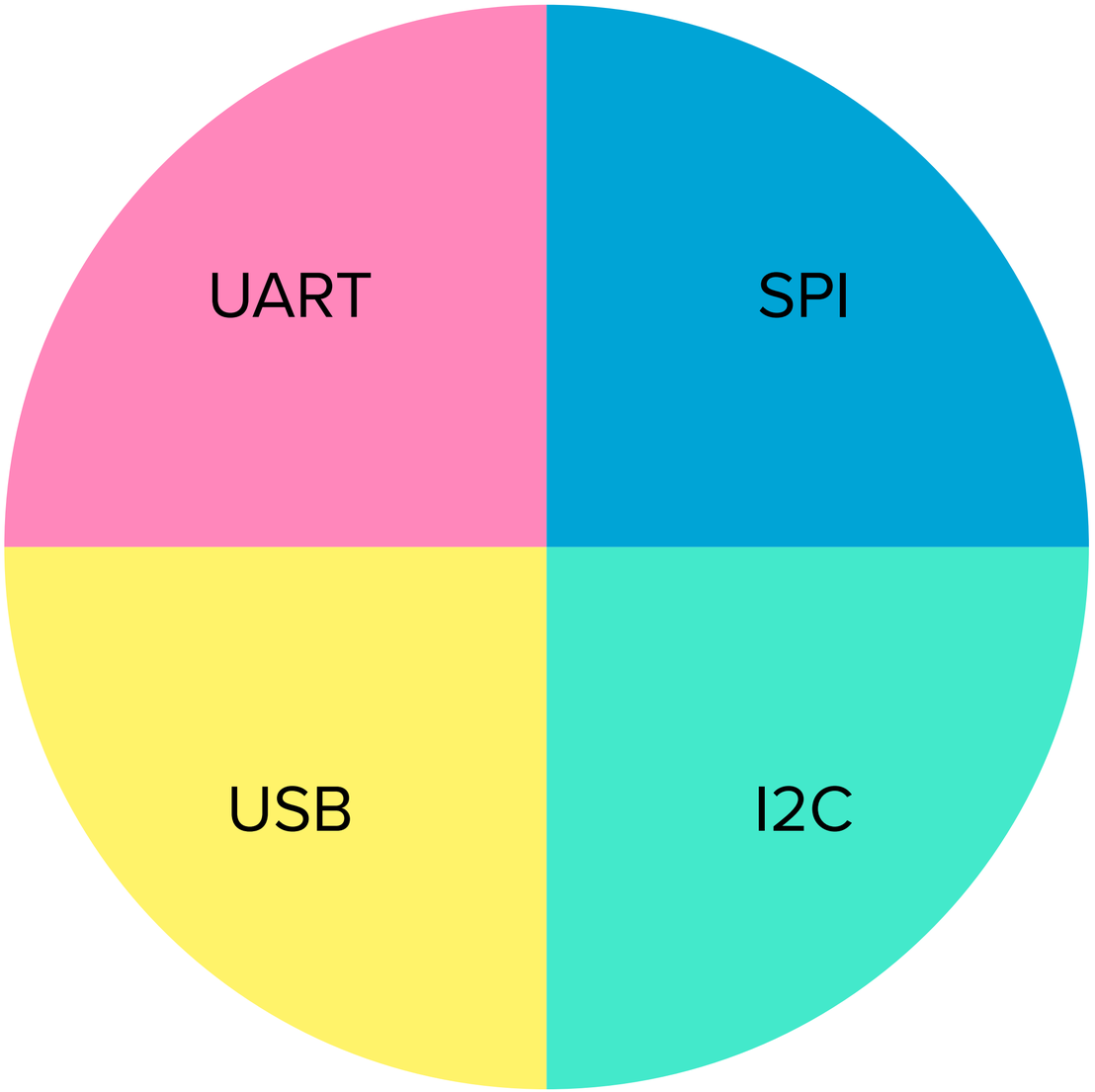
INTERACTION DESIGN

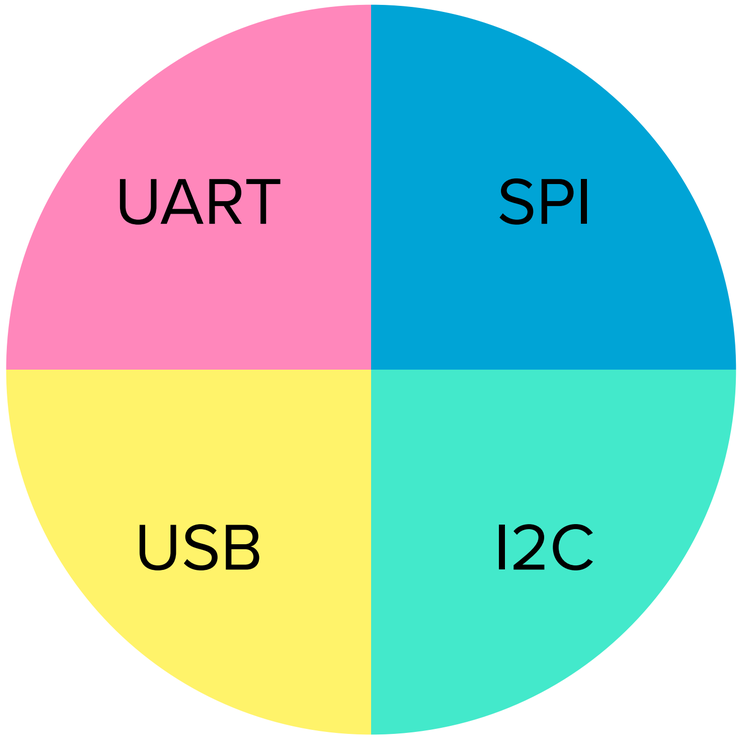
ARDUINO + P5.JS

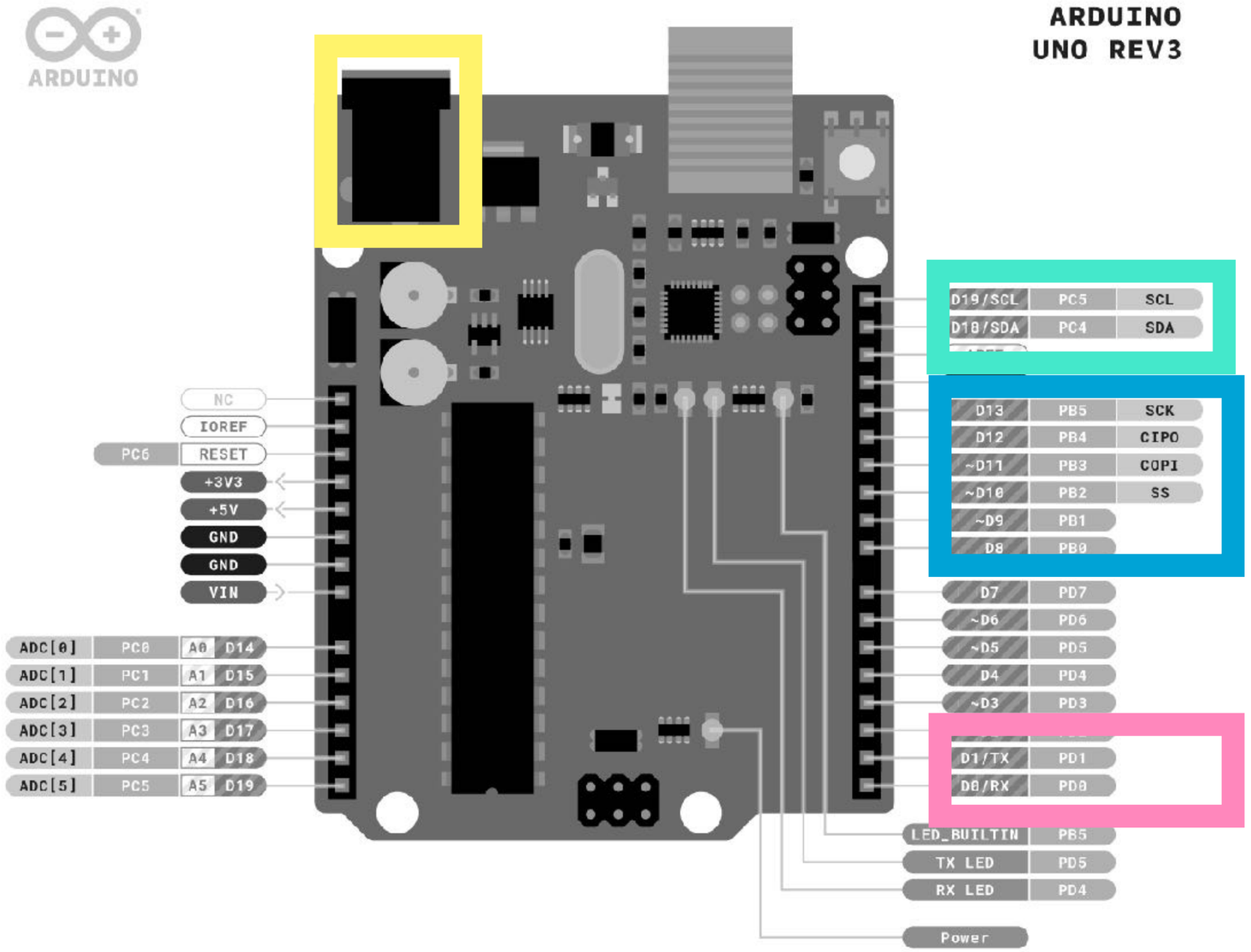
Physical Computing HS21

SERIAL COMMUNICATION

