

## BalloonSat Sensor Array

The PICAXE-08M2 in the BalloonSat flight computer is a digital device. Being digital, it functions best with a series of on and off voltages and does not interact very well with a voltage that varies between being fully on and fully off (analog). However, there is a circuit inside the PICAXE-08M2 that interfaces to the world of analog voltages. The circuit is called an analog to digital converter, or ADC and is activated by the **READADC** command.

The PICAXE's **READADC** command converts an analog voltage into a digital value with eight bits of resolution. A bit is either a 0 or a 1, so eight bits ranges from the low value of 0000 0000 to the high value of 1111 1111. In binary, 0000 0000 is equivalent to a decimal value of 0 and a binary 1111 1111 is equivalent to a decimal value of 255. Therefore, the ADC converts a voltage between 0 and +5 volts to a decimal value between 0 and 255. The conversion is linear, so a decimal value of 0 is zero volts and a decimal value of 255 is five volts. This also means that a change of just one decimal value is equal to a change of 5 volts divided by 256, or 0.0195 volts (or 19.5 mV). Why did we divide five volts by 256 and not 255? Because there are 256 values between 0 and 255.

Okay, the **READADC** command converts an analog voltage into a decimal value, but how do we connect the sensor to the PICAXE and how do we get the decimal number of the sensor's voltage out? The entire command looks like this.

**READADC** channel,variable

For example, **READADC C.1,B0**

This example converts a voltage on the PICAXE's input 1 (on bank C) and stores the result in a variable named B0. The variable B0 is the first one byte variable in RAM memory. We know its one byte in size because its name begins with the letter B. A byte is eight bits wide, which is the same number of bits that the **READADC** command returns. Once the value is stored inside a variable, B0 in this case, it can be stored in RAM memory and displayed on a PC.

The simplest way to display the value stored in B0 use the command, **DEBUG**. The **DEBUG** command opens a window on the PICAXE Editor that displays every variable (B0 to B13) in the PICAXE's memory. You'll need to look at the listing for B0 to see what value is stored in it. If you create a program loop that reads an analog value into memory and then displays it in the **DEBUG** Window, you can watch the sensor connected to your flight computer react to changing environmental conditions.

The problem with the **DEBUG** command is that it only displays data stored in RAM memory. Only 14 pieces of data can be stored in RAM memory and what is stored there, is forgotten as soon as the battery is disconnected. Therefore, you don't want to use RAM memory to store data for display data after a near space mission. To save a value

currently stored in variable B0 into permanent memory location for later retrieval, move the data to EEPROM memory. There are 256 bytes of EEPROM memory for the long term storage of flight data and the **WRITE** command moves data from RAM into EEPROM. The **WRITE** command has the following syntax.

**WRITE** location,data  
**WRITE** location,variable

For example, **WRITE 120,8**  
or, **WRITE 95,B0**

The first example saves the decimal value 8 into memory location 120. That's not very useful because we are storing a specific number (8) into a specific EEPROM location (120). The second example is much better; it stores the value currently in variable B0 and stores it in memory location 95. Since there are 256 memory locations for data, you'll need to write this command 256 times to record all the data your BalloonSat flight computer can collect. How could you possibly add the **WRITE** command 256 times to the flight computer's program without running out of memory to collect data and take pictures? The best way to store the value in a variable into a memory location pointed to by the number stored inside a second variable.

For example, **WRITE B0,B1**

In this much better example, the value currently in variable B1 is stored in a memory location pointed to by the value in variable B0. After storing the value, increase the number stored inside B0 by 1 so it points to the next available memory location. You increment the value stored in variable B0 using the following command.

**B0 = B0 + 1**

After incrementing the value in B0, use the same **WRITE** command to store the newest value in variable B1 into the next available memory location. Executing these commands over and over stores a list of values into memory, without overwriting the earlier ones. This lets the flight computer record sensor values throughout the mission. Then after recovery of the BalloonSat, you can see how the sensor reacted to environmental conditions as the BalloonSat ascended.

After recovery, use the **READ** command to retrieve the values stored in memory. The **READ** command uses the following syntax.

**READ** location,variable

Like the **WRITE** command, the location can be either a specific number or a variable. By incrementing the value in the variable, the number stored in successive memory locations can be read out.

