Arduino Uno programming

Contents

What is an Arduino, Download and learn the IDE, Our first programs

Download the software for the Arduino.

You can load new programs onto the main chip - ATmega328p – via USB using the Arduino IDE. Visit the link below to download the latest Arduino IDE:

http://arduino.cc/en/Main/Software

You can get a free e-book about arduino here :

http://randomnerdtutorials.com/ebook/.

What is an Arduino ?

The Arduino is a small development board with a brain (also known as a micro-controller) that you can program. It interacts with the real world through leds, sensors, motors, LCDs, buzzers, etc...

If you type on your search engine the query “Arduino projects”, you will find tons of amazing Projects.

Arduino is essentially a tiny computer that can connect to electrical circuits. The Arduino Uno is powered by an ATmega328P chip, it is the biggest chip on the board as you can see on Figure 1. That’s where you store your programs.

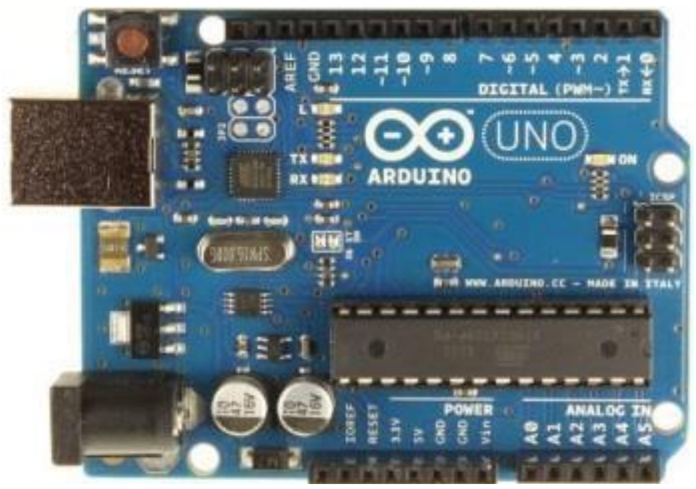


Figure 1 Arduino UNO R3 board with ATmega328P

The top row of the Arduino has 14 digital pins, labeled 0-13. These pins can act as either inputs or outputs. You can connect them to your circuits to turn them on or off. You can also read buttons – see if a button is either pressed or not.

