An Invisible Headlight...

Wink has a special light under his nose. This light is on the bottom side of the circuit board at the very front edge. It is yellowish clear and has a round dome facing toward the front. Have a look at your robot and see if you can find it.

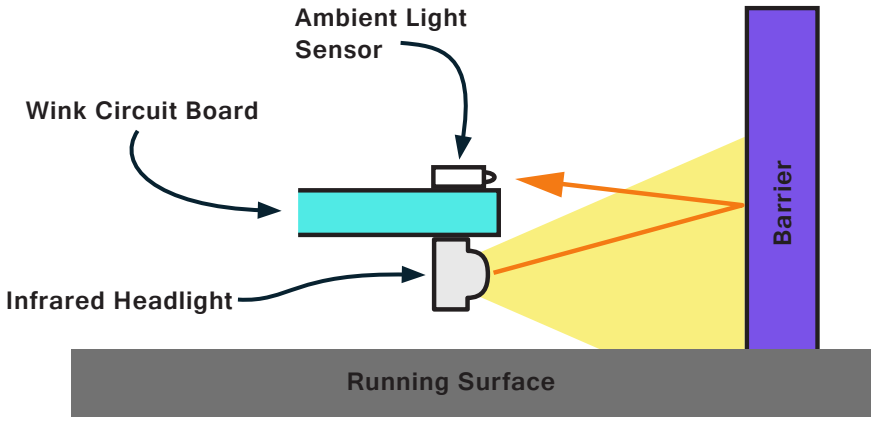
This special light can be turned on by your code. When the light turns on, you can't see it with your own eyes, because your eyes are not sensitive to the specific wavelength of light it produces, but there is indeed light coming out. Remember in the previous lessons where we used Wink's ambient light sensors to see a flashlight? These ambient light sensors can also see this "invisible" light produced by the special light under Wink's nose.

We will refer to this special light as a "Headlight" because it is placed about where you would expect a robot to have a headlight.

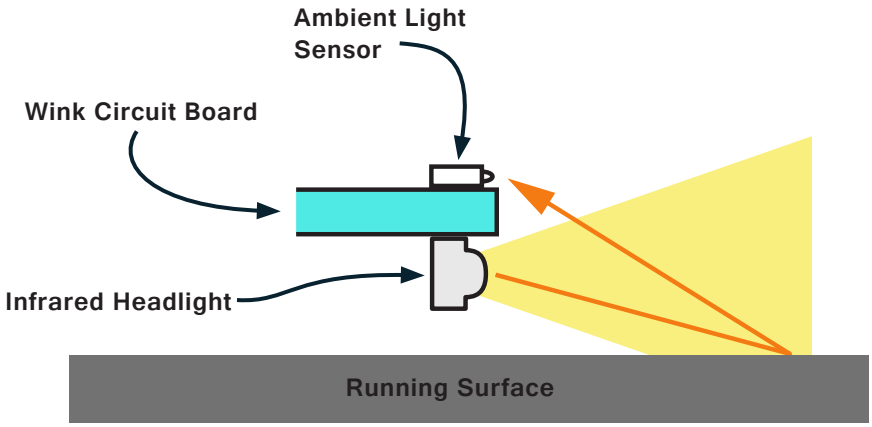
Study the picture below to get an idea of how we can use this headlight to see objects in front of Wink.

The drawing is a side view of Wink's nose. It shows Wink's headlight on the bottom side, his circuit board, and the ambient light sensor on the top side of the circuit board. You can also see the surface Wink is sitting on and a barrier in front of Wink's nose. The yellow area shows where the invisible light shines and the orange arrow shows that some of this light will reflect off the barrier and return back to Wink's ambient light sensor.

As Wink gets closer to the barrier, more and more light will reflect off the barrier to the ambient light sensor.



Wink Side View



Wink Side View

We should also consider what happens if no barrier is present. As you can see from the above picture, some of the light produced by the headlight will shine on the running surface and in most cases, a small amount of this light will reflect back off the surface and be seen by the ambient light sensor. We will need to keep this in mind as we eventually try to sense whether or not a barrier is present.

Here are a few things to remember about different barriers and surfaces.

- A barrier that is lighter color (like white) will tend to reflect much more light than a barrier that is a darker color. This makes sense because darker colors are “dark” because they absorb light waves.
- A similar thing happens with the running surface. A light color surface (like white paper) will reflect more of the light than a darker surface (like a black or brown colored desk).

