EP1000

Digital I/O

Digital I/O

- The Uno uses the ATMega328 processor, which has 14 digital I/O pins (Some of these pins are multifunctional)

- A digital I/O pin can input or output digital (0, 5V) signals.

Digital I/O functions

- The Arduino system provides 3 functions for the manipulation of digital I/O.

- You need to
    1. Configure the pin (**pinMode()**), before
    2. Using the pin
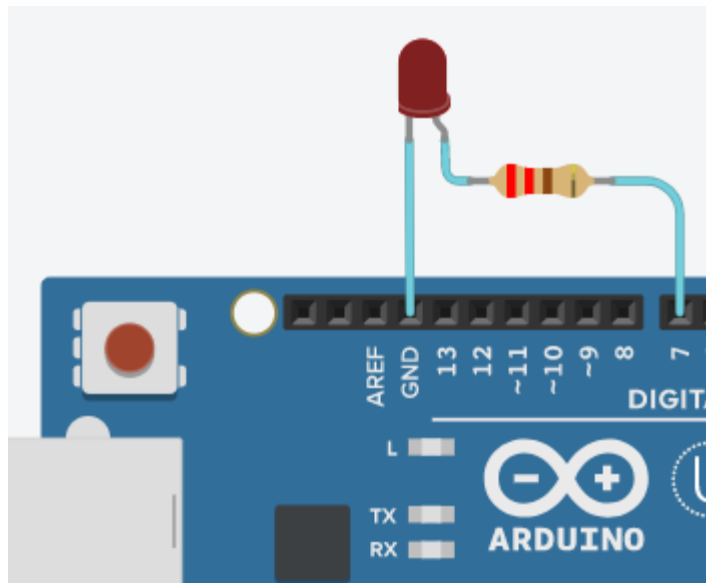   - **digitalWrite()** output
        - **digitalRead()** input

Digital I/O functions
• digitalRead() • digitalWrite() • pinMode()

pinMode(pin, MODE)

- Configures specified pin to behave either as in input or an output.

- Modes available:
    - INPUT
    digital input mode (high-impedance state)
    - INPUT_PULLUP
    digital input mode with internal 20K~50K ohm pull-up resistor
    - OUTPUT
    digital output mode able to source up to 40mA per pin, total of 200mA per chip

digitalWrite(pin, {LOW|HIGH})

- Outputs a LOW (0V) or HIGH (5V) to a digital pin.

- The digital pin must be configured as OUTPUT.


```
 2
 3  // Red LED connected to pin 7
 4  #define RED    7
 5
 6  ...
 7
 8  // set as digital OUTPUT
 9  pinMode(RED,OUTPUT);
10
11  // flash the LED
12  digitalWrite(RED, HIGH);
13  delay(300);
14  digitalWrite(RED, LOW);
15  delay(300);
16
```

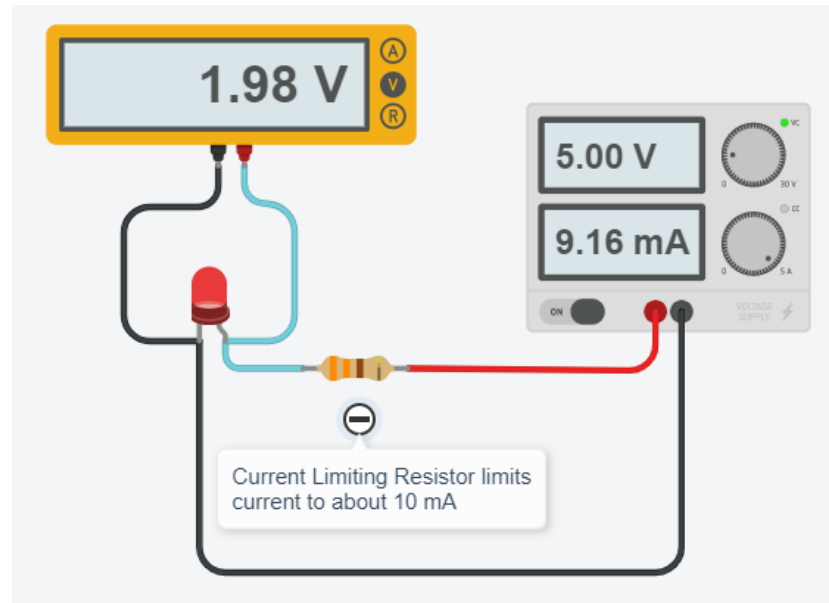You can also output a LOW to create a GND for sinking current!

