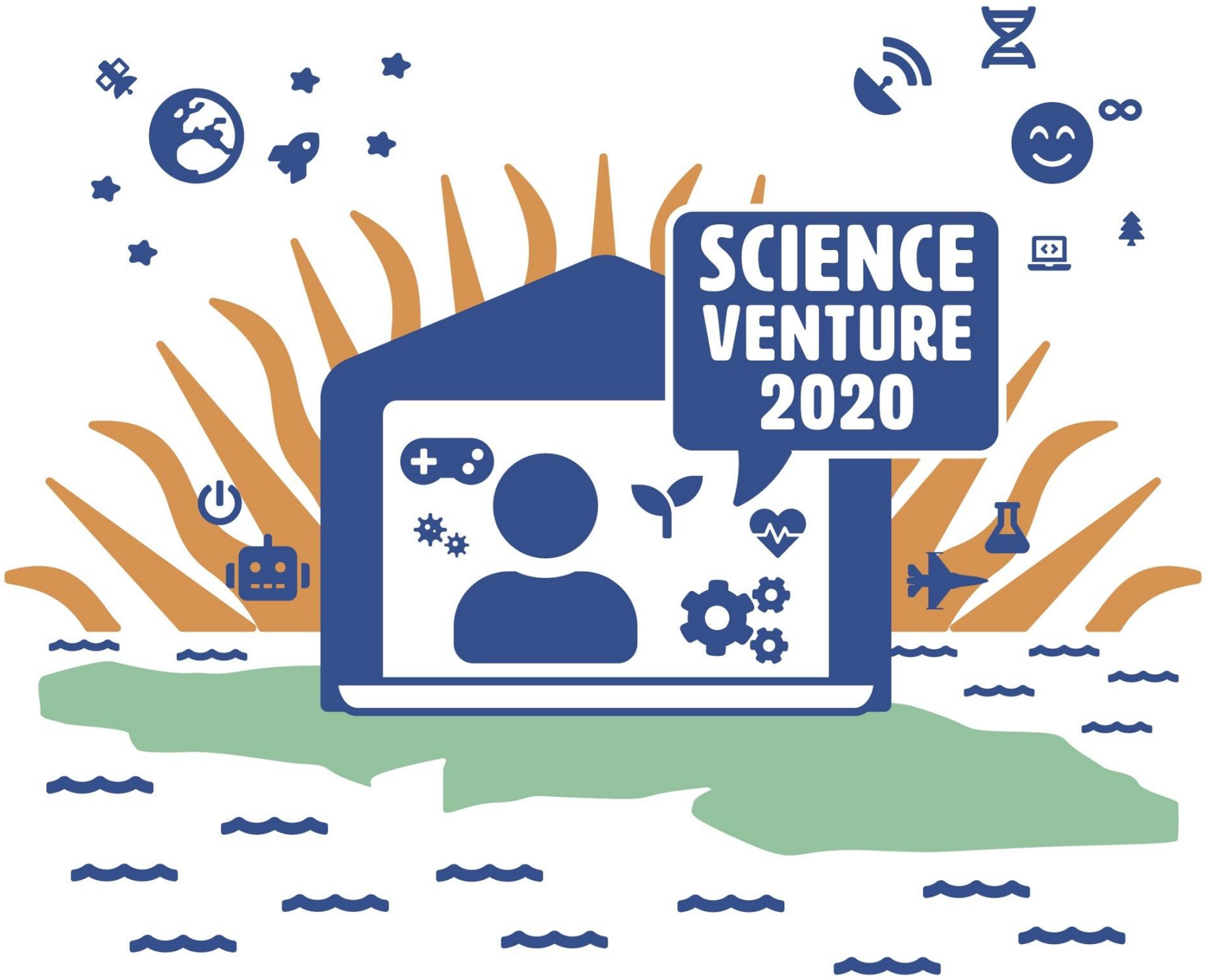


Arduinos

Arduino Intro

Introduction to Arduino microcontrollers. Youth will learn how to connect their computer to the Arduino web editor and cloud. Practice coding C++ and program a Morse code message using the onboard LED.