Let's put it all together and see what we get.

**Mission:**

**READADC 2,B0**

**WRITE B1,B0**

**B1 = B1 + 1**

**GOTO Mission**

**Readout:**

**READ B1,B0**

**DEBUG**

**B1 = B1 + 1**

**GOTO Readout**

Now of course it won't be this easy. For one thing, how many sensor measurements does your flight computer need to collect, how often does it need to collect them, and when does the flight computer need to read out stored values? In addition, the values stored in memory are safe, even if power is lost to the PICAXE. However, reprogram the PICAXE to retrieve values stored in memory and the values will be overwritten before you can get them. Therefore, the program in the flight computer must be capable of both storing values during a mission and retrieving them after recovery.

So please read about the following commands and see what solutions you can develop.

**PAUSE**

**READADC**

**SEROUT**

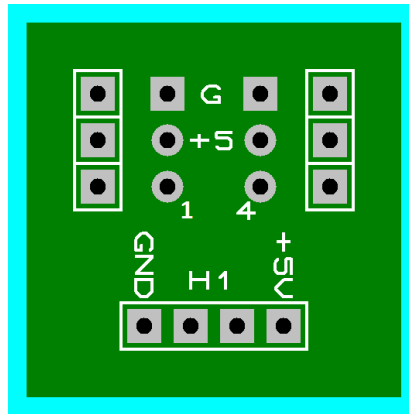
**IF**

**THEN**

**PIN**

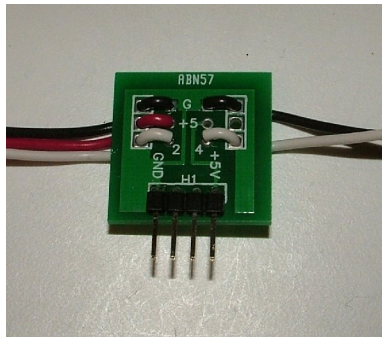
**Building Sensor Arrays**

The BalloonSat flight computer digitizes two analog voltages on I/O pins 1 and 4. Virtually all sensors require power, so the flight computer's input port has connections for +5 volts and ground along with connections to I/O pins 1 and 4. When a sensor array of two sensors is plugged into the input port, the sensor array begins producing output as soon as the flight computer has power. The four-pin header PCB shown below plugs into the receptacle on the flight computer and connects the two sensors of your BalloonSat's sensor array.



**Figure 1. The four pin header used with the flight computer.**

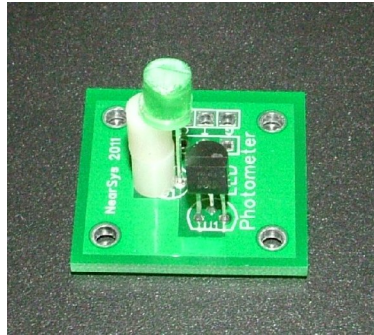
You soldered a right-angle header of four pins to the bottom of a header when you built your flight computer. Now let's look at the soldering pads on the left and right side of the header. Notice there are two pairs of three pads on the left and right sides. A sensor soldered to the left side gets a connection to +5 volts, ground, and I/O pin 1 while a sensor soldered to the right side gets a connection to +5 volts, ground, and I/O pin 4. The top pad has a letter G, indicating these pads are for ground, or the zero volt wire of the sensor. The middle pad has +5 printed next to it, indicating these pads provide five volts to power to the sensors. The bottom pads have a 1 or a 4. Therefore, the power and grounds are shared between sensors, but not the I/O pins. A four pin header, when complete, looks like this.



**Figure 2. This four pin header has two sensors soldered to it (the sensors are outside the image). Note that the right sensor only has two wires, ground and signal, connecting it to the four pin header. The left sensor is using all three wires, the black wire for ground, the red wire for +5 volts, and the white wire to connection to I/O pin #1 (although the header in this picture says its I/O #2).**

Normally a cable of three wires connects sensors to the four pin header. The cables are necessary because sensors are usually located away from the flight computer (the flight computer is inside the BalloonSat while sensors are located outside the airframe). Also, notice that the header PCB has strain relief holes on the outside edges. Now let's look at the sensors.

