I/O FUNCTIONS

The pins on the Arduino board can be configured as either inputs or outputs. A majority of Arduino analog pins, may be configured, and used, in exactly the same manner as digital pins.

Pins Configured as INPUT

- Arduino pins are by default configured as inputs, so they do not need to be explicit declared as inputs with pinMode().
- Pins configured this way are said to be in a high-impedance state.
- Input pins make extremely small demands on the circuit that they are sampling.
- It takes very little current to switch the input pin from one state to another.
- Useful pins fo such tasks as implementing a capacitive touch sensor or reading a LED as photodiode.
- Pins configured as pinMode(pin, INPUT) with nothing connected to them.
- With wires connected to them that are not connected to other circuits.
- Report seemingly random changes pin state.
- Picking-up electrical noise from environment.
- Capacitively coupling the state of nearby pin.

Pull-UP Resistors

Pull - up resistors are often useful to steer an input pin to a known state if no input is present. This can be done by adding a pull-up resistor (to +5V), or a pull-down resistor (resistor to ground) on the input. A 10K resistor is a good value for a pull-up or pull-down resistor.

Using Built-in Pull-up Resistor with Pins configured as Input

There are 20,000 pull-up resistors built into the Atmega chip that can be accessed from software. These built-in pull-up resistors are accessed by setting the **pinMode() as INPUT_PULLUP.**
This effectively inverts the behaviour of the INPUT mode, where HIGH means the sensor is OFF and LOW means the sensor is ON. The value of this pull-up depends on the microcontroller used. On most AVR-bassed boards, the value is guaranteed between 20kΩ and 50kΩ.

When connecting a sensor to a pin configured with INPUT_PULLUP, the other end should be connected to the ground. This causes the pin to read HIGH when the switch is open and LOW when th switch is passed.
The pull-up resistors provide enough current to light an LED dimly connected to a pin configured as an input.

Same registers (internal chip memory locations) that control whether a pin is HIGH or LOW control the pull-up resistors. A pin that is configured to have pull-up resistors turned on when the pin is in INPUTmode, will have the pin configured as HIGH if the pin is then switched OUTPUT mode with pinMode(). This works in the other direction as well.

EXAMPLE

```
pinMode( 3, INPUT ) ; // set pin to input without using built in pull up resistor
pinMode( 5, INPUT_PULLUP ) ; // set pin to input without using built in pull up
resistor
```

Pins Configured as OUTPUT

- Pins configured as OUTPUT with pinMode() are said to be in a low-impedance state.
- They can provide a substantial amount of current to other circuits.
- Atmega pins can source* or sink* up to 40 mA of current to other devices/ circuits.
- Is enough to brightly light up a LED, or run many sensors.
- It's not enough current to run relays, solenoids, or motors.
- High current devices that runs from the output pins, can damage or destroy the output transistors in the pin.

pinMode() Function

The pinMode() function is used to configure a specific pin to behave either as an input or an output. It is possible to enable the internal pull-up resistors with the mode INPUT_PULLUP.

