Temperature Sensing Coaster

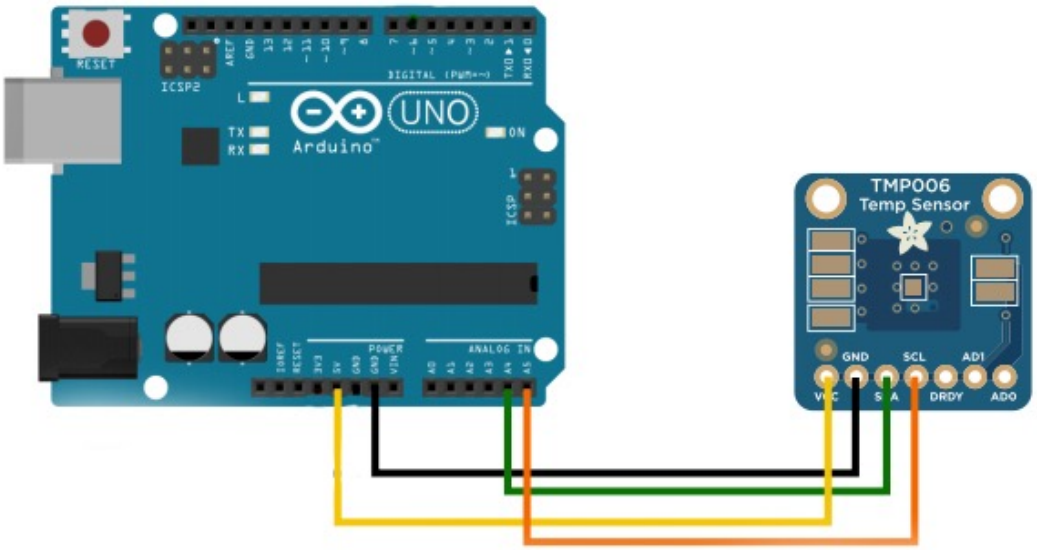
Tutorial - Smart Coaster

1. Connect the VCC on the temp sensor to 5V on .Arduino.

2. Connect the GND on the temp sensor to GND on Arduino.

3. Connect the SDA on the temp sensor to A4 on Arduino

4. Connect the SCL on the temp sensor to A5 on Arduino.

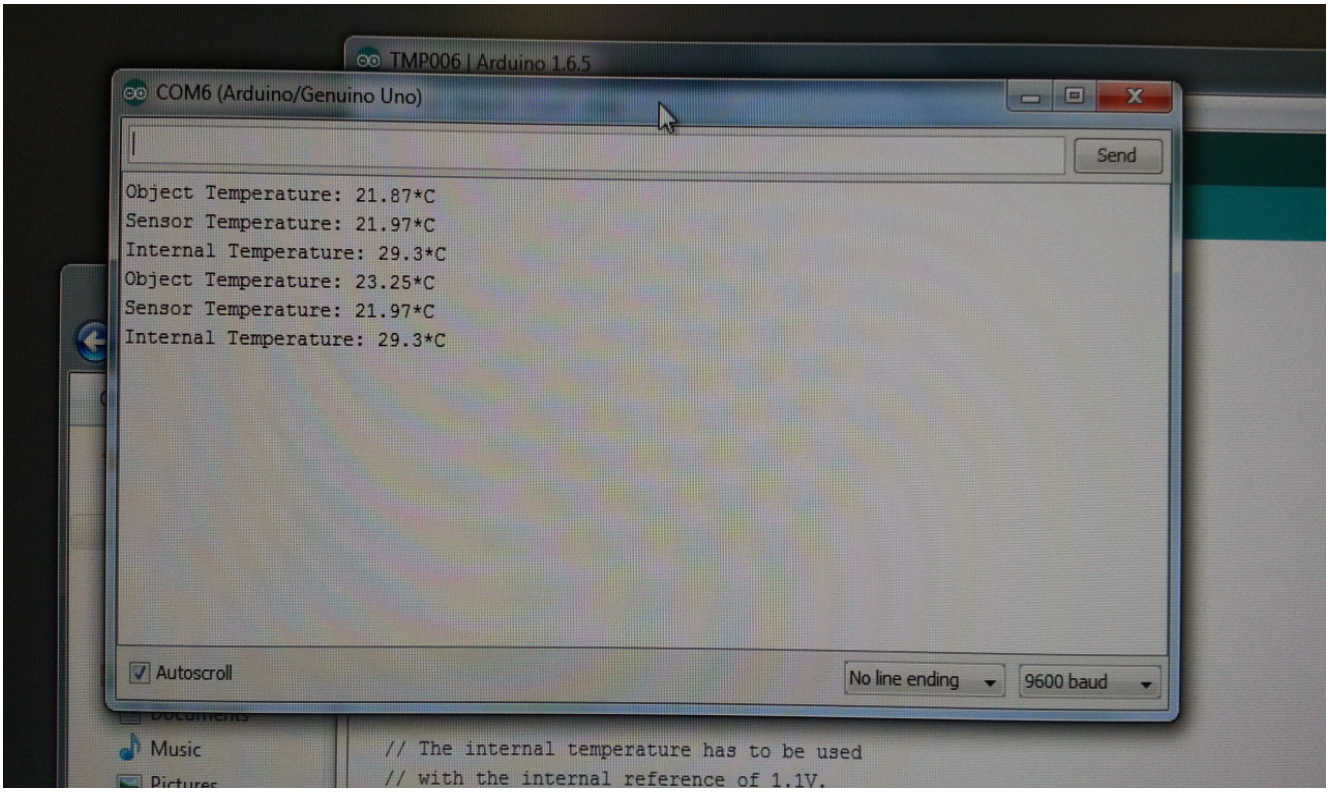


