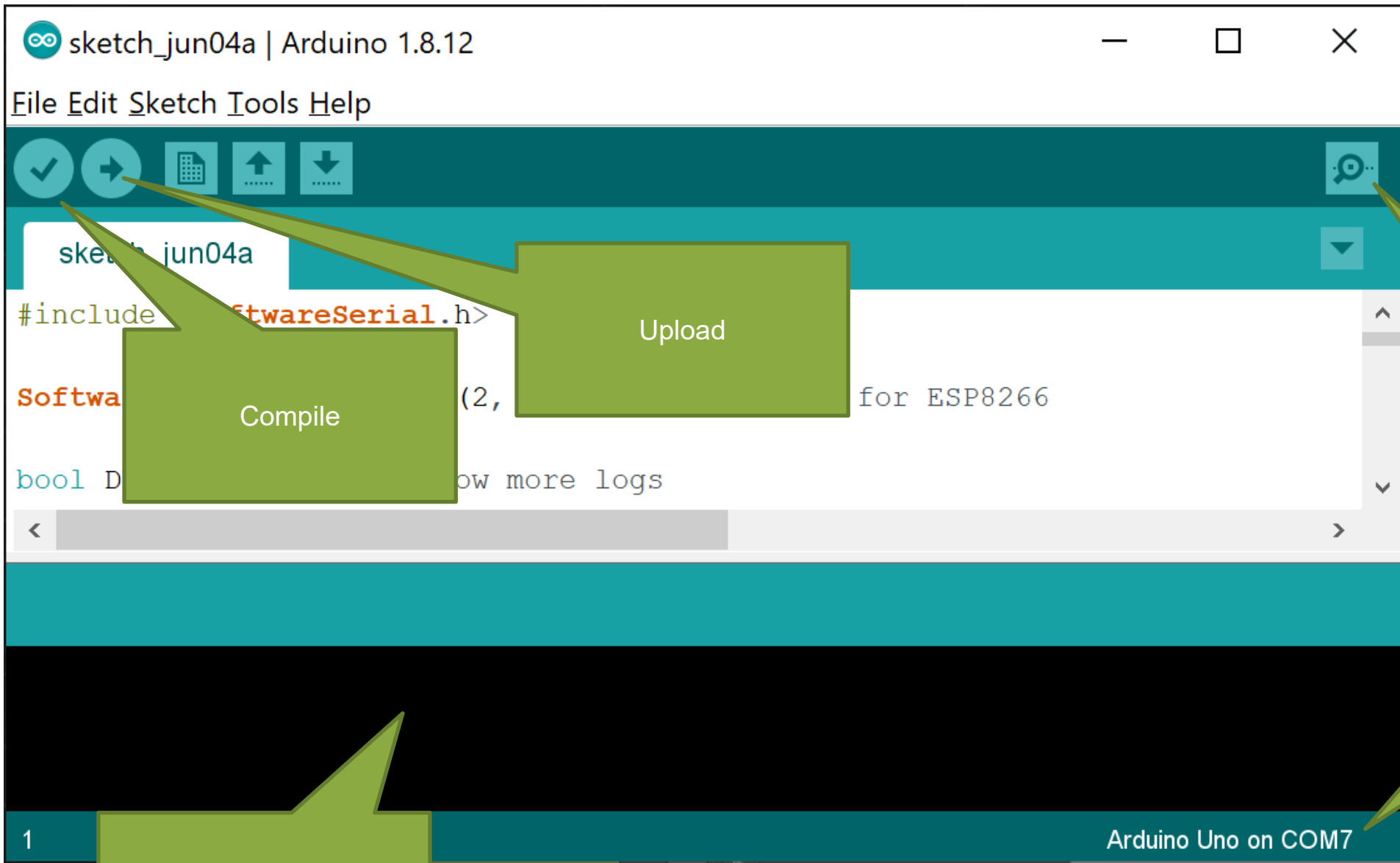


420-N23-LA Introduction to IOT

Arduino Programming, the Arduino Platform and the ESP8266 IC

Arduino Programming



Serial Monitor Shortcut

Virtual COM port used,
and Board Detected

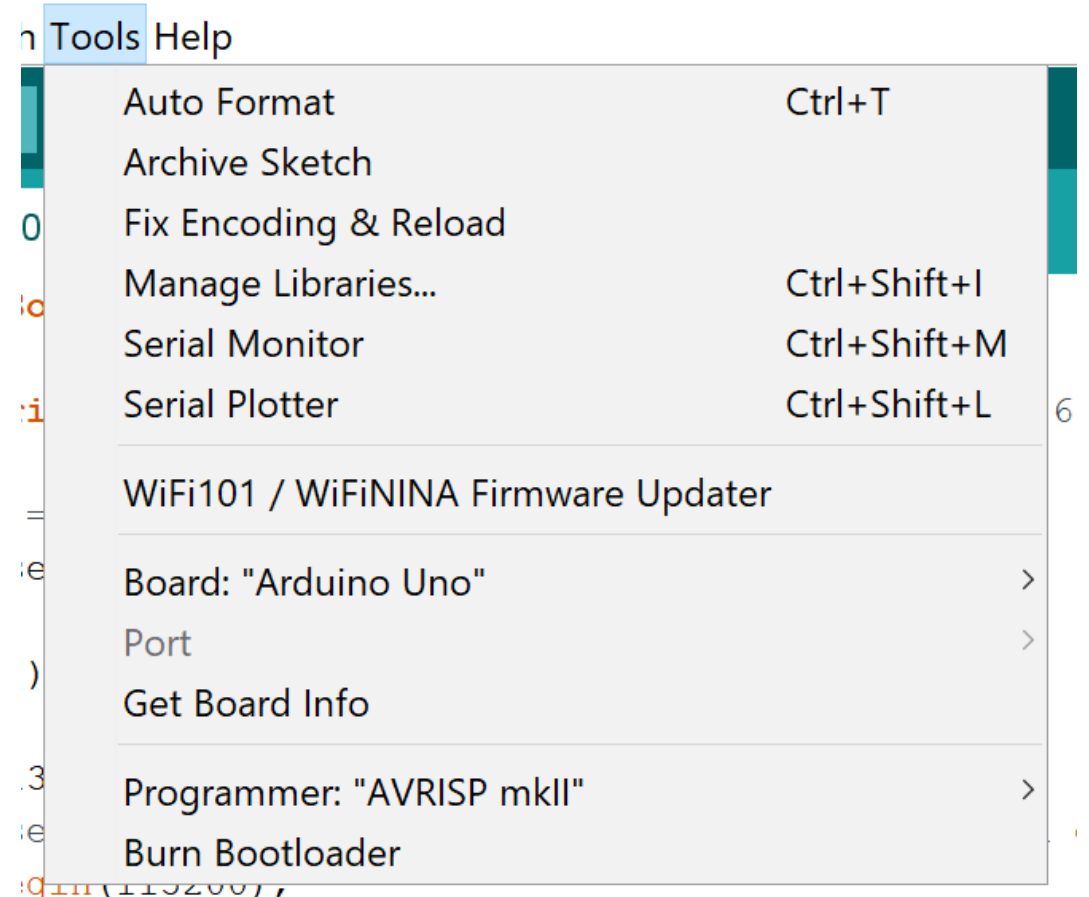
Compilation Logs –
Other Logs

1 Arduino Uno on COM7

Arduino IDE

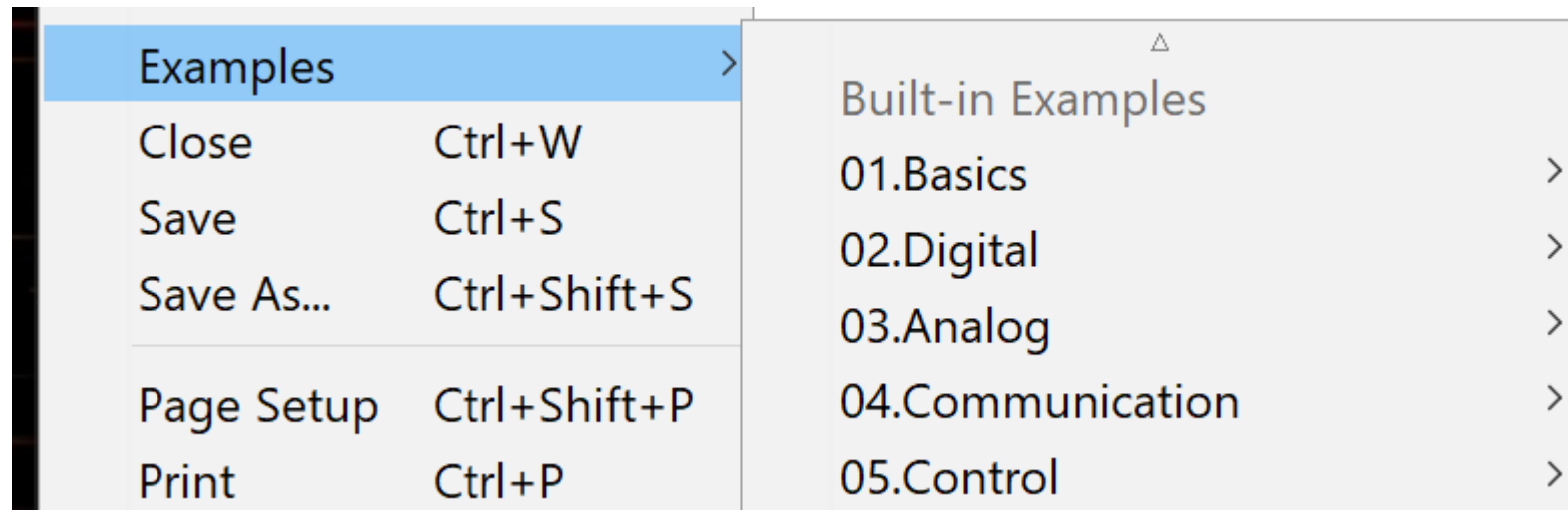
Tools Menu

- Auto Format
 - Formats your code to “beautify” it.
- Manage Libraries
 - Most input-output devices, and extra functions require libraries you can specify here.
- Monitor/Plotter
 - See output text from device. Plotter shows a value changing over time.
- Board/Port
 - It is very important to specify the board you are using otherwise it won't flash properly.



Examples

- Arduino IDE comes with lots of examples.
- Saves you the trouble to Google examples on the web.



Setup and Loop

- You must always have two methods in your code, setup and loop.
- These methods are used to control execution.
 - Setup:
 - Runs once at the start, used for initialization of variables.
 - Loop:
 - The main loop of the program.
 - If not used, include an empty method.

```
void setup() {}  
void loop() {}
```

PinMode

- ▣ Configures the specified pin to behave either as an input or an output.

`pinMode(pin, mode)`

`mode: INPUT, OUTPUT, or INPUT_PULLUP`

digitalWrite()

- Writes a HIGH or a LOW value to a digital pin.
 - If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V for HIGH, 0V for LOW.

digitalWrite(**pin**, **value**)

pin: the Arduino pin number.

value: HIGH or LOW.

Delay()

- ▣ Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

delay(**ms**)

ms: the number of milliseconds to pause.

