

JROBOT

# AVRcam

## User's Manual



Version 1.2

JROBOT

## Table Of Contents

1	Revision History .....	3
2	Copyright/Licensing/Disclaimer Info .....	4
3	Introduction .....	5
4	Building the AVRcam .....	7
4.1	Assembly Instructions for the Serial and Power Cables .....	7
4.1.1	Assembling the Power Cable .....	7
4.1.2	Assembling the Serial Cable .....	8
4.2	Assembly Instructions for the AVRcam .....	11
4.3	Before Beginning Assembly .....	12
4.4	Validating the AVRcam Assembly .....	14
5	Installing the AVRcamVIEW PC Software .....	15
5.1	Windows Installation .....	15
5.2	Linux Installation .....	15
6	Communicating with the AVRcam .....	16
6.1	Electrical Interface to the AVRcam .....	16
6.2	Protocol Overview .....	16
6.3	AVRcam Modes .....	18
6.3.1	Idle Mode .....	18
6.3.2	Frame Dump Mode .....	18
6.3.3	Color Tracking Mode .....	18
6.4	Testing the AVRcam User-Interface .....	18
6.5	Testing the AVRcam with AVRcamVIEW .....	19
6.6	How Lighting Affects the AVRcam .....	22
7	Advanced Development with the AVRcam .....	24
7.1	Programming the AVRcam .....	24
8	Appendix A .....	25
9	Appendix B .....	26
9.1	Complete AVRcam Command Protocol .....	26
10	Appendix C .....	30

# 1 Revision History

Date	Version	Author	Update
11/06/2004	1.0	John O	Initial release
12/02/2004	1.1	John O	-Added pictures and instructions for cable assembly -Added more detail in section 7 for in-system reprogramming -verified all previously marked CHECK items ☺
02/07/2005	1.2	John O	-Updated the schematic, board layout, and assembly instructions so they match version 1.2 of the hardware (the only change is that the 0.1 uF caps got new “Cxx” numbers on the schematic).

## 2 Copyright/Licensing/Disclaimer Info

AVRcam Hardware and Embedded Software Copyright © 2004 John Orlando  
AVRcamVIEW PC Software Copyright © 2004 Brent Taylor  
All material in this document is Copyright © 2004 by John Orlando

Both the AVRcam embedded software and the AVRcamVIEW PC software are licensed under the GNU General Public License. If interested in an alternative licensing scheme, please contact John Orlando at [john@jrobot.net](mailto:john@jrobot.net).

The AVRcam system is distributed in the hope that it will be useful. However, no warranties, either expressed or implied, are made regarding the operation, use or results of this system. The AVRcam should not be used in any life-critical applications.

### 3 Introduction

Image processing has always been a task left for systems that have a considerable amount of computing resources to provide any reasonable functionality. These systems typically require fast processors, lots of RAM for storing images, and large amounts of power while images are being processed. The end result is a fairly complicated, expensive system that is out of reach for many developers who have moderate image processing requirements.

The AVRcam was designed to fill this gap in image processing.

The AVRcam, shown in Photo 1, is a real-time image processing engine that utilizes an Omnivision OV6620 CMOS image sensor, mated with an Atmel AVR mega8 microcontroller to perform all of the image processing tasks. It can be thought of as a vision sub-system, with a well-defined interface that is accessible through a standard serial port. This interface provides high-level, post-processed image information to a primary system (PC, another embedded controller, BASIC Stamp, etc.) to utilize. This removes the burden of performing the image-processing on the main system, and allows the developer to concentrate on how to use the high-level image data to perform the task at hand.



Photo 1: Front view of the AVRcam v1.1

The AVRcam provides an assortment of capabilities:

- ◆ Track up to 8 different colorful objects at 30 frames/second
- ◆ Configure the system to recognize up to 8 different user-defined colors
- ◆ Provide real-time tracked object statistics (number of objects, color of objects, bounding box) through a standard serial port
- ◆ Tracked image resolution of 88 x 144 pixels at 30 frames/second
- ◆ Perform full-resolution color image dumps
- ◆ Dumped image resolution of 176 x 144 pixels
- ◆ Low power consumption (the entire system only draws 57 mA)
- ◆ Small size (the entire system is 2.4" x 1.9")

## JROBOT

In addition, a PC application called AVRcamVIEW was developed to support the AVRcam for purposes of evaluating and testing the system. The AVRcamVIEW provides the following capabilities:

- ◆ Take full-color snapshots (176 x 144 pixels) with the system and display the images (both raw Bayer data and interpolated color data)
- ◆ Easily create a Color Map of colors to track based on a snapshot (just click on the colors of interest and add them to the Color Map)
- ◆ Adjust the precision of each tracked color (i.e. provide a range of acceptable R-G-B values for each color), allowing the user to adjust the Color Map to the surrounding environment
- ◆ Display the real-time tracking results of each tracked object (with color and bounding box information)
- ◆ Record a tracking session for playback at a later time
- ◆ Test the system out in multiple OS platforms that are supported by Java 1.5 (both Windows and Linux are currently supported)

Finally, one of the most important benefits to the AVRcam is that it is completely open-source: hardware, software, design docs, everything. In this regard, the AVRcam should also be considered as a development platform for not only the existing capabilities provided by the system, but new features that are tailored to the individual needs of a developer. A set of core routines are provided in the source code that facilitate all of the low-level interactions with the CMOS image sensor. These routines can be utilized in a custom image processing application running on the AVRcam, within the space/speed capabilities of the system. The existing AVRcam embedded software (with all the features mentioned above) occupies approximately 4K of the 8K on-chip flash memory, 700 bytes of the 1K on-chip RAM, and 48 bytes of the 512 bytes on-chip EEPROM. The system allows for complete in-circuit re-programmability of the image processing software using the standard 10-pin AVR ISP programmer. All of the software on the AVRcam was developed using GCC (an open-source, free C compiler that has been ported to the AVR family of microcontrollers), meaning that the development environment for the AVRcam is completely free for any and all to use. For more information, see section 7 of this document.

Hopefully the AVRcam will open up many new possibilities to the developer for easily adding a vision system to your next project. Please check <http://www.jrobot.net> for discussions on the AVRcam, as well as technical details and the latest source code.

## 4 Building the AVRcam

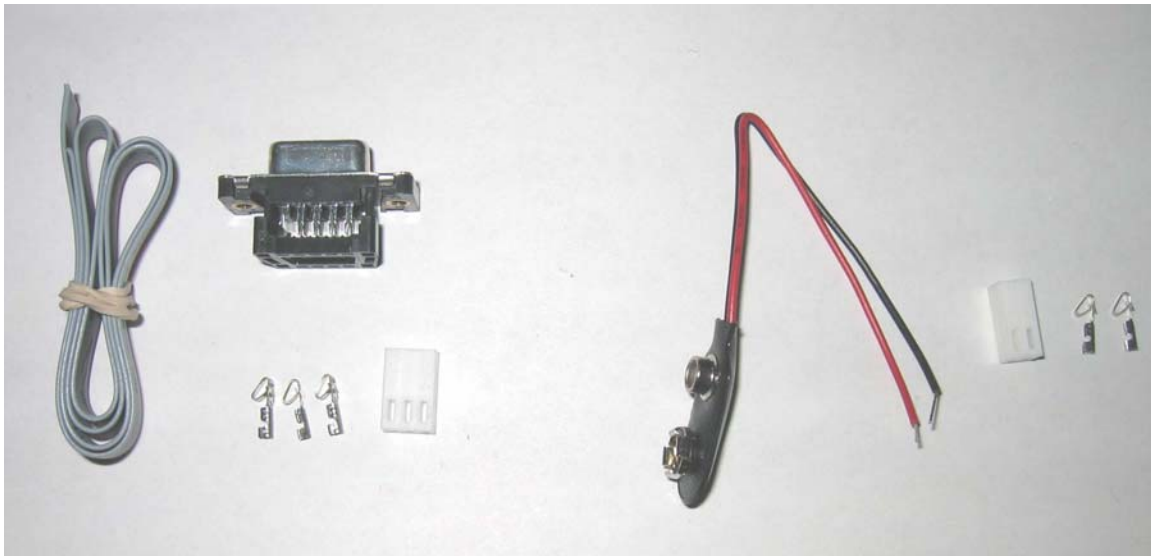
The AVRcam consists of several different components that should have been shipped with the system. The list of components is listed below.

- AVRcam bare circuit board
- Assortment of electronic components (see Appendix A for a complete list of all components)
- CD containing the AVRcamVIEW PC software, as well as this User's Manual, complete source code for both the AVRcam and AVRcamVIEW software, and the WinAVR development environment containing the GCC C compiler and instructions for using it
- Power supply cable with connector (to power the AVRcam)
- Serial port cable with 9-pin D connector

If the system was purchased already assembled and tested, then the AVRcam circuit board will be populated, with the assortment of electronic components already soldered and tested to be functional.

