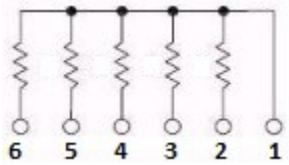
Putting It All Together - Controlling The Hardware With The Software
Programming Part 2

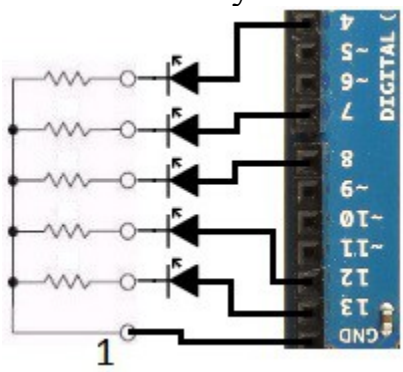
This following is what I got for Chase4. The pin variable declarations and setup() are the same as in Chase2 and Chase3. The main difference in those parts is that an LED has been added and connected to pin 4. It is now designated LED1 and the others follow:

```
// Chase4

int LED1_Pin_4 = 4;
int LED2_Pin_7 = 7;
int LED3_Pin_8 = 8;
int LED4_Pin_12 = 12;
int LED5_Pin_13 = 13;

void setup()
{
  pinMode(LED1_Pin_4, OUTPUT);
  pinMode(LED2_Pin_7, OUTPUT);
  pinMode(LED3_Pin_8, OUTPUT);
  pinMode(LED4_Pin_12, OUTPUT);
  pinMode(LED5_Pin_13, OUTPUT);
  digitalWrite(LED1_Pin_4, LOW);   // turn the LED off
  digitalWrite(LED2_Pin_7, LOW);   // turn the LED off
  digitalWrite(LED3_Pin_8, LOW);   // turn the LED off
  digitalWrite(LED4_Pin_12, LOW);   // turn the LED off
  digitalWrite(LED5_Pin_13, LOW);   // turn the LED off
}
```

There are now 5 LEDs rather than 4. At some point, individual current limiting resistors take up too much room. A resistor array is a good solution which typically looks like this, with the physical device on the left and the schematic representative on the right. The dot or some other mark indicates pin 1, which is common to one side of all of the resistors in the array:



Connect the LEDs to the UNO and the array as follows:



It's also just fine to use individual resistors rather than an array. The rest of Chase4 looks like the following. Notice that TotalMilliseconds is declared as a constant integer. This is necessary since a variable cannot be used in a case statement; only constants. In this case, it is set to 300 when it is declared. It cannot be changed by the program, but only set at declaration.

A for loop starts the counter, x, at 0 and runs as long as it is less then TotalMilliseconds. The first LED is turned on and the last turned off at 0. Then, since there are 5 LEDs, subsequent LEDs are turned on and the previous one turned of in 1/5 steps of the amount set by TotalMilliseconds. Thus, the on and off times are set by TotalMilliseconds, and thus the speed of the chase:

```
int x;  // counter
const int TotalMilliseconds = 300; // total time used in for loop at 1ms per step

void loop()
{
	for(x=0; x<TotalMilliseconds; x++)
	{
		switch(x)
		{
			case 0:
				digitalWrite(LED1_Pin_4, HIGH);   // turn LED1 on at 0 ms
				digitalWrite(LED5_Pin_13, LOW);   // and LED5 off
			break;

			case TotalMilliseconds / 5: // 1/5 -- TotalMilliseconds = 300 yields 60
				digitalWrite(LED2_Pin_7, HIGH);   // turn LED2 on
				digitalWrite(LED1_Pin_4, LOW);   // and LED1 off
			break;

			case (TotalMilliseconds * 2) / 5: // 2/5 -- TotalMilliseconds = 300 yields 120
				digitalWrite(LED3_Pin_8, HIGH);   // turn LED3 on
				digitalWrite(LED2_Pin_7, LOW);   // and LED2 off
			break;

			case (TotalMilliseconds * 3) / 5: // 3/5 -- TotalMilliseconds = 300 yields 180
				digitalWrite(LED4_Pin_12, HIGH);   // turn LED4 on
				digitalWrite(LED3_Pin_8, LOW);   // and LED3 off
			break;

			case (TotalMilliseconds * 4) / 5: // 4/5 -- TotalMilliseconds = 300 yields 240
				digitalWrite(LED5_Pin_13, HIGH);   // turn LED5 on
				digitalWrite(LED4_Pin_12, LOW);   // and LED4 off
			break;

		} // end switch(x)

		delay(1); // wait 1ms

	} // end for(x=0; x<TotalMilliseconds; x++)

} // end void loop()

// end Chase4
```

One problem with Chase4 is that there are more lines of code than might be wanted or needed. Please note that that is not always a problem, since code that can do without for loops and switch statements is often faster, which can be a benefit in many applications.

In addition, please note that Chase1, modified for 5 LEDs, which does not have a for loop or switch statement, ended up being 1248 bytes, whereas Chase4 with a for loop and switch was 1316 bytes. Small control computers are often memory starved, so the less room a program takes, the better.

Still another consideration is readability. It is very important that code does not get cold. Most programmers have looked back at code they wrote several years ago and wondered why in the world they did it that way. They might also be followed by other programmers. Those are also good reasons for commenting code. Generally speaking, the shortest, simplest code is often easiest to understand.

We will next look at a way to shorten and simplify Chase.

