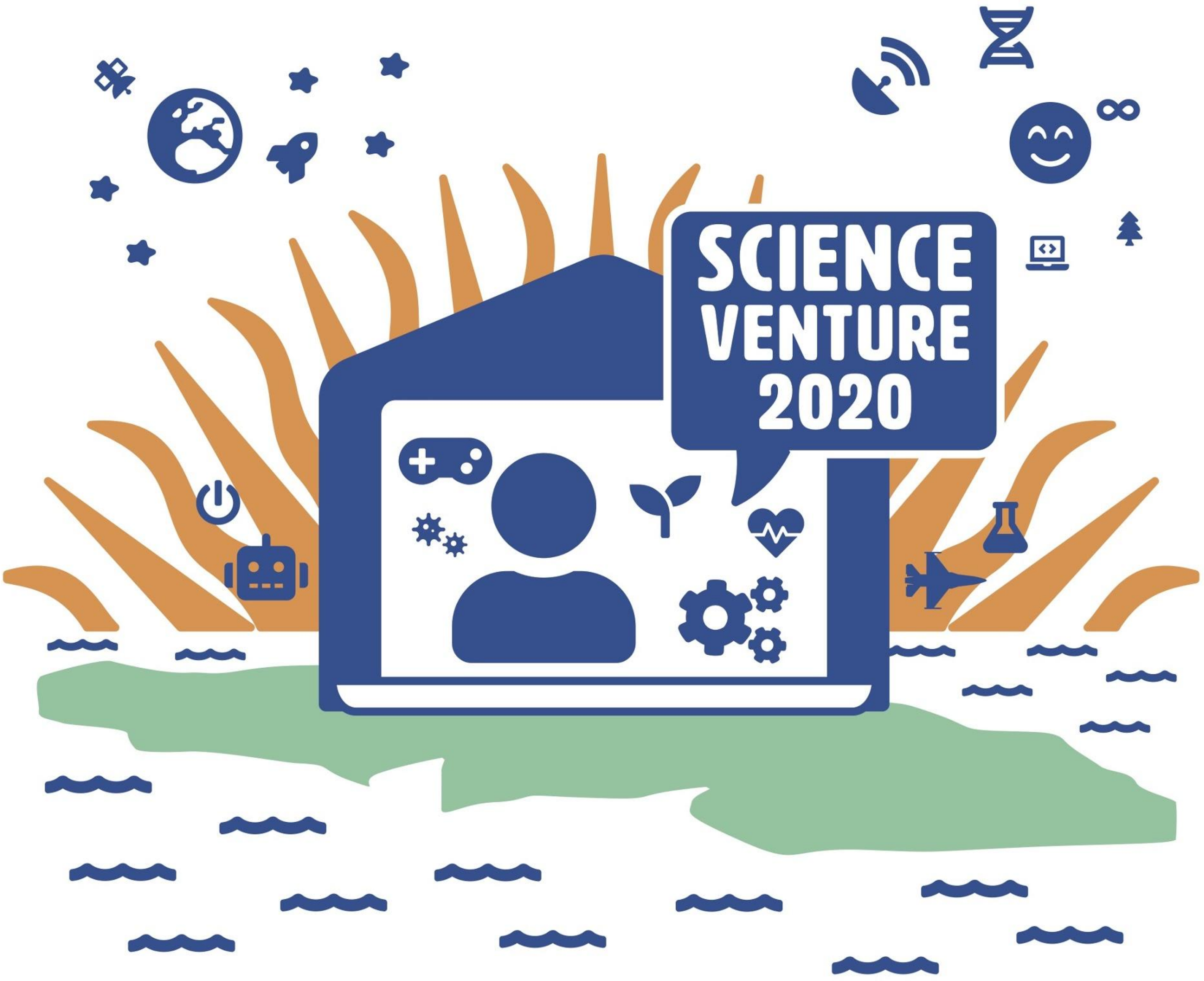


Arduinos

Arduino Metronome

Experienced C++ coding for Arduino. Practice using while and if/else loops in the Arduino web editor. Learn about Servo and DC motors, and how we can add motion to our Arduino circuit. Code a metronome project and set the beats per minute (BPM) to flash an LED in time with the metronome. Add motor movement to your metronome and customize Arduino microcontroller, breadboard wiring, and C++ code.

