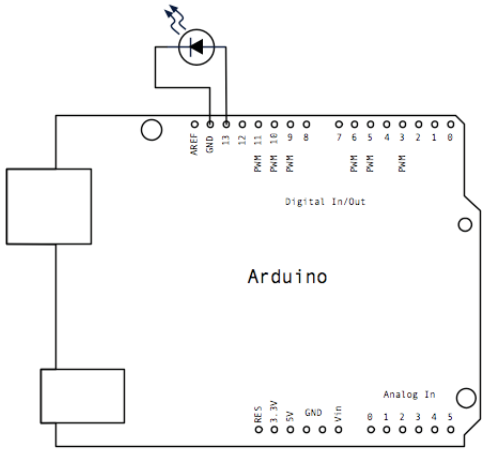
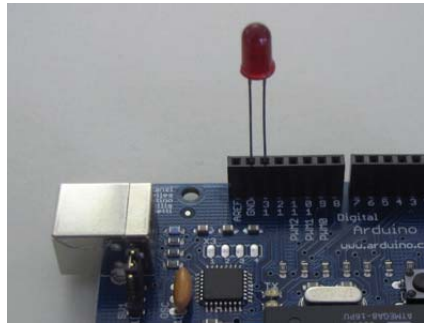
DIGITAL I/O_QUICK RECAP

BLINK

FILE> Examples>Digital>Blink

```
int ledPin =  13;    // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts
void setup()
{
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power
void loop()
{
  digitalWrite(ledPin, HIGH);   // set the LED on
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // set the LED off
  delay(1000);                  // wait for a second
}
```