(Connect the temp sensor to the Arduino)

5. Download the code from GitHub:TMP006 Temp sensor breakout.

https://github.com/adafruit/Adafruit_TMP006

6. Upload the code to Arduino,open “Tools”,” Serial Monitor” you can the output temperature.



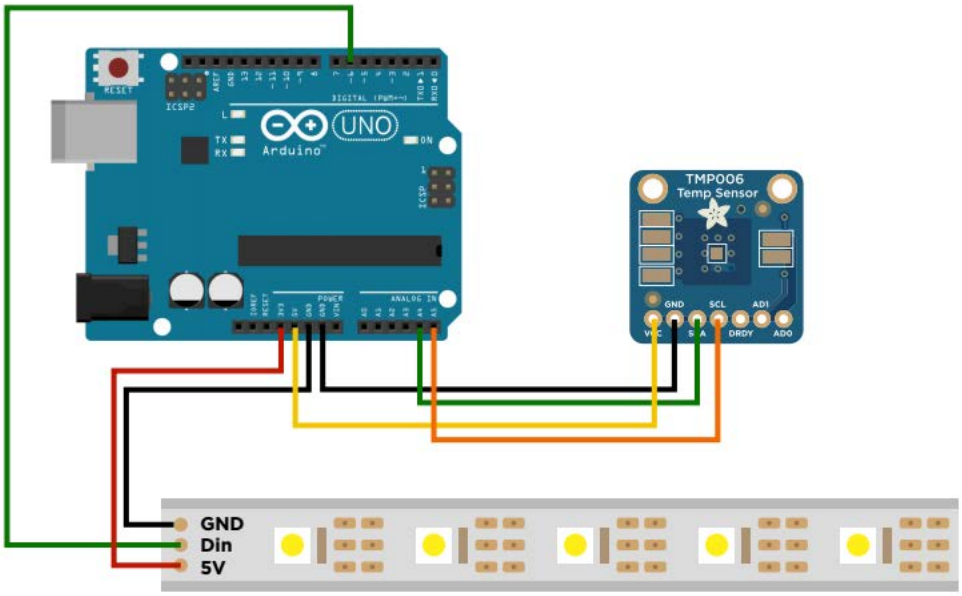
(Serial window output the temperature)