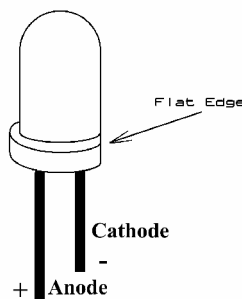
## LED Photometer



**Figure 3. The LED photometer with a green LED the LM335 temperature sensor.**

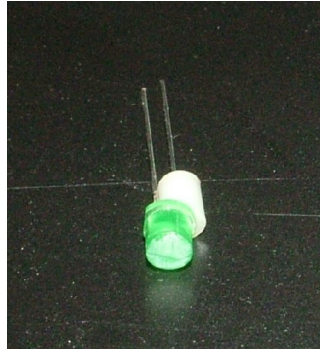
A photometer produces a voltage proportional to the intensity of the light shining on it (double the intensity of the light and the voltage also doubles). By using an LED in the photometer, it only detects the color of the LED. The photometer circuit converts the weak current signal from the LED into a voltage and amplifies it. The circuit that does this is called a transconductance amplifier and is based the LTC272 operational amplifier (op-amp). An LED is sensitive to its temperature and not just the amount of light shining on it. Therefore, it is important that the temperature of the LED also be known. That's why next to the LED is a black-colored LM335 temperature sensor. **Note:** the LM335 looks a lot like the LP2950 voltage regulator used in the flight computer. Look carefully on the LM335 and you'll see that LM335 printed on it face. When you use the photometer as a sensor, the flight computer is reading two pieces of data, the voltage from the LED (light intensity) and the voltage from the LM335 (temperature).

LEDs have a positive and negative terminal (called the anode and cathode). The LED Photometer PCB has an A next to the solder pad for the anode lead of the LED. The opposite of the anode, the cathode, is marked with a flat edge on the LED case.



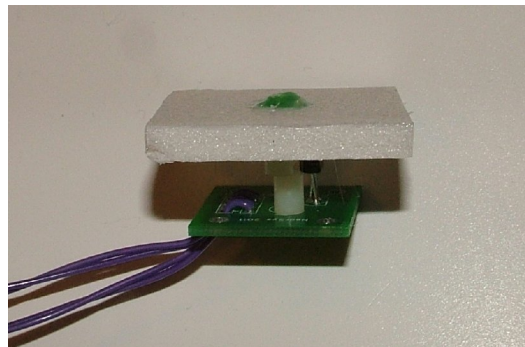
**Figure 4. The cathode of the LED is marked by a flat edge and usually, a shorter lead.**

To work properly, the LED in the photometer must stand higher than the LM335 temperature sensor. Use two plastic stand-offs on one of the LED leads to make the LED stand above the PCB as shown below. The top of the LED **must** be cut off and the top sanded smooth so that the LED is not as sensitive to its pointing direction. Do this before soldering the LED to the PCB.



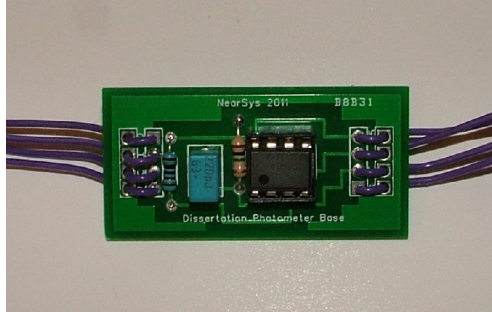
**Figure 5. A green LED with one of its leads (wire) sticking through a plastic spacer (a stand-off). Stack two stand-offs on one lead to get the LED to stand higher than the LM335 temperature sensor. The other lead of the LED can be left bare, you don't need to cover it up with a stand-off.**

The LM335 temperature sensor in the Photometer is black and will absorb sunlight. This makes the LM335 hotter than the LED. To keep the temperature of the LM335 closer to the temperature of the LED, add a sun shield to the LED photometer like the one shown below. The sun shield is a small sheet of white Styrofoam.

