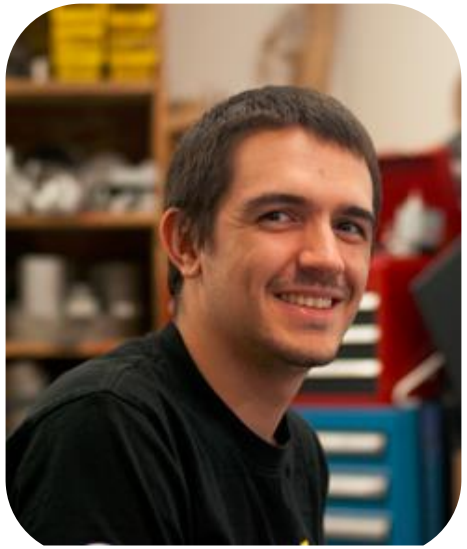
Measurements, Sensors and Data Logging Course

Week 1

Introductions

Who We Are

- Jordan Krell
  - FRSEF Fair Director
  - Motorsport Engineer @ Bosch Engineering
- Chris Bergsneider
  - Motorsport Engineer @ Bosch Engineering
- Kathy Dutton
  - FRSEF Outreach, Marketing, SRC
  - Teacher @ Homeschool Connections



Who We Are

- Tracy Zhang
  - Outreach Specialist @ MSU St. Andrews
- John Furcean
  - Outreach Specialist @ MSU St. Andrews
- Clare Light
  - Project/Event Coordinator @ MSU St. Andrews

FLINT REGIONAL SCIENCE & ENGINEERING FAIR

Who We Are

- Nick Henton
  - Research and Instructional Technologist @ MSU St. Andrews

- Ivan Lysenko
  - Research and Instructional Technologist @ MSU St. Andrews

Safety

- Your safety is paramount
  - Please read the virtual handbook
  - If at any point you feel unsafe, logout immediately and contact the FRSEF.

- Contact us in an Emergency or for Routine Assistance
  - Jordan: 248-330-4269 jkrell@flintsciencefair.org
  - FRSEF: 810-797-5290 kdutton@flintsciencefair.org

Arduino Hardware and IDE

Introduction to the Arduino Integrated Development Environment and Arduino Language

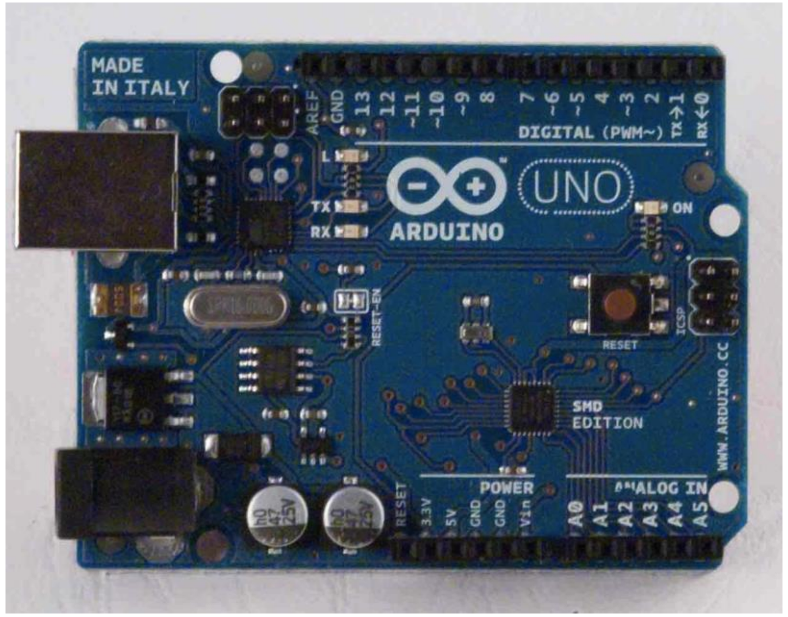
Arduino Hardware – add pic

What is an Arduino?

- “Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects. Arduino senses the environment by receiving inputs from many sensors, and affects its surroundings by controlling lights, motors, and other actuators.” – Arduino Website

Find more info:

https://www.arduino.cc/en/guide/introduction



Arduino Hardware

Microcontroller

What is a Microcontroller?

- A microcontroller, or MCU, is a miniature computer with a processor, RAM, non-volatile storage and peripherals.
- The peripherals can include general purpose input/output (GPIO), communications (serial, I2C, etc.), timers, interrupts, PWM, and analog functions (ADC).

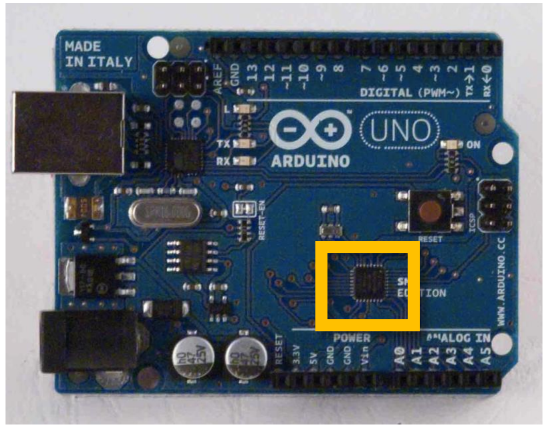
Where are Microcontrollers used?

- Microcontrollers are used in embedded applications, such as appliances, vehicles, industrial controls, wireless devices, and much more.
- The Arduino boards are based on a microcontroller, an ATmega328P in our case.
- Activity: can you name an object near you that has a microcontroller?

Find more info:

- https://en.wikipedia.org/wiki/Microcontroller
- https://www.allaboutcircuits.com/technical-articles/what-is-a-microcontroller-introduction-component-characteristics-component/
- https://www.microchip.com/wwwproducts/en/ATMEGA328P

Image from https://www.microchip.com/wwwproducts/en/ATMEGA328P



Grove Beginner's Kit Hardware

The Grove Beginner’s Kit is a variant of the Arduino Uno, with many sensors and output modules included and pre-wired.

It includes:

- Seeeduino Lotus (Arduino Uno derivative)
- Grove LED . . . . (D4 Output)
- Grove Buzzer . . . . (D5 Output)
- Grove OLED Display . . . . (I^2C Bus)
- Grove Button . . . . (D6 Input)
- Grove Rotary Potentiometer . . . . (A0 Analog Input)
- Grove Light Sensor . . . . (A6 Analog Input)
- Grove Sound Sensor . . . . (A2 Analog Input)
- Grove Temp and Humidity Sensor . . . (D3 I/O)
- Grove Air Pressure Sensor . . . . (I^2C Bus)
- Grove 3-axis Accelerometer . . . . (I^2C Bus)

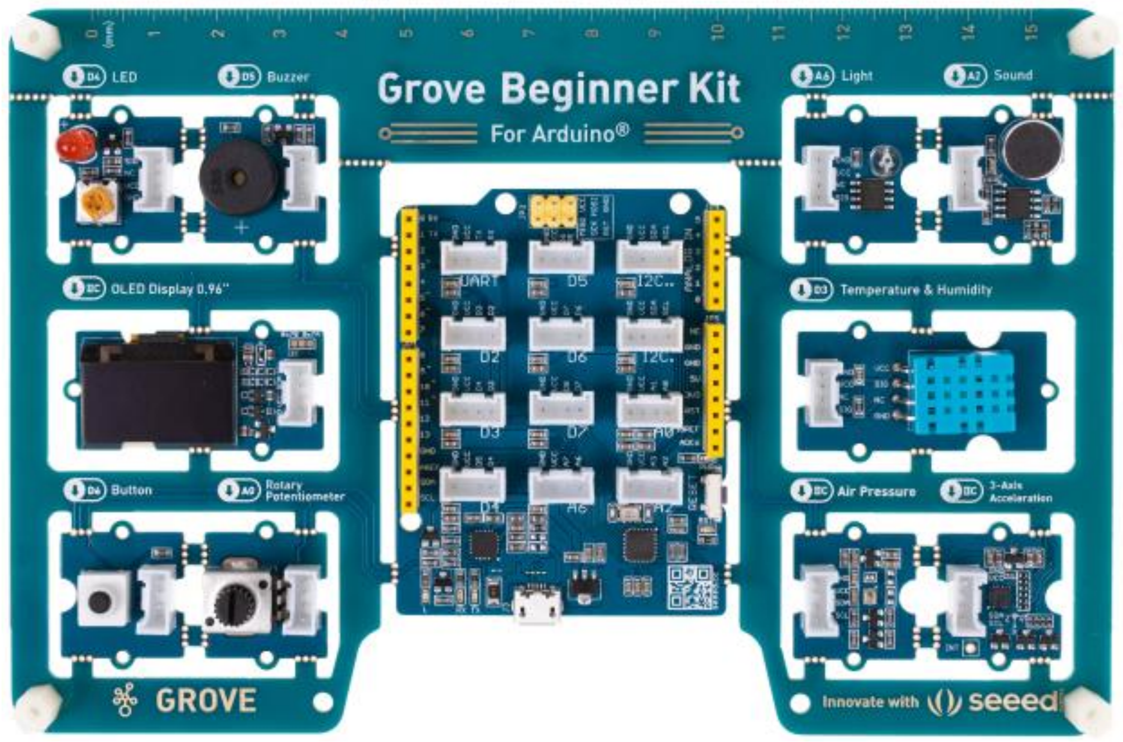


Image from https://www.seeedstudio.com/Grove-Beginner-Kit-for-Arduino-p-4549.html

Find more info:

- https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf
- https://www.seeedstudio.com/Grove-Beginner-Kit-for-Arduino-p-4549.html

Example Connection D4 LED

- Grove LED: (D4 Output)

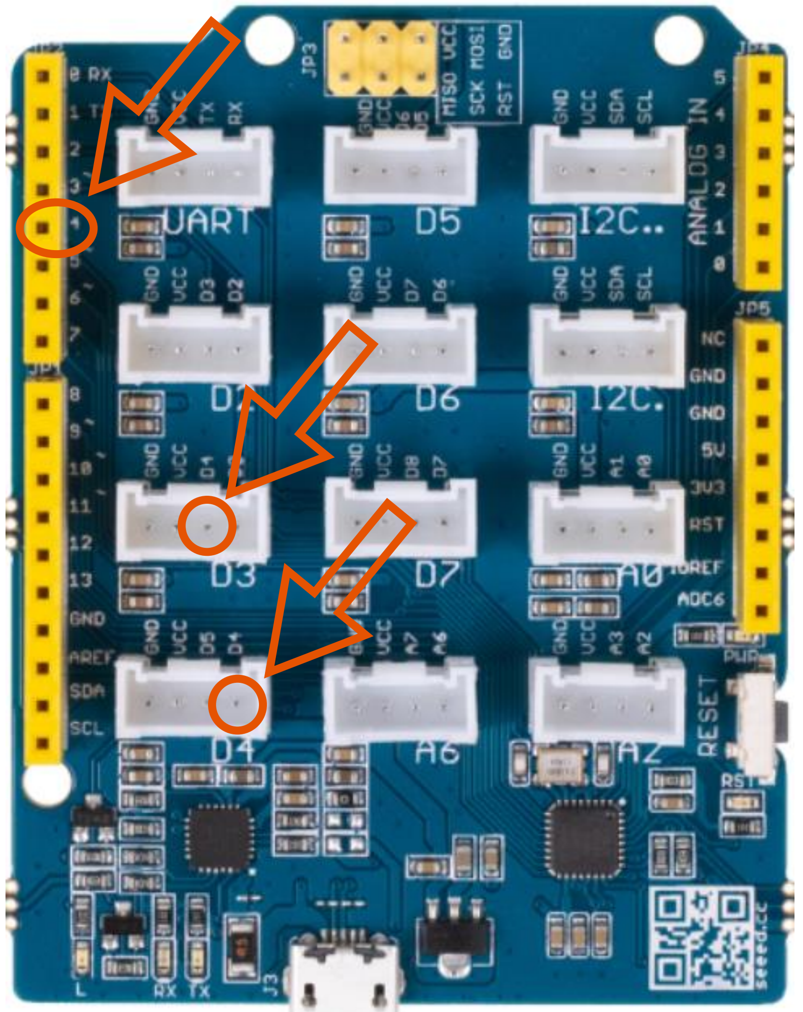


Image from https://www.seeedstudio.com/Grove-Beginner-Kit-for-Arduino-p-4549.html

