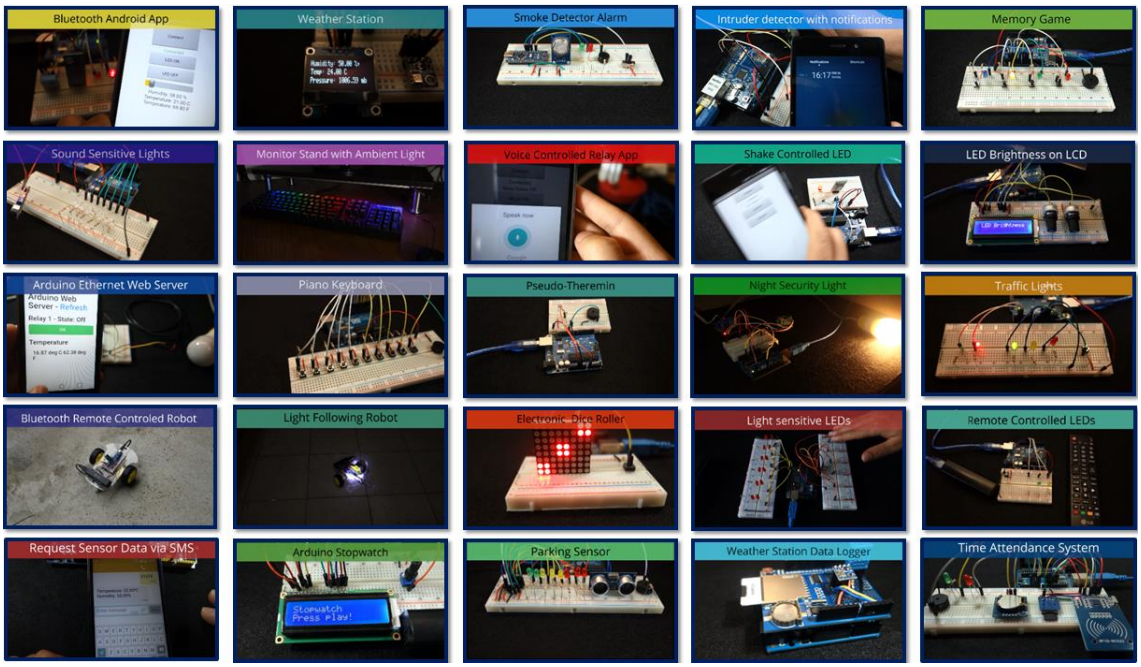
Arduino
Step-by-step Projects



Build 25 Projects!

Version 2.0

Created by Rui Santos and Sara Santos |Random Nerd Tutorials

Security Notice

This is the kind of thing we hate having to write about, but the evidence is clear: piracy for digital products is over all the internet.

For that reason we've taken certain steps to protect our intellectual property contained in this eBook.

This eBook contains hidden random strings of text that only apply to *your* specific eBook version that is unique to your email address. You probably won't see anything different, since those strings are hidden in this PDF. We apologize for having to do that – **but it means if someone were to share this eBook I know exactly who shared it and I can take further legal consequences.**

You cannot redistribute this eBook. **This eBook is for personal use and** is only available for purchase at:

- http://randomnerdtutorials.com/products
- https://rntlab.com

Please send an email to Rui Santos - hello@ruisantos.me, if you find this eBook anywhere else.

What we really want to say is thank you for purchasing this eBook and we hope you have fun with it!

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 authors (Rui Santos and Sara Santos) do not warrant that the information contained in this eBook is fully complete and shall not be responsible for any errors or omissions.

The authors (Rui Santos and Sara 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.

Throughout this eBook you will find some links and some of them are affiliate links. This means the authors (Rui Santos and Sara Santos) earn a small commission from each purchase with that link. Please understand that the authors have experience with all of these products, and they recommend them because they are useful, not because of the small commissions they make if you decide to buy something. Please do not spend any money on these products unless you feel you need them.

Other Helpful Links:

- Join Private Facebook Group
- Terms and Conditions

About the Authors

Hey there,

Thank you for reading the “Arduino Step-by-step Projects” course!

> *“I’m Rui Santos, an electrical engineer student and blogger. I have more than four years of experience teaching electronics and programming with the Random Nerd Tutorials blog. I’m also founder of RNTLab.com and author of BeagleBone For Dummies. Most of my projects and tutorials are related with Arduino, Home Automation, ESP8266, and Raspberry Pi.”*

> *“I’m Sara Santos, I have a master’s degree in Bioengineering and I’ve been working with Rui at Random Nerd Tutorials since 2015. My main tasks at Random Nerd tutorials include: recording and editing video, photo shooting, content creation, making projects, proofreading, etc… I also write and build courses with Rui.”*

Contents

Course Intro

This eBook is the offline version of the *Arduino Step-by-step Projects Course*. The course contains video and GIF images that cannot be displayed in this eBook. For the complete online version with video and animated images, please go to https://rntlab.com/arduino-projects/ and insert the password given when you bought the course.

Welcome to Arduino Step-by-step Projects Course – Build 25 Projects!

This Arduino Course is a compilation of 25 projects that you can build by following clear step-by-step instructions with schematics and downloadable code.

This is a completely practical course in which you're going to build several Arduino projects.

This course covers a wide variety of subjects from reading sensors, controlling outputs, building robots, data logging, home automation, android apps, and much more. You're going to use a wide variety of sensors and modules like: temperature and humidity sensor, bluetooth module, sound sensor, ultrasonic sensor, Ethernet shield, RFID reader, GSM shield and more.

This course is designed for people that like the Arduino and that like to build cool projects. The projects in this course have different levels of difficulty since beginner to advanced, however we put an effort to make all projects accessible to everyone.

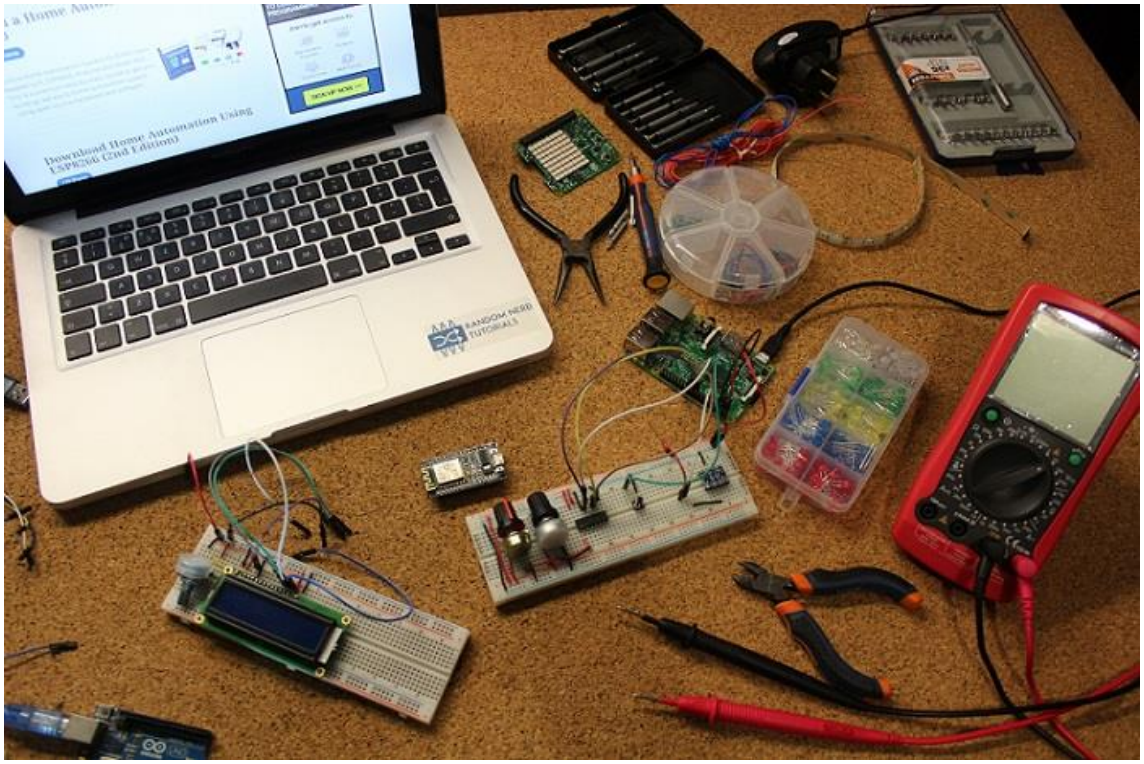
If you're just getting started with the Arduino don't worry: all projects have clear step-by-step instructions that anyone can follow along with schematics and downloadable code.

Download Source Code

Each project contains the source code and everything you need to complete the project. However, if you prefer you can click the button below and instantly download the source code for all projects.

DOWNLOAD CODE >>

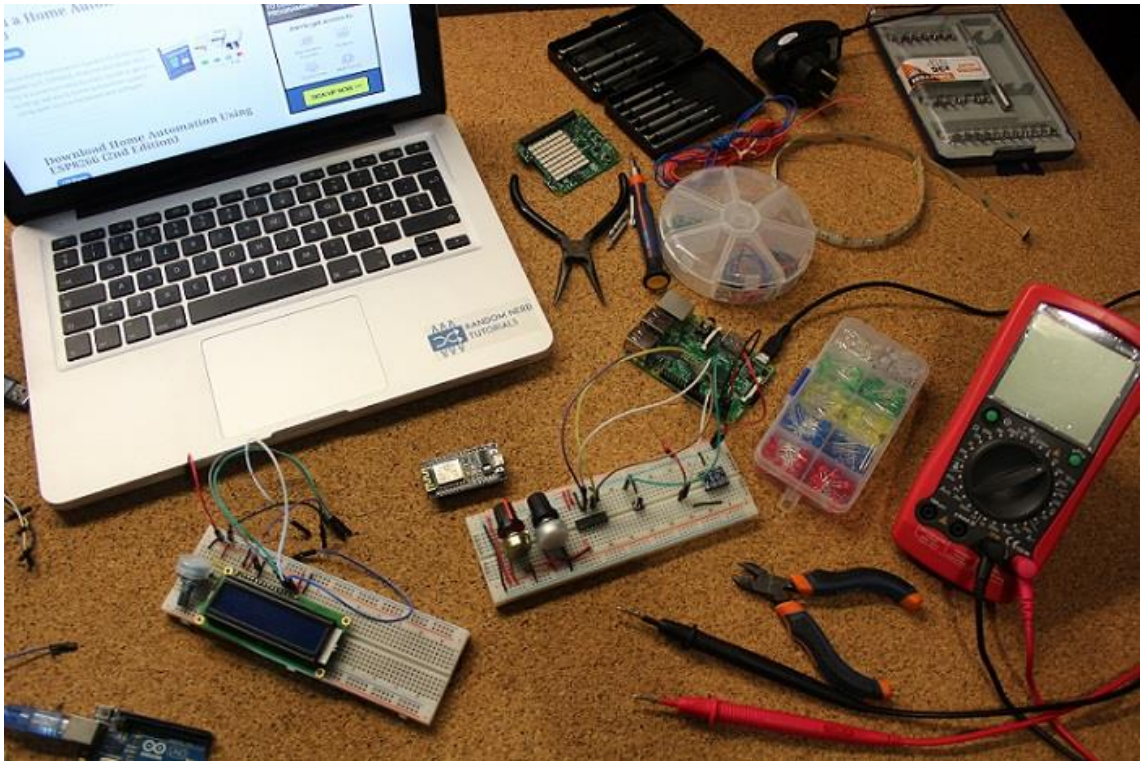
Setting Up Your Workspace



Setting Up Your Workspace

Making electronics projects is a lot of fun and it is nice to have a specific space to be our lab, a space where we have all we need to make our projects. It doesn't have to be a full room or a big space. But it's important to have "the" space! A space that is ours and where we can have all things we need to get the most of making electronics projects.

Your workspace needs a table or a flat surface big enough for your laptop and for building your circuits, with a comfortable chair and a good work lamp. You'll also need to keep your components and tools at hand, at least the ones you use more often.



The table where we build the circuits and make the projects is coated in cork, which is great for working with tools without damaging the table and is also great for the circuitry because cork is an insulator. However, it is not a very good space to place your computer, as the cork will "warm" your computer.

Once you have the space, you need to set it up! We'll show you the tools we think every electronics lab must have.

Getting the Tools

Here's a list with essential and optional tools for your electronics lab.

Multimeter (essential)

A multimeter is a measurement tool that is absolutely necessary. It helps you troubleshooting your circuits. Many times, your circuits may not work because of faulty wires. Checking the connections with the multimeter can save a lot of time.

If you're looking to buy one, I truly recommend you an auto-range multimeter. The auto-ranging is a great advantage, because it saves you of the hassle of having to know which range of value the electrical characteristic you're measuring falls under.

My multimeter is the one in the following figure. It doesn't have auto-range, it measures resistance, current, voltage and checks for continuity. It works great.

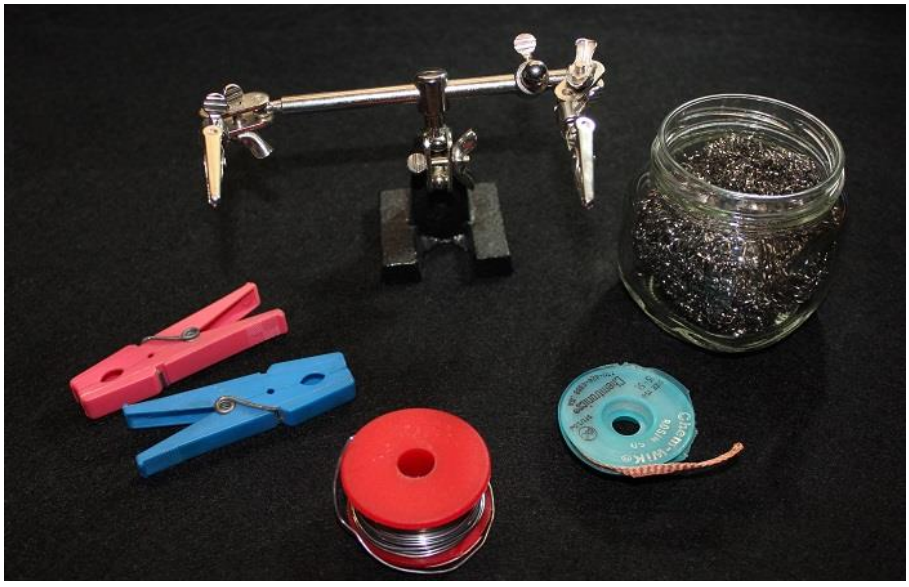
Soldering station (essential)

Having a soldering station in an electronics lab is absolutely necessary, even if you don't usually do permanent circuits. Many electronics components don't come with headers or breadboard-friendly pins attached. In these cases, you'll need some soldering work. If you don't do a lot of soldering a simple soldering station will do the work. I actually started with a simple and cheap soldering station– it did well its job – and later I decided to buy a more professional one – the one at the figure below.

Soldering accessories (essential)

Besides the soldering station you also need some soldering accessories. You need solder and a cleaning sponge. I use a glass cup with steel wool inside and it cleans the soldering tip really well.

Once you start soldering, you'll find it difficult to hang, keep the cables together and solder at the same time. So, a helping hand can help you in the process. I also use clothes pegs quite frequently.



Voltage supplier (optional)

A voltage supplier is not essential in simple and beginner projects. But once you moved to more advanced projects, a voltage supplier may help you test your circuits. If you're a beginner you don't need to get a voltage supplier now. In fact, I only get mine some years after starting with electronics.

Plastic boxes (essential)

Organization can save you a lot of time searching for the components and tools. I have several types of plastic boxes to store components and tools. Old plastic boxes can also be useful to store stuff. I have closed boxes where I save the components I don't use frequently. For storing and organizing the components I use frequently I use a cabinet organizer.

Cabinet organizer (optional)

A cabinet organizer is essential so that you have all your components organized. There are a lot of cabinet organizers for electronics components with drawers that you can label, so that you always know where everything is.

I have two cabinets as shown in the picture below, and I absolutely love them. I store my sensors, buttons, potentiometers, resistors and much more.

Wire stripper (essential)

A wire stripper is essential in an electronics lab as you'll have to strip cables many times. You can get a simple wire stripper, but I absolutely recommend you a self-adjusting wire stripper/cutter instead. It is very easy to use, adapts to a wide variety of cables and works perfectly.

With this wire stripper all you have to do is set the strip length, insert a wire, squeeze the handles, and you're finished.

Needle-nose pliers and wire cutter (essential)

A needle-nose pliers is always handy. You should absolutely get one. Wire cutters are also absolutely essential. The following figure shows the pliers and wire cutter I use more often.

Hot glue gun (optional)

Hot glue is great. It is very handy for a wide variety of applications if you want to make something stick together. In electronics, hot glue is perfect to fix your circuit into a surface or to attach cables together. As the glue is an insulating material, it can help you protect your circuitry.

My hot glue gun is an old one, but is works great in my projects. I always use it with an old plate below for not spreading glue over the table.

If you're curious, check this post about hot glue tips and tricks for your electronics projects.

Precision Screwdriver Set (optional)

There are components with really small screws and with specific shapes and so, at some point in your projects, you'll need a precision screwdriver. I recommend you getting a set take comes with extension bits because you'll need all of them sooner or later.

Tweezers (optional)

Tweezers are very useful to manipulate the small electronics components. The ones in the picture below have an extremely small and pointed tip, which is perfect for electronics.

Rotary Tool and accessories (optional)

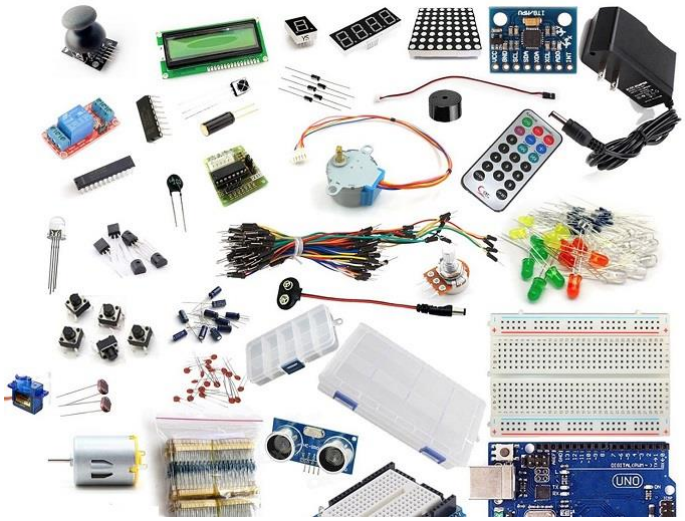
This tool is great if you like to give your projects good finished looks. This tool is good for working with acrylic, plastic, wood and much more. I have the one in the following figure. It was quite cheap. However, I don't use it very often.

Assortment kits

Besides the previous tools, it is also handy to get some components assortment kits with components that you'll use often. Here's what I recommend.

Arduino starter kit

Arduino starter kits usually come with everything you need to get started doing projects with the Arduino. They usually include the Arduino board, electronics components such as buttons, motors, potentiometers, resistors, breadboard and jumper wires.



Jumper wires

In Arduino projects, you'll use mostly male-to-male jumper wires. Some sensors require male-to-female jumper wires and sometimes it may also be handy to have female-to-female jumper wires. So, get a bunch of each type.

You should also get simple electronics wires that are useful for a wide variety of applications as you can easily cut and solder them.

LEDs assortment

LEDs are awesome and are always a good fit in every electronics projects. Get an assortment with several colors and sizes.

Resistors assortment

You'll need resistors in most of your electronics projects. Get a 1/4 watt resistors assortment. These will meet most of your needs.

Breadboards

Once you start doing electronics projects, you'll realize that you'll never have enough breadboards for your projects.

Potentiometers and pushbuttons

Potentiometers and pushbuttons are a must. You'll use them often.

Varied Arduino sensors and modules

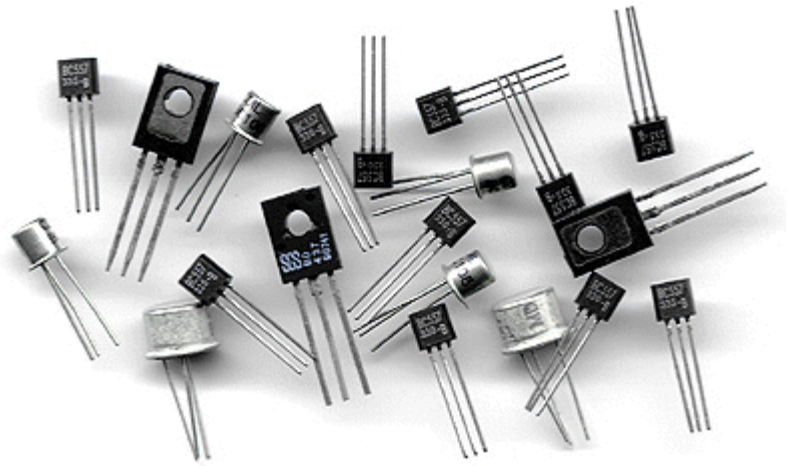
There are a wide variety of Arduino compatible sensors and modules. You can get some of them for a very low price. Consider buying some, especially the ones you find more interesting. I have a post with 21 Arduino compatible sensors that you can get for less than $2. You can check it here.

Prototype PCB

If you like to transform your breadboard circuit into permanent circuits, you should get some prototyping PCBs.

Transistors assortment

Get a transistors assortment. These come with the most widely used transistors.



Wrapping up

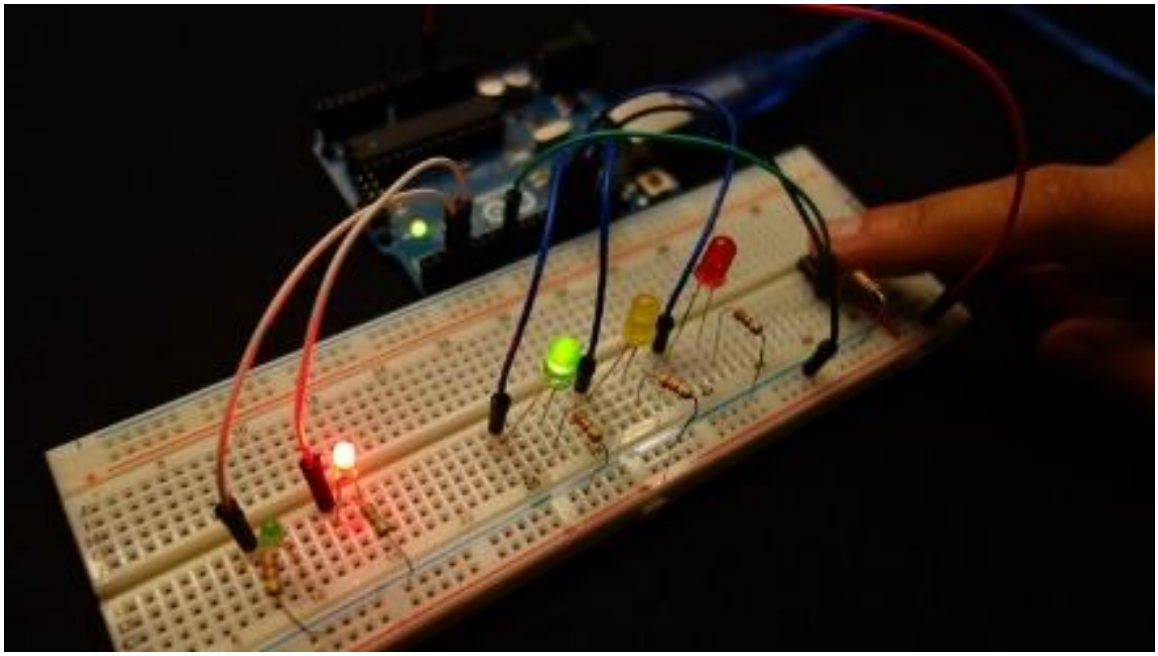
Well, it's a lot of stuff for setting up your electronics lab. However, you don't need to get overwhelmed. Electronics components' kits usually come with several component units at a very low price. You can start with a small Arduino starter kit and then, as go you through the projects, you can buy more components.

So, are you excited for setting up your electronics lab?

We hope you found these tips useful.

Have fun!

Traffic Lights



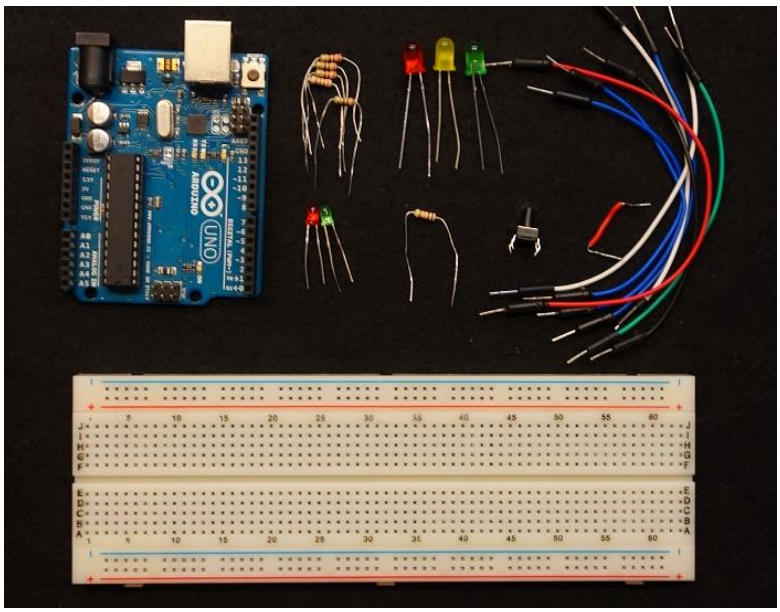
Traffic Lights

Level: Beginner – Time: 25 minutes

In this project you're going to build a traffic lights system:

- There are 3 LEDs with different colors (green, yellow and red) to mimic the traffic lights for the cars;
- There are 2 LEDs with different colors (green and red) to mimic the traffic lights for the pedestrians;
- There is a pushbutton to mimic the ones in the pedestrians' traffic lights.

Parts required



Grab all the needed components for this project. If you don't have any of them, just click on the link to buy it on eBay.

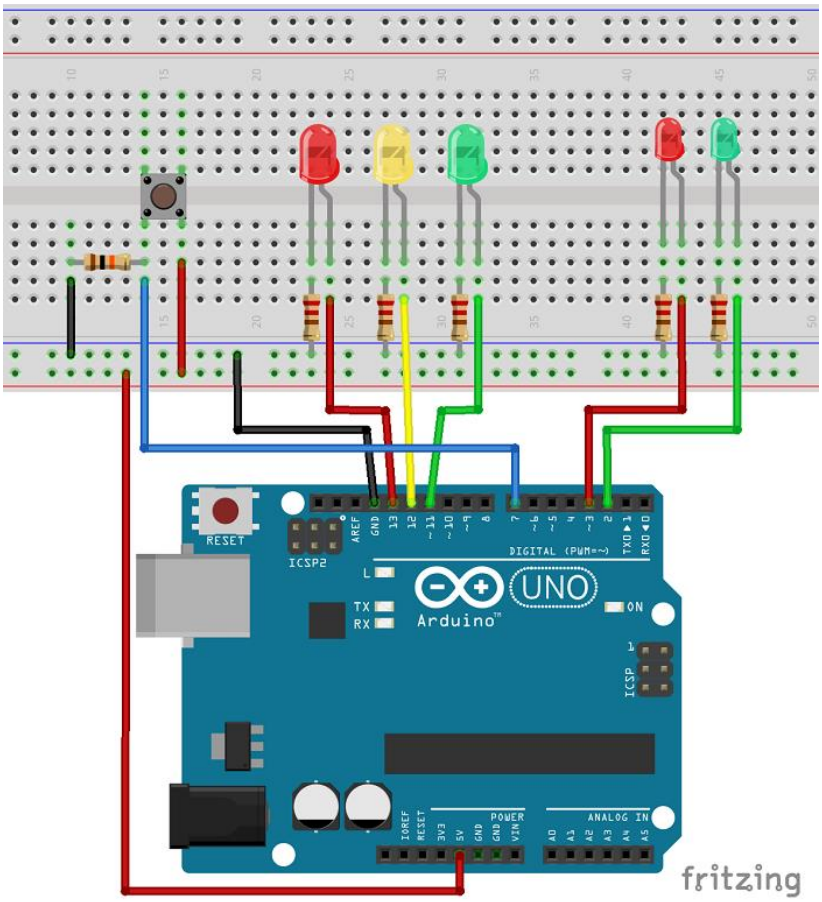
Figure	Name	eBay
	Breadboard	http://ebay.to/21bEojM
	Arduino UNO	http://ebay.to/1SQda0R
	3x 5mm LED (1x red, 1x yellow, 1x green)	http://ebay.to/20H2Oyy

	2x 3mm LED (1x red, 1x green)	http://ebay.to/20H2Oyy
	5x 220Ohm Resistor	http://ebay.to/1KsMYFP
	1x 10kOhm Resistor	http://ebay.to/1KsMYFP
	1x pushbutton	http://ebay.to/211vcRv
	Jumper Wires	http://ebay.to/1PXeaJz

I'm using LEDs of different sizes but if you don't have LEDs of different sizes, it is ok. The project still works.

