The introduction of iOS Evo Laser app

Introduction:

iOS **Evo Laser** is a iPhone application for Controlling Laser device via 3.5mm Audio Port.

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

The Protocol for controlling Laser device are as follows:

-16% and lower of duty cycle for laser off

-18 to 84 % of duty cycle for laser on

The workflow of iOS Evo Laser application

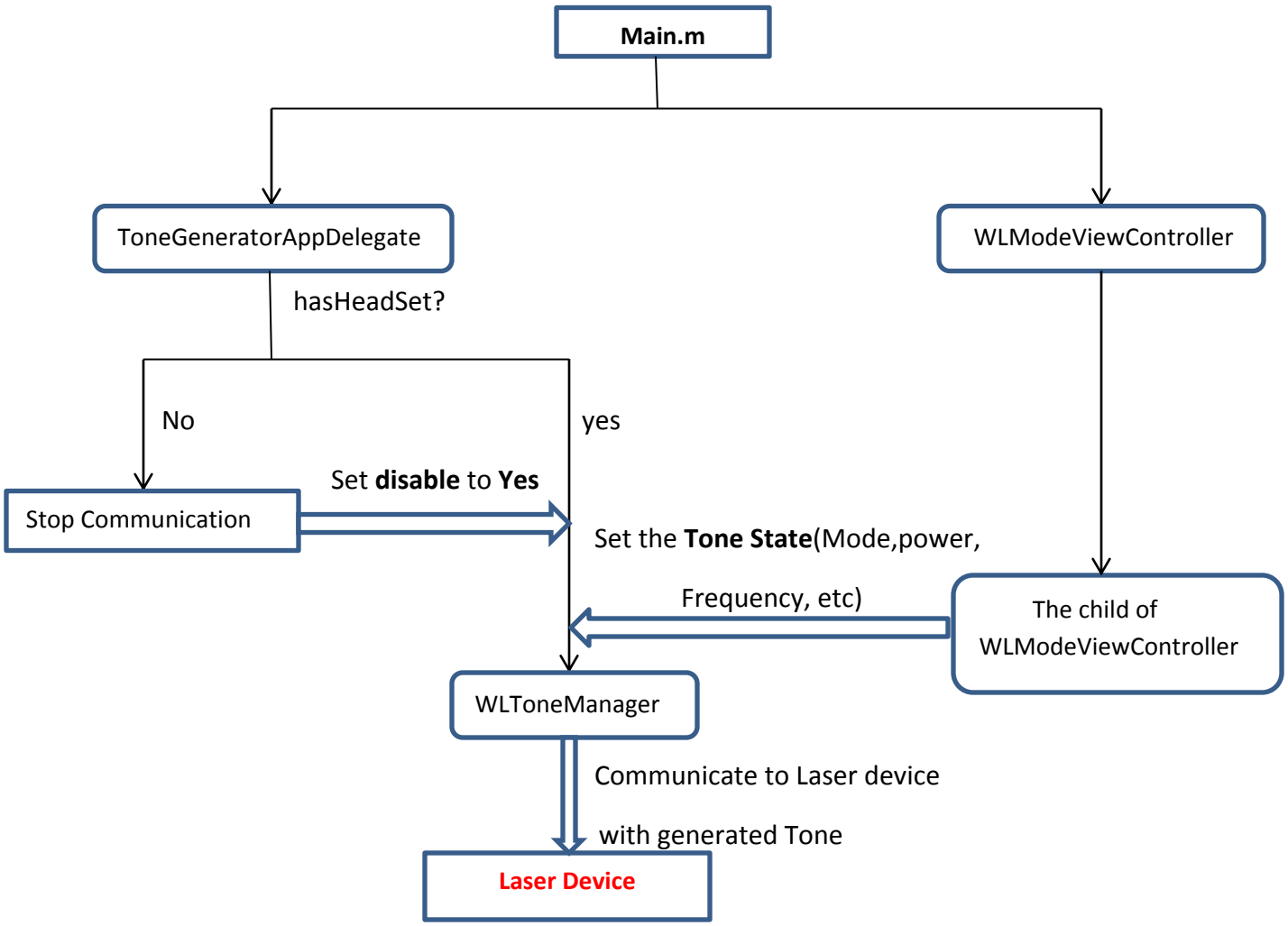

Figure1. Work flow of iOS Evo Laser app

How to Generate Tone and Communicate to Laser Device?(WLToneManager)

This is the class for generating tone and communication to Laser Device.

