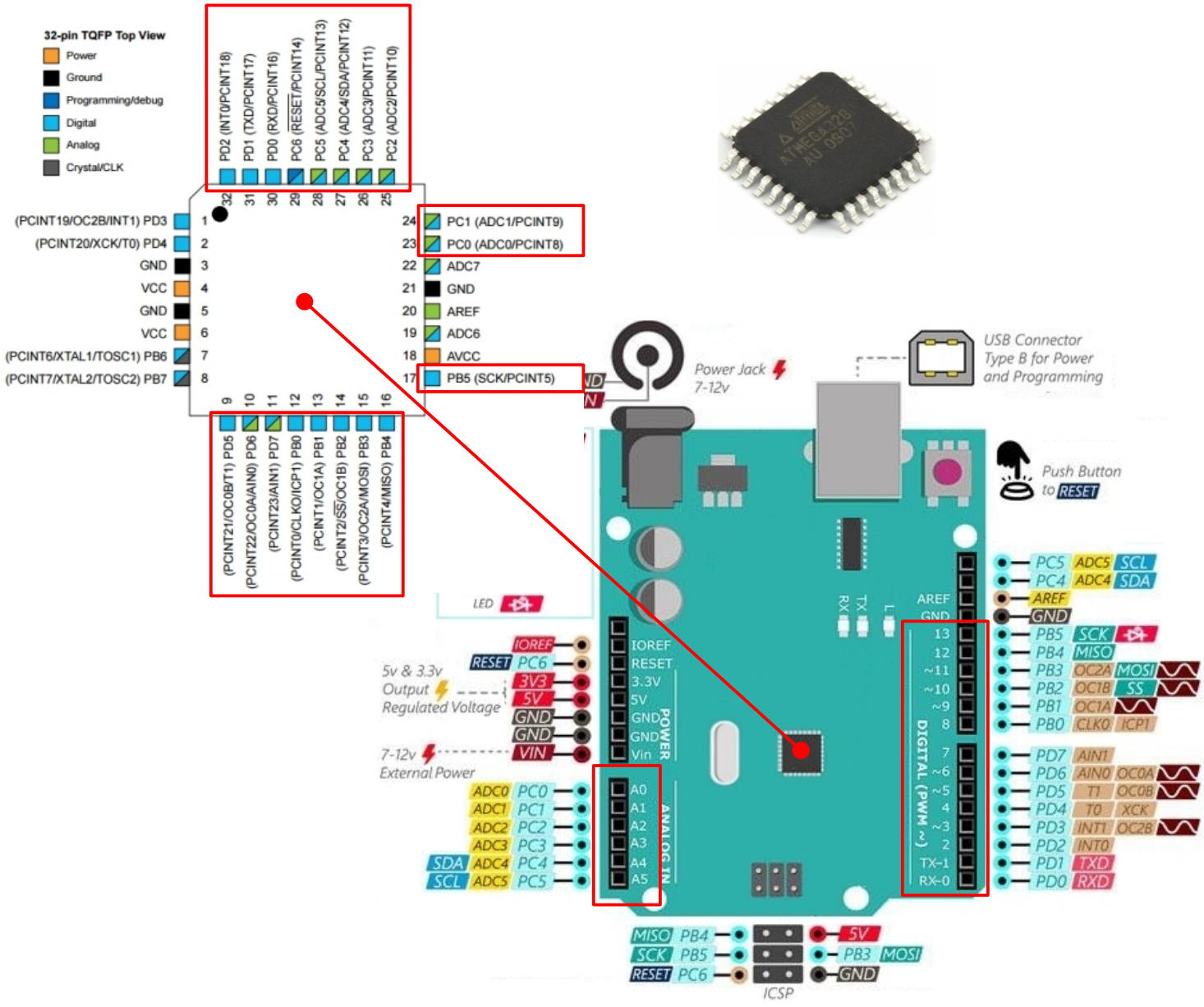
ATMEGA328 – I/O pins – OUTPUT
https://github.com/teaksoon/lmaewapm

Apart from the **Power Supply Pins (GND, VCC, AVcc)**, the ATMEGA328P micro-controller have many other pins coming out from its physical chip packaging. Those other Pins are known as **I/O Pins** connected to **Arduino Uno Board with label A0 to A5 and 0 to 13**



When I/O Pins are set to become OUTPUT Pin (from our Program), they become very similar to the Arduino Uno "5V Pin"

- Both Arduino "5v Pin" and I/O Pins are +ve Terminal of a Circuit and have +ve 5V in them

