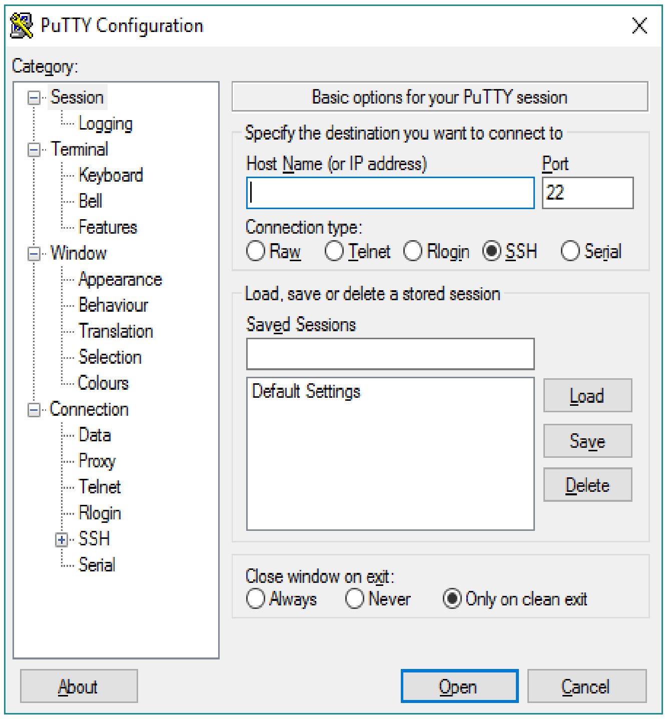


Figure 9. Working Flow of the network based SOCKET Formation and Client Server communication

IMPLEMENTATION

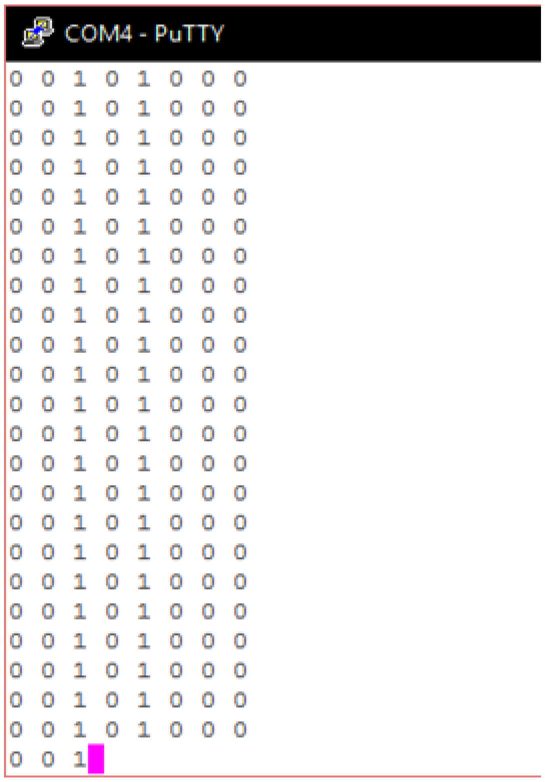


Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

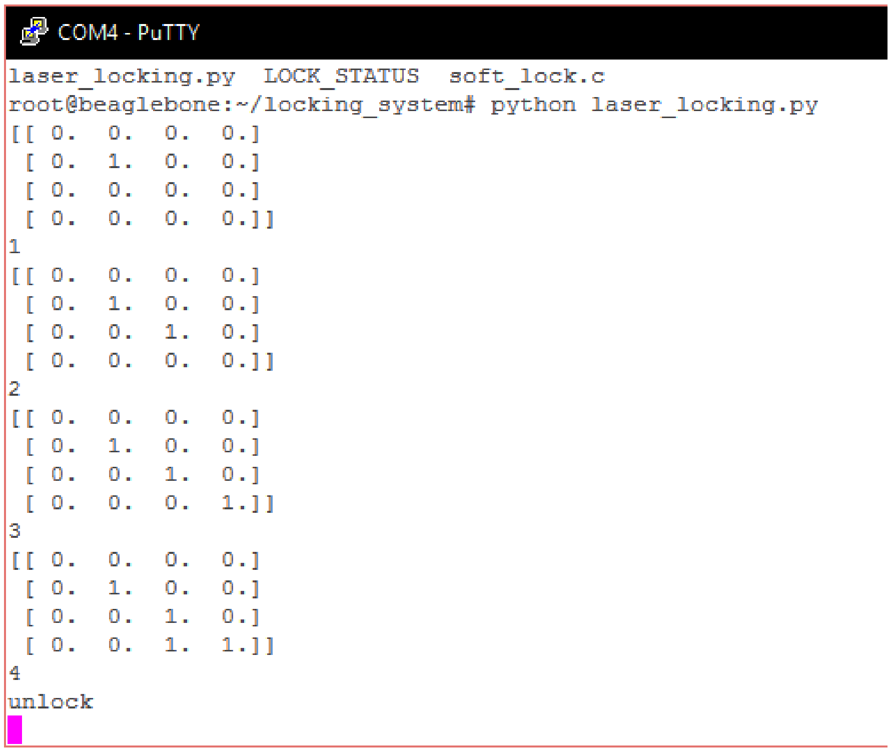
Close window on exit: Always Never [✓ Only on clean exit]