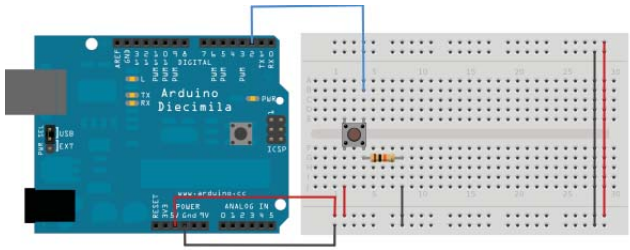


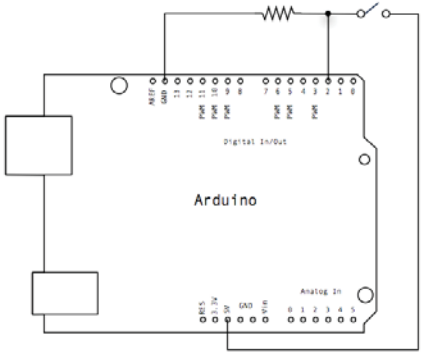


BUTTON

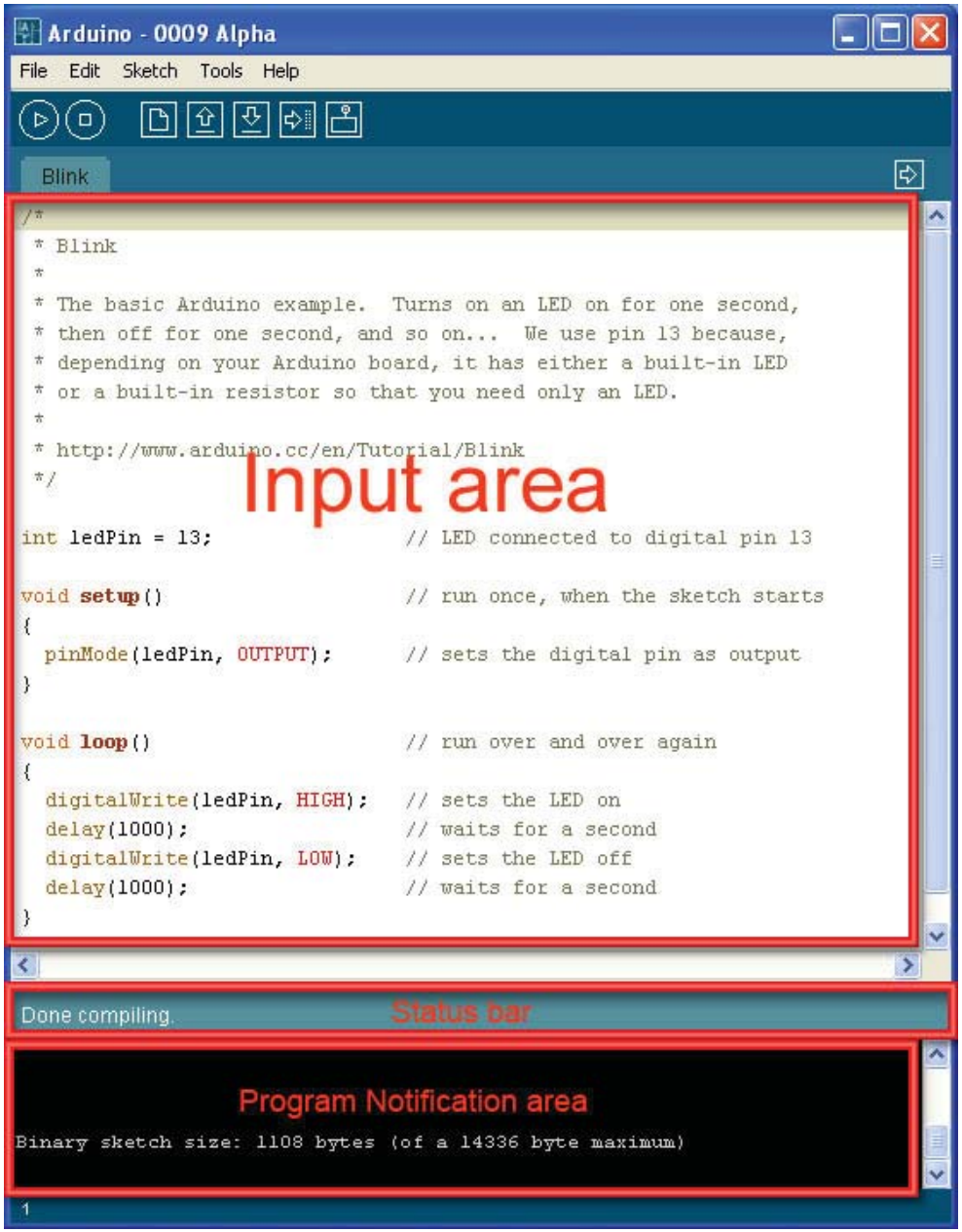
FILE> Examples>Digital>Button

```
 int buttonPin = 2;     // the number of the pushbutton pin
int ledPin =  13;      // the number of the LED pin
int buttonState = 0;         // variable for reading the pushbutton status

void setup()
{
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop()
{
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH)
{
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```





ARDUINO PLATFORM



MEMORY

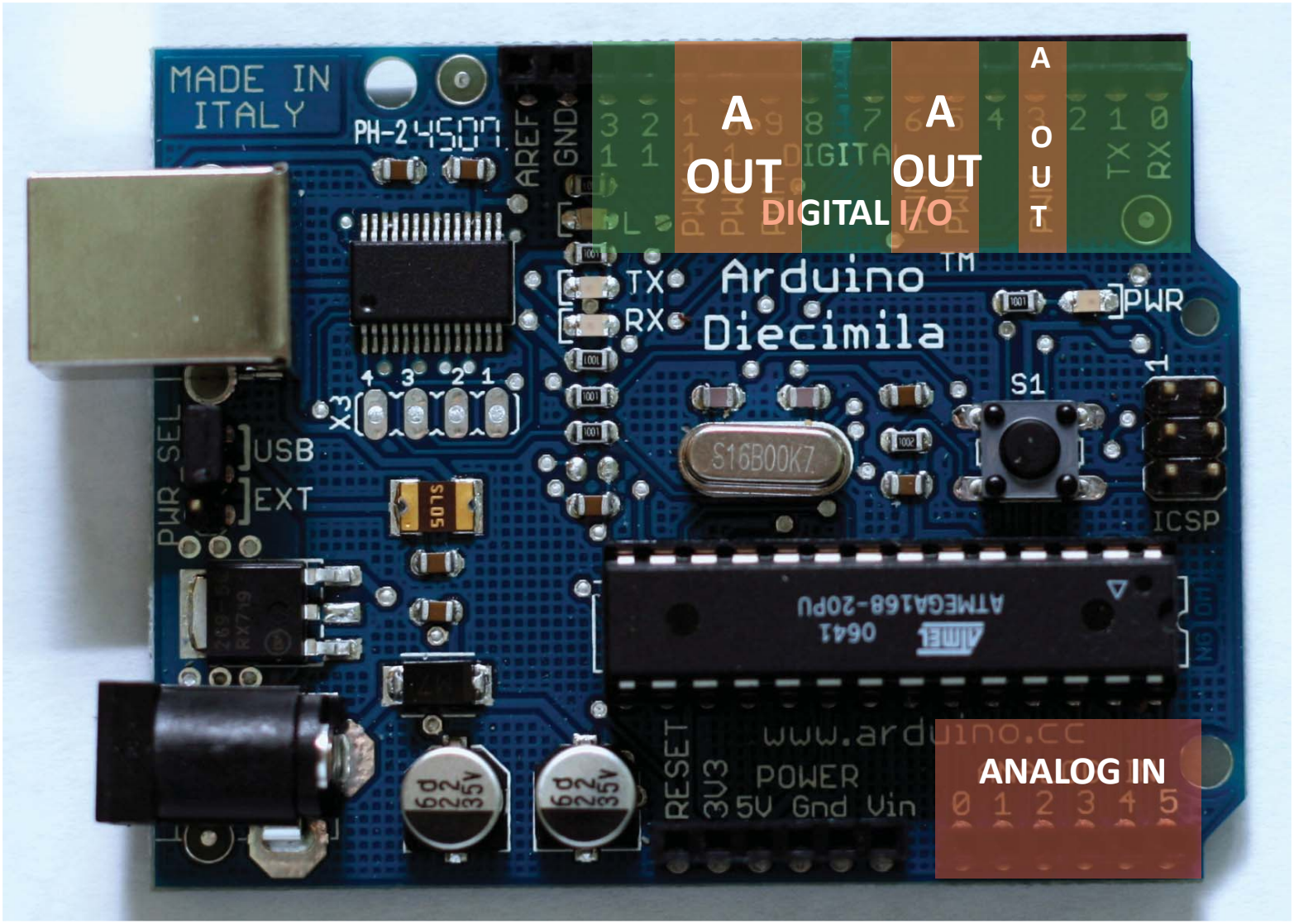
There are three pools of memory in the microcontroller used on Arduino
boards (ATmega168):

- **Flash** memory (program space), is where the Arduino sketch is stored.