Difference between I/O Pin and the VCC are,

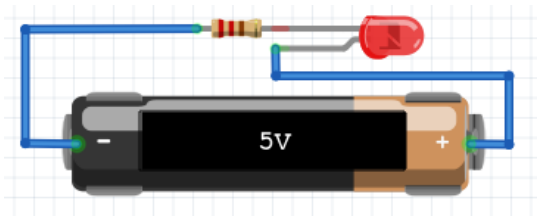
- **Arduino "5V Pin" fixed to have +ve 5V** in it. While the **+ve 5V in the OUTPUT I/O Pin can be "turned OFF(0V) or turned ON(+ve 5V)"**

- We can control OUTPUT I/O Pin to have 0V or 5V from our **Program**

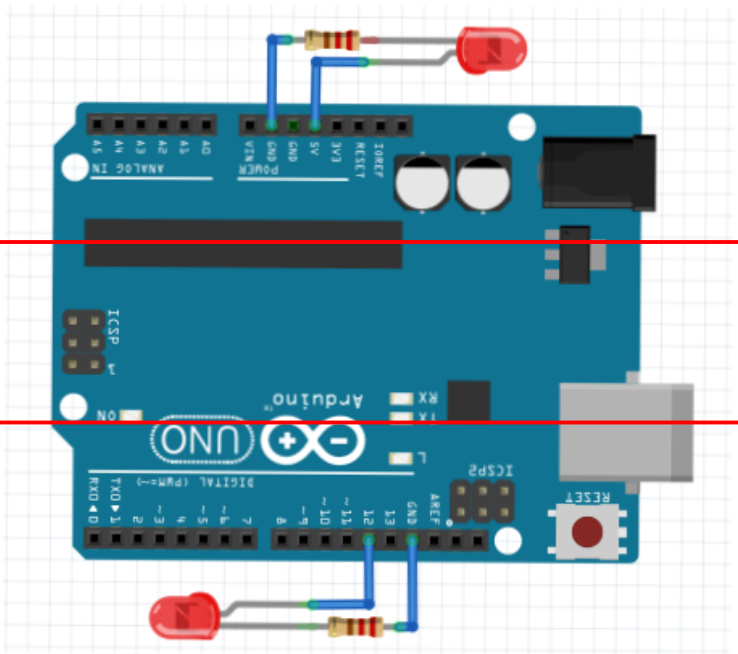
Programming "OUTPUT I/O pin" is all about changing the I/O pin from our Program to have either 0V or 5V

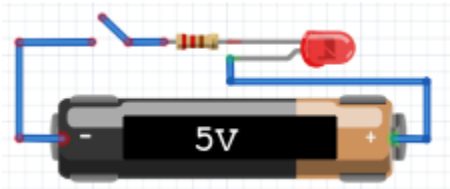
These two setup are the same (vs Arduino Uno 5V Pin)

This is a LED connected to
a Power Source
+ve Terminal and -ve Terminal



This is a LED connected to
5V Pin(+ve) and GND Pin(-ve)





These two setup are ALMOST the same (vs Arduino Uno I/O Pins)

This is a LED connected to
a Power Source
+ve Terminal and -ve Terminal
- With an external **"Manual switch"**

This is a LED connected to
I/O Pin 12(+ve) and GND Pin (-ve)
(can be other Pin A0-A5, 0-13)
- With an internal **"Programable Switch"**

SAFETY NOTE:

1. The reason for the 220ohm Resistor to be used together with the LED in this 5V circuit is to prevent the LED from damage. In our basic electricty lesson, besides the Voltage, there is another eletrical elements involved, "Current". This 5mm bulb LED can only handle about 20mA current, a 5V circuit like this will surely have more than 20mA current, If the Resistor (220ohm or more) is not there, the LED will surely be damaged.

