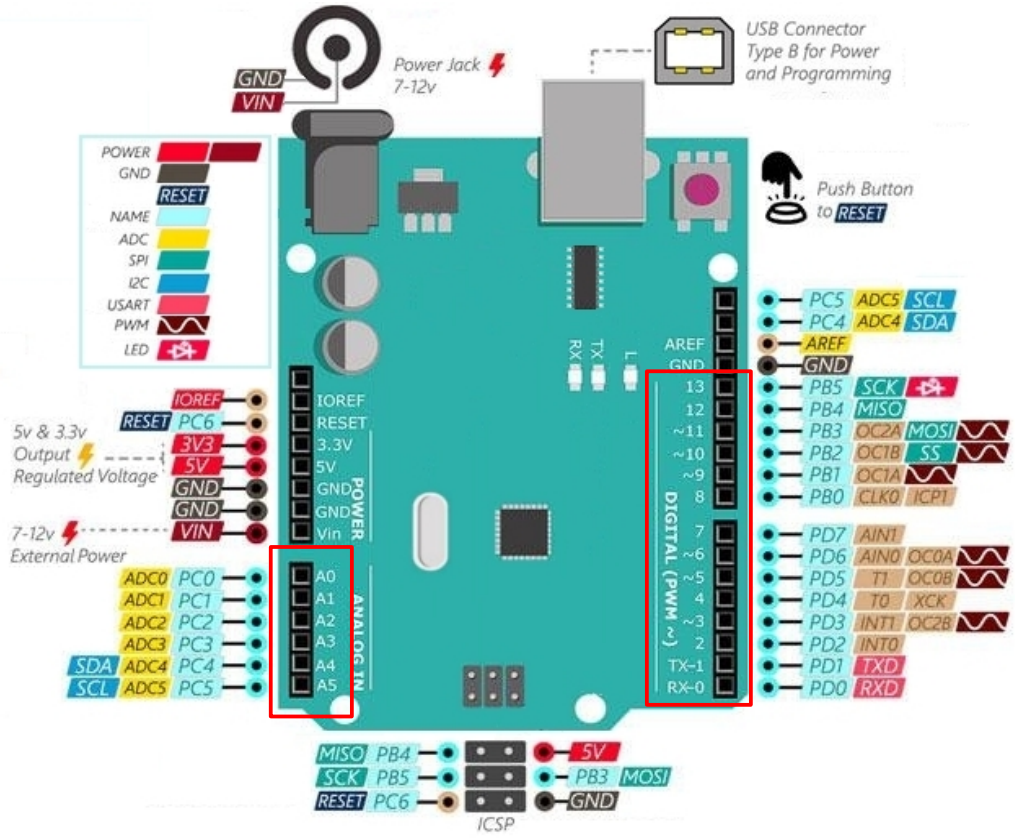
ATMEGA328/Arduino Uno - I/O Pins - INPUT with Tactile Switch
https://github.com/teaksoon/lmaewapm



All the I/O Pins on the Arduino Uno Board with label A0 to A5 and 0 to 13 when set as **INPUT**, can be used to **physically receive Voltage**

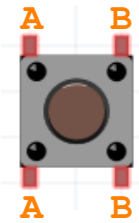
We can only read 0 or 1 from the Digital INPUT I/O Pins (0 to 13) using **digitalRead() function**
0 or LOW is for Input Voltage that is Less than 3V.
1 or HIGH is for Input Voltage that is 3V or More
We cannot use the analogRead() function on these Pins

We can read the real voltage (anything from 0V to 5V) from the Analog INPUT I/O Pins (A0 to A5) using **analogRead() function**. **We can still use the digitalRead() function on the Analog Pins (which will give us 0 or 1, same like the regular Digital Pins)**. There are only 6 Analog Pins on the Arduino Uno board, therefore the Analog Pins are very precious. We normally reserve them for Analog Sensors, where we need the real Voltage from the Analog Sensor where we cant get from the Digital Pins

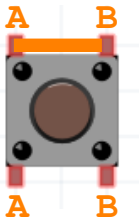
For Tactile Switch, we are only interested in ON and OFF. That is all that a Switch can do for us, OFF=0V=LOW or ON=5V=HIGH. Normally, we will use the Arduino Uno Digital Pins (0 to 13) for Tactile Switch and keep the Analog Pins for Analog Sensors

NOTE:
For educational purpose, we want to learn about digitalRead() and analogRead() and the Voltage in the INPUT I/O Pins. We will use Analog Pin (A0) this time with our Tactile Switch, since Analog Pin can use both digitalRead() and analogRead(). We want to see Voltage, analogRead() and digitalRead() in action, in the same Program.

