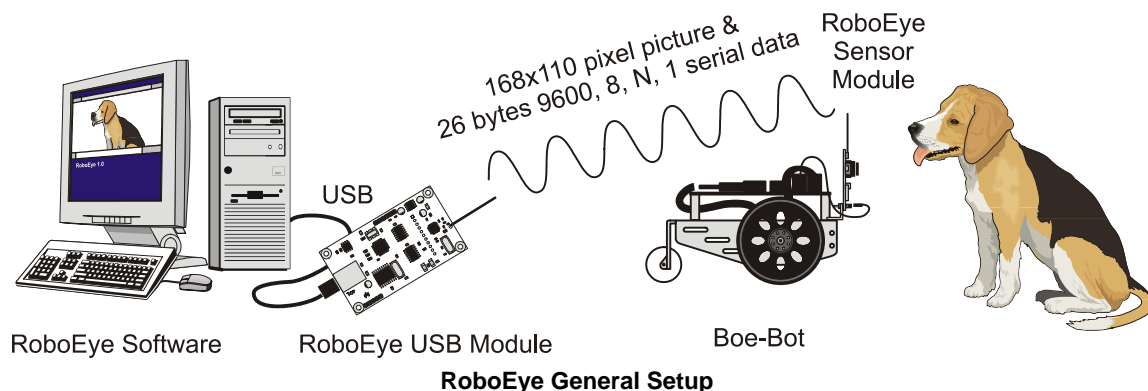


## RoboEye Wireless Vision Sensor (#30008)

RoboEye provides both a simple and sophisticated method of robotics vision and control. In its simple form, the RoboEye Sensor sends a 168x110 pixel image over a 2.4 GHz wireless band for display on your PC using the RoboEye software. An additional serial I/O line may be used to send control commands or data between the RoboEye software terminal and a Parallax BASIC Stamp<sup>®</sup> microcontroller module. The wireless data link includes built-in error checking.

In its more advanced form, software developers can take advantage of RoboEye's open-sourced image transmission protocol for further decision making and analysis on a computer. This is done using an OCX application and intercepting a data packet stream which describes pixel colors and location. This type of use is similar to the popular CMU Cam, except that image analysis can be done with PC software instead of the BASIC Stamp.

Pay attention to the setup process described in this documentation. Installation of the USB driver, setup of the RoboEye Sensor Module and configuration works best if done in a particular order.



## Packing List

Verify that your package is complete and contact us if anything is missing. This kit also includes the basic hardware you will need to connect the RoboEye Sensor Module to the Parallax Boe-Bot<sup>®</sup> robot.

- (1) RoboEye Sensor Module and (1) RoboEye USB Module (#727-30008)
- (1) 3-pin 10" female/female cable (white, red, black) (#805-00001)
- (1) Lens and lens holder (black plastic) (#721-30008)
- (2) Right-angle aluminum bracket parts (#720-28215)
- (4) 4/40 1/4" screws (#700-00028)
- (4) 4/40 nuts (#700-00003)
- (2) M2 x 10 screws (#710-10001)

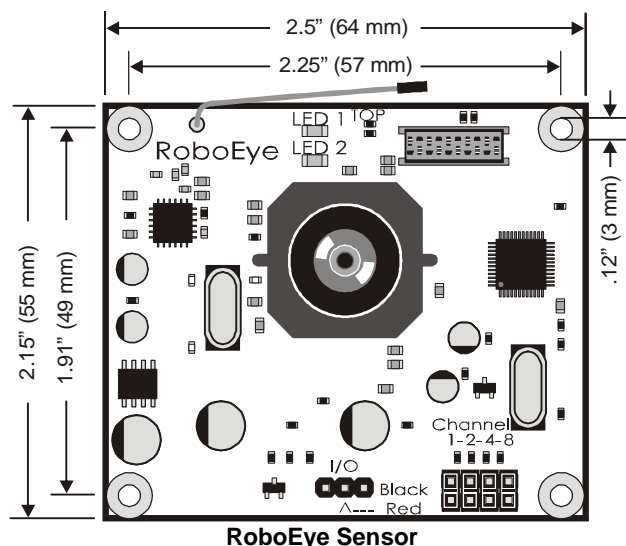
**NOTE:** You will also need a standard A-male to B-male USB cable to connect the RoboEye USB module to your PC. A USB cable is not included in this kit. If you do not already have a USB cable, you may order one from Parallax (Part ##805-00007).

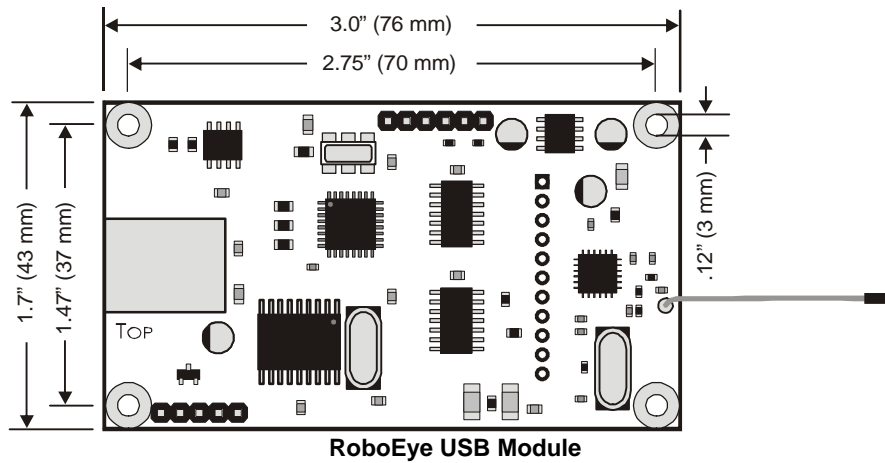
RoboEye PC software and BASIC Stamp source code are available for free download from our RoboEye web page on the Parallax web site ([http://www.parallax.com/detail.asp?product\\_id=30008](http://www.parallax.com/detail.asp?product_id=30008)).

