ControlManual.java

|  |
| --- |
| package com.parrillapp.parrillapp; |
|  |  |
|  | import android.bluetooth.BluetoothAdapter; |
|  | import android.bluetooth.BluetoothDevice; |
|  | import android.bluetooth.BluetoothSocket; |
|  | import android.content.Intent; |
|  | import android.os.Bundle; |
|  | import android.os.Handler; |
|  | import android.view.KeyEvent; |
|  | import android.view.View; |
|  | import android.widget.RadioButton; |
|  | import android.widget.TextView; |
|  |  |
|  | import java.io.IOException; |
|  | import java.io.InputStream; |
|  | import java.io.OutputStream; |
|  | import java.util.UUID; |
|  |  |
|  | public class ControlManual extends UIControl { |
|  |  |
|  | public static Arduino arduino; |
|  | private ManejoSensores sensores; |
|  |  |
|  | private Handler bluetoothIn; |
|  | final int handlerState = 0; |
|  |  |
|  | private BluetoothAdapter btAdapter = null; |
|  | private BluetoothSocket btSocket = null; |
|  | private StringBuilder recDataString = new StringBuilder(); |
|  | private static String dirMac = null; |
|  |  |
|  | public static HiloConexion hiloConexion; |
|  | private static final UUID BTMODULEUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"); |
|  |  |
|  | public static boolean enviarOrdenArduino(String orden){ |
|  | switch(orden){ |
|  | case Constantes.APAGAR\_BUZZER: |
|  | if(!arduino.isEncendidoBUZZER()) |
|  | return false; |
|  | break; |
|  | case Constantes.APAGAR\_LED: |
|  | if(arduino.getEstadoLED() == 0) |
|  | return false; |
|  | break; |
|  | } |
|  | hiloConexion.enviarOrdenArduino(orden); |
|  | return true; |
|  | } |
|  |  |
|  | public static boolean enviarOrdenMotor(String orden, float altura){ |
|  | switch(orden){ |
|  | case Constantes.SUBIR\_MOTOR: |
|  | if(!arduino.puedeSubir(altura)) |
|  | return false; |
|  | break; |
|  | case Constantes.BAJAR\_MOTOR: |
|  | if(!arduino.puedeBajar(altura)) |
|  | return false; |
|  | break; |
|  | } |
|  | hiloConexion.enviarOrdenMotor(orden, altura); |
|  | return true; |
|  | } |
|  |  |
|  | @Override |
|  | protected void onResume(){ |
|  | super.onResume(); |
|  |  |
|  | tEstado.setText("Estado de "+arduino.getNombre()+":"); |
|  |  |
|  | if(this.btAdapter.isEnabled()){ |
|  | BluetoothDevice device = btAdapter.getRemoteDevice(this.dirMac); |
|  |  |
|  | try{ |
|  | this.btSocket = createBluetoothSocket(device); |
|  | }catch (IOException e){ |
|  | crearToast(getApplicationContext(), "No se pudo conectar (SOCKET\_DEVICE)."); |
|  | finish(); |
|  | } |
|  | try{ |
|  | this.btSocket.connect(); |
|  | }catch (IOException e){ |
|  | try{ |
|  | this.btSocket.close(); |
|  | }catch (IOException e2){ |
|  | crearToast(getApplicationContext(), "No se pudo conectar (SOCKET\_CLOSE)."); |
|  | finish(); |
|  | } |
|  | } |
|  |  |
|  | hiloConexion = new HiloConexion(btSocket); |
|  | hiloConexion.start(); |
|  |  |
|  | if(!this.btSocket.isConnected()){ |
|  | crearToast(getApplicationContext(), "Fallo en conexion con Parrilla."); |
|  | finish(); |
|  | }else |
|  | entrarModoManual(); |
|  |  |
|  | }else{ |
|  | Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION\_REQUEST\_ENABLE); |
