Hands-on Arduino

March 24, 2012

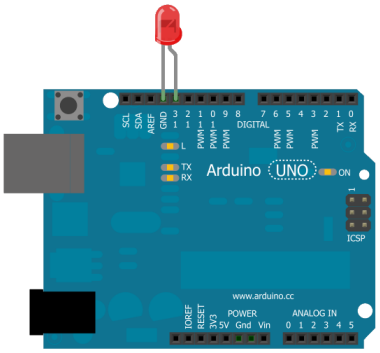
For this lab you will need a computer with Arduino Software installed. It can be downloaded from http://www.arduino.cc/. A set of Arduino Uno and USB cable will be provided per group.

For every section you will need to plug the Arduino to your computer, run the Arduino Software and load the code provided for each section. You can find it in your email inbox.

1 Traffic Lights

Take 5 LEDS: 1 yellow, 2 green and 2 red. Also five 220 Ω resistances.

We are going to start by placing one of the LEDs on the digital output 13. Plug the negative leg in the GND of the Arduino. You can follow the next illustration:

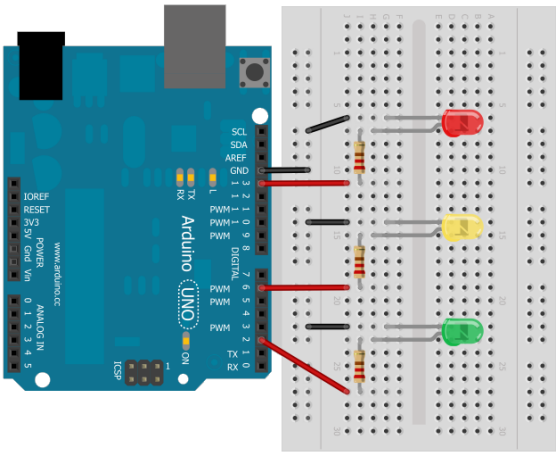


The next step is compiling the provided code. Upload your code clicking on the right arrow button. You will need your Arduino to be plugged to your computer and the serial port to be already set up.

You will be able to observe a LED turning on and off in an endless cycle. Try to understand the code because you will have to use it to complete the following tasks.

1.1 Build a car traffic light

You will need red, yellow and green LEDs. One of each color. Using the functions *digitalWrite(PIN,HIGH)* and *digitalWrite(PIN,LOW)* you will be able to turn on and off the LEDs attached to the corresponding pin numbers *PIN*. You will need to use 3 different pins, one per LED. Build an endless loop where the green light turns on for 4 seconds, the amber light for a second and the red light for 8 seconds.



1.2 Add a pedestrian traffic light

This will be done using an additional pair of LEDs, green and red. Follow the sketch showed and use the 220 Ω resistors to control the current in your LEDs. Be sure that the green light is on 4 seconds and then blinks 4 times, one per second before turning red. For this task you will need to build a function named *blink* with the following structure:

```
void blink(int PIN)
{
  digitalWrite(PIN, LOW);
  delay(500);
  digitalWrite(PIN, HIGH);
  delay(500);
...
}
```

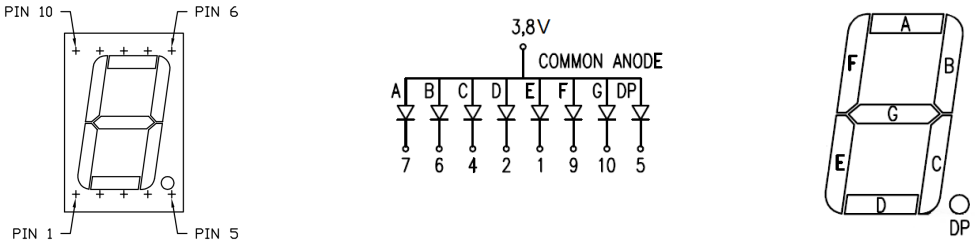
1.3 Synchronize the pedestrian and car traffic lights

Reusing the code you have written before, build a loop where the car traffic light turns green for 4 seconds right after the pedestrian red light turns on. Then the amber light will be on for a second, followed by the activation of the green pedestrian traffic light and the red car traffic light at the same time. Both will light for 8 seconds, but the last 4 seconds the pedestrian light will blink. This cycle should repeat over and over.