Schematics

Assemble all the parts by following the schematics below.



Code

You don't need any library for this code. The code is very simple. Here's some tips to better understand what's going on.

- The car light is always green, and so the pedestrian light is always red unless someone presses the button.
- When someone presses the button here's what happens:
  - The car light changes to yellow and then to red
  - The pedestrian light changes to green
  - The lights are in this state for a while (in the code this time is the variable `crossTime`)
  - The pedestrian green light flashes and goes to red
  - The car light changes from red to green

All this actions will be inside the `changeLights()` function. Every time you want to change the lights, you just need to call the `changeLights()` function.

Download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure you have the right board and COM port selected.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

int redCar = 13;
int yellowCar = 12;
int greenCar = 11;
int greenPed = 2;
int redPed = 3;
int button = 7;
int crossTime = 2000;
unsigned long changeTime;

void setup() {
  // initialize timer
  changeTime = millis();
  // here we are initializing our pins as outputs
  pinMode(redCar, OUTPUT);
  pinMode(yellowCar, OUTPUT);
  pinMode(greenCar, OUTPUT);
  pinMode(redPed, OUTPUT);
  pinMode(greenPed, OUTPUT);
  pinMode(button, INPUT);
  //turn on the green light
  digitalWrite(greenCar, HIGH);
  digitalWrite(redPed, HIGH);
  digitalWrite(redCar, LOW);
  digitalWrite(yellowCar, LOW);
  digitalWrite(greenPed, LOW);
```

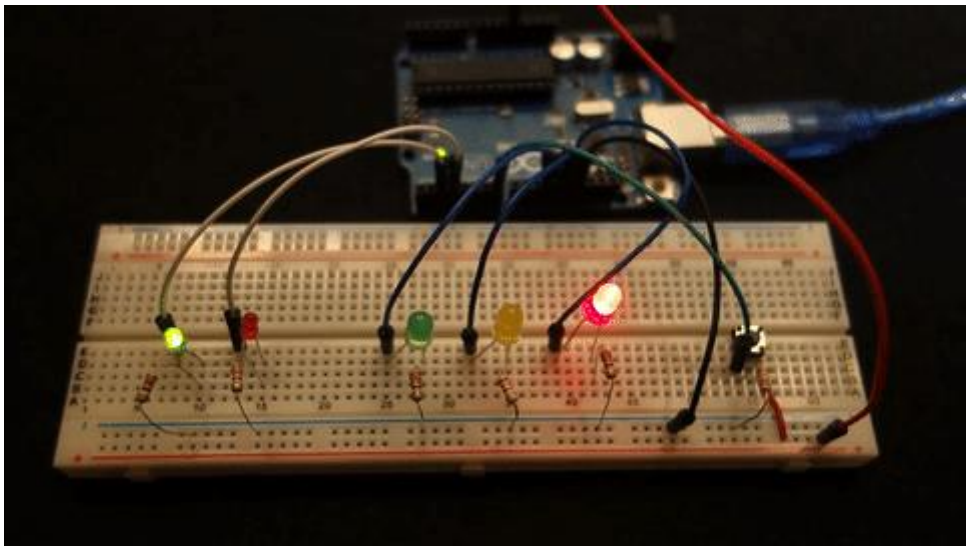
```
  Serial.begin(9600);
}

void loop() {
  // this variable will tell us if the button is pressed
  int state = digitalRead(button);
  Serial.println(state);
  // if the button is pressed and if it has passed 5 seconds since
last button press
  if (state == HIGH && (millis() - changeTime) > 5000) {
    //call the function to change the lights
    changeLights();
  }
}

void changeLights() {
  digitalWrite(greenCar, LOW);    // the green LED will turn off
  digitalWrite(yellowCar, HIGH); // the yellow LED will turn on for
2 second
  delay(2000);

  digitalWrite(yellowCar, LOW); // the yellow LED will turn off
  digitalWrite(redCar, HIGH); // the red LED will turn on for 5
seconds

  digitalWrite(redPed, LOW);
  digitalWrite(greenPed, HIGH);
  delay(crossTime);

  // flash the ped green
  for (int x=0; x<10; x++) {
    digitalWrite(greenPed, LOW);
    delay(100);
    digitalWrite(greenPed, HIGH);
    delay(100);
  }
  digitalWrite(greenPed, LOW);
  digitalWrite(redCar, LOW);
  digitalWrite(redPed, HIGH);
  digitalWrite(greenCar, HIGH);

  changeTime = millis();
 }
```

Source code:

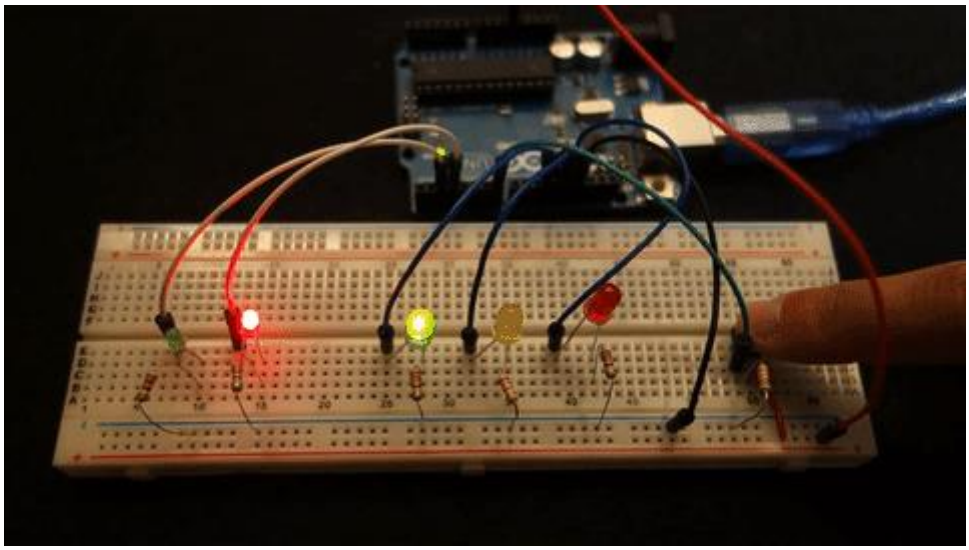
https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/4_traffic_lights.ino

Demonstration

Here's the traffic lights in action. When you press the button the light for the cars changes from green to red, and the pedestrian light changes from red to green.



After the `crosstime` , the pedestrian green led flashes and changes to red. The light for the cars changes from red to green.

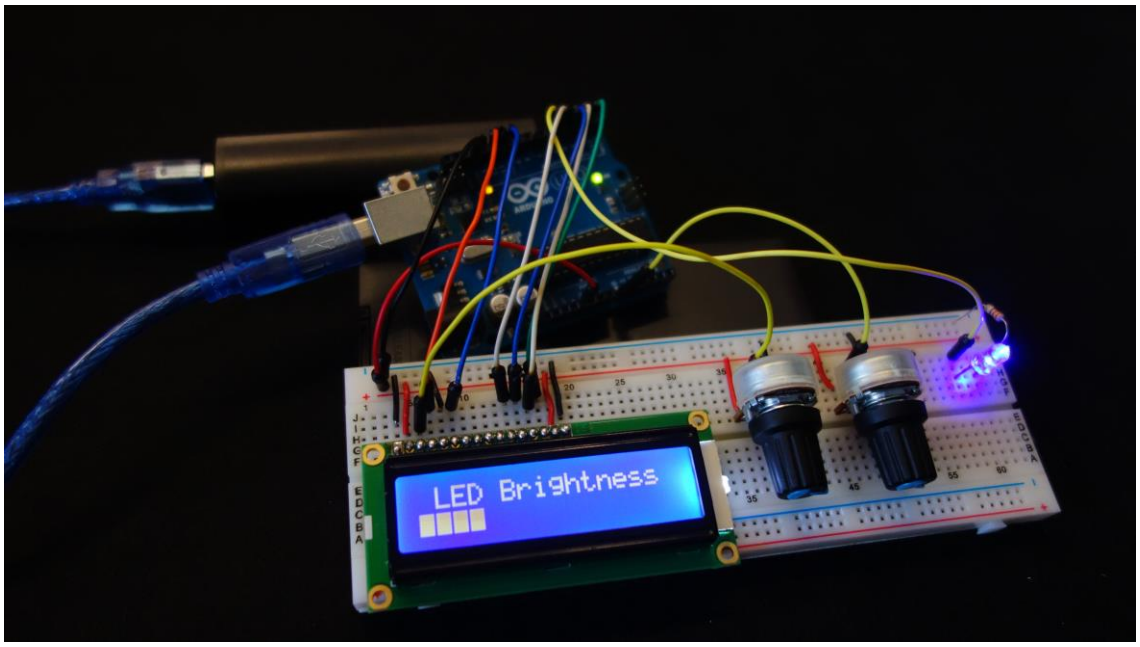


Wrapping up

If you're starting with the Arduino, a good exercise is to change the value of some variables like `crossTime` and `changeTime` and see what happens.

If you want something a little bit more challenging, try to mimic what happens in a junction, with several lights for both the cars and the pedestrians.

LED brightness on LCD



Display the LED brightness on a 16×2 LCD

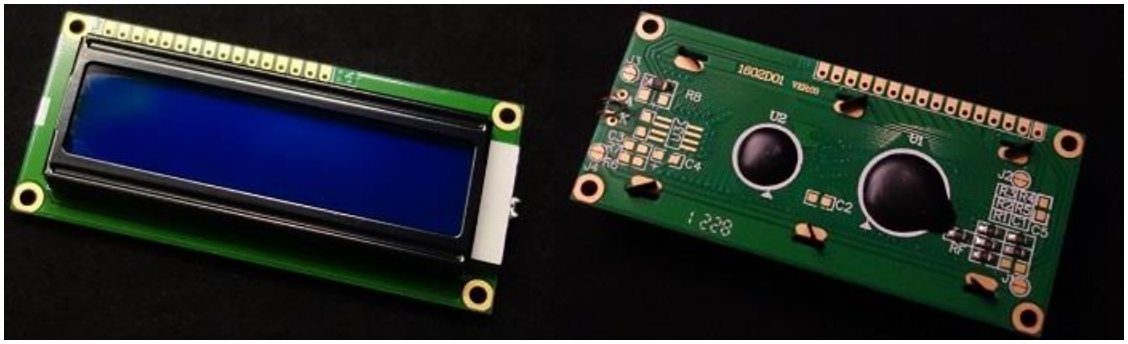
Level: Beginner – Time: 25 minutes

This is a beginner project where you'll use a 16×2 LCD to show the LED brightness. This is a good project for you to get started with this kind of display.

Shortly, in this project we will control the LED brightness using a potentiometer. The LED brightness will be displayed on the LCD screen using a progress bar.

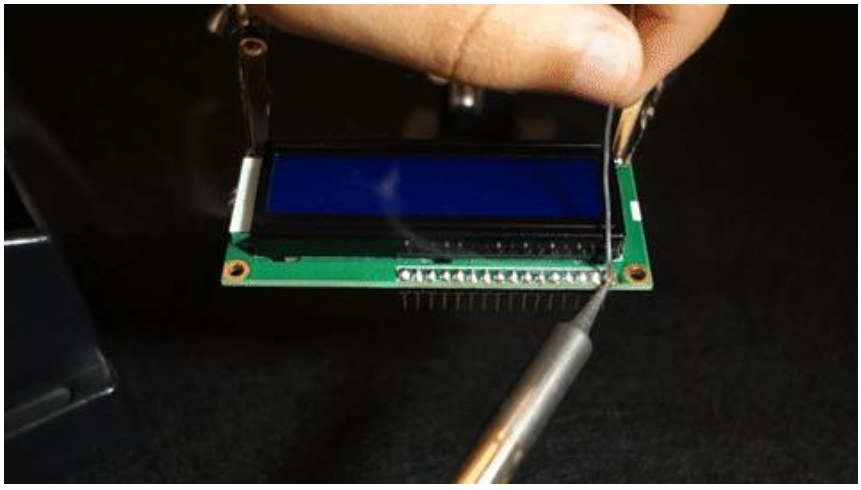
16×2 Liquid Crystal Display (LCD)

The 16×2 LCD is the one in the following figure (front and the back view).

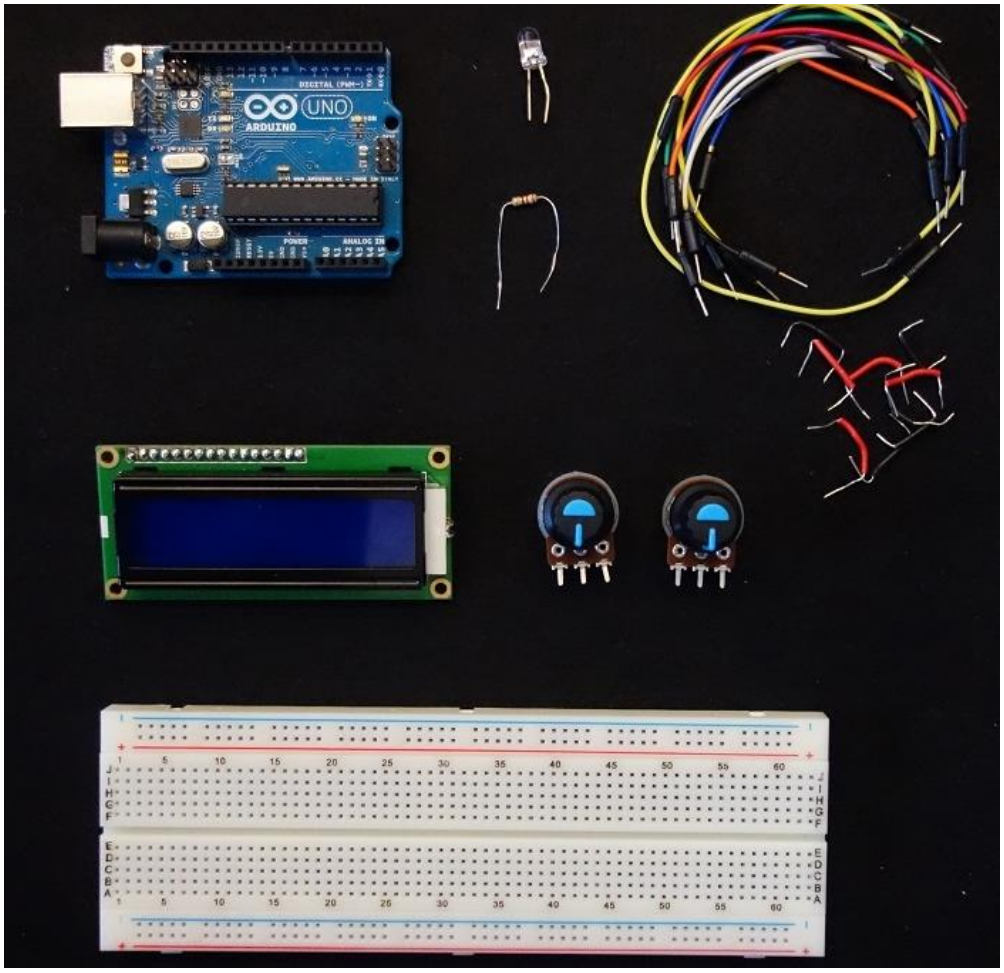


The 16×2 LCD is commonly used in several circuits and devices. This LCD has 2 rows, and each row can display 16 characters. It also has LED backlight that can be adjusted accordingly to the contrast between the background and the characters that you want.

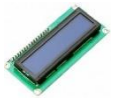
When you buy a 16×2 LCD, usually it doesn't come with breadboard friendly pins. So, you may need some headers. Solder the headers to your LCD, and it is ready to use.



Parts required



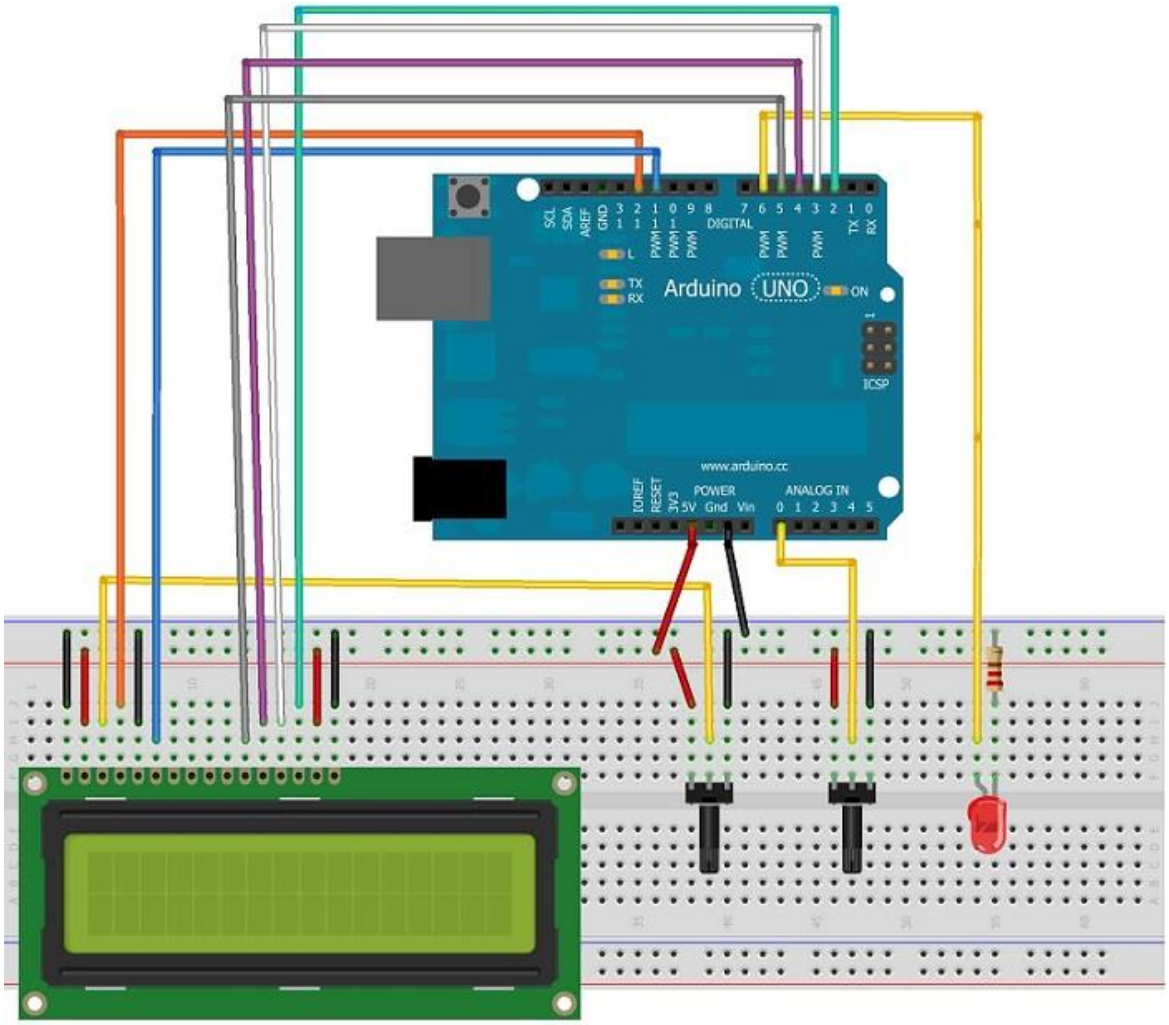
Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	16x2 LCD display (Hitachi HD44780 driver)	http://ebay.to/2cgnKeW
	Breadboard	http://ebay.to/21bEojM
	Arduino UNO	http://ebay.to/1SQda0R
	LED	http://ebay.to/20H2Oyy

	220Ohm Resistor	http://ebay.to/1KsMYFP
	Jumper Wires	http://ebay.to/1PXeaJz
	2x 10kOhm Potentiometers	http://ebay.to/1PUefOb

Schematics

Assemble all the parts by following the schematics below.



Note that you need two potentiometers. The one at the left is connected to the LCD to adjust the backlight. The one at the right is connected to the Arduino to control the LED brightness.

Code

To use the LCD with the Arduino, you don't need to install any library. The library used, `LiquidCrystal.h`, is installed in the Arduino IDE by default.

Download or copy the following code to your Arduino IDE and upload it to your Arduino board. Make sure that you have the right board and COM port selected.

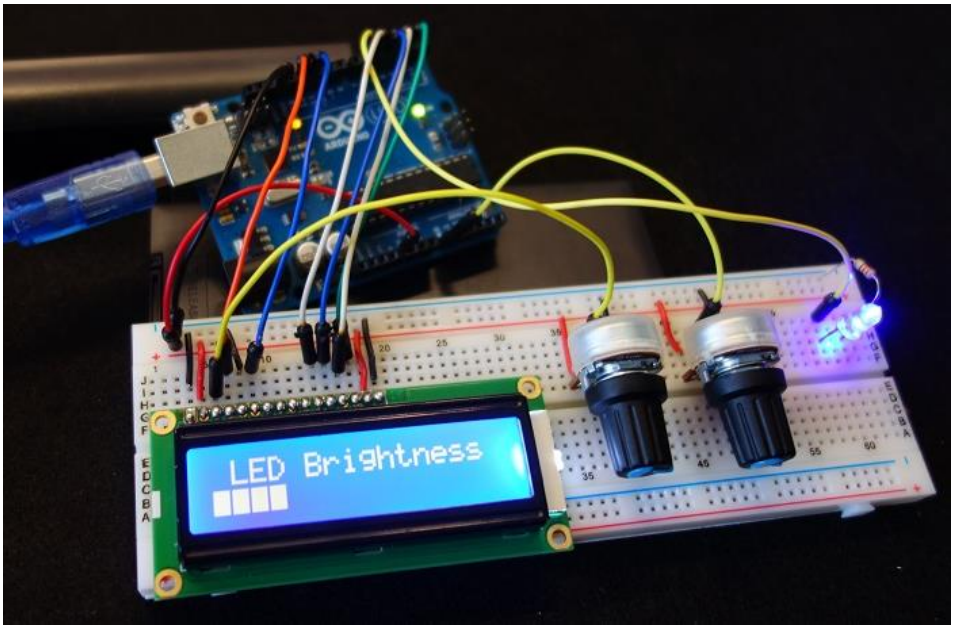
```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
*/

// include the LCD library
#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,
  B11111,
  B11111,
  B11111,
};