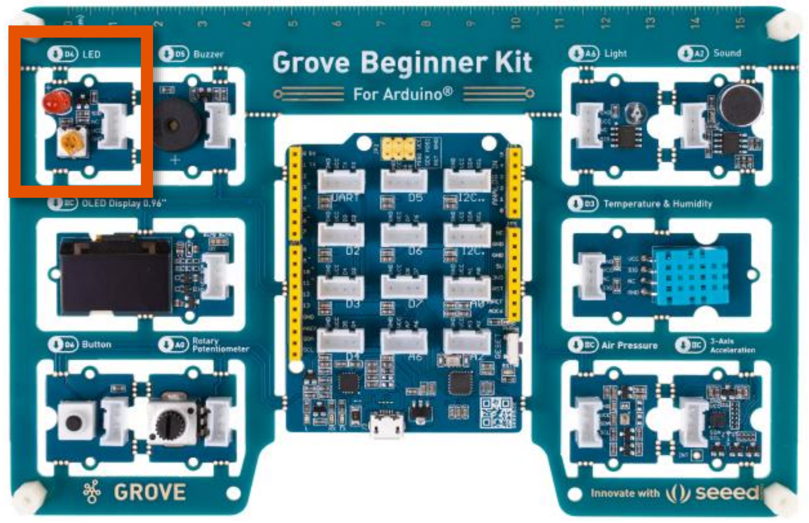


Image from https://www.seeedstudio.com/Grove-Beginner-Kit-for-Arduino-p-4549.html

Grove Beginner's Kit Hardware

- Grove Sound Sensor : A2 Analog Input

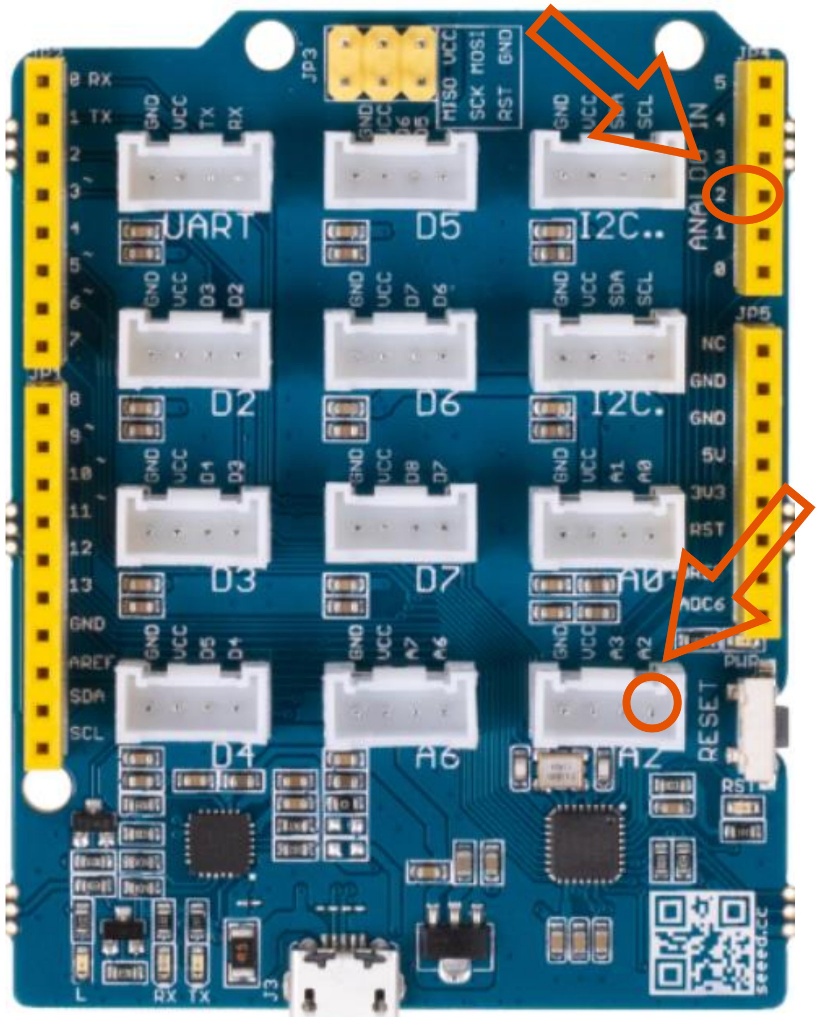


Image from https://www.seeedstudio.com/Grove-Beginner-Kit-for-Arduino-p-4549.html

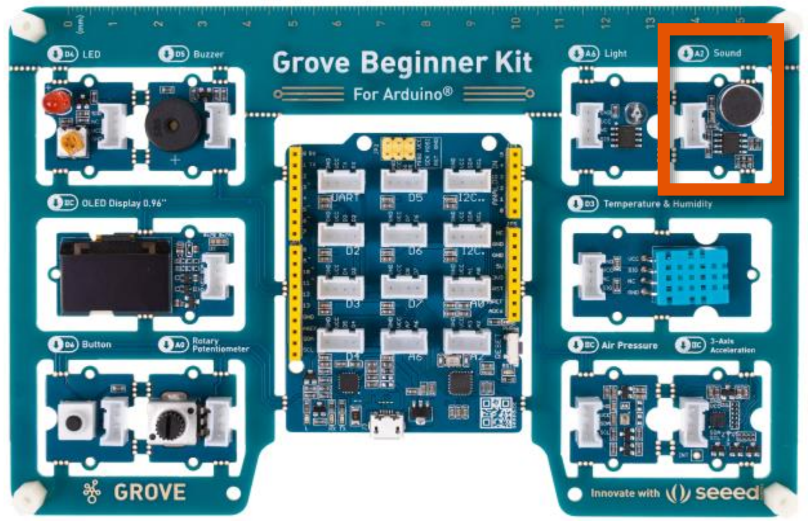


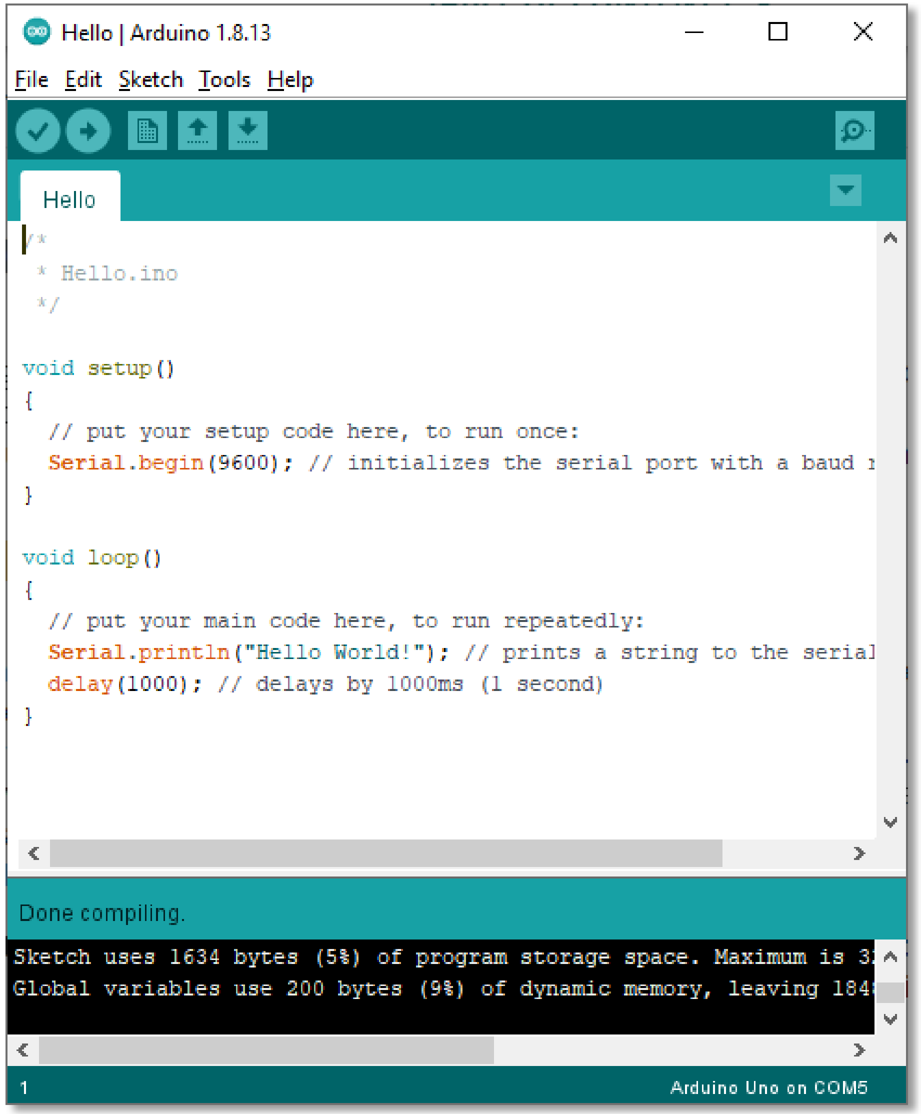
Image from https://www.seeedstudio.com/Grove-Beginner-Kit-for-Arduino-p-4549.html

Arduino IDE

The Arduino IDE is an **Integrated Development Environment** for creating programs for Arduino, called **Sketches**. It contains all the necessary functions such as a text editor, message area, text console, toolbars, compiler and drivers to write, compile and upload your sketches. It also has a few useful features that makes writing and testing your sketches easier.

Find more info:

https://www.arduino.cc/en/Guide/Environment



```
/*
 * Hello.ino
 */

void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600); // initializes the serial port with a baud r
}

void loop()
{
  // put your main code here, to run repeatedly:
  Serial.println("Hello World!"); // prints a string to the serial
  delay(1000); // delays by 1000ms (1 second)
}
```

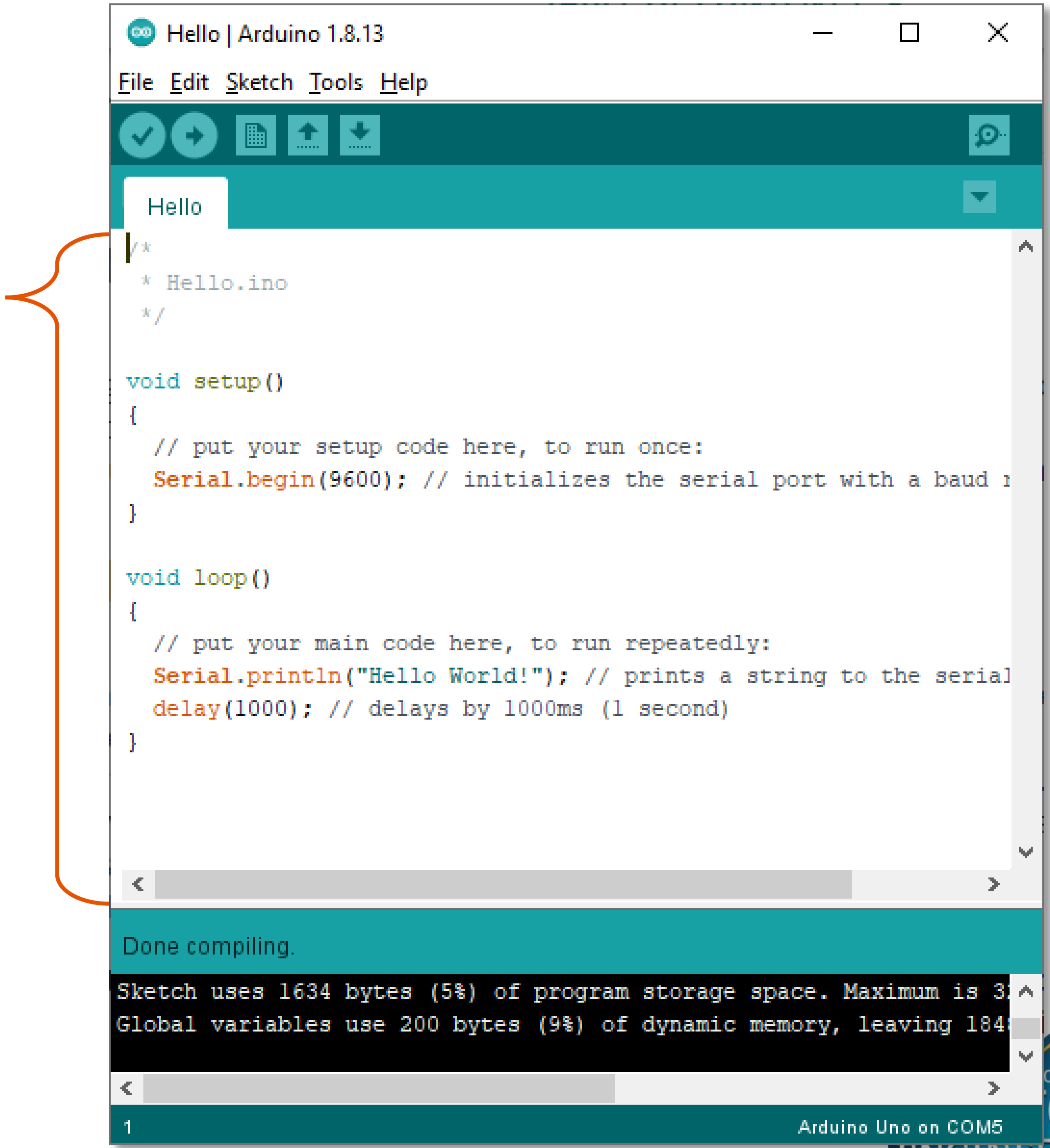
```
Sketch uses 1634 bytes (5%) of program storage space. Maximum is 3
Global variables use 200 bytes (9%) of dynamic memory, leaving 184
```