## Electrical Specifications

<b>Power supply</b>	RoboEye Sensor 4-9 VDC (5 recommended); RoboEye USB Module (USB port provides 5 V and 100 mA to run the hardware)
<b>RoboEye Sensor Supply Current</b>	50 mA (typical)
<b>I/O Line Voltages</b>	Maximum 5 V input voltage; 3 V output voltage (logical high); 0.3 V output voltage (logical low)
<b>Serial Communication</b>	UART, 9600 Baud, 8 bit data, 1 stop bit with 26 byte buffer
<b>Frequency</b>	2400 – 2524 MHz with 16 different channels (jumper configured)

## Physical Dimensions





## Lens and Lens Holder Mounting to RoboEye Sensor Module



1. Locate the (2) M2 x 10 screws, RoboEye Sensor Module and Lens/Lens Holder assembly.

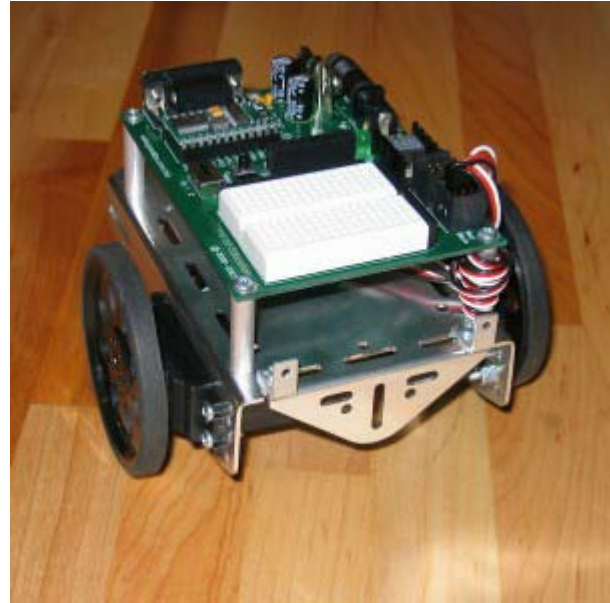
2. Thread the M2 x 10 screws through the back of the printed circuit board, into the lens/lens holder assembly. No nuts are required since the screws are self-threading.

## Mount the RoboEye Sensor Module to a Boe-Bot

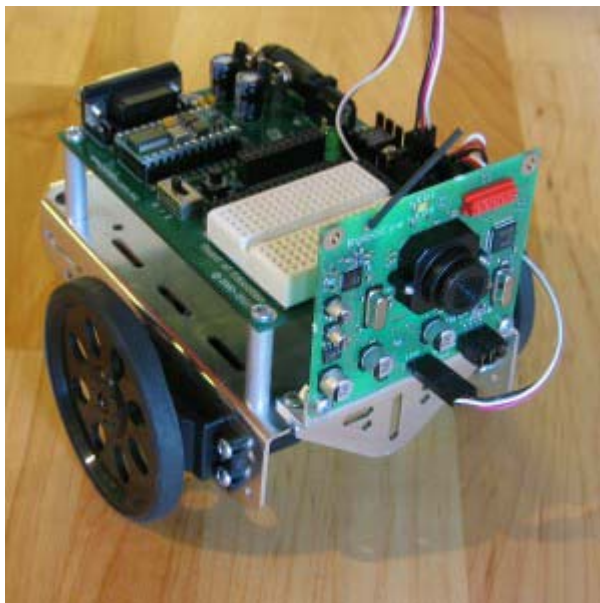
To mount the RoboEye Sensor Module on a Boe-Bot, follow these additional steps. Disconnect power from the Board of Education® before you begin.



1. Locate the (4) 4/40 1/4" screws, (4) 4/40 nuts, (2) right-angle brackets, and the 10" female/female three-pin cable.



2. Using (2) 4/40 1/4" screws and (2) 4/40 nuts, mount the (2) right-angle brackets to the Boe-Bot robot's front slots.



3. Using (2) 4/40 1/4" screws and (2) 4/40 nuts, mount the RoboEye Sensor Module to the brackets. Connect the 3-pin cable between the RoboEye Sensor Module and the BOE's P15 servo port, paying particular attention to polarity.

## USB Driver Installation

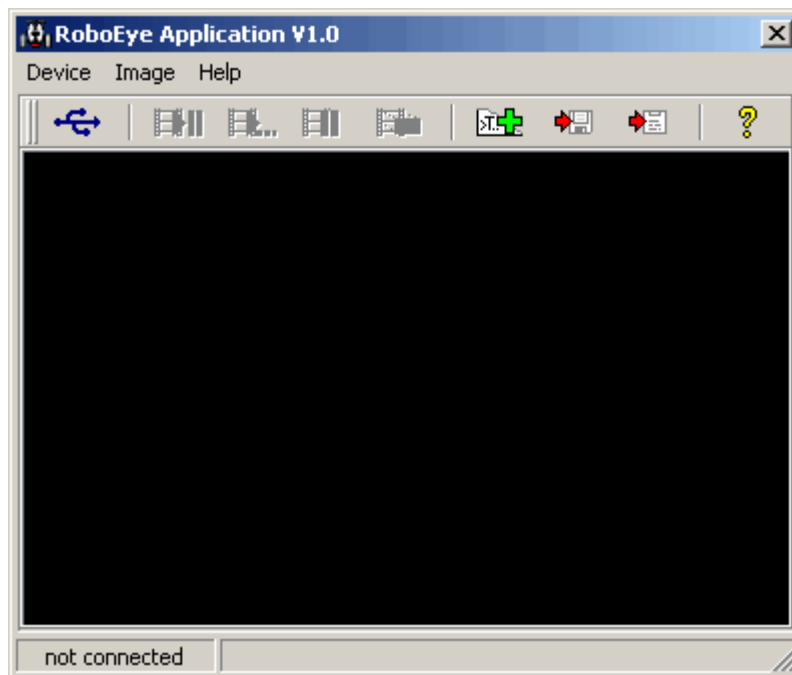
The USB drivers need to be installed first.