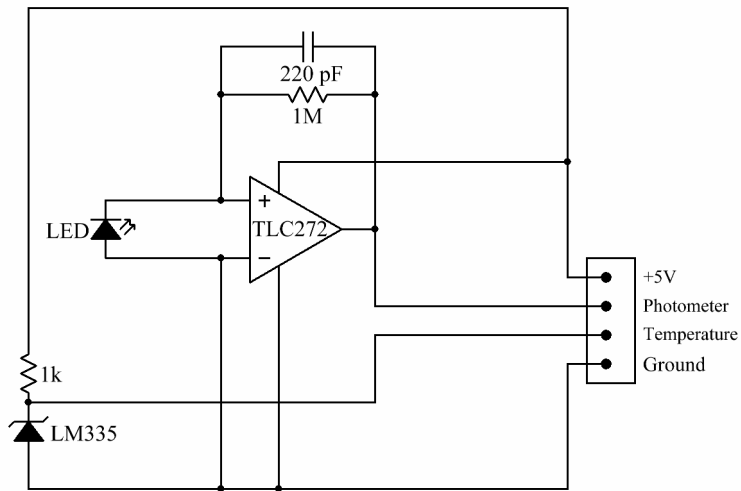


**Figure 6. The LED Photometer with its sun shield. The flattened top of the LED is looking out a small hole in the sun shield. The LM335 temperature sensor is shielded underneath the Styrofoam.**

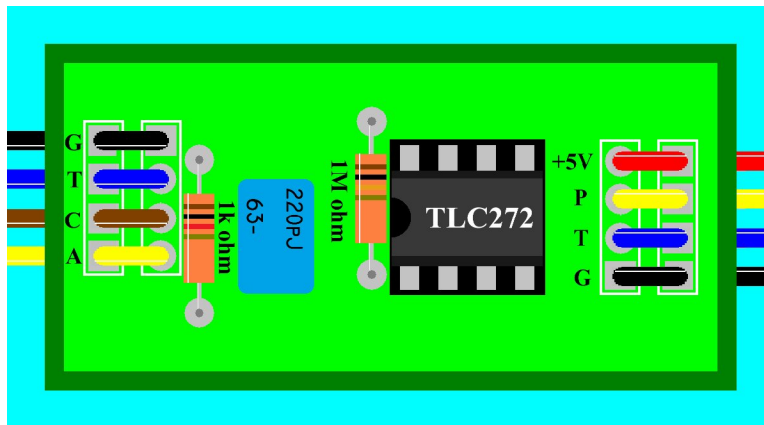
The small current produced by sunlight shining on the LED must be converted to a larger voltage before the flight computer can use it. The TLC272 op amp is part of a circuit called a transconductance amplifier that does this conversion.



**Figure 7. The transconductance amplifier is on a separate PCB and connects to the LED photometer by four wires.**



**Figure 8. Schematic for the LED Photometer.**



**Figure 9. Place of parts for the photometer's transconductance amplifier.**

The capacitor and resistors are not polarized, but the op-amp is. Watch that you point the notch in the IC in the correct direction before you plug it into its socket. Use the one

mega-ohm resistor (brown, black, green, gold) for a red, yellow, or green LED. An infrared LED (in a water clear case) can use a lower value, like the 220 kilo-ohm resistor (red, red, yellow, gold). The 1,000 ohm resistor is for the LM335 temperature sensor.

Use the strain relief holes when soldering the four wires to the PCB. The wires then run to the four-pin header PCB. At the four-pin header, solder the +5V and GND wires to opposite sides of the PCB. Then watch carefully where you solder the sensor wires to the I/O pads of the four-pin header. The diagram below shows there are two wires on each side of the four-pin header.

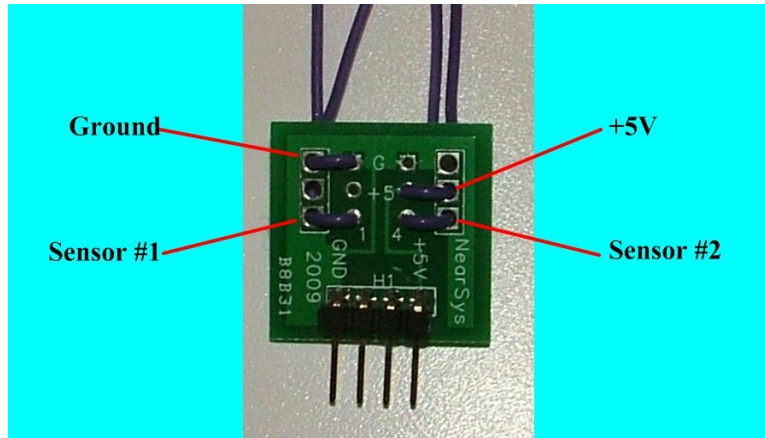


Figure 9. Solder two wires to each side of the four-pin header.