Text Editor

Arduino IDE

Text Editor

- This is where you write your code. It has features for cut/copy/paste, find/replace text, commenting, indenting, highlighting, etc. It makes a basic but competent code editor.



```
/*
 * Hello.ino
 */

void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600); // initializes the serial port with a baud r
}

void loop()
{
  // put your main code here, to run repeatedly:
  Serial.println("Hello World!"); // prints a string to the serial
  delay(1000); // delays by 1000ms (1 second)
}
```

Done compiling.

```
Sketch uses 1634 bytes (5%) of program storage space. Maximum is 3
Global variables use 200 bytes (9%) of dynamic memory, leaving 184
```

Message Area and Console

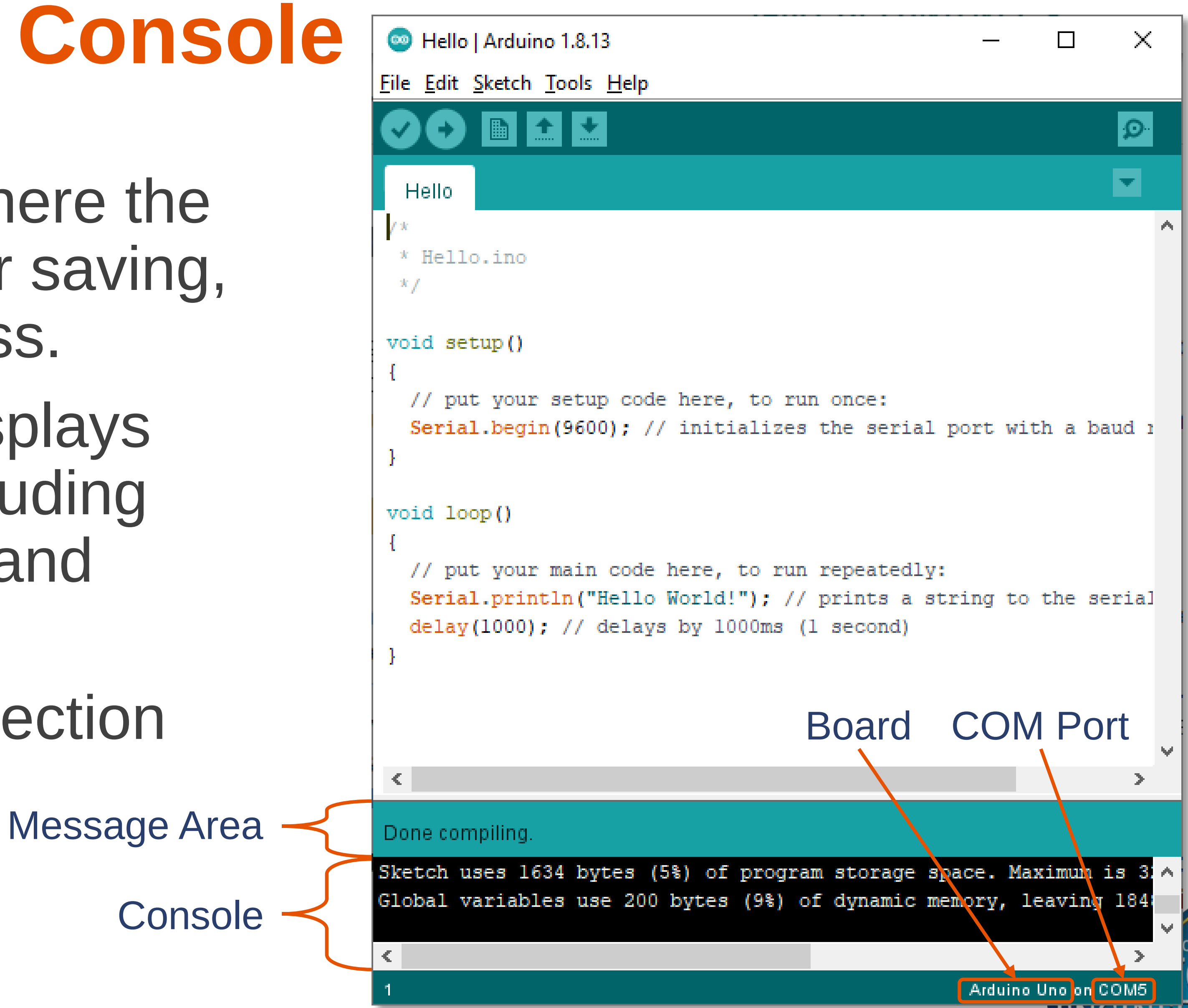
Arduino IDE

Message Area - This is where the IDE gives you feedback for saving, errors and compile progress.

Console - The console displays text output by the IDE, including complete error messages and other information.

Board – Current board selection

COM Port – Current COM port selection



```
/*
 * Hello.ino
 */

void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600); // initializes the serial port with a baud r
}

void loop()
{
  // put your main code here, to run repeatedly:
  Serial.println("Hello World!"); // prints a string to the serial
  delay(1000); // delays by 1000ms (1 second)
}
```

Done compiling.

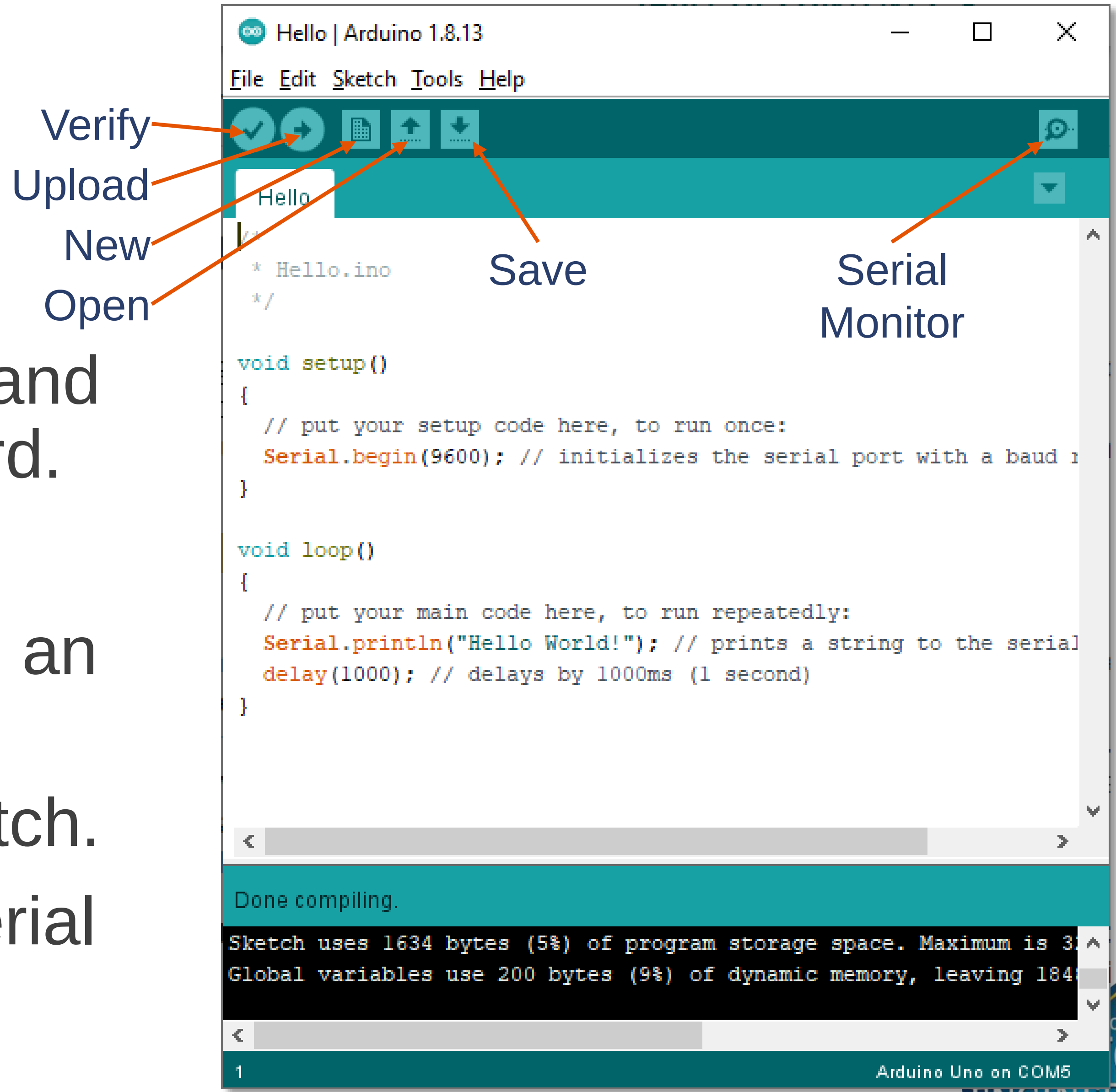
```
Sketch uses 1634 bytes (5%) of program storage space. Maximum is 3
Global variables use 200 bytes (9%) of dynamic memory, leaving 184
```

Some text copied directly from sites listed in the for more info

Toolbar Buttons

Arduino IDE

- ***Verify*** – Compiles your code to check for errors.
- ***Upload*** – Compiles your code and uploads it to your selected board.
- ***New*** – Creates a new sketch.
- ***Open*** – Opens a menu to open an existing sketch in this window.
- ***Save*** – Saves your current sketch.
- ***Serial Monitor*** – Opens the Serial Monitor.



```
/*
 * Hello.ino
 */

void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600); // initializes the serial port with a baud r
}

void loop()
{
  // put your main code here, to run repeatedly:
  Serial.println("Hello World!"); // prints a string to the serial
  delay(1000); // delays by 1000ms (1 second)
}
```