2 Counter

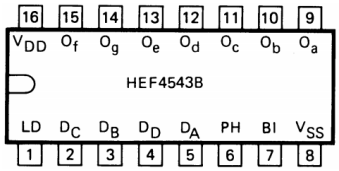
Counters are a very important part in any circuit. Any digital clock is just a counter, but you can also count people that enter in a store, cars crossing a road and in general any kind of stuff that is detectable.

In this section we are going to build a simple counter. It will count up from 0 to 9 and it will be displayed using a 7 segment LED. The code for this section will be provided since the connections and code to use a 7 segment LED are not trivial. To simplify them we are going to use a BCD to 7-segment decoder, namely HEF4543BP. From the 7 segment data sheets we get:



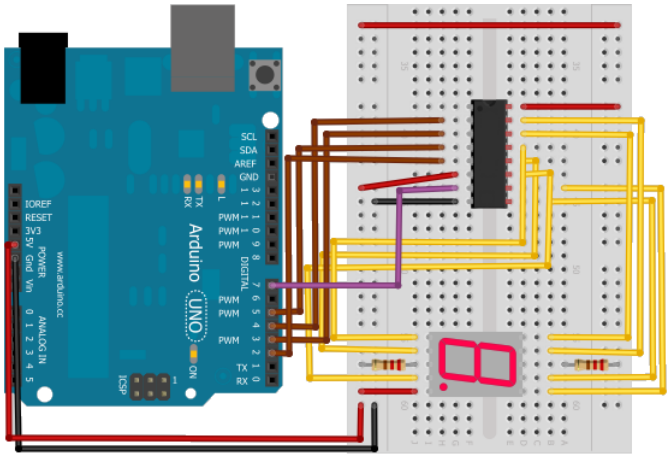
7 segment display pins

INPUTS							OUTPUTS							
LD	BI	PH (4)	D_D	D_C	D_B	D_A	O_a	O_b	O_c	O_d	O_e	O_f	O_g	DISPLAY
X	H	L	X	X	X	X	L	L	L	L	L	L	L	blank
H	L	L	L	L	L	L	H	H	H	H	H	H	L	0
H	L	L	L	L	L	H	L	H	H	L	L	L	L	1
H	L	L	L	L	H	L	H	H	L	H	H	L	H	2
H	L	L	L	L	H	H	H	H	H	H	L	L	H	3
H	L	L	L	H	L	L	L	H	H	L	L	H	H	4
H	L	L	L	H	L	H	H	L	H	H	L	H	H	5
H	L	L	L	H	H	L	H	L	H	H	H	H	H	6
H	L	L	L	H	H	H	H	H	H	L	L	L	L	7
H	L	L	H	L	L	L	H	H	H	H	H	H	H	8
H	L	L	H	L	L	H	H	H	H	H	L	H	H	9
H	L	L	H	L	H	L	L	L	L	L	L	L	L	blank
H	L	L	H	L	H	H	L	L	L	L	L	L	L	blank
H	L	L	H	H	L	L	L	L	L	L	L	L	L	blank
H	L	L	H	H	L	H	L	L	L	L	L	L	L	blank
H	L	L	H	H	H	L	L	L	L	L	L	L	L	blank
H	L	L	H	H	H	H	L	L	L	L	L	L	L	blank
L	L	L	X	X	X	X	(5)							(5)
as above		H	as above				inverse of above							as above



7 segment decoder pins

The circuit looks as follows:



2.1 Counting down and reset

To count down we are going to change the order of the loop. The integer *int* will be initialized to 10, the condition to keep the loop may be changed to greater than 0, and each step should decrease the integer by one unit.