1. Go to the Parallax RoboEye web page [http://www.parallax.com/detail.asp?product\\_id=30008](http://www.parallax.com/detail.asp?product_id=30008) and download "RoboEye Software" and "FTDI VCP Drivers". These are zipped software packages.
2. Extract the contents of these zipped folders to your computer so you can access the files.
3. Connect the RoboEye USB Module to the PC using a USB cable (not included).
4. The PC will report that new hardware has been found. Point the PC to the FTDI drivers unzipped to your PC.









The RoboEye control software consists of two parts: CamViewXControl.ocx and RoboEye.exe. Install both of these by running RoboEyeInstall.exe.

## RoboEye Software Setup

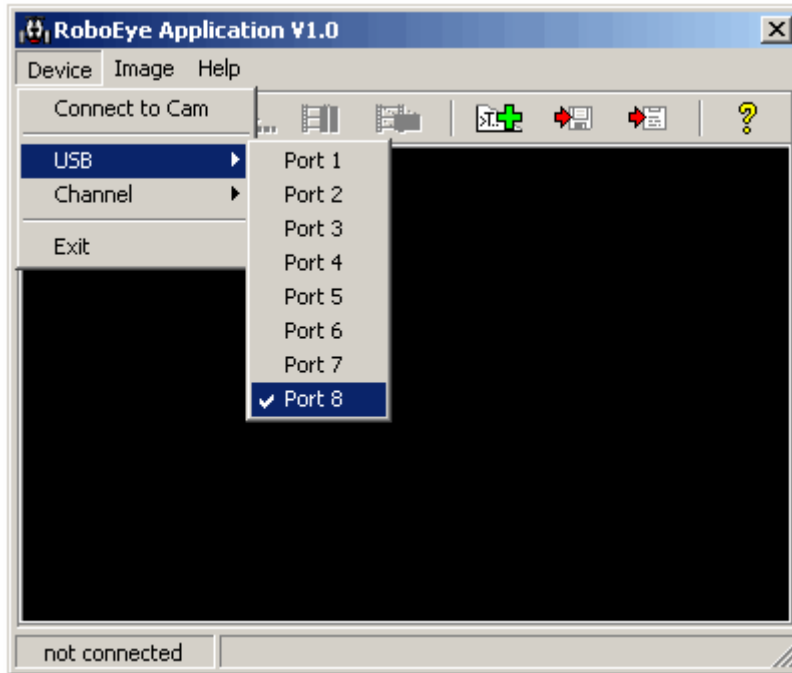
1. Run RoboEyeInstall.exe software. Then run RoboEye.exe.



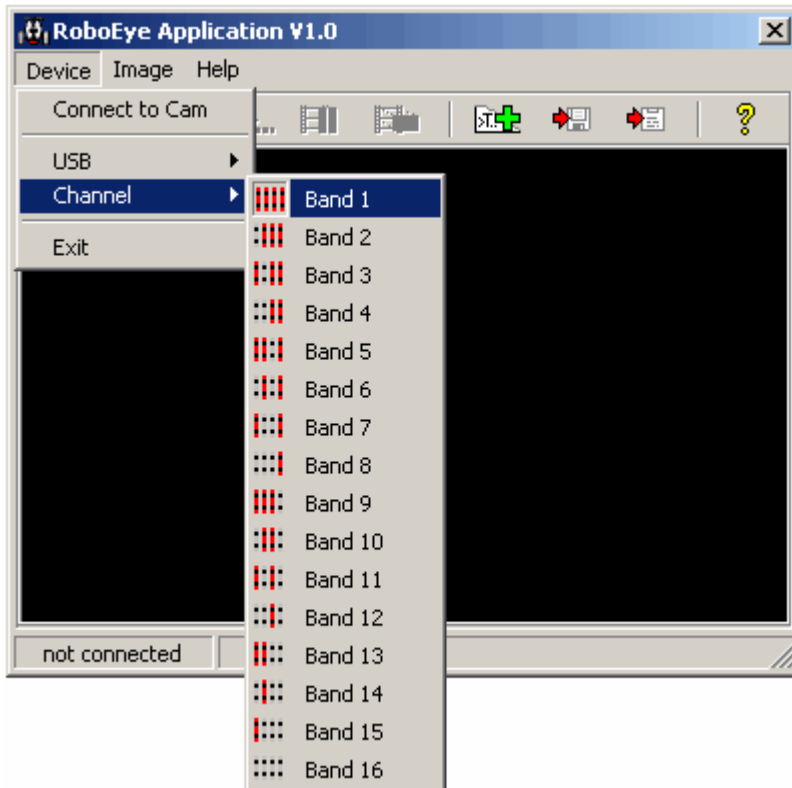
The icons provide the following control:

	connect USB interface		double the picture size
	capture a single picture		opens terminal
	start repeat capturing		save picture to disk
	stops capturing		copy picture to clipboard

2. Choose the appropriate USB port in the RoboEye software. This must match the USB virtual COM port assigned by the Windows Device Manager.



3. Select the wireless transmitting channel, based on the placement of jumpers on the RoboEye Sensor Module.



4. Apply power to the remote RoboEye Sensor Module. After power is applied, the green LED2 flashes.
5. Click the green "+" button to show the terminal window.
6. Click on the "Connect to camera" icon under the Device menu. Now you can test the connection between the RoboEye Sensor Module and the RoboEye USB Module connected to the PC.
7. In the white terminal field, type "ping" and press the >>> button to send the data. RoboEye Sensor Module will respond with the firmware revision number.



Real-time moving picture transmission is not possible with RoboEye. It is also impossible to store a complete picture on-board the picture sensor. Sharp and clear pictures are obtained using a fixed camera position on fixed scenery.

If you have connection troubles these are often solved by unplugging the RoboEye USB Module.



## Special Operations with RoboEye

Apart from pure picture transmission, RoboEye can do a lot more. The data transmission between the USB-module and the camera unit is bidirectional. That means data can be transmitted to the camera, and additional information can be sent from camera to the PC. Before we check these possibilities we will have a look at the different data sources and receivers of information. In particular, the different data receivers, camera sensor and serial interface are confusing at the beginning.

