The Introduction of Android Evo Laser

Introduction:

Android Evo Laser is a mobile application for controlling Laser device via 3.5mm Audio Jack connected to Smartphone device .

The EvoLaser is controlled by **PWM**(pulse with modulated),meaning the power state of the laser is determined by the duration of active ON period of each cycle.

-Power On State of Laser: 18 to 84 % active ON of duty cycle.

-Power Off State of Laser: 16% and lower active ON of duty cycle.

The workflow of Android Evo Laser app

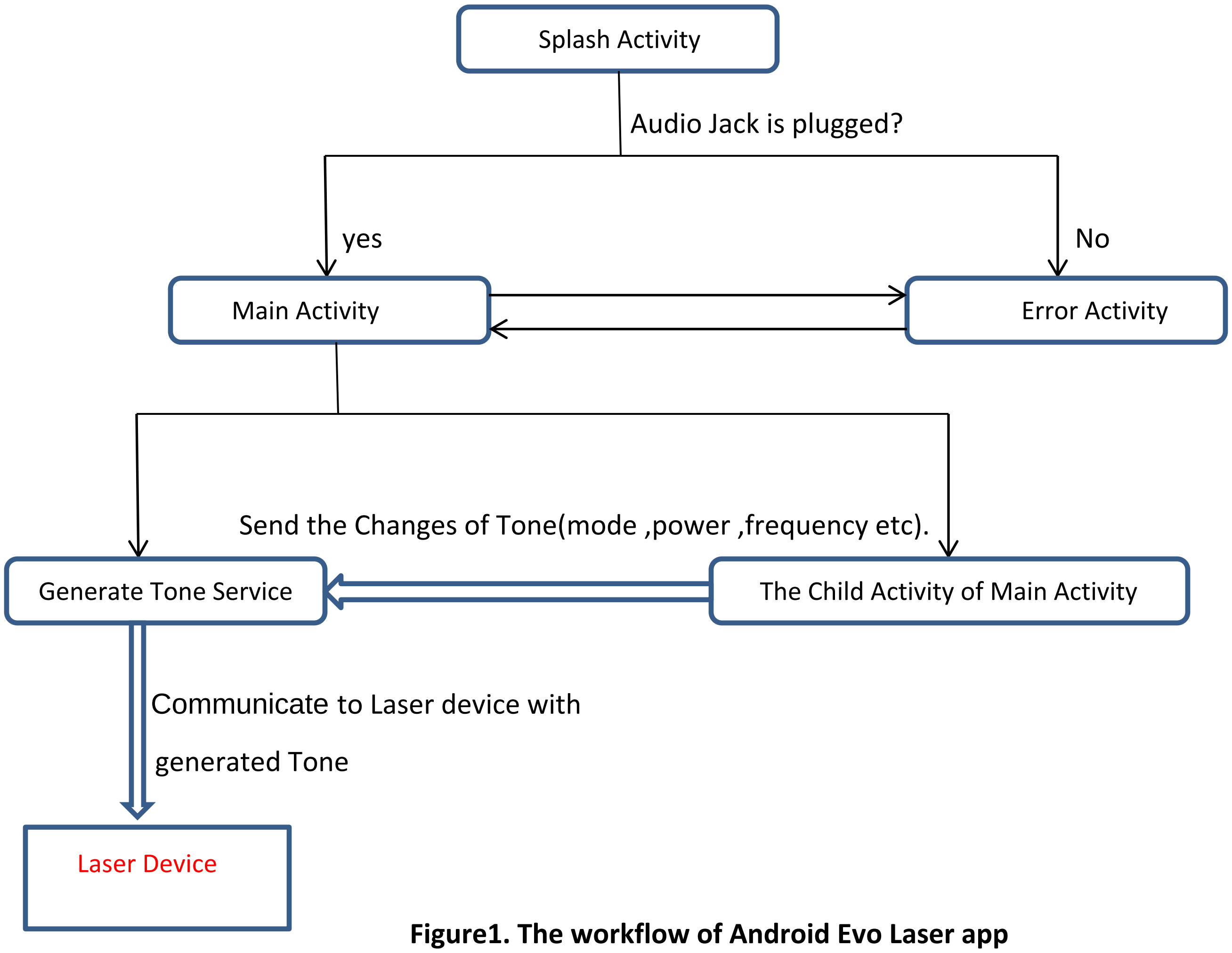

Splash Activity
Audio Jack is plugged?
yes
No
Main Activity
Error Activity
Send the Changes of Tone(mode ,power ,frequency etc).
Generate Tone Service
The Child Activity of Main Activity
Communicate to Laser device with
generated Tone
Laser Device