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 bightness
  analogWrite(ledPin, brightness);
  // turns the brighness into a percentage for the bar
  pBari=map(brightness, 0, 255, 0, 17);
  //prints the progress bar
  for (i=0; i<pBari; i++)
  {
    lcd.setCursor(i, 1);
    lcd.write(byte(0));
  }
  // delays 750 ms
  delay(750);
}
```

Source code:

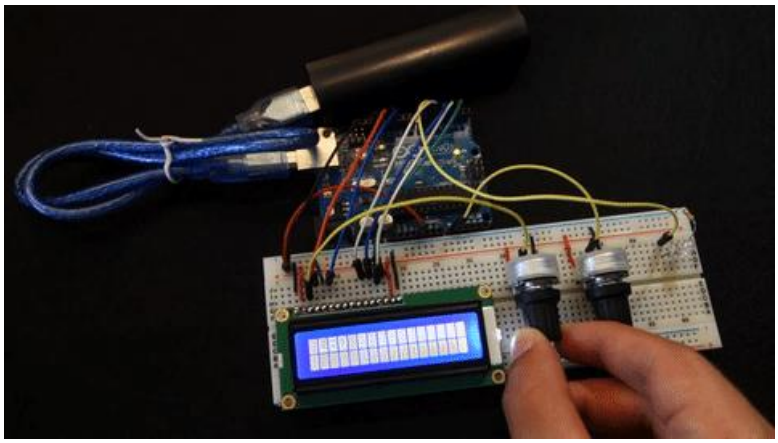
https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/2_led_brightness_on_a_16x2_lcd_display.ino

Here's how your circuit looks like after assembling all the parts and uploading the code.

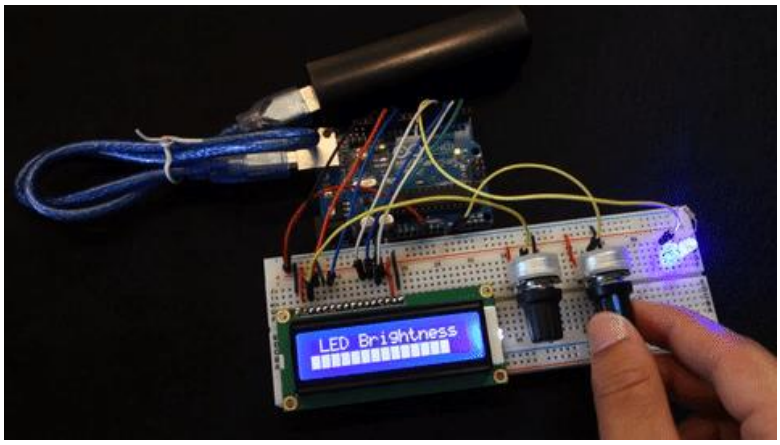


Demonstration

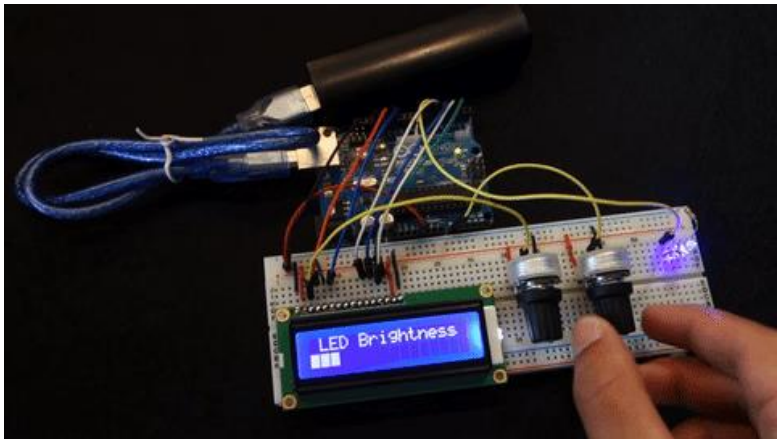
In the end, you can adjust the LCD backlight.



If you increase the LED brightness, the progress bar increases.



And if you decrease the LED brightness, the progress bar decreases accordingly:

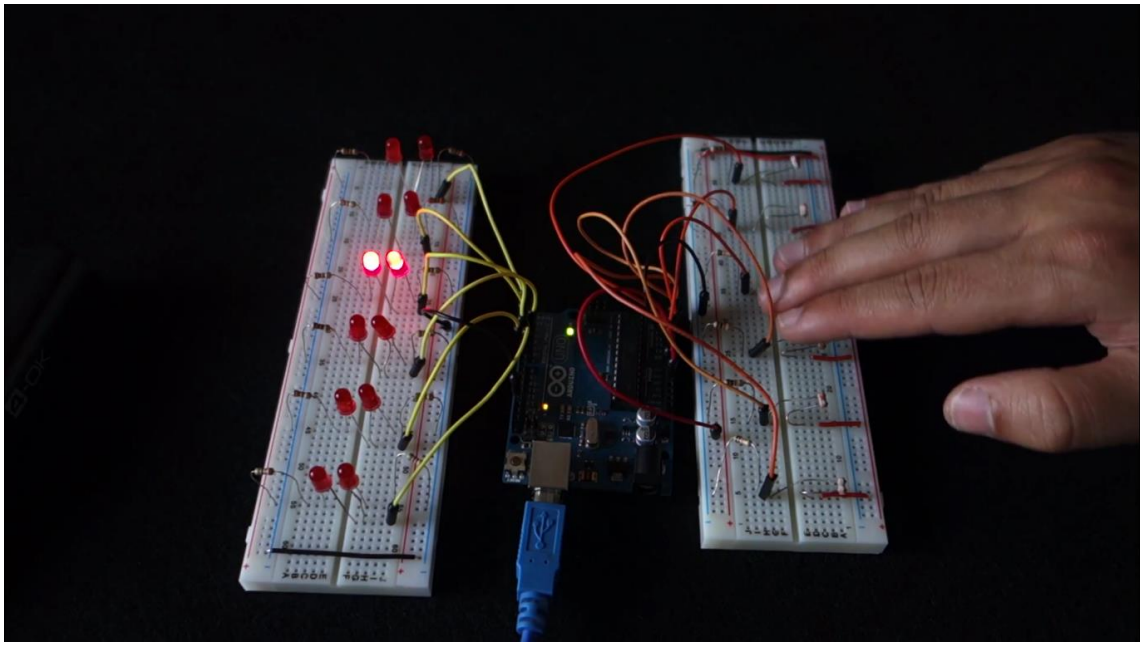


Wrapping up

This is just a basic example on how to use this kind of display.

Now, the idea is to modify the sketch and use the display in other projects.

Light Sensitive LEDs



Light Sensitive LEDs

Level: Beginner – Time: 45 minutes

In this project you'll build a circuit with some LEDs that are controlled accordingly to the light intensity. You'll need one breadboard with LEDs and another breadboard with photoresistors.

The idea of this project is to place your hand over the photoresistors to light up the LEDs. This project is divided into two parts:

- You'll adjust the sensitivity of your photoresistor
- You'll build the final circuit

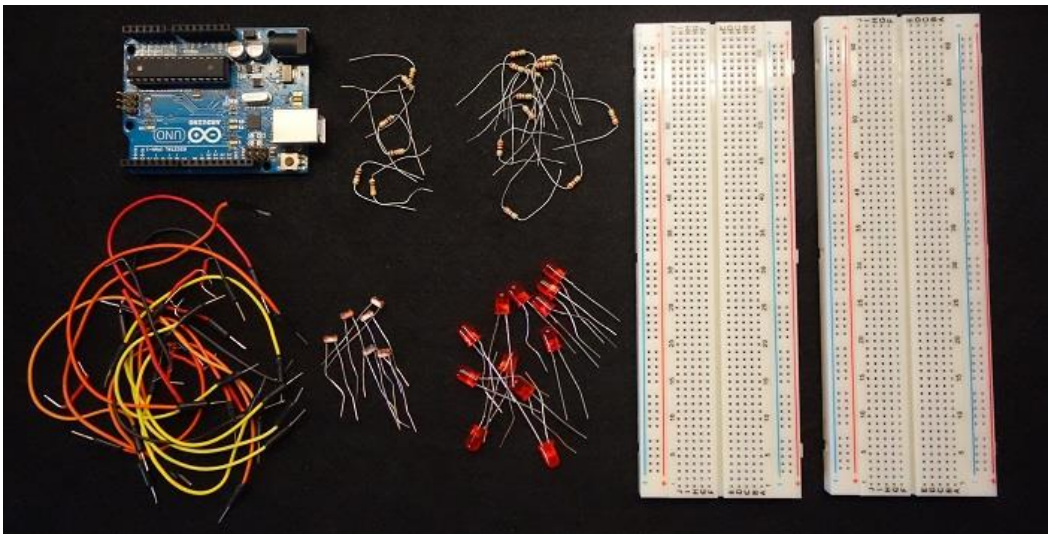
Photoresistor

The photoresistor is the component in the following figure.

It is a Light-Dependent Resistor (LDR). This means that the resistance across the photoresistor varies accordingly to the light intensity. Here's how the resistance varies:

- with increasing light, the less the resistance (more current flowing)
- with decreasing light, the more the resistance (less current flowing)

Parts required



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

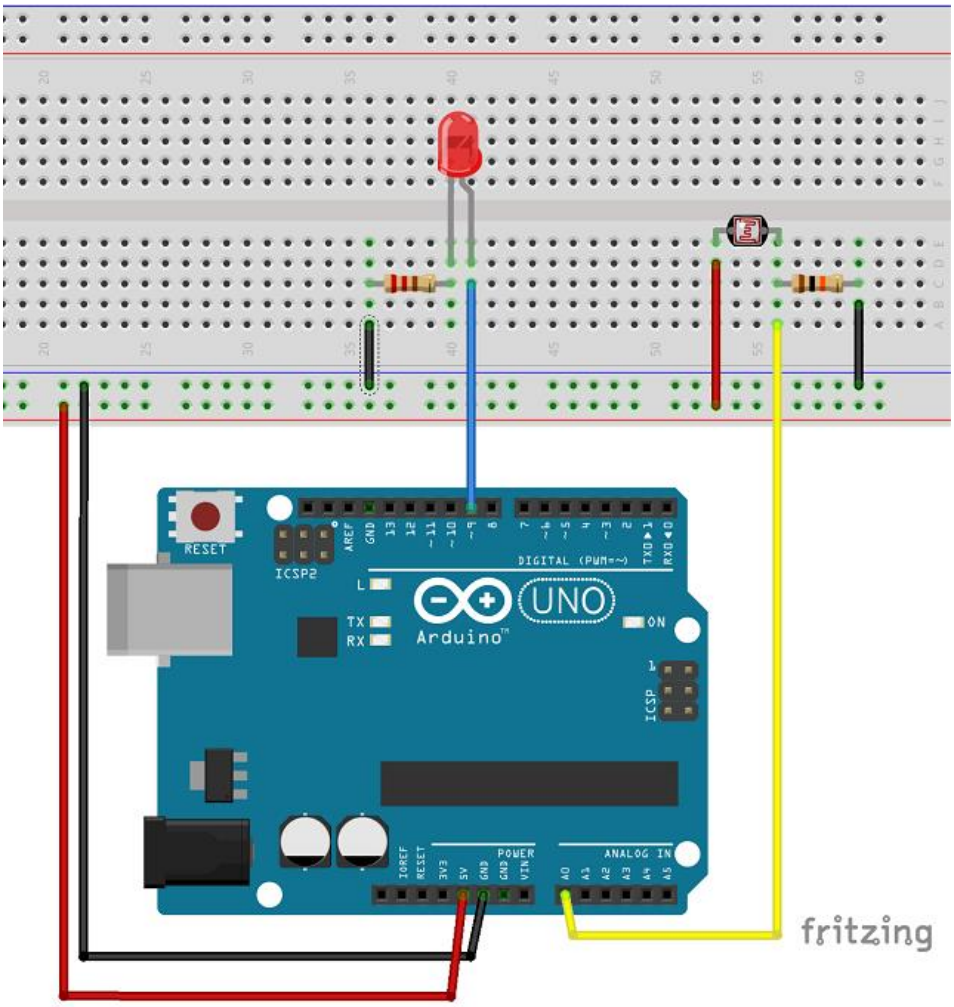
Figure	Name	eBay
	2x Breadboard	http://ebay.to/21bEojM
	Arduino UNO	http://ebay.to/1SQda0R
	12x LED	http://ebay.to/20H2Oyy
	6x Photoresistors	http://ebay.to/2bMWpSV
	12x 220Ohm Resistor	http://ebay.to/1KsMYFP
	6x 10kOhm Resistor	http://ebay.to/1KsMYFP
	Jumper Wires	http://ebay.to/1PXeaJz

Determine the threshold value

Before actually building the final circuit, you'll need to define a threshold value. This means that you need to select the value of brightness that turns the LEDs on.

You need to do this because your ambient light is different than mine and varies accordingly to the lighting of your office.

For that, assemble the following circuit with one LED and one photoresistor.



Code

Download or copy the following code to your Arduino IDE, and upload it to your Arduino board.

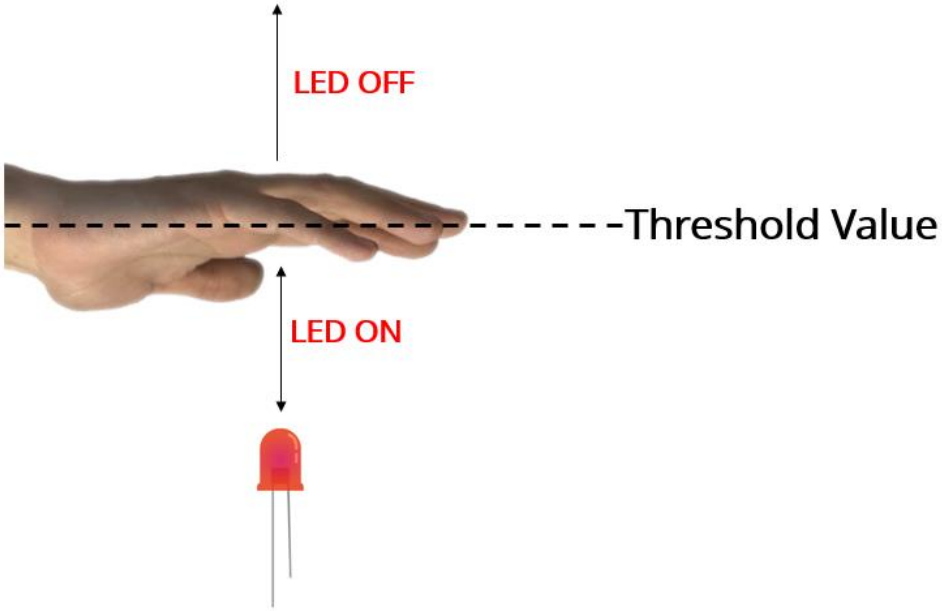
```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
*/

// You might need to change the threshold value to find a suitable
value
```

```
// to turn the LEDs on and off when you move your hand over the
photoresistor
int threshold = 600;

// Connect the LED to digital pin 9
int led = 9;

// The photoresistor lead is connected to analog pin 0
int sensor = A0;
// stores the analog reading value
int sensorValue = 0;

void setup() {
  // sets the LED as an output
  pinMode(led, OUTPUT);
  // sets the photoresistor as an input
  pinMode(sensor, INPUT);
  // starts the serial communication at a baud rate of 9600
  Serial.begin(9600);
}

void loop(){
  // Reads the current photoresistor value
  sensorValue = analogRead(sensor);
  // if the value is below a certain threshold it turns the LED on,
otherwise the LED remains off
  if (sensorValue < threshold) {
      digitalWrite(led, HIGH);
  }
  else {
      digitalWrite(led,LOW);
  }
  // prints the current readings in the Arduino IDE serial monitor
  Serial.print ("Sensor reading: ");
  Serial.println(sensorValue);
  delay(50);
}
```

Source code

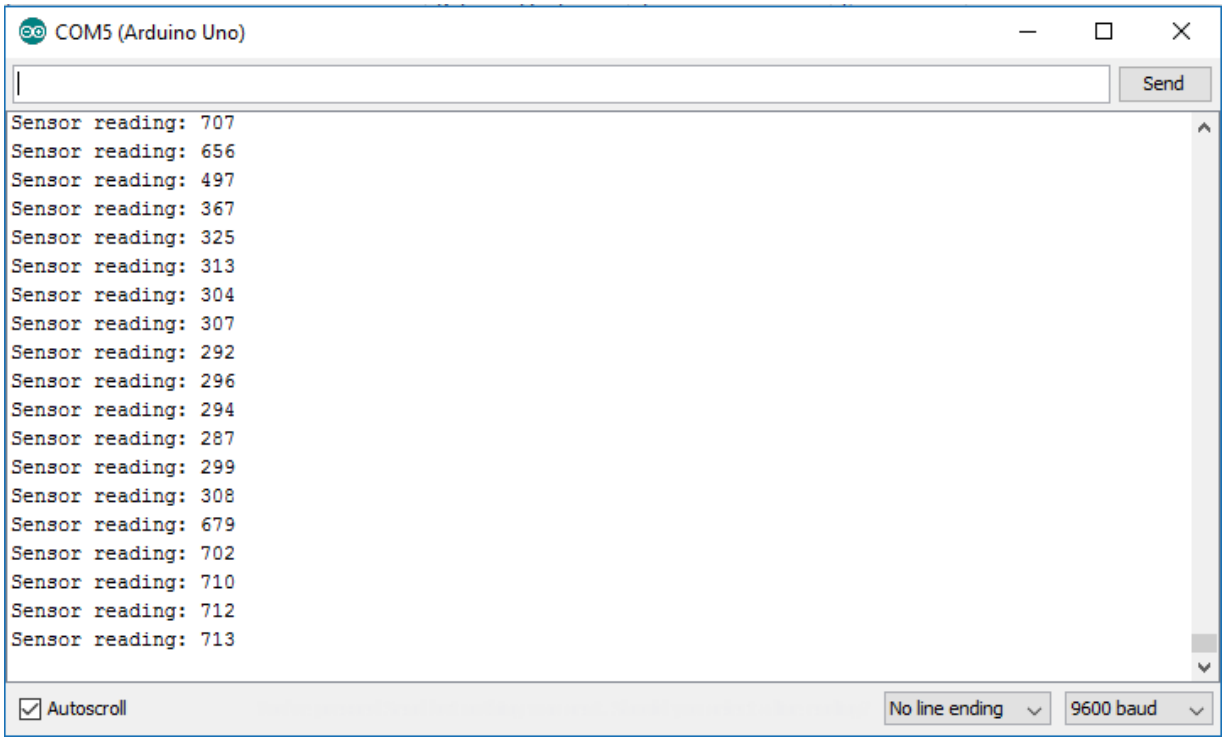
https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/1a_ldr_tester.ino

This code changes the LED state from OFF to ON, when the photoresistor reaches a certain value of light intensity. That value is set to 600 at the moment. However, you should change that value accordingly to the light of your surroundings.

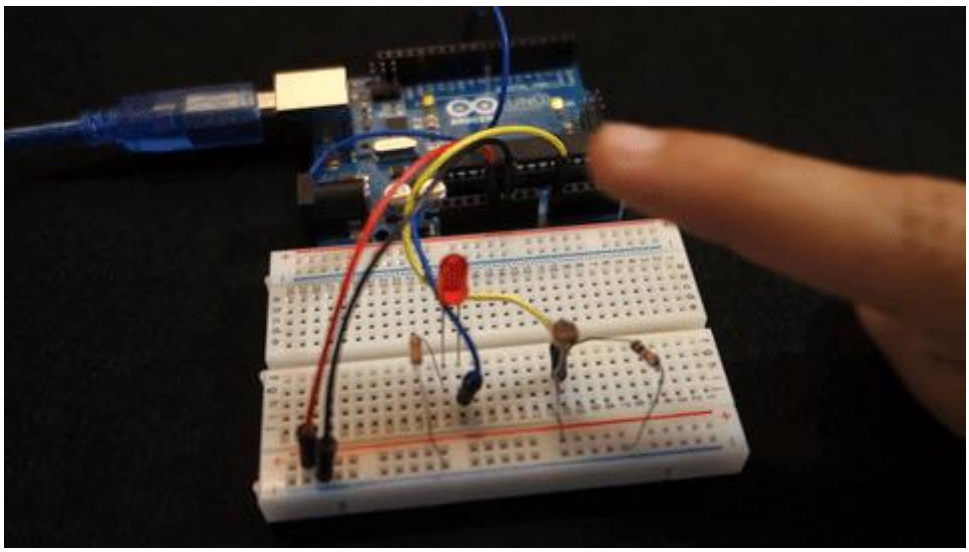


Open the serial monitor at a baud rate of 9600.

You'll see the light intensity values of your surroundings. Place your hand over the photoresistor and the light intensity values will decrease.



Define the distance between the photoresistor and your hand at which your LED should light up. Check what's the value you read on the serial monitor and save that number, because that's your threshold value. In my case, I set the threshold value to 600.



Change the `threshold` variable in the code and test the circuit until you find the right one.

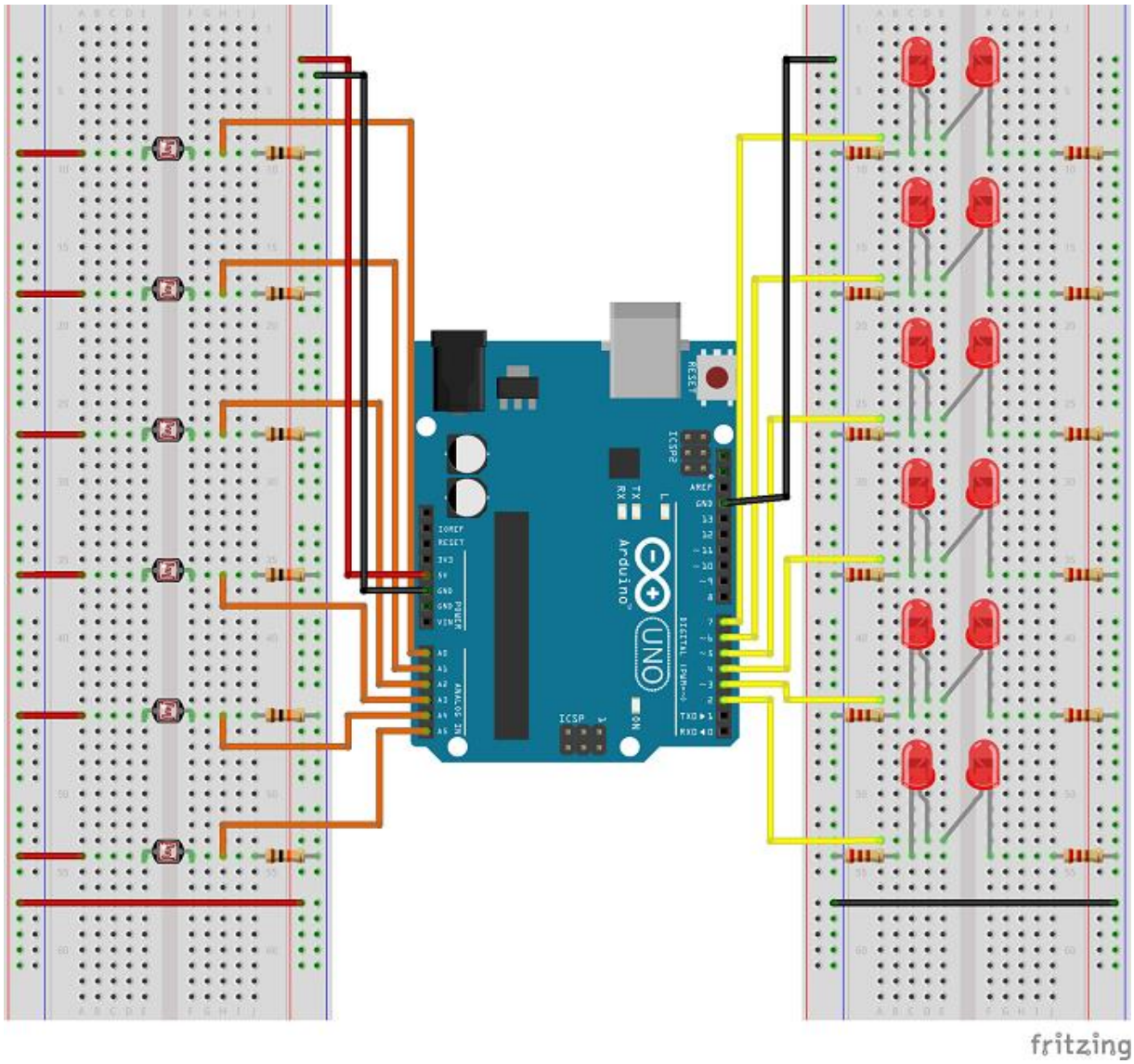
```
int threshold = 600;
```

Assembling the final circuit

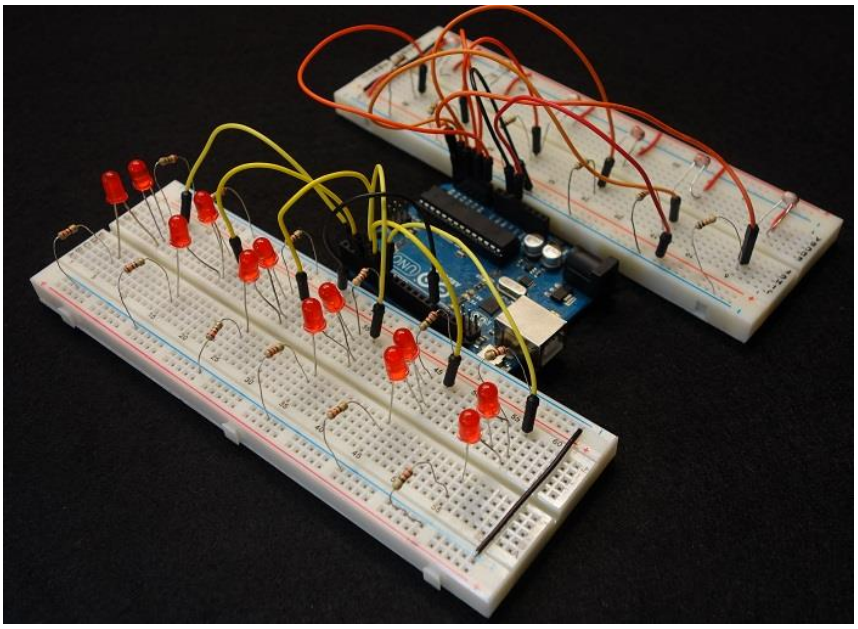
The final circuit is very similar to the previous one. Instead of having just one set of LED + photoresistor, you'll have 12 LEDs for a cool visual effect.

Schematics

Assemble all the parts as in the schematics below. I really recommend that you follow this layout in order to have the same visual effect. Note that I've placed LEDs in parallel in order to control 2 LEDs per photoresistor.



Your circuit should look similar to this one:



Code

The sketch for this project is very straightforward. You don't need to install any library.

Download or copy the following code to your Arduino IDE, and upload it to your Arduino Board. Make sure that you have the right board and COM port selected.

Don't forget to change the threshold variable with your threshold value.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
*/

// You might need to change the threshold value to find a suitable
value
// to turn the LEDs on and off when you move your hand over the
photoresistor
int threshold = 600;

// Connect the LEDs from digital pin 7 to digital pin 2
int led1 = 7;
int led2 = 6;
int led3 = 5;
int led4 = 4;
int led5 = 3;
int led6 = 2;

// Connect photoresistor leads are connected from analog 0 to
analog 5
int sensor1 = A0;
int sensor2 = A1;
int sensor3 = A2;
int sensor4 = A3;
int sensor5 = A4;
int sensor6 = A5;

// stores the analog reading values for each photoresistor
int sensorValue1 = 0;
int sensorValue2 = 0;
int sensorValue3 = 0;
int sensorValue4 = 0;
int sensorValue5 = 0;
int sensorValue6 = 0;

void setup() {
  // sets the LEDs as an output
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  // sets the photoresistors as an input
  pinMode(sensor1, INPUT);
  pinMode(sensor2, INPUT);
```

```
  pinMode(sensor3, INPUT);
  pinMode(sensor4, INPUT);
  pinMode(sensor5, INPUT);
  pinMode(sensor6, INPUT);
  // starts the serial communication at a baud rate of 9600
  // Serial.begin(9600);
}

void loop(){
  // Reads the current photoresistor value
  sensorValue1 = analogRead(sensor1);
  // if the value is below a certain threshold it turns the LED on,
otherwise the LED remains off
  // This process is repeated for all photoresistors and LEDs
  if (sensorValue1 < threshold) {
      digitalWrite(led1, HIGH);
  }
  else {
      digitalWrite(led1,LOW);
  }
  sensorValue2 = analogRead(sensor2);
  if (sensorValue2 < threshold) {
      digitalWrite(led2, HIGH);
  }
  else {
      digitalWrite(led2,LOW);
  }

  sensorValue3 = analogRead(sensor3);
  if (sensorValue3 < threshold) {
      digitalWrite(led3, HIGH);
  }
  else {
      digitalWrite(led3,LOW);
  }

  sensorValue4 = analogRead(sensor4);
  if (sensorValue4 < threshold) {
      digitalWrite(led4, HIGH);
  }
  else {
      digitalWrite(led4,LOW);
  }
  sensorValue5 = analogRead(sensor5);
  if (sensorValue5 < threshold) {
      digitalWrite(led5, HIGH);
  }
  else {
      digitalWrite(led5,LOW);
  }
  sensorValue6 = analogRead(sensor6);
  if (sensorValue6 < threshold) {
      digitalWrite(led6, HIGH);
  }
  else {
      digitalWrite(led6,LOW);
  }
  // You can uncomment for debugging purporses
```

