In this project, I have worked on Bluetooth connection between Arduino and WABBS-face. I have created the Android app so the phone could connect to Arduino, then we will use it to make the phone stay awake.

I also have the Arduino test code, but it is not included in here.

-----Vinh Tran---

*/\*\*  
 \* Called when the app gets opened  
 \* <p>  
 \* Sets up the Main Activity and creates the view required.  
 \* Also gets the Camera Views and sets up the image Runnable  
 \* </p>  
 \*  
 \** ***@param*** *\** ***@return*** *\*/*@Override

//----------------\_Vinh -------------------------------------------------------------------------------  
//-------------------------------------------------------------------------------------------------------  
// GUI Components  
private TextView mBluetoothStatus; // show in the box the status of the connection (failed/connected)  
private TextView mReadBuffer; // show in the box the data that it receive  
private Button mScanBtn; // turn on bluetooth of the phone  
private Button mOffBtn; // turn off bluetooth of the phone  
private Button mListPairedDevicesBtn; //show paired devices  
private Button mDiscoverBtn; // find new bluetooth devices  
private BluetoothAdapter mBTAdapter; //this one will query a list of paired devices, initiate device discovery  
private Set<BluetoothDevice> mPairedDevices; //create a set of paired devices  
private ArrayAdapter<String> mBTArrayAdapter; //array of mBTAdapter  
private ListView mDevicesListView; //list of devices  
private CheckBox mLED1; // LED box to send out the signal  
  
private Handler mHandler; // Our main handler that will receive callback notifications  
private ConnectedThread mConnectedThread; // bluetooth background worker thread to send and receive data  
private BluetoothSocket mBTSocket = null; // bi-directional client-to-client data path  
  
private static final UUID *BTMODULEUUID* = UUID.*fromString*("00001101-0000-1000-8000-00805F9B34FB"); // "random" unique identifier  
//we could change this if we want  
  
  
// #defines for identifying shared types between calling functions  
private final static int *REQUEST\_ENABLE\_BT* = 1; // used to identify adding bluetooth names  
private final static int *MESSAGE\_READ* = 2; // used in bluetooth handler to identify message update  
private final static int *CONNECTING\_STATUS* = 3; // used in bluetooth handler to identify message status  
  
  
//-------------------------------------------------------------------------------------------------------  
//-------------------------------------------------------------------------------------------------------  
//-------------------------------------------------------------------------------------------------------

//-------------------------------------Vinh Bliuetooth--------------------------------------------------  
 //-------------------------------------------------------------------------------------------------------  
 mBTArrayAdapter = new ArrayAdapter<String>(this, android.R.layout.*simple\_list\_item\_1*);  
 mBTAdapter = BluetoothAdapter.*getDefaultAdapter*(); // get a handle on the bluetooth radio  
  
  
 //mDevicesListView.setAdapter(mBTArrayAdapter); // assign model to view  
 //mDevicesListView.setOnItemClickListener(mDeviceClickListener);  
  
  
 //This handler will show the message on the RX if receive any message and also show  
 //the device which is connected or failed connection.  
 mHandler = new Handler() {  
 public void handleMessage(android.os.Message msg) {  
 if (msg.what == *MESSAGE\_READ*) {  
 String readMessage = null;  
 try {  
 readMessage = new String((byte[]) msg.obj, "UTF-8");  
 } catch (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 }  
 //mReadBuffer.setText(readMessage);  
 }  
  
 if (msg.what == *CONNECTING\_STATUS*) {  
 if (msg.arg1 == 1)  
 {  
  
 }  
 // mBluetoothStatus.setText("Connected to Device: " + (String) (msg.obj));  
 else  
 {  
 //mBluetoothStatus.setText("Connection Failed");  
 }  
  
 }  
 }  
 };  
  
 if (mBTArrayAdapter == null) {  
 // Device does not support Bluetooth  
 //mBluetoothStatus.setText("Status: Bluetooth not found");  
 Toast.*makeText*(getApplicationContext(), "Bluetooth device not found!", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 //when we check the box mLED1, it write the message  
 bluetoothOn();  
  
