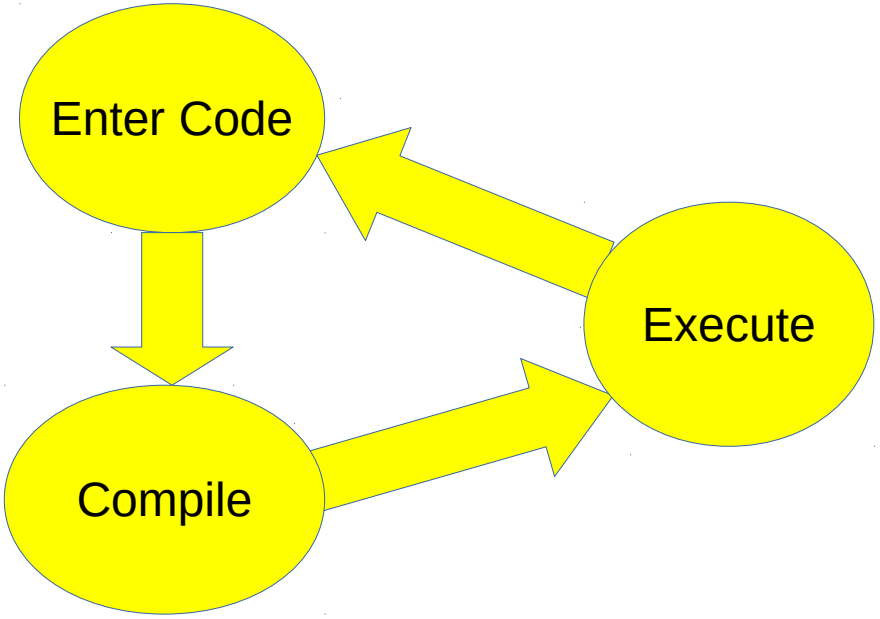
# Programming

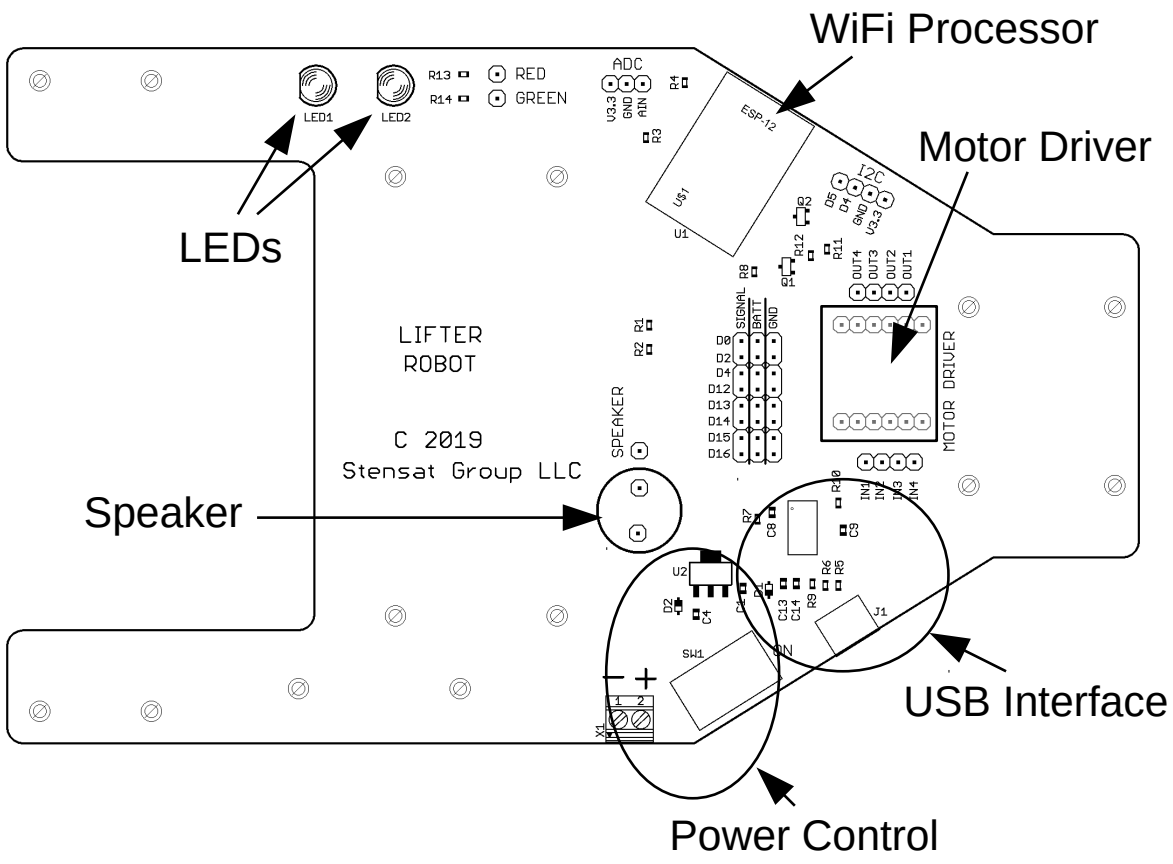
# Introduction

- There is a cycle to programming the processor board.
- First, you write code using instructions people can read which is called a programming language.
- Next, the code gets compiled which translates what you typed to machine code that the computer understands.
- Lastly, the code is executed by the computer.
- If there is an error, you change you code and start the process over.



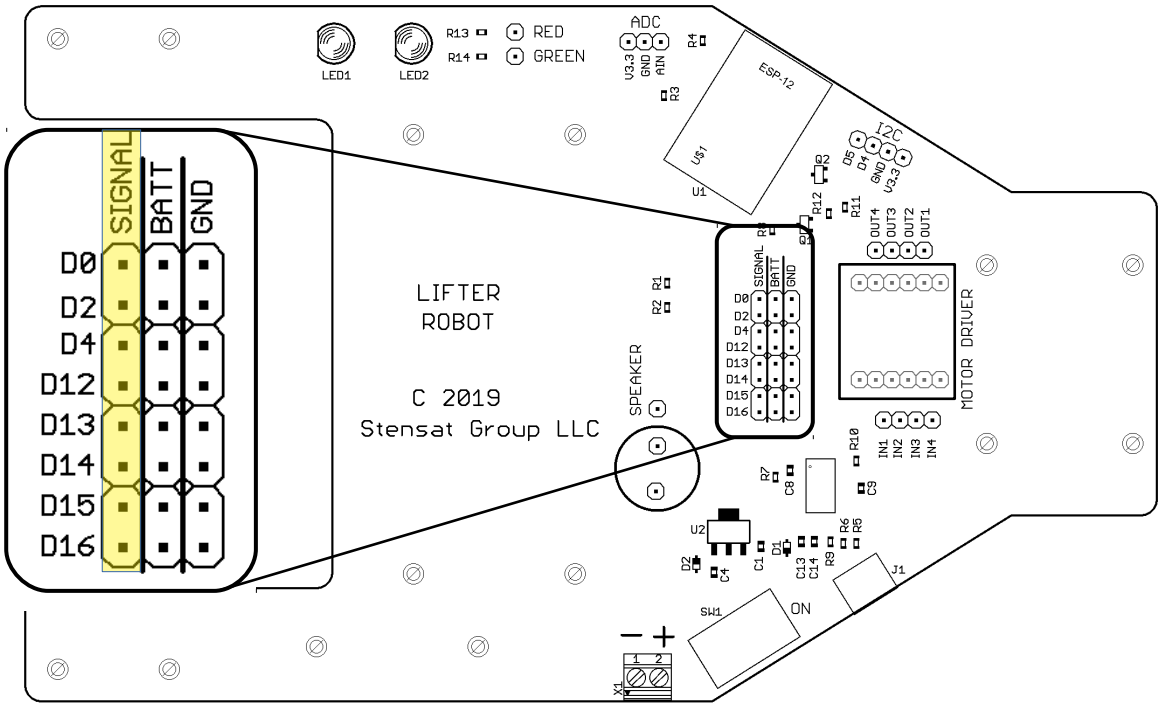
# Introduction

- The base plate is the processor board.
- It includes a WiFi processor, motor driver, LEDs and speaker.
- Uploading code is through the USB interface.
- The power control allows the USB interface to power the electronics but not the motors. Only batteries can power the motors.



# Introduction

- On board are pin connectors. The allow jumpers to be used to connect the various components on the base plate.
- Shown up close are the digital pins. These pins can control motors, LEDs and the speaker.
- The column marked SIGNAL is what will be used.



