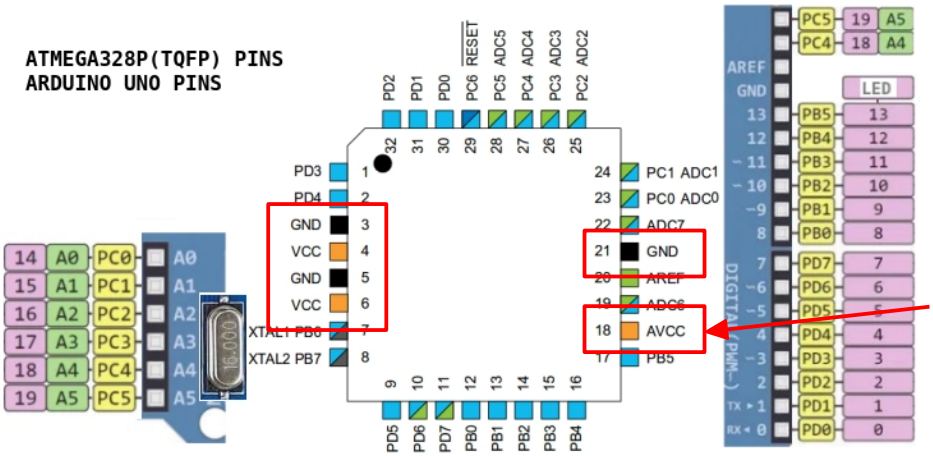
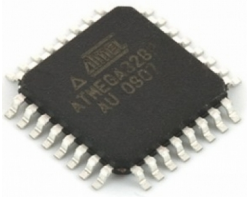
ARDUINO BASICS - INPUT / OUTPUT PINS
https://github.com/teaksoon/stemkraf

by TeakSoon Ding (OCT-2021)
for STEMKRAF

Power Supply for Arduino Uno and Atmega328 micro-controller





Note: AVCC is required to be connected to 5V, because the Analog Pins uses seperate power supply.

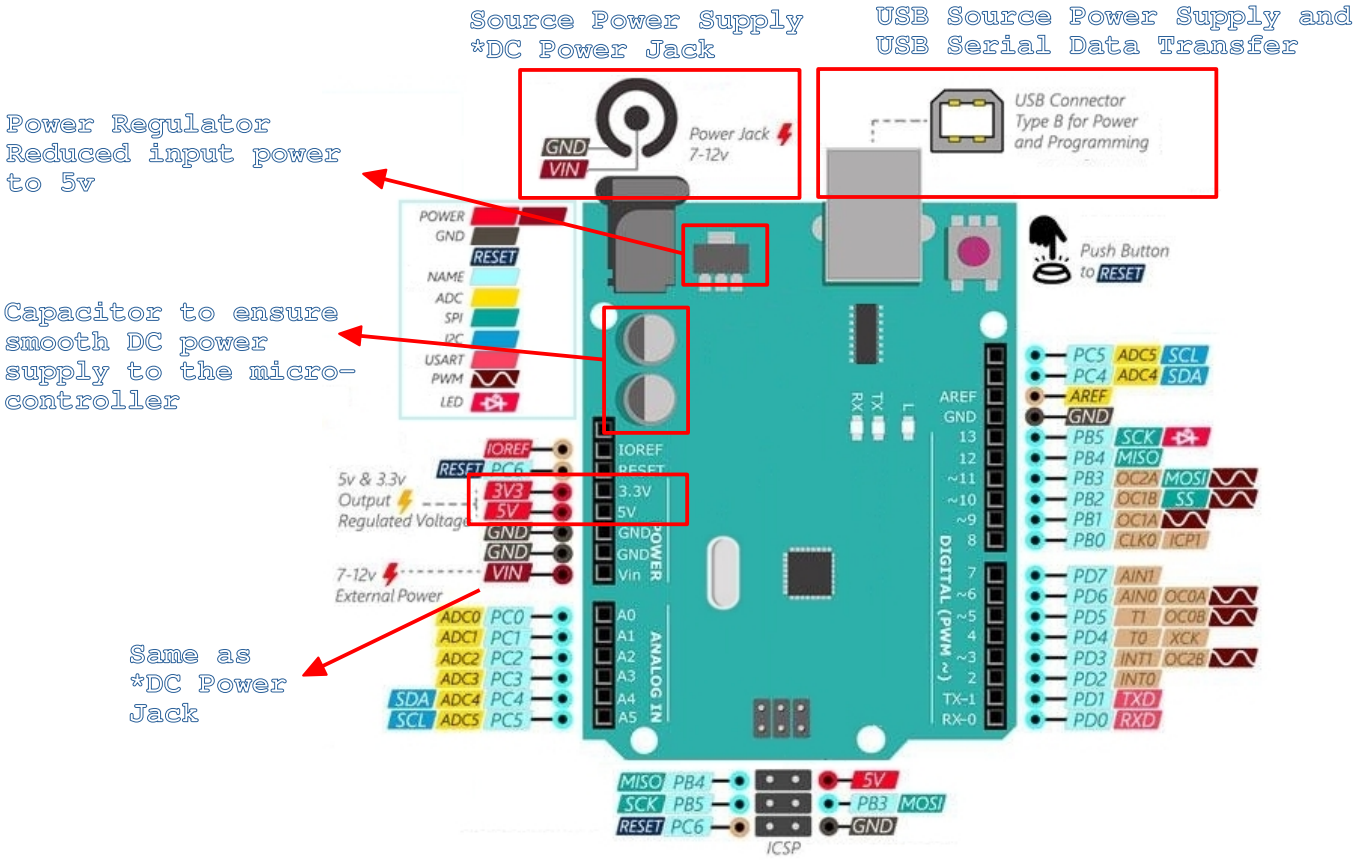
In order for the Atmega328 micro-controller to work, it needs to receive electricity.
DC Power +ve Terminal to be connected to VCC
DC Power -ve Termnial to be connected to GND