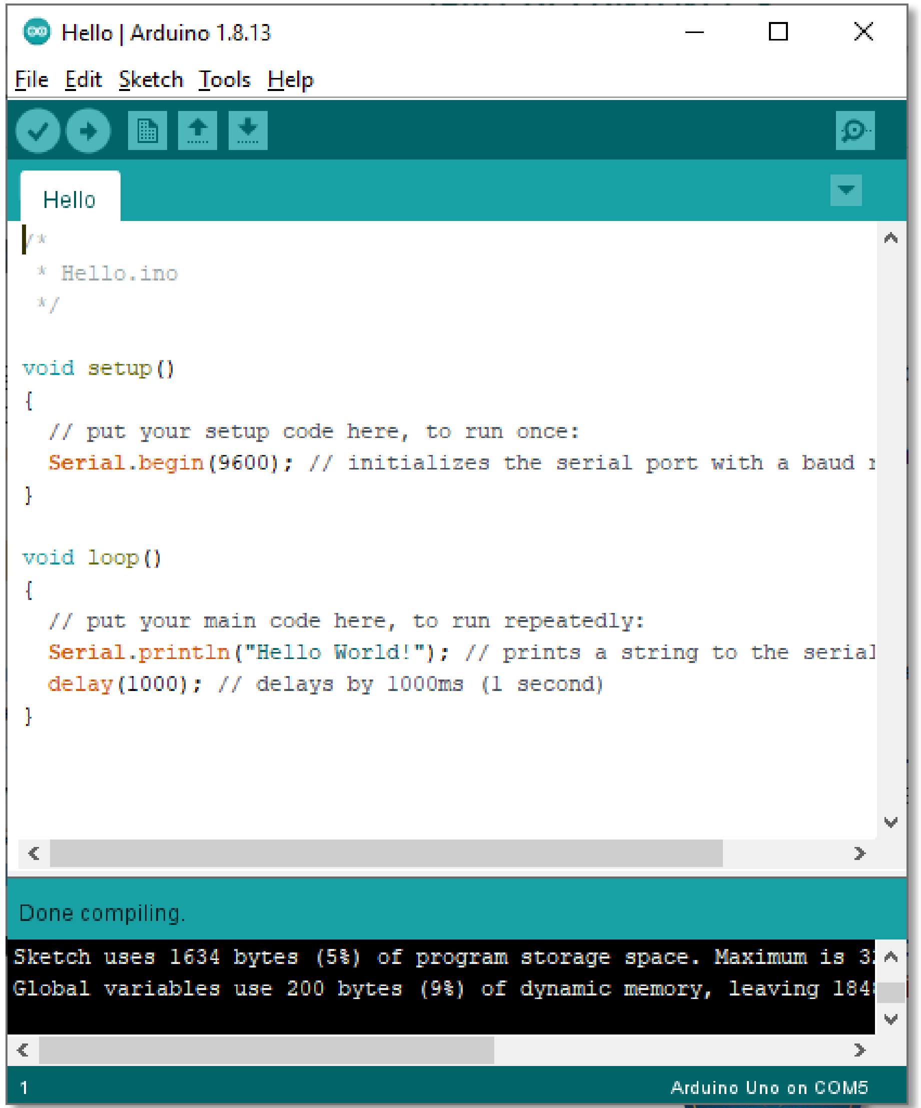
Done compiling.

```
Sketch uses 1634 bytes (5%) of program storage space. Maximum is 3
Global variables use 200 bytes (9%) of dynamic memory, leaving 184
```

1 Arduino Uno on COM5

Arduino IDE

- **Verify (Compile):** Converts your sketch into a program that can be used by the Arduino
- **Upload:** Process of sending the program to the Arduino
  - I often call it ***Flash***



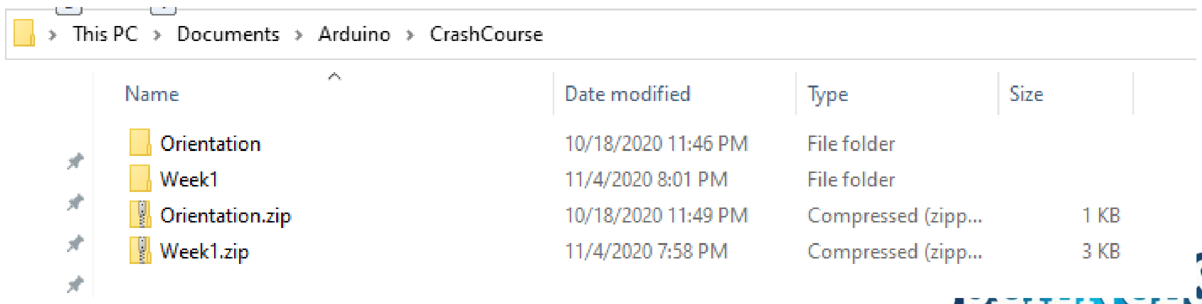
```
/*
 * Hello.ino
 */

void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600); // initializes the serial port with a baud r
}

void loop()
{
  // put your main code here, to run repeatedly:
  Serial.println("Hello World!"); // prints a string to the serial
  delay(1000); // delays by 1000ms (1 second)
}
```

Done compiling.

```
Sketch uses 1634 bytes (5%) of program storage space. Maximum is 3
Global variables use 200 bytes (9%) of dynamic memory, leaving 184
```

Week 1 Prerequisites

- Download the week 1 sketches zip from the course website:
- https://www.flintsciencefair.org/students/crash-course-measurements-sensors-and-data-logging/
- Unzip them into your Sketchbook folder:
  - Windows: C:\Users\<YOUR_USERNAME>\Documents\Arduino\FRSEF_Crash_Course
  - Mac: /Users/<YOUR_USERNAME>/Documents/Arduino/FRSEF_Crash_Course
  - Chromebook: Arduino\CrashCourse\

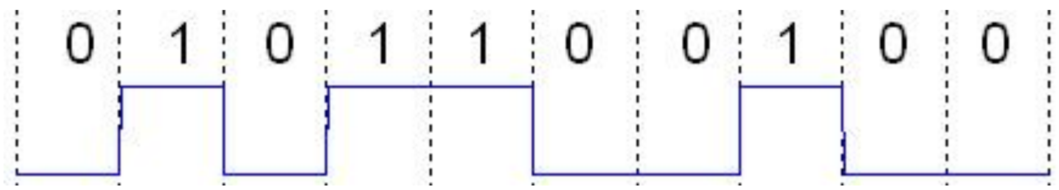


This PC › Documents › Arduino › CrashCourse

Name	Date modified	Type	Size
Orientation	10/18/2020 11:46 PM	File folder	
Week1	11/4/2020 8:01 PM	File folder	
Orientation.zip	10/18/2020 11:49 PM	Compressed (zipp...	1 KB
Week1.zip	11/4/2020 7:58 PM	Compressed (zipp...	3 KB

Digital Signal Introduction

- What is a digital signal?
  - A digital signal is a signal that can only have one of a finite number of values at a given time.
  - For most digital electronics (including the Arduino), this value is Binary, either 1 (HIGH) or 0 (LOW).
  - For us this means that a digital signal going into or coming from our MCU is either
    - a HIGH signal, near 5V (supply voltage), or
    - a LOW signal near 0V (ground).
- Activity: Can you name a device that uses digital signals in your house?
- More Info:
  - https://en.wikipedia.org/wiki/Digital_signal
  - https://www.allaboutcircuits.com/textbook/digital/



By El pak at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20667098

`digitalRead()` Function

```
digitalRead(buttonPin);
```

- Gets the current state of the buttonPin.

- The digitalRead function reads the value from a specified digital pin.
- Syntax:

```
    digitalRead(pin);
```

  - Pin: Arduino pin number to read
  - Returns:
    - `HIGH`
    - `LOW`
- More information:
  - https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread/

`digitalWrite()` Function

```
digitalWrite(ledPin, HIGH);
```

- Set the ledPin as HIGH.

- Writes a `HIGH` or `LOW` value to the specified pin.
- Syntax:

```
    digitalWrite(pin, value);
```

  - Pin: Arduino pin number to set the value of
  - Value: options are
    - `HIGH`, set pin to a high (5V) value
    - `LOW`, set pin to a low (0V) value
- For most applications `pinMode(pin, OUTPUT)` function should be run on the desired pin before calling `digitalWrite()`
- More information:
  - https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

Analog Signal Introduction

- What is an analog signal?
  - An analog signal is a continuous, time varying signal.   It can be any of an infinite number of values within its amplitude range.
  - Analog signals can be measured with digital electronics with the help of an Analog-to-Digital Converter (ADC).
  - Activity: Can you name a device that uses analog signals in your house?
- More Info:
  - https://en.wikipedia.org/wiki/Analog_signal
  - https://www.elprocus.com/differences-between-analog-signal-and-digital-signal/

Difference between Analog and Digital Signals

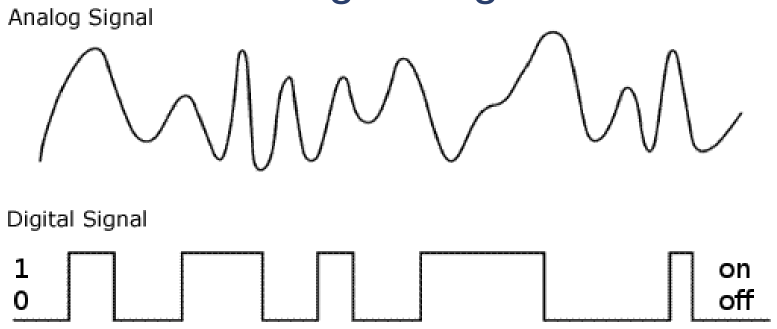
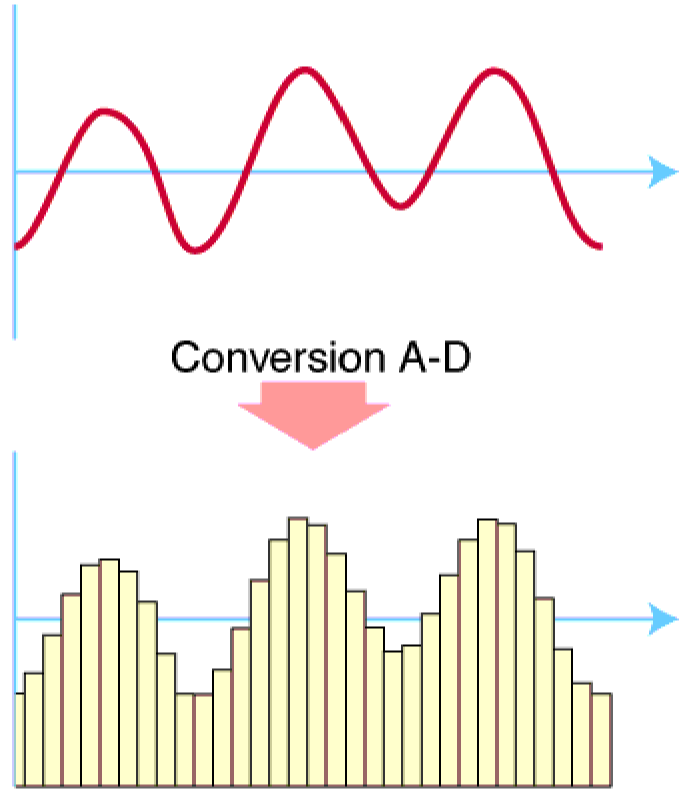


Image from https://36projectsblog.wordpress.com/9-raspberry-pi-analog-to-digital-conversion-using-mcp3008-adc/

ADC (Analog to Digital Converter) introduction

- What is an ADC?
  - An ADC is an electronic circuit that converts from a continuous analog signal to a discrete digital signal.
    - *Convert from a voltage to a digital value or number*
  - ADCs are classified by their resolution, measured in bits.
    - Ex: Ardruino Uno = 10 bit ADC

- More information:
  - https://en.wikipedia.org/wiki/Analog-to-digital_converter
  - https://learn.sparkfun.com/tutorials/analog-to-digital-conversion/all



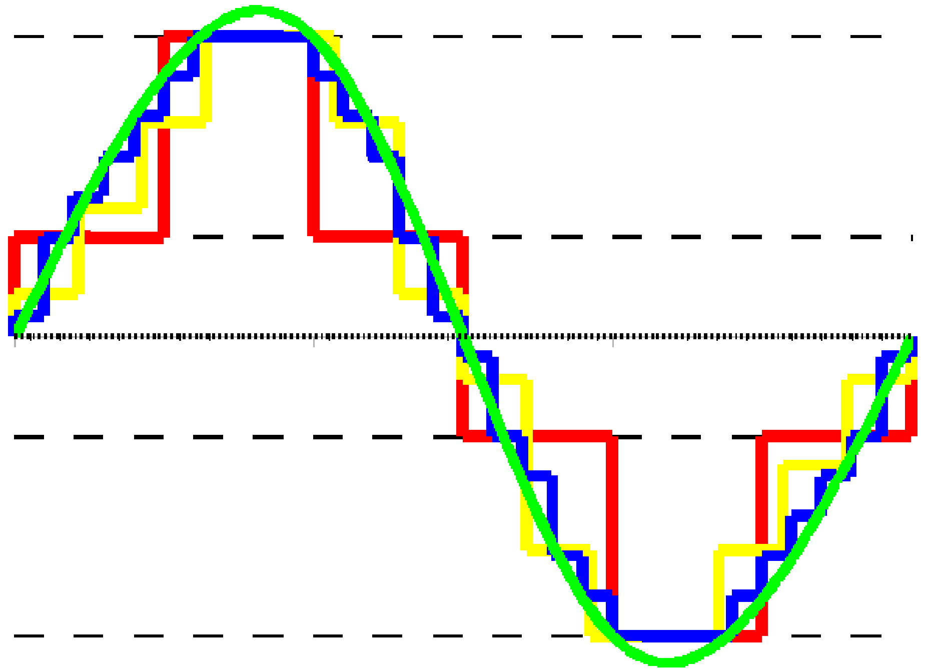
By Megodenas - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=36972702

