

Deepak | February 21, 2015

Blueno

An android based app to control robot wirelessly

# Objective:

# To build an Android app which can controller a robot through Bluetooth, a touch interface to make a virtual joystick.

Prerequisites:

* Basics of android environment and app systems
* Java
* Object oriented paradigm
* UART Communication
* Arduino hardware and programming

Hardware requirements:

* Android Smartphone with Bluetooth capability
* A Robot with Arduino controller
* A HC-05 Bluetooth module
* A windows computer with eclipse for building the app

Approach:

Basically this is an Android app which communicates with the HC-05 Bluetooth module on the bot. The interface is made like a joystick working on touch sense, whenever we touch the screen and move the finger the relative position of the finger is calculated with respect to the initial touch with some threshold, this angle is sent to the bot through Bluetooth. On bot side a program is written in Arduino which takes data command serially and to move its motors accordingly.

Overall System:

UART

Wireless

Android Smartphone

HC-05 Bluetooth module

Robot

Communication System:

Kinect side

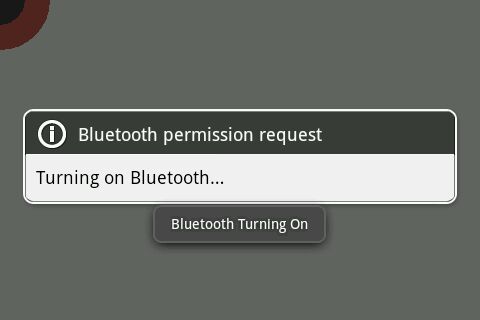
Bot side

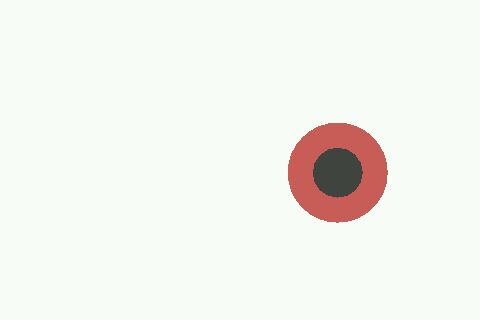
Request data and wait until it comes

Sends latest data and wait until next request

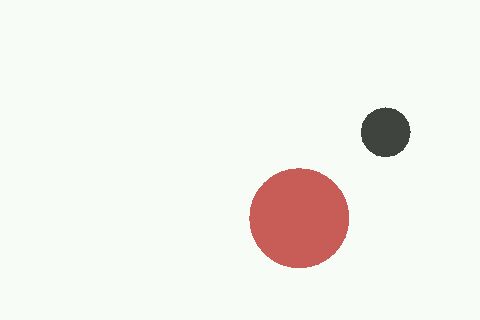


It requests to turn on Bluetooth and turns it on. After that it searches for the device with name specifies in the code and connects to it.





After Initialization and Bluetooth prompt, it opens a white area where, whenever to tap, a reference circle (brown) is made and as you move finger black circle moves with it.



Key Codes Snippets:

//From App Bluetooth.java

This class is used to connect to the Bluetooth device after found

private class ConnectThread extends Thread

{

private final BluetoothSocket mmSocket;

private final BluetoothDevice mmDevice;

public ConnectThread(BluetoothDevice device)

{

BluetoothSocket tmp = null;

mmDevice = device;

try

{

tmp = device.createRfcommSocketToServiceRecord(MY\_UUID);

}

catch (IOException e)

{

}

mmSocket = tmp;

}

public void run()

{

btAdapter.cancelDiscovery();

try

{

mmSocket.connect();

}

catch (IOException connectException)

{

try

{

mmSocket.close();

}

catch (IOException closeException)

{

}

return;

}

connectedThread = new ConnectedThread(mmSocket);

String s = "successfully connected";

connectedThread.write(s.getBytes());

loop();

}

Call loop function after connection, where all data is sent

public void cancel()

{

try

{

mmSocket.close();

}

catch (IOException e)