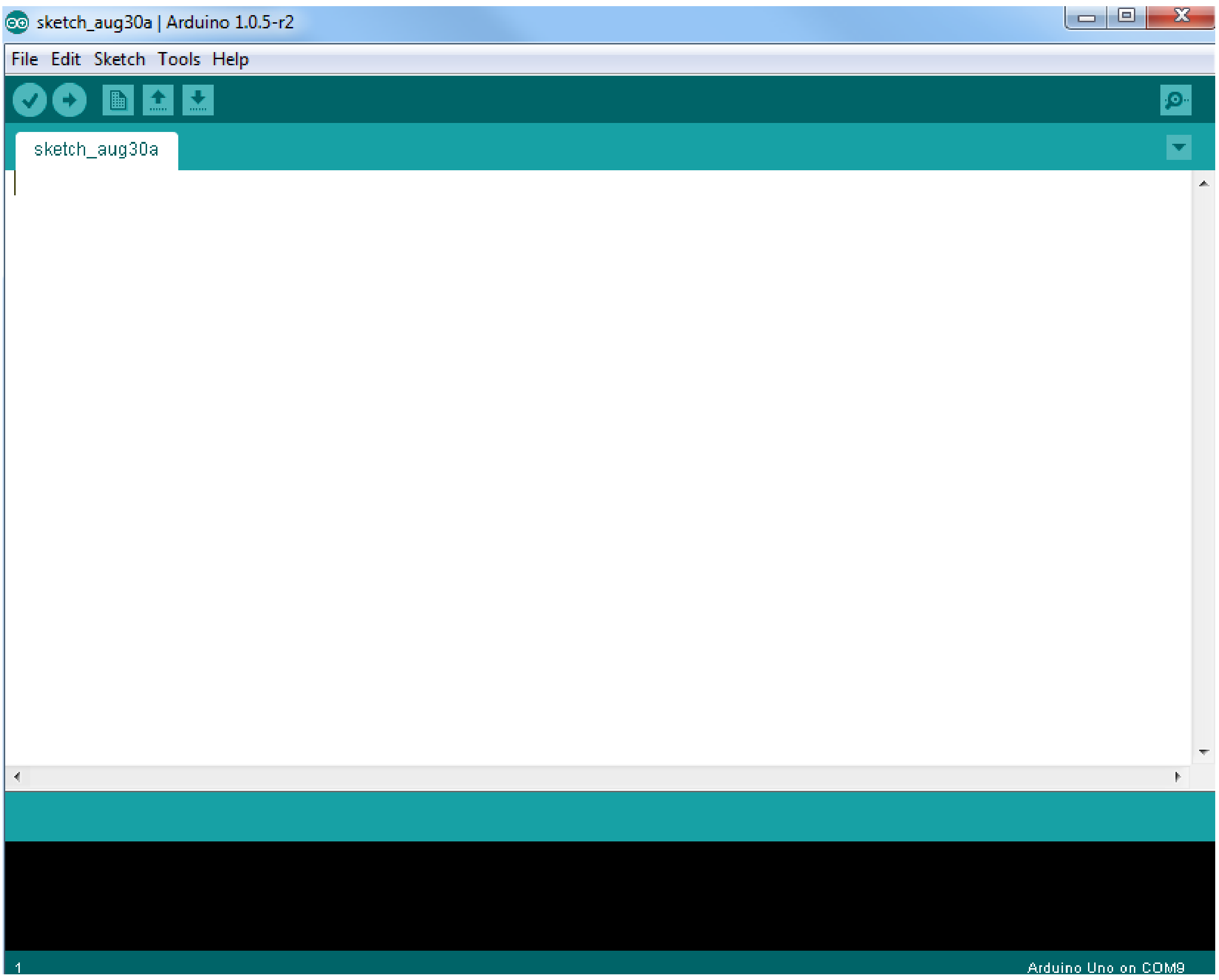
On the bottom left row, you can see the power pins. The Arduino has 3.3V or 5V supply. This is really useful since most components require 3.3V or 5V. You will also find some pins labeled "GND" on the Arduino, these are ground pins On the bottom right row, you can see the analog input pins, labeled A0-A5. These pins are used to make analog measurements of sensors or other components. Analog inputs are especially good for measuring things with a range of possible values. For example, measuring temperature sensors or potentiometers.

Start your Arduino IDE.

Now you should see a similar window on your computer.

Note :
Check the COM port mentioned in the lower right corner. It must be the COM port, that you use. But the Arduino fails sometimes in recognizing the correct.

Under “Tools” it’s possible to change it manually.



Uploading your first sketch.

