

Introduction to Arduino Programs :

Sketches



Objectives

- ▣ Understand the basic structure of an Arduino sketch
- ▣ Identify each part of the structure
- ▣ Understand the importance and significance of well documented code using comments
- ▣ Understand variable types and how to use them
- ▣ Understand the importance of “grammar and punctuation” in a C program
- ▣ Understand how digital outputs are implemented



Arduino programs: “Sketches”

- ▣ A “Sketch” is a single program that runs on an Arduino device.

- ▣ There are four basic parts of a sketch:
 1. Header – definition of what the program does, who wrote it, and the date.
 2. Variable definitions
 3. setup function – defines how I/O pins will be used
 4. loop function – main code that runs indefinitely



Header

- ▣ The header is a group of *comments* that provides important information about the program including:

- Title of program
- Purpose of program
- Author of program
- Date program was created

Most important

- Revisions
- Other files the program uses
- I/O connections
- TO DO list

Very helpful



Simple Header example

```
/*
```

```
Name of program: Blink#2
```

```
Purpose of program: Flashes LED on  
board using a constant and variable.
```

```
Created by: Tom Thoen
```

```
Date: 1/20/2017
```

```
*/
```

This is the MINIMUM information required for any program you are writing in class!!



Comments are created two ways

example #1 : block

```
/* ← (This is the beginning of the comment block)
```

```
Name of program: Blink#2
```

```
Purpose of program: Flashes on board LED  
using a constant and variable.
```

```
Created by: Tom Thoen
```

```
Date: 1/20/2017
```

```
*/ ← (This is the end of the comment block)
```

Note - the comments do not “do” anything in the program - they are there just to make the code more readable and document what's going on.



Comments

Example #2 - single line:

```
blinkNum = 20; // Number of LED flashes when "stop" pressed
```

Usually used on the same line as the code

Not very exciting or fun – but INCREDIBLY important! Helps the reader (and you) understand what the code means!

And no, you don't have to comment every line of code – but it helps to keep track of what the code does.



Parts of an Arduino program: Variable definitions

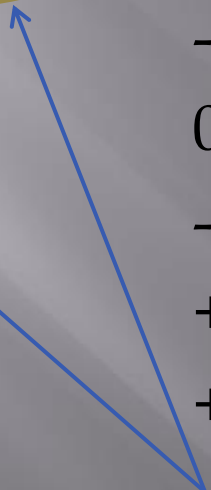
- ❑ A *variable* is simply a small amount of memory used to store a numeric value or character while the program is running.
- ❑ It can be a single bit, a signed value, a byte, or a floating point value.
- ❑ Before using a variable, we need to define its *name* and *type*.



Variable definitions - Examples of different types

<u>Type:</u>	<u>Value range</u>	<u>Bytes Used</u>
<u>boolean</u>	1 or 0 ("true" or "false")	1
char	-128 to +128	1
byte	0 to 255 (no negative values)	1
<u>int</u>	-32,768 to 32,767	2
long	+/- 2,147,483,647	4
float	+/- 3.xxxxxxxx E ³⁸	4

most often used



How are variables defined in a sketch??

`boolean flashMode = 0;` declare variable and initialize to zero
(type) (name) (value)

`int longDelay = 2000;` declare variable and initialize to 2000
`byte redLED = 13;` Create a variable name (for an LED on pin 13)
`int delayTime = 500;` Create a variable for the flash time *

* Note – what would happen if we used byte instead of int?

Although it is not necessary to initialize a variable to a value, it's generally a good idea as it is not guaranteed to be equal to zero at the beginning!

Parts of a sketch: the *setup* function

The *setup* function initializes the Input and Output pins, and serial (communication) functions.

