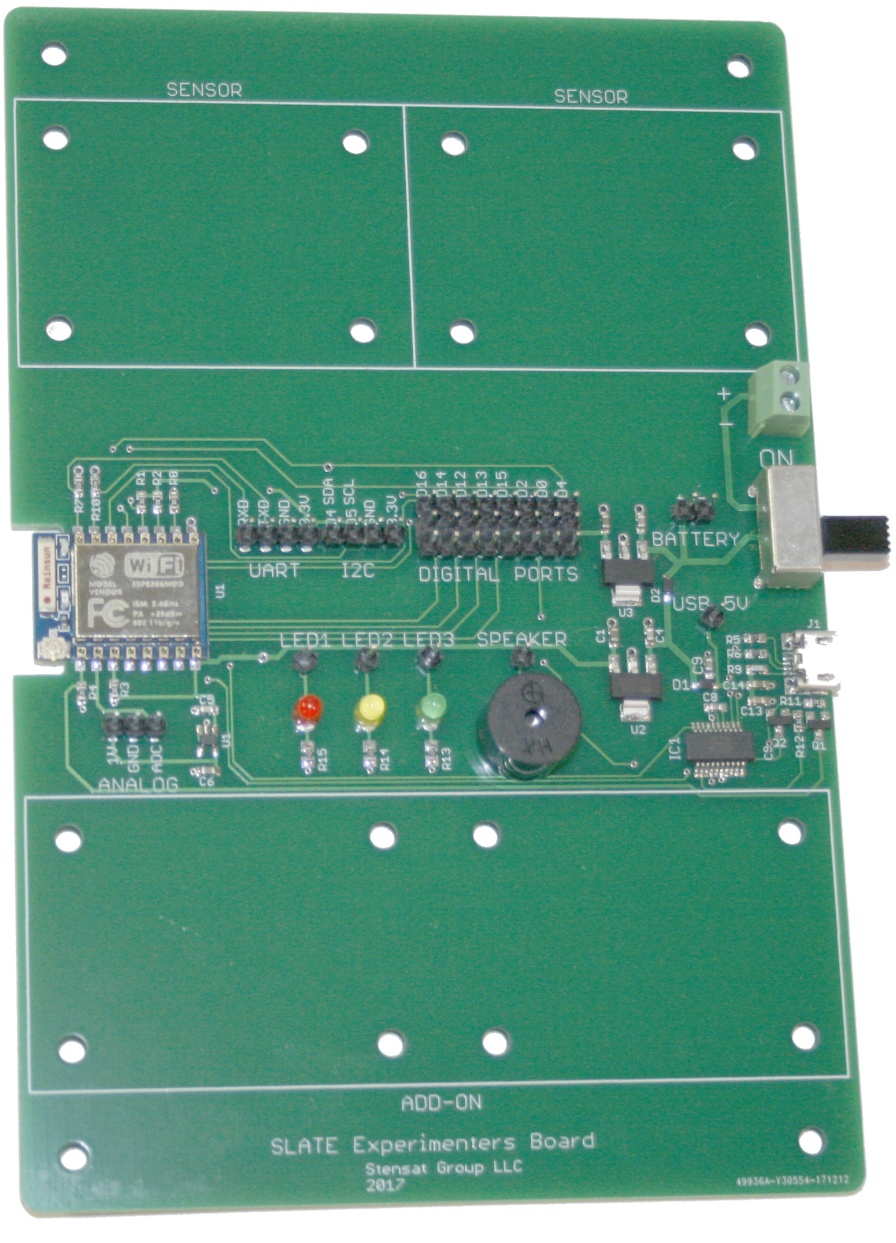
Sten-SLATE ESP Kit

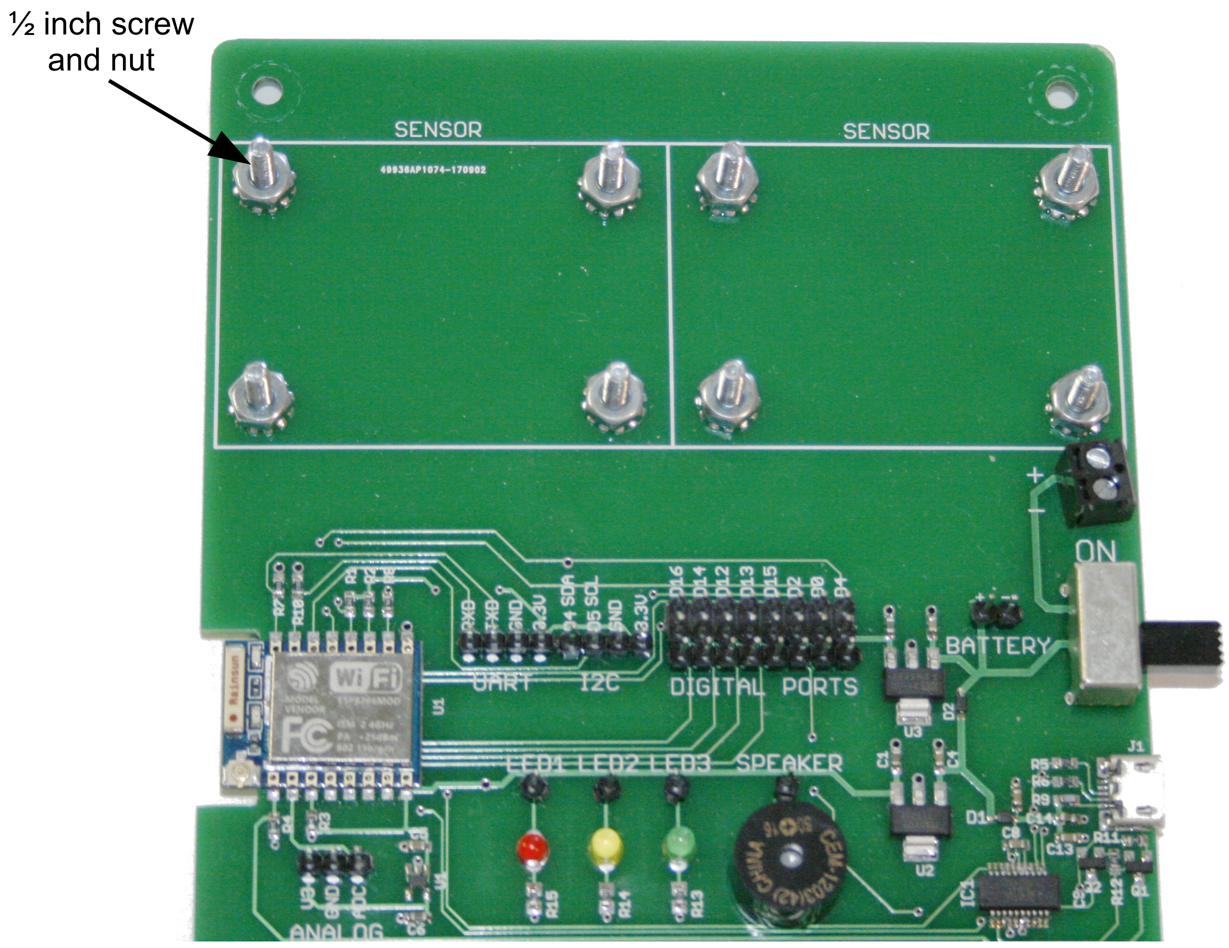
Sensors



Sensors

Mounting Sensors

There are two locations marked SENSOR for mounting sensor modules. They require screws and nuts to be installed for mounting sensor modules. Eight screws and nuts are required. Insert the screw through