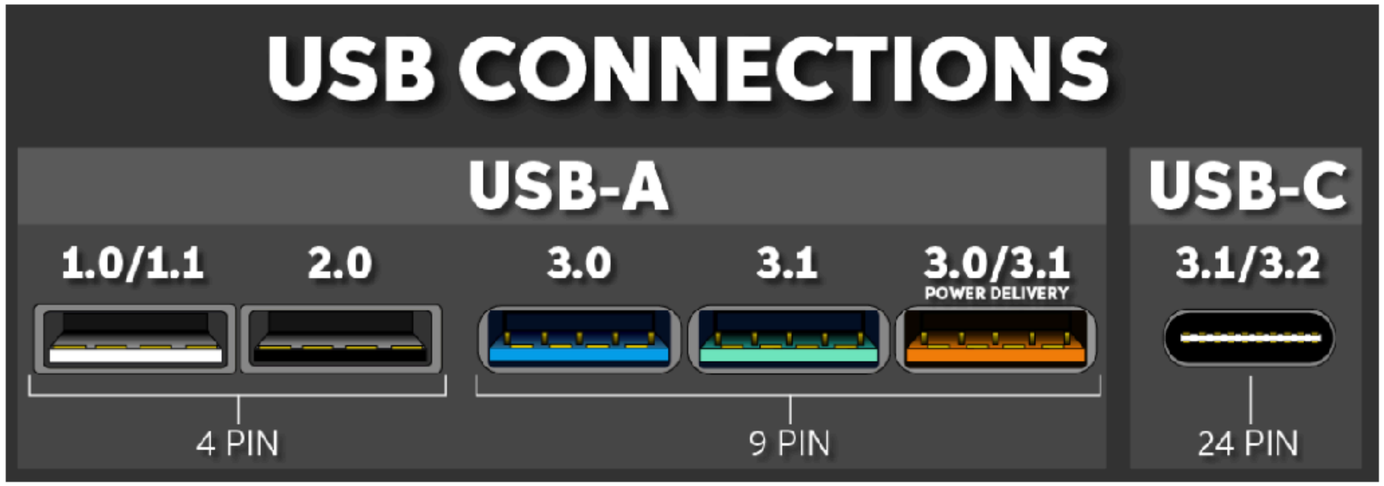


SERIAL COMMUNICATION





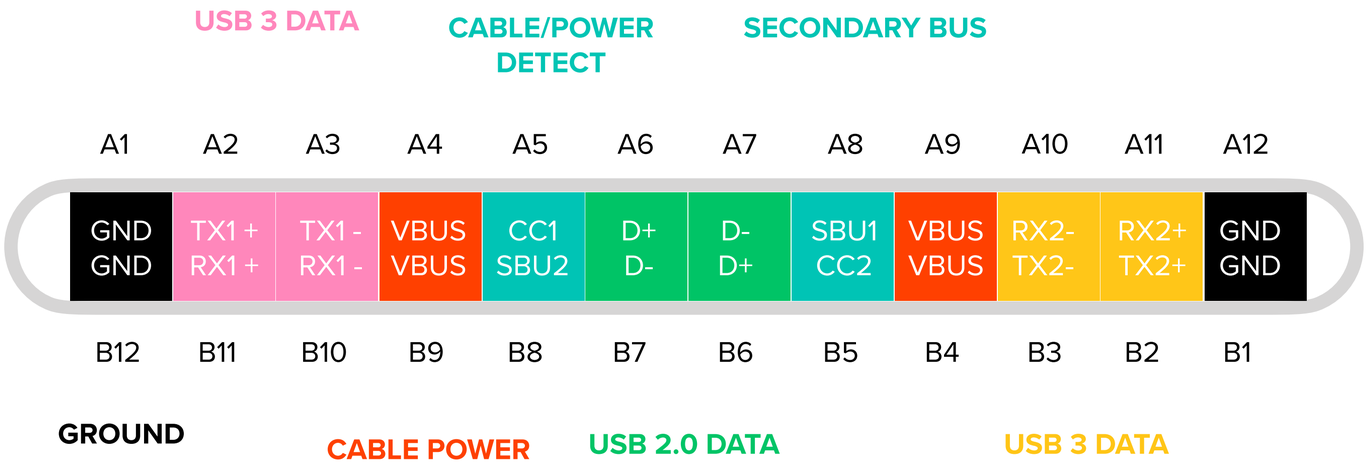
USB SERIAL COMMUNICATION



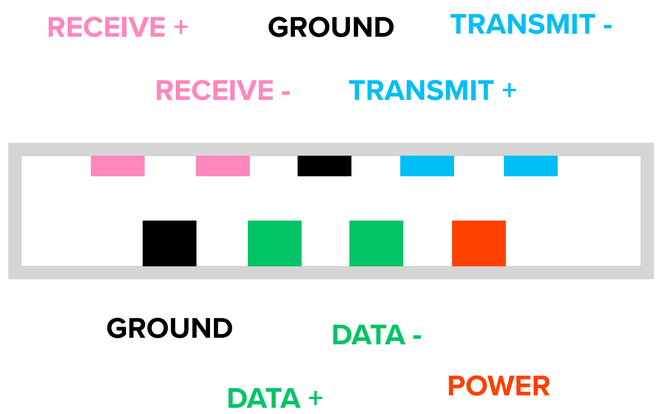
USB 4	Up to 40 Gbps	Up to 100W at 20V
USB 3 (USB 3.1 Gen 2)	Up to 10 Gbps	Up to 15W at 5V
USB 3 (USB 3.1 Gen 1)	Up to 5 Gbps	Up to 900 mA at 5V
USB 2	Up to 480 Mbps	Up to 500 mA at 5V
USB 1.1	Up to 12 Mbps	Up to 500 mA at 5V