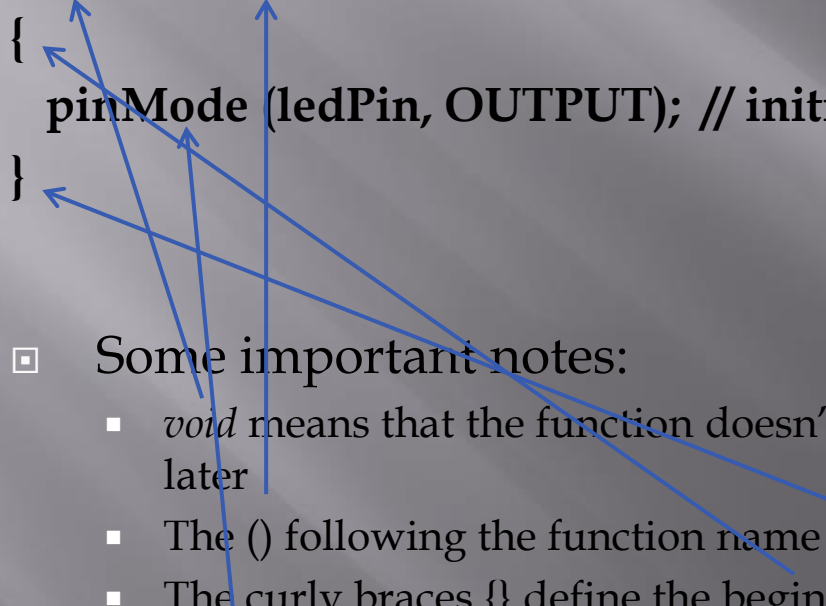
The *setup* function only runs one time at the beginning of the program.

So, what is a *function* anyway? We'll be discussing these in a lot of detail; for now think of them as little sub-programs.

Setup example:

```
Int ledPin = 13;           // define ledPin = 13
```

```
void setup()  
{  
  pinMode (ledPin, OUTPUT); // initialize digital pin 13 as an output.  
}
```



□ Some important notes:

- *void* means that the function doesn't return a value – we'll talk about this more later
- The () following the function name *setup* shows that it is a function.
- The curly braces {} define the beginning and end of the function
- *pinMode* is a pre-defined command in the Arduino programming language. It is used to define whether a pin on the I/O header is used for input or output.

And last but not least... loop

The **loop** function is the main part of the program that runs continuously once **setup** has completed.

```
void loop()  
{  
    // The main program goes here...  
    // and loops forever until you reset or  
    // turn off the power...  
}
```

Digital Outputs

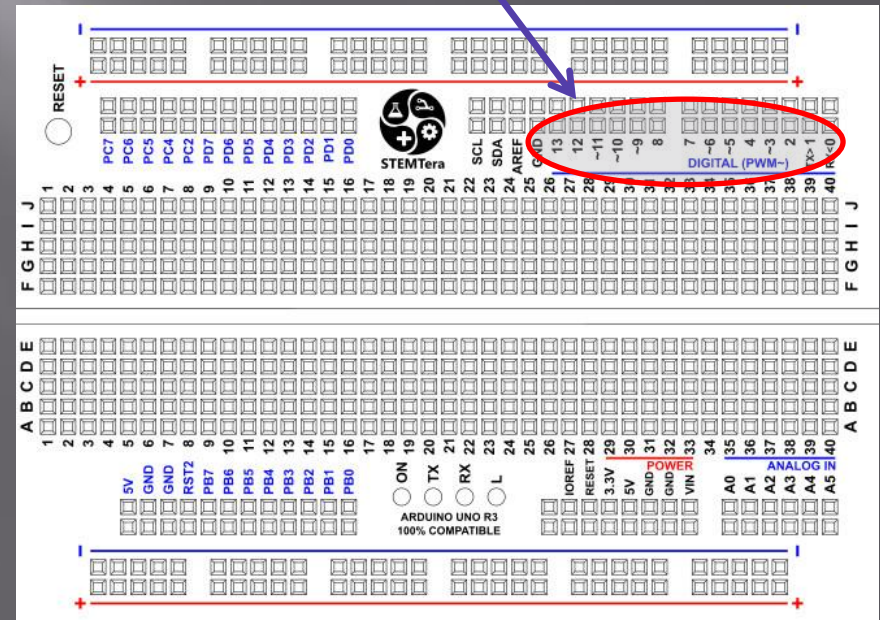
In order to turn a digital output on or off, we need to do two things:

1. Define which pin we will turn on or off
2. Set it to a high voltage (5 Volts) or a low voltage (0 volts)

```
digitalWrite(13, HIGH);
```

Pin number
(this is the LED on the board)

