

RANDOM NERD TUTORIALS - PROJECTS

18 + RANDOM NERD TUTORIALS PROJECTS



Parts + Schematics + Code + Demonstration



Compilation of 18+ Arduino Projects and Tutorials!
by Rui Santos of The RandomNerdTutorials.com Blog

Table of Contents

Disclaimer	3
Introduction	4
Connect with Rui	5
Parts required	6
Introducing the Arduino	9
Ultrasonic Sensor with LEDs and buzzer	14
Poor Man's Oscilloscope.....	18
Control LEDs with IR Remote Control	23
Control Servo with Visual Basic	29
Control DC Motor via Bluetooth	32
Temperature Displayed on 4 Digit 7 segment (common anode).....	37
LED Cube 3x3x3 with Arduino	43
Control 2 DC Motors via Bluetooth	46
How to Use App Inventor with Arduino	50
Webserver with an Arduino + Ethernet Shield	54
Complete Guide for Ultrasonic Sensor HC-SR04	61
Datalogger with Temperature Sensor and Photoresistor	67
Teensy/Arduino - Memory Game	73
Android App that Sends a Message to Your Arduino.....	81
Control your Arduino with Voice Commands [Android App]	85
Display the LED Brightness on a LCD 16x2.....	89
Teensy - Username and Password Auto Filler.....	93
Arduino with PIR Motion Sensor.....	97
Resources	100
Final Words from Rui	103
Download Arduino Step-by-step Projects Course.....	104

Disclaimer

This eBook has been written for information purposes only. Every effort has been made to make this eBook as complete and accurate as possible.

The purpose of this eBook is to educate. The author (Rui Santos) does not warrant that the information contained in this eBook is fully complete and shall not be responsible for any errors or omissions. The author (Rui Santos) shall have neither liability nor responsibility to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by this eBook.

This eBook contains code examples which you can use on your own projects, excepted where otherwise noted.

You cannot redistribute this eBook.

This eBook is only available for free download at:

<http://randomnerdtutorials.com/download>

Please send an email to the author (Rui Santos - hello@ruisantos.me), if you have found this eBook anywhere else.

Throughout this eBook you will find some Amazon links and some of them are affiliate links. This means the author (Rui Santos) earns a small commission from each purchase with that link. **That has no additional cost to you**, but helps paying for the expenses of purchasing new components for the projects presented in this eBook.

Introduction

You've probably found your way to this eBook because you found my blog through Google, YouTube or you simply saw one of my projects featured on Instructables, Freetronics, Hackaday, etc... Anyway I'm truly happy you are here.

This eBook is a compilation of my projects. It wasn't created to read from start to finish, you can skip from one project to another.

I encourage you to watch some of the video demonstrations, since some of my projects are easier to understand if you can see the circuit in action.

This eBook has the purpose to inspire you create something amazing with electronics and programing. After you create something cool, I hope you share it with others. That's the whole goal of this awesome community.

To all my readers, thank you for your interest in my work. I really appreciate!

Have fun with your projects,

Rui Santos

P.S. Make sure you visit my website to see the latest projects!

<http://RandomNerdTutorials.com>

Connect with Rui

If at any point while you are reading you have any questions, please don't hesitate to contact me. Here are some ways to stay in touch. Choose your preferred way and click one of those links below!

Through social media, I usually share once a day a cool project I've found on the web or simply what I'm currently working on.



[Visit my website](#)

(<http://RandomNerdTutorials.com>)



[Subscribe on YouTube](#)

(<https://www.youtube.com/user/RandomNerdTutorials>)



[Like on facebook](#)

(<https://www.facebook.com/RandomNerdTutorials>)



[Follow me on Twitter](#)

(<https://twitter.com/RuiSantosdotme>)



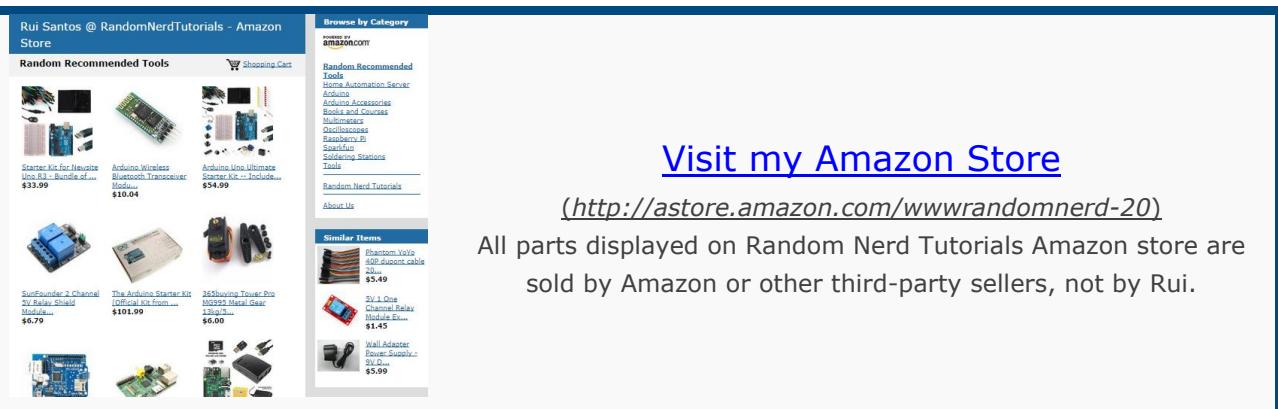
[Fork me on GitHub](#)

(<https://github.com/RuiSantosdotme>)

Parts required

I thought it would be helpful to create a parts required topic. Since it makes much easier to show most of the components/tools used throughout this eBook and where you can get them.

Purchase your parts on Random Nerd Tutorials Amazon Store!



The screenshot shows the homepage of the Random Nerd Tutorials Amazon Store. It features a sidebar with 'Random Recommended Tools' including an Arduino Uno Ultimate Starter Kit, an Arduino Wireless Bluetooth Transceiver, and an Arduino Starter Kit. Below this are sections for SunFounder 2 Channel 2V Relay Shield, The Arduino Starter Kit, and 26Shiny Tower Pro. To the right is a 'Browse by Category' sidebar with links like Home Automation, Arduino, Books and Courses, and Raspberry Pi. A 'Similar Items' section lists items like a Phantom Volo 400P Support cable, a 5V 1 Channel Relay, and a Wall Adapter Power Supply. At the bottom right is a link to 'Visit my Amazon Store' with the URL <http://astore.amazon.com/wwwrandomnerd-20>.

[Visit my Amazon Store](#)
(<http://astore.amazon.com/wwwrandomnerd-20>)

All parts displayed on Random Nerd Tutorials Amazon store are sold by Amazon or other third-party sellers, not by Rui.

Figure	Part	Where to buy
	Arduino Uno Ultimate Starter Kit - Includes 72 page Instruction Book	Click to see on Amazon (http://amzn.to/1BNsQX6)
	Arduino UNO R3 board with ATmega328P	Click to see on Amazon (http://amzn.to/YMjlZg)
	Breadboard 400-point	Click to see on Amazon (http://amzn.to/1nIJNhR)



Wire Jumper Cable
Male-Male, Female-
Female, Female-Male
(3 x 40P)

[Click to see on Amazon](#)
(<http://amzn.to/1q5Li8g>)



Ethernet Shield W5100
with MicroSD Card Slot
for Arduino

[Click to see on Amazon](#)
(<http://amzn.to/1qzJvVB>)



Resistor Kit, 10 Ohm -
1M Ohm (400 Piece, 16
Value)

[Click to see on Amazon](#)
(<http://amzn.to/1tCQpwk>)



Assorted Clear LED w/
Resistors (8 Colors,
Pack of 80, 5mm)

[Click to see on Amazon](#)
(<http://amzn.to/1l2V1fv>)



Hc-sr04 Arduino
Ultrasonic Distance
Measuring Sensor
Module

[Click to see on Amazon](#)
(<http://amzn.to/1pSPpVs>)



Arduino Wireless
Bluetooth Transceiver
Module Slave 4Pin
Serial + DuPont Cable

[Click to see on Amazon](#)
(<http://amzn.to/XHaTKJ>)



LCD Module 20 x 4
(White on Blue)

[Click to see on Amazon](#)
(<http://amzn.to/1nlLDzc>)



Capacitor Kit (33 Value
645 Piece)

[Click to see on Amazon](#)
(<http://amzn.to/1tCRHal>)



Micro small servo
motor RC Robot
Helicopter (2 Pcs
TowerPro SG90 9G)

[Click to see on Amazon](#)
(<http://amzn.to/1ppjEn0>)



MultiMeter - Extech
MN35 Digital Mini

[Click to see on Amazon](#)
(<http://amzn.to/1wkiRbI>)



Soldering Station
Variable Power
Between 5-40W, a
1.5mm Pointed Tip

[Click to see on Amazon](#)
(<http://amzn.to/1ppjSdX>)



Rigol DS1102E 100MHz
Digital Oscilloscope,
Dual Analog Channels,
1 GSa/s Sampling, USB
Storage

[Click to see on Amazon](#)
(<http://amzn.to/1tCSp7Q>)

Introducing the Arduino

The Arduino is a small development board with a brain (also known as a microcontroller) 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.

What's an Arduino?

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 the picture above. That's where you store your programs.



Arduino UNO R3 board with ATmega328P – ([Click to see on Amazon](#) <http://amzn.to/YMjIZq>)

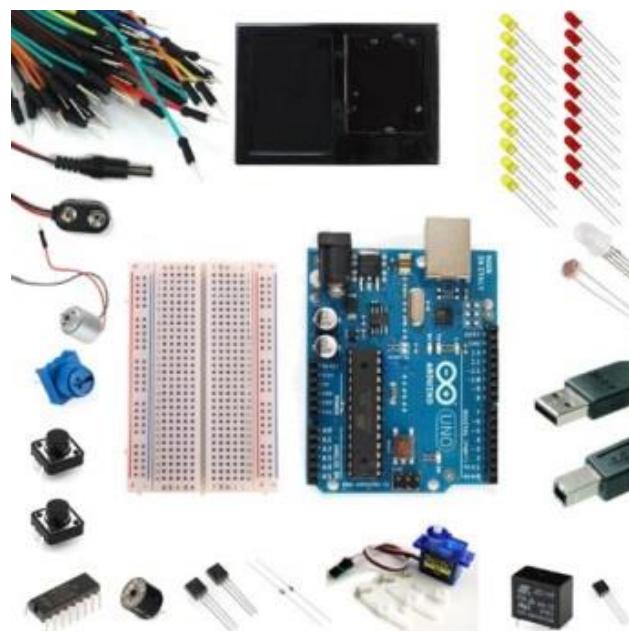
The top row of the Arduino has 14 digital pins, labeled 0-13. These pins can act as either inputs or outputs. You can connect them to your circuits to turn them on or off. You can also read buttons – see if a button is either pressed or not.

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.

What you will need?

In my opinion, the best way to enter the Arduino world is by purchasing one of those Arduino Uno starter kits that contain all the basic components – take a look at the starter kit below.



Arduino Uno Ultimate Starter Kit – ([Click to see on Amazon](http://amzn.to/1BNsQX6) <http://amzn.to/1BNsQX6>)

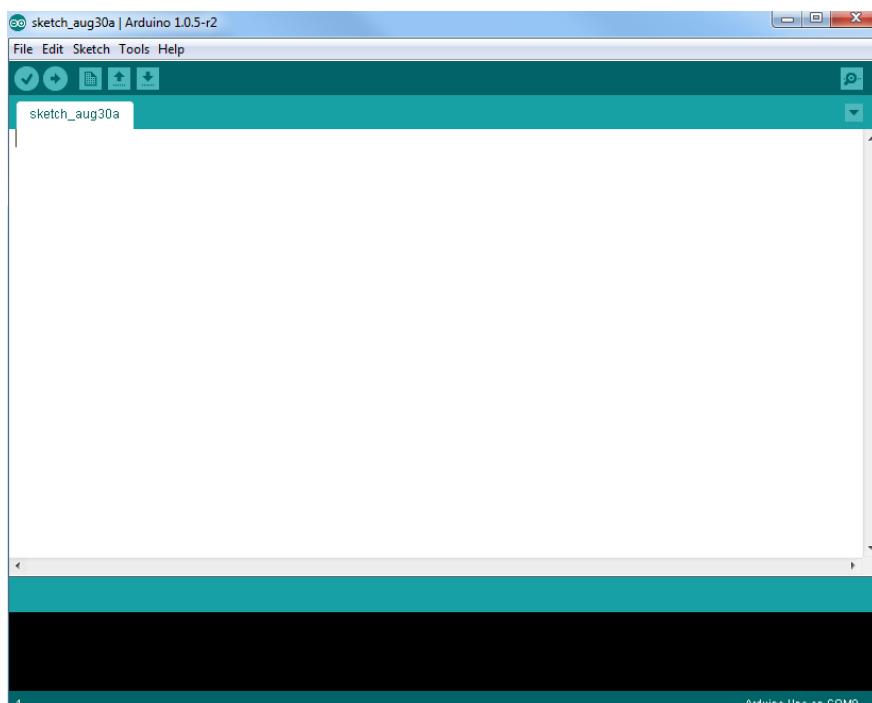
Downloading the Arduino IDE

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>

I won't go into much detail how to install this software, since the official Arduino website does a great job explaining how to do it any of the three operating systems (Windows, Mac and Linux).

In the end, you should see a similar window on your computer.



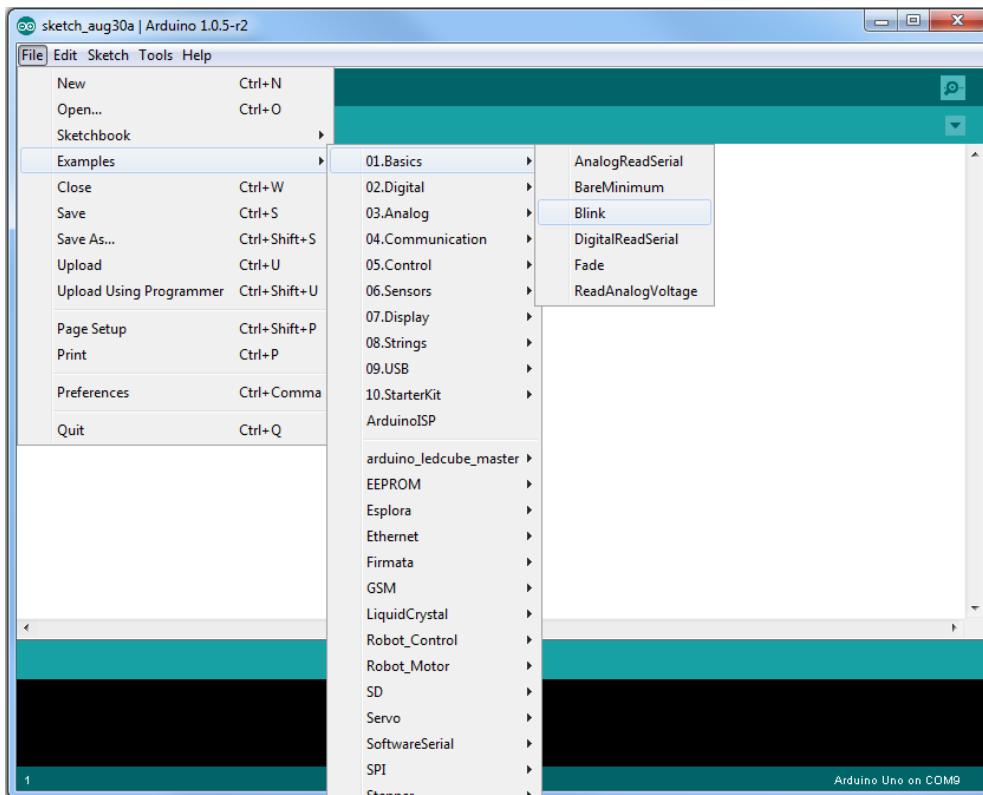
Uploading an Arduino Sketch

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**



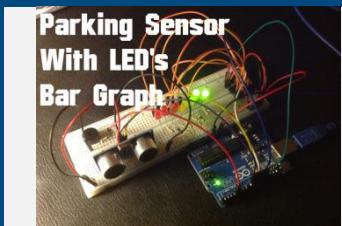
By default your Arduino IDE comes pre-configured for the Arduino UNO. Left-click the “*Upload*” button and wait a few seconds until a “*Done uploading.*” message appears.

A screenshot of the Arduino IDE showing the "Blink" sketch. The code is displayed in the central workspace. At the top, there's a toolbar with icons for upload, refresh, and other functions. Below the toolbar is a status bar with "Blink" and "Upload" buttons. The status bar also shows "Done uploading." and "Binary sketch size: 1.084 bytes (of a 32.256 byte maximum)". The status bar at the bottom right shows "Arduino Uno on COM9".

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.



Ultrasonic Sensor with LEDs and buzzer



View Project on Random Nerd Tutorials

Click [here](#)

Watch on YouTube

Click [here](#)

View code on GitHub

Click [here](#)

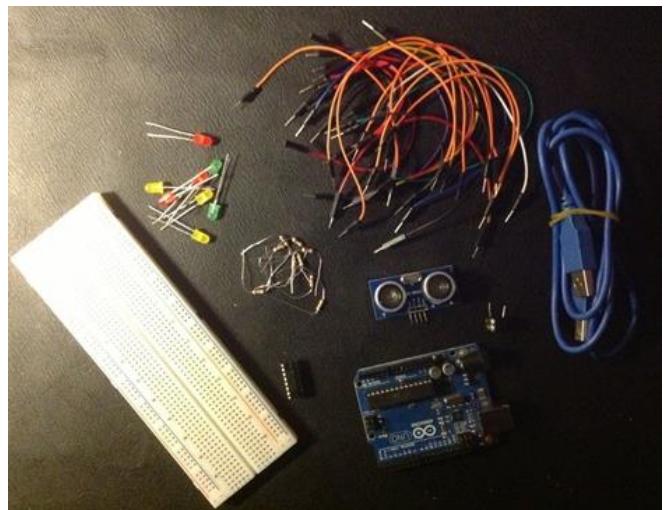
Introduction

In this project we have an ultrasonic sensor that measures the distance and the LEDs bar graph will light up according to our distance from the sensor. As we get closer to the sensor the buzzer beeps in a different way.

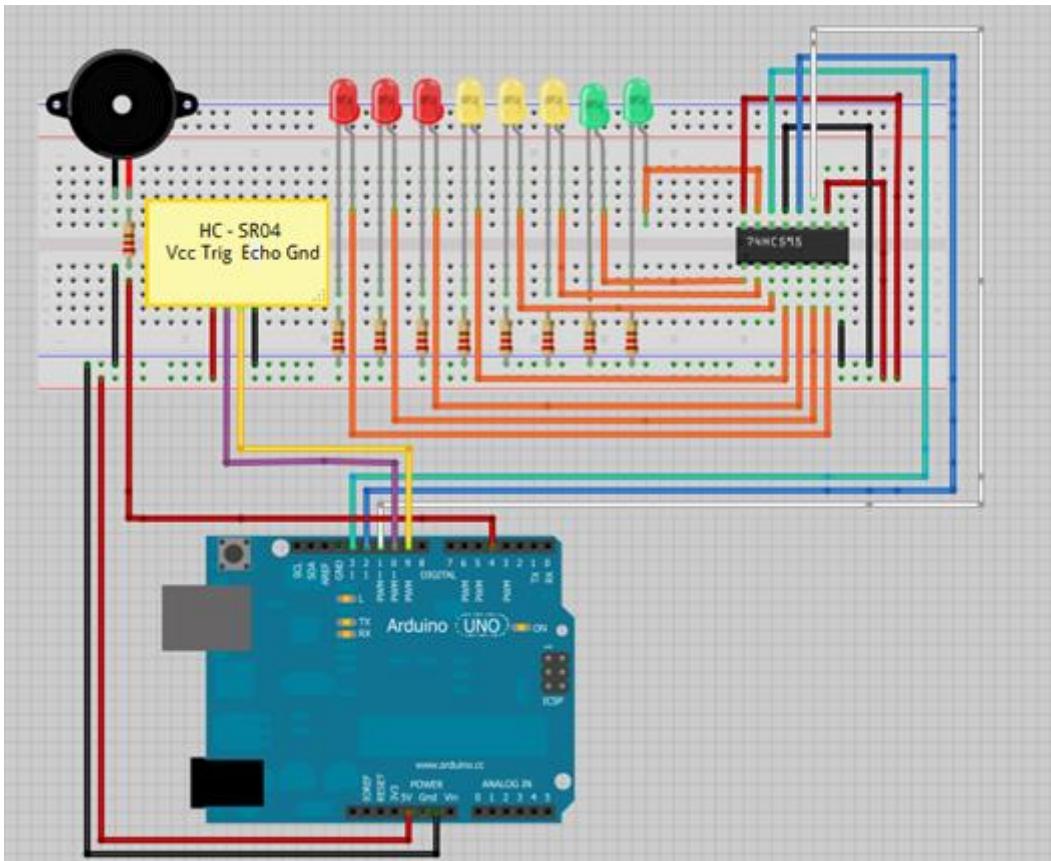
This circuit can work as a parking sensor! This project is great, since it is cheap and really easy to assemble.