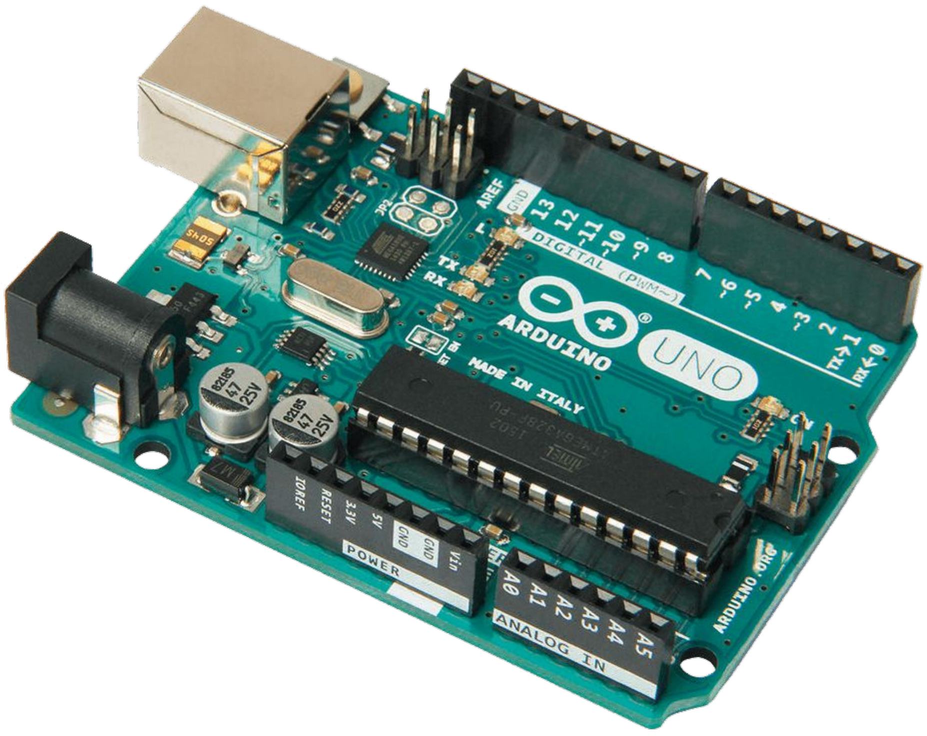
What is a Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing

Arduino coding environment: https://create.arduino.cc/editor/

Materials

- Arduino Uno Kit
  - USB Cable
  - White LED
- Computer with Arduino Create



Arduino Intro

Why Arduino?

Inexpensive

- Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand and even the pre-assembled Arduino modules cost less than $50

Cross-platform

- The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

Simple, clear programming environment

- The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

Open source and extensible software

- The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

Open source and extensible hardware

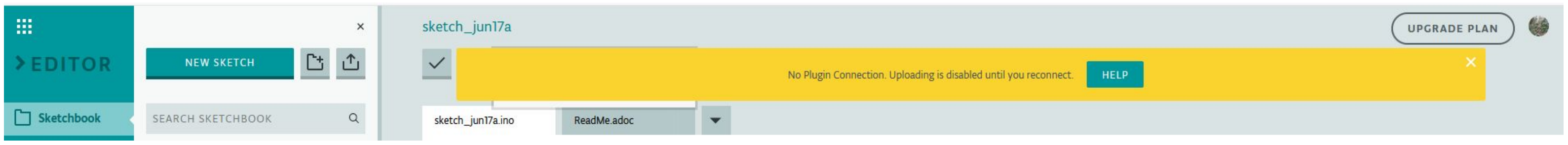
- The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Arduino Intro

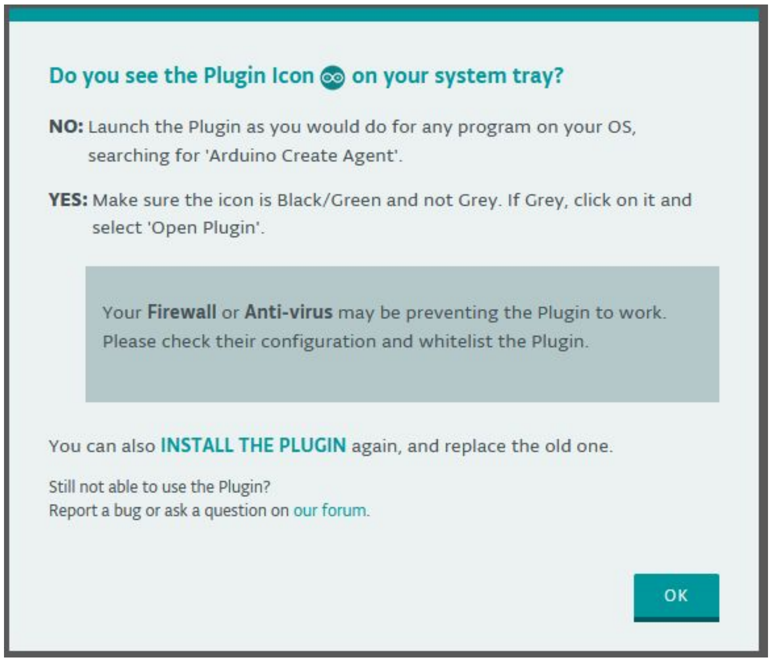
Testing Arduino upload - simple test program

The Arduino Web Editor is the whole Arduino Desktop IDE online in a web browser. Create and document your Arduino projects, create sketches, flash sketches onto your Arduino board and share your work with your friends.