Image source

USB SERIAL COMMUNICATION



USB-C

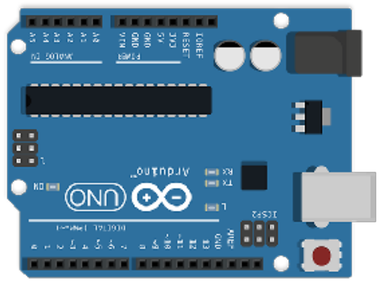


USB-A 3.0

ARDUINO <-> PROCESSING

```
import processing.serial.*;
```

```
Serial.begin(9600);
```

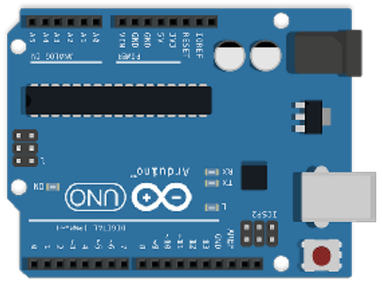


Link to WIKI

ARDUINO <-> P5.js

```
?????
```

```
Serial.begin(9600);
```



WHY P5?

EXERCISE 1

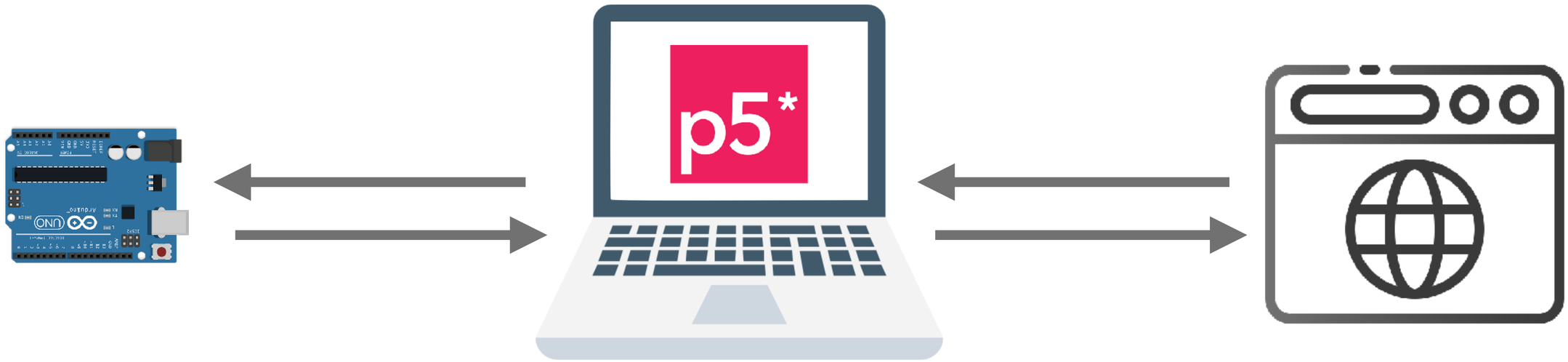
Working in pairs convert processing sketch to p5.js sketch

Person A gives 5 drawing rules to person B e.x:

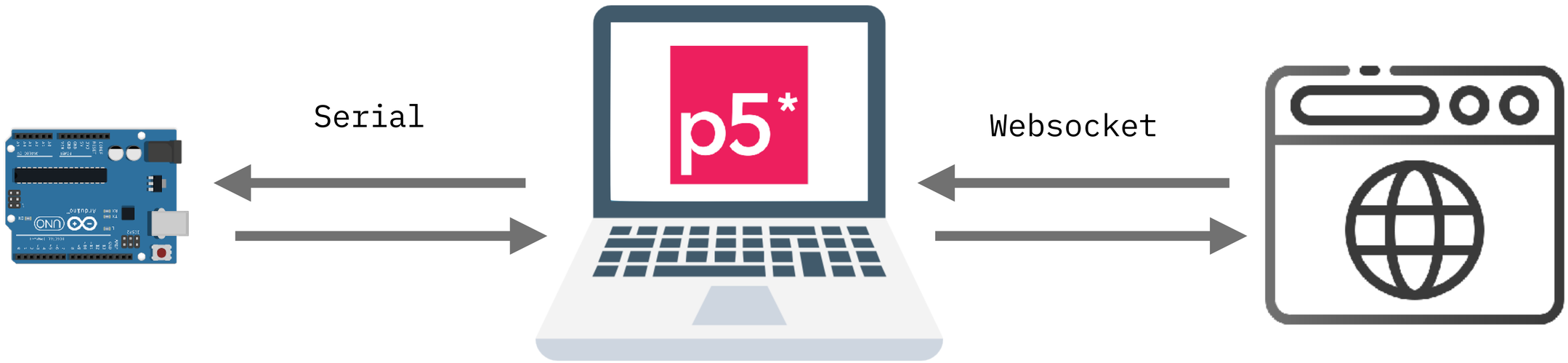
Draw a white circle with 10 diameter in the middle of a screen, that will be multiplied in size by 2 each time mouse is clicked. When size is bigger than 100, decrease the size by 4 till it reaches diameter 10.