Parts Required

- 1x Arduino ([Click to see on Amazon](#))
- 1x 74HC595 8 Bit Shift Register
- 1x Breadboard
- 8x LEDs (for example: 3x red, 3x yellow, 2x green)
- 9x 220 Ohm Resistors
- 1x Buzzer
- 1x Ultrasonic Sensor (for example: HC-SR04)
- Jumper Wires



Schematics



Upload the Code below

[View code on GitHub](#)

```
/*
 * created by Rui Santos, http://randomnerdtutorials.com
 * Ultrasonic Sensor with LED's bar graph and buzzer
 */
int tonePin = 4;      //Tone - Red Jumper
int trigPin = 9;      //Trig - violet Jumper
int echoPin = 10;     //Echo - yellow Jumper
int clockPin = 11;    //IC Pin 11 - white Jumper
int latchPin = 12;    //IC Pin 12 - Blue Jumper
int dataPin = 13;     //IC Pin 14 - Green Jumper

byte possible_patterns[9] = {
B00000000,
B00000001,
B00000011,
B00000111,
```

```

B00001111,
B00011111,
B00111111,
B01111111,
B11111111,
};

int proximity=0;
int duration;
int distance;

void setup() {
    //Serial Port
    Serial.begin (9600);

    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(clockPin, OUTPUT);
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(tonePin, OUTPUT);
}

void loop() {
    digitalWrite(latchPin, LOW);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(1000);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = (duration/2) / 29.1;

    /*if (distance >= 45 || distance <= 0) {
        Serial.println("Out of range");
    }
    else {
        Serial.print(distance);
        Serial.println(" cm");
    }*/
}

proximity=map(distance, 0, 45, 8, 0);
//Serial.println(proximity);

if (proximity <= 0) {
    proximity=0;
}
else if (proximity >= 3 && proximity <= 4) {

```

```

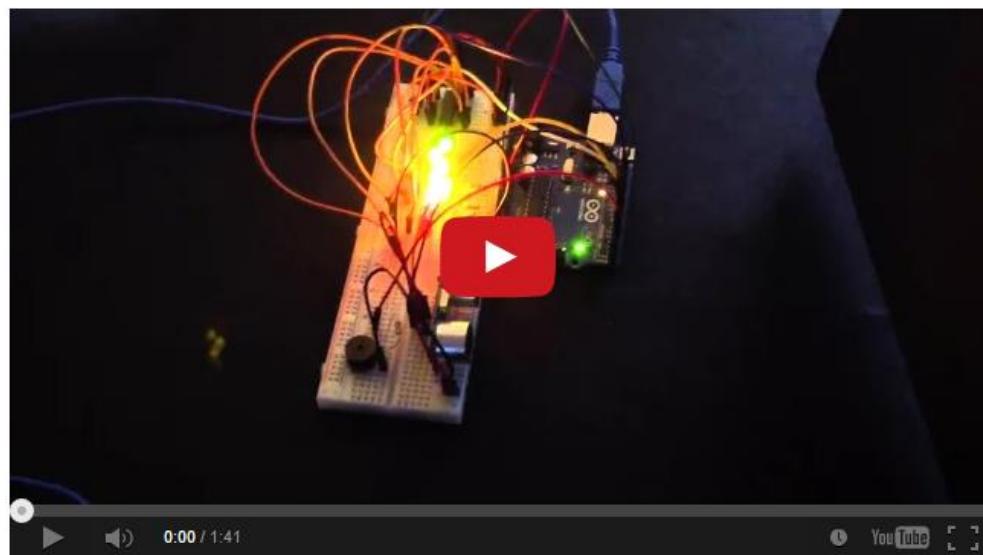
        tone(tonePin, 200000, 200);
    }
    else if (proximity >= 5 && proximity <= 6) {
        tone(tonePin, 5000, 200);
    }
    else if (proximity >= 7 && proximity <= 8) {
        tone(tonePin, 1000, 200);
    }
    shiftOut(dataPin, clockPin, MSBFIRST,
possible_patterns[proximity]);
    digitalWrite(latchPin, HIGH);

}

delay(600);
noTone(tonePin);
}

```

Watch the video demonstration



Watch on YouTube: http://youtu.be/7ZPc_5tL3c

Poor Man's Oscilloscope



View Project on Random Nerd Tutorials

Click [here](#)

Watch on YouTube

Click [here](#)

View Arduino code on GitHub

Click [here](#)

View Processing code on GitHub

Click [here](#)

Introduction

In this project you can create a cheap oscilloscope, I didn't write this code, I've found it on the internet a while back ago and I've decided to share this awesome project. Let's start...

First, download processing software. It's free [Click here to download](#). You don't need to install anything, it runs like the Arduino IDE.

Upload this code to your Arduino

[View Arduino code on GitHub](#)

```
/*
  Complete project details:
  http://randomnerdtutorials.com/arduino-poor-mans-
  oscilloscope/
*/
```

```
#define ANALOG_IN 0

void setup() {
  Serial.begin(9600);
  //Serial.begin(115200);
}

void loop() {
  int val = analogRead(ANALOG_IN);
  Serial.write( 0xff );
  Serial.write( (val >> 8) & 0xff );
  Serial.write( val & 0xff );
```

}

Run this code in Processing IDE

[View Processing code on GitHub](#)

```
/*
 * (c) 2008 Sofian Audry (info@sofianaudry.com)
 *
 * This program is free software: you can redistribute it
 * and/or modify
 * it under the terms of the GNU General Public License as
 * published by
 * the Free Software Foundation, either version 3 of the
 * License, or
 * (at your option) any later version.
 */
import processing.serial.*;

Serial port; // Create object from Serial class
int val; // Data received from the serial port
int[] values;
float zoom;

void setup()
{
    size(1280, 480);
    // Open the port that the board is connected to and use
    // the same speed (9600 bps)
    port = new Serial(this, Serial.list()[0], 9600);
    values = new int[width];
    zoom = 1.0f;
    smooth();
}

int getY(int val) {
    return (int) (height - val / 1023.0f * (height - 1));
}

int getValue() {
    int value = -1;
    while (port.available() >= 3) {
        if (port.read() == 0xff) {
            value = (port.read() << 8) | (port.read());
        }
    }
}
```

```

    }
    return value;
}

void pushValue(int value) {
    for (int i=0; i<width-1; i++)
        values[i] = values[i+1];
    values[width-1] = value;
}

void drawLines() {
    stroke(255);

    int displayWidth = (int) (width / zoom);

    int k = values.length - displayWidth;

    int x0 = 0;
    int y0 = getY(values[k]);
    for (int i=1; i<displayWidth; i++) {
        k++;
        int x1 = (int) (i * (width-1) / (displayWidth-1));
        int y1 = getY(values[k]);
        line(x0, y0, x1, y1);
        x0 = x1;
        y0 = y1;
    }
}

void drawGrid() {
    stroke(255, 0, 0);
    line(0, height/2, width, height/2);
}

void keyReleased() {
    switch (key) {
        case '+':
            zoom *= 2.0f;
            println(zoom);
            if ( (int) (width / zoom) <= 1 )
                zoom /= 2.0f;
            break;
        case '-':
            zoom /= 2.0f;
            if (zoom < 1.0f)
}
}

```

```

        zoom *= 2.0f;
    break;
}
}

void draw()
{
    background(0);
    drawGrid();
    val = getValue();
    if (val != -1) {
        pushValue(val);
    }
    drawLines();
}

```

And then you just need to connect the Arduino Analog Pin 0 to the signal you want to read.

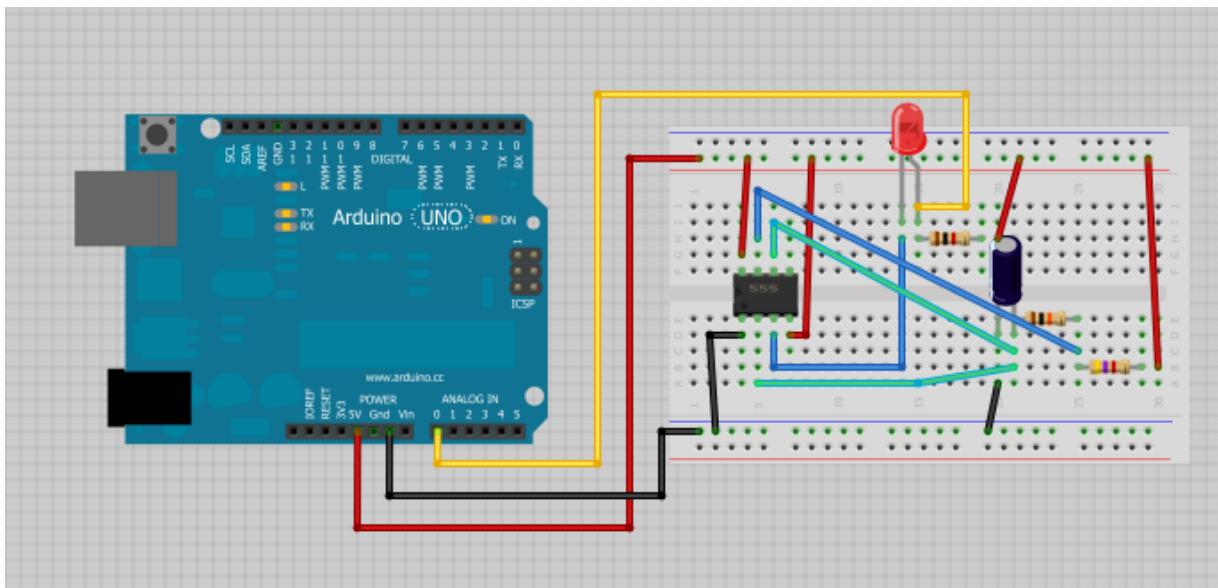
And It's done!

Parts required

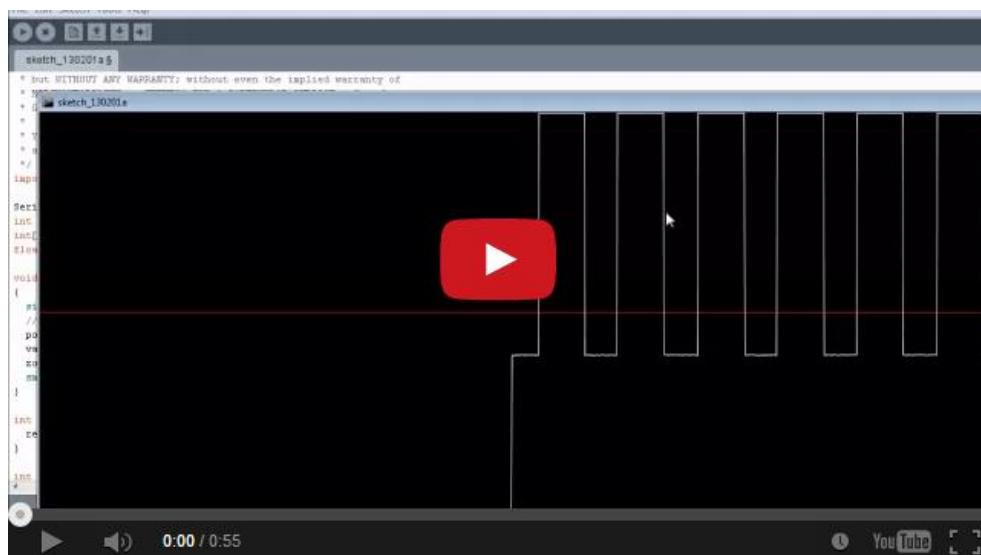
- 1x Arduino ([Click to see on Amazon](#))
- 1x Breadboard
- 1x LED
- 1x 10k resistor
- 1x 4.7k resistor
- 1x 1k resistor
- 1x 100nF electrolytic capacitor
- Jumper cables

Schematics

This is the circuit I'll be measuring, it's a simple 555 timer circuit that flashes an LED.

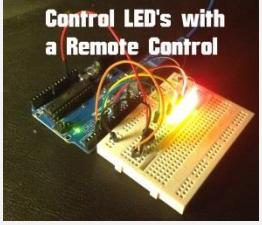


Watch the video demonstration



Watch on YouTube: <http://youtu.be/ZMhQInlBhPA>

Control LEDs with IR Remote Control

 Control LED's with a Remote Control	View Project on Random Nerd Tutorials Click here	Watch on YouTube Click here
	View library code on GitHub Click here	
	View Arduino code on GitHub Click here	

Introduction

In this project you will learn how you can control any device using a remote control and your Arduino.

First you need to [go to this page](#) and download the IR library. Simply follow the read me file to install. I think that any IR remote control can do this, I'll be using a Philips Universal remote control.

Parts required

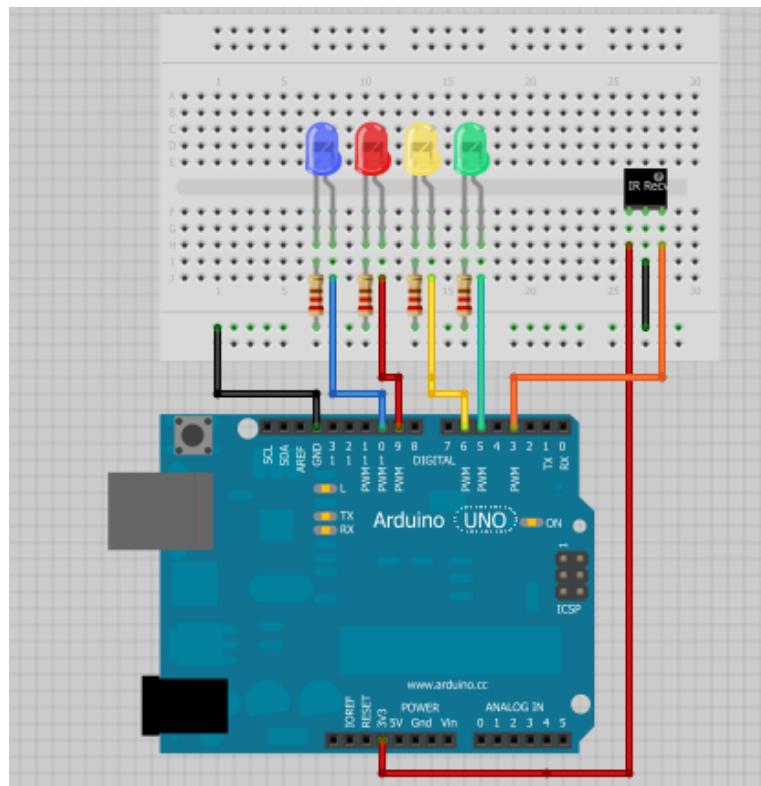
- 1x Arduino ([Click to see on Amazon](#))
- 1x Breadboard
- 1x Remote control
- 1x IR receiver (I'll be using TSOP4838)
- 4x LED's
- 4x 220ohm resistors
- Jumper cables

Infrared receiver pins

- First pin: Vout, outputs HIGH when no signal is present and LOW when a mark is received
- Second pin: GND
- Third pin: Vcc



Schematics



Open your Arduino IDE.

Got to **File > examples > IRremote > IRrecvDemo**. You need to upload the sketch to your arduino, open the serial monitor and start using your remote control and see which values the arduino is receiving.

After a while I've wrote down which values appear in the serial monitor when you press the volume up key or any other key, and write it down for all the keys you want to use. And they were:

- Power: E240
- Forward: E250
- Reverse: E248
- Volume+: E244
- Volume-: E254
- Mute: E24C7



You will need to convert those hexadecimal numbers to decimal, [you can use this tool for that](#). Then just need to change your remote control values from my arduino code.

Upload the code below

Visit the link below, click to “Download ZIP”, unzip the library and move the library to your Arduino libraries folder.

<https://github.com/RuiSantosdotme/Arduino-IRremote>

[View code on GitHub](#)

```
/*
 * Control LED's with a remote control
 * Modified by Rui Santos, http://randomnerdtutorials.com
 * based on IRremote Library - Copyright Ken Shirriff
 */

#include <IRremote.h>

int IR_Recv = 3;      //IR Receiver Pin 3
int g_ledPin = 5;    //green LED pin 5
int y_ledPin = 6;    //yellow LED pin 6
int r_ledPin = 9;    //red LED pin 9
int b_ledPin = 10;   //blue LED pin 10
int ledPins[] = {5, 6, 9, 10}; //array with all the LED's pins
int ledStates[] = {0, 0, 0, 0}; //this means the LED's states at first is 0 = LOW
int i=0; //LED index for the arrays

IRrecv irrecv(IR_Recv);
decode_results results;

//variables to make the LED blink when selected
int ledState = LOW;           // ledState to turn the LED on or off
long previousMillis = 0;       // stores last time LED was updated
long interval = 1000;         // interval at which to blink (milliseconds)

void setup() {
```

```

Serial.begin(9600); //starts serial communication
irrecv.enableIRIn(); // Starts the receiver
pinMode(g_ledPin, OUTPUT); // sets the digital pin
as output
pinMode(y_ledPin, OUTPUT); // sets the digital pin
as output
pinMode(r_ledPin, OUTPUT); // sets the digital pin
as output
pinMode(b_ledPin, OUTPUT); // sets the digital pin
as output
}

void loop() {
    //decodes the infrared input
    if (irrecv.decode(&results)){
        long int decCode = results.value;
        Serial.println(decCode);
        //switch case to use the selected remote control button
        switch (results.value){
            case 57936: //when you press the Forward button
                //this if/else statement makes sure that LED is ON
                or OFF before move to the next LED
                if(ledStates[i]==0)
                    digitalWrite(ledPins[i], LOW);
                else
                    digitalWrite(ledPins[i], HIGH);
                Serial.println("Next LED");
                //makes sure that when we reach the last LED it
                goes to the first LED again
                if(i>=3)
                    i=-1;
                i+=1;
                break;

            case 57928: //when you press the Reverse button
                //this if/else statement makes sure that LED is ON
                or OFF before move to the previous LED
                if(ledStates[i]==0)
                    digitalWrite(ledPins[i], LOW);
                else
                    digitalWrite(ledPins[i], HIGH);
                Serial.println("Previous LED");
                //makes sure that when we reach the first LED it
                goes to the last LED
                if(i<=0)
}
}

```

```

        i=4;
        i-=1;
        break;

case 57932: //when you press the Mute button
    if(ledStates[i]==0){ //if the LED is off, It will
turn on
        Serial.println("Turns ON the LED Selected");
        digitalWrite(ledPins[i], HIGH); //sets the LED
on
        ledStates[i]=1; //updates the
LED state
    }
    else{
        Serial.println("Turns OFF the LED Selected");
//else: the LED is on, It will turn off
        digitalWrite(ledPins[i], LOW); //sets the LED
off
        ledStates[i]=0; //updates the
LED state
    }
    break;

case 57920: //when you press the Power button
    Serial.println("Turns OFF all the LED's");
    digitalWrite(g_ledPin, LOW); // sets the green
LED off
    ledStates[0] =0; // updates the LED
state
    digitalWrite(y_ledPin, LOW); // sets the yellow
LED off
    ledStates[1] =0; // updates the LED
state
    digitalWrite(r_ledPin, LOW); // sets the red LED
off
    ledStates[2] =0; // updates the LED
state
    digitalWrite(b_ledPin, LOW); // sets the blue
LED off
    ledStates[3] =0; // updates the LED
state
    break;

default:
    Serial.println("Waiting");

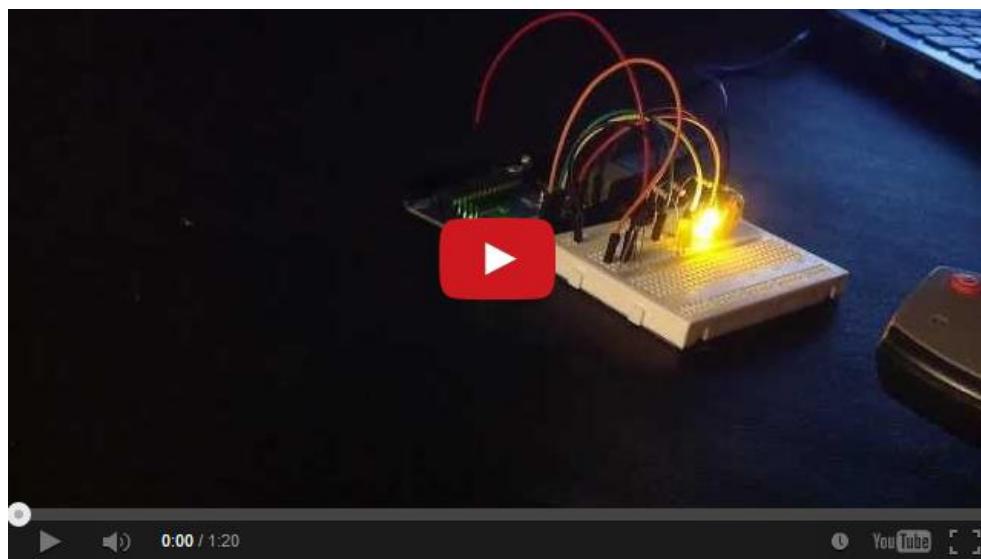
```

```

    }
    irrecv.resume(); // Receives the next value from the
button you press
}
//this if statement makes the LED blink if it's selected
and off
if(ledStates[i]==0) {
    unsigned long currentMillis = millis();
    if(currentMillis - previousMillis > interval) {
        // save the last time you blinked the LED
        previousMillis = currentMillis;
        // if the LED is off turn it on and vice-versa:
        if (ledState == LOW)
            ledState = HIGH;
        else
            ledState = LOW;
        // set the LED with the ledState of the variable:
        digitalWrite(ledPins[i], ledState);
    }
}
}

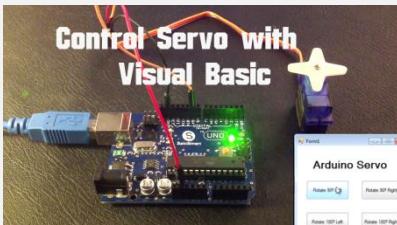
```

Watch the video demonstration



Watch on YouTube: <http://youtu.be/IQ7K8Jp3Jns>

Control Servo with Visual Basic

	View Project on Random Nerd Tutorials	Click here
	Watch on YouTube	Click here
	View arduino code on GitHub	Click here
	View Visual Basic Code	Click here

Introduction

In this project, you will learn how you can control a servo with Visual basic. Most people usually use the Serial Monitor of the Arduino IDE to communicate with the Arduino, but today your will use a visual basic program that I've created. Basically in the VB program we have 4 buttons that will interact with the Arduino when we press them.

I'll be showing program in Visual Basic that allows the user to rotate a servo attached to the Arduino. You need to make 3 connections from the servo to your arduino:

- Red: 3.3V (depends on your Servo motor)
- Brown: Ground
- Orange: Digital Pin 9

Upload the Arduino code below

[View code on GitHub](#)

```
/*
 * Control a servo motor with Visual Basic
 * Created by Rui Santos, http://randomnerdtutorials.com
 */

#include <Servo.h>

Servo myservo; // create servo object to control a servo

void setup()
```

```

{
    myservo.attach(9); // attaches the servo on pin 9 to
the servo object
    Serial.begin(9600); //begins serial communication
}

void loop()
{
    int pos;
    if (Serial.available()) {
        delay(100);
        while (Serial.available()>0) {
            pos=Serial.read(); //reads the value sent from
Visual Basic
            if (pos=='0')
                myservo.write(90); //rotates the servo 90
degrees (Left)
            else if (pos=='1')
                myservo.write(-90); //rotates the servo 90
degrees (right)
            else if (pos=='2')
                myservo.write(180); //rotates the servo 180
degrees (Left)
            else if (pos=='3')
                myservo.write(-180); //rotates the servo 180
degrees (right)
        }
    }
}

```

Run the Visual Basic script

Visit the link below to download the Visual Basic script:

[Click here to download the Visual Basic Script](#)

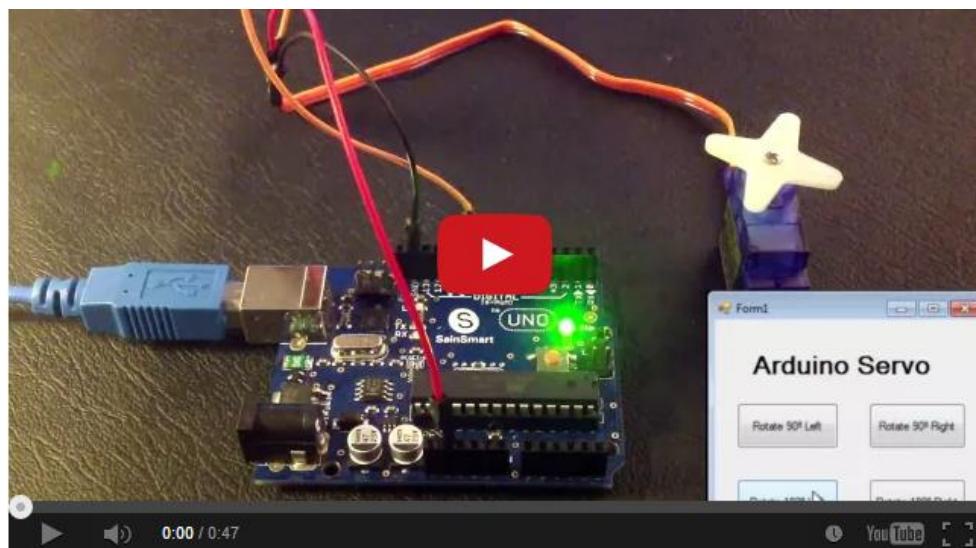
Note

When you're using the Visual Basic Program the Serial monitor on the Arduino IDE must be closed

You need to match the com port on your Visual Basic Code to the right com port you're Arduino is using

I don't know why, but YouTube cropped my video, the last two buttons are "Rotate 180° Left" and "Rotate 180° right".