Figure 10. The serial/SSH terminal used to debug the system



```
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0
0 0 1
```

Figure 11. LDR Laser System Testing



```
laser_locking.py  LOCK_STATUS  soft_lock.c
root@beaglebone:~/locking_system# python laser_locking.py
[[ 0.   0.   0.   0.]
 [ 0.   1.   0.   0.]
 [ 0.   0.   0.   0.]
 [ 0.   0.   0.   0.]]
1
[[ 0.   0.   0.   0.]
 [ 0.   1.   0.   0.]
 [ 0.   0.   1.   0.]
 [ 0.   0.   0.   0.]]
2
[[ 0.   0.   0.   0.]
 [ 0.   1.   0.   0.]
 [ 0.   0.   1.   0.]
 [ 0.   0.   0.   1.]]
3
[[ 0.   0.   0.   0.]
 [ 0.   1.   0.   0.]
 [ 0.   0.   1.   0.]
 [ 0.   0.   1.   1.]]
4
unlock
```

Figure 12. Gesture based LDR-Laser System in Debug mode

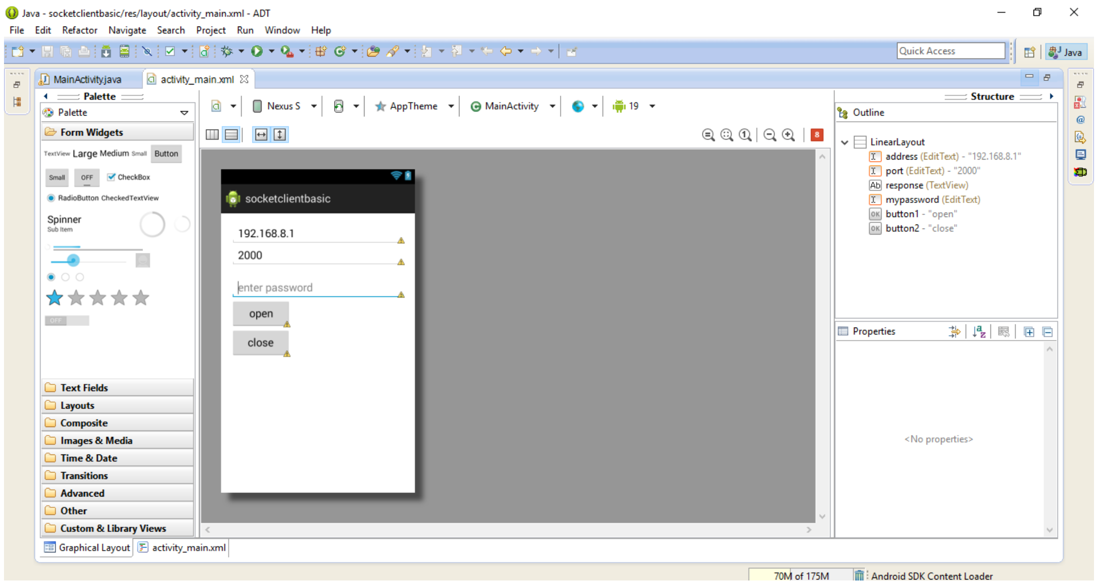


Figure 13. APP development on Eclipse android SDK development

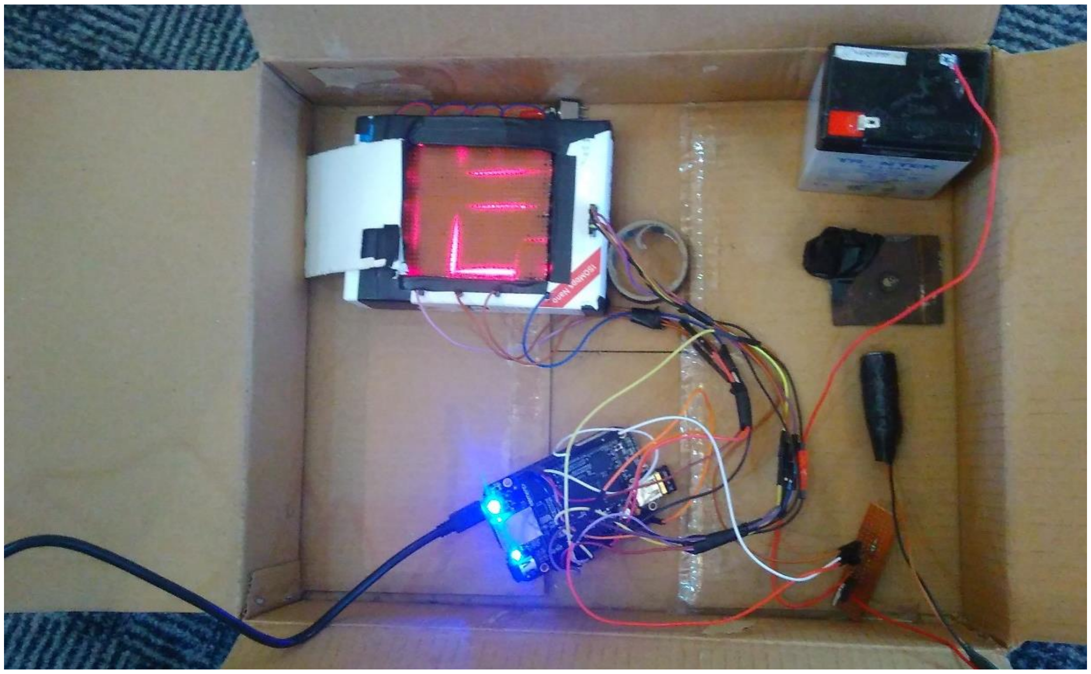


Figure 14. Final Product demonstration

Putty was used for secure shell connection with BBB. Further for setting up the Access point network initially Drivers of the USB Wifi adapter Chipset (RTL8188CUS) were installed. Further packages such as Hostapd and Dnsmasq were used for access point network and IP distribution. The LASER LDR system was coded in Python due to the easy to use environment provided by the prebuilt Adafruit Libraries. Futher the server was coded in C for a better connection establishment with Android App. To Combine the programs in the system Shell files and file handling was used. The software app was developed on Android Development Tools on Eclipse IDE. The system to run in standalone service files were made.