Note: AVCC is also connected to +ve DC Terminal, this is for internal Analog Device power supply (reduce electrical noise for analog device)

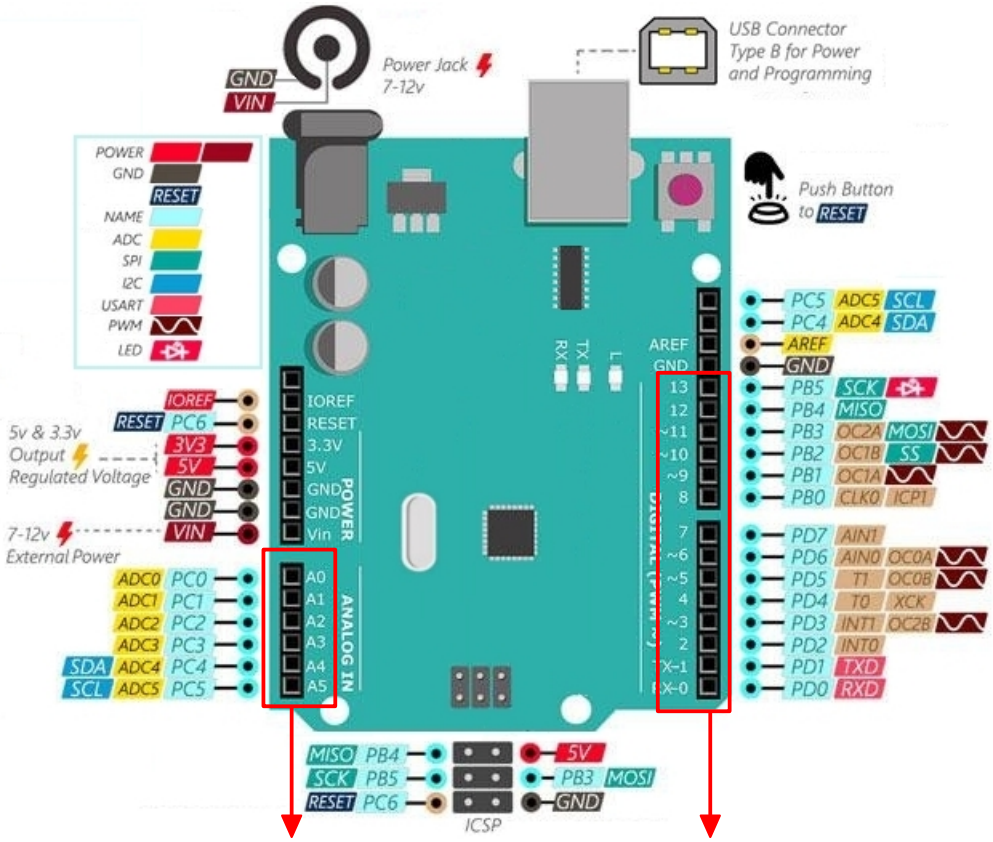
We must make sure power does not exceed the limit, otherwise the chip will be burnt. We must also make sure the power is sufficient, otherwise the micro-controller will not function.

However, when using the Arduino Uno, it is very much safer and easier. The Arduino Uno board have a power regulator to limit the current going into the micro-controller and also Capacitors to ensure stable power input.