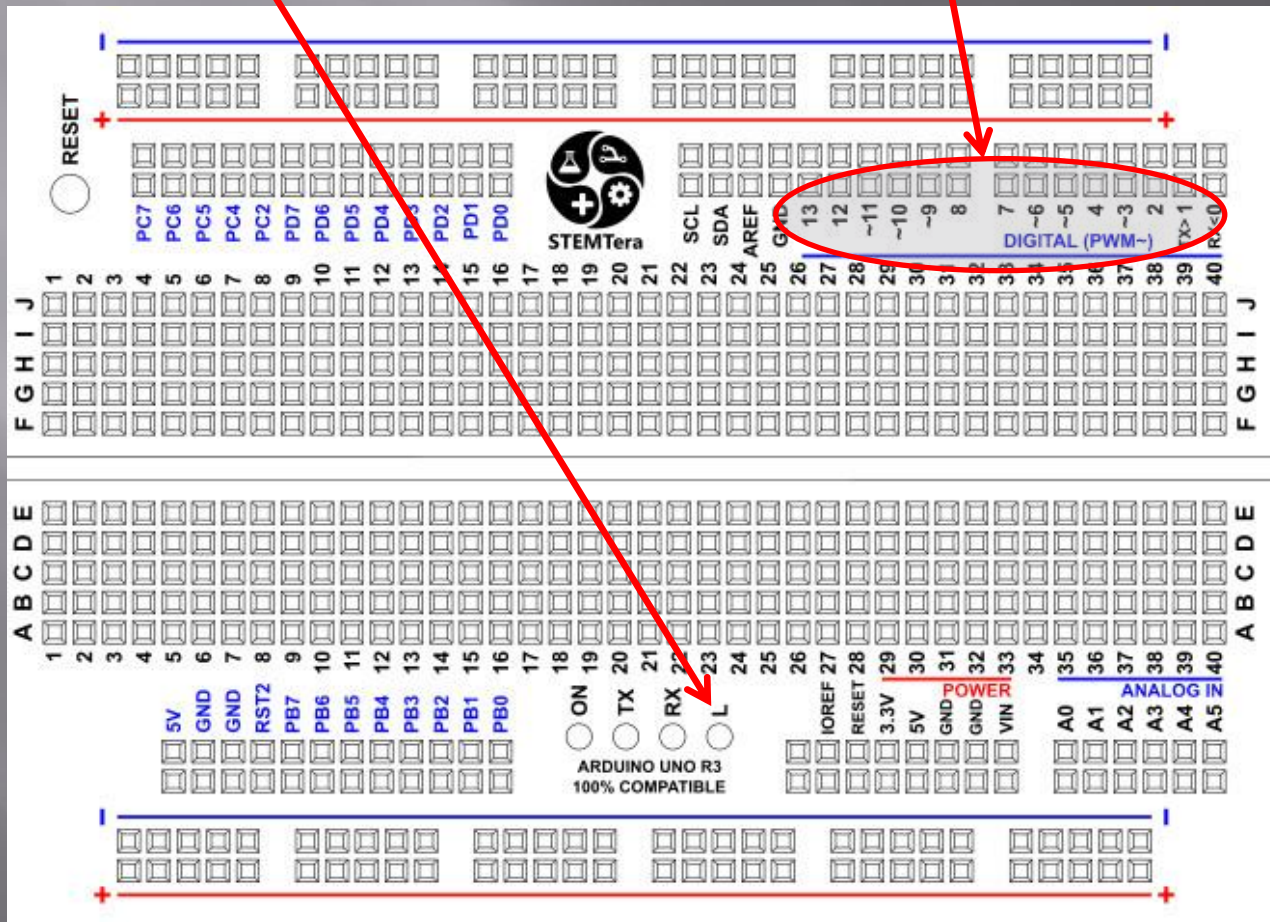
Output voltage
(HIGH = 5V, LOW = 0V)



Digital Outputs

Pin number 13 is also connected to the LED on the board

Usually outputs 2 - 13 are used on the UNO



Digital Outputs

Instead of using a number it is better to name the pin as a *constant** at the beginning of the program as follows:

```
const int redLED = 4;           // Red LED on pin 4
const int greenLED = 5;        // Green LED on pin 5
```

In the program, we can now use the names instead of the pin numbers:

```
digitalWrite(redLED, HIGH);    // Turn on red LED
digitalWrite(greenLED, LOW);   // Turn off Green LED
```

* *A constant is a special type of variable that can't change in the program*

The Delay function

A very simple function used in programs is the *delay* function.

- *delay* essentially stops the program for a fixed time
- *delay* is an example of *built-in functions* in the Arduino library

```
delay(1000);    // delay for one second
```



Note the parentheses following the function name

The value 1000 is called an *argument*, and it is *passed* to the function.