- **SRAM**(static random access memory) is where the sketch creates and manipulates variables when it runs.
- **EEPROM**(Electronically Erasable Programmable Read Only Memory) is memory space that programmers can use to store long-term information.

The ATmega168 chip has the following amounts of memory:

- Flash 16k bytes (of which 2k is used for the bootloader)
- SRAM 1024 bytes
- EEPROM 512 bytes

DIGITAL AND ANALOG I/O ON THE BOARD



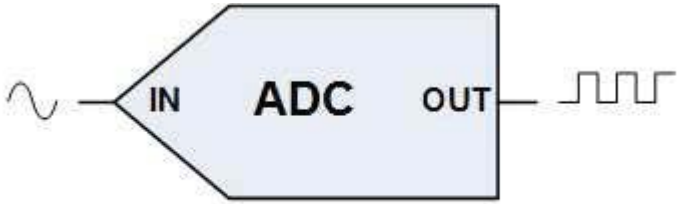
ANALOG DIGITAL CONVERTERS (ADCs)

_ADC is an electronic device that converts an input analog voltage (or current) to a digital number proportional to the magnitude of the voltage or curren

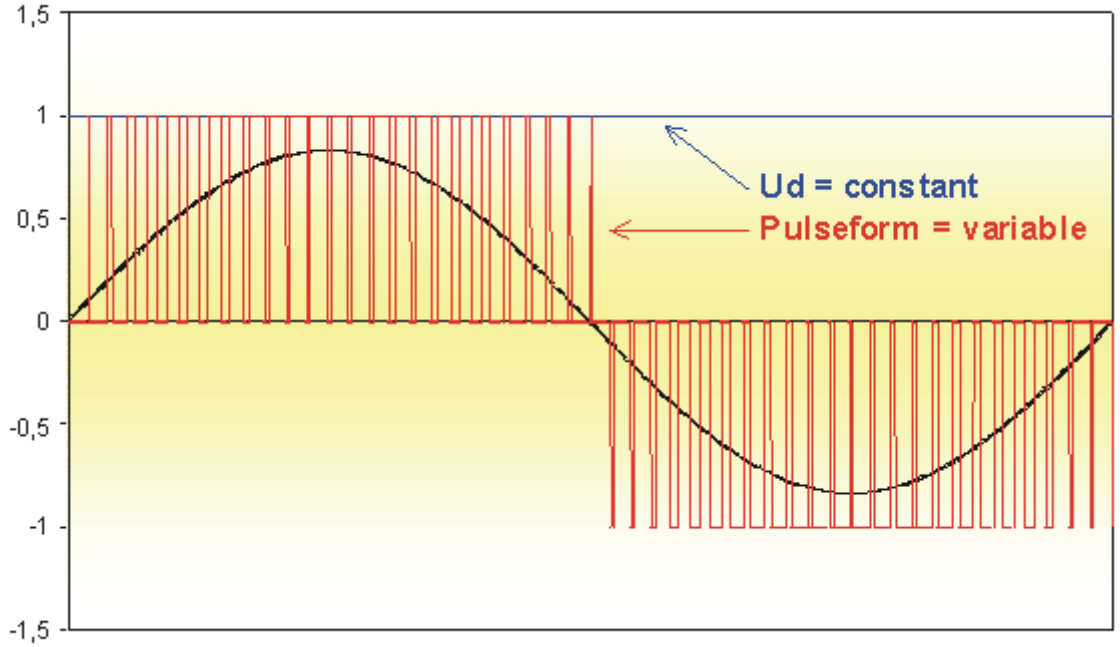
_Arduino has a built in 10-bit ADC which keeps checking the value of the voltage input and returns back the values.

It takes (0.0001 s) per reading.