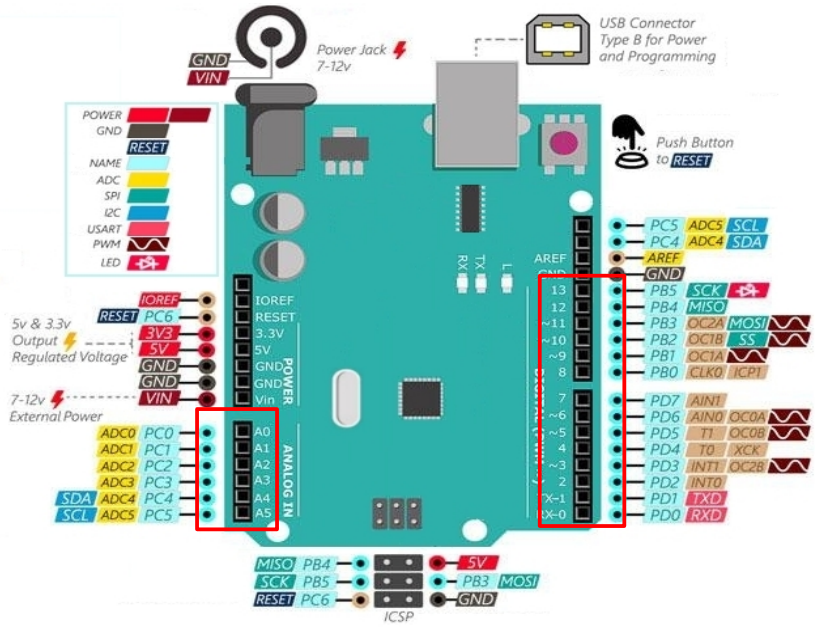
We will learn about Resistor in another lesson, you will know why we use 220 Ohm or Resistor with some other resistance value and the current involved.

2. Although this circuit can supply 5V, do not attach anything we wish to the Arduino Uno board, especially the DC Motors.

Motor generally uses more "Current" than what the IO Pins or the 5V Pin on the Arduino Uno can cope. Apart from that, DC Motors also generates electricty, we do not want it to flow back into the Arduino Uno and potentially damage the board.

We can use Motor with Arduino Uno board but not by connecting directly to the I/O Pin like the LED (we will learn about this in another lesson).

ATMEGA328 - I/O pins - OUTPUT
https://github.com/teaksoon/lmaewapm



On the Arduino Uno Board(Pin A0 to Pin A5 and Pin 0 to Pin 13), all of them can be used as OUTPUT I/O Pin. How do we Program them ?

Programming OUTPUT I/O Pin is very simply. We only have 2 things to do,
1.Tell the CPU to set the I/O Pin to OUTPUT Pin
2.Tell the CPU to supply 5V or 0V to the OUTPUT Pin

1.Tell the CPU to set the I/O Pin to OUTPUT Pin

To make the I/O Pin an OUTPUT Pin,
we simply use the following Program Code,

```
pinMode(12, 1); // this will set Pin 12 to be an OUTPUT Pin
pinMode(A0, 1); // this will set Pin A0 to be an OUTPUT Pin
and etc... ( can be any other I/O Pins )
```

To make our Program code easier to read, we can also code

```
pinMode(12, OUTPUT); // this will set Pin 12 to be an OUTPUT Pin
pinMode(A0, OUTPUT); // this will set Pin A0 to be an OUTPUT Pin
and etc... ( can be any other I/O Pins )
```

2.Tell the CPU to supply 5V or 0V to the OUTPUT Pin

Once any Pin has already been set as OUTPUT Pin, we can ask the CPU to either supply 0V or 5V to the OUTPUT I/O Pin,

by using the following Program Code,

```
digitalWrite(12,1); // This will make Pin 12 to have 5V
digitalWrite(12,0); // This will make Pin 12 to have 0V
and etc... ( can be any other I/O Pins )
```

To make our Program code easier to read, we can also code

```
digitalWrite(12,HIGH); // This will make Pin 12 to have 5V
digitalWrite(12,LOW); // This will make Pin 12 to have 0V
and etc... ( can be any other I/O Pins )
```

Program Structure in "C-Language with Arduino Library" Style

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

Start the Arduino IDE Software and enter the following codes

```
void setup() { }
void loop() { }
```

We always start with "skeleton code". The above is the bare mininum code for Program written in "C-Language with Arduino Library" style

The Program above have two "functions". A function name **"setup"** and another function name **"loop". These two function must exist** in any Program written in "C-Language with Arduino Library" Style

Each function will contain zero or more Program Codes in them. At the moment both are have zero Program Codes in them.

The above is already a working Program, you can upload it into to your ATMEGA328 micro-controller on your Arduino Uno board.

The Program Codes inside any C-Language functions will run from Top to Left to Bottom, each C-Language Program Code is separated by a semi-colon ;

The first line of Program Code inside the setup() function is the starting point for any Program written in the "C-Language with Arduino Library" style.

After completion of setup() function,

the Program Codes inside the loop() function will start, also from Top to Left to Bottom.