The units of the delay value are milliseconds.

Example Program

```
/*  
Name: Blink227  
Purpose of program: Flashes an LED at a rate of 1 Hz, repeatedly.  
Author: Jose Programmer  
Date: 2/27/2017  
*/
```

```
const int LED = 13;
```

```
void setup()  
{  
  pinMode(LED, OUTPUT); // initialize digital pin 13 as an output.  
}
```

```
void loop() // the loop function runs over and over again forever  
{  
  digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000); // wait for a second  
  digitalWrite(LED, LOW); // turn the LED off by making the voltage LOW  
  delay(1000); // wait for a second  
}
```

[yawn] - not that exciting, but it has all four of the elements described...can you spot them all??

Grammar and Punctuation

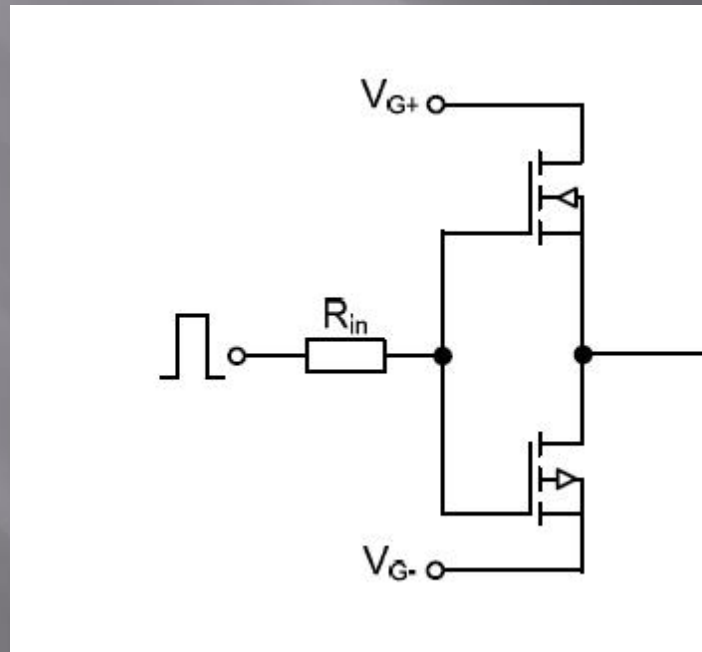
- ▣ The “C” programming language is a structured language, but has some rules:
 - White spaces are ignored
 - Variable definitions and functions are followed by a semicolon ; Think of it as the period at the end of a sentence.
 - Functions are followed by parenthesis () – if nothing is passed to the function the “insides” are empty
 - The “body” of a function is surrounded by curly braces {}
 - For readability, the body of a function is indented

Wiring outputs to the Arduino

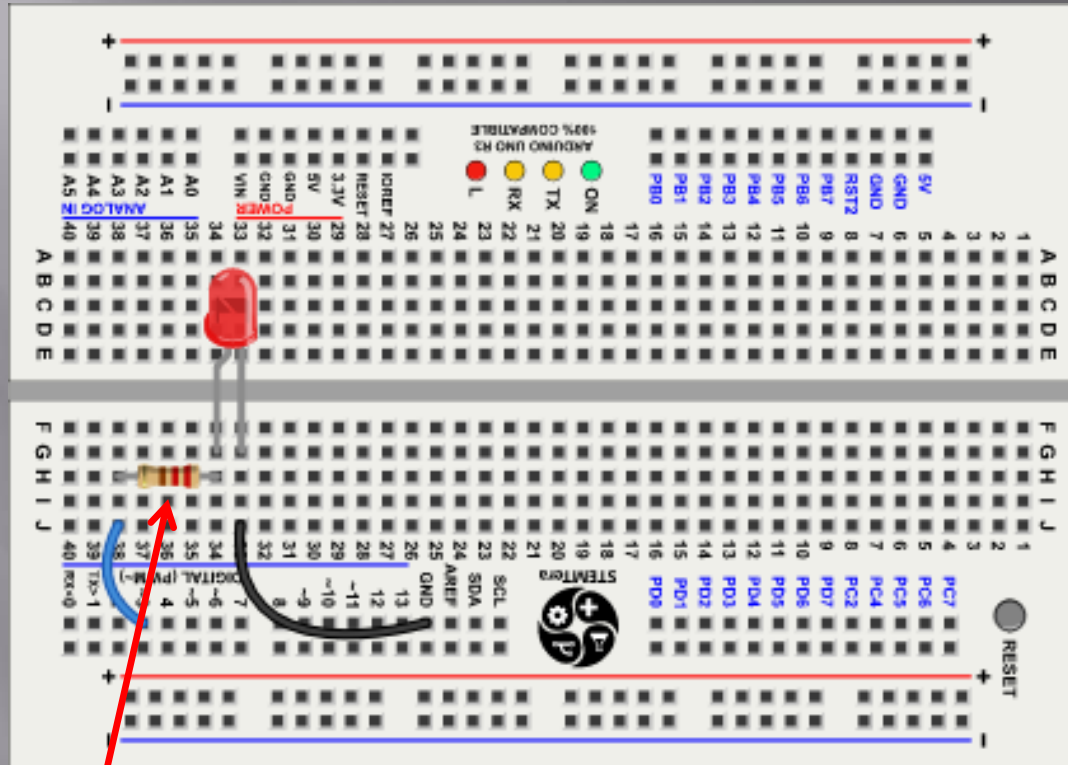
- ▣ The output voltage of each digital pin is 5 volts.
- ▣ The output current of the Arduino pins is fairly low – 20 milliAmps (mA) is a safe value to use.
- ▣ However, the Maximum current for groups of outputs is:
 - Sourcing: 300mA
 - Sinking: 400mA
- ▣ If we are just using LED's and limit the current to 10mA there shouldn't be a problem.
- ▣ However, watch for shorting outputs!

