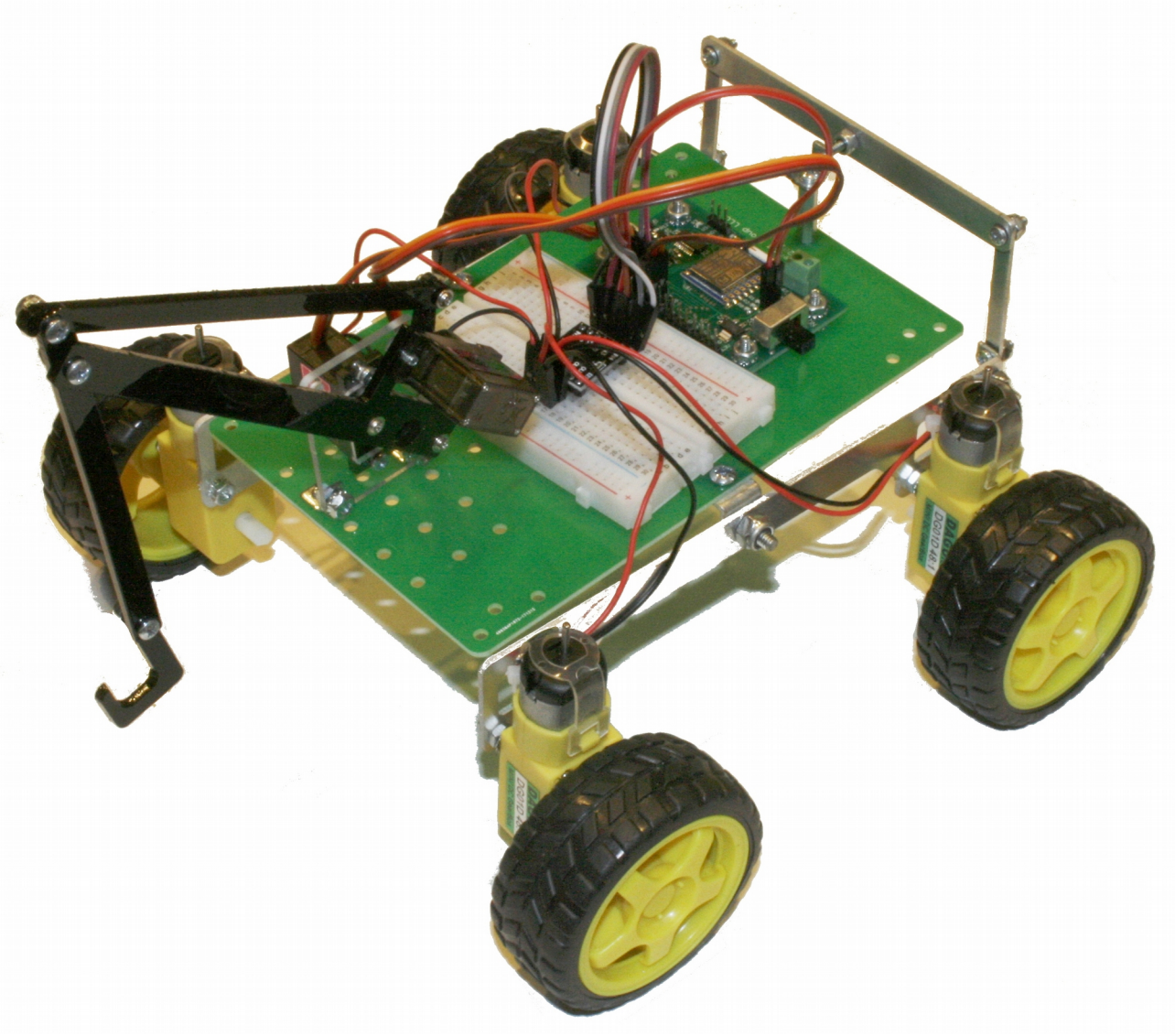
*Sten**BOT*** Robot Kit



Legal Stuff

- Stensat Group LLC assumes no responsibility and/or liability for the use of the kit and documentation.
- There is a 90 day warranty for the Quad-Bot kit against component defects. Damage caused by the user or owner is not covered.
    - Warranty does not cover such things as over tightening nuts on standoffs to the point of breaking off the standoff threads, breaking wires off the motors, causing shorts to damage components, powering the motor driver backwards, plugging the power input into an AC outlet, applying more than 9 volts to the power input, dropping the kit, kicking the kit, throwing the kit in fits of rage, unforeseen damage caused by the user/owner or any other method of destruction.
- If you do cause damage, we can sell you replacement parts or you can get most replacement parts from online hardware distributors.
- This document can be copied and printed and used by individuals who bought the kit, classroom use, summer camp use, and anywhere the kit is used. Stealing and using this document for profit is not allowed.
- If you need to contact us, go to www.stensat.org and click on contact us.

References

- www.arduino.cc
- esp8266.github.io/Arduino/versions/2.4.0

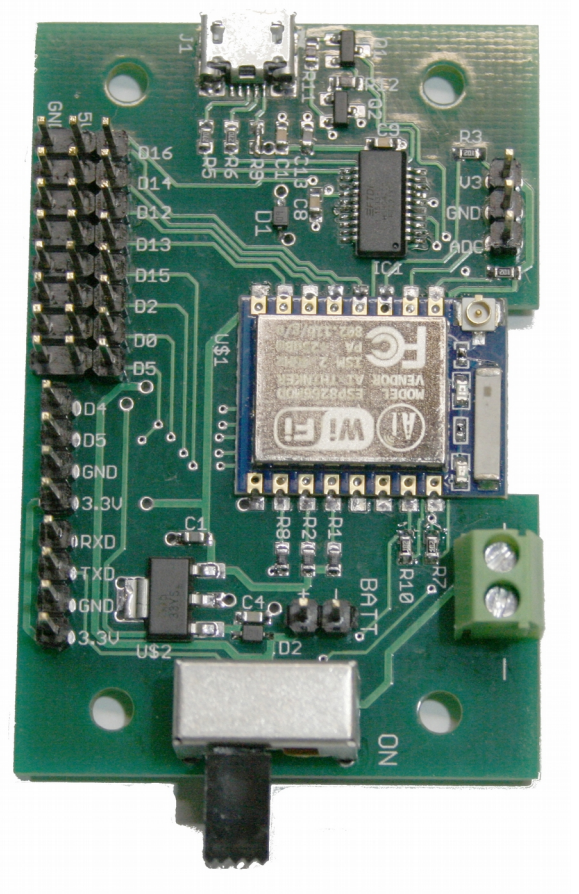
Processor Board
and
Arduino Software

Overview

- In this section, you will be introduced to the processor board electronics and the arduino software.
- At the end of this section, you will be able to write software, control things and sense the environment.