Using digitalWrite...

In the next few examples, you will see a new function called digitalWrite. All of the little electrical connections on Wink's brain are wired to his different parts through electrical traces on the circuit board. digitalWrite is the primary function used in the Arduino environment to turn these connections on and off. Each connection has a tiny electrical switch inside Wink's brain and digitalWrite is the function used to turn this little switch on and off.

In electronics, the word "digital" is used to describe a switch that can be either on or off, but not somewhere in between. When the switch is turned "on" it is said to be "high". When the switch is turned "off", it is said to be "low".

Using digitalWrite is easy. You simply tell the function which connection to turn on or off, then you tell it whether to set this connection "high" to turn it on, or to set it "low" to turn if off.

The people who design electronics call the little electrical connections on electronic parts "pins". We can give these pins meaningful names in our code. The pin that controls Wink's headlight is called "Headlight".

(If you're interested, these pins are actually named in the tab called WinkHardware.h You can see them listed toward the top of the code on that tab. We'll touch on that in a later lesson).

Here is an example of how digitalWrite is used...

```
digitalWrite(Headlight, HIGH);      //turn on IR Headlight
digitalWrite(Headlight, LOW);       //turn off IR Headlight
```

} Examples of how to use digitalWrite to turn a connection on or off.

The first item included in digitalWrite is the name of the pin you would like to control. You then tell the function whether to set this pin HIGH or LOW. Notice there is a comma after "Headlight" and the word HIGH or LOW must be all capital letters.

How to sense a barrier...

As we've seen from the previous pictures, we can shine Wink's headlight and measure how much light gets reflected back to his ambient light sensors. That sounds simple enough. Let's try a quick example and see what happens.

```
int centerLight;   //declare variable

void loop(){

 digitalWrite(Headlight, HIGH);     //turn on IR Headlight
 centerLight = analogRead(AmbientSenseCenter);  //read sensor

 if (centerLight < 100)             //if centerLight < 100
  {
    motors(100,100);                //drive forward
  }
  else                              //otherwise...
  {
    motors(0,0);                    //be still
  }
} //end of loop()
```

} Use digitalWrite to turn on Headlight
} Read center light sensor

} If the light reflected is less than 100, drive forward. Otherwise, stop the motors.

Wink_Ch10Barrier_Ex01

Load up this code and see how Wink responds. He should drive forward when nothing is in front of him, then stop if he gets close to an object like your hand or a piece of paper. Try with different barrier objects and also try on different surfaces. Once the center light sensor crosses a threshold of 100, his motors will stop.

As with our other examples, after playing for a while, you may notice a few strange things happening. First off, you may notice that Wink has to get really close to an object to make him stop. The other thing you may notice is that Wink doesn't want to drive at all. He may just sit there. What would cause this?

Remember that when we read the light sensor, we are reading the total amount of light falling on the sensor. This amount includes any light reflected from a barrier from the headlight, as well as any light from the headlight that was reflected from the surface. It also includes any other light that was already present in the room. For this reason, simply placing Wink in a brighter location in the room will make him stop driving. Anything that causes the light level to rise above 100 will make our "if" condition false and will cause Wink to stop moving.

So what is the best threshold value to use? (A "threshold" by the way, is a word often used in robotics, electronics, and programming to refer to any value where something specific happens). In this case we really need to know what the light sensor is reading so we can set this threshold to a good value. Remember the Serial.print() lesson earlier?

Let's use a Serial.print() function so Wink can tell us how much light he is seeing when he reads his sensor. We will insert this right after the sensor is measured.

```
int centerLight;   //declare variable

void loop(){

 digitalWrite(Headlight, HIGH);     //turn on IR Headlight
 centerLight = analogRead(AmbientSenseCenter);  //read sensor
 Serial.println(centerLight)        //print light level

 if (centerLight < 100)             //if centerLight < 100
  {
    motors(100,100);                //drive forward
  }
  else                              //otherwise...
  {
    motors(0,0);                    //be still
  }
} //end of loop()
```

Wink_Ch10Barrier_Ex02

} Same example as before, except we add Serial.println() to print the centerLight level to the Arduino Serial Monitor.

Open your serial monitor and see what readings you are getting from the light sensor. See how this changes when you place Wink on the running surface, how it changes when you pick him up, and how it changes as you place your hand in front of his nose.