The camera module has a modified serial interface. Data is transmitted as an asynchronous data stream via this port with a transmission rate of 9600 Baud.

To perform a simple test, open the terminal window of the PC application and type the short text "Hello world", then press the send button. After pressing the send button, this message is sent to the camera module and output serially. A feedback of information from camera to PC is just as simple. When, for example, a send-string is sent to the serial entry of the camera via the BASIC Stamp, the terminal window will show this data.

Important: Because the camera has only one input/output connection, you must provide for conflict-free transmission. From a technical point of view the bidirectional communication is only half-duplex. Therefore, it can either send or receive data, but not both at the same time.

For a simple solution, one of the stations, e.g. the PC, is defined as a master. Then the slave can only answer when a request is sent by the PC. The consideration of the transmission medium radio is very important, too. Radio is ALWAYS uncertain. That means radio messages can be lost. That is perfectly clear in an environment with lots of different transmitters or as soon there is no direct optical sight between the USB-module and the camera, or when the distance is getting too big. In such cases a request from the master to the slave must be repeated several times. The structuring of this action, often called data transmission protocol, is the programmer's job.

Aside from pure data transmission that does not have any influence on the camera, some control commands are implemented. But why? In the simplest case, the user releases the picture transmission from the PC by pressing the send button, and the camera sends the picture data. However, the capability of having a timed or event-triggered picture transmission from the camera would also be useful, wouldn't it?

Let's make a simple example. Again the camera is connected to a BASIC Stamp microcontroller. The BASIC Stamp shall request a picture every 5 minutes. Therefore, a time loop is programmed and after every 5 minutes the command <ESC>T<SPACE>100<CR> is sent. What does that mean? The escape-sign (\$1b) signals the camera that not a simple data string is following but a sequence of commands. In other words, everything that follows <ESC> is very interesting for the camera. The next sign after the <ESC> specifies the wanted action. <T> means "transmit", so the camera shall send a picture. The parameter behind the <T> specifies the amount of repetitions. <CR> ends the command and the processor inside the camera starts instruction execution now. Note: The commands are case sensitive.

With three other commands, complex actions can be accomplished. The search instruction <S> makes the camera processor calculate the x-coordinate of continuous monochrome objects. Here, X-coordinate means the number of the pixel that represents the found center line (For details, please, look at section "Object Detection with RoboEye"). That e.g. a robot can handle the object coordinates it asks for the found value with the command <P>.

More for specialists is the command <I>. With this instruction the internal registers of the picture sensor are described. To use this command it is absolutely necessary to study the data sheet of the picture sensor (OV6630) **before action**. In the already described examples the sequences of commands come



from an external gadget (like a BASIC Stamp at the serial entry of RoboEye. Sometimes that can be awkward and involved, especially when the <S> command is used, which means that the parameters in the command have to be changed regularly. So, you would be programming the BASIC Stamp, watching if everything works properly, then downloading a new program, again watching,... That is no fun. To shorten the procedure it would be nice to use the command <S> from PC, wouldn't it?

That is possible. The so-called "echo" command does not only give out the sequences from the PC-terminal window to the serial interface, but it also reads them back into RoboEye. That means that we can do without an external control. While "echo" switches on this mode, "noecho" finishes it. Because of that, quickly playing with camera commands is possible.

Now let's talk about the very "tough" cases. With the delivered PC-software all functions of the camera can be tried out. Nevertheless, there will be users who want to solve a very special problem and therefore write their own software. For this purpose the transmission protocol is open (For details, please, look at the section "Data Protocol between PC and RoboEye Sensor Module").

The PC and the receiving module communicate via USB. However, the original USB-interface is changed to a so-called virtual COM-Port by a special circuit. The needed driver is a part of the RoboEye package. Anyway, it would do no harm if one looks for updates at the website of the interface-circuit's manufacturer. The circuit FT232BM by FTDI ([www.ftdichip.com](http://www.ftdichip.com)) is used.

After driver installation, the programmer has obtained another fast serial COM-Port. With a simple terminal program the camera sensor can be controlled without PC application. But there is an important "trick" to pay attention to. The transmission rate is "asymmetrical": commands from the PC to the USB-module are sent with 9600 Baud, but data from USB-module to PC come with 923,076 Baud! Because it is not possible to control the interface "asymmetrically", it is necessary to change Baud rate regularly. That means sending the control word to the USB-Module with 9600 Baud, changing the Baud rate, receiving and analyzing the answer from the camera module with 923,076 Baud. The USB-module is largely transparent, and the built-in microcontroller only creates the necessary control signals for the high frequency circuit. We do not give a closer look at this kind of programming; corresponding questions for further support are to send to Parallax directly.

## Serial Communication Between RoboEye Sensor Module and PC

### Sending Commands to the RoboEye Sensor Module

Open the RoboEye software terminal window (press the open terminal button) to get access to the serial I/O line. The serial I/O line is configured as a UART with 9600 Baud (9600 Baud, one stop bit, no parity, no hardware handshake) in half duplex mode. This means that command strings are sent directly from the terminal input line through the RoboEye USB Module to the RoboEye Sensor Module.

For example, if you want to transmit the string "Hello", type the individual characters and press <Enter> or click on the send button (>>>) at the end of the line. The five characters are packed into a message and will be transmitted to the picture sensor wirelessly. No special data transmission protocol is required.