the hole from underneath the board and secure with a nut on the top side. Repeat for the remaining seven holes.



Analog Sensors

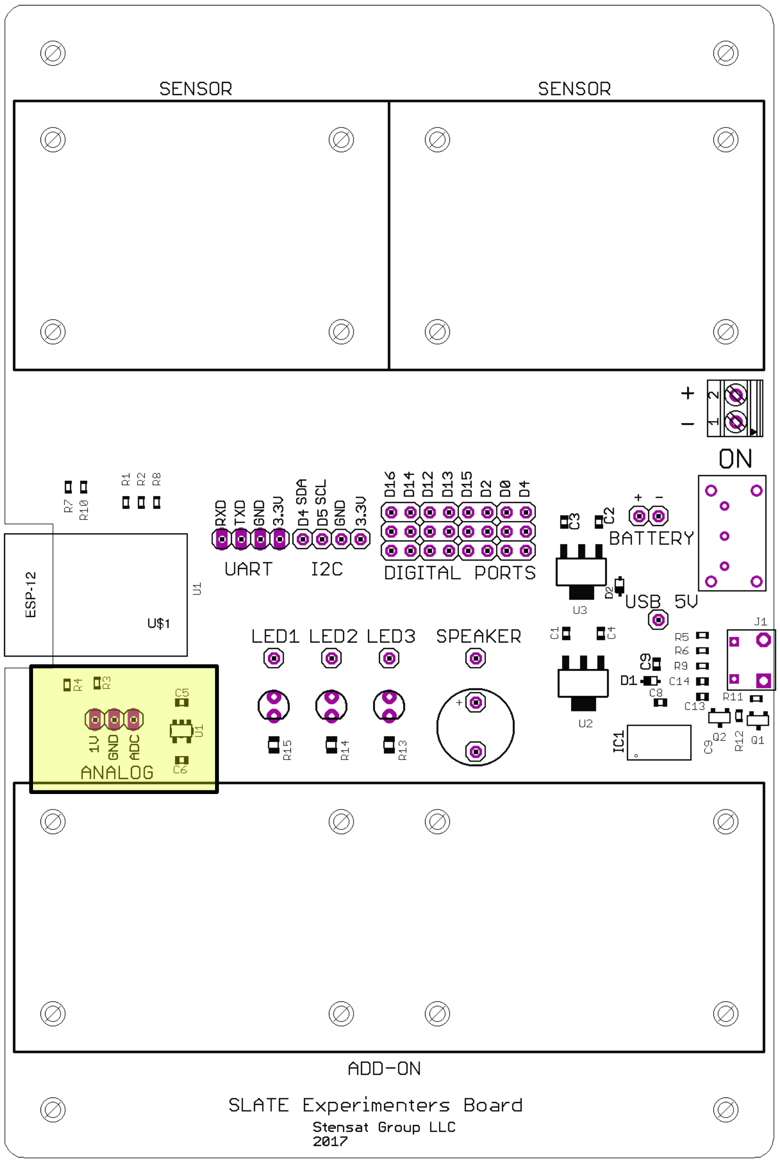
This section introduces analog sensors, a light sensor and a thermal sensor. These two sensors are variable resistors that change resistance based on what they measure.

