Disclaimer

- Many of these slides are mine
- But, some are stolen from various places on the web
- todbot.com – Bionic Arduino and Spooky Arduino class notes from Todd E. Kurt
- ladyada.net – Arduino tutorials by Limor Fried

Part 1 – Arduino SW

Procedure

Using Arduino

- Write your sketch
- Press Compile button (to check for errors)
- Press Upload button to program Arduino board with your sketch

Try it out with the “Blink” sketch!

Load “File/Sketchbook/Examples/Digital/Blink”
Blink Sketch (program)

/*
 * Blink
 * The basic Arduino example. Turns on an LED on for one second,
 * then off for one second, and so on... We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 */

int ledPin = 13; // LED connected to digital pin 13
void setup() {
  pinMode(ledPin, OUTPUT); // run once, when the sketch starts
  pinMode(12, INPUT); // sets the digital pin as output
}

void loop() {
  digitalWrite(ledPin, HIGH); // run over and over again
  delay(1000); // sets the LED on
  digitalWrite(13,LOW); // wait for a second
  delay(1000); // sets the LED off
  delay(1000); // wait for a second
}

Arduino Functions

- Each of the 14 digital pins is controlled by program statements
  - pins are numbered 13 to 0
  - pinMode(<pinNumber>, <INPUT/OUTPUT>)
  - Define whether the pin is used for input or output
    - e.g. pinMode(13, OUTPUT);
    - Pins are OUTPUT by default...
  - digitalWrite(<pinNumber>, <HIGH/LOW>)
  - Drive the output to a HIGH or LOW voltage (5v or 0v)
    - e.g. digitalWrite(13,HIGH);
  - digitalRead(<pinNumber>)
  - read a value on an input pin
    - e.g. digitalRead(8);

Almost all statements end with a semicolon!

Add Comments...

- One section for setting things up
  - pinMode(13, OUTPUT);
    - pinMode(12, INPUT);
  - One section repeats forever – lines of code execute
    - one at a time
      - digitalWrite(13,HIGH);
        - delay(1000);
      - digitalWrite(13,LOW);
        - delay(1000);
    - repeat forever...

Comments are just notes to the reader. They are NOT code
Variables

```c
int ledPin = 13; // LED connected to digital pin 13
```

- `ledPin` is a variable that holds a 16-bit value
- 16 binary digits is enough for -32768 to 32767
- Default starting value is defined to be 13
- There are other data types you can use

- Variables are placeholders for values
- Think of them as mailboxes
- You can store a value in them, and pick it up later
- Lets you refer to things by name, instead of just number

- Assigned with `=`
  - e.g. `ledPin = 12;` // This updates the value of `ledPin` to be 12

- Variables
  - Variable names must start with a letter or underscore
  - Case sensitive!
  - `Foo` and `foo` are different variables!
  - After the letter or underscore you can use numbers too
  - Are these valid names?
    - `Abc` ❌
    - `1st_variable` ❌
    - `_123_` ❌
    - `pinName` ✓
    - `another name` ✓
    - `a23-d` ❌
    - `aNiceVariableName` ✓

Use Variables

- One section for setting things up
  - int ledPin; // define an int variable
  - ledPin = 13; // set ledPin to 13
  - pinMode(ledPin, OUTPUT); // pin 13 is the output LED

- One section repeats forever – lines of code execute one at a time
  - digitalWrite(ledPin,HIGH); // Set 13 high (LED lit)
  - delay(1000); // wait for a second
  - digitalWrite(ledPin,LOW); // set 13 low (LED Off)
  - delay(1000); // wait for 1 sec

If you want to change pins, you only need to change one line of code!

Blink Sketch (program)

```c
int ledPin = 13; // LED connected to digital pin 13
void setup() {
  pinMode(ledPin, OUTPUT); // set ledPin to 13
  digitalWrite(ledPin, HIGH); // Set 13 high (LED lit)
  delay(1000); // delay for 1 sec
  digitalWrite(ledPin, LOW); // set 13 low (LED Off)
  delay(1000); // wait for 1 sec
}
void loop() {
  digitalWrite(ledPin, HIGH); // Set 13 high (LED lit)
  delay(1000); // delay for 1 sec
  digitalWrite(ledPin, LOW); // set 13 low (LED Off)
  delay(1000); // wait for 1 sec
}
```

Required Arduino Functions

```c
/* define global variables here */

void setup() {
  // run once, when the program starts
  int initialization statement; // typically pin definitions
  ...
  <initialization statement>
}

void loop() {
  // run over and over again
  <main loop statement>
  ...
  <main loop statement>
}
```