Figure1. The workflow of Android Evo Laser app

How does the Android Evo Laser app work?

Splash Activity:

This is the first activity that runs on launching app.

To control Laser device, it is necessary that phone device is connected with laser device via audio bus.

So when Audio Jack is plugged in, the control logic goes to Main Activity, otherwise goes to Error Activity.

Code:

```
public void gotoMainPage() {
		if(audioManager.~~isWiredHeadsetOn~~()) {
			Intent intent = new Intent(this, MainActivity.class);
			intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
			startActivity(intent);
			finish();
		}

		else{
			Intent intent = new Intent(this, ErrorActivity.class);
			intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
			startActivity(intent);
			finish();

		}

}
```

Main Activity:

If audio jack is plugged in, the control logic goes to Main Activity when app launches.

In this activity start GenerateToneService for communicating to Laser device.

Code:

```
Intent intent=new Intent(MainActivity.this,GenerateToneService.class);
			startService(intent);
```

And start child activities (Continuous Activity, Momentary Activity, Strobe Activity etc).

The Protocol for Generating Tone (ToneManager):

To control laser device, it is needed to make a tone signal.

In this protocol we refer the formulas as follows:

-Power On State of Laser: 18 to 84 % active ON of duty cycle.

-Power Off State of Laser: 16% and lower active ON of duty cycle.

-The Member Variable of **ToneManager** class.

mode:The Integer variable which indicates the tone mode. This can be one of 4 modes (Continuous,Momentary,Strobe,Fade).
power:The Double variable which controls the strength of the laser. The means of this variable is the duration of active On period of each cycle.
strobe_frequency: The Double variable which controls the frequency of laser (the number of laser flash per second) in Strobe mode.
fade_frequency: The Double variable which controls the frequency of laser (the number of laser fade in and out per second) in Fade mode.
sampling_rate: The Integer variable which define the number of sample per second. In this project we use 44100 as the sampling_rate.
sample_position: The Integer variable which use to control the laser first need to enter 100Hz loop. In this project we use 0~441 as the sample_position.
long_sample_position: The Double variable which is used in Strobe and Fade mode.
pressed:The Boolean variable which used in Momentary mode.

1.generate Tone in Continuous Mode

In Continuous Mode, Laser pen outputs continuous laser light.

Power will be vary from 0 to 1. If the user increases the power, the strength of laser light will be increased.

Code:

```
public int generate_continuous(){

   double pulsewidth = 70;
       if( this.power >= 0.01){
              pulsewidth = 80 + this.power * 290;
           }
              return 1-(2*(this.sample_position > pulsewidth?1:0));
}
```