The SLATE ESP has an analog-to-digital converter, ADC for short, that converts a voltage to a value. The SLATE ESP ADC has a voltage range of 0 to 1 volt. It is a 10 bit ADC which means it generates a number with a range from 0 to 1023. 0 volts results in a value of 0. 1 volt will generate a value of 1023. There is a linear relationship between the voltage and the value generated by the ADC. The voltage can be calculated using the equation below:

V = ADC / 1023.0

If 0.5 volts is applied to the ADC, the value generated will be 51..

The Experimenters board has one ADC input as highlighted in the image. A 1 volt reference is included at pin **V3** for any sensor circuit that needs it such as the light and thermal sensor. Pin **GND** is the 0 volt reference. Pin ADC is the input.

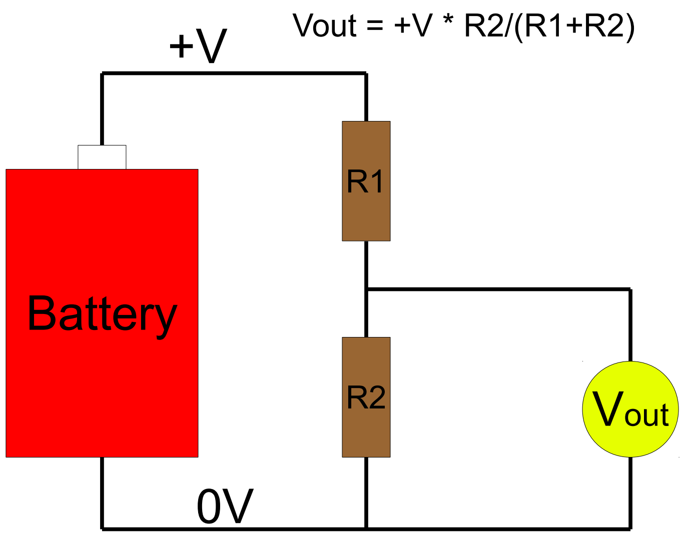


Analog Sensors

The light and thermal sensors are resistive type sensors. Their resistance changes based on what they measure. The ADC cannot measure resistance, only voltage. A voltage divider can generate a voltage based on the resistance of the sensor. The only problem is that the full range of the ADC may not be used since the resistor divider will not allow a full voltage swing from 0 to 1 volt.

The thermal sensor is called a thermistor. It's resistance changes with temperature. Their nominal resistance is specified at 25C. This particular sensor is at 10 Kohms at 25C. The table below shows the resistance at different temperatures.

The thermistor is resistor R2 as shown in the circuit. R1 is a 10 Kohm resistor. At 25C, the thermistor will be 10 Kohm. This divides the 1 voltage to 0.5 volts. The ADC will generate 511. As the temperature goes higher, the resistance decreases and the voltage to the ADC also decreases. As the temperature goes lower, the resistance increases and the voltage increases.



