Introduction

The Bubble Display is a tiny 4-digit, 7-segment display. It's perfect if you need some user feedback from your system, but don't want to fiddle with LCDs or other display options. The Bubble Display comes in an easy-to-use DIP package and can be used in breadboards, protoboards, or PCBs.
Covered in This Tutorial

This tutorial will go over the basic hardware requirements for the bubble display, show you how to hook one up to an Arduino microcontroller, and then show you a few different example sketches.

Required Materials

To follow along with all of the examples in this tutorial, you will need the following parts:

<table>
<thead>
<tr>
<th>Bubble Display</th>
<th>SparkFun Wish List</th>
</tr>
</thead>
</table>
Suggested Reading

Before you go any further, check the tutorials below, and makes sure you have a good understanding of the concepts mentioned.

- What is an Arduino?
- Working with Wire
- How to Use a Breadboard
- Integrated Circuits (ICs)
- LEDs
- Current Limiting Resistors

Hardware Overview

Let's briefly discuss the characteristics and functionality of the bubble display before we connect it to other hardware.

All the information found on this page and more can be found in the product's datasheet. Consult it, if you need more information.

Power

The bubble display is unique in that, unlike most ICs, these displays don't need a supply voltage. The bubble display is four, 7-segment displays packed into one package, and 7-segment displays are really just seven LEDs lined up in a particular pattern. In this case, the number '8' shape we're all familiar with. Thus, to turn on and off a particular part of the display, you send it a digital HIGH or LOW signal just like
you would with a regular LED.

However, you may have noticed that there are more LEDs than there are pins on the IC. There are 32 segments total (counting the decimal points) and only 12 pins. What gives? The answer is that these displays use *common-cathode* LEDs, which means all the LEDs in a given segment share one ground pin. This is the same method used in *RGB LEDs* that allows you to control three different colored LEDs with only four pins. With these displays, there are eight pins for the anodes (+) (seven segments and one decimal point per digit) and four pins for the cathode (-) of each digit.

![A closeup of the bubble display.](image)

With that said, you still want to limit the amount of current flowing through the LEDs. If you are using a 5V microcontroller such as a [SparkFun RedBoard](https://www.sparkfun.com/products/11996), you'll need to add *current limiting resistors* to your circuit. Even if you're using a 3.3V micro, you should still add resistors, albeit much smaller values. The following tables from the datasheet give us the information we need to calculate the correct resistor value.
LED calculators, such as this one make it easy to figure out which value resistor to use. For example, if you're using a 5V source to control the segments, you'll want to have a 680 Ω resistor in line with each, assuming that each segment is operating under normal conditions. If you find that the display isn't as bright as you'd like it, you can always add a smaller value resistor. We've found that a 330Ω resistor works well on all four cathode pins.

Pinout

Now that we know how to properly provide power to the display, let's go over the segment configuration so we know which pins light up which segments. The pinout for the bubble display is as follows.
## Device Pin Description

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CATHODE 1</td>
</tr>
<tr>
<td>2</td>
<td>ANODE e</td>
</tr>
<tr>
<td>3</td>
<td>ANODE c</td>
</tr>
<tr>
<td>4</td>
<td>CATHODE 3</td>
</tr>
<tr>
<td>5</td>
<td>ANODE dp</td>
</tr>
<tr>
<td>6</td>
<td>CATHODE 4</td>
</tr>
<tr>
<td>7</td>
<td>ANODE g</td>
</tr>
<tr>
<td>8</td>
<td>ANODE d</td>
</tr>
<tr>
<td>9</td>
<td>ANODE f</td>
</tr>
<tr>
<td>10</td>
<td>CATHODE 2</td>
</tr>
<tr>
<td>11</td>
<td>ANODE b</td>
</tr>
<tr>
<td>12</td>
<td>ANODE a</td>
</tr>
</tbody>
</table>

Next, take a look at the segment key.
Last, here is a diagram showing the numbering of the digits as they correspond to the pins labeled above.

Don't forget that most ICs are polarized. Pin 1 is indicated by the small dot on the underside of the board.

Building from what we discussed above, we can start to make sense of this. For example, let's say we wanted to turn on Segment_A on Digit_1. Looking at the table above we see that Segment_A (labeled Anode a) is pin 12. Looking at the same table, we can also see that the cathode for Digit_1 is pin 1. If you connect pin 1 to GND, and pin 12 to 3.3V or 5V (don't forget that resistor!), you should see the topmost segment on the leftmost digit light up. Yay LEDs! From here you can extrapolate how the rest works. If you wanted to light up Segment_A on both Digit_1 and Digit_4, you would simply add another connection to our first example from GND to pin 6 (cathode 4), and so on...

