SMART SECURITY SYSTEM

Submitted by

ANGAD BORAKLAR

HASSAM KHAN WAZIR

in partial fulfilment for the course

of

ADVANCED MECHATRONICS

MECHATRONICS AND ROBOTICS ENGINEERING

TANDON SCHOOL OF ENGINEERING

11201, 6 MetroTech Center, Brooklyn, NY 11201

Abstract

Security is a major concern in today's world and home security is no different. With the advent of modern technology, many high-end solutions are available such as electronic security systems, very high strength doors and deadbolts, and 24/7 monitoring using sophisticated cameras. However, one thing in common in all of the security systems mentioned above is the cost. A basic security lock costs more than a hundred dollars and the price only goes up as the security systems gets more features.

The concept of Internet of Things (IoT) is currently very popular and several powerful microcontroller and System on Chip (SoC) boards are available at a very low cost. This gives immense power into the hands of engineers as well as enthusiasts that can build their own solutions to existing problems instead of relying on big manufacturers that usually sell the same solutions for exorbitant prices. With access to microcontrollers, cameras, and the ubiquity of smartphones, it is within the realm of possibility to create a security system that will suit the need of a common person without costing them a lot of money.

1. Introduction

Security is a major concern in today's world and home security is no different. With the advent of modern technology, many high-end solutions are available such as electronic security systems, very high strength doors and deadbolts, and 24/7 monitoring using sophisticated cameras. However, one thing in common in all of the security systems mentioned above is the cost. A basic security lock costs more than a hundred dollars and the price only goes up as the security systems gets more features.

The concept of Internet of Things (IoT) is currently very popular and several powerful microcontroller and System on Chip (SoC) boards are available at a very low cost. This gives immense power into the hands of engineers as well as enthusiasts that can build their own solutions to existing problems instead of relying on big manufacturers that usually sell the same solutions for exorbitant prices. With access to microcontrollers, cameras, and the ubiquity of smartphones, it is within the realm of possibility to create a security system that will suit the need of a common person without costing them a lot of money.

2. Hardware

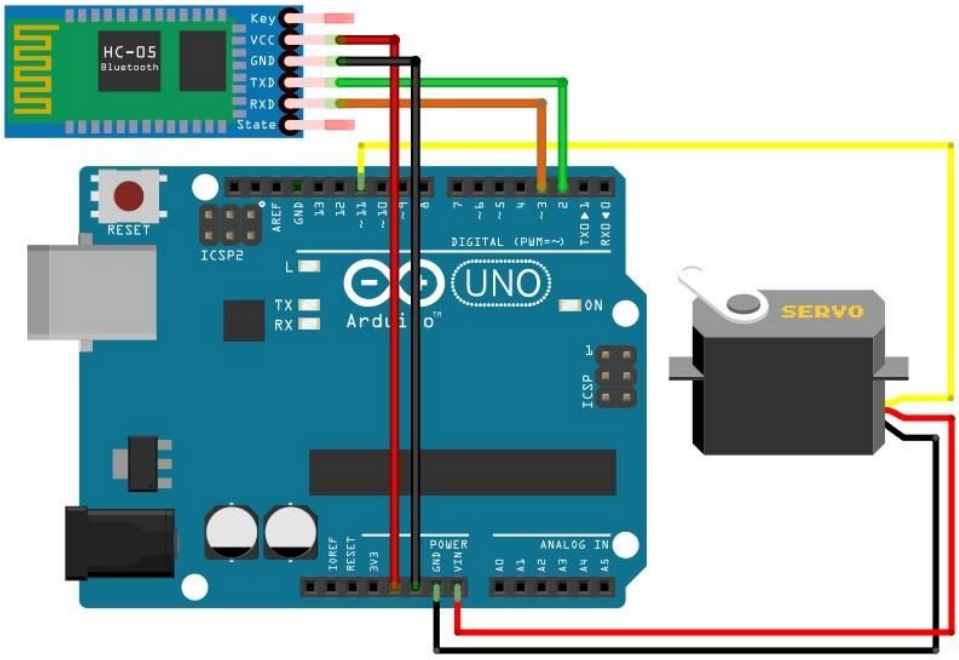
The hardware used for this project is as follows:

- Arduino Uno Rev 3
- HC-05 Bluetooth module
- Raspberry Pi 3
- Pi camera
- Adafruit Fona 808 Cellular + GPS Breakout board.
- Servo Motor - TowerPro MG995
- 6V Power Supply

