ManejoSensores.java

|  |
| --- |
| package com.parrillapp.parrillapp; |
|  |  |
|  |  |
|  | import android.content.Context; |
|  | import android.hardware.Sensor; |
|  | import android.hardware.SensorEvent; |
|  | import android.hardware.SensorEventListener; |
|  | import android.hardware.SensorManager; |
|  |  |
|  | import static android.content.Context.SENSOR\_SERVICE; |
|  |  |
|  | public class ManejoSensores implements SensorEventListener{ |
|  |  |
|  | private float altura; |
|  | private float pesoAnt; |
|  | private float proxAnt; |
|  | private float luzAnt; |
|  | public int luz\_permitida\_min; |
|  | public int luz\_permitida\_max; |
|  | public int luz\_restablece\_min; |
|  | public int luz\_restablece\_max; |
|  | private boolean estaHorizontal = true; |
|  | public Context contexto; |
|  |  |
|  | private Sensor sensorAc, sensorLuz, sensorProx; |
|  | private SensorManager sensorManager; |
|  |  |
|  | public void setAltura(float altura){ |
|  | this.altura = altura; |
|  | } |
|  |  |
|  | public ManejoSensores(Context c){ |
|  |  |
|  | this.altura = Constantes.ALTURA\_DEFECTO; |
|  | this.contexto = c; |
|  | this.pesoAnt = 0; |
|  | this.proxAnt = 0; |
|  | this.luzAnt = 0; |
|  | this.luz\_permitida\_min = Constantes.LUZ\_PERMITIDA\_MIN\_MEDIA; |
|  | this.luz\_permitida\_max = Constantes.LUZ\_PERMITIDA\_MAX\_MEDIA; |
|  | this.luz\_restablece\_min = Constantes.LUZ\_RESTABLECER\_MIN\_MEDIA; |
|  | this.luz\_restablece\_max = Constantes.LUZ\_RESTABLECER\_MAX\_MEDIA; |
|  |  |
|  | sensorManager = (SensorManager) this.contexto.getSystemService(SENSOR\_SERVICE); |
|  | sensorAc = sensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER); |
|  | sensorLuz = sensorManager.getDefaultSensor(Sensor.TYPE\_LIGHT); |
|  | sensorProx = sensorManager.getDefaultSensor(Sensor.TYPE\_PROXIMITY); |
|  |  |
|  | } |
|  |  |
|  | public void iniciar(){ |
|  | sensorManager.registerListener(this, sensorAc, SensorManager.SENSOR\_DELAY\_NORMAL); |
|  | sensorManager.registerListener(this, sensorLuz, SensorManager.SENSOR\_DELAY\_NORMAL); |
|  | sensorManager.registerListener(this, sensorProx, SensorManager.SENSOR\_DELAY\_NORMAL); |
|  | } |
|  |  |
|  | public void detener(){ sensorManager.unregisterListener(this); } |
|  |  |
|  | public void setLuzBaja(){ |
|  | this.luz\_permitida\_min = Constantes.LUZ\_PERMITIDA\_MIN\_BAJA; |
|  | this.luz\_permitida\_max = Constantes.LUZ\_PERMITIDA\_MAX\_BAJA; |
|  | this.luz\_restablece\_min = Constantes.LUZ\_RESTABLECER\_MIN\_BAJA; |
|  | this.luz\_restablece\_max = Constantes.LUZ\_RESTABLECER\_MAX\_BAJA; |
|  | } |
|  |  |
|  | public void setLuzMedia(){ |
|  | this.luz\_permitida\_min = Constantes.LUZ\_PERMITIDA\_MIN\_MEDIA; |
|  | this.luz\_permitida\_max = Constantes.LUZ\_PERMITIDA\_MAX\_MEDIA; |
|  | this.luz\_restablece\_min = Constantes.LUZ\_RESTABLECER\_MIN\_MEDIA; |
|  | this.luz\_restablece\_max = Constantes.LUZ\_RESTABLECER\_MAX\_MEDIA; |
|  | } |
|  |  |
|  | public void setLuzAlta(){ |
|  | this.luz\_permitida\_min = Constantes.LUZ\_PERMITIDA\_MIN\_ALTA; |
|  | this.luz\_permitida\_max = Constantes.LUZ\_PERMITIDA\_MAX\_ALTA; |
|  | this.luz\_restablece\_min = Constantes.LUZ\_RESTABLECER\_MIN\_ALTA; |
|  | this.luz\_restablece\_max = Constantes.LUZ\_RESTABLECER\_MAX\_ALTA; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Si se detecta proximidad, envia orden de apagar Buzzer |
|  | \*/ |
|  | public boolean manejarProximidad(float actualMedido){ |
|  | if(actualMedido > Constantes.PROX\_RESTABLECER) |