### 4.1 Assembly Instructions for the Serial and Power Cables

If the kit form of the AVRcam was purchased, the serial and power cables have not been assembled yet. The unassembled view of the cables can be seen below.



#### 4.1.1 Assembling the Power Cable

- 1) Connect the two metal crimp connectors to the end of the 9-volt battery clip. This can be done with a standard crimper, or a pair of needle nose pliers. If desired, a small amount of solder can be applied to hold the connection in place. The result should look like the following picture:

## JROBOT



- 2) Next, insert the two crimped ends into the white Molex housing, ensuring that the angled catch on the metal connectors locks into place in the Molex housing:



The power cable is now complete.

### 4.1.2 Assembling the Serial Cable

- 1) Connect the female 9-pin serial connector to the 9-strand ribbon cable by feeding the ribbon cable through the serial connector. NOTE: It is important to make sure that the ribbon cable is lined up with the “teeth” in the 9-pin serial connector to ensure proper connection. Once completed, the cable should look like the following:



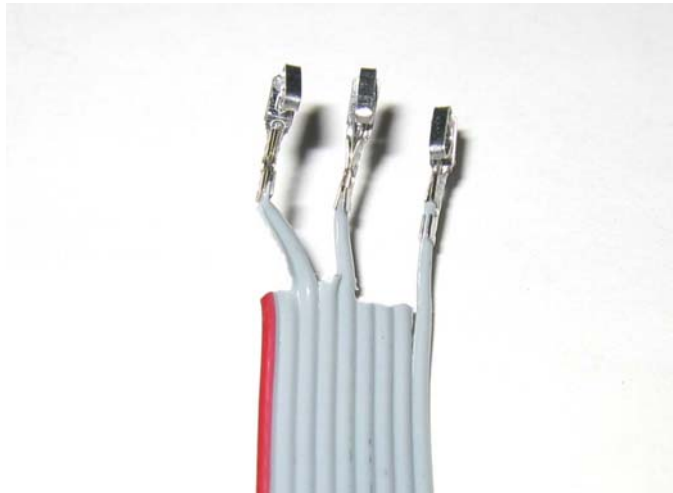


## JROBOT

- 2) Next, the opposite end of the ribbon cable needs to have some of its wires shortened. The serial connection only needs 3 of the 9 (transmit, receive, and ground) signals. Cut and strip the opposite end of the cable so that it looks like the following photo:

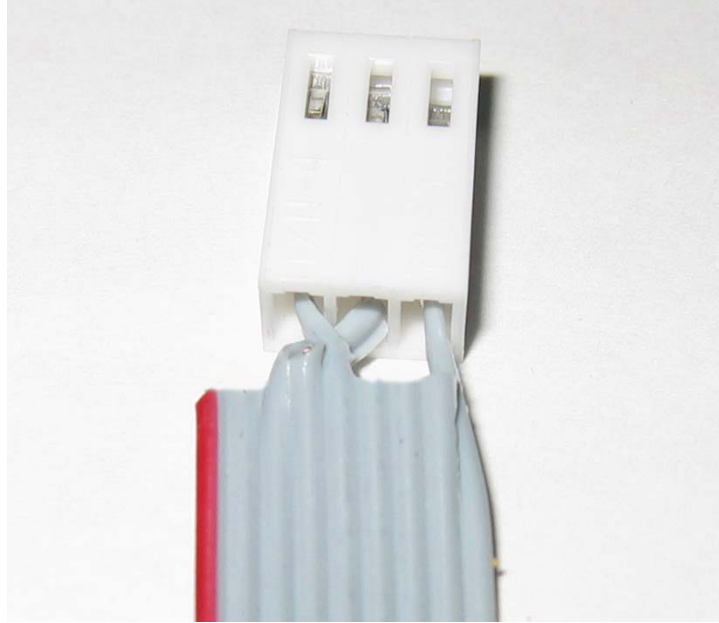


- 3) Connect the three metal crimp connectors to the exposed end of the wire using a crimp tool or a pair of needle-nose pliers. Again, a small amount of solder can be used to secure the connection if needed. The result should look like the following photo:

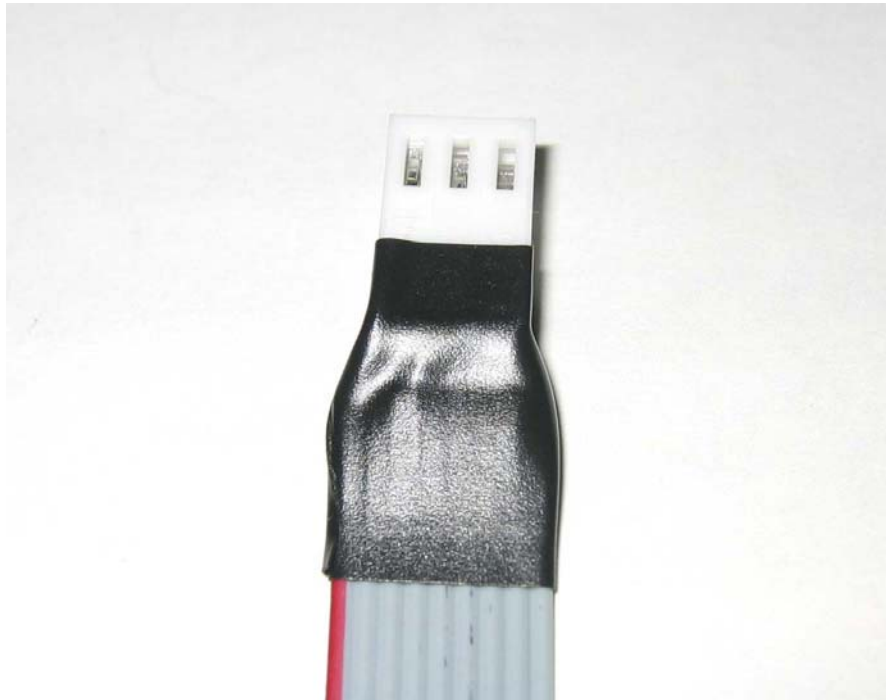


- 4) Insert the three metal crimp connectors into the white Molex housing, ensuring that the catch on each connector locks securely in place. NOTE: This step requires that second and fourth wire (from the red strip in the ribbon cable) are crossed over before inserting into the housing, as shown in the following photo:

## JROBOT



- 5) Slide the heat-shrink tubing over the Molex connector, so that it protects the ribbon cable where it connects to the housing. Apply heat to this tubing (with either a heat gun, or a hair-dryer) so that it shrinks and seals up this cable, as shown in the following photo:



The serial cable is now complete. Both completed cables are shown in the following photo:



## 4.2 Assembly Instructions for the AVRcam

The AVRcam can be assembled with only a basic knowledge of soldering and electrical components. The complete parts placement diagram can be found in Figure 1.

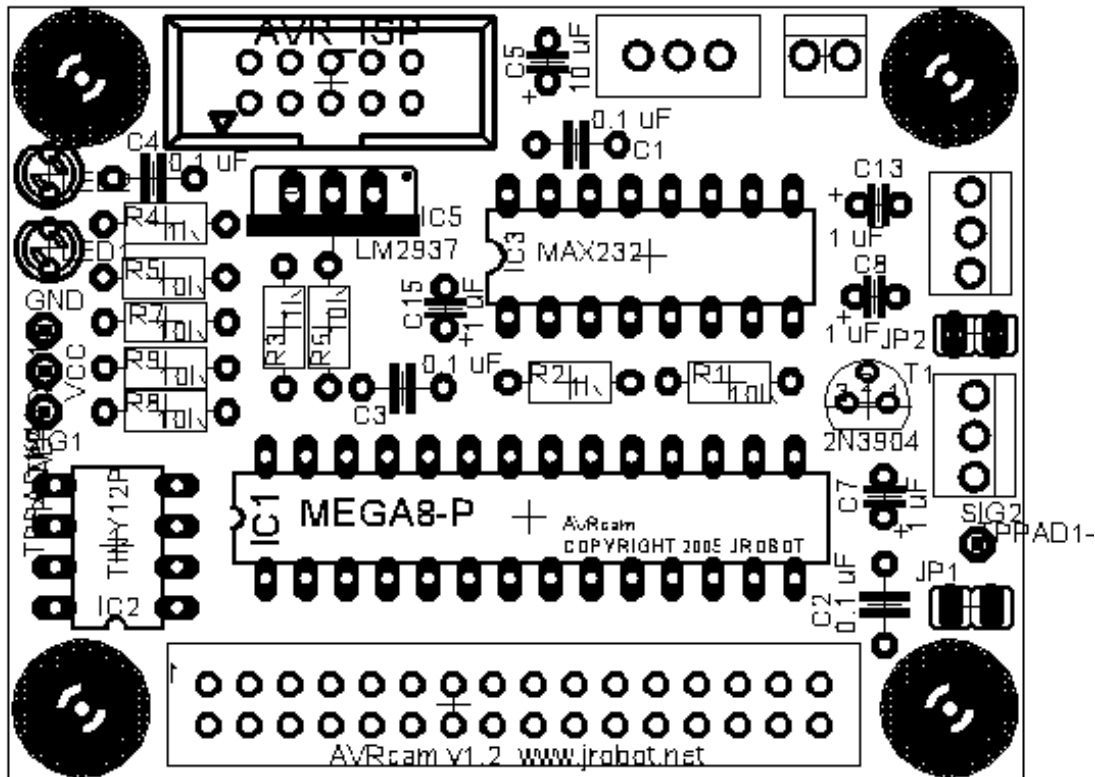


Figure 1: AVRcam v1.2 circuit board layout

### 4.3 Before Beginning Assembly...

As the board is being assembled, it may be helpful to the user to secure each component to the circuit board before soldering it down. Small strips of tape work very well to hold the components in place while soldering.