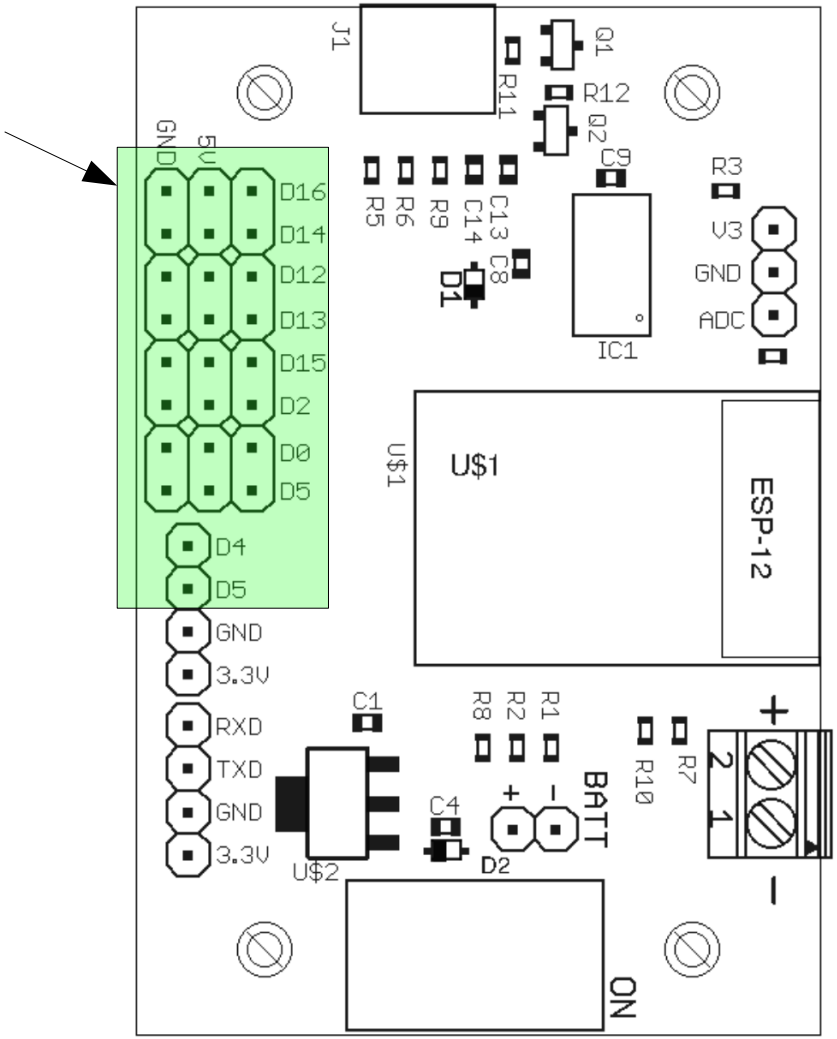
Processor Specifications

- The processor is shown to the right. It is called an embedded computer because it is to be integrated or embedded in something, this case a robot.
- The processor board has connections that allow devices to be interfaced such as lights, motors and sensors.
- There are two types of interfaces that will be used for the robot, digital interfaces and analog.
- The digital interfaces can be configured as an input or output.
  - As an input, the digital interface can detect the state of switches or other signals as being on or off.
  - As an output, the digital interface can turn things on and off such as lights.
- The analog interface is an input only can can measure voltages.



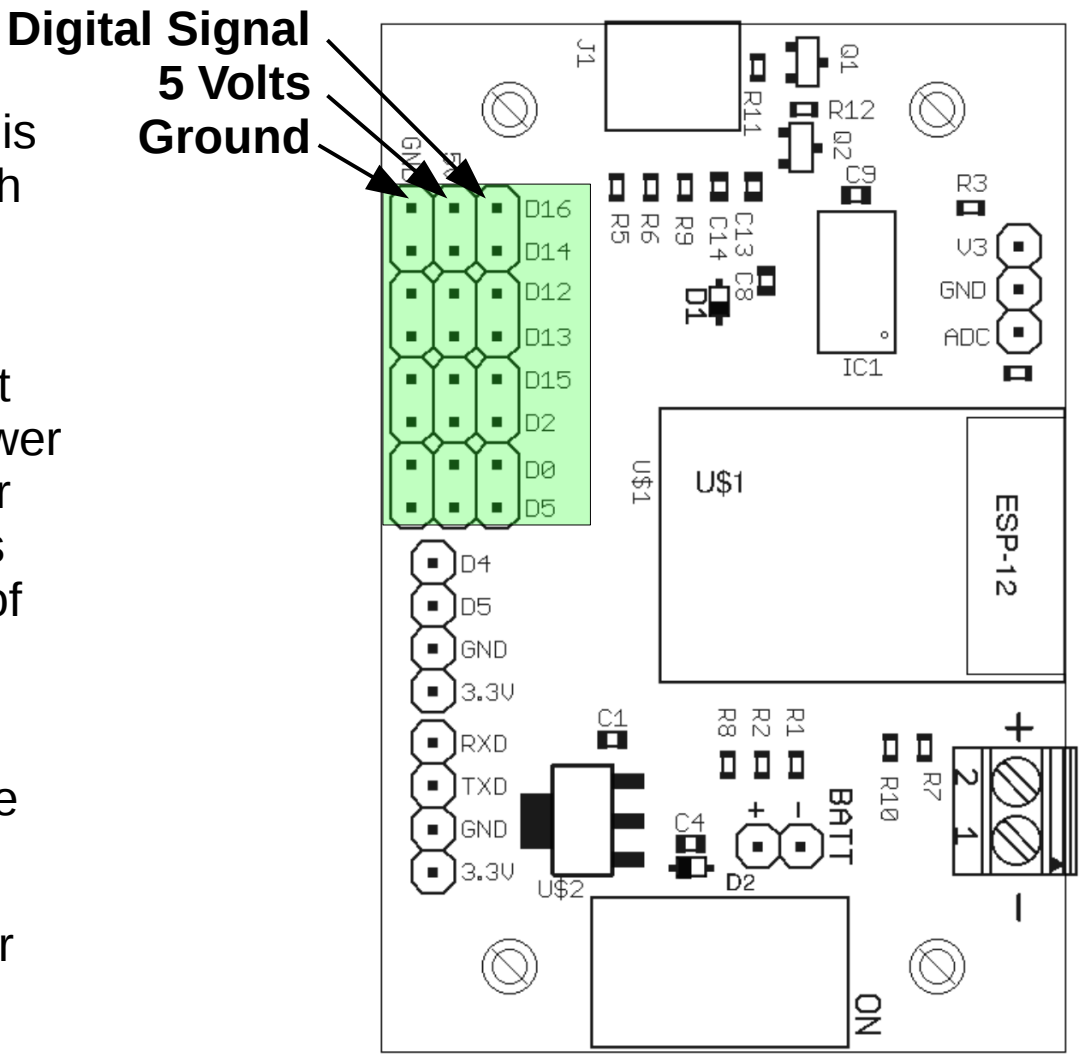
Digital Pins

- The digital pins are highlighted in green.
- There are eleven digital ports total. Some are dedicated and others have multiple purposes.
- The digital pins can be used to control things and detect things.
- When the digital pin is set low, the voltage is set to zero volts.
- When the digital pin is set high, the voltage is set to 3.3 volts.
- Digital pins 0,5,2,15,13,12,14,16 can be used as digital signals, servo controllers and digital inputs.
- RXD and RXD can be used to communicate with serial devices but will be used for program upload and debugging.



Processor Board Pinout

- The digital pin configuration is shown to the right.
- The digital pin is the most inward pin. This is the digital signal that can be set to high or low, logic level 1 or 0, 3.3 volts or 0 volts.
- The next pin is 5 volts. This is power that can be used. It is available when the power switch is set to on and an external power source is connected. The USB port does not power this 5 volts for the protection of the host computer.
- The last pin closest to the edge of the board is the ground. This is the reference voltage of zero volts.
- The pins are positioned so a servo motor can be directly plugged in.
- Each pin can be configured as an input or output.