It works same as **ToneManager** and **GenerateToneService** in Android **Evo Laser** app.

Generating tones in 4 modes(*Continuous, Momentary, Strobe, Fade*) are same as android

The function of generating Tone in 4 Modes:

- **The Member Variable of WLToneManager 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

disable: The Boolean variable which is used to stop communicate to laser device.

toneUnit: The AudioComponentInstance variable for Playing tone.

_fm: The WLFavoritesManager variable for save tone state to Favorite Manager and load from Favorite Manager

1.The function of generating 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:

```
int generate_continuous(void *tm){
        WLToneManager *tmanager = (WLToneManager *) tm; double pulsewidth = 70;
  if( tmanager->power >= 0.01){
    pulsewidth = 80 + tmanager->power * 290;
  }
return 1-(2*(tmanager->sample_position > pulsewidth));
}
```

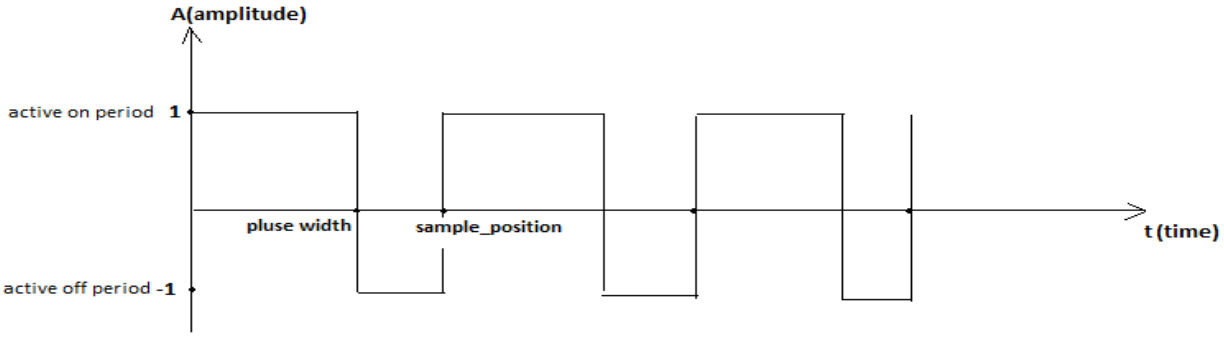

Figure 2. Generating Tone in Continuous Mode

2. The function of generating Tone in Momentary Mode

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

Code:

```
int generate_pulse(void *tm){
WLToneManager *tmanager = (WLToneManager *) tm; double pulsewidth = 70;
   if( tmanager->power >= 0.01 && tmanager->pressed){
      pulsewidth = 80 + tmanager->power * 290;
   }
return 1-(2*(tmanager->sample_position > pulsewidth));
}
```