7. Now the temperature sensor is working, then try to connect the NeoPixel strip to Arduino.

8. Connect the GND from the strip to any GND on Arduino.

9. Connect the +5V input on the strip to the 3.3V on Arduino.

10. Connect the DIN to pin 6 on Arduino.



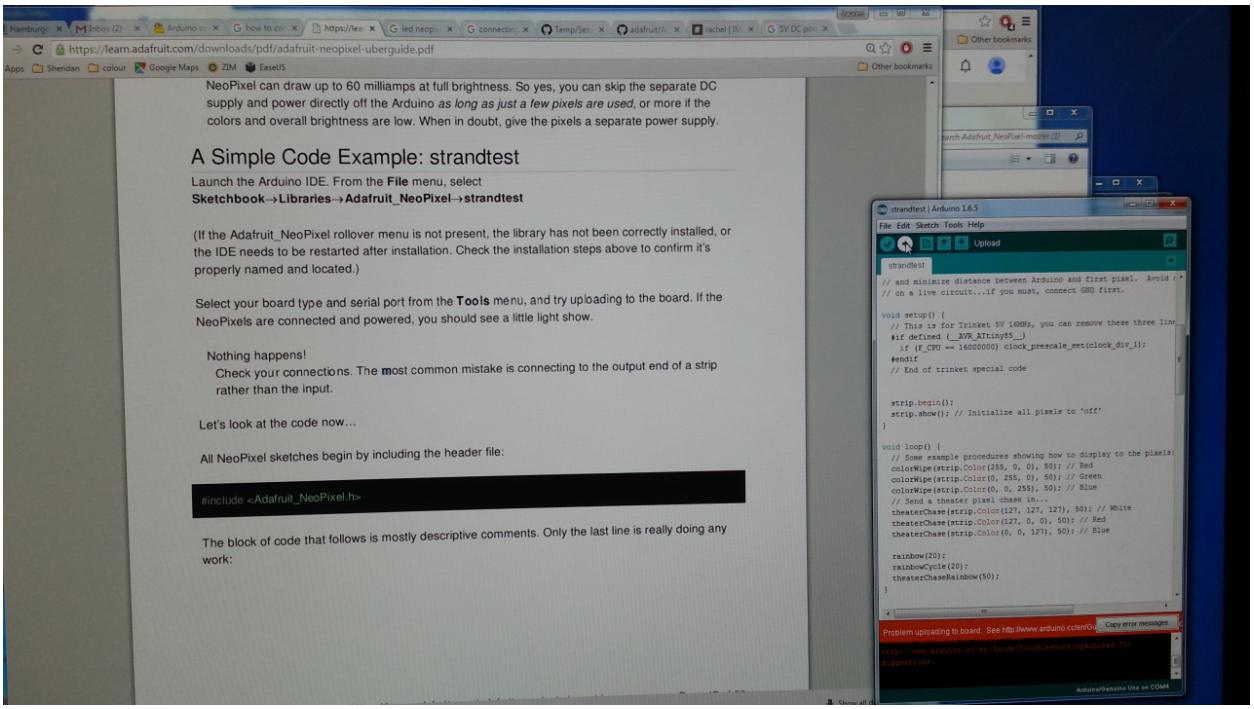
(Connect the Neo Pixel with the Arduino and temp sensor)

11. Visit https://github.com/adafruit/Adafruit_NeoPixel,Select the “Download ZIP” button, or simply click this link to download directly.

12. Uncompress the ZIP file after it’s finished downloading.

13. The resulting folder should contain the files “Adafruit _ Neo Pixel.cpp”, “Adafruit _ Neo Pixel.h” and an “examples” sub-folder. Sometimes in Windows you’ll get an intermediate-level folder and need to move things around.

14. Rename the folder (containing the .cpp and .h files) to “Adafruit_Neo Pixel” (with the underscore and everything), and place it alongside your other Arduino libraries,typically in your (home folder)/Documents/Arduino/Libraries folder. Libraries should not be installed alongside the Arduino application itself.