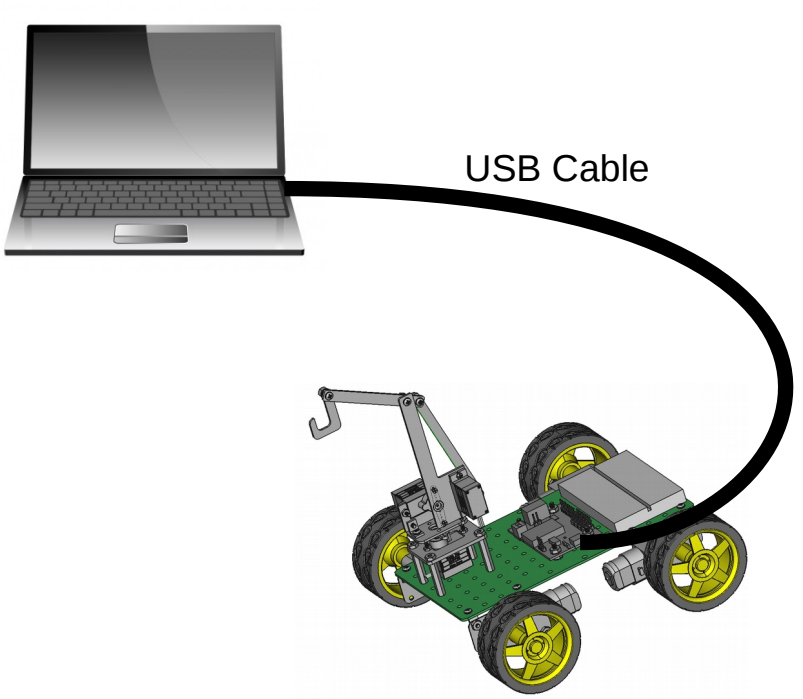


Software

- Now that the processor features have been covered, it is time to learn about programming it.
- The processor uses the arduino software. This software allows you to write programs, compile them and upload them to the processor. It also allows you to interact with the software running on the processor.
- Only one program can be installed and run at a time. The processor is small and does not have an operating system.
- Embedded computers are designed to perform a specific task and not operate like a desktop computer or laptop.
- More information about the arduino software can be found at
  - www.arduino.cc

How Things Work

- The Arduino software on the laptop is where you write code and compile it. The compiling translates your written code into machine code that is executed on the rover.
- The rover has the processor board that executes the code you wrote. When you upload the code, the Arduino software in the laptop sends the program over the USB cable to the rover to be executed on the rover.
- The program you write runs on the rover processor.



Loading and Configuring Arduino Software

- Download the proper version of the Arduino software from www.arduino.cc
- Once downloaded and installed, run the program.
- Under the “File” menu, select “Preferences”
- Find the text entry space to the right of “Additional Board Manager URLS:
- Enter the following into the text area
  - http://arduino.esp8266.com/stable/package_esp8266com_index.json
- Click OK.
- Select the menu “Tools” and then “Boards:” and then “Boards Manager”.
- A window will open. Scroll down until “esp8266” is located. Click on it and click Install.
- The software will load the compiler for the processor board.

Configuring The Software

- Select the “Tools” menu again and then “Board:”.
- Select “Generic ESP8266 Module”
- Go back to “Tools” menu and select “Reset Method”. Select “nodemcu”
- Nothing else needs to be changed. This completes setting up for the processor board.
- To the right is a representaion of the Tools Menu. All selections should look like it. Only the Port: selection may be different.
- On the next page, you configure the COM port.

Auto Format
Archive Sketch
Fix Encoding & Reload
Serial Monitor
Serial Plotter

Board: “Generic ESP8266 Module”
Flash Mode: “QIO”
Flash Size: “512K (64K SPIFFS)”
Debug Port: “None”
Reset Method: “nodemcu”
Flash Frequency: “40MHz”
CPU Frequency: “80 MHz”
Upload Speed: “115200”
Port: “COM3”

Programmer: “AVRISP mkII”
Burn Bootloader

Fading LEDs

- The **for()** loop is a powerful statement. It lets the code execute a group of instructions multiple times but not for ever.
- Leave the circuit as is. The red LED will be faded.
- Create a new program and enter the code to the right.

```
void setup()
{
  pinMode(16,OUTPUT);
}

void loop()
{
  for(int i=0;i<1023;i++) {
    analogWrite(16,i);
    delay(20);
  }
}
```

Configuring Arduino Software

- Plug the processor board into the computer USB port
  - Let the operating system find the drivers. (network connection required)
    - The driver is also included with arduino software
  - In the arduino program select menu “Tools”
  - Select “serial Port”
  - Select the appropriate COM port.
    - If you have a modem built in or existing COM ports, the COM number for the processor will usually be the highest number.
  - Next two pages describe how to install the drivers if needed.

Windows COM Port

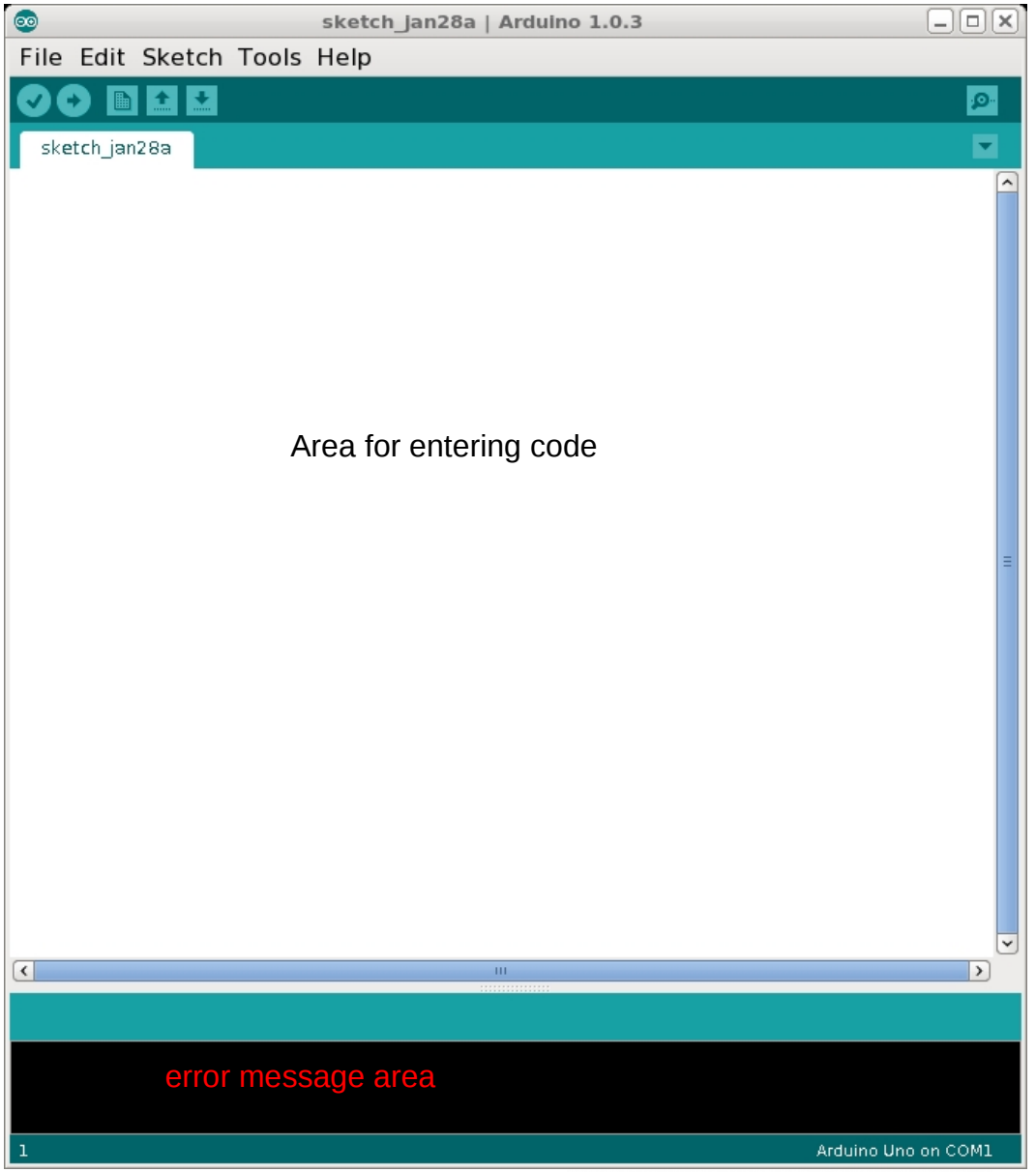
- If the Serial Port menu does not show any COM ports try the following:
  - Go to http://www.ftdichip.com/Drivers/VCP.htm
  - In the table on the website, locate the row specifying **`Windows`**.
  - Click on the link **`setup executable`** and download the software.
  - Right click on the icon labeled **`CDM v2.12.00.....`** and select **`Run as Administrator`** in the menu that pops up.
  - Follow the instructions to install. You have to run the installation program as an administrator or the driver will not install.
  - Go back to Arduino and select the COM Port it gives you.
    - Most likely, the COM port will be COM3. If there is more than one COM port, choose the higher number.