Finally, the last change to our counter will be the addition of a reset button to stop the counter. We will attach the pin 6 of our Arduino to a switch with a pull-up resistor. Then, using the function *digitalRead(PIN)*, we will reset the counting when the button is pushed. To do that I have substituted the last line of the provided code with the next lines:

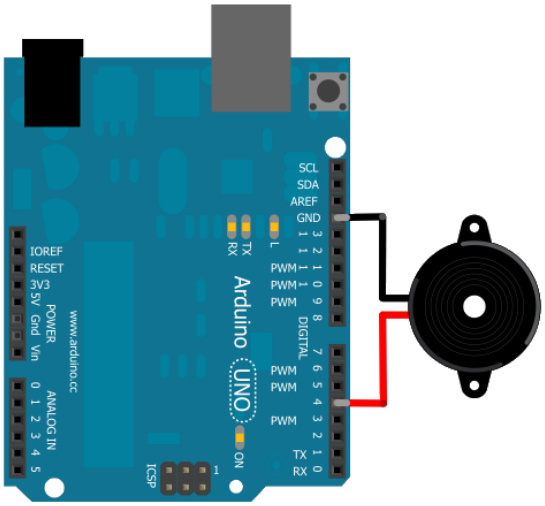
```
...
delay(1000);//This is the time that we are going to display our digit.

if(digitalRead(6)==LOW){
//stops counting and resets the counter if the button is pushed
digitalWrite(BL,HIGH);
break;
  }
```

What is wrong with this code? Try the push button with fast taps and you will see what I mean. To assure the correct functioning of the reset we will need to hold the botton steadily for some time, the time that Arduino spends running the loop *for* in which our code is embedded.

Do you have any idea on how to minimize this time? If so try to improve this code and check if it works, it will be helpful in your projects. If not, go to the next section.

3 Music note generator

In this section we will use a buzzer to play music notes. If you don't have a piezoelectric you can use the speaker inside your work station. The next sketch should be used:



3.1 One pitch, one octave

Prove the code Buzzer and the function *void buzz(int PIN, long frequency, long length)* to make a loop where you play an octave. An octave is defined as the interval between one musical pitch and another with half or double its frequency. As an example you can use the following frequencies: 996, 1050, 1180, 1250, 1320, 1400, 1490, 1580, 1670, 1770 and 1870 Hz. The length of the pitches should be 1000 ms.

3.2 Play a note

In order to have a faster way to choose our notes we are going to build a matrix to store all the frequencies. You can find it in the Buzzer_Play_Note class. Run this code and try out different "breaths" to understand their effect.

3.3 The Can-Can!

Now you are ready to play any song with your Arduino, you just need a music sheet and some patience writing down the frequencies into your code. To give some atmosphere to the lab you can run the French Can-Can. Just read the class *Buzzer_Playing_Can_Can* and try it out. Since Arduino has not the musical fame of Moulin Rouge you can try to improve the sound by changing octaves (add N multiples of 12 to the *noteInt* index in order to raise N octaves). Build a loop where the melody is played in different octaves. Let the musician you have inside compose the combination that sounds better.

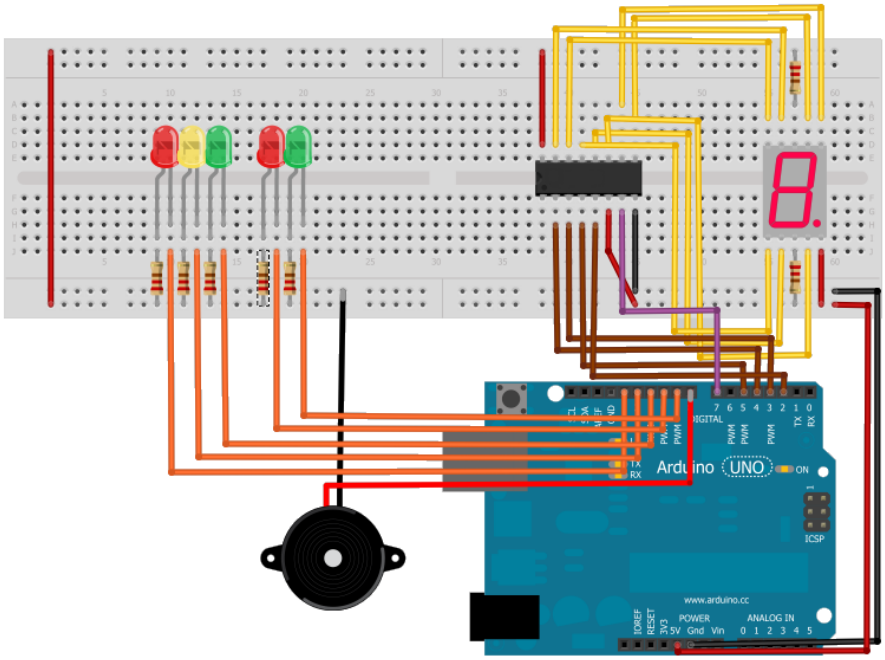
4 All together. Blind friendly traffic light