Lighting up individual segments is fun for about a minute, so let's move on and get this thing displaying some useful information.

**Hooking it Up**

In order to talk to the bubble display, we'll need a microcontroller. This example will use a SparkFun RedBoard to control the display, but you can use whichever you choose. You'll also need some jumper wires, some resistors, and a breadboard.

This IC is polarized, so pay attention to which way you stick it in the breadboard! Pin one is indicated by a white dot on the underside of the part.
Once you have the bubble display oriented correctly on the breadboard, hook everything up like the image below shows:

Note the polarity mark on the bottom-left corner of the bubble display. Click the image for a larger view.

While the display may work without current limiting resistors, it's always a good idea to have them in your circuit to avoid burning out your display. You may also opt to have your resistors on the eight anode pins rather than the four cathode pins. You will need to add a little more hardware, but the result will be a much more balanced brightness in your display no matter which character you're displaying.
Now, it's time to light up the display with some code.

**Firmware**

Before you can start writing code to control the bubble displays, you'll need to install the SevSeg Arduino Library first. You can download it from the [GitHub repo](https://github.com). If you need more details on installing a library, visit our [Installing an Arduino Library](#) tutorial, and, if you are unfamiliar with GitHub, there's a [tutorial](#) for that too.

Once the library is installed, we can use the included example sketches to get started using the bubble display.

**Example 1: Counter**

Go to File -> Examples -> SevSeg -> Example -> Bubble_Display.Counter.

Here is the code for those who would like to see it without downloading it.

```c
language:c
/*
 * March 6, 2014
 * Spark Fun Electronics
 * Nathan Seidle
 * Updates by Joel Bartlett
 *
 * This code is originally based Dean Reading's Library deanreading@hotmail.com
 * http://arduino.cc/playground/Main/SevenSegmentLibrary
 * He didn't have a license on it so I hope he doesn't mind me making it public domain:
 * This code is public domain but you buy me a beer if you use this and we meet someday (Beerware license).
 *
 * This sketch provides a simple counter example for the HP Bubble display from SparkFun.
 * https://www.sparkfun.com/products/12710
 *
 * Pinout for HP Bubble Display:
 * 1: Cathode 1
 * 2: Anode E
 * 3: Anode C
 * 4: Cathode 3
 * 5: Anode dp
 * 6: Cathode 4
 * 7: Anode G
 * 8: Anode D
 * 9: Anode F
 * 10: Cathode 2
 * 11: Anode B
 * 12: Anode A
 */

#include "SevSeg.h"

//Create an instance of the object.
SevSeg myDisplay;
```
//Create global variables
unsigned long timer;
int deciSecond = 0;

void setup()
{
    int displayType = COMMON_CATHODE; //Your display is either common cathode or common anode

    //This pinout is for a bubble dispaly
    //Declare what pins are connected to the GND pins (cathodes)
    int digit1 = 8; //Pin 1
    int digit2 = 5; //Pin 10
    int digit3 = 11; //Pin 4
    int digit4 = 13; //Pin 6

    //Declare what pins are connected to the segments (anodes)
    int segA = 7; //Pin 12
    int segB = 6; //Pin 11
    int segC = 10; //Pin 3
    int segD = 3; //Pin 8
    int segE = 9; //Pin 2
    int segF = 4; //Pin 9
    int segG = 2; //Pin 7
    int segDP = 12; //Pin 5

    int numberOfDigits = 4; //Do you have a 1, 2 or 4 digit display?

    myDisplay.Begin(displayType, numberOfDigits, digit1, digit2, digit3, digit4, segA, segB, segC, segD, segE, segF, segG, segDP);

    myDisplay.SetBrightness(100); //Set the display to 100% brightness level

    timer = millis();
}

void loop()
{
    //Example ways of displaying a decimal number
    char tempString[10]; //Used for sprintf
    sprintf(tempString, "%4d", deciSecond); //Convert deciSecond into a string that is right adjusted
    //sprintf(tempString, "%d", deciSecond); //Convert deciSecond into a string that is left adjusted
    //sprintf(tempString, "%04d", deciSecond); //Convert deciSecond into a string with leading zeros
    //sprintf(tempString, "%4d", deciSecond * -1); //Shows a negative sign infront of right adjusted number
    //sprintf(tempString, "%4X", deciSecond); //Count in HEX, right adjusted