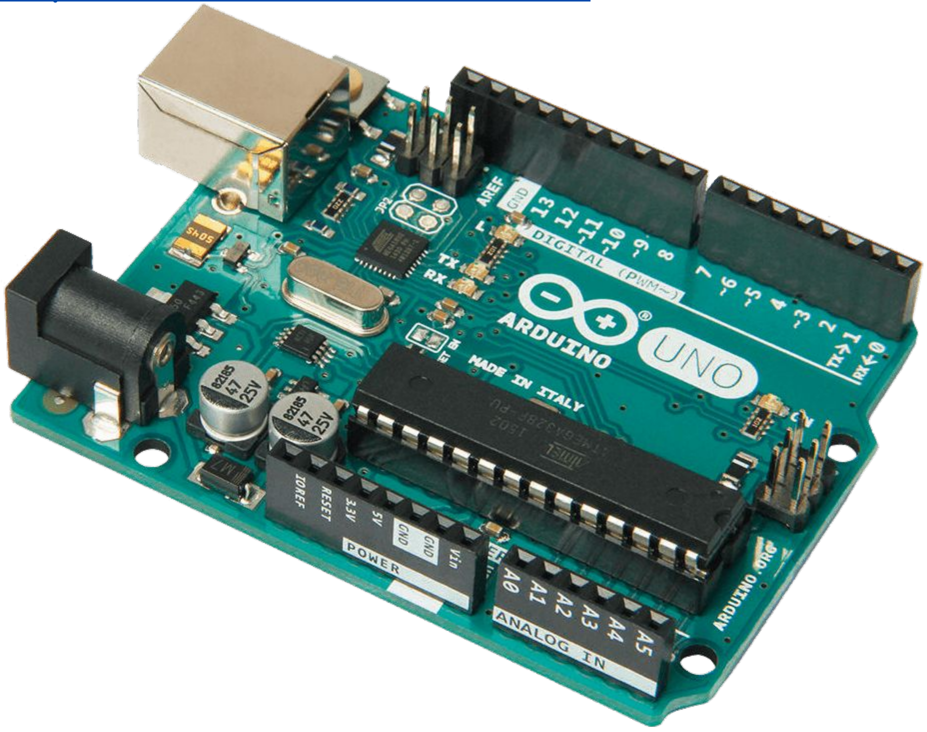
What is a Arduino?

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
  - 5 LEDs
  - Breadboard
- Computer with Arduino Create
- Tape, glue or sticky putty



Arduino Metronome

Information on Metronomes

A **metronome** has a mechanical arm that swings back and forth. One "beat" is the time it takes for the metronome to swing from the middle, to one side, back to the middle again.

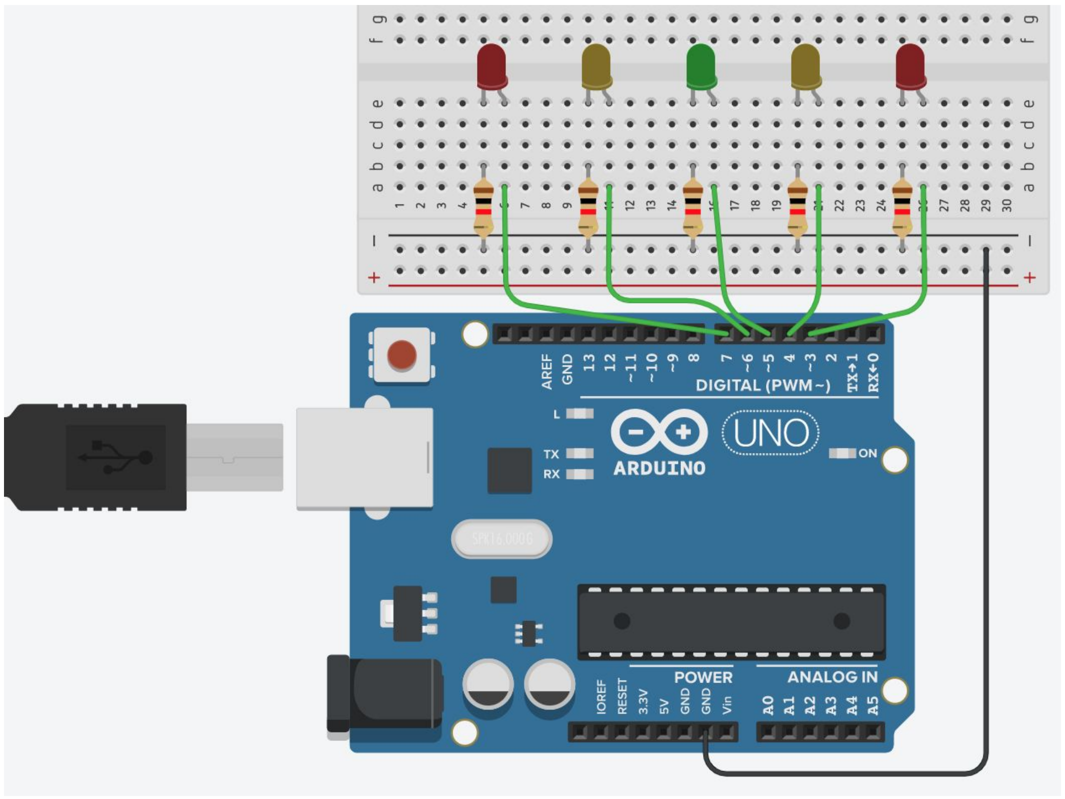
Musical timing is expressed in beats per minute (bpm). So, we want to be able to give the metronome a speed in units of bpm, and then have the program calculate how quickly to "swing" back and forth. Rather than using a mechanical arm, in this program we will light LEDs in a pattern that moves back and forth.

