## Wink IR Remote Control Ability...

Wink may be fitted with an infrared ("IR") remote control receiver. This part is not standard. The part is an upgrade available if requested. If your Wink has the IR receive ability, you will find a square black part about 1/4 inch in size just behind his right motor. This part is soldered to three holes in the circuit board. Please contact plumgeek.com if you would like to add the IR ability to an existing Wink robot (the part is easy to solder yourself with a small soldering iron).

Wink can receive IR communications from a remote control. Additionally, Wink can receive IR communications from other Wink robots or Ringo robots.

The optional Plum Geek remote control is configured to send packets of 4 bytes whenever a button is pressed. For this reason, by default we assume Wink will be sending and/or receiving packets that are 4 bytes long. You can control the number of bytes for your own purpose if you like.

**NOTE:** The IR functions in this lesson require Wink Base Sketch Rev01_3 or higher.

## Receiving IR Data...

Wink can be set to receive and react to data from an IR remote control or another robot. This generally requires these steps:

1. Wink is set to begin receiving data with RxIRRestart().
2. Your code occasionally checks if an IR packet has been received by calling IsIRDone().
3. Your code determines what data was received and does something based on what was received.

```
RxIRRestart(4);   //enable IR receive handler, wait for 4 bytes
```

} Enables background IR receive handler, set to receive 4 byte packets

The IR receive handler runs in the background. It sets up a "buffer" which is a string of variables called an array that will hold the data received from the remote control. Once a packet is received, the background handler stops and will not receive any more packets. The data in the received packet will be held in this buffer forever until RxIRRestart() is called again, which clears the buffer and re-enables the receive handler.

```
if(IsIRDone())    //will be true if a packet has been received
```

} IsIRDone() checks to see if a packet has been received.

You can occasionally call IsIRDone() in your code. This function will return a non-zero value if a packet has been received. This means that if it is placed in the condition of an "if" CS, the condition will be true if a packet was received.

# Using IR Remote Control

## Bonus Lesson 1

Once you have determined that a packet has been received, your code must look at this packet to see what it contains. If you are receiving from the IR remote control, the function GetIRButton() will do this for you automatically and tell you which button was pressed. This is the easiest way to make your Wink respond to button presses on the remote.

For advanced users, know that the buffer is actually a global array called IRBytes[] which is 20 bytes long. After IsIRDone() returns a non-zero value, you can read IRBytes[] directly to determine what was received.

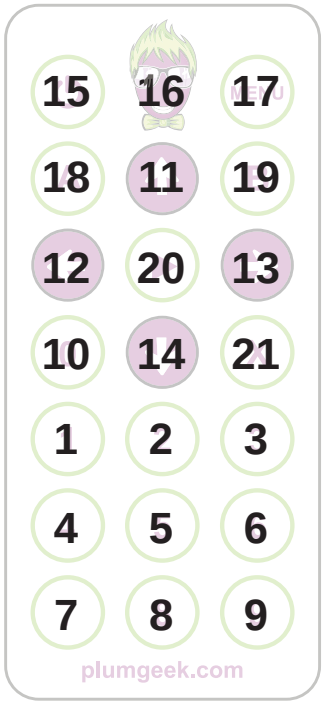
```
byte button;      //declare variable "button" as a byte type

button = GetIRButton();    //automatically puts pressed button
                           //number into "button" variable.
```

} The variable you use to hold the button number should be "byte" type.

} Automatically reads received packet and returns the button number that was pressed. (See diagram below).

If the packet received did not match the coding of a button on the remote control, GetIRButton() will return zero. So in the above example, if the GetIRButton() function didn't recognize the received packet data as matching a button on the Plum Geek remote control, the "button" variable would equal zero.

After you have read the buffer data (or figured out what button was pressed with the GetIRButton() function), you will usually want to re-start the IR receive function so you can receive another packet.




### Packet Sent by Keys

| | |
|---|---|
| 00 FF 45 BA | 00 FF 0C F3 |
| 00 FF 47 B8 | 00 FF 18 E7 |
| 00 FF 44 BB | 00 FF 5E A1 |
| 00 FF 40 BF | 00 FF 08 F7 |
| 00 FF 43 BC | 00 FF 1C E3 |
| 00 FF 07 F8 | 00 FF 5A A5 |
| 00 FF 15 EA | 00 FF 42 BD |
| 00 FF 09 F6 | 00 FF 52 AD |
| 00 FF 16 E9 | 00 FF 4A B5 |
| 00 FF 19 E6 | 00 FF 46 B9 |
| 00 FF 0D F2 | |

## Remote Control Button Map

3 SKILL LEVEL

## IR Remote Example...

Study the following example to see how Wink can receive remote control commands and react to them.

```
void setup(){
 hardwareBegin();
 playStartChirp();
 RxIRRestart(4);           //prepare to receive 4 byte packet
}

byte button;               //declare "button" variable as a byte

void loop(){

 if(IsIRDone()){  //true if an IR packet has been received
   button = GetIRButton(); //get remote button number
   Serial.println(button); //print button number to serial mon

   if(button == 11){      //if FWD key pressed
     motors(100,100);     //drive forward
     delay(500);          //wait 1/2 second
     motors(0,0);         //stop driving
     RxIRRestart(4);      //restart IR handler to get next packet
   }
   else if(button == 14){  //BACK key pressed
     motors(-100,-100);     //drive backward
     delay(500);            //wait 1/2 second
     motors(0,0);           //stop driving
     RxIRRestart(4);        //rst IR handler to get next packet
   }
   else{
     // do nothing except re-start the IR handler
     RxIRRestart(4);        //rst IR handler to get next packet
   }
 } //end of if(IsIRDone())

} //end of loop()
```