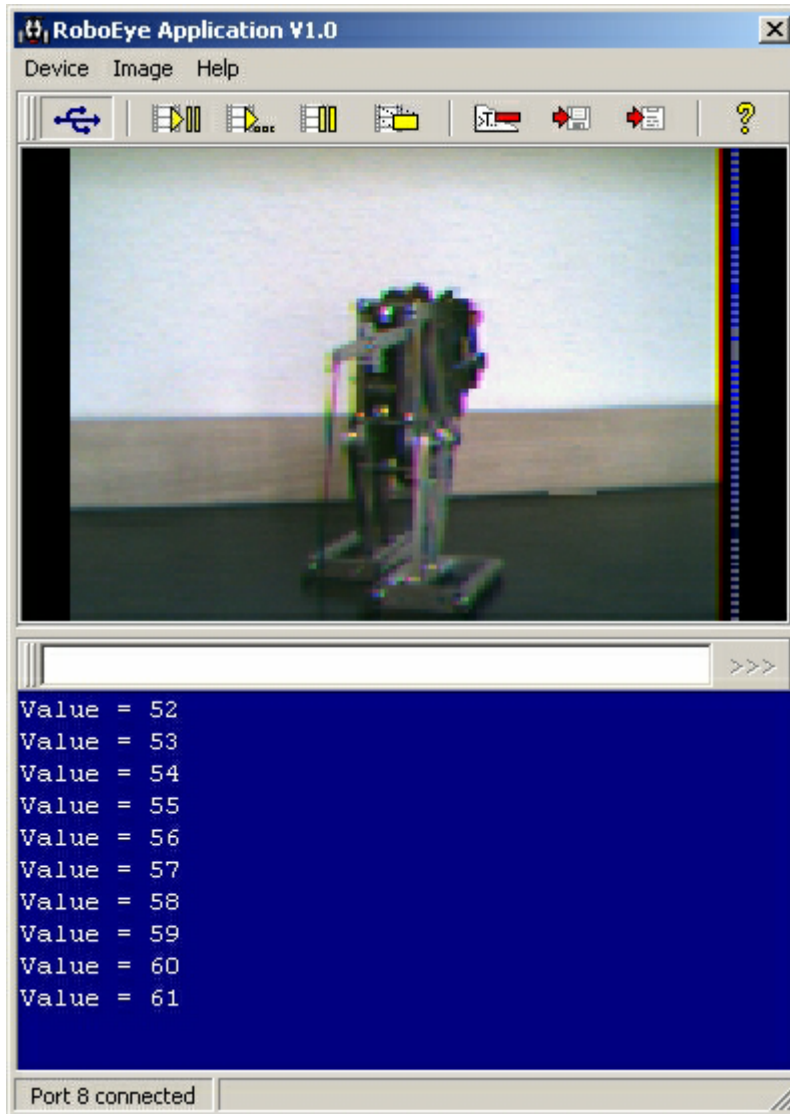
Transmission between the PC and RoboEye Sensor Module is most reliable if the picture auto-capture is turned off. It is also important to note that hitting the send button (>>>) sends the entire string of characters present in the white field.

### Sending Data from the BASIC Stamp to the PC

Each incoming message will be buffered until a <CR> is detected. Incoming messages are limited to 26 bytes. Here is a very simple example using the BASIC Stamp 2 module:

```
' ----- [ Title ] -----  
' BASIC Stamp to the PC Basic Communication - RoboEyeSimpleTransmission.BS2  
  
' {$STAMP BS2}           ' Stamp directive  
' {$PBASIC 2.5}         ' PBASIC directive  
  
' ----- [ Variables ] -----  
  
counter          VAR      Byte          ' FOR...NEXT loop counter  
  
' ----- [ Main Routine ] -----  
  
controlPanel:  
  FOR counter = 1 TO 256  
    SEROUT 15, 84, ["Value = ",DEC counter, CR]  
    PAUSE 1000  
  NEXT  
END
```

The example above sends the value of "counter" to the BASIC Stamp P15 at 9600 Baud (see the SEROUT command in the BASIC Stamp Syntax and Reference Manual).



**RoboEyeSimpleTransmission.BS2 Example Output**

### **RoboEye Sensor Direct Command Control from the BASIC Stamp**

In some cases it would be very useful if the RoboEye Sensor Module could initiate a picture transmission without any user interaction from the PC.

For such an application a few “escape” sequences are defined. These start with the escape character (character number \$1B hexadecimal, 27 decimal) and terminate with a carriage return (\$0D, or 13). The following escape commands are implemented in the RoboEye firmware:

<b>Escape T</b> Transmits a Picture	<ESC>T<SPACE><Number><CR> \$1B \$54 \$20 \$64 \$0D hexadecimal characters #27T 100#13 hidden command
<b>Escape S</b> Object detection parameters	<ESC>Sc s d<CR> c - searched color {R, G, B} s - object size {0...255} d - detected object {0, 1} \$1B \$53 \$31 \$30 \$30 \$20 \$31 \$30 \$20 \$0D #27SR 100 10 1#13
<b>Escape P</b> Position of the detected object	<ESC>P<CR> the picture sensor gives the x coordinate of the detected object x {0...176, 0 := no object found}
<b>Escape I</b> Command to the I2C bus at the RoboEye Sensor Module	<ESC>In m<CR> \$1B \$49 \$15 \$40 \$0D changes the color output sequence (blue picture) \$1B \$49 \$15 \$41\$0D default value (natural picture color)

### RoboEye Sensor Module Escape Commands

The most interesting commands are the T, S and P commands. See the "Object Detection" section for more detailed use of the S and P commands.

RoboEye has several other very useful commands which can be activated from the RoboEye software or from the BASIC Stamp.

To avoid data conflicts the I/O line works in half duplex mode. In output mode the UART is deactivated. Under some circumstances it would be very helpful to get more access to the RoboEye Sensor Module by sending and receiving commands at the same time. For example you might want to use some of the escape commands without external hardware. RoboEye has an "echo" command for this purpose. To test type "echo" on the terminal of the PC software and press enter. The RoboEye Sensor Module answers "Echo On". Now you can use the escape sequences, like "#27SR100 10 1#13". The # character signals that the following numbers specify the ASCII code of the character (#27 := \$1B := <ESC>). The mode ends by sending "noecho".

<b>Echo On / Echo Off</b> Send and receive commands at the same time	Type "echo" in the RoboEye software terminal to enable the use of the escape sequences. Type "noecho" to turn off.
---	---

### Echo On / Off Command

Two other commands are available, "ping" and "color". The "ping" command returns the revision number of the RoboEye firmware. The "color" command is very helpful to specify the search parameter for the 'S' command.