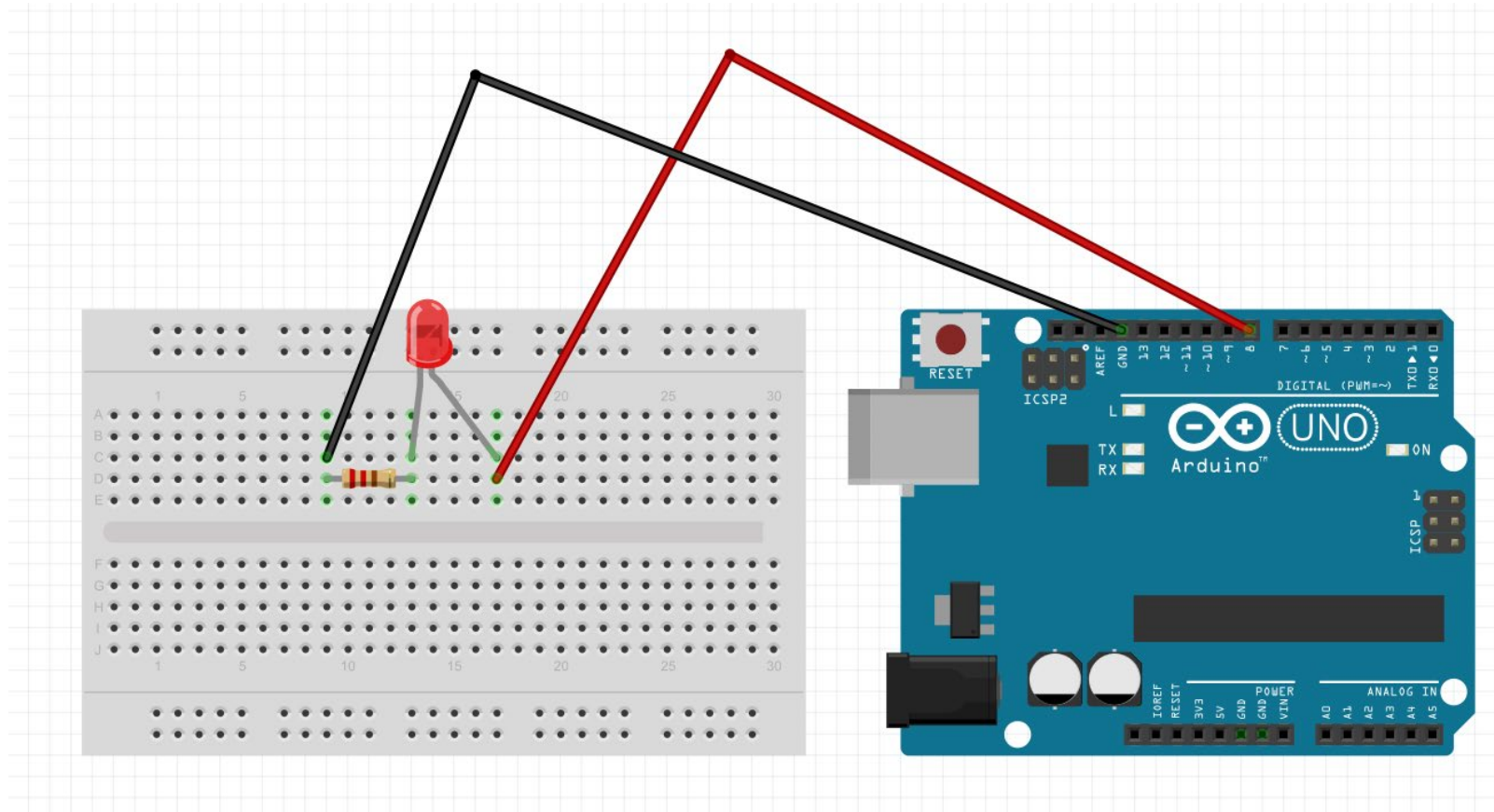
Delays

- Often, we need to pause code execution, it may be showing results too fast.
- Some sensors can't be polled too fast, they may report wrong results (some may even overheat).
- The delay is in milliseconds.

<https://www.arduino.cc/reference/en/language/functions/time/delay/>

```
delay( 500)
```

Example: Blinking LED



Blinking Example

This is the built-in LED

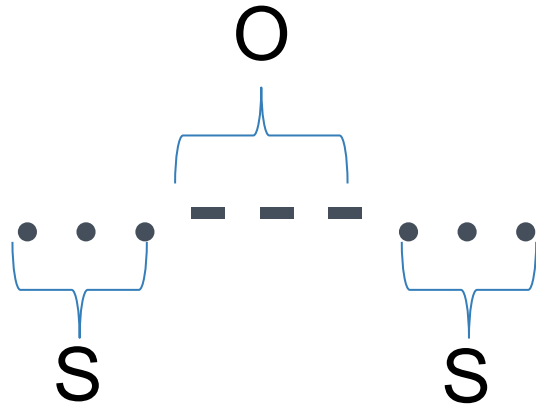
ŒÁÝ□‡] ∂pŒÁ 1/3□ ۞□□□□□□□□□□□□ ! □W5 1 □∞ÓÁÁ] ∞Ý] ∂ □ÝÓ□∂Œ≈ŒÝ£ ‡ □° ŒÁ□