{

}

}

}

The Initializes the I/O stream and has functions to write and read to the stream

private class ConnectedThread

{

private final BluetoothSocket mmSocket;

private final InputStream mmInStream;

private final OutputStream mmOutStream;

public ConnectedThread(BluetoothSocket socket)

{

mmSocket = socket;

InputStream tmpIn = null;

OutputStream tmpOut = null;

try {

tmpIn = socket.getInputStream();

Getting I/O stream out of the socket

tmpOut = socket.getOutputStream();

} catch (IOException e) { }

mmInStream = tmpIn;

mmOutStream = tmpOut;

}

public void write(byte[] bytes)

{

A function to write data to Bluetooth socket

try

{

mmOutStream.write(bytes);

} catch (IOException e) { }

}

public void cancel()

{

try

{

This Broadcast receiver listens the Bluetooth states

mmSocket.close();

} catch (IOException e) { }

}

}

BroadcastReceiver bluetoothState = new BroadcastReceiver()

{

public void onReceive(Context context, Intent intent)

{

String stateExtra = BluetoothAdapter.EXTRA\_STATE;

String toast;

int state = intent.getIntExtra(stateExtra, -1);

if (state == BluetoothAdapter.STATE\_TURNING\_ON) {

toast = "Bluetooth Turning On";

} else if (state == BluetoothAdapter.STATE\_ON) {

toast = "Bluetooth On";

SetupUI();

} else if (state == BluetoothAdapter.STATE\_OFF) {

toast = "Bluetooth Off";

SetupUI();

} else

toast = "Unknown Bluetooth State";

Toast.makeText(MainActivity.this, toast, Toast.LENGTH\_SHORT).show();

}

This Broadcast receiver listens to the Bluetooth search results`

};

BroadcastReceiver discoveryResult = new BroadcastReceiver()

{

public void onReceive(Context context, Intent intent)

{

String action = intent.getAction();

if (BluetoothAdapter.ACTION\_DISCOVERY\_FINISHED.equals(action)) { //print.append("Finished\n");

}

if (BluetoothDevice.ACTION\_FOUND.equals(action))

{

BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA\_DEVICE); btDeviceList.add(device);

if (device.getName().equals("SLAVE\_TMP") && !deviceFound)

{

deviceFound = true;

btAdapter.cancelDiscovery();

ConnectThread connect = new ConnectThread(device);

connect.start();

}

Call ConnectThread class on successful search of SLAVE\_TMP

}

else

{

if (BluetoothDevice.ACTION\_UUID.equals(action))

{

BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA\_DEVICE);

Parcelable[] uuidExtra = intent.getParcelableArrayExtra(BluetoothDevice.EXTRA\_UUID);

}

};

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

myView = new MyView(this);

turnOn = new Button(this);

setContentView(R.layout.activity\_main);

parent = (LinearLayout)findViewById(R.id.parent);

parent.addView(myView,0);

valid = false;

refX = 0;

refY = 0;

SetupUI();

findDevice();

}

Getting touch coordinates

public boolean onTouchEvent(MotionEvent ev)

{

x = ev.getX();

y = ev.getY();

if(ev.getAction() == MotionEvent.ACTION\_DOWN)

{

refX = ev.getX();

refY = ev.getY();

}

if(Math.sqrt(Math.pow(x-refX,2)+Math.pow(y-refY,2)) < 30)

small = true;

else

small = false;

if(ev.getAction() == MotionEvent.ACTION\_UP)

{

small = true;

}

return true;

This function starts device search

}

public void findDevice()

{

String lastUsedRemoteDevice = getLastUsedRemoteBTDevice();

if (lastUsedRemoteDevice != null)

{

Set<BluetoothDevice> pairedDevices = btAdapter.getBondedDevices();

for (BluetoothDevice pairedDevice : pairedDevices)

{

if(pairedDevice.getAddress().equals(lastUsedRemoteDevice)) {

remoteDevice = pairedDevice;

}

}

}

if (remoteDevice == null)