Macintosh OS X USB Driver Installation

- Go to http://www.ftdichip.com/Drivers/VCP.htm
- Select VCP driver (Virtual COM Port) for the Mac OS X.
- Download and double click on the file
  - A window will open showing two packages
- Double click the package for the OS version you have
- Follow instructions for the installation
- Go back to Arduino and select the Serial Port: /dev/tty.usbserial-DAxxxxxx
  - The xxxxxx will be some combination of letters and numbers

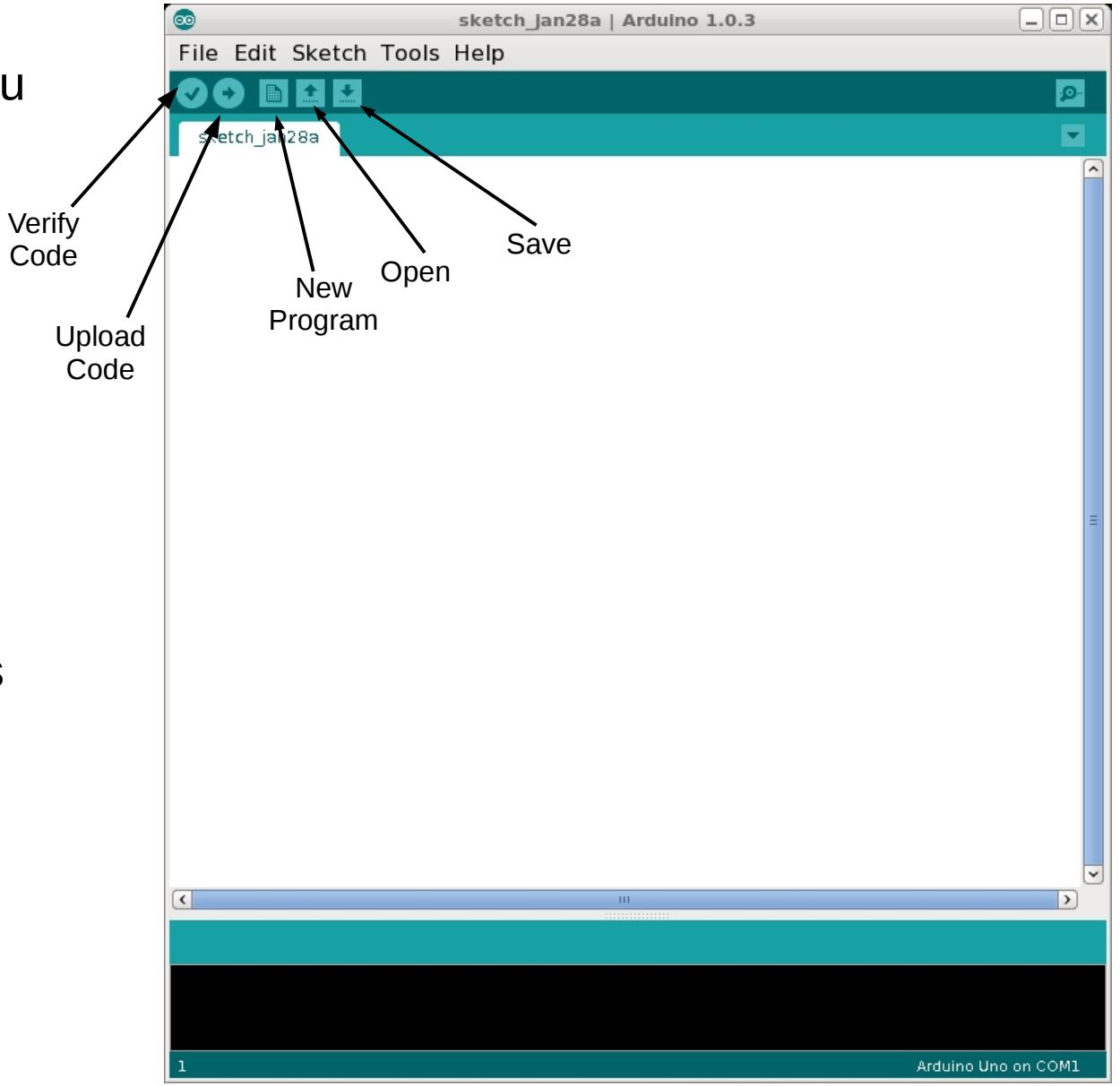
Using Arduino

- This is the arduino software.
- The software will let you enter programs and upload the code to the processor board.
- The large white area is where the code is entered.
- The black area below is where error messages will be displayed such as when there is an error in the code or the software cannot upload code for some reason.