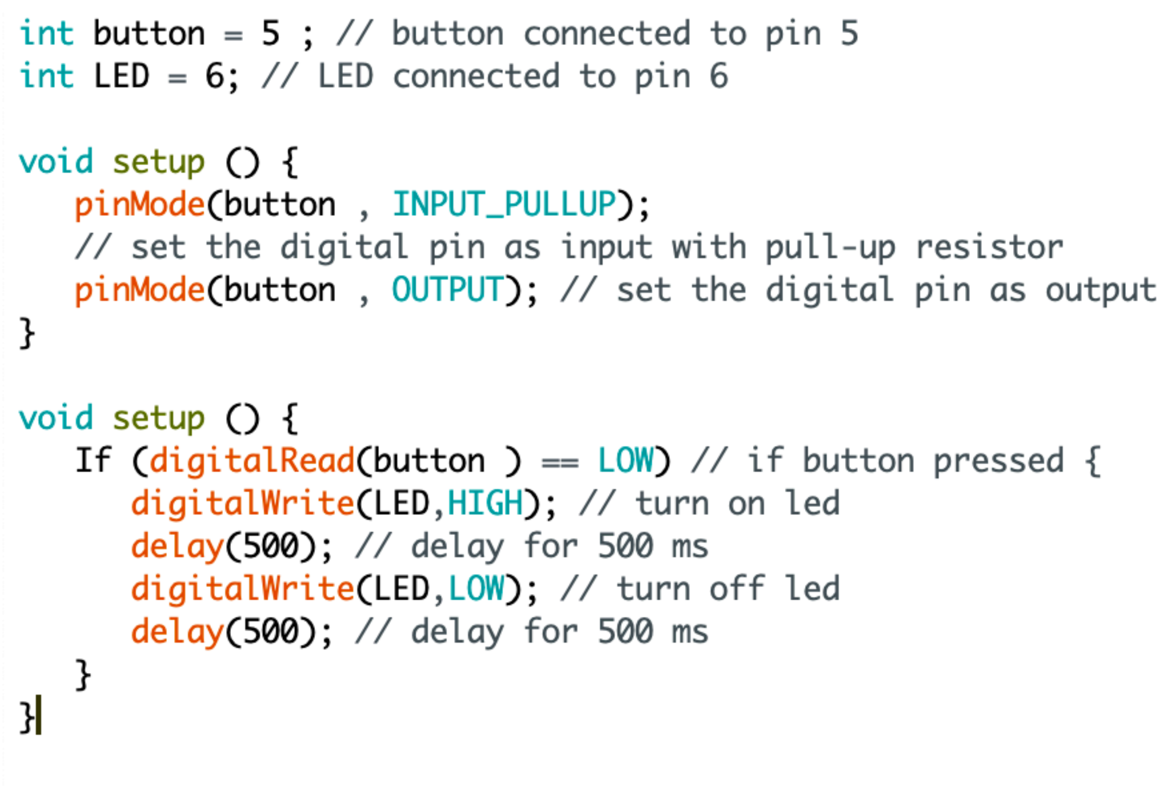
SYNTAX

```
void setup( ) {
        pinMode( pin, mode );
}
```

— **pin**: the number of the pin whose mode you wish to set
— **mode**: INPUT, OUTPUT, or INPUT_PULLUP.

*source: provide positive current | *snik: provide negative current

EXAMPLE



```
int button = 5 ; // button connected to pin 5
int LED = 6; // LED connected to pin 6

void setup () {
   pinMode(button , INPUT_PULLUP);
   // set the digital pin as input with pull-up resistor
   pinMode(button , OUTPUT); // set the digital pin as output
}

void setup () {
   If (digitalRead(button ) == LOW) // if button pressed {
      digitalWrite(LED,HIGH); // turn on led
      delay(500); // delay for 500 ms
      digitalWrite(LED,LOW); // turn off led
      delay(500); // delay for 500 ms
   }
}
```

digitalWrite() Function

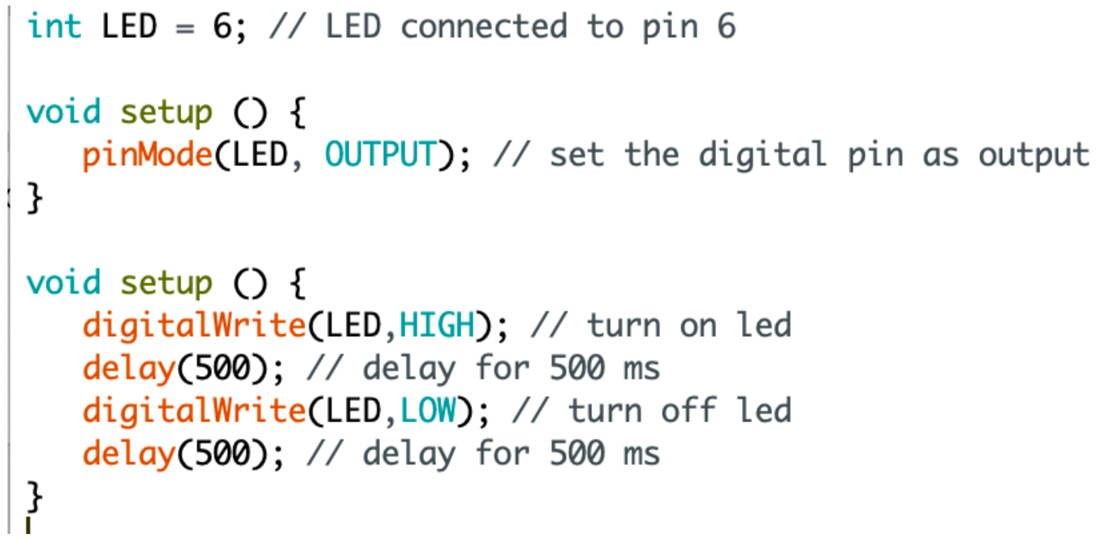
The digitalWrite() function is used to write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V for HIGH, 0V for LOW. If the pin id configured as an INPUT, digitalWrite() will enable or disable the internal pulp on the input pin. It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor.