"void" means that those functions do not return any values

Arduino Language Recap

```c
int pinMode(pin, mode); // set pin direction
  - pin is a number, mode can be INPUT or OUTPUT
  - Used in the setup() function

digitalWrite(pin, value); // set pin value
  - Value can be HIGH (1) or LOW (0)

digitalRead(pin); // read value from pin
  - Returns an int – value either HIGH or LOW

delay(val); // pause the program for a bit
  - Pauses for val milliseconds (1/1000's of a sec)
  - 1000 msec = 1 sec
  - val can be up to "unsigned long max" (i.e. huge)
```
Data Types on Arduino

- By default, types are signed unless you say “unsigned”…

<table>
<thead>
<tr>
<th>Type</th>
<th>Size (bits)</th>
<th>Size (bytes)</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>1</td>
<td>1</td>
<td>0 (false)</td>
<td>1 (true)</td>
</tr>
<tr>
<td>unsigned byte</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>byte</td>
<td>8</td>
<td>1</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>unsigned int</td>
<td>16</td>
<td>2</td>
<td>0</td>
<td>65,535</td>
</tr>
<tr>
<td>int</td>
<td>16</td>
<td>2</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>unsigned long</td>
<td>32</td>
<td>4</td>
<td>0</td>
<td>4,294,967,295</td>
</tr>
<tr>
<td>long</td>
<td>32</td>
<td>4</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>float (double)</td>
<td>32</td>
<td>4</td>
<td>-3.4028235E+38</td>
<td>3.4028235E+38</td>
</tr>
</tbody>
</table>

Blink Modifications

- Change so that blink is on for 500msec and off for 100msec
  - What happens?
- Change so that blink is on for 50msec and off for 50msec
  - What happens?
- Change so that blink is on for 10ms and off for 10ms
  - What happens?

Blink Modifications

- Change to use an external LED rather than the one on the board
  - Connect to pin 13
  - LED is on if current flows from Anode to Cathode
  - LED is on if the digital pin is HIGH, off if LOW
  - How much current do you use?
    - not more than 20mA
  - How do you make sure you don’t use too much?
    - use a resistor
  - Pay attention to current! Use a current-limiting resistor!

LEDs and Resistors

On LEDs, polarity matters.
- Shorter lead is “negative” side, goes to ground.
- Long lead goes to positive source.
- Current flows from Anode to Cathode
- Lights up when current flows

LEDs and Resistors

On LEDs, polarity matters.
- Shorter lead is “negative” side, goes to ground.
- Long lead goes to positive source.
- Current flows from Anode to Cathode
- Lights up when current flows

Anode + Cathode -

Arduino Pin 13

Anode + Cathode -

Ground
Making Circuits

heart pumps, blood flows voltage pushes, current flows

Wiring it Up

Electricity flows in a loop. Can stop flow by breaking the loop

Wiring it Up

Arduino Duemilanove board has this circuit built-in
To turn on LED use digitalWrite(13, HIGH)

Proto Boards

numbers & letter labels just for reference
All connected, a “bus”
groups of 5 connected
not connected

Wire it Up

plugged into “ground” bus
We just made an LED blink

**Big Deal?**

- Most actuators are switched on and off with a digital output
  - The `digitalWrite(pin,value);` function is the software command that lets you control almost anything
- LEDs are easy!
  - Motors, servos, etc. are a little trickier, but not much
  - More on that later...
- Arduino has 14 digital pins (inpts or outputs)
  - Can easily add more with external helper chips
  - More on that later...

---

**Current Limiting Resistor**

- Every LED has a $V_f$ “Forward Voltage”
- How much voltage is dropped (used up) passing through the LED

\[
R = \frac{V - V_f}{I}
\]

- Example - If $V_f$ is 1.9v (red LED), and $V = 5v$, and you want 15mA of current (0.015A)
  - $R = (5 - 1.9)/0.015 = 3.1/0.015 = 206\,\Omega$
  - Exact isn’t critical – use next size up, i.e. 220\,\Omega
  - Or be safe and use 330\,\Omega or 470\,\Omega
  - This would result in 9.4mA or 6.6mA which is fine

---

**Resistor Color Codes**

- What’s the color code for a 220\,\Omega resistor?
- What’s the color code for a 1k\,\Omega resistor?
- What’s the color code for a 470\,\Omega resistor?