ŞÓŒ □Š] Ý×° ى□۞
° ŒÁ] Ó∂] و ‡] ∂pŒÁف □dÉ} pÉ} ى□□□□ ! □Š] ÝŠ □ÝÀ] □∂Œ≈ŒÝ£ ‡ □° ŒÁ□£ Š □Ó×Ý° ×Ý
ھ

ŞÓŒ □‡ ÓÓ° ى□۞
∂Œ≈ŒÝ£ ‡ è, ŒÝ] و ‡] ∂pŒÁف □EH@E ى□□□ ! □Š] ÝŠ □ÝÀ] □W5 1 □ÓÁ
∂] ‡£- آ و ء ء ء ى□□□□□□□□□□□□□□□□□□□□□□□□ ! □\$ £ŒÝŠ □f Ó□ □£ □Š] ∞ÓÁ∂
∂Œ≈ŒÝ£ ‡ è, ŒÝ] و ‡] ∂pŒÁف □Wdè ى□□□ ! □Š] ÝŠ □ÝÀ] □W5 1 □Óf f
∂] ‡£- آ و ء ء ء ى□□□□□□□□□□□□□□□□□□□□□□□□ ! □\$ £ŒÝŠ □f Ó□ □£ □Š] ∞ÓÁ∂
ھ

Example: Morse Code Program

- Blink out SOS on a LED



Including Necessary Libraries

- You must include libraries to help you talk to certain devices, or give you extra commands.
 - Example: Temperature sensor, LCD Display, JSON Parser.
- This follows the “C” standard for including libraries.
- Names are usually filename.h (.h means header file).

```
#include <dht.h>
```

Serial Monitor

- The Arduino software (IDE) comes with a tool called **serial monitor** that enables you to report back results from microcontroller.
- Using the serial monitor you can get information about status of your sensor, and get an idea about what is happening in your circuit and code as it runs.

- w| , ⒼⓈ ‡ ك∅ ≈ ⒼⓈ' وئى لى ء ء ء

- 17

Sending the Information from Arduino to Serial Monitor

- The following command sends information from the Arduino to a connected computer. You can see this information in Serial monitor.
 - If you give `Serial.print()` an argument in quotation marks, it will print out the text you typed.
 - `Serial.println("Hello World!");`
 - If you give it a variable as an argument, it will print out the value of the variable.
 - `Serial.println(sensorVal);`

Example Code

```
int ledPin = 8;
int ledDelay=1000;
void setup() {
  // initialize digital pin LED_BUILTIN as
  // an output.
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}
```

