# Lab 3: Extending our PD Patch and App

In this weeks lab we will extend the app that we worked with last week so that we can control two oscillators. Obviously we’ll need to edit the PD patch. However we will also need to add some new GUI widgets to our app to control the new parts of our patch.

## Setup

Download the example from Github: <https://github.com/Hydroxate/Lab03_Template>

Open the provided template in android studio.

## Edit The PD Patch

Lets create a more interesting PD patch.

1. Locate the raw folder in Android Studio.
2. Right click and locate the synth.zip in finder.
3. Unzip the archive.
4. Delete the archive.
5. Open the synth.pd file in pure data.
6. Edit the patch. Make the following changes:

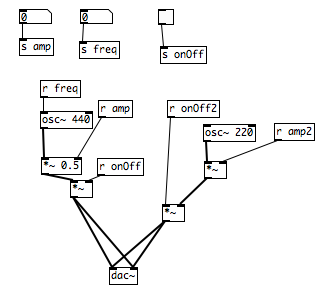
Create a new oscillator with a default frequency of 220Hz.

Create a new [\*~ ] to control its amplitude.

Create a new receive object called [r amp2] and connect it to the multiplier

Connect the output of the [\*~ ] to the existing [dac~]

Test your patch in pd using toggles and number boxes.



### Include the new patch in the project

1. Save the patch
2. Close the patch
3. Right click the synth patch, and click “Compress synth.pd”.
4. This will create a zip file called “synth.pd.zip”. Rename it “synth.zip”.
5. Delete synth.pd

## Part 1: Creating a GUI in Android Studio and Connecting it to a PD Patch.

Now we need to create some new GUI objects in our app that will communicate with the new [receive ] objects in our PD patch. This is a three step process

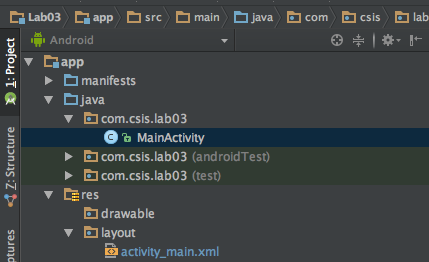
1. First we place the object graphically on our screen and set its properties such as location, color and most importantly, we give it a unique id.
2. Then we create an instance of the widget type (e.g. button, or slider or switch) in our code with a unique name and we link this to the graphical object we made in step 1.
3. Finally we specify what we want to happen when the object is interacted with. We do this by creating a piece of code linked to the instance that allows us to specify for example, “when the button is clicked send a number to pd”. This piece of code is called a listener.