  
 //list paired devices  
 listPairedDevices();  
  
 if(!mBTAdapter.isEnabled()) {  
 Toast.*makeText*(getBaseContext(), "Bluetooth not on", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 else{  
 //mBluetoothStatus.setText("Connecting...");  
 // Get the device MAC address, which is the last 17 chars in the View  
  
 //String info = ((TextView) v).getText().toString();  
 //final String address = info.substring(info.length() - 17);  
 //final String name = info.substring(0,info.length() - 17);  
  
 // Spawn a new thread to avoid blocking the GUI one  
 new Thread()  
 {  
 public void run() {  
 boolean fail = false;  
  
 BluetoothDevice device = mBTAdapter.getRemoteDevice("00:06:66:08:61:A5");  
  
 try {  
 mBTSocket = createBluetoothSocket(device);  
 } catch (IOException e) {  
 fail = true;  
 Toast.*makeText*(getBaseContext(), "Socket creation failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 // Establish the Bluetooth socket connection.  
 try {  
 mBTSocket.connect();  
 } catch (IOException e) {  
 try {  
 fail = true;  
 mBTSocket.close();  
 mHandler.obtainMessage(*CONNECTING\_STATUS*, -1, -1)  
 .sendToTarget();  
 } catch (IOException e2) {  
 //insert code to deal with this  
 Toast.*makeText*(getBaseContext(), "Socket creation failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 if(fail == false) {  
 mConnectedThread = new ConnectedThread(mBTSocket);  
 mConnectedThread.start();  
  
 mHandler.obtainMessage(*CONNECTING\_STATUS*, 1, -1, "FireFly-61A5")  
 .sendToTarget();  
 }  
 }  
 }.start();  
  
 }  
  
  
  
 }  
  
 //-------------------------------------------------------------------------------------------------------  
 //-------------------------------------------------------------------------------------------------------

private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws IOException {  
 return device.createRfcommSocketToServiceRecord(*BTMODULEUUID*);  
 //creates secure outgoing connection with BT device using UUID  
}  
  
  
private class ConnectedThread extends Thread {  
 private final BluetoothSocket mmSocket;  
 private final InputStream mmInStream;  
 private final OutputStream mmOutStream;  
  
 public ConnectedThread(BluetoothSocket socket) {  
 mmSocket = socket;  
 InputStream tmpIn = null;  
 OutputStream tmpOut = null;  
  
 // Get the input and output streams, using temp objects because  
 // member streams are final  
 try {  
 tmpIn = socket.getInputStream();  
 tmpOut = socket.getOutputStream();  
 } catch (IOException e) {  
 }  
  
 mmInStream = tmpIn;  
 mmOutStream = tmpOut;  
 }  
  
 public void run() {  
 byte[] buffer = new byte[1024]; // buffer store for the stream  
 int bytes; // bytes returned from read()  
 // Keep listening to the InputStream until an exception occurs  
 while (true) {  
 try {  
 // Read from the InputStream  
 bytes = mmInStream.available();  
 if (bytes != 0) {  
 SystemClock.*sleep*(100); //pause and wait for rest of data. Adjust this depending on your sending speed.  
 bytes = mmInStream.available(); // how many bytes are ready to be read?  
 bytes = mmInStream.read(buffer, 0, bytes); // record how many bytes we actually read  
 mHandler.obtainMessage(*MESSAGE\_READ*, bytes, -1, buffer)  
 .sendToTarget(); // Send the obtained bytes to the UI activity  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
  
 break;  
 }  
 }  
 }  
  
 /\* Call this from the main activity to send data to the remote device \*/  
 public void write(String input) {  
 byte[] bytes = input.getBytes(); //converts entered String into bytes  
 try {  
 mmOutStream.write(bytes);  
 } catch (IOException e) {  
 }  
 }  
  