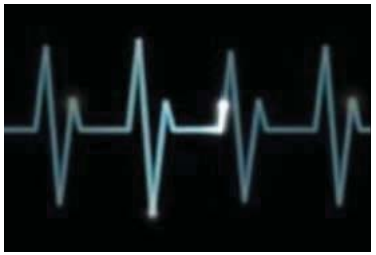
10,000 readings / sec.



'MBOL FOR ANALOG TO DIGITAL CONVERTER (ADC)



PWM



_Pulse Width Modulation

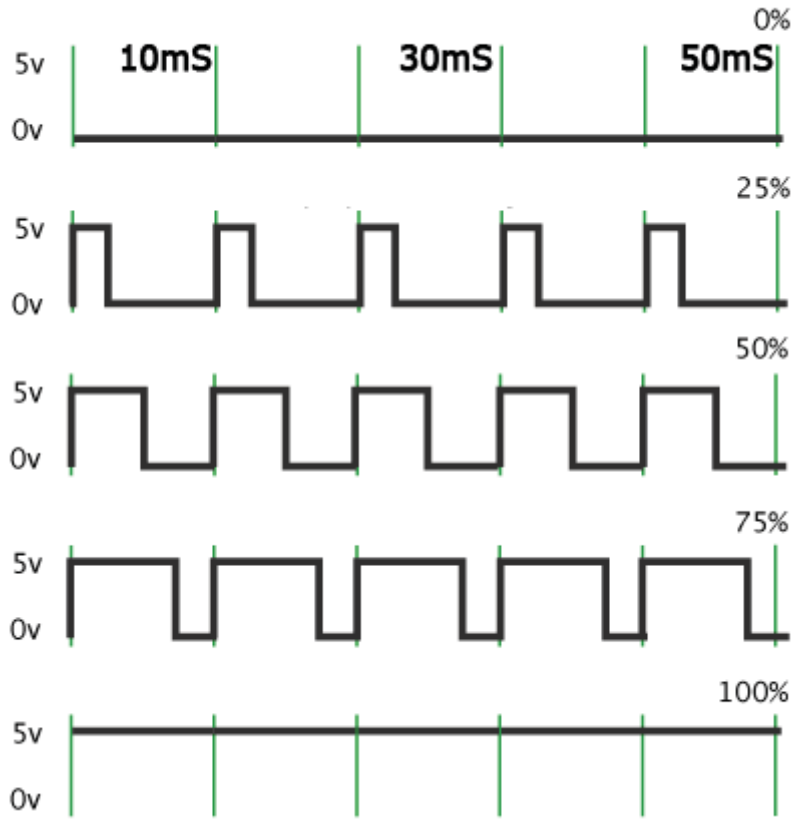
_Digital way of getting an analog current supply.

_ a PWM variable-power scheme switches the power quickly between fully on and fully off

_Pulse: sudden increase and decrease in current flow.

_ The term **duty cycle** describes the proportion of on time to the regular interval or period of time; a low duty cycle corresponds to low power, because the power is off for most of the time

_High = 256 (0-255)



DIGITAL I/O

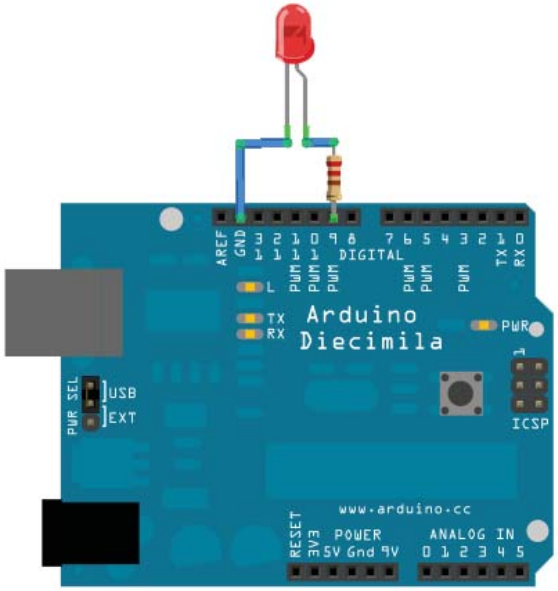
```
pinMode(pin#, INPUT/OUTPUT);  //assigns pin as input or output
digitalWrite(pin#, STATE: HIGH/LOW);  //OUTPUT
digitalRead(pin#); //INPUT
```

ANALOG I/O