All components are placed on the BACK side of the board, with the exception of the 32-pin dual row camera header, which is placed on the FRONT side of the board. See Photos 2 and 3 for reference. Photo 2 also provides a reference for right/left/top-edge/bottom-edge when assembling the board.

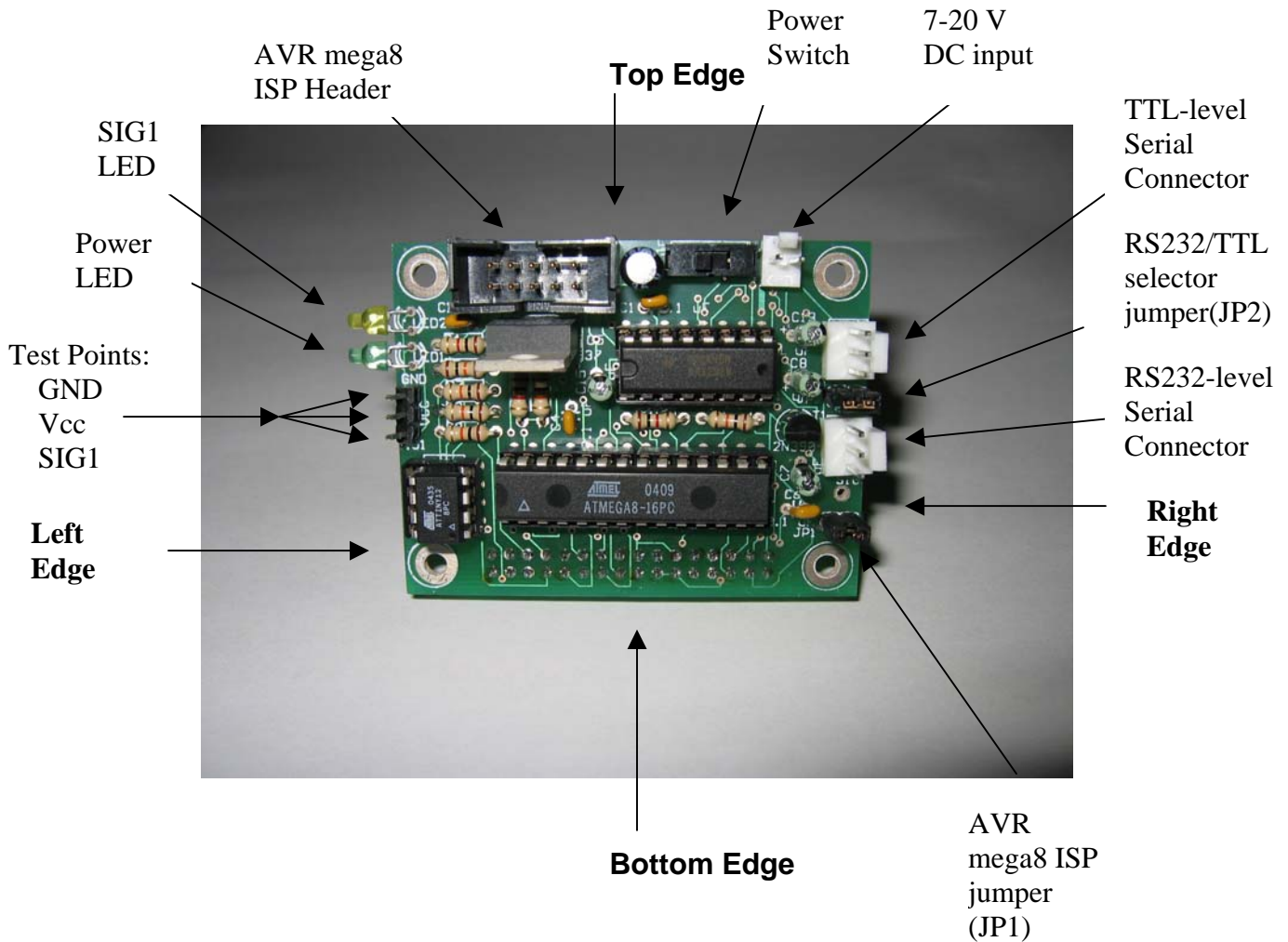


Photo 2: BACK side of an assembled AVRcam

After each component is placed and soldered, it may be necessary to clip off any excess portion of the leads for the components.

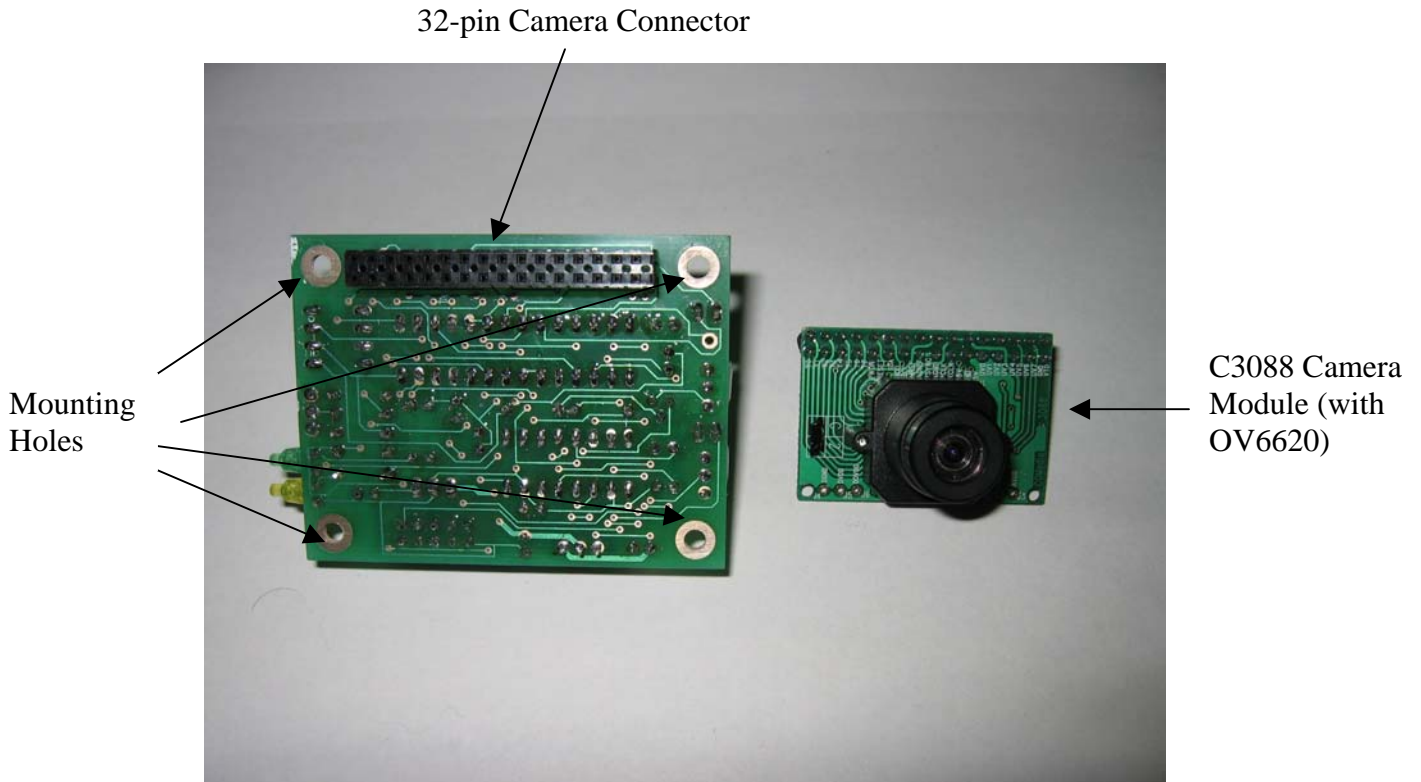


Photo 3: FRONT side of assembled AVRcam, with camera detached

The AVRcam should be assembled in the following order:

- 1) Insert and solder R1, R2, R3, R4, R5, R6, R7, R8, and R9.
- 2) Insert and solder C1, C2, C3, and C4. None of these capacitors are polarized, so their orientation is not critical. NOTE: It may be necessary to bend the leads to these capacitors slightly inward so that they will fit into their holes.
- 3) Turn the board over, and insert the 32-hole dual row female camera header on the FRONT side of the board. This should be the only component placed on the front side of the board, with it being soldered on the BACK side of the board. When finished, turn the board back over to continue working on the BACK side of the board.