Person A rewrites the processing sketch into p5.js sketch. Person B opens p5.js reference page and helps with finding the corresponding variables, functions etc.

ARDUINO <-> P5.js



ARDUINO <-> P5.js



ARDUINO <-> P5.js

WEB SERIAL API

The Web Serial API provides a way for websites to read from and write to serial devices. These devices may be connected via a serial port, or be USB or Bluetooth devices that emulate a serial port.

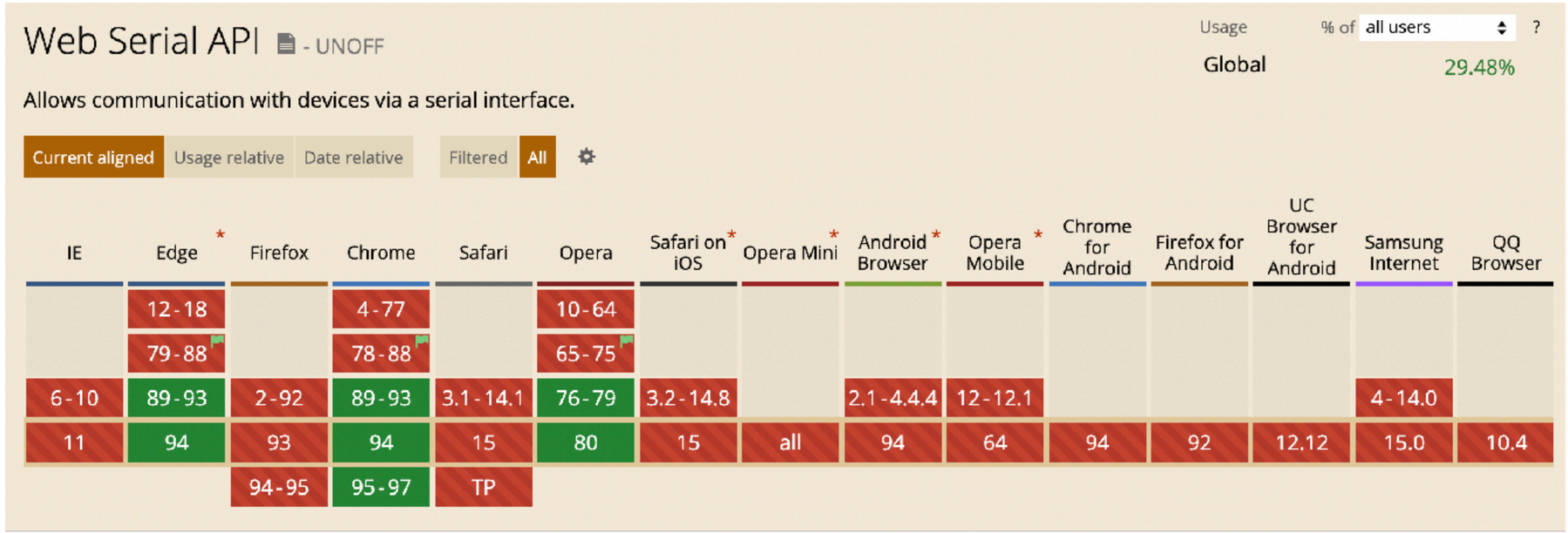


Image source

WEB SERIAL API

1. Enable flag `chrome://flags/#enable-experimental-web-platform-features`

2. Open dev tool console: `ctrl-shift-i` on Windows, `cmd-alt-i` on Mac

3. Type in console: `"serial" in navigator`

```
true //if supported
```

4. Plug in your Arduino board

5. Type in console: `await navigator.serial.requestPort();`

WEB SERIAL API

port.open(SerialOptions)

dictionary SerialOptions {

- baudRate: the only required option that must be an integer value like 9600 or 115200
- dataBits: The number of data bits per frame (either 7 or 8).
- stopBits: The number of stop bits at the end of a frame (either 1 or 2).
- parity: The parity mode (either “none”, “even” or “odd”).
- bufferSize: The size of the read and write buffers that should be created (less than 16MB).
- flowControl: The flow control mode (either “none” or “hardware”).