ADC (Analog to Digital Converter) introduction

- ADC Math
  - 10 bits = 2^10 = 1024 steps
    - – Counting starts at 0, 0 – 1023
  - When measuring 0 – 5V, divide 5 / 1023
    - – Each step or value = 0.004887
    - – ADC value of 100 = 0.4887V

- Common ADC Types
  - 8 bits = 2^8 = 256
  - 10 bits = 2^10 = 1024
  - 12 bits = 2^10 = 4096

- Comparison of 2, 3 and 4 bit ADCs
  - – Analog Signal
  - – 2 Bit
  - – 3 Bit
  - – 4 Bit



By Hyacinth - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=34641795

`analogRead()` Function

- Exercise (use the Arduino's 10 bit ADC)
  - What is the ADC value for an input of:
    - 0V
    - 5V
    - 1V
    - **Formula = (1023 / 5V) * Input Voltage**
  - What is the voltage for an ADC reading of:
    - 0
    - 1023
    - 100
    - **Formula = 5V * (ADC # / 1023)**
- More information:
  - https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

$$ADC_{value} = V_{in} \frac{\left(2^{bits} - 1\right)}{V_{ref}}$$

$$V_{in} = V_{ref} \frac{ADC_{value}}{\left(2^{bits} - 1\right)}$$

`analogRead()` Function

```
potValue = analogRead(potPin);
```

- Reads the analog value from potPin analog input.

- The analogRead function reads the value from a specified analog pin using its built in ADC.
- The Arduino Uno and derivates have a 10bit ADC that returns a value between 0 to 1023.
- Syntax:

```
    analogRead(pin);
```

  - Pin: Arduino pin number to read.
  - Returns an int between 0 and 1023.
- More information:
  - https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

Lesson 1: Blink

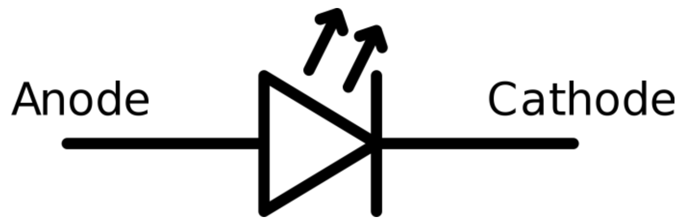
Blinking an LED

LED Introduction

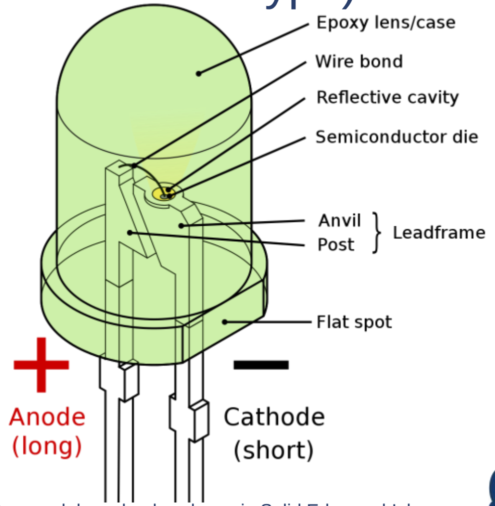
Lesson 1: Blink

- What is an LED?
  - A LED, or Light Emitting Diode, is an electronic device that produces light when a current is passed from its Anode (positive terminal) to its Cathode (negative terminal). They are commonly used for indication and lighting.
- Where are LEDs used?
  - LEDs are used in many devices, from light bulbs, computers, road signs, cell phones, clocks, industrial equipment, home appliances and much, much more.
  - Activity: find a device not listed above that uses an LED.

LED Symbol



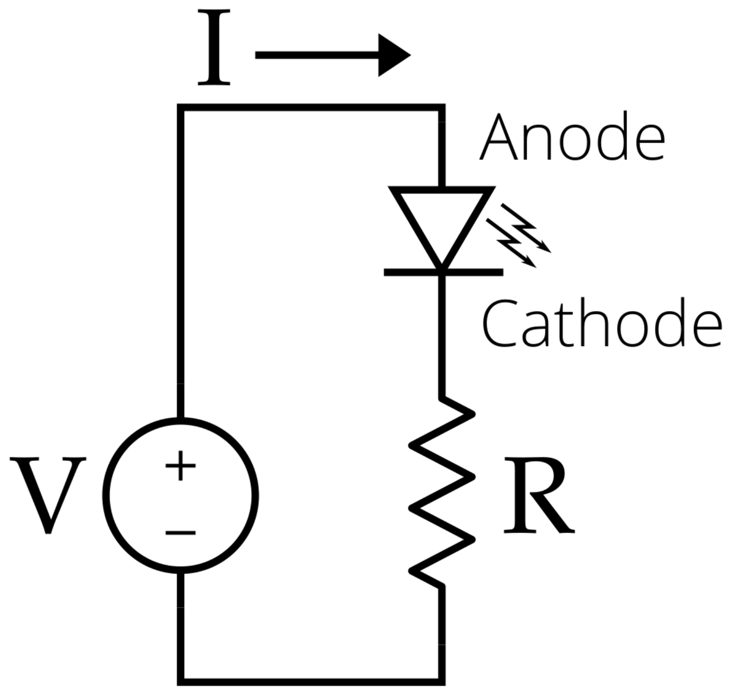
Drawing of a 5 mm round (the "normal" type) LED



LED Introduction (cont.)

Lesson 1: Blink

- How do I use an LED?
  - LEDs are current driven light sources, therefore we must limit the current flowing through them. We normally do this with a resistor.
    - *I* is the current flowing through the circuit
    - *V* is the voltage source
    - *R* is the resistor, used to limit current through the LED
    - Note the direction of the current flow through the LED, from the Anode to the Cathode. Like all diodes, LEDs only allow current flow in one direction.
  - In our application, the 5V microcontroller on our Grove Board will provide the voltage to the LED connected to D4.
    - The resistor has already been included to limit the current.
- More Info:
  - https://en.wikipedia.org/wiki/Light-emitting_diode
  - https://www.allaboutcircuits.com/textbook/semiconductors/chpt-3/special-purpose-diodes/#Light-emitting%20diodes
  - https://www.allaboutcircuits.com/tools/led-resistor-calculator/



Combining Digital Signals with LEDs
Lesson 1: Blink

- We can use the MCU on our Arduino to create a digital signal to control the LED.
  - Outputting a HIGH signal can turn the LED ON, and
  - Outputting a LOW signal turns the LED OFF.
- What hardware will we need for this Lesson?
  - Grove LED Module on pin D4
  - Seeeduino Lotus (Arduino Uno compatible board)

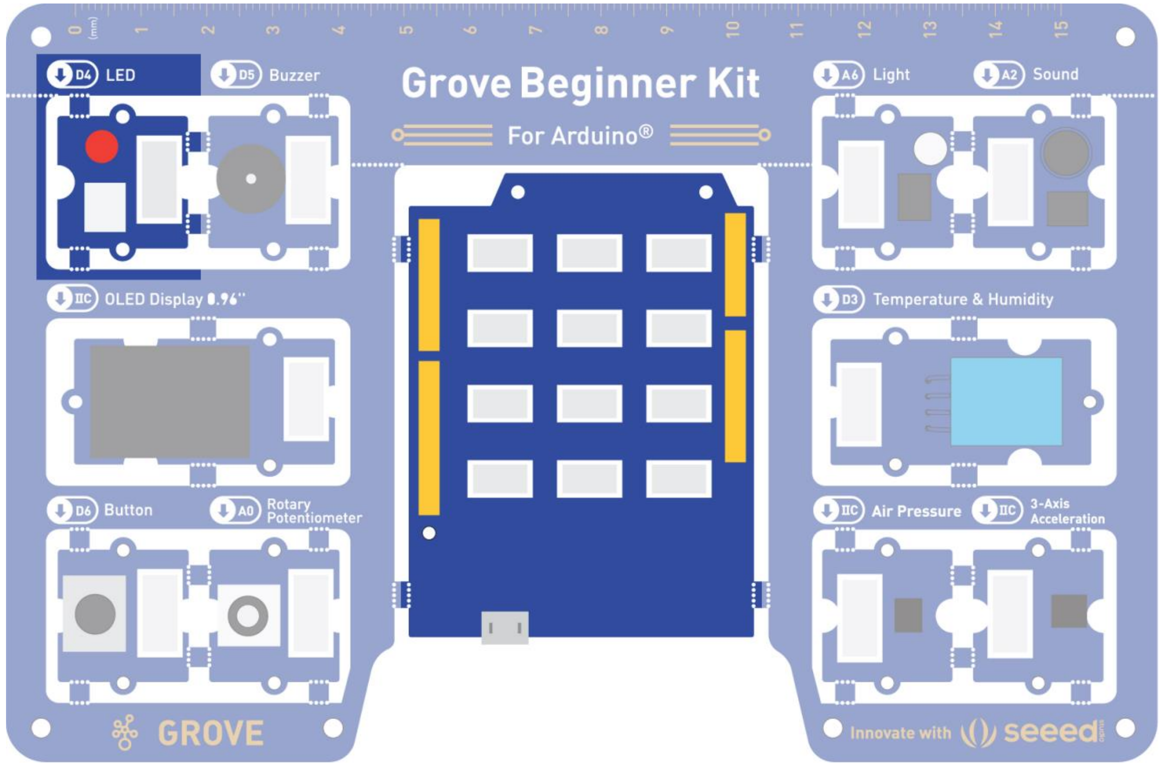
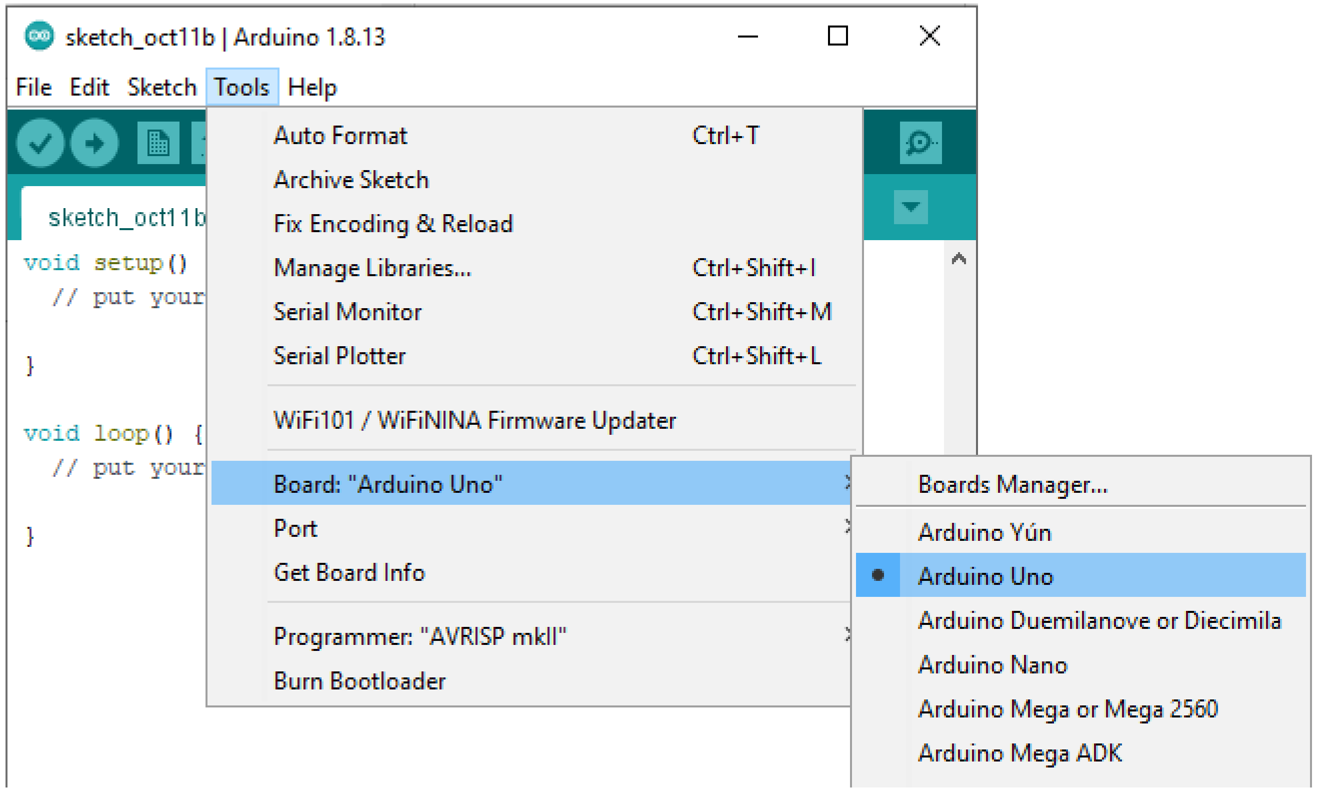