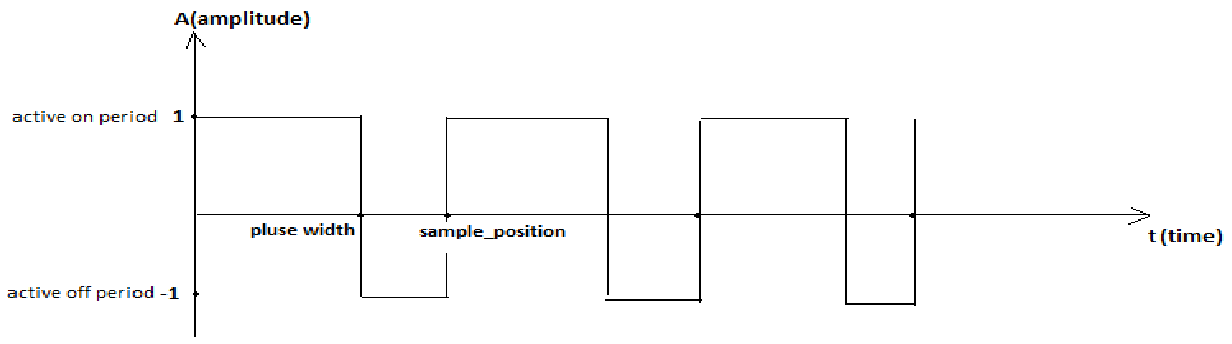


Figure 2. Generating Tone in Continuous Mode

2.generate Tone in Momentary Mode

In Momentary mode, if the user pressed is true, laser light is powered on, otherwise powered off

Code:

```
public int generate_pulse(){

   double pulsewidth = 70;
    if( this.power >= 0.01 && this.pressed){
        pulsewidth = 80 + this.power * 290;
    }
        return 1-(2*(this.sample_position > pulsewidth?1:0));
}
```

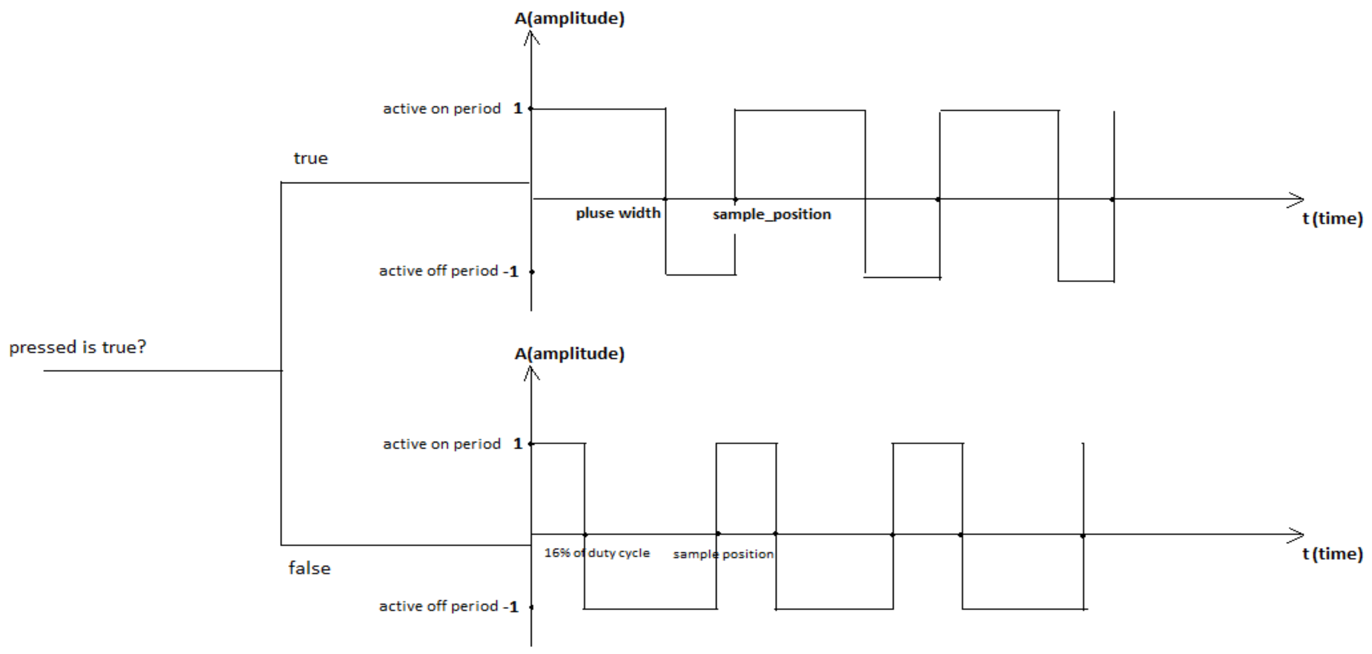


Figure 3. Generating Tone in Momentary Mode

3.generate Tone in Strobe Mode.

In Strobe mode, Laser pen outputs flash laser light.