# Digital Pin

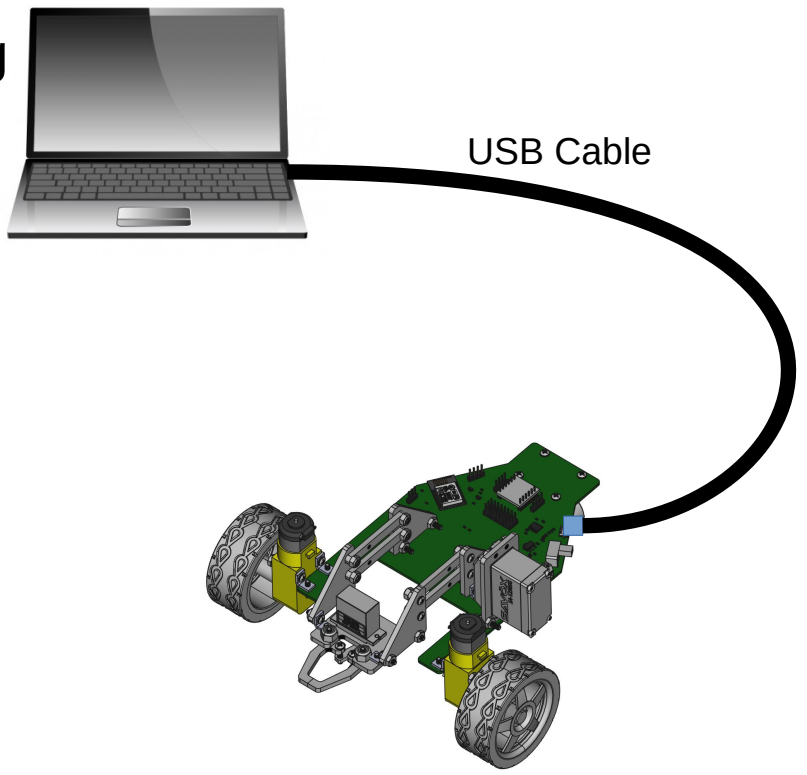
- Digital pins work similar to a switch. The digital pin can be turned on or off. In a computer, the digital pin is set high or low which is equivalent to to be turned on or off.
- When set high, the pin will be set to 3.3 volts.
- When set low, the pin will be set to 0 volts.

# 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 robot 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 representation 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”

---

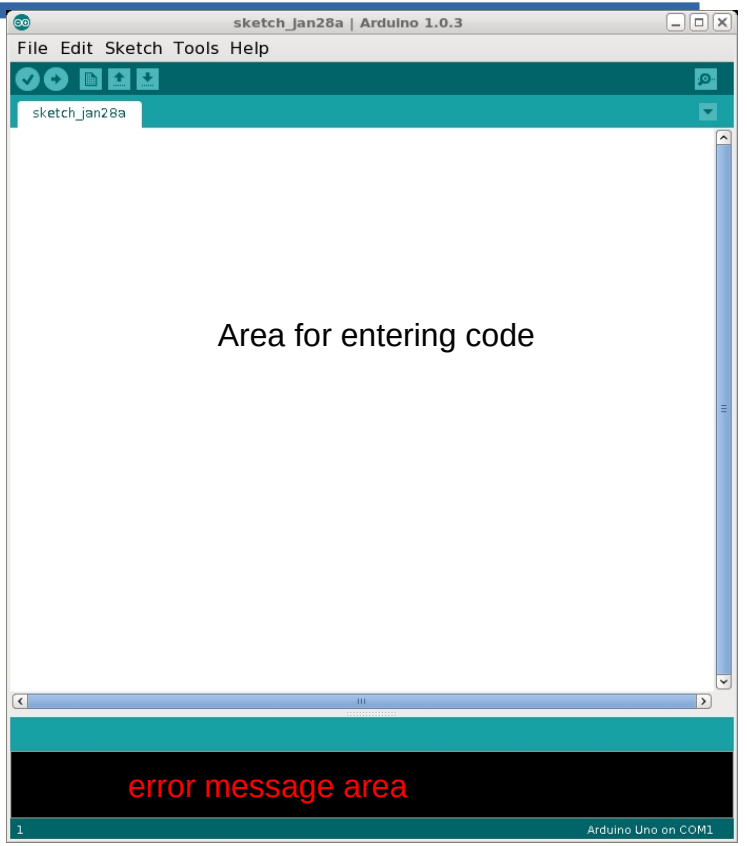
Programmer: “AVRISP mkII”
Burn Bootloader