The DIP sockets for holding IC1, IC2, and IC3 will be connected next. It is important to note that the actual ICs will be added to the circuit at the end of the assembly process after everything on the board has been tested. Again: DO NOT INSERT IC1, IC2, OR IC3 UNTIL AFTER THE BOARD HAS BEEN TESTED! Otherwise, there is a potential to destroy the ICs.

- 4) Insert and solder the two 14-pin DIP sockets to form a 28-pin DIP socket for IC1. Make sure that the notch on the socket is properly oriented, pointing to the left-edge of the board.
- 5) Insert the 8-pin DIP socket into the board for IC2. Solder it down, ensuring that the notch on the socket is properly oriented (pointing to the bottom-edge of the board).

## JROBOT

- 6) Insert the 16-pin DIP socket into the board for IC3. Solder it down, ensuring that the notch on the socket is properly oriented (it should be pointing to the left-edge of the board).
- 7) Insert and solder IC5 to the board. Make sure that the front of this chip (where it says "LM2937") is facing the top-edge of the circuit board.
- 8) Insert and solder the AVR\_ISP programming header. Make sure that the **CENTER** notch in the header is not facing the outer edge of the circuit board. If the AVR\_ISP programming header also has an offset notch, this should be facing the outer edge of the circuit board.
- 9) Insert and solder C5, C7, C8, C13, and C15. Note that all of these capacitors ARE polarized, and the positive terminal (+) should line up with the positive sign (+) shown on the circuit board.
- 10) Insert and solder T1, the 2N3904 transistor. Make sure that the flat edge of the transistor is facing the bottom-edge of the circuit board.
- 11) Insert and solder the two 3-prong male Molex headers on the right edge of the board. These headers are used to communicate with the serial port (one for TTL-level signalling, and one for RS232-level signalling). Make sure that the tall back of each header is closest to the right-edge of the board.
- 12) Insert and solder the only 2-prong male Molex header that is used to deliver a power source to the circuit. Make sure that the tall back of the header is closest to the top-edge of the board.
- 13) Insert and solder LED1 and LED2, the green and yellow LEDs. The yellow LED sits closer to the top-edge of the board, and the green LED goes right under it. The LEDs are polarized, so their orientation on the board is important. The longer leg of each LED is the anode, and should be inserted so that the anode is closer to the bottom-edge of the board for each LED.
- 14) Insert and solder both JP1 and JP2, the two 2-pin jumpers. Take extra care to make sure they are standing perpendicular to the board when soldered down, since they can easily wiggle while performing the soldering. Once these jumpers are soldered down, place the actual jumper inserts to connect each jumper together.
- 15) Insert and solder the 3-pin male header that provides Vcc, GND, and SIG1 close to the left-edge of the board. Again, take extra care to make sure this header is standing perpendicular to the board when soldering it down.
- 16) Insert and solder the 1-pin male header that provides SIG2 near the right-edge of the board. This signal is an extra signal that can be used by the user if needed. Again, take extra care to make sure this header is standing perpendicular to the board when solder it down.
- 17) Insert and solder the SPST power switch close to the top edge of the board. There is no orientation for this component.

Once all of the components have been placed, the bottom of the board should be visually inspected to ensure that all solder joints look clean and shiny. The board should also be carefully inspected to ensure that no solder bridges have accidentally been formed, connecting signals together that weren't meant to be connected. If there are any questionable connection, or solder bridges, use a solder-removal tool (solder sucker, solder wick) to clean up the offending connection.

### 4.4 Validating the AVRCam Assembly

Once this inspection is complete, use an ohmmeter to measure the resistance between the VCC and GND header pins near the left-edge of the board to ensure that they are not shorted together. Use the ohmmeter to carefully trace out each connection in the circuit if a short is found.

Plug in the header-end of the AVRCam power cable to the 2-prong male Molex header near the top-edge of the board. Connect a suitable power supply (such as a 9-volt battery, or other voltage source between 7 and 20 Volts) to the supply-end of the AVRCam power cable.

Switch the power switch on, and the green LED should light up. If it doesn't, disconnect the power cable immediately, and go back and re-check all connections on the board. If the green LED does light up, use a voltmeter to measure the voltage between the VCC and GND headers near the left-edge of the board. The voltage should be close to 5 Volts. If not, disconnect the power cable immediately, and go back and re-check all connections on the board.

Once the proper voltage is measured between VCC and GND, disconnect the power from the circuit by turning the power switch off. Then, insert IC1, IC2, and IC3 into their appropriate sockets on the board, making sure that the notch on the IC matches the notch on the socket (see Photo 2). It may be necessary to bend the legs on each IC slightly to make it fit into the socket.

Once the ICs are inserted, switch the power switch back on. Using a voltmeter, measure the voltage between VCC and GND again, making sure that it is still close to 5 V. If not DISCONNECT the power cable immediately, and go back and check all connections on the board.

Once the proper voltage is measured between VCC and GND, turn the power switch off. Turn the board over, and insert the OV6620 camera module into the 32-pin female header on the FRONT of the board (see Photo ). Remove the lens cap on the camera if it is still connected.

Finally, turn the power switch back to the ON position. The green LED should light up, and then about two seconds later, the yellow LED should blink three times and then stay turned on. This indicates that the board has successfully powered on and is running. If not, turn the power switch back to the off position. Wait a few seconds, and try turning it on again. If the yellow LED still doesn't light up, disconnect the power cable immediately, and re-check all connections on the board.

## 5 Installing the AVRcamVIEW PC Software

The enclosed CD contains the both the Windows installer as well as the Linux RPMs for the AVRcamVIEW PC application. This installer will install the Java Runtime Environment (JRE) to the directory where the software is installed, so that it will not cause a problem if other Java installations exist on the PC.

### 5.1 Windows Installation

Insert the CD into the PC, and navigate to the Windows directory. Double-click on the setup.exe file. This will install the AVRcamVIEW application and place an shortcut on the Windows desktop. The AVRcamVIEW application can then be launched by double-clicking on this icon.

### 5.2 Linux Installation

It is necessary to install and run the AVRcamVIEW software with root privledges, to allow for reading/writing to the serial port under Linux. So, if not already done, log in as root. Next, insert the CD into the PC, and mount the CD-ROM drive if not already mounted. Change directories to the Linux directory on the CD, and, from a shell, install the RPMs for the AVRcamVIEW software using the following command (if a newer version of the AVRcamVIEW application is on the CD, replace 01.03 with this name):

```
> rpm -i AVRcam-01.03-1.i586
```

This will automatically install the AVRcamVIEW application under /opt/AVRcam. To launch the AVRcamVIEW application, change directories to /opt/AVRcam, and execute the following command:

```
>jre/bin/java -jar AVRcam.jar
```

## 6 Communicating with the AVRcam

### 6.1 Electrical Interface to the AVRcam

The AVRcam provides its interface to an external host system through a UART-based serial port. This serial port can operate at both TTL voltage levels as well as RS232 voltage levels, depending on the configuration of Jumper2 (JP2) near the right-edge of the board (see Photo 2). If the AVRcam will be communicating with a PC that uses standard RS232 voltage levels, the JP2 jumper should be inserted on the board. If the AVRcam will be communicating with another microcontroller, or another system using TTL voltage levels, JP2 should be removed from the board. The pinouts for the both the RS232 and TTL serial ports is shown in Photo 4.