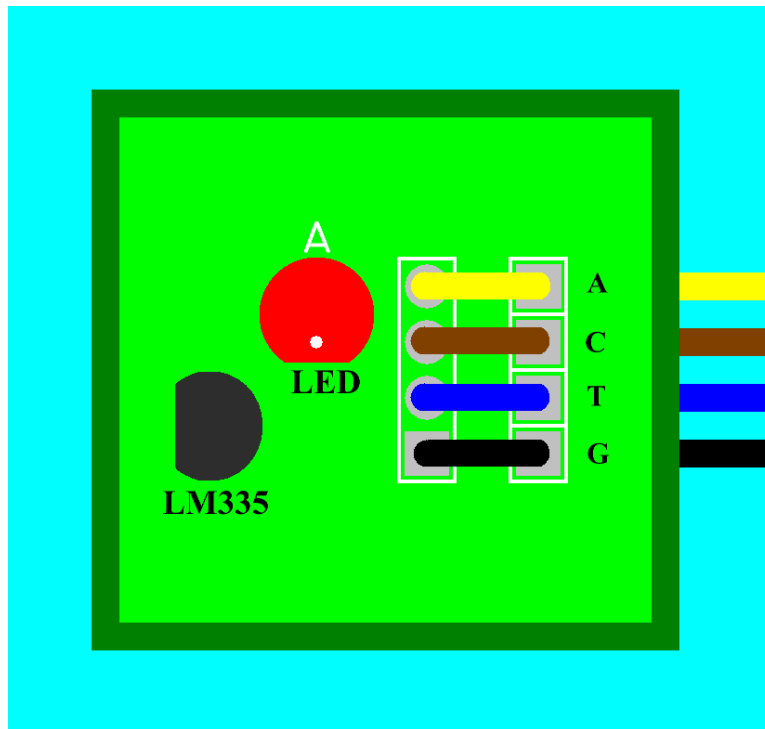


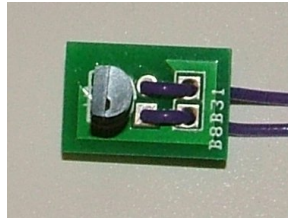
Figure 10. Place of parts for the photometer.

Don't forget, the domed-top of the LED must be cut off. You can use a hack saw or coarse sandpaper to flatten the top. After removing the top, sand the top of the LED until it's flat and smooth. The sandpaper will leave the top looking frosty. This is needed so the LED housing doesn't focus sunlight on the silicon inside the LED. Both the LED and LM335 are polarized and must be soldered in their correct orientation for the circuit to work. Make sure the anode (A), cathode (C), temperature (T), and ground (G) wires properly connect between the transconductance PCB and the photometer PCB.

Also, note that the LED Photometer PCB can be used with any current device. So for example, a small solar cell can replace the LED. This allows the output of the solar sell to be measured during a mission.

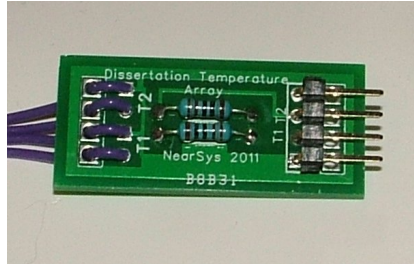
### Temperature Sensors

This is an LM335 electronic thermometer. Specifically, it is a temperature-controlled zener diode that produces a voltage between 0 and 5 volts for temperatures between 0 and 500 kelvins. The sensor looks like a little transistor. The white lettering (top silk) on the Temp Sensor PCB shows the proper orientation for the LM335.