Watch the video demonstration



Watch on YouTube: http://youtu.be/rza_ujeK398

Control DC Motor via Bluetooth



View Project on Random Nerd Tutorials

Click [here](#)

Watch on YouTube

Click [here](#)

View arduino code on GitHub

Click [here](#)

Introduction

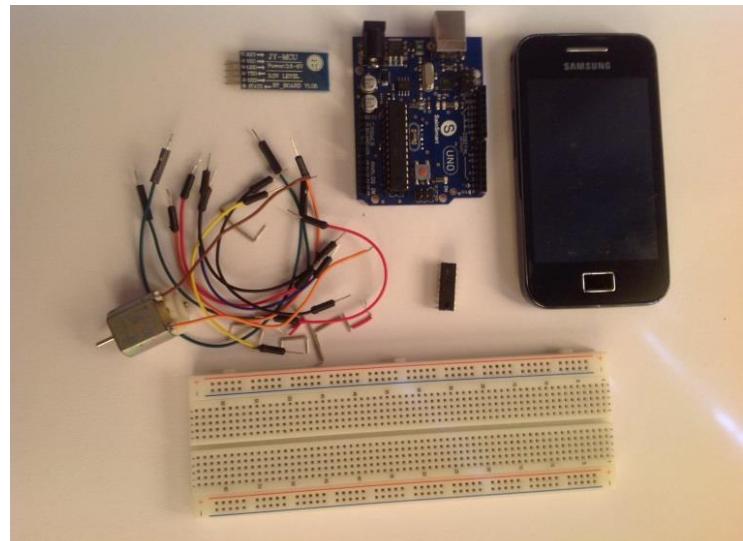
In this project we will control a DC motor with a smartphone via bluetooth. This project is great to learn more about:

- DC motors
- Interfacing Arduino with your smartphone
- Bluetooth
- L293D IC

If you don't have the L293 IC you can make the same circuit using the H bridge, anyway I really recommend you to read more about that and the IC datasheet. There's plenty of tutorials about that.

Parts required

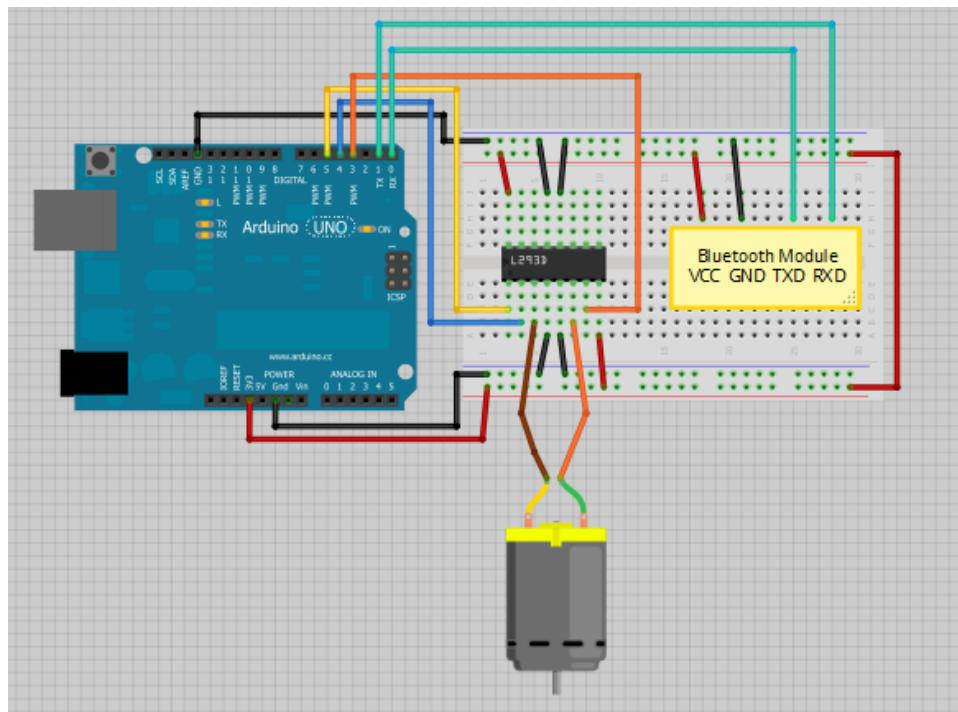
- 1x Arduino ([Click to see on Amazon](#))
- 1x Bluetooth Module
(for example: HC-05 or 06 - [Click to see on Amazon](#))
- 1x Smartphone (any Android will work)
- BlueTerm application
- 1x L293D IC
- 1x DC motor
- 1x Breadboard
- Jumper Cables



Schematics

Note

You can only connect TX and RX cables to your Arduino, after you upload the code.



Two common mistakes

1. You need to remove the RX and TX cables when you're uploading the sketch to your Arduino.
2. Sometimes people connect the TX from the bluetooth module to the TX of the Arduino... that's wrong and it won't work. Make sure you connect it properly, the TX into RX and the RX into the TX.

Note

If the HC-05 Bluetooth Module asks for a password, it's '1234'.

Upload the code below

Make sure you remove the wires from RX and TX otherwise the code won't upload properly!

[View code on GitHub](#)

```
/*
 * Control DC motor with Smartphone via bluetooth
 * created by Rui Santos, http://randomnerdtutorials.com
 */

int motorPin1 = 3; // pin 2 on L293D IC
int motorPin2 = 4; // pin 7 on L293D IC
int enablePin = 5; // pin 1 on L293D IC
int state;
int flag=0; //makes sure that the serial only prints once the state

void setup() {
    // sets the pins as outputs:
    pinMode(motorPin1, OUTPUT);
    pinMode(motorPin2, OUTPUT);
    pinMode(enablePin, OUTPUT);
    // sets enablePin high so that motor can turn on:
    digitalWrite(enablePin, HIGH);
    // initialize serial communication at 9600 bits per second:
    Serial.begin(9600);
}

void loop() {
    //if some date is sent, reads it and saves in state
    if(Serial.available() > 0) {
        state = Serial.read();
        flag=0;
    }
    // if the state is '0' the DC motor will turn off
    if (state == '0') {
        digitalWrite(motorPin1, LOW); // set pin 2 on L293D low
        digitalWrite(motorPin2, LOW); // set pin 7 on L293D low
        if(flag == 0) {
            Serial.println("Motor: off");
        }
    }
}
```

```

        flag=1;
    }
}

// if the state is '1' the motor will turn right
else if (state == '1') {
    digitalWrite(motorPin1, LOW); // set pin 2 on
L293D low
    digitalWrite(motorPin2, HIGH); // set pin 7 on
L293D high
    if(flag == 0) {
        Serial.println("Motor: right");
        flag=1;
    }
}
// if the state is '2' the motor will turn left
else if (state == '2') {
    digitalWrite(motorPin1, HIGH); // set pin 2 on
L293D high
    digitalWrite(motorPin2, LOW); // set pin 7 on
L293D low
    if(flag == 0) {
        Serial.println("Motor: left");
        flag=1;
    }
}
}

```

For the android communication with our bluetooth module I've used the BlueTerm app, It's completely free, so you just need to go to "Play store" and download it. Then you just need to connect your smartphone with the bluetooth module. Remember to remove the TX and RX cables. (you can see in youtube video below how that's done).

I've only set 3 commands to control the DC motor:

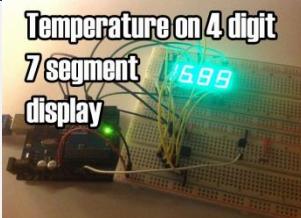
- '0' - Turns off the DC motor
- '1' - DC motor rotates to right
- '2' - DC motor rotates to left

Watch the video demonstration



Watch on YouTube: <http://youtu.be/nXymP6txD4>

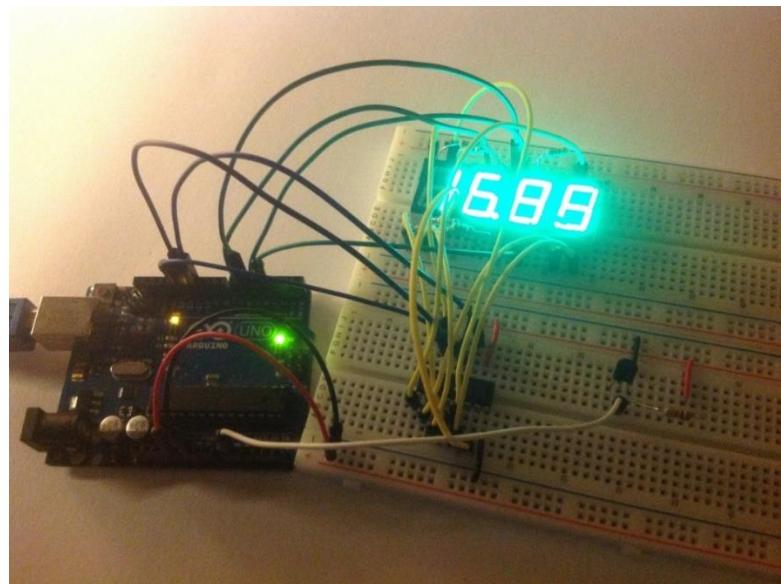
Temperature Displayed on 4 Digit 7 segment (common anode)

 Temperature on 4 digit 7 segment display	View Project on Random Nerd Tutorials	Click here
	Watch on YouTube	Click here
	View arduino code on GitHub	Click here

Introduction

In this project I'll display the temperature in a 4 digit 7 segment display (common anode).

The sensor is the cheapest you can find so actually the temperature changes pretty easily which makes the display to show always different temperatures. But the idea is to apply this code to other projects with 7 segment displays that I might do later. [You can also read more about 7 segment displays in this post.](#)



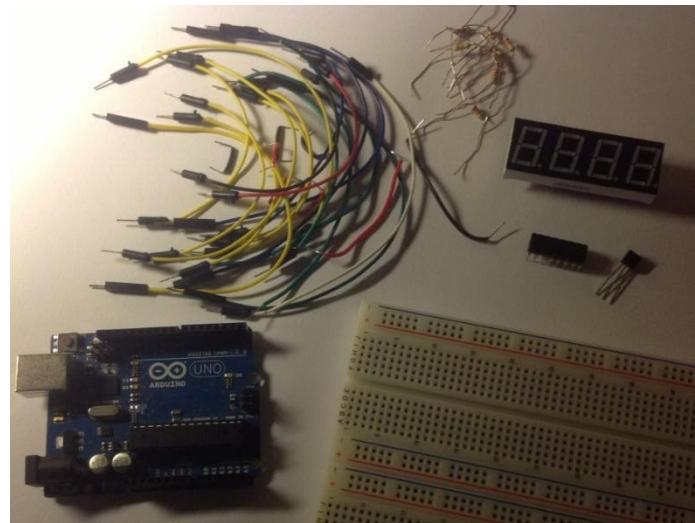
This project is great to learn more about:

- Reading sensors (in this case temperature)
- 7 segment displays (4 digit 7 segment displays)
- 8 bit Shift Registers (74HC595)

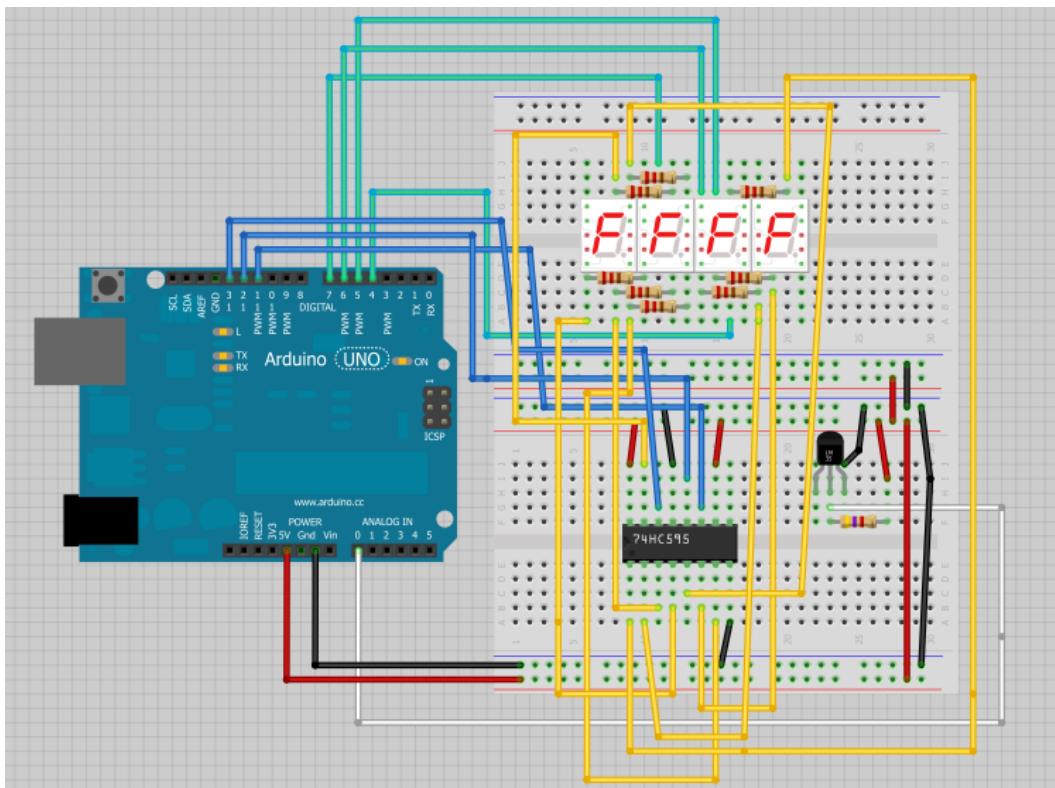
- Practice wiring

Parts required

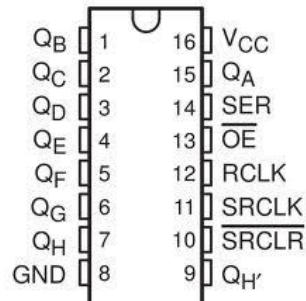
- 1x Arduino ([Click to see on Amazon](#))
- 1x Temperature Sensor (I'm using the LM335Z)
- 1x 4 Digit 7 Segment Display (common anode)
- 1x 74HC595 8 Bit Shift Register
- 8x 220 Ohm Resistors
- 1x 4700 ohm Resistor
- 1x Breadboard (or two)
- Jumper Cables



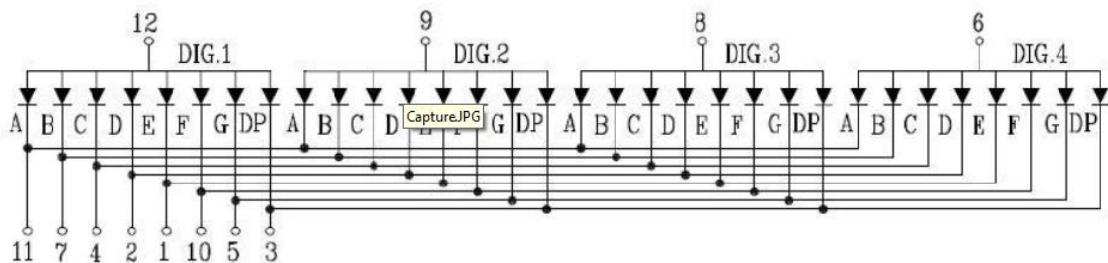
Schematics



This can also help you wiring because the Schematics I've made using fritizing turned out a bit confusing. This is the 7 segment display internal circuit diagram



INTERNAL CIRCUIT DIAGRAM



Basically the pin 11 connects to the QA, the pin 7 to the QB and so on...

Upload the code below

[View code on GitHub](#)

```
/*
 * Temperature Sensor Displayed on 4 Digit 7 segment
common anode
 * Created by Rui Santos, http://randomnerdtutorials.com
 */

const int digitPins[4] = {
  4,5,6,7};                                //4 common anode pins of the
display
const int clockPin = 11;                     //74HC595 Pin 11
const int latchPin = 12;                      //74HC595 Pin 12
const int dataPin = 13;                      //74HC595 Pin 14
```

```

const int tempPin = A0;           //temperature sensor pin
const byte digit[10] =          //seven segment digits in bits
{
    B00111111, //0
    B00000110, //1
    B01011011, //2
    B01001111, //3
    B01100110, //4
    B01101101, //5
    B01111101, //6
    B00000111, //7
    B01111111, //8
    B01101111 //9
};
int digitBuffer[4] = {
    0};
int digitScan = 0, flag=0, soft_scaler = 0;
;
float tempK, tempC, tempF, temp;

void setup() {
    for(int i=0; i<4; i++)
    {
        pinMode(digitPins[i], OUTPUT);
    }
    pinMode(tempPin, INPUT);
    pinMode(latchPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(tempPin, INPUT);
}

//writes the temperature on display
void updateDisp() {
    for(byte j=0; j<4; j++)
        digitalWrite(digitPins[j], LOW);

    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, B11111111);
    digitalWrite(latchPin, HIGH);

    delayMicroseconds(100);
    digitalWrite(digitPins[digitScan], HIGH);

    digitalWrite(latchPin, LOW);
}

```

```

if (digitScan==2)
    shiftOut(dataPin, clockPin, MSBFIRST,
~(digit[digitBuffer[digitScan]] | B10000000)); //print the
decimal point on the 3rd digit
else
    shiftOut(dataPin, clockPin, MSBFIRST,
~digit[digitBuffer[digitScan]]);

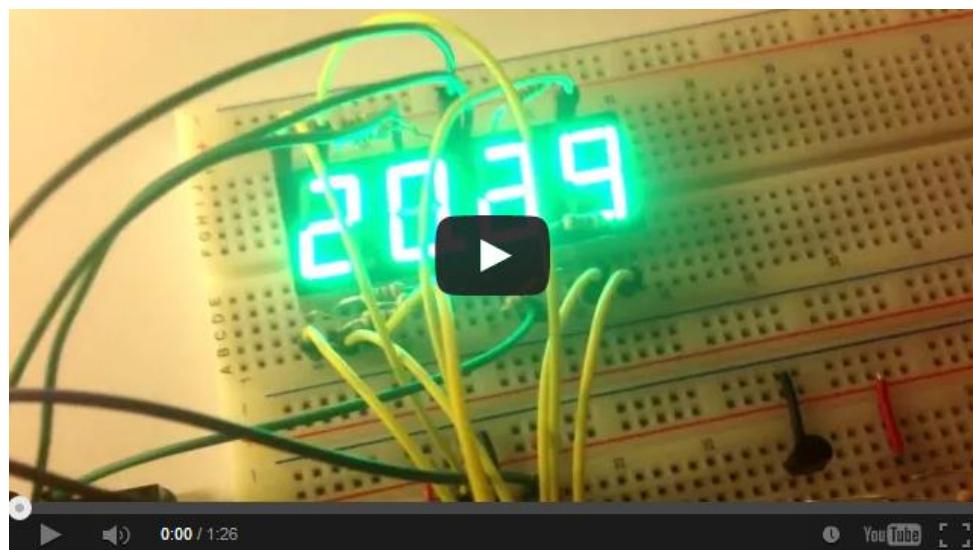
    digitalWrite(latchPin, HIGH);
    digitScan++;
    if (digitScan>3) digitScan=0;
}

void loop() {
    tempK = (((analogRead(tempPin)/ 1023.0) * 5.0) * 100.0);
    //Converts Kelvin to Celsius minus 2.5 degrees error
    tempC = tempK - 273.0;
    tempF = ((tempK - 2.5) * 9 / 5) - 459.67;
    //Celsius temperature display
    tempC = int(tempC*100);
    digitBuffer[3] = int(tempC)/1000;
    digitBuffer[2] = (int(tempC)%1000)/100;
    digitBuffer[1] = (int(tempC)%100)/10;
    digitBuffer[0] = (int(tempC)%100)%10;
    updateDisp();
    delay(2);

/*
//Fahrenheit temperature display
tempF = int(tempF*100);
digitBuffer[3] = int(tempF)/1000;
digitBuffer[2] = (int(tempF)%1000)/100;
digitBuffer[1] = (int(tempF)%100)/10;
digitBuffer[0] = (int(tempF)%100)%10;
updateDisp();
delay(2);
*/
}

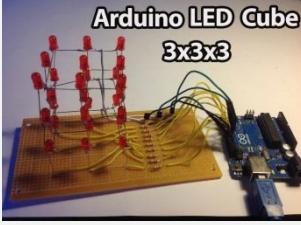
```

Watch the video demonstration



Watch on YouTube: <http://youtu.be/qYK6By37-Oo>

LED Cube 3x3x3 with Arduino

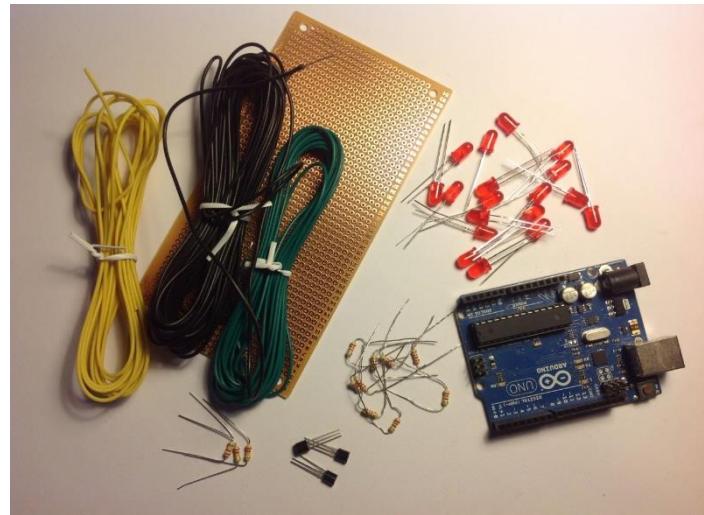
	View Project on Random Nerd Tutorials	Click here
	Watch on YouTube (full tutorial)	Click here
	Watch on YouTube (demo)	Click here
	Direct download link	Click here

Introduction

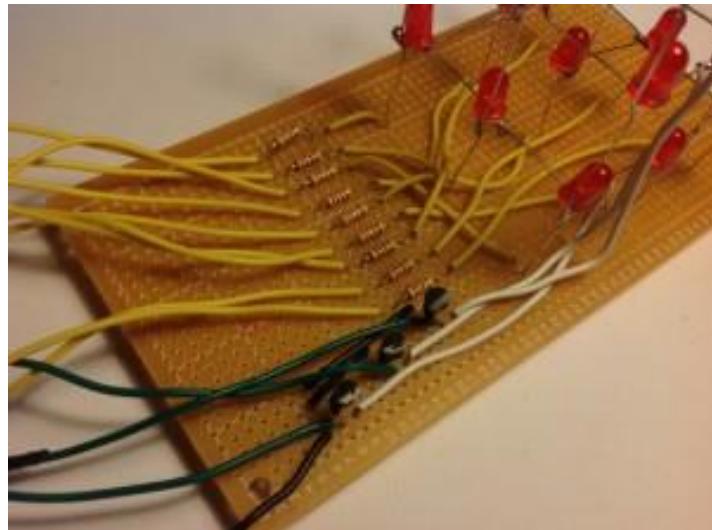
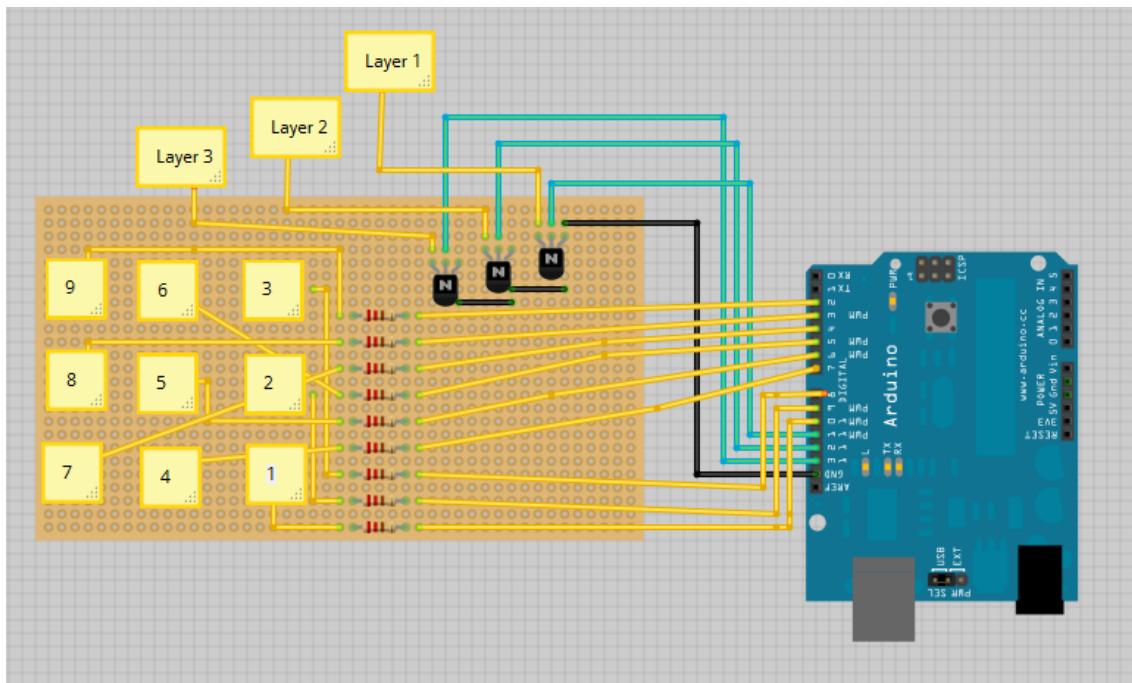
In this tutorial I'll show how you can create your own LED Cube 3x3x3. I'm sure you already saw some similar projects to this one but you never took action and made your own. Now it's time to make your own!