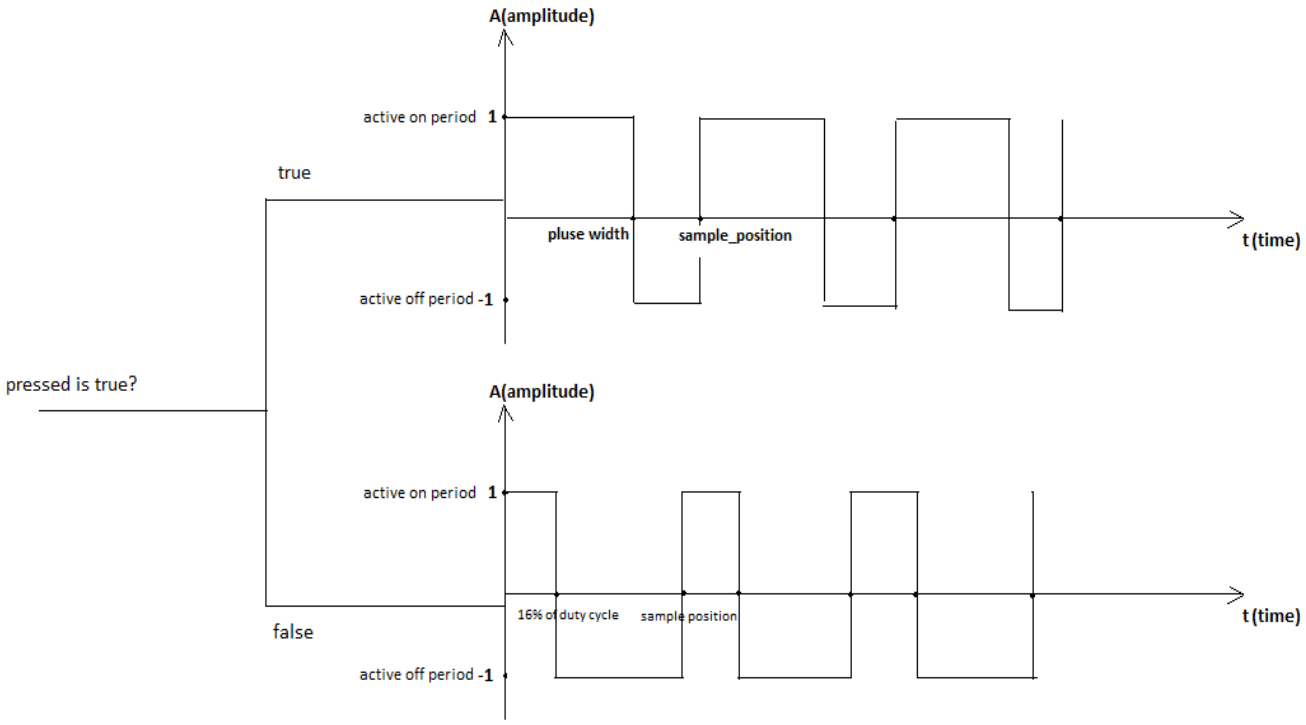

Figure 3. Generating Tone in Momentary Mode

3. The function of generating 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:

```
int generate_strobe(void *tm){
   WLToneManager *tmanager = (WLToneManager *) tm; double pulsewidth = 70;
   if( tmanager->power >= 0.01){
      pulsewidth = 80 + tmanager->power * 290;
      double period = tmanager->sampling_rate / (tmanager->strobe_frequency * 100);
      double    half_period = period / 2;
      if(tmanager->long_sample_position <=  half_period) pulsewidth = 70;
      if(++tmanager->long_sample_position >= period) tmanager->long_sample_position = 0;
   }
return 1-(2*(tmanager->sample_position > pulsewidth));
}
```