SYNTAX

```
void loop( ) {
        digitalWrite( pin, value );
}
```

— **pin**: the number the pin whose mode you wish to set
— **value**: HIGH, or LOW

EXAMPLE



```
int LED = 6; // LED connected to pin 6

void setup () {
   pinMode(LED, OUTPUT); // set the digital pin as output
}

void setup () {
   digitalWrite(LED,HIGH); // turn on led
   delay(500); // delay for 500 ms
   digitalWrite(LED,LOW); // turn off led
   delay(500); // delay for 500 ms
}
```

analogRead() Function

Arduino is able to detect whether there is a voltage applied to one of its pins and report through the digitalRead() function. There is difference between on/off sensor and an analog sensor, whose value continuously changes. To read this type of sensor, we need a different type of pin.

In the lower-right part of the Arduino board, you will see six pins marked “Analog in”. These special pins not only wether there is voltage applied to them, but also its value. By using the analogRead() function we can read the voltage applied to one of the pins.

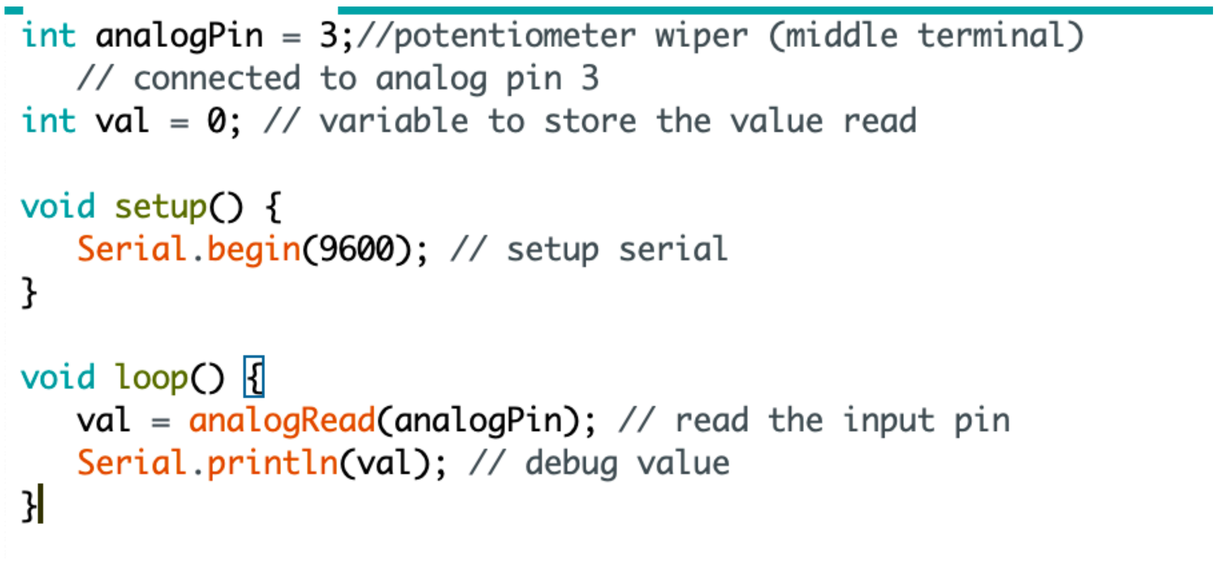
This function returns a number between 0 and 1023, which represents voltages between 0 and 5 volts.

SYNTAX

analogRead(pin);

— **pin**: the number of the analog input pin to read

EXAMPLE



```
int analogPin = 3;//potentiometer wiper (middle terminal)
   // connected to analog pin 3
int val = 0; // variable to store the value read

void setup() {
   Serial.begin(9600); // setup serial
}

void loop() {
   val = analogRead(analogPin); // read the input pin
   Serial.println(val); // debug value
}
```