Parts Required

- 1x Arduino Board
- 27x LED's
- 1x Stripboard
- 3x 22k ohm Resistors
- 9x 220 ohm Resistors
- 3x NPN Transistors (for example: 2N2222, BC547, 2N3904)
- Wire



Schematics

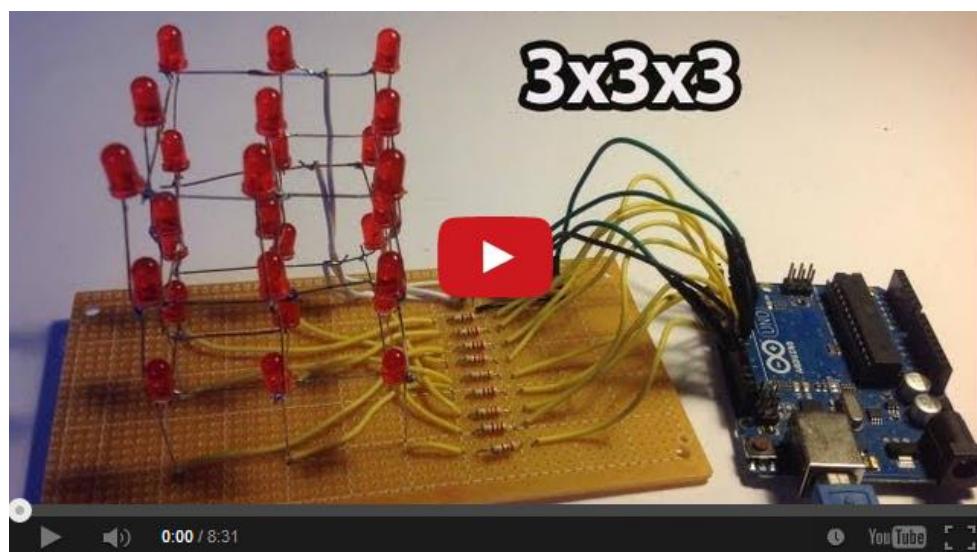


Download the Arduino library below

[Click here to Download the library](#)

Be sure to download the library and unzip it to the Arduino libraries folder. If everything is correct you should find an example in the Arduino software under **File > Examples > LedCube > ledcube**.

Watch this video tutorial

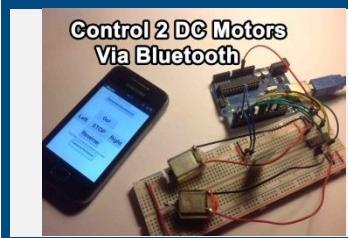


Watch full tutorial on Youtube: <http://youtu.be/GLx6aA75CZY>



Watch this demo of the final product: <http://youtu.be/vx-Dq9GDF0>

Control 2 DC Motors via Bluetooth



View Project on Random Nerd Tutorials

Click [here](#)

Watch on YouTube

Click [here](#)

View arduino code

Click [here](#)

Update, 23-12-2013

appinventor.mit.edu changed their whole website... So the source file that previously you could use to edit on their website, only works with AppInventor version 1.0 or also called classic that you can see here <http://appinventor.mit.edu/explore/classic.html> Click the button: "**Invent your own Apps now**". This project still works just fine with my app and with my Arduino code. But you can only edit the source code on Appinventor classic version.

Introduction

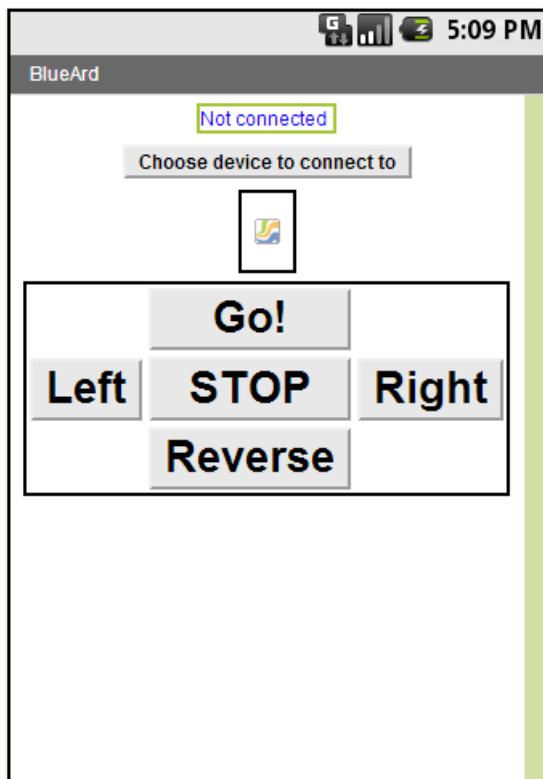
In this tutorial I'll show you how you can control 2 DC motors via bluetooth with my brand new Android app. It's called "BlueArd" and It's the 1.0 version. I want to upgrade my app later and add more features.

This app was created with [MIT App Inventor](#). It's a great place to start with android development.

If you remember my previous tutorial ([Click here to see that project](#)) where I was controlling 1 DC motor, I was using an app called "BlueTerm". That app did the job but I didn't like the design that much. So I've decided to make my own.

It's my very first app and it's working for me but I'm not sure if this will work for everyone. If you want to make some improvements to my app, feel free to do that.

Let's take a look to "BlueArd":



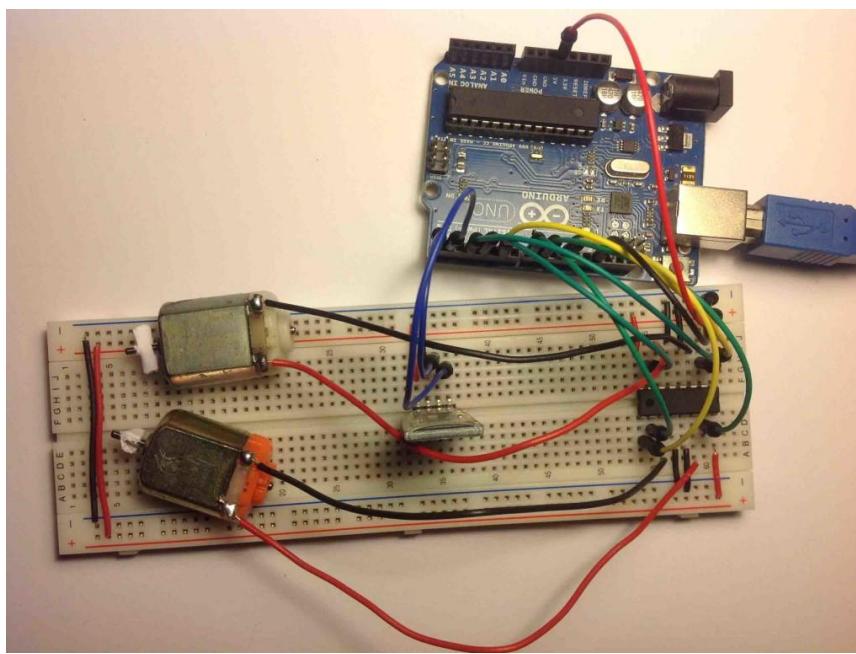
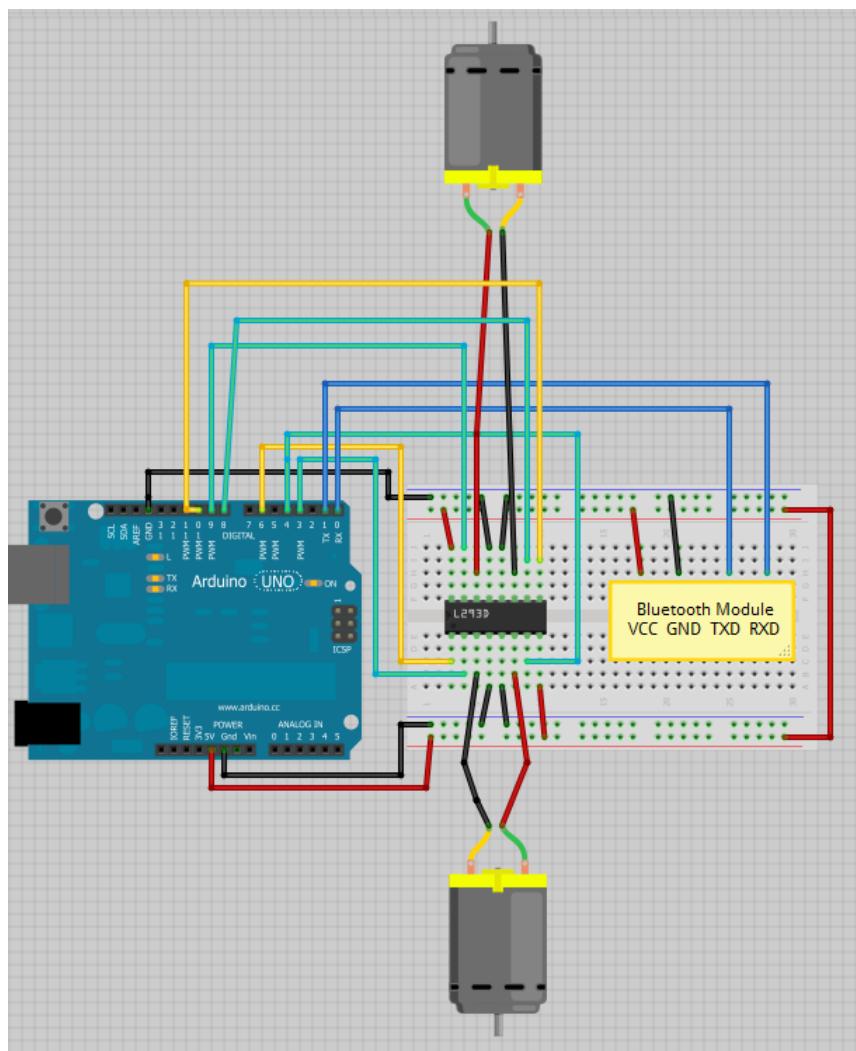
This app is perfect to control anything via bluetooth. You can edit this app for your needs (you have the source code below... just click share to unlock the source code). But my idea is to create a robot later that will be controlled via bluetooth through this app.

Parts Required

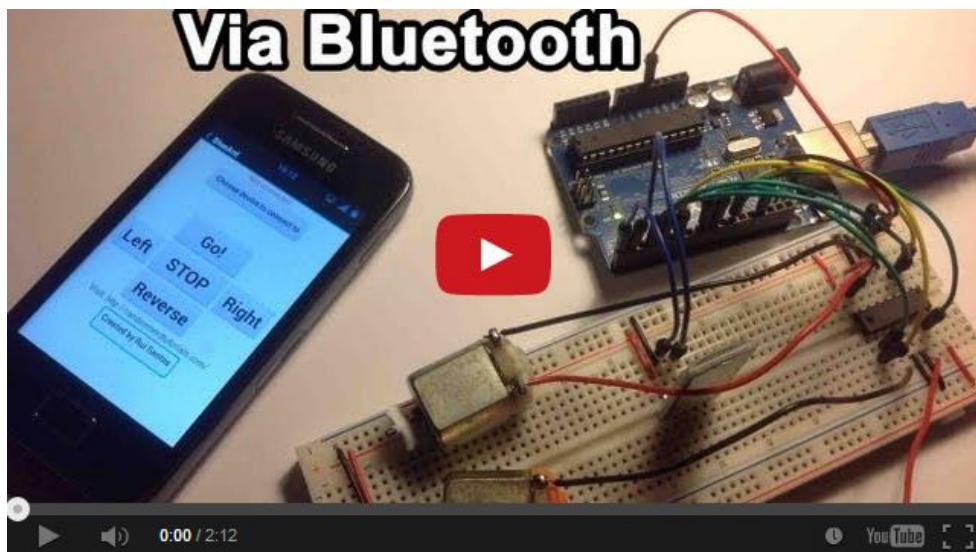
- 1x Arduino
- 1x Bluetooth Module (for example: HC-05)
- 1x Smartphone
- BlueArd Application
- 1x L293D IC
- x DC motor
- 1x Breadboard
- Jumper Cables



Schematics



Watch this video tutorial



Watch on YouTube:<http://youtu.be/3RsfA6ngLdQ>

Download all my source code below

- Arduino Sketch
- BlueArd.apk
- BlueArd Source files (for editing purpose)

[Click here to download all the files.](#)

Notes and Tips

1. You need to remove the RX and TX cables when you're uploading the sketch to your Arduino.
2. Sometimes people connect the TX from the bluetooth module to the TX of the Arduino... that's wrong and it won't work. Make sure you connect it properly, the TX into RX and the RX into the TX.
3. If the HC-05 Bluetooth Module asks for a password, It's '1234'.
4. Before Testing my "BlueArd" app, test if you've made all the connections correctly. How you can do that? Simply enter numbers ('1', '2', '3', '4' and '5') into your serial monitor and your DC motors should be working properly...

How to Use App Inventor with Arduino

	View Project on Random Nerd Tutorials	Click here
	Watch on YouTube	Click here
	View arduino code	Click here

Update, 23-12-2013

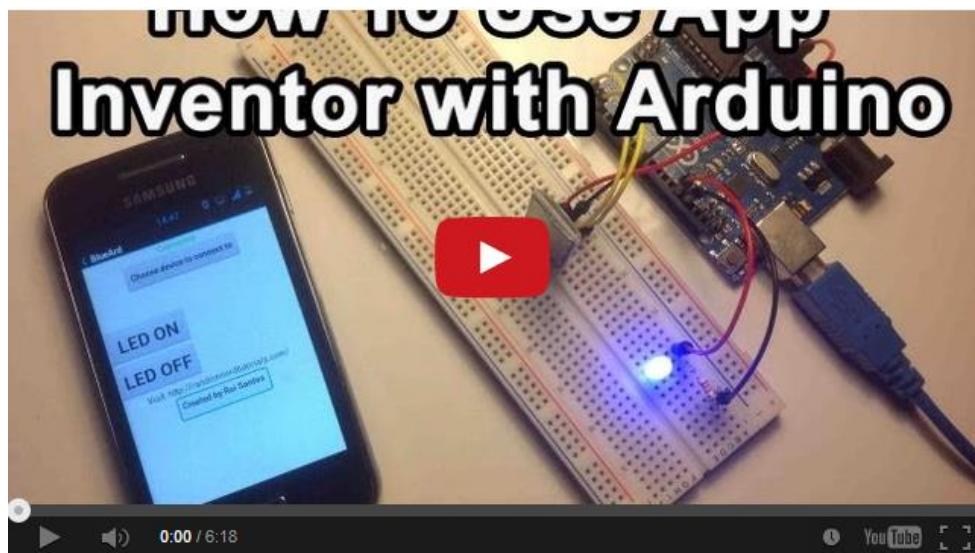
appinventor.mit.edu changed their whole website... So the source file that previously you could use to edit on their website, only works with AppInventor version 1.0 or also called classic that you can see here <http://appinventor.mit.edu/explore/classic.html> Click the button: "**Invent your own Apps now**". This project still works just fine with my app and with my Arduino code. But you can only edit the source code on Appinventor classic version.

Introduction

After so many requests, I've decided to create this quick tutorial! I hope you enjoy.

This will help you understand how App Inventor works and how it can interact with your arduino via bluetooth.

Watch this video tutorial



Watch on YouTube: <http://youtu.be/cVj0jqOIGXo>

Download all my source code below

- Arduino Sketch
- BlueLED.apk
- BlueLED Source files (for editing purpose)

[Click here to download all the files.](#)

Note

If you want to edit my app this is what you need to do: After you download this folder make sure you unzip it. Only upload the BlueLED.zip into App Inventor.

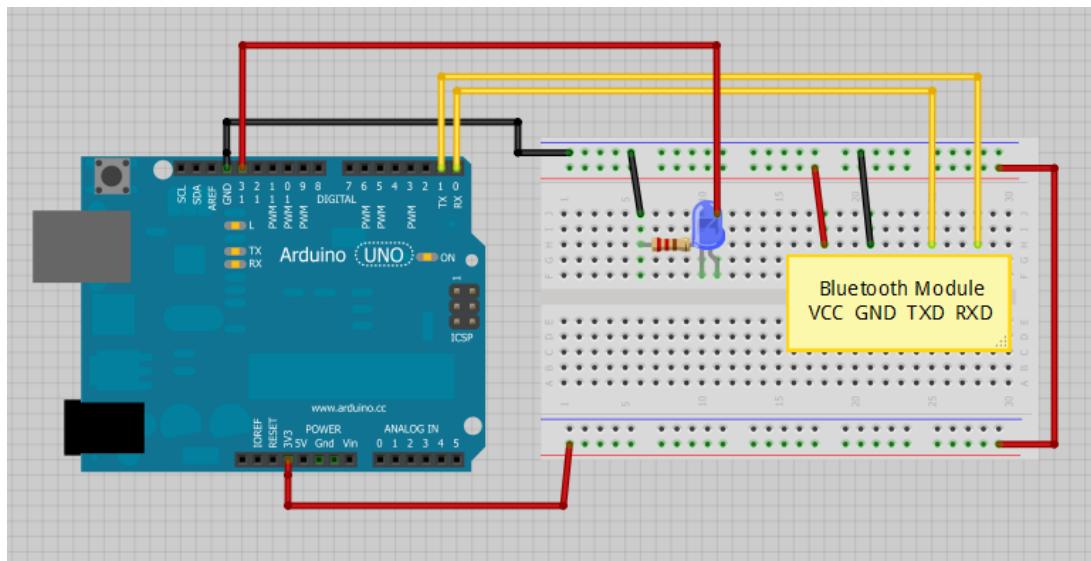
Parts Required

- 1x Arduino Uno
- 1x Bluetooth Module (for example: HC-05, [click here to read my review here about this Bluetooth module](#))
- 1x Smartphone (any Android will work, I've only tested with Samsung Galaxy Ace)
- Android Application (you can download it in the next step)
- 1x 220Ohm Resistor

- 1x LED
- 1x Breadboard
- Jumper Cables



Schematics



Tips

1. You need to remove the RX and TX cables when you're uploading the sketch to your Arduino.

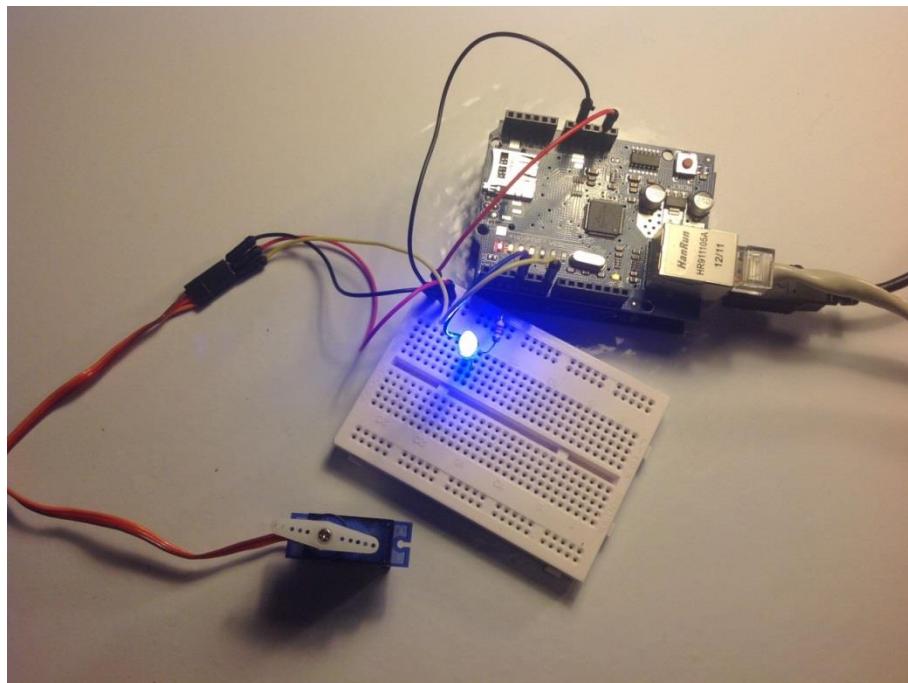
2. Sometimes people connect the TX from the bluetooth module to the TX of the Arduino... that's wrong and it won't work. Make sure you connect it properly, the TX into RX and the RX into the TX.
3. If the HC-05 Bluetooth Module asks for a password, It's '1234'.
4. Before Testing my "BlueLED" app, test if you've made all the connections correctly. How you can do that? Simply enter numbers ('1', '0') into your serial monitor and your LED should be turning on and off.