Image from https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf

Start and Setup the Arduino IDE

Lesson 1: Blink

1. Open the **Arduino IDE**
   a. Note: if Windows Firewall asks for an exception for Java SE Binary, allow on both Public and Private Networks (not secure, but this is a datalogging and sensors class, not a security class)
2. Select **Arduino Uno Board**
   a. **Tools → Board:… → Arduino Uno**

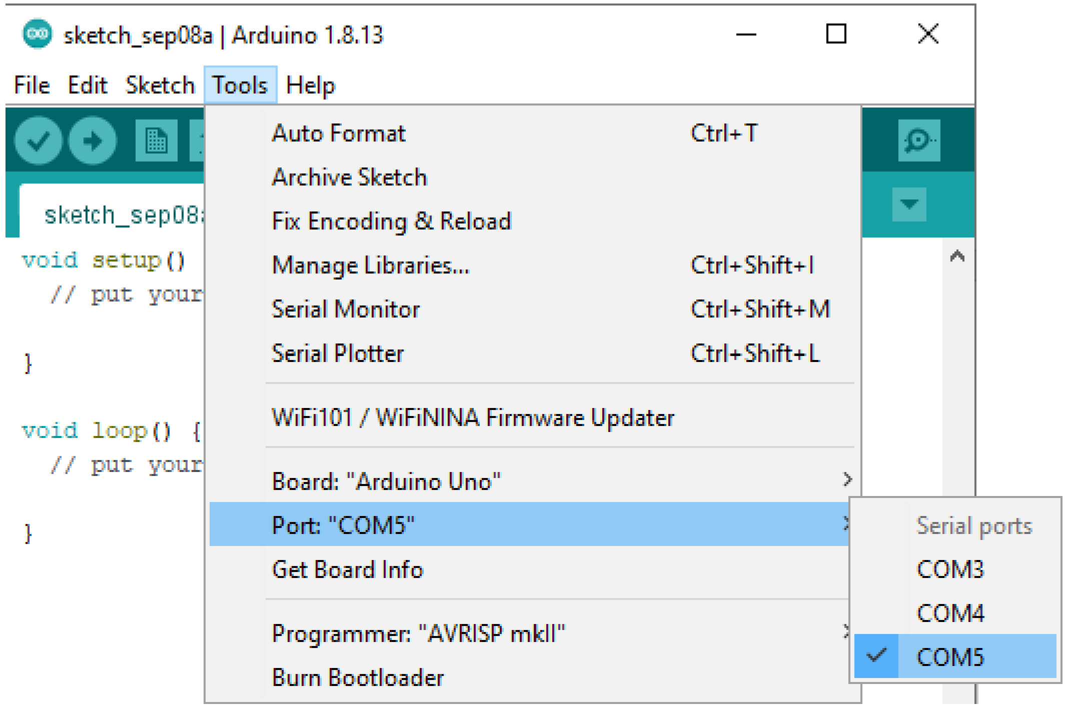

sketch_oct11b | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_oct11b
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino Yún
● Arduino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino Mega or Mega 2560
Arduino Mega ADK

```
void setup()
  // put your

}

void loop() {
  // put your

}
```

Start and Setup the Arduino IDE

Lesson 1: Blink

3. Connect your Arduino to your computer with the USB cable.
4. Find your **COM Port**
   a. Reference the Orientation documentation for how to do this for your particular operating system.
5. Select your **COM Port**
   a. **Tools → Port:… → COMx**
      I. "COMx" is the COM Port you identified in the previous step. In this example it was COM5. Your COM Port may be different.

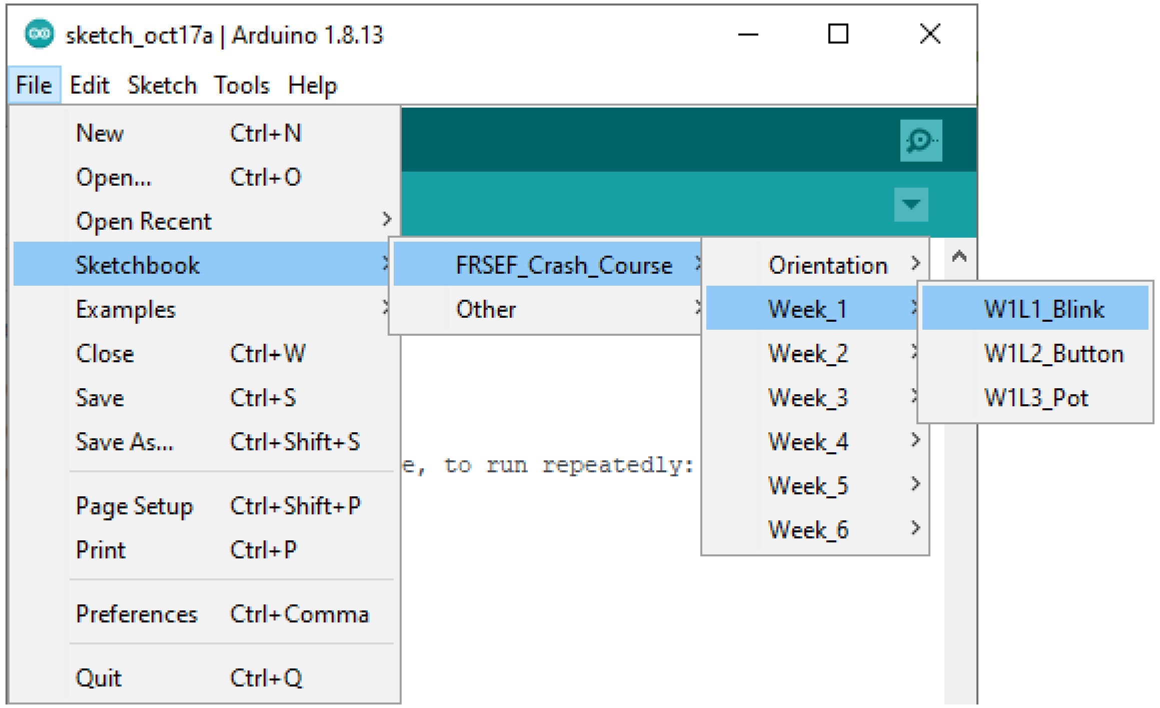

sketch_sep08a | Arduino 1.8.13

```
void setup() {
  // put your
}

void loop() {
  // put your
}
```

Open and Upload Sketch

Lesson 1: Blink

6. Open Blink Sketch
   a. **File → Sketchbook → FRSEF_Crash_Course → Week_1 → W1L1_Blink**

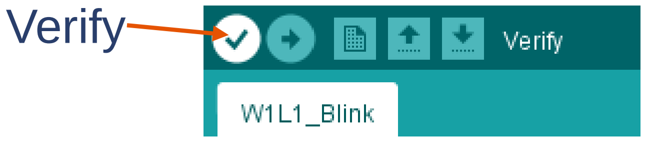


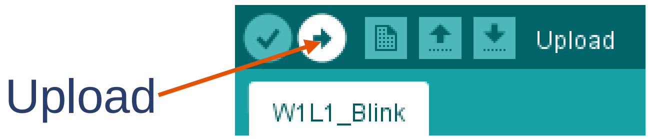
```
e, to run repeatedly:
```

Open and Upload Sketch

Lesson 1: Blink

6. Verify the sketch by clicking the Verify Button.
   a. The sketch should compile with no errors.
7. Upload the sketch to your Arduino by clicking the Upload Button.
   a. The sketch should re-compile, and then upload to your Arduino.
8. Watch as the LED blinks about once every two seconds.





Code Analysis - Comments

Lesson 1: Blink

- Text that the compiler ignores.
  - Used to document your code so it is easier to understand for yourself or others
  - and comment out lines of code so that the compiler ignores them without deleting them
- Syntax (two methods):
  - `/* your comment here */`
    - Everything between the opening identifier and closing identifier is commented, even across new lines.
  - `// your comment here`
    - Everything after the start identifier is a comment until a new line is reached
- More information:
  - https://www.arduino.cc/reference/en/language/structure/further-syntax/blockcomment/
  - https://www.arduino.cc/reference/en/language/structure/further-syntax/singlelinecomment/

```
/*
 * This is a block comment
 *  it can span multiple lines
 */

// This is a single line comment
```

Example with two different types of comments in **bold.**

Code Analysis – Setup Function

Lesson 1: Blink

- The `setup()` function is called when a sketch starts.
- Use it to initialize variables, pin modes, start using libraries, etc.
- The `setup()` function will only run once, after each powerup or reset of the Arduino board.
- Syntax:

```
void setup()
{

}
```

- More information:
  - https://www.arduino.cc/reference/en/language/structure/sketch/setup/

```
void setup()
{
  // put your setup code here
  // to run once:
}
```

Example Setup Function

Code Analysis – Loop Function

Lesson 1: Blink

- The `loop()` runs after the `setup()` function.
- It then repeats continuously until the Arduino looses power or is reset.
- It is where the main portion of your code goes.
- Use it to allow you board to continuously control your Arduino.
- Syntax:

```
void loop()
{

}
```

- More information:
  - https://www.arduino.cc/reference/en/language/structure/sketch/loop/

```
void loop()
{
  // put your main code here
  // to run repeatedly:
}
```

Example Loop Function

Code Analysis – Variable
Lesson 1: Blink

What is a variable?

- A variable is a way of naming and storing a value for later use by the program, such as data from a sensor or an intermediate value used in a calculation.

What are the different types of variables?

Variable Data Type	Range
bool	0 or 1
byte	Integers from 0 to 255
char	Characters, ex ‘a’, ‘1’, ‘+’, or ‘H’
double	floating point ex. 1.234 or 56.78
int	Integers from -32,768 to 32,767
unsigned int	Integers from 0 to 65,535
More types available in the more information link	

- More information:
  - https://www.arduino.cc/reference/en#data-types

Code Analysis – Variable Declaration

Lesson 1: Blink

- Before a variable can be used it must be declared.
- To declare a variable, its type and name must be defined.
- You may optionally initialize its value or apply a qualifier.
- Syntax:

REQUIRED
Name Definition

REQUIRED
Punctuator

```
qualifier type name = value;
```

OPTIONAL
Qualifier

REQUIRED
Type Definition

OPTIONAL
Value Initializer

- More information:
  – https://www.arduino.cc/en/Reference/VariableDeclaration

```
const int ledPin = 4;
```

Example variable declaration with constant qualifier and value initialization.

```
byte state;
```

Example variable declaration.

