There are many types of sensors that people can add to their hobby robot projects, but one type of sensor that you almost never see is an image sensor. For humans, sight is our primary method of getting information about our world. Wouldn't it be great to allow our robots to perceive the world in the same way that we do? This month's column will show you how you can add an image sensor to your robot so that it can learn much more about the environment that it is in.

The image sensor that will be used is not what immediately comes to mind when you think of an image sensor. This column will be using the Taos TSL3301. It is a linear array of 102 pixels. This chip comes in a clear eight-pin DIP package, which makes it handy for those of us who like to prototype on breadboards or on perfboard. This chip can divide its array into three 34-pixel sections. Each section can have a separate gain and offset values though this column will set all three sections to the same settings. This chip can run off of a single five-volt supply and has a completely digital interface. This is quite handy when you are using a low-end microcontroller that doesn't have an analog-to-digital converter.

The TSL3301 has one of the smallest number of pixels in the series of chips that Taos produces, but this small number fits well with small embedded processors that have limited amounts of RAM to use. Despite the RAM limitation, generally your robot is going to have all the time that it needs to process the information that it receives so the actual pattern recognition tasks shouldn't be much of a limiting factor.

Let's look at the pinout for the TSL3301. As you can see, it only has three pins that you will be using to communicate with this chip. This makes it really easy to interface with your microcontroller. The interface is a strange mishmash of the RS232 protocol and SPI.

The data lines communicate using one start bit, eight bits of data, and one stop bit as RS232 does, but this data can come and go at almost any baud rate because you are also providing a clock signal. You will need a pretty fast processor to hit its speed limit, which is a clock of 10 MHz.

The other quirk about this chip is that it has no internal clock to drive its functionality so the clock that you provide for the serial communications is also what drives its internal functionality. Because of this, sometimes you will need to send a few extra clock pulses to the chip so that it can finish doing things internally.

Before going further ahead into how the chip operates, let's back up and look at how to get an image projected onto the pixel array in the first place. Working with optics can be an involved process if you are trying to achieve a high quality image. Fortunately for us, having a low-quality image is more than sufficient for our purposes since we only have 102 pixels to capture the image anyway.

A single, double convex lens was used to project the image onto the chip. The lens was part number NT32-019, purchased through Edmund Industrial Optics. This lens is 9 mm in diameter and has a 9 mm focal length. Because of the short focal length, this

Figure 1. The pinout for the TSL3301.

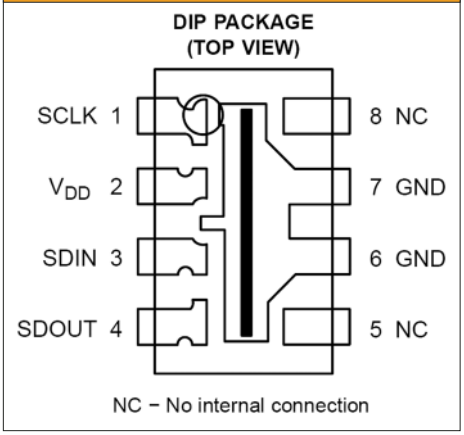


Figure 2. A side view of the image sensor assembly.

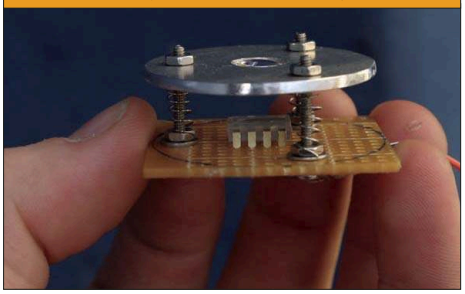
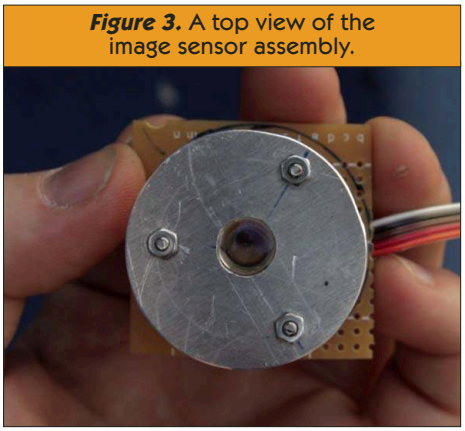


Figure 3. A top view of the image sensor assembly.



