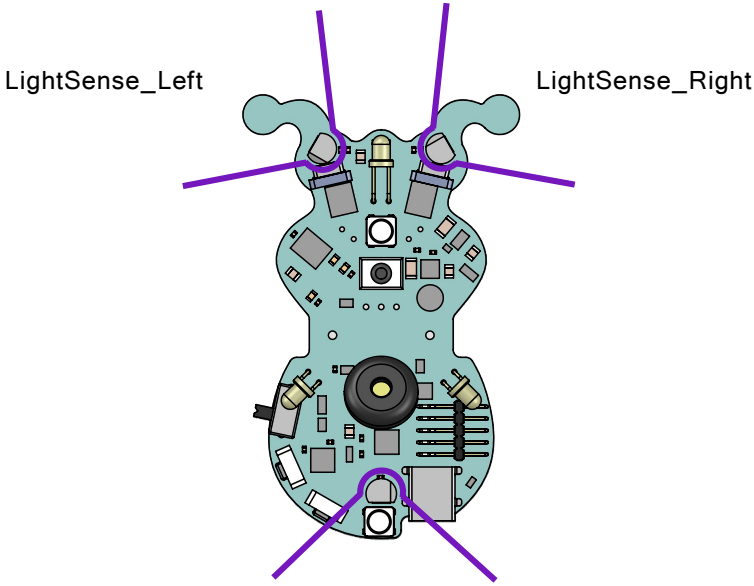
Using Ringo's Light Sensors

Ringo has some light sensors that he can use to observe his environment. These sensors can be used to control all kinds of behaviors and they are easy to use. Each sensor returns a value between 0 and 1024. The value goes up as more light reaches the sensor. Easy.

Setup...

Ringo's brain has three inputs that can read the analog light sensors. However, there are actually six light sensors on Ringo's body - there are the three ambient light sensors on the top side, and there are also three light sensors on the bottom side of Ringo to sense lines and edges. We can select which set of sensors are active by changing the `Source_Select` pin to either `HIGH` or `LOW`. Like this:

```
digitalWrite(Source_Select, HIGH);   //This selects the top light sensors
digitalWrite(Source_Select, LOW);    //This selects the bottom light sensors
```



Using Ringo's Light Sensors

The best way to see how the sensors work is to use the Arduino `Serial.print` functions to send the light sensor readings to your computer screen so you can see how they change as you point lights at Ringo, or cast shadows on his sensors. Plug Ringo into his programming adaptor, then plug the programming adaptor into your computer. Go ahead and leave the adaptor connected to Ringo during this lesson. Let's load up the following code example and see what happens.

```
#include "RingoHardware.h"  //include Ringo background functions

int sensorValue;    //declares variable "sensorValue"

void setup(){
  HardwareBegin();    //initialize Ringo's circuitry
  PlayStartChirp();  //play startup chirp and blink eyes
}

void loop(){
 digitalWrite(Source_Select, HIGH);  //select top side light sensors
 sensorValue = analogRead(LightSense_Right);  //read right light sensor
 Serial.println(sensorValue);         //print sensorValue through serial port
 Serial.println();                   //print a blank line feed
 delay(250);                         //delay 250 milliseconds
}
```

In previous examples we simply told Ringo to do something. In this example, we're going to read a value from a sensor. Ringo needs to mark a location in his memory space to remember the value when it is read from the sensor. The line `int sensorValue;` "declares" the variable "sensorValue". Once the variable is declared, the variable can be used to store a value, to perform an operation on a value (like adding a number to it), or to recall the value from it.

We then use `digitalWrite` to set the `Source_Select` pin to `HIGH`, which enables the top side light sensors. Next we tell Ringo to perform an analog read of the `LightSense_Right` pin, and place the result of that analog read into our variable called `sensorValue`. An analog read simply creates a numerical value based on a voltage created by the light sensor. As the light is increased, this voltage is increased and thus, the result of the `analogRead` will increase.

Ringo now knows the present light level of his sensor (stored in the `sensorValue` variable), but you can't see it yet because it's only stored in Ringo's brain. The line `Serial.println(sensorValue);`

tells Ringo to send this value out his serial port to the programming adaptor, and the programming adaptor sends it to your computer. Open the serial monitor window and you will see the sensor value pop up in the window. The next line `Serial.println();` tells Ringo to print a line with nothing on it. This is basically a carriage return in your serial monitor window to create a space between readings. The last line `delay(250);` causes a quarter second delay between readings.

When you run this code, you should see a value between 0 and 1024 pop up in your serial monitor window four times per second. Let this run for a few seconds, then place your hand over Ringo's right eye. You should see the value go down a bit. It should go really low if you cover Ringo in a towel or blanket that will almost completely block out the light. If you turn him toward a bright light or window, you should see this value go up.

You may notice a few interesting things. First, it is impossible to bring this value all the way down to zero. This is beyond the scope of the lesson, but the light sensor always leaks a tiny bit of current which prevents it going all the way to zero. This is normal. In a similar way, the light sensors are designed to never totally max out the input of Ringo's brain, so even in direct sunlight you won't reach all the way to 1024.

You will also notice that the value goes up and down a bit even when the light in your room isn't changing. This is normal and can be caused by several things. Most room lighting actually flashes brighter and dimmer 60 times per second, which Ringo's sensors can pick up. This can also be caused by tiny bits of electronic "noise" coming from Ringo's power supply. If Ringo is charging his battery you'll see even more noise caused by the charging system.

