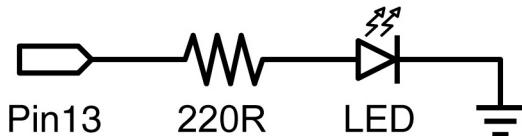


appendix

digital output



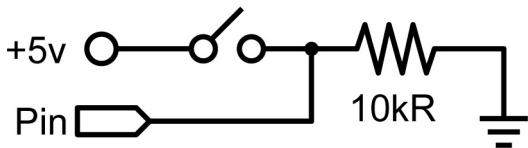
This is the basic 'hello world' program used to simply turn something on or off. In this example, an LED is connected to pin13, and is blinked every second. The resistor may be omitted on this pin since the Arduino has one built in.

```
int ledPin = 13;                      // LED on digital pin 13

void setup()                          // run once
{
    pinMode(ledPin, OUTPUT);        // sets pin 13 as output
}

void loop()                           // run over and over again
{
    digitalWrite(ledPin, HIGH);     // turns the LED on
    delay(1000);                  // pauses for 1 second
    digitalWrite(ledPin, LOW);      // turns the LED off
    delay(1000);                  // pauses for 1 second
}
```

digital input



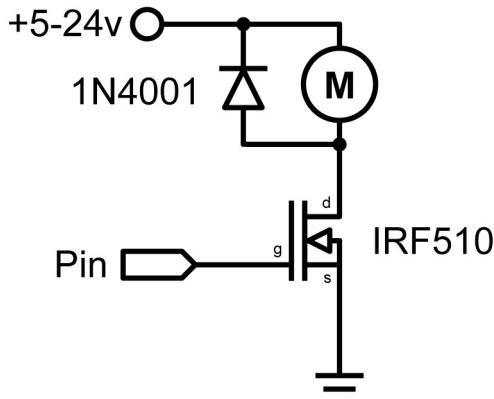
This is the simplest form of input with only two possible states: on or off. This example reads a simple switch or pushbutton connected to pin2. When the switch is closed the input pin will read HIGH and turn on an LED.

```
int ledPin = 13;           // output pin for the LED
int inPin = 2;             // input pin (for a switch)

void setup()
{
    pinMode(ledPin, OUTPUT); // declare LED as output
    pinMode(inPin, INPUT);  // declare switch as input
}

void loop()
{
    if (digitalRead(inPin) == HIGH) // check if input is HIGH
    {
        digitalWrite(ledPin, HIGH); // turns the LED on
        delay(1000);             // pause for 1 second
        digitalWrite(ledPin, LOW); // turns the LED off
        delay(1000);             // pause for 1 second
    }
}
```

high current output



Sometimes it is necessary to control more than 40ma from the Arduino. In this case a MOSFET or transistor could be used to switch higher current loads. The following example quickly turns on and off the MOSFET 5 times every second.

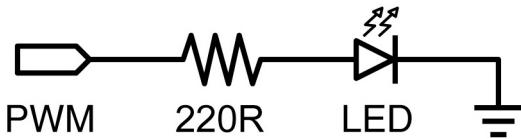
Note: The schematic shows a motor and protection diode but other non-inductive loads could be used without the diode.

```
int outPin = 5;           // output pin for the MOSFET

void setup()
{
    pinMode(outPin, OUTPUT);      // sets pin5 as output
}

void loop()
{
    for (int i=0; i<=5; i++)    // loops 5 times
    {
        digitalWrite(outPin, HIGH); // turns MOSFET on
        delay(250);             // pauses 1/4 second
        digitalWrite(outPin, LOW); // turns MOSFET off
        delay(250);             // pauses 1/4 second
    }
    delay(1000);               // pauses 1 second
}
```