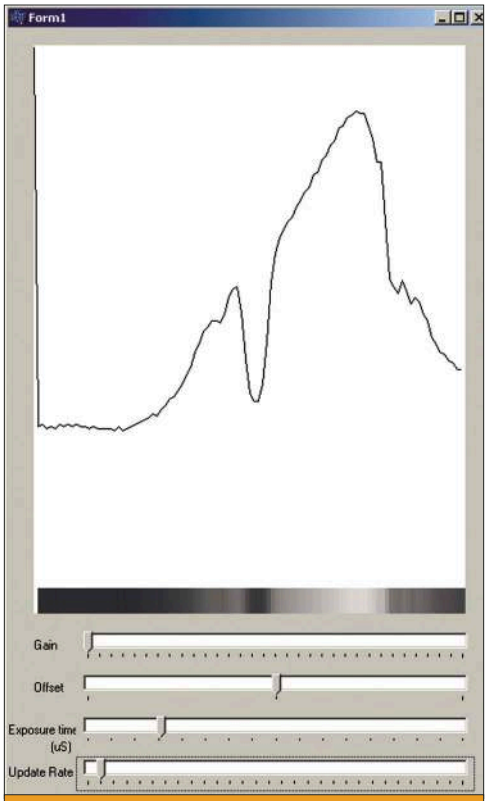


Figure 4. This program is on *SERVO*'s website and allows you to view the images from the TSL3301.

lens allows for a wide field of view. This can give you a good overview of the room that your robot is in, but won't allow you to see detail.

This lens was mounted, as shown in Figures 2 and 3. A piece of aluminum was cut into a circle and a hole was drilled in its center that was just slightly bigger than the lens diameter. Next, three holes were drilled into the perimeter of the aluminum piece that allow 1-72 bolts to pass through. Matching holes were drilled into a prototyping circuit board that the image sensor was mounted to. Then 1-72 bolts were put through the holes in the prototyping board and nuts were put onto the other side to keep them mounted firmly in place.

Small springs were made to go around the bolts. These springs keep the aluminum piece and lens away from the sensor. You can make a spring by wrapping piano wire around a drill bit or any other round piece of metal. Next, the lens is mounted to the aluminum piece by first laying the aluminum piece flat onto a table and placing the lens inside of its hole. Now take some super glue and put three drops of it around the lens on the aluminum. Make sure that no super glue touches the lens at this point.

Take a toothpick and carefully drag the drops over to the edge of the lens. Let this sit for a few minutes and your lens will be firmly bonded to the aluminum. Make sure that the super glue is fully dried or else you risk getting some onto the lens with your fingers when you pick it up. This type of mount is a little more involved to make than others, but allows for higher precision focusing due to the high number of threads per inch in the bolts.

Slide the aluminum disk over the three bolts and thread some nuts onto the bolts. These nuts won't be tightened but instead will allow you to adjust the distance of the lens from the image sensor. When you find the place that is in focus, put a little locktite onto the nuts to keep the lens in place.

Okay, you can now project an image onto your sensor, so let's go back to how to talk to this chip. This chip is really easy to communicate with. It does, however, require that you write your own bit-banged receive and send routines because of its quirky interface. The TSL3301 chip has three communication lines. These are called: SCK, which is the clock line; SDIN, which is the line that the chip receives data on; and SDOUT, which is its transmit line. SDOUT and SCK will be used to transmit.

Do the following to send a byte to the LTC3301:

• Drive the SDOUT line low.

• Pulse the SCK line by driving it high and then low again. If you have a fast processor, be mindful of the maximum clock rate of 10 MHz.

• Create a loop that repeats eight times and does the following:
  – Look at the least significant bit in the byte that will be sent and set the SDOUT line to match.
  – Pulse the SCK line.
  – Shift the byte that is being output one bit to the right.

• Drive SDOUT high.

• Pulse SCK.

There is some source code that runs on a PIC16F873 processor that's available on the *SERVO* website (**www.servomagazine.com**) that you can reference if you are having trouble with something that you see in this month's column.

To receive a byte from the TDL3301, you need to do the following:

• Pulse the SCK line once to skip over the start bit.

• Clear a register that will hold the received byte. We'll call this DATA.