```
analogWrite(pin#, the duty cycle: 0 (always off)-255 (always on));  //OUTPUT
analogRead(pin#); //INPUT
```

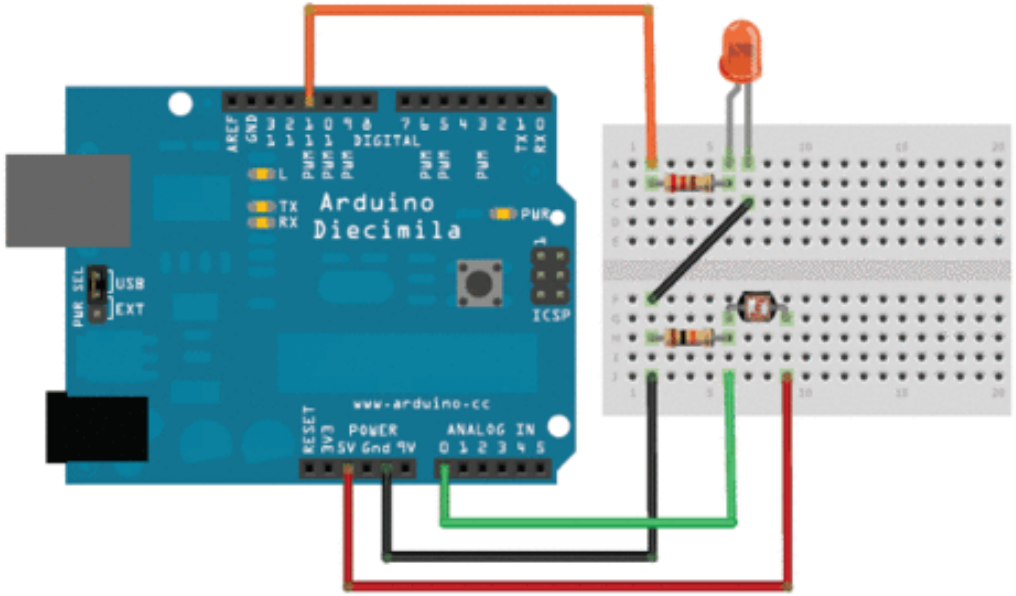
PWM EXAMPLE_led/speaker