pwm output



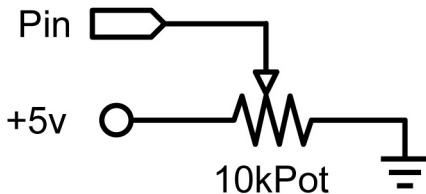
Pulsewidth Modulation (PWM) is a way to fake an analog output by pulsing the output. This could be used to dim and brighten an LED or later to control a servo motor. The following example slowly brightens and dims an LED using for loops.

```
int ledPin = 9;      // PWM pin for the LED

void setup(){}        // no setup needed

void loop()
{
    for (int i=0; i<=255; i++) // ascending value for i
    {
        analogWrite(ledPin, i); // sets brightness level to i
        delay(100);           // pauses for 100ms
    }
    for (int i=255; i>=0; i--) // descending value for i
    {
        analogWrite(ledPin, i); // sets brightness level to i
        delay(100);           // pauses for 100ms
    }
}
```

potentiometer input



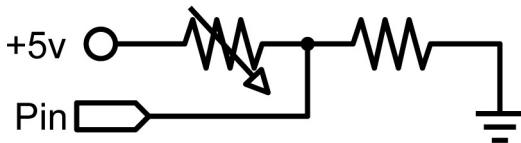
Using a potentiometer and one of the Arduino's analog-to-digital conversion (ADC) pins it is possible to read analog values from 0-1024. The following example uses a potentiometer to control an LED's rate of blinking.

```
int potPin = 0;      // input pin for the potentiometer
int ledPin = 13;     // output pin for the LED

void setup()
{
    pinMode(ledPin, OUTPUT); // declare ledPin as OUTPUT
}

void loop()
{
    digitalWrite(ledPin, HIGH); // turns ledPin on
    delay(analogRead(potPin)); // pause program
    digitalWrite(ledPin, LOW); // turns ledPin off
    delay(analogRead(potPin)); // pause program
}
```

variable resistor input



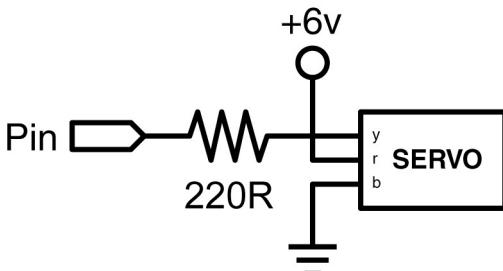
Variable resistors include CdS light sensors, thermistors, flex sensors, and so on. This example makes use of a function to read the analog value and set a delay time. This controls the speed at which an LED brightens and dims.

```
int ledPin    = 9;      // PWM pin for the LED
int analogPin = 0;      // variable resistor on analog pin 0

void setup(){}
void loop()
{
    for (int i=0; i<=255; i++) // ascending value for i
    {
        analogWrite(ledPin, i); // sets brightness level to i
        delay(delayVal());     // gets time value and pauses
    }
    for (int i=255; i>=0; i--) // descending value for i
    {
        analogWrite(ledPin, i); // sets brightness level to i
        delay(delayVal());     // gets time value and pauses
    }
}

int delayVal()
{
    int v;                  // create temporary variable
    v = analogRead(analogPin); // read analog value
    v /= 8;                 // convert 0-1024 to 0-128
    return v;                // returns final value
}
```

servo output



Hobby servos are a type of self-contained motor that can move in a 180° arc. All that is needed is a pulse sent every 20ms. This example uses a servoPulse function to move the servo from 10° -170° and back again.

```
int servoPin = 2;      // servo connected to digital pin 2
int myAngle;          // angle of the servo roughly 0-180
int pulseWidth;        // servoPulse function variable

void setup()
{
    pinMode(servoPin, OUTPUT);    // sets pin 2 as output
}

void servoPulse(int servoPin, int myAngle)
{
    pulseWidth = (myAngle * 10) + 600;    // determines delay
    digitalWrite(servoPin, HIGH);         // set servo high
    delayMicroseconds(pulseWidth);       // microsecond pause
    digitalWrite(servoPin, LOW);         // set servo low
}

void loop()
{
    // servo starts at 10 deg and rotates to 170 deg
    for (myAngle=10; myAngle<=170; myAngle++)
    {
        servoPulse(servoPin, myAngle);    // send pin and angle
        delay(20);                      // refresh cycle
    }
    // servo starts at 170 deg and rotates to 10 deg
    for (myAngle=170; myAngle>=10; myAngle--)
    {
        servoPulse(servoPin, myAngle);    // send pin and angle
        delay(20);                      // refresh cycle
    }
}
```