CODING

SETTING UP A WIFI HOTSPOT (AP MODE) ALONG WITH WIFI ACCESS ON A BBB USING A EDIMAX WIFI ADAPTER (RTL8188CUS/RTL8192CU CHIPSET)
Specific to version – 3.8.13-bone79

STEP 1: SETTING UP THE KERNEL

Execute the following commands one by one on the command window

apt-get update
apt-get install lsb-release
apt-get install git
apt-get install make
apt-get install gcc
apt-get install build-essential python-dev python-pip python-smbus -y
apt-get install chkconfig
apt-get upgrade
mkdir ~/beaglebone-ubuntu-scripts
cd ~/beaglebone-ubuntu-scripts
nano script.sh

SPEACIAL NOTE: The folder linux-3.8.16-bone70 must be added to the folder /usr/src of the beagle bone black before excecuting the shell script mentioned below.

Link to the folder: https://drive.google.com/file/d/0B2DABrJSbxM0STFyN2JXekZ5RTQ/view?usp=sharing

Note: Extract the folder and copy to the exact location as specified above in the /usr/src directory on the BBB

Contents of the file: script.sh

```
#!/bin/sh

mkdir -p "/lib/modules/${KVER}"
rm -f "/lib/modules/${KVER}/build"
ln -s "/usr/src/linux-3.8.13-bone70" "/lib/modules/build"
prepare_kernel_source ()
{
        CURPWD=$PWD
        cd "/usr/src/linux-3.8.13-bone70"

        make oldconfig
        make prepare
        make scripts
```

```
                cd "${CURPWD}"
}
prepare_kernel_source
exit 0
```

```
chmod +x script.sh
./script.sh
reboot
******************
```