Voltage Divider Circuit

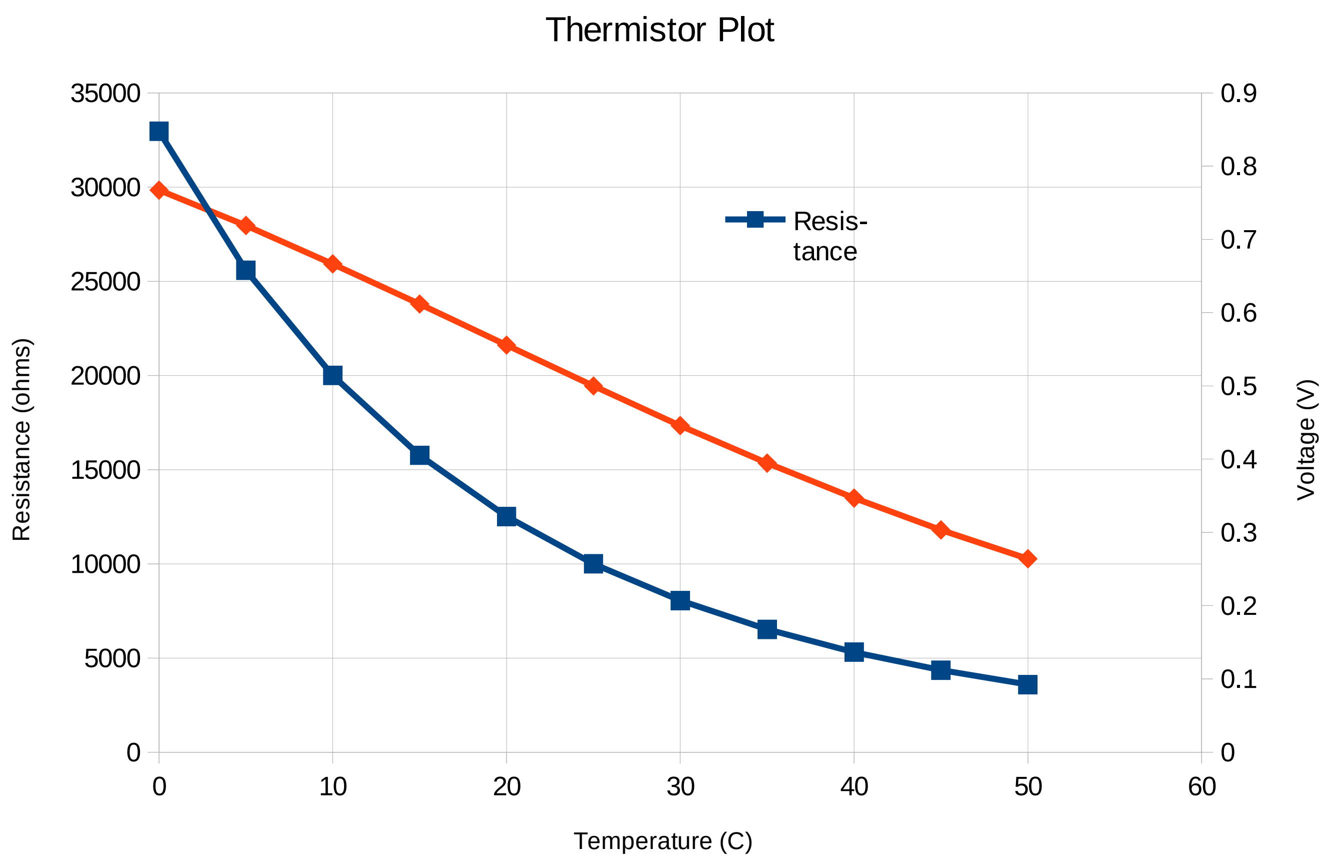
Temperature C	Resistance Ohms
0	32960
5	25580
10	20000
15	15760
20	12510
25	10000
30	8048
35	6518
40	5312
45	4354
50	3588

Thermistor Resistance Table

Thermistor

The plot of resistance versus temperature shows the thermistor does not have a linear response to temperature. The thermistor table can be entered into a spreadsheet and plotted using the scatter plot. The voltage can also be added and shown in the plot. The shows the relationship of voltage out of the voltage divider and resistance versus temperature.

The voltage plot almost looks linear and if a rough temperature is only needed, then a linear equation can be generated to calculate the temperature based on voltage.