*Wink_BCh01Remote_Ex01*

- Start remote receive handler
- Declare "button" variable
- Determine if a packet has been received
- Get remote control button number
- Print number (useful for debugging)
- If "FORWARD" key was pressed, drive forward for 1/2 second.
- Re-start remote receive handler
- If "BACKWARD" key was pressed, drive backward for 1/2 second.
- Re-start remote receive handler
- If packet didn't match a key, we still need to re-start the handler so it will see the next packet the next time a button is pressed.

## "Drive with remote control"...

We haven't included it here as it is rather long, but check out the example "Wink_BCh01Remote_Ex02" for a fully functioning "drive around with remote control" example. Feel free to edit what Wink does when his buttons are pressed. Have fun!

3 SKILL LEVEL

## Sending IR data with Wink's Headlight...

Wink can create IR packet data with his headlight as if he was a remote control. This can be used to communicate between robots. This can be done the "easy way", or the "more complicated way" if you want to do something more advanced.

Let's start with the easy way...

```
TxIRKey(1);       //creates signal as if you pressed the "1"
                  //button on the remote

TxIRKey(12);      //creates signal as if you pressed the "LEFT"
                  //arrow key on the remote
```

Easy way to transmit the IR packet corresponding to a given key on the remote control.

Easy enough right? If you want to transmit your own custom data, the process is to create an array, then populate that array with the data you want to send. You then call the function TxIR() where you include the name of your array and the length of the array.

Like this...

```
byte dataToSend[]={0xF3,0xFF,0xE3}  //declare and populate array

TxIR(dataToSend,3);    //transmit dataToSend array, which is 3
                       //bytes long
```

Use TxIR to send an array via the IR headlight. The second argument is the length of the array.

## An example...

Load the above example "Wink_BCh01Remote_Ex02" on to a Wink with the IR receiver part installed. Then put the code below into the loop() function of a different Wink. This second Wink should command the first Wink to rotate back and forth when the two are close enough together.

```
void loop(){
  int i;
  for(i=0,i<5,i++){         //repeat 5 times
    TxIRKey(12);            //send "LEFT" key
    delay(50);
  }
  for(i=0,i<5,i++){         //repeat 5 times
    TxIRKey(13);            //send "RIGHT" key
    delay(50);
  }
}
```

*Wink_BCh01Remote_Ex03*

## Some notes about IR and the remote...

Here are a few notes to keep in mind when working with the IR handler.

1. When you press and hold a button on the Plum Geek remote control, the first packet that is transmitted contains the data for the key that was pressed. As long as you hold the key, the remote transmits the very beginning of the packet over and over every few milliseconds. This beginning of the packet is called a “preamble”. The preamble is the same no matter what key you pressed.

   For this reason, we have written the code to assume that if a fragment of a packet is received, the handler assumes it has seen this “preamble” signal and that you are still pressing the last key it received correctly. If Wink is nearly out of range of the remote control (and can’t see it very well), if you press a new key, he may assume you re-pressed the last key he received correctly.

   In most cases this isn’t a problem. If this will cause a problem for the behavior you are trying to write, we suggest reading IRBytes[] directly instead of using GetIRButton().

2. The coding of the remote control is hard-coded into the remote and cannot be updated. The remote we are using is based on a similar remote commonly supplied with room lighting systems. The factory that produces the remote has custom printed the Plum Geek overlay for the surface of the remote but the coding has not been customized for us.

3. If you want one remote to control several different robots, but you don’t want them all to react to the same button press at the same time, you could write your own code to accept a two button combination. For example, you code a given Wink to only accept a command if the “A” button was recently pressed on the remote, and you code a different Wink to only accept a command if the “B” button was recently pressed on the remote. That is beyond the scope of this lesson, but if anyone writes code to handle this, please share and we will include it in future versions of this lesson. (Hint: to determine if a button was “recently” pressed, you can mark time with millis(), which tells you how many milliseconds Wink has been running for. It will roll over after about 72 days).

4. The part on Wink that sees the IR remote control will occasionally trigger if the lighting around Wink changes quickly. Some lighting (especially LED room lighting) actually pulses on and off very quickly. This can cause the IR receiver to trigger. If this happens, Wink will think he has received a packet of data, and the IsIRDone() function will return true. This is not usually a problem however as the GetIRButton() function will return zero if the data doesn’t exactly match the coding for a remote control button. If you transmit your own custom data, you may want to include a checksum at the end of your packet to avoid this issue. (A checksum is good practice as the packet will occasionally be mis-read anyway. This is a challenge to all wireless systems).

5. The receive handler occasionally hangs for some reason. Normally after a lot of communication. We’re not sure why this is happening, but it can sometimes be corrected by pressing more buttons on the remote or last resort resetting the robot. If anyone with advanced skills wants to trouble shoot we’d appreciate any insight.