NOTE: Before executing the following commands make sure the time is up-to-date if not update it using ntpdate or RTC

```
cd /usr/src
cd linux-3.8.13-bone*    NOTE : * means its differs for different bone
make oldconfig
make prepare
make modules_prepare
```

STEP 2: SETTING UP THE DRIVER FOR THE USB WIFI ADAPTER

NOTE: The following driver creates 2 wireless access points on which one will be WIFI Access and other will be HOTSPOT (AP mode)

Note: Make sure the date is updated from the net

Download the drivers from a github repository and compile and install

GIT REPO LINK: https://github.com/jlucidar/FabMo-Platform/tree/master/Arch_linux_config/rtl8188CUS-driver-beaglebone

Execute the following commands to download, compile and install the Driver

```
git clone https://github.com/jlucidar/FabMo-Platform
cd FabMo-Platform/Arch_linux_config/rtl8188CUS-driver-beaglebone
nano Makefile
```

Edit the following in the make-file <on line 533>

```
ifeq ($(CONFIG_PLATFORM_BEAGLEBONE), y)
EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN
EXTRA_CFLAGS += -DCONFIG_IOCTL_CFG80211 -DRTW_USE_CFG80211_STA_EVENT
ARCH=arm
CROSS_COMPILE := arm-linux-gnueabi-
KVER  := $(shell uname -r)
KSRC ?= /lib/modules/$(KVER)/build
MODDESTDIR := /lib/modules/$(KVER)/kernel/drivers/net/wireless/
INSTALL_PREFIX :=
endif
```

---

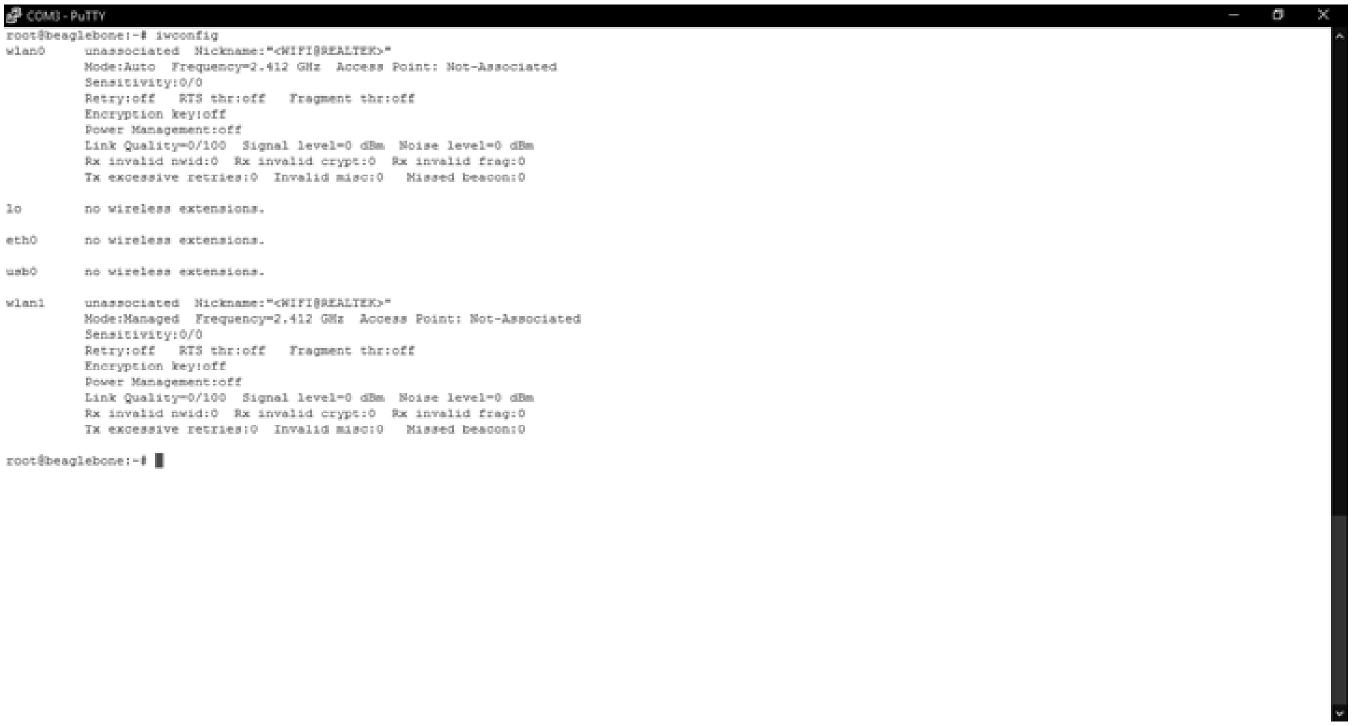
to

```
ifeq ($(CONFIG_PLATFORM_BEAGLEBONE), y)
EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN
EXTRA_CFLAGS += -DCONFIG_IOCTL_CFG80211 -DRTW_USE_CFG80211_STA_EVENT
ARCH=arm
CROSS_COMPILE ?=
KVER  := $(shell uname -r)
KSRC ?= /lib/modules/build
MODDESTDIR := /lib/modules/$(KVER)/kernel/drivers/net/wireless/
INSTALL_PREFIX :=
endif
```