Metronome Coding

Adapted from https://www.instructables.com/id/DIY-Metronome/

Changed to blink an array of LEDs instead of buzzing the piezo.

Wiring Diagram:



Arduino Metronome

Metronome.ino - LED

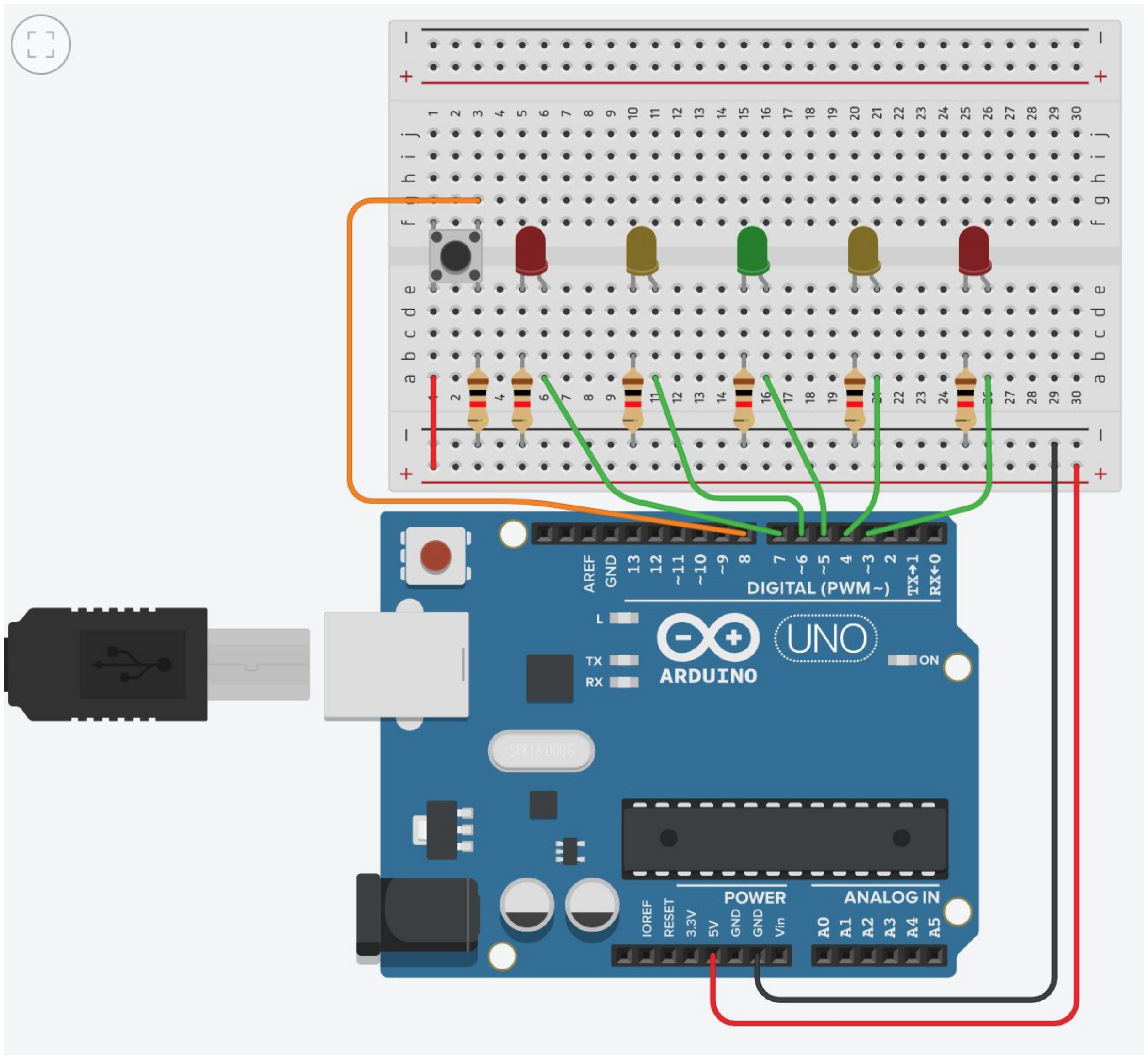
```
/*  for Science Venture, June 2020
*/

// declare the constants for the five LEDS
const int LED1 = 3;
const int LED2 = 4;
const int LED3 = 5;
const int LED4 = 6;
const int LED5 = 7;

// calculating the speed required
float ledpb = 4;               // LEDs per beat. led3 is the "on beat" so, 3, 4, 5, 4 would be one beat, (or 3, 2, 1, 2).
float bpm = 60;                // number of beats in one minute
float ledpm = (ledpb*bpm);       // LEDs per minute
float ledpms = ledpm/(60.*1000.);   // LEDs per millisecond
float mspled = 1/ledpms;         // reciprocal of frequency is period (1/f=T)

int ledArray[8] = {LED1, LED2, LED3, LED4, LED5, LED4, LED3, LED2};
int led_index;

void setup() {
    Serial.begin(9600);        // initialize Serial connection
    led_index = 0;

    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
    pinMode(LED4, OUTPUT);
    pinMode(LED5, OUTPUT);
}

void loop() {

  // Blink the LED when MIN angle is reached
    digitalWrite(ledArray[led_index],HIGH);
    delay(mspled);
    digitalWrite(ledArray[led_index],LOW);

    led_index++;
    if (led_index >= 8) {
        led_index = 0;
    }
}
```

Arduino Metronome

Extension

Make a timer activated by a button press. Use LEDs and the Serial monitor to show when time has run up, and optionally add Servo or DC motor code to show motion as the timer counts down. Use the same breadboard configuration from the Fibonacci program, and adapt the same button code to initiate the timer countdown.

Diagram for wiring the button switch on the breadboard - the button part is similar to the Fibonacci Project, but now we have five coloured LEDs as well.



Arduino Metronome

timerCountdownButton.ino - using LEDs

```
/*
  for Science Venture, June 2020
  Starts a 10 second time when the button is pressed. (then resets after two seconds)
*/

// declare the constants for the five LEDS
const int LED1 = 3;
const int LED2 = 4;
const int LED3 = 5;
const int LED4 = 6;
const int LED5 = 7;

// declare button variables
const int button = 8;           // button Pin
int buttonState;                // will hold the current state of button

int timer_length = 10;          // timer length (integer in seconds)
float timer_length_ms = timer_length*1000.;   // timer length in milliseconds
float increment = timer_length_ms/5;

int ledArray[5] = {LED5, LED4, LED3, LED2, LED1};
int led_index = 0;

void setup() {
        Serial.begin (9600);
        pinMode(button,INPUT);  // set button as input

         pinMode(LED1, OUTPUT);
         pinMode(LED2, OUTPUT);
         pinMode(LED3, OUTPUT);
         pinMode(LED4, OUTPUT);
         pinMode(LED5, OUTPUT);

        delay(2000);            // give the serial connection time to start
        Serial.println("Welcome to the timer program");
}
```

Arduino Metronome

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

        // start timer if button is pressed:
        if(buttonState == HIGH) {

                Serial.print("Starting timer for: ");
                Serial.print(timer_length, DEC);
                Serial.println(" seconds.");

                 for (led_index = 0; led_index <= 4; led_index++){

                        digitalWrite(ledArray[led_index],HIGH);
                        delay(increment);
                        digitalWrite(ledArray[led_index],LOW);
         }

          Serial.println("Time's up!");
          led_index = 0; // reset the LED timer position
          delay(2000);
        }
}
```

#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