Strobe frequency will be vary 1~10Hz. If strobe frequency is 2, laser light flash two times.

Code:

```
public int generate_strobe() {


  double pulsewidth = 70;
       if (this.power >= 0.01) {
              pulsewidth = 80 + this.power * 290;
              double period = this.sampling_rate / (this.strobe_frequency * 100);
              double half_period = period / 2;
              if (this.long_sample_position <= half_period) pulsewidth = 70;
             if (++this.long_sample_position >= period) this.long_sample_position = 0;
       }
  return 1 - (2 * (this.sample_position > pulsewidth ? 1 : 0));
}
```

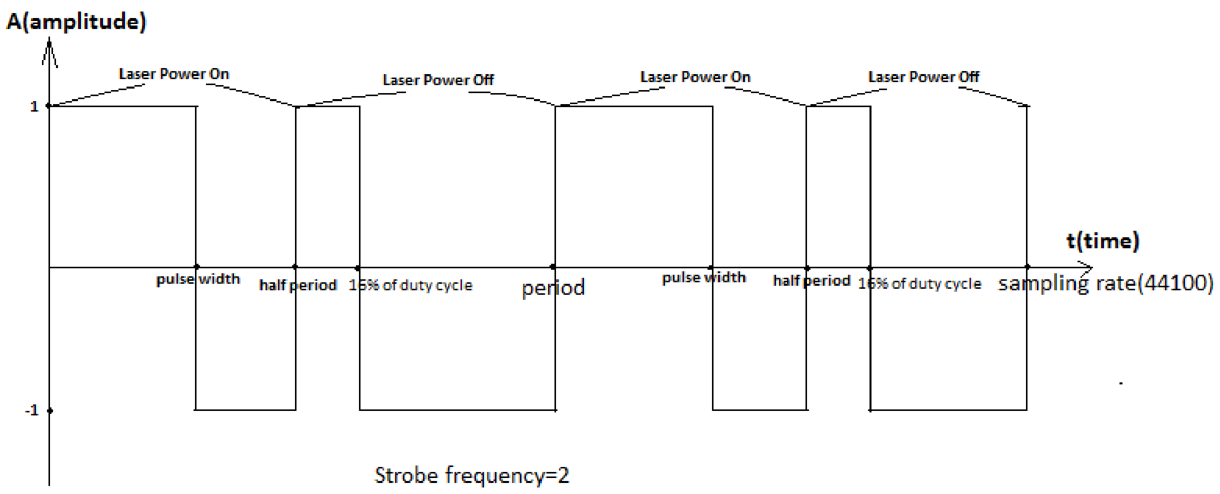


Figure4. Generating Tone in Strobe Mode

4.generate Tone in Fade Mode.

Same as Strobe mode, however in this mode don't flash laser light, only repeat fade in and out of laser light. Fade frequency will be vary 1~5Hz.

Code:

```
public int generate_fade(){
    double pulsewidth = 70;
    if( this.power >= 0.01){
          double period = this.sampling_rate / (this.fade_frequency * 100);
          double half_period = period / 2;
          double  ratio  =  1  -  Math.abs(half_period  -  this.long_sample_position)  /
       half_period;
          pulsewidth = 80 + this.power * ratio * 290;
         if(++this.long_sample_position >= period) this.long_sample_position = 0;
       }
  return 1-(2*(this.sample_position > pulsewidth?1:0));
}
```

GenerateToneService:

This class works as background service to generate tone and communicates to Laser device.

This implements the communication between phone device and laser device by using AudioTrack.

AudioTrack.write(toneSignal,0,toneSignal.length);

```
AudioTrack.play();
```

If app flow runs this code, phone outputs the toneSignal to Laser Control circuit via Audio Jack Port.

Then Laser Control Circuit (Arduino circuit) receives that signal and control the Laser Device.

The important functions of this Class are as follows:

-signal_generator();

The function which generate tone according to Tone Mode.
This function returns the value of one of 4 functions (generate_continuous(), generate_pulse(), generate_strobe(), generate_fade()) according to the Tone Mode.

-generate_tone()

The function which save generated tones by signal_generator() and write to AudioTrack for communication to Laser device.