Webserver with an Arduino + Ethernet Shield

	View Project on Random Nerd Tutorials	Click here
	Watch on YouTube	Click here
	View code on GitHub	Click here

Introduction

This project is all about using an Arduino with an Ethernet shield. I'll be controlling one LED and a servo, but you can apply this method to control any electronic device you want. (such as DC motors, buzzers, relays, stepper motors, etc..)

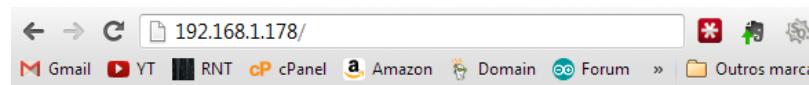


How it works

The code provided when uploaded and connected to the internet it creates a webserver in your LAN and you simply use the IP to access that webserver through your browser. After that it shows a webpage similar to that one below. When you press the button "Turn On LED"

your url will change to: "http://192.168.1.178/?button1on" the arduino will read that information and It turns the LED On.

By default the IP is "192.168.1.178". That also can be found on the arduino code provided.



Random Nerd Tutorials Project

Arduino with Ethernet Shield

[Turn On LED](#) [Turn Off LED](#)

[Rotate Left](#) [Rotate Right](#)

Created by Rui Santos. Visit <http://randomnerdtutorials.com> for more projects!

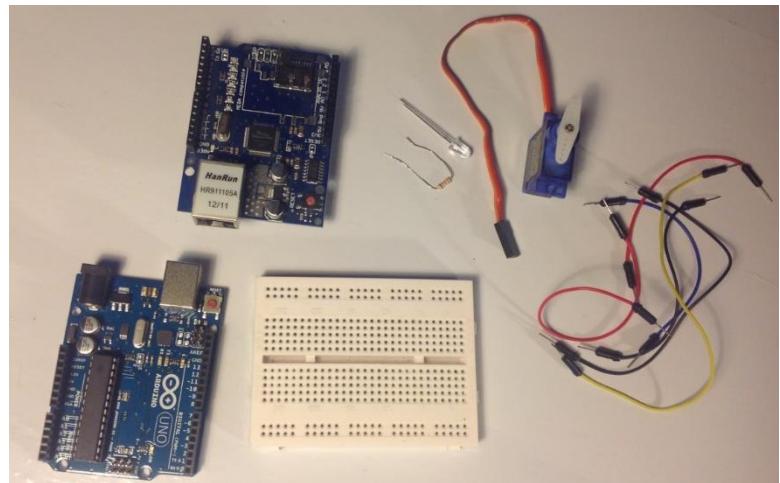
Watch the video demonstration



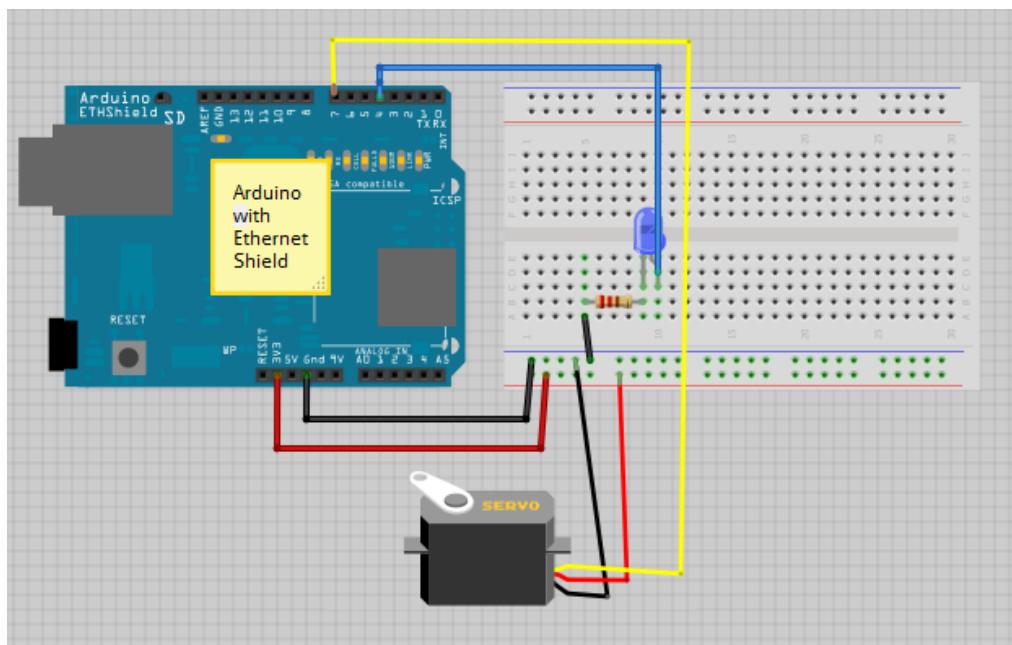
Watch on YouTube: <http://youtu.be/2J4cx2gA7DQ>

Parts Required

- 1x Arduino Uno
- 1x Ethernet Shield
- 1x 220 Ohm Resistor
- 1x LED
- 1x Micro Servo Motor
- 1x Breadboard
- Jumper Cables



Schematics



Upload the code below

[View code on GitHub](#)

```
/*
Created by Rui Santos
```

Visit: <http://randomnerdtutorials.com> for more arduino projects

Arduino with Ethernet Shield
*/

```
#include <SPI.h>
#include <Ethernet.h>
#include <Servo.h>
int led = 4;
Servo microservo;
int pos = 0;
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
//physical mac address
byte ip[] = { 192, 168, 1, 178 }; // ip in lan (that's what you need to use in your browser.
("192.168.1.178")
byte gateway[] = { 192, 168, 1, 1 }; // internet access via router
byte subnet[] = { 255, 255, 255, 0 };
//subnet mask
EthernetServer server(80);
//server port
String readString;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }
  pinMode(led, OUTPUT);
  microservo.attach(7);
  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}

void loop() {
  // Create a client connection
  EthernetClient client = server.available();
```

```

if (client) {
    while (client.connected()) {
        if (client.available()) {
            char c = client.read();

            //read char by char HTTP request
            if (readString.length() < 100) {
                //store characters to string
                readString += c;
                //Serial.print(c);
            }

            //if HTTP request has ended
            if (c == '\n') {
                Serial.println(readString); //print to serial
monitor for debugging

                client.println("HTTP/1.1 200 OK"); //send new
page
                client.println("Content-Type: text/html");
                client.println();
                client.println("<HTML>");
                client.println("<HEAD>");
                client.println("<meta name='apple-mobile-web-
app-capable' content='yes' />");
                client.println("<meta name='apple-mobile-web-
app-status-bar-style' content='black-translucent' />");
                client.println("<link rel='stylesheet'
type='text/css'
href='http://randomnerdtutorials.com/ethernetcss.css'
/>");

                client.println("<TITLE>Random Nerd Tutorials
Project</TITLE>");
                client.println("</HEAD>");
                client.println("<BODY>");
                client.println("<H1>Random Nerd Tutorials
Project</H1>");
                client.println("<hr />");
                client.println("<br />");
                client.println("<H2>Arduino with Ethernet
Shield</H2>");
                client.println("<br />");
                client.println("<a href=\"/?button1on\">Turn
On LED</a>");

            }
        }
    }
}

```

```

        client.println("<a href=\"/?button1off\">Turn
Off LED</a><br />");
        client.println("<br />");
        client.println("<br />");
        client.println("<a
href=\"/?button2on\">Rotate Left</a>");
        client.println("<a
href=\"/?button2off\">Rotate Right</a><br />");
        client.println("<p>Created by Rui Santos. Visit
http://randomnerdtutorials.com for more projects!</p>");
        client.println("<br />");
        client.println("</BODY>");
        client.println("</HTML>");

delay(1);
//stopping client
client.stop();
//controls the Arduino if you press the buttons
if (readString.indexOf("?button1on") >0) {
    digitalWrite(led, HIGH);
}
if (readString.indexOf("?button1off") >0) {
    digitalWrite(led, LOW);
}
if (readString.indexOf("?button2on") >0) {
    for(pos = 0; pos < 180; pos += 3) // goes
from 0 degrees to 180 degrees
    {
        microservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15); // waits 15ms for the servo to reach the position
    }
}
if (readString.indexOf("?button2off") >0) {
    for(pos = 180; pos>=1; pos-=3) // goes
from 180 degrees to 0 degrees
    {
        microservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15); // waits 15ms for the servo to reach the position
    }
}

```

```
//clearing string for next read  
readString="";  
  
}  
}  
}  
}  
}
```

Note

If you try this project. You can only access that IP address from your home. This means you must be connected to the same router that you're ethernet shield is connected to. That picture below is from me accessing my webserver with my iPad.

Complete Guide for Ultrasonic Sensor HC-SR04

	View Project on Random Nerd Tutorials	Click here
	View code on GitHub (complex code)	Click here
	View code on GitHub (easy code with library)	Click here
	NewPing library	Click here

Introduction

This post is all about the Ultrasonic Sensor HC - SR04. I'll explain how it works, show some features and share an Arduino Project example to help you with your projects.

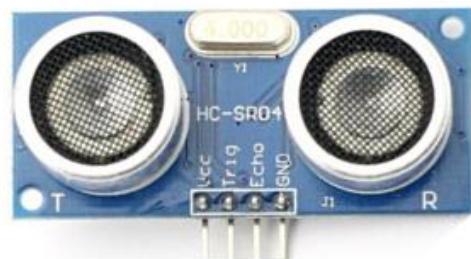
Description

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 2cm to 400 cm or 1" to 13 feet. Its 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.

Features

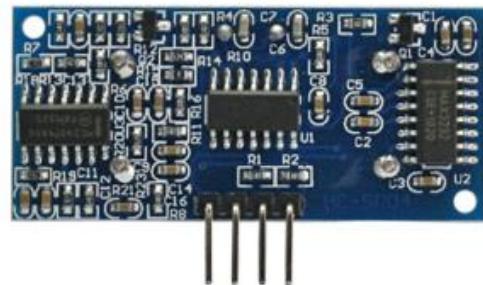
- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Current: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree

- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm



Sensor Pin

- VCC: +5VDC
- Trig : Trigger (INPUT)
- Echo: Echo (OUTPUT)
- GND: GND



Arduino with HC - SR04 Sensor

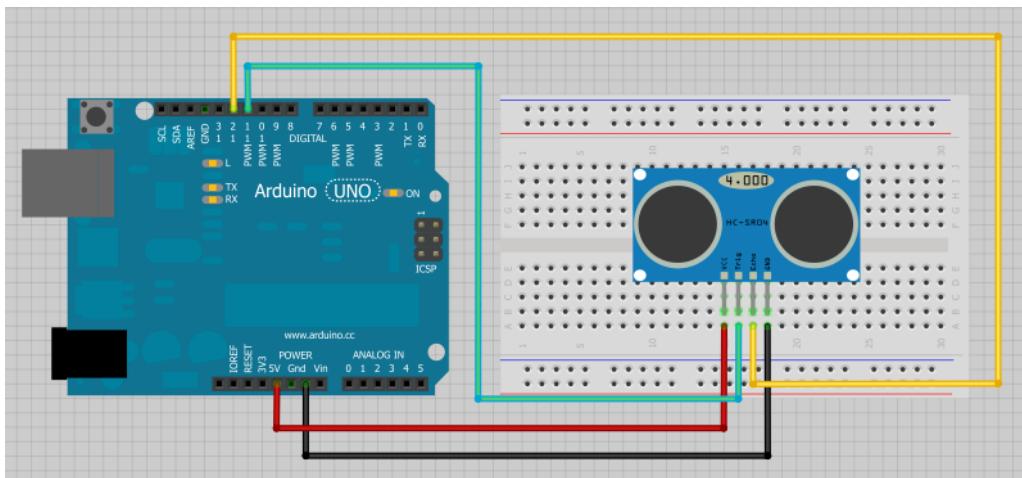
This sensor is really cool and popular among the Arduino Tinkerers. So I've decided to post a project example using this sensor. In this project the ultrasonic sensor read and write the distance in the serial monitor. It's really simple.

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.

Schematics



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);
}

```

Source code with NewPing

Below is an example using the NewPing library. Download the library [here](#).

[View code on GitHub](#)

```
/*
```

```

* Posted on http://randomnerdtutorials.com
* created by http://playground.arduino.cc/Code/NewPing
*/
#include <NewPing.h>

#define TRIGGER_PIN 12
#define ECHO_PIN 11
#define MAX_DISTANCE 200

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // 
NewPing setup of pins and maximum distance.

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

void loop() {
    delay(50);
    unsigned int uS = sonar.ping_cm();
    Serial.print(uS);
    Serial.println("cm");
}

```

Where to buy?

[Click here to get one.](#)



Note

If the HC-SR04 does not receive an echo then the output never goes low. Devantec and Parallax sensors time out after 36ms and I think 28ms respectively. If you use Pulsin as above then with no return echo

the program will hang for 1 second which is the default timeout for Pulsin. You need to use the timeout parameter.

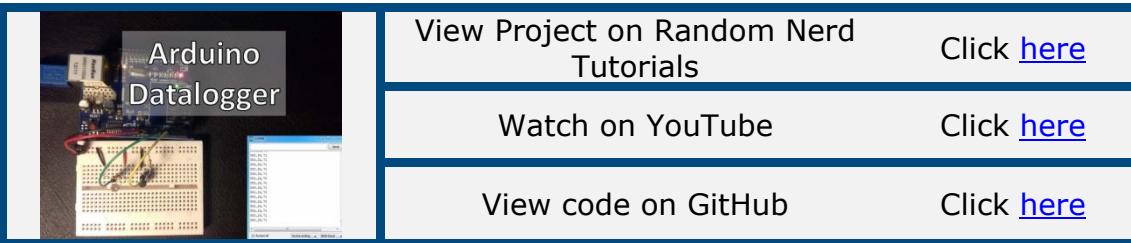
<http://arduino.cc/en/Reference/PulseIn>

The HC-SR04 barely works to 10 feet giving a total path length of 20 feet and a path time of about 20ms so set the timeout to something above that, say 25 or 30ms.

If you put a resistor, say 2k2 between E and T then only connect to T you can use the HC-SR04 from just one Arduino pin. Look up single pin operation of ultrasonic sensors.

Also if you are using a HC-SR04 with a PicAxe you need to up the clockspeed to at least 8MHz otherwise they don't see the start of the echo pulse so pulsin never starts. The HC-SR04 works fine with a BS2.

Datalogger with Temperature Sensor and Photoresistor



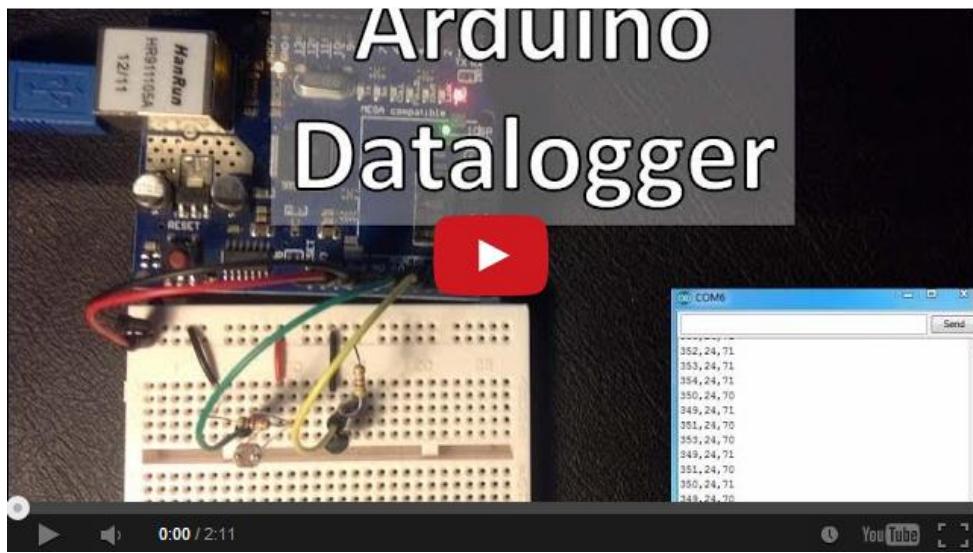
Introduction

In this project I'm going to create a simple Datalogger with my Arduino and an Ethernet shield.

With this concept you can change my code and monitor any sensor you desire.

I'll be using a photoresistor and a temperature sensor and all the information will be stored in a micro SD card.

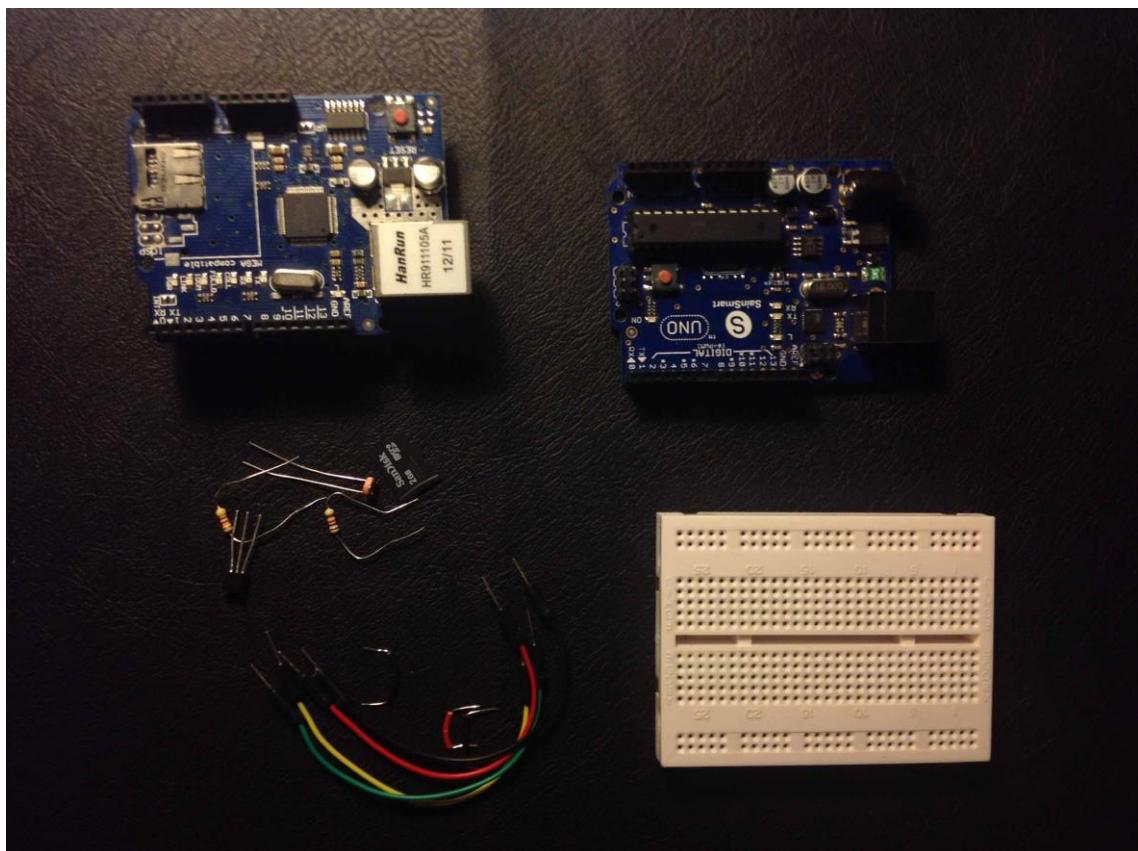
Watch the video below for a complete tutorial



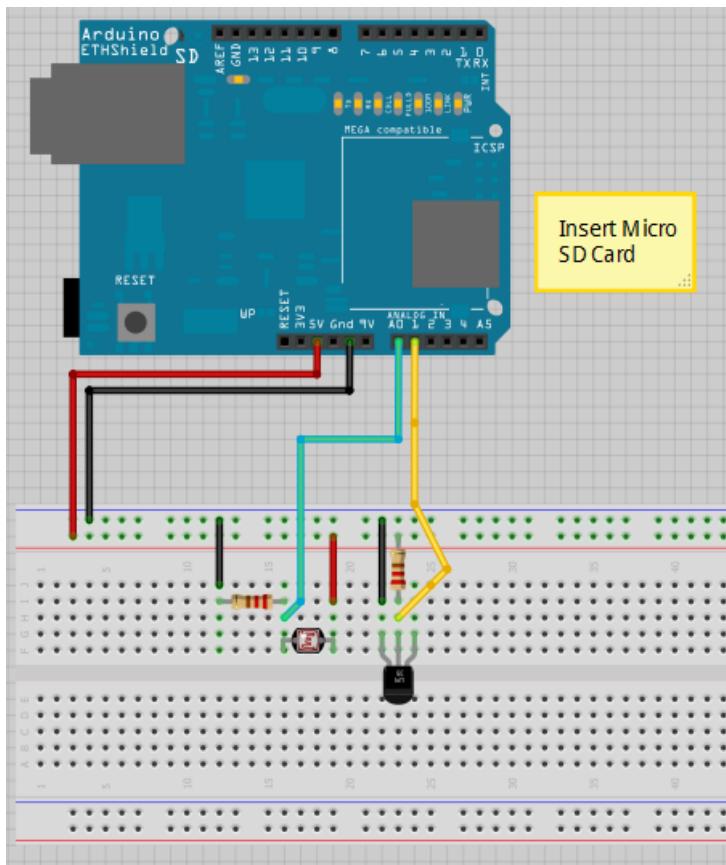
Watch on YouTube: <http://youtu.be/R19On2w5qbU>

Parts Required

- 1x Arduino Uno
- 1x Ethernet Shield
- 1x Micro SD Card
- 1x Temperature Sensor (I'm using the LM335Z)
- 1x Photoresistor
- 1x 1k ohm Resistor
- 1x 4700 ohm Resistor
- 1x Breadboard (or two)
- Jumper Cables



Schematics



Upload the code below

[View code on GitHub](#)

```
/*
Modified by Rui Santos
For more Arduino Projects: http://randomnerdtutorials.com
```

SD card datalogger

This example shows how to log data from three analog sensors to an SD card using the SD library.

The circuit:

- ```
* analog sensors on analog ins 0, 1, and 2
* SD card attached to SPI bus as follows:
** MOSI - pin 11
** MISO - pin 12
** CLK - pin 13
```