Electronic Circuit

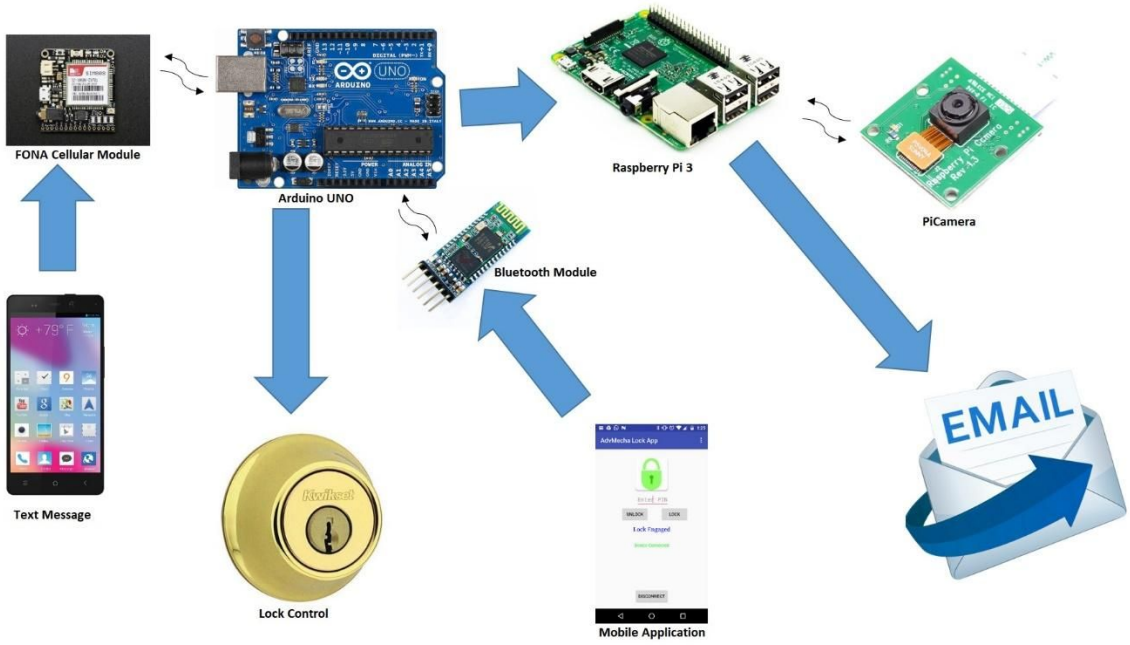
The lock is connected to a clamp attached to a servo motor that is controlled using an Arduino UNO board. The servo stays at a neutral 90o position and upon receiving a locking or unlocking command, it rotates to 180o or 0o respectively, before returning to the neutral position. The electronic circuit for the lock control is shown below.

Figure 2.1 Electronic circuit for the lock control.



The raspberry pi and the Adafruit Fona 808 Cellular board are also connected to the Arduino board via a digital pin and the RX/TX pins respectively. The relationship of all the components with each other is shown in the figure below.

Figure 2.2 Overview of the Hardware components used in the project.



The mobile application connects to the Bluetooth module which sends the control commands to the Arduino from the user. The Arduino, then, controls the servo motor that eventually engages or disengages the lock. On the other hand, the user sends a text message to the FONA cellular module which communicates with the Arduino in order to tell the raspberry pi to capture an

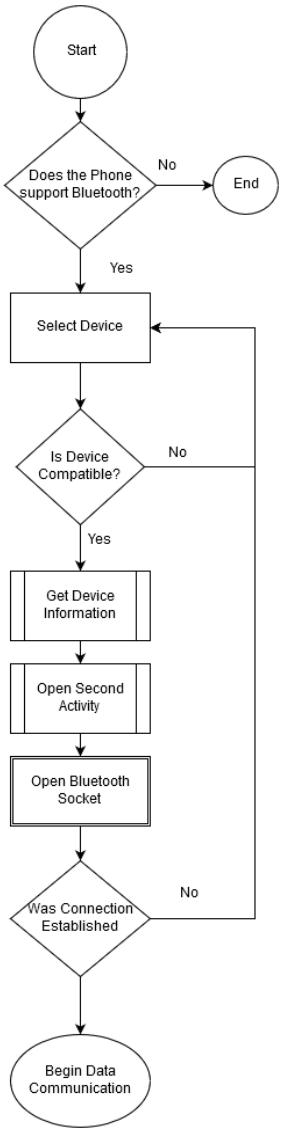
image and send it to the user via email. The cellular module also texts the user back with a confirmation message.

3. Software

The software comprises of two parts. The first part is the Android application development whereas the second part is capturing and sending images to email address.

3.1 Application Development

A mobile application was developed to control the electronic circuitry of the lock. The application was developed on the Android platform using the Android Studio 2.3.2 IDE.