};

IDE

To work with p5.js and Arduino we’re going to use Processing IDE with p5.js mode enabled. In this mode Processing automatically creates index.html file, where you can add external libraries:

- Using CDN (like jsDelivr)

- Importing the library directly in ./libraries/ folder inside your Processing sketch. index.html will include the file automatically in the header

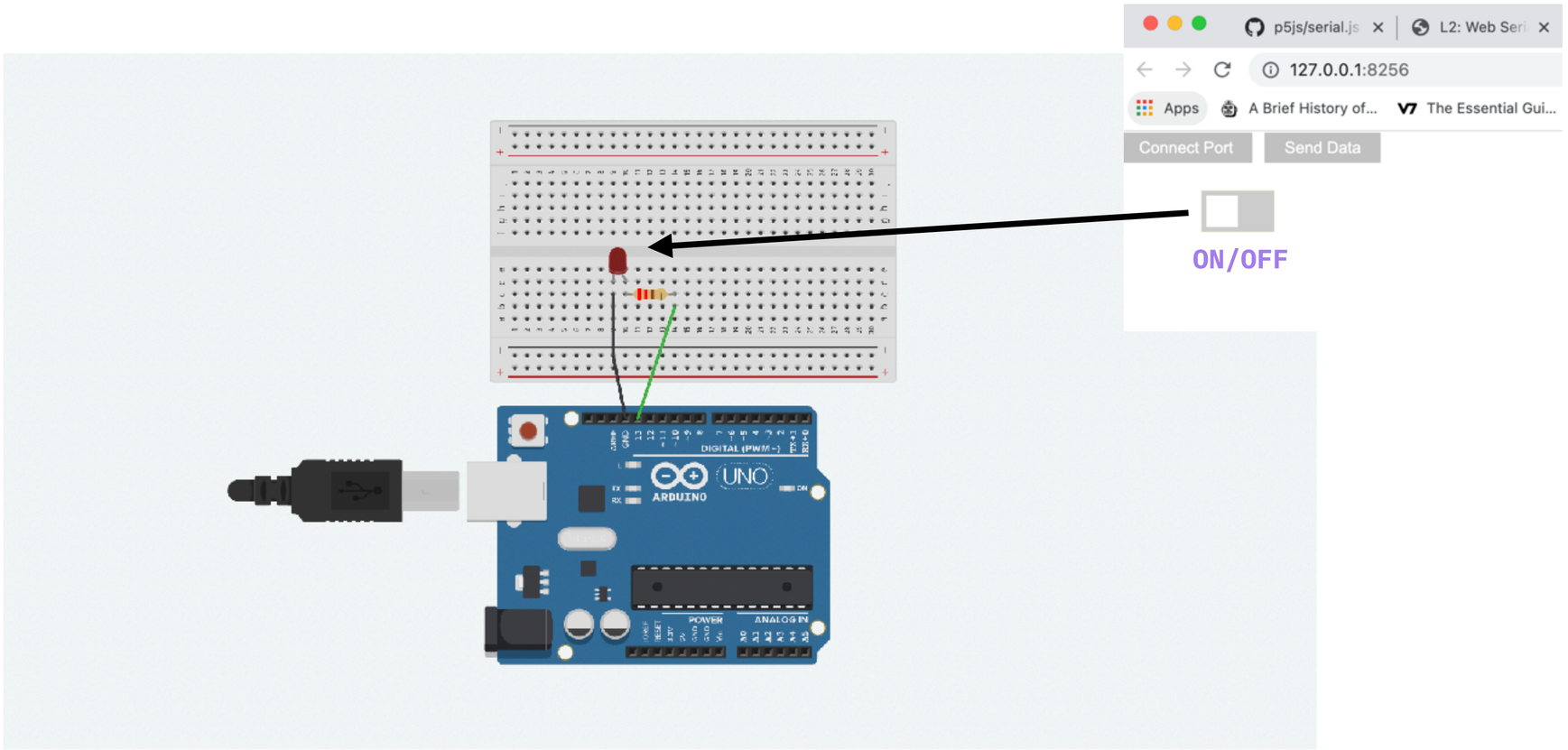
serial.js

To allow communication between an Arduino board and p5.js we'll use `serial.js` library written by Jon E. Froehlich. You can clone it from here and include `serial.js` or use the jsDelivr service:

```
<script src="https://cdn.jsdelivr.net/gh/makeabilitylab/p5js/_libraries/serial.js"></script>
```