<b>Ping</b> Obtain RoboEye firmware revision number	Type "ping" in the RoboEye software terminal to obtain the RoboEye Sensor Module firmware revision number.
<b>Color</b> Used to obtain the average color	Type "color" in the RoboEye software terminal to obtain the average color present.

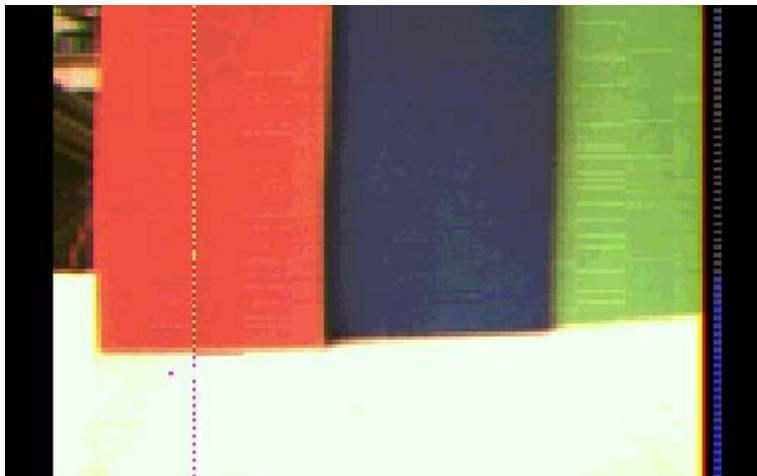
### Ping and Color Commands

### Object Detection with RoboEye

RoboEye's camera sensor is controlled by a powerful 16 bit microcontroller. Most of the processor's power is needed for data handling between the CMOS picture sensor and the wireless radio. But a portion of the program is reserved for picture processing.

The microcontroller is able to detect the x coordinates of a specified object. It helps to have a brief overview of the object detection algorithm.

The picture pixel data comes in line by line at a very high speed. Suppose you want to search out the red color.



Each pixel consists of three color components: red, green and blue. A red pixel has a "red" value that is larger than the green and blue values. In our example the microcontroller looks at all pixel values in the line. If it finds a red pixel, the x coordinate for that pixel will be stored.

But how can the RoboEye Sensor find a colored object? Objects with more than one red pixel in the line are marked so that only areas with many red pixels following each other are noted. The algorithm measures the sizes of the areas in every row, marks the largest ones and computes the main emphasis (x coordinate) of every area. The average value of all x coordinates is now the position of the object.

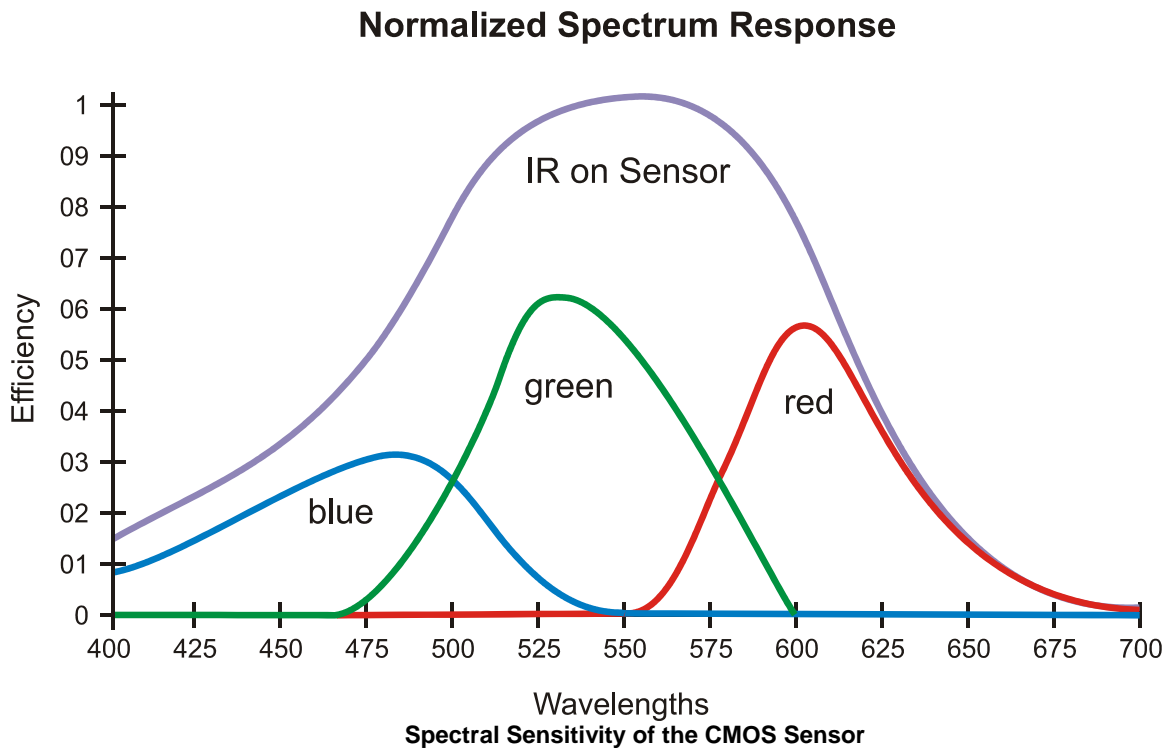
But what will happen if two nearly same sized red objects exist in different positions? Try it out.

Because of the limited computational power, the search algorithm is as simple as possible. Only green, red or blue objects can be found. During development of the search algorithm it was very easy to read out the found color pixel value from the microcontroller.