NOTE: The Pins labelled 5V and 3.3v are normally used as as output power supply where we draw 5V or 3.3v DC power supply from them



Arduino Uno and Input / Output Pins

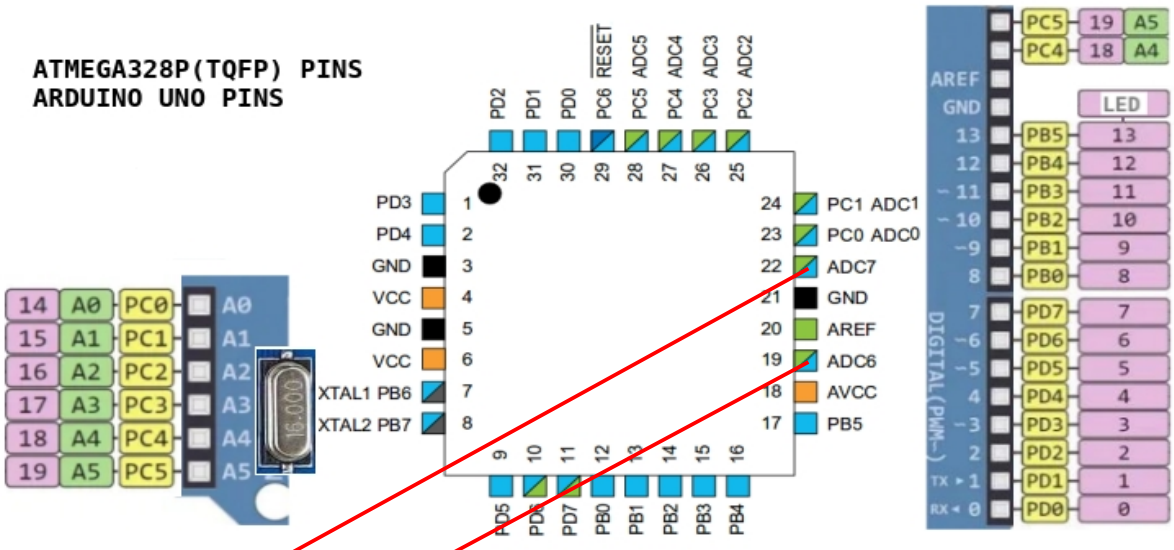


Pin 0 to 13 and Pin A0 to A5 are also dealing with Voltage. Unlike the VIN, 5V, 3.3V Pin fixed behaviour, the behaviour Pin 0 to 13 and Pin A0 to A5 can be programmed.

- They can Receive Power from External Source (set as INPUT PIN)

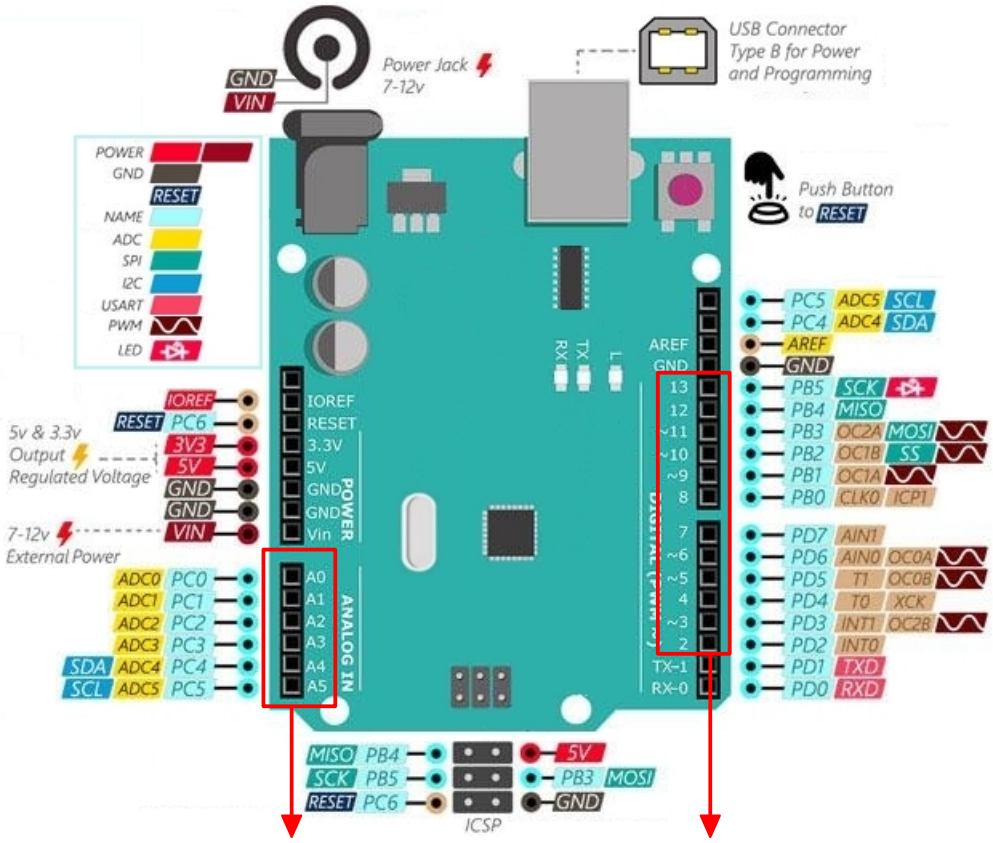
- They can Supply Power to External Device (set as OUTPUT PIN)