```

** CS - pin 4

based on Tom Igoe example.
 */

#include <SD.h>
float tempK; //stores kelvin temperature
int sensor; //stores sensor value everytime we use the
analog read function
// On the Ethernet Shield, CS is pin 4. Note that even if
it's not
// used as the CS pin, the hardware CS pin (10 on most
Arduino boards,
// 53 on the Mega) must be left as an output or the SD
library
// functions will not work.
const int chipSelect = 4;
int analogPin=0; //help us read the analogpin of our
INPUTS
void setup()
{
 // Open serial communications and wait for port to open:
 Serial.begin(9600);
 while (!Serial) {
 ; // wait for serial port to connect. Needed for
Leonardo only
 }
 Serial.print("Initializing SD card...");
 // make sure that the default chip select pin is set to
 // output, even if you don't use it:
 pinMode(10, OUTPUT);

 // see if the card is present and can be initialized:
 if (!SD.begin(chipSelect)) {
 Serial.println("Card failed, or not present");
 // don't do anything more:
 return;
 }
 Serial.println("card initialized.");
 //Defines Photoresistor as an INPUT on PIN Number 0
 pinMode(0, INPUT);
 //Defines Temperature sensor as an INPUT on PIN Number 1
 pinMode(1, INPUT);
}

```

```

void loop()
{
 // make a string for assembling the data to log:
 String dataString = "";
 // reads three sensors and append to the string:
 for (analogPin = 0; analogPin < 3; analogPin++) {
 //reads the photoresistor on PIN0
 if(analogPin==0) {
 sensor = analogRead(analogPin);
 }
 //reads the kelvin Tempeature on PIN1
 //then converts our tempeature to Degrees
 else if(analogPin==1) {
 //reads temperature and converts to kelvin
 tempK = (((analogRead(analogPin)/ 1023.0) * 5.0) *
100.0);
 //Converts Kelvin to Celsius minus 1.5 degrees error
 sensor = tempK - 273.0;
 }
 //reads the Kelvin temperature on PIN1
 //then converts our tempeature to Farenheit
 else{
 //reads temperature and converts to kelvin
 tempK = (((analogRead(analogPin-1)/ 1023.0) * 5.0) *
100.0);
 //Converts Kelvin to Farenheit
 sensor = ((tempK - 2.5) * 9 / 5) - 459.67;
 }
 //stores our values
 dataString += String(sensor);
 if (analogPin < 2) {
 dataString += ",";
 }
 }

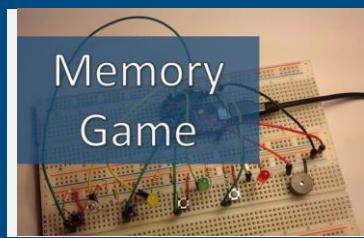
 // open the file. note that only one file can be open at
 a time,
 // so you have to close this one before opening another.
 File dataFile = SD.open("data.txt", FILE_WRITE);

 // if the file is available, write to it:
 if (dataFile) {
 dataFile.println(dataString);
 dataFile.close();
 // print to the serial port too:
 }
}

```

```
 Serial.println(dataString);
}
// if the file isn't open, pop up an error:
else {
 Serial.println("error opening datalog.txt");
}
}
```

# Teensy/Arduino - Memory Game



View Project on Random Nerd Tutorials

Click [here](#)

Watch on YouTube

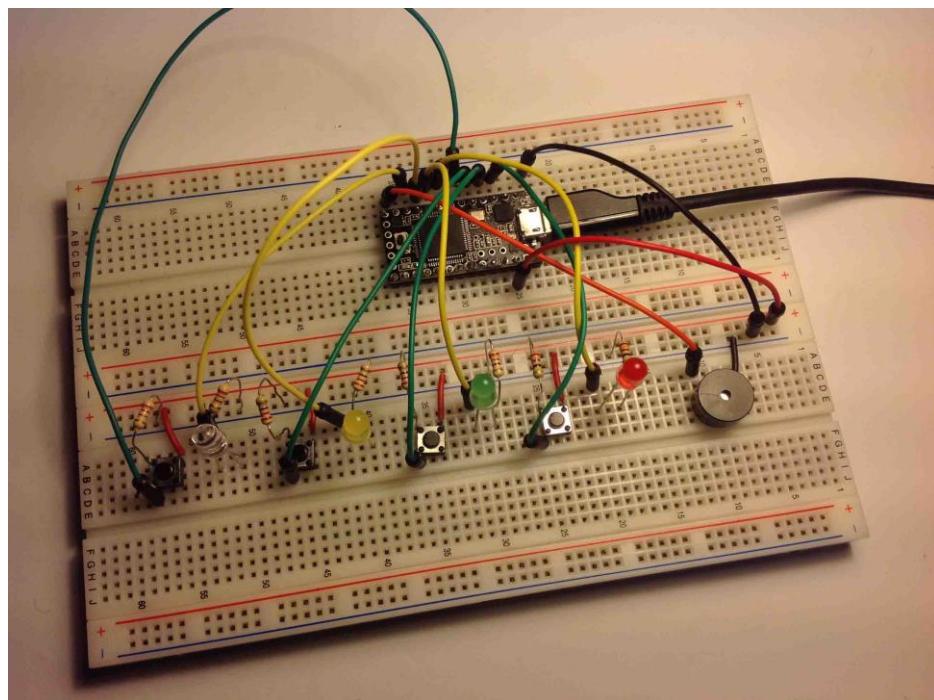
Click [here](#)

View code on GitHub

Click [here](#)

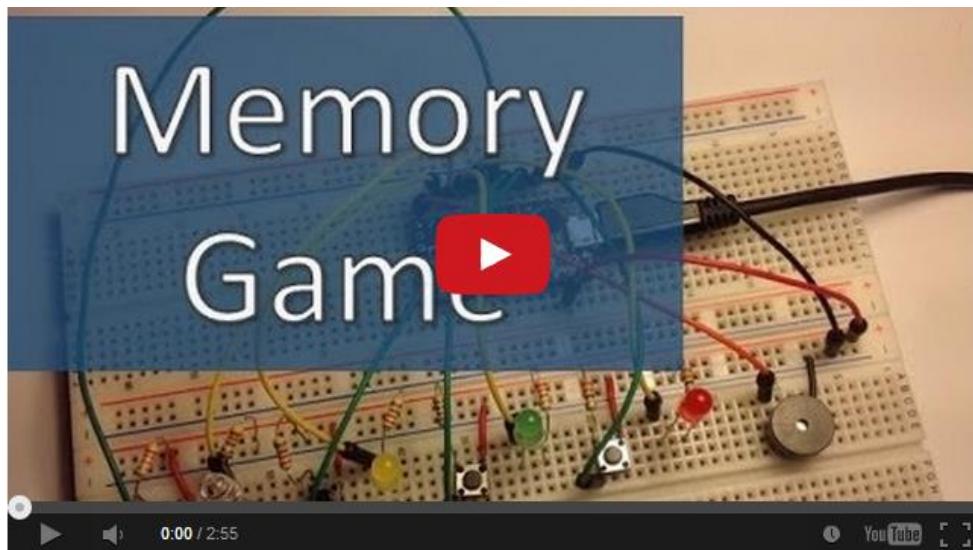
## Introduction

This project is all about creating a simple game to test your memory.



I'll be using a Teensy 3.0 board. If you want to know more about this board please [click here](#) to read a Getting Started Guide I've created a few days ago. This project is also 100% compatible with the Arduino.

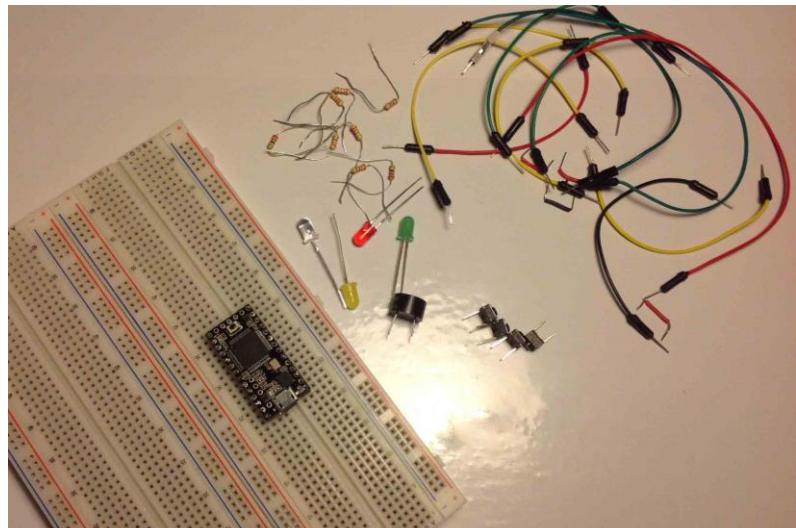
**Watch the video below to see how it works**



Watch on YouTube: <http://youtu.be/cDEmH0iguMw>

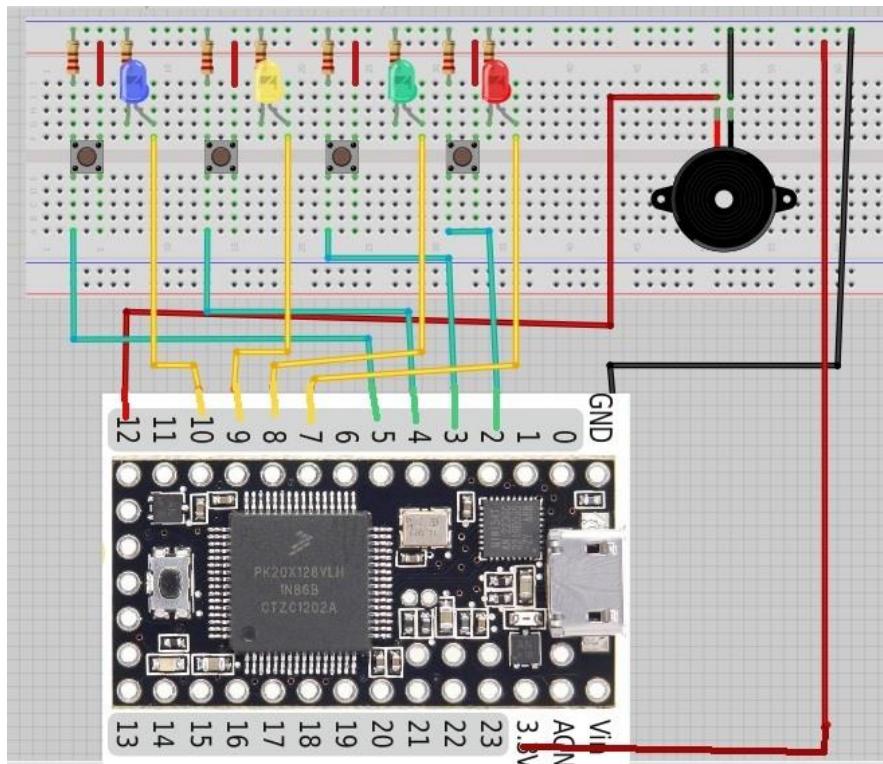
## Parts Required

- 1x [Teensy 3.0](#) (or an Arduino)
- 8x 220 Ohm Resistor
- 4x LED's
- 4x Pushbuttons
- 1x Buzzer
- 1x Breadboard
- Jumper Cables

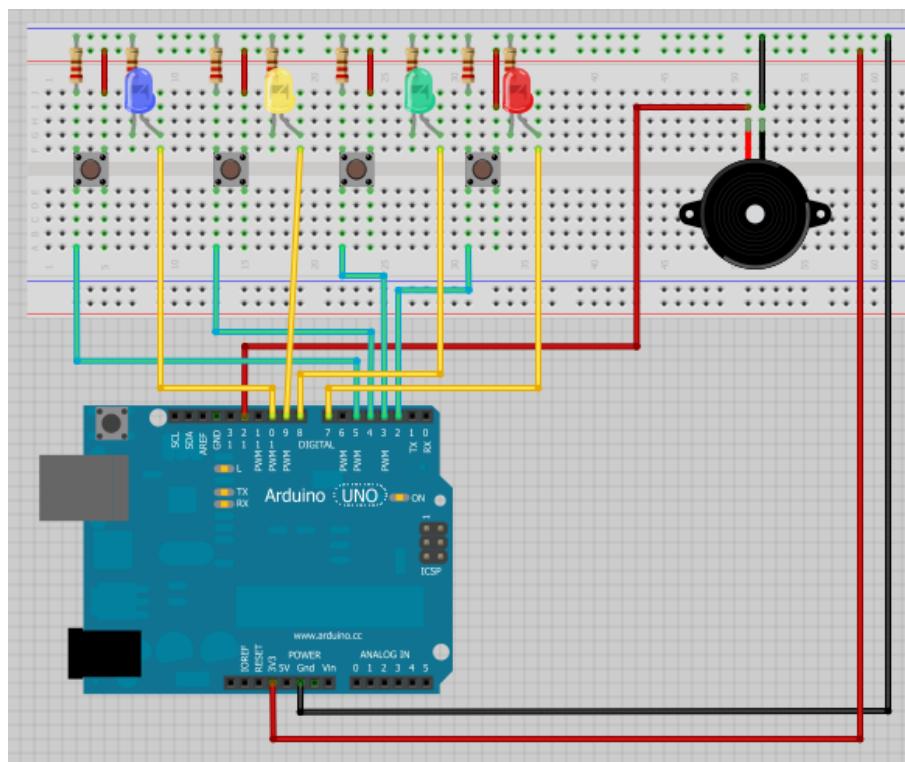


## Schematics

Teensy



Arduino



## Upload the code below

This code works with Teensy and with the Arduino.

[View code on GitHub](#)

```
/*
Memory Game with Arduino
Based on a project by Jeremy Wilson
Modified by Rui Santos
Visit: http://randomnerdtutorials.com
*/
// Constants
const int button1 = 2; // 1st button controls
Blue LED
const int button2 = 3; // 2nd button controls
Yellow LED
const int button3 = 4; // 3rd button controls
Green LED
const int button4 = 5; // 4th button controls Red
LED
const int led1 = 7; // Blue LED
const int led2 = 8; // Yellow LED
const int led3 = 9; // Green LED
const int led4 = 10; // Red LED
const int buzzer = 12; // Buzzer Output
const int tones[] = {1915, 1700, 1519, 1432, 2700}; //
tones when you press the LED's - the last one is when you
fail.

// Variables
int buttonState[] = {0,0,0,0}; // current state of
the button
int lastButtonState[] = {0,0,0,0}; // previous state
of the button
int buttonPushCounter[] = {0,0,0,0};

void playTone(int tone, int duration) {
 for (long i = 0; i < duration * 1000L; i += tone * 2) {
 digitalWrite(buzzer, HIGH);
 delayMicroseconds(tone);
 digitalWrite(buzzer, LOW);
 delayMicroseconds(tone);
```

```

 }
 }

void setup() {
 // initialize inputs :
 randomSeed(analogRead(0));
 pinMode(button1, INPUT);
 pinMode(button2, INPUT);
 pinMode(button3, INPUT);
 pinMode(button4, INPUT);
 // initialize outputs:
 pinMode(led1, OUTPUT);
 pinMode(led2, OUTPUT);
 pinMode(led3, OUTPUT);
 pinMode(led4, OUTPUT);
 pinMode(buzzer, OUTPUT);
 // initialize serial communication for debugging:
 //Serial.begin(9600);
}

int game_on = 0;
int wait = 0;
int currentlevel = 1; // This is the level (also the
number of button presses to pass to next level)
long rand_num = 0; //initialize long variable for random
number from 0-100.
int rando = 0; //initialize random integer for
loopgame_on. Will be from 1-4 later.
int butwait = 500; //amount of time to wait for next
button input (ghetto de-bounce)
int ledtime = 500; //amount of time each LED flashes for
when button is pressed
int n_levels = 10; //number of levels until the game is
won
int pinandtone = 0; //This integer is used when the
sequence is displayed
int right = 0; //This variable must be 1 in order to go to
the next level
int speedfactor = 5; //This is the final speed of the
lights and sounds for the last level. This increases as
more games are won
int ledelay = 200; //Initializing time for LED. This will
decrease as the level increases

void loop() {

```

```

int n_array[n_levels];
int u_array[n_levels];

int i;
//clears arrays both "n_array" and "u_array" and starts a
new game
if (game_on == 0) {
for(i=0; i<n_levels; i=i+1) {
 n_array[i]=0;
 u_array[i]=0;
 rand_num = random(1,200);
 if (rand_num <= 50)
 rando=0;
 else if (rand_num>50 && rand_num<=100)
 rando=1;
 else if (rand_num>100 && rand_num<=150)
 rando=2;
 else if (rand_num<=200)
 rando=3;
 //saves a random number in our n_array
 n_array[i]=rando;
}
game_on = 1;
}

//shows the user the current sequence
if (wait == 0) {
 delay (200);
i = 0;
for (i = 0; i < currentlevel; i= i + 1) {
 leddelay =
ledtime/(1+(speedfactor/n_levels)*(currentlevel - 1));
 pinandtone = n_array[i];
 digitalWrite(pinandtone+7, HIGH);
 playTone(tones[pinandtone], leddelay);
 digitalWrite(pinandtone+7, LOW);
 delay(100/speedfactor);
}
 wait = 1;
}
i = 0;
int buttonchange = 0;
int j = 0; // This is the current position in the sequence

```

```

while (j < currentlevel) {
 while (buttonchange == 0) {
 for (i = 0; i < 4; i = i + 1) {
 buttonState[i] = digitalRead(i+2);
 buttonchange = buttonchange + buttonState[i];
 }
 }
 for (i = 0; i < 4; i = i + 1) {
 if (buttonState[i] == HIGH) {
 digitalWrite(i+7, HIGH);
 playTone(tones[i], ledtime);
 digitalWrite(i+7, LOW);
 wait = 0;
 u_array[j]=i;
 buttonState[i] = LOW;
 buttonchange = 0;
 }
 }
 if (u_array[j] == n_array[j]) {
 j++;
 right = 1;
 }
 else {

 right = 0;
 i = 4;
 j = currentlevel;
 wait = 0;
 }
}

if (right == 0) {
 delay(300);
 i = 0;
 game_on = 0;
 currentlevel = 1;
 for (i = 0; i < 4; i = i + 1) {
 digitalWrite(i+7, HIGH);
 }
 playTone(tones[4], ledtime);
 for (i = 0; i < 4; i = i + 1) {
 digitalWrite(i+7, LOW);
 }
 delay (200);
 for (i = 0; i < 4; i = i + 1) {
}

```

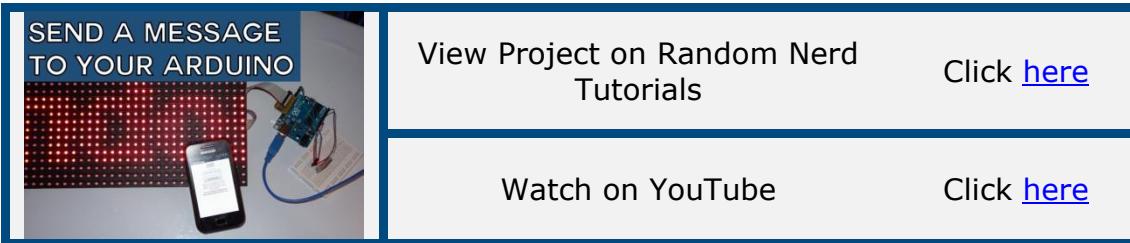
```

 digitalWrite(i+7, HIGH);
 }
 playTone(tones[4], ledtime);
for (i = 0; i < 4; i = i + 1){
 digitalWrite(i+7, LOW);
}