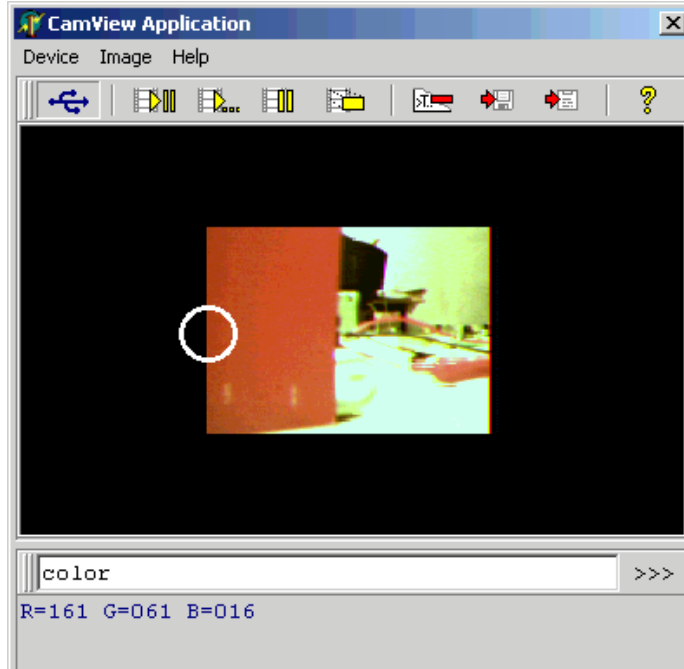
But the CMOS sensor used has its own idea of what is red. What does the sensor do? Looking into the data sheet shows how the sensor works.

Each pixel has a color filter, so that pixels with red, green and blue sensitivity exist. The sum of the three colors results in the real (natural) color of the picture detail. That looks easy, but it isn't. One of the problems is the "white balance". If you are sitting at a candlelight dinner (with ordinary candles), your eyes will have a few problems separating colors.

With a CMOS sensor, an electronic system has to build the white balance, so that colors can be detected. It is also very important to know what is red, or green or blue from the view of the sensor. That is the function of the hidden command "color".



Suppose you need to know "how" red or green or blue the object is. The exact object color value gives you some information if the search algorithm detects something.



**The White Circle Specifies the Measurement Area (Note, the circle wasn't shown in picture)**

See the white circle at the left of the above picture. From this area the RoboEye Sensor Module gives you the values of the red, green and blue pixel. In the example the CMOS sensor measures a value of 161 for red, 61 for green and 16 for blue.

Now you can specify your search parameter. For example, you choose `<ESC>SR 80 10 1<CR>`. The value 80 tells the search algorithm to look for a red pixel with a value greater than 80 from that of the green or blue component. In other words, only pixels with  $(red > green + 80)$  and  $(red > blue + 80)$  are detected. The value 10 finds pixels with a value greater than 10 from that of the green or blue component. The number 1 in the last argument tells the controller to mark found objects with a vertical line.

If you chose 120, instead of 80, no object will be found because no pixel's red component minus the green and/or blue components is greater than 120. With a value of 5, instead of 80, the difference is too small to separate objects. In this case everything is an object, the result of the found x coordinate is useless.

The detecting algorithm is much faster than the picture transmission. The green LED on the RoboEye Sensor Module's printed circuit board flashes when no object is detected and activates when an object is found.



## Boe-Bot Control Example

This example program demonstrates how to control the Boe-Bot using RoboEye's terminal. The program is also available for download from the RoboEye web page at [www.parallax.com](http://www.parallax.com). Once connected, press "reset" on the Board of Education. Type either F, B, L or R for direction followed by "enter" or the >>> button to send a carriage return.

Enter distances in three digits, in decimal format. In other words, a distance of 20 should be entered as 020 and so on.



## RoboEyeBoeBot.bs2 Example Program

```
' -----[ Title ]-----  
' Basic Boe-Bot Control with RoboEye - RoboEyeBoeBot.bs2  
' Control the Boe-Bot with the RoboEye Terminal while observing incoming  
' video feed.
```

```

' {$STAMP BS2}                                ' Stamp directive
' {$PBASIC 2.5}                               ' PBASIC directive

' -----[ Variables ]-----
pulseCount    VAR    Byte                    ' FOR...NEXT loop counter
direction     VAR    Byte                    ' Direction control
distance      VAR    Byte                    ' asdf

' -----[ Initialization ]-----
' -----[ Main Routine ]-----

controlPanel:
  SEROUT 15, 84, ["Direction?",CR]
  SERIN 15, 84, [direction]
  PAUSE 10
  SEROUT 15, 84, ["Distance?",CR]
  SERIN 15, 84, [DEC3 distance]
  IF direction = "F" THEN Forward_Pulse
  IF direction = "B" THEN Back_Up
  IF direction = "L" THEN Turn_Left
  IF direction = "R" THEN Turn_Right
  GOTO controlPanel

' -----[ Subroutines ]-----

Forward_Pulse:                                ' Send a single forward pulse.
  FOR pulseCount = 0 TO distance
    PULSOUT 12,650
    PULSOUT 13,850
  PAUSE 20
  NEXT
  RETURN

Turn_Left:                                    ' Left turn, about 90-degrees.
  FOR pulseCount = 0 TO distance
    PULSOUT 12, 650
    PULSOUT 13, 650
  PAUSE 20
  NEXT
  RETURN

Turn_Right:                                   ' Right turn, about 90-degrees.
  FOR pulseCount = 0 TO distance
    PULSOUT 12, 850
    PULSOUT 13, 850
  PAUSE 20
  NEXT
  RETURN

Back_Up:                                       ' Back up.
  FOR pulseCount = 0 TO distance
    PULSOUT 12, 850
    PULSOUT 13, 650
  PAUSE 20
  NEXT
  RETURN

```

## Data Protocol Between PC and RoboEye Sensor Module

The data transmission protocol is packet oriented, meaning that each message is organized into a formatted packet before transmission. The protocol used by RoboEye depends on the task that has been accomplished.

The wireless message is defined by the technical specifications of the radio modem. The modem works in the "shock burst" mode. In this mode 28 bytes are sent as the packet of one message. Every message contains a unique ID, address and a CRC value. Depending on the environment, messages can be corrupted by other radio modems, like Bluetooth or a Wireless LAN. Corrupted messages are ignored by the receiver and their data is lost. Corrupted messages are not repeated automatically.