Bulk of the micro-controller programming is all about changing and reading the Voltage on the programable pins



There are another 2 Analog Pin from the atmega328 that is not used in Arduino Uno board, ADC7 and ADC6 (they are available on the Arduino Nano board). If you must use them from the Arduino Uno board, you can physically solder a connection to those pin

Arduino Uno and Input / Output Pins



INPUT PIN (receive power from connected device)

We can make pin to become an OUTPUT PIN by coding `pinMode(pin, INPUT);` often not required as this is the default.

When the pin is set as INPUT PIN, we can use 2 Arduino functions to read the Voltage that is currently received by this Pin

- digitalRead() function
- analogRead() function

- digitalRead() function

When we code `digitalRead(pin);`
- if the Voltage on the pin is 5V or very near to 5V, the function will give us the value of HIGH or 1

- if the Voltage on the pin is 0V or very near to 0V, the function will give us the value of LOW or 0

- analogRead() function

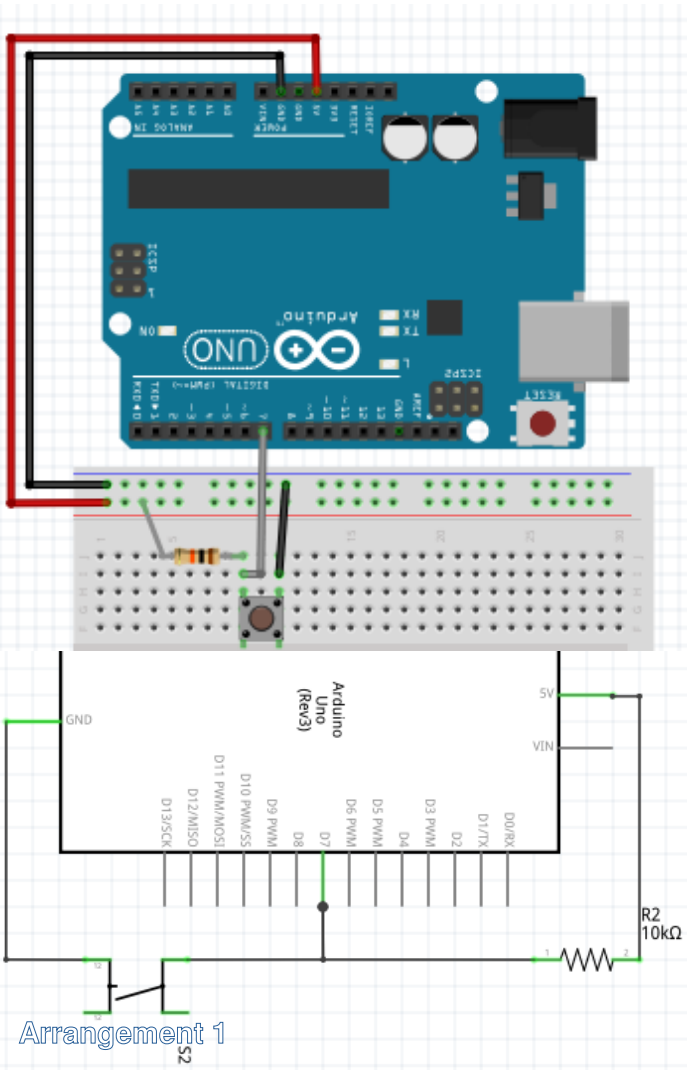
This function only works with the Analog Pins (A0 to A5)

When we code `analogRead(pin);`
- We will get the actual Voltage on the pin, represented by a 10-bit numbering system (0 to 1023), meaning

if the voltage on the pin is 5V, we will get 1023
if the voltage on the pin is 0V, we will get 0

