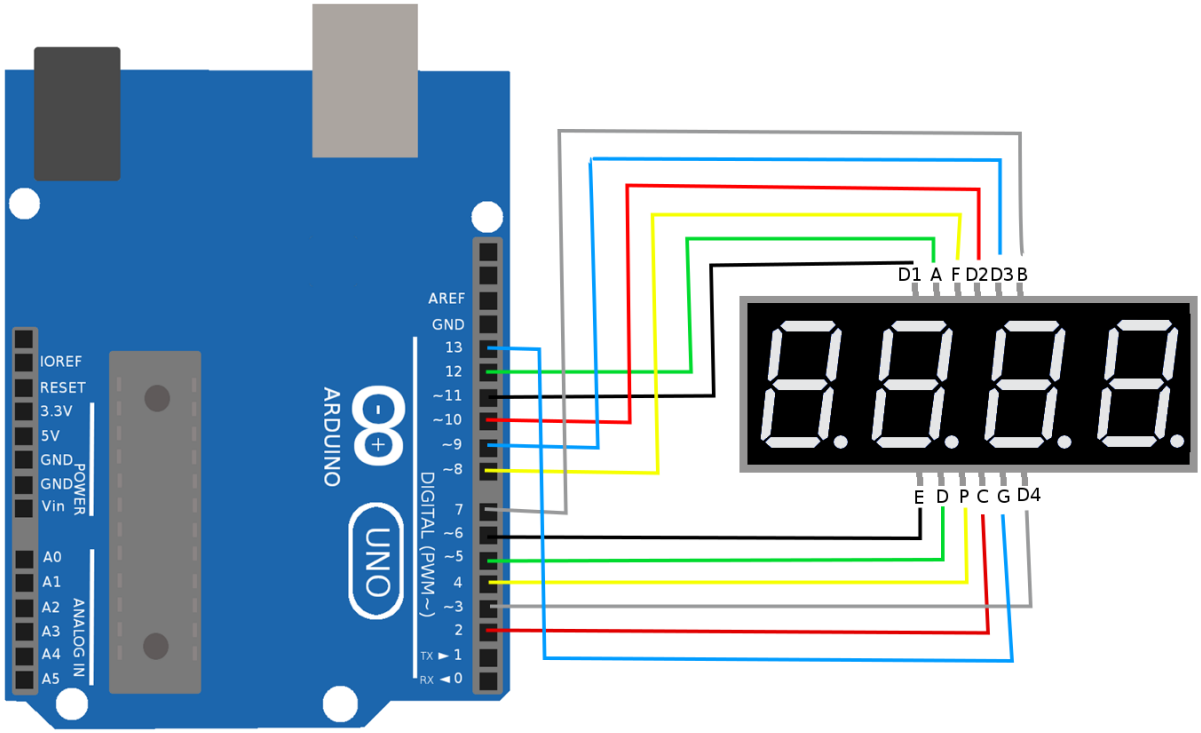
7 segment Display

You've likely seen many examples of 7 seg displays in clocks and microwaves and such; though they are starting to go out of style. The 7 segment display is our token example of a parrallel device. Like the LED, it is techanically a digital device; but decuase it only includes diodes, its one of the simplest digital devices possible. Since each LED has its own independent wire connecting it to the processor, we consider this a parallel device.