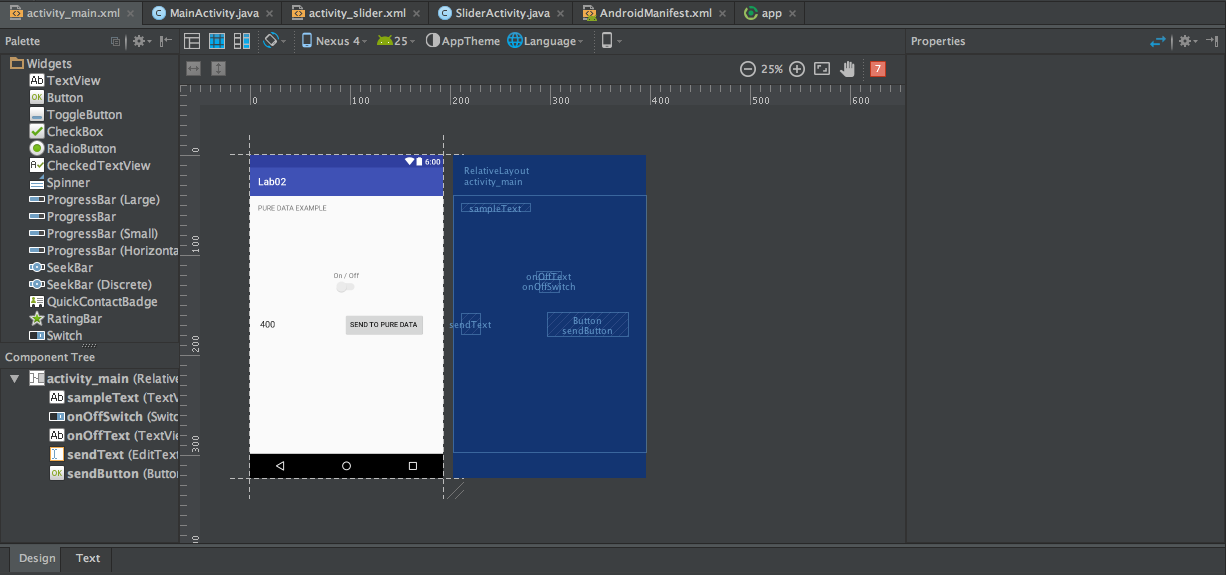
### Add a Button

We will now add a button to our GUI. When this button is clicked it will send a value of “1” to the pd patch.

### Step 1: Add it graphically

1. Click on activity\_main.xml in the project pane, it is located under res/layout.



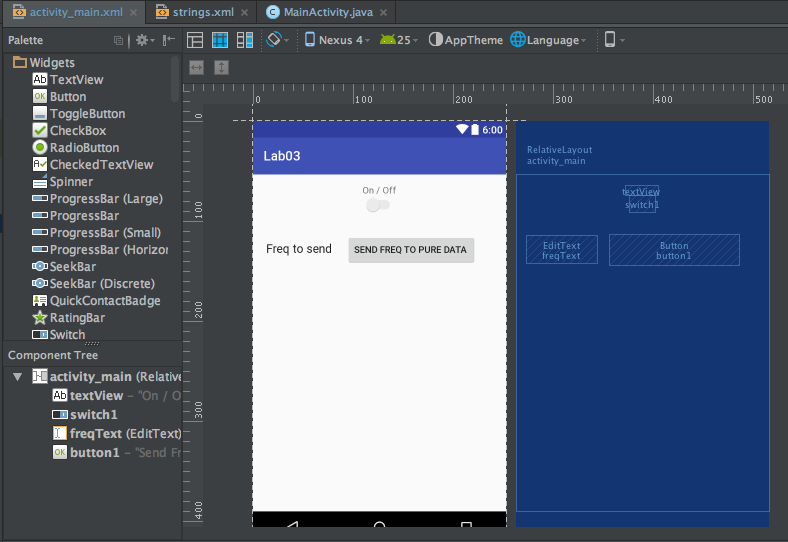


There are two different views possible when working with the xml files. One is a graphical drag and drop view called “design view”. The other is a text view where we can edit the actual xml directly. Its called “text view”. We are going to work with design view for now.

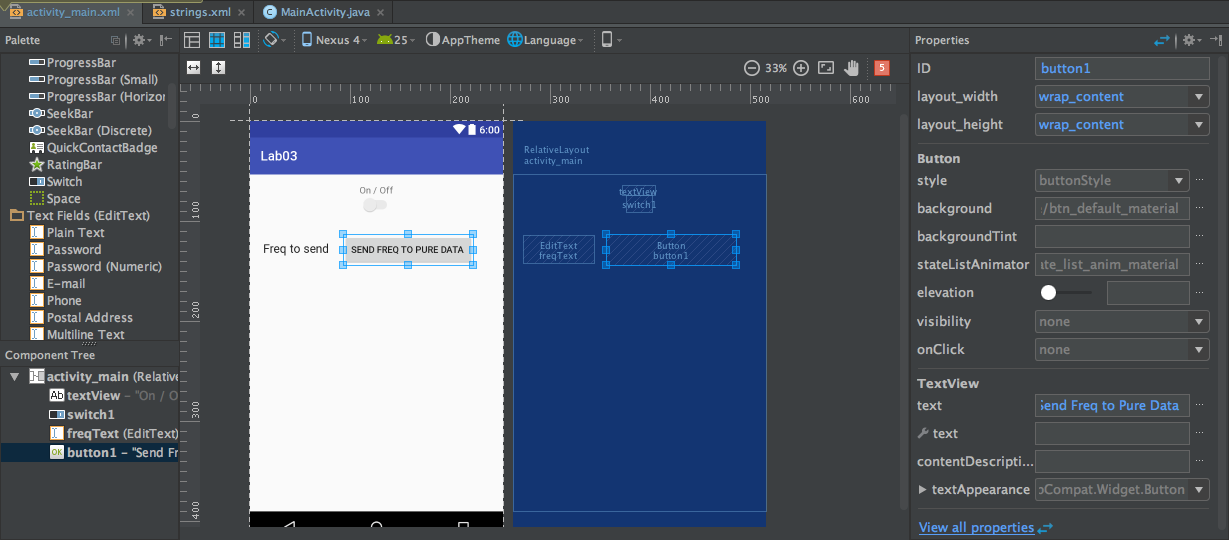
You can switch between text and design view by pressing the highlighted button in the above figure.