Save the Make file and execute the following commands

make
make install
reboot
****

iwconfig <NOTE: To check whether the following the following driver works , In this driver no need to blacklist any other preinstalled driver>



```
root@beaglebone:~# iwconfig
wlan0     unassociated  Nickname:"<WIFI@REALTEK>"
          Mode:Auto  Frequency=2.412 GHz  Access Point: Not-Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=0/100  Signal level=0 dBm  Noise level=0 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

lo        no wireless extensions.

eth0      no wireless extensions.

usb0      no wireless extensions.

wlan1     unassociated  Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency=2.412 GHz  Access Point: Not-Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=0/100  Signal level=0 dBm  Noise level=0 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

root@beaglebone:~# 
```

You shall see the wireless interfaces wlan0 and wlan1 which confirms that the following driver is installed
You can edit the interfaces file and the following
nano /etc/network/interfaces

Add the following in the /etc/network/interfaces

```
auto wlan0
iface wlan0 inet static
address 192.168.8.1
netmask 255.255.255.0
```

```
hostapd -dd /etc/hostapd/hostapd.conf
post-up /usr/sbin/service hostapd restart

auto wlan1
iface wlan1 inet dhcp
   wpa-ssid "Phynart Technologies"
   wpa-psk  "Deployment_Phase_1"
```

You can start and end manually WIFI and HOTSPOT by using

```
ifup wlan0/1
ifdown wlan0/1
```

NOTE: IN CASE THERE IS PACKET LOSS IN THE DATA DISABLE THE HDMI

```
nano /boot/uEnv.txt
# uncomment disable HDMI
reboot
```

****

STEP 3: SETTING UP THE HOSTAPD FOR THE HOTSPOT

Download the HOSTAPD module from the GITHUB repository, compile and configure it.

GIT REPOSITORY LINK: https://github.com/jenssegers/RTL8188-hostapd

```
apt-get autoremove hostapd
wget https://github.com/jenssegers/RTL8188-hostapd/archive/v2.0.tar.gz
tar -zxvf v2.0.tar.gz
cd RTL8188-hostapd-2.0/hostapd
sudo make
sudo make install
```

NOTE: This last step will move the created hostapd binary to /usr/local/bin, add a startup script and create a configuration file in /etc/hostapd/hostapd.conf. Edit this configuration file and start the hostapd service

```
sudo service hostapd restart
```

Change the following things in the following files

Add the following lines into the file **/etc/default/hostapd**

```
RUN_DAEMON="yes"
DAEMON_CONF="/etc/hostapd/hostapd.conf";
DAEMON_OPTS="-dd -t";
```

Add the path if not added in the file **/etc/init.d/hostapd**

```
DAEMON_SBIN=/usr/local/bin/hostapd
DAEMON_CONF=/etc/hostapd/hostapd.conf # add path
```

Since the WAP and WAP2 network do not work edit the following file **/etc/hostapd/hostapd.conf** as follows

```
# Basic configuration

interface=wlan0
ssid=wifi
channel=1
#bridge=br0

# Static WEP key configuration

auth_algs=2
wep_default_key=0
wep_key0=123456789a

# WPA and WPA2 configuration

#macaddr_acl=0
#auth_algs=1
#ignore_broadcast_ssid=0
#wpa=1
#wpa_passphrase=12345678
#wpa_key_mgmt=WPA-PSK
#wpa_pairwise=TKIP
#rsn_pairwise=CCMP

# Hardware configuration

driver=rtl871xdrv
ieee80211n=1
hw_mode=g
device_name=RTL8192CU
manufacturer=Realtek
```

To test the hostapd you can use the command
hostapd –dd /etc/hostapd/hostapd.conf