Summing up, we already know how to manage traffic lights, how to countdown and how to play music with an Arduino. Why don't we try to add all the features in one single device? Let's make a blind friendly traffic light. Although the Can Can would be a nice choice to make the pedestrians' day, it is more realistic to use a pair of high frequencies as the real traffic lights do. For instance, you can try these out:

```
buzz(buzzer, 440, 125);
buzz(buzzer, 400, 250);
```

Reusing all the code you have done till now make a traffic light that beeps and shows a countdown from 8 to 0 when the pedestrian traffic lights are green. You will need to turn off the counter display when the pedestrian traffic lights are red. Due to the concurrence of so many processes you will encounter some timing problems that you will need to solve. Pay special attention to control the beeps, LEDs and counter in a proper way.

The circuit will look somewhat like this:



Notice that we almost make use of all the digital pins in the Arduino, since the pin 0 and 1 are reserved for Tx and Rx purposes, only the pin 6 is being idle. What would have happened if we didn't use the BCD to 7 segment decoder? Could we have built the traffic light? Why yes or why not?

5 Thermostat

In some buildings you freeze in summer and melt down in winter. Probably they don't know how to use an Arduino as a Thermostat. We are going to learn how to do it:

5.1 Using a thermal sensor

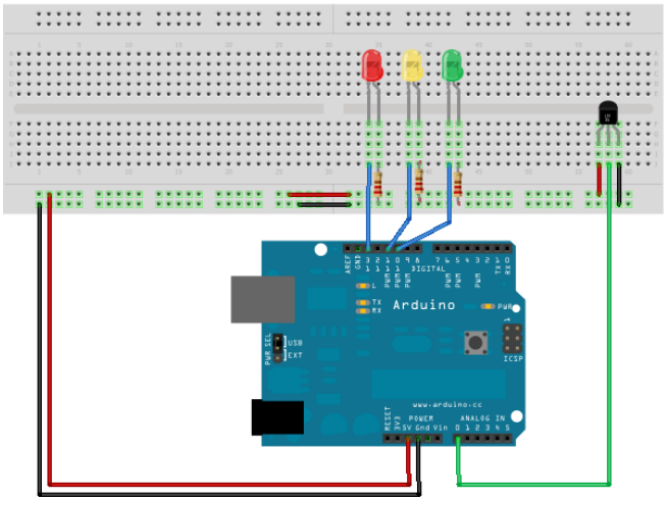
The LM35 is a "cheap" thermal sensor that will be used to measure the temperature of the room. We will need to plug it to the analog inputs of the Arduino. As a first approach we will use it to find out whether the room temperature is between a certain range of temperatures or not. Using the class *Thermometer* check the temperature of the room. If the green LED lights then your temperature will be under the range of temperatures, if it's the yellow the one that lights you will be inside the range, and finally if the red turns on you will be over the range of temperatures. Try out different values for your range and try to determine the temperature of the room.

In this section you will use for first time an analogic input to get the LM35 readout. The function *analogRead()* returns an integer between 0 and 1023, being the former 0 V and the latter 5 V. Therefore the resolution is 4.9 mV per unit. If we set *temp* to be the returned integer and *LM35* the analog input pin number the code should look as follows:

```
temp = analogRead(LM35);
```

To obtain the real temperature of the room we would need to use some kind of calibration that maybe it's not available in the lab. For example you could apply both, ice on the sensor to set the 32 Fahrenheit degrees and the temperature of your body that we know is somewhat around 96 Fahrenheit degrees. Knowing these two values you can measure accurately the temperature of the room. This is not required to complete the lab but it's something to take into account in a real project.