    //Produce an output on the display
    myDisplay.DisplayString(tempString, 0); //(numberToDisplay, decimal point location)

    //Other examples
    //myDisplay.DisplayString(tempString, 0); //Display string, no decimal point
    //myDisplay.DisplayString("-23b", 3); //Display string, decimal point in third position

    //Check if 10ms has elapsed
    if (millis() - timer >= 100)
    {
        timer = millis();
        deciSecond++;
    }
With this sketch uploaded, the display will count up, with each order of magnitude spilling onto the next digit to the left.

**Example 2: Text**

The next example shows how to display text. Since these displays are 7-segment and not alphanumeric, they are limited in which characters they can and can't display. The sketch has a table listing all the characters and whether or not they can be displayed.

To open this sketch, go to File -> Examples -> SevSeg -> Example -> Bubble_Display_Text.

```c
/*
March 6, 2014
Spark Fun Electronics
Nathan Seidle
Updates by Joel Bartlett

This code is originally based Dean Reading's Library deanreading@hotmail.com
http://arduino.cc/playground/Main/SevenSegmentLibrary
He didn't have a license on it so I hope he doesn't mind me making it public domain:
This code is public domain but you buy me a beer if you use this and we meet someday (Beerware license).

The example shows you how to display characters to the bubble display. Since this is a 7-segment display and not an alpha-numeric display, there are numerous characters that cannot be displayed. The table below should give you an idea which ones can.

Pinout for HP Bubble Display:
1: Cathode 1
2: Anode E
3: Anode C
4: Cathode 3
5: Anode dp
6: Cathode 4
7: Anode G
8: Anode D
9: Anode F
10: Cathode 2
11: Anode B
12: Anode A

Here is the character map found in the SevSeg.h file for reference as to which characters can be displayed and which can't.

// ABCDEFG Segments
0b1111110, // 0
0b0110000, // 1
0b1101101, // 2
0b1111001, // 3
0b0110011, // 4
```
```cpp
#include "SevSeg.h"

//Create an instance of the object.
SevSeg myDisplay;

//Create global variables
unsigned long timer;
int deciSecond = 0;

void setup()
{

    int displayType = COMMON_CATHODE; //Your display is either common cathode or common anode

    //This pinout is for a bubble display
    //Declare what pins are connected to the GND pins (cathodes)
    int digit1 = 8; //Pin 1
    int digit2 = 5; //Pin 10
    int digit3 = 11; //Pin 4
    int digit4 = 13; //Pin 6

    //Declare what pins are connected to the segments (anodes)
    int segA = 7; //Pin 12
    int segB = 6; //Pin 11
    int segC = 10; //Pin 3
    int segD = 3; //Pin 8
    int segE = 9; //Pin 2
    int segF = 4; //Pin 9
    int segG = 2; //Pin 7
    int segDP = 12; //Pin 5

    int numberOfDigits = 4; //Do you have a 1, 2 or 4 digit display?

    myDisplay.Begin(displayType, numberOfDigits, digit1, digit2, digit3, digit4, segA, segB, segC, segD, segE, segF, segG, segDP);

    myDisplay.SetBrightness(100); //Set the display to 100% brightness level

    timer = millis();
}

void loop()
{

    for(int i = 0; i < 100; i++)
    {

        myDisplay.DisplayString(tempString, 0); // (numberToDisplay, decimal point location)
    }
}
```
delay(10);
}

for(int i =0;i<100;i++)
{
    tempString[0] = 58;
    tempString[1] = 'I';
    tempString[2] = 'S';
    tempString[3] = 58;

    myDisplay.DisplayString(tempString, 0);

delay(10);
}

for(int i =0;i<100;i++)
{
    tempString[0] = 'C';
    tempString[1] = 'O';
    tempString[2] = 'O';
    tempString[3] = 'L';

    myDisplay.DisplayString(tempString, 0);

delay(10);
}

}

Example 3: Sensor Data

This last example is more of a real-world application. By adding a temperature and humidity sensor to the RedBoard, we can make a tiny, digital thermometer and hygrometer.

First, you need to add a SHT15 to your circuit. You may use whatever sensors you like, but you will need to alter the code accordingly.
Now, go to File -> Examples -> SevSeg -> Example -> Bubble_Display_SHT15 to open up the sketch.