You can manually switch to hostapd by
ifup wlan0 and *ifdown wlan0*

STEP 4: SETTING UP THE DNSMASQ FOR THE IP DISTRIBUTION

Excecute the following commands

```
sudo apt-get -y install dnsmasq
cd /etc/
sudo mv dnsmasq.conf dnsmasq.conf.orig
sudo touch dnsmasq.conf
```

Contents of the dnsmasq.conf

```
interface=wlan0
dhcp-range=192.168.8.20,192.168.8.254,255.255.255.0,12h
```

```
reboot
```

****

```
service dnsmasq start
service hostapd start
```

CHECK THE STATUS OF BOTH HOSTAPD AND DNSMASQ BY

```
service dnsmasq status
service hostapd status
```

SOCKET PROGRAMMING CODE ON C:

```
# include<stdio.h>
# include<netinet/in.h>
# include<sys/types.h>
# include<arpa/inet.h>
# include<string.h>
# include<sys/socket.h>
# include<math.h>
# include<string.h>

int port=8888;

int main()
{
    FILE *fp;
    char passopen[]="12345:YES";
    char passclose[]="12345:N0";
    fp=fopen("LOCK_STATUS","w");
    struct sockaddr_in servaddr,cliaddr;
    int servid,clid;
    int status;
    char buffer[1024];
    int ct=0;
    servid=socket(AF_INET,SOCK_STREAM,0);
    if(servid>=0)
        printf("Socket Created");
    else
        printf("Socket Not Created");
    memset(buffer,0,sizeof(buffer));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
    servaddr.sin_port=htons(port);

    status=bind(servid,(struct sockaddr*)&servaddr,sizeof(servaddr));
    if(status>=0)
        printf("\nBinded\n");
    status=listen(servid,5);
    if(status>=0)
        printf("Listening...");
    int siz = sizeof cliaddr;
    while(ct<5){
        ct++;
        clid = accept(servid,(struct sockaddr*)&cliaddr,&siz);
        recv(clid,buffer,sizeof(buffer),0);
        printf("trial : %d : %s ",ct,buffer);
        close(clid);
        if(strcmp(buffer,passopen)==0 || strcmp(buffer,passclose)==0)
        {
                fprintf(fp,"%s\n",buffer);
                fclose(fp);
                break;
        }
    }
    close(servid);
}
```

LASER-LDR TEST CODE:

```
import Adafruit_BBIO.GPIO as GPIO

GPIO.setup("P9_11", GPIO.IN)
GPIO.setup("P9_12", GPIO.IN)
GPIO.setup("P9_13", GPIO.IN)
GPIO.setup("P8_9", GPIO.IN)
GPIO.setup("P8_7", GPIO.IN)
GPIO.setup("P9_24", GPIO.IN)
GPIO.setup("P9_26", GPIO.IN)
GPIO.setup("P8_8", GPIO.IN)
while True:
        print GPIO.input("P9_11"),GPIO.input("P8_9"),GPIO.input("P9_13"),GPIO.in
put("P9_12"),GPIO.input("P8_8"),GPIO.input("P9_26"),GPIO.input("P8_7"),GPIO.inpu
t("P9_24")
```

LASER LDR UNLOCK CODE:

```
import Adafruit_BBIO.GPIO as GPIO
import numpy as np

c0 = "P9_11"
c1 = "P8_9"
c2 = "P9_13"
c3 = "P9_12"

r0 = "P9_24"
r1 = "P8_7"
r2 = "P9_26"
r3 = "P8_8"

led1 = "P8_10"

f=0;
b = np.zeros((4,4))

GPIO.setup(c0,GPIO.IN)
GPIO.setup(c1,GPIO.IN)
GPIO.setup(c2,GPIO.IN)
GPIO.setup(c3,GPIO.IN)
GPIO.setup(r0,GPIO.IN)
GPIO.setup(r1,GPIO.IN)
GPIO.setup(r2,GPIO.IN)
GPIO.setup(r3,GPIO.IN)

GPIO.setup(led1,GPIO.OUT)

while((GPIO.input(c1)==0) or (GPIO.input(r1)==0)):
    a=1
if((GPIO.input(c1)==1) and (GPIO.input(r1)==1)):
    f=f+1
    b[1][1]=1
    print b
    print f
    while((GPIO.input(c2)==0) or (GPIO.input(r2)==0)):
        a=1
    if (GPIO.input(c2)==1 and GPIO.input(r2)==1 and b[2][2]==0):
        f=f+1
        b[2][2]=1
        print b
        print f
        while((GPIO.input(c3)==0) or (GPIO.input(r3)==0)):
            a=1
        if (GPIO.input(c3)==1 and GPIO.input(r3)==1 and b[3][3]==0):
            f=f+1
            b[3][3]=1
            print b
            print f
            while((GPIO.input(c3)==0) or (GPIO.input(r2)==0)):
                a=1
            if (GPIO.input(c3)==1 and GPIO.input(r2)==1 and b[3][2]==0):
                f=f+1
                b[3][2]=1
                print b
                print f
                print "unlock"
                while(1):
                    GPIO.output(led1,GPIO.HIGH)
```