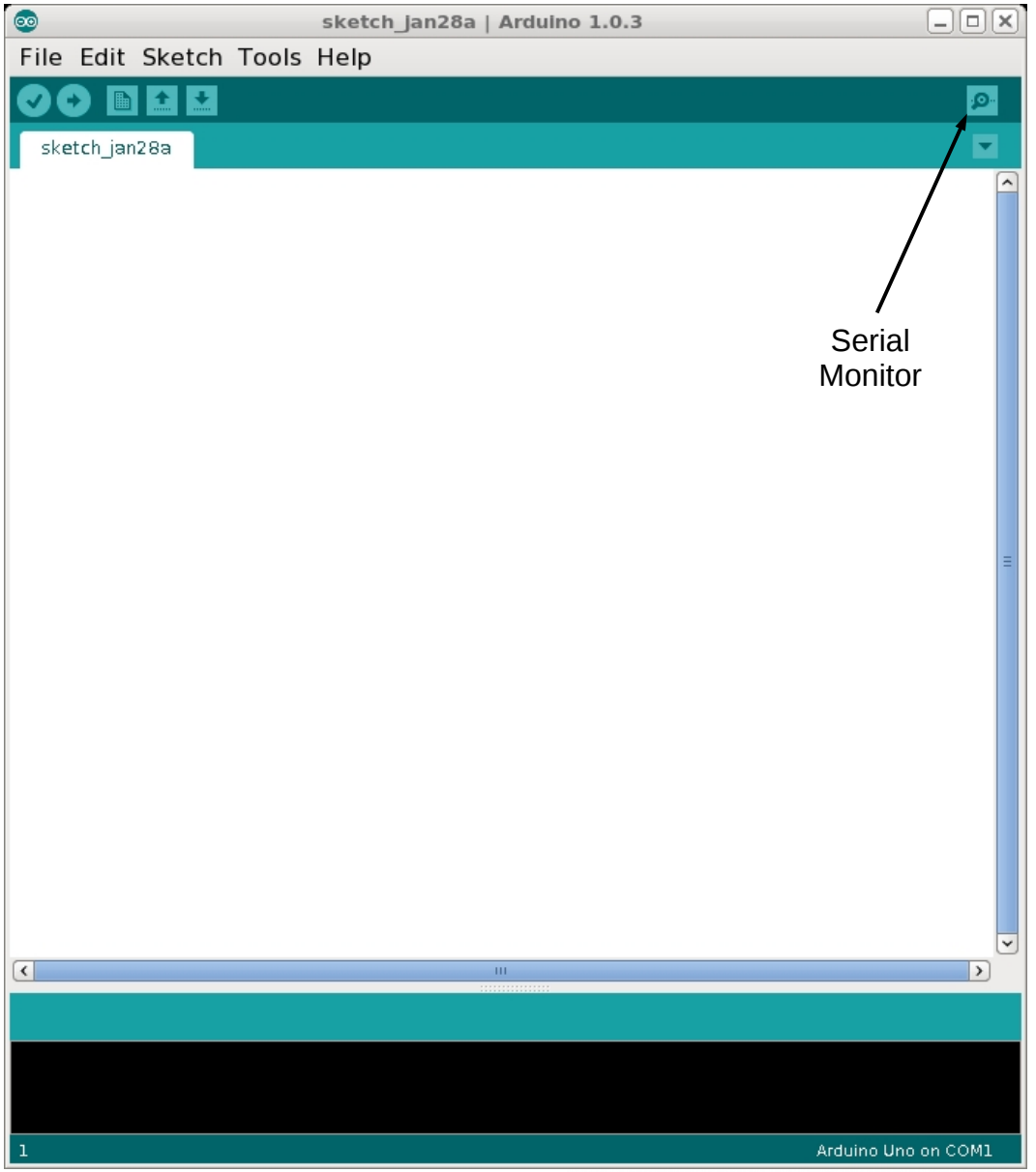
Using Arduino

- The buttons below the menu have different functions.
- The first called Verify Code will compile the code and check for errors but not upload the code.
- The next button will do the same as the first but also upload the code.
- New Program button opens a new copy of the program allowing you to start writing another program.
- Open and Save are for opening and saving the code you have written.



Using Arduino

- Serial Monitor button opens a new window allowing you to interact with the processor.
- The Serial Monitor window allows the processor to display information and you to send information.
- This will be used quite a bit in this section.



First Program to Test

- Enter the program in the editor on the right. **Do not copy and paste from the pdf file**. It doesn't work. The compiler is case sensitive so pay attention to capitalized letters.
- Plug the processor board into the USB port.
- Click on the upload Code button to compile and upload the program.
- When the status message at the bottom of the window says done uploading, click on the serial monitor button.
- The Serial Monitor window pops up with the message being displayed. In the lower right of the window, change **9600 baud** to **115200 baud.**
- Save your program. Pick a file name.

```
void setup()
{
    Serial.begin(115200);
}

void loop()
{
    Serial.print("Hello World");
}
```

Serial Monitor Window

What are Functions

- A function is basically a set of instructions grouped together. A function is created to perform a specific task.
- The set of instructions for a function are bounded by the curly brackets as seen to the right.
- The **`setup()`** function is used to initialize the processor board, variables, and devices.
- Inside functions, you can call other functions. **`Serial.begin()`** is a function. It is located somewhere else in the arduino software.

```
void setup()
{
    Serial.begin(115200);
}

void loop()
{
    Serial.print("Hello World");
}
```

Other Syntax Requirements

- You will notice that some lines end with a semi-colon. This is used to identify the end of an instruction. An instruction can be an equation or function call.
- When you create a function such as **`setup()`**, you do not need a semi-colon.

```
void setup()
{
    Serial.begin(115200);
}

void loop()
{
    Serial.print(“Hello World”);
}
```

Arduino Programming Basics

- The program is made up of two functions.
- **`setup()`** function is run at reset, power up or after code upload only once.
  - It is used to initialize all the needed interfaces and any parameters.
- **`loop()`** function is run after the **`setup()`** function and is repeatedly run hence the name loop.
- This program configures the serial interface to send messages at 115200 bits per second.
- The message is “Hello World” and is repeatedly displayed.

```
void setup()
{
    Serial.begin(115200);
}
```

```
void loop()
{
    Serial.print(“Hello World”);
    delay(500);
}
```

•**`Serial.begin()`** is a function that intializes the serial interface and sets the bit rate.
•**`Serial.println()`** sends the specified message over the serial interface and move the cursor to down one line.
•**`delay(500)`** is a command to stop the program for 500 milliseconds.

What is in the Software

- In the **`setup()`** function, it executes the function **`Serial.begin(9600);`**
  - This function initializes the UART which is connected to the USB port to allow for communications.
- In the **`loop()`** function, it executes the function **`Serial.print("Hello world");`**
  - This function send the text in quotes to the UART. This is displayed in the Serial Monitor window.
- The other function is called **`delay()`**.
  - This function stops the program for a specified period of time. The unit is in milliseconds. The code to the write displays the text every half second.

```
void setup()
{
    Serial.begin(115200);
}
```

```
void loop()
{
    Serial.print("Hello World");
    delay(500);
}
```

What is in the Software

- In the Serial Monitor window, you may have noticed that the text displayed scrolls to the right. That is just how Serial.print() works.
- To have the text displayed on its own line, change the Serial.print() to Serial.println().
- Serial.println() adds a line feed which forces the text in the Serial Monitor to move down one line.
- Make the change, upload the code and open the Serial Monitor window.

```
void setup()
{
    Serial.begin(115200);
}
```

```
void loop()
{
    Serial.println("Hello World");
    delay(500);
}
```

Electronics

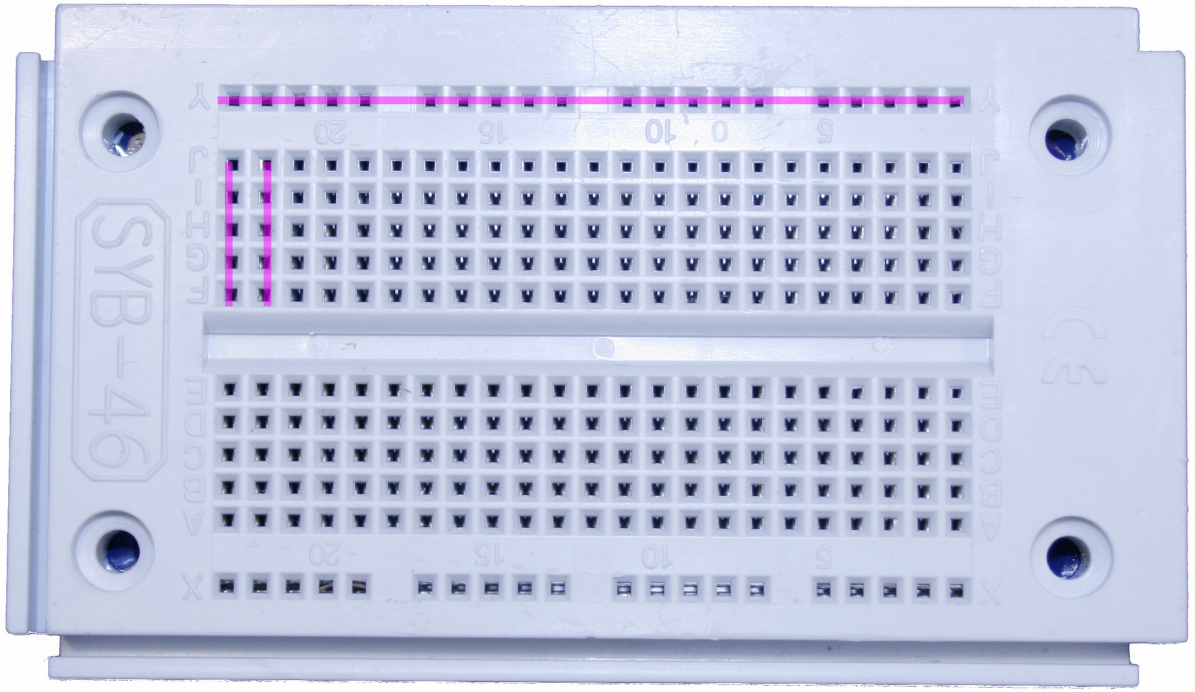
- At this point, you should be able to run the arduino software.
- You should know that the software consists of two functions
  - setup()
  - loop()
- You should know how to initialize the UART and write a program to display text in the Serial Monitor window.
- You should know how to open the Serial Monitor window.
- Next is learning a little about electronics and how to control things.