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(ledPin, HIGH); // turn the LED on
  Serial.println("LED ON");
  delay(ledDelay);           // wait for a second
  digitalWrite(ledPin, LOW); // turn the LED off
  Serial.println("LED off");
  delay(ledDelay);           // wait for a second
}
```

Analog & Digital Read and Write

Arduino Input

- The pins on the Arduino can be configured as either inputs or outputs.
- The majority of Arduino analog pins, may be configured, and used, in exactly the same manner as digital pins.
- The default state of a pin is INPUT.

Analog I/O

- The Arduino can not vary the output voltage on its pins, it can only output 5v.
- The **Arduino does** not **have** a built-in digital-to-**analog** converter (DAC).
- It uses a technique called **pulse-width modulate** (PWM) to achieve some of the functions of an **analog output**.
- PWM rapidly turns the output pin high and low over a fixed period of time.

Analog OutPut Pins

- The Arduino has six pins set aside for PWM, they can be identified by the ~ next to their number on board:
 - Digital pins 3, 5, 6, 9, 10 and 11
- The PWM pins are 8-bit pins, meaning that you can set the duty cycle somewhere between 0 -255.

analogWrite() Command

- analogWrite command writes an analog value ([PWM wave](#)) to a pin.
- The analogWrite comes handy when you plan to control the motor speed or the intensity of any [LED](#).

analogWrite Syntax

- The syntax of analogWrite is given as follows:

analogWrite(int pin, int value);

- where:
 - pin: The arduino pin to write to.
 - value: The duty cycle between 0 (always off) and 255 (always on).

Controlling the Motor Speed with analogWrite:

- when controlling the motor speed with analogWrite:
 - The value you write on the PWM pins will control the speed.
 - For example, if you intend to run the motor at full speed, you will set the value 255 i.e. the maximum value it can handle that will ultimately run the motor at full speed.
 - Similarly, setting value as "0" will be sending no signal and motor won't start.
 - And if the motor requires to be run at half speed, then you will set the value 127 or 128 - half of the maximum value that will cause the motor to be running at half speed.