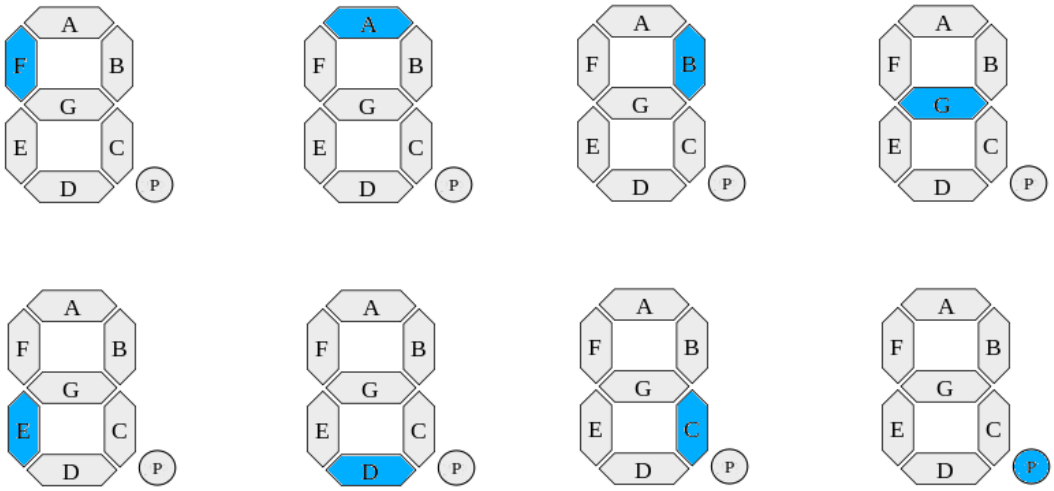
Wiring

Start by wiring the 7 segment display as shown in the image below. We're making a big jump here going from devices that have 3 pins to wire to this device which has 12. Don't panic, just take it one wire at a time, and take comfort in the fact that this is the most complicated wiring we will implement in this course. The color coding on this device is just to make the wires easies to distinguish; they don't mean anything. Every pin on this device is a digital input pin; 4 of them control which of the 4 displays is active, and 8 of them control the segments.



Testing:

Open and load to the Arduino the SevenSegTest sketch. This sketch will confirm that your device is properly wired by cycling through each segment on each display in a specific order. You can see from the image below that the sequence draws a figure eight through the segments and then on to the decimal point. It will repeat this sequence for each of the 4 dislpays, moving from left to right.



If you see that any of the segments or display are not lighting, or are lighting out of order, then you likely need to recheck the wiring for those particular segments or displays.

Coding:

Now that we have our device wired properly, we are going to be writing some of the code that makes the display show us numbers. You can close the SevenSegTest sketch and instead open up the SevenSeg sketch. Upload and run the code and you should see that number 1234 is printed to the device.

Code Walkthrough:

```
//display selectors
int d1 = 11;
int d2 = 10;
int d3 = 9;
int d4 = 3;
int displays[] = {d1,d2,d3,d4};
```

Here we define four variables that hold the numbers for the pins that connect the displays to the Arduino. We also load those variables into a list, so that we can later loop over the list when we want to affect all the dislays.

```
//7 seg selectors
int A = 12;
int G = 13;
int D = 5;
int F = 8;
int E = 6;
int B = 7;
int C = 2;
int P = 4;
int segments[] = {A,G,D,F,E,B,C,P};
```

Here we define eight variables that hold the numbers for the pins that connect the segments to the Arduino. We also load those variables into a list, so that we can later loop over the list when we want to affect all the segments.

```
void setup() {
  Serial.begin(9600);
  while(!Serial){}
  Serial.println("Starting 7 Segment");
  pinMode(A1, OUTPUT);
  int lenDisp = sizeof(displays)/sizeof(int);
  for(int i = 0; i< lenDisp; i++){
    int pin = displays[i];
    pinMode(pin, OUTPUT);
    digitalWrite(pin, HIGH);
  }
  int lenSeg = sizeof(segments)/sizeof(int);
  for(int i=0; i< lenSeg; i++){
    int pin = segments[i];
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
  }

}
```

Here in the setup function we start by initiating the serial port and printing a message, you should be getting pretty familiar this pattern by now. Next we are going to initialize the pins for all the displays and all the segments by looping over both the lists we created to contain those values. We use a for loop to move through these lists. For each pin specified in the lists, we use the pinMode function to tell the Arduino to use the pin as a digital output pin. For each pin in the displays list, we initialize the pin as being high (5V). For each pin in the segments list, we initialize the pin as being low (0V).

```
void loop() {
    numberToDisplay(1,d4);
    numberToDisplay(2,d3);
    numberToDisplay(3,d2);
    numberToDisplay(4,d1);
}
```

In the main loop, we simply cycle through the 4 displays and write a number to each of them. We write "1" to the first display, "2" to the second, "3" to the third, and "4 for the fourth. You might notice that since this is the loop function, so this code is being called over and over again. This is required for this device, because the device does not have any kind of memory. We only show a number on a single display at a time, and we cycle through them fast enough to give the illusion that all of them are being shown at the same time.