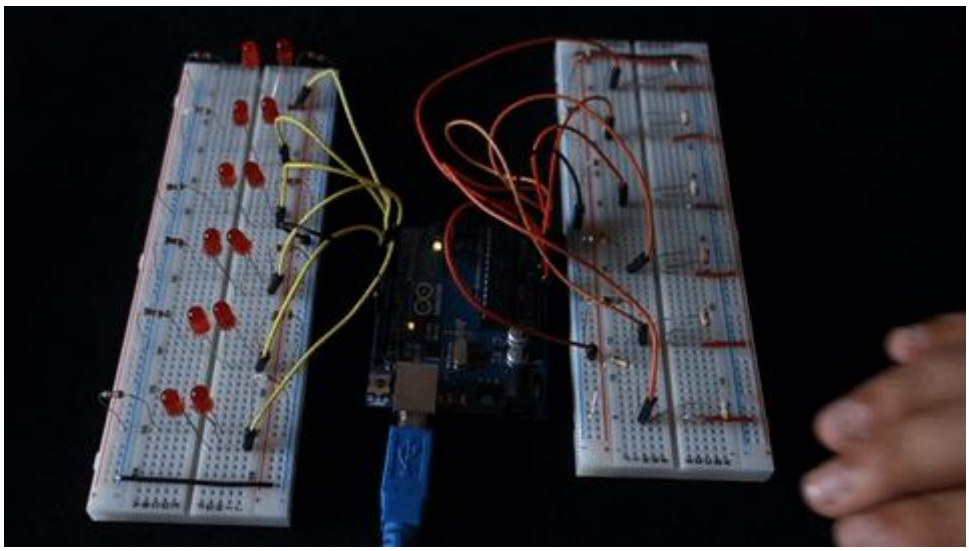
```
  //Serial.print ("Sensor reading: ");
  //Serial.println(sensorValue1);
  delay(50);
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/1b_light_sensitive_leds.ino

Demonstration

At the end, you'll have a cool visual effect.



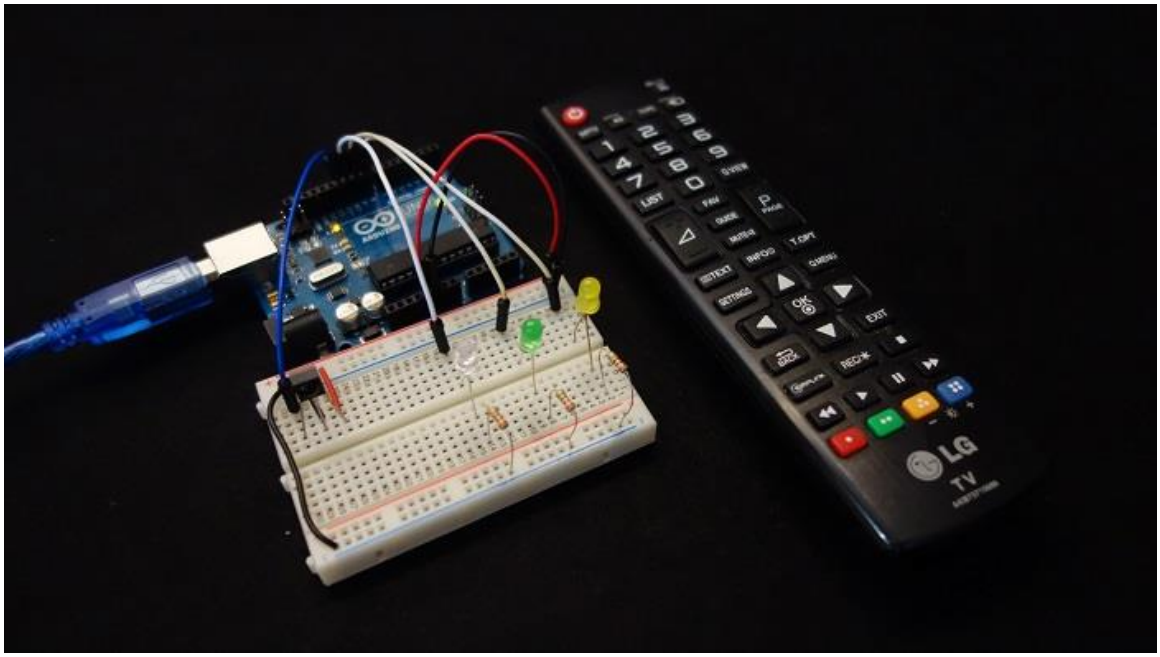
Wrapping up

This is a simple project with a pretty straightforward code, yet with a very cool outcome.

You can make several changes to the code or to the circuit to have different variations of this project.

For example, try to change the LED brightness accordingly to the light intensity of your surroundings. This way, you can create automatic LEDs that light up when the night comes or when it's dark.

Control LEDs with IR Remote



Control LEDs with IR Remote

Level: Beginner – Time: 45 minutes

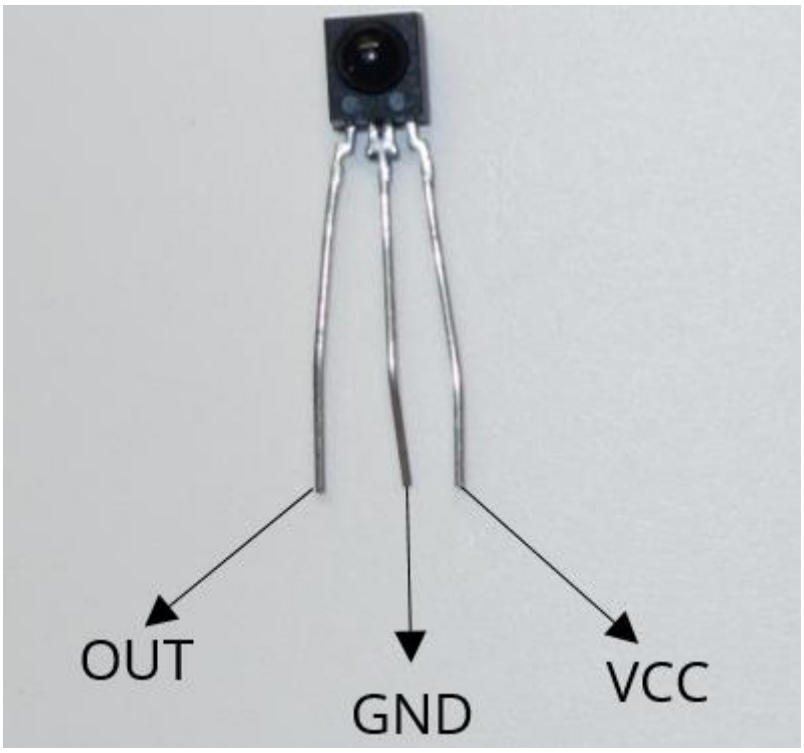
In this project you'll use an infrared (IR) receiver to control 3 LEDs with a remote control. You'll turn them on and off with the buttons of your remote control. This is a good beginner project to get you introduced to the infrared receiver.

This project is divided into two parts:

- You'll decode the IR signals transmitted by your remote control
- You'll use that info to perform a task with your Arduino (control 3 LEDs)

Infrared (IR) Receiver

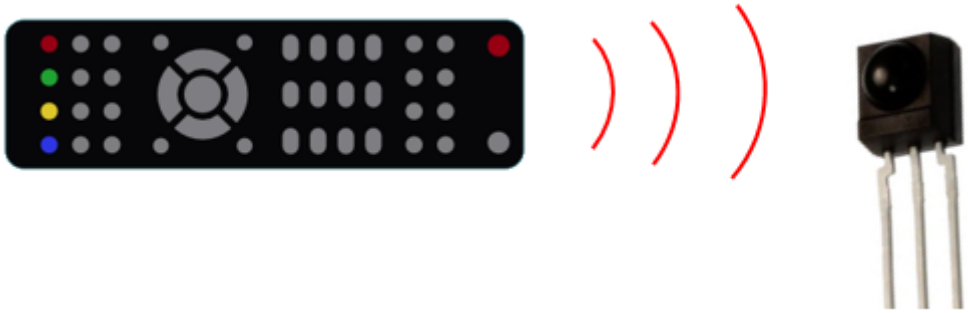
The infrared receiver is the component in the figure below. This is the TSOP4838.



The TSOP4838 has three pins:

- OUT (OUTPUT)
- GND (Ground)
- VCC (Voltage source)

When you press your remote control, it sends infrared modulated signals. These signals contain information that your receiver collects.

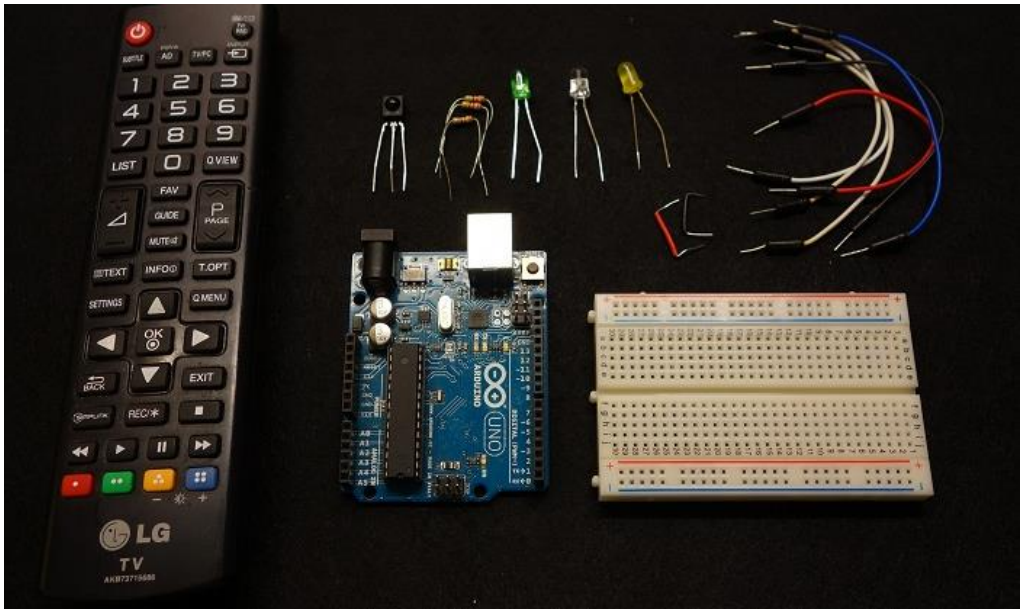


Each button sends specific information. So, we can assign that information to a specific button.

Parts required

For this project you'll need a remote control. If you don't have any remote control at home, you need to buy one.

Any remote control will do, even if it is a very old one that you don't use. Actually, if it is an old one that you have laying around that's perfect. It's a new purpose for something old.



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Breadboard	http://ebay.to/21bEojM

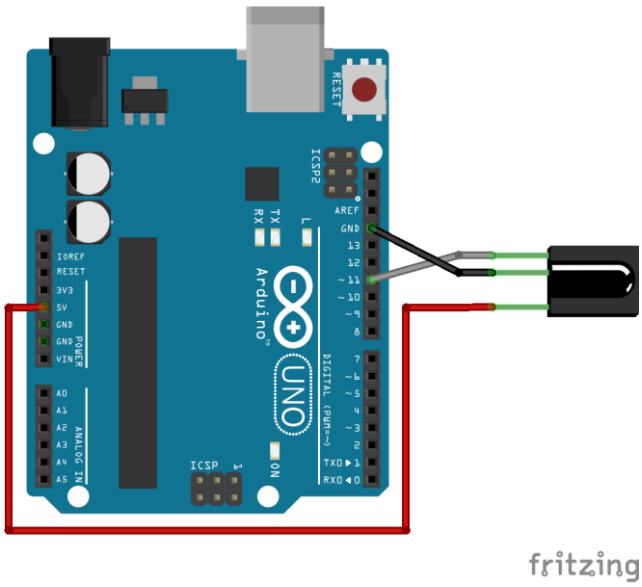
	Arduino UNO	http://ebay.to/1SQda0R
	Infrared Receiver (TSOP4838)	http://ebay.to/2cqMHHz
	3x LED	http://ebay.to/20H2Oyy
	3x 220Ohm Resistor	http://ebay.to/1KsMYFP
	Jumper Wires	http://ebay.to/1PXeaJz
	1x Remote control	http://ebay.to/2bOAUnB

Decode the IR signals

In this part of the project you need to decode the IR signals associated with each button.

Schematics

Connect the IR receiver accordingly to the schematics below.



Code

To control the IR receiver, you need to install the IRremote Library in the Arduino IDE.

Installing the IRremote library

- Click here to download the IRremote library. You should have a .zip folder in your Downloads
- Unzip the .zip folder and you should get IRremote-master folder
- Rename your folder from ~~IRremote-master~~ to IRremote
- Move the IRremote folder to your Arduino IDE installation libraries folder
- Finally, re-open your Arduino IDE

Note: this library conflicts with the RobotIRremote library. So, in order to use it, you should move temporarily the RobotIRremote library out of the Arduino IDE libraries folder. If you don't do this, the code will not compile!

Download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure that you have the right board and COM port selected.

```
/*
 * IRremote: IRrecvDemo - demonstrates receiving IR codes with
IRrecv
 * An IR detector/demodulator must be connected to the input
RECV_PIN.
 * Version 0.1 July, 2009
 * Copyright 2009 Ken Shirriff
 * http://arcfn.com
 */

#include <IRremote.h>

int RECV_PIN = 11;

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
  delay(100);
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/3a_ir_receiver_demo.ino

Open the serial monitor at a baud rate of 9600.

In this project want to control 3 LEDs. Choose 6 buttons for the following tasks:

- LED1 – ON
- LED1 – OFF
- LED2 – ON
- LED2 – OFF
- LED3 – ON
- LED3 – OFF

Press, for example, the button number 1. You shoud see a code on the serial monitor. Press the same button several times to make sure you have the right code for that button. If you see something like `FFFFFFFF` ignore it, it's trash.

Do the same for the other buttons.

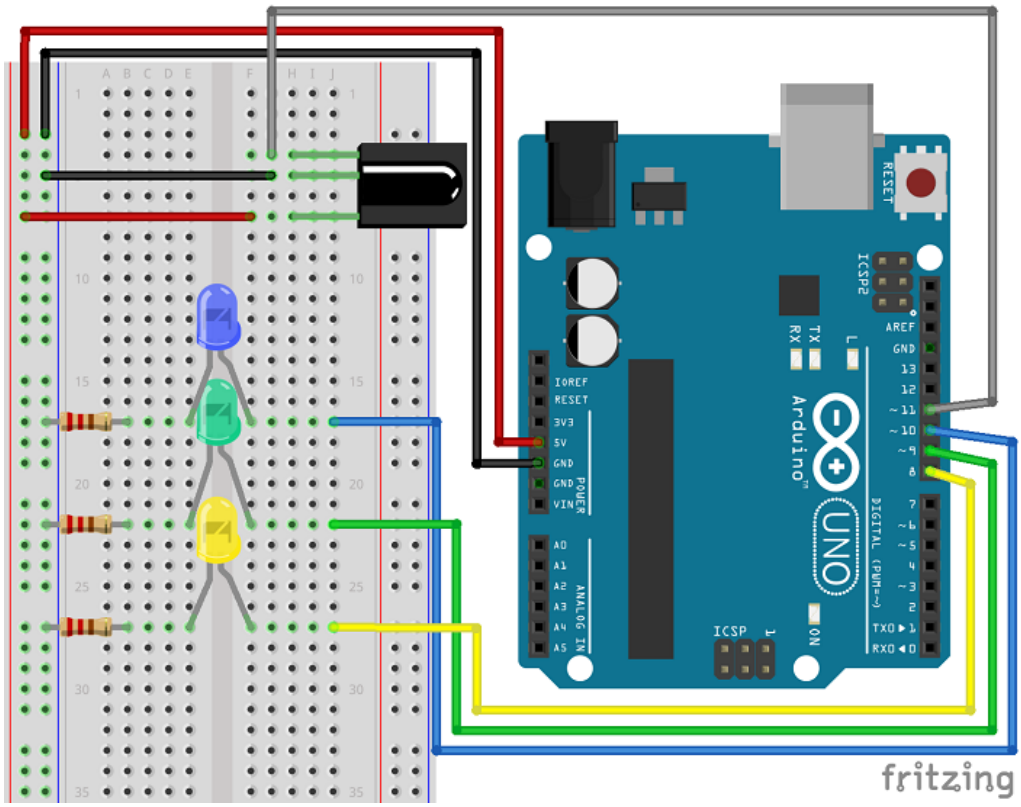
Write down the code associated with each button, because you'll need that information later.

Building the final circuit

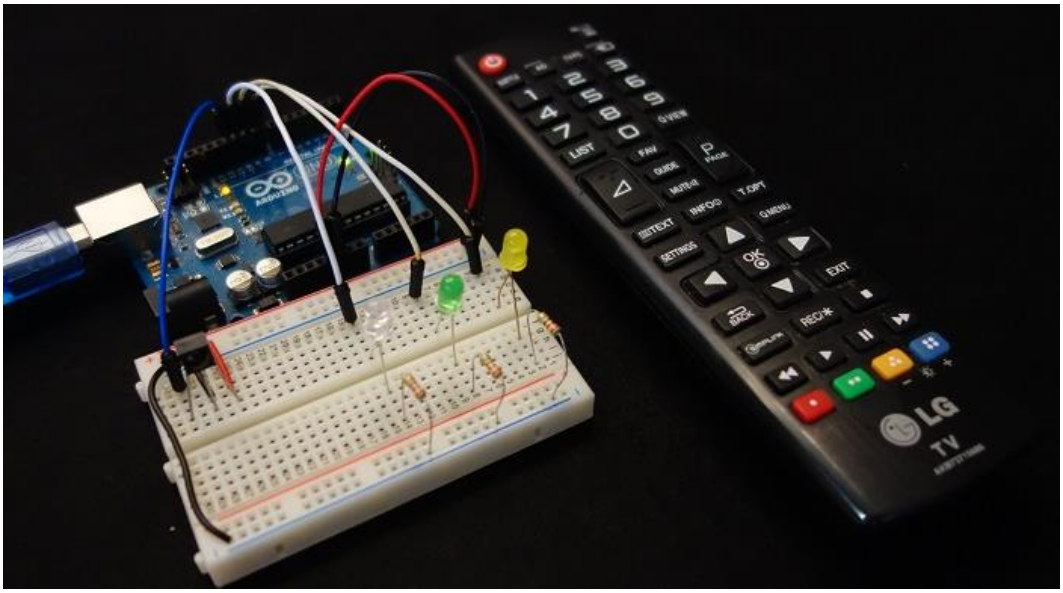
In this part, you'll build the circuit with three LEDs that will be controlled using your remote.

Schematics

Assemble all the parts by following the schematics below.



Here's how your circuit looks like in the end:

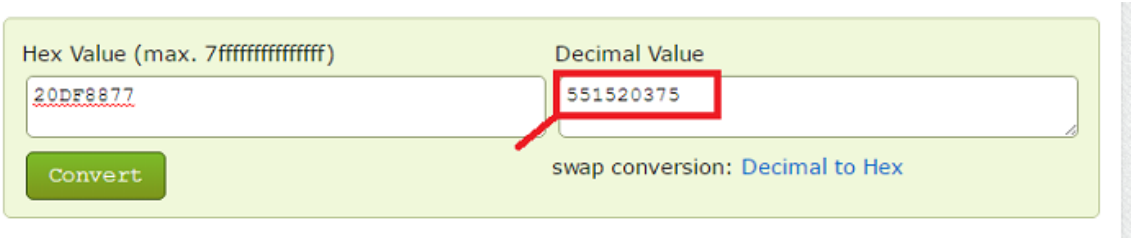


Code

Now, grab the codes you captured in the previous step. You need to convert your codes from hex to decimal.

For that, you can use the following website: www.binaryhexconverter.com/hex-to-decimal-converter

Here's a conversion example for one of my codes:



Repeat that process to all your hex values and save the decimal values. These are the ones you need to replace in the code below.

Download or copy the following sketch to your Arduino IDE. Write your own decimal values in the sketch provided in the `case` lines and upload it to your Arduino board. Make sure that you have the right board and COM port selected.

```
/*
 * Modified by Rui Santos, http://randomnerdtutorialscom
 * based on IRremote Library - Ken Shirriff
*/

#include <IRremote.h>

int IR_Recv = 11;   //IR Receiver Pin 3
int bluePin = 10;
int greenPin = 9;
int yellowPin = 8;

IRrecv irrecv(IR_Recv);
decode_results results;

void setup(){
  Serial.begin(9600);  //starts serial communication
  irrecv.enableIRIn(); // Starts the receiver
  pinMode(bluePin, OUTPUT);      // sets the digital pin as output
  pinMode(greenPin, OUTPUT);      // sets the digital pin as output
  pinMode(yellowPin, OUTPUT);      // sets the digital pin as
output

}

void loop(){
  //decodes the infrared input
  if (irrecv.decode(&results)){
    long int decCode = results.value;
    Serial.println(results.value);
```

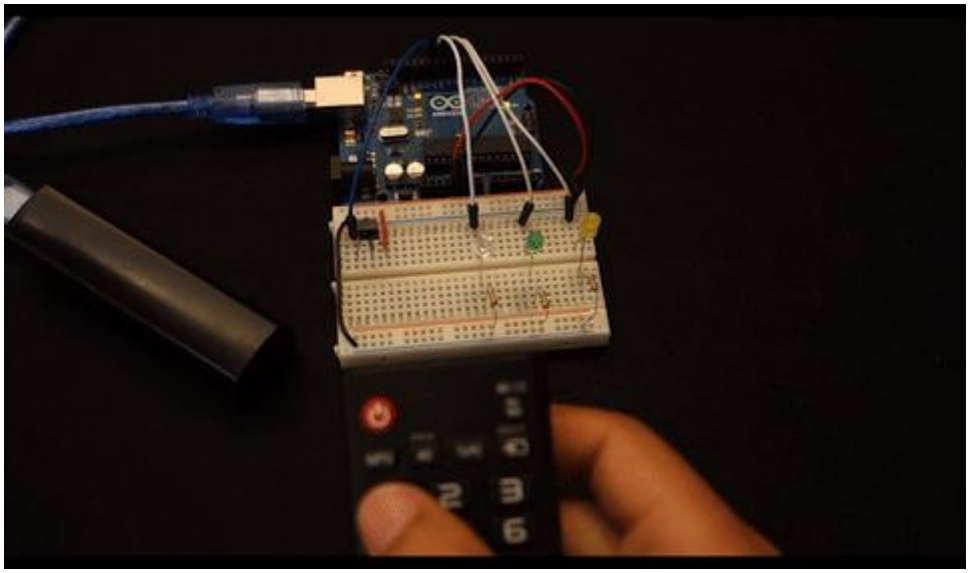
```
    //switch case to use the selected remote control button
    switch (results.value){
      case 551520375: //when you press the 1 button
        digitalWrite(bluePin, HIGH);
        break;
      case 551495895: //when you press the 4 button
        digitalWrite(bluePin, LOW);
        break;
       case 551504055: //when you press the 2 button
        digitalWrite(greenPin, HIGH);
        break;
       case 551528535: //when you press the 5 button
        digitalWrite(greenPin, LOW);
        break;
       case 551536695: //when you press the 3 button
        digitalWrite(yellowPin, HIGH);
        break;
       case 551512215: //when you press the 6 button
        digitalWrite(yellowPin, LOW);
        break;
    }
    irrecv.resume(); // Receives the next value from the button you
press
  }
  delay(10);
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/3b_control_leds_remote_control.ino

Demonstration

In the end you can control each LED individually using the buttons of your remote control.



Wrapping up

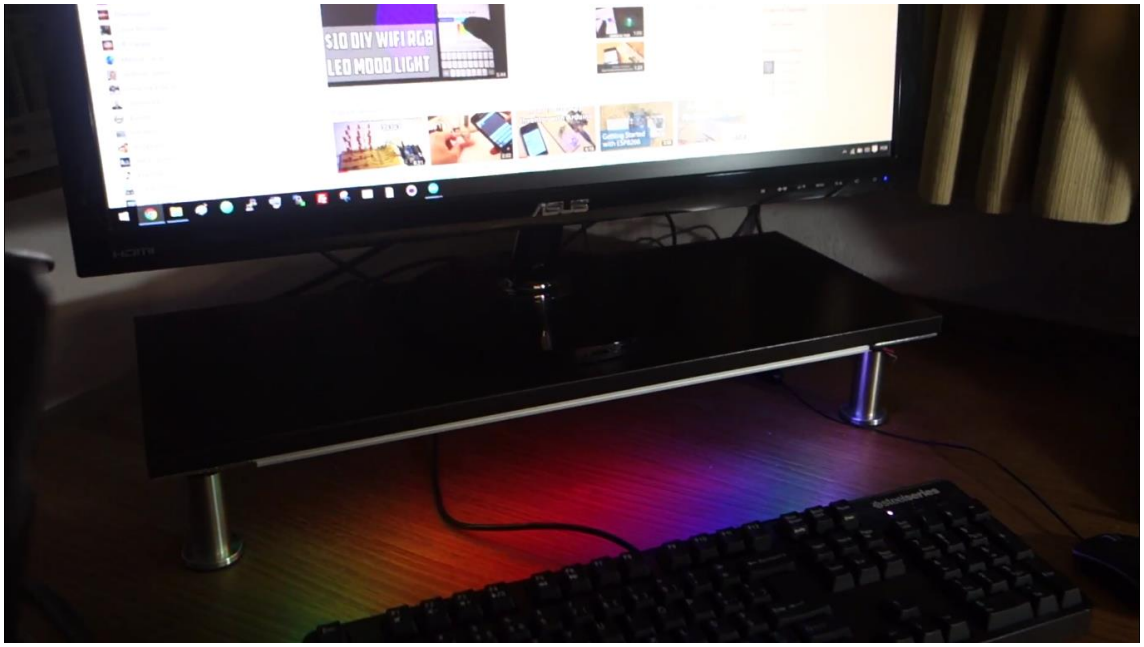
This is a great project to learn about the IR receiver. There are endless possibilities for what you can do with it.

For example, you can replace those LEDs with a relay to control your house appliances.

This can be particularly useful because some remotes have a bunch of buttons that you never use. So, why not use them to do something useful?

P.S. We should warned you that we've found some issues with the **IRremote library**. For example, we've found that this library conflicts with the **analogwrite()** arduino function.

Monitor Stand with Ambient Light



Monitor Stand with Ambient Light

Level: Intermediate – Time: 1h 30 minutes

In this project you're going to create a monitor stand with ambient light.

The monitor stand is an ergonomic boost for your posture, bringing the monitor to eye level while you work.

Additionally, it also brings space to store your desktop accessories below the table.

You'll add an RGB led strip to provide the ambient light.

This is a very simple project that you can actually build for your office.

I've been using my monitor stand with ambient light in my desk for a while and I'm very happy with the final outcome.

Monitor Stand

The monitor stand will be built using parts from IKEA (only $15).

- Shelf: EKBY LAIVA – IKEA.
- Adjustable legs: CAPITA – IKEA.

Follow the instructions provided in the CAPITA enclosure to build your monitor stand.

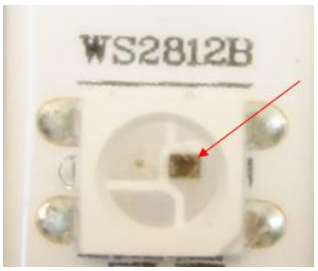
WS2812B Addressable RGB LED

We will use a WS2812B addressable RGB LED strip for the ambient light look.

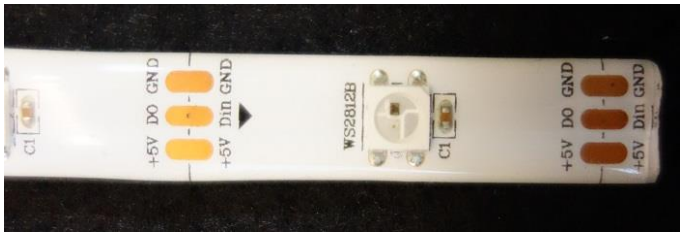
The preceding figure shows an addressable LED strip, which means that you can control the brightness and the color of each LED individually.

This LED strip has WS2812B LEDs wired in series. These LEDs have a built-in IC and you can control all of them with one-wire interface using one Arduino digital pin.

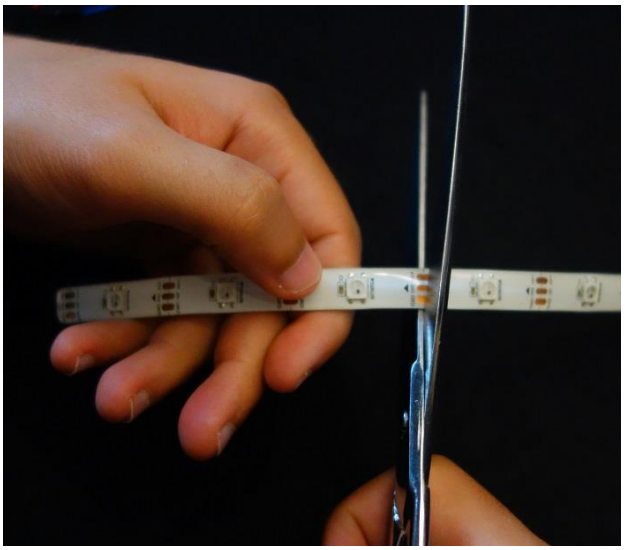
In the next figure you can see the chip inside the RGB LED.



This kind of strips are very flexible and can be cut to any desired length. As you can see, the strip is divided into segments, and each segment contains one RGB LED.



You can adjust its size by cutting the strip with a scissors in the right place (the proper places to cut the strip are marked).



These strips come with connectors at each end. I've decided to cut the connectors and solder header pins. That way is more handy to connect the strip to an Arduino or a breadboard.

Using an LED strip case

These strips usually come with a removable tape, so that you can stick them wherever you want. The problem is that they don't stick very well, so chances are that you'll find your strip in the floor the following day.

The solution: I found a strip case that diffuses the light and you can screw it permanently to a shelf.

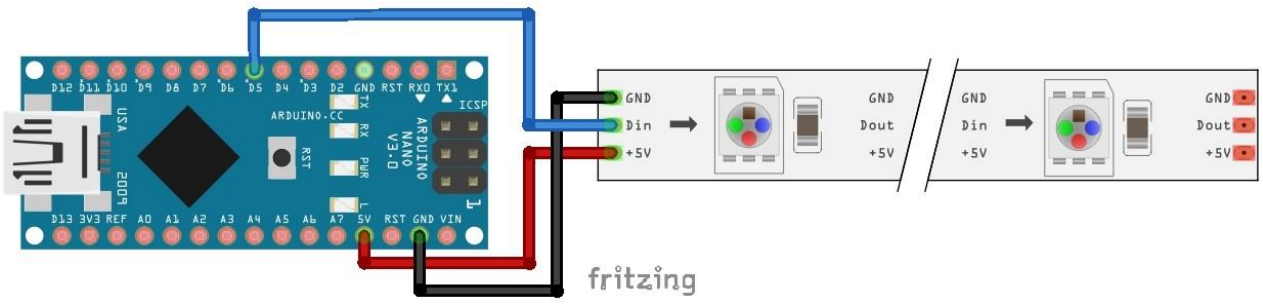
Parts required

Grab all the parts needed for this project.

Figure	Name	eBay
	WS2812 RGB LED strip	http://ebay.to/2ekEPsw
	LED Strip Case	http://ebay.to/2fSoTPi
	Shelf (EKBY LAIVA)	http://bit.ly/2fCCpmS
	4 Adjustable Legs (CAPITA)	http://bit.ly/2ekFF8B
	Jumper Wires	http://ebay.to/1PXeaJz

Schematics

Wiring the strip to the Arduino is very straightforward. It only requires three connections.



Code

To control the WS2812B LED strip, you need to download the `FastLED` library.

Installing the FastLED library

- Click here to download the FastLED library. You should have a .zip folder in your Downloads folder
- Unzip the .zip folder and you should get FastLED-master folder
- Rename your folder from ~~FastLED-master~~ to FastLED
- Move the FastLED folder to your Arduino IDE installation libraries folder
- Finally, re-open your Arduino IDE

After installing the needed library, upload the following code to your Arduino board.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

#include <FastLED.h>

#define LED_PIN     5
// Change the variable below to the number of RGB LEDs that your
strip has
#define NUM_LEDS    14
#define BRIGHTNESS  255
#define LED_TYPE    WS2812
#define COLOR_ORDER GRB
CRGB leds[NUM_LEDS];

#define UPDATES_PER_SECOND 50

CRGBPalette16 currentPalette;
TBlendType    currentBlending;

void setup() {
    delay(3000); // power-up safety delay
    FastLED.addLeds<LED_TYPE, LED_PIN, COLOR_ORDER>(leds,
NUM_LEDS).setCorrection(TypicalLEDStrip);
    FastLED.setBrightness(  BRIGHTNESS );

    currentPalette = RainbowColors_p;
    currentBlending = LINEARBLEND;
}

void loop(){
    ChangePalettePeriodically();

    static uint8_t startIndex = 0;
    startIndex = startIndex + 1;

    FillLEDsFromPaletteColors(startIndex);
```

```
    FastLED.show();
    FastLED.delay(2000 / UPDATES_PER_SECOND);
}

void FillLEDsFromPaletteColors(uint8_t colorIndex){
    uint8_t brightness = 255;

    for( int i = 0; i < NUM_LEDS; i++){
        leds[i] = ColorFromPalette( currentPalette, colorIndex,
brightness, currentBlending);
        colorIndex += 3;
    }
}

// There are several different palettes of colors demonstrated
here.
//
// FastLED provides several 'preset' palettes: RainbowColors_p,
RainbowStripeColors_p,
// OceanColors_p, CloudColors_p, LavaColors_p, ForestColors_p, and
PartyColors_p.
//
// Additionally, you can manually define your own color palettes,
or you can write
// code that creates color palettes on the fly.  All are shown
here.

void ChangePalettePeriodically(){
    uint8_t secondHand = (millis() / 1000) % 60;
    static uint8_t lastSecond = 99;

    if(lastSecond != secondHand){
        lastSecond = secondHand;
        if(secondHand ==  0){ currentPalette = RainbowColors_p;
currentBlending = LINEARBLEND;}
    }
}
```

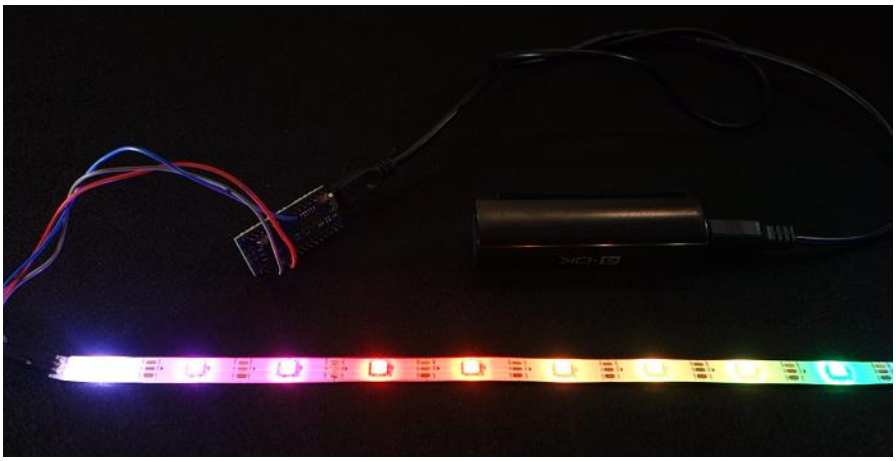
Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/16_rgb_ws2812b_rainbow_effect.ino

Note: you have to change `#define NUM_LEDS` to the number of LEDs that your strip currently has, in this example I have 14 LEDs.