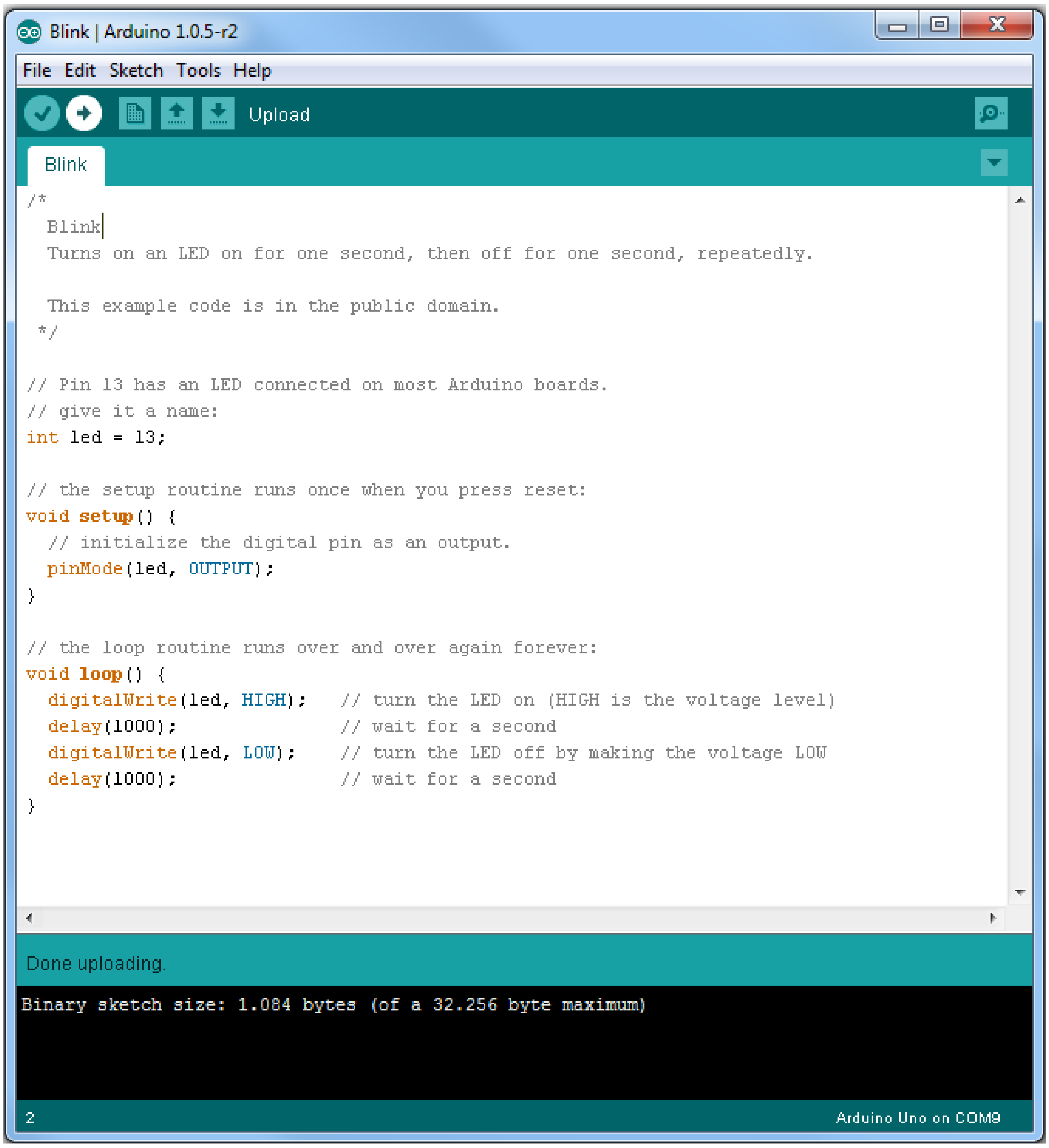
Connect your Arduino UNO to your computer via USB.
For this example, you will be uploading the most basic example that the Arduino has. Which is blinking an on-board LED or digital pin 13.

Open your Arduino IDE.

Go to :

```
    File > Examples > 01.Basics > Blink
```

When the sketch is loaded, left click on upload



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

```
Done uploading.
Binary sketch size: 1.084 bytes (of a 32.256 byte maximum)
```

Wait until "Done uploading" message appear at the button line. This code simply blinks the on-board LED on your Arduino UNO (highlighted with red color). If you look closely you should see the LED staying on for one second and off for another second repeatedly.

Note :
There is NOT a `main()` procedure.
Instead we have a `setup()` and a `loop()` procedure.

The Arduino Uno has only one processor and no operating system. So we can run one and only one program on it. It begins with `setup()`, which runs a single time and continues with **loop()** (yes you guessed it!) an infinite number of times.