if the voltage is nV, we will get (n/5) x 1023

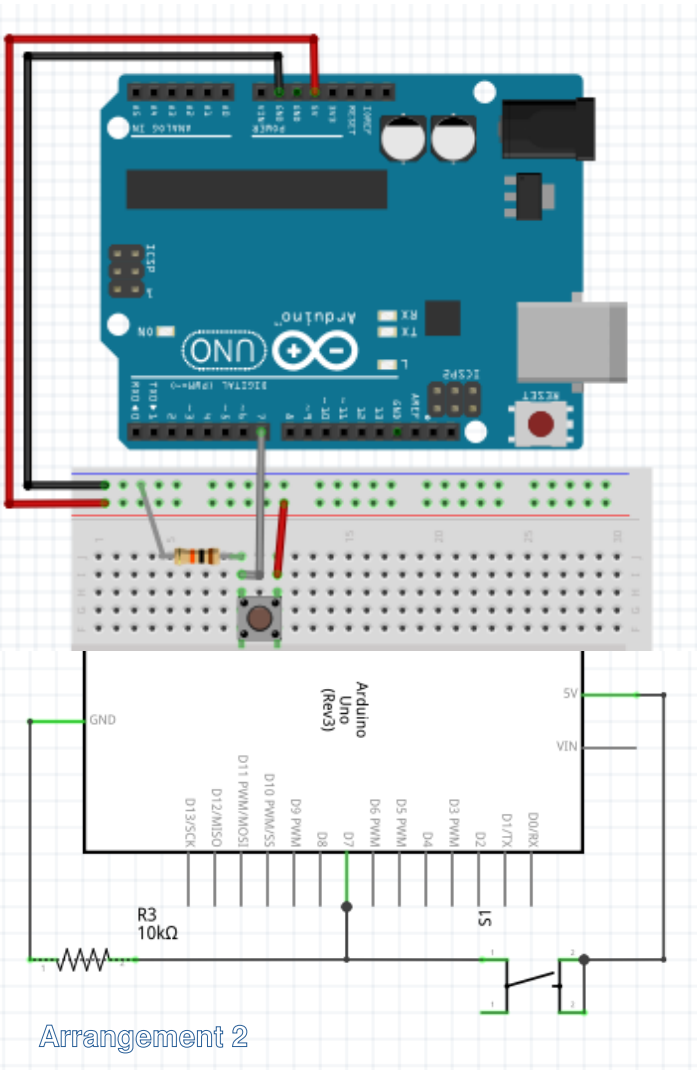
Arduino Uno and Input / Output Pins



Arrangement 1

Button is NOT PRESSED
– 5V going to Pin 7
digitalRead(7) will give is HIGH or 1

Button is PRESSED
– 0V going to Pin 7
digitalRead(7) will give us LOW or 0



Arrangement 2

Button is NOT PRESSED
– 0V going to Pin 7
digitalRead(7) will give us LOW or 0

Button is PRESSED
– 5V going to Pin 7
digitalRead(7) will give us HIGH or 1

We are pysicallly connecting and disconnecting 5V from the Arduino Uno board to the digital Pin 7.

The digitalRead() function will tell us whether digital Pin 7 has HIGH (5V) or LOW (0V). This reading will tell us whether the Button is pressed or released. digitalRead() can work for all the Pins (Pin 0 to 13, A0 to A5)

The analogRead() function however, can only read from the Analog Pins (A0 to A5). Usage is similar to digitalRead() where it will read a Pin for Voltage. Instead of HIGH and LOW, analogRead() gives us a range of values (0 to 1023)

Why do we need a Resistor ?

Without a resistor, we will get alot of "electrical noise" on the pin, we may get wierd results when reading Voltage from the Pin. Any resistor above 1Kohm should be enough to filter the noise for the tactile switch

In this example, we are only interested to find out whether the button is pressed or released

This is not all that it can do. This is the foundation for all the other advanced digital technologies.

Different external device will alter the voltage on the Pin differently. We read the voltage on the pins and determine what the device is telling us. Radio Signals device for example, will set the digital Pin in a series HIGH and LOW, which when decoded... represents a message

```
// program name = input_digitalread_7
//
void setup() {
  pinMode(7,INPUT); // optional
  Serial.begin(9600);
}

void loop(){
  Serial.println(digitalRead(7));
}
```

Connect Tactile Button to Pin 7, Open the Serial Monitor from Arduino IDE Software. Press and Release the button and see what is shown on the Serial Monitor.

```
// program name = input_digitalread_A0
//
void setup() {
  pinMode(A0,INPUT); // optional
  Serial.begin(9600);
}

void loop(){
  Serial.println(digitalRead(A0));
}
```

Connect Tactile Button to Pin A0, Open the Serial Monitor from Arduino IDE Software. Press and Release the button and see what is shown on the Serial Monitor.

```
// program name = input_analogread_A0
//
void setup() {
  pinMode(A0,INPUT); // optional
  Serial.begin(9600);
}

void loop(){
  Serial.println(analogRead(A0));
}
```