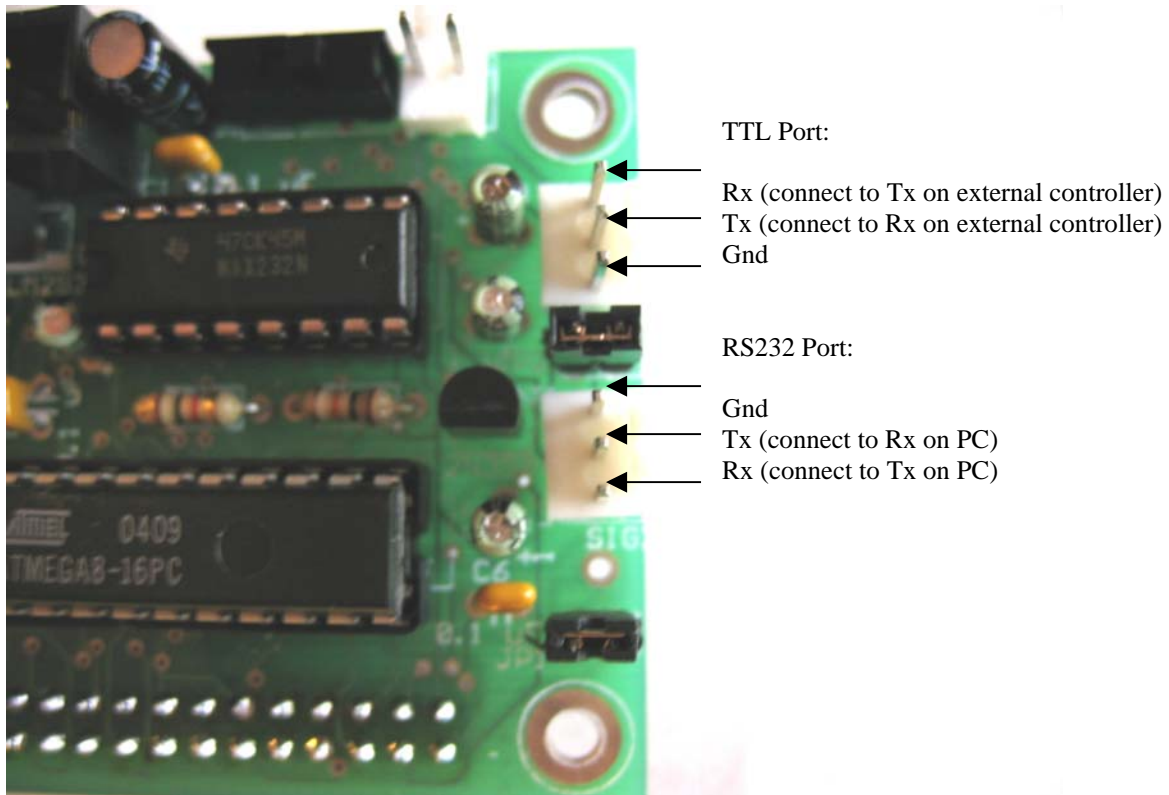


Photo 4: Pinouts of TTL and RS232 serial ports

The protocol used to communicate with the board is a very simple command/response interface. Commands are sent to the AVRcam through transmit line (TX) on the serial port, and responses are returned from the AVRcam through the receive line (RX) on the same serial port. It should be noted that no flow-control is used on the AVRcam. The only three lines needed to communicate with the system are the TX, RX, and GND. Currently, the AVRcam only supports a baud rate of 115.2 kbps, with 8 data bits, 1 stop bit, and no parity. The entire AVRcam command protocol is described in Appendix B.

### 6.2 Protocol Overview

A typical AVRcam command consists of a simple two-character ASCII operation-code (op-code), followed by optional ASCII parameters, followed by a carriage-return character. The AVRcam



## JROBOT

then responds with either a positive acknowledgement (ACK) indicating the command was successful, or a negative acknowledgement (NCK), indicating that the command was not successful and should be re-transmitted.

For example, if the user wants to ping the AVRcam to ensure that the board is functioning properly, the PING command can be sent to the board. The AVRcam should then respond with a positive acknowledgement indicating that the board is functioning. This example is illustrated below:

```
<host system> PG\r
<AVRcam> ACK\r
```

Here, the host system is able to determine that the AVRcam is alive and operating properly because it responded with an positive acknowledgement. Note: The PING command has the op-code "PG". For a complete list of op-codes, and their meaning, see Appendix B.

Some command/response pairs can also cause the AVRcam to provide a supplemental line of information in response to the received command. For example, the host system can request the software version of the AVRcam through the GET VERSION command ("GV"). The response to this command contains the normal acknowledgement, immediately followed by the version information on the next line. This example is illustrated below:

```
<host system> GV\r
<AVRcam> ACK\r
<AVRcam> AVRcam v1.0\r
```

Finally, in addition to the normal command/response protocol, the AVRcam has different modes that can be invoked that enable additional information packets to be sent from the AVRcam to the host system asynchronously. These modes will always be invoked by a standard command/response pair. For example, to enable the color tracking mode, the user sends an ENABLE TRACKING command ("ET") to the AVRcam. The AVRcam responds with a positive acknowledgement if the command was received successfully. The AVRcam then begins to send tracking packets back to the host system to be processed. This example is illustrated below:

```
<host system> ET\r
<AVRcam> ACK\r
<AVRcam> {formatted tracking packet containing hex characters}
<AVRcam> {formatted tracking packet containing hex characters}
<AVRcam> {formatted tracking packet containing hex characters}
...
```

The above example shows how the ACK from the AVRcam is followed up with multiple tracking packets for as long as there are objects being tracked (or until the DISABLE TRACKING command is sent to the AVRcam). This example is also the first time that a non-ASCII-printable character could potentially be sent to the host system. The tracking packets contain 8-bit hexadecimal values (between 0x00 and 0xFF) in a well-defined packet format described in Appendix B. These tracking packets will not be viewable with a normal terminal emulator, since terminal emulators only display ASCII characters typically.

The complete formatting for each user-interface command/response can be found in Appendix B. This section also contains the complete formatting for the supplemental information packets that are sent by the AVRcam according to the mode of the AVRcam.

## 6.3 AVRcam Modes

The AVRcam can be put into three different modes. Each mode allows for different functionality in the system.

### 6.3.1 Idle Mode

The first mode is the Idle Mode. This is the default mode of the AVRcam. In this mode, the system is not sending any additional information packets back to the host system.

### 6.3.2 Frame Dump Mode

The second mode is the Frame Dump Mode. This mode is entered by sending the DUMP FRAME (“DF”) command to the AVRcam. This command causes the AVRcam to start sending frame-dump packets back to the host system, until an entire image frame has been sent. This mode is only temporary, while the frame-dump packets are being sent. Once all frame-dump packets for a complete frame have been sent, the AVRcam transitions back to the Idle Mode.

### 6.3.3 Color Tracking Mode

The third mode is the Color Tracking Mode. This mode is entered by sending the ENABLE TRACKING (“ET”) command to the AVRcam. This command causes the AVRcam to start sending tracking packets back to the host system. These tracking packets will continue to be sent as long as the system is in the Color Tracking Mode. If no trackable objects are found in the image stream, and thus no tracking packets can be sent back to the host system, the mode still persists. The only way to exit this mode is to send the DISABLE TRACKING (“DT”) command to the AVRcam. This causes the system to go back to the Idle Mode.

## 6.4 Testing the AVRcam User-Interface

The easiest way to get started with the AVRcam user-interface is to use a serial terminal emulator running on a PC (such as Hyperterminal, or minicom) as the host system. This allows the user to send simple commands to the board and view the responses sent by the board.

First, power-down the AVRcam if it is currently powered on. Then plug in the DB9 9-pin connector on the serial cable into the serial port on the back of the PC. Plug the other end of the serial cable into the AVRcam’s 3-pin RS232 header (see Photo 4). Turn on the power switch to the system.

Next, start up a terminal emulator on the host PC (such as Hyperterminal, or minicom). The terminal emulator should be set up to run at a baud rate of 115.2 kbps, 8 data bits, 1 stop bit, and no flow-control. Make sure that the terminal sends only a carriage-return when the user presses the Enter key (some terminals default to sending carriage-return/line-feed when the Enter key is pressed). Also, turning local-echo on will allow the user to see the commands as they are typed in.