(Install the Neo Pixel library)

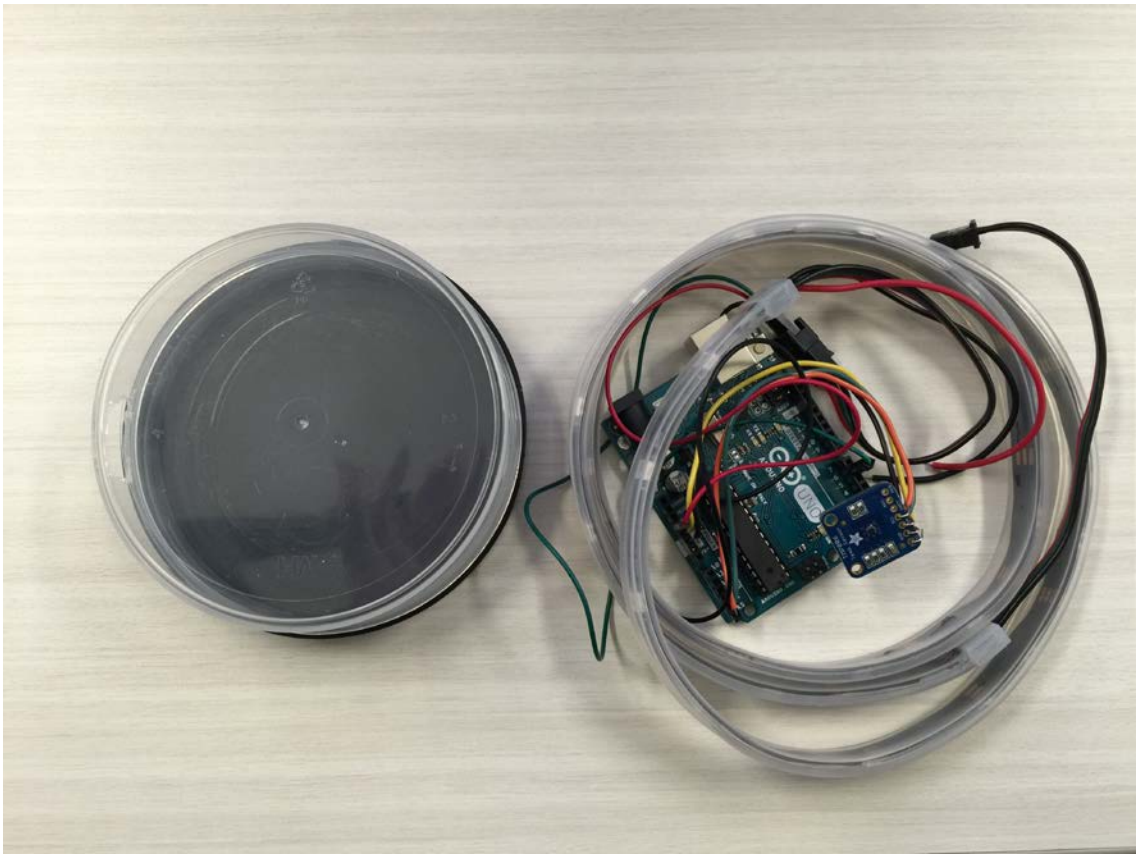
15. Re-start the Arduino IDE if it is currently running.
16. Upload the code try to light up the Neo Pixel.

