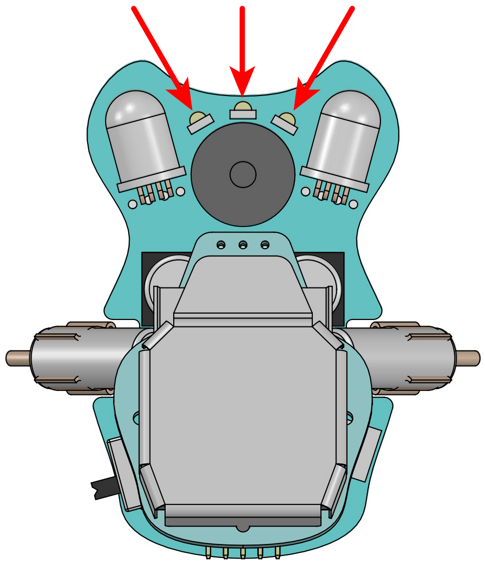
Reading Sensors...

One of the things that makes robots “smart” is their ability to sense what is going on around them. By sensing what is happening around them, they can eventually be programmed to make choices based on what they sense. (We’ll go into the “Making Decisions” part in our next lesson by learning how the “if” statement works).

Lets start by learning how to read Wink’s light sensors. He has three light sensors near his nose facing outward. They are small white rectangles with dome lenses. One of them points straight ahead, and two of them face outward to the side at forty five degree angles, with one slanted toward the left and one slanted toward the right.

They are illustrated in the drawing below.



The light sensors produce a small electric voltage that Wink’s brain can measure and turn into a number. This number represents how much light is falling on each of the sensors. The sensors can provide a reading between 0 and 1024. It should be noted that the level may not reach all the way down to 0, and it will likely not reach all the way to 1024, but it will get close. This is okay and normal.

Wink’s light sensors work well in normal room lighting. If he points directly at a window or sunlight, this will max out his sensors. The bright light won’t harm his sensors, just be aware that they won’t work as well in bright light. After you learn the examples below, experiment with Wink in different lighting to get a feel of how bright and how dim he can see.

Using the analogRead() function...

Don't let the name of this function scare you. It's easy. Let's talk about this "analog" word for a second.

A computer brain, like the one in Wink, can sense what it sees on it's electrical pins. (The "pins" are the electrical contacts around the sides of the part. In electronics, we generally call these connections "pins"). Wink's brain can sense these pins in different ways depending on what is connected to them.

In the case of Wink's light sensor, a small electrical voltage is produced by the sensor which is then connected to a pin on Wink's brain. As the light level changes, this voltage changes. It tends to rise higher when more light falls on the sensor, and it tends to drop lower when less light falls on the sensor. This kind of signal is called an "analog" signal.

To read this signal, it would make sense to name the function "analogRead", and that's exactly what they did with Arduino. The three front light sensors we will be using in this lesson are called Ambient sensors, that is to say, they sense the "ambient" light in the room. "Ambient" is like saying "the overall general amount of light around the robot".

When writing your code, the three ambient sensors are given very specific names as follows: AmbientSenseLeft, AmbientSenseCenter, AmbientSenseRight. I bet you can figure out which sensor goes with each name.

Now let's look at some code...

```
# include WinkStuff.h

void setup(){
  hardwareBegin();
  playStartChirp();
}

int sensorValue;     //declare the variable sensorValue

void loop(){
  sensorValue = analogRead(AmbientSenseCenter);  //read sensor
  Serial.println(sensorValue);                   //print value
  delay(100);                                    //delay
} //end of loop()
```

} Normal setup

} Call the analogRead() function to read the sensor, place the result in the variable called sensorValue, then use Serial.println() to print the value of the variable with an automatic new-line character, then delay 1/10th of a second before looping.

Wink_Ch08LightSensors_Ex01

When you call the analogRead() function, all you have to do is include the name of the sensor you want to read inside the parethasis. It's just that easy. The function will read the voltage on the given pin, then turn this voltage into a number between 0 and 1024, then return with this value, which it stores in the sensorValue variable. We then use Serial.println() as we did in the previous lesson to print this value to our Serial Monitor window. Run this code and point Wink at some different light sources while watching the output in the Serial Monitor window.