analogRead Command

- The analogRead reads the voltage value from the specified analog pin.
- The analogRead is a command mainly used to program the analog pins on the board.
- If you are using analogRead functions, it indicates you are making the pins as input
 - i.e. you can connect the Arduino analog pins with any [sensor](#) and read its value by making the analog pins as input.

analogRead

- Arduino has 6 input Analog pins: A0, A1, A2, A3, A4, A5
- Analog pins are 10-bit pins. It means each pin can store 0 - 1023 values.
- The voltage values are directly proportional to the values stored in the Arduino Pins.
 - For example, if the [sensor](#) voltage is around 2.5 V then the value we get on an analog pin will be half the total value it can store in the pin i.e. 512

analogRead Syntax

- The syntax of analogRead is given as follows:

int data = analogRead(int pin);

- **where:**

- Pin defines the number of a pin you are targeting.

- **Return:** analogRead returns value anywhere between 0 to 1023 depending on the voltage it gets from the pin.

map()

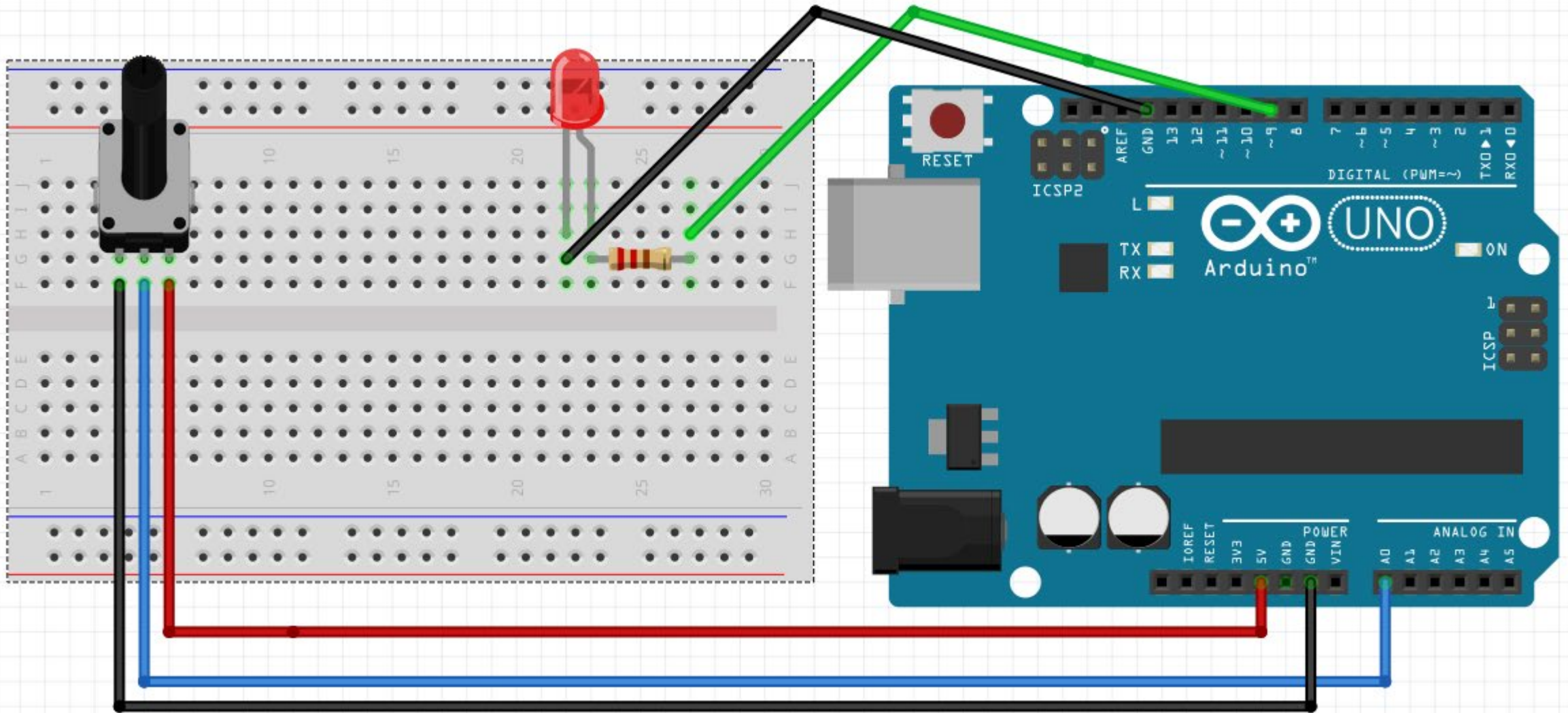
- Re-maps a number from one range to another. That is, a value of **fromLow** would get mapped to **toLow**, a value of **fromHigh** to **toHigh**, values in-between to values in-between, etc.
- The syntax of map is given as follows:

map(value, fromLow, fromHigh, toLow, toHigh)

Example: Analog In

- In this example you built a circuit that is able to change the brightness of an LED by using a PWM signal. This will build upon the previous example, adding an element of user input by allowing manual control over the brightness of the LED. To accomplish this, you will use a component called a potentiometer. Using this component, we can create a voltage divider circuit that can indirectly control the brightness of our LED.
- This example shows you:
 - How to read an analog input pin
 - Map the result to a range from 0 to 255,
 - Use that result to set the pulse width modulation (PWM) of an output pin to dim or brighten an LED
 - Print the values on the serial monitor of the Arduino Software (IDE).

Example: Analog In Diagram



Example: Analog In