-playTone()

The function which is used to start communicate to Laser device with generated tones.

-stopTone()

The function which is used to stop communicate .

-updateTone()

The function which update tones according to the changes of Tone parameter(Tone Mode, Power,Frequency, pressed status).
This function receives the changes of tone parameter throughout the **BroadCastReceiver** from the child Activities of **MainActivity.**

Continuous Activity:

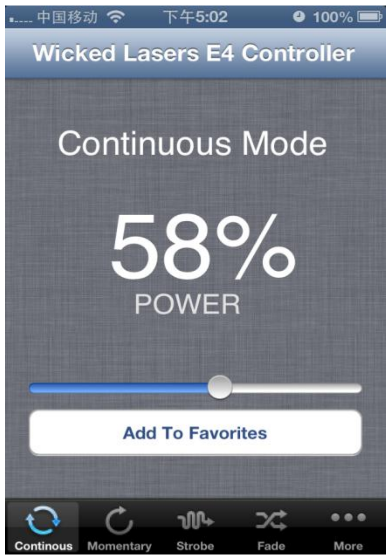


Figure 5. The UI screen of Continuous Activity

This activity makes the **GenerateToneService** work as **Continuous** Mode.
If user move the seekbar , the strength of Laser light will be vary.

Code:

```
public void changeValue(int value)
{
      powerValue.setText(String.format("%d", value)+"%");
      Intent broadcastIntent = new Intent();
      broadcastIntent.setAction(Constants.TONE_CHANGED_ACTION);
      broadcastIntent.putExtra("mode", 0);
      broadcastIntent.putExtra("power", (double)value/100.0);
      sendBroadcast(broadcastIntent);

}
```

And if user clicks “Add To Favorites” button, app saves the tone parameter to Favorite List.

Momentary Activity:

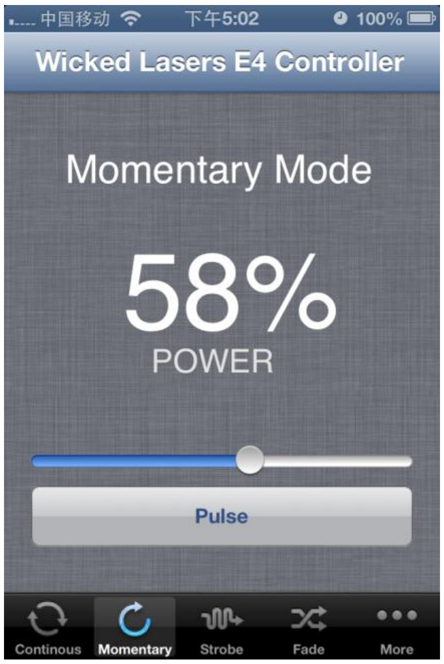


Figure 6. The UI screen of Momentary Activity

This activity make the **GenerateToneService** works as **Momentary** Mode.
If user touch on "Pulse" button, Laser light will be **Turn On**

Code:

```
public void changeValue(int value)
{
        powerValue.setText(String.format("%d", value)+"%");
        Intent broadcastIntent = new Intent();
        broadcastIntent.setAction(Constants.TONE_CHANGED_ACTION);
        broadcastIntent.putExtra("mode", 0);
        broadcastIntent.putExtra("power", (double)value/100.0);
        sendBroadcast(broadcastIntent);

 }
```

Strobe Activity:

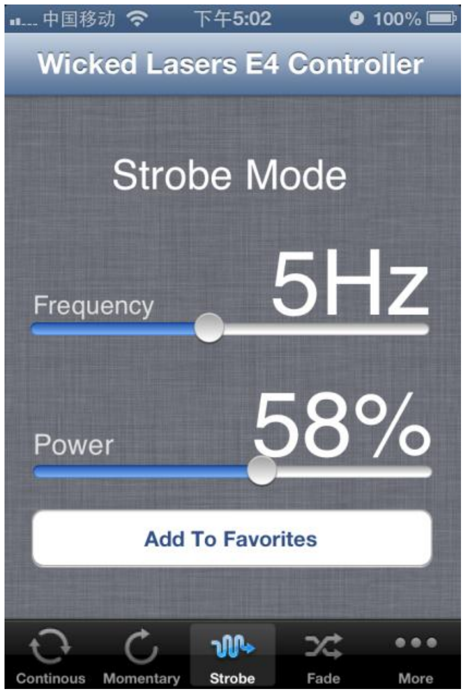


Figure 7. The UI screen of Strobe Activity

This activity make the **GenerateToneService** works as **Strobe** Mode.
If user move the seekbar ,send the changed Tone parameter(power,Frequency) to **GenerateToneService** by using **sendBroadcast** function .

Code:

```
public void changeValue()
 {
  mgr.power = (double) power.getProgress() / 100.0;
  mgr.strobe_frequency = minvalue + frequency.getProgress()
    * (maxvalue - minvalue) / 100;

  current_frequency = (int) Math.round(mgr.strobe_frequency * 100);
  current_power = power.getProgress();

  frequency_value.setText(String.format("%d", current_frequency) + "Hz");
  power_value.setText(String.format("%d", current_power) + "%");

Intent broadcastIntent = new Intent();
  broadcastIntent.setAction(Constants.TONE_CHANGED_ACTION);
  broadcastIntent.putExtra("mode", 2);
  broadcastIntent.putExtra("power",mgr.power );
  broadcastIntent.putExtra("strobe_frequency", mgr.strobe_frequency);
  sendBroadcast(broadcastIntent);
 }
```

And if user click “Add To Favorites” button, app saves the tone parameter to Favorite List.

Fade Activity:

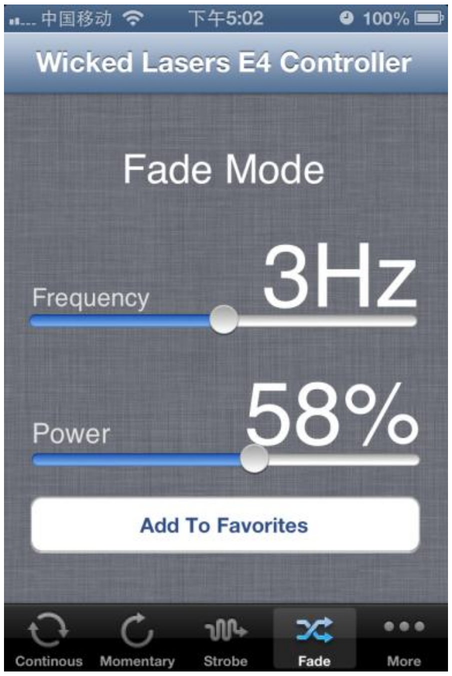


Figure 8. The UI screen of Fade Activity

This activity make the **GenerateToneService** works as **Fade** Mode.
If user move the seekbar ,send the changed Tone parameter(power,Frequency) to **GenerateToneService** by using **sendBroadcast** function .

Code:

```
public void changeValue()
 {
     mgr.power=(double)power.getProgress()/100.0;
mgr.fade_frequency=minvalue+frequency.getProgress()*(maxvalue-      minvalue)/100;

     current_frequency = (int) Math.round(mgr.fade_frequency * 100);
  current_power = power.getProgress();


  frequency_value.setText(String.format("%d", current_frequency)+"Hz");
     power_value.setText(String.format("%d", current_power)+"%");


     Intent broadcastIntent = new Intent();
  broadcastIntent.setAction(Constants.TONE_CHANGED_ACTION);
  broadcastIntent.putExtra("mode", 3);
  broadcastIntent.putExtra("power",mgr.power );
  broadcastIntent.putExtra("fade_frequency", mgr.fade_frequency);
  sendBroadcast(broadcastIntent);

 }
```

And if user click "Add To Favorites" button, app saves the tone parameter to Favorite List.