We’re using 4-band 5% resistors with a ¼ watt rating
Wire it Up
- Wire up an external LED of your choice, and change the Blink program to use that external LED.
- Choose your resistor based on the Vf of the LED you’re using.

Moving on…

Varying LED Brightness
Some circuit as Blink circuit but pin 9 instead of pin 13.

The PWM pins work with the "analogWrite(value)" command where value ranges between 0 and 255.
To turn LED to half-bright, use analogWrite(128, 255).

Pulse Width Modulation
- analogWrite(pin, value);
  - value can be 0 to 255
  - Must be one of the “PWM pins” - pins 3, 5, 6, 9, 10, 11
  - Don’t need to set pinMode to OUTPUT (but won’t hurt)

Load “File/Sketchbook/Examples/Analog/Fading”

C “for” loop
for (<initialization>; <condition>; <increment>) {
  // do something…
}

You can also define the variable right in the loop
for (int i = 0; i <= 255; i++) { // repeat 256 times
  analogWrite(pin, i); // write a value to the pin
  delay(50); // wait 50msec (0.05 sec)
} // The loop will take 50*256 msec to execute (12.8 sec)

Aside: C Compound Operators
x = x + 1; // adds one to the current value of x
x += 5; // same as x = x + 5
x++; // same as x = x + 1
x = x - 2; // subtracts 2 from the current value of x
x -= 3; // same as x = x - 3
x--; // same as x = x - 1
x = x * 3; // multiplies the current value of x by 3
x *= 5; // same as x = x * 5
Fading Program

int ledPin = 9; // LED connected to digital pin 9

void setup() {
  // nothing happens in setup (Why not?)
}

void loop() {
  // fade in from min to max in increments of 5 points:
  for (int fadeValue = 0; fadeValue <= 255; fadeValue += 5) {
    analogWrite(ledPin, fadeValue); // sets the value (range from 0 to 255):
    delay(30); // wait for 30 milliseconds between brightness steps
  }
  // fade out from max to min in increments of 5 points:
  for (int fadeValue = 255; fadeValue >= 0; fadeValue -= 5) {
    analogWrite(ledPin, fadeValue); // sets the value (range from 0 to 255):
    delay(30); // wait for 30 milliseconds between dimming steps
  }
}

Modified Fading

- What would you change to make things behave differently?
- Can you predict the effect of your changes?
- Loops are important – a general way to repeat things over and over
  - You don’t always have to repeat a fixed number of times
  - Assume that "foo" is a variable that you can set in your program
    - for (int i = 0; i < foo; i++) { } // loop "foo" times

Moving on…

- Write a program to make the LED flicker like a flame
- Choose a random intensity
- For a random amount of time
- Use `analogWrite(ledPin, val)` to change brightness
- Main loop repeats itself forever…
  - Set the value of the brightness to a random value
  - Wait for a random amount of time
  - repeat
- The effect looks like flickering...

Candle Program

- `random(min, max)` will return a random number between min and max.
- `randomSeed(int)` will initialize the random function
  - Not really needed...
  - `foo = random(10, 200);` // assign foo to a random number between 10-200
- Remember `delay(val);` // waits for "val" milliseconds

Hints…

- `int bright;` // make a new variable called bright
- `bright = random(100, 255);` // set "bright" to a random value
  - between 100 and 255
- Remember: `analogWrite(pin, value);` // sets a brightness on a pin
  - "pin" is the pin number, "value" is between 0 – 255

Silly LED Tricks
LED Wiring – 2 ways

Hints… Overall Algorithm

void setup() {  
    // set pin directions  
    // set global values if needed
}  

void loop() {  
    // set LED on/off values
    // delay for some amount of time...
    // set LED on/off values.
    // delay for some amount of time...
    // more LED values followed by more delays...
    // etc...
}  

// this code repeats when you get to the end

Hints… setup()

void setup() {
    pinMode(0,OUTPUT);
    pinMode(1,OUTPUT);
    pinMode(2,OUTPUT);
    pinMode(3,OUTPUT);
    pinMode(4,OUTPUT);
    pinMode(5,OUTPUT);
    pinMode(6,OUTPUT);
    pinMode(7,OUTPUT);
}

OR…

void setup() {
    for (int i=0; i<8; i++) {
        pinMode(i,OUTPUT);  
        // this loop will repeat 8 times
    }  
    // i will be 0, 1, 2, 3, 4, 5, 6, 7 on each iteration of the loop
}

OR…

void setup() {
    // do nothing (why?)
}

Hints… loop()

// loop is the function that repeats forever

void loop() {
    int delayTime = 100;  
    // a basic unit of delay (in msec)
    digitalWrite(0, HIGH);  
    // set LED 0 on
    delay(delayTime);  
    // wait delayTime milliseconds
    digitalWrite(0, LOW);  
    // set LED 0 off
    digitalWrite(1, HIGH);  
    // set LED 1 on
    delay(delayTime);  
    // wait delayTime milliseconds
    // more setting and delaying...
}  

Or use for (int i=0; i<foo; i++) or random(min,max), etc…

Everybody start coding!