Code online with the Arduino web editor (account required)
https://create.arduino.cc/editor/


No Plugin Connection. Uploading is disabled until you reconnect.

Press “HELP” on the left sidebar or top yellow message, then you will see this pop-up.


Do you see the Plugin Icon on your system tray?

NO: Launch the Plugin as you would do for any program on your OS, searching for 'Arduino Create Agent'.

YES: Make sure the icon is Black/Green and not Grey. If Grey, click on it and select 'Open Plugin'.

Your **Firewall** or **Anti-virus** may be preventing the Plugin to work. Please check their configuration and whitelist the Plugin.

You can also INSTALL THE PLUGIN again, and replace the old one.

Still not able to use the Plugin?
Report a bug or ask a question on our forum.

Near the bottom of the help screen, click “INSTALL THE PLUGIN”
Run the installer once downloaded.
(ArduinoCreateAgent-1.1-windows-installer-firefox.exe on our machine)

Allow access to the program if asked by your firewall. You may need to go back to the webpage where you downloaded the plugin and refresh to confirm the app is installed and connected to the Arduino cloud.

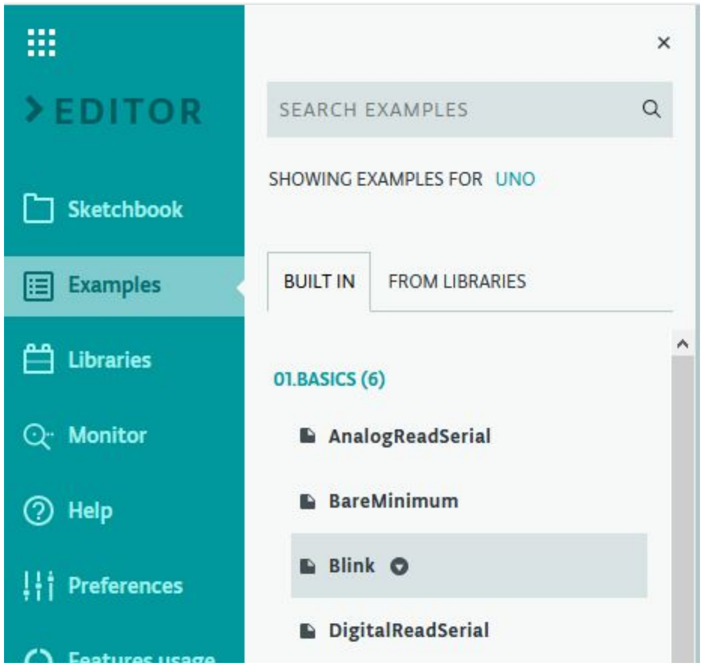
When finished, you should see this symbol in your system tray:

Arduino Intro

Instructions for setting up the Arduino web editor

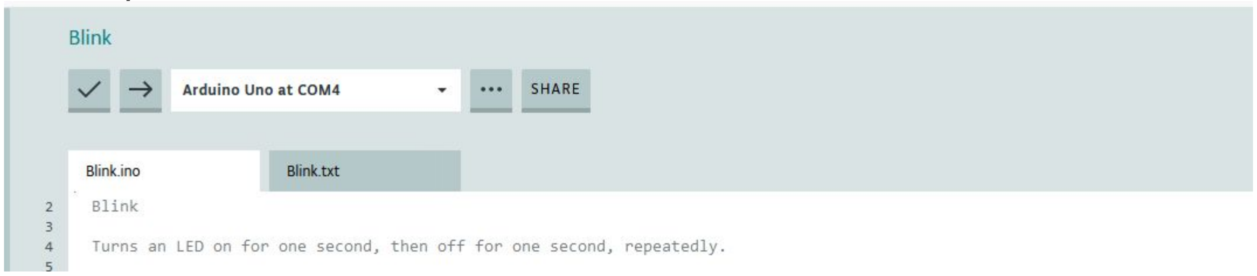
Return to the Arduino Create website. You can do this by clicking the symbol in the system tray and choosing "Arduino Create", then "Arduino web editor".