Driving an LED

- An LED lights up (conducts) if a correct voltage is applied to the pins.

- When the LED conducts, current is allowed to pass through. The LED drops about 2V.

- We need to limit this current (10~20mA) otherwise, we will get a short-circuit.
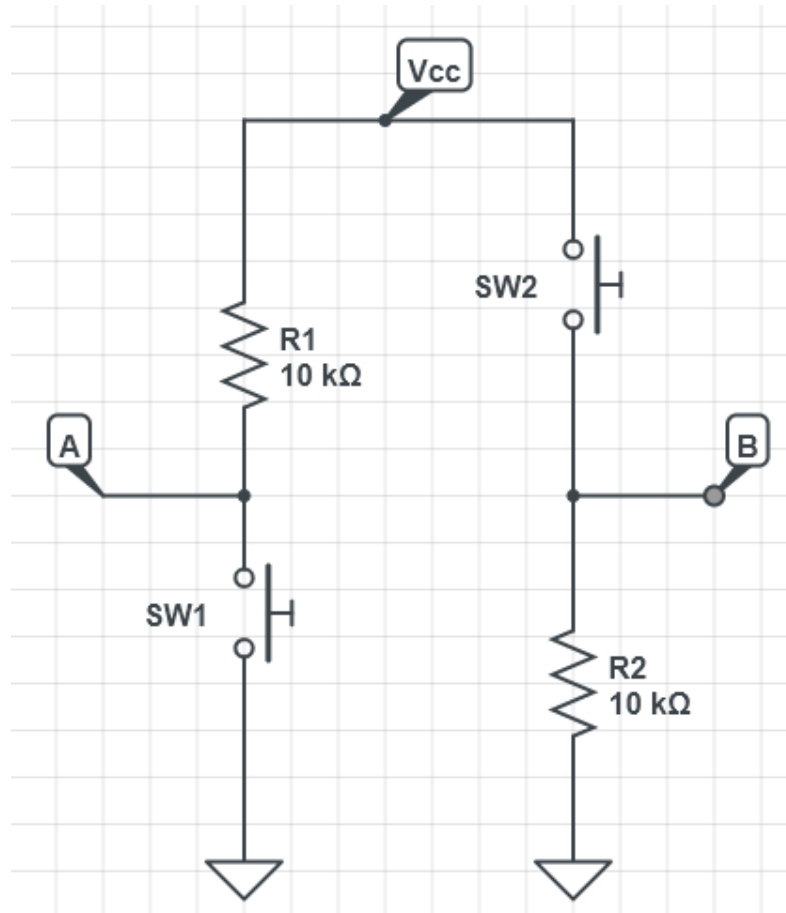
- Current limiting resistor value:
  R = V / I
     = (5 – 2V) / 10 mA
     = 300 ohms

LED current depends on type, check data sheet for forward voltage and current limits. Watch: GreatScott! Everything about LEDs

Digital Input

Circuits by CircuitLab

- We can use digital inputs to read the status of the switches in a circuit.

- Need to add a current-limiting resistor to prevent short circuits.

- Usual value is 10 kOhm

- States:
    - A - normal HIGH, when closed LOW
   - B – normal LOW, when closed HIGH