{

Toast.makeText(MainActivity.this, "Discovering", Toast.LENGTH\_SHORT).show();

if (btAdapter.startDiscovery())

{

IntentFilter filter = new IntentFilter(

BluetoothDevice.ACTION\_FOUND);

filter.addAction (BluetoothAdapter.ACTION\_DISCOVERY\_STARTED);

filter.addAction (BluetoothAdapter.ACTION\_DISCOVERY\_FINISHED);

registerReceiver(discoveryResult, filter);

}

}

}

private String getLastUsedRemoteBTDevice()

{

SharedPreferences prefs = getPreferences(MODE\_PRIVATE);

String result = prefs.getString("LAST\_REMOTE\_DEVICE\_ADDRESS", null);

return result;

}

public void SetupUI()

{

btAdapter = BluetoothAdapter.getDefaultAdapter();

if (btAdapter.isEnabled())

{

}

else

{

String actionStateChanged = BluetoothAdapter.ACTION\_STATE\_CHANGED;

String actionRequestEnable = BluetoothAdapter.ACTION\_REQUEST\_ENABLE;

startActivityForResult(new Intent(actionRequestEnable),0);

IntentFilter filter = new IntentFilter(actionStateChanged);

registerReceiver(bluetoothState, filter);

}

}

After all connections the program control transfers to this loop function in which all communication heppens

public void loop()

{

byte[] ch;

int i;

String total;

while(true)

{

try

{

if(connectedThread.mmInStream.available() != 0)

{

ch = new byte[1];

//connectedThread.mmInStream.read(ch);

i = connectedThread.mmInStream.read();

ch[0] = (byte)i;

if(ch[0] == 'a')

{

mAngle = (float)Math.atan2((y-refY),(x-refX));

if(mAngle < 0)

{

mAngle += 2\*Math.PI;

}

i = (int)(mAngle\*100);

if(small)

i += 1000;

total = String.valueOf(i);

connectedThread.write(total.getBytes());

ch[0] = '\n';

connectedThread.write(ch);

}

}

}

catch(Exception e)

{

}

}

}

This is the graphical view of the touch senses

public class MyView extends View

{

public MyView(Context context)

{

super(context);

}

@Override

protected void onDraw(Canvas canvas)

{

super.onDraw(canvas);

int radius;

radius = 50;

Paint paint = new Paint();

paint.setStyle(Paint.Style.FILL);

paint.setColor(Color.WHITE);

canvas.drawPaint(paint);

// Use Color.parseColor to define HTML colors

paint.setColor(Color.parseColor("#CD5C5C"));

canvas.drawCircle(refX, refY, radius,paint);

paint.setColor(Color.parseColor("#444444"));

canvas.drawCircle(x, y, radius/2,paint);

invalidate();

}

}

}

In remote.ino (bot side)

void remoteread()

Sends character ‘a’ to indicate data request

{

Serial1.write('a');

while(!Serial1.available())

{

}

Waits until new data comes

angle = Serial1.parseInt();

if(angle>700)

{

//Serial.print("Stall");

stall();

Stalls if 1000 is added as angle goes to from 0-628

}

else

{

angle = ((angle)/100);

magFeed(60,angle);

}

magFeed is the navigation function with pwm and angle ans arguements

//Serial.println(angle);

}

Links:

https://github.com/Deepak61900/Blueno

Email id: [deepak61900@gmail.com](mailto:deepak61900@gmail.com)

The Verdict:

This app has evolved out of a need of a cheap and very handy remote control for a bot, the interface can be made much better and user friendly with more controls and also other controls like accelerometer, magnetometer etc. A smartphone is a great piece of technological art, and it is hard to believe that so many things are inside such a small volume, this potential of Smartphone can be used in many fields, recently NASA has launched a project in which a smartphone is used as the main controller of the spaceship. This is because of the fact that smartphones give so much for a very less price.

The possibilities are endless, another version of this app reads the accelerometer reading and drives the bot accordingly.