As you move your hand closer, you should see this number go up. The closer your hand gets, the more light will reflect from your hand which will cause the sensor to see more light and increase the value. If you have a lot of light in your room, see how this value changes as you face Wink toward a light.

As you can see, this value can change a lot depending on what surface he is on and how much other light is present. Try to adjust the threshold value from the starting point of 100 and see if you can make Wink respond better.

You may quickly realize it's very hard to find a good threshold. If you lower the threshold, Wink will be more sensitive and he will see your hand further away. But this introduces another problem: when you make the threshold too low, Wink will stop moving just from the light already present in the room, or from the light reflected from the running surface.

Do you have any idea how we can solve this problem? We want Wink to be as sensitive as possible (so he sees the object further away) but the room light and the light from the surface are making this difficult. Think about that for a while or discuss with your group then go on to the next page.

How to remove the effect of ambient light...

An easy way to remove the effect of the ambient light already present in the room is to take two different measurements of the ambient light sensor.

We can measure the sensor one time with the headlight turned off and store the result in a variable. We can then turn the headlight on and measure the sensor a second time. This second value will include the amount of light reflected from a barrier and it will also include the amount of light that was already present from the room.

If we subtract the first value from the second value, we can effectively remove the amount of light that was already present in the room. The result will be the amount that was reflected from the barrier and the surface only. Study this example...

```
int centerLightOff;   //declare variables
int centerLightOn;
int centerLightOnly;

void loop(){

 digitalWrite(Headlight, LOW);      //turn off IR Headlight
 delay(1);                          //delay 1 millisecond
 centerLightOff = analogRead(AmbientSenseCenter);  //read sensor

 digitalWrite(Headlight, HIGH);     //turn on IR Headlight
 delay(1);                          //delay 1 millisecond
 centerLightOn = analogRead(AmbientSenseCenter);  //read sensor

 centerLightOnly = centerLightOn - centerLightOff;

 Serial.println(centerLightOnly);  //print light level

} //end of loop()
```

Wink_Ch10Barrier_Ex03

} Measure the light sensor with the IR headlight turned off. This measures how much light is already present in the room.

} Measure the light sensor with the IR headlight turned on. This measures room light as well as IR headlight light.

} Subtract the first measurement from the second. The result is light from the headlight only.

} Use Serial.println to print value to screen.

In this example we're measuring the amount of light with the headlight off, then with the headlight turned on. When we subtract the headlight off value from the headlight on value, we will be left with only the amount of light that was produced by the headlight reflecting off of the surface or any obstacle. We then use Serial.println() to display this value to the serial monitor.

We also wait a delay of one millisecond after turning the headlight on or off. This is because it takes a short amount of time for the sensor to react to the new light level. We need to wait this short delay so the sensor will read the correct value after it is stabilized to its new level.

Load this example onto Wink and experiment by placing Wink on and off of the surface and putting your hand in front of him. When you are holding Wink away from the surface and your hand is not near his nose, the amount should be very low because there's nothing to reflect the light off of.

Place Wink on the running surface you intend to use and see what values you are seeing when placing your hand a distance in front of him. See how close you need to get your hand to make this value start to rise. Pick a value somewhere between the lower "no hand present" value and the higher "hand is present" value. Remember this number. This will be a good threshold value for the next example. We are going to take the example above and add on the "if" CS with some code to control the motors.

```
int centerLightOff;   //declare variables
int centerLightOn;
int centerLightOnly;

void loop(){

 digitalWrite(Headlight, LOW);      //turn off IR Headlight
 delay(1);                          //delay 1 millisecond
 centerLightOff = analogRead(AmbientSenseCenter);  //read sensor

 digitalWrite(Headlight, HIGH);     //turn on IR Headlight
 delay(1);                          //delay 1 millisecond
 centerLightOn = analogRead(AmbientSenseCenter);  //read sensor

 centerLightOnly = centerLightOn - centerLightOff;

 Serial.println(centerLightOnly);   //print light level

 if (centerLightOnly < 15)          //if centerLight < 15
  {
    motors(100,100);                //drive forward
  }
  else                              //otherwise...
  {
    motors(0,0);                    //be still
  }

} //end of loop()
```

Wink_Ch10Barrier_Ex04

} Adjust threshold here. 15 is a good starting value for a dark table, and 90 to 100 is a good starting value for a white table or white paper.