ATMEGA328/Arduino Uno - I/O Pins - INPUT with Tactile Switch
https://github.com/teaksoon/lmaewapm



Button
is **released**
A/B is **disconnected**



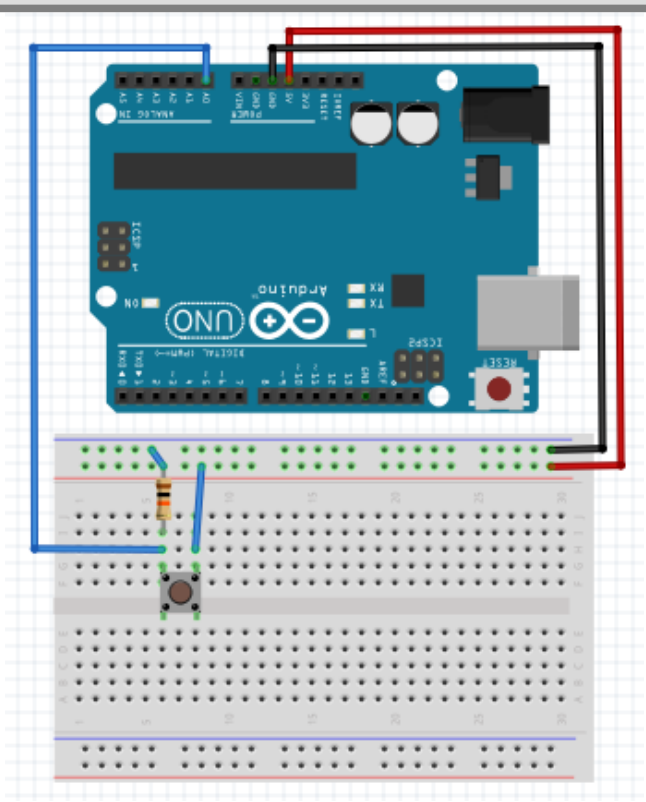
Button
is **held-down**
A/B is **connected**

Tactile Switch is a mechanical switch.

It has a button on the outside and two separate metal plates inside the casing, Side A and Side B with both coming out of the casing.

When the Button is held-down, Side A and Side B will be connected, when Button is released, Side A and Side B will automatically be disconnected

INPUT I/O Pin with Tactile Switch



1x Computer with Arduino IDE Software
1x USB 2.0 Type A/B Data Cable
1x Arduino Uno Board

1x Solderless Breadboard
Jumper wires
1x Tactile Switch
1x 10 KOhm Resistor

Tactile B Side, to Arduino 5V
Tactile A Side, to Arduino A0
Tactile A Side, to Resistor to GND

When **button is released**, A0 will only have path to Resistor and GND since the Tactile A/B sides are not connected. A0 will get 0V

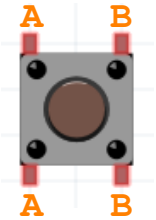
When **button is held-down**, A0 will have a path directly to 5V Pin through the connected Tactile A/B Sides. A0 will get 5V

Program: io_input_tactile_raw_reading

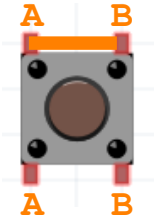
```
void setup() {
  pinMode(A0, INPUT);
  Serial.begin(9600);
}

void loop() {
  int pinDigital = digitalRead(A0);
  int pinAnalog  = analogRead(A0);

  Serial.print("\nA0: ");
  Serial.print(" Digital=");Serial.print(pinDigital);
  Serial.print(", Analog=");Serial.print(pinAnalog);
  Serial.print(", Voltage=");Serial.print( (float) (pinAnalog*5)/1023,2 );
}
```

While watching the Serial Monitor screen, hold-down the Tactile Switch Button and then release the Tactile Switch button

ATMEGA328/Arduino Uno – I/O Pins – INPUT with Tactile Switch
https://github.com/teaksoon/lmaewapm



Button
is **released**
A/B is **disconnected**



Button
is **held-down**
A/B is **connected**

Mechanical switch do have a behaviour that will cause problem for us in some applications. During the process of holding down the Button or releasing the Button, there will be some electrical “jitters” that we did not want, called “DEBOUNCE”. Jitters may happen for a very short period of time in a few miliseconds.