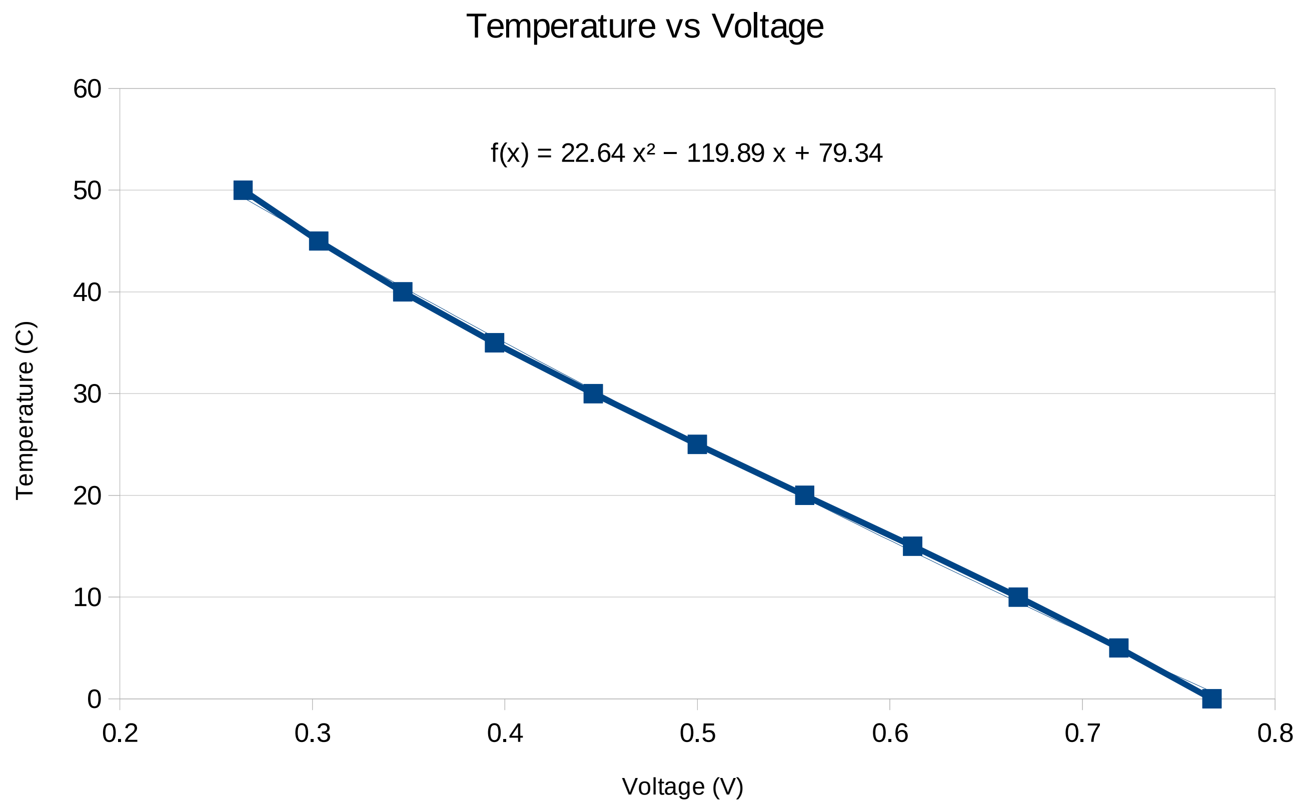


Thermistor

For better temperature accuracy, the voltage and temperature can be plotted in the spreadsheet with the temperature on the Y axis. The spreadsheet can generate a polynomial that can approximate the curve in the plot. Below is the equation generated.

Temperature = 22.64 * voltage * voltage - 119.89 * voltage + 79.34

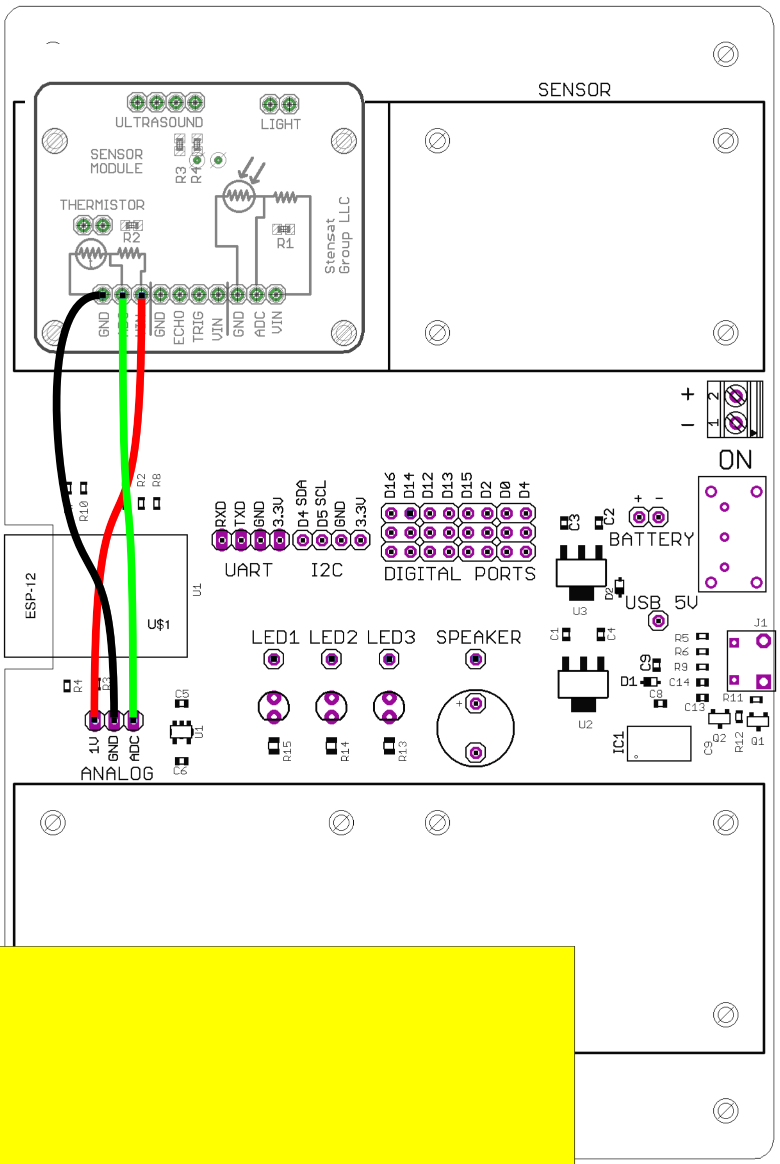
This equation will be used in the program on the next page.



Thermistor

Insert the Sensor module onto the screws in the left sensor area. Secure with #4 nuts. Use three jumper wires to connect the thermistor to the ADC.

Enter the code below and upload and run it. The result will be the air temperature. Place a finger on the sensor and see if the temperature rises.