Control an Output and read an Input

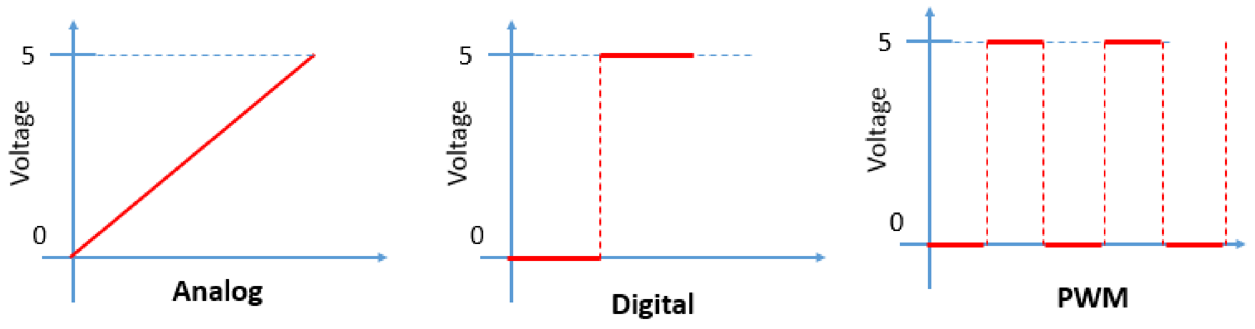
An Arduino board contains digital pins, analog pins and PWM pins.

Difference between digital, analog and PWM

In **digital pins**, you have just two possible states, which are on or off. These can also be referred as High or Low, 1 or 0 and 5V or 0V.
For example, if an LED is on, then, its state is High or 1 or 5V. If it is off, you'll have Low, or 0 or 0V.

In **analog pins,** you have unlimited possible states between 0 and 1023. This allows you to read sensor values. For example, with a light sensor, if it is very dark, you'll read 1023, if it is very bright you'll read 0 If there is a brightness between dark and very bright you'll read a value between 0 and 1023.

PWM pins are digital pins, so they output either 0 or 5V. However, these pins can output "fake" intermediate voltage values between 0 and 5V, because they can perform "Pulse Width Modulation" (PWM). PWM allows to "simulate" varying levels of power by oscillating the output voltage of the Arduino.



Controlling an output

To control a digital output you use the function `digitalWrite()` and between
brackets you write, the pin you want to control, and then HIGH or LOW if you want to turn something on or off.
To control a PWM pin you use the function `analogWrite()` and between Brackets you write the pin you want to control and a number between 0 and 255.

Reading an input

To read an analog input you use the function `analogRead()` and for a digital input you use `digitalRead()`.

Assignments 1

1. Let the Arduino write "Hello World" on the serial monitor.
2. Blink the internal LED (Pin 13) 10 sec on and 5 sec off.
3. Input a number in the Serial monitor, let the Arduino multiply by 2 and display the result on the Serial monitor.
4. Try to make a simple stop watch Display minutes and seconds on the Serial Monitor (use delay).
5. For you who has a button and small wires to connect to Arduino. Take ass 4 and connect the button from pin 4 to ground. The reset button will start the clock, the other button will stop the watch.

For today there are two projects, to getting closer to the Arduino, and a little extra.

Turn your LED on and OFF by the light.

Components you need.

- LED – Light emitting diode.
- LDR – Light dependent resistor.
- Resistor 220 Ohm.
- Resistor 1000 Ohm.

In this project we are going to read data from a light sensor (photoresistor). Whit this sensor we can know whether there's light or not. This is useful in automatic lights. When the night is coming, the lights turn on automatically. In our project:

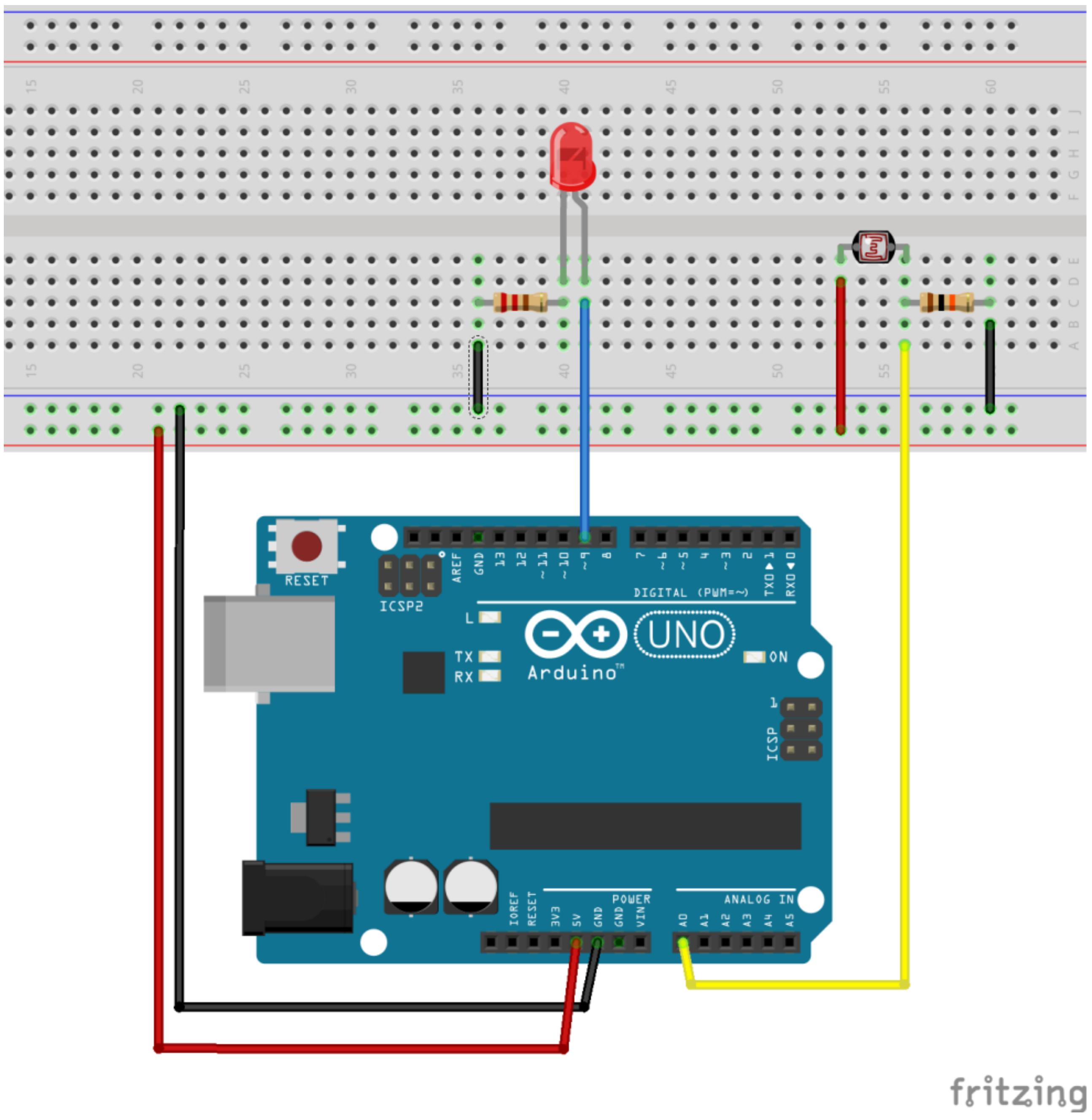
- If there isn't light, the LED will turn on
- If there is light, the LED will turn off

What is a photoresistor?

A photoresistor is a light-dependent resistor. The resistance of a photoresistor decreases with increasing of light intensity. So:

- When there is light, the resistance decreases, we will have more current flowing.
- When there is no light, the resistor increases, we will have less current flowing.