• Now create a loop that does the following eight times:
  – Shift DATA one bit to the right.
  – If SDIN is high, then it will set the highest bit of DATA.
  – Pulse SCK.

• Finally, pulse SCK once to skip over the stop bit.

The TSL3301 chip needs to be initialized when you first power it up. Here is the routine that you follow to make it happy so that you can start sending it commands:

• Drive the SCK line low.
• Drive the SDIN line low.
• Pulse the SCK line 30 times.
• Drive the SDIN line high.
• Pulse the SCK line 10 times.
• Send 0x1B to the chip.
• Pulse the clock five times.
• Send 0x5F to the chip.
• Send 0x00 to the chip.

Before you start reading the data from the chip, you may want to change the gain and offset values. Gain adjusts the scaling of the values that are read. Increasing gain can add noise to the image but may be necessary if you are taking hundreds of images per second. The gain variable can be anything from 0 to 31. Offset adds or subtracts a fixed value from each pixel. It is an eight-bit sign magnitude variable so it can represent any value from –128 to 127.

To adjust your gains and offsets,

you will need to write to a few registers. There are three gain and three offset registers that correspond to the different 34-pixel sections of the array. To write to a register, first you will send its address and then the value that you want to write to it. The addresses for the offset registers are 0x40, 0x42, and 0x44. The addresses for the gain registers are 0x41, 0x43, and 0x45.

Now you are ready to capture your image. To capture an image, you will need to do the following:

- Send 0x80 to the chip to start capturing the image.
- Pulse SCK 22 times.
- Delay for the amount of time necessary to capture the image. This would be equivalent to how long the shutter would be open in a real camera. Shutter times of one microsecond to 255 microseconds make for a pretty good range that can see in bright sunlight and in candlelight.
- Send 0x10 to the chip to stop capturing the image.
- Pulse SCK five times.
- Send 0x02 to start reading the pixels from the chip.
- Pulse SCK repeatedly until you see a start bit (low SDOUT).
- For all 102 pixels, receive a byte.

Wow! There were a lot of things that you needed to set up, but once you have all of the routines that were described here written, you can start to have some fun with this chip. One thing that you might like to do with this sensor is to see in color. This sensor simply responds to the amount of light that strikes it, so if you want a color image, then you will need to use filters to read red, green, and blue images. You can then combine these to make a full-color image.

Buying professional optical filters can be expensive. A cheap way to get around that problem is to go to a local store that sells or rents motion picture, stage lighting, or maybe photography equipment. You can often find sample booklets of filters that are used to color lights. The samples are far too small to put over a light but are more than big enough to put over your robot's tiny lens. The nice thing is that these filter booklets have graphs of the colors that they allow to pass through so filter selection is easy. Making a filter wheel that rotates in front of your sensor would allow you to capture color images.

Something that you should be aware of is that if your robot is in a room with fluorescent lights, then your images will vary a lot in brightness due to the flickering of the fluorescent bulbs. You might want to put a dark filter over the sensor and increase your exposure time to a full cycle of the bulb's flicker rate; 8.3 milliseconds should work for fluorescents with older ballasts. Newer electronic ballasts might not create this flicker problem.

If you want an image that you can display on a computer, you could mount the sensor and lens onto a hobby servo and slowly sweep it around the room. The software that is provided on *SERVO*'s website allows you to see a graph of the brightness of each pixel and a grayscale version of what it is seeing, as well. It would be fairly simple to modify it into a program that progressively captured images and displayed them on successive columns or rows.

RESOURCES

Mouser Electronics
www.mouser.com
Sells the TSL3301 chip.

LEE Filters USA
www.leefiltersusa.com
Sells filters for the motion picture industry.

Edmund Optics
www.edmundoptics.com/US/
Sells lenses.

Custom Computer Services, Inc.
www.ccsinfo.com
Sells the C compiler used for the PIC code on *SERVO*'s website.

Borland
www.borland.com/us
Sells the C++ compiler used for the PC code on *SERVO*'s website.

Visual input is not something that you commonly see in hobby robotics though it isn't terribly difficult or expensive to integrate into your projects. There are endless possibilities of things that you can do with robotic vision. You could track moving objects. You could do optical range finding. You could locate objects of a certain color or determine the motion of something without any physical contact. What could you do with a sensor like this? **SV**