## Do something when you press Wink's button...

Up to this point, our examples just run in a quick loop over and over. Sometimes you may want your program to wait for something to happen - like button press. You can cause your program to stop and wait for a button press with the WaitForButton() function.

```
void loop(){
  WaitForButton();        //wait here till' button pressed
  EyesRed();              //make eyes red
  Wait(1);                //wait 1 second
  EyesOff();              //turn eyes back off
}
```

Wink_Ch04Button_Ex01

Wait for button to be pressed. Program continues as soon as button is pressed, turning eyes on for one second, then back off, then program loops and again waits for a button press.

By waiting for the button press, you can trigger an event (like blinking the eyes) whenever you want to. If you notice, when you hold the button down in this example, the eyes appear to stay on. They are indeed being turned off for a very short amount of time at the end of the loop, but then WaitForButton() is run again quickly following - if you're still holding the button, the light will be turned back on for another second.

Let's make the light turn back off quickly as soon as you release the button. We can do this by making the delay time much shorter. Instead of using Wait() which waits for a number of seconds, let's use the much more precise delay() function. The delay() function is a core Arduino function so it will change color when you type it. It also has a lower case "d" at the start. The delay() function works just like "Wait()" except it delays a number of milliseconds. There are 1000 milliseconds in one second, so delay(1000); works the same as Wait(1);

```
void loop(){
  WaitForButton();        //wait here till' button pressed
  EyesRed();              //make eyes red
  delay(25);              //wait 25 milliseconds
  EyesOff();              //turn eyes back off
}
```

Wink_Ch04Button_Ex02

Wait for button to be pressed. Program continues as soon as button is pressed, turning eyes on for 25 milliseconds, then back off, then program loops and again waits for a button press.

You'll notice this version responds more quickly when you release the button. The first version is good if you want the eyes to stay on for a while after a quick press, and the second version is good if you want the eyes to only turn on while you're pressing the button. Once you reach the Skill Level 2 lessons, you'll learn how to use "if" and "else" for much greater control of this.

## Adding sound...

Let's learn to control Wink's sound chirp. The functions work like this.

```
Chirp(200);     //chirp for 200 milliseconds
OnChirp();      //turn on the chirp and leave it on
OffChirp();     //turn off the chirp sound
```

} Use chirp functions to control the sound.

*Wink_Ch04Button_Ex03*

You'll quickly notice that Wink's sound chirp is fairly loud. It's a good idea to use the Chirp() function in most cases because it turns on the chirp, then turns it back off after a certain number of milliseconds. You can also use OnChirp() to turn on the chirp and leave it on. But don't forget to turn it back off in your code by calling OffChirp().

Let's add sound to our button example from above.

```
void loop(){
  WaitForButton();        //wait here till' button pressed
  EyesRed();              //make eyes red
  Chirp(25);              //chirp for 25 milliseconds
  EyesOff();              //turn eyes back off
}
```

} Chirp works just like delay() except it turns on the chirp sound during the delay.

*Wink_Ch04Button_Ex04*

## Experiment on your own...

Now you know how to control Wink's eyes, his motors, his sound, and do stuff when the button is pressed. Easy, right? This is a good time to have some fun and experiment on your own. In the next lesson we'll put these ideas together to do something fun.