Upon application startup, the software runs an internal query determining if the device can support Bluetooth. If the outcome is positive, the software checks if Bluetooth on the device is

currently turned on, and prompts the user to do so if it isn't. The user, then, chooses from a list of available paired devices on their device and will see the locking mechanism listed there if it has already been paired with the mobile device. When the lock is selected from the list, the software gets the address of the lock from the Bluetooth adapter and opens a Bluetooth Socket to create a connection, through which communication can be initiated between the lock and the smart device. The Bluetooth adapter stops searching for other devices and finally, the Bluetooth Socket is connected. Now the smart device is connected with the lock via Bluetooth and the current Activity (screen of the smart device) opens up a new Activity. Figure 3.2 shows the first Activity and a list of paired Bluetooth devices.

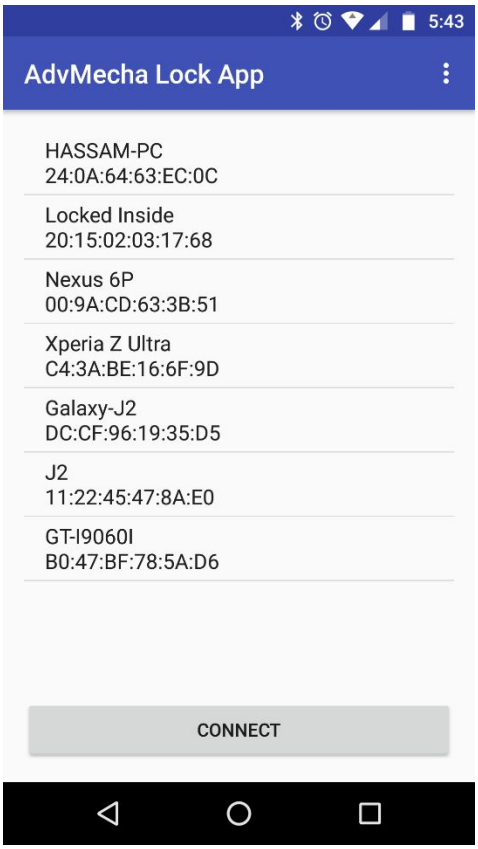
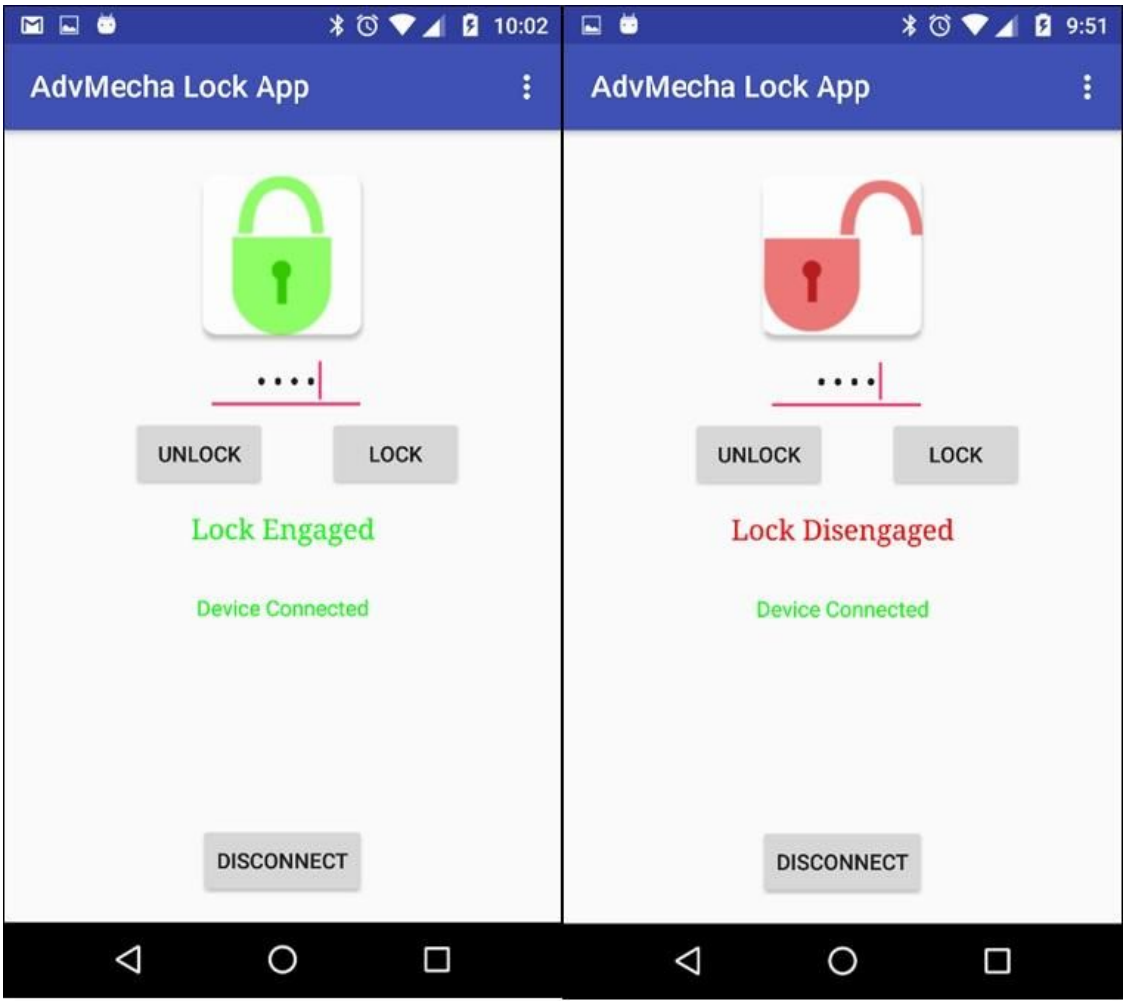


Figure 3.2 The First Activity, showing a list of paired devices.

In the new Activity, the user has the option to enter the password for the lock and then press one of the two buttons to either engage of disengage the lock. The status of the lock is also displayed under the buttons along with the status of the connection between the lock and the mobile device. The user also has the option to disconnect from the lock, which will sever the connection

between the mobile device and the lock and the user will need to initiate the connection process from the beginning. Figure 3.3 shows the second Activity of the application, responsible for the



lock controls and disconnecting the mobile device from the lock. One safety mechanism that has bene integrated into the application is that in case the application disconnects from the lock, the lock will automatically engage.

In this new Activity, the user has the option to disconnect from the lock, which will sever the connection between the smart device and the lock and the user will need to start over from the beginning. Another option is to enter the password for the lock and press one of the two button that will either engage the locking mechanism or disengage it. The status of the locking mechanism will be displayed on the screen along with the status of the connection and a visual

indication of the current state of the locking mechanism (whether it is engaged or disengaged). Figure 4 shows the second Activity as well as the two states of the locking mechanism. At the bottom of the screen, the “Disconnect” button serves the purpose of closing the Bluetooth Socket which, in turn, closes the connection between the smart device and the locking mechanism. However, before the connection is severed, the smart device will send a final command to the locking mechanism and the lock will become engaged. This is done for safety purposes so that disconnecting the smart device automatically engages the locking mechanism, serving as an auto locking feature.

3.2 Capturing and Sending Images to Email Address

There are several steps involved in capturing and sending images to an email address. A brief explanation of each step is given below.

Capturing images to email address

To capture and send the image from the Raspberry Pi microcontroller to an email address is one of the major aspects of this project. The procedure to do so can be split into three parts as described below.

Raspberry Pi Camera

In this project, a Raspberry Pi camera (PiCam) is used to capture the image of a person standing in front of the device. Based on the commands from the Arduino, the PiCam takes a picture and saves it locally on the SD card of the microcontroller. After this, the microcontroller extracts the captured image from the folder and sends it as an attachment to the authorized email address over Wi-Fi.

Sending Images

Once the image is captured and is ready to be sent as an email attachment, a python script is used to connect to a remote Google server. Google gives the option of accessing their remote email

servers for free with the correct credentials to identify each user. On this Google server, an email id is set as the sender of the message and correct authentication process is initiated. The user can pre-authorize his/her email address along with the correct password using the python script. Once the script is run, the remote Gmail server is accessed using valid email (and password), a 'log-in' process is initiated virtually. The script also provides options to change the body of the email, the subject line as well as the number of recipients. Once a secure connection is established and valid user is logged in, the script pulls the required image file from the SD card and attaches it to the email structure. After this, it sends the email over Wi-Fi to the designated user address and the recipient can view this as an email attachment.

Python code

In this project, we have used an SMTP server to send the image as an email attachment over the internet. An SMTP server is the machine that takes care of the whole email delivery process: that's why to send your messages with an email client or software you need first of all to configure the correct SMTP settings – in particular, the right SMTP address you're using. (For instance, Gmail's is smtp.gmail.com). Remember anyway that normal servers generally come with strict limits in terms of how many emails you can send and how many addresses you can handle per day. So if you're planning to send bulk emails you should definitely switch to a professional SMTP server that will allow you to manage unlimited messages and guarantee the highest deliverability.

For our project, we are using the most basic version of Gmail SMTP to test the results of our project. For testing purposes, we have also created a test email address on Gmail called 'advmechspring2017@gmail.com' to be used as the sender address of the email. Using the above email ID, we will send the image to the recipient's email ID ('angadboralkar@gmail.com') over Wi-Fi. The PiCamera library is imported into the python script to access hardware and software functionality of the PiCam.

4. Conclusion

The project successfully demonstrates that using a simple design and low cost electronic components, common everyday problems can be solved very easily. The project itself took very less time to make, and the majority of time was spent learning the Android API, the Bluetooth sensor, and sending data automatically via email and coming up with a simple design that works flawlessly and looks elegant. Working on this project provided a better insight into the world of electronics and the role of microcontrollers and sensors, since in a single project, out team was exposed to designing, 3d modelling, 3d printing, microcontrollers, sensors and actuators, data acquisition, mobile application development and hardware-software integration.

APPENDIX

Code

PYTHON

```
import smtplib
from email.mime.multipart import
MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
import os
import picamera
import time

os.chdir('/home/pi/Desktop/pics')

camera = picamera.PiCamera()

camera.start_preview()
time.sleep(3)
camera.capture("test.png")
camera.stop_preview()

fromaddr = "advmechspring2017@gmail.com"
toaddr = "angadboralkar@gmail.com"

msg = MIMEMultipart()

msg['From'] = fromaddr
msg['To'] = toaddr
msg['Subject'] = "Attachment Test"

body = "See the attachment"

msg.attach(MIMEText(body, 'plain'))

filename = "test.png"
attachment =
open("/home/pi/Desktop/pics/test.png", "rb")

part = MIMEBase('application', 'octet-stream')
part.set_payload((attachment).read())
encoders.encode_base64(part)
part.add_header('Content-Disposition',
"attachment; filename= %s" % filename)

msg.attach(part)

server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login(fromaddr, "angadhassam")
text = msg.as_string()
server.sendmail(fromaddr, toaddr, text)
server.quit()
```

ARDUINO

Lock Code:

```
#include <Servo.h>
#include <SoftwareSerial.h>
byte myPass[4];
String myString;
boolean isUnlock = false;

#define servo_pin 6                    // Set
servo_pin Pin

SoftwareSerial myBluetooth(2, 3); // RX, TX
Servo daServo;


void setup() {
 myBluetooth.begin(9600);
 //Arduino Pin Configuration
 pinMode(servo_pin, OUTPUT);

 servo(90);              // Make sure servo is in
mean position

 //Protocol Configuration
```

```
  Serial.begin(9600);       // Initialize serial
communications with PC
}


////////////////////////////////////// Main
Loop ////////////////////////////////
void loop () {

   if (myBluetooth.available() > 0)
   {myString = "";}

   while(myBluetooth.available() > 0)
   {
     command = ((byte)myBluetooth.read());

     if(command == ':')
     {
       break;
     }
     else
     {
       myString += command;
     }
     delay(1);
   }
   //Serial.println(myString);
   if(myString == "1")// && isUnlock == false)
   {
     myBluetooth.write("1");
       Unlock();
       Serial.println(myString);
   }
   else if(myString == "0")// && isUnlock ==
true)
   {
       myBluetooth.write("0");
       Lock();
      Serial.println(myString);
   }
   myString = "";
   successRead = getID();            // sets
successRead to 1 when we get read from reader
otherwise 0

////////////////////Servo
Method////////////////////////////////////
 void Lock()
 {
 if (digitalRead(limitswitch) ==
LOW||isUnlock==false)
  {
    daServo.attach(servo_pin);
    servo(0);            // tell servo to go to position
in variable 'pos'
    delay(1000);
    servo(90);
    delay(1000);
    isUnlock=false;
    daServo.detach();
  }
 }
  void Unlock()
  {
    {
   daServo.attach(servo_pin);
    servo(180);            // tell servo to go to
position in variable 'pos'
    delay(1000);
    servo(90);
    delay(1000);
    isUnlock=true;
    daServo.detach();
  }
  }

void servo(int datPos)
{
  daServo.write(datPos);
  delay(10);
}
```

FONA Code:

```
#include "Adafruit_FONA.h"

#define FONA_RX 2
#define FONA_TX 3
#define FONA_RST 4

// this is a large buffer for replies
char replybuffer[255];

// We default to using software serial. If you
want to use hardware serial
```

```
// (because softserial isnt supported) comment
out the following three lines
// and uncomment the HardwareSerial line
#include <SoftwareSerial.h>
SoftwareSerial fonaSS =
SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

// Hardware serial is also possible!
//  HardwareSerial *fonaSerial = &Serial1;

Adafruit_FONA fona =
Adafruit_FONA(FONA_RST);

uint8_t readline(char *buff, uint8_t maxbuff,
uint16_t timeout = 0);

void setup() {
 pinMode(10, OUTPUT);
 digitalWrite(10, LOW);
 while (!Serial);

 Serial.begin(115200);
 Serial.println(F("FONA SMS caller ID test"));
 Serial.println(F("Initializing....(May take 3
seconds)"));

 // make it slow so its easy to read!
 fonaSerial->begin(4800);
 if (! fona.begin(*fonaSerial)) {
  Serial.println(F("Couldn't find FONA"));
  while(1);
 }
 Serial.println(F("FONA is OK"));

 // Print SIM card IMEI number.
 char imei[16] = {0}; // MUST use a 16 character
buffer for IMEI!
 uint8_t imeiLen = fona.getIMEI(imei);
 if (imeiLen > 0) {
  Serial.print("SIM card IMEI: ");
Serial.println(imei);
 }

 Serial.println("FONA Ready");
}

char fonaInBuffer[64];         //for notifications
from the FONA

void loop() {

 char* bufPtr = fonaInBuffer;    //handy buffer
pointer
 digitalWrite(10, LOW);
 if (fona.available())      //any data available
from the FONA?
 {
  int slot = 0;           //this will be the slot
number of the SMS
  int charCount = 0;
  //Read the notification into fonaInBuffer
  do  {
   *bufPtr = fona.read();
   Serial.write(*bufPtr);
   delay(1);
  } while ((*bufPtr++ != '\n') &&
(fona.available()) && (++charCount <
(sizeof(fonaInBuffer)-1)));

  //Add a terminal NULL to the notification
string
  *bufPtr = 0;

  //Scan the notification string for an SMS
received notification.
  //  If it's an SMS message, we'll get the slot
number in 'slot'
  if (1 == sscanf(fonaInBuffer, "+CMTI:
\"SM\",%d", &slot)) {
    Serial.print("slot: "); Serial.println(slot);

    char callerIDbuffer[32];  //we'll store the
SMS sender number in here
    char replybuffer[255];
    //char checksms[255] = "Open"
    // Retrieve SMS sender address/phone
number.
    if (! fona.getSMSSender(slot, callerIDbuffer,
31)) {
     Serial.println("Didn't find SMS message in
slot!");
    }
    Serial.print(F("FROM: "));
Serial.println(callerIDbuffer);
```

```
    uint16_t smslen;
    if (! fona.readSMS(slot, replybuffer, 250,
&smslen)){
      Serial.println("Cannot Read the SMS!");
    }

    Serial.println(replybuffer);

    //Send back an automatic response
    Serial.println("Sending reponse...");

    if (strcmp(replybuffer, "Open") == 0){
      Serial.println("Well, it's a match!");
      fona.sendSMS(callerIDbuffer, "Door is now
OPEN! Check your email to see who's at the
door.");
      digitalWrite(10, HIGH);
      delay(500);
      //digitalWrite(10, LOW);
    }else{
      Serial.println("Nope, not a match!");
      fona.sendSMS(callerIDbuffer, "This is an
incorrect passphrase. Door is now CLOSED!");
    }

    if (fona.deleteSMS(slot)) {
      Serial.println(F("OK!"));
    } else {
      Serial.println(F("Couldn't delete"));
    }
  }
 }
}
```

APPLICATION CODE

```
package hassamwazirnyu.superduperlockapp;
import android.content.Intent;
import android.os.Bundle;
import
android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Set;

public class MainActivity extends
AppCompatActivity {

  //widgets
  Button btnPaired;
  ListView MainActivity;
  //Bluetooth
  private BluetoothAdapter myBluetooth = null;
  private Set<BluetoothDevice> pairedDevices;
  public static String EXTRA_ADDRESS =
"device_address";

  @Override
  protected void onCreate(Bundle
savedInstanceState)
  {
    super.onCreate(savedInstanceState);

setContentView(R.layout.main_activity_layout);

    //Calling widgets
    btnPaired =
(Button)findViewById(R.id.bt_connect);
    MainActivity =
(ListView)findViewById(R.id.lv_devices);

    //if the device has bluetooth
    myBluetooth =
BluetoothAdapter.getDefaultAdapter();

    if(myBluetooth == null)
    {
      //Show a mensag. that the device has no
bluetooth adapter
      Toast.makeText(getApplicationContext(),
"Bluetooth Device Not Available",
Toast.LENGTH_LONG).show();
```

```
        //finish apk
        finish();
    }
    else if(!myBluetooth.isEnabled())
    {
        //Ask to the user turn the bluetooth on
        Intent turnBTon = new
Intent(BluetoothAdapter.ACTION_REQUEST_EN
ABLE);
        startActivityForResult(turnBTon,1);
    }

    btnPaired.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v)
        {
            pairedDevicesList();
        }
    });

  }

  private void pairedDevicesList()
  {
    pairedDevices =
myBluetooth.getBondedDevices();
    ArrayList list = new ArrayList();

    if (pairedDevices.size()>0)
    {
        for(BluetoothDevice bt : pairedDevices)
        {
            list.add(bt.getName() + "\n" +
bt.getAddress()); //Get the device's name and
the address
        }
    }
    else
    {
        Toast.makeText(getApplicationContext(),
"No Paired Bluetooth Devices Found.",
Toast.LENGTH_LONG).show();
    }

    final ArrayAdapter adapter = new
ArrayAdapter(this,android.R.layout.simple_list_i
tem_1, list);
    MainActivity.setAdapter(adapter);

MainActivity.setOnItemClickListener(myListClick
Listener); //Method called when the device
from the list is clicked

  }

  private AdapterView.OnItemClickListener
myListClickListener = new
AdapterView.OnItemClickListener()
  {
    public void onItemClick (AdapterView<?>
av, View v, int arg2, long arg3)
    {
        // Get the device MAC address, the last
17 chars in the View
        String info = ((TextView)
v).getText().toString();
        String address =
info.substring(info.length() - 17);

        // Make an intent to start next activity.
        Intent i = new Intent(MainActivity.this,
LockControl.class);

        //Change the activity.
        i.putExtra(EXTRA_ADDRESS, address);
//this will be received at LockControl (class)
Activity
        startActivity(i);
    }
  };


  @Override
  public boolean onCreateOptionsMenu(Menu
menu)
  {
    // Inflate the menu; this adds items to the
action bar if it is present.

getMenuInflater().inflate(R.menu.menu_main_
activity, menu);
    return true;
  }
```

```
    @Override
    public boolean
onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The
action bar will
        // automatically handle clicks on the
Home/Up button, so long
        // as you specify a parent activity in
AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

```
package hassamwazirnyu.superduperlockapp;

import android.graphics.Color;
import
android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import
android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;

import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.os.AsyncTask;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class LockControl extends
AppCompatActivity {

    int status = 0;
    Button btUnlock, btLock,
btDisconnectControl;
    EditText etPin;
    boolean isCommReceived = false,
isPassReceived = false;
    TextView tvActionResult, tvStatus;
    ImageView ivImage;
    String address = null;
    private ProgressDialog progress;
    BluetoothAdapter btAdapter = null;
    BluetoothSocket btSocket = null ;
    private boolean isBtConnected = false;
    //SPP UUID. Look for it
    static final UUID myUUID =
UUID.fromString("00001101-0000-1000-8000-0
0805F9B34FB");
    private String password = "";
    String myPass = "1234";
    boolean isEngaged = false;

    @Override
    protected void onCreate(Bundle
savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        Intent newint = getIntent();
        address =
newint.getStringExtra(MainActivity.EXTRA_ADD
RESS); //receive the address of the bluetooth
device


setContentView(R.layout.activity_lock_control);

        //call the widgtes
        btUnlock =
(Button)findViewById(R.id.bt_unlock);
        btLock =
(Button)findViewById(R.id.bt_lock);
        btDisconnectControl =
```

```
(Button)findViewById(R.id.bt_disconnect_control);

      tvActionResult = (TextView)findViewById(R.id.tv_action_result);
      tvStatus = (TextView)findViewById(R.id.tv_status);
      etPin = (EditText)findViewById(R.id.et_enter_pin);
      ivImage = (ImageView)findViewById(R.id.iv_image);

      //GetQuery();

     new ConnectBT().execute(); //Call the class to connect

      //commands to be sent to bluetooth
      btUnlock.setOnClickListener(new View.OnClickListener()
      {
         @Override
         public void onClick(View v)
         {
            password = etPin.getText().toString();
            //Log.d("Password: ", password);
            if(password.equals(myPass))
            {
               DisengageLock();     //method to Unlock Door
               //readAcknowledgement();

ivImage.setImageResource(R.mipmap.unlocked);

tvActionResult.setText(R.string.lock_disengaged);

tvActionResult.setTextColor(Color.RED);
            }
            else{
               msg("The password is incorrect");
            }
         }
      });

      btLock.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            password = etPin.getText().toString();
            //Log.d("Password: ", password);
            if(password.equals(myPass) )
            {
               EngageLock();     //method to Lock Door
               //GetQuery();

ivImage.setImageResource(R.mipmap.locked);

tvActionResult.setText(R.string.lock_engaged);

tvActionResult.setTextColor(Color.GREEN);
               //readAcknowledgement();
            }
            else{
               msg("The password is incorrect");
            }
         }
      });


btDisconnectControl.setOnClickListener(new View.OnClickListener()
      {
         @Override
         public void onClick(View v)
         {
            EngageLock();     //method to Lock Door
            Disconnect(); //close connection
         }
      });

   }

   void GetQuery()
   {
      if (btSocket!=null)
      {
         try
         {
            Log.d("PassVal: ", password);

btSocket.getOutputStream().write("5".getBytes());
```

```
        }
        catch (IOException e)
        {
            msg("Error");
        }


        try
        {
            String Ack;
            Ack =
String.valueOf(btSocket.getInputStream().read()
);
            Log.d("Ack val: ", String.valueOf(Ack));

            if(Ack.equals("1"))
            {
                isEngaged = true;
            }
            else{
                isEngaged = false;
            }
        }
        catch (IOException e)
        {
            msg("Error");
        }
    }

  }

  private void Disconnect()
  {

        if (btSocket != null) //If the btSocket is
busy
        {
            try {
                btSocket.close(); //close connection
            } catch (IOException e) {
                msg("Error");
            }
        }

    finish(); //return to the first layout

  }

  private void readAcknowledgement()
  {
    String Ack = "";
    if(btSocket != null)
    {
        try
        {
            Ack =
String.valueOf(btSocket.getInputStream().read()
);
            Log.d("Ack val: ", String.valueOf(Ack));

            switch (Ack) {
                case "0":
                  //
tvActionResult.setText(R.string.lock_engaged);
                  //
tvActionResult.setTextColor(Color.BLUE);
                    break;
                case "1":
                  //
tvActionResult.setText(R.string.lock_disengaged
);

//tvActionResult.setTextColor(Color.GREEN);
                    break;
                case "2":
                    msg("Lock Already Disengaged");
                    break;
                case "3":
                    msg("Lock Already Engaged");
                    break;
                case "4":
                    msg("Password Incorrect");
                    break;
            }
        }
        catch (IOException e)
        {
            msg("Error");
        }
    }
    else{
        msg("Bluetooth Not Connected. Please
Restart the Application");
    }
  }
```

```
   private void sendPassword()
   {
      if (btSocket!=null)
      {
         try
         {
            String a = etPin.getText().toString();
            Log.d("PassVal: ", password);

btSocket.getOutputStream().write(password.ge
tBytes());
         }
         catch (IOException e)
         {
            msg("Error");
         }
      }
   }

   private void EngageLock()
   {
      if (btSocket!=null)
      {
         try
         {
            String data = "0";

btSocket.getOutputStream().write(data.getByte
s());
            Log.d("Data: ", data);
         }
         catch (IOException e)
         {
            msg("Error");
         }
      }
   }

   private void DisengageLock()
   {
      if (btSocket!=null)
      {
         try
         {
            String data = "1";

btSocket.getOutputStream().write(data.getByte
s());
            Log.d("Data: ", data);
         }
         catch (IOException e)
         {
            msg("Error");
         }
      }
   }

   // fast way to call Toast
   private void msg(String s)
   {

Toast.makeText(getApplicationContext(),s,Toast
.LENGTH_SHORT).show();
   }

   @Override
   public boolean onCreateOptionsMenu(Menu
menu) {
      // Inflate the menu; this adds items to the
action bar if it is present.

getMenuInflater().inflate(R.menu.menu_lock_c
ontrol, menu);
      return true;
   }

   @Override
   public boolean
onOptionsItemSelected(MenuItem item) {
      // Handle action bar item clicks here. The
action bar will
      // automatically handle clicks on the
Home/Up button, so long
      // as you specify a parent activity in
AndroidManifest.xml.
      int id = item.getItemId();

      //noinspection SimplifiableIfStatement
      if (id == R.id.action_settings) {
         return true;
      }

      return super.onOptionsItemSelected(item);
   }

   private class ConnectBT extends
```

```
AsyncTask<Void, Void, Void>  // UI thread
   {
      private boolean connSuccess = true; //if it's here, it's almost connected

      @Override
      protected void onPreExecute()
      {
         progress = ProgressDialog.show(LockControl.this, "Connecting...", "Please wait!!!");  //show a progress dialog
      }

      @Override
      protected Void doInBackground(Void... devices) //while the progress dialog is shown, the connection is done in background
      {
         try
         {
            if (btSocket == null || !isBtConnected)
            {
               btAdapter = BluetoothAdapter.getDefaultAdapter();//get the mobile bluetooth device
               BluetoothDevice btDevice = btAdapter.getRemoteDevice(address);//connects to the device's address and checks if it's available
               btSocket = btDevice.createInsecureRfcommSocketToServiceRecord(myUUID);//create a RFCOMM (SPP) connection

               BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
               btSocket.connect();//start connection
            }
         }
         catch (IOException e)
         {
            connSuccess = false;//if the try failed, you can check the exception here
         }
         return null;
      }

      @Override
      protected void onPostExecute(Void result) //after the doInBackground, it checks if everything went fine
      {
         super.onPostExecute(result);

         if (!connSuccess)
         {
            msg("Connection Failed. Is it a compatible device? Try again.");
            tvStatus.setText("Device not Connected");
            tvStatus.setTextColor(Color.BLUE);
            finish();
         }
         else
         {
            msg("Connected.");
            isBtConnected = true;
            tvStatus.setText("Device Connected");
            tvStatus.setTextColor(Color.GREEN);

         }
         progress.dismiss();
      }
   }
}
```