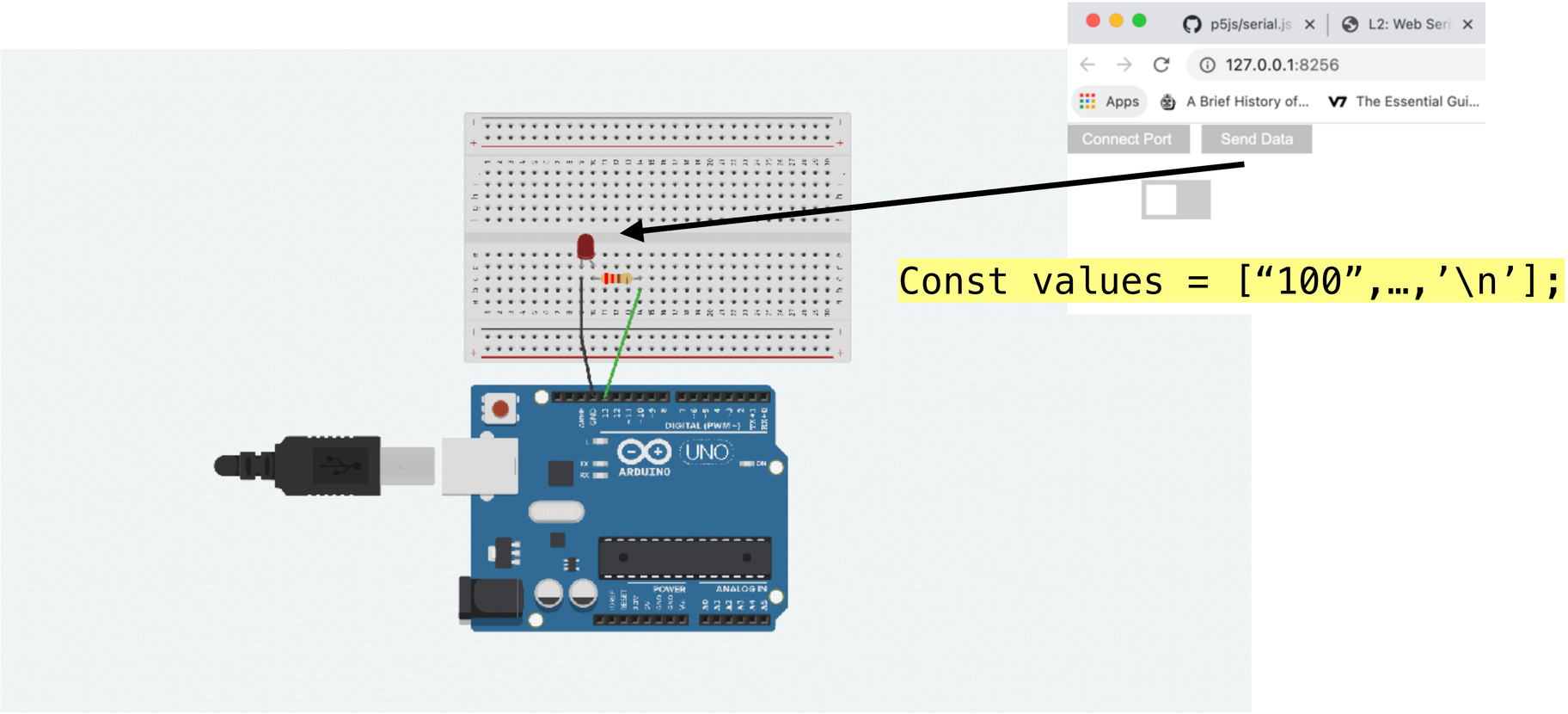
P5.JS -> ARDUINO

Make a LED light blink using web serial



Exercise 1

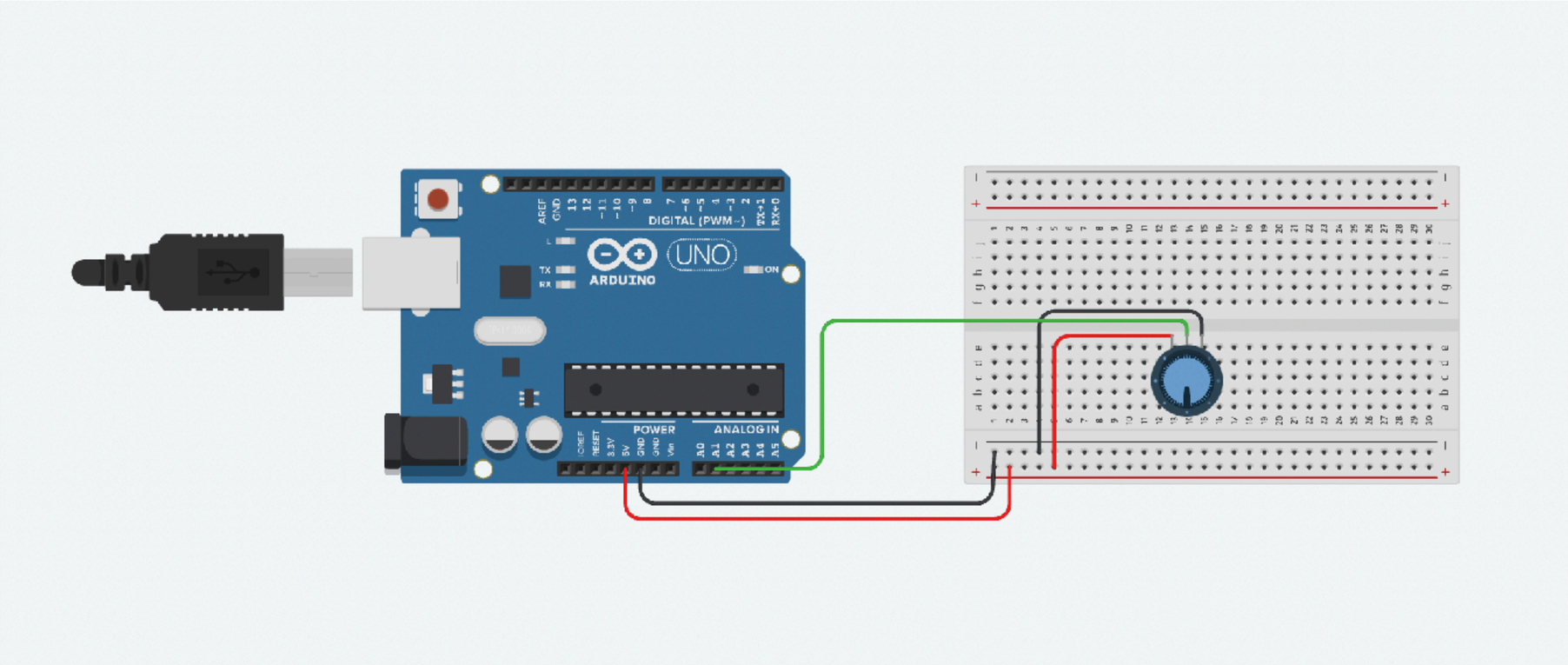
Make “Send Data” button control the brightness of LED by sending an array of numbers.