|  | startActivityForResult(enableBtIntent, 1); |
|  | finish(); |
|  | } |
|  | } |
|  |  |
|  | @Override |
|  | protected void onCreate(Bundle savedInstanceState) { |
|  | super.onCreate(savedInstanceState); |
|  | setContentView(R.layout.activity\_control\_manual); |
|  |  |
|  | iniciarView(); |
|  |  |
|  | this.btAdapter = BluetoothAdapter.getDefaultAdapter(); |
|  | this.bluetoothIn = Handler\_Msg\_Hilo\_Principal(); |
|  |  |
|  | this.sensores = new ManejoSensores(getApplicationContext()); |
|  | this.arduino = new Arduino(); |
|  |  |
|  | String aux = getIntent().getStringExtra("conexion"); |
|  | this.dirMac = aux.substring(aux.length() - 17); |
|  | arduino.setNombre(aux.substring(0,aux.indexOf(this.dirMac)-1)); |
|  |  |
|  | tAltura.setOnEditorActionListener(new TextView.OnEditorActionListener() { |
|  | @Override |
|  | public boolean onEditorAction(TextView v, int actionId, KeyEvent event) { |
|  | if (actionId != 0 || event.getAction() == KeyEvent.ACTION\_DOWN) { |
|  | float altura = verificarCondiciones(tAltura.getText().toString(), Constantes.ALTURA\_MIN\_PERMITIDA, Constantes.ALTURA\_MAX\_PERMITIDA); |
|  | if(altura == -1) //si el TextView es incorrecto, uso la altura por defecto |
|  | sensores.setAltura(Constantes.ALTURA\_DEFECTO); |
|  | else |
|  | sensores.setAltura(altura); |
|  | //} |
|  | } else { |
|  | return false; |
|  | } |
|  | return true; |
|  | } |
|  | }); |
|  | } |
|  |  |
|  | @Override |
|  | //Cuando se ejecuta el evento onPause se cierra el socket Bluetooth, para no estar recibiendo datos |
|  | public void onPause() { |
|  | super.onPause(); |
|  | sensores.detener(); |
|  | if(btSocket != null && btSocket.isConnected()){ |
|  | salirModoManual(); |
|  | try { |
|  | btSocket.close(); |
|  | } catch (IOException e2) { |
|  | crearToast(getApplicationContext(), "Fallo de Conexion (SOCKET\_CLOSE)"); |
|  | finish(); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | private void entrarModoManual(){ |
|  | hiloConexion.enviarOrdenArduino(Constantes.ENTRAR\_MODO\_MANUAL); |
|  | sensores.iniciar(); |
|  | } |
|  |  |
|  | private void salirModoManual(){ |
|  | hiloConexion.enviarOrdenArduino(Constantes.ENTRAR\_MODO\_AUTOMATICO); |
|  | sensores.detener(); |
|  | } |
|  |  |
|  | public void onRadioButtonClicked(View view){ |
|  | boolean checked = ((RadioButton) view).isChecked(); |
|  | switch(view.getId()) { |
|  | case R.id.rLuzBaja: |
|  | if(checked) |
|  | this.sensores.setLuzBaja(); |
|  | break; |
|  | case R.id.rLuzMedia: |
|  | if(checked) |
|  | this.sensores.setLuzMedia(); |
|  | break; |
|  | case R.id.rLuzAlta: |
|  | if(checked) |
|  | this.sensores.setLuzAlta(); |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Valida los parametros recibidos y si son correctos y hay conexion por bluetooth, los envia. |
|  | \*/ |
|  | public void enviarTemp(View view){ |
|  | float tempMin; |
|  | float tempMax; |
|  | int enviable = 0; |
|  |  |
|  | tempMin = verificarCondiciones( tMin.getText().toString(), Constantes.TEMP\_MIN\_PERMITIDA, Constantes.TEMP\_MAX\_PERMITIDA ); |
|  | tempMax = verificarCondiciones( tMax.getText().toString(), Constantes.TEMP\_MIN\_PERMITIDA, Constantes.TEMP\_MAX\_PERMITIDA ); |