2
SKILL LEVEL

Wink has three of these sensors as we discussed before. How about we read them all at the same time and see what they are all reading? How would we do this? Think about it for a minute then study the next code example.

```
int left,center,right;     //declare variables

void loop(){
  left = analogRead(AmbientSenseLeft);      //read left
  center = analogRead(AmbientSenseCenter);  //read center
  right = analogRead(AmbientSenseRight);    //read right

  Serial.print(left);       //left value
  Serial.print(“\t”);       //tab key
  Serial.print(center);     //center value
  Serial.print(“\t”);       //tab key
  Serial.print(right);      //right value
  Serial.println();         //print a new line character
  delay(100);               //delay 1/10th second
} //end of loop()
```

Wink_Ch08LightSensors_Ex02

} Declare variables we will be using

} Read each sensor and put the value in the corresponding variable.

} Print the results to the Serial Monitor window, using the tab special character between each to form nice columns. Short delay at the end before the loop repeats.

Does this example look similar to what you were thinking of above? You may have noticed I slipped in a couple new concepts for you in this example that have nothing to do with reading light sensors. I figured you could handle them at this point.

First, have a look at the variable declaration at the top. I could have declaired each variable on its own line of code with three different statements, which is fine. But you can often save space in your code by declairing them all together in one statement. This doesn’t actually save memory space on the robot, the three variables still take up the same amount of memory space - but the length of your written code is shorter. This can be useful if you need to declaire a handful of variables that work together. When you use this trick, you put the type of variable at the front as always, then simply separate each variable with a comma, with the normal semi-colon at the end.

The second thing you may have noticed is that I printed the special character “\t”, which is interpreted as pressing the Tab key on your keyboard. This will cause the output in your Serial Monitor to line up in nice columns.

Get a flashlight and play around with this example a bit. Try to cover each sensor one at a time with your finger and see what happens. (Note that some LED flashlights actually pulse on and off very quickly. If you see the levels jumping around a lot, this may be the cause as the robot may sometimes read the sensor while the light is on and sometimes while it is off).

That was too easy...

I wish there was more to say about reading the light sensors, but that's really about all there is to it. I bet that was WAY easier than you thought it would be.

Let's cover a few more points so you feel challenged, then we'll be done here.

Remember in the previous lesson where we included millis() directly in the Serial.print() function? Do you think you could to do the same thing with analogRead? Of course you can. See if you can figure it out on your own before looking at the example below.

```
void loop(){
  Serial.print(analogRead(AmbientSenseLeft));    //left value
  Serial.print("\t");                            //tab key
  Serial.print(analogRead(AmbientSenseCenter)); //center value
  Serial.print("\t");                           //tab key
  Serial.print(analogRead(AmbientSenseRight));  //right value
  Serial.println();                            //new line
  delay(100);                                  //delay
} //end of loop()
```

analogRead() included directly inside the Serial.print() functions.

Wink_Ch08LightSensors_Ex03

Awesome right? You should be feeling very proud of yourself at this point. A quick glance at the above block of code and it looks pretty scary right? Before you started Lesson 1, that may have even made you think coding was hard. It turns out that computer code, like many other things, follows certain rules and patterns and once you understand these rules and patterns, the whole thing is actually fairly easy and readable.

One thing you may be struggling with at this point is writing the longer statements. Writing the statement of motors(100,100); is pretty easy, but writing something much longer like Serial.print(analogRead(AmbientSenseCenter)); is a whole different thing. If you're writing each example, the code has probably failed to compile a few times do to simple typing mistakes.

One tip is to use Copy and Paste as much as possible. Learn and use your keyboard shortcuts for these. On most computers, Ctrl-C is "copy" and Ctrl-V is "paste". When writing several lines that are similar, I find it best to highlight the first line, hit Ctrl-C, use my arrow keys to drop down, then hit Ctrl-V to paste a copy. Then you can quickly edit the important part. In the above example, the only thing that changes between the three longer statements is the word "Left", "Center", and "Right".

In a future lesson on writing your own functions, you'll learn how to turn that entire long statement into something super short, like prs(); which stands for "print right sensor".

In the next lesson we're going to learn how to make Wink do smart things by making choices. You're about to learn about the powerful "if" statement.