For now make sure you are in design view.

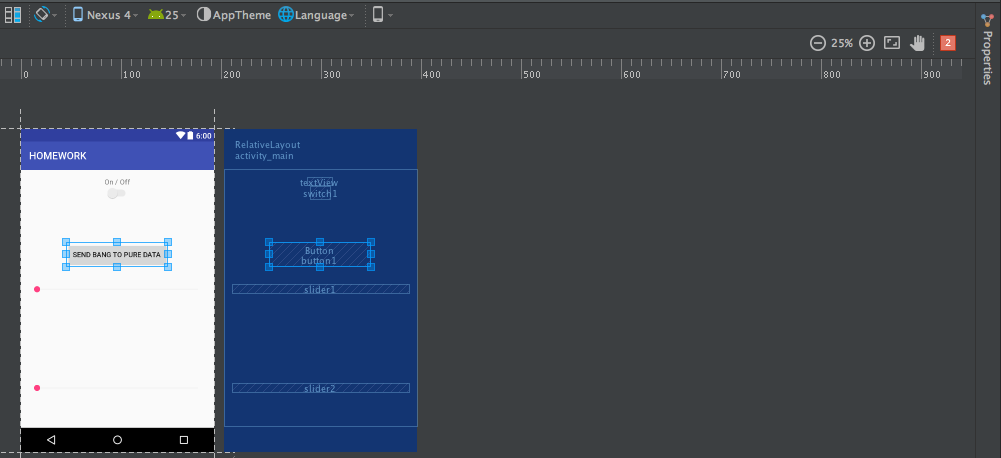
This is the design tab. This is where you can drag and drop items to create your GUI. The widgets (individual GUI objects like buttons, switches, sliders etc) are listed on the left in the palette, these can be dragged and dropped onto the screen.



1. Click on the existing “send freq to pure data’ button widget that is already in place . Its properties should now be shown in the properties tab to the right hand side.



If the properties tab is hidden, click the button to the right to display the tab.



The most important property to note is ID. This ID is what we use to refer to the widget when we are creating behaviours in our code.

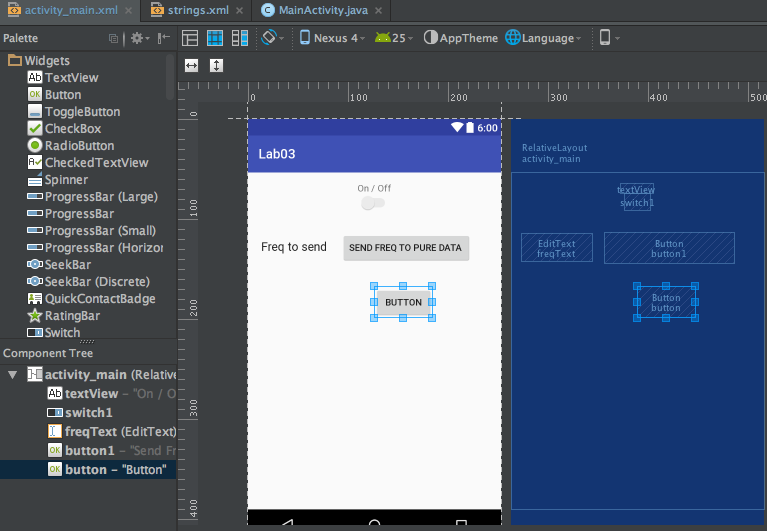
We create behaviours or functionality in code and then attach this behaviour to the widget by specifying the ID that the behaviour should be connected to.

What is the id of the existing “send freq to pure data” button?

There are also properties to set the text of the buttons, their width, and height.