|  |  |
|  | if(tempMin == -1) |
|  | enviable = 1; |
|  |  |
|  | if(tempMax == -1) |
|  | if(enviable == 1) |
|  | enviable = 3; |
|  | else |
|  | enviable = 2; |
|  |  |
|  | if(enviable == 0) |
|  | if(tempMin >= tempMax) |
|  | enviable = 3; |
|  |  |
|  | switch (enviable){ |
|  | case 0: |
|  | this.hiloConexion.enviarTemperaturasArduino(tempMin, tempMax); |
|  |  |
|  | break; |
|  | case 1: crearToast(getApplicationContext(), "Temperatura Minima Incorrecta."); |
|  | break; |
|  | case 2: crearToast(getApplicationContext(), "Temperatura Maxima Incorrecta."); |
|  | break; |
|  | case 3: crearToast(getApplicationContext(), "Temperaturas Incorrectas."); |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | ///////////////////////////////////////////////////////////////// |
|  | private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws IOException { |
|  | return device.createRfcommSocketToServiceRecord(BTMODULEUUID); |
|  | } |
|  |  |
|  | private boolean esNumero(String s){ |
|  | if(s.isEmpty()) |
|  | return false; |
|  | for(int i = 0; i < s.length(); i++) { |
|  | if(Character.digit(s.charAt(i),10) < 0) |
|  | return false; |
|  | } |
|  | return true; |
|  | } |
|  |  |
|  | private Handler Handler\_Msg\_Hilo\_Principal () |
|  | { |
|  | return new Handler() { |
|  | public void handleMessage(android.os.Message msg) |
|  | { |
|  | //si se recibio un msj del hilo secundario |
|  | if (msg.what == handlerState) { |
|  |  |
|  | //voy concatenando el msj |
|  | String readMessage = (String) msg.obj; |
|  | recDataString.append(readMessage); |
|  |  |
|  | int endOfLineIndex = recDataString.indexOf("\n"); |
|  |  |
|  | if (endOfLineIndex > 0) { |
|  | String dataIn = recDataString.substring(0, 1); |
|  |  |
|  | switch (dataIn) { |
|  |  |
|  | case Constantes.ENVIO\_ALTURA: |
|  | if(esNumero(recDataString.substring(1, endOfLineIndex))){ |
|  | tAlturaActual.setText(recDataString.substring(1, endOfLineIndex)+" cm"); |
|  | arduino.setAlturaParrila(Integer.parseInt(recDataString.substring(1, endOfLineIndex))); |
|  | } |
|  | break; |
|  |  |
|  | case Constantes.ENVIO\_TEMPERATURA: |
|  | if(esNumero(recDataString.substring(1, endOfLineIndex))) { |
|  | tTemperatura.setText(recDataString.substring(1, endOfLineIndex) + " Cº"); |
|  | arduino.setTemperatura(Integer.parseInt(recDataString.substring(1, endOfLineIndex))); |
|  | } |
|  | break; |
|  |  |
|  | case Constantes.AVISO\_TEMPERATURA://W |
|  | if(recDataString.substring(1, endOfLineIndex).equals("1")) { |
|  | tEstadoAltura.setText("Elevada"); |
|  | arduino.setTemperaturaAlta(); |
|  | } |
|  | if(recDataString.substring(1, endOfLineIndex).equals("2")) { |
|  | tEstadoAltura.setText("Insuficiente"); |
|  | arduino.setTemperaturaBaja(); |
|  | } |
|  | if(recDataString.substring(1, endOfLineIndex).equals("0")) { |
|  | tEstadoAltura.setText("Correcta"); |
|  | arduino.setTemperaturaNormal(); |
|  | } |
|  |  |
|  | break; |
|  |  |
|  | case Constantes.ENVIO\_BUZZER: |
|  | if(recDataString.substring(1, endOfLineIndex).equals("1")) { |
|  | tBuzzer.setText("Encendido"); |
|  | arduino.setEncendidoBUZZER(true); |
