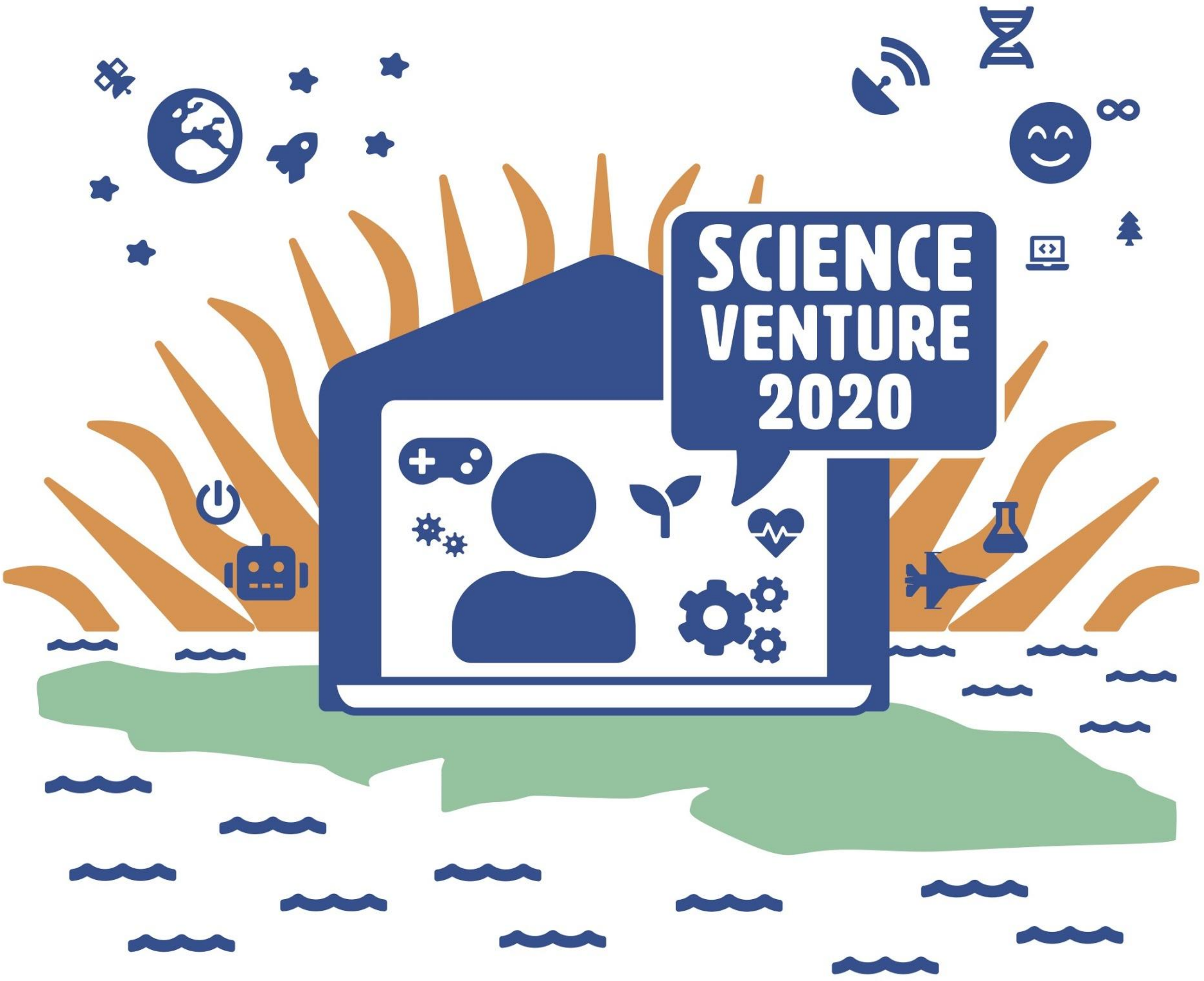


Arduinos

Arduino Fibonacci Sequence

Practice using variables in C++ code. Wire an Arduino to a breadboard and program a button to produce the values of the Fibonacci sequence. Learn about using the console, debugging, and value ranges related to binary maximums. Continue coding by extending the code into the tribonacci sequence.