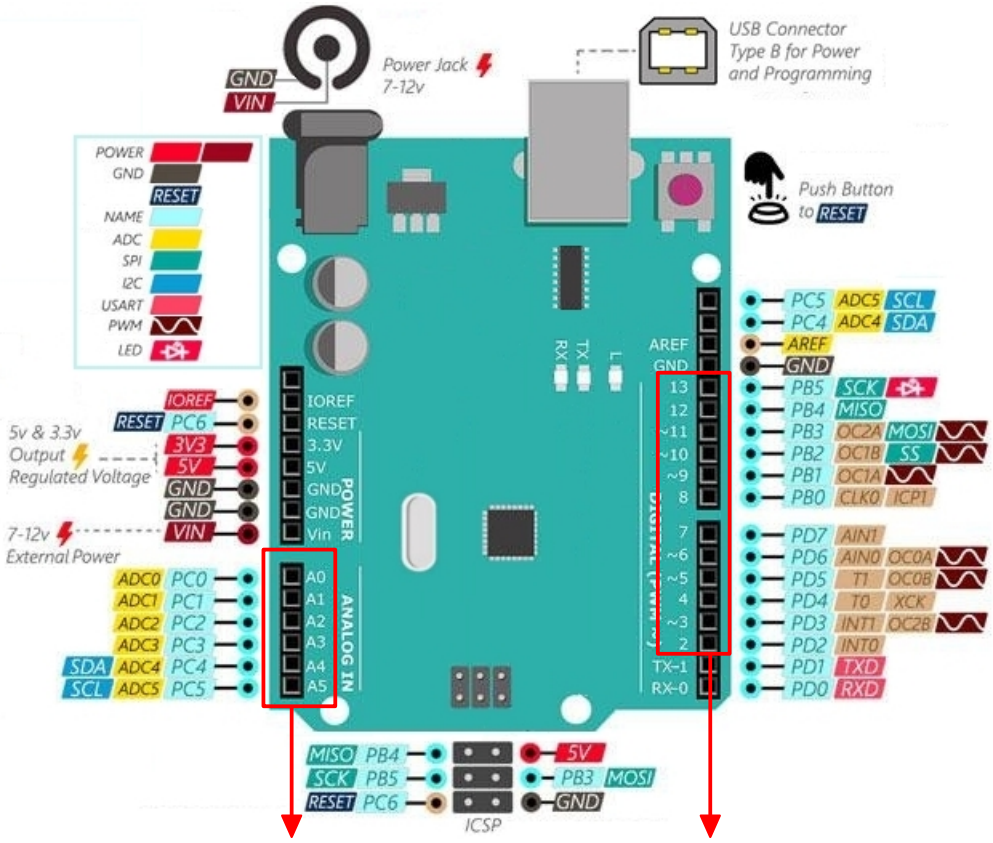
Connect Tactile Button to Pin A0, Open the Serial Monitor from Arduino IDE Software. Press and Release the button and see what is shown on the Serial Monitor.

```
// program name = input_analogread_7 ( will not work )
//
void setup() {
  pinMode(7,INPUT); // optional
  Serial.begin(9600);
}

void loop(){
  Serial.println(analogRead(7)); // THIS WILL NOT WORK,
  // You may get some random numbers. analogRead only works with Analog Pins
}
```

Connect Tactile Button to Pin 7, Open the Serial Monitor from Arduino IDE Software. Press and Release the button and see what is shown on the Serial Monitor.

Arduino Uno and Input / Output Pins



OUTPUT PIN (supply power to connected device)

We can make pin to become an OUTPUT PIN by coding `pinMode(pin, OUTPUT);`

When the pin set as OUTPUT PIN, we can use 2 Arduino functions to control how much Voltage this pin will supply to the connected device.

- digitalWrite() function
- analogWrite() function

- digitalWrite() function

1. When we code digitalWrite(pin, HIGH), pin will supply 5V to whatever device connected to it.

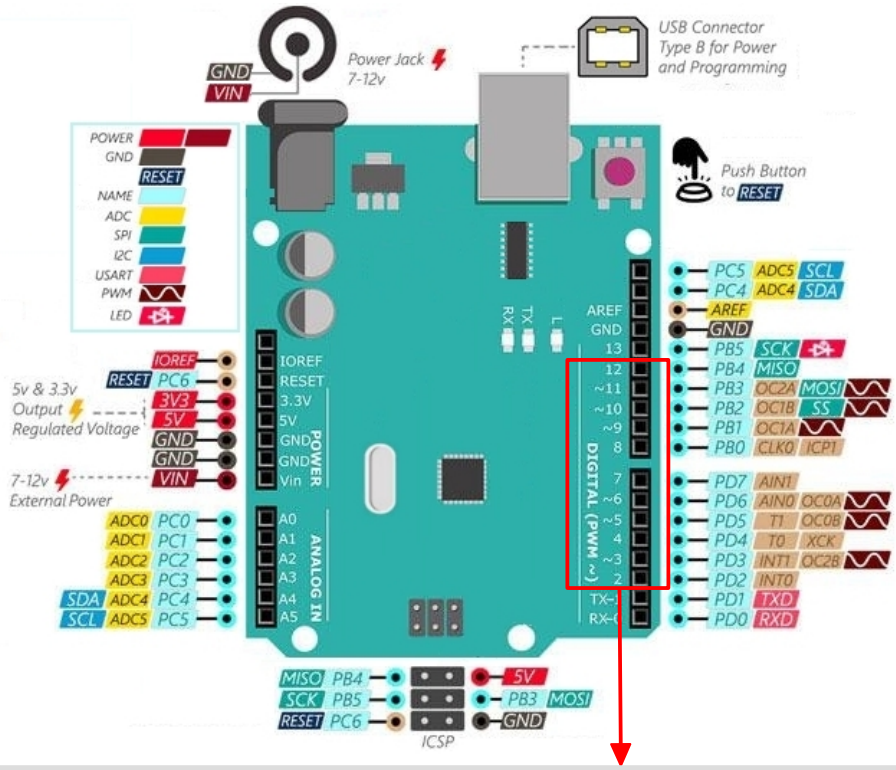
2. When we code digitalWrite(pin, LOW), pin will supply 0V to whatever device connected to it.