|  | } |
|  | if(recDataString.substring(1, endOfLineIndex).equals("0")) { |
|  | tBuzzer.setText("Apagado"); |
|  | arduino.setEncendidoBUZZER(false); |
|  | } |
|  |  |
|  | break; |
|  |  |
|  | case Constantes.ENVIO\_LLUVIA: |
|  | if(recDataString.substring(1, endOfLineIndex).equals("1")) { |
|  | tLluvia.setText("Detectada"); |
|  | arduino.setHayLluvia(true); |
|  | } |
|  |  |
|  | if(recDataString.substring(1, endOfLineIndex).equals("0")) { |
|  | tLluvia.setText("No Detectada"); |
|  | arduino.setHayLluvia(false); |
|  | } |
|  |  |
|  | break; |
|  |  |
|  | case Constantes.ENVIO\_PROXIMIDAD: |
|  | if(recDataString.substring(1, endOfLineIndex).equals("1")) { |
|  | tProximidad.setText("Detectada"); |
|  | arduino.setHayProximidad(true); |
|  | } |
|  |  |
|  | if(recDataString.substring(1, endOfLineIndex).equals("0")) { |
|  | tProximidad.setText("No Detectada"); |
|  | arduino.setHayProximidad(false); |
|  | } |
|  |  |
|  | break; |
|  |  |
|  | case Constantes.ENVIO\_LED: |
|  | if(recDataString.substring(1, endOfLineIndex).equals("1")) { |
|  | tLED.setText("Rojo"); |
|  | arduino.setLEDRojo(); |
|  | } |
|  | if(recDataString.substring(1, endOfLineIndex).equals("2")) { |
|  | tLED.setText("Azul"); |
|  | arduino.setLEDAzul(); |
|  | } |
|  | if(recDataString.substring(1, endOfLineIndex).equals("0")) { |
|  | tLED.setText("Apagado"); |
|  | arduino.setLEDApagado(); |
|  | } |
|  |  |
|  | break; |
|  | } |
|  | recDataString.delete(0, recDataString.length()); |
|  | } |
|  | } |
|  | } |
|  | }; |
|  | } |
|  |  |
|  | private class HiloConexion extends Thread { |
|  | private final InputStream mmInStream; |
|  | private final OutputStream mmOutStream; |
|  |  |
|  | public HiloConexion(BluetoothSocket socket) { |
|  |  |
|  | InputStream tmpIn = null; |
|  | OutputStream tmpOut = null; |
|  |  |
|  | try { |
|  | tmpIn = socket.getInputStream(); |
|  | tmpOut = socket.getOutputStream(); |
|  | } catch (IOException e) { } |
|  |  |
|  | mmInStream = tmpIn; |
|  | mmOutStream = tmpOut; |
|  | } |
|  |  |
|  | public void run() { |
|  | byte[] buffer = new byte[256]; |
|  | int bytes; |
|  |  |
|  | while (true) { |
|  | try { |
|  | //se leen los datos del Bluetooth |
|  | bytes = mmInStream.read(buffer); |
|  | String readMessage = new String(buffer, 0, bytes); |
|  | bluetoothIn.obtainMessage(handlerState, bytes, -1, readMessage).sendToTarget(); |
|  | } catch (IOException e) { |
|  | break; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | public void write(String input) { |
|  | String aux = input.concat("\n"); |
|  | byte[] msgBuffer = aux.getBytes(); |
|  | try { |
|  | mmOutStream.write(msgBuffer); |
|  | } catch (IOException e) { |
|  | crearToast(getApplicationContext(), "La conexion fallo (WRITE\_THREAD)."); |
|  | } |
|  | } |
|  |  |
|  | public void enviarTemperaturasArduino(float tMin, float tMax){ write(Constantes.ENVIO\_TEMPERATURA+tMin+"\_"+tMax); } |
|  |  |
|  | public void enviarOrdenArduino(String orden){ |
|  | write(orden); |
|  | } |
|  |  |
|  | public void enviarOrdenMotor(String orden, float altura){ |
|  | write(orden+altura); |
|  | } |
|  | } |
|  | } |