The following example illustrates the structure of a message. The message contains of two header bytes followed by the data packet. CRC and address bytes are added and removed by the radio.

```

/*****
/* Structure of the wireless message protocol */
/*
/* SYNC | ID | NR | 12 pixel (26 Byte) |
/* 0xdb 0x4n 10 GR GB GR GB GR GB GR GB GR GB GR GB |
/*
/* | |--- blue
/* | |--- green
/* |----- red
/* Color format: 4:2:2
/*
/* SYNC - 0xdb start of message
/* ID - indicates the type of the payload, i.e. 0x5n == picture data
/* n (5 bit low nibble) contains the column number
/* NR - row number (0...72 for actual size)
/* Data - pixel data (12 x 2 byte = 12 pixel)
*****/
```

Because the picture sensor also transmits data from the built-in UART, other ID numbers indicate different data types. The different numbers and the type of the messages have to be defined. The PC receives the messages and creates the picture. If some messages are lost, the referring picture part isn't refreshed. Messages from PC to picture sensor have a similar structure.

```

/*****
/* Structure of the messages to the picture sensor */
/*
/* HEADER | ID | raw data (size as given in the header) |
/* 0b.11nn.nnnn 0xkk x x x x x x ..... (size = n)
/*
/* Header - 0b.11nn.nnnn, contains SYNC (0b11) and size of the message
/* ID - indicates the type of the message
/* enum {
/* enConfigReg = 1, /* config-data for nRF2401 */
/* enDataReg /* message data */
/* };
/* xx - data
*****/
```

The transmission unit receives data at 9600 Baud. Two different data messages exist. The first type of message is a direct command to the nRF2401 radio. The data of this message type is sent to the config register of the radio.

The other type of message is sent directly to the transmitting register of the radio. In this case the transmission unit doesn't have its own intelligence. In other words, if a user wants to make his own

programs, he has full control of the radio modem. Note that the baud rate of the transmission is fixed. Incoming data (data from unit to PC) has a baud rate of 923,076 Baud. Outgoing data, from the PC to the RoboEye Sensor, has to be sent with 9600 Baud. Because the microcontroller only has a small RAM buffer, the transmission unit utilizes a hardware handshaking mechanism. After a complete message is received, the CTS signal goes high.

## Examples for using the RoboEye with your Own Software Applications

### Commands to RoboEye USB interface

	Header	Target	Message
<b>Tx-Mode</b>	211	1	142 8 28 64 224 0 0 0 0 231 0 0 0 0 231 35 239 16
<b>Rx-Mode</b>	211	1	142 8 28 64 224 0 0 0 0 231 0 0 0 0 231 35 239 1
			Command string for nRF2401

The combination of header and target number signals the USB module microcontroller that the following message should be shifted into the transceiver programming registers. That means these commands program the transceiver chip on the RoboEye USB module directly. For further information about the transceiver chip programming, see the manufacturer's website at [www.nvlsi.com](http://www.nvlsi.com) (nRF2401 chip).

### Commands to RoboEye

Adr - Address of nRF2401 (must be 231)

Cmd - Command to the RoboEye USB-Module

- 1 - Message is sent directly to UART (variable packet size)
- 2 - Message is sent to I<sup>2</sup>C (size must be 3 bytes)
- 3 - transmit command

Par - Parameter (varies depending on used command)

ID - ID number (new message get a new number)

	Header	Target	Adr	Cmd	Par	ID	Message	unused
<b>UART1</b>	222	2	231	1	10	1	2 3 4 5 6 7 8 9 10 11	15 bytes
<b>UART2</b>	222	2	231	1	10	2	6 6 6 6 6 7 8 9 10 11	15 bytes
<b>UART3</b>	222	2	231	1	13	3	Hello World!	12 bytes
<b>IIC1</b>	222	2	231	2		1	192 18 110	23 bytes
<b>IIC2</b>	222	2	231	2		2	192 18 108	23 bytes
<b>SendPic</b>	222	2	231	3			200	26 bytes
							Payload size must be 28 byte long	

The commands specified by header number 222 and target number 2 transmit a wireless message to the RoboEye sensor. The message payload is fixed with 28 bytes. The sent message can be structured differently.

The commands in the table above show some examples. Note, in some cases a so called ID number is needed. The ID separates messages. In example UART3 the text "Hello World!" should be transmitted from PC to the UART of the RoboEye. In noisy environments wireless messages can be lost. In this case it is highly recommended to repeat message transmitting frequently. Nonetheless the receiver doesn't know if the received message is new or only repeated. As long as the ID isn't changed, following messages are ignored. With other words, if you want to send "Hello World!" twice, the ID has to be changed between the messages. The easiest way of changing the ID is to increment the value after transmitting.

The USB interface works completely transparently. For transmitting data the RoboEye USB module has to be programmed as a transmitter. That means that the program sequence goes to the transceiver circuit. After programming the transceiver, messages can be transmitted. To receive answers from RoboEye the

USB unit has to be programmed into receiving mode. For programming the RoboEye USB module interface, a baud rate of 9600 Baud must be used. The same baud rate has to be used for transmitting messages.

But in receiving mode, a complete other baud rate has to be programmed. Normally higher baud rates are multiplied numbers of the often used baud rate of 9600 Baud, i.e. 19200 Baud, 38400 Baud (the highest baud rate of the normal serial interface of the PC is 115200 Baud: 12 x 9600 Baud). Because of the high data volume of the pictures the USB module uses a baud rate of 923.076 Baud. The exact multiple number should be 921600 Baud (96 x 9600 Baud), but the USB circuit runs with a clock of 6MHz, so that only a small error occurs.

## **Special Technical Considerations**

The range of the wireless radio depends heavily on environmental conditions. In direct line of sight the range can be up to 100 feet or more. Obstructions between the transmitter and receiver, like human bodies, reduce the range. At the furthest range the error rate of the data transmission increases. This is seen when only a few percent of the picture is transmitted back to the PC.

RoboEye's data messages are secured using a fixed address, a rolling ID and CRC check. But noise at the receiver input can still produce a valid message that contains incorrect data. The content of such a message might include characters like "5fdg#~^" or others you did not expect.

Avoid touching RoboEye's electronic components and antenna. Electrostatic charges can damage the electronic components.

BASIC Stamp, Boe-Bot and Board of Education are federally registered trademarks of Parallax Inc.