- analogWrite() function

This function only works with the PWM Digital pins, on Arduino board they are labeled with ~ symbol, pin 3,5,6,9,10 and 11.

What analogWrite(pin,value) does is, based on the specified value, it runs a series of digitalWrite(pin,HIGH) and digitalWrite(pin,LOW) in various combinatons to create a PWM voltage effect on the connected Device (more on PWM-Pulse Wave Modulation in the next slide)

NOTE:
analogWrite() function has nothing to do with the physical Arduino Analog Pins, A0 to A5. This function should be called "pwmWrite" instead.



PWM – Pulse Wave Modulation

PWM is the process of alternating between HIGH and LOW. It is actually similar to a series of "digitalWrite()".

On Arduino Uno, PWM pins are labelled with ~ symbol, pin 3,5,6,9,10 and 11.

Digital PWM Pins only works when set as OUTPUT mode. pinMode(pin, OUTPUT);

analogWrite(pin,value);
value = 0 to 255 (Binary 00000000 to 11111111 since it is stored in an 8-bit memory). Although the name is "analogWrite", it has nothing to do with Analog Pins... it is actually for digital PWM Pins.

1. analogWrite(pin,255);
We make the pin HIGH, continue making it HIGH, it is called 100% duty cycle
Meaning, the pin provide 5V,5V,5V,... to the device connected to this pin.
The connected device will get a "constant 5V" power supply

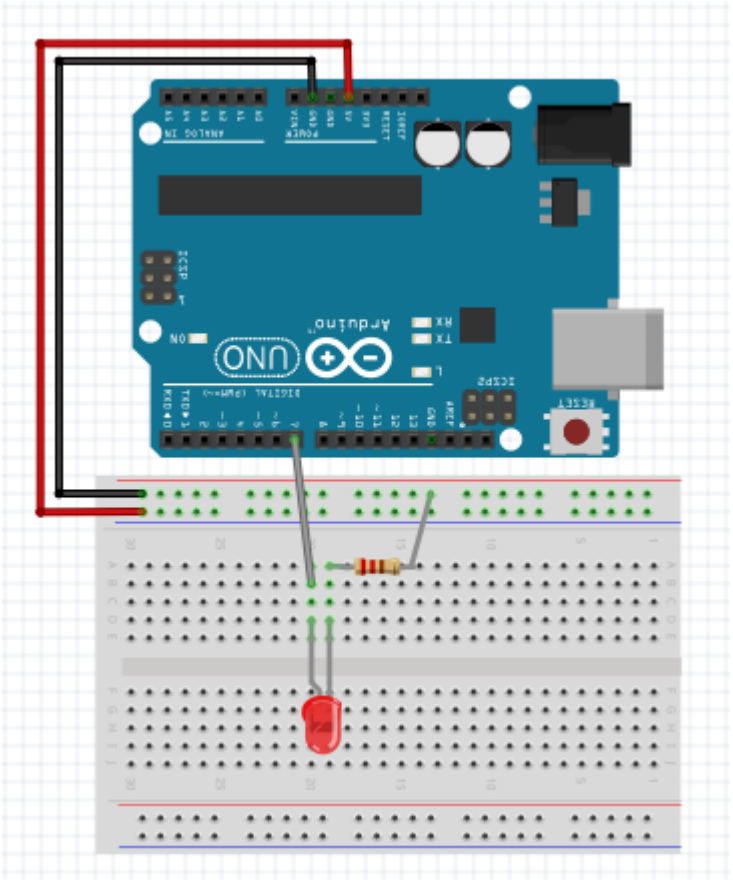
2. analogWrite(pin,0);
We make the pin LOW, continue making it LOW, it is called 0% duty cycle
Meaning, the pin provide 0V,0V,0V,... to the device connected to this pin.
The connected device will get a "constant 0V" power supply