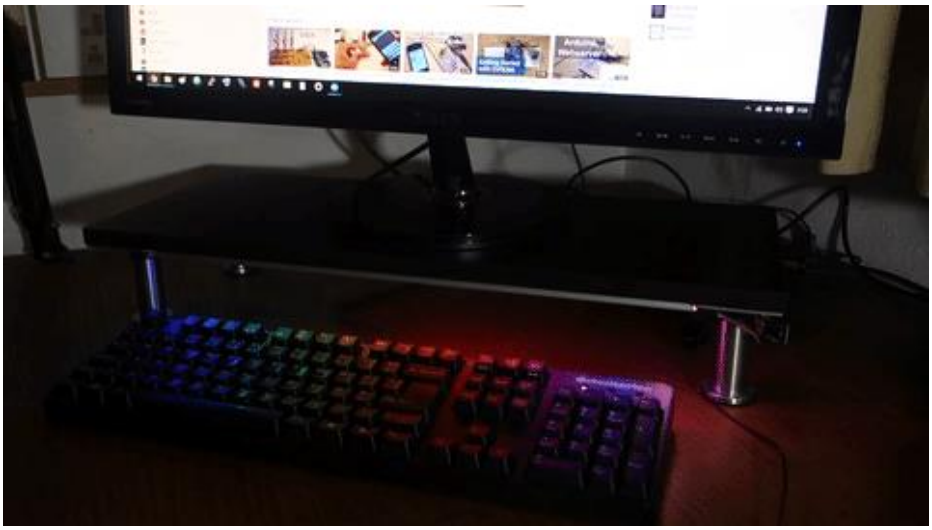
Demonstration

Here's your RGB LED strip after uploading the code.



Here's how your shelf looks like in the end.

And here's the monitor stand in action.



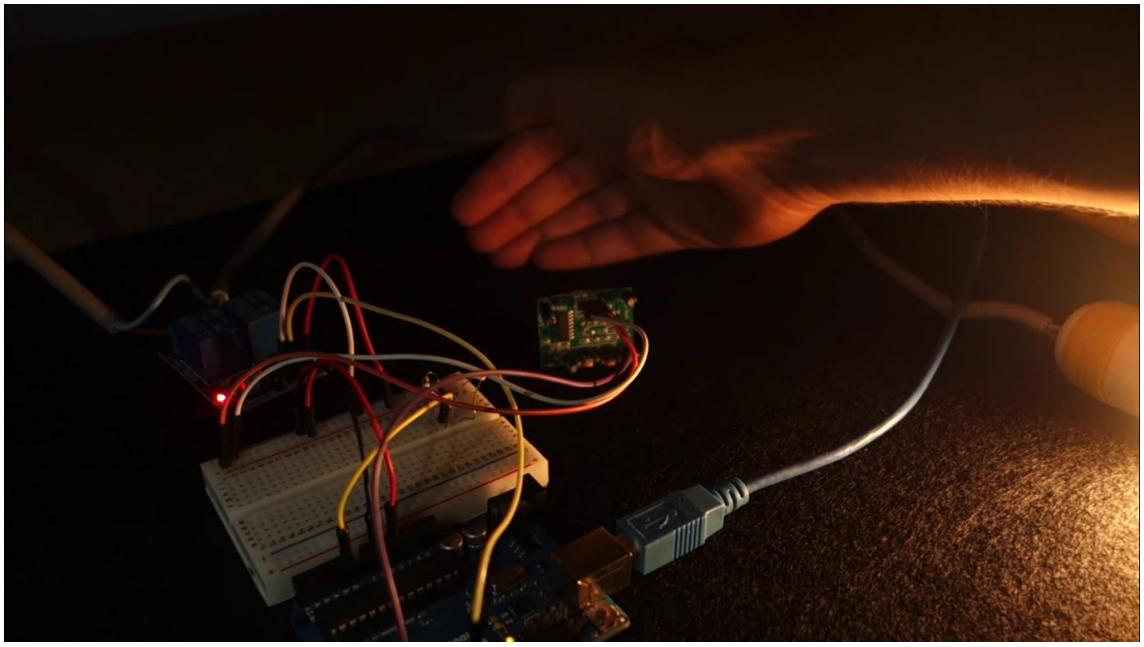
Wrapping up

In this project you've built a monitor stand with ambient light.

This is a great project to get you introduced to addressable RGB LED strips.

Feel free to change the Arduino code to generate other patterns that you like.

Night Security Light



Night Security Light

Level: Advanced– Time: 1h

In this project you're going to build a night security light.

SAFETY WARNING!

When you are making projects that are connected to mains voltage, you really need to know what you are doing, otherwise **you may shock yourself.** This is a serious topic and I want you to be safe. If you are not 100% sure what you are doing, do yourself a favor and don't touch anything. Ask someone who knows!

A night security light only turns on at night or in a dark environment as long as movement is detected. You probably have seen one of these lamps in some houses or maybe you even have one at your house.

Here's the main features of this project:

- the lamp turns on when it's dark AND movement is detected
- when movement is detected the lamp stays on for 10 seconds
- when the lamp is ON and detects movement, it starts counting 10 seconds again
- when there's light, the lamp is turned off, even when motion is detected.

Note: if you're not comfortable dealing with mains voltage, but you still want to try to do the project, you can replace the relay module with an LED, for example. The code and schematic are very similar.

Photoresistor

The light intensity is detected using a photoresistor (light dependent resistor). You can adjust the threshold value between dark and light in the code provided.

PIR motion sensor

The PIR motion sensor detects movement.

PIR stands for “Passive Infrared”. The PIR motion sensor measures infrared light from objects in its field of view. So, it can detect motion based on changes in infrared light in the environment. It is ideal to detect if a human has moved in or out of the sensor range.

The sensor in the figure above has two built-in potentiometers to adjust the delay time (the potentiometer at left) and the sensitivity (the potentiometer at right).

Pinout

Wiring the PIR motion sensor to an Arduino is pretty straightforward – the sensor has 3 pins.

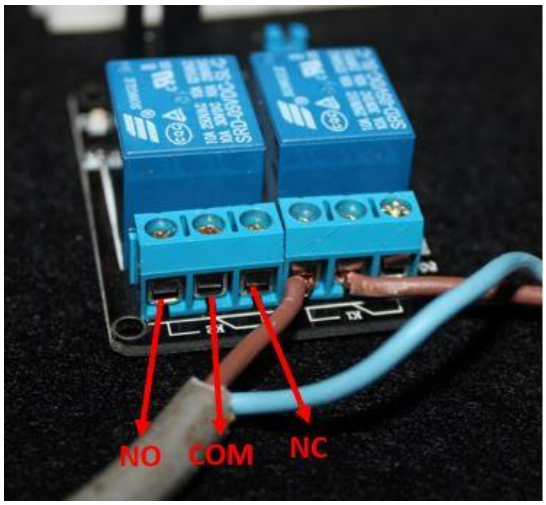
- GND – connect to ground
- OUT – connect to an Arduino digital pin
- 5V – connect to 5V

Relay module

A relay is an electrically operated switch. It means that it can be turned on or off, letting the current going through or not. The relay module is the one in the figure below.

This particular relay module comes with two relays (those blue cubes).

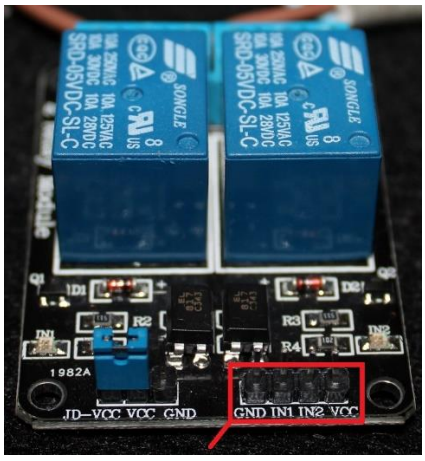
About mains voltage, relays have 3 possible connections:



- COM: common pin
- NO: normally open – there is no contact between the common pin and the normally open pin. So, when you trigger the relay, it connects to the COM pin and power is provided to the load (a lamp, in our case).
- NC: normally closed – there is contact between the common pin and the normally closed pin. There is always contact between the COM and NC pins, even when the relay is turned off. When you trigger the relay, the circuit is opened and there is no power provided to the load.

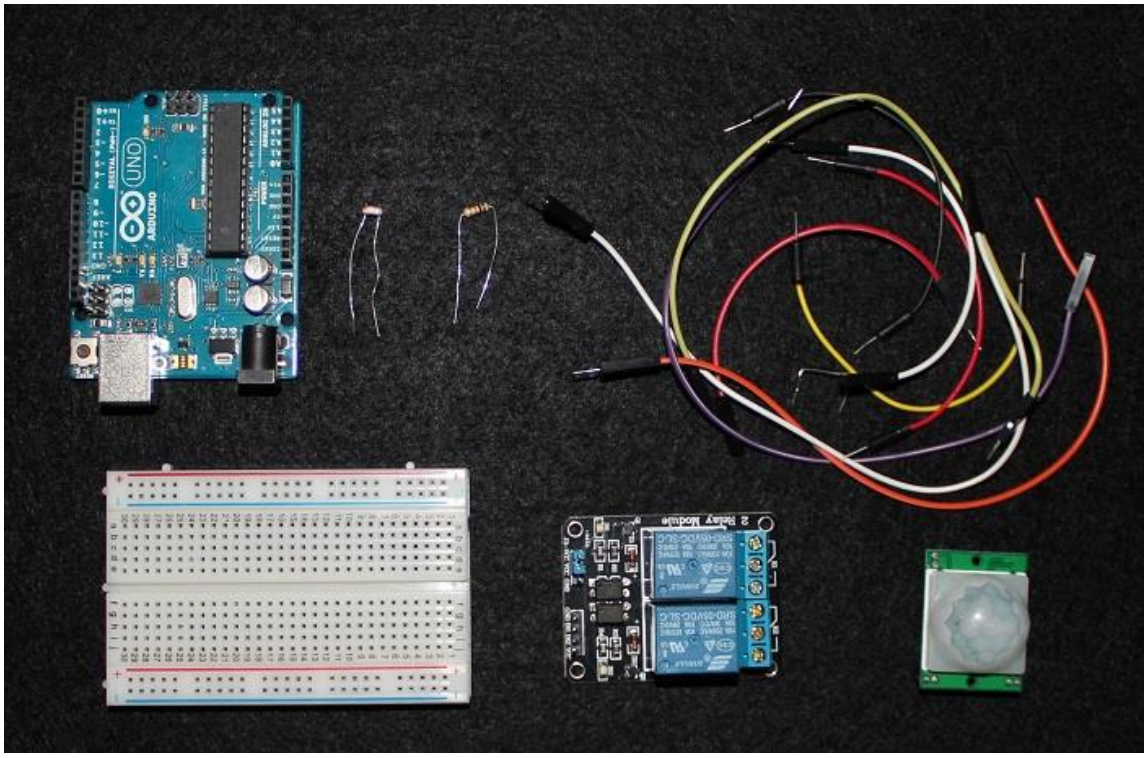
Relating this project, it is better to use a normally open circuit, because we want to light up the lamp occasionally.

The connections between the relay and the Arduino are really simple:

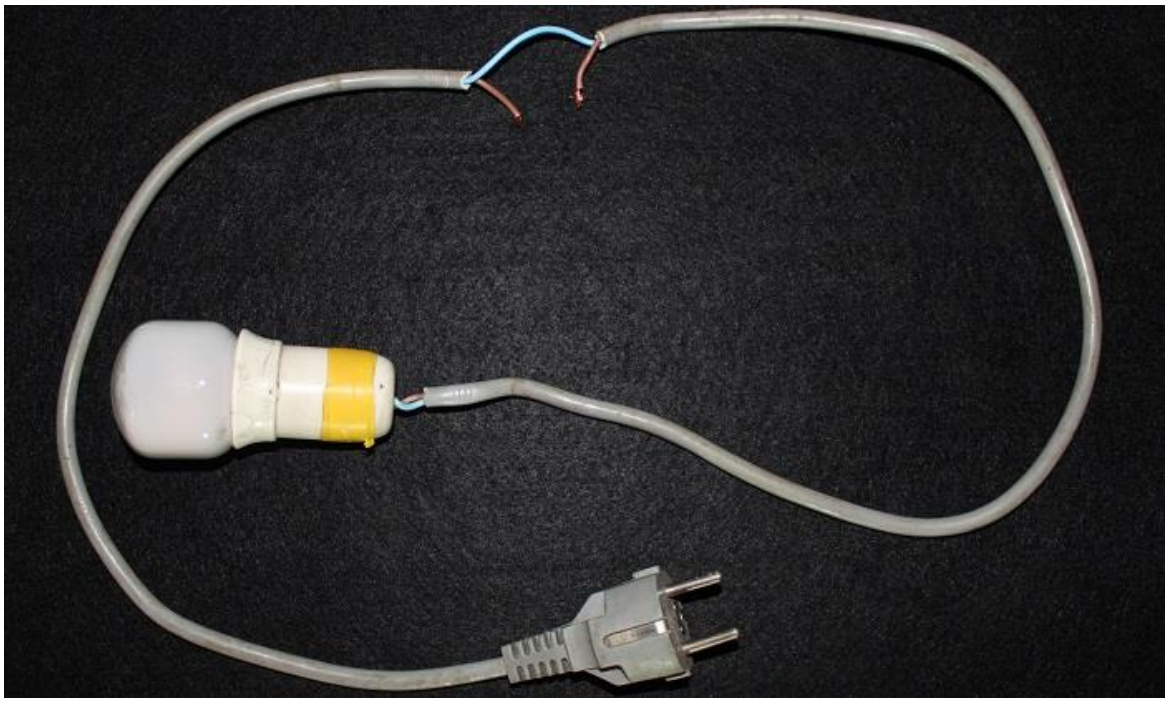


- GND: goes to ground
- IN1: controls the first relay. Should be connected to an Arduino digital pin
- IN2: controls the second relay. Should be connected to an Arduino digital pin
- VCC: goes to 5V

Parts required



Besides these electronics components, you also need an AC male socket, an AC wire and a lamp bulb holder (a lamp cord set).



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Arduino Uno	http://ebay.to/1SQda0R
	PIR Motion Sensor	http://ebay.to/1Wqh9jL
	Photoresistor	http://ebay.to/2bMWpSV
	10kOhm Resistor	http://ebay.to/1KsMYFP
	Relay Module	http://ebay.to/2dRv8lh
	Lamp Cord Set	http://ebay.to/2feHELc
	Breadboard	http://ebay.to/21bEojM
	Jumper Wires	http://ebay.to/1PXeaJz

Code

For this project, you don't need to install any Arduino libraries. So, you just have to download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure that you have the right Board and COM port selected.

Warning: do not upload a new code to your Arduino board while your lamp is connected to the mains voltage. You should unplug the lamp from mains voltage, before upload a new sketch to your Arduino.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

// Relay pin is controlled with D8. The active wire is connected to
Normally Closed and common
int relay = 8;
```

```
volatile byte relayState = LOW;

// PIR Motion Sensor is connected to D2.
int PIRInterrupt = 2;

// LDR pin is connected to Analog 0
int LDRPin = A0;
// LDR value is stored on LDR reading
int LDRReading;
// LDR Threshold value
int LDRThreshold = 300;

// Timer Variables
long lastDebounceTime = 0;
long debounceDelay = 10000;

void setup() {
  // Pin for relay module set as output
  pinMode(relay, OUTPUT);
  digitalWrite(relay, HIGH);
  // PIR motion sensor set as an input
  pinMode(PIRInterrupt, INPUT);
  // Triggers detectMotion function on rising mode to turn the
relay on, if the condition is met
  attachInterrupt(digitalPinToInterrupt(PIRInterrupt),
detectMotion, RISING);
  // Serial communication for debugging purposes
  Serial.begin(9600);
}

void loop() {
  // If 10 seconds have passed, the relay is turned off
  if((millis() - lastDebounceTime) > debounceDelay && relayState ==
HIGH){
    digitalWrite(relay, HIGH);
    relayState = LOW;
    Serial.println("OFF");
  }
  delay(50);
}