However, when loop() function completes, it will run loop() function from the beginning again (meaning, the Program Codes inside the loop() function will be repeated forever)

Please read the "Arduino Uno QuickStart" tutorial if you have forgotten how to create a Program and send the Program into the micro-controller

Basic Program to manipulate the OUTPUT I/O Pin

We will write a Program to set an I/O Pin to OUTPUT Pin, then we supply 5V to it. After 3 seconds, we supply 0V to it.

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

Start the Arduino IDE Software and enter the following codes...
We always start with the bare minimum "skeleton code", then we expand from there.

```
void setup() { }
void loop() { }
```

1.Lets set an I/O Pin to OUTPUT Pin. The code below, tells the micro-controller CPU to set Pin 13 as OUTPUT Pin.

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() { }
```

2.Now the Pin 13 has become an OUTPUT Pin, Lets continue to supply Voltage to it

```
void setup() {
  pinMode(13, OUTPUT);

  digitalWrite(13,HIGH);
  delay(3000);
  digitalWrite(13,LOW);
}

void loop() { }
```

We are done with our Program now

Upload this program into the ATMEGA328 micro-controller on the Arduino Uno board and watch out for one LED on the Arduino Uno board being TURN ON for 3 seconds then, automatically TURNED OFF

If you missed it, press the "Reset" button on the Arduino Uno board and observe the Arduino Uno board. That LED is connected to Pin 13

If you have LED+Resistor, instead of Pin 13, you can try other Pins

Reset Button

Although this looks simple, this is the foundation for more advanced I/O OUTPUT activities (we will learn those in the future lessons, you will find out they will all relate to this simple ON/OFF activities)

ATMEGA328 – I/O pins – OUTPUT
https://github.com/teaksoon/lmaewapm

This Page is for those already familiar with beginners topic and nothing to learn from this tutorial, I hope this page can be useful to some of you

Each digitalWrite(13,HIGH) or digitalWrite(13,LOW) requires about 50 clock cycle to complete. For 16Mhz micro-controller, one clock cycle is 62.5 nano-seconds. Means, one digitalWrite() will take about 3125 nano-seconds to complete. Which is still too fast for our eyes, that is why we have delay(3000); 3000 miliseconds delay. However, sometimes we need faster speed from our digitalWrite(). To do that, we can use the "C-Language with AVR Library" style to get better speed.

We can mix the "Arduino Library" and the "AVR Library" Codes together

```
#include "avr/io.h"
void setup() {
  pinMode(13, OUTPUT);

  PORTB |=  (1 << PB5);
  delay(3000);
  PORTB &= ~(1 << PB5);
}
void loop(){ }
```

Replace digitalWrite(13,HIGH) with PORTB |= (1 << PB5);
Replace digitalWrite(13,LOW) with PORTB &= ~(1 << PB5);

The two lines of Codes above is from the "C-Language with AVR Library". When a C Program is compiled, the C-Compiler Software will covert the C-Language Codes to INSTRUCTION SET Codes,

PORTB |= (1 << PB5); // is compiled to a single INSTRUCTION SET: sbi 0x05, 5, according to ATMEGA328 datasheet, SBI requires 2 clock cycle
PORTB &= ~(1 << PB5); // is compiled to a single INSTRUCTION SET: cbi 0x05, 5, according to ATMEGA328 datasheet, CBI requires 2 clock cycle

So, it is 125 nano-seconds (2 cycles) vs 3125 nano-seconds (50 cycles)

What is digitalWrite() and why does it take so many clock-cycle to complete?

Below is the source code for digitalWrite(). As you can see, it has alot of Codes to run (when compiled, it will be even more in INSTRUCTION SET codes). Which not only slows down but also increases Program size

```
void digitalWrite(uint8_t pin, uint8_t val) {
uint8_t timer = digitalPinToTimer(pin);
uint8_t bit   = digitalPinToBitMask(pin);
uint8_t port  = digitalPinToPort(pin);
volatile uint8_t *out;
if (port == NOT_A_PIN) return;
  if (timer != NOT_ON_TIMER) turnOffPWM(timer);
  out = portOutputRegister(port);
  uint8_t oldSREG = SREG;
  cli();
  if (val == LOW) {
    *out &= ~bit;
  } else {
    *out |= bit;
  }
  SREG = oldSREG;
}
```

While "PORTB |= (1 << PB5);" or "PORTB &= ~(1 << PB5);" is just one line of code, calling a single INSTRUCTION SET directly.