Microphone Activity:

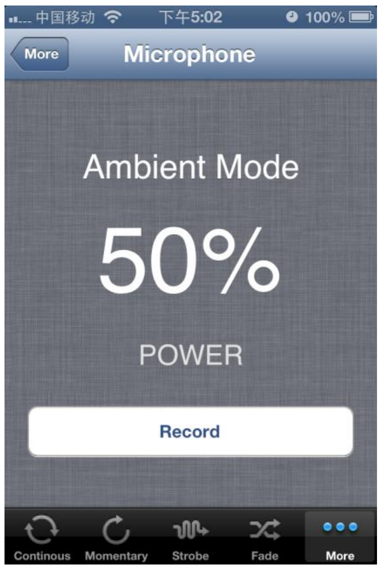
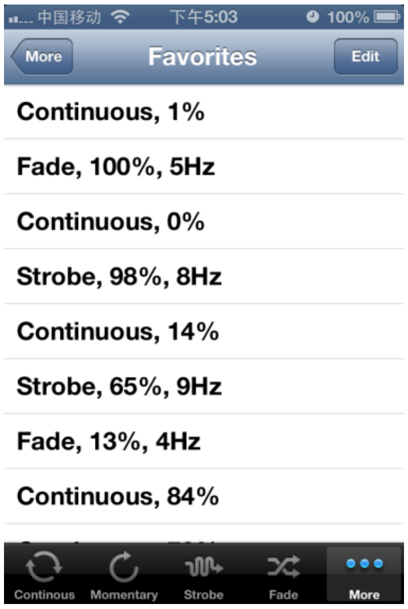


Figure 9. The UI screen of Microphone Activity

This activity will load the Tone parameters which is saved in Favorites List and send the changed Tone parameter
(mode,power,frequency) to **GenerateToneService** by using **sendBroadcast** function.

Favorite Activity:



Continuous, 1%

Fade, 100%, 5Hz

Continuous, 0%

Strobe, 98%, 8Hz

Continuous, 14%

Strobe, 65%, 9Hz

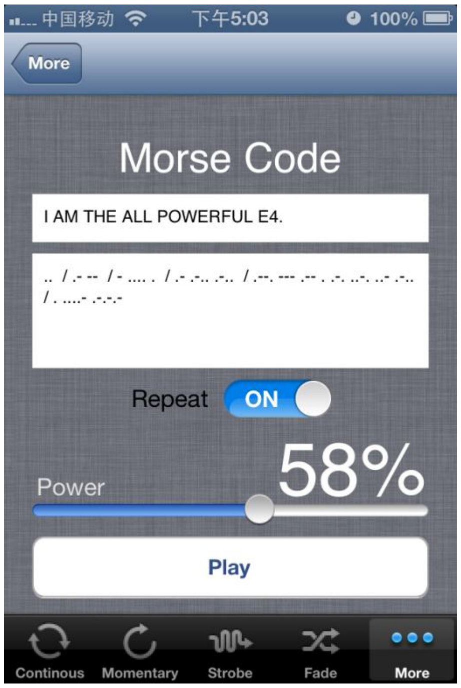
Fade, 13%, 4Hz

Continuous, 84%

Figure 10. The UI screen of Favorite Activity

This activity will load the Tone parameters which is saved in Favorites List and send the changed Tone parameter
(mode,power,frequency) to **GenerateToneService** by using **sendBroadcast** function.

Morse Activity:



Morse Code

I AM THE ALL POWERFUL E4.

.. /.- -- /- .... . /.- .-.. .-.. /.--. --- .-- . .-. ..-. ..- .-..
/ . ....- .-.-.-

58%

Figure 11. The UI screen of Morse Activity

This activity generates Morse Code from Plain Text and send generated morse code to Laser device.

> Morse code “.” corresponds to the 150ms duration of Power On in Momentary Mode.
> Morse code “-” corresponds to the 450ms duration of Power On in Momentary Mode.

First convert plain Text to Morse code by using **`textToMorseCode()`** `function`,
And generate the command List from Morse code by using generateCommandList().
The means of command List are as follows.
If Morse code is “..- -“ , command List is (150ms Power On, 150ms Power Off, 150ms Power On, 150ms Power Off, 450ms Power On, 150ms Power Off, 450ms Power On, 150ms Power Off).

After generating command List, the logic sends the command of list one by one to **GenerateToneService** by using **sendBroadcast** function.