A final item to note, is that Ringo's top side light sensors are "logarithmic". This basically means that they respond to differences in light levels the same way your eyes do, and they can see differences in a wide range of light ranging from very dim to very bright.

Using Ringo's Light Sensors

Three sensors at once...

Reading one sensor is great, but if we read more than one sensor, Ringo can make better decisions based on what he sees. Let's try an example where we read and serial print all three sensors at once. Can you guess how we would do this?

```
#include "RingoHardware.h"  //include Ringo background functions

void setup(){
  HardwareBegin();  //initialize Ringo's circuitry
}

int sensorRight;   //declares variable "sensorRight"
int sensorLeft;    //declares variable "sensorLeft"
int sensorRear;    //declares variable "sensorRear"

void loop(){
 digitalWrite(Source_Select, HIGH);  //select top side light sensors
 sensorLeft = analogRead(LightSense_Left);  //read left light sensor
 sensorRight = analogRead(LightSense_Right);  //read right light sensor
 sensorRear = analogRead(LightSense_Rear);  //read rear light sensor

 Serial.print(sensorLeft);         //print sensorValue through serial port
 Serial.print("  ");               //print a couple blank spaces
 Serial.print(sensorRight);        //print sensorValue through serial port
 Serial.print("  ");               //print a couple blank spaces
 Serial.print(sensorRear);         //print sensorValue through serial port
 Serial.println();                 //print a blank line feed
 Serial.println();                 //print a blank line feed
 delay(250);                       //delay 250 milliseconds
}
```

Example Sketch: Ringo_Guide_LightSense_01

I think you can see what is happening here. We're using three variables now instead of just one. We read each sensor value into each variable, then use `Serial.print` to send it to your computer screen. Note in this example we're using `Serial.print()` rather than `Serial.println()`. The function `Serial.print()` basically prints information without a carriage return at the end, so all the values end up on the same line. The two `Serial.println();` functions at the end create a carriage return at the end of the output data, as well as a blank line between data sets.

Using Ringo's Light Sensors

Using top side light sensors when Ringo's eyes are lit up...

Have a close look at Ringo's eyes. You'll notice that a NeoPixel LED shines through the clear ambient light sensor. Whenever the NeoPixel is turned on (Ringo's eyes are lit up), this light from the NeoPixel is picked up by the top side light sensor directly in front of it. If you have Ringo's eyes lit up and want to read the light sensors, you will need to briefly turn off Ringo's eyes before reading the light sensors. The sensor read is very quick (about 100 microseconds per sensor) which is much too fast for your eye to see. If Ringo's NeoPixel eye lights are turned off, then the light sensors are read, then the eye lights are turned back on immediately following, it will appear that Ringo's eyes were lit up the entire time.

Try this code and see what happens:

```
#include "RingoHardware.h"  //include Ringo background functions

int sensorRight;   //declares variable "sensorRight"
int sensorLeft;    //declares variable "sensorLeft"
int sensorRear;    //declares variable "sensorRear"

void setup(){
  HardwareBegin();    //initialize Ringo's circuitry
  PlayStartChirp();   //play startup chirp and blink eyes
  digitalWrite(Source_Select, HIGH);  //select top side light sensors
}

void loop(){
 SetAllPixelsRGB(0,0,0);              //turn off all NeoPixel lights
 delay(2);                            //delay 2 milliseconds. Important!

 sensorLeft = analogRead(LightSense_Left);  //read left light sensor
 sensorRight = analogRead(LightSense_Right);  //read right light sensor
 sensorRear = analogRead(LightSense_Rear);  //read rear light sensor

 // ... continued on next page ....
```

```
// ... continued from previous page ....

SetPixelRGB( 4, 220, 30, 160);  //set pixel 4 (Right eye)
SetPixelRGB( 5, 220, 30, 160);  //set pixel 5 (Left eye)
RefreshPixels();                //turn on the pixels

Serial.print(sensorLeft);        //print sensorValue through serial port
Serial.print("  ");               //print a couple blank spaces
Serial.print(sensorRight);       //print sensorValue through serial port
Serial.print("  ");               //print a couple blank spaces
Serial.print(sensorRear);        //print sensorValue through serial port
Serial.println();                //print a blank line feed
Serial.println();                //print a blank line feed
delay(500);                      //wait a half second before looping
}
```

Now we'll talk about what is happening. You'll notice a few differences in this example.

Firstly, we placed the `digitalWrite(Source_Select, HIGH);` in the `setup()` function because in this case, it only really needs to run once (as we're not switching to read the lower sensors in this case). This is fine to do, but you probably want to include this in your main `loop()` function if you're reading both top and bottom sensors.

We first start by turning off all the pixels. The next step `delay(2);` is super important. The light sensors are amplified through a part called an "op-amp", and this particular op-amp takes about a millisecond to adjust to the lower light level after Ringo's eyes are turned off. By delaying 2 milliseconds, we can be certain that Ringo is no longer seeing any "after glow" from having his eyes turned on. As soon as this delay is complete, we can go grab the three readings we want with `analogRead()` functions. As soon as the readings are complete, we flip Ringo's eyes back on. Once the eyes are back on, we send the serial data of the readings back to your computer, and finally, we wait a half second before doing it all again.

In the next section, we'll talk about reading Ringo's bottom sensors which can be used to sense lines and edges.