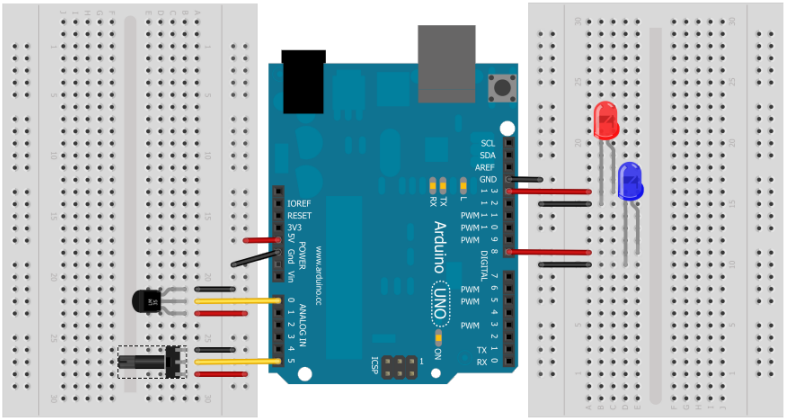
The circuit scheme is provided:



5.2 Build a thermostat

This manual process can be automatized by your Arduino, what is a much better idea. Furthermore, we can use this information to control a fan or a heater to make the room a cozier place. For the moment we are going to use two LEDs to simulate whether the fan or heater were on. Both could be off if the room reached the wanted temperature. To vary the measured temperature you can use your breath or any source of heat you have at **hand**. Should I have named this Hands-on LM35? To control the thermostat we are going to use a potentiometer, by varying it's current we will be able to set up different temperatures.

Follow the next scheme for your circuit:



You can also try to display the temperature in a pair of 7-segment displays. As we did with the counter we will use BCD to 7 segment decoders to minimize the number of digital pins busy. Divide the integer temp by 10 to obtain the first digit, and use the modulus of 10 to get the last one. In other words:

```
int firstDigit=int(temp)/10;
int lastDigit=int(temp) % 10;
```

Then using the next lines from the counter code you will be able to display the measured temperature:

```
num=firstDigit; //use lastDigit respectively.
for(int i=0; i<4; i++){
   if(int(num%2)!=0){
     num=int(num/2);
     values[i]=HIGH;
   }
   else{
     values[i]=LOW;
     num=int(num/2);
   }
}
```

```
digitalWrite(a, values[0]);//least significant value D3
digitalWrite(b, values[1]);//D2
digitalWrite(c, values[2]);//D1
digitalWrite(d, values[3]);//most significant value D0

delay(40);//displays the number for 40 ms
```

6 How to use Arduino in your projects

One last word. As you have already observed Arduino is a powerful tool with many applications. I will give you some ideas to improve some designs, but they are nothing but hints, you have to decide what to do trying to be original.

First of all, you can attach to your counter an infra-red sensor to detect objects. Also you can have a better counter using double digits so you can build a clock that displays minutes, hours and why not the temperature of the room. If you find out that you don't have enough pins to do it you should plan to use a shift register. We have 74HC595 shift registers available in the lab.

You can make a musical game, e.g., drum or piano hero, using an array of LEDs to display the notes falling in cascade and a set of buttons as your fake instrument. If a note reaches the last raw of LEDs and you hit the correct button at the same time the score will increase using a set of displays to show it, if the timing of your hit is wrong you shouldn't count it as a hit. This project would be really challenging and you can add many features to it. For example an interactive menu to choose different songs, the music sheets would be saved in the Arduino memory. You can use a red-yellow-green indicator to show how well you are doing (hit streaks can be used to determine this) and many other ideas. So many outputs may require the application of logic to multiplex the outputs of the Arduino. As I said, the application of all these ideas at the same time would require a thoughtful planning.

The thermostat can be used to control a real fan or the current in a resistor instead of the LEDs we have used in this lab. You can also set up programs to active your thermostat only on weekends or weekdays (Since we don't have so much time in the lab, it would be enough to turn it on every other minute).

There are many other applications and tutorials out there. Use google as a way to find new ideas. The limit is where your imagination decides it is.