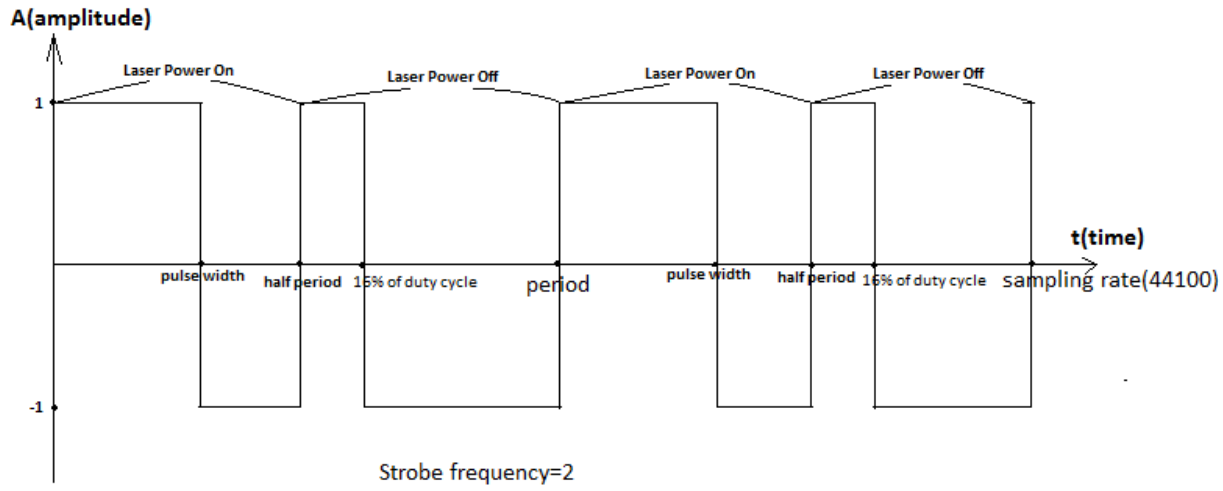

Figure4. Generating Tone in Strobe Mode

4. The function of generating 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.

```
int generate_fade(void *tm){
   WLToneManager *tmanager = (WLToneManager *) tm; double pulsewidth = 70;
   if( tmanager->power >= 0.01){
        double period = tmanager->sampling_rate / (tmanager->fade_frequency * 100);
        double half_period = period / 2;
        double ratio = 1 - abs(half_period - tmanager->long_sample_position) / half_period;
       pulsewidth = 80 + tmanager->power * ratio * 290;
      if(++tmanager->long_sample_position >= period) tmanager->long_sample_position = 0;
   }
 return 1-(2*(tmanager->sample_position > pulsewidth));
}
```

The MainFunction of *WLToneManager:*

-signal_generator();

The function for generating 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.

- **renderTone**

  This function which generate tone for communication.

  Code:

```
for (UInt32 frame = 0; frame < inNumberFrames; frame++)
{
     if(tmanager->disable)   buffer[frame] = 0;
      else
      {
         buffer[frame] = signal_generator(tmanager);
         if (++tmanager->sample_position >= 441) tmanager->sample_position -= 441;
      }
}
```

In this function if disable is set to true, write to buffer array (0 value).

Because Laser Control device did not recognize 0 value, laser pen doesn't work .

So we use **disable** field to stop communication.

-CreateToneUnit

This function configure parameters to find the default playback output unit, get the default playback output unit , create a new unit that we'll use for output, set tone rendering function on the unit (**renderTone** function) and set the format to 32 bit, single channel, floating point, linear PCM.

To see more details, refer source code.

-start

This function create tone unit for playing tone by callig **CreateToneUnit** function and start communication to laser device.

-init

This function initialize tone parameters and start communication by calling **start** function.

- saveCurrentStateToFavorites

This function save current Tone state to Favorite Manager.

Code:

```
- (BOOL) saveCurrentStateToFavorites{
        WLFavorite *f = [[WLFavorite alloc ] initWithPower: self->power strobeFrequency:
        self->strobe_frequency fadeFrequency: self->fade_frequency mode: self->mode];
        [self.favorites_manager add:f];
        [f release];
        return YES;
}
```

- restoreFavorite

This function load the Tone state from which is saved in Favorite Manager

Code:

```
- (BOOL) restoreFavorite: (WLFavorite *) favorite{
        self->power = favorite.power;
        self->strobe_frequency = favorite.strobe_frequency;
        self->fade_frequency = favorite.fade_frequency;
        self->mode = favorite.mode;
        [self selectSignalGeneratorAtIndex:self->mode];
        return YES;
}
```

The Structure of Application

WLContinousModeViewController:

Figure 5. The UI screen of WLContinousModeViewController

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

Code:

```
- (IBAction)powerSliderValueChanged:(UISlider *)slider
 {
      WLToneManager*tmanager=(WLToneManager*)[[UIApplication sharedApplication].
     delegate tone_manager];
      tmanager->power = self->power_slider.value;
      [self sync];
 }
```

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

WLMomentaryModeViewController:

Figure 6. The UI screen of WLMomentaryModeViewController

This class makes the **WLToneManager** works as **Momentary** Mode.
If user touch on "Pulse" button, Laser light will be **Turn On**

Code:

```
- (void)handleGesture:(UIGestureRecognizer *)gestureRecognizer{
        if(gestureRecognizer.state==1 || gestureRecognizer.state==3){
        WLToneManager*tmanager=(WLToneManager*)[[UIApplicationsharedApplication].
        delegate tone_manager];

        if(tmanager->pressed)   tmanager->pressed = NO;
        else   tmanager->pressed = YES;
        }
 }
```

WLStrobeModeViewController:

Figure 7. The UI screen of WLStrobeModeViewController

This class makes the **WLToneManager** works as **Strobe** Mode.
If user move the seekbar ,send the changed Tone parameter(power,Frequency) to **WLToneManager**.