void detectMotion() {
  Serial.println("Motion");
  LDRReading = analogRead(LDRPin);
  // LDR Reading value is printed on serial monitor, useful to get
your LDRThreshold
  //Serial.println(LDRReading);
  // Only turns the Relay on if the LDR reading is higher than the
LDRThreshold
  if(LDRReading > LDRThreshold){
    if(relayState == LOW){
      digitalWrite(relay, LOW);
    }
    relayState = HIGH;
    Serial.println("ON");
    lastDebounceTime = millis();
  }
}
```

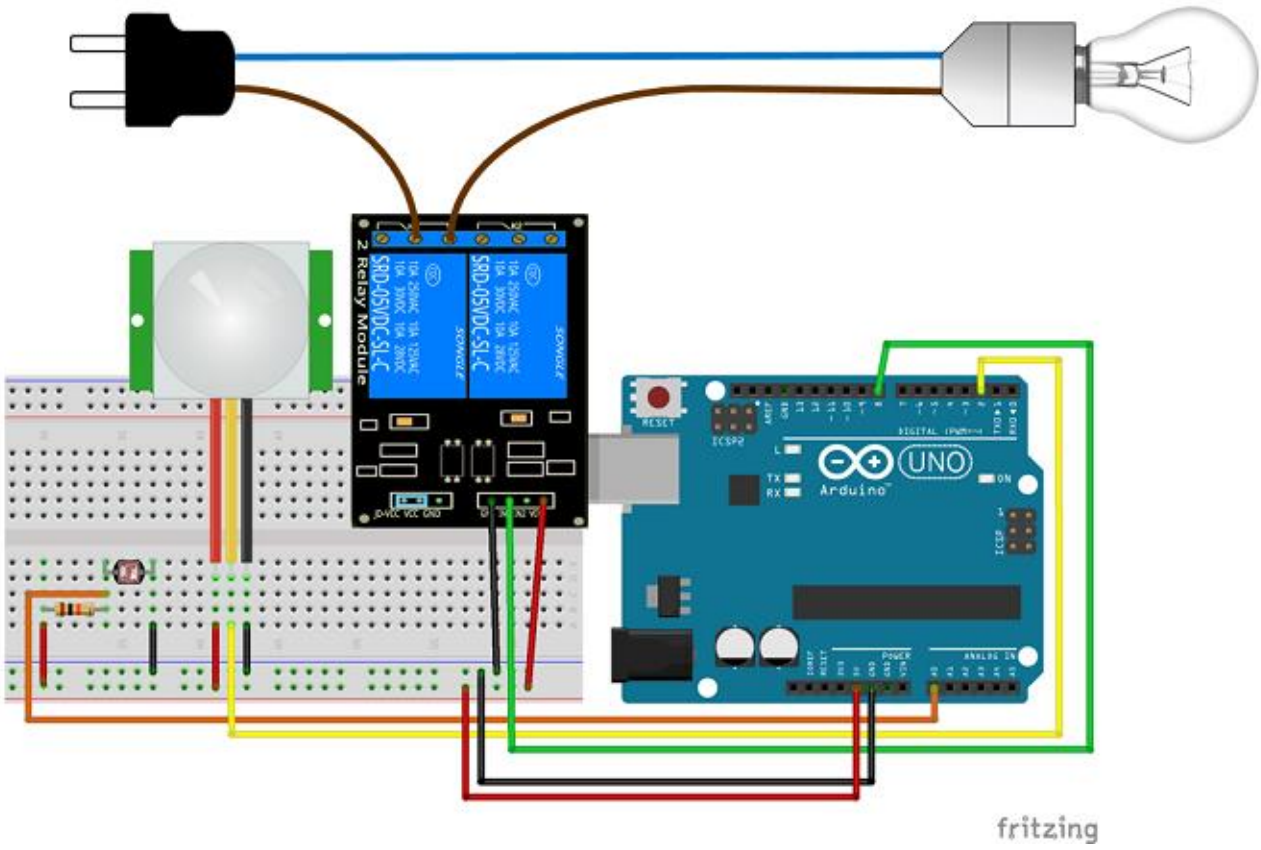
Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/11_night_security_light.ino

Schematics

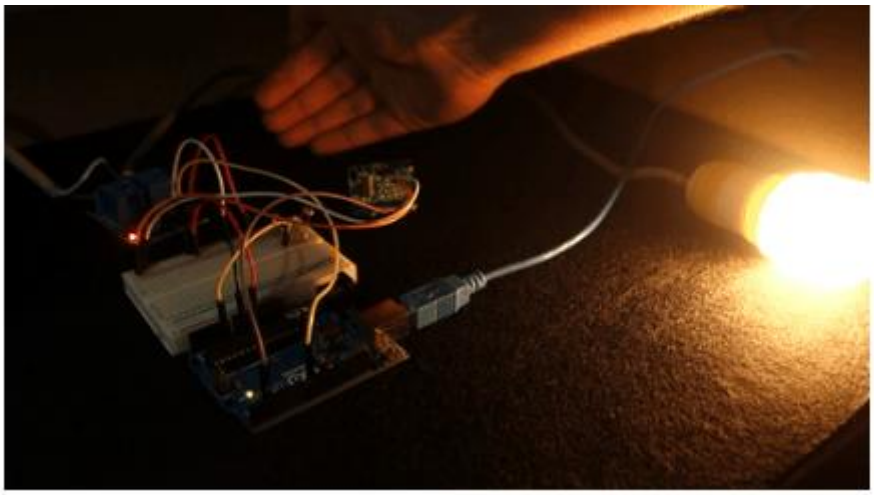
Here's the schematics for this project. Be very careful with all the connections, mainly with the mains voltage connections.

Warning: I can't stress this enough, but do not touch the relay or any live wires while they are connected to the mains voltage. This can be dangerous, if you're not comfortable dealing with mains voltage, do yourself a favor and don't touch anything.



Demonstration

Here's your circuit in action:



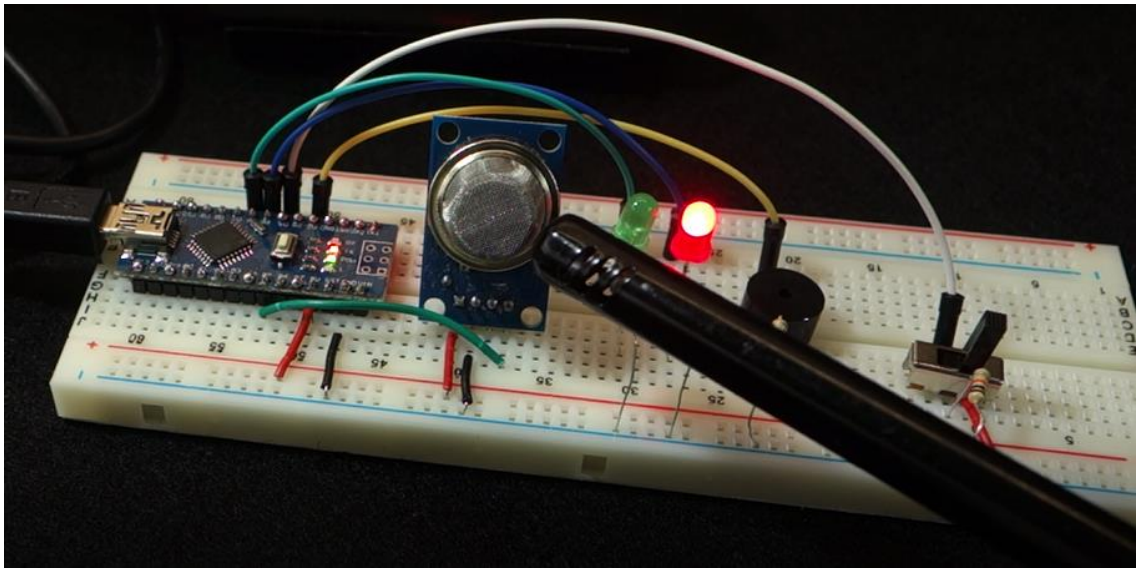
Wrapping up

In this project you've built a night security light with a photoresistor and a PIR motion sensor.

This is a great project to practice with relays and with the PIR motion sensor.

With just a few changes you can modify this project to work as a burglar detector. So, instead of turning on a lamp, it would turn on a buzzer.

Smoke/Gas Detector Alarm



Smoke/Gas Detector Alarm

Level: Beginner – Time: 40 minutes

In this project you're going to build a smoke/gas detector alarm.

You will use an MQ-2 gas sensor to measure the gas levels in the environment. The alarm will be built using a buzzer. You will also have a green and a red LED to show the gas levels status.

MQ-2 gas/smoke sensor

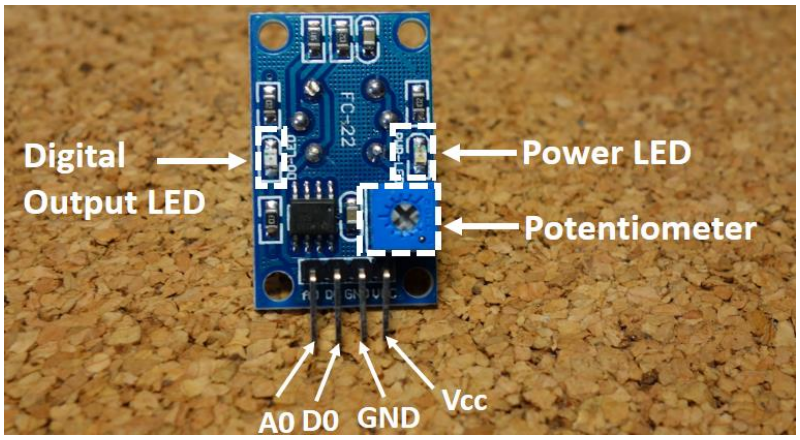
In this project you're going to measure the gas levels in the environment using an MQ-2 gas sensor.

This sensor is sensitive to smoke and the following flammable gases: LPG, butane, propane, methane, alcohol and hydrogen. The resistance of the sensor is different depending on the type of gas.

The sensor has analog and digital output pins.

- Analog output (A0) – gives you a value between 0 and 1023
- Digital output (D0) – returns HIGH or LOW depending if the smoke levels are below or above a certain threshold.

The threshold for the digital output pin is adjusted using the built-in potentiometer.

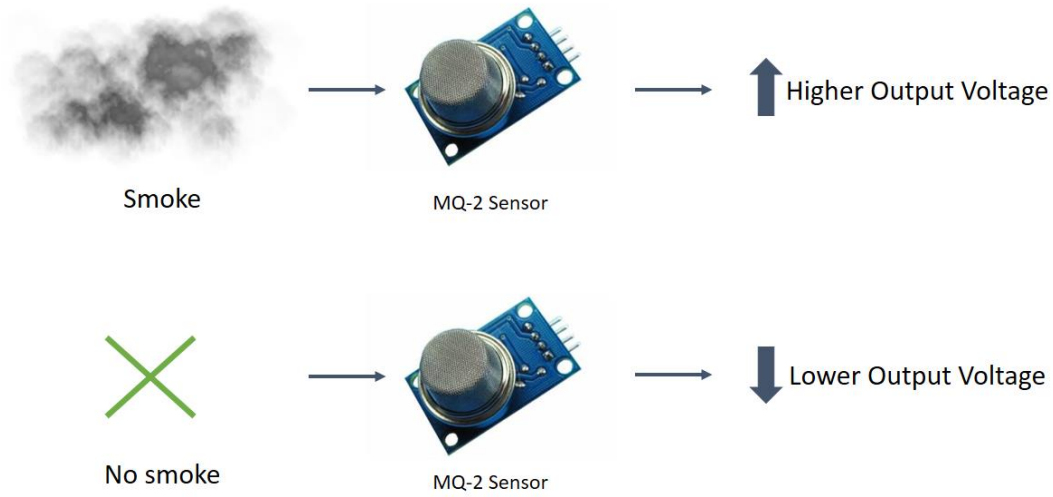


How does it work?

The voltage that the sensor outputs changes accordingly to the gas levels on the atmosphere. The sensor outputs a voltage that is proportional to the gas concentration.

In other words, the relationship between voltage and gas concentration is the following:

- The greater the gas concentration, the greater the output voltage
- The lower the gas concentration, the lower the output voltage



Parts required

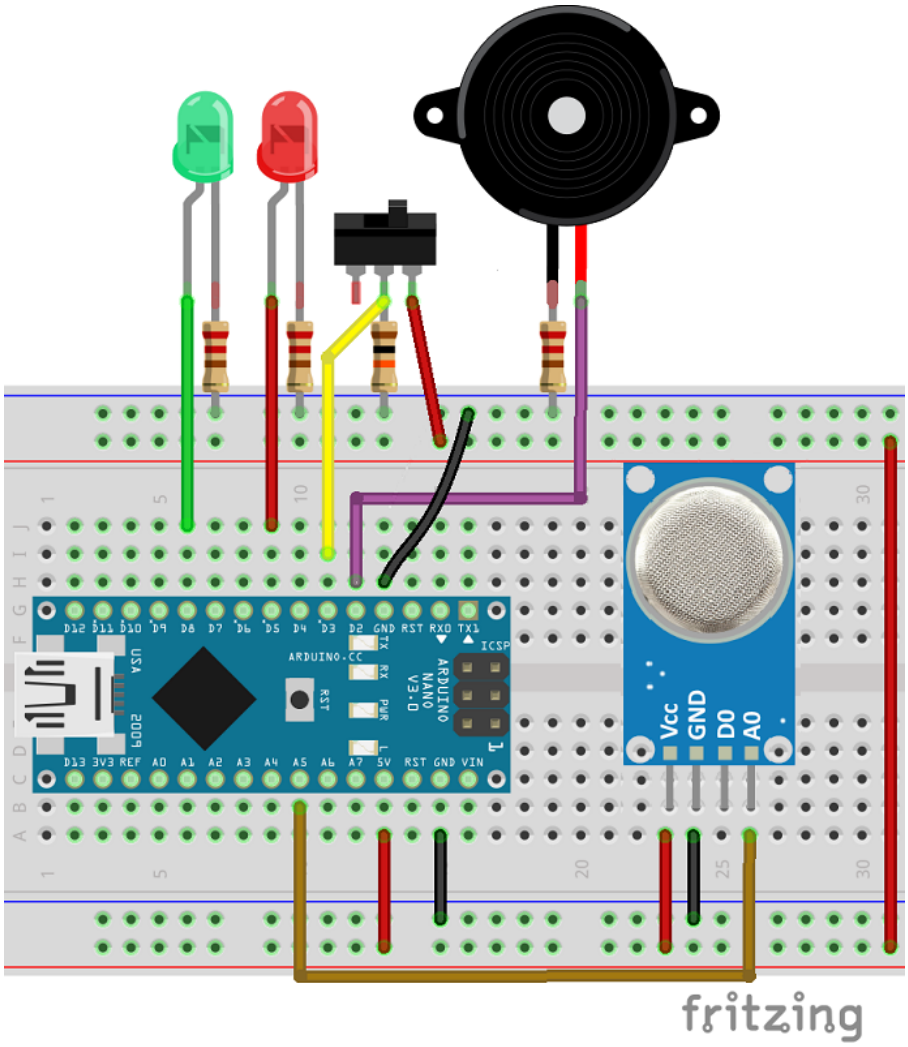
Grab all the components needed for this project:

Figure	Name	eBay
	Breadboard	http://ebay.to/21bEojM
	Arduino UNO	http://ebay.to/1SQda0R
	MQ-2 Smoke/Gas Sensor	http://ebay.to/1SsN4ib
	1x Red LED 1x Green LED	http://ebay.to/20H2Oyy

	1x buzzer	http://ebay.to/2dcur1W
	3x 220Ohm Resistor 1x 10KOhm Resistor	http://ebay.to/1KsMYFP
	1x Toggle Switch	http://ebay.to/1Q5VuuE
	Jumper Wires	http://ebay.to/1PXeaJz

Schematics

Assemble all the parts by following the schematics below.



Code

Copy the next code to your Arduino IDE and upload it to your Arduino board.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

int redLed = 5;
int greenLed = 8;

int buzzer = 2;
int button = 3;

int smokeSensor = A5;
int sensorThres = 100;

void setup() {
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(button, INPUT);
  Serial.begin(9600);
}

void loop() {
  if(digitalRead(button) == HIGH) {
    int sensorValue = analogRead(smokeSensor);
    Serial.println(sensorValue);

    if(sensorValue > sensorThres) {
      tone(buzzer, 1000);
      digitalWrite(redLed, HIGH);
      digitalWrite(greenLed, LOW);
    }

    else {
      digitalWrite(redLed, LOW);
      digitalWrite(greenLed, HIGH);
      noTone(buzzer);
    }
    delay(10);
  }
  else {
    digitalWrite(redLed, LOW);
    digitalWrite(greenLed, LOW);
    noTone(buzzer);
  }
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/18_smoke_detector_alarm.ino

Note: you might need to adjust the `sensorThres` variable. In my case, I have `sensorThres = 100`. You should change that value depending on your sensor sensitivity.

Follow these next instructions:

1) Open the serial monitor at a baud rate of 9600

2) Check the readings for a while in order to gather a value that represents a typical scenario with no gas

3) Then, grab a lighter and press it next to the sensor in order to release some gas. You should see the values increasing

4) Chose a value between the typical scenario and the values when the sensor detects gas (in my case I chose 100)

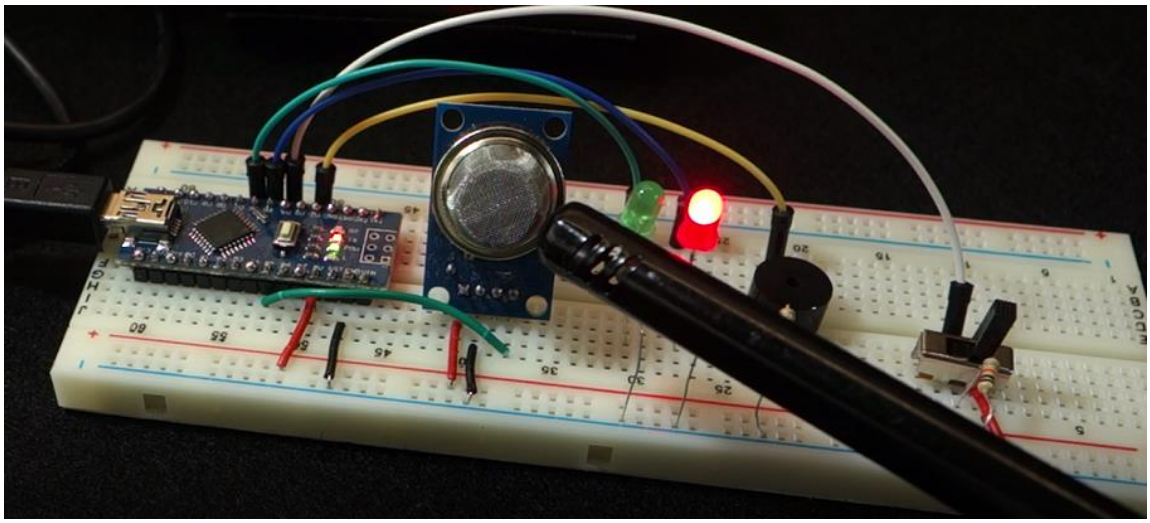
5) Change your `sensorThres` variable value (`int sensorThres = 100;`)

6) Upload the updated sketch

7) Test the circuit

Demonstration

When you approximate and press the lighter next to the smoke sensor, the buzzer starts beeping and the red LED lights up.

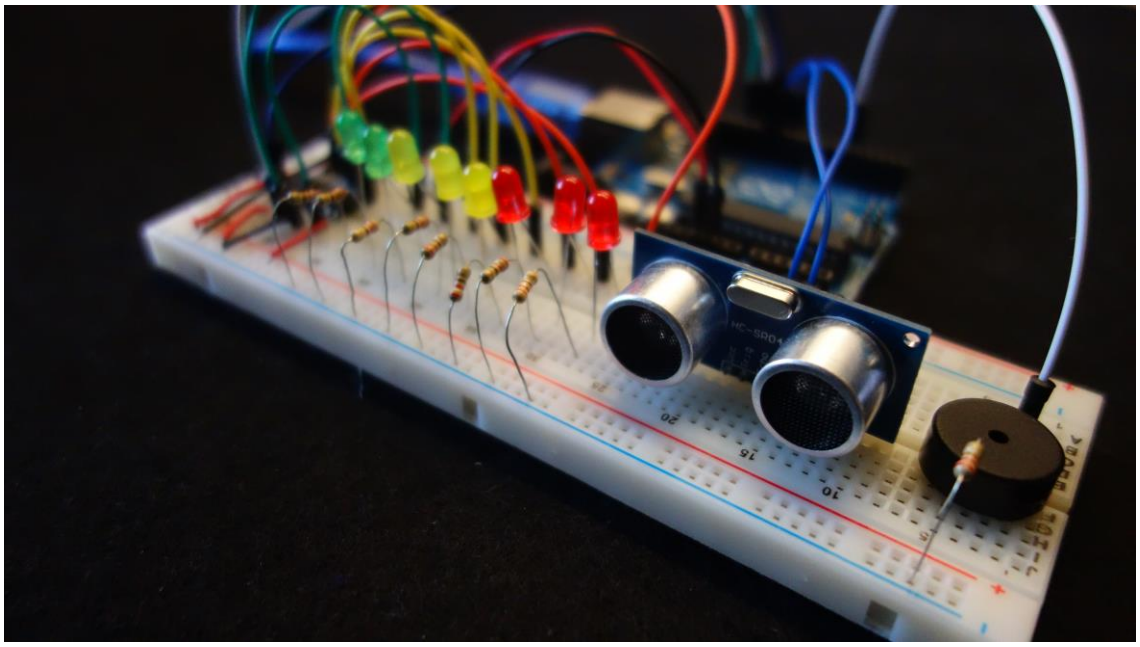


Wrapping up

In this project you learned how to use the gas/smoke sensor module.

This project gave you knowledge to work with other sensors whose output voltage depends on the environment conditions.

Parking Sensor



Parking Sensor

Level: Intermediate – Time: 45 minutes

In this project you're going to build a simple circuit that can work as parking sensor.

Here's the main components you'll use in this project:

- ultrasonic sensor to measure distance to an object
- buzzer that beeps accordingly to how far you are from an object
- several LEDs linked to a Bit Shift Register making a progress bar that increases or decreases accordingly to the distance to an object.

Ultrasonic sensor

The HC-SR04 (which is the model that I'm using) reads from 2 cm to 400 cm with an accuracy of 0.3 cm, which is good for most projects. This module comes with an ultrasonic transmitter and receiver.

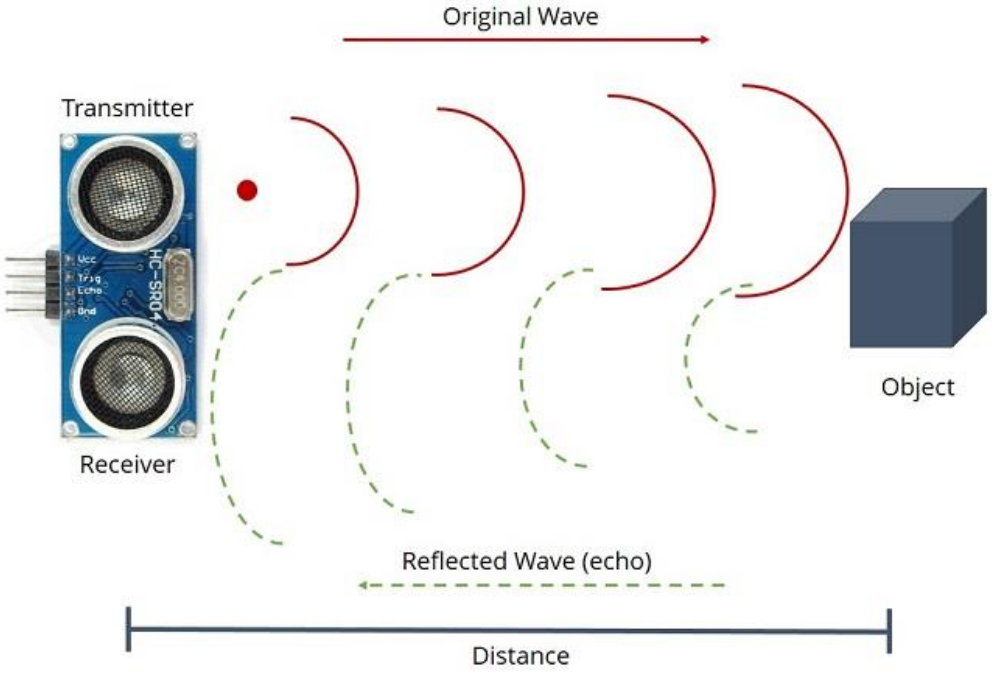
This sensor has 4 pins. Wiring the ultrasonic sensor to your Arduino is pretty straightforward. The Vcc pin should be connected to 5V and the GND pin to GND. The trig and the echo pin should be connected to digital pins (see the schematics section).

How it works?

The ultrasonic sensor uses sonar to determine the distance to an object. Here's what happens:

1. the transmitter (trig pin) sends a signal: a high-frequency sound
2. when the signal finds an object, it is reflected and
3. the transmitter (echo pin) receives it.

The time between the transmission and reception of the signal allows us to know the distance to an object. This is possible because we know the velocity of the sound in the air as shown in figure below.

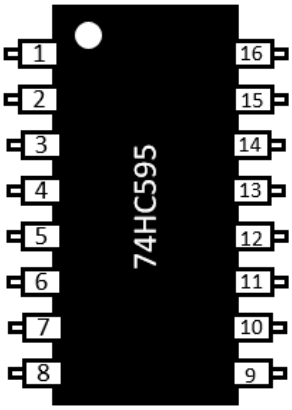


Bit Shift Register

The shift register is the IC in the following figure.

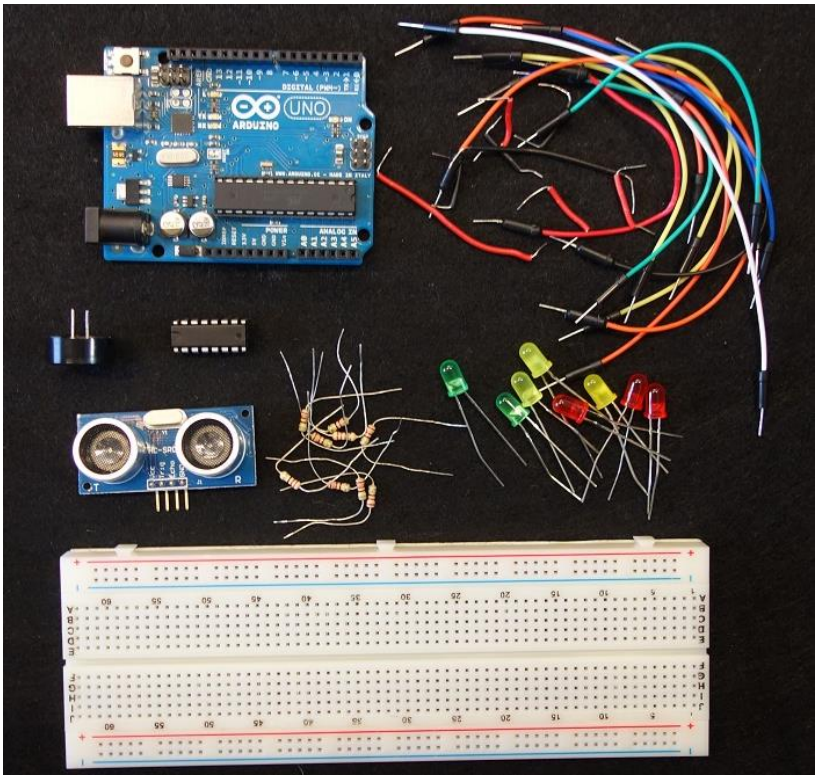
It allows you to add additional inputs or outputs to the Arduino. In our case, we want to control a large number of LEDs (8 LEDs + the sensor + buzzer) and without the shift register we wouldn't have enough digital pins in the Arduino.

The shift register allows you to control 8 outputs using just a few Arduino pins. Here's the 74HC595 Shift Register Pinout.



Pin	Symbol	Description
1	Q1	Parallel data output (bit-1)
2	Q2	Parallel data output (bit-2)
3	Q3	Parallel data output (bit-3)
4	Q4	Parallel data output (bit-4)
5	Q5	Parallel data output (bit-5)
6	Q6	Parallel data output (bit-6)
7	Q7	Parallel data output (bit-7)
8	GND	GND
9	Q7’	Serial Data Output
10	$\overline{MR}$	Master Reset (Active Low)
11	SH_CP	Shift Register Clock Input
12	ST_CP	Storage Register Clock Input
13	$\overline{OE}$	Output Enable (Active Low)
14	DS	Serial Data Input
15	Q0	Parallel data output (bit-8)
16	Vcc	Positive Supply Voltage

Parts required

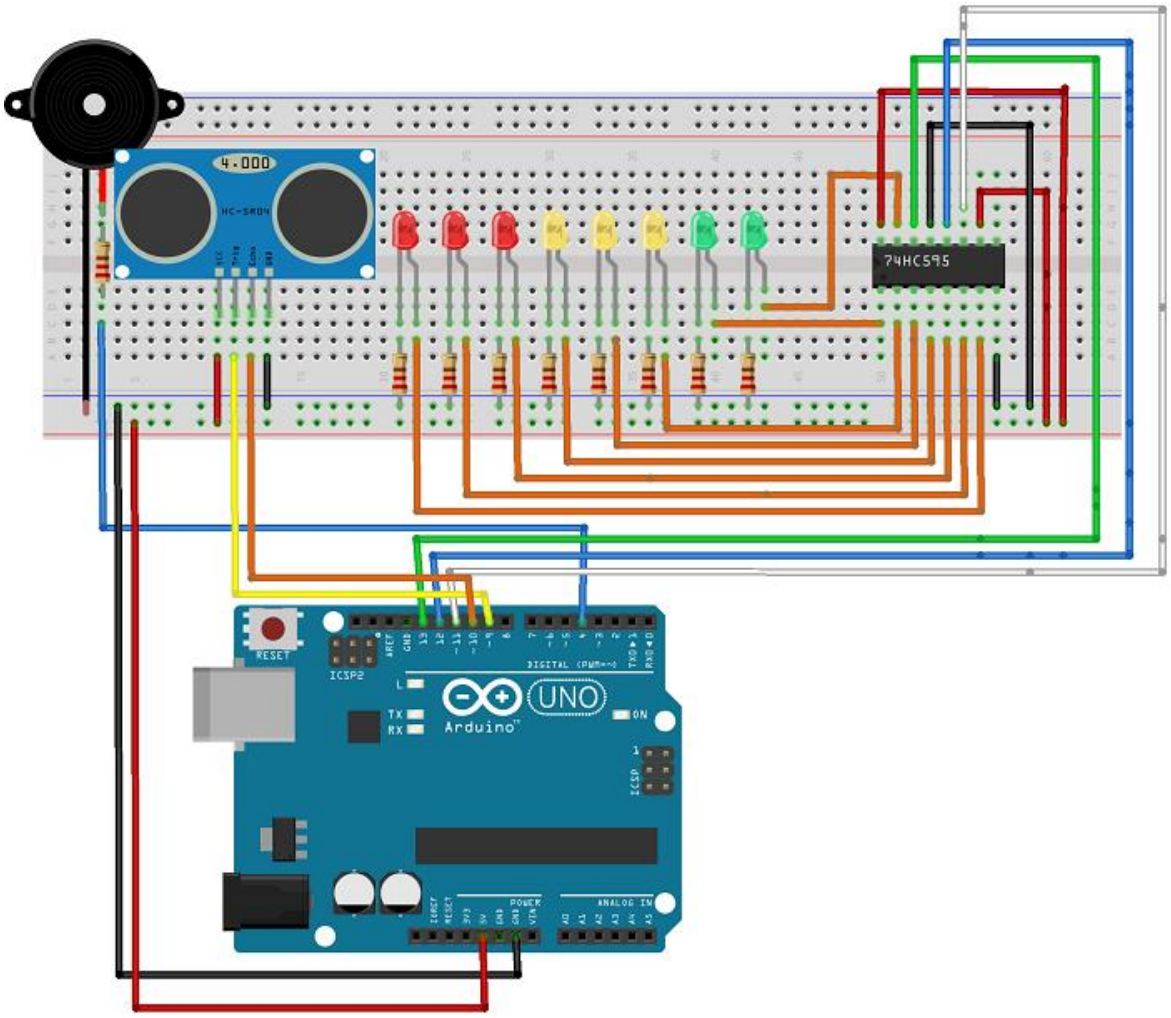


Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Breadboard	http://ebay.to/21bEojM
	Arduino UNO	http://ebay.to/1SQda0R
	1x ultrasonic sensor	http://ebay.to/1UzSdYv
	1x 74HC595 8 Bit Shift Register	http://ebay.to/2cOpJw3
	8x 5mm LED (3x red, 3x yellow, 2x green)	http://ebay.to/20H2Oyy
	1x buzzer	http://ebay.to/2dcur1W
	9x 220Ohm Resistor	http://ebay.to/1KsMYFP
	Jumper Wires	http://ebay.to/1PXeaJz

Schematics

Assemble all the parts by following the schematics below.



Code

You have to download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure that you have the right board and COM port selected.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

int tonePin = 4;     // Tone
int trigPin = 9;     //Trig
int echoPin = 10;    //Echo
int clockPin = 11;   // IC Pin 11
int latchPin = 12;   // IC Pin 12
int dataPin = 13;    // IC Pin 14
```

```
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 < 3){
    proximity=0;
  }
  else if (proximity >= 3 && proximity <= 4){
    tone(tonePin, 1000, 250);
  }
  else if (proximity >= 5 && proximity <= 6){
     tone(tonePin, 1000, 500);
  }
  else if (proximity >= 7 && proximity <= 8){
    tone(tonePin, 1000, 900);
```