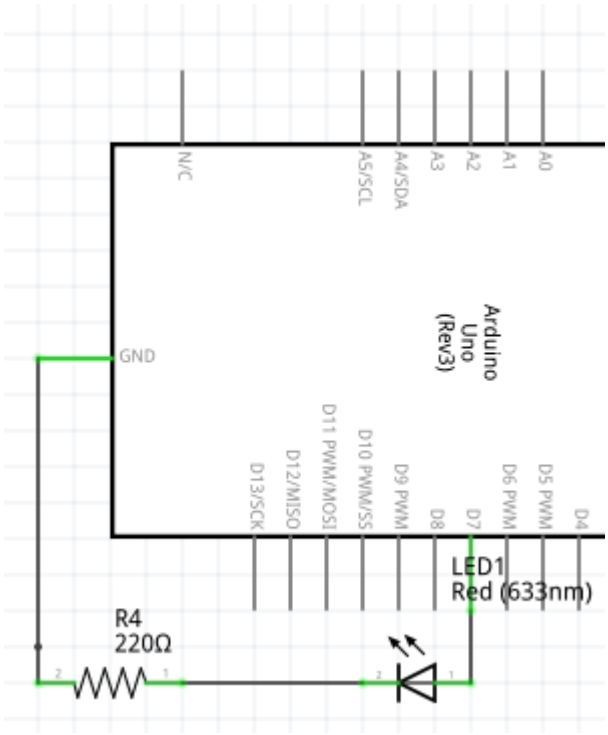
3. analogWrite(pin,127); // this is between 0 and 255
We issue "HIGH, LOW", and repeats that combination, it is called 50% duty cycle. Meaning, the pin provide 5V,0V,5V,0V,... to the device connected to this pin. This is not a constant Voltage, it repeats sets of "5V,0V". The connected device will react accordingly

4. analogWrite(pin,191); // this is between 127 and 255
We issue "HIGH, HIGH, LOW", and repeats that combination, it is called 75% duty cycle. Meaning, this pin provide 5V,5V,0V,5V,5V,0V,... to the device connected to this pin. This is not a constant Voltage, it repeats the set of "5V,5V,0V". The connected device will react accordingly

5. and so on... The "value" used in analogWrite() can be changed to achieve different PWM effect voltage for the connected device.

Arduino Uno and Input / Output Pins





After setting the Pin 7 to OUTPUT Pin, by default the OUTPUT Pin 7 has 0V.

LED will only light up when there is a minimum amount of Voltage (usually minimum 1.3V, varies from one LED to another). At this point of time, Pin 7 has 0V, so the connected LED will not light up

When we code digitalWrite(7,HIGH);
The Micro-controller will make 5V flow to Pin 7, providing 5V to the LED and that will cause LED to turn ON

When we code digitalWrite(7,LOW);
The micro-controller will make 0V flow to Pin 7, providing 0V to the LED and that will cause LED to turn OFF

We can use digitalWrite() function on the Analog Pins (A0 to A5) the same way like we use on the digital Pins (0 to 13)

Why do we need a Resistor ?

A direct 5V from Arduino Uno Board will destroy the LED, that is why we have the 220 ohm resistor there to prevent the LED from being destroyed.

In this example, we are just switching the LED, ON and OFF.

This is not all that it can do. This is the foundation for all the other advanced digital technologies.

Different external device will behave differently when different voltage combinations is applied to them. Some device will require us to send multiple digitalWrite() with a specific sequence to perform a task. For example, a Radio Signals will consist of a series of ON and OFF in various combinations, representing messages.

```
// program name = output_digitalwrite_7
//
void setup() {
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH);
}

void loop(){ }
```

Connect LED to Pin 7 and watch. The LED connected to Pin 7 will receive 5V

```
// program name = output_digitalwrite_A0
//
void setup() {
  pinMode(A0,OUTPUT);
  digitalWrite(A0,HIGH);
}

void loop(){ }
```

Connect LED to Pin A0 and watch. The LED connected to Pin A0 will receive 5V

```
// program name = output_analogwrite_9
//
void setup() {
  pinMode(9,OUTPUT);
  analogWrite(9,127);
}

void loop(){ }
```

Connect LED to PWM Pin 9 and watch. The LED will receive alternating 5V/0V, this process go so fast that you will not notice the "blink" going on with the LED. You will just see the LED switched ON.

```
// program name = output_analog_7 ( will not work )
//
void setup() {
  pinMode(7,OUTPUT);
  analogWrite(7,127); // THIS WILL NOT WORK.
  // analogWrite will only work on PWM Pins
}

void loop(){ }
```

Connect LED to Pin 7 and watch. The LED connected to Pin 7 will not receive PWM alternating 5V/0V because Pin 7 is not a PWM pin, analogWrite() function will not work correctly on this Pin