From the terminal emulator, send a PING command to the AVRcam by typing “PG”, followed by hitting the Enter key. Pressing the Enter key is shown below through the standard convention of a ‘r’ character:

```
<host system> PG\r
<AVRcam> ACK
```

Once this is successfully received, request the software version of the system through the GET VERSION command:

```
<host system> GV\r
```

# JROBOT

```
<AVRcam>  ACK\r  
<AVRcam>  AVRcam v1.0\r
```

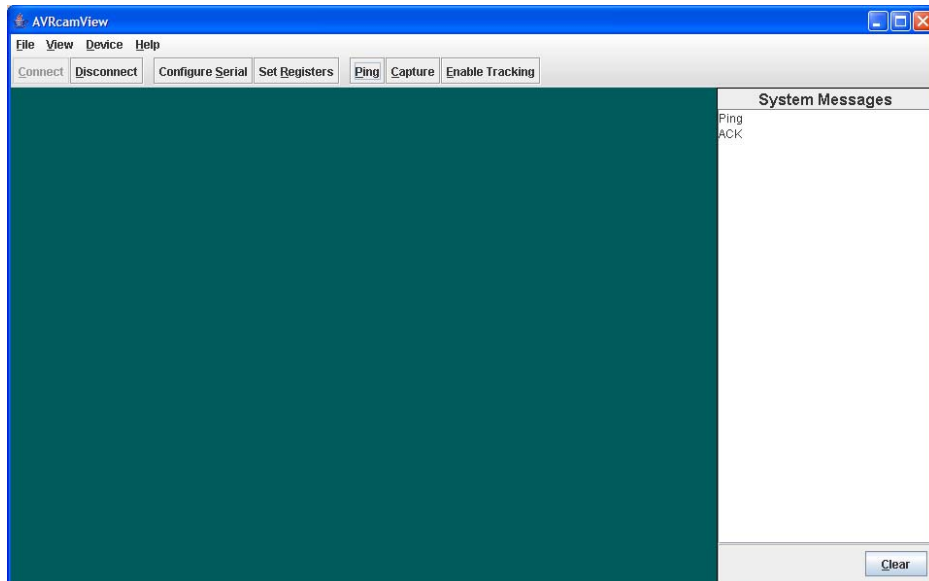
Congratulations! You have now verified that the AVRcam is functioning properly, and your serial connection is working properly. Testing the AVRcam with a terminal emulator has its limitations, since any command that causes the AVRcam to generate non-ASCII (i.e., non-printable) data will not be properly viewable on the PC. The next step is to use the AVRcamVIEW software to further test the system. But before this can be accomplished, the terminal emulator running on the PC should be closed so that the serial port on the PC will be available for use by other programs.

## 6.5 Testing the AVRcam with AVRcamVIEW

The AVRcamVIEW PC application provides a simple way for the user to easily test out all of the features of the AVRcam. Before the application can be used, it must be installed on the host PC. See Section 4 if the AVRcamVIEW software has not been installed yet.

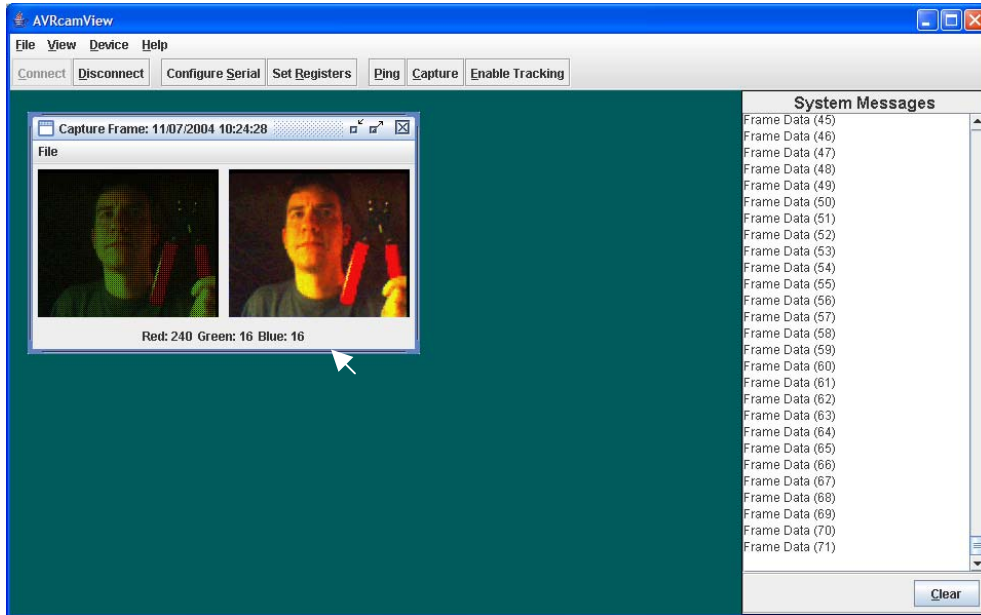
If the AVRcam isn't already connected to the PC, connect it using the serial cable and turn on the AVRcam. Next, start the AVRcamVIEW software on the host PC. This will start the application, displaying the main Command Bar, as well as the Message Viewer and the Display Window. The first thing to do with the AVRcamVIEW application is to configure the serial port for 115.2 kbps, 8 data bits, 1 stop bit, and no parity (8N1). These parameters are set through the "Configure Serial" button on the Command Bar. Next, left-click on the "Connect" button, which will allow the user to select which COM port on the PC to use. Once this is set, the first thing the AVRcamVIEW software does is to ping the camera to determine if it is present. If the AVRcam is not present, then the response timer associated with the ping command will expire after 5 seconds, and a timeout message will be displayed to the user. This indicates that the ping was not received by the AVRcam. This could be due to incorrect serial settings, wrong COM port, disconnected serial cable, or powered off AVRcam.

It is now time to send a user-initiated command from the AVRcamVIEW application to the AVRcam. Left-click on the "Ping" button in the Command Bar. The user should see the MessageViewer display update, showing that it sent the PING command, followed by the response sent back by the AVRcam (either ACK or NCK). If no response is received from the AVRcam, the command will time out after 5 seconds. It is also possible to manually inject a NCK into the system by pressing CTRL-N on the keyboard.

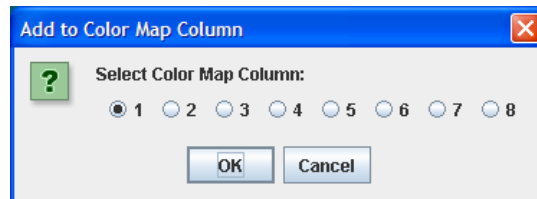


# JROBOT

Next, left-click on the “Capture” button in the Command Bar. This will take a snapshot of the camera’s current field of view, and display it. A complete image acquisition takes about 4 seconds to complete. Once all the image lines have been sent to the AVRcamVIEW application, the complete image will be shown. If the image appears fuzzy, the lens on the camera module can be adjusted to focus the image better.

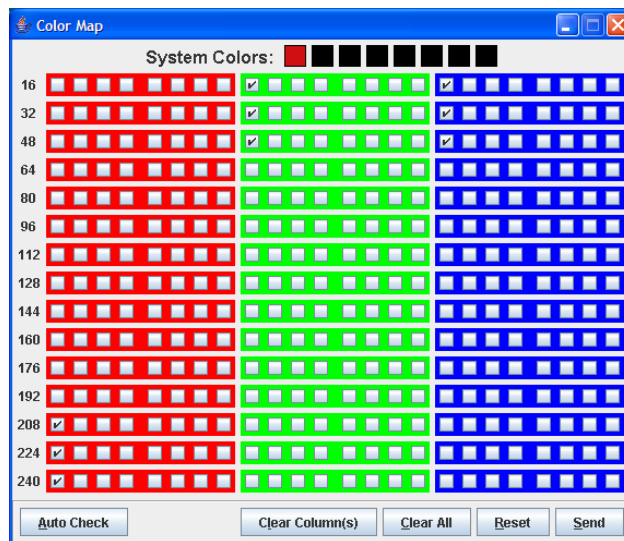
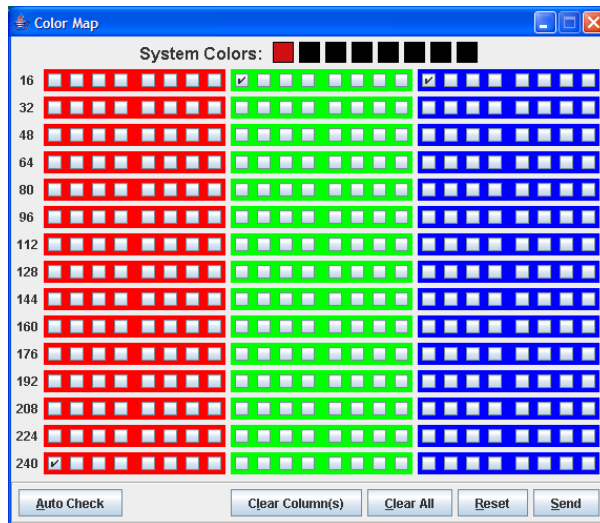


Once a useful image has been captured, the user can then select a color from the image to track by moving the mouse pointer over the region of interest in the completed image on the right. Notice that the red/green/blue values for the pixels underneath the mouse pointer are updated below the image to indicate what color combination is represented. When the mouse pointer is placed over the color of interest, right-click the mouse to add the color to the color map.