```
const int analogInPin = A0; // Analog input pin that
the                          potentiometer is
attached to

const int analogOutPin = 9; // Analog output pin that
the                          LED is attached
to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM
(analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}
```

```
void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the Serial Monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);
  // wait 2 milliseconds before the next loop for the
  //analog-to-digital converter to settle after the last
  reading:
  delay(2);
}
```

Getting Input From the Serial Monitor

Getting input from the Serial Monitor

- Prompt the user for information
 - Use Serial.print to prompt the user
 - “How many times do you want the LED to blink?”
 - “Choose an option from the Menus?”
- The input can be : int, float or String
- Wait for the user to input the value
 - Use Serial.available() as the condition of an empty while loop.
- Read the information input from the user
 - Parse the data in the serial buffer

Getting input from the Serial Monitor

- When the user inputs data and press the “Enter”, Serial.available() returns a non-zero value.
- Use a while loop to make the program wait for an input from the user.

```
while (Serial.available() != 0 ){  
  
}
```

Parse the information in the Serial Buffer

- functions to parse the information in the Serial Buffer :
 - Serial.parseInt()
 - Serial.parseFloat()
 - Serial.readString()
- e.g. to parse an integer

```
int integerValue=Serial.parseInt();
```

Example: Turn LED ON/OFF using User Input

```
int input;  
int ledPin=8;  
  
void setup() {  
  Serial.begin(9600);  
  pinMode(ledPin, OUTPUT);  
}
```

```
void loop() {  
  Serial.println("Please enter 1 to turn LED ON and 0 to  
  turn LED OFF: ");  
  // Wait for the user Input  
  while(Serial.available() == 0){  
  }  
  // Read the input from the user  
  input = Serial.parseInt();  
  if ( input == 1 ) {  
    Serial.println ("LED ON");  
    digitalWrite (ledPin , HIGH );  
  }  
  else{  
    Serial.println ( "LED OFF" );  
    digitalWrite ( ledPin, LOW );  
  }  
}
```