ANDROID UNLOCK CODE:

```
import Adafruit_BBIO.GPIO as GPIO

GPIO.setup("P8_10",GPIO.OUT)

f=open("LOCK_STATUS","r")
x=f.readlines(0)
y=[]
y.append('12345:YES\n')
z=[]
z=z.append('12345:NO\n')
print x
print y
print z
while(True):
    if(x[0]==y[0]):
        GPIO.output("P8_10",GPIO.LOW)
    elif(x[0]==z[0]):
        GPIO.output("P8_10",GPIO.HIGH)
```

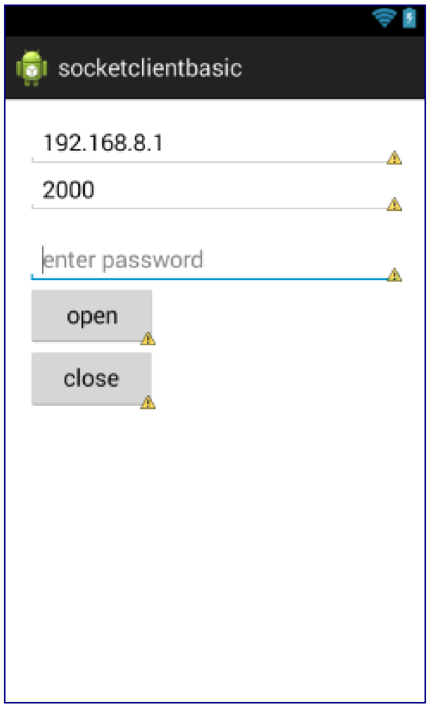
SOFT APP CODE

JAVA FILE

```
package com.example.socketclientbasic;

import java.io.BufferedWriter;
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;

import android.location.Address;
import android.os.AsyncTask;
import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {

       TextView textResponse;
       EditText editTextAddress, editTextPort, mypassword;
       Button buttonConnect, buttonClear;
       private static Socket socket;
       String host;
       InetAddress address;
       int port;
       @Override
       protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editTextAddress = (EditText)findViewById(R.id.address);
        editTextPort = (EditText)findViewById(R.id.port);
        textResponse = (TextView)findViewById(R.id.response);
        mypassword = (EditText) findViewById(R.id.mypassword);
       }
       public class MyClientTask extends AsyncTask<Void, Void, Void> {

              String dstAddress;
              int dstPort;
              String Message;
              String response = "";

              MyClientTask(String addr, int port,String mymessage){
               dstAddress = addr;
               dstPort = port;
               Message = mymessage;
              }

              @Override
              protected Void doInBackground(Void... arg0) {

               Socket socket = null;
```

```
            try {
             socket = new Socket(dstAddress, dstPort);

             OutputStream os = socket.getOutputStream();
            OutputStreamWriter osw = new OutputStreamWriter(os);
            BufferedWriter bw = new BufferedWriter(osw);

            bw.write(Message);
            bw.flush();
            Toast.makeText(getApplicationContext(), Message,
Toast.LENGTH_LONG).show();
           // textResponse.setText("Message sent to the server : "+"hello");

        } catch (UnknownHostException e) {
             // TODO Auto-generated catch block
             e.printStackTrace();
             response = "UnknownHostException: " + e.toString();
            } catch (IOException e) {
             // TODO Auto-generated catch block
             e.printStackTrace();
             response = "IOException: " + e.toString();
            }finally{
             if(socket != null){
              try {
               socket.close();
              } catch (IOException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
              }
             }
            }
            return null;
           }

           @Override
           protected void onPostExecute(Void result) {
            textResponse.setText(response);
            super.onPostExecute(result);
           }

          }
      private void makeConnection(){
            MyClientTask myClientTask = new MyClientTask(
                        editTextAddress.getText().toString(),
                        Integer.parseInt(editTextPort.getText().toString()),"heelo");
                      myClientTask.execute();

      }
      private void sendData(String sendMessage) throws IOException{
            //Send the message to the server
        OutputStream os = socket.getOutputStream();
        OutputStreamWriter osw = new OutputStreamWriter(os);
        BufferedWriter bw = new BufferedWriter(osw);

        bw.write(sendMessage);
        bw.flush();
        textResponse.setText("Message sent to the server : "+sendMessage);
      }
      public void connectSystem(View v){
            makeConnection();
      }
      public void connectClose(View v) throws IOException{
            socket.close();
      }
      public void clearData(View v){
            editTextAddress.setText("");
             editTextPort.setText("");
             mypassword.setText("");
      }
      public void sendYes(View v) throws IOException{
            String data = mypassword.getText().toString() + ":YES";
            MyClientTask myClientTask = new MyClientTask(
                        editTextAddress.getText().toString(),
                        Integer.parseInt(editTextPort.getText().toString()),data);
                      myClientTask.execute();
      }
      public void sendNo(View v) throws IOException{
            String data = mypassword.getText().toString() + ":NO";
            MyClientTask myClientTask = new MyClientTask(
                        editTextAddress.getText().toString(),
                        Integer.parseInt(editTextPort.getText().toString()),data);
                      myClientTask.execute();

      }

     }
```