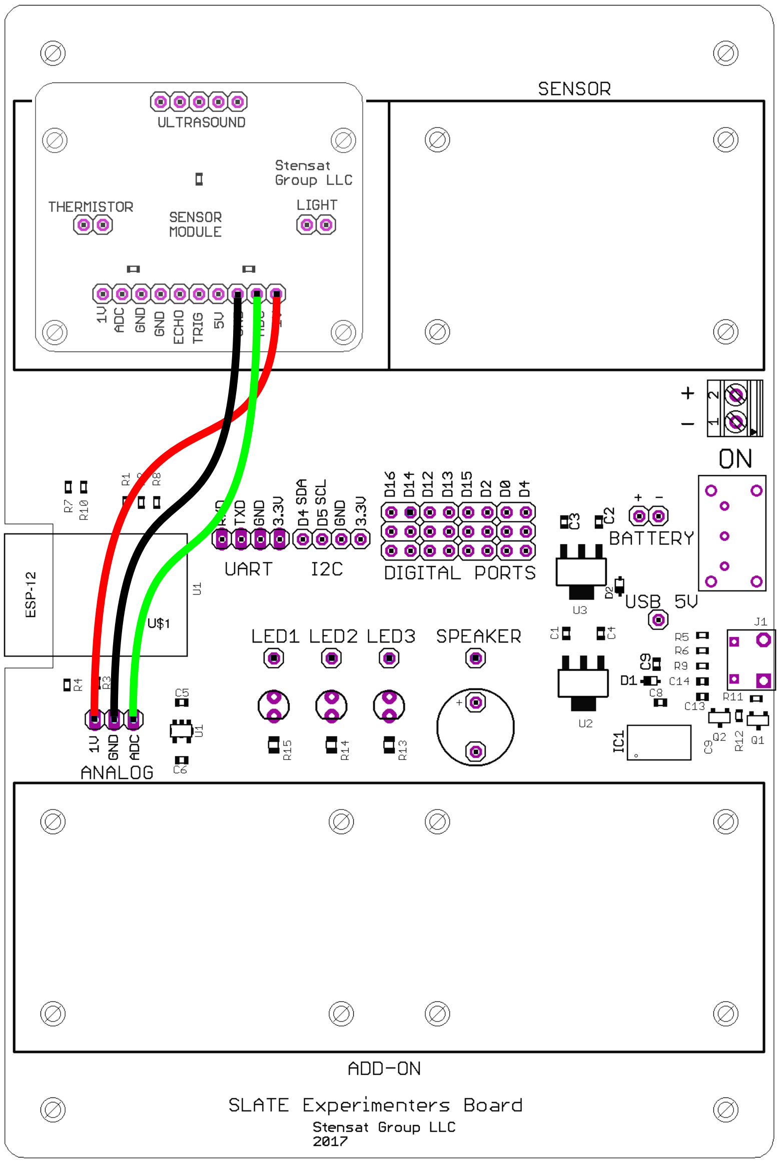


```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  int a;
  float volts;
  a = analogRead(0);
  volts = (float)a/1023.0;
  float t = (22.64*volts*volts) - (119.89 * volts) +79.34;
  Serial.println(t,2);
  delay(200);
}
```

Light Sensor

The light sensor is a light sensitive resistor. It's resistance decreases as the light intensity increases. At 100 lux, the resistance is 5 Kohms. In the dark, the resistance can reach up to 20 megaohm. It is connected similarly as the thermistor taking the place of R1 in the voltage divider circuit. Voltage will increase as the light intensity increases and the voltage decreases as light intensity decreases. There is no calibration data for the sensor. It is used as general light intensity detector and generally used to detect darkness to turn lights on.

Enter the code in a new program and try it out. Use a flash light to shine light on the sensor and see how the voltage changes.

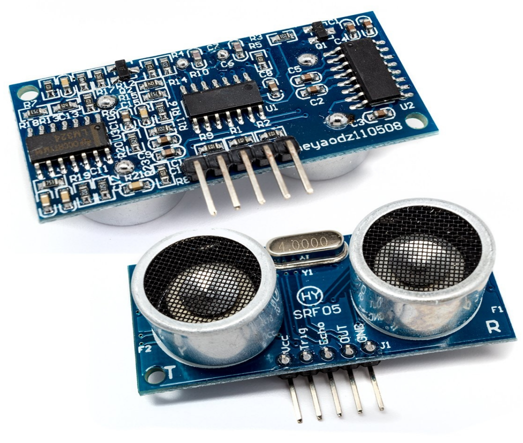


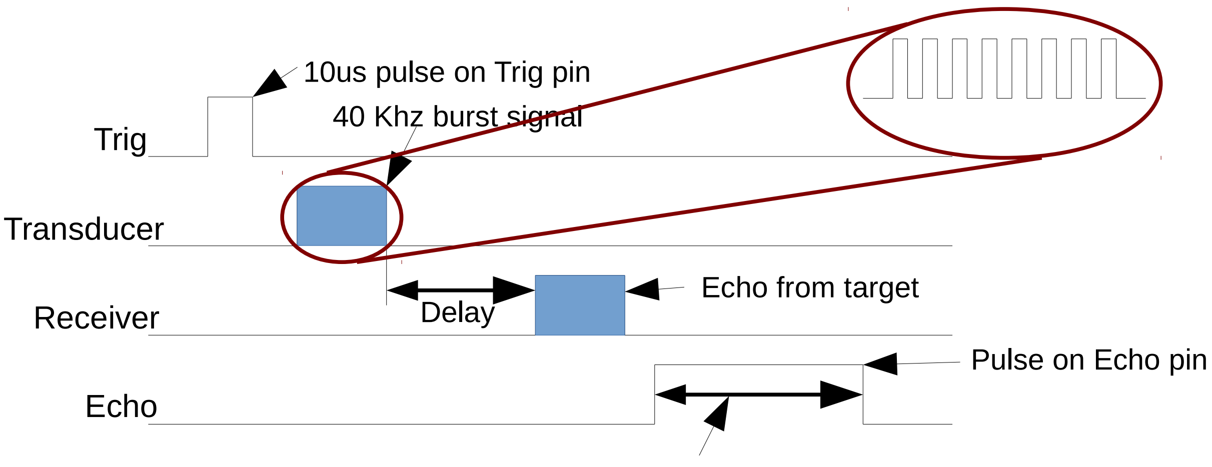
```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  int a;
  float volts;
  a = analogRead(0);
  Serial.println(a);
  volts = (float)a/1023.0;
  Serial.println(volts,2);
  delay(200);
}
```

Ultrasonic Sensor

The ultrasonic sensor uses bursts of sound to measure distance. The sensor transmits a short 40 KHz tone and then measures the time it takes the tone to be reflected back.