delay(500);
game_on = 0;
}
//if you insert the right sequence it levels up
if (right == 1){
 currentlevel++;
 wait = 0;
}
//if you finish the game
if (currentlevel == n_levels){
 delay(500);
 // The following is the victory sound:
 int notes[] = {2, 2, 2, 2, 0, 1, 2, 1, 2};
 int note = 0;
 int tempo[] = {200, 200, 200, 400, 400, 400, 200, 200,
600};
 int breaks[] = {100, 100, 100, 200, 200, 200, 300, 100,
200};
 for (i = 0; i < 9; i = i + 1){
 note = notes[i];
 digitalWrite(note+7, HIGH);
 playTone(tones[note], tempo[i]);
 digitalWrite(note+7, LOW);
 delay(breaks[i]);
 }
 //sets game_on to 0, so it restarts a new game
 game_on = 0;
 currentlevel = 1;
 n_levels = n_levels + 2;
 speedfactor = speedfactor + 1;
}
}

```

# Android App that Sends a Message to Your Arduino



## Introduction

In this project I'm going to show an Android app that sends a message to your Arduino via bluetooth. And that message will be displayed in a Doted Matrix display.

### Resources for this project:

- [How To Use App Inventor With an Arduino](#)
- [Control 2 DC Motors Via Bluetooth with an android app created with MIT App Inventor](#)
- [Review of the Freetronics DMD 32x16 Red](#)
- [Review of the HC-05 Bluetooth Module](#)

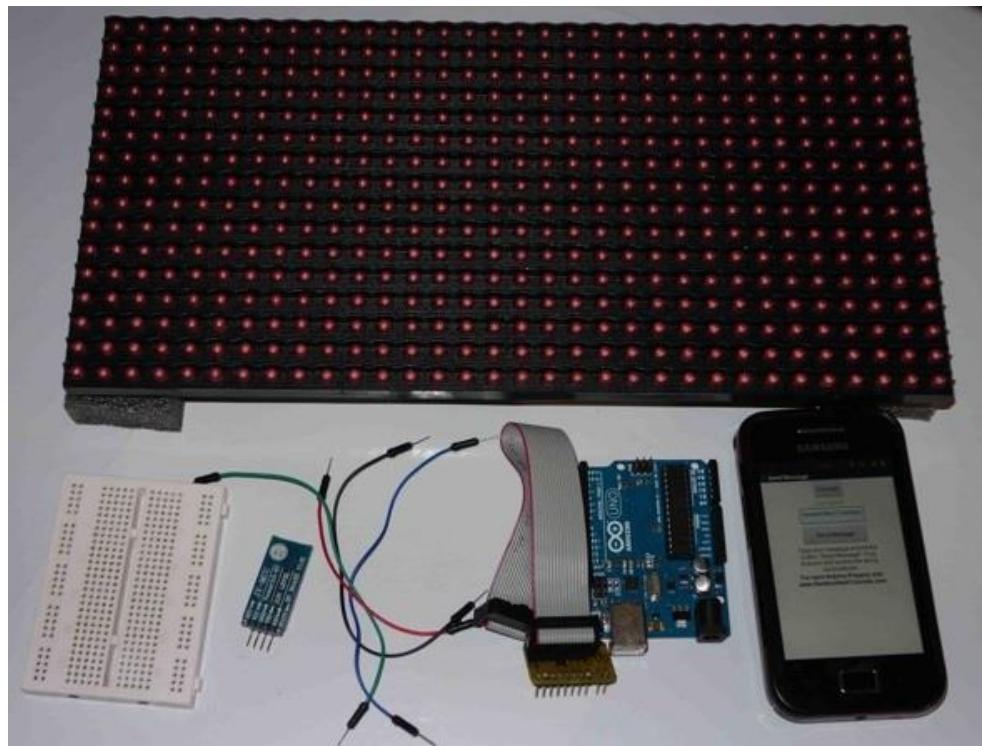
## Watch the video below



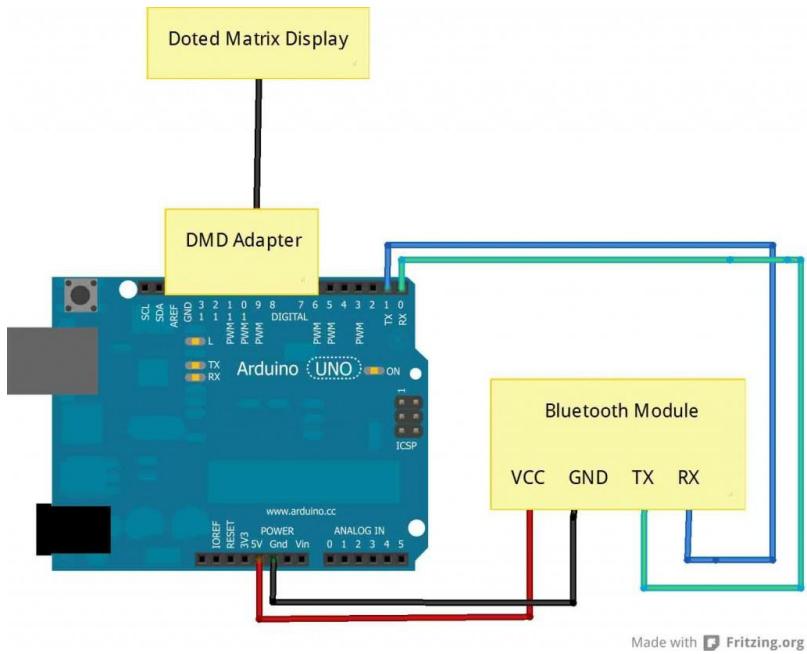
Watch on YouTube: <http://youtu.be/RQvSMiZiNoc>

## Parts Required

- 1x Arduino Uno
- 1x DMD
- 1x Bluetooth Module (for example: HC-05, [click here to read my review about this Bluetooth module](#))
- 1x Smartphone (any Android will work, I've only tested with Samsung Galaxy Ace)
- Android Application (you can download it in the next step)
- 1x Breadboard
- Jumper Cables



## Schematics



## Source Code

- [Arduino Sketch](#)
- [Install the DMD library](#)
- [Send\\_Message.apk](#)
- [Send\\_Message.aia](#) (to edit the android app)

## Note

If you want to edit my app this is what you need to do. Download [Send\\_Message.aia](#) and upload it to [MIT App Inventor](#).

## Tips

1. You need to remove the RX and TX cables when you're uploading the sketch to your Arduino.
2. Sometimes people connect the TX from the bluetooth module to the TX of the Arduino... that's wrong and it won't work. Make sure you connect it properly, the TX into RX and the RX into the TX.
3. If the HC-05 Bluetooth Module asks for a password, It's '1234'.
4. Before Testing my android app, test if you've made all the connections correctly. How you can do that? Simply enter some

messages (for example: "Random") into your serial monitor and your DMD should display that message.

# Control your Arduino with Voice Commands [Android App]



View Project on Random Nerd Tutorials

Click [here](#)

Watch on YouTube

Click [here](#)

## Introduction

In this project we're going to control an Arduino with Voice commands with a Simple android App that I've create with MIT App Inventor.

## Watch the video below



Watch on YouTube: <http://youtu.be/TEKQUMzg6UY>

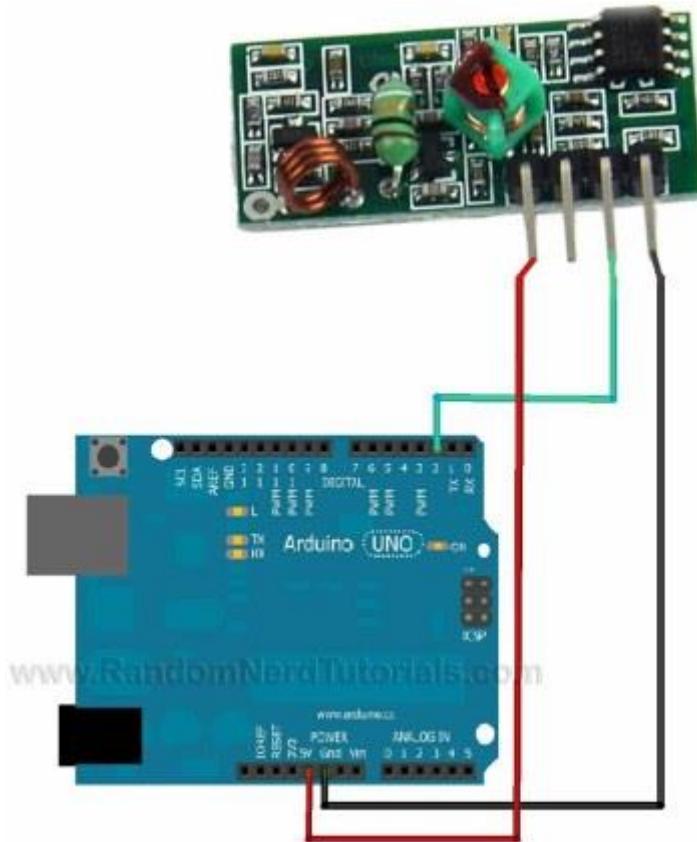
## Resources for this project:

- [How To Use App Inventor With an Arduino](#)
- [How to control outlets with 433Mhz Transmitter](#)
- [Review of the HC-05 Bluetooth Module](#)

## Parts Required

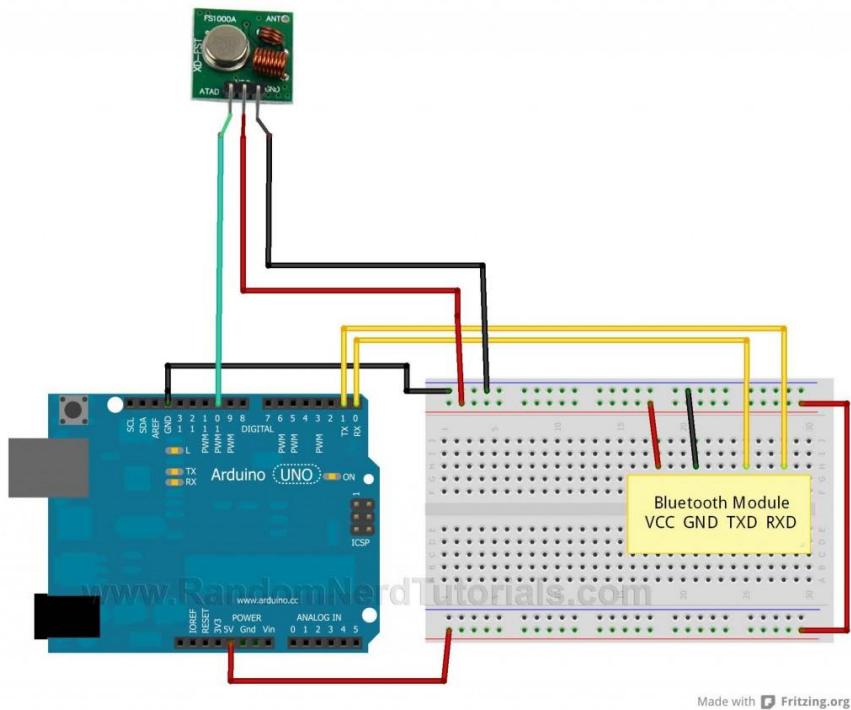
- 1x Arduino ([Click to see on Amazon](#))
- 1x Smartphone
- 1x Bluetooth Module (for example HC-06 - [Read my review here](#) or [click to see on Amazon](#))
- 1x 433Mhz Receiver and Transmitter ([Click to see on Amazon](#))
- 2x Remote Controlled Sockets with a Remote Control (Controlled by 433Mhz Frequency) ([Click to see on Amazon](#))
- 1x Breadboard ([Click to see on Amazon](#))
- Jumper Cables

## Receiver Circuit



[Click here to Download the RCSwitch Library](#). Install it and Re-open the Arduino IDE. Then open the example “ReceiveDemo\_Advanced”. Upload the code and open the serial monitor. Start pressing the buttons from the Remote you’re going to use and save them.

# Final Circuit



**Upload and install the source code below**

- Arduino Sketch
  - Install the RC Switch Library
  - Voice\_Control.apk
  - Voice\_Control.aia (to edit the android app)

[Click here to download all](#)

## Note

If you want to edit my app this is what you need to do. Download Voice\_Control.aia and upload it to [MIT App Inventor](#).

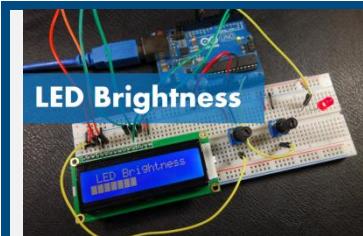
## Tips

1. You need to remove the RX and TX cables when you're uploading the sketch to your Arduino.
  2. Sometimes people connect the TX from the bluetooth module to the TX of the Arduino... that's wrong and it won't work. Make sure you connect it properly, the TX into RX and the RX into the TX.

3. If the HC-05 Bluetooth Module asks for a password, It's '1234'.
4. Before Testing my "BlueLED" app, test if you've made all the connections correctly. How you can do that? Simply enter numbers ('1', '0') into your serial monitor and your LED should be turning on and off.

[P.S. Click here to learn how to use the 433Mhz Transmitter/Receiver circuit to control outlets.](#)

# Display the LED Brightness on a LCD 16x2



View Project on Random Nerd Tutorials

Click [here](#)

Watch on YouTube

Click [here](#)

View code on GitHub

Click [here](#)

## Introduction

In this project we're going to display the LED brightness on a LCD 16x2 with a progress bar.

## Watch the video below

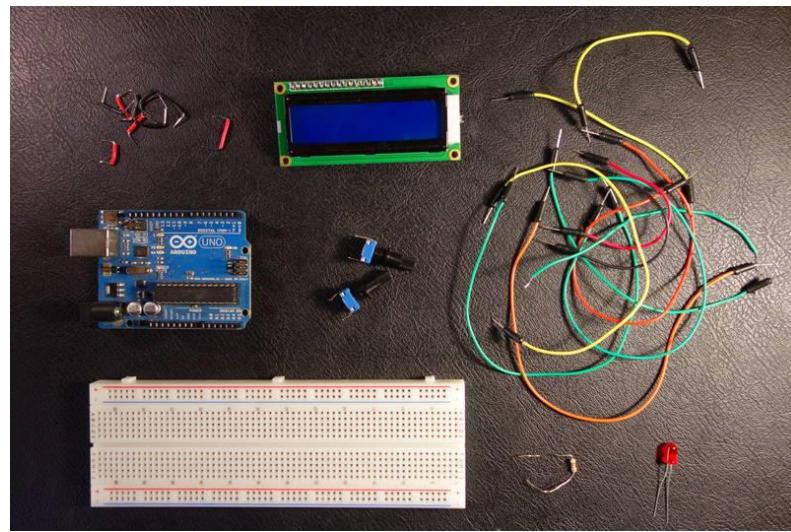


Watch on YouTube: <http://youtu.be/sAklcqiywPw>

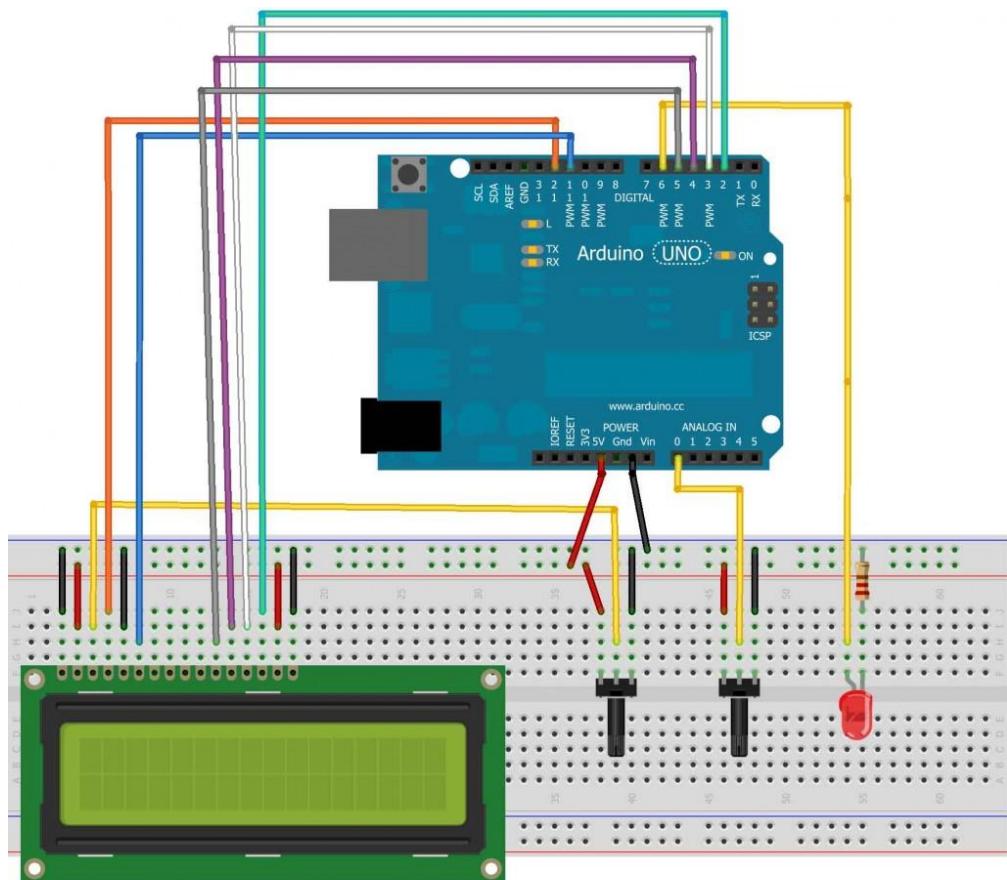
## Parts Required

- 1x Arduino ([Click to see on Amazon](#))
- 1x Breadboard ([Click to see on TaydaElectronics](#))
- 1x LCD 16x2 ([Click to see on TaydaElectronics](#))
- 2x 10k Ohm Potentiometers([Click to see on TaydaElectronics](#))
- 1x 5mm LED ([Click to see on TaydaElectronics](#))

- 1x 220Ohm Resistor ([Click to see on TaydaElectronics](#))
- Jumper Cables



## Schematics



#### INTERFACE PIN CONNECTIONS

| Pin No. | Symbol  | Level   | Description                                                  |
|---------|---------|---------|--------------------------------------------------------------|
| 1       | VSS     | ---     | Ground for Logic ( 0V)                                       |
| 2       | VDD     | ---     | Power supply for Logic (+5.0V)                               |
| 3       | V0      | ---     | Power supply for LCD drive                                   |
| 4       | RS      | H/L     | Register selection ( H:Data register,L:Instruction register) |
| 5       | R/W     | H/L     | Read/Write selection ( H:read,L:Write)                       |
| 6       | E       | H/L → L | Enable signal for LCM                                        |
| 7~14    | DB0~DB7 | H/L     | Data Bus Lines                                               |
| 15      | LEDA    | ---     | Power supply for backlight(+5.0V)                            |
| 16      | LEDK    | ---     | Power supply for backlight(-)                                |

#### LCD Pinout

### Upload the code below

[View code on GitHub](#)

```
/*
Created by Rui Santos
```

All the resources for this project:  
<http://randomnerdtutorials.com/>

Based on some Arduino code examples  
\*/

```
// include the library code
#include <LiquidCrystal.h>

// initialize the library with the numbers of the
interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int potPin = A0; // Analog pin 0 for the LED
brightness potentiometer
int ledPin = 6; // LED Digital Pin with PWM
int potValue = 0; // variable to store the value
coming from the potentiometer
int brightness = 0; // converts the potValue into a
brightness
int pBari = 0; // progress bar
int i = 0; // foor loop

//progress bar character for brightness
byte pBar[8] = {
 B11111,
 B11111,
 B11111,
 B11111,
```

```

B1111,
B1111,
B1111,
};

void setup() {
 // setup our led as an OUTPUT
 pinMode(ledPin, OUTPUT);
 // set up the LCD's number of columns and rows:
 lcd.begin(16, 2);
 // Print a message to the LCD
 lcd.print(" LED Brightness");
 //Create the progress bar character
 lcd.createChar(0, pBar);
}

void loop() {
 // clears the LCD screen
 lcd.clear();
 // Print a message to the LCD
 lcd.print(" LED Brightness");
 //set the cursor to line number 2
 lcd.setCursor(0,1);
 // read the value from the potentiometer
 potValue = analogRead(potPin);
 // turns the potValue into a brightness for the LED
 brightness=map(potValue, 0, 1024, 0, 255);
 //lights up the LED according to the brightness
 analogWrite(ledPin, brightness);
 // turns the brightness into a percentage for the bar
 pBari=map(brightness, 0, 255, 0, 16);
 //prints the progress bar
 for (i=0; i<pBari; i++)
 {
 lcd.setCursor(i, 1);
 lcd.write(byte(0));
 }
 // delays 750 ms
 delay(750);
}

```

# Teensy - Username and Password Auto Filler



View Project on Random Nerd Tutorials

Click [here](#)

Watch on YouTube

Click [here](#)

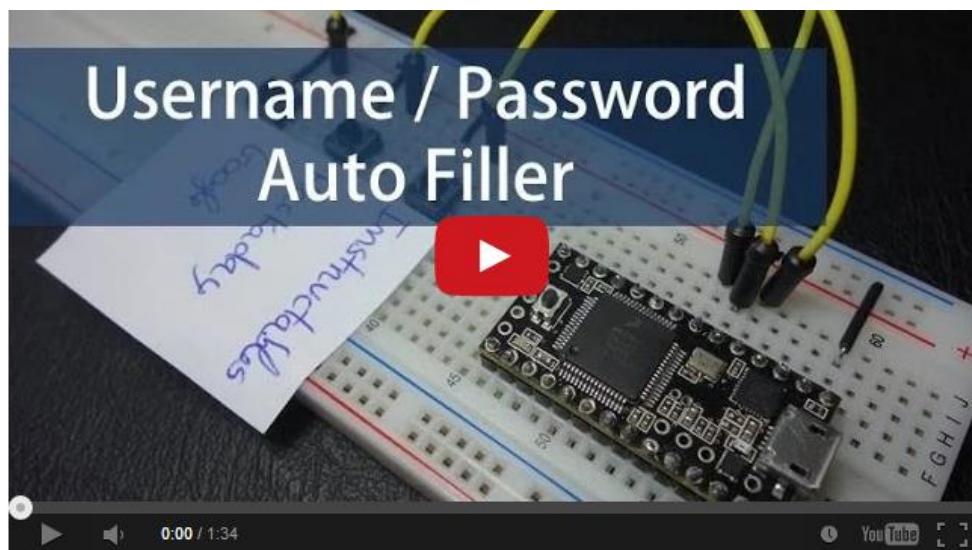
View code on GitHub

Click [here](#)

## Introduction

In this project we're going to make our own hardware password management tool.

**Watch the video below to see how it works**



Watch on YouTube: <http://youtu.be/NYISDdXiGmY>

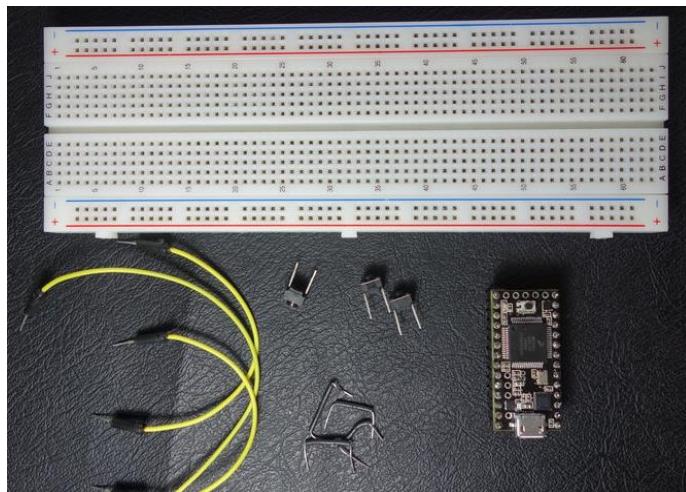
## Resources

I recommend you to take a look at those posts below to learn more about the Teensy board.

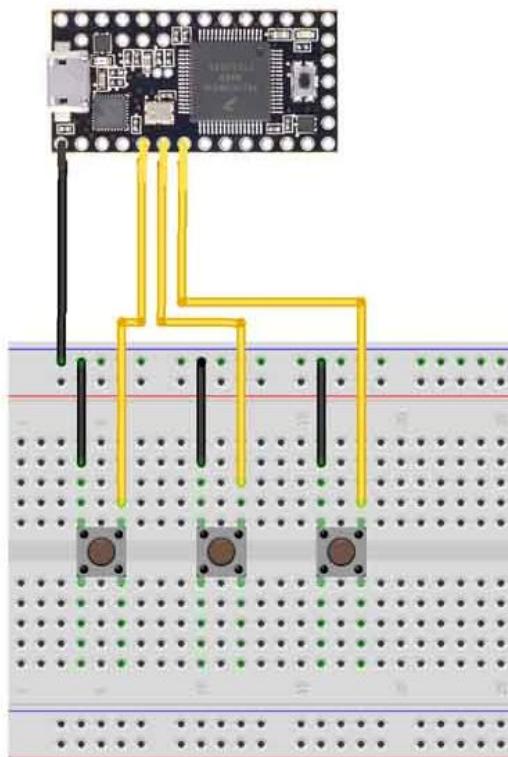
- [Getting Started with Teensy](#)
- [Teensy/Arduino - Memory Game](#)

## Parts Required

- 1x Teensy  
([Getting Started with Teensy](#))
- 3x Pushbuttons
- 1x Breadboard  
([Click to see on Amazon](#))
- Jumper Cables



## Schematics



## Upload the code below

### Note

With a few changes this project also works with the Arduino Leonardo

[View code on GitHub](#)

```
/*
Created by Rui Santos

All the resources for this project:
http://randomnerdtutorials.com/

Based on some Arduino code examples
*/

//include the bounce library
#include <Bounce.h>

// Creating Bounce objects for each button makes detecting
// changes very easy.
Bounce button3 = Bounce(3, 10);
Bounce button4 = Bounce(4, 10);
Bounce button5 = Bounce(5, 10);

void setup() {
 pinMode(3, INPUT_PULLUP);
 pinMode(4, INPUT_PULLUP);
 pinMode(5, INPUT_PULLUP);
}

void loop() {
 // updates all the buttons
 // don't use long delays in the loop
 button3.update();
 button4.update();
 button5.update();

 // when you press the buttons
 if (button3.fallingEdge()) {
 Keyboard.print("username_1");
 Keyboard.set_key1(KEY_TAB);
 Keyboard.send_now();
 Keyboard.print("password_1");
 Keyboard.set_key1(KEY_ENTER);
 }
}
```

```

 Keyboard.send_now();
}

if (button4.fallingEdge()) {
 Keyboard.print("username_2");
 Keyboard.set_key1(KEY_TAB);
 Keyboard.send_now();
 Keyboard.print("password_2");
 Keyboard.set_key1(KEY_ENTER);
 Keyboard.send_now();
}

if (button5.fallingEdge()) {
 Keyboard.print("username_3");
 Keyboard.set_key1(KEY_TAB);
 Keyboard.send_now();
 Keyboard.print("password_3");
 Keyboard.set_key1(KEY_ENTER);
 Keyboard.send_now();
}