Put everything together as the schematics below:



Then, upload the following code:

```
int ledPin          =  9;
int ledBrightness   =  0;
int sensorPin       = A0;
int sensorValue     =  0;

void setup(void) {
    pinMode(ledPin, OUTPUT);
    // We'll send debugging information
    // via the Serial monitor
    Serial.begin(9600);
}
```

```
void loop(void) {
    sensorValue   = analogRead( sensorPin );
    Serial.print( "Sensor reading: " );
    Serial.println( sensorValue );

    // LED gets brighter the darker it is at the sensor
    // that means we have to -invert- the reading
    // from 0-1023 back to 1023-0
    sensorValue = 1023 - sensorValue;

    // now we have to map 0-1023 to 0-255 since
    // that's the range analogWrite uses
    ledBrightness = map( sensorValue, 0, 1023, 0, 255);
    analogWrite( ledPin, ledBrightness);
    delay( 50 );
}
```

In the code we use the `map()` function. This function should be used as follows:

```
Map( value, fromLow, fromHigh, toLow, toHigh)
```

We use this function when we have a value within a certain range, and we want to fit that value into another range. In our case, we are reading an analog value that is between 0 and 1023. Then, we want to output that value. However, the output pins, output a range between 0 and 255, so we need to adjust that value to this new range.

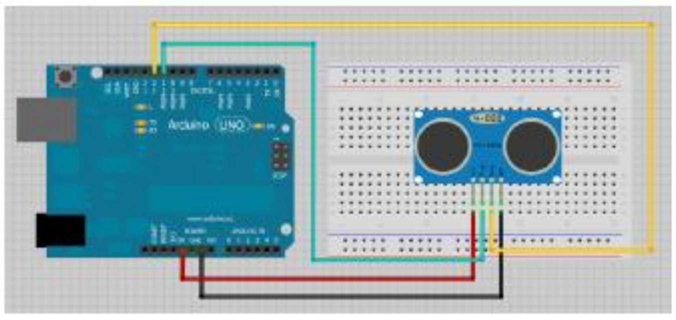
The ultrasonic sensor.

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2 cm to 400 cm or 1” to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

This sensor is really cool and popular among the Arduino Tinkerers. So I've decided to post a project example using this sensor. In this project the ultrasonic sensor read and write the distance in the serial monitor.

It's really simple. My goal is to help you understand how this sensor works and then you can use this example in your own projects.

Note
There's an Arduino library called NewPing that can make your life easier when using this sensor.



Source code

View code on GitHub

```
/*
* created by Rui Santos, http://randomnerdtutorials.com
*
* Complete Guide for Ultrasonic Sensor HC-SR04
*
* Ultrasonic sensor Pins:
* VCC  : +5VDC
* Trig :  Trigger (INPUT) - Pin11
* Echo :  Echo (OUTPUT) - Pin 12
* GND  :  GND
*/
int trigPin = 11;  // Trig - green Jumper
int echoPin = 12;  // Echo - yellow Jumper
long duration, cm, inches;
void setup() {
    //Serial Port begin
    Serial.begin( 9600 );

    //Define inputs and outputs
    pinMode( trigPin, OUTPUT );
    pinMode( echoPin, INPUT  );
}

void loop()
{
    // The sensor is triggered by a HIGH pulse of 10 or
    // more microseconds.
    // Give a short LOW pulse beforehand to ensure
    // a clean HIGH pulse:
    digitalWrite( trigPin, LOW  );
    delayMicroseconds(  5 );
    digitalWrite( trigPin, HIGH );
    delayMicroseconds( 10 );
    digitalWrite( trigPin, LOW  );

    // Read the signal from the sensor: a HIGH pulse whose
    // duration is the time (in microseconds) from the
    // sending of the ping to the reception of its echo
    // off of an object.
    pinMode(  echoPin, INPUT);
    duration = pulseIn(echoPin, HIGH);

    // convert the time into a distance
    cm       = (duration/2) / 29.1;
    inches   = (duration/2) / 74;
    Serial.print( inches );
    Serial.print( "in, " );
    Serial.print(  cm );
    Serial.print( "cm" );
    Serial.println( );
    delay(250);
}
```

