Code Basics

Let's cover a few basics...

I know you're itching to make Ringo do cool stuff, so play with some of the pre-loaded behaviors for a while. Eventually, you'll want to customize how Ringo behaves. For this, you'll need to learn a few basics about the code. This guide is not intended to be a complete course in programming. There are lots of great resources for this (see the end of this section for some links!) However, we'll try our best to start you off in the right direction. Let's dive right in.

Below you'll see a box containing code. We'll use boxes like this throughout the guide to represent code you'll write or edit in the Arduino IDE. The important parts of the code in a given example are highlighted in purple.

```
/*
    Opening Heading. This area includes basic information about the program.
    It is not actually loaded onto Ringo. These are just notes for the reader.
*/

#include “RingoHardware.h”  //this makes the IDE include some outside
                            //files when loading code into Ringo

void setup(){          //the setup() function configures Ringo on startup
  HardwareBegin();     //Sets all of Ringo’s I/O pins correctly
  otherstuff();        //there may be other lines of code inside setup()
}                      //this closing curley bracket is the end
                       //of the setup() function

void loop(){             //this is where the real action happens!
  commandsToDoStuff();   //functions inside the loop() function make Ringo
  doMoreCoolStuff();     //do all his cool stuff. You will write most of
  evenMoreFun();         //your code inside the loop() function
}                        //this closing curley bracket is the end
                         //of the loop() function
```

1
SKILL
LEVEL

Code! Yay! Let's talk about the above example...

Each sketch includes code and comments. Comments are super important because they give human readers important notes about what's going on in the sketch. Comments are ignored by the Arduino IDE when it compiles the code before sending it to Ringo's brain. Comments can be written two ways.

A pair of forward slashes denotes everything else following them on the same line as a comment, and thus is ignored by the compiler.

```
thisIsCode(); //this part is a comment. This part is ignored by the compiler.
```

But what if your comment is going to require more than one line to write? There's a solution for that. You can begin a comment block with a forward slash followed by a asterisk. Then you can write as many lines of comment as you like. You then end the comment block with an asterisk followed by a forward slash. Here is an example of a comment block including the starting and ending characters.

```
/*
 This is a comment block. You can write as much as you want inside a
 code block and the comment can take up many lines. You do have to end
 a code block eventually with closing characters.
*/
```

Great. Now let's talk about the `#include “somefile”` line. This tells the compiler to copy and paste the referenced file at this point in the code. It is then compiled in as if you had actually pasted it in this place. This is a good way to include some external files (which may be really long) at the top of your code without making your main code window look cluttered.

You'll learn more about include files eventually. For now, just leave the existing `#include` lines in the examples and be careful not to edit or delete them. If they are altered, the Arduino IDE may refuse to compile your code sketch.

1
SKILL LEVEL

Now lets talk about the void setup() function. Everything between the opening curley brace { and the closing curley brace } of this function is run one time when Ringo first powers up (and also after you press and release the Reset user button on Ringo's body). This function is useful for "setting up" Ringo's brain for operation. You should include a function inside setup() called HardwareBegin(); which sets up Ringo's brain to use the electronic parts included on Ringo's body.

The setup() function for many of the Ringo examples is shown below.

```
void setup(){
  HardwareBegin();         //initialize Ringo's brain to work with his circuitry
  PlayStartChirp();        //play startup chirp and blink eyes
  delay(1000);             //wait 1 second
  NavigationBegin();       //startup the gyro and accelerometer
  RestartTimer();          //capture start time
}
```

After setup() finishes, the loop() function runs over and over again, forever (or until you turn off or reset Ringo, or until his battery runs too low and he shuts off automatically). The loop() function is where the bulk of you program usually lives.

Further learning...

You'll learn a lot as you work your way though this guide which will show you how to use all of Ringo's parts on a basic level. However, this guide will not teach you all the nuances of programming. If you'd like to learn more on your own, please check out the following links.

Learn the basics about sketches and programming technique here:
http://www.arduino.cc/en/Tutorial/Foundations

Learn the core Arduino functions. This is the "go to" reference for functions:
http://www.arduino.cc/en/Tutorial/HomePage

(continued on next page)

Code Basics

If you look around on YouTube you'll find lots of great explanations as well as awesome projects people around the world have created with Arduino. Here are a few video courses we think you'll enjoy. Remember that Ringo is basically an Arduino board without connecting headers - we've just connected the electronics directly to the microcontroller, so many of the examples you'll find online are still applicable to Ringo.

Arduino Course for Absolute Beginners
https://youtu.be/QNTaQ5qjniE?list=PLYutciIGBqC2rqIBw3wVX4LjFIcjtWjGP

Tutorial Series for Arduino by Jeremy Blum
https://youtu.be/fCxzA9_kg6s?list=PLA567CE235D39FA84

If you come across any other great resources you feel would be helpful to others, please send us a link and we'll add them to this guide.

1
SKILL
LEVEL