```
int analogPin = 9;

void setup()
{
   Serial.begin(9600);
}
void loop()
{
   analogWrite(analogPin, 0);      //0%
   delay(1000);
   analogWrite(analogPin, 64);     //25%
   delay(1000);
    analogWrite(analogPin, 127);    //50%
   delay(1000);
    analogWrite(analogPin, 191);    //75%
   delay(1000);
    analogWrite(analogPin, 255);    //100%
   delay(1000);
}
```

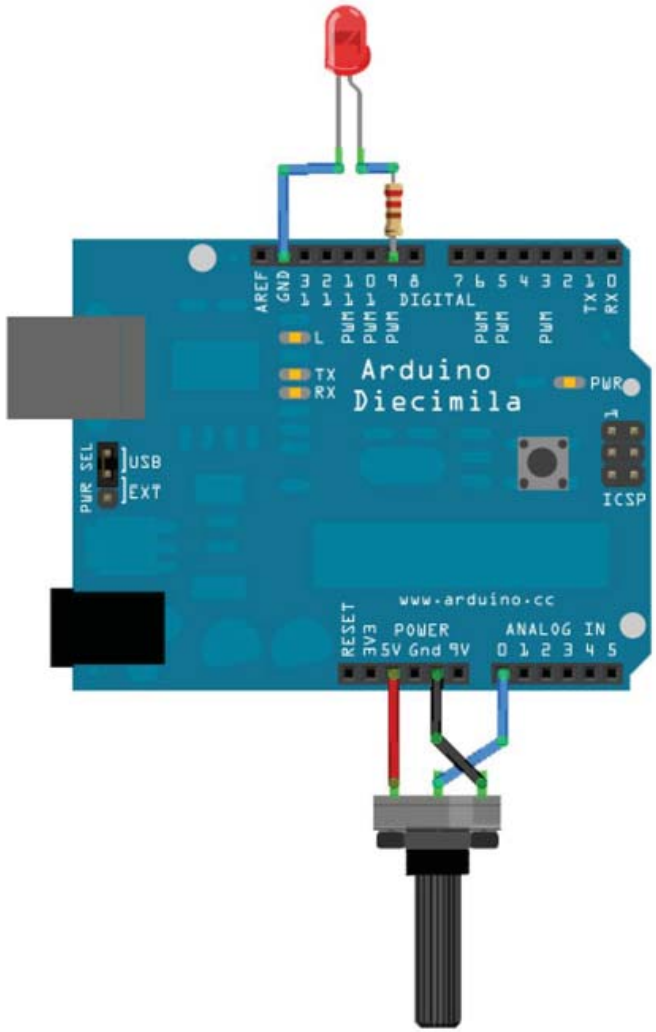


ANALOG IN_photocell/potentiometer

Photocell:



Potentiometer:



ANALOG IN_photocell/potentiometer CODE

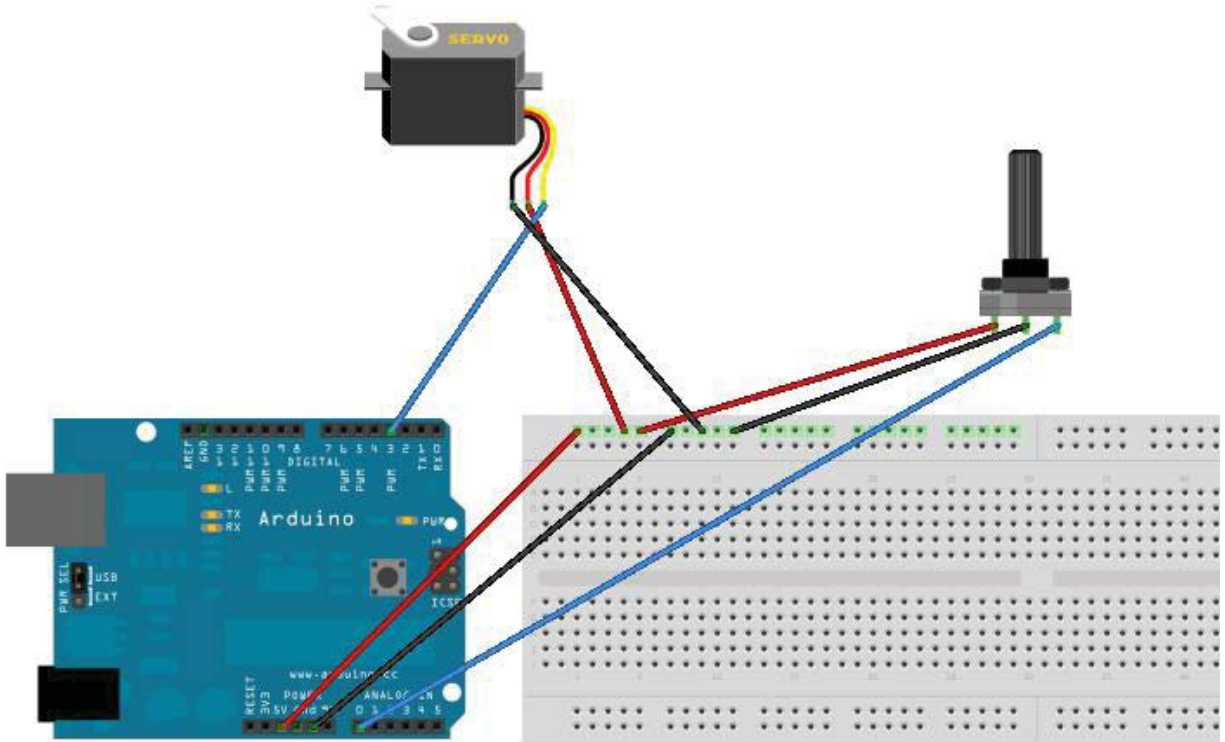
FILE> Examples> Analog> AnalogInput