|  | this.proxAnt = actualMedido; |
|  |  |
|  | if(actualMedido < Constantes.PROX\_PERMITIDO && this.proxAnt > Constantes.PROX\_RESTABLECER){ |
|  | this.proxAnt = actualMedido; |
|  | return true; |
|  | } |
|  | return false; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Si se detecta baja de luz, envia orden de apagar LED |
|  | \*/ |
|  | public boolean manejarLuz(float actualMedido){ |
|  | if(actualMedido < this.luz\_restablece\_min || actualMedido > this.luz\_restablece\_max) |
|  | this.luzAnt = actualMedido; |
|  |  |
|  | if(actualMedido > this.luz\_permitida\_min && actualMedido < this.luz\_permitida\_max && (this.luzAnt < this.luz\_permitida\_min || this.luzAnt > this.luz\_permitida\_max)){ |
|  | this.luzAnt = actualMedido; |
|  | return true; |
|  | } |
|  | return false; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Si hay shake, envia altura por BT junto a orden (que depende de la direccion del shake) |
|  | \*/ |
|  | public boolean manejarAcelerometroSubir(float actualMedido){ |
|  |  |
|  | if(actualMedido > Constantes.AC\_MIN\_RESTABLECER && actualMedido < Constantes.AC\_MAX\_RESTABLECER) |
|  | this.pesoAnt = actualMedido; |
|  |  |
|  | if(actualMedido < Constantes.AC\_PERMITIDO\_SUBIR && (this.pesoAnt > Constantes.AC\_MIN\_RESTABLECER && this.pesoAnt < Constantes.AC\_MAX\_RESTABLECER)){ |
|  | this.pesoAnt = actualMedido; |
|  | return true; |
|  | } |
|  | return false; |
|  | } |
|  |  |
|  | public boolean manejarAcelerometroBajar(float actualMedido){ |
|  |  |
|  | if(actualMedido > Constantes.AC\_MIN\_RESTABLECER && actualMedido < Constantes.AC\_MAX\_RESTABLECER) |
|  | this.pesoAnt = actualMedido; |
|  |  |
|  | if(actualMedido > Constantes.AC\_PERMITIDO\_BAJAR && (this.pesoAnt > Constantes.AC\_MIN\_RESTABLECER && this.pesoAnt < Constantes.AC\_MAX\_RESTABLECER)){ |
|  | this.pesoAnt = actualMedido; |
|  | return true; |
|  | } |
|  | return false; |
|  | } |
|  |  |
|  | @Override |
|  | public void onSensorChanged(SensorEvent event) { |
|  | //Si telefono en Vertical, uso el de luz. |
|  | //Si telefono en horizontal, uso el de proximidad. |
|  |  |
|  | if(event.sensor.getType() == Sensor.TYPE\_PROXIMITY){ |
|  | if(estaHorizontal){ |
|  |  |
|  | if(manejarProximidad(event.values[0])){ |
|  |  |
|  | ControlManual.enviarOrdenArduino(Constantes.APAGAR\_BUZZER); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | if(event.sensor.getType() == Sensor.TYPE\_LIGHT){ |
|  | if(!estaHorizontal){ |
|  | if(manejarLuz(event.values[0])){ |
|  | ControlManual.enviarOrdenArduino(Constantes.APAGAR\_LED); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | if(event.sensor.getType() == Sensor.TYPE\_ACCELEROMETER){ |
|  |  |
|  | float y = event.values[1]; |
|  | float z = event.values[2]; |
|  |  |
|  | if(z > Constantes.Z\_MAX\_H || z < Constantes.Z\_MIN\_H) { |
|  | estaHorizontal = true; |
|  | } |
|  | else{ |
|  | if(y > Constantes.Y\_MIN\_V && (z > Constantes.Z\_MIN\_V && z < Constantes.Z\_MAX\_V)){ |
|  | estaHorizontal = false; |
|  | } |
|  | } |
|  |  |
|  | if(manejarAcelerometroSubir(event.values[0])){ |
|  | ControlManual.enviarOrdenMotor(Constantes.SUBIR\_MOTOR, altura); |
|  | } |
|  | else |
|  | if(manejarAcelerometroBajar(event.values[0])){ |
|  | ControlManual.enviarOrdenMotor(Constantes.BAJAR\_MOTOR, altura); |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  | @Override |
|  | public void onAccuracyChanged(Sensor sensor, int accuracy) { |
|  | } |
|  | } |