Code:

```
- (IBAction)frequencySliderValueChanged:(UISlider *)slider
{
	WLToneManager *tmanager = (WLToneManager *) [[UIApplication sharedApplication].
	delegate tone_manager];
	tmanager->strobe_frequency = self->frequency_slider.value;;
	[self sync];
}

- (IBAction)powerSliderValueChanged:(UISlider *)slider
{
	WLToneManager *tmanager = (WLToneManager *) [[UIApplication sharedApplication].
	delegate tone_manager];
	tmanager->power = self->power_slider.value;;
	[self sync];
}
```

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

WLFadeModeViewController:

Figure 8. The UI screen of Fade Activity

This class makes the **WLToneManager** works as **Fade** Mode.
If user move the seekbar ,send the changed Tone parameter(power,Frequency) to **WLToneManager** .

Code:

```
- (IBAction)frequencySliderValueChanged:(UISlider *)slider
{
        WLToneManager*tmanager=(WLToneManager*)[[UIApplication  sharedApplication].
        delegate tone_manager];
        tmanager->fade_frequency = self->frequency_slider.value;
        [self sync];
}

- (IBAction)powerSliderValueChanged:(UISlider *)slider
{
        WLToneManager*tmanager=(WLToneManager*)[[UIApplication sharedApplication].
        delegate tone_manager];
        tmanager->power = self->power_slider.value;
        [self sync];
}
```

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

WLMicrophoneViewController:

Figure 9. The UI screen of WLMicrophoneViewController

This class detect the speech signal volume and change the power of **Tone** according to the input volume.

Code:

```
void RecordVolumeCallback(float volume)
{
   UITabBarController*tabcontroller=(UITabBarController*)[[UIApplication sharedApplication].
         delegate viewController];
   if([tabcontroller.selectedViewController isKindOfClass:[WLMicrophoneViewController class]])
   {
      WLToneManager*tmanager=(WLToneManager*)[[UIApplication sharedApplication].
         delegate tone_manager];
      float ajdusted_vol = volume * 10;
      ajdusted_vol = ajdusted_vol > 1? 1 : ajdusted_vol;
      tmanager->power = ajdusted_vol;
      NSLog([NSString stringWithFormat:@"power:%f",tmanager->power]);

      [tabcontroller.selectedViewController sync];
   }
}
```

volume is the input speech signal volume (0~1).

WLFavoritesViewController:

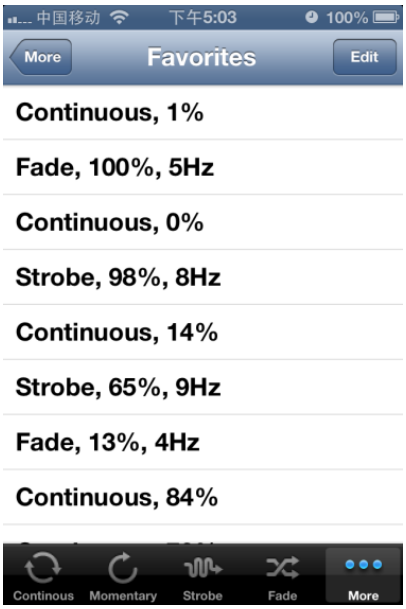

Figure 10. The UI screen of WLFavoritesViewController

This class will load the Tone parameters which is saved in Favorites List and send the changed Tone parameter
(mode,power,frequency) to **WLToneManager** .

WLMorseCodeViewController:

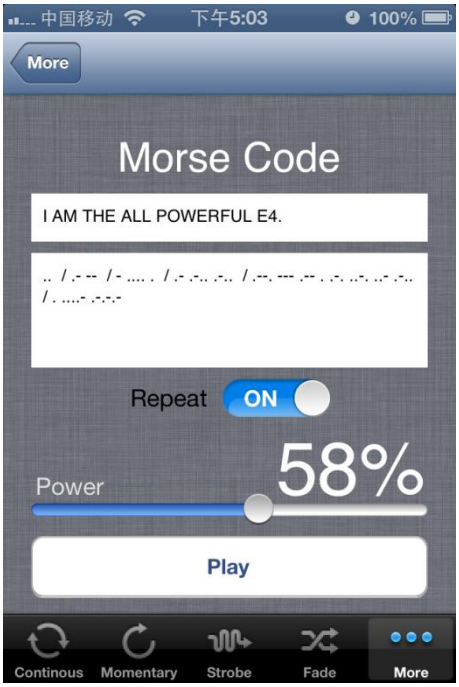

Figure 11. The UI screen of WLMorseCodeViewController

This class 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 **GetMorseCodeFrom ()** function,
And generate the command List from Morse code by using **GenerateMorseCommand ()**.
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 **WLToneManager** .