March 6, 2014
Spark Fun Electronics
Nathan Seidle
Updates by Joel Bartlett

This code is originally based Dean Reading's Library deanreading@hotmail.com
http://arduino.cc/playground/Main/SevenSegmentLibrary
He didn't have a license on it so I hope he doesn't mind me making it public domain:
This code is public domain but you buy me a beer if you use this and we meet someday (Beerware license).

This example gives you a real world scenario in which we take sensor data (in this case temperature and humidity data from a SHT15) and print it to the bubble display using characters and numbers.

Pinout for HP Bubble Display:
1: Cathode 1
2: Anode E
3: Anode C
4: Cathode 3
5: Anode dp
6: Cathode 4
7: Anode G
8: Anode D
9: Anode F
10: Cathode 2
11: Anode B
12: Anode A

Pinout for SHT15:
Vcc: 3.3V
Data: A4 or SDA on newer Arduino boards
SCK: A5 or SCK on newer Arduino boards
GND: GND

Here is the character map found in the SevSeg.h file for reference as to which characters can be displayed and which can't.

// ABCDEFG Segments
0b111110, // 0
0b0110000, // 1
0b1101101, // 2
0b1111001, // 3
0b0110011, // 4
0b1011011, // 5
0b1011111, // 6
0b1110000, // 7
0b1111111, // 8
0b1111011, // 9
0b1110111, // 10 "A"
0b1101111, // 11 "B"
0b1001110, // 12 "C"
0b1011101, // 13 "D"
SHT15 code based on the code found at Wiring.org
*include "SevSeg.h"

//Create an instance of the object.
SevSeg myDisplay;

int temperatureCommand  = B00000011;  // command used to read temperature
int humidityCommand = B00000101;  // command used to read humidity

int clockPin = A5;  // pin used for clock
int dataPin  = A4;  // pin used for data
int ack;  // track acknowledgment for errors
int val;
float tempF;
float humidity;
float tempC;

char tempString[5];

void setup()
{
  int displayType = COMMON_CATHODE; //Your display is either common cathode or common anode

  //This pinout is for a bubble dispaly
  //Declare what pins are connected to the GND pins (cathodes)
  int digit1 = 8; //Pin 1
  int digit2 = 5; //Pin 10
  int digit3 = 11; //Pin 4
  int digit4 = 13; //Pin 6

  //Declare what pins are connected to the segments (anodes)
  int segA = 7; //Pin 12
  int segB = 6; //Pin 11
  int segC = 10; //Pin 3
  int segD = 3; //Pin 8
  int segE = 9; //Pin 2
  int segF = 4; //Pin 9
  int segG = 2; //Pin 7
  int segDP= 1; //Pin 5

  int numberOfDigits = 4; //Do you have a 1, 2 or 4 digit display?

  myDisplay.Begin(displayType, numberOfDigits, digit1, digit2, digit3, digit4, segA, segB, segC, segD, segE, segF, segG, segDP);
  myDisplay.SetBrightness(100); //Set the display to 100% brightness level
}

void loop()
{
  sht15();
  clear();
  for(int i =0; i<500; i++)
  {
  
}
tempString[0] = 'T';
tempString[1] = 'F';

myDisplay.DisplayString(tempString, 0);

sprintf(tempString, "%4d", (long)tempF, DEC);
myDisplay.DisplayString(tempString, 0);
}
clear();
for(int i =0;i<500;i++)
{
    tempString[0] = 'R';
tempString[1] = 'H';

    myDisplay.DisplayString(tempString, 0);

    sprintf(tempString, "%4d", (long)humidity, DEC);
    myDisplay.DisplayString(tempString, 0);
}

//-------------------------------------------------------------------------------------------
void clear()
{
    //write a non-displayable character to each position to clear the display
    tempString[0] = 'm';
tempString[1] = 'm';
tempString[2] = 'm';
tempString[3] = 'm';

    myDisplay.DisplayString(tempString, 0);
}

//-------------------------------------------------------------------------------------------
int sht15()
{
    // read the temperature and convert it to centigrades
    sendCommandSHT(temperatureCommand, dataPin, clockPin);
    waitForResultSHT(dataPin);
    val = getData16SHT(dataPin, clockPin);
    skipCrcSHT(dataPin, clockPin);
    tempC = (float)val * 0.01 - 40;
tempF = (float)tempC * 1.8 + 32;