XML FILE


socketclientbasic

192.168.8.1

2000

enter password

Figure 15 Android App

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <EditText
```

```xml
        android:id="@+id/address"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="dstAddress"
        android:text="192.168.8.1" />

    <EditText
        android:id="@+id/port"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="dstPort"
        android:text="2000" />

    <TextView
        android:id="@+id/response"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <EditText
        android:id="@+id/mypassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="enter password" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:text="open"
        android:onClick="sendYes" />

    <Button
        android:id="@+id/button2"
        android:layout_width="99dp"
        android:layout_height="wrap_content"
        android:text="close"
        android:onClick="sendNo" />

</LinearLayout>
```

APPLICATION ORIENTED LEARNING

The following Project is a prototype of the future generation security systems. Further the scope of the project can be enhanced by adding the following features to the project:

- Adding a guest mode to the LOCK which allows permission to a desirable person / guest to unlock the lock after proper authentication via a camera enabling IoT and sending guest pictures to the owner and asking him to authenticate the guest
- Further the gesture based pattern unlock could be made a 3D gesture record and unlock system where a door unlocks when made a static gesture

Operating Systems Concepts Learned:

- Socket Programming in JAVA and C
- Understanding Wireless network and Types
- Understanding Wireless Encryption Standards: WEP WPSK WPSK2
- Setting up Wireless Drivers on Devices
- Services used in Setting up a Access Point Network on a real time Linux based OS
- Configuring Interfaces and networks
- Understanding Net- mask and IP-configuration on a Linux OS
- Practically understanding the concepts of UDP and TCP through Socket Programming
- How does a device assign IP's to other devices – Services involved
- Understanding SSH connection formed via a BBB and computer forming a local connection

CONCLUSION

The Project was successfully completed except for the APP crashing problems were faced as the client app sent the PASSKEY data string.

REFERENCES

https://ariandy1.wordpress.com/2013/04/18/wifi-access-point-with-tp-link-tl-wn722n-on-ubuntu-12-04/

https://groups.google.com/forum/#!category-topic/beagleboard/bMQTXi1Jt80

https://seravo.fi/2014/create-wireless-access-point-hostapd

http://elinux.org/RPI-Wireless-Hotspot

https://jenssegers.com/43/Realtek-RTL8188-based-access-point-on-Raspberry-Pi

http://linux.die.net/man/8/ifconfig

http://w1.fi/gitweb/gitweb.cgi?p=hostap.git;a=blob_plain;f=hostapd/hostapd.conf

https://wiki.gentoo.org/wiki/Hostapd

https://fleshandmachines.wordpress.com/2012/10/04/wifi-acces-point-on-beaglebone-with-dhcp/

https://github.com/RahulMahale/Talks-and-workshops/blob/master/DebConf/steps.md

http://www.cyberciti.biz/faq/howto-ubuntu-debian-squeeze-dhcp-server-setup-tutorial/

http://lists.shmoo.com/pipermail/hostap/2009-June/019919.html

http://www.howtogeek.com/167783/htg-explains-the-difference-between-wep-wpa-and-wpa2-wireless-encryption-and-why-it-matters/

http://beej.us/guide/bgnet/

https://beagleboard.org/black

IRF44N Data-Sheet

BC547 Data-Sheet