 /\* Call this from the main activity to shutdown the connection \*/  
 public void cancel() {  
 try {  
 mmSocket.close();  
 } catch (IOException e) {  
 }  
 }  
}  
  
private AdapterView.OnItemClickListener mDeviceClickListener = new AdapterView.OnItemClickListener() {  
 public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {  
  
 if (!mBTAdapter.isEnabled()) {  
 Toast.*makeText*(getBaseContext(), "Bluetooth not on", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 //mBluetoothStatus.setText("Connecting...");  
 // Get the device MAC address, which is the last 17 chars in the View  
 String info = ((TextView) v).getText().toString();  
 final String address = info.substring(info.length() - 17);  
 final String name = info.substring(0, info.length() - 17);  
  
 // Spawn a new thread to avoid blocking the GUI one  
 new Thread() {  
 public void run() {  
 boolean fail = false;  
  
 BluetoothDevice device = mBTAdapter.getRemoteDevice(address);  
  
 try {  
 mBTSocket = createBluetoothSocket(device);  
 } catch (IOException e) {  
 fail = true;  
 Toast.*makeText*(getBaseContext(), "Socket creation failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 // Establish the Bluetooth socket connection.  
 try {  
 mBTSocket.connect();  
 } catch (IOException e) {  
 try {  
 fail = true;  
 mBTSocket.close();  
 mHandler.obtainMessage(*CONNECTING\_STATUS*, -1, -1)  
 .sendToTarget();  
 } catch (IOException e2) {  
 //insert code to deal with this  
 Toast.*makeText*(getBaseContext(), "Socket creation failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 if (fail == false) {  
 mConnectedThread = new ConnectedThread(mBTSocket);  
 mConnectedThread.start();  
  
 mHandler.obtainMessage(*CONNECTING\_STATUS*, 1, -1, name)  
 .sendToTarget();  
 }  
 }  
 }.start();  
 }  
};  
  
private void bluetoothOn() {  
 if (!mBTAdapter.isEnabled()) {  
 Intent enableBtIntent = new Intent(BluetoothAdapter.*ACTION\_REQUEST\_ENABLE*);  
 startActivityForResult(enableBtIntent, *REQUEST\_ENABLE\_BT*);  
 //mBluetoothStatus.setText("Bluetooth enabled");  
 Toast.*makeText*(getApplicationContext(), "Bluetooth turned on", Toast.*LENGTH\_SHORT*).show();  
  
 } else {  
 Toast.*makeText*(getApplicationContext(), "Bluetooth is already on", Toast.*LENGTH\_SHORT*).show();  
 }  
}  
  
private void listPairedDevices() {  
 mPairedDevices = mBTAdapter.getBondedDevices();  
 if (mBTAdapter.isEnabled()) {  
 // put it's one to the adapter  
 for (BluetoothDevice device : mPairedDevices) {  
 mBTArrayAdapter.add(device.getName() + "\n" + device.getAddress());  
 Log.*e*("Bluetooth: ", device.getName() + "\n" + device.getAddress());  
 Log.*d*("tag", "Value: " + device.getName() + "\n" + device.getAddress());  
 }  
  
  
 Toast.*makeText*(getApplicationContext(), "Show Paired Devices", Toast.*LENGTH\_SHORT*).show();  
  
 } else  
 Toast.*makeText*(getApplicationContext(), "Bluetooth not on", Toast.*LENGTH\_SHORT*).show();  
}  
  
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent Data) {  
 // Check which request we're responding to  
 if (requestCode == *REQUEST\_ENABLE\_BT*) {  
 // Make sure the request was successful  
 if (resultCode == *RESULT\_OK*) {  
 // The user picked a contact.  
 // The Intent's data Uri identifies which contact was selected.  
 //mBluetoothStatus.setText("Enabled");  
 } else {  
 //mBluetoothStatus.setText("Disabled");  
 }  
 }  
  
}