Digital Input
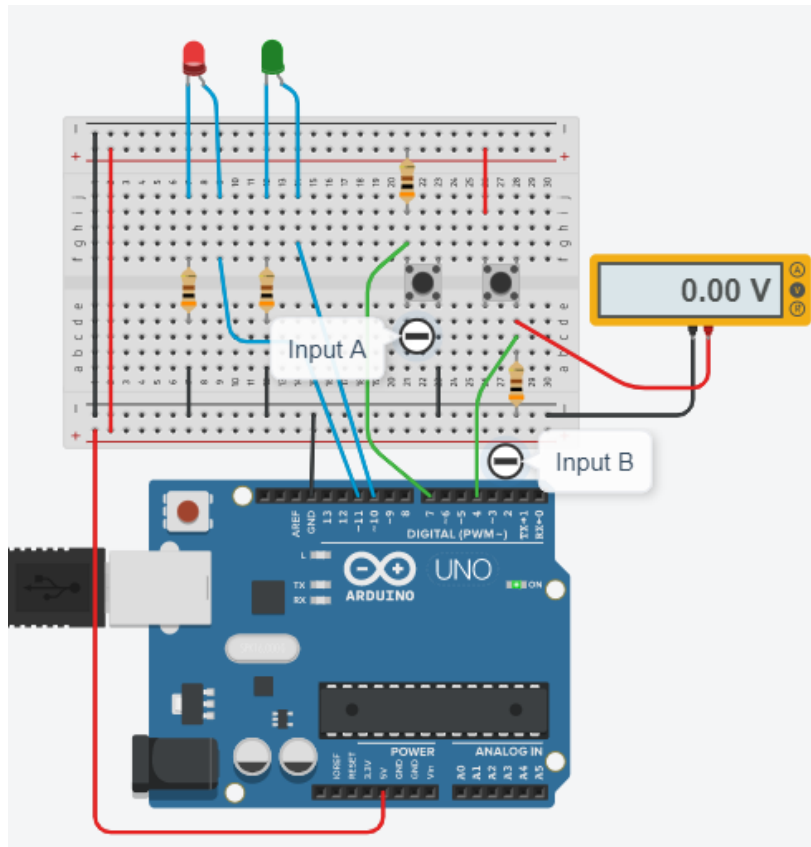
[How to use PB switches with digitalRead()](#)

```cpp
#define RED 11
#define GREEN 10
#define A    7
#define B    4

void setup()
{
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(A, INPUT);
  pinMode(B, INPUT);
}

void loop()
{
  if (digitalRead(A) == HIGH){
    digitalWrite(RED, HIGH);
  }
  else{
    digitalWrite(RED, LOW);
  }
  if (digitalRead(B) == LOW){
    digitalWrite(GREEN, LOW);
  }
  else{
    digitalWrite(GREEN, HIGH);
  }
}
```

Internal Input Pullup Resistor
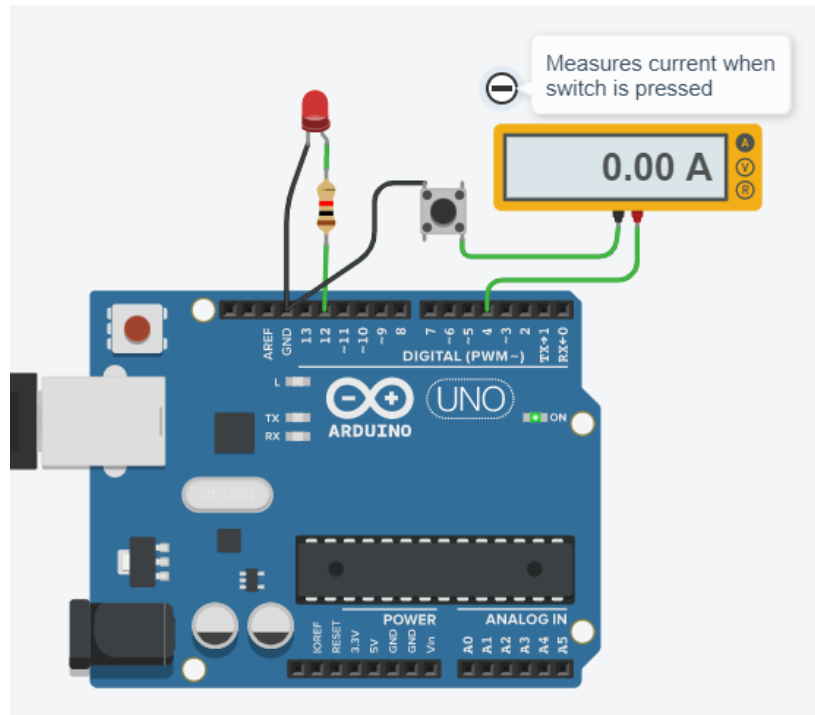
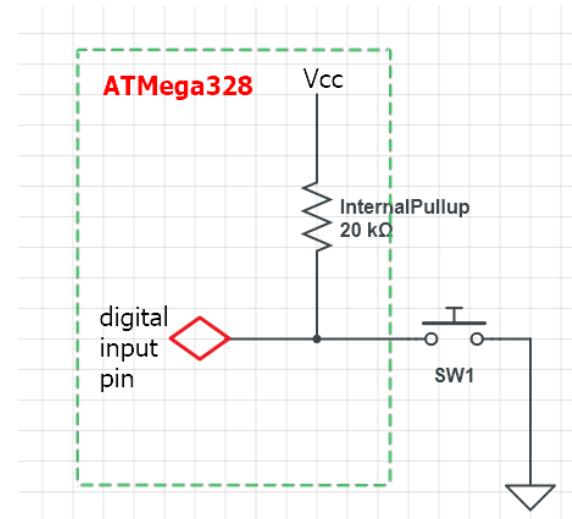Uno input pullup resistor demo

- We can use the internal pullup resistor by changing the mode

- pinMode(pin, INPUT_PULLUP)

- Internal pullup resistor is 20K~50K which limits the current.