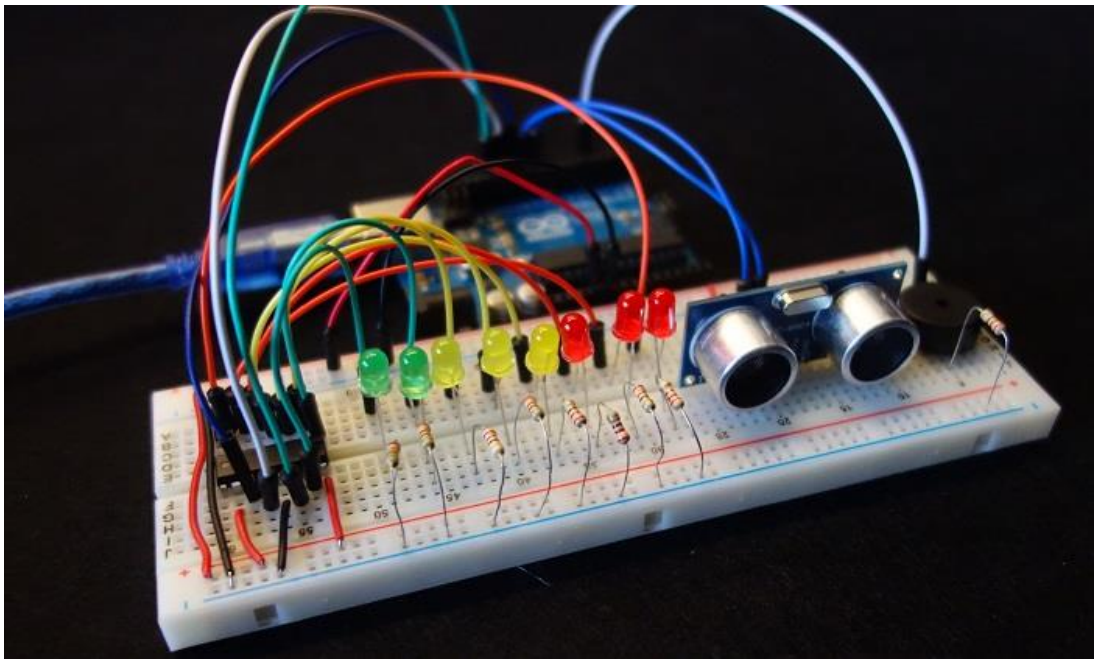
```
  }
  shiftOut(dataPin, clockPin, MSBFIRST,
possible_patterns[proximity]);
  digitalWrite(latchPin, HIGH);

  delay(600);
  noTone(tonePin);
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/5_parking_sensor.ino

Demonstration

After assembling the circuit and uploading the code, your circuit looks like this:



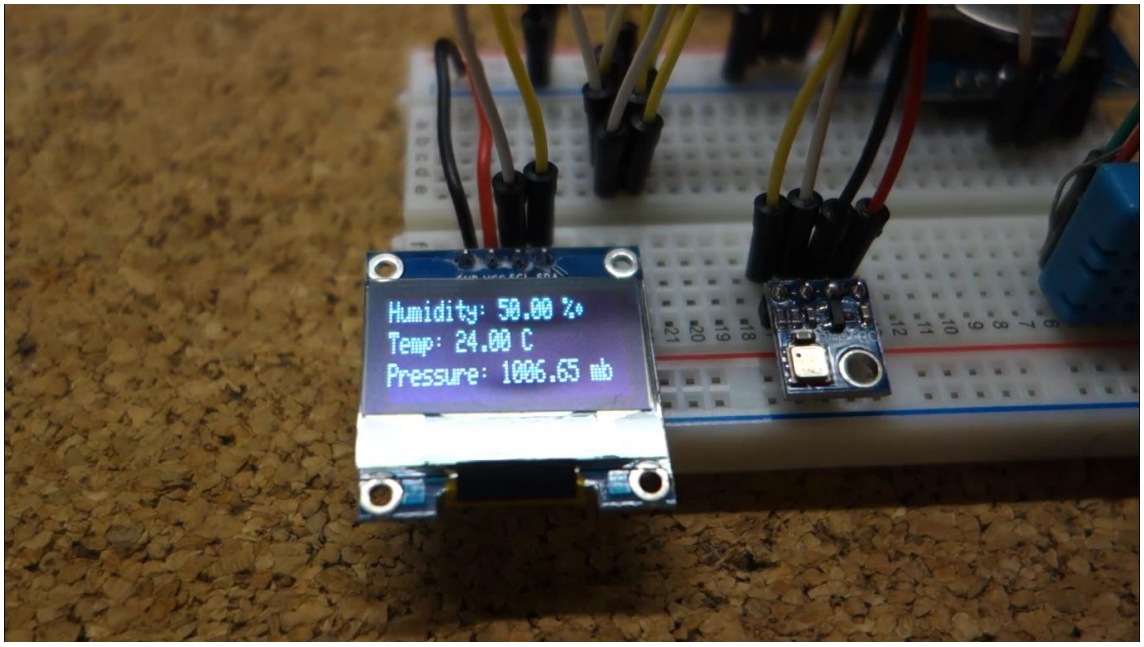
Wrapping up

In this project you learned how to use an ultrasonic sensor.

You can use the ultrasonic sensor in a robot, for example in an obstacle-avoiding robot.

You also learned about the bit shift register, which allows you to expand the number of Arduino output pins.

Arduino Weather Station



Arduino Weather Station (AWS) with Temperature, Humidity, Pressure, Date and Time

Level: Intermediate – Time: 1h30

In this project you're going to build an Arduino Weather Station (AWS) that shows: temperature, humidity, pressure, date and time.

Everything will be displayed in an OLED display. For the measurements, we will use the following sensors/modules:

- **DTH11** – for measuring humidity and temperature
- **BMP180 barometric** sensor – for measuring pressure
- **DS1307 Real Time Clock (RTC)** – for setting date and time

This project is divided into two parts:

- Getting familiar with the sensors/modules we're going to use and installing the needed libraries;
- Building the circuit and writing/uploading the code.

During this project, you will need to install a couple of libraries. But don't worry, everything will be well documented. I recommend that you start installing them as you follow along this tutorial.

OLED display

The OLED display is the one in the following figure.

It is a very small display, the screen has 25mm x 14mm (0.98in x 0.55in). It is made of 128 by 64 individual OLED pixels and no backlight is required. My OLED display is monochrome, but there are other models with colors.

I2C communication

This OLED display uses I2C communication. This means that the display communicates with the Arduino using just 2 pins.

These pins are the SCL and SDA pins. You should connect the SCL OLED pin to the SCL Arduino pin and the SDA OLED pin to the SCL Arduino pin.

Here are the SDA and SCL pins of the Arduino Uno (and other boards – in case you're using other than the UNO board):

- UNO: SDA (A4) – SCL (A5);
- Nano: SDA (A4) – SCL (A5);
- MEGA: SDA (20) – SCL (21);
- Leonardo: SDA (20) – SCL (21);

Libraries

To control the OLED display you need the `Adafruit_GFX.h` library and `Adafruit_SSD1306.h` library.

Installing the Adafruit_GFX library

1. Click here to download the Adafruit GFX library. You should have a .zip folder in your Downloads folder
2. Unzip the .zip folder and you should get Adafruit-GFX-Library-master folder
3. Rename your folder from ~~Adafruit-GFX-Library-master~~ to Adafruit_GFX_Library (you really need ro replace those "-" by "_")
4. Move the Adafruit_GFX_Library folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

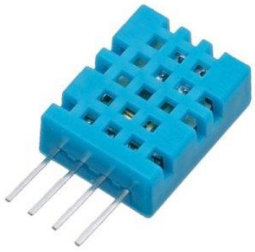
Installing the adafruit_SSD1306 library

1. Click here to download the Adafruit_SSD1306 library. You should have a .zip folder in your Downloads folder
2. Unzip the .zip folder and you should get Adafruit-GFX-Library-master folder
3. Rename your folder from ~~Adafruit_SSD1306-master~~ to Adafruit_SSD1306
4. Move the Adafruit_SSD1306 folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

DHT11 temperature and humidity sensor

In this project, you'll use the DHT11 temperature and humidity sensor – you can also use the DHT22 with a small code change.

The DHT11 is the sensor in the following figure:



This sensor contains a chip that does analog to digital conversion and spits out a digital signal with the temperature and humidity. So, it is wired to an Arduino digital pin.

To use the DHT11 or DHT22 sensor, you need to install the **DHT.h** library.

Installing the DHT sensor library

1. Click here to download the DHT-sensor-library. You should have a .zip folder in your Downloads folder
2. Unzip the .zip folder and you should get DHT-sensor-library-master folder
3. Rename your folder from ~~DHT-sensor-library-master~~ to DHT
4. Move the DHT folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

BMP180 barometric sensor

The BMP180 barometric sensor allows you to collect data about: pressure, altitude and temperature. It is a very small sensor with 1mm x 1.1mm (0.039in x 0.043in). When you first use the module, you need to solder some header pins.

This sensor communicates via I2C.

To control the BMP180 barometric sensor, you need to install the "SFE_BMP180.h" library.

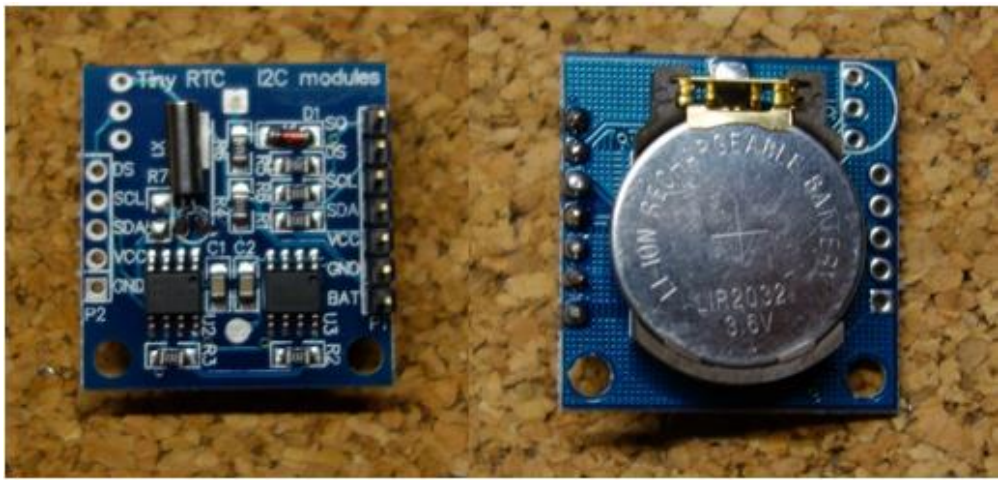
Installing the SFE_BMP180 Library

- Click here to download the SFE_BMP180 library. You should have a .zip folder in your Downloads folder
- Unzip the .zip folder and you should get BMP180_Breakout_Arduino_Library-master folder
- Rename your folder from ~~BMP180_Breakout_Arduino_Library-master~~ to BMP180_Breakout
- Move the BMP180_Breakout folder to your Arduino IDE installation libraries folder
- Finally, re-open your Arduino IDE

RTC (Real Time Clock) module

I'm using the DS1307 RTC module. However, this project works just fine with the DS3231, which is very similar. One main difference between them is the accuracy. The DS3231 is much more accurate than the DS1307.

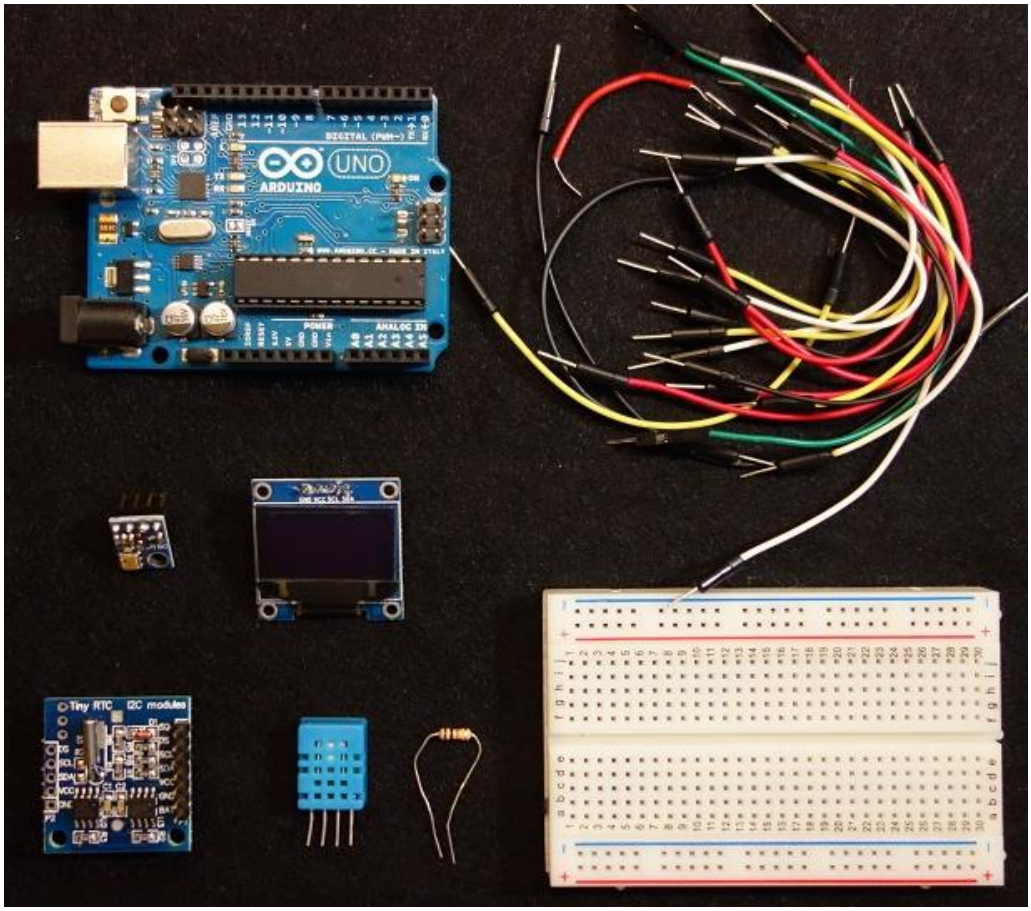
The DS1307 RTC is the one in the following figure. When you first use this module, you need to solder some header pins.



As you can see, the module has a backup battery installed. This allows the module to retain the time, even when it's not being powered up.

This module uses I2C communication.

Parts required



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Arduino UNO	http://ebay.to/1SQda0R
	Breadboard	http://ebay.to/21bEojM
	OLED display	http://ebay.to/2ddDlPz
	DHT11 temperature and humidity sensor	http://ebay.to/2ddDgLv

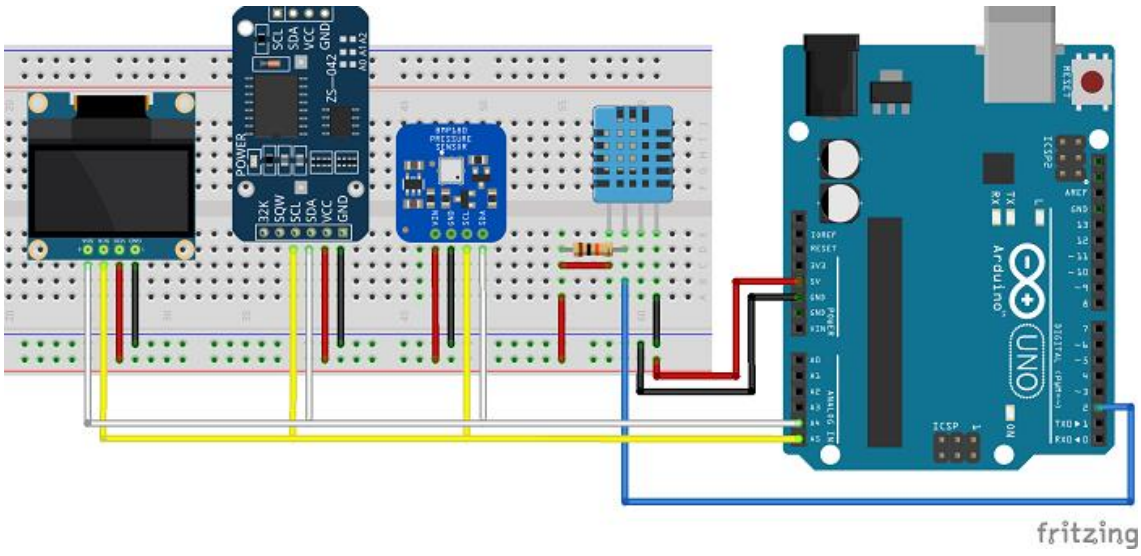
	1x 10kOhm Resistor	http://ebay.to/1KsMYFP
	BMP180 barometric sensor	http://ebay.to/2cw6qRC
	DS1307 Real Time Clock	http://ebay.to/2d3fGRl
	Jumper Wires	http://ebay.to/1PXeaJz

Building the weather station

Now, you have everything prepared to start building your weather station.

Schematics

Assemble the circuit by following the schematics below.



Note: be careful if you're not using the same components. For example, if you're using other model of the BMP180 barometric sensor, make sure you check out its input voltage. Some sensors should be powered up using 3.3V instead of 5V.

Code

You need to modify the code provided to set your own parameters. You need to set the altitude and the current time. Everything is explained after the code. Don't upload the code until reading everything in this section.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

#include <Wire.h> //for I2C communication
#include <Adafruit_GFX.h> // include library for the OLED
#include <Adafruit_SSD1306.h> // include library for the OLED
#include <SFE_BMP180.h> // include library for the pressure sensor
#include <DHT.h> //include library for the temperature and humidity
sensor

#define DHTPIN 2     // pin that DTH11 is connected
#define DHTTYPE DHT11   // DHT 11
#define OLED_RESET 4    //OLED
#define DS3231_I2C_ADDRESS 0x68
Adafruit_SSD1306 display(OLED_RESET);
#define ALTITUDE 100 // set with your altitude

SFE_BMP180 pressure;

// Initialize DHT sensor for normal 16mhz Arduino
DHT dht(DHTPIN, DHTTYPE);

// Convert normal decimal numbers to binary coded decimal (used to
convert time)
byte decToBcd(byte val){
  return( (val/10*16) + (val%10) );
}
// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val){
  return( (val/16*10) + (val%16) );
}

void setup(){
  if (pressure.begin())
    Serial.println("BMP180 init success");
  else {
    // Oops, something went wrong, this is usually a connection
problem,
    Serial.println("BMP180 init fail\n\n");
    while(1); // Pause forever.
  }
  Wire.begin();  // initialize wire communication
  dht.begin();  //initialize  dht sensor
  Serial.begin(9600); // initialize  serial monitor
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //initialize display

  // set the initial time here:
  // DS3231 seconds, minutes, hours, day, date, month, year
  // uncomment the following line after setting up the time once
  setDS3231time(33,21,11,5,29,9,16);
}

void setDS3231time(byte second, byte minute, byte hour, byte
dayOfWeek, byte dayOfMonth, byte month, byte year){
  //set the time and date data to DS3231
  Wire.beginTransmission(DS3231_I2C_ADDRESS);
```

```
  Wire.write(0); // set next input to start at the seconds register
  Wire.write(decToBcd(second)); // set seconds
  Wire.write(decToBcd(minute)); // set minutes
  Wire.write(decToBcd(hour)); // set hours
  Wire.write(decToBcd(dayOfWeek)); // set day of week (1=Sunday,
7=Saturday)
  Wire.write(decToBcd(dayOfMonth)); // set date (1 to 31)
  Wire.write(decToBcd(month)); // set month
  Wire.write(decToBcd(year)); // set year (0 to 99)
  Wire.endTransmission();
}
void readDS3231time(byte *second, byte *minute, byte *hour, byte
*dayOfWeek, byte *dayOfMonth, byte *month, byte *year){
  Wire.beginTransmission(DS3231_I2C_ADDRESS);
  Wire.write(0); // set DS3231 register pointer to 00h
  Wire.endTransmission();
  Wire.requestFrom(DS3231_I2C_ADDRESS, 7);
  // request seven bytes of data from DS3231 starting from register
00h
  *second = bcdToDec(Wire.read() & 0x7f);
  *minute = bcdToDec(Wire.read());
  *hour = bcdToDec(Wire.read() & 0x3f);
  *dayOfWeek = bcdToDec(Wire.read());
  *dayOfMonth = bcdToDec(Wire.read());
  *month = bcdToDec(Wire.read());
  *year = bcdToDec(Wire.read());
}

// Displays data and time on the OLED display
void displayTime(){
  byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
  // retrieve data from DS3231
  readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth,
&month, &year);
  display.clearDisplay(); //clearing the display
  display.setTextColor(WHITE); //setting the color to white
  display.setTextSize(1); //setting the color size
  display.setCursor(0,0); //set the cursor coordinates
  // dispaying the day of the week
  switch(dayOfWeek){
    case 1:
      display.print("Sunday");
      break;
    case 2:
      display.print("Monday");
      break;
    case 3:
      display.print("Tuesday");
      break;
    case 4:
      display.print("Wednesday");
      break;
    case 5:
      display.print("Thursday");
      break;
    case 6:
      display.print("Friday");
      break;
```

```
    case 7:
     display.print("Saturday");
      break;
  }
  display.setCursor(0,20); // setting the cursor coordinates
  display.print(hour, DEC); //display the hour
  // convert the byte variable to a decimal number when displayed

  display.print(":");
  if (minute<10){
    display.print("0");
  }
  display.print(minute, DEC);
  display.print(":");
  if (second<10){
    display.print("0");
  }
  display.print(second, DEC);
  display.setCursor(0,10);
  display.print(dayOfMonth, DEC);
  display.print("/");
  display.print(month, DEC);
  display.print("/");
  display.print(year, DEC);
}
// Displays the temperature and humidity
void displayTempHumid(){

  delay(2000);
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very
slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
   display.clearDisplay();
   display.setTextColor(WHITE);
   display.setTextSize(1);
   display.setCursor(5,0);
   display.print("Failed to read from DHT sensor!");
   return;
  }
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0,0);
  display.print("Humidity: ");
  display.print(h);
  display.print(" %\t");

  // Comments this section and uncomment below if you prefer to
display temperature in Fahrenheit
  display.setCursor(0,10);
```

```
  display.print("Temp: ");
  display.print(t);
  display.print(" C");

  // Uncomment this part if you want to display the temperature in
Fahrenheit
  /*display.setCursor(0,10);
  display.print("Temp: ");
  display.print(t*1.8+32);
  display.print(" F"); */
}

// Displays pressure
void displayPressure(){
  char status;
  double T,P,a;
  status = pressure.startTemperature();
  if (status != 0){
    // Wait for the measurement to complete:
    delay(status);
    // Retrieve the completed temperature measurement:
    // Note that the measurement is stored in the variable T.
    // Function returns 1 if successful, 0 if failure.
    status = pressure.getTemperature(T);
    if (status != 0){
      // Start a pressure measurement:
      // The parameter is the oversampling setting, from 0 to 3
(highest res, longest wait).
      // If request is successful, the number of ms to wait is
returned.
      // If request is unsuccessful, 0 is returned.

      status = pressure.startPressure(3);
      if (status != 0){
        // Wait for the measurement to complete:
        delay(status);

        // Note that the measurement is stored in the variable P.
        // Note also that the function requires the previous
temperature measurement (T).
        // (If temperature is stable, you can do one temperature
measurement for a number of pressure measurements.)
        // Function returns 1 if successful, 0 if failure.

        status = pressure.getPressure(P,T);
        if (status != 0){
          // Print out the measurement:
          display.setTextColor(WHITE);
          display.setTextSize(1);
          display.setCursor(0,20);
          display.print("Pressure: ");
          display.print(P,2);
          display.print(" mb ");
        }
        else Serial.println("error retrieving pressure
measurement\n"); //for debugging
      }
      else Serial.println("error starting pressure measurement\n");
```

```
//for debugging
    }
    else Serial.println("error retrieving temperature
measurement\n"); //for debugging
  }
  else Serial.println("error starting temperature measurement\n");
//for debugging
}
void loop(){
  displayTempHumid();
  displayPressure();
  display.display();
  delay(3000);
  displayTime();
  display.display();
  delay(3000);
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/6_weather_station.ino

Setting the altitude

In the code provided you need to set up the altitude of your weather station (an approximated value is fine). This is an important step to calibrate the barometric sensor.

Go to elevationmap.net, insert your address and check your altitude's location. Set your altitude in the code. The place where you should write your altitude is commented.

```
#define ALTITUDE 100 // set with your altitude
```

Replace the "100" with your altitude in meters.

Setting the time

For setting the current time you do the following:

- set your current time in the function setDS3231time()

```
// set the initial time here:
// DS3231 seconds, minutes, hours, day, date, month, year
setDS3231time(33,53,16,1,25,9,16);
```

These are the parameters for the function: seconds, minutes, hours, day of the week, day, month and year (in this order). Sunday is the 1 day of the week and Saturday is 7.

- upload the code
- comment the function that sets the time

```
// set the initial time here:
// DS3231 seconds, minutes, hours, day, date, month, year
//setDS3231time(33,53,16,1,25,9,16);
```

- upload the commented code.

This is a very important step to set up the time in your RTC. If you don't do this, every time your RTC resets, it will not know the current time.

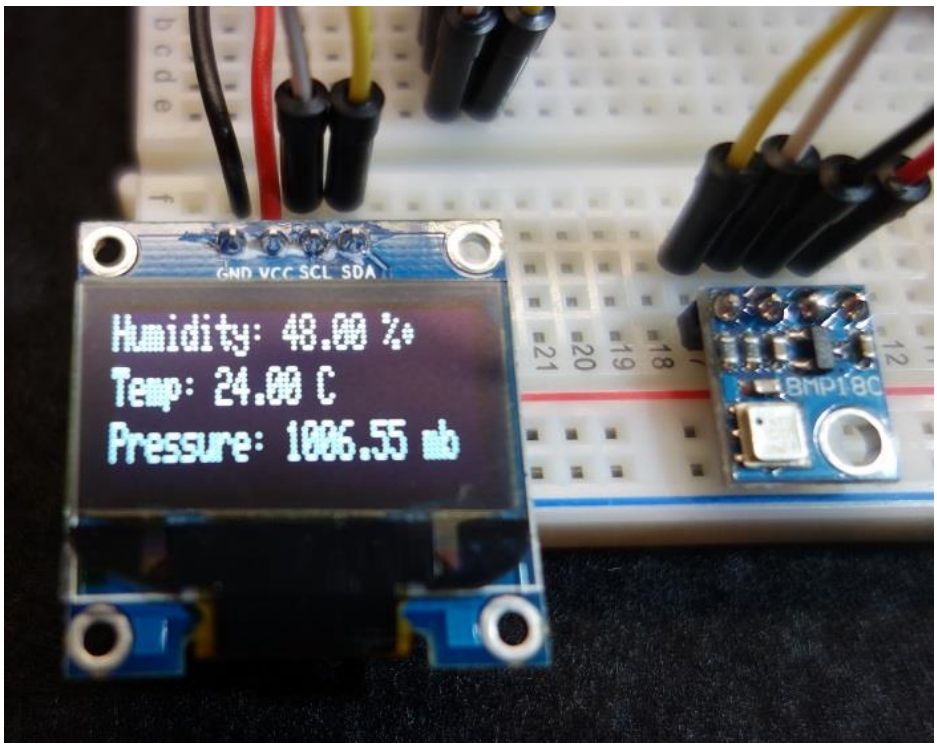
Troubleshooting

If your code doesn't compile, or if the weather station is not working properly, check the following:

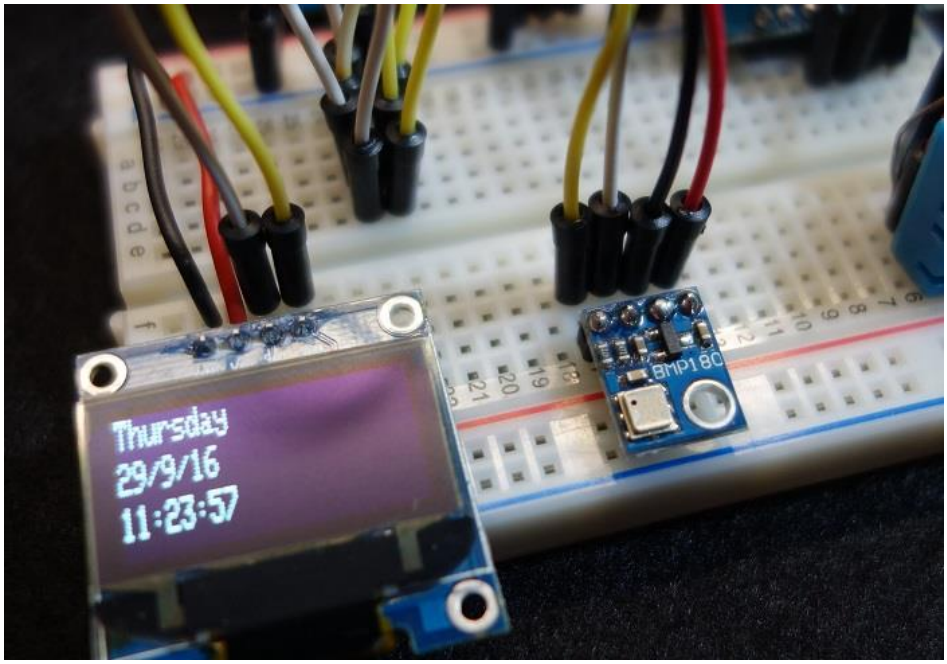
- make sure you have all the needed libraries installed: Adafruit_GFX, Adafruit_SSD1306, DHT and SFE_BMP180.
- make sure you have set the time in your RTC module.
- make sure you edited the code to set up your altitude.

Demonstration

In the end, you have humidity temperature and pressure displayed in the OLED for a couple of seconds.



Then, the current time and date is also displayed until it starts over again with the weather information.



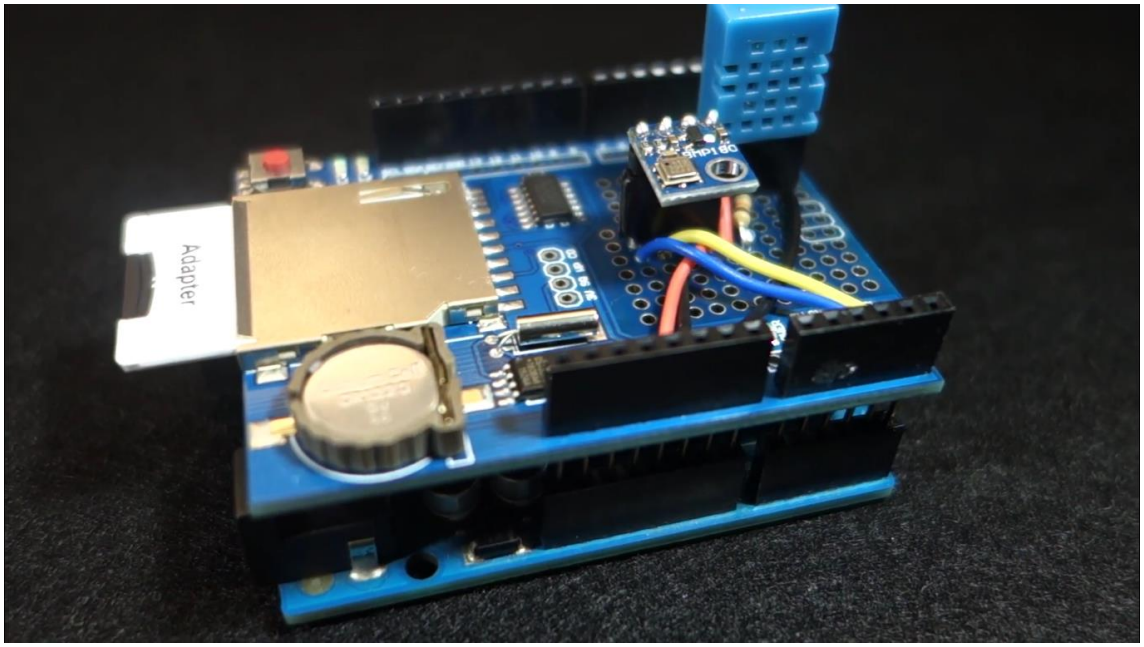
Wrapping up

This is a good project to get you acquainted with the OLED display, barometric sensor, temperature and humidity sensor and the RTC module.

This is a great project to practice with sensors/modules that require I2C communication.

You can build your own version of the weather station by adding more sensors or changing the OLED display layout.

Weather Station – Data Logging to Excel



Weather Station – Data Logging to Excel

Level: Intermediate – Time 1h 15 minutes

In this project you're going to build a weather station data logger.

You will save your temperature, humidity and pressure data on an SD card with time stamps.

Then, the data collected on the SD card will be processed using Microsoft Excel.

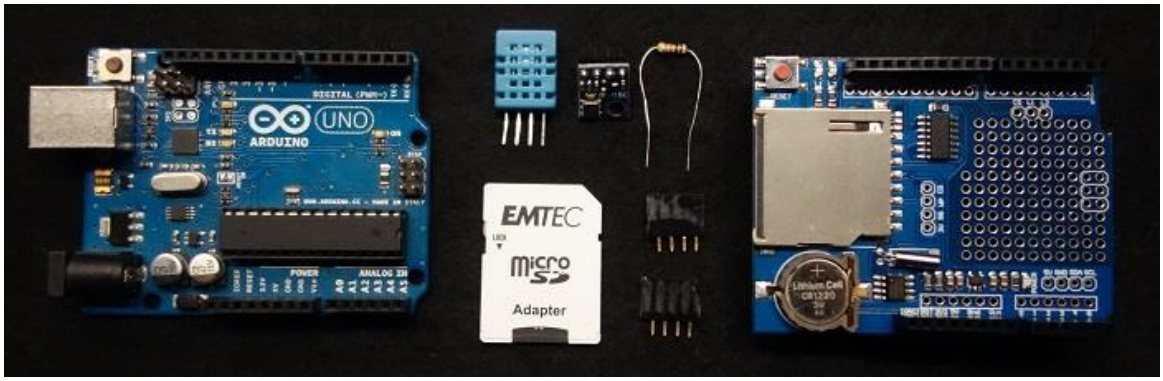
Data Logging Shield

There are several ways of building a data logger with Arduino. In this project, you're going to use a data logging shield as shown in the figure below:

Here are the main features of the data logging shield:

- it comes with an SD card slot. It accepts SD cards and not microSD cards. If you have a microSD card, you need to use an SD card adapter.
- It has with a built-in RTC with a battery holder for a CR1220 3V lithium metal coin cell battery. This battery is needed so that the RTC keeps time, even when the Arduino is not being powered or when it resets.
- It has a prototyping area for soldering connections, sensors, circuits, etc.

Parts required

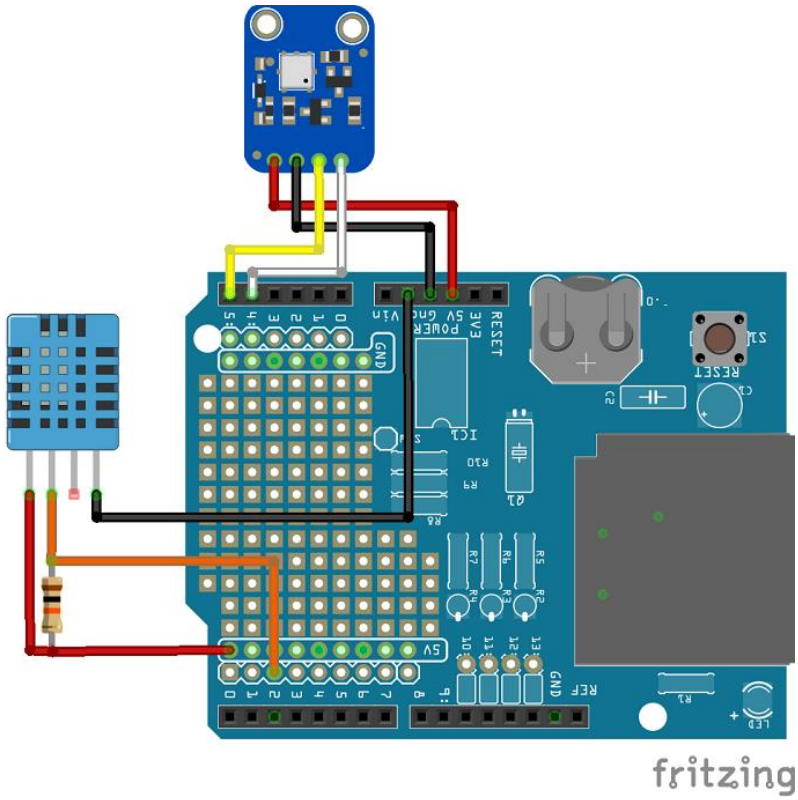


Grab all the components needed for this project. If you don't have them, just visit the links in the table below.

Figure	Name	eBay
	Arduino UNO	http://ebay.to/1SQda0R
	Breadboard	http://ebay.to/21bEojM
	XD-05 Data logging shield	http://ebay.to/2etdmTN
	SD Card	http://ebay.to/2fJos93
	DHT11 temperature and humidity sensor	http://ebay.to/2ddDgLv
	1x 10kOhm Resistor	http://ebay.to/1KsMYFP
	BMP180 barometric sensor	http://ebay.to/2cw6qRC
	Jumper Wires	http://ebay.to/1PXeaJz

Schematics

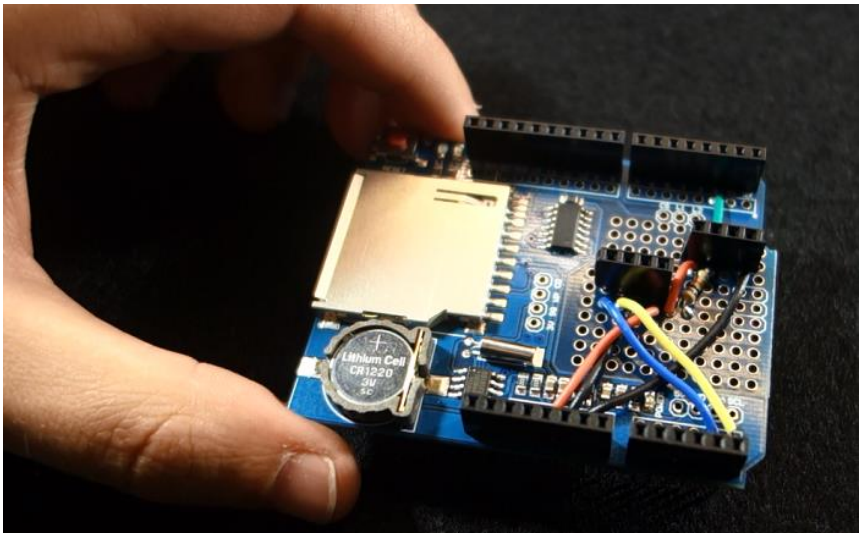
Wire the circuit as in the schematics below.



In the schematics, the parts are out of the prototyping area so that you can clearly see the connections.

The idea is that you solder your sensors to the prototyping area in order to have a permanent weather station data logger. I encourage you to use headers to attach your sensors.

Here's how my shield looks after soldering.

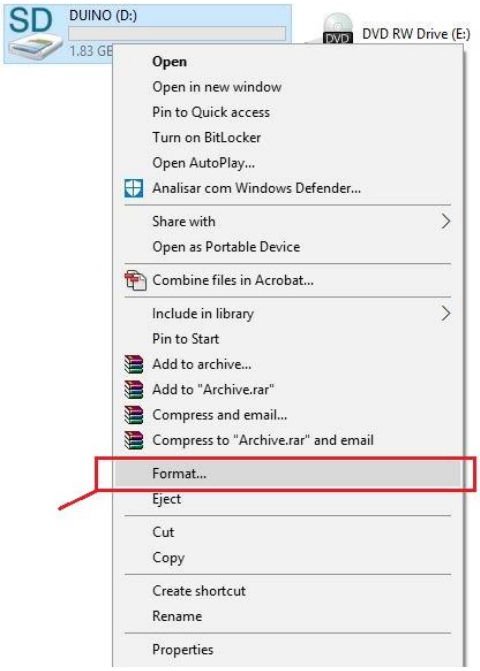


Note: if you don't want to solder the circuit to the prototyping area, you can use a breadboard.

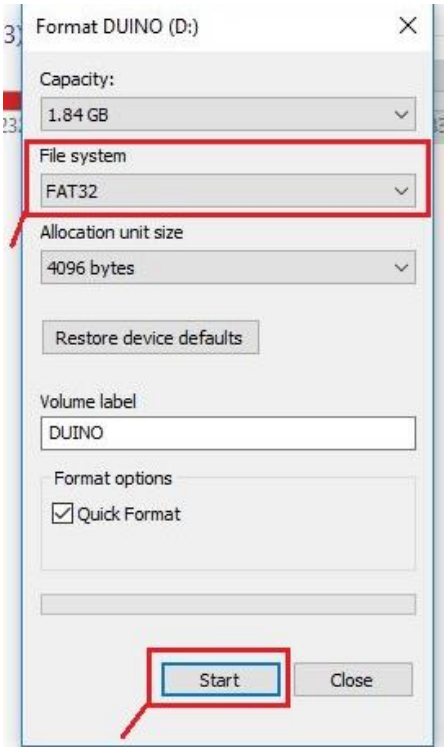
Preparing your SD Card

Your SD card needs to be formatted before use.

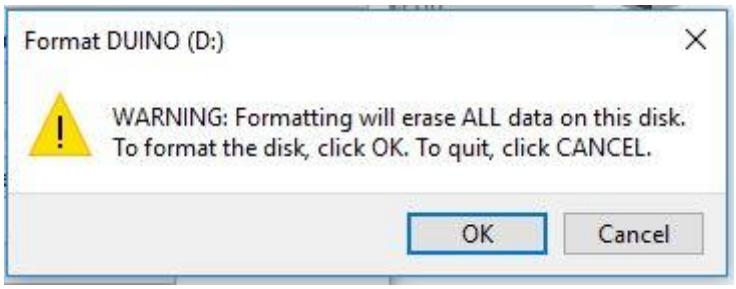
Go to "My Computer" folder, right-click on the SD card icon and select "Format...".



Select File system "FAT32" and press the "Start" button.



A warning message pops-up:



Press “OK”, wait a few seconds and your card is ready to use.

Code

Upload the following code to your Arduino board:

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

// Example testing sketch for various DHT humidity/temperature
sensors
#include <SPI.h>
#include <SD.h>
#include <DHT.h>
#include <RTClib.h>
#include <SFE_BMP180.h>

//how many milliseconds between grabbing data and logging it
#define LOG_INTERVAL 2000

//define DHT pin
#define DHTPIN 2     // what pin we're connected to

// Uncomment whatever type you're using!
#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT22   // DHT 22  (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

// Initialize DHT sensor for normal 16mhz Arduino
DHT dht(DHTPIN, DHTTYPE);

// change this to match your SD shield or module;
// Arduino Ethernet shield: pin 4
// Data loggin SD shields and modules: pin 10
// Sparkfun SD shield: pin 8
const int chipSelect = 10;

//creating a file to store the data
File myFile;

//For the RTC
RTC_DS1307 rtc;
```

```
//You need to create an SFE_BMP180 object, here called "pressure":
SFE_BMP180 pressure;

#define ALTITUDE 100 // Set here your altitude

void setup() {

  //initializing the DHT sensor
  dht.begin();

  //initializing Serial monitor
  Serial.begin(9600);

  // setup for the RTC
  while (!Serial); // for Leonardo/Micro/Zero
    if (! rtc.begin()) {
      Serial.println("Couldn't find RTC");
      while (1);
    }
    else{
      // following line sets the RTC to the date & time this sketch
was compiled
      //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    }
    if (! rtc.isrunning()) {
      Serial.println("RTC is NOT running!");
    }
    Serial.print("Initializing SD card...");

    if (!SD.begin(chipSelect)) {
      Serial.println("initialization failed!");
      return;
    }
    Serial.println("initialization done.");

    if(pressure.begin())
      Serial.println("BMP180 init success");
    else {
      // Oops, something went wrong, this is usually a connection
problem,
      // see the comments at the top of this sketch for the proper
connections.
      Serial.println("BMP180 init fail\n\n");
      while(1); // Pause forever.
    }

  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.

  myFile=SD.open("DATA.txt", FILE_WRITE);

  // if the file opened ok, write to it:
  if (myFile) {
    Serial.println("File opened ok");
    myFile.println("Date,Time,Temperature °C,Humidity %,Pressura
hPa");
  }
  myFile.close();
```

```
}

void loggingTime(){
  DateTime now = rtc.now();
  myFile = SD.open("DATA.txt", FILE_WRITE);
  if (myFile) {
    myFile.print(now.year(), DEC);
    myFile.print('/');
    myFile.print(now.month(), DEC);
    myFile.print('/');
    myFile.print(now.day(), DEC);
    myFile.print(" ");
    myFile.print(now.hour(), DEC);
    myFile.print(':');
    myFile.print(now.minute(), DEC);
    myFile.print(':');
    myFile.print(now.second(), DEC);
    myFile.print(",");
  }
  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.println(now.day(), DEC);
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.println(now.second(), DEC);
  myFile.close();
  delay(1000);
}

void loggingTemperature () {

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very
slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit
  //float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) /*|| isnan(f)*/) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  //debugging purposes
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.println(" %");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C");
  //Serial.print(f);
```

```
  //Serial.println(" *F\t");

  myFile = SD.open("DATA.txt", FILE_WRITE);
  if (myFile) {
    Serial.println("open with success");
    myFile.print(t);
    myFile.print(",");
    myFile.print(h);
    myFile.print(",");
  }
  myFile.close();
}
void loggingPressure(){
  char status;
  double T,P,a;
  status = pressure.startTemperature();
  if (status != 0){
    // Wait for the measurement to complete:
    delay(status);
    // Retrieve the completed temperature measurement:
    // Note that the measurement is stored in the variable T.
    // Function returns 1 if successful, 0 if failure.
    status = pressure.getTemperature(T);
    if (status != 0){
      // Start a pressure measurement:
      // The parameter is the oversampling setting, from 0 to 3
(highest res, longest wait).
      // If request is successful, the number of ms to wait is
returned.
      // If request is unsuccessful, 0 is returned.

      status = pressure.startPressure(3);
      if (status != 0){
        // Wait for the measurement to complete:
        delay(status);

        // Note that the measurement is stored in the variable P.
        // Note also that the function requires the previous
temperature measurement (T).
        // (If temperature is stable, you can do one temperature
measurement for a number of pressure measurements.)
        // Function returns 1 if successful, 0 if failure.

        status = pressure.getPressure(P,T);
        if (status != 0){
          myFile = SD.open("DATA.txt", FILE_WRITE);
          if (myFile) {
            myFile.println(P,2);
            Serial.println("Pressure is ok");
          }
          myFile.close();
        }
        else Serial.println("error retrieving pressure
measurement\n"); //for debugging
      }
      else Serial.println("error starting pressure measurement\n");
//for debugging
    }
```

```
    else Serial.println("error retrieving temperature
measurement\n"); //for debugging
  }
  else Serial.println("error starting temperature measurement\n");
//for debugging

}

void loop() {
  loggingTime();
  loggingTemperature();
  loggingPressure();
  delay(5000);
}
```

