



# Project 5 - LED Chase Effect

We are now going to use a string of LED's (10 in total) to make an LED chase effect, similar to that used on the car KITT in the Knightrider TV Series and on the way introduce the concept of arrays.

## What you will need

10 x Red Diffused LED's	
10 x 220Ω Resistors	



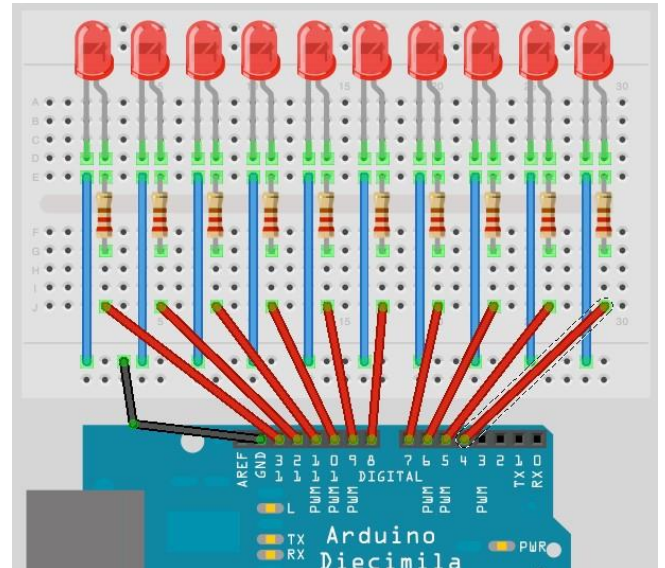
## Enter the code

```
// Project 5 - LED Chase Effect
// Create array for LED pins
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
int ledDelay(65); // delay between changes
int direction = 1;
int currentLED = 0;
unsigned long changeTime;

void setup(){
  // set all pins to output
  for (int x=0; x<10; x++) {
    pinMode(ledPin[x],OUTPUT);
    changeTime = millis();
  }
}

void loop(){
  // if it has been ledDelay ms since last change
  if (millis() - changeTime > ledDelay) {
    changeLED();
    changeTime = millis();
  }
}

void changeLED() {
  // turn off all LED's
  for (int x=0; x<10; x++) {
    digitalWrite(ledPin[x],LOW);
  }
  // turn on the current LED
  digitalWrite(ledPin[currentLED],HIGH);
  // increment by the direction value
  currentLED += direction;
  // change direction if we reach the end
  if (currentLED == 9) {direction = -1;}
  if (currentLED == 0) {direction = 1;}
}
```



# Project 5 - Code Overview

Our very first line in this sketch is

```
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
```

and this is a declaration of a variable of data type array. An array is a collection of variables that are accessed using an index number. In our sketch we have declared an array of data type byte and called it ledPin. We have then initialised the array with 10 values, which are the digital pins 4 through to 13. To access an element of the array we simply refer to the index number of that element. Arrays are zero indexed, which simply means that the first index starts at zero and not 1. So in our 10 element array the index numbers are 0 to 9.

In this case, element 3 (ledPin[2]) has the value of 6 and element 7 (ledPin[6]) has a value of 10.

You have to tell the size of the array if you do not initialise it with data first. In our sketch we did not explicitly choose a size as the compiler is able to count the values we have assigned to the array to work out that the size is 10 elements. If we had declared the array but not initialised it with values at the same time, we would need to declare a size, for example we could have done this:

```
byte ledPin[10];
```

and then loaded data into the elements later on. To retrieve a value from the array we would do something like this:

```
x = ledpin[5];
```

In this example x would now hold a value of 8. To get back to your program, we have started off by declaring and initialising an array and have stored 10 values that are the digital pins used for the outputs to our 10 LED's.

In our main loop we check that at least ledDelay milliseconds have passed since the last change of LED's and if so it passes control to our function. The reason we are only going to pass control to the changeLED() function in this way, rather than using delay() commands, is to allow other code if needed to run in the main program loop (as long as that code takes less than ledDelay to run).

The function we created is

```
void changeLED() { // turn off all LED's    for (int x=0; x<10; x++)
{      digitalWrite(ledPin[x], LOW);
}
// turn on the current LED    digitalWrite(ledPin[currentLED], HIGH); // increment
by the direction value    currentLED += direction;
// change direction if we reach the end    if (currentLED == 9) {direction = -1;}
if (currentLED == 0) {direction = 1;}
}
```

and the job of this function is to turn all LED's off and then turn on the current LED (this is done so fast you will not see it happening), which is stored in the variable currentLED.

This variable then has direction added to it. As direction can only be either a 1 or a -1 then the number will either increase (+1) or decrease by one (currentLED +(-1)).

We then have an if statement to see if we have reached the end of the row of LED's and if so we then reverse the direction variable.

By changing the value of `ledDelay` you can make the LED ping back and forth at different speeds. Try different values to see what happens.

However, you have to stop the program and manually change the value of `ledDelay` then upload the amended code to see any changes. Wouldn't it be nice to be able to adjust the speed whilst the program is running? Yes it would, so let's do exactly that in the next project by introducing a way to interact with the program and adjust the speed using a potentiometer.