Problems with Mechanical Switches
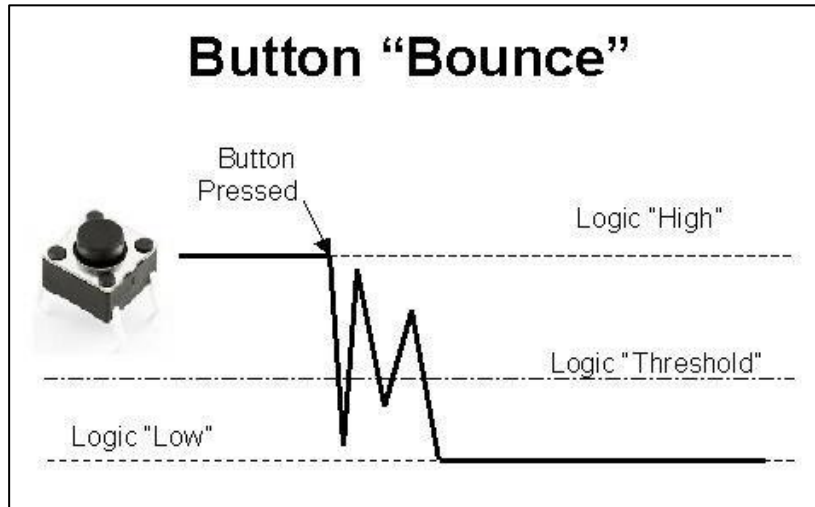
Software Debounce of buttons

- When a mechanical switch is pressed, it creates transients (bouncing) which causes incorrect states to be read.

- Solution:
  - Software debounce
  - Add a delay
  - Use states
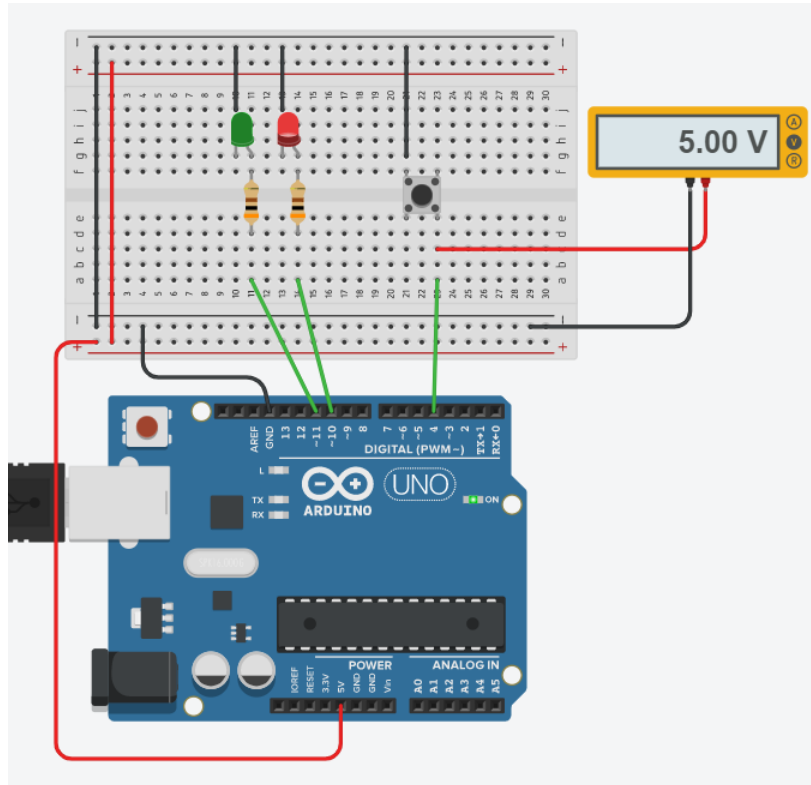
Counting with a Switch

- PBSW uses internal pullup resistor.