The distance measurement is started by the Trig pin being pulsed high for at least 10 us. The sensor will then transmit a very short 40 KHz tone and wait for the echo to be detected. The sensor calculates the delay time and generates a pulse with the width proportional to the delay. Distance can be calculated by measuring the pulse width in microseconds and dividing by 58 to get centimeter values.



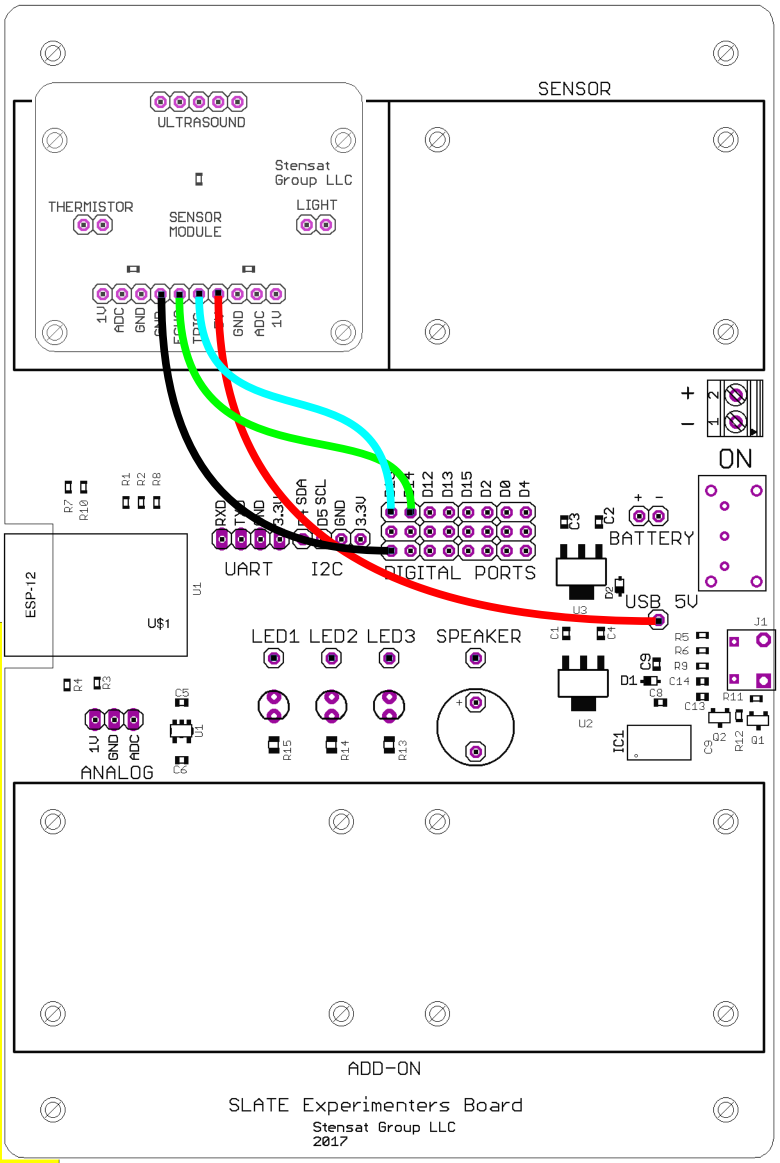


distance = pulse width (us) / 58

Ultrasonic Range Sensor

The picture shows where to install the jumper wires to connect the ultrasonic range sensor. **GND** is connected to the **GND** pin on the digital port. **5V** is connected to the USB 5V pin. Trigger is connected to **D16**. Echo is connected to **D14**.

Enter the code below in a new program and run it. Use a solid object or hand and move it to and from the sensor. A ruler can be used to verify the accuracy of the sensor. It can measure down to 3 centimeters.



```
void setup()
{
      Serial.begin(115200);
      pinMode(14,INPUT);
      pinMode(16,OUTPUT);
}

void loop()
{
      unsigned long distance;
      digitalWrite(16,LOW);
      delayMicroseconds(2);
      digitalWrite(16,HIGH);
      delayMicroseconds(10);
      digitalWrite(16,LOW);
      distance = pulseIn(14,HIGH);
      distance = distance/58;
      Serial.println(distance);
      delay(500);
}
```

Creating a Function

Using the ultrasonic range sensor program, it will be modified to become a standalone function. A function is a group of instructions with a name that can be called from a program. Functions are useful for where there is an operation that is used in multiple places in a program. This helps eliminate the need to rewrite the same code in different parts of the program. It also allows the function to be used in different programs.