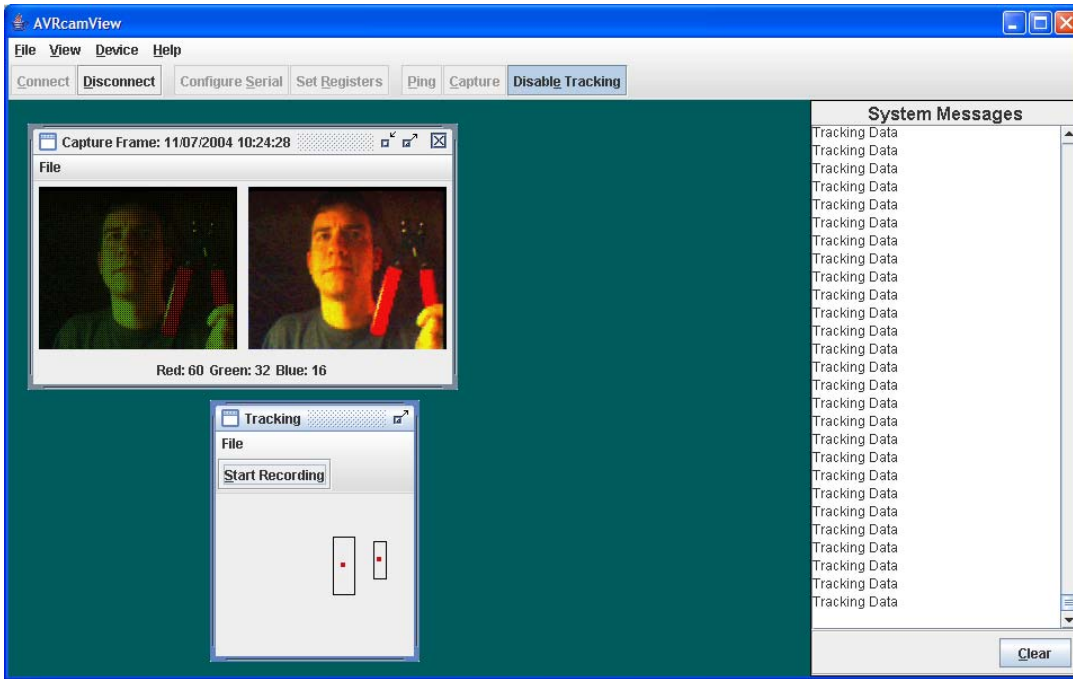
The color map supports up to 8 different tracked colors, represented by one color per column in the color map. Each tracked color is made up of a red component, a green component, and a blue component. Thus for the tracked color, a single check-box has been filled in each column. This sets the acceptable red/green/blue pixel values that will be considered “tracked” for the color. This map can be manually expanded to decrease the precision with which the color is tracked by simply checking a few of the surrounding boxes in each column. This will allow for some variance in the sampled pixels while still considering the pixels to be tracked.

# JROBOT

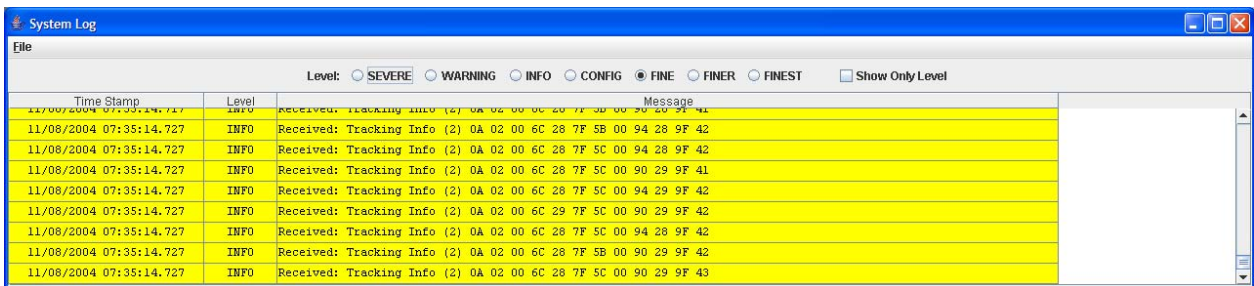


Next, send the color map down to the AVRcam by left-clicking on the “Send” button in the Color Map window. The Message Viewer display should show an ACK was received after the color map was sent, and the newly tracked color is displayed at the top of the Color Map window. Close the Color Map window to return to the main window.

Now that the color map has been set, left-click on the “Enable Tracking” button in the Command Bar. This will start the camera tracking any color that has been programmed into the Color Map. Note: The Command Bar has certain operations disabled while the system has tracking enabled. The Tracking window is displayed to show each tracked object’s bounding box (as well as each object’s color, shown with the filled in box in the center of each bounding box). As objects that match the colors in the color map move through the AVRcam’s field of view, the Tracking window updates in real-time to show that the object is being tracked.



The user can also view a real-time stream of the communication between the AVRcam and the PC. This can be viewed by selecting View-> Show Log from the main menu. Ensure that the log-level is set to Fine. With this set, all communication between the AVRcam and the PC will be displayed with a timestamp denoting when the message was sent. This information is quite useful when testing out the system to understand how certain commands are formatted. It should also be noted that this setup allows the user to view packets that are sent as binary values (such as tracking and dump packets), as well as those sent in ASCII (such as basic commands). The entire log file can be saved as either a text file or an XML file to be analyzed or parsed offline at a later time. Note: See Appendix B for a complete description of the different packet types and their payloads.



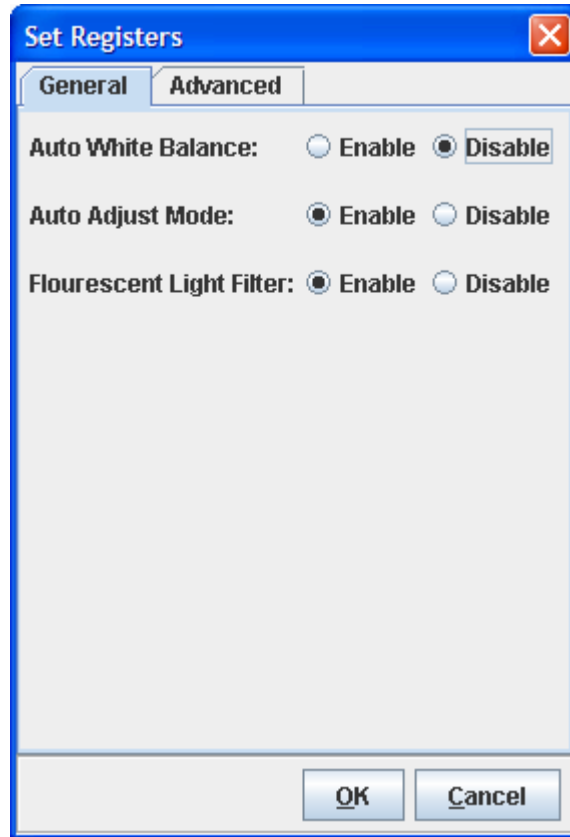
Tracking can be disabled by left-clicking on the Disable Tracking button in the Command Bar. This will allow the user full access to the Command Bar again.

## 6.6 How Lighting Affects the AVRcam

It is important to note here that lighting conditions can have a significant impact on tracking capabilities. For example, objects that are highly reflective tend to appear as bright white spots in the presence of significant light, regardless of color. The best way to understand how lighting affects the AVRcam is to capture different snapshots in different lighting conditions to see how

## JROBOT

the images are seen by the camera. It is also possible for the user to set parameters such as auto-adjust mode, auto-white balance, and fluorescent light filtering by left-clicking on the Set Registers button in the Command Bar.



It should also be noted that different types of light will affect the AVRcam in different ways. Specifically, infrared light from incandescent lights (or outdoor lighting) can flood into the camera and cause the images to appear very red. One way to deal with this situation is to enable the auto-white balance. However, this can also cause other colors to appear slightly different than their normal color. The best situation is to experiment with the system in different environments to determine what seems to work and what doesn't regarding lighting conditions.

## 7 Advanced Development with the AVRcam

The AVRcam project is an open-source development effort. Both the embedded software running on the AVRcam as well as the AVRcamVIEW PC application are both available in source form, licensed under the GNU General Public License. This allows any developer to have access to the source to make modifications, or add new functionality to the system as needed according to the terms of the GPL.

The AVRcam provides the standard 10-pin AVR ISP header, which is compatible with the STK200 and STK300 programming cables common among AVR enthusiasts. A suitable programming cable can be purchased at <http://www.jrobot.net/> that connects to the parallel port of any PC. This in-system programming also requires software that is capable of programming the mega8, such as the open-source AVRDUDE programming software ( <http://savannah.nongnu.org/projects/avrdude> ), or the free PonyProg software ( <http://www.lancos.com/prog.html> ). This provides a standard way to update the flash memory on the mega8, as well as modify the fuse settings on the mega8 if needed.

The AVRcam software was developed using AVR-GCC, a port of the standard GCC C compiler to the AVR family of microcontrollers. Version 3.2 of the GCC compiler was used for development, though later version should also work. The compiler is bundled into a package known as WinAVR, which is a simple installer that sets up the entire GCC development environment under Windows. This software can be downloaded at <http://winavr.sourceforge.net/>.