- Each time the switch is pressed, the count is incremented.
  The LEDs should show the binary equivalent of the count

- Sequence:
  --, -R, G-, GR
     repeats

- Simulate and examine result, does it work as stipulated?

Code: Counting with a switch

```
1
2    #define RED     10
3    #define GREEN   11
4    #define PBSW     4
5
6    int count = 0;  // inital count value
7
8    void setup()
9    {
10     pinMode(RED, OUTPUT);
11     pinMode(GREEN, OUTPUT);
12     pinMode(PBSW, INPUT_PULLUP);
13     decode(count);
14   }
15
16   void loop()
17   {
18     if (digitalRead(PBSW) == LOW){
19       // switch was pressed
20       count = (count + 1) % 4;  // only 4 states
21       decode(count);  // display it
22     }
23   }
24
```

```
25   // decodes count into binary
26   void decode(int v)
27   {
28     switch(v){
29       case 0:
30           digitalWrite(RED, 0);
31           digitalWrite(GREEN, 0);
32           break;
33       case 1:
34           digitalWrite(RED, 1);
35           digitalWrite(GREEN, 0);
36           break;
37       case 2:
38           digitalWrite(RED, 0);
39           digitalWrite(GREEN, 1);
40           break;
41       case 3:
42           digitalWrite(RED, 1);
43           digitalWrite(GREEN, 1);
44           break;
45     }
46   }
47
```

How do we debug this code?

Arduino Serial Mode

- The Arduino System provides a Serial Mode for displaying text messages.

- Uses the Uno's serial port to transmit data to and from the board to the IDE

- Allows data to be displayed in text as well as in graphical format.

- Uses the Arduino Built-in library: Serial

- When using Serial, you must not use the Tx,Rx pins for any I/O.

[Serial](#) Library

```
2
3   #define LED 8
4   int count = 0;
5
6   void setup()
7 ˅ {
8     // initialise TxRx speed
9     Serial.begin(9600);
10    pinMode(LED, OUTPUT);
11  }
12
13  void loop()
14 ˅ {
15    // send msg to Serial port
16    Serial.print("Count is now ");
17    Serial.println(count);
18    flash(LED);
19    count = count + 1;
20  }
21
22  void flash(int led)
23 ˅ {
24    digitalWrite(led, HIGH);
25    delay(1000);
26    digitalWrite(led, LOW);
27    delay(1000);
28  }
29
```

- Use Serial to display the count value from a sketch. The sketch updates the count value and flashes an LED with a delay of 1 second between flashes.

- [Serial.begin(9600)](#)
  Sets the data rate in bits/sec for serial data transmission

- [Serial.print(data)](#)
  [Serial.println(data)](#)
  Sends data to the serial port for conversion and output. If a string of text is sent, it must be delimited with double quotes (")

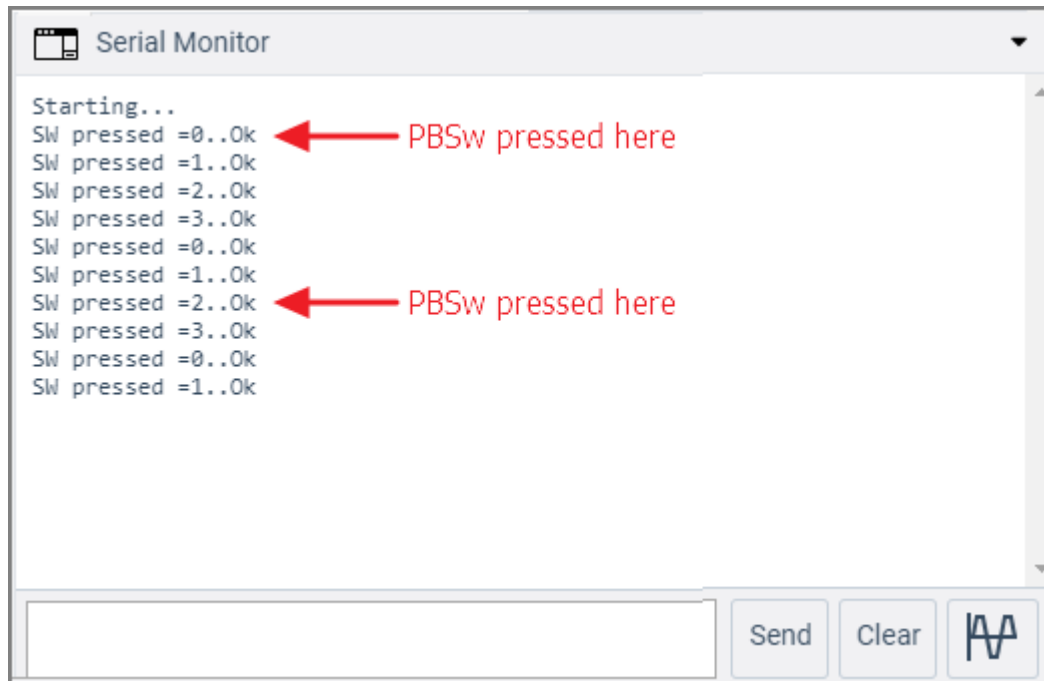- There are other functions, but usually not used in embedded circuits.