Code Analysis – Pin Definition

Lesson 1: Blink

- We first define which pin the LED is driven by. In our case it is prewired to pin D4.

```
const int ledPin = 4; // Grove LED is on pin D4
```

- We defined it with the `const` qualifier which tells the compiler that that variable cannot change.
- Pin D4 is electrically connected to many points on the Seeeduino Lotus (circled). You can use these points later when you need to break out the modules or other Arduino compatible shields.
- More information:
  - https://wiki.seeedstudio.com/Seeeduino_Lotus/

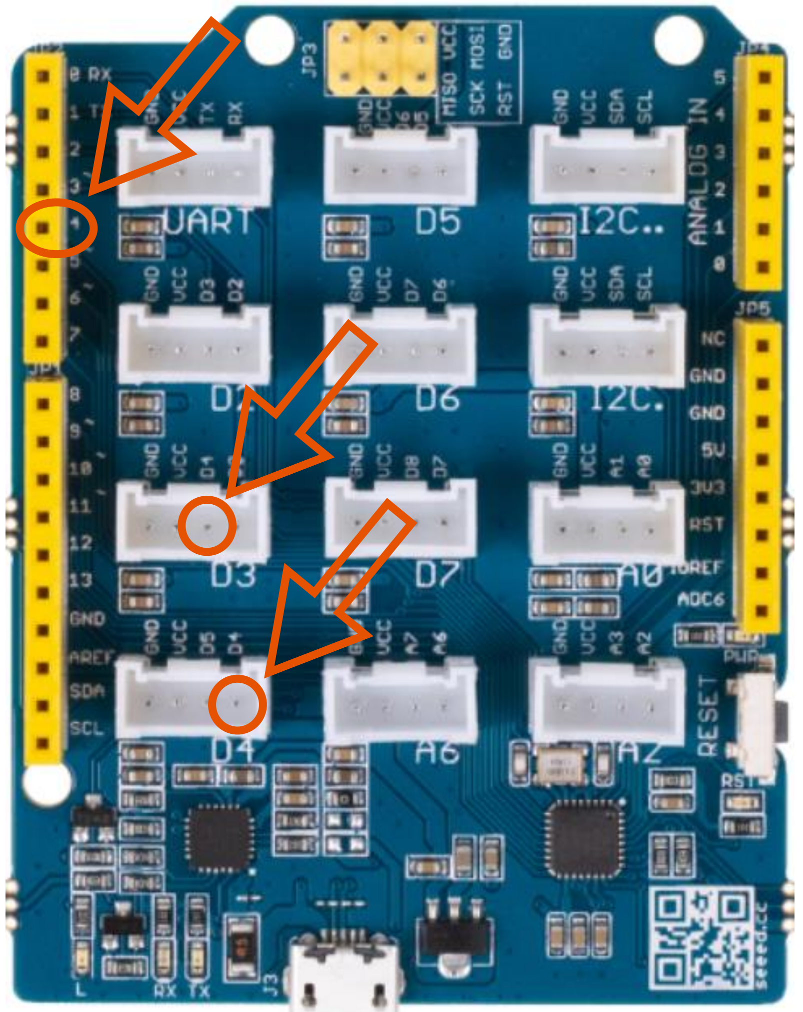


Image from https://www.seeedstudio.com/Grove-Beginner-Kit-for-Arduino-p-4549.html

Code Analysis – pinMode Function

Lesson 1: Blink

```
pinMode(ledPin, OUTPUT);
```

- Configures the ledPin as an output.

- The pinMode function configures the specified pin to behave either as an input or an output.
- Syntax:

```
pinMode(pin, mode);
```

  - Pin: Arduino pin number to set the mode of
  - Mode: options are
    - `INPUT`, set pin as an input
    - `OUTPUT`, set pin as an output
    - `INPUT_PULLUP`, set pin as an input and enable a weak internal pullup resistor.
- More information:
  - https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/

Code Analysis – delay Function

Lesson 1: Blink

`delay(1000);`

- Wait for 1 second.

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

```
    delay(ms);
```

  - ms: number of milliseconds (ms) to pause.
    - Data type is unsigned long with a range of 0 to 4,294,967,295ms (about 49.7 days)
- Warning – the delay function blocks most other activity of the Arduino. This means no other sensor readings, math functions, communications* and pin manipulation* can happen during the delay. It is recommended to only use delay for short pauses or simple sketches.
- More information:
  - https://www.arduino.cc/reference/en/language/functions/time/delay/

* Interrupts and functions using interrupts still work

Blink Activities

Lesson 1: Blink

- Activity 1
  - Change the blink rate to 2Hz (2 blinks per second).
- Activity 2
  - Change the Pin to use the `LED_BUILTIN` keyword. Watch the results on your board.
- Activity 3 (Bonus / Homework)
  - Alternate LEDs that blink.
- Activity 4 (Bonus / Homework)
  - Toggle the LED of your choice, without explicitly defining the state of the LED with `HIGH` or `LOW`. (This will require knowledge from Lesson 2)

Lesson 2: Button

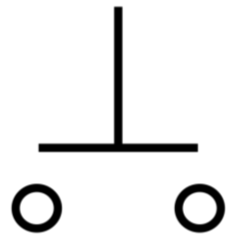
Use a pushbutton to change the state of the LED

Pushbutton Introduction

Lesson 2: Button

- What is a pushbutton?
  - A pushbutton is a momentary type of switch used as an input to an electrical system. When **closed** it allows an electrical current to flow through it. When **open** it prevents electrical current flow through it.
- Where are pushbuttons used?
  - Pushbuttons are used in many devices, from keyboards, cell phones, alarm clocks, industrial equipment, home appliances and much, much more.
  - Activity: find a device not listed above that uses a pushbutton.

Pushbutton Symbol



By Michel Bakni - Derived from files [1] and [2].(in English) (1993) 315-1975 - IEEE Standard American National Standard Canadian Standard Graphic Symbols for Electrical and Electronics Diagrams (Including Reference Designation Letters), IEEE, p. 59 DOI: 10.1109/IEEESTD.1993.93397. ISBN: 0738109479., CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=94264073

Example Pushbutton

Image from https://www.digikey.com/en/products/detail/e-switch/TL2230EEF140/4029358

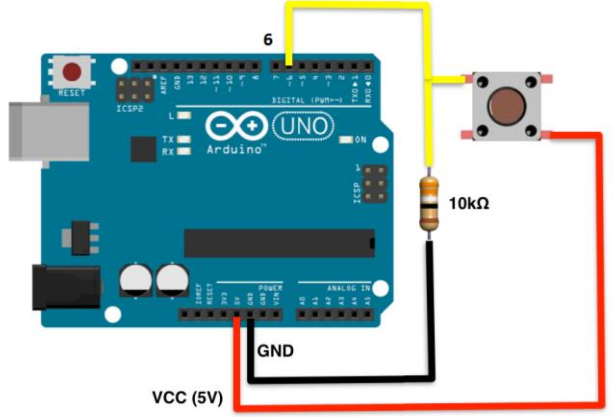
Pushbutton Introduction

Lesson 2: Button

- How do I use a pushbutton?
  - Follow the connection diagram to the right. Your Grove Beginner's Kit has already done this for you.
  - We then read the state of the input using the digitalRead function.

- More Info:
  - https://www.allaboutcircuits.com/textbook/digital/chpt-4/switch-types/
  - https://en.wikipedia.org/wiki/Push-button

Example Pushbutton Connection



Modified from https://sites.google.com/site/ardunitydoc/getting-started/run-examples/push-button

Lesson 2 Hardware

Lesson 2: Button

- We can use the MCU on our Arduino to receive a digital reading of the state of a pushbutton.
  - Receiving a HIGH signal means the button is pressed.
  - Receiving a LOW signal means the button is released.
- What hardware will we need for this Lesson?
  - Grove LED Module on pin D4
  - Grove Button Module on pin D6
  - Seeeduino Lotus (Arduino Uno compatible board)

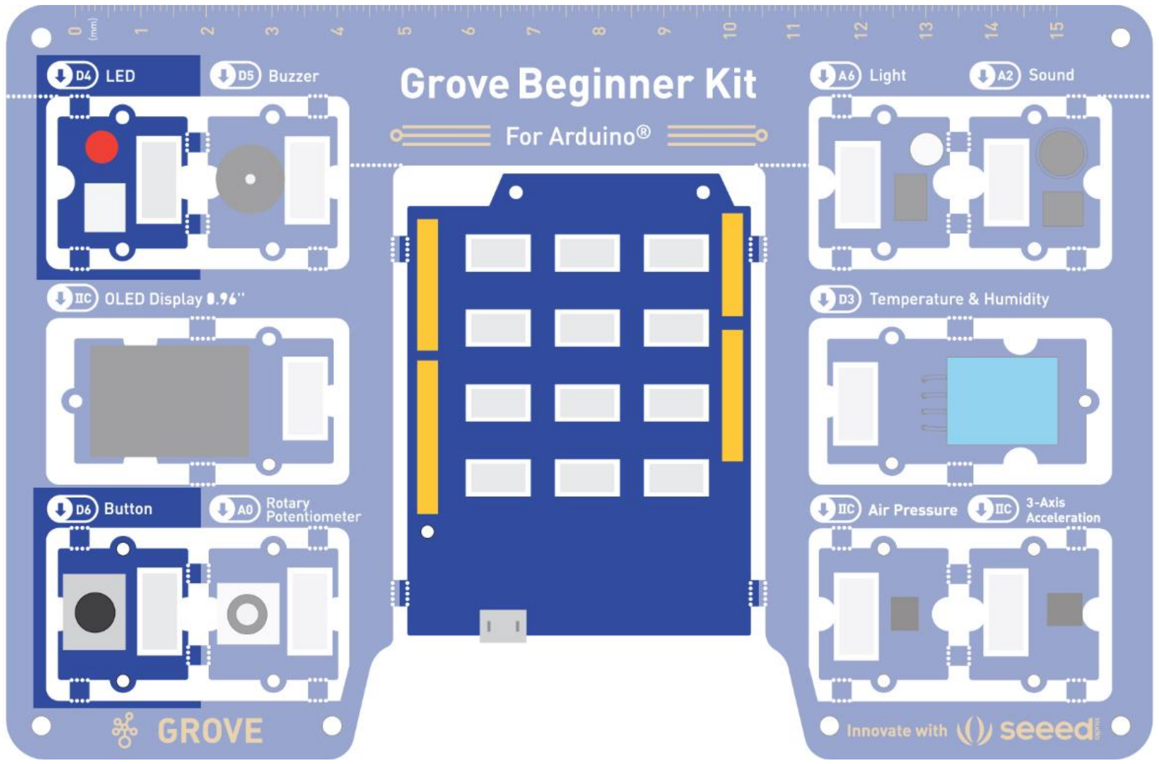


Image from https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf

Open and Upload Sketch

Lesson 2: Button

1. Open Button Sketch
   a. **File → Sketchbook → FRSEF_Crash_Course → Week_ → W1L2_Button**
2. Verify the sketch by clicking the Verify Button.
   a. The sketch should compile with no errors.
3. Upload the sketch to your Arduino by clicking the Upload Button.
   a. The sketch should re-compile, and then upload to your Arduino.
4. Watch the LED as you press the button.

Code Analysis – pinMode Function

Lesson 2: Button

```
pinMode(buttonPin, INPUT);
```

- Configures the buttonPin as an input.

- The pinMode function configures the specified pin to behave either as an input or an output.
- Syntax:

```
pinMode(pin, mode);
```

  - Pin: Arduino pin number to set the mode of
  - Mode: options are
    - `INPUT`, set pin as an input
    - `OUTPUT`, set pin as an output
    - `INPUT_PULLUP`, set pin as an input and enable a weak internal pullup resistor.
- More information:
  - https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/

Code Analysis – Variable Assignment

Lesson 2: Button

```
buttonValue = digitalRead(buttonPin);
```

- Assigns the state of read buttonPin to buttonValue.

- The assignment operator (**=**) puts whatever is on the right side of the equal sign into the variable on the left side.
- Syntax:

```
    variable = value;
```

  - Variable: stores the value of the statement on the right side of the equals sign.
  - Value: statement, function or equation whose value is to be stored in variable.
- More information:
  - https://www.arduino.cc/en/Reference/VariableDeclaration

Code Analysis – if…else if…else Conditionals

Lesson 2: Button

- The **`if`** statement checks a condition and executes the proceeding statement(s) if the condition is TRUE.
- The **`else`** statement executes if the previous **`if`** conditional evaluated as FALSE. The **`else`** statement is optional.
- The **`else if`** statement combines the **`else`** statement with the **`if`** statement. The **`else if`** statement is optional.
- Syntax:

```
if(condition1)
{
   // do this
}
else if(condition2) // OPTIONAL
{
   // do that
}
else // OPTIONAL
{
   // do something else
}
```

  – **`conditionX`** must evaluate to TRUE (not 0) or FALSE (0)

- More information:
  – https://www.arduino.cc/reference/en/language/structure/control-structure/if/
  – https://www.arduino.cc/reference/en/language/structure/control-structure/else/

```
if(buttonValue == HIGH)
{
  digitalWrite(ledPin, HIGH);
}
else
{
  digitalWrite(ledPin, LOW);
}
```

Example of an if…else conditional

Button Activities

Lesson 2: Button

- Activity 1
  - Change the LED to turn OFF if the button is pressed and turn ON when the button is released.
- Activity 2 (Bonus / Homework)
  - Keep the same function as the original W1L2_Button.ino sketch, without using a conditional statement.
- Activity 3 (Bonus / Homework)
  - Toggle the LED every time the button is pressed.