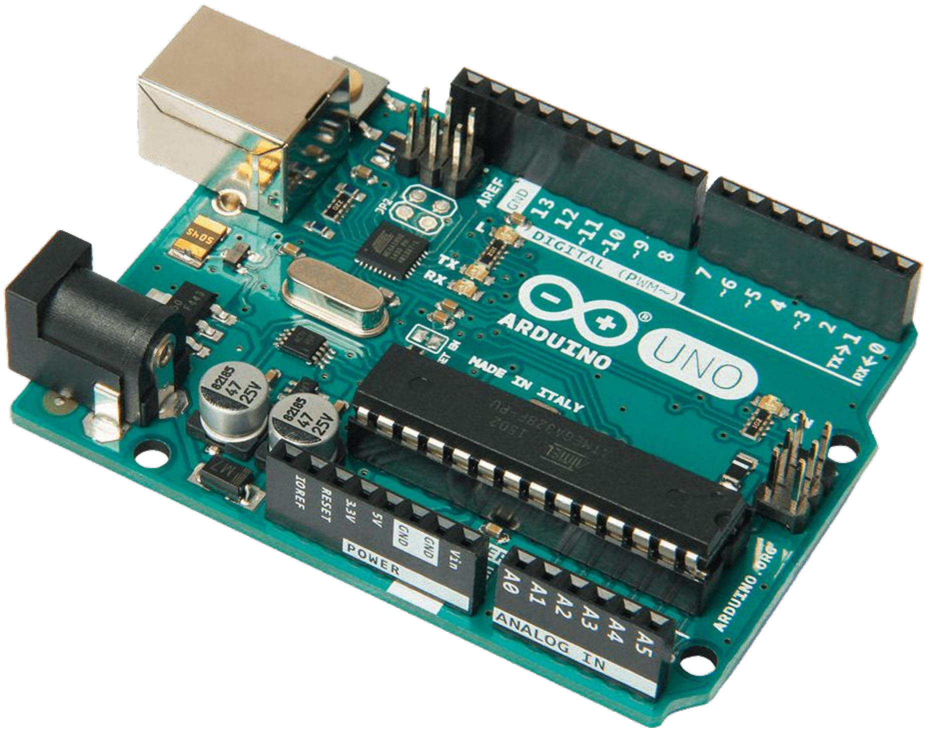
What is a Arduino?

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

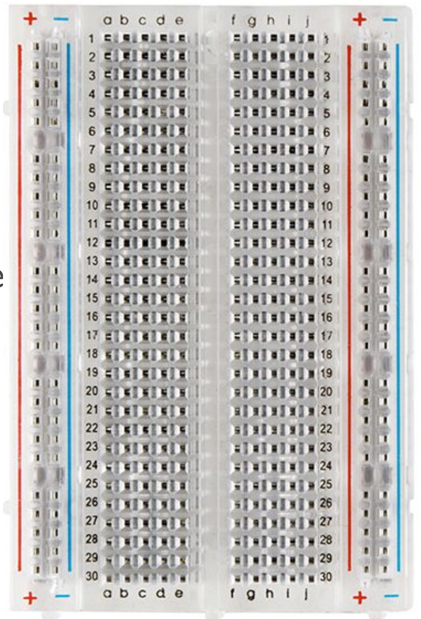
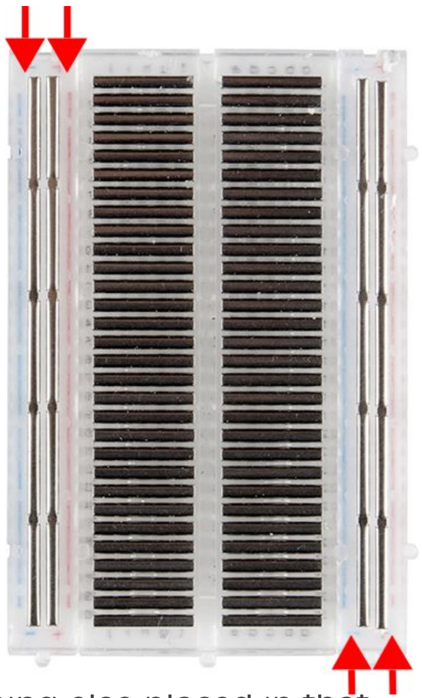
- Arduino Uno Kit
  - USB Cable
  - White LED
  - Breadboard
  - Button switch
- Computer with Arduino Create



Arduino Fibonacci Sequence

What is a breadboard?

An electronics breadboard is actually referring to a **solderless breadboard**. Solderless means that we don't have to make permanent attachments for our circuit components, but rather we can use the metal conductors within the board to conduct electricity between the rows and columns. These are great units for making temporary circuits and prototyping, and they require absolutely no soldering.

Once inserted that component will be electrically connected to anything else placed in that row. This is because the metal rows are conductive and allow current to flow from any point in that strip.

You'll also notice that each horizontal row is separated by a ravine, or crevasse, in the middle of the breadboard. This ravine isolates both sides of a given row from one another, and they are not electrically connected.

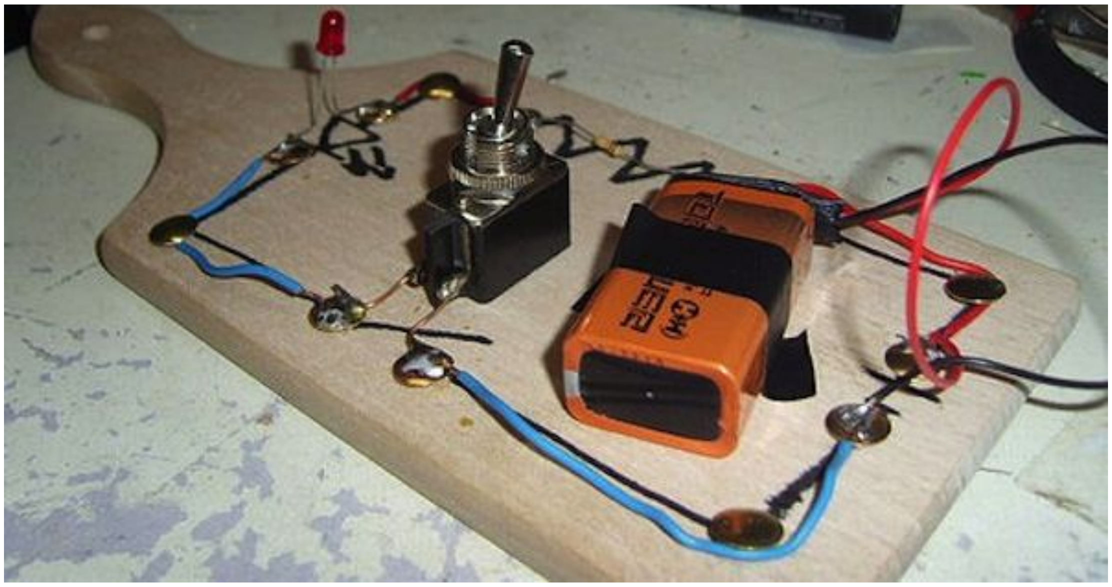
These power rails are metal strips that are identical to the ones that run horizontally, except they are, typically*, all connected. When building a circuit, you tend to need power in lots of different places. The power rails give you lots of easy access to power wherever you need it in your circuit. Usually they will be labeled with a '+' and a '-' and have a red and blue or black stripe, to indicate the positive and negative side.

It is important to be aware that the power rails on either side are not connected, so if you want the same power source on both sides, you will need to connect the two sides with some jumper wires. Keep in mind that the markings are there just as a reference. There is no rule that says you have to plug power into the '+' rail and ground into the '-'rail, though it's good practice to keep everything in order.

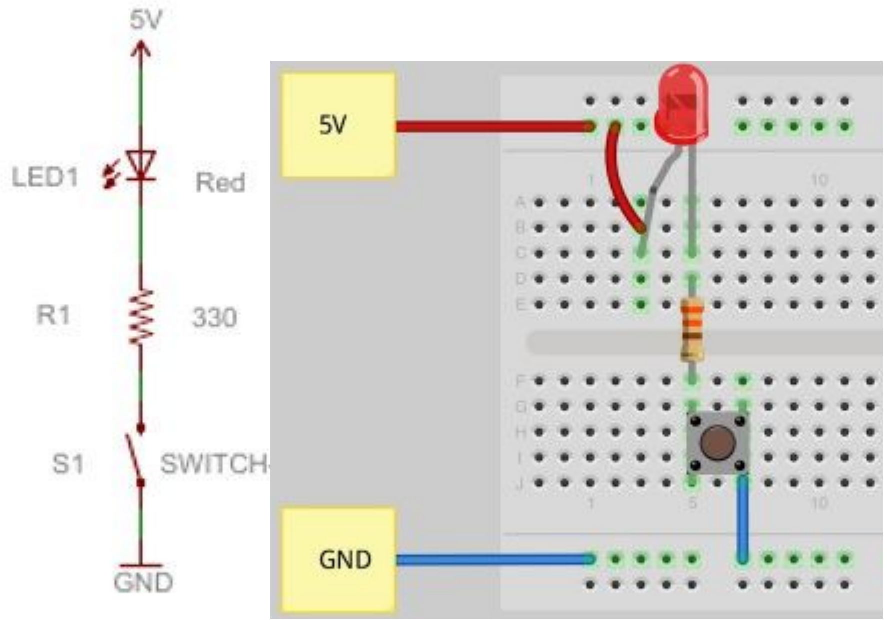
- As long as all the electrical connections are being made, you can build your circuit any way you'd like!
- You can simply pull power from the Arduino's female headers. The Arduino has multiple power and ground pins that you can connect to the power rails or other rows on a breadboard.
- The Arduino usually gets its power from the USB port on a computer or an external power supply such as a battery pack

Arduino Fibonacci Sequence

So why do we call this electronic “circuit builder” a breadboard? Many years ago, when electronics were big and bulky, people would grab their mom’s breadboard, a few nails or thumbtacks, and start connecting wires onto the board to give themselves a platform on which to build their circuits.



Example breadboard connection for Button switch LEDs:



Arduino Fibonacci Sequence

The **Fibonacci Sequence** is a series of numbers, wherein the next number is found by adding up the two numbers before it.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, ...

We can write the rule:

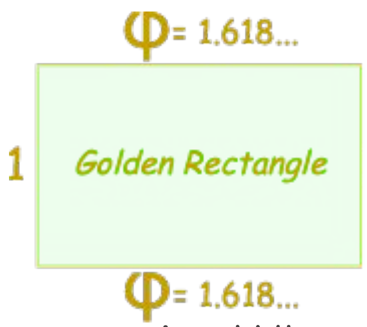
The Rule is $x_n = x_{n-1} + x_{n-2}$

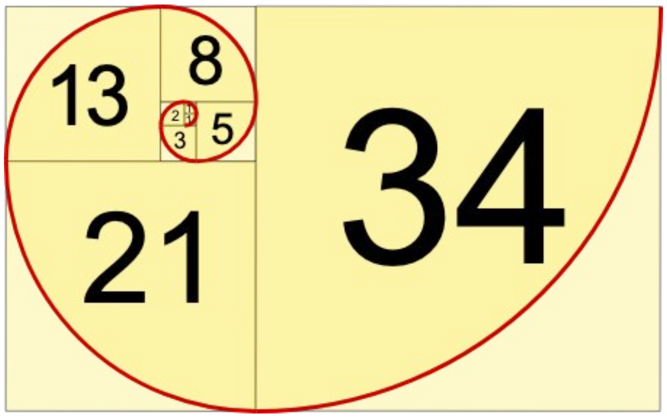
where:

- x_n is term number "n"
- x_{n-1} is the previous term (n−1)
- x_{n-2} is the term before that (n−2)

Example: term 9 is calculated:

$$\begin{aligned} x_9 &= x_{9-1} + x_{9-2} \\ &= x_8 + x_7 \\ &= 21 + 13 \\ &= 34 \end{aligned}$$





And here is a surprise. When we take any two successive *(one after the other)* Fibonacci Numbers, their ratio is very close to the **Golden Ratio "φ"** which is approximately **1.618034...**

In fact, the bigger the pair of Fibonacci Numbers, the closer the approximation.

You can also calculate a Fibonacci Number by multiplying the previous Fibonacci Number by the Golden Ratio and then rounding (works for numbers above 1)

Arduino Fibonacci Sequence

Fibonacci.ino

```
/*

A sequence is a list of numbers.
For some sequences, you pick some of the first few numbers. Then, the next
number in the sequence is a result of the older ones.

For the Fibonacci sequence, we start with the first two numbers being 1.
To get the next number in this sequence, you add together the last two numbers
that are in the list.


	Circuit:
	button to Arduino Pin4

*/

// VARIABLES
const int button = 4;			// button Pin
int buttonState;			// will hold the current state of button
int lastButtonState = HIGH;		// will store the previous state of button

int a = 1;			// variables used to calculate the Fib. numbers
int b = 1;		// the values for a and b are the "seed" for the sequence
int c;			// c is used to get the next number in the sequence.


void setup() {
  pinMode(button,INPUT);			// set button as input
  Serial.begin(9600);		// begin serial communication
  delay(2000);			// 2s delay, so we can start the Serial Monitor
}
```

Arduino Fibonacci Sequence

Fibonacci.ino

```
void loop() {
  buttonState = digitalRead(button);   // read the button state

                                       // if button is just pressed
                                       // this logic tries to make sure that each button
                                       // press only causes one new line of Fib numbers.
  if(buttonState == LOW && buttonState != lastButtonState){
        c = a + b;                     // calculate next Fib. number
        printNumbers(a,b,c);                // send the last three values to the print function
        a = b;                         // The last number becomes the second-to-last
        b = c;                         // The new number becomes the last number in the list
  }
  lastButtonState = buttonState;      // store last button state
}



void printNumbers(int d, int e, int f){
  // this funcion is used to print the Fib. numbers into a "table" on the Serial Monitor

  Serial.print(d);                     // print F[n-2]
  Serial.print("\t");
  Serial.print("+");
  Serial.print("\t");
  Serial.print(e);                     // print F[n-1]
  Serial.print("\t");
  Serial.print("=");
  Serial.print("\t");
  Serial.println(f);                   // print F[n], the new Fib. number
  // notice how the println function adds a newline to the end (compared to print, which
doesn't)
}
```

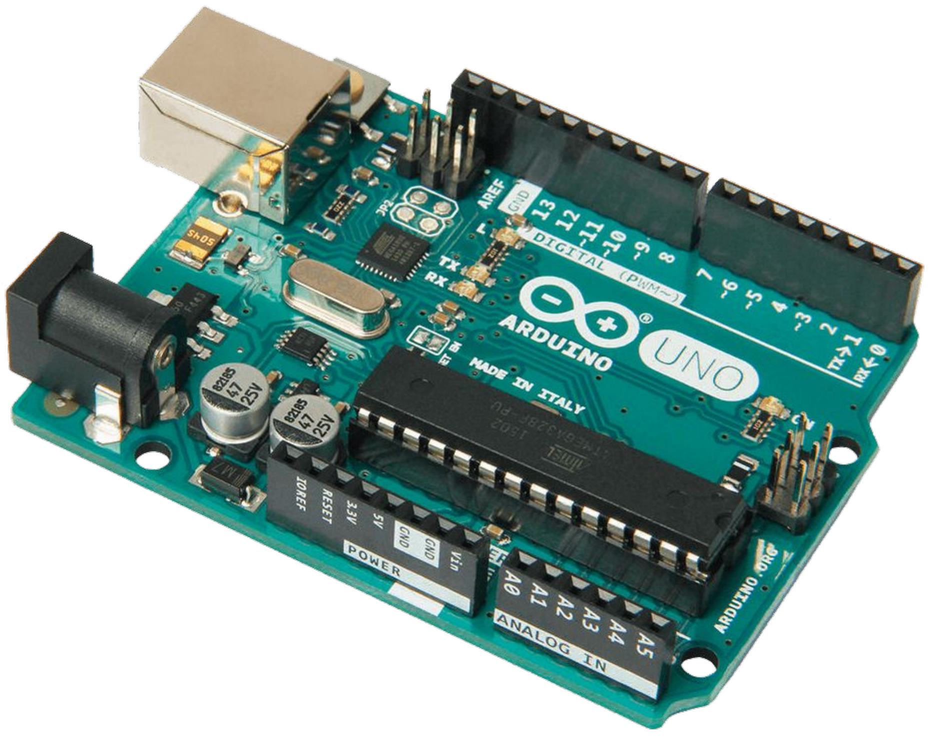
Arduino Fibonacci Sequence

As you can see in our output, when we get to the:
17711 + 28657 =
instead of getting 46368 we get -19168

Why is this happening?

Integers are **16 bit** on the Arduino with on bit reserved for the sign in the signed version, so 32767 is the highest number such a variable is able to hold, afterwards it overflows and goes negative. If you want to play with higher numbers, use the long version or better the explicit types: **int32_t**.

The **32 bit** unsigned number is being truncated to a 16 bit signed number type.



Arduino Fibonacci Sequence

Tribonacci.ino

```
/*

A sequence is a list of numbers.
For some sequences, you pick some of the first few numbers. Then, the next
number in the sequence is a result of the older ones.

For the Tribonacci sequence, we start with the first two numbers being 0,
and the third number being 1.
To get the next number in this sequence, you add together the last three numbers
that are in the list.

        Circuit:
        button to Arduino Pin4

*/

// VARIABLES
const int button = 4;           // button Pin
int buttonState;                // will hold the current state of button
int lastButtonState = HIGH;             // will store the previous state of button

int a = 0;                      // variables used to calculate the Trib. numbers
int b = 0;              // the values for a, b and c are the "seed" for the sequence
int c = 1;                      // d is used to get the next number in the sequence.
int d;

void setup() {
  pinMode(button,INPUT);                // set button as input
  Serial.begin(9600);           // begin serial communication
  delay(2000);                  // 2s delay, so we can start the Serial Monitor
}
```

Arduino Fibonacci Sequence

```
void loop() {
  buttonState = digitalRead(button);   // read the button state

                                   // if button is just pressed
                                   // this logic tries to make sure that each button
                                   // press only causes one new line of Trib numbers.
  if(buttonState == LOW && buttonState != lastButtonState){
        d = a + b + c;                 // calculate next Trib. number
        printNumbers(a,b,c,d);           // send the last four values to the print function
        a = b;
        b = c;
        c = d;                         // these three assignment lines shuffle each of the
                                   // variables forward one space in the sequence
  }
  lastButtonState = buttonState;     // store last button state
}

void printNumbers(int e, int f, int g, int h){
  // this funcion is used to print the Trib. numbers into a "table" on the Serial Monitor

  Serial.print(e);                     // print the third to last Trib. number
  Serial.print("\t");
  Serial.print("+");
  Serial.print("\t");
  Serial.print(f);                     // print the second to last Trib. number
  Serial.print("\t");
  Serial.print("+");
  Serial.print("\t");
  Serial.print(g);                     // print the last Trib. number
  Serial.print("\t");
  Serial.print("=");
  Serial.print("\t");
  Serial.println(h);                   // print the new Trib. number
  // notice how the println function adds a newline to the end (compared to print, which
doesn't)
}
```

#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