Where does the serial output go?

- There is an icon/text "Serial Monitor" on both the TinkerCAD and Arduino IDE interface to show the contents of the serial monitor.

- You can clear, input and output data as well as graphically chart the data you receive from the embedded system.
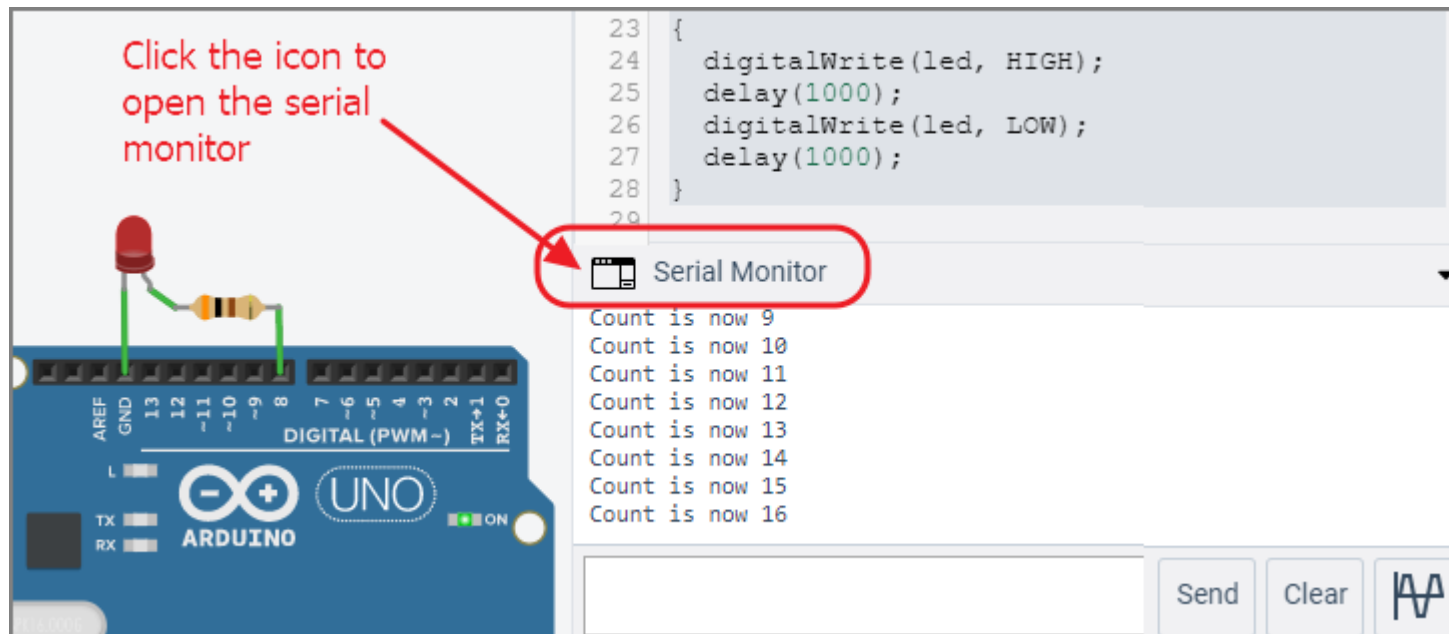
[Using the Serial Monitor](#)

Debug our Counting SW program


```
    Serial Monitor                                    ▼
Starting...
SW pressed =0..Ok      ◄────── PBSw pressed here
SW pressed =1..Ok
SW pressed =2..Ok
SW pressed =3..Ok
SW pressed =0..Ok
SW pressed =1..Ok
SW pressed =2..Ok      ◄────── PBSw pressed here
SW pressed =3..Ok
SW pressed =0..Ok
SW pressed =1..Ok

                                    Send   Clear   ᴧ∨
```