```
int sensorPin = 0;    // select the input pin for the potentiometer
int ledPin = 9;       // select the PWM pin for the LED
int sensorValue = 0; // variable to store the value coming from the sensor
void setup()
{
      Serial.begin(9600);
}
void loop()
{
      // read the value from the sensor
      sensorValue = analogRead(sensorPin);
      //prints sensor value to program notification area using Serial.print or
      //Serial.println
      Serial.println(sensorValue);
      delay(500);
      // turn the ledPin on according to potentiometer's modulation
      analogWrite(ledPin, sensorValue);
}
```

MAPPING VALUES

```
int sensorPin = 0;    // select the input pin for the potentiometer
int ledPin = 9;       // select the PWM pin for the LED
int sensorValue = 0; // variable to store the value coming from the sensor
int outputData = 0; // Assigning integer value to the variable outputData

void setup()
{
      Serial.begin(9600);
}
void loop()
{
      // read the value from the sensor
      sensorValue = analogRead(sensorPin);
      // map it to the range of the analog out
      outputData = map(sensorValue, 0, 1023, 0, 255);
      // turn the ledPin on according to potentiometer's modulation
      analogWrite(ledPin, sensorValue);
}
```

SERVOS



SERVOS

http://www.arduino.cc/playground/ComponentLib/Servo

```
#include <Servo.h>

Servo myservo;        //create servo object to control a servo
int potpin = 0;       // analog pin used to connect the potentiometer
int val = 0;          // variable to read the value from the analog pin
void setup()
{
      myservo.attach(2); // attaches the servo on pin 2 to the servo object
}
void loop()
{
      val = analogRead(potpin);          // reads the value of the potentiometer (value between 0 and 1023)
      val = map(val, 0, 1023, 0, 180);   // scale it to use it with the servo (value between 0 and 180)
      myservo.write(val);                // sets the servo position according to the scaled value
      delay(15);                         // waits for the servo to get there
}
```

SERVOS_SWEEP

FILE> Examples>Servo> Sweep

```
#include <Servo.h>
 Servo myservo;  // create servo object to control a servo
                 // a maximum of eight servo objects can be created
int pos = 0;     // variable to store the servo position

void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}


void loop()
{
  for(pos = 0; pos < 180; pos ++)  //goes from 0 degrees to 180 degrees in steps of 1 degree
  {
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for(pos = 180; pos>=1; pos--)     // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```