The AVRcamVIEW PC application was written entirely in Java, and utilizes the RXTX serial package for Java (available at <http://www.rxtx.org> ). This software can be expanded to add new buttons and associated commands to the GUI, as well as other image processing features if needed. The Java 1.5 development environment was used for this effort.

Finally, it is important to check <http://www.jrobot.net> often for new updates regarding the system, as well as additional discussion in the forums section at <http://www.jrobot.net/Forums>.

### 7.1 Programming the AVRcam

Each time the AVRcam powers up, the yellow debug LED will blink four times and stay powered on. During this blinking period, the user has the opportunity to re-program the AVR through the 10-pin STK200/300 programming header. The programming session **must** be initiated during this blinking period. Otherwise, it is possible for the programming to fail since the lines on the mega8 used for programming (MISO/MOSI/SCK) or only guaranteed to be in the required tri-state level while the yellow LED is blinking. If programming does fail, simply power-cycle the board. If the yellow LED does not blink after a few seconds, it is safe to attempt to re-program the mega8 again. If the LED does blink, follow the programming guidelines stated above.



## 8 Appendix A

### AVRcam Bill of Materials

<b>Part Description</b>	<b>Quantity</b>
RS232 Level Shifter	1
AVR mega8 microcontroller	1
AVR tiny12 microcontroller	1
C3088 Camera Module	1
LM2937 5V regulator	1
Molex 2-pin connector (male)	1
Molex 3-pin connector (male)	2
10-pin AVR ISP header	1
SPST power switch	1
1K Ohm resistor	3
10K Ohm resistor	6
1 uF capacitor	4
0.1 uF capacitor	4
10 uF capacitor	1
Green power LED	1
Yellow signal LED	1
2N3904 NPN transistor	1
8-pin IC socket	1
14-pin IC socket	2
16-pin IC socket	1
Dual-row 32 hole female header	1
AVRcam double-sided circuit board	1
DB-9 Female serial header	1
9-volt connector	1
Molex 2-pin connector (female)	1
Molex 3-pin connector (female)	1
Crimp terminals for female conns	5
Single-pin male headers	4

## 9 Appendix B

### 9.1 Complete AVRcam Command Protocol

The following tables list the various commands that can be used to control the camera and its capabilities.

Mnemonic	Ping
OpCode	PG
Payload	<none>
Response	ACK – if command successfully received NCK – if command not successfully received
Supplemental Response Packet	<none>

Example: The user wants to ping the AVRcam to ensure it is responding.

User: PG\r

AVRcam: ACK\r

Mnemonic	Change Camera Registers
OpCode	CR
Payload	RegAddr1 – indicating the address of the first register to modify RegVal1 – indicating the new value of the register specified in RegAddr1 <space char> RegAddr2 – indicating the address of the second register to modify RegVal2 – indicating the new value of the register specified in RegAddr2 <space char> ... RegAddr8 – indicating the address of the eighth register to modify RegVal8 – indicating the new value of the register specified in RegAddr8 (up to 8 registers may be modified with this format)
Response	ACK – if command successfully received NCK – if command not successfully received
Supplemental Response Packet	<none> (The camera will begin using the new camera settings)

Example: The user wants to update registers 11d (with a value of 14d) and 13d (with a value of 26d) in the OV6620 camera module.

User: CR 11 14 13 26\r

AVRcam: ACK\r

NOTE: This command can cause the AVRcam to not function properly if certain registers within the OV6620 are modified. Use caution when modifying the registers!

# JROBOT

Mnemonic	Dump a Frame
OpCode	DF
Payload	<none>
Response	ACK – if command successfully received NCK – if command not successfully received
Supplemental Response Packet	Raw color frame-dump packets:
	Byte 0: 0x0B – indicating the start of raw color data line
	Byte 1: line number in <b>hex</b> , indicating which line number is being sent
	Byte 2-178: 176 <b>hex</b> pixel stacks, where a pixel stack is structured as follows: Each pixel stack consists of a <i>one-byte</i> value, where the high nibble represents one color value (either red or green), and the low nibble represents one color value (either green or blue). The pixels on the camera are arranged in Bayer format.  Even pixel high nibbles: green value    Even pixel low nibbles: red value Odd pixel high nibbles: green value    Odd pixel low nibbles: blue value
	Byte 179: 0x0F – indicating an end of line
	Note: The AVRcam will send a total of 77 frame-dump packets to the host system (where each packet contains information about two consecutive rows of image data) each time the DF command is received.

Example: The user wants a single frame of image data sent to them (actual RGB values).

User: DF\r

AVRcam: ACK\r

AVRcam: 0x0B 0x01 0x83 0xD3 0x4A ...<173 more pixels> 0x0F

```

|      |      | |      | |      | |--blue value          |--end of line
|      |      | |      | |      | |--green value
|      |      | |      | |--red value
|      |      | |      | |--green value      (Note: No spaces exist between data)
|      |      | |--blue value
|      |      | |--green value
|      |--Line Number
|--Indicates start of frame-dump packet

```

AVRcam: 0x0B 0x02 0x83 0xD4 0x3A ...<173 more pixels>0x0F

...

AVRcam: 0x0B 0x48 0x83 0xD4 0x3A ...<173 more pixels>0x0F

(Note: The hex values 0x0B through 0x0F are reserved and will not be used in the data stream other than to indicate the starting and ending marks of a line of pixel data. In addition, the frame-dump packets can be sent out of order, which is why the line number is included in the packet. The user should be able to reassemble the lines in their proper order for displaying the complete image.)

Mnemonic	Get AVRcam Software Version Number
OpCode	GV
Payload	<none>
Response	ACK – if command successfully received NCK – if command not successfully received
Supplemental Response Packet	<The camera will send a string indicating the software version number>

Example: The user wants to determine what version of software is running on their AVRcam.

User: GV\r

AVRcam: ACK\r

AVRcam: AVRcam v1.0\r

# JROBOT

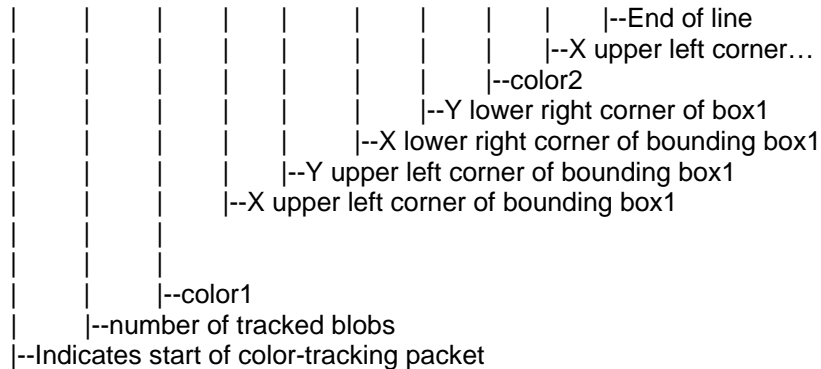
Mnemonic	Enable color-blob tracking
OpCode	ET
Payload	<none>
Response	ACK – if command successfully received NCK – if command not successfully received
Supplemental Response Packets	<p>&lt;The camera begins to generate color-tracking packets in <b>hex</b> to indicate what color-blobs are found in each frame. There are NO spaces between bytes in a color-tracking packet. The current color map will be used to map the sampled pixel values into actual colors.&gt;</p> <p>Byte 0: 0x0A – Indicating the start of a tracking packet            Byte 1: Number of tracked objects (0x00 – 0x08 are valid)            Byte 2: Color of object tracked in bounding box 1            Byte 3: X upper left corner of bounding box 1            Byte 4: Y upper left corner of bounding box 1            Byte 5: X lower right corner of bounding box 1            Byte 6: Y lower right corner of bounding box 1            Byte 7: Color object tracked in bounding box 2            ....            Byte x: 0xFF (indicates the end of line, and will be sent after all tracking info for the current frame has been sent)</p>

Example: The user wants to begin tracking the colors set in the color map.

User: ET\r

AVRcam: ACK\r

AVRcam: 0x0A 0x05 0x04 0x12 0x09 0x36 0x38 0x01 0x25...0xFF



AVRcam: 0x0A 0x05 0x06 0x32 0x39 0x76 0x98...0x0F

(Note: The AVRcam is capable of tracking up to eight color blobs (values 1 through 8 as defined by the current color map) per frame. This number is variable, and may also be zero. All currently tracked blobs will be returned inside of each color-tracking packet.

Mnemonic	Disable color-blob tracking
OpCode	DT
Payload	<none>
Response	ACK – if command successfully received NCK – if command not successfully received
Supplemental Response Packet	<none> (Color blob tracking will end).

Example: The user wants to disable color-blob tracking.

User: DT\r

AVRcam: ACK\r



