3.9 Sketches:

Start the Arduino IDE application. Open the Blink example sketch: Select **File → Examples → 01.Basics → Blink** on the top Arduino IDE menu.

Arduino Comments in Sketches

The Blink sketch contains a big comment block at the top. Scroll down to find the code that blinks the on-board LED. Comments are shown in a gray color between the /* and */ characters. Text typed between an opening forward slash and asterisk /* and closing asterisk forward slash */ is ignored by the software tools that run when the sketch is uploaded to an Arduino board. Place any comments or notes that you want between these characters. This type of comment can span multiple lines. The sketch name and description are contained in the top comment block of the Blink sketch. Detailed information about the on-board LED, and a history of the sketch follows.

A second type of comment starts with double forward slashes //. In this case, everything after the // is part of the comment. This is a single-line comment that turns everything after the // on the same line into a comment. Consequently the next line after the comment is not part of the comment.

Arduino Blink Sketch Code

The image below shows the Blink sketch code with the top comment block removed. As can be seen, it contains a mix of single-line comments and code. The code is everything in the image that is not a comment. Comments describe what is happening in the sketch.

When a sketch is loaded to an Arduino board, it is first built by software tools that automatically run. The build process consists of preprocess, compile and link stages. This converts the code from human readable text to something that runs on the Arduino board.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

Blink Sketch Code is one of the Arduino Sketches for Beginners

Arduino Sketch Syntax Highlighting

Colored words in the sketch are a result of the Arduino IDE using syntax highlighting. Syntax highlighting is the highlighting of Arduino language keywords, definitions and functions. This is more easily understood after more is learned about writing Arduino sketches.

Modify the Arduino Blink Sketch

Change the rate that the on-board LED blinks at in the Blink sketch, as follows. Change 1000 to **200** in both instances it is found in the code. The following image shows the modified Arduino Blink sketch. Red dots in the image mark the modified lines of code. Notice that the comments at the right of each modified line of code are updated to reflect the changes made in the code.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(200);                        // wait for 200 milliseconds (0.2 seconds)
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(200);                        // wait for 200 milliseconds (0.2 seconds)
}
```

Modified Blink Sketch Code

Click the Upload button in the Arduino IDE. This uploads the modified sketch to the target Arduino board. Notice that the on-board L LED blinks faster. As a result of the modified code, the LED is now on for 0.2 seconds and off for 0.2 seconds. The on and off times were originally 1000 milliseconds, also written 1000ms, which is one second (1s). This is because there are 1000ms in 1s. When 1000 is changed to 200 in the sketch, the LED on and off times change from 1000ms to 200ms. 200ms is 0.2s.

Save the Modified Sketch

Arduino example sketches are read-only. This means that they cannot be overwritten. When an example sketch is modified and saved, it must be saved to a new location. Three ways of saving a sketch in the Arduino IDE are firstly, click the Save toolbar icon (the arrow pointing down). Hover the mouse cursor over any toolbar icon and its name is shown at the right of the icons. Secondly, save the file using the keyboard shortcut **Ctrl + S** (hold down the Ctrl key and then press the s key). Thirdly, select **File → Save** on the top Arduino IDE menu bar. Because the file is read-only, the IDE prompts to save the file to a different location. Use the dialog box that opens to save the Blink sketch to your Arduino folder. Change the name to something like **Blink_Fast** befor

Start the Arduino IDE application. Open the Blink example sketch: Select **File → Examples → 01.Basics → Blink** on the top Arduino IDE menu.

Arduino Comments in Sketches

The Blink sketch contains a big comment block at the top. Scroll down to find the code that blinks the on-board LED. Comments are shown in a gray color between the /* and */ characters. Text typed between an opening forward slash and asterisk /* and closing asterisk forward slash */ is ignored by the software tools that run when the sketch is uploaded to an Arduino board. Place any comments or notes that you want between these characters. This type of comment can span multiple lines. The sketch name and description are contained in the top comment block of the Blink sketch. Detailed information about the on-board LED, and a history of the sketch follows. After that, find a link to the Arduino Blink sketch tutorial page.

A second type of comment starts with double forward slashes //. In this case, everything after the // is part of the comment. This is a single-line comment that turns everything after the // on the same line into a comment. Consequently the next line after the comment is not part of the comment.

Arduino Blink Sketch Code

The image below shows the Blink sketch code with the top comment block removed. As can be seen, it contains a mix of single-line comments and code. The code is everything in the image that is not a comment. Comments describe what is happening in the sketch.

When a sketch is loaded to an Arduino board, it is first built by software tools that automatically run. The build process consists of preprocess, compile and link stages. This converts the code from human readable text to something that runs on the Arduino board.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

Blink Sketch Code is one of the Arduino Sketches for Beginners

Arduino Sketch Syntax Highlighting

Colored words in the sketch are a result of the Arduino IDE using syntax highlighting. Syntax highlighting is the highlighting of Arduino language keywords, definitions and functions. This is more easily understood after more is learned about writing Arduino sketches.

Modify the Arduino Blink Sketch

Change the rate that the on-board LED blinks at in the Blink sketch, as follows. Change 1000 to **200** in both instances it is found in the code. The following image shows the modified Arduino Blink sketch. Red dots in the image mark the modified lines of code. Notice that the comments at the right of each modified line of code are updated to reflect the changes made in the code.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(200);                        // wait for 200 milliseconds (0.2 seconds)
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(200);                        // wait for 200 milliseconds (0.2 seconds)
}
```

Modified Blink Sketch Code

Click the Upload button in the Arduino IDE. This uploads the modified sketch to the target Arduino board. Notice that the on-board L LED blinks faster. As a result of the modified code, the LED is now on for 0.2 seconds and off for 0.2 seconds. The on and off times were originally 1000 milliseconds, also written 1000ms, which is one second (1s). This is because there are 1000ms in 1s. When 1000 is changed to 200 in the sketch, the LED on and off times change from 1000ms to 200ms. 200ms is 0.2s.