# 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.
- On a Mac, the port is not a bluetooth device.

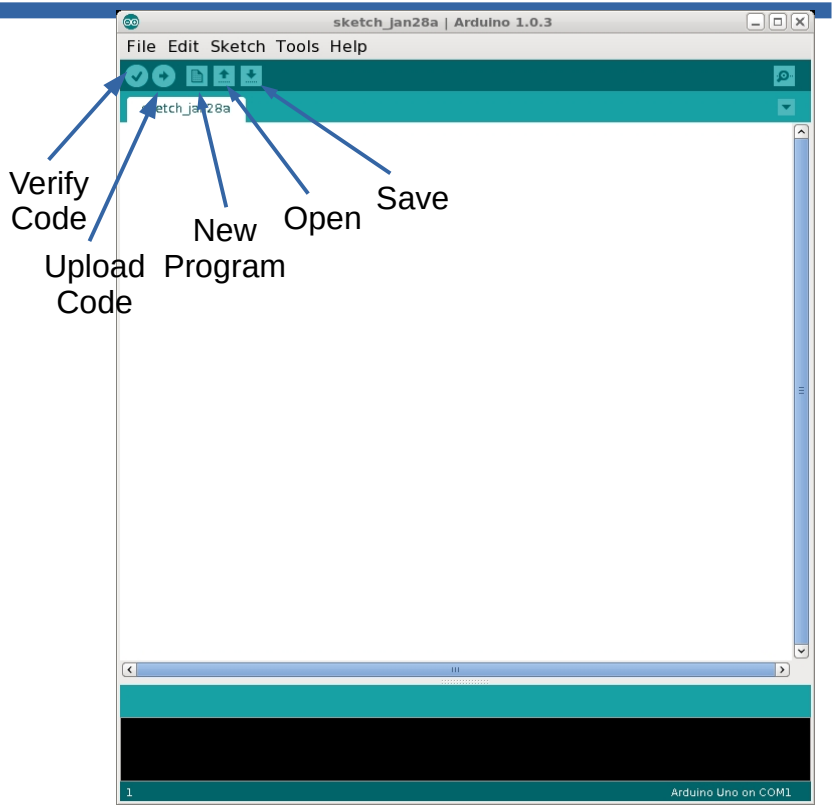
# Arduino Software

- This is the arduino software.
- The software will let you enter programs and upload the code to the processor board.
- The programming language is C and C++.
- 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.



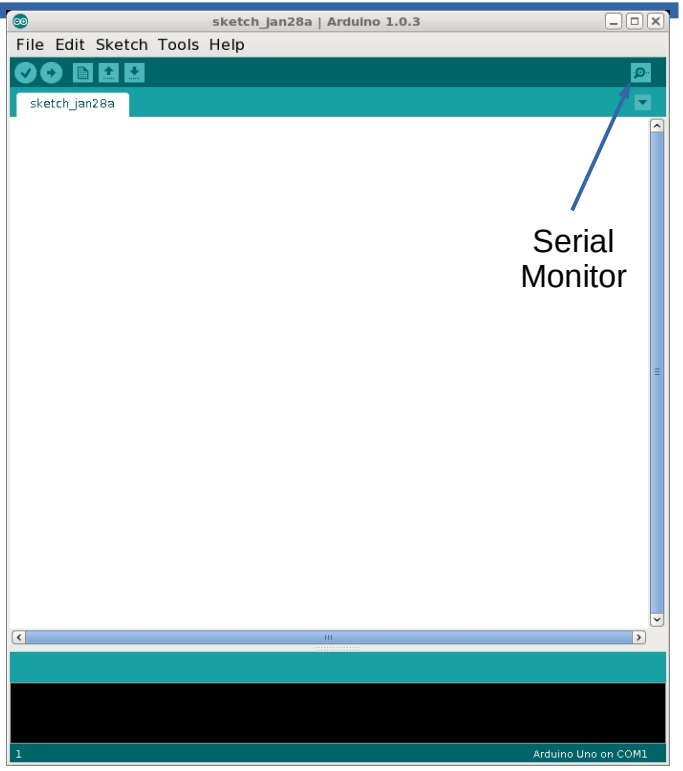
# Arduino Software

- 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.