// release all the keys
Keyboard.set_modifier(0);
Keyboard.set_key1(0);
Keyboard.send_now();
}

```

# Arduino with PIR Motion Sensor



View Project on Random Nerd Tutorials

Click [here](#)

Watch on YouTube

Click [here](#)

View arduino code

Click [here](#)

## Introduction

In this project we're going to create a simple circuit with an Arduino and PIR motion sensor that can detect movement.

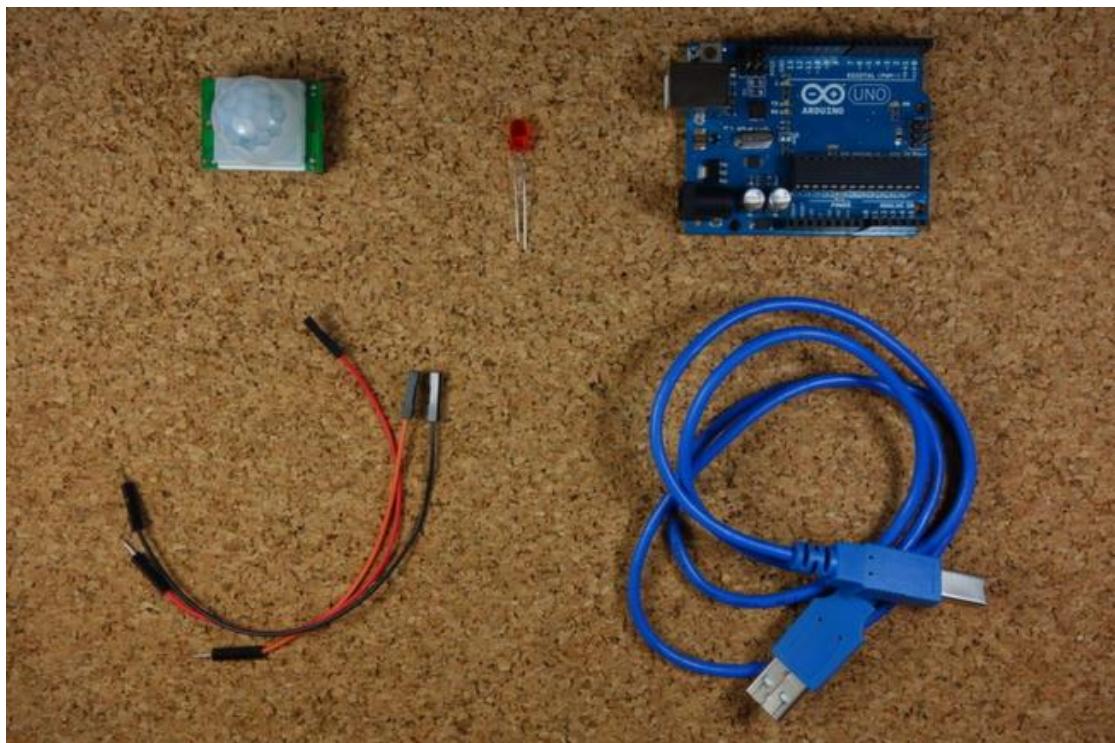
**Watch the video below to see how it works**



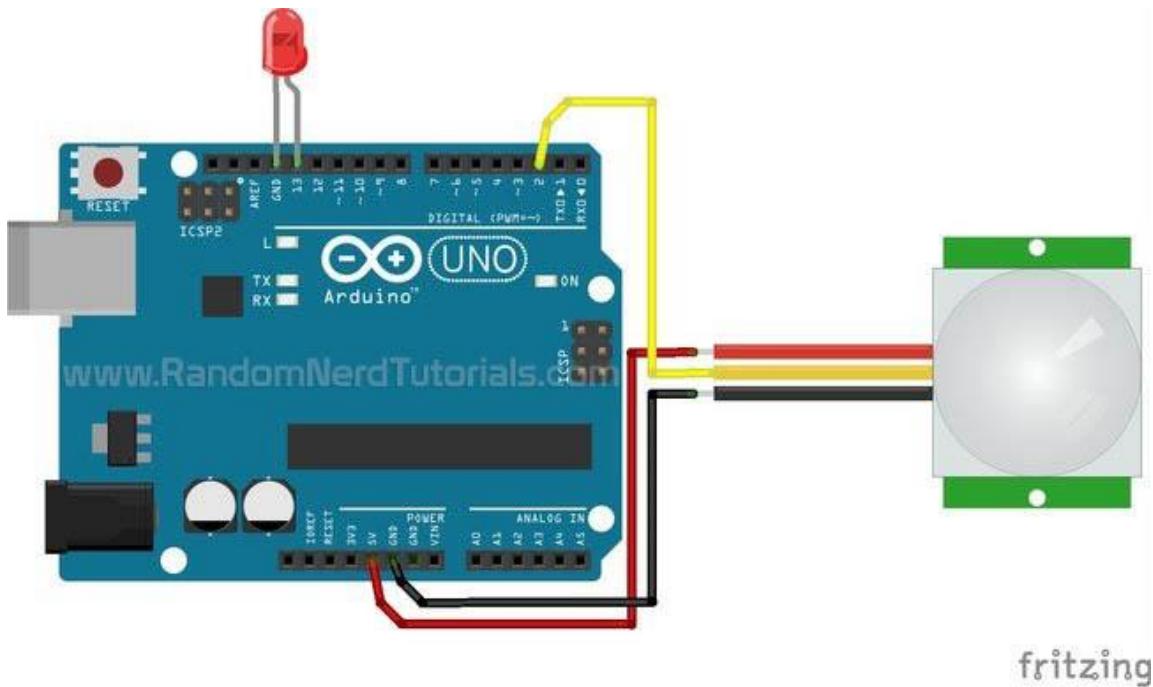
Watch on YouTube: <http://youtu.be/vJgtckLzoKM>

## Parts required

- 1x PIR Motion Sensor ([Click to see on Amazon](#))
- 1x Arduino ([Click to see on Amazon](#))
- 1x LED
- Jumper Cables ([Click to see on Amazon](#))



## Schematics



**Upload the code below**

[View code on GitHub](#)

/\*

```

Arduino with PIR motion sensor
For complete project details, visit:
http://RandomNerdTutorials.com/pirsensor
Modified by Rui Santos based on PIR sensor by Limor
Fried
*/

```

```

int led = 13; // the pin that the LED is
attached to
int sensor = 2; // the pin that the sensor is
attached to
int state = LOW; // by default, no motion
detected
int val = 0; // variable to store the
sensor status (value)

void setup() {
 pinMode(led, OUTPUT); // initialize LED as an output
 pinMode(sensor, INPUT); // initialize sensor as an
input
 Serial.begin(9600); // initialize serial
}

void loop() {
 val = digitalRead(sensor); // read sensor value
 if (val == HIGH) { // check if the sensor is
HIGH
 digitalWrite(led, HIGH); // turn LED ON
 delay(100); // delay 100 milliseconds

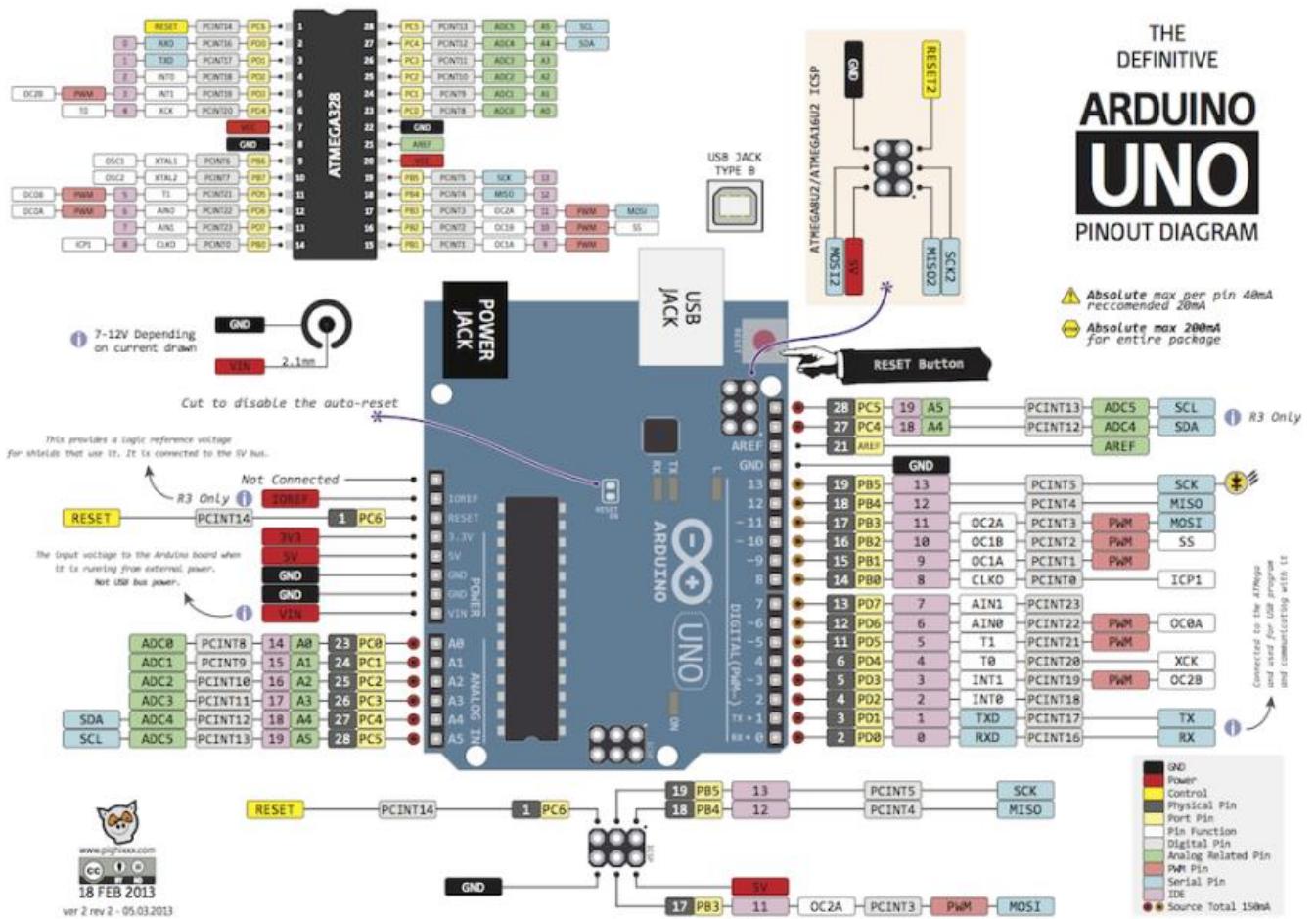
 if (state == LOW) {
 Serial.println("Motion detected!");
 state = HIGH; // update variable state to HIGH
 }
 }
 else {
 digitalWrite(led, LOW); // turn LED OFF
 delay(200); // delay 200 milliseconds

 if (state == HIGH) {
 Serial.println("Motion stopped!");
 state = LOW; // update variable state to LOW
 }
 }
}

```

# Resources

## Arduino Pinout



## ASCII Table

| Dec | Hex | Name              | Char | Ctrl-char | Dec | Hex | Char  | Dec | Hex | Char | Dec | Hex | Char |
|-----|-----|-------------------|------|-----------|-----|-----|-------|-----|-----|------|-----|-----|------|
| 0   | 0   | Null              | NUL  | CTRL-@    | 32  | 20  | Space | 64  | 40  | @    | 96  | 60  | '    |
| 1   | 1   | Start of heading  | SOH  | CTRL-A    | 33  | 21  | !     | 65  | 41  | A    | 97  | 61  | a    |
| 2   | 2   | Start of text     | STX  | CTRL-B    | 34  | 22  | "     | 66  | 42  | B    | 98  | 62  | b    |
| 3   | 3   | End of text       | ETX  | CTRL-C    | 35  | 23  | #     | 67  | 43  | C    | 99  | 63  | c    |
| 4   | 4   | End of xmit       | EOT  | CTRL-D    | 36  | 24  | \$    | 68  | 44  | D    | 100 | 64  | d    |
| 5   | 5   | Enquiry           | ENQ  | CTRL-E    | 37  | 25  | %     | 69  | 45  | E    | 101 | 65  | e    |
| 6   | 6   | Acknowledge       | ACK  | CTRL-F    | 38  | 26  | &     | 70  | 46  | F    | 102 | 66  | f    |
| 7   | 7   | Bell              | BEL  | CTRL-G    | 39  | 27  | '     | 71  | 47  | G    | 103 | 67  | g    |
| 8   | 8   | Backspace         | BS   | CTRL-H    | 40  | 28  | (     | 72  | 48  | H    | 104 | 68  | h    |
| 9   | 9   | Horizontal tab    | HT   | CTRL-I    | 41  | 29  | )     | 73  | 49  | I    | 105 | 69  | i    |
| 10  | 0A  | Line feed         | LF   | CTRL-J    | 42  | 2A  | *     | 74  | 4A  | J    | 106 | 6A  | j    |
| 11  | 0B  | Vertical tab      | VT   | CTRL-K    | 43  | 2B  | +     | 75  | 4B  | K    | 107 | 6B  | k    |
| 12  | 0C  | Form feed         | FF   | CTRL-L    | 44  | 2C  | ,     | 76  | 4C  | L    | 108 | 6C  | l    |
| 13  | 0D  | Carriage feed     | CR   | CTRL-M    | 45  | 2D  | -     | 77  | 4D  | M    | 109 | 6D  | m    |
| 14  | 0E  | Shift out         | SO   | CTRL-N    | 46  | 2E  | -     | 78  | 4E  | N    | 110 | 6E  | n    |
| 15  | 0F  | Shift in          | SI   | CTRL-O    | 47  | 2F  | /     | 79  | 4F  | O    | 111 | 6F  | o    |
| 16  | 10  | Data line escape  | DLE  | CTRL-P    | 48  | 30  | 0     | 80  | 50  | P    | 112 | 70  | p    |
| 17  | 11  | Device control 1  | DC1  | CTRL-Q    | 49  | 31  | 1     | 81  | 51  | Q    | 113 | 71  | q    |
| 18  | 12  | Device control 2  | DC2  | CTRL-R    | 50  | 32  | 2     | 82  | 52  | R    | 114 | 72  | r    |
| 19  | 13  | Device control 3  | DC3  | CTRL-S    | 51  | 33  | 3     | 83  | 53  | S    | 115 | 73  | s    |
| 20  | 14  | Device control 4  | DC4  | CTRL-T    | 52  | 34  | 4     | 84  | 54  | T    | 116 | 74  | t    |
| 21  | 15  | Neg acknowledge   | NAK  | CTRL-U    | 53  | 35  | 5     | 85  | 55  | U    | 117 | 75  | u    |
| 22  | 16  | Synchronous idle  | SYN  | CTRL-V    | 54  | 36  | 6     | 86  | 56  | V    | 118 | 76  | v    |
| 23  | 17  | End of xmit block | ETB  | CTRL-W    | 55  | 37  | 7     | 87  | 57  | W    | 119 | 77  | w    |
| 24  | 18  | Cancel            | CAN  | CTRL-X    | 56  | 38  | 8     | 88  | 58  | X    | 120 | 78  | x    |
| 25  | 19  | End of medium     | EM   | CTRL-Y    | 57  | 39  | 9     | 89  | 59  | Y    | 121 | 79  | y    |
| 26  | 1A  | Substitute        | SUB  | CTRL-Z    | 58  | 3A  | :     | 90  | 5A  | Z    | 122 | 7A  | z    |
| 27  | 1B  | Escape            | ESC  | CTRL-[    | 59  | 3B  | :     | 91  | 5B  | [    | 123 | 7B  | {    |
| 28  | 1C  | File separator    | FS   | CTRL-\    | 60  | 3C  | <     | 92  | 5C  | \    | 124 | 7C  |      |
| 29  | 1D  | Group separator   | GS   | CTRL-]    | 61  | 3D  | =     | 93  | 5D  | ]    | 125 | 7D  | }    |
| 30  | 1E  | Record separator  | RS   | CTRL-^    | 62  | 3E  | >     | 94  | 5E  | ^    | 126 | 7E  | ~    |
| 31  | 1F  | Unit separator    | US   | CTRL-_    | 63  | 3F  | ?     | 95  | 5F  | _    | 127 | 7F  | DEL  |

## Resistor Color Code

The diagram illustrates two types of resistor color coding:

- 4-Band-Code:** A standard resistor with four color bands. The first three bands represent the resistance value (in ohms), and the fourth band represents the tolerance.
- 5-Band-Code:** A precision resistor with five color bands. The first four bands represent the resistance value, and the fifth band represents the tolerance.

**4-Band-Code:**

- Color bands: Green, Blue, Yellow, Brown
- Resistance:  $560 \text{ k}\Omega$
- Tolerance:  $\pm 5\%$
- Color code values: 560, 0, 0, 1 (representing  $560 \times 10^3 \Omega$ )
- Color tolerance: 2%, 5%, 10%

**5-Band-Code:**

- Color bands: Red, Orange, Yellow, Blue, Gold
- Resistance:  $237 \Omega$
- Tolerance:  $\pm 1\%$
- Color code values: 2, 3, 4, 6, 0.1 (representing  $237 \Omega$ )
- Color tolerance: 0.1%, 0.25%, 0.5%, 1%

**Color Code Reference Table:**

| COLOR  | 1 <sup>ST</sup> BAND | 2 <sup>ND</sup> BAND | 3 <sup>RD</sup> BAND | MULTIPLIER   | TOLERANCE        |
|--------|----------------------|----------------------|----------------------|--------------|------------------|
| Black  | 0                    | 0                    | 0                    | $1\Omega$    |                  |
| Brown  | 1                    | 1                    | 1                    | $10\Omega$   | $\pm 1\%$ (F)    |
| Red    | 2                    | 2                    | 2                    | $100\Omega$  | $\pm 2\%$ (G)    |
| Orange | 3                    | 3                    | 3                    | $1K\Omega$   |                  |
| Yellow | 4                    | 4                    | 4                    | $10K\Omega$  |                  |
| Green  | 5                    | 5                    | 5                    | $100K\Omega$ | $\pm 0.5\%$ (D)  |
| Blue   | 6                    | 6                    | 6                    | $1M\Omega$   | $\pm 0.25\%$ (C) |
| Violet | 7                    | 7                    | 7                    | $10M\Omega$  | $\pm 0.10\%$ (B) |
| Grey   | 8                    | 8                    | 8                    |              | $\pm 0.05\%$     |
| White  | 9                    | 9                    | 9                    |              |                  |
| Gold   |                      |                      |                      | $0.1\Omega$  | $\pm 5\%$ (J)    |
| Silver |                      |                      |                      | $0.01\Omega$ | $\pm 10\%$ (K)   |

# Final Words from Rui

If you've made it to this point, thank you so much!

Seriously, thanks for your interest in my projects. That's why I keep sharing my new projects. So you can read and hopefully have fun trying them.

Lastly, I hope you enjoyed this eBook as much as I loved creating it! If you did, please share one of my projects with your friends! Or you can send them the link below, so they can also download my free eBook:

<http://RandomNerdTutorials.com/download>

Thanks again and have fun with your projects,

Rui Santos.

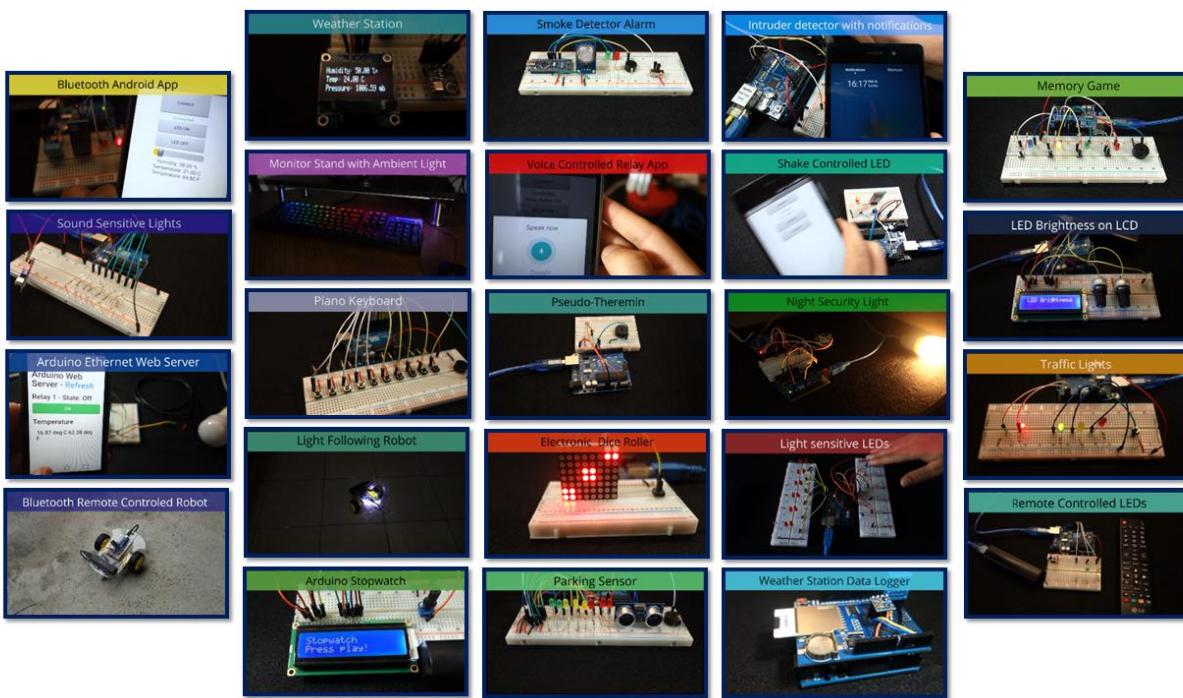
P.S. Make sure you visit my website to see the latest projects!

<http://RandomNerdTutorials.com>

# Download Arduino Step-by-step Projects Course

## Build 23 Projects

This Arduino Course is a compilation of 23 projects divided in 5 Modules that you can build by following clear step-by-step instructions with schematics and downloadable code.



## **Included Projects:**

### Module #1: Lights and LEDs

- Traffic Lights
- LED Brightness on LCD
- Light Sensitive LEDs
- Remote Controlled LEDs
- Monitor Stand with Ambient Light
- Night Security Light

## Module #2: Sensor Projects

- Smoke Detector Alarm
- Parking Sensor
- Arduino Weather Station (AWS) with Temperature, Humidity, Pressure, Date and Time
- Weather Station – Data Logging to Excel

## Module #3: Sound Projects

- Pseudo-Theremin
- Sound Sensitive Lights
- Piano Keyboard

## Module #4: Creating Applications

- Bluetooth Android App
- Voice Controlled Relay App
- Arduino Ethernet Web Server (Relay + Temperature)
- Shake Controlled LED
- Intruder Detector with Notifications

## Module #5: Miscellaneous Projects

- Arduino Stopwatch
- Electronic Dice Roller
- Memory Game
- Light following Robot
- Bluetooth Remote Controlled Robot