To make the code more compact we will use an array of integers defined and initialized as follows, found in Chase5:

```
const int LEDs[] = {4,7,8,12,13};
```

The five cells of the array are referenced by a number or integer variable inside the brackets ([]). The first cell is in position 0 and the 5th one in position 4. The numbers in the above initialization represent the pin numbers. So,
LEDs[0] = Pin 4, LEDs[1] = Pin 7, LEDs[2] = Pin 8,

LEDs[3] = Pin 12 and LEDs[4] = Pin 13

The number of cells can be determined by getting the size of the whole array, then divided that by the size of an individual integer by using the built-in sizeof() function. Note that the loop stops before reaching ArrayCount. The array's range is from 0 through 4, not through 5.:

```
int ArrayCount = sizeof(LEDs) / sizeof(int);
int x; // counter
```

A loop can be used in setup to initialize the pins:

```
void setup()
{
   // make pins outputs and set low
   for(x=0; x<ArrayCount; x++)
   {
     pinMode(LEDs[x], OUTPUT);
     digitalWrite(LEDs[x], LOW);
   }
  //Serial.begin(9600);
}
```

We are going to do the same thing that has been done in other versions of Chase, which is to turn on an LED then turn off the last one. The first one turned on will be the one for cell 0, and the first one turned off will be the last one. After that, the last one turned on will be turned off for each iteration of the for loop.

This is what the rest of Chase5 looks like:

```
int DelayTime = 60, LastOn = ArrayCount - 1;
void loop()
{
        for(x=0; x<ArrayCount; x++)
        {
                digitalWrite(LEDs[x], HIGH); // turn selected LED on =  0, 1, 2, 3, 4
                digitalWrite(LEDs[LastOn], LOW);  // turn selected LED off = 4, 0, 1, 2, 3

                LastOn = x;

                delay(DelayTime); // wait DelayTime milliseconds

        } // end for(x=0; x<ArrayCount; x++)

} // end void loop()

// end Chase5
```

Speed is set by the DelayTime variable at 60ms per iteration of the loop. LastOn is initialized to the last LED, which is not ArrayCount, but one less. After that, it's set to the current loop count, x.

Chase5 starts at 0 and goes through ArrayCount less 1, then starts over. Build Chase6, which will run one direction then back, then repeat the process. My version is Chase6.c

Experiment with the delay time. Note that virtually no movement is perceived with fast rates or if delay(DelayTime); is remarked out (// delay(DelayTime);). The LEDs appear to be constantly on, albeit at somewhat less than full brightness. All of Chase5 is on the next page.

```
// Chase5

/*
 LEDs[0] = Pin 4
 LEDs[1] = Pin 7
 LEDs[2] = Pin 8
 LEDs[3] = Pin 12
 LEDs[4] = Pin 13
*

//          positions = 0,  1,2,  3,  4
const int LEDs[] = {4, 7, 8, 12, 13};
int ArrayCount = sizeof(LEDs) / sizeof(int); // number of pins = size of whole array divided by size
of integer
int x;

void setup()
{
   // make pins outputs and set low
   for(x=0; x<ArrayCount; x++)
   {
    pinMode(LEDs[x], OUTPUT);
    digitalWrite(LEDs[x], LOW);
   }
  //Serial.begin(9600);
}

int DelayTime = 60, LastOn = ArrayCount - 1;

void loop()
{
        for(x=0; x<ArrayCount; x++)
        {
                digitalWrite(LEDs[x], HIGH); // turn selected LED on =  0, 1, 2, 3, 4
                digitalWrite(LEDs[LastOn], LOW);  // turn selected LED off = 4, 0, 1, 2, 3

                LastOn = x;

                delay(DelayTime); // wait DelayTime milliseconds

        } // end for(x=0; x<ArrayCount; x++)

} // end void loop()

// end Chase5
```

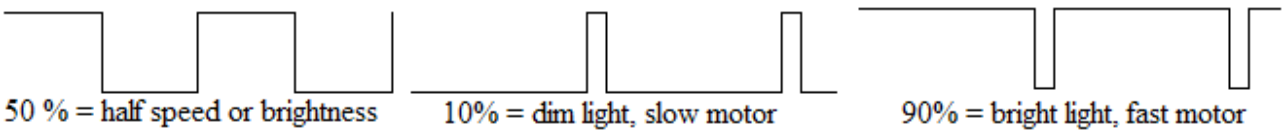
[Download Chase5.c Here]

8

Thus far, we have covered many aspects of programming, including the basic way an Arduino sketch is constructed, flowcharts, making comments in code so as to keep track of what's going on, designing and calling subroutines, sending test information to the host computer via USB, for loops and switch statements, connecting simple devices to an UNO (an LED), arrays and constants.

We will next explore fading LEDs rather than just turning them on and off. Move the LED currently connected to pin 8 to pin 9, then load and compile the Fade.c sketch.

Notice that pin 9 is designated 9~ on the UNO board. That indicates that it is an analog output. It uses Pulse Width Modulation (PWM) to produce an analog output. In fact, some very high quality audio amplifiers use PWM. PWM is generally a continuous train of pulses. The brightness of a lamp or speed of a motor, for example, is determined by the width of the on pulse. On time is called the duty cycle, expressed as a percentage of a full cycle:



Note that, in the case of LEDs, the change is only apparent. The instantaneous output is the same at low duty cycles, but the brain interprets the brightness as being lower.

Table Of Contents

Previous: Boolean Logic - AND, OR, XOR, NAND, NOR, NXOR

Next: Programming Part 3