Another sensor is the obstacle avoidance sensor, easy to use try this out.

```
int led       = 13; // define LED Interface
int buttonpin =  3; // define the obstacle avoidance
                     // sensor interface
int val       =  0; // define numeric variable val

void setup ()
{
    // define LED as output interface
    pinMode ( led, OUTPUT);

    // define the obstacle avoidance
    // sensor output interface
    pinMode ( buttonpin, INPUT) ;
}

void loop ()
{
    // digital interface will be assigned
    // a value of 3 to read val
    val = digitalRead (buttonpin) ;

    // When the obstacle avoidance sensor
    // detects a signal, LED flashes
    if (val == HIGH)
    {
          digitalWrite ( led, HIGH);
    }
    else
    {
          digitalWrite ( led, LOW);
    }
}
```

Assignments 2

6. Connect an LDR and an LED to the Arduino as shown above. Lay your hand over the LDR, now the LED should lite on, when you remove your hand the LED turns off.

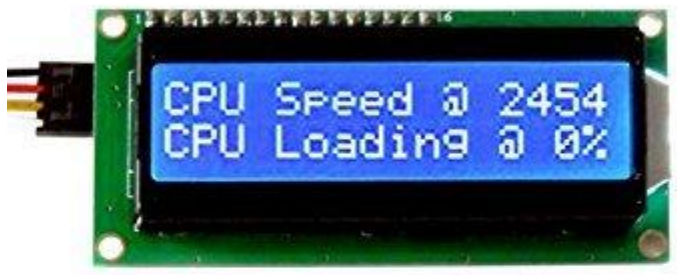
7. Dim the LED up and down depending on the light on the LDR (use `analogWrite()` (PWM)

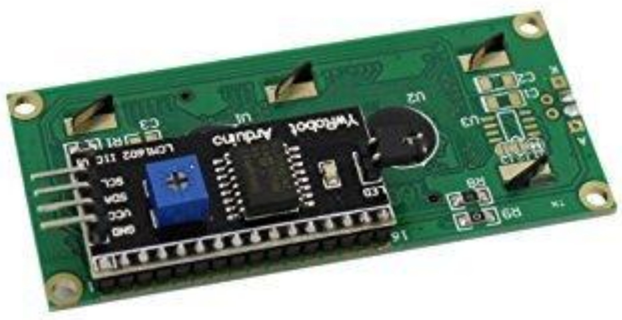
8. Replace the LDR with a potentiometer, see if you can get the led to turn on and off by turning.

9. Connect the Sonic sensor as shown try now to set it up, and use it as a counter, count up every time something I passing.

10. Setup your equipment so that you can measure distance, control if it is correct.

Today we will connect a display and a keypad to the Arduino





This 16 character by 2 lines is a very clear display with high contrast.
White text on blue background.
It contains a serial adaptorcard on its backside. It means that only two wires are used for communication to the display. SDA, SCL, 5V power and GND is all what we are using.
We have to be sure that we have the LiquidCrystal_I2C.h file, in this are all the commands for the display.

https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/

This is a display using the protocol I2C. I hope that you have this.

Connect it to the Arduino as shown, only 4 lines.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//   I2C LCD 1602    Arduino
//      GND         GND
//      VCC         5V
//      SDA         A4
//     SCL        A5

//  Set the LCD address to 0x27 for a 16 chars and
//  2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{
    // initialize the LCD
    lcd.init();

    // Turn on the blacklight and print a message.
    lcd.backlight();
    lcd.print("Hello, world!");
}
void loop()
{
    // Do nothing here...
}
```

This is a simple program, now we try to get something real to happen.

Connect your keypad to the Arduino, make a program that can sense what button you press.

You can find a lot of examples on the Internet.

Display it on the Serial monitor.

Assignments 3

11. Combine the program with the display, and the program with the light sensor (LDR) display the value when you have sun on your LDR.
12. Combine the Clock program and display the time (Real time). Enter the time when you start, and let the clock show the time.
13. Combine the distance sensor with the display program, show the distance, and with a button change between meter and inch.
14. On the display show text you enter in the serial monitor.
15. From the keypad enter 4 digits, make a secret code turn the internal LED pin13 on and off when you reach the secret code, display on the external display.