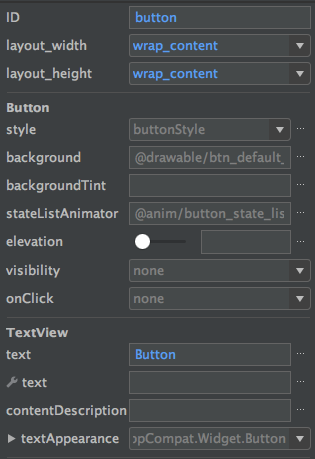
Lets add the new button to our GUI

1. Drag a button widget from the Widget palette onto the screen and place it just under the “send freq to pure data” button.



Now we need to set the properties for the button.

1. Go to the properties tab.

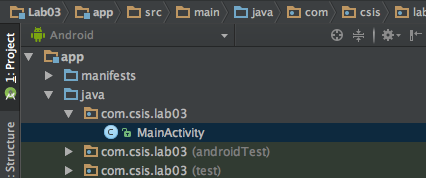


1. Rename the button ID to button2.
2. Rename the button text to Button2

### Step 2: Create an instance in code

At this stage our widget only exists as a graphical object, essentially a picture on the canvas. We have finished step 1 of our 3 step process. On to step two, “create an instance of the widget type (e.g. button, or slider or switch) in code with a unique name and link this to the graphical object we just made.”

1. Double click the MainActivity.java file in the project pane. This is located under the java folder.



1. Go to line 41. It looks like the below.



When this line of code is executed it will create an instance of the Button family called “button1” that is connected with the graphical button that has an ID of “button1”. Read that line again a few times. It’s important. What part of the line of code specifies that the instance will be called “button1”? Where in the code does it specify what graphical button we are referring to?

Answer: Well, look at the beginning of the line before the = (i.e. Button button1). This means create an instance of the Button family and name is “button1”. Now, where in the line to we create out newly created instance to the exisitng graphical object? Look at the end of the line where it says “r.id.button1”. “button1” is the ID of the “send freq to pure data” button widget that was typed in the properties section in the GUI design tab. By specifying it here we link the Button instance (called button1) to the existing graphical object with id of button1.

Lets add the code to create an instance of Button which we will call “button2” for our new button widget (which has an ID of button2).

1. Underneath the above line of code add the following:

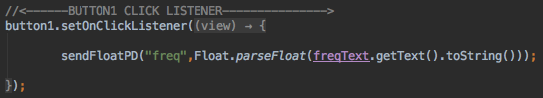
**Button button2 = (Button) findViewById(R.id.button2);**

**It should look like the below now:**



### Step 3: Add functionality to instance

Now that we have created a Button instance with a unique name (button2) we are ready for the third and final step. We must add a listener: code that detects interaction. Lets look at the listener code for the included “send freq to pure data” button. Its just below the line we just added. See the next figure.



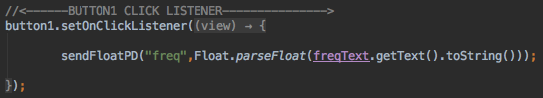
Look at the beginning of this code i.e. button1.onClickListener. We could read

this as “attach an onclick listener to the instance called “button1””.

**Things to note:**

I have code here to send floats to puredata. This sendFloatPD method can be called. It needs the name of the receiveEvent in the patch and a float!

In button1 it goes:



sendFloatPD needs two Strings: receiveEvent name and a float value to send.

So if you added a editText in the GUI for amplitude called “ampText”, you could then send it to a receiveEvent called “amp” with:

sendFloatPD(“amp”, Float.parseFloat(ampText.getText().toString()));

We want to attach an onClickListener to our instance that we named button2.

1) Create the listener for the button **button2**.

We’ll copy the format of the “send freq to pure data” button!

Type this at line 75.

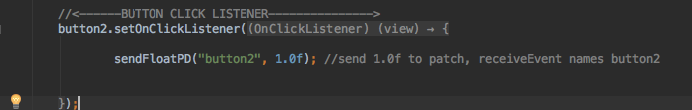
button2.setOnClickListener(newView.OnClickListener()

{  
 @Override  
 public void onClick(View view)

{  
 sendFloatPD("button2", 1.0f);   
}

});

It will look like this:



This will send a 1 to the pure data patch when the button is clicked.

Make sure there is a receiveEvent named “button2” in your pd patch to handle this float.

2) Build and run app and test!

Any errors raise your hand.