What do the output pins look like inside the Arduino?

VERY simplified view...



Wiring LED's



Typical resistor values are 330 – 470 Ohms

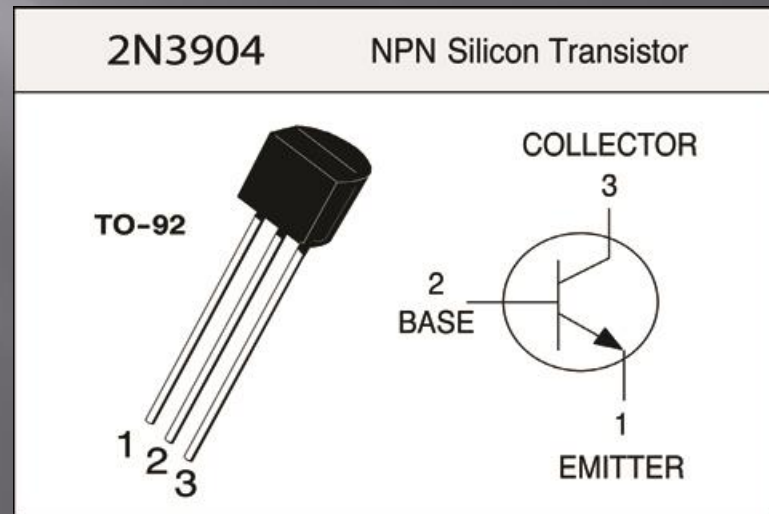
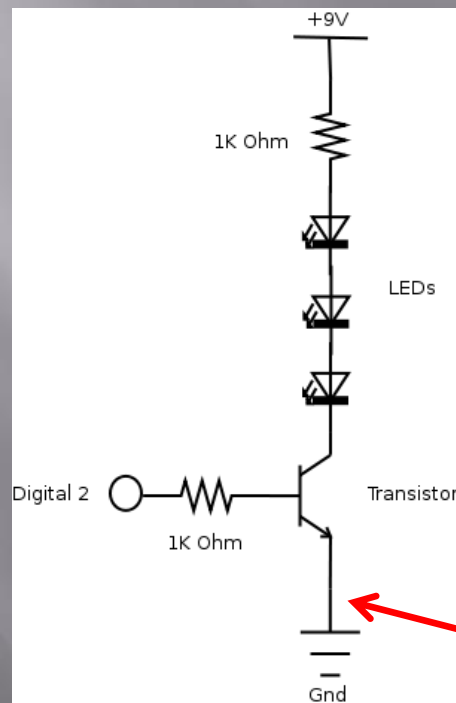
Each output requires a resistor if multiple LED's are on at the same time

NEVER connect an LED to an output without a resistor!!!
Why not??



What if I need to switch a device that requires more current?

- ▣ Use a transistor



Note: Ground **MUST** be also connected to the Arduino GND pin!!!