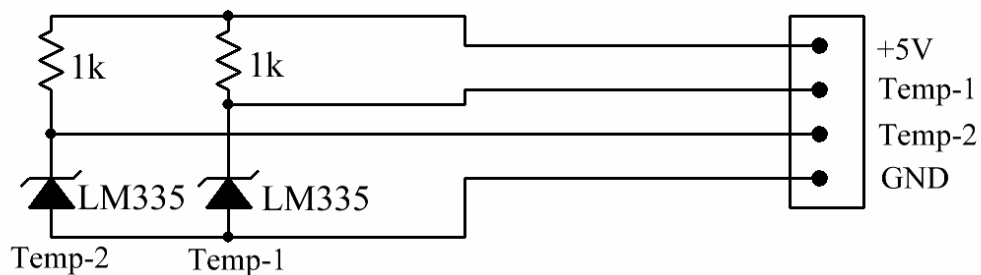


**Figure 11. The sensor portion of the temperature sensor. Look at its flat face to verify it really is a LM335 and not a LP2950. The name is stamped in small white letters.**

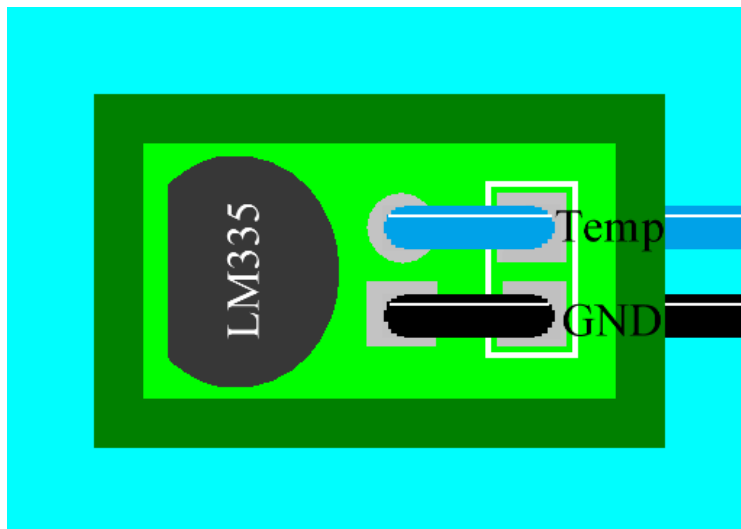
To make the LM335 produce a voltage, it needs a current limiting resistor of 1,000 ohms (brown, black, red, gold). The resistor also acts like a voltage divider when it is combined with the LM335. The PCB for the temperature sensors supports two LM335's and is called the temperature support board. Since it is a separate PCB, it can remain inside the BalloonSat airframe while two-wire cables attach it to each of the temperature sensor PCBs. The temperature support board can attach to a four-pin header PCB or you can just solder a four-pin header directly to the PCB as shown below.



**Figure 12. The temperature support board. It contains two 1,000 ohm resistors and four wires running off to two temperature sensors. A right-angle four-pin header is soldered on the right side so the PCB will plug directly into the I/O port of the flight computer.**



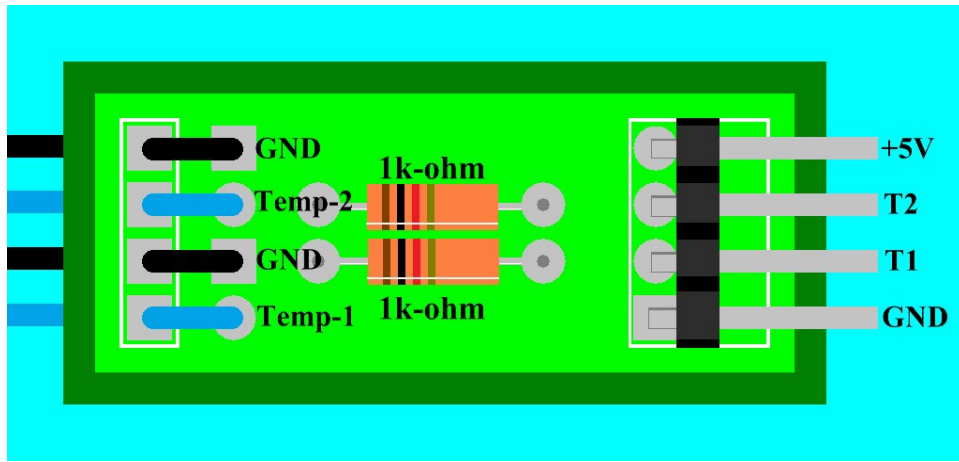
**Figure 13. The schematic for the temperature sensor.**



**Figure 14. The placement of the LM335 and wires on the temperature PCB.**

The LM335 is polarized, make sure you solder it to the PCB according to the white silk drawing on the PCB. Make note of which wire is ground and which is the output from the

temperature sensor. The wires connect to the temperature support board and the wires must be soldered to the same pads on this PCB.