Lesson 3: Potentiometer

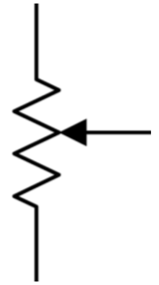
Use a potentiometer to change the brightness of the LED

Potentiometer Introduction

Lesson 3: Pot

- What is a potentiometer (pot)?
  - A pot is a type of variable resistor that has 3 terminals, two end terminals and a moveable wiper terminal. They are commonly used as adjustable voltage dividers. Engineers often use these properties to make position sensors.
- Where are pots used?
  - Pots are used in many devices, from volume knobs, industrial equipment, servos, home appliances, vehicles, and much, much more.
  - Activity: find a specific device that uses a pot.

Potentiometer Symbol



By Potentiometer with load.png: OmegatronDerivative work: DesbWit - This file was derived from: Potentiometer with load.png, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20737313

Example Potentiometers

By Junkyardsparkle - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=39291275

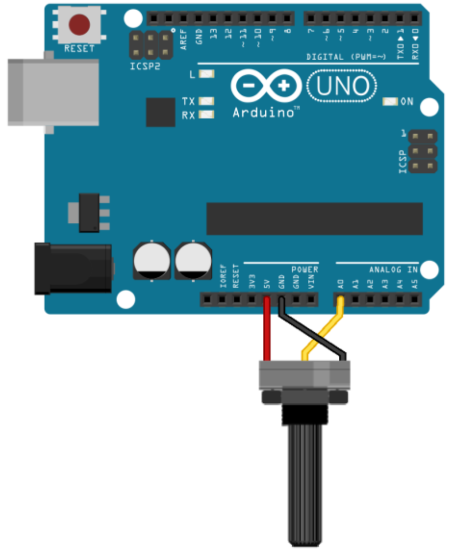
Potentiometer Introduction

Lesson 3: Pot

- How do I use a potentiometer?
  - Follow the connection diagram to the right. Your Grove Beginner's Kit has already done this for you.
  - We then read the state of the input using the analogRead function.

- More Info:
  - https://en.wikipedia.org/wiki/Potentiometer
  - https://www.allaboutcircuits.com/textbook/direct-current/chpt-6/voltage-divider-circuits/

Example Potentiometer Connection



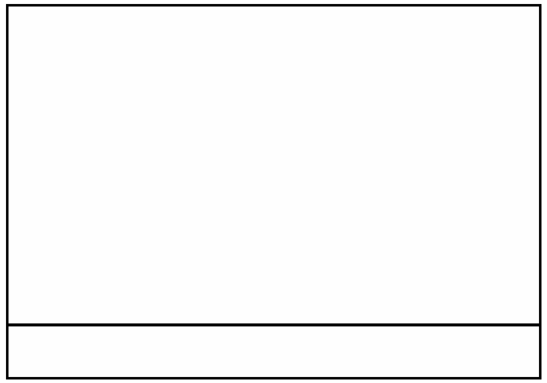
Modified from https://www.arduino.cc/en/Tutorial/BuiltInExamples/AnalogInput

Pulse Width Modulation (PWM) Introduction

Lesson 3: Pot

- What is Pulse Width Modulation?
  - PWM is a type of digital signal that varies its value using the width of the pulse.
  - The value of the PWM signal is called the duty cycle (D). The duty cycle can be calculated as follows:
    - $\mathrm{D} = \frac{t_H}{t_H + t_L}$
    - Where t_H is the time the signal is high,
    - And t_L is the time the signal is low.
- It is easy to convert a PWM signal back to an analog signal with a low pass filter.
- We can use this to control the brightness of the LED.
- More Info:
  - https://en.wikipedia.org/wiki/Pulse-width_modulation
  - https://www.allaboutcircuits.com/textbook/semiconductors/chpt-11/pulse-width-modulation/

PWM Signal



By Eighthave - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=2821502

Combining Analog, PWM, and LEDs

Lesson 3: Pot

- We can use the MCU on our Arduino to read the value of the pot and output a PWM signal to the LED to control the brightness.
  - Outputting a higher value is a larger duty cycle which means a brighter LED.
- What hardware will we need for this Lesson?
  - Grove LED Module on pin D4
  - Grove Rotary Potentiometer Module on pin A0
  - Seeeduino Lotus (Arduino Uno compatible board)

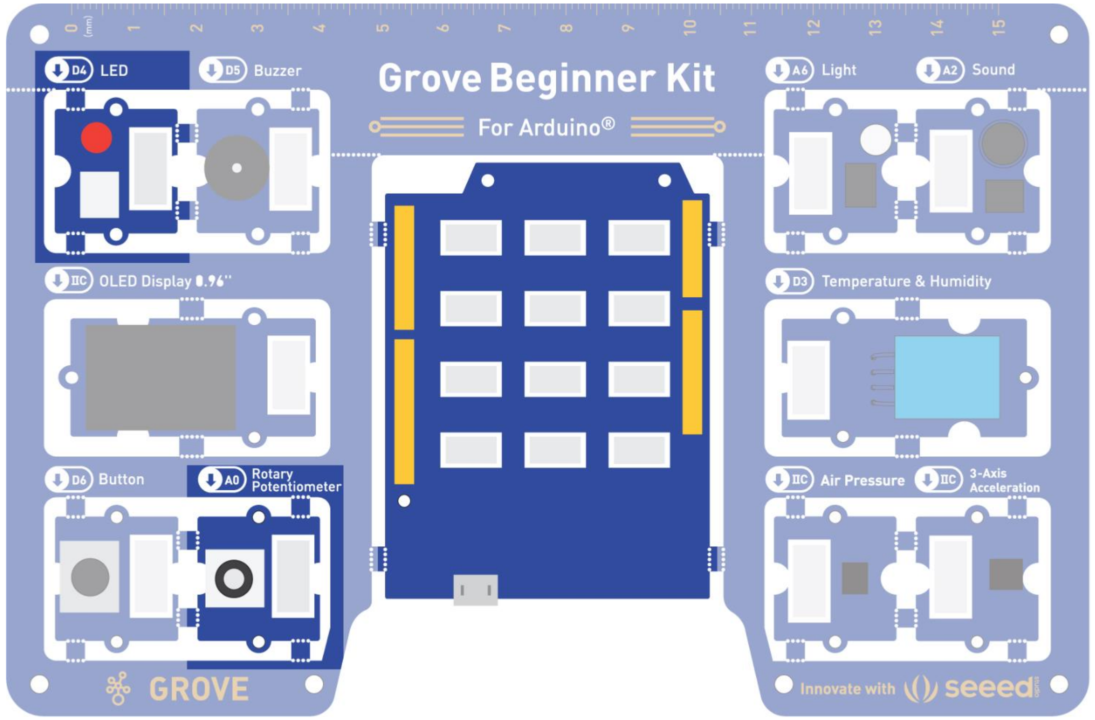


Image from https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf

Open and Upload Sketch

Lesson 3: Pot

1. Open Pot Sketch
   a. **File → Sketchbook → FRSEF_Crash_Course → Week_ → W1L3_Pot**
2. Verify the sketch by clicking the Verify Button.
   a. The sketch should compile with no errors.
3. Upload the sketch to your Arduino by clicking the Upload Button.
   a. The sketch should re-compile, and then upload to your Arduino.
4. Watch the LED as you rotate the potentiometer.

Code Analysis – delayMicroseconds Function

Lesson 3: Pot

```
delayMicroseconds(potValue);
```

- Wait for the number of microseconds (µs) stored in potValue.

- This function is similar to the delay function from Lesson 1, except it pauses by microseconds instead if milliseconds.
- There are 1000µs in 1ms and 1,000,000µs in 1s.
- Syntax:

```
delayMicroseconds(µs);
```

  - µs: number of microseconds (µs) to pause.
    - Data type is unsigned int with a range of 0 to 16,383µs (about 16 ms)
- More information:
  - https://www.arduino.cc/reference/en/language/functions/time/delaymicroseconds/

Pot Activities

Lesson 3: Pot

- Activity 1
  - Change the LED PWM to get brighter with a clockwise rotation of the potentiometer.
- Activity 2 (Bonus / Homework)
  - If the light sensor is on pin A6, modify the sketch to use the light sensor instead of the potentiometer.

Summary

Week 1

Week 1 topics

- Arduino overview
- Arduino IDE
- Microcontrollers
- LEDs
- Buttons (switches)
- Potentiometers
- Digital Signals
- Analog Signals
- PWM
- IDE setup (Board and COM port selection)
- Sketch Verification and Upload
- `//   /*  */` Comments
- `setup()` and `loop()`
- Variables and data types
- Pin definitions
- `pinMode()`
- `digitalWrite()`
- `digitalRead()`
- `analogRead()`
- `delay()`
- `delayMicroseconds()`
- `if()` and `else` conditionals

W1L1_Blink.ino Sketch

```
/*
 * W1L1_Blink.ino
 * Turn the LED ON for 1 second then OFF for 1 second
 *
 * FRSEF Crash Course V1 Week 1 Lesson 1
 * Written by Chris Bergsneider
 * cbergsneider@flintsciencefair.org
 * October 17, 2020
 */

const int ledPin = 4; // Grove LED is on pin D4

void setup()
{
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT); // set the ledPin to be an output
}

void loop()
{
  // put your main code here, to run repeatedly:
  digitalWrite(ledPin, HIGH); // Turn the LED ON
  delay(1000); // wait for 1 second
  digitalWrite(ledPin, LOW);  // Turn the LED OFF
  delay(1000); // wait for 1 second
}
```

W1L2_Button.ino Sketch

```
/*
 * W1L2_Button.ino
 * Turn the LED when button is pressed
 *
 * FRSEF Crash Course V1 Week 1 Lesson 2S
 * Written by Chris Bergsneider
 * cbergsneider@flintsciencefair.org
 * October 19, 2020
 */

const byte buttonPin = 6; // pushbutton is on D6
const byte ledPin = 4; // LED is on D4

byte buttonValue = 0; // global variable for storing the value of the button

void setup()
{
  pinMode(buttonPin, INPUT); // initialize the button as an input
  pinMode(ledPin, OUTPUT); // initialize the LED as an output
}

void loop()
{
  // Note: commented out the next line for more a more educational method
  //digitalWrite(ledPin, digitalRead(buttonPin)); // set LED to match button state

  // read state of the button and store it in variable buttonValue
  buttonValue = digitalRead(buttonPin);

  if(buttonValue == HIGH)  // Check if button is pressed
  {
    digitalWrite(ledPin, HIGH); // Turn ON LED
  }
  else  // if it is not pressed
  {
    digitalWrite(ledPin, LOW);  // Turn OFF LED
  }
}
```

W1L3_Pot.ino Sketch

```
/*
 * W1L3_Pot.ino
 * Change the brightness of the LED with the potentiometer
 *
 * FRSEF Crash Course V1 Week 1 Lesson 3
 * Written by Chris Bergsneider
 * cbergsneider@flintsciencefair.org
 * October 21, 2020
 */

const byte ledPin = 4;  // LED is on pin D4
const byte potPin = A0; // Potentiometer is on pin A0

const int analogHigh = 1023;  // maximum analogRead value

unsigned int potValue = 0;

void setup()
{
  pinMode(ledPin, OUTPUT); // set the ledPin to be an output
}

void loop()
{
  potValue = analogRead(potPin);  //read the potentiometer
  // Create the PWM signal
  digitalWrite(ledPin, HIGH); // Write the LED pin high
  delayMicroseconds(potValue * 16);  // delay by the pot value microseconds *16
  digitalWrite(ledPin, LOW);  // Write the LED pin low
  delayMicroseconds((analogHigh - potValue) * 16);  // wait for the remainder of the peroid
}
```