# Arduino Software

- 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

- 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

# 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");
}
```

# 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");
}
```

# 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.

# Programming Basics

- In the **`setup()`** function, it executes the function **`Serial.begin(115200);`**
- 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);
}
```

# Programming Basics

- 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.print("Hello World");
    delay(500);
}
```

# Variables

- Variables are used to store information. There will be three different types variables used for the robotic arm:
  - int which store integer values also known as whole numbers
  - float which store fractional values, decimal numbers
  - char which store bytes.

# Variables

- Variables can have any name as long as it starts with a letter. It can contain all the letters in upper and lower case, numbers and _.
- No other characters are allowed.
- The C/C++ language is case sensitive meaning that a variable of the same name but different case letters are different variables.
- variable **dog** is a different variable than **Dog**.

```
int a;
float pulse;
char buf;
```

# Variables

- Math operations can be performed on variables.
- Integers and float variables use the same symbols for the math operations.
- Float values and integers can be mixed in equations. Just be careful as the results may not always be as expected.
- To be safe, integers can be type case to float by inserting **(float)** in front of the variable. Floats can be converted to integers by inserting **(int)** in front the variable.

```
int a,b;

float p1,p2;

a = 45;       // assign a value
b = a + 32;   // add two numbers
b = a - 98;   // subtract two numbers
a = b * a;    // multiply
b = 18342 / a; // divide
p1 = 43.6;
p2 = p1 * 3.14159;
p2 = p1 * a;
p2 = (float)a / p1; // convert int
```

# Conditional Statements

- Programs are limited if there is no ability to make decisions.
- The **if()** statement is used to determine how programs function.
- The program to the right creates and assigns values to two variables.
- The **if()** statement has a comparison inside the parentheses. If the comparison is true, the code in the brackets right after is executed. If not, the code in the brackets after else is executed.
- Change the variable values and run the program.

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

void loop() {
  int a = 45;
  int b = 32;
  if(a > b) {
    Serial.println("A is greater");
  } else {
    Serial.println("B is greater");
```

# Conditional Statements

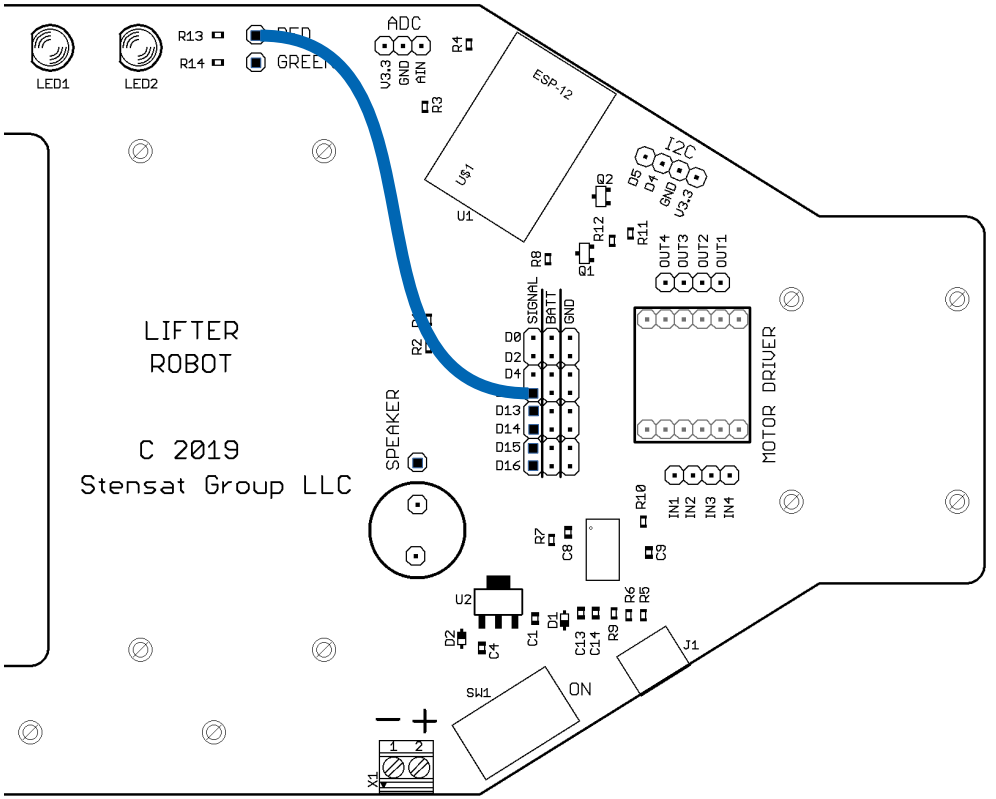
- Listed to the right are the different types of comparisons.
- The comparisons work for both integer and float type variables.