In the left side bar, click "Examples", then find the "Blink" program under "Basics".



This is a simple program which turns the Arduino onboard LED on and off in one second intervals.

The check mark under the program name verifies the program, and the arrow uploads it to your Arduino Uno.



```
Blink

Turns an LED on for one second, then off for one second, repeatedly.
```

Some LEDs will quickly flash on your Arduino boards as the program uploads. Then, the one second blinks should start.

Congratulations! You have successfully uploaded and run your first Arduino program.

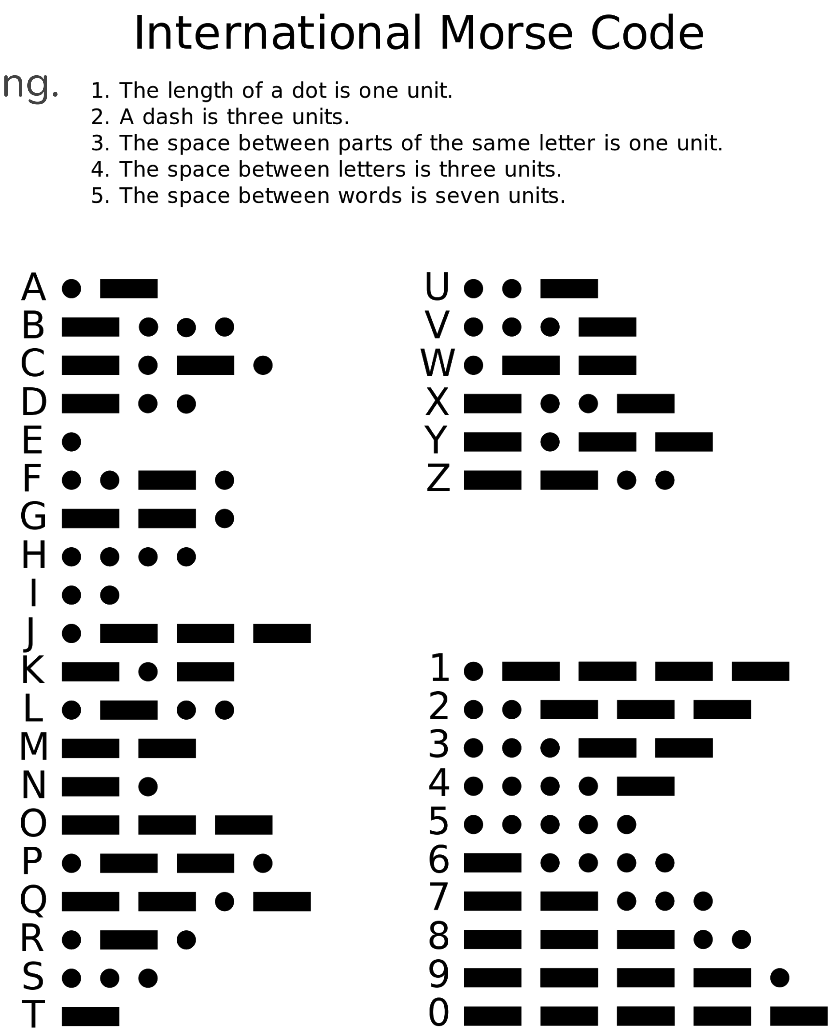
Morse Code with Arduino

Code messages in morse code using the Arduino software.

Let's explore using the onboard LED to transmit morse code.

Basics of Morse code:

- We call the shortest duration of time used in Morse code a "**unit**".
- The unit is usually made as short as possible while still being understandable to the receiver.
- A "**dot**" is a short blink, one unit long.
- A "**dash**" is a longer blink, three units long.
- Spaces between dots and dashes in a letter are one unit long. (in-letter space)
- Spaces between letters are three units long.
- Spaces between words are seven units long.



International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	● ▬
B	▬ ● ● ●
C	▬ ● ▬ ●
D	▬ ● ●
E	●
F	● ● ▬ ●
G	▬ ▬ ●
H	● ● ● ●
I	● ●
J	● ▬ ▬ ▬
K	▬ ● ▬
L	● ▬ ● ●
M	▬ ▬
N	▬ ●
O	▬ ▬ ▬
P	● ▬ ▬ ●
Q	▬ ▬ ● ▬
R	● ▬ ●
S	● ● ●
T	▬
U	● ● ▬
V	● ● ● ▬
W	● ▬ ▬
X	▬ ● ● ▬
Y	▬ ● ▬ ▬
Z	▬ ▬ ● ●
1	● ▬ ▬ ▬ ▬
2	● ● ▬ ▬ ▬
3	● ● ● ▬ ▬
4	● ● ● ● ▬
5	● ● ● ● ●
6	▬ ● ● ● ●
7	▬ ▬ ● ● ●
8	▬ ▬ ▬ ● ●
9	▬ ▬ ▬ ▬ ●
0	▬ ▬ ▬ ▬ ▬

NOTE: if the onboard LED is not bright enough, put one side of an LED into the GND and the other into pin 13. The code doesn't need to be modified for this to work.

