Installation & Basic Usage

Add dependency in your gradle.

```
dependencies {
    compile files('libs/eBeamSmartPenSDK-v1.0-20180515-release.aar')
}
```

Register UartService in your manifest.

```
<service
    android:name="com.luidia.ebeam.pen.sdk.UartService"
    android:enabled="true" />
```

Import the eBeamSDK and create Smartpen Controller that is singleton object in your applicatioin context.

```
import com.luidia.ebeam.pen.sdk.EBeamSPController;

public class ExampleApplication extends android.app.Application {
    @Override
    public void onCreate() {
        super.onCreate();

        EBeamSPController.create(this);
    }
}
```

```
public class MyActivity extends Activity {
    private EBeamSPController penController;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        penController = EBeamSPController.getInstance();
    }
}
```

Connecting eBeam Smartpen via BLE

```
public void connect(BluetoothDevice btDevice) {
    final String address = btDevice.getAddress();
    final String name = btDevice.getName();
```

```
    penController.connect(name, address);
}
```

Disconnecting

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.btn_connect:
            if (penController.isPenMode()) {
                penController.disconnect();
            }
            break;
    }
}
```

Registration / Unregistration Listeners

```
public class MainActivity extends Activity implements PenEventListener, PenMessageListener {

    @Override
    protected void onResume() {
        super.onResume();

        penController.setPenMessageListener(this);
        penController.setPenEventListener(this);
    }

    @Override
    protected void onPause() {
        penController.setPenMessageListener(null);
        penController.setPenEventListener(null);

        super.onPause();
    }
}
```

Handling PenMessage

```
@Override
public void onPenMessage(int i, int i1, int i2, Object o) {
    final int what = i;

    switch (what) {
        case PenMessage.PNF_MSG_DISCONNECTED:
        case PenMessage.PNF_MSG_CONNECTED:
        case PenMessage.PNF_MSG_DI_FAIL:
            updateButtons();
```

```
            updateDeviceInfo();
            break;
    }
}
```

PenEvent Handling

```
@Override
public void onPenEvent(int i, int i1, int i2, Object o) {
    final int what = i;
    final int x = i1;
    final int y = i2;

    switch (what) {
        case PenEvent.PEN_DOWN:
        case PenEvent.PEN_HOVER:
        case PenEvent.PEN_MOVE:
        case PenEvent.PEN_UP:
            break;
    }
}
```

Calibration

See CalibrationActivity.java in example project.

```
penController.requestCalibrationChanging(calibrationPositionData[TOP_LEFT],
        calibrationPositionData[BOTTOM_RIGHT],
        new CalibrationResultCallback() {
            @Override
            public void onCalibrationCompleted() {
                runOnUiThread(() -> Toast.makeText(getApplicationContext(), "Calibration changed!",
Toast.LENGTH_SHORT).show());
                finish();
            }

            @Override
            public void onCalibrationFailed() {
                Toast.makeText(getApplicationContext(), "Calibration failed!",
Toast.LENGTH_SHORT).show();
            }
        });
```

Changing Pen Name

See ChangeNameActivity.java in example project.

```java
@Override
public void onClick(View v) {
    switch(v.getId()) {
        case R.id.btn_change:
            penController.setDeviceName(etNewName.getText().toString());
            break;
    }
}
```