Debugging the SW counting program

```cpp
3   void setup()
4   {
5       ...
6       Serial.begin(9600);
7       Serial.println("Starting...");
8   }
9
10  void loop()
11  {
12      if (digitalRead(PBSW) == LOW){
13          // switch was pressed
14          Serial.print("SW pressed =");
15          Serial.print(count);
16          count = (count + 1) % 4;
17          decode(count);  // display it
18          Serial.println("..Ok");
19      }
20  }
```

- Looks like we have bouncing problem and/or the switch is being read too fast (SW not recovered yet)

Pushbutton SW

1 push count

HIGH

threshold

LOW

STATE → HIGH → LOW → HIGH

Record SW event
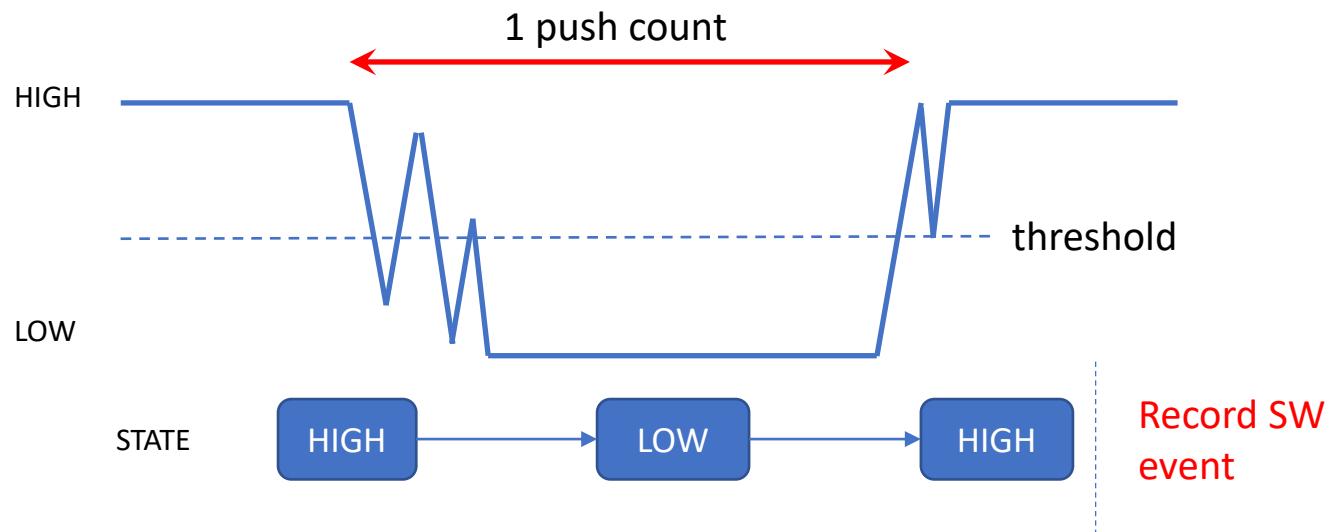
- We keep reading the PBSW taking note of the digital values

- We maintain the state so that we know which part of the sequence we are currently in and when the PBSW returns to normal

- Register the entire sequence as a single push.

Using states

- Use states to track the keypress.

- HIGH = normal
  LOW = in a keypress
  HIGH = returns to normal

- Add short delay when key is pressed to remove bouncing

- Record the keypress only when the sequence is complete.

[Reading a Pushbutton using states](#)

```
17
18  void loop()
19  {
20    if(digitalRead(PBSW) == LOW){
21      // switch pressed
22      if (state == HIGH){
23        // ok, lets process
24        state = LOW;
25        delay(25);
26      }
27      else {
28        // ignore, since state = LOW
29      }
30    }
31    else {
32      // switch is at normal
33      if (state == LOW){
34        // register keypress
35        count = (count + 1) % 4;
36        decode(count);
37        state = HIGH;
38      }
39    }
40  }
41
42
```

Assignment: Programming

- Work out [Assignment 12 Introduction to Arduino Programming](#) using TinkerCAD.

- Simulate your solution using TinkerCAD.
  Use Serial to display messages showing the states and to show that you know how to use the library and the serial monitor.

- Document your work on your site.

EP1000

Digital I/O

End