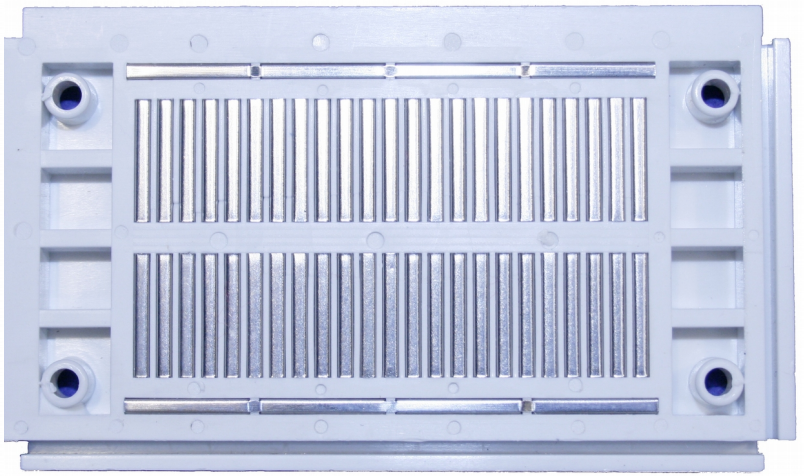
Electrical Circuits

- In this section, you will learn how to connect electrical circuits.
- Electrical circuits is nothing more than connecting wires between devices to allow the flow of electrons. A lamp plugged into an outlet has two wires to make an electrical circuit.
- **CAUTION: Always disconnect power when wiring up circuits. The processor is powered when the USB cable is connected. Always disconnect the USB cable when making or changing any wire connections. The processor can be damaged.**

How the Solderless BreadBoard Works

- The solderless bread board allows circuits to be quickly connected.
- Each row of holes that go left to right on the top and bottom are all connected together.
- The columns of 5 holes are all connected together.
- The lines in the picture show the connections.
- Components and wires are inserted in the holes to make connections.

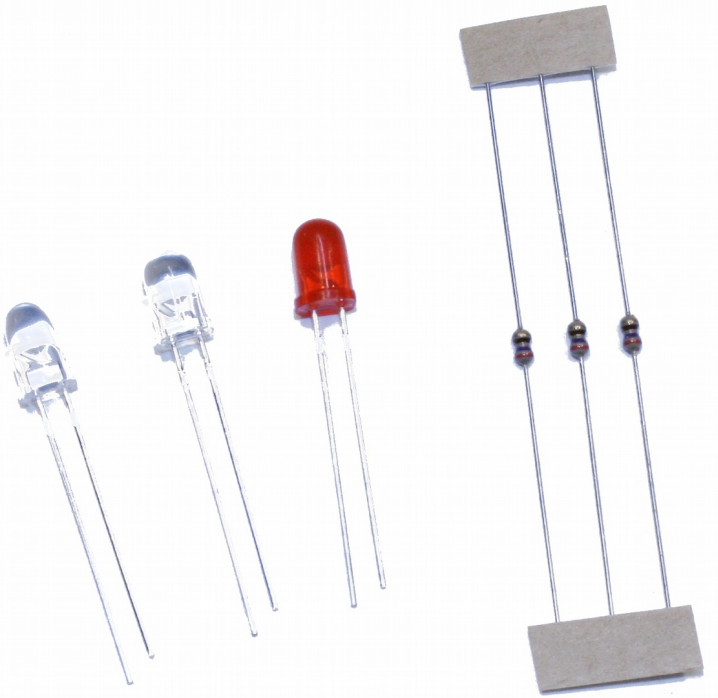




Back side showing how holes are connected

Components

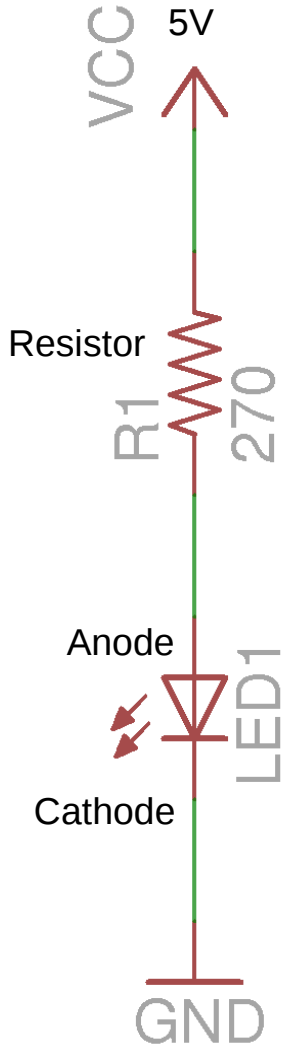
- The first circuit will use a Light Emitting Diode or LED.
- The LED is a polarized device and only works in one direction and gives off light when current flows through it.
- The positive pin on the LED is the longer pin. It is called the anode. The other end is the cathode.
- LEDs need the current to be limited otherwise it will take too much and burn out. A resistor will be used to limit the current flow through the LED.



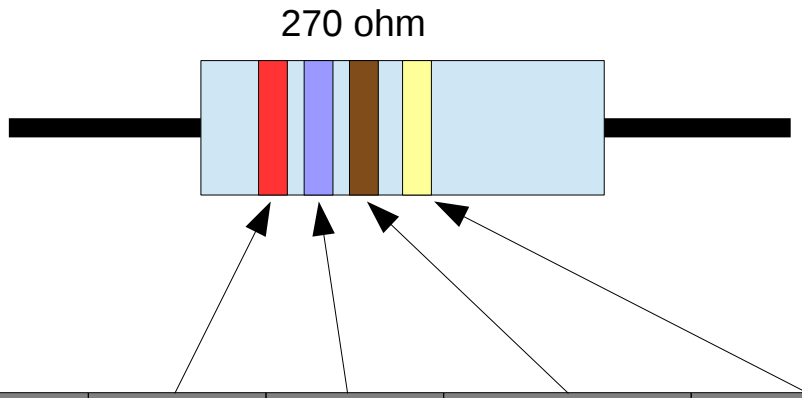
LED Resistors

First Circuit

- The first circuit will connect the LED straight to 5 volts so the LED will always be lit when there is power.
- The schematic for the circuit is shown to the right.
  - The symbol at label R1 is for the resistor.
  - LED1 is next to the symbol for the LED.
- The symbol at the top is the +5V connection. It is called VCC.
- The GND symbol is for ground. This is the zero volt reference.
- The LED has an anode and a cathode. The anode is the long pin.
- When the anode is at a higher voltage than the cathode, the LED will light.



Resistor Code



Color	Name	Digit 1	Digit 2	Multiplier	Tolerance
	Black	0	0	x1	
	Brown	1	1	x10	1%
	Red	2	2	x100	2%
	Orange	3	3	x1,000	3%
	Yellow	4	4	x10,000	4%
	Green	5	5	x100,000	
	Blue	6	6	x1,000,000	
	Violet	7	7		
	Grey	8	8	Gold	5%
	White	9	9	Silver	10%

A resistor is a device that is used to limit the flow of current or reduce the voltage depending on how it is used.