    //Serial.print("Temperature: ");
    //Serial.print((long)tempF, DEC);
    //Serial.print("_");
    //Serial.print((char)176);
    //Serial.print("F ");

    // read the humidity
    sendCommandSHT(humidityCommand, dataPin, clockPin);
    waitForResultSHT(dataPin);
    val = getData16SHT(dataPin, clockPin);
    skipCrcSHT(dataPin, clockPin);
humidity = -4.0 + 0.0405 * val + -0.0000028 * val * val;

    //Serial.print(" Relative Humidity: ");
    //Serial.print((long)humidity, DEC);
    //Serial.println("%");
// commands for reading/sending data to a SHTx sensor

int shiftIn(int dataPin, int clockPin, int numBits) {
    int ret = 0;

    for (int i=0; i<numBits; ++i) {
        digitalWrite(clockPin, HIGH);
        //delay(10); not needed ;)
        ret = ret*2 + digitalRead(dataPin);
        digitalWrite(clockPin, LOW);
    }
    return(ret);
}

// send a command to the SHTx sensor
void sendCommandSHT(int command, int dataPin, int clockPin) {
    int ack;

    // transmission start
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
    digitalWrite(dataPin, HIGH);
    digitalWrite(clockPin, HIGH);
    digitalWrite(dataPin, LOW);
    digitalWrite(clockPin, LOW);
    digitalWrite(clockPin, HIGH);
    digitalWrite(dataPin, HIGH);
    digitalWrite(clockPin, LOW);

    // shift out the command (the 3 MSB are address and must be 000, the last 5 bits are the command)
    shiftOut(dataPin, clockPin, MSBFIRST, command);

    // verify we get the right ACK
    digitalWrite(clockPin, HIGH);
    pinMode(dataPin, INPUT);
    ack = digitalRead(dataPin);
    if (ack != LOW)
        Serial.println("ACK error 0");
    digitalWrite(clockPin, LOW);
    ack = digitalRead(dataPin);
    if (ack != HIGH)
        Serial.println("ACK error 1");
}

// wait for the SHTx answer
void waitForResultSHT(int dataPin) {
    int ack;

    pinMode(dataPin, INPUT);
    for(int i=0; i<100; ++i) {
        delay(10);
        ack = digitalRead(dataPin);
        if (ack == LOW)
            break;
    }
}
break;
}
if (ack == HIGH)
    Serial.println("ACK error 2");
}

// get data from the SHTx sensor
int getData16SHT(int dataPin, int clockPin) {
    int val;
    // get the MSB (most significant bits)
    pinMode(dataPin, INPUT);
    pinMode(clockPin, OUTPUT);
    val = shiftIn(dataPin, clockPin, 8);
    val *= 256; // this is equivalent to \texttt{val << 8};
    // send the required ACK
    pinMode(dataPin, OUTPUT);
    digitalWrite(dataPin, HIGH);
    digitalWrite(dataPin, LOW);
    digitalWrite(clockPin, HIGH);
    digitalWrite(clockPin, LOW);
    // get the LSB (less significant bits)
    pinMode(dataPin, INPUT);
    val |= shiftIn(dataPin, clockPin, 8);
    return val;
}

// skip CRC data from the SHTx sensor
void skipCrcSHT(int dataPin, int clockPin) {
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
    digitalWrite(dataPin, HIGH);
    digitalWrite(dataPin, LOW);
    digitalWrite(clockPin, HIGH);
    digitalWrite(clockPin, LOW);
}

//------------------------------------------------------------------------------------------------------------------

When you run the sketch, it should start off by printing $tF$ followed by the temperature in fahrenheit. After ten seconds, it should display $rH$ followed by the current relative humidity. After ten more seconds, the cycle will repeat.
Check out the Product Showcase video to see the display in action.

Resources and Going Further

For more information on the bubble display, check out the resources below:

- Datasheet (PDF)
- Product Showcase: Bubblicious

For more great tutorials and 7-segment displays and LCDs, check out these other offerings from SparkFun Electronics:

- OpenSegment Hookup Guide
- Using the Serial 7-Segment Display
- Galileo Unread Email Counter
- Reaction Timer
- Dungeons and Dragons Dice Gauntlet
- Serial Graphic LCD Hookup
- Graphic LCD Hookup Guide
- OLED Display Hookup Guide