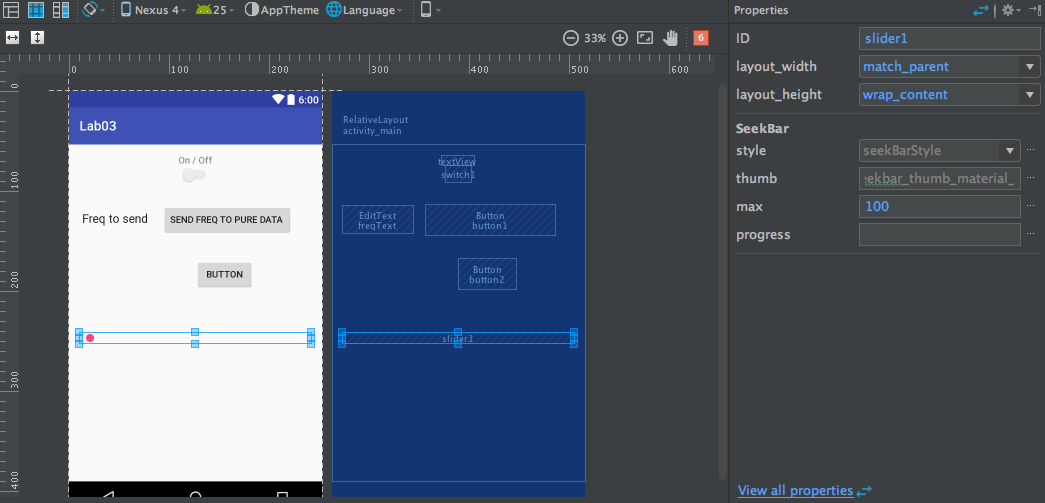
## Part 2: Adding a Slider to the GUI

You will repeat the steps for adding a button, but instead using a slider. In android this is called a **seekbar**.

1. Return to the Design view of the GUI by double clicking

activity\_main.xml.

1. Drag and drop a seekbar under the button. It will look like this:

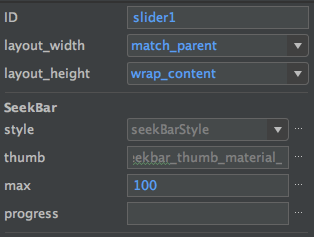


1. Edit the properties as following:

Rename seekbar ID to slider1.

Set “max” to 100.

It will look like the following:



You now need to add a listener for when the slider’s value changes.

Go to line 82 in MainActivity.java:



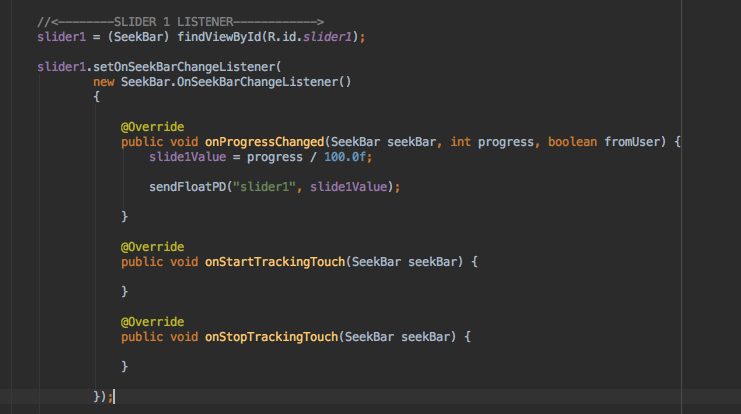
The slider has been declared for you, under this you will add the listener.

1. Add the listener code as follows:

slider1.setOnSeekBarChangeListener(  
 new SeekBar.OnSeekBarChangeListener()  
 {  
  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)

{  
 slide1Value = progress / 100.0f;  
  
 sendFloatPD("slider1", slide1Value);  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 });

It will look like the following:



5) Add a slider1 receiveEvent to your patch that will handle this data. It

will send a number between 0.0 and 1.0.

### Mandatory Homework:

**You’ll need to edit the pure data patch for more functionality.**

**Utilize the second oscillator in the patch.**

**In total, have at least 6 widgets for both oscillators including the amplitude and frequency ( get creative!).**

**Use buttons, switches, sliders and text fields to make a more complicated synthesiser!**