Select **Save As...** under the **File** menu. Give the program the name **ultrasound_f**. Next delete the function **setup()** that is highlighted. Also delete the last two function calls at the end of the **loop()** function.

As shown in the lower right, change the **void loop()** to the name of the function. Insert the return command at the end of the new function.

Select **Save** from the **File** menu.

```
~~void setup()~~
~~{~~
~~  Serial.begin(115200);~~
~~  pinMode(14,INPUT);~~
~~  pinMode(16,OUTPUT);~~
~~}~~

void loop()
{
  unsigned long distance;
  digitalWrite(16,LOW);
  delayMicroseconds(2);
  digitalWrite(16,HIGH);
  delayMicroseconds(10);
  digitalWrite(16,LOW);
  distance = pulseIn(14,HIGH);
  distance = distance/58;
~~  Serial.println(distance);~~
~~  delay(500);~~
}
```

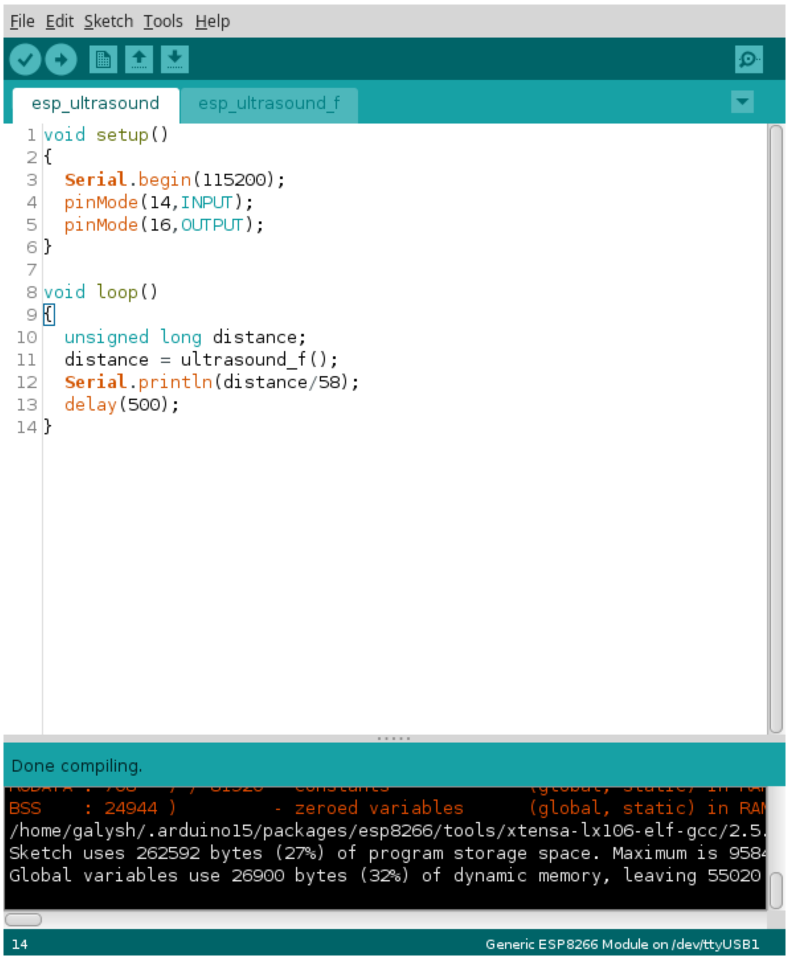
```
unsigned long ultrasound_f()
{
  unsigned long distance;
  digitalWrite(16,LOW);
  delayMicroseconds(2);
  digitalWrite(16,HIGH);
  delayMicroseconds(10);
  digitalWrite(16,LOW);
  distance = pulseIn(14,HIGH);
  distance = distance/58;
  return(distance);
}
```

Creating a Separate Function File

Create a new program. Enter the program shown. Save it with the name **ranging**. You will notice the tab for the program is now named ranging.

Under the **Sketch** menu, select **Add File...** Locate and select **ultrasound_f** file. You have to go to the **ultrasound_f** folder to locate **ultrasound_f.ino**. A new tab will appear with the ultrasound function. Compile and upload it.

```
void setup()
{
      Serial.begin(115200);
      pinMode(14,INPUT);
      pinMode(16,OUTPUT);
}

void loop()
{
      unsigned long distance;
      distance = ultrasound_f();
      Serial.println(distance/58);
      delay(500);
}
```



```
void setup()
{
  Serial.begin(115200);
  pinMode(14,INPUT);
  pinMode(16,OUTPUT);
}

void loop()
{
  unsigned long distance;
  distance = ultrasound_f();
  Serial.println(distance/58);
  delay(500);
}
```

Done compiling.

```
BSS    : 24944 )          - zeroed variables      (global, static) in RAM
/home/galysh/.arduino15/packages/esp8266/tools/xtensa-lx106-elf-gcc/2.5.
Sketch uses 262592 bytes (27%) of program storage space. Maximum is 958
Global variables use 26900 bytes (32%) of dynamic memory, leaving 55020
```

14 Generic ESP8266 Module on /dev/ttyUSB1

Arduino with two tabs.