Next Task: 8 LEDs

- connect LEDs (through resistors!) to 8 Arduino pins
- use pins 0, 1, 2, 3, 4, 5, 6, 7
  o Remember, pem on pins 3, 5, 6, 9, 10, 11 only...
- Now you can turn the LEDs on and off with
  digitalWrite(0, HIGH);  
  // turn LED 0 on
  digitalWrite(1, LOW);  
  // turn LED 1 off
  analogWrite(3, 180);  
  // turn LED 3 partly on
- Use those commands, also delay(), also perhaps loops, and
  random(min,max) to make the 8 LEDs do something!

- We’ll have demos in a few minutes…
**Blink Subtlety**

- When the delay(val); function runs, nothing else can happen
- Arduino just sits there counting milliseconds
- For blink this is just fine, but later you may want other things to be going on while the Arduino is counting
- Load BlinkWithoutDelay from the examples
- Let’s look at what it does…

**Comparison Operators**

- `x == y` (x is equal to y)
- `x != y` (x is not equal to y)
- `x < y` (x is less than y)
- `x > y` (x is greater than y)
- `x <= y` (x is less than or equal to y)
- `x >= y` (x is greater than or equal to y)

Beware of `x = y;` This does an assignment, not a comparison!

**BlinkWithoutDelay**

```c
const int ledPin = 13; // const says this won’t change
int ledState = LOW;
long previousMillis = 0;
long interval = 1000; // interval at which to blink the LED

void setup()
{
  pinMode(ledPin, OUTPUT); // set LED pin mode
}

void loop()
{
  if (millis() – previousMillis > interval) {
    previousMillis = millis(); // save the time you made the change
    if (ledState == LOW) {
      ledState = HIGH; }
    else {
      ledState = LOW; }
  digitalWrite(ledPin, ledState); // set the LED with ledState
  }
  // you can do other things here if it’s not time to change the LED state
}
```

**Summary – Whew!**

- **Digital Pins**
  - use pinMode(<pin>, <INPUT/OUTPUT>) for setting direction
  - Put these in the setup() function
  - pinMode(13, OUTPUT); // set pin 13 as an output

- use digitalWrite(<pin>, <HIGH/LOW>) for on/off
  - digitalWrite(13, HIGH); // turn on LED connected to pin 13

- use analogWrite(<pin>, <val>) for PWM dimming
  - values from 0 – 255
  - PWM pins are 3, 5, 6, 9, 10, 11
  - analogWrite(9, 235); // set LED on pin 9 to somewhat bright

**More Summary**

- `delay(val)` delays for val-number of milliseconds
  - milliseconds are thousands of a sec
    - (1000msec = 1sec)
  - `delay(500)`; // delay for half a second

- `random(min,max)` returns a random number between min and max
  - You get a new random number each time you call the function
  - `foo = random(10, 255);` // assign foo a random # from 10 to 255

- Two required Arduino functions
  - `void setup() { ... }` // executes once at start for setup
  - `void loop() { ... }` // loops forever
    - statements execute one after the other inside loop, then repeat after you run out

- `int i = 10;` // define an int variable, initial value 10

- Other types of variables:
  - char – 8 bits
  - long – 32 bits
  - unsigned…
  - float – 32 bit floating point number
Still More Summary

- `for (<start>; <stop>; <change>) { ... }`
  - `for (int i=0; i<8; i++) { ... }` // loop 8 times
    - the value of i in each iteration is 0, 1, 2, 3, 4, 5, 6, 7

- `if (<condition>) { ... }`
  - `if (foo < 10) {digitalWrite(ledPin, HIGH);}`

- `if (<condition>) { ... } else { ... }`
  - `if (num == 10) {<do something>} else {<do something else>} />

Last Summary (for now)

- LEDs – turn on when current flows from anode to cathode
- Always use a current-limiting resistor!
- Remember your resistor color codes
- 220 ohm is a good, general-purpose value for LEDs
- Drive from Arduino on digital pins
- Use PWM pins if you want to use analogWrite for dimming

![LED Diagram]