Experiment with the threshold value unit you get something that works well. If Wink tends to stop, or stop and go without your hand present, then raise the threshold a bit. (This can happen if the light reflected from the surface is enough to exceed the threshold). Keep in mind that as Wink rocks forward, he is looking more directly at the surface, so the amount of light he sees from the surface will increase.

If Wink has to get very close to your hand before he stops, then lower the threshold a bit.

Dealing with light reflected from the surface...

So now we have our robot working pretty well, except we still need to provide a threshold number that changes depending on what surface Wink is driving on. It can be annoying to have to measure and test this number every time Wink changes surfaces.

We can deal with the light reflected from the surface in many different ways. I am going to share a method that is fairly simple. There are other ways that may work somewhat better, but they are more complex and beyond the level of this lesson.

The basic idea is to take a measurement while Wink is sitting still on the surface, with no barrier in front of him. We can store this measurement in a variable. As long as the surface doesn't change in brightness, then this initial measurement can be used as a "baseline" that represents the amount of light we can usually expect to be reflected from the surface.

Once this "baseline" value is known, Wink can begin driving around and taking more measurements along the way. He can compare each new measurement to this baseline measurement, and if the new measurement increases by a certain amount above the baseline value, we can assume a barrier is present.

Scary code warning...

The code on the next page looks scary, but it's not so bad. It's the longest example you've seen so far but please don't let that scare you. In a future lesson we're going to learn how to write our own "functions" that we will use to make this code much shorter easier to understand.

The reason the next example looks so long is mostly because we're repeating the large block of code that turns the headlight off and on, reads the ambient sensor, and subtracts the two readings.

In the next example, we are doing exactly what I described above. Notice there is now a large block of code inside the setup() function. Up to this point, we haven't added any significant code to setup(). The setup() is a great place to put code that you only want to run one time at startup. This is a perfect place to put the code to read the sensor to get our "baseline" reflected light.

You will also notice that we have declared our variables this time before the setup() function. Any variable needs to be declared before it can be used. Because we are using some of our variables in the code inside our setup() function, we need to move the declarations above the setup() or the program won't compile.

Study the code in the next example then load it on to Wink. From the time Wink turns on and you hear the start chirp, you have two seconds to put Wink down on the surface before he runs the code to get his baseline measurement.

If you don't set him down quickly enough, this baseline will not be correct. This is also true if you change surfaces. If you need to re-set Wink's baseline value, turn him off with the PWR button, then turn him back on. The program will start from the beginning with the start chirp and a fresh two second delay.

Detecting Barriers

Lesson 10

```
#include “WinkHardware.h”

int centerLightOff,centerLightOn,centerLightOnly;
int baseline,threshold;

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

  delay(2000);   //wait 2 seconds

  digitalWrite(Headlight, LOW);     //turn off IR Headlight
  delay(1);                         //delay 1 millisecond
  centerLightOff = analogRead(AmbientSenseCenter); //read sensor

  digitalWrite(Headlight, HIGH);    //turn on IR Headlight
  delay(1);                         //delay 1 millisecond
  centerLightOn = analogRead(AmbientSenseCenter); //read sensor

  baseline = centerLightOn - centerLightOff;  //subtract values
  threshold = baseline + 10;
}

void loop(){

 digitalWrite(Headlight, LOW);     //turn off IR Headlight
 delay(1);                         //delay 1 millisecond
 centerLightOff = analogRead(AmbientSenseCenter);  //read sensor

 digitalWrite(Headlight, HIGH);    //turn on IR Headlight
 delay(1);                         //delay 1 millisecond
 centerLightOn = analogRead(AmbientSenseCenter);  //read sensor

 centerLightOnly = centerLightOn - centerLightOff;

 if (centerLightOnly < threshold)  //threshold from above
  {
    motors(100,100);               //drive forward
  }
  else                             //otherwise...
  {
    motors(0,0);                   //be still
  }
} //end of loop()
```

Wink_Ch10Barrier_Ex05

} Declare variables before setup()

} Normal setup functions

} Wait 2 seconds to put Wink down on surface

} Read sensor with Headlight turned off then turned on. Subtract the two. If this is done with no barrier present, but still sitting on the surface, then “baseline” should have only the amount of light reflected from the surface.

} Set “threshold” to baseline plus 10. By adjusting this “10” value you can set how sensitive Wink will be.