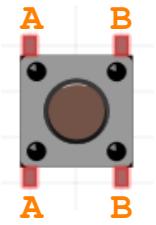
If we are making a simple ON/OFF switch, we do not need to care about debounce BUT for some other cases like: if each Button click is supposed to something different, “DEBOUNCE” will become a serious issue (a single Click will become a multiple Clicks and our Program will be doing wrong thing). We need to get rid of this “DEBOUNCE”, we can either do it via extra hardware or we simply use some creative Prgramming to deal with this issue.

Program: io_input_tactile_debounce

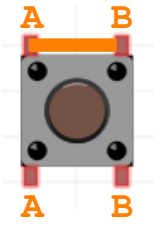
```
#define T1_PIN   A0

unsigned long debounce_delay = 40;
unsigned long debounce_timer;

int T1_data;
int prev_T1_data;

int hitCounter = 0;

void setup() {
  pinMode(T1_PIN, INPUT);
  Serial.begin(9600);
  prev_T1_data = digitalRead(A0); // Initial value
}

void loop() {
int new_reading;

  // start - Tactile Switch Processing -
  new_reading = digitalRead(T1_PIN);
  if (new_reading != prev_T1_data) {
    debounce_timer = millis();
  }
  if ((millis()-debounce_timer) > debounce_delay) {
    if (new_reading != T1_data) {
      T1_data = new_reading;
      if (T1_data == HIGH) {
        Serial.print("\nButton Pressed: "); Serial.print(++hitCounter);
      }
    }
  }
  prev_T1_data = new_reading;
  // end - Tactile Switch Processing -

}
```

While watching the Serial Monitor screen, hold-down the Tactile Switch Button and then release the Tactile Switch button

ATMEGA328/Arduino Uno – I/O Pins – INPUT with Tactile Switch
https://github.com/teaksoon/lmaewapm



Button
is **released**
A/B is **disconnected**



Button
is **held-down**
A/B is **connected**

Below is code for very simple Tactile Switch use, without dealing with "DEBOUNCE". Whether the "DEBOUNCE" will become an issue, depends on what our Tactile Switch is used for. There is nothing wrong with this code, this code simply dont care about "DEBOUNCE"

Program: io_input_tactile_no_debounce

```
#define T1_PIN   A0
int T1_data;
int hitCounter = 0;

void setup() {
  pinMode(T1_PIN, INPUT);
  Serial.begin(9600);
}

void loop() {
int new_reading;

  // start - Tactile Switch Processing -
  new_reading = digitalRead(T1_PIN);
  if (new_reading != T1_data) {
    T1_data = new_reading;
    if (T1_data == 1) {
      Serial.print("\nButton Pressed: "); Serial.print(++hitCounter);
      // delay(40); // put this in for a "crude" debounce filter
    }
  }
  // end - Tactile Switch Processing -
}
```

While watching the Serial Monitor screen, hold-down the Tactile Switch Button and then release the Tactile Switch button.

This code is very much simpler compared to the one dealing with "DEBOUNCE". From the Serial Monitor, we can see that there are "Button Pressed" message that is not physically pressed by us.

We can still have a very "crude" debounce filter by adding in a delay() function after the INPUT I/O Pin reading (40ms should be enough, you can try different delay time). Try that and watch the Serial Monitor Screen while pressing and releasing the Button, we now have a very "crude" debounce filter

"DEBOUNCE" only happen during the Pressing and Releasing of Button

When a Button has already been Held Down or when a Button has already been Released (for a few milli-seconds), the Voltage will not jitter anymore, it will stay stable and no more "DEBOUNCE"