```
int a,b;

a == b        equals
a < b         less than
a > b         greater than
a <= b        less than or equal
a >= b        greater than or equal
a != b        not equal
```

# LED Control

- Take a jumper wire and put one end onto digital pin 12.
- Put the other end of the jumper onto the Red LED pin.
- It does not matter which color jumper is used.



# LED Control

- Start a new program in the Arduino IDE.
- Enter the code as shown.
- In the **setup()** function, the **pinMode()** function configures the digital pin 12 to be an output. This will allow the pin to turn on and off or go high and low.
- In the **loop()** function, the digital pin is set high and low repeatedly using the **digitalWrite()** function. A half second delay is inserted between turning the LED on and off.
- Upload the code and give the program the name **blinky** when asked. After the code uploads, the LED should start blinking.

```
void setup() {
  pinMode(12,OUTPUT);
}

void loop() {
  digitalWrite(12,HIGH);
  delay(500);
  digitalWrite(12,LOW);
  delay(500);
}
```

# LED Control

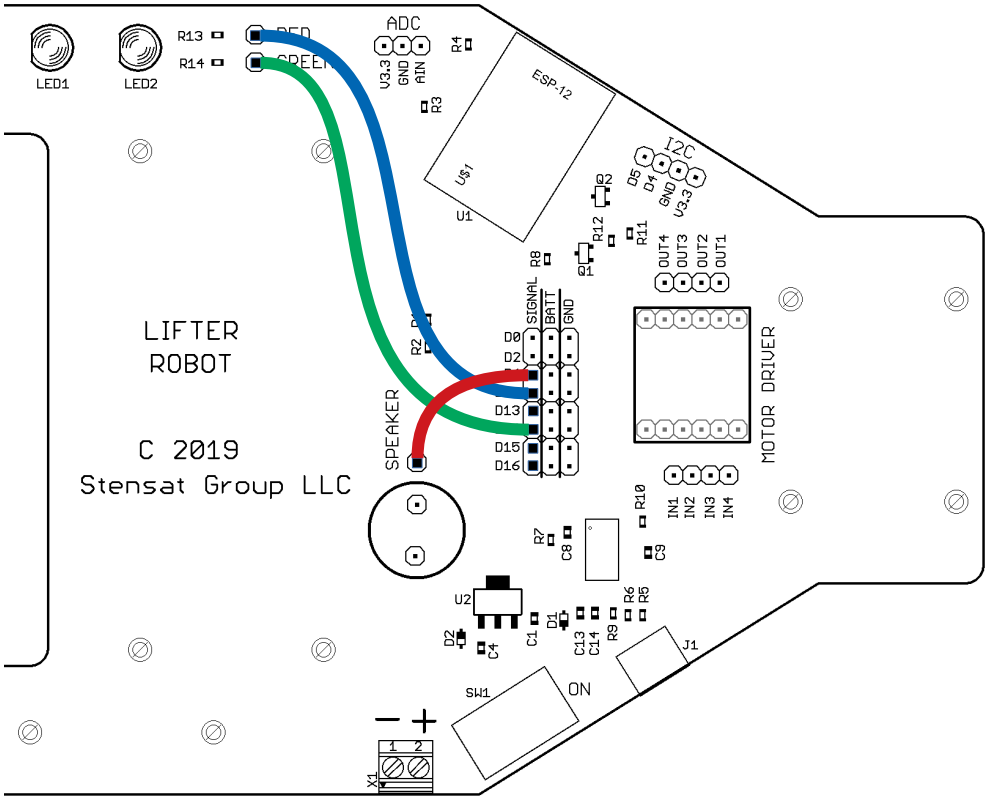
- Connect the green LED to digital pin 13.
- Modify the code to have digital pin 13 blink the green LED opposite to the red LED. When the red LED is on, the green LED is off. When the red LED is off, the green LED is on.
- What did you have to do to the program to make this happen?