Image from: https://upload.wikimedia.org/wikipedia/commons/thumb/b/b5/International_Morse_Code.svg/2000px-International_Morse_Code.svg.png

Morse Code with Arduino

MorseCodeInitial.ino

Copy and paste Arduino code (in blue) into the online editor or into a text file and saved as a .ino file.

```
/*
	Let's explore using the onboard LED to transmit morse code!

	Basics of Morse code:
	We call the shortest duration of time used in Morse code a "unit".
	The unit is usually made as short as possible while still being understandable to
	the receiver.

	A "dot" is a short blink, one unit long.
	A "dash" is a longer blink, three units long.

	Spaces between dots and dashes in a letter are one unit long. (in-letter space)
	Spaces between letters are three units long.
	Spaces between words are seven units long.

	NOTE: if the onboard LED is not bright enough, put one side of an LED into the
	GND and the other into pin 13. The code doesn't need to be modified for this to
work.

*/

const int unit = 250; //Let's start off by using 250 milliseconds as the shortest unit.

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

Morse Code with Arduino

```
// the loop function runs over and over again forever
void loop() {
  //short blink (dot)
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(unit);                      // wait for 1 unit
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(unit);                      // wait for 1 unit

  //long blink (dash)
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(3*unit);                    // wait for 3 unit
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(unit);                      // wait for 1 unit

  //letter A (.-)
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(unit);                      // wait for 1 unit (dot)
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(unit);                      // wait for 1 unit (in-letter space)
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(3*unit);                    // wait for 3 unit (dash)
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(3*unit);                    // wait for 3 units (end of letter space)


  //space
  delay(4*unit);                    // Spaces between words require 7 units of space.
                  // Each letter already has 3 units of space at the end.
                  // So an additional 4 units of space signify the end
                  // of a word.
}
```

Morse Code with Arduino

Morse Code Functions: MorseCodeInitial.ino

```
/*
        Let's explore using the onboard LED to transmit morse code!

        Basics of Morse code:
        We call the shortest duration of time used in Morse code a "unit".
        The unit is usually made as short as possible while still being understandable to
        the receiver.

        A "dot" is a short blink, one unit long.
        A "dash" is a longer blink, three units long.

        Spaces between dots and dashes in a letter are one unit long. (in-letter space)
        Spaces between letters are three units long.
        Spaces between words are seven units long.

        NOTE: if the onboard LED is not bright enough, put one side of an LED into the
        GND and the other into pin 13. The code doesn't need to be modified for this to
        work.

*/

const int unit = 250; //Let's start off by using 250 milliseconds as the shortest unit.


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

// the loop function runs over and over again forever
void loop() {
  letter_a();
  letter_b();
  letter_b();
  letter_a();
  end_of_word();
}
```

Morse Code with Arduino

```
void letter_a() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(unit);          // dot
  digitalWrite(LED_BUILTIN, LOW);
  delay(unit);          // in-letter space
  digitalWrite(LED_BUILTIN, HIGH);
  delay(3*unit);              // dash
  digitalWrite(LED_BUILTIN, LOW);
  delay(3*unit);              // end of letter space
}

void letter_b() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(3*unit);              // dash
  digitalWrite(LED_BUILTIN, LOW);
  delay(unit);          // in-letter space
  digitalWrite(LED_BUILTIN, HIGH);
  delay(unit);          // dot
  digitalWrite(LED_BUILTIN, LOW);
  delay(unit);          // in-letter space
  digitalWrite(LED_BUILTIN, HIGH);
  delay(unit);          // dot
  digitalWrite(LED_BUILTIN, LOW);
  delay(unit);          // in-letter space
  digitalWrite(LED_BUILTIN, HIGH);
  delay(unit);          // dot
  digitalWrite(LED_BUILTIN, LOW);
  delay(3*unit);              // end of letter space
}

void end_of_word() {
  delay(4*unit);    // adds 4 units of space to the end of the last letter
            // this signifies the end of the word
}
```

#SVatHome

Want to share your project or results with us?

Email or tag us @ScienceVenture

Have a question?

Reach us at svcamp@engr.uvic.ca