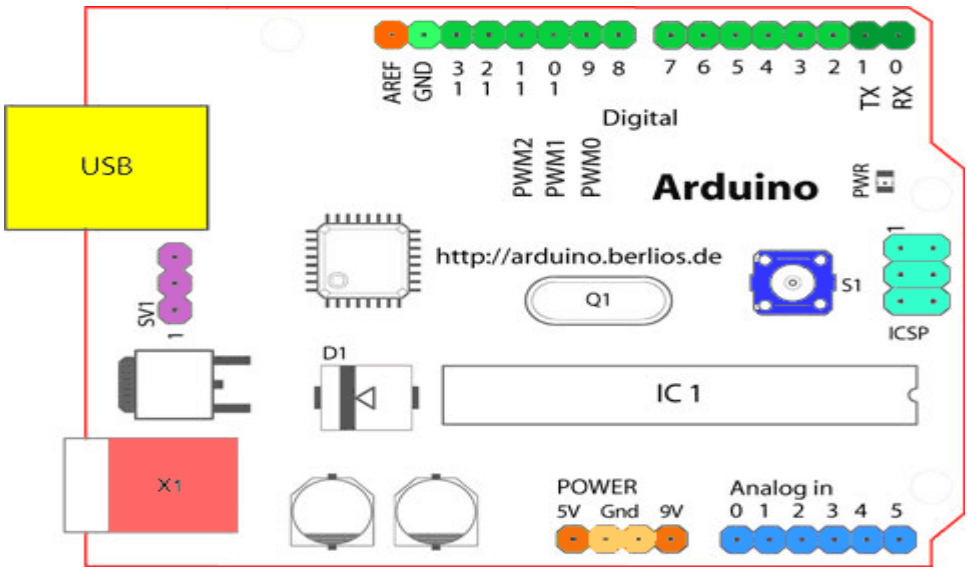
Save the Modified Sketch

Arduino example sketches are read-only. This means that they cannot be overwritten. When an example sketch is modified and saved, it must be saved to a new location.

- Three ways of saving a sketch in the Arduino IDE are firstly, click the Save toolbar icon (the arrow pointing down). Hover the mouse cursor over any toolbar icon and its name is shown at the right of the icons.
- Secondly, save the file using the keyboard shortcut **Ctrl + S** (hold down the Ctrl key and then press the s key). Thirdly, select **File → Save** on the top Arduino IDE menu bar. Because the file is read-only, the IDE prompts to save the file to a different location.
- Use the dialog box that opens to save the Blink sketch to your Arduino folder. Change the name to something like **Blink_Fast** before for saving.

3.10 Arduino Pins:

Fig: Arduino UNO



Starting clockwise from the top center:

- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital i/o (digitalRead and digitalWrite) if you are also using serial communication (e.g. Serial.begin).
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)
- Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

Microcontrollers

ATmega328P (used on most recent boards)

- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6 (DIP) or 8 (SMD)
- DC Current per I/O Pin: 40 mA
- Flash Memory: 32 KB
- SRAM: 2 KB
- EEPROM: 1 KB

ATmega168 (used on most Arduino Diecimila and early Duemilanove)

- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6 (DIP) or 8 (SMD)
- DC Current per I/O Pin: 40 mA
- Flash Memory 16 KB:
- SRAM: 1 KB
- EEPROM: 512 bytes

ATmega8 (used on some older board)

- Digital I/O Pins: 14 (of which 3 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA
- Flash Memory: 8 KB
- SRAM: 1 KB
- EEPROM: 512 bytes

Digital Pins

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pinMode(), digitalRead(), and digitalWrite() commands. Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40 mA.

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. On the Arduino Diecimila, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. On the Arduino BT, they are connected to the corresponding pins of the WT11 Bluetooth® module. On the Arduino Mini and LilyPad Arduino, they are intended for use with an external TTL serial module (e.g. the Mini-USB Adapter).
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.
- BT Reset: 7. (Arduino BT-only) Connected to the reset line of the Bluetooth® module.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- LED: 13. On the Diecimila and LilyPad, there is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

Analog Pins

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analogRead() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

- I2C: 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website).

Power Pins

- VIN (sometimes labelled "9V"). The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. Note that different boards accept different input voltages ranges, please see the documentation for your board. Also note that the LilyPad has no VIN pin and accepts only a regulated input.
- 5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- 3V3. (Diecimila-only) A 3.3 volt supply generated by the on-board FTDI chip.
- GND. Ground pins.

Other Pins

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. (Diecimila-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

3.11 Input/Output From Pins Using Sketches:

pinMode(), digitalWrite(), and delay()

The pinMode() function configures a pin as either an input or an output. To use it, you pass it the number of the pin to configure and the constant INPUT or OUTPUT. When configured as an input, a pin can detect the state of a sensor like a pushbutton; As an output, it can drive an actuator like an LED.

The digitalWrite() functions outputs a value on a pin. For example, the line:

digitalWrite(ledPin, HIGH);

set the ledPin (pin 13) to HIGH, or 5 volts. Writing a LOW to pin connects it to ground, or 0 volts.

The delay() causes the Arduino to wait for the specified number of milliseconds before continuing on to the next line. There are 1000 milliseconds in a second, so the line:

delay(1000);

creates a delay of one second.

setup() and loop()

There are two special functions that are a part of every Arduino sketch:

setup() and loop(). The setup() is called once, when the sketch starts. It's a good place to do setup tasks like setting pin modes or initializing libraries. The loop() function is called over and over and is heart of most sketches. You need to include both functions in your sketch, even if you don't need them for anything.