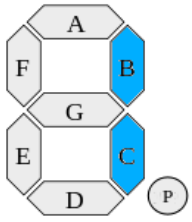
```
void numberToDisplay(int num, int disp)
{
  digitalWrite(disp, LOW);
  displayNumber(num);
  delay(1);
  clearSegments();
  digitalWrite(disp, HIGH);
}
```

This is the function that gets called from within the loop function that shows a single number on a single display. First, we drive the selected display low. Unintuitively, setting the display low is how we select it and turn it on. Next we call the function displayNumber to show the specified number. We use the delay function to wait 1 millisecond; while we wait the number is being displayed. Finally, we turn off all the segments and then we turn off the display by driving it high again.

```
void displayNumber(int i)
{
  switch(i){
    case 1: one(); break;
    case 2: two(); break;
    case 3: three(); break;
    case 4: four(); break;
  }
}
```

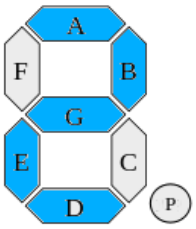
Here we have a function that uses a switch statement. This lets us handle some set of predefined cases for the value of the number we input into the function. As written, we can handle the cases where the input to this function is either 1,2,3 or 4. You'll extend this function to handle the numbers 5-9 and 0 later. For any of the predefined cases we give it, this function calls the appropriate lower level function.

```
void one(){
   digitalWrite(B, HIGH);
   digitalWrite(C, HIGH);
}
```



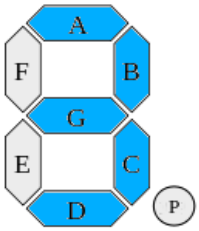
This function calls the appropriate segments (B and C) to display the character 1.

```
void two(){
   digitalWrite(A, HIGH);
   digitalWrite(B, HIGH);
   digitalWrite(G, HIGH);
   digitalWrite(E, HIGH);
   digitalWrite(D, HIGH);
}
```



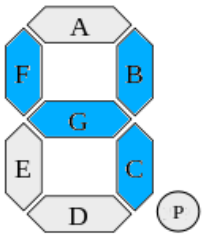
This function calls the appropriate segments (A,B,G,E,D) to display the character 2.

```
void three(){
   digitalWrite(A, HIGH);
   digitalWrite(B, HIGH);
   digitalWrite(G, HIGH);
   digitalWrite(C, HIGH);
   digitalWrite(D, HIGH);
}
```



This function calls the appropriate segments (A,B,G,C,D) to display the character 3.

```
void four(){
  digitalWrite(F, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(C, HIGH);
}
```



This function calls the appropriate segments (F,B,G,C) to display the character 4.

Extension:

Now that you've seen how the current code can display the numbers 1-4, you will write some additional

code that displays the numbers 5-8 and displays the number 5678 on the display.

1) Start by copying the pattern you were shown above and create the functions `five()` `six()` and `seven()` by selecting the pattern of segments and displaying them using the function `digitalWrite(segment, HIGH);`

2) Next, add a case into the swtich statement inside the display number function for the cases where i is equal to 5,6,7 or 8. This should look something like:

```
case #: number(); break;
```

3) Finally, change the numbers in the main loop function to show 5,6,7 and 8 instead of 1,2,3 and 4.

4) If you're feeling confident, go ahead and repeat this process one more time and add the last two characters we haven't written yet, 0 and 9.