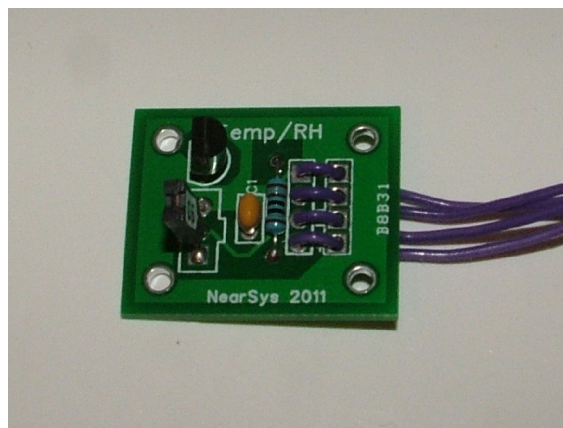
**Figure 15. Placement of parts for the temperature support board.**

The resistors are not polarized, just be sure to line up the wires from the two temperature PCBs with the solder pads on the board. Instead of soldering the output pads to wires and finally to the four-pin header PCB, just solder a four-pin, right-angle header directly to the PCB. Now the temperature support board will plug directly into the flight computer.

Consider shielding the temperature sensor from sunlight with the white plastic lid. That way solar heating doesn't affect the air temperature reading.

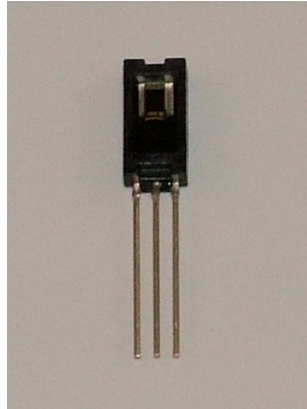
### **Temperature and Relative Humidity Sensor**

The HIH-4000 produces a voltage in proportion to the relative humidity. When combined with a LM335, this sensor array measures both relative humidity and temperature.



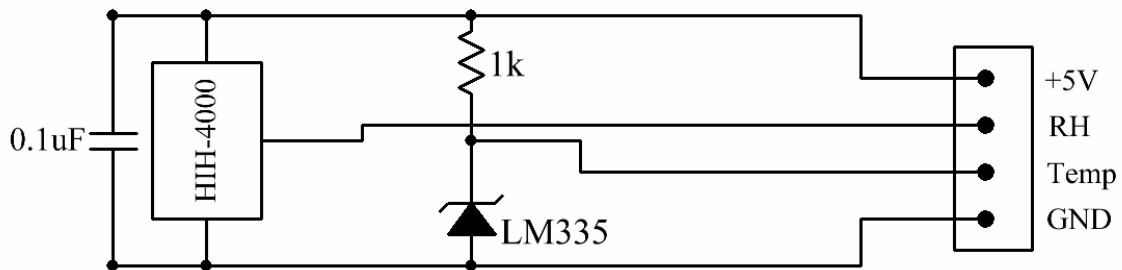
**Figure 16. The completed temperature and relative humidity sensor array.**

The relative humidity sensor produces an output that varies linearly with the humidity. The HIH 40000 sensor produces 0.8 volts at 0% relative humidity and increases by 0.031 volts for every 1% increase. The sensor has three leads, so it possible to solder it backwards. **Avoid this.** Notice the sensor has an open face on one side and is closed on the back. Inside the open face, the actual silicon sensor is visible as shown in figure 17.



**Figure 17. Visible on this side of the HIH 40000 is the tiny humidity sensor.**

When you solder the HIH-4000 to the PCB, point the open face of the HIH-4000 away from the tiny 0.1 uF capacitor or towards the nearest edge of the PCB.