Resistor values are determined by the color bands. The picture to the left shows how to decipher the color bands. The first two bands determine numerical value and the third band is the multiplier. A 270 ohm resistor has a red, violet, and brown band. The last band indicates the how far off the value can be.

The bands start at one end of the resistor.

Wiring Diagram for LED

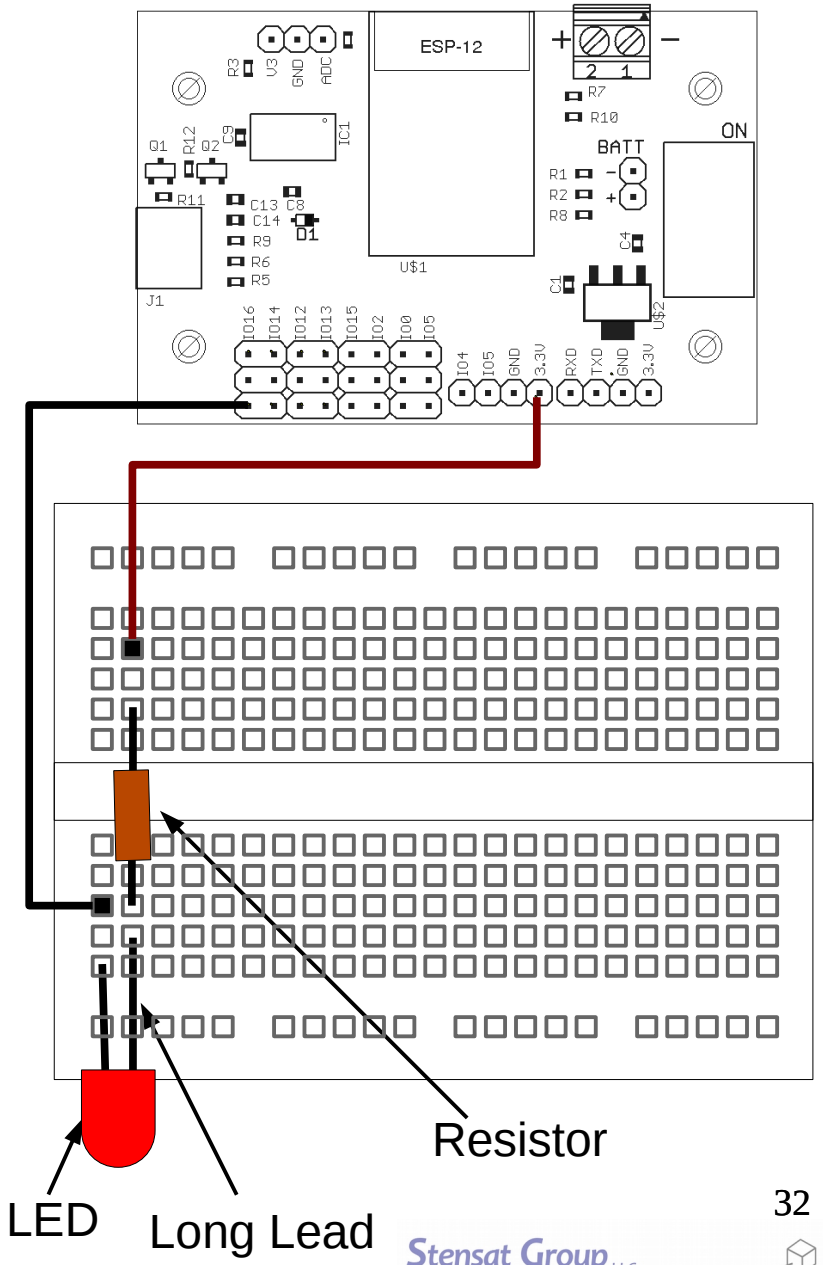
With the USB cable disconnected, insert the LED into the bread board as shown. The short lead on the LED should be on the left side.

Insert the 270 ohm (red, violet, brown) resistor as shown. One lead should be installed in the same column as the long lead of the LED. The other resistor lead is inserted in any other column.

Take the black jumper wire and insert the pin into the column of the short LED lead. Plug the other end into the ground pin as shown.

Take a red jumper and connect the pin into the same column as the resistor lead and the other end into 3.3V pin as shown.

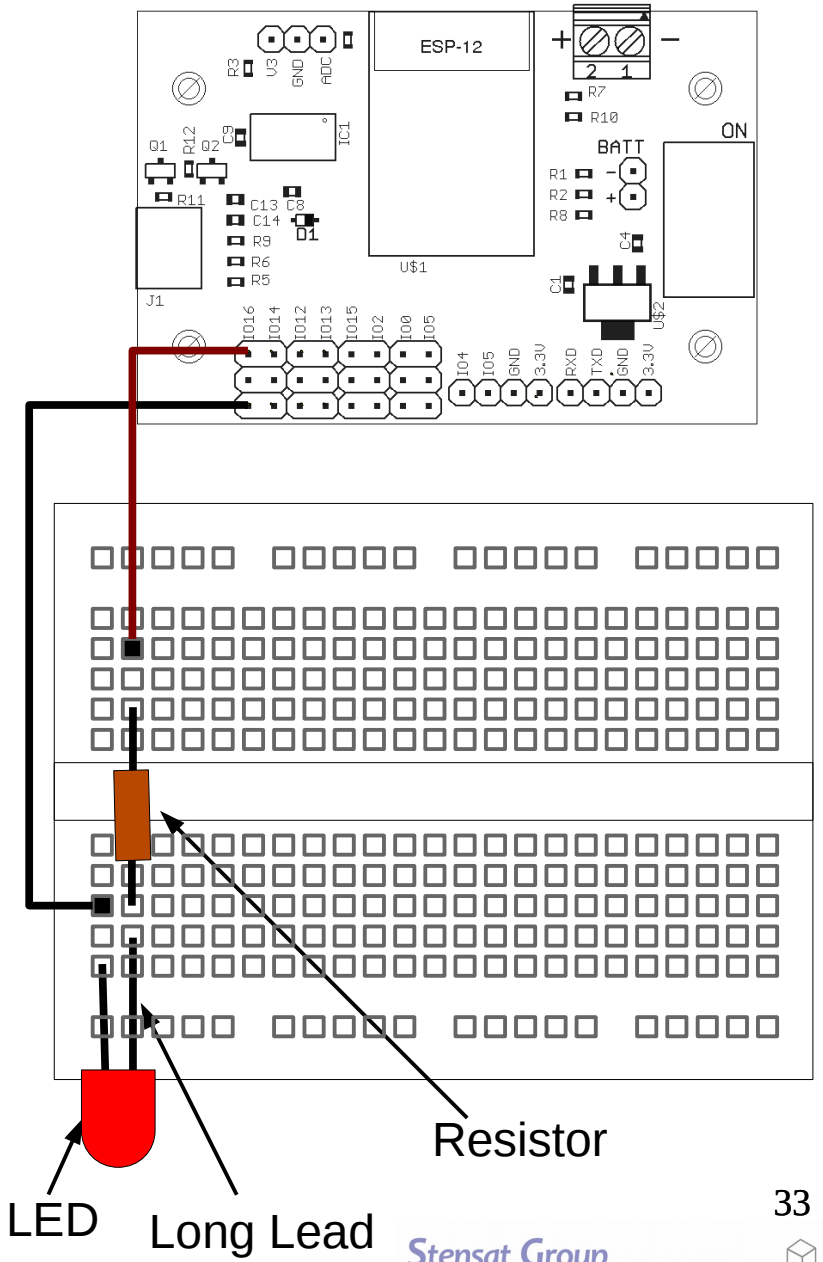
Plug the processor board into the computer USB port. The LED should light up. If not, reinsert the LED in the opposite orientation.