(Light up the Neo Pixel)

17. Add four if statement in the code, tell the strip light to change color according to the temperature.

(Add four if statement in the code)

18. The last step is to put Arduino, temp sensor and Neo Pixel in the container.



(Arduino, temp sensor and Neo Pixel and the container)

(Done! put the hot drink on the coaster)

Our Code

```
//temperature sensor part
#include <stdint.h>
#include <math.h>
#include <Wire.h>
#include "I2C_16.h"
#include "TMP006.h"

uint8_t sensor1 = 0x40; // I2C address of TMP006, can be 0x40-0x47
uint16_t samples = 16; // # of samples per reading, can be 1/2/4/8/16

//void setup()<-this part was inserted with the neopixel void setup()
//{
//  Serial.begin(9600);
//  Serial.println("TMP006 Example");
//
//  config_TMP006(sensor1, samples);
//
//  pinMode(3, OUTPUT);
//}
//end of sensor part



// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// released under the GPLv3 license to match the rest of the AdaFruit NeoPixel library

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
  #include <avr/power.h>
#endif
```

```
// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1
#define PIN            6

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS      30

// When we setup the NeoPixel library, we tell it how many pixels, and which pin to use to send signals.
// Note that for older NeoPixel strips you might need to change the third parameter--see the strandtest
// example for more information on possible values.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

int delayval = 100; // delay for half a second

void setup() {
//temperature sensor
  Serial.begin(9600);
  Serial.println("TMP006 Example");

  config_TMP006(sensor1, samples);

  //neopixel void setup()
  // This is for Trinket 5V 16MHz, you can remove these three lines if you are not using a Trinket
#if defined (__AVR_ATtiny85__)
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
#endif
  // End of trinket special code

  pixels.begin(); // This initializes the NeoPixel library.
}
```

```
void loop() {

  float object_temp = readObjTempC(sensor1);
 Serial.print("Object Temperature: ");
 Serial.print(object_temp); Serial.println("*C");

 float sensor_temp = readDieTempC(sensor1);
 Serial.print("Sensor Temperature: ");
 Serial.print(sensor_temp); Serial.println("*C");

 Serial.print("Internal Temperature: ");
 Serial.print(GetTemp(),1); Serial.println("*C");


 // For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way up to the count of pixels minus one.


 if(object_temp>34){
  for(int i=0;i<NUMPIXELS;i++){

    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels.setPixelColor(i, pixels.Color(200,0,0)); // Lights up red for too hot

    pixels.show(); // This sends the updated pixel color to the hardware.

    delay(delayval); // Delay for a period of time (in milliseconds).

  }
 }

 if(object_temp>32 && object_temp<34){
```

```
  for(int i=0;i<NUMPIXELS;i++){

    //pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels.setPixelColor(i, pixels.Color(0,150,0)); //Green color for best temperature.

    pixels.show(); // This sends the updated pixel color to the hardware.

    delay(delayval); // Delay for a period of time (in milliseconds).

  }
}

  if(object_temp>29 && object_temp<32){
   for(int i=0;i<NUMPIXELS;i++){

    //pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels.setPixelColor(i, pixels.Color(249,207,0)); // Yellow color for a good range.

    pixels.show(); // This sends the updated pixel color to the hardware.

    delay(delayval); // Delay for a period of time (in milliseconds).

  }
}



   if(object_temp<29){
    for(int i=0;i<NUMPIXELS;i++){

    //pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels.setPixelColor(i, pixels.Color(0,0,0)); // no color
```

```
    pixels.show(); // This sends the updated pixel color to the hardware.

    delay(delayval); // Delay for a period of time (in milliseconds).

  }
 }

}

double GetTemp(void)
{
  unsigned int wADC;
  double t;

  // The internal temperature has to be used
  // with the internal reference of 1.1V.
  // Channel 8 can not be selected with
  // the analogRead function yet.

  // Set the internal reference and mux.
  ADMUX = (_BV(REFS1) | _BV(REFS0) | _BV(MUX3));
  ADCSRA |= _BV(ADEN);  // enable the ADC

  delay(20);            // wait for voltages to become stable.

  ADCSRA |= _BV(ADSC);  // Start the ADC

  // Detect end-of-conversion
  while (bit_is_set(ADCSRA,ADSC));

  // Reading register "ADCW" takes care of how to read ADCL and ADCH.
  wADC = ADCW;
```

```
  // The offset of 324.31 could be wrong. It is just an indication.
  t = (wADC - 324.31 ) / 1.22;

  // The returned temperature is in degrees Celcius.
  return (t);
}
```