```
void setup() {
  pinMode(12,OUTPUT);
}

void loop() {
  digitalWrite(12,HIGH);
  delay(500);
  digitalWrite(12,LOW);
  delay(500);
}
```

# Speaker

- Connect a jumper wire from digital pin 4 to the speaker.
- The speaker is a device that makes sound when a signal is toggled high and low quickly.
- There is a function that does this called **tone()**. Tone requires two parameters, the digital pin number and the frequency in Hertz or Hz.



# Speaker

- Try this program. It toggles between two tones like the last program that toggles between the two LEDs.
- The first tone() function will cause the digital pin 4 to toggle between 3.3 volts and 0 volts 2000 times a second or at a frequency of 2000 Hz.
- The second tone() function will toggle the pin at a frequency of 1500 Hz.
- The frequency range allowed is 30 Hz to 20000 Hz.

```
void setup() {
  pinMode(4,OUTPUT);
}

void loop() {
  tone(4,2000);
  delay(500);
  tone(4,1500);
  delay(500);
}
```

# Speaker

- Musical notes are just different frequencies. A middle **C** note has a frequency of about 261 Hz.
- Open a new program and enter the code to the right.
- The **#define** statement assigns a value to a name. The **#define** statements in this program assign frequency numbers to the musical notes.
- Try the program and replace the notes in the loop to see how the define statements make programming easier.

```
#define C 261
#define D 294
#define E 329
#define F 349
#define G 392
#define A 440
#define B 493
#define C2 523

void setup() {
  pinMode(4,OUTPUT);
}

void loop() {
  tone(4,C);
  delay(500);
  tone(4,C2);
  delay(500);
}
```

# Speaker

- Music can be played similar to musical greeting cards and other toys that play simple tunes.
- Shown to the right is the top part of the program.
- Add another **#define** statement which identifies the number of notes in the song.
- Below that are two arrays of data. The **note** array contains the notes of the song and the **dur** array contains the length of the note. 4 is quarter note and 1 is a whole note.

```
#define C 261
#define D 294
#define E 329
#define F 349
#define G 392
#define A 440
#define B 493
#define C2 523
#define p 7

int note[p] = {C,C,G,G,A,A,G};
int dur[p] = {4,4,4,4,4,4,1};
```

# Speaker

- The **loop()** function gets rewritten. The **setup()** function stays the same.
- The **for()** loop lets a program cycle through specific code marked by the bold brackets for some number of times. There are three parts of the **for()** loop. The first part sets the start condition which in this code declares variable **`i`** and sets it to zero. The second part is a comparison and as long as it is true, the **for()** loop keeps executing. The third part modifies the variable **`i`** which is incremented by 1.
- Once the comparison is not true, the **for()** loop stops executing. This will be indicated by a 1 second pause in the music.

```
void setup() {
  pinMode(4,OUTPUT);
}

void loop() {
  for(int i = 0;i<p;i++) {
    int t = 1000/dur[i];
    tone(4,note[i]);
    delay(t);
    noTone(4);
    delay(20);
  }
  delay(1000);
}
```

# Speaker

- The code in the **for()** loop creates another variable **t** that calculates the duration of the note. A whole note is 1 second long. A quarter note has a value of 4 which divides 1000 ms to 250 ms or quarter second.
- **dur[i]** is the duration. The value of **i** is used to select the duration of the note.
- **tone(4,note[i])** generates the tone based on the value of **note[i]**. The value of **i** selects the current note.
- **note[0]** is the first note in the array. **note[6]** is the last note in the array.

```
void setup() {
  pinMode(4,OUTPUT);
}

void loop() {
  for(int i = 0;i<p;i++) {
    int t = 1000/dur[i];
    tone(4,note[i]);
    delay(t);
    noTone(4);
    delay(20);
  }
  delay(1000);
}
```

# Speaker

- After the note has played. the **noTone(4)** function turns off the note being played. **delay(20)** creates a 20 ms worth of silence to help distinguish between the notes. This is especially useful when the same note is played more than once in sequence.
- Outside the **for()** loop is the one second delay. This indicates the end of the song. The program will repeat the music continuously.

```
void setup() {
  pinMode(4,OUTPUT);
}

void loop() {
  for(int i = 0;i<p;i++) {
    int t = 1000/dur[i];
    tone(4,note[i]);
    delay(t);
    noTone(4);
    delay(20);
  }
  delay(1000);
}
```

# End

- By the end of this section, you should have a basic understanding of writing code and controlling devices such as LEDs and speakers.