**Figure 18. The schematic for the relative humidity and temperature array.**



## **Photometer**

Since the BalloonSat spins during its flight, it's best if you do not place the photometer on the side of the airframe. The LED is sensitive to temperature, so even if the light intensity does not change, its voltage will change as it gets colder. The change is linear, so if you can take a reading at the same light intensity but at different temperatures, you can determine the equation for adjusting the photometer's output based on temperature. Probably the best way to do this is to collect light and temperature data outside on a sunny day. Place the warm photometer outside during a cold day and leave the LED pointing in the same direction while the flight computer collects both temperature and light intensity data. Note that you are rapidly cooling the photometer while keeping it pointed in the same direction. As long as the data is collected over a few minutes and the LED can't change its orientation with respect to the sun, any change in the light intensity reading can only be due to the LED's temperature. Take two extreme temperature and light readings and find the slope and intercept of the equation that relates them. Then use this equation to adjust the photometer's readings after the BalloonSat mission to determine the change in light intensity.

The output of the photometer is linear with respect to light intensity (once you account for temperature effects). Therefore, if the photometer voltage has increased by 50%, then the light intensity has increased by 50%. In the spreadsheet, compare the photometer output to the first reading on the ground. However, first remember to adjust the photometer output for temperature.

The BalloonSat kit contains a sheet of Polaroid filter. If the Polaroid filter is placed over the photometer, then the photometer becomes a photopolarimeter. Knowing the relationship of the sun to the BalloonSat might be important information in interpreting the results from a photopolarimeter. How could you tell which direction the photometer is pointed during its mission in near space?

## **Temperature Array**

Since there are two temperature sensors in this array, you can place each sensor in different locations and make comparisons. Inside and outside the airframe is one example. The sensors will also let you compare the effectiveness of insulation or the effects of color on temperature.

The output of each temperature reading must be converted to a temperature scale before it's useful. The digital readings go from 0 to 255 while the temperature in Kelvins goes from 0 to 500. What spreadsheet equation will make this conversion? The relationship between digital reading and temperature is linear, so you only need to determine a slope and intercept. After converting the temperature into Kelvins, you'll probably want to convert the temperature into Celsius or Fahrenheit.

## **Relative Humidity and Temperature**

The lowest digital reading from the HIH-4000 is 41 and that occurs at 0% relative humidity. The highest reading is 199 and that occurs at 100% relative humidity. The

output is linear, so if you can find the slope and intercept of the relationship described above, you can create an equation for converting digital readings into relative humidity.

The temperature sensor was described above.

### Getting Data from the Sensor Arrays

After your BalloonSat has been returned from its mission, your flight software must download the data it collected without you having to reprogram it. **Remember**, if you reprogram the flight computer, you will erase all the data collected. After your flight computer has finished downloading its data into the terminal program, click **Edit**, then **Copy Input Buffer**. Now start up Notepad and click on **Edit** and then **Paste**. The results will look like this.

```
150,210,150,209,149,208,147,208,146,207,144,206,142,205.....
```

Since there are two sensors, break the single line of data into two columns. The result looks like this.

```
150,210
150,209
149,208
147,208
146,207
144,206
142,205
.....
```

Save the text file and then open a spreadsheet program. Open the text file and import it into the spreadsheet. Name the columns after the data that was collected. In the example above, the data came from the relative humidity and temperature sensor array. I'll name the first column Temperature and the second column Relative Humidity.

In the next columns, add the equations needed to convert the flight computer's digital values into actual readings you can use. Then are you ready to make charts?

### Importing Altitude into the Spreadsheet

Your BalloonSat probably recorded data once per minute. In some experiments, knowing just the time of the reading is good enough; however, in other cases, you will want to know the altitude of the BalloonSat instead of the time. On the NearSys website, you will find a spreadsheet generated from the APRS log used to track the balloon that carried your BalloonSat. The spreadsheet will have the time and altitude of the balloon. You can take each altitude at each minute of the flight and add it to a new column in your BalloonSat's spreadsheet.

Your BalloonSat will begin recording data a minute or two before liftoff. Therefore, you'll need to take that into account when adding altitudes to your spreadsheet. After this, you can begin making charts of your data.