LED Connected to Digital Pin

Move the red jumper from the 3.3V pin on the processor board to the pin marked D16.

Leave everything else as is.



Connecting the LED to a Digital Pin

- The LED is not lit at this time because the digital pin 16 needs to be programmed to generate a voltage.
- The program to the right will cause the LED to blink.
- Create a new program and enter the code.
- Upload the code to the processor board. Make sure the processor boar is plugged into the USB port.
- Save the program and use a new file name such as “blinky”.

```
void setup()
{
    pinMode(16,OUTPUT);
}

void loop()
{
    digitalWrite(16,HIGH);
    delay(500);
    digitalWrite(16,LOW);
    delay(500);
}
```

Connecting the LED to a Digital Pin

- In the **`setup()`** function, digital pin 16 is configured as an output.
  - The function **`pinMode()`** configures digital pin 16 to be an output.
  - **`pinMode()`** takes two arguments separated by a coma.
    - The first argument selects the digital pin.
    - The second argument configures the digital pin as an output.
- in the **`loop()`** function, digital pin 16 is set high which causes the pin to generate 3.3 volts. The LED turns on.
- The **`delay()`** function halts the program for 500 milliseconds.
- The next **`digitalwrite()`** command sets digital pin 16 to 0 volts turning off the LED.
- Save the program and use a new file name such as “blinky”.

```
void setup()
{
    pinMode(16,OUTPUT);
}

void loop()
{
    digitalWrite(16,HIGH);
    delay(500);
    digitalWrite(16,LOW);
    delay(500);
}
```

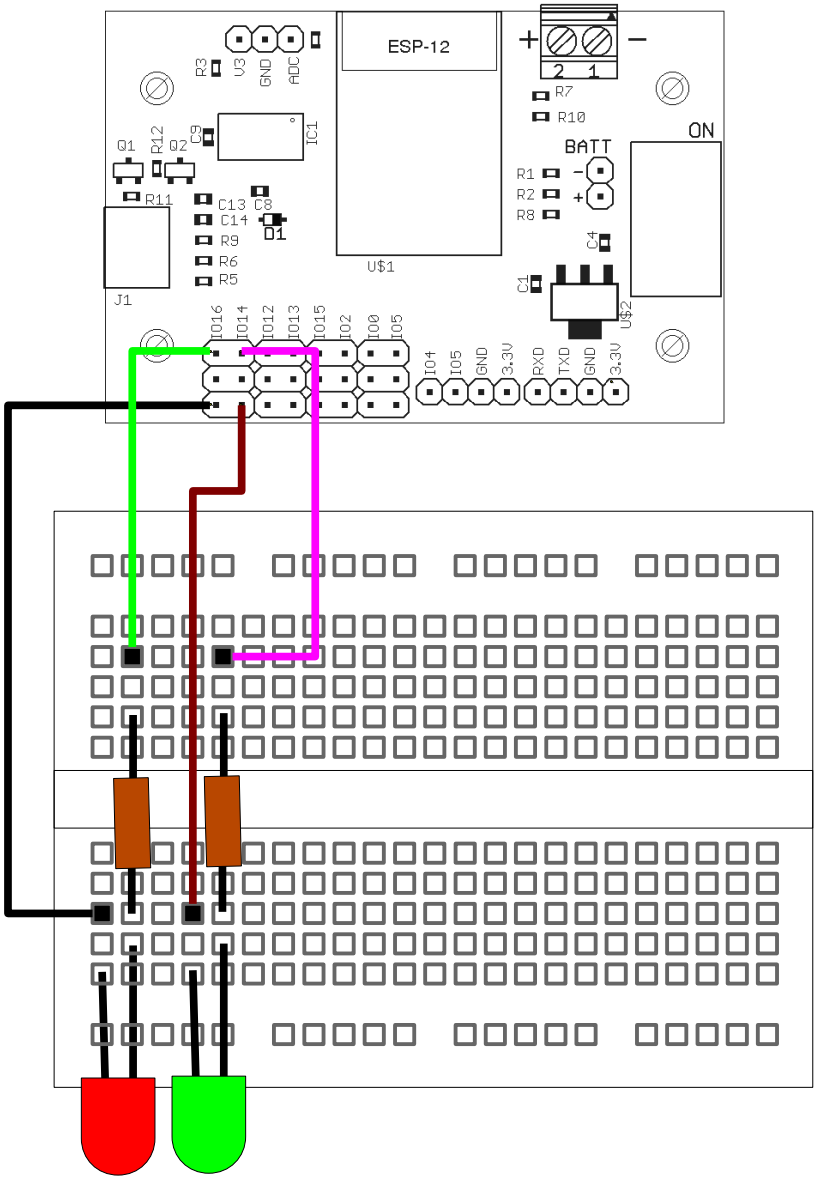
digitalWrite()

- The digitalWrite() function controls a pin and can set it high or low.
  - The function has two arguments separated by a coma.
    - The first argument selects the digital pin.
    - The second argument sets the digital pin.
      - When set high, the pin is set to 3.3 volts.
        - In logic terms, a high signal is logic level 1.
      - When set low, the pin is set to 0 volts.
        - In logic terms, a low signal is logic level 0.
  - The function is written as
    - digitalWrite(pin,setting)
      - setting is HIGH or LOW
        - The letters need to be capital.

HIGH = 5 volts = Logic 1
LOW = 0 volts = Logic 0

Two LEDs

- Add a second LED to digital pin 14 just like the first LED circuit. Use a second resistor. Make sure to not use the same columns as the first LED.



Two LEDs

- Add the highlighted code into the existing code.
- The second LED will turn on when the fist LED turns off and turn off when the first LED turns on.

```
void setup()
{
    pinMode(16,OUTPUT);
    pinMode(14,OUTPUT);
}

void loop()
{
    digitalWrite(16,HIGH);
    digitalWrite(14,LOW);
    delay(500);
    digitalWrite(16,LOW);
    digitalWrite(14,HIGH);
    delay(500);
}
```

Fading LEDs

- The **for()** loop is a powerful statement. It lets the code execute a group of instructions multiple times but not for ever.
- Leave the circuit as is. The red LED will be faded.
- Create a new program and enter the code to the right.

```
void setup()
{
  pinMode(16,OUTPUT);
}

void loop()
{
  for(int i=0;i<1023;i++) {
    analogWrite(16,i);
    delay(20);
  }
}
```

Fading LEDs

- First, the command **int a;** declares a variable. A variable is a place to store numbers that can be changed.
- The **for()** loop has three parts.
  - The first part is the initial condition.
  - The second part is the test condition.
  - The third part is the modifier.
- The initial condition sets the variable to **a** start value. In this code, the variable a is set to 0.
- The test condition compare **a** to the number 256. As long as **a** is less than 256, the condition is true. That means the code in the brackets get executed.
- The modifier changes the value of **a.** In this code, the variable a is incremented by 1.

```
void setup()
{
  pinMode(16,OUTPUT);
}

void loop()
{
  int a;
  for(a=0;a<1023;a=a+1) {
    analogWrite(16,a);
    delay(20);
  }
}
```

Fading LEDs

- Upload the code and run it. See what happens with the LED.
- Over time the LED should get brighter and then go dark and start over.
- Add a second **for()** loop to make the LED fade from bright to dark.

```
void setup()
{
  pinMode(16,OUTPUT);
}

void loop()
{
  int a;
  for(a=0;a<1023;a=a+1) {
    analogWrite(16,a);
    delay(20);
  }
}
```