} This part is the same as our previous example.

} This time we’re going to compare centerLightOnly to the value in “threshold” which was determined in our setup() function above.

2 SKILL LEVEL

Load the above example then play with Wink for a while. In most cases he will do a good job of getting a baseline for your exact surface. I'll remind you again that when Wink makes the reading to set his baseline (two seconds after the start chirp and right before he starts moving) it is important to have no obstacles or fingers in front of him. Try to give him at least 4 to 5 inches of open space in front while he makes this measurement.

To adjust the sensitivity, play with the value in the last line of the setup() function. Lower values will allow him to see barriers further away.

```
threshold = baseline + 10;
```

} Adjust the "10" value to set how sensitive Wink will be. Lower numbers make him more sensitive to seeing an obstacle.

Lets consider what is happening with this line of code. Remember the value in "baseline" is the amount of light being reflected from the running surface. We are then adding 10 to this value and storing it in the variable "threshold". Later in our code, in the "if" CS, we determine if the centerLightOnly value is less than this threshold value. If it is less, then the motors go forward. If it is no longer less than, the motors will stop moving.

This basically means that if a barrier or object causes the value to increase by more than 10 above the baseline value calculated in the setup() function, that the motors will stop.

If this value is made lower, then the threshold is lower. This means Wink can see an object further away, but it also means that as he rocks around on the surface, he may be more likely to think he sees a barrier even if a barrier is not present.

Another challenge...

Now you know how to sense a barrier in front of Wink. Let's add one more step to the previous example as a final challenge for this lesson. In the code above, Wink will drive forward until he sees a barrier then stop. Once the barrier is moved away, he will start moving again.

You have probably noticed that if you put your hand in front of him, then slowly move your hand back away from him, he will continue driving forward following your hand. He will tend to remain about the same distance away from your hand each time.

In this next challenge, lets see if we can make him back up when your hand gets closer. We will build on the previous example so that he drives until he sees your hand then stops. But if you then move your hand closer to Wink, can we make him back up and move away?

Remember that as your hand gets closer to Wink, the amount of light he measures will become larger and larger. Can we use "if", "else if", and "else" to make Wink either drive forward, drive backward, or be still?

In this new example, we want to establish a threshold where Wink will stop driving forward, then a different larger threshold where he will drive backward. Consider how you may do this then continue to the next example.

Challenge: Wink backs up from barrier...

There are many ways to solve this challenge. Here is what I came up with. In this example, I create a second threshold value, then add another condition to our "if" CS down below. This should cause Wink to move toward your hand and stop, then move back away from your hand if you move toward him. You can play with the two threshold values in the setup() to adjust how sensitive Wink is. You can also adjust his speed by changing the motor speeds down below. Experiment and have fun, then write your own code that makes Wink do something when he sees a barrier.

```
int centerLightOff,centerLightOn,centerLightOnly;
int baseline,stopThreshold,revThreshold;

void setup(){
 //... the rest of setup() same as last example
  stopThreshold = baseline + 10;
  revThreshold = baseline + 15;
}

void loop(){

 digitalWrite(Headlight, LOW);     //turn off IR Headlight
 delay(1);                         //delay 1 millisecond
 centerLightOff = analogRead(AmbientSenseCenter);  //read sensor

 digitalWrite(Headlight, HIGH);    //turn on IR Headlight
 delay(1);                         //delay 1 millisecond
 centerLightOn = analogRead(AmbientSenseCenter);  //read sensor

 centerLightOnly = centerLightOn - centerLightOff;

 if (centerLightOnly < stopThreshold)       //"stop" threshold
 {
   motors(100,100);                         //drive forward
 }
 else if (centerLightOnly > revThreshold)   //"reverse" threshold
 {
   motors(-100,-100);                       //drive backward
 }
 else                                       //otherwise...
 {
   motors(0,0);                             //be still
 }
} //end of loop()
```

Wink_Ch10Barrier_Ex06

} using "stopThreshold" and "revThreshold" for our two threshold levels

} First part of setup() same as before

} Added new "revThreshold"

} If the light level is less than the "stopThreshold" value, Wink must be far from a barrier. Drive in forward direction.

} Else If the light level is greater than the "revThreshold" value, Wink must be really close to a barrier. Drive in reverse direction.

} Else, light level must be between the two thresholds. Stop moving.

2 SKILL LEVEL