Source code

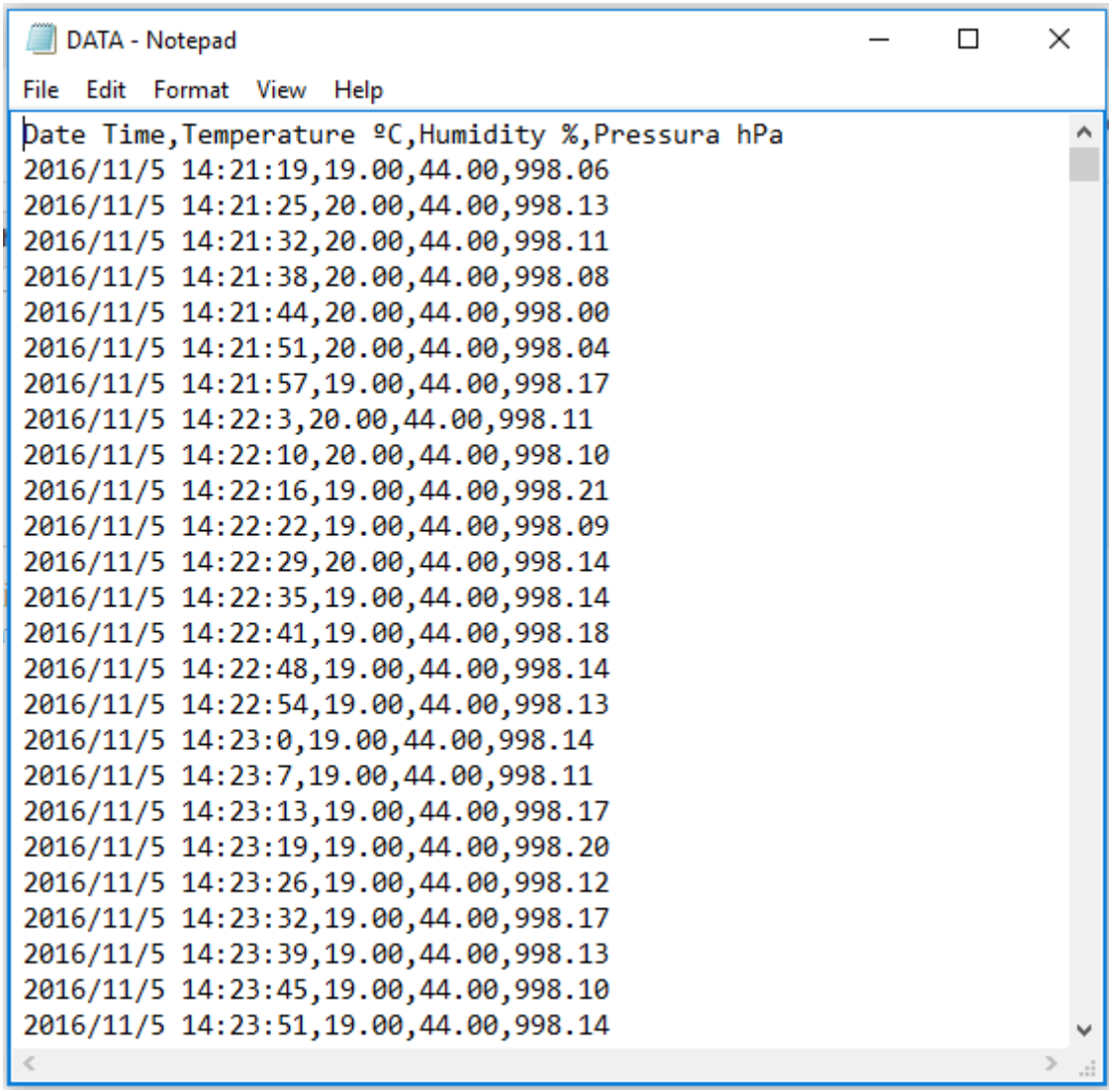
https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/17_weather_station_datalogger.ino

Getting Data from your SD Card

To get the data collected from your data logger follow these next instructions:

1) Open your SD card. You should have a .txt file with the name that you've given in the code. In my case, it is called DATA.txt.

2) Open the file using your default .txt editor. You have your data in comma separated values (CSV) format.

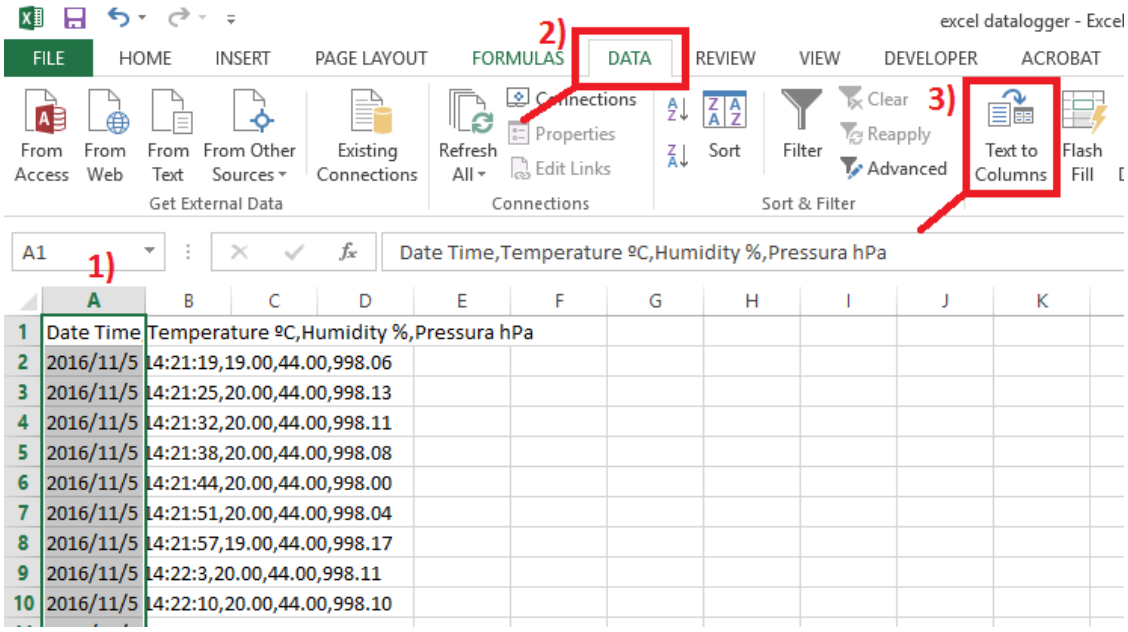


Processing Data using Microsoft Excel

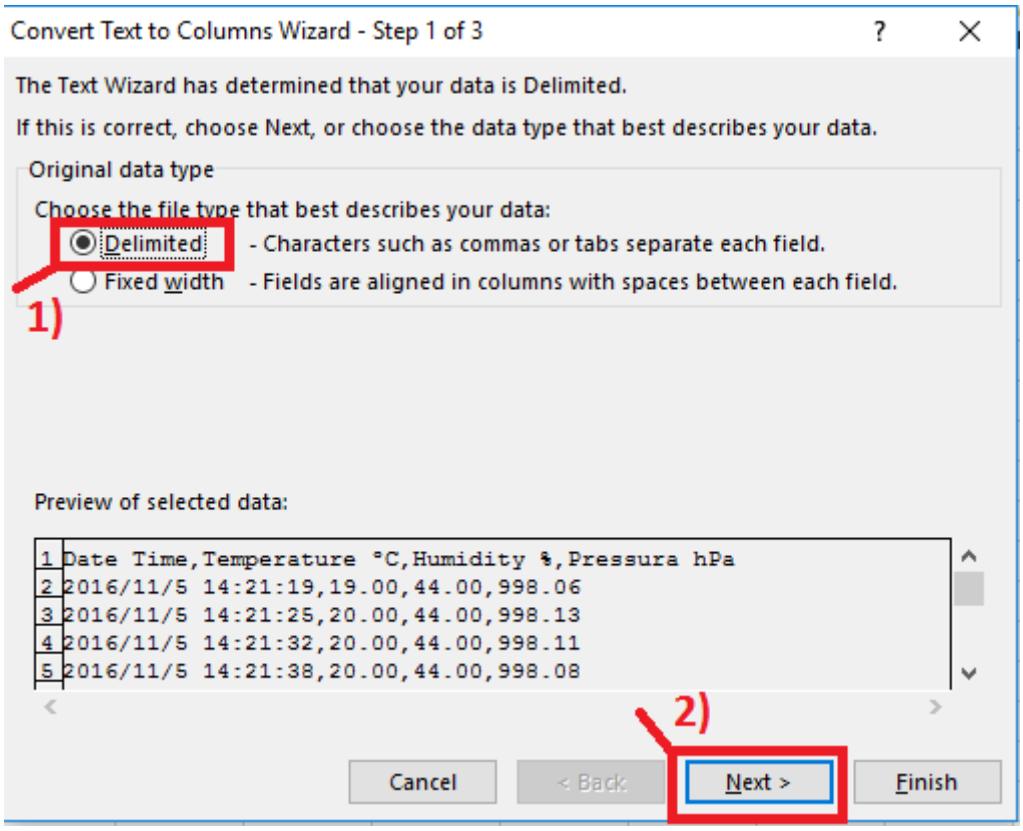
Open a blank Excel Spreadsheet. Copy all data from the .txt file and paste it into a cell on a new Excel Spreadsheet cell.

Note: if you prefer, you can download this sample Excel Spreadsheet and replace the values with your own data.

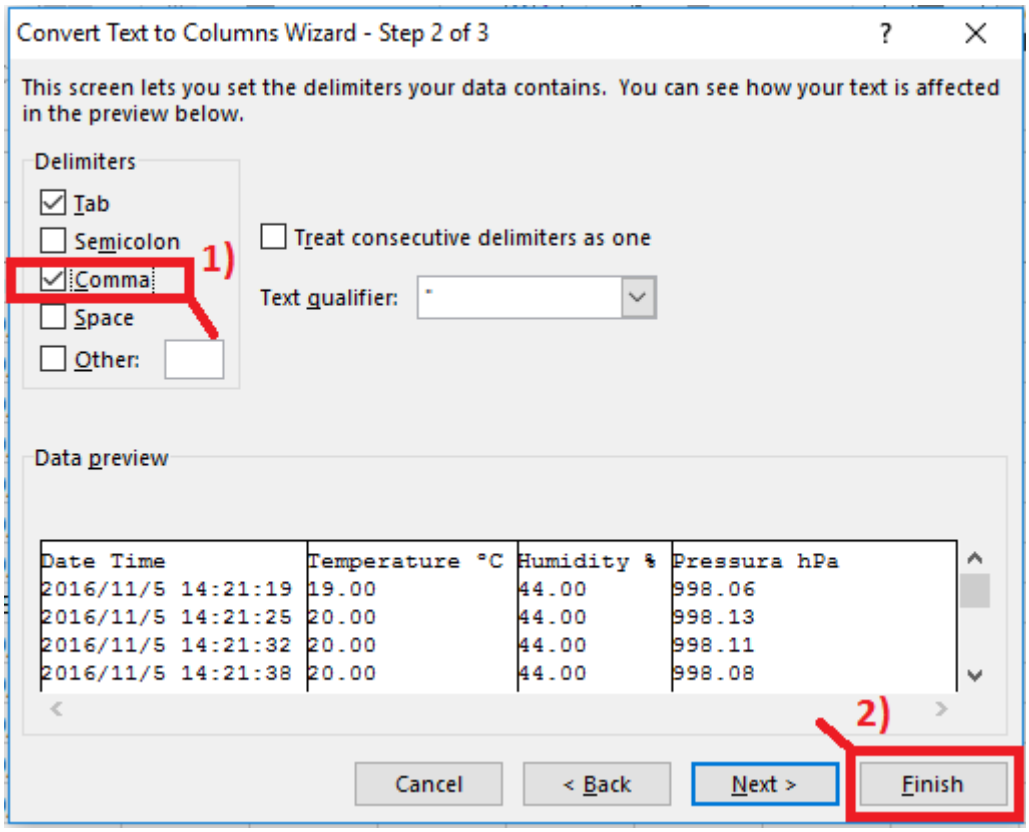
Select the whole column with the data. Select the DATA tab and then select Text to Columns.



Select the Delimited option. Then, press the Next button.



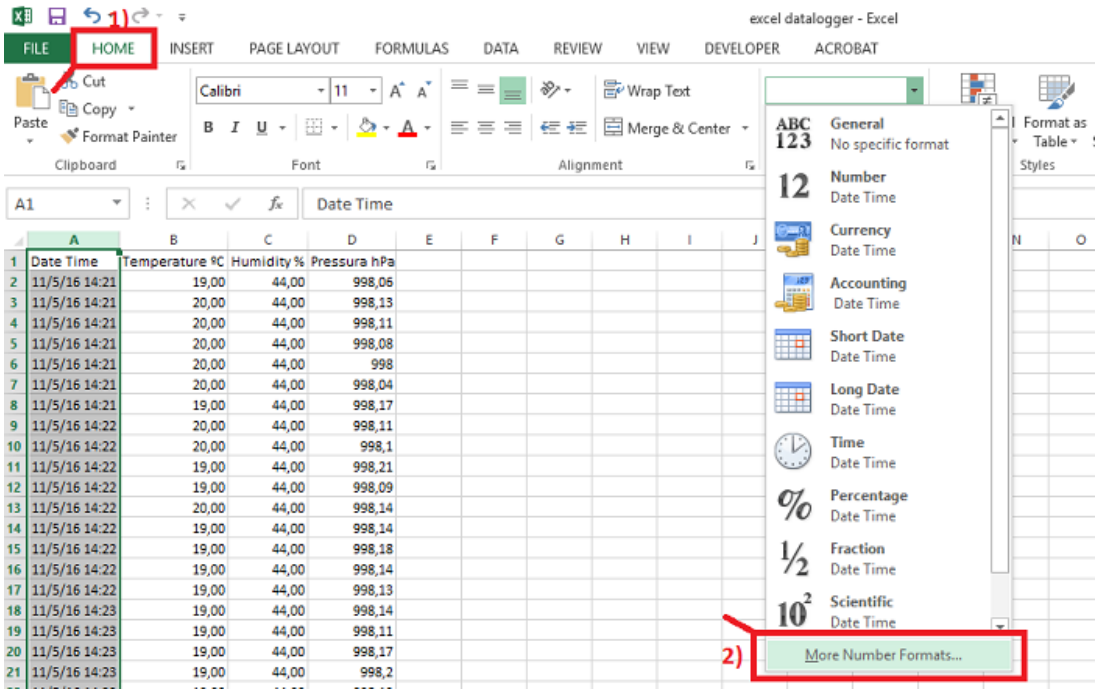
Select Comma and press Finish.



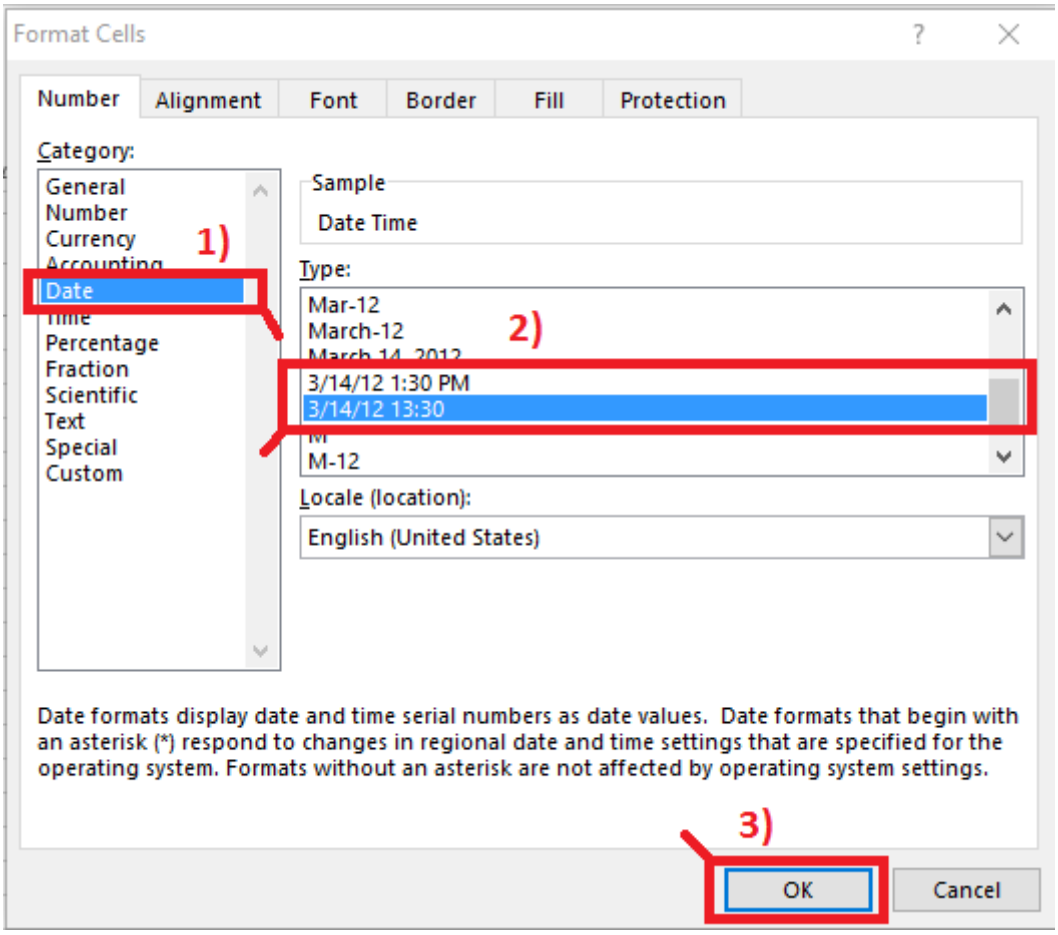
This process splits the data into four different columns: Date+Time, Temperature, Humidity and Pressure.

Then, you need to select the right format for each column. Here's what you need to do next:

