CS 2310: Uploader for TDR System on Android Platform

Mingda Zhang / miz44
mzhang@cs.pitt.edu

1 Project Summary

During the past decades, mobile phones have become a necessary component in modern life. The improved hardware, especially the increased computation power and the various sensors, greatly expanded the functionalities of the modern phones. In this project, we plan to expand the TDR system to Android platform and take the development of *Uploader* as a prototype implementation.

2 System Design

Original TDR system for Chi is developed for PC platform and in this project, we explore the possibility of migrating the system to Android platform. As a prototype, we firstly focus on some of the core components, and we pick the *Android Uploader*. It is worth noting that the *Android Data Collector* component is implemented by another colleague (Zhenjiang Fan) thus this project only address the problems related to *Android Uploader*. The system overview is shown in Fig. 1.

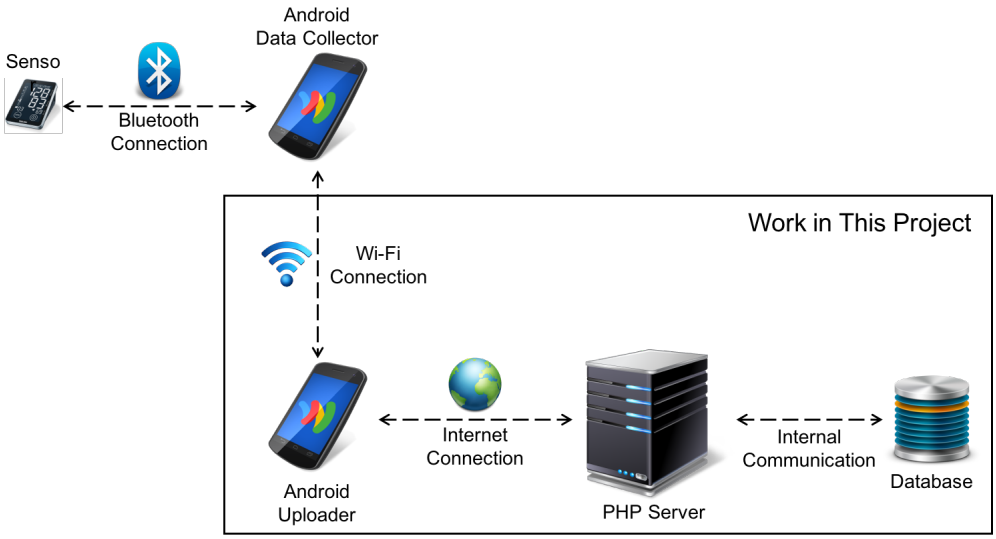


Figure 1: System overview on Android platform

(1) Graphical Interface

In Fig. 2 we show the interface of the Android Uploader. At the top the IP and corresponding port number is displayed to facilitate other components to connect to the *Uploader*. In the middle the display box is

used to show the data received from sensor component. In the bottom there is a status display box to show important information about the the *Uploader* status, such as new connections, closed connections and new received packets.
It is worth noting that we put a *simulate* button at the bottom of the interface, which is used to simulate the connection and packets sending functions.

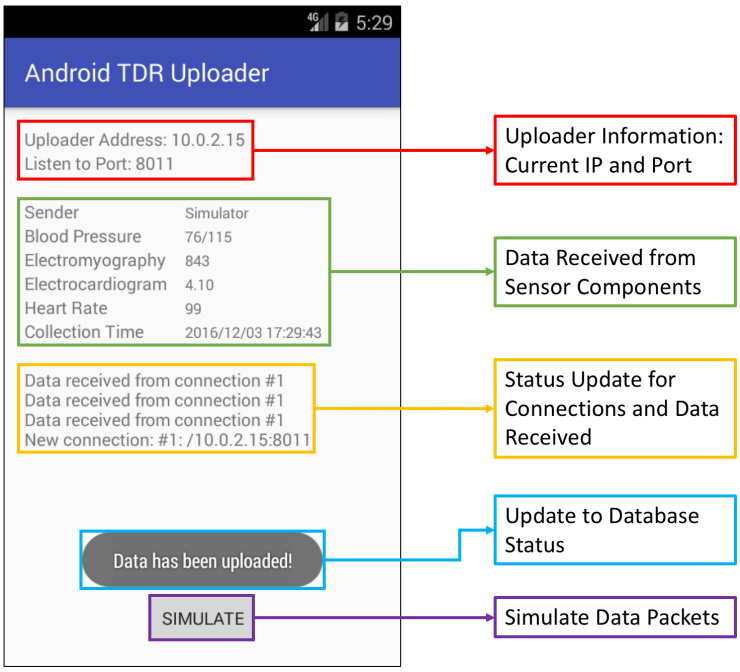


Figure 2: Sample user interface on Android device

(2) Demonstration

In Fig. 3 we show the screenshots of several functions in *Uploader*. When the *Uploader* initiates, it will show the uploader address and the port at the top, as in left figure. Since we also have a simulation function, a simulator client is also initiated and connected to the server. From the screenshot the connection #1 is from the simulator.
It is worth noting that *Uploader* is compatible with existing SIS system. As an example, we can easily debug the system using *PrjRemote* provided in the original TDR SIS system. In the middle figure we show the packets sent from the PrjRemote. Since it is used for test purpose, we do not transfer any real data but just a string `TEST_PURPOSE`. The content is shown in the middle. The screenshot of the *PrjRemote* is shown in Fig. 4
In the right figure, we show the results of receiving packets from simulator. The content of the packet is shown in the middle, for example, we can see the "Blood Pressure" is `78/120`. Since this packet comes from the simulator, the sender is known as "Simulator".
Fig. 5 shows the screenshots from the real database to prove that our uploader did send the information to the database for storage.

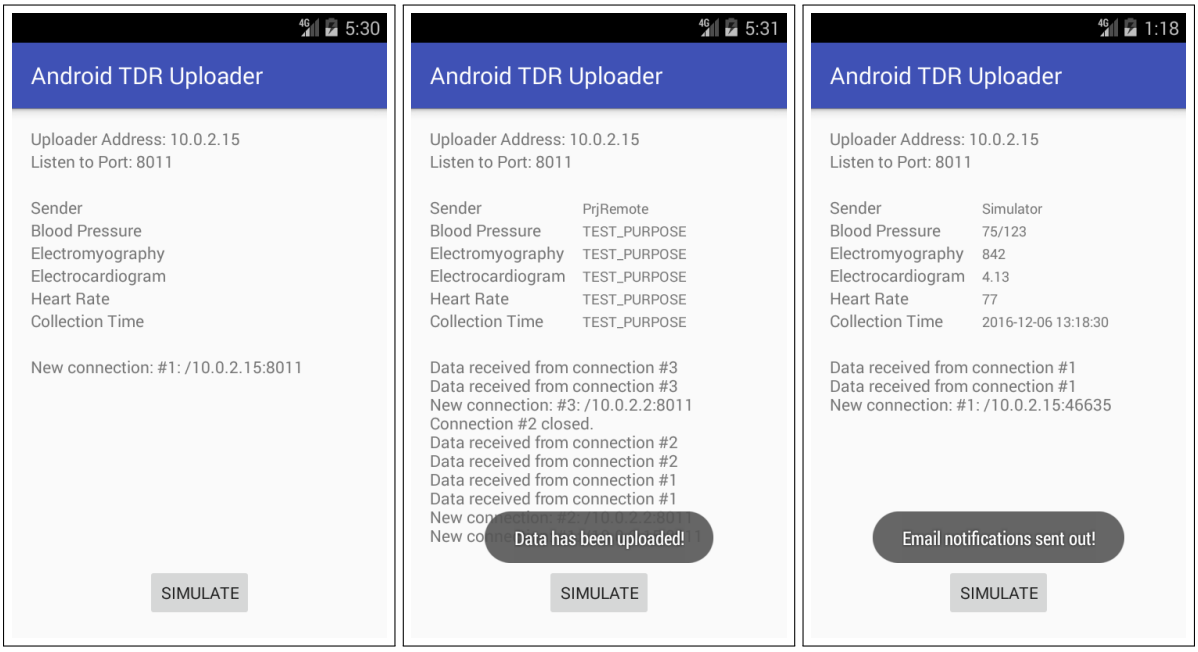


Figure 3: Demonstrations of the Android Uploader

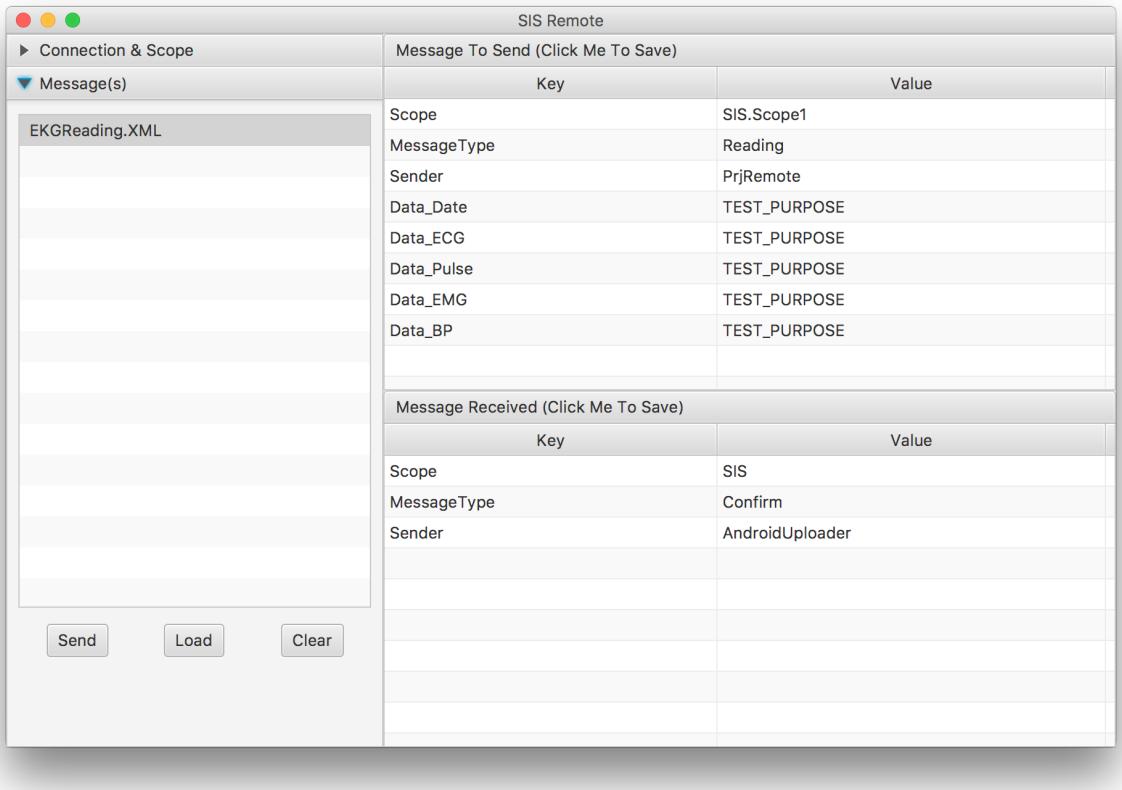
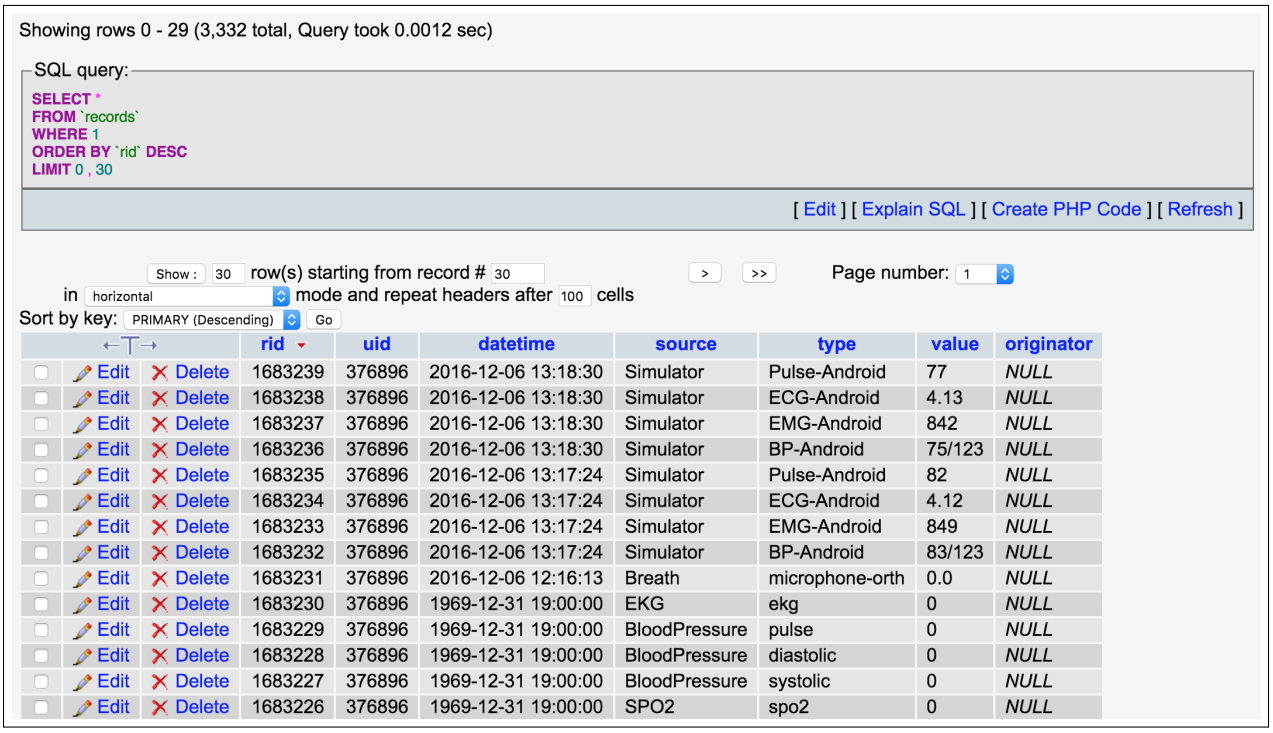


Figure 4: Screenshot of the PrjRemote when connecting with Android Uploader



Showing rows 0 - 29 (3,332 total, Query took 0.0012 sec)

SQL query:

```
SELECT *
FROM `records`
WHERE 1
ORDER BY `rid` DESC
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 30 > >> Page number: 1

in horizontal mode and repeat headers after 100 cells

Sort by key: PRIMARY (Descending) Go

	rid	uid	datetime	source	type	value	originator
Edit Delete	1683239	376896	2016-12-06 13:18:30	Simulator	Pulse-Android	77	NULL
Edit Delete	1683238	376896	2016-12-06 13:18:30	Simulator	ECG-Android	4.13	NULL
Edit Delete	1683237	376896	2016-12-06 13:18:30	Simulator	EMG-Android	842	NULL
Edit Delete	1683236	376896	2016-12-06 13:18:30	Simulator	BP-Android	75/123	NULL
Edit Delete	1683235	376896	2016-12-06 13:17:24	Simulator	Pulse-Android	82	NULL
Edit Delete	1683234	376896	2016-12-06 13:17:24	Simulator	ECG-Android	4.12	NULL
Edit Delete	1683233	376896	2016-12-06 13:17:24	Simulator	EMG-Android	849	NULL
Edit Delete	1683232	376896	2016-12-06 13:17:24	Simulator	BP-Android	83/123	NULL
Edit Delete	1683231	376896	2016-12-06 12:16:13	Breath	microphone-orth	0.0	NULL
Edit Delete	1683230	376896	1969-12-31 19:00:00	EKG	ekg	0	NULL
Edit Delete	1683229	376896	1969-12-31 19:00:00	BloodPressure	pulse	0	NULL
Edit Delete	1683228	376896	1969-12-31 19:00:00	BloodPressure	diastolic	0	NULL
Edit Delete	1683227	376896	1969-12-31 19:00:00	BloodPressure	systolic	0	NULL
Edit Delete	1683226	376896	1969-12-31 19:00:00	SPO2	spo2	0	NULL

Figure 5: Demonstrations of the Android Uploader

(3) Responses

The uploading function contains two parts: the first one is to transfer data to remote SIS database; the second is to send emails to specified user to notify them of the new data. In Fig. 3 we also show the results of the two functions. Note that the pop up notifications will appear for a few seconds once it receives success signal from the remote server. In the middle figure we show the data has been successfully uploaded to database, and in the right figure we show the emails have been sent out.

3 Challenges

(1) Communication

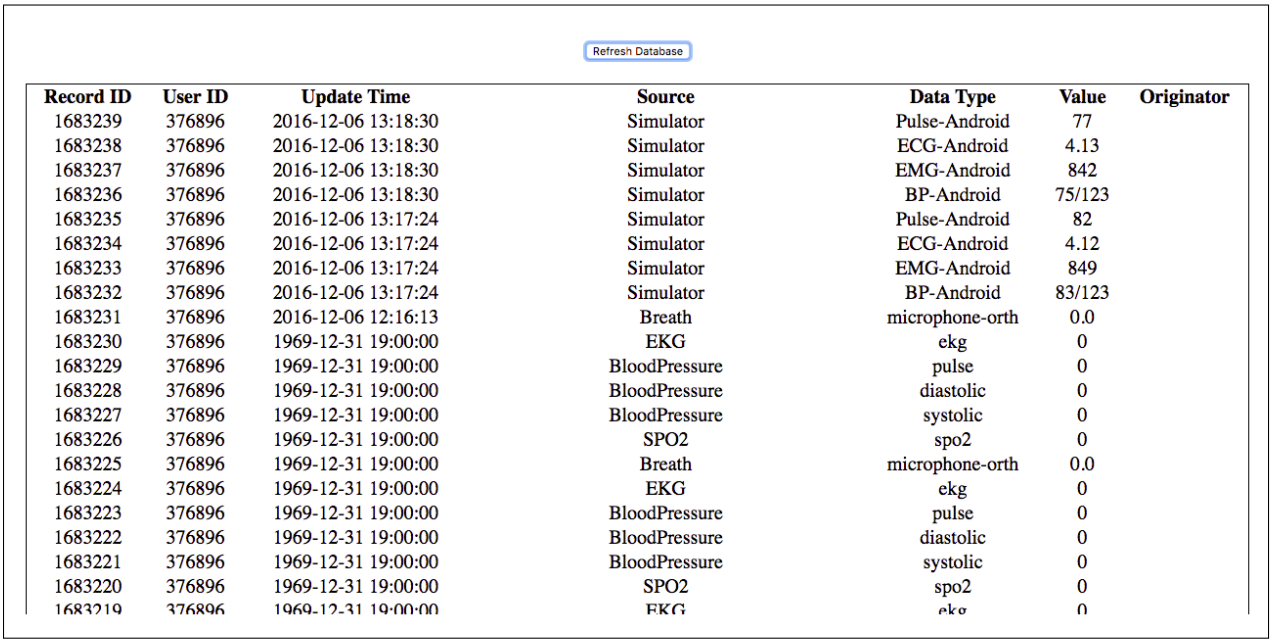
In this phase, since the TDR SIS server has not been set up on Android device yet, the *Uploader* has to take over part of responsibilities of the SIS server.
Specifically, to handle multiple concurrent connections from multiple components, the *Uploader* needs to open up a port (shown in the user interface) and wait for connections from components. To address this problem, we adapted the idea of SIS server and implemented a light weight version using similar thread pool strategy in the *Uploader* component.
Currently the only assumptions are that both sensor component and the *Uploader* are in the same local network, thus they can connect using socket communications.

(2) Uploader

The core function of this component is to upload received sensor data to remote server and save it to database. In this way, the collected data can be stored and further analysis of data can be performed. The connection is also using socket communication, and a PHP server is set up in the remote server. The rationality for such design is that database is not allowed for remote connection directly for security concern. Therefore, the data packet needs to be delivered to the PHP server first, then gets decoded by corresponding PHP programs and lastly adds to the database.

(3) Database Checker

Since the *Uploader* needs to send data to database, it is necessary to access the database to check if the process is successful. However, repeatedly accessing database using username and password increased the security risk. To address this problem, we developed a web database checker, and the only functionality is to display the latest 50 items in the database. In this way, developers do not need to send emails to the administrators or requires username and password to debug their program. Since its functionality is determined, the security risk is reduced to minimal.



Record ID	User ID	Update Time	Source	Data Type	Value	Originator
1683239	376896	2016-12-06 13:18:30	Simulator	Pulse-Android	77	
1683238	376896	2016-12-06 13:18:30	Simulator	ECG-Android	4.13	
1683237	376896	2016-12-06 13:18:30	Simulator	EMG-Android	842	
1683236	376896	2016-12-06 13:18:30	Simulator	BP-Android	75/123	
1683235	376896	2016-12-06 13:17:24	Simulator	Pulse-Android	82	
1683234	376896	2016-12-06 13:17:24	Simulator	ECG-Android	4.12	
1683233	376896	2016-12-06 13:17:24	Simulator	EMG-Android	849	
1683232	376896	2016-12-06 13:17:24	Simulator	BP-Android	83/123	
1683231	376896	2016-12-06 12:16:13	Breath	microphone-orth	0.0	
1683230	376896	1969-12-31 19:00:00	EKG	ekg	0	
1683229	376896	1969-12-31 19:00:00	BloodPressure	pulse	0	
1683228	376896	1969-12-31 19:00:00	BloodPressure	diastolic	0	
1683227	376896	1969-12-31 19:00:00	BloodPressure	systolic	0	
1683226	376896	1969-12-31 19:00:00	SPO2	spo2	0	
1683225	376896	1969-12-31 19:00:00	Breath	microphone-orth	0.0	
1683224	376896	1969-12-31 19:00:00	EKG	ekg	0	
1683223	376896	1969-12-31 19:00:00	BloodPressure	pulse	0	
1683222	376896	1969-12-31 19:00:00	BloodPressure	diastolic	0	
1683221	376896	1969-12-31 19:00:00	BloodPressure	systolic	0	
1683220	376896	1969-12-31 19:00:00	SPO2	spo2	0	
1683219	376896	1969-12-31 19:00:00	EKG	ekg	0	

Figure 6: Screenshot of the database checker

Fig. 6 shows the screenshots of our database checker. Note that it is the same as the real database data.

4 Specifications

(1) Development Environment

The development is using Android Studio 2.2.2 on macOS Sierra 10.12.1. It is tested on a Nexus S device. The device resolution is 480×800 and device satisfies Android API 19.

(2) Message Definitions

The message used in this system is compatible with the original TDR SIS message type definition. An example message is shown below in XML format. Currently we implemented "Register", "Confirm" and "Reading" types of messages, but this could be easily expanded to other message types once we decided the specific functions of other components.

```
<?xml version="1.0" standalone="yes"?>
<Msg>
    <Item>
        <Key>Scope</Key>
        <Value>SIS.Scope1</Value>
    </Item>
    <Item>
        <Key>MessageType</Key>
        <Value>Reading</Value>
    </Item>
    <Item>
        <Key>Sender</Key>
        <Value>AndroidSensor</Value>
    </Item>
    <Item>
        <Key>Data_BP</Key>
        <Value>80/117</Value>
    </Item>
    <Item>
        <Key>Data_Date</Key>
        <Value>2016/12/03 17:40:07</Value>
    </Item>
    <Item>
        <Key>Data_ECG</Key>
        <Value>4.10</Value>
    </Item>
    <Item>
        <Key>Data_Pulse</Key>
        <Value>108</Value>
    </Item>
    <Item>
        <Key>Data_EMG</Key>
        <Value>842</Value>
    </Item>
</Msg>
```

5 Contributions (Extra Deeds)

We claim the following perspectives as our extra deeds in this project.

Designed a nice GUI on Android platform to display the progress of the communications between components and the remote database/email service. The system is also fully compatible with existing TDR SIS system and the message types are defined in the same way.

The *Uploader* is able connect to multiple components and this functionality makes it feasible to debug multiple components since the Android TDR server is not available yet. It needs minimal changes to work with an Android TDR server once it is is online.

Since accessing database requires username and password, it is not convenient for developers to test their program. We developed a dedicated checker to return only the latest top 50 items in the database. The only functionality is to display the latest items in database thus no security risk is added. In this way, developers can easily tell if their uploading function succeeds without the need to access the real database, or sending emails to the administrators.

We developed a *Simulator* system that can generates pseudo data for different components. The only difference of our simulator with real sensor collector is that data is not collected by device. Apart from that everything is the same as in real application. The data is transferred through connection and get processed by the *Uploader*. It is of great help for debugging the system.

The connection from other components to *Uploader* is implemented and can be easily extended to other components by just copy-and-paste an independent Java class. This simplifies the process of developing new components on Android.

Appendix

Here is the source code we used to develop the Android system. Note that the implementation can be obtained from the following address.

https://github.com/zhmd/AndroidTDR/

It is worth noting that **we only show the code written for project and some existing codes inherited from TDR SIS system is not shown**. We strongly encourage you to refer to the GitHub for original source code if you need to use this code.

UploaderActivity.java

```
package pitt.cs2310.mzhang.androidtdr;

public class UploaderActivity extends AppCompatActivity {
    private static final String SERVER_ADDR = "http://ksiresearch.org/chronobot/android_upload.php";

    private static final int duration = Toast.LENGTH_SHORT;

    private static final String SCOPE = "SIS.Scope1";
    private static final int serverPort = 8011;
    private static final String NAME = "AndroidUploader";
    private static final List<String> TYPES = new ArrayList<String>(
            Arrays.asList(new String[]{"Reading", "Connect"}));
    private static final String TAG = "UploaderActivity";
    static TextView infoport, infoip, msg;
    static TextView bp_display;
    static TextView emg_display;
    static TextView ecg_display;
    static TextView pulse_display;
    static TextView username_display;
    static TextView time_display;
    static Context context;
    static Toast toast;
    ServerSocket serverSocket;
    String message;
    Button send;
    View thisView;
    String myIpAddress;
    AcceptSockets uploaderServer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_uploader);

        infoport = (TextView) findViewById(R.id.infoport);
        infoip = (TextView) findViewById(R.id.infoip);
        msg = (TextView) findViewById(R.id.msg);
        send = (Button) findViewById(R.id.send_btn);
        message = "";
        username_display = (TextView) findViewById(R.id.Username);
        bp_display = (TextView) findViewById(R.id.BP_data);
        emg_display = (TextView) findViewById(R.id.EMG_data);
        ecg_display = (TextView) findViewById(R.id.ECG_data);
        pulse_display = (TextView) findViewById(R.id.Pulse_data);
        time_display = (TextView) findViewById(R.id.Time_data);
        thisView = findViewById(android.R.id.content);
        context = getApplicationContext();

        myIpAddress = getIpAddress();
        infoip.setText("Uploader␣Address:␣" + myIpAddress);

        new Thread(new Runnable() {
            @Override
            public void run() {
                uploaderServer = new AcceptSockets(myIpAddress, serverPort, UploaderActivity.this);
                uploaderServer.start();
            }
        }).start();

        send.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                ClientSimulator simulator = new ClientSimulator();
                simulator.oneMessage();
            }
        });
    }
```

```
@Override
protected void onDestroy() {
    super.onDestroy();

    if (serverSocket != null) {
        try {
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

private String getIpAddress() {
    String ip = "";
    try {
        Enumeration<NetworkInterface> enumNetworkInterfaces = NetworkInterface.getNetworkInterfaces();
        while (enumNetworkInterfaces.hasMoreElements()) {
            NetworkInterface networkInterface = enumNetworkInterfaces.nextElement();
            Enumeration<InetAddress> enumInetAddress = networkInterface.getInetAddresses();
            while (enumInetAddress.hasMoreElements()) {
                InetAddress inetAddress = enumInetAddress.nextElement();
                String sAddr = inetAddress.getHostAddress();
                if (!inetAddress.isLoopbackAddress() && sAddr.indexOf(':') < 0) {
                    Log.d(TAG, "IP␣Address:␣" + sAddr);
                    return sAddr;
                }
            }
        }
    } catch (SocketException e) {
        ip += "Something␣Wrong!␣" + e.toString() + "\n";
        Log.d(TAG, "Something␣wrong!␣" + e.toString());
    }
    return ip;
}


public static class SendToPhpJob extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... params) {
        String message = params[0];
        Log.d(TAG, message);
        URL url;
        HttpURLConnection connection = null;
        OutputStream os = null;
        InputStream is = null;
        String ret;

        try {
            url = new URL(SERVER_ADDR);
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Content-Type", "application/json;charset=utf-8");
            connection.setRequestProperty("X-Requested-With", "XMLHttpRequest");
            connection.setDoInput(true);
            connection.setDoOutput(true);

            os = connection.getOutputStream();
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(os, "UTF-8"));
            writer.write(message);
            writer.flush();
            writer.close();
            os.close();

            connection.connect();

            int response_code = connection.getResponseCode();

            if (response_code == HttpURLConnection.HTTP_OK) {
                Log.d(TAG, "connection␣successful.");
            }
            is = connection.getInputStream();
            BufferedReader reader = new BufferedReader(new InputStreamReader(is));

            StringBuilder result = new StringBuilder();
            String line;

            while ((line = reader.readLine()) != null) {
                result.append(line);
            }

            Log.d(TAG, "input␣shown␣below:");

            ret = result.toString();

        } catch (UnknownHostException e) {
            Log.d(TAG, "UnknownHostException:␣" + e.toString());
            ret = "";
        } catch (Exception e) {
            Log.d(TAG, "UnknownHostException:␣" + e.toString());
            ret = "";
```

```
            } finally {
                try {
                    if (is != null) {
                        is.close();
                    }
                    if (os != null) {
                        os.close();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
                if (connection != null) {
                    connection.disconnect();
                }
            }
            return ret;
        }

        @Override
        protected void onPostExecute(String response) {
            Log.d(TAG, response);
            if (response.equals("success")) {
                String result = "Data has been uploaded!";
                toast = Toast.makeText(context, result, duration);
            } else {
                String result = "Something wrong happened!";
                toast = Toast.makeText(context, result, duration);
            }
            toast.show();
        }
    }
}
```

ClientSimulator

```
package pitt.cs2310.mzhang.androidtdr;

class ClientSimulator {
    private static final String TAG = "ClientSimulator";
    private static String serverAddress;
    private static int serverPort;
    private static MsgEncoder msgEncoder;

    ClientSimulator() {
    }

    ClientSimulator(String addr, int port) {
        serverAddress = addr;
        serverPort = port;
        Log.d(TAG, "Server Address: " + serverAddress);
        Log.d(TAG, "Server Port: " + serverPort);
    }

    void initialize() {
        try {
            Socket socket = new Socket(serverAddress, serverPort);
            msgEncoder = new MsgEncoder(socket.getOutputStream());
        } catch (UnknownHostException e) {
            Log.d(TAG, "UnknownHostException: " + e.toString());
        } catch (IOException e) {
            Log.d(TAG, "IOException: " + e.toString());
        }
    }

    void oneMessage() {
        final KeyValueList message = random_data();
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    msgEncoder.sendMsg(message);
                } catch (IOException e) {
                    Log.d(TAG, "IOException: " + e.toString());
                }
            }
        }).start();
    }

    void oneMessage(KeyValueList kvList) {
        final KeyValueList content = kvList;
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    msgEncoder.sendMsg(content);
                } catch (IOException e) {
                    Log.d(TAG, "IOException: " + e.toString());
                }
```

```
            }
        }).start();
    }

    private KeyValueList random_data() {
        String data;

        KeyValueList sensor_data = new KeyValueList();
        sensor_data.putPair("Scope", "SIS.Scope1");
        sensor_data.putPair("MessageType", "Reading");
        sensor_data.putPair("Sender", "Simulator");

        Random r = new Random();

        data = String.valueOf(r.nextInt(10) + 75) + "/" + String.valueOf(r.nextInt(10) + 115);
        sensor_data.putPair("Data_BP", data);

        data = String.valueOf(r.nextInt(10) + 840);
        sensor_data.putPair("Data_EMG", data);

        data = "4.1" + String.valueOf(r.nextInt(10));
        sensor_data.putPair("Data_ECG", data);

        data = String.valueOf(r.nextInt(50) + 70);
        sensor_data.putPair("Data_Pulse", data);

        long curr_time = System.currentTimeMillis();
        sensor_data.putPair("Data_Date", String.valueOf(curr_time));

        return sensor_data;
    }
}
```

AccessSockets

```
package pitt.cs2310.mzhang.androidtdr;

class AcceptSockets {
    private static final String TAG = "AcceptSockets";
    private String serverAddress;
    private int serverPort;
    private ServerSocket universal;
    private ExecutorService service;
    private UploaderActivity activity;

    AcceptSockets(String address, int port, UploaderActivity activity) {
        this.serverPort = port;
        this.serverAddress = address;
        this.activity = activity;
    }

    void start() {
        Log.d(TAG, "Starting␣server.");
        try {
            universal = new ServerSocket(serverPort);
            Log.d(TAG, "Server␣started.␣Port␣Number:␣" + universal.getLocalPort());
            activity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    UploaderActivity.infoport.setText("Listen␣to␣Port:␣" + universal.getLocalPort());
                }
            });
            service = Executors.newCachedThreadPool();

            ClientSimulator simulator = new ClientSimulator(serverAddress, serverPort);
            simulator.initialize();

            while (true) {
                Log.d(TAG, "Prepare␣to␣accept␣new␣connection.");
                Socket socket = universal.accept();
                service.execute(new SocketsThread(socket, activity));
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (universal != null) {
                try {
                    universal.close();
                    Log.d(TAG, "Server␣Shut␣down.");
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

SocketsThread

```
package pitt.cs2310.mzhang.androidtdr;

class SocketsThread implements Runnable {
    static final String SMTP_HOST_NAME = "smtp.ksiresearch.org.ipage.com";
    static final String SMTP_PORT = "587";
    static final String emailFromAddress = "chronobot@ksiresearch.org";
    private static final int duration = Toast.LENGTH_SHORT;
    static final private String TAG = "SocketsThread";
    static final private String MyName = "AndroidUploader";
    private static UploaderReading reading = new UploaderReading();
    private static int counter = 1;
    private int connectionNumber;
    private MsgDecoder msgDecoder;
    private MsgEncoder msgEncoder;
    private Activity activity;

    SocketsThread(final Socket socket, UploaderActivity activity) {
        this.activity = activity;
        Log.i(TAG, "Received␣Connection␣#" + counter + ".␣Port␣Number:␣" + socket.getPort());
        connectionNumber = counter;
        counter++;
        try {
            msgDecoder = new MsgDecoder(socket.getInputStream());
            msgEncoder = new MsgEncoder(socket.getOutputStream());
        } catch (IOException e) {
            Log.e(TAG, "Initialize␣Failed.");
        }
        activity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                String message = UploaderActivity.msg.getText().toString();
                message = "New␣connection:␣#" + connectionNumber + ":␣" + socket.getInetAddress() + ":" + socket.getPort() + "\n"
                      + message;
                UploaderActivity.msg.setText(message);
            }
        });
    }

    private static boolean execute(String query) throws Exception {
        String url = "http://ksiresearch.org/chronobot/PHP_Post_copy.php";
        return PostQuery.PostToPHP(url, query);
    }

    private static String formQuery(long datetime, String source, String type, Object value) {
        return "Insert␣into␣`records`␣(`uid`,␣`datetime`,␣`source`,␣`type`,␣`value`)␣values␣('"
                + reading.uid
                + "',"
                + "FROM_UNIXTIME(" + datetime / 1000 + ")"
                + ",'"
                + source
                + "','"
                + type
                + "','"
                + value.toString()
                + "')";
    }

    @Override
    public void run() {
        try {
            KeyValueList kvList;
            while (true) {
                kvList = msgDecoder.getMsg();
                Log.d(TAG, "Received␣raw:␣<" + kvList.encodedString() + ">");
                if (kvList.size() == 0) {
                    break;
                }
                processMessage(kvList);
            }
            Log.i(TAG, "Connection␣#" + connectionNumber + "␣closed.");
            activity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    String message = UploaderActivity.msg.getText().toString();
                    message = "Connection␣#" + connectionNumber + "␣closed.\n" + message;
                    UploaderActivity.msg.setText(message);
                }
            });
        } catch (Exception e) {
            Log.e(TAG, "Error␣in␣decoding␣messages.");
        }
    }

    private void processMessage(KeyValueList kvList) {
        final String scope = kvList.getValue("Scope");
        final String messageType = kvList.getValue("MessageType");
        final String sender = kvList.getValue("Sender");
        Log.d(TAG, "Message␣Received:␣" + kvList.toString());
```

```
        switch (messageType) {
            case "Reading":
                reading.message_sender = sender;
                reading.BP = kvList.getValue("Data_BP");
                reading.ECG = kvList.getValue("Data_ECG");
                reading.EMG = kvList.getValue("Data_EMG");
                reading.collection_time = Long.valueOf(kvList.getValue("Data_Date"));
                reading.Pulse = kvList.getValue("Data_Pulse");

                SimpleDateFormat date_format = new SimpleDateFormat("yyyy-MM-dd␣HH:mm:ss");

                reading.readable_time = date_format.format(new Date(reading.collection_time));
                Log.d(TAG, "Readable␣Date:␣" + reading.readable_time);

                activity.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        UploaderActivity.username_display.setText(sender);
                        UploaderActivity.bp_display.setText(reading.BP);
                        UploaderActivity.emg_display.setText(reading.EMG);
                        UploaderActivity.ecg_display.setText(reading.ECG);
                        UploaderActivity.pulse_display.setText(reading.Pulse);
                        UploaderActivity.time_display.setText(reading.readable_time);
                        String message = UploaderActivity.msg.getText().toString();
                        message = "Data␣received␣from␣connection␣#" + connectionNumber + "\n" + message;
                        UploaderActivity.msg.setText(message);
                    }
                });

                String emailSubject = "New␣Sensor␣Readings␣from␣" + sender;
                String emailContent = kvList.toString();
                try {
                    send_msg();
                    send_email(emailFromAddress, reading.recipients, emailSubject, emailContent);
                } catch (MessagingException e) {
                    Log.d("Email", e.toString());
                } catch (Exception e) {
                    Log.d("UpdateDatabase", e.toString());
                }
                break;
            case "Register":
                KeyValueList reply = new KeyValueList();
                reply.putPair("MessageType", "Confirm");
                reply.putPair("Sender", MyName);
                reply.putPair("Scope", scope);

                try {
                    msgEncoder.sendMsg(reply);
                } catch (IOException e) {
                    Log.e(TAG, "IOException:␣" + e.toString());
                }
                break;
        }
    }

    private void send_msg() throws Exception {
        boolean result;
        result = execute(formQuery(reading.collection_time, reading.message_sender, "BP-Android", reading.BP));
        result = result && execute(formQuery(reading.collection_time, reading.message_sender, "EMG-Android", reading.EMG));
        result = result && execute(formQuery(reading.collection_time, reading.message_sender, "ECG-Android", reading.ECG));
        result = result && execute(formQuery(reading.collection_time, reading.message_sender, "Pulse-Android", reading.Pulse));

        Log.d(TAG, String.valueOf(result));
        if (result) {
            activity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    String notification = "Data␣has␣been␣uploaded!";
                    UploaderActivity.toast = Toast.makeText(UploaderActivity.context, notification, duration);
                    UploaderActivity.toast.show();
                }
            });
        } else {
            activity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    String notification = "Something␣wrong␣happened!";
                    UploaderActivity.toast = Toast.makeText(UploaderActivity.context, notification, duration);
                    UploaderActivity.toast.show();
                }
            });
        }
        Log.d(TAG, "Update␣to␣database␣completed.");
    }

    private void send_msg(KeyValueList kvList) {
        String bp_string = kvList.getValue("Data_BP");
        String emg_string = kvList.getValue("Data_EMG");
        String ecg_string = kvList.getValue("Data_ECG");
        String pulse_string = kvList.getValue("Data_Pulse");
        String time_string = kvList.getValue("Data_Date");
        String username_string = kvList.getValue("Sender");
```

```
        JSONObject messageJSON = new JSONObject();

        try {
            messageJSON.put("Username", username_string);
            messageJSON.put("BP", bp_string);
            messageJSON.put("ECG", ecg_string);
            messageJSON.put("EMG", emg_string);
            messageJSON.put("Pulse", pulse_string);
            messageJSON.put("Timestamp", time_string);
        } catch (JSONException e) {
            e.printStackTrace();
        }

        String message = messageJSON.toString();
        Log.i(TAG, "Upload␣to␣TDR␣Database:␣" + message);
        new UploaderActivity.SendToPhpJob().execute(message);
    }

    void send_email(String from, String[] recipients,
                    String subject, String content) throws MessagingException {
        Log.d("Email", "Start␣to␣send␣Email.");
        boolean debug = false;
        Properties props = new Properties();
        props.put("mail.smtp.host", SMTP_HOST_NAME);
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.port", SMTP_PORT);
        props.put("mail.smtp.ssl.enable", "true");
        Log.d("Email", "Set␣parameters␣successfully.");

        Session session = Session.getDefaultInstance(props, new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication("chronobot@ksiresearch.org", "Health14");
            }
        });
        Log.d("Email", "Sessions␣created␣successfully.");
        session.setDebug(debug);

        Message msg = new MimeMessage(session);
        InternetAddress addressFrom = new InternetAddress(from);
        msg.setFrom(addressFrom);

        InternetAddress[] addressTo = new InternetAddress[recipients.length];
        for (int i = 0; i < recipients.length; i++) {
            addressTo[i] = new InternetAddress(recipients[i]);
        }
        msg.setRecipients(Message.RecipientType.TO, addressTo);

        msg.setSubject(subject);

        msg.setText(reading.toString());

        Log.d("Email", "Ready␣to␣send␣out␣messages.");
        Transport.send(msg);
        activity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                String notification = "Email␣notifications␣sent␣out!";
                UploaderActivity.toast = Toast.makeText(UploaderActivity.context, notification, duration);
                UploaderActivity.toast.show();
            }
        });
        Log.d("Email", "Send␣sensor␣readings␣to␣recipients'␣Email␣successfully!");
    }
}
```

activity_uploader.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_uploader"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    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="pitt.cs2310.mzhang.androidtdr.UploaderActivity">

    <LinearLayout
        android:layout_width="0px"
        android:layout_height="0px"
        android:focusable="true"
        android:focusableInTouchMode="true" />

    <LinearLayout
        android:id="@+id/info_box"
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp"
    android:orientation="vertical">

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

    <TextView
        android:id="@+id/infoport"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>

<LinearLayout
    android:id="@+id/data_box"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/info_box"
    android:layout_marginBottom="20dp"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:layout_width="130dp"
            android:layout_height="wrap_content"
            android:text="@string/menu_Username" />

        <TextView
            android:id="@+id/Username"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:maxLines="1"
            android:textSize="12sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:layout_width="130dp"
            android:layout_height="wrap_content"
            android:text="@string/menu_BP" />

        <TextView
            android:id="@+id/BP_data"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:maxLines="1"
            android:textSize="12sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:layout_width="130dp"
            android:layout_height="wrap_content"
            android:text="@string/menu_EMG" />

        <TextView
            android:id="@+id/EMG_data"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:maxLines="1"
            android:textSize="12sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:layout_width="130dp"
            android:layout_height="wrap_content"
            android:text="@string/menu_ECG" />

        <TextView
            android:id="@+id/ECG_data"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:maxLines="1"
            android:textSize="12sp" />
    </LinearLayout>
```

```
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:layout_width="130dp"
                android:layout_height="wrap_content"
                android:text="@string/menu_Pulse" />

            <TextView
                android:id="@+id/Pulse_data"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:maxLines="1"
                android:textSize="12sp" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:layout_width="130dp"
                android:layout_height="wrap_content"
                android:text="@string/menu_Time" />

            <TextView
                android:id="@+id/Time_data"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:maxLines="1"
                android:textSize="12sp" />
        </LinearLayout>


    </LinearLayout>

    <LinearLayout
        android:id="@+id/message_box"
        android:layout_width="match_parent"
        android:layout_height="180dp"
        android:layout_below="@id/data_box">

        <TextView
            android:id="@+id/msg"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/message_box"
        android:gravity="center">

        <Button
            android:id="@+id/send_btn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/simulate" />
    </LinearLayout>

</RelativeLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="pitt.cs2310.mzhang.androidtdr">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />


    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".UploaderActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

```
    </application>

</manifest>
```

check_database.html

```
<html>

<body>
    <form action="http://ksiresearch.org/chronobot/android_show.php" method="post" target="display">
        <div style="text-align:␣center;␣margin-top:␣40px;">
            <input type="submit" value="Refresh␣Database">
        </div>
    </form>

    <div style="text-align:␣center;">
        <iframe name="display" width="80%" frameborder="0" height="550px"></iframe>
    </div>
</body>
</html>
```

android_show.php

```
<head>
    <style>
        table {
            font-size: 20px;
        }
        td {
            vertical-align: middle;
            text-align: center;
        }
    </style>
</head>

<body>
    <div style="text-align:␣center;">
        <?php
        $sql = 'SELECT␣*␣FROM␣␣`records`␣WHERE␣1␣ORDER␣BY␣␣`rid`␣DESC␣LIMIT␣0␣,␣50';

        if($sql){
            $link=mysql_connect("ksiresearchorg.ipagemysql.com","duncan","duncan");
            if($link){
                mysql_select_db("chronobot");
                $result=mysql_query($sql);
                if($result){
                    echo "<table␣style='border-collapse:␣collapse;␣border:␣1px␣solid␣black;␣width:␣100%;'>" . "\n";
                    echo "<tr><th>Record␣ID</th><th>User␣ID</th><th>Update␣Time</th><th>Source</th><th>Data␣Type</th><th>Value</th
                          ><th>Originator</th></tr>";
                    while($row=mysql_fetch_row($result)){
                        echo "<tr>";
                        foreach($row as $value){
                            echo "<td>".$value."</td>";
                        }
                        echo "</tr>";
                    }
                }
            }else{
                echo "␣MySQL␣connect␣failed␣!␣\n";
            }
        }else{
            echo "␣No␣Query␣!␣\n";
        }
        exit();
        ?>
    </div>
</body>
```