ADVANCED: Add a motor and change its speed based on a slider.
Turn the LED on once speed reached the maximum. Turn off when it’s 0.

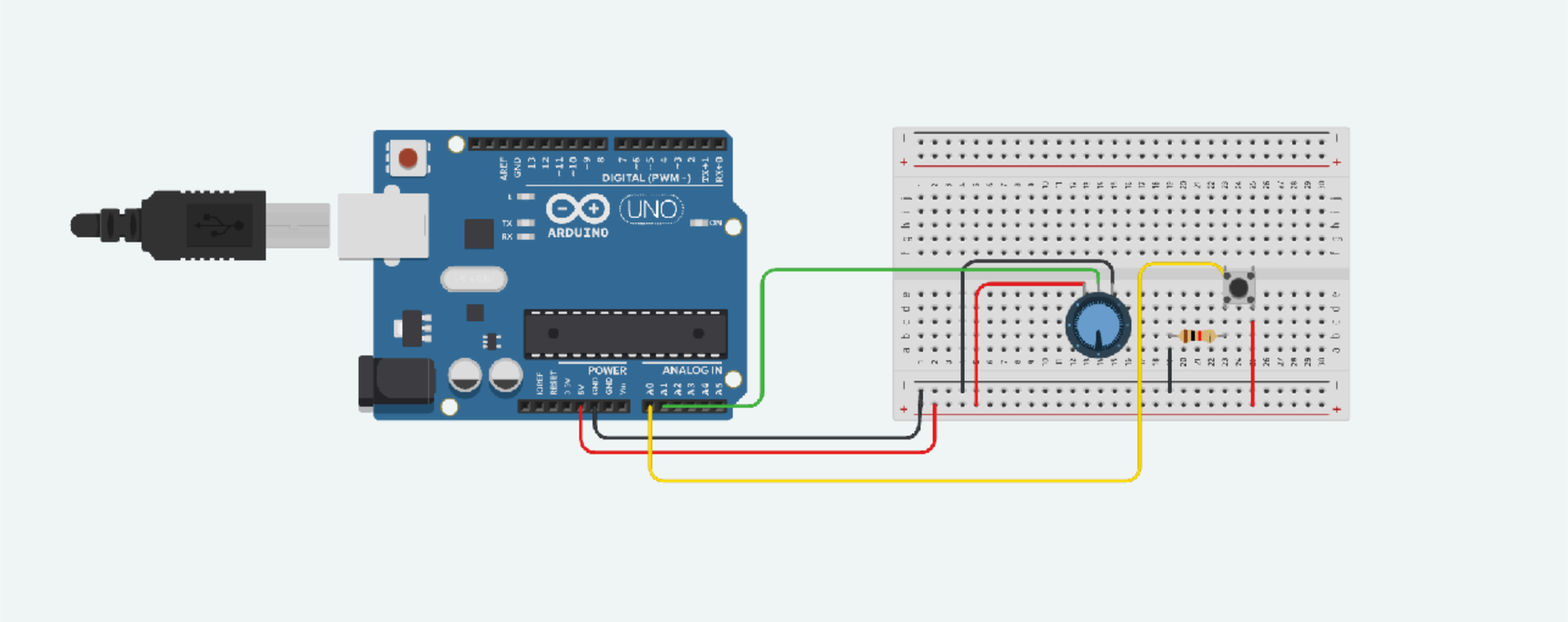
ARDUINO –> P5.JS

Manipulate a shape in p5.js sketch with potentiometer.



Exercise 2

Invert the sketch colours on each click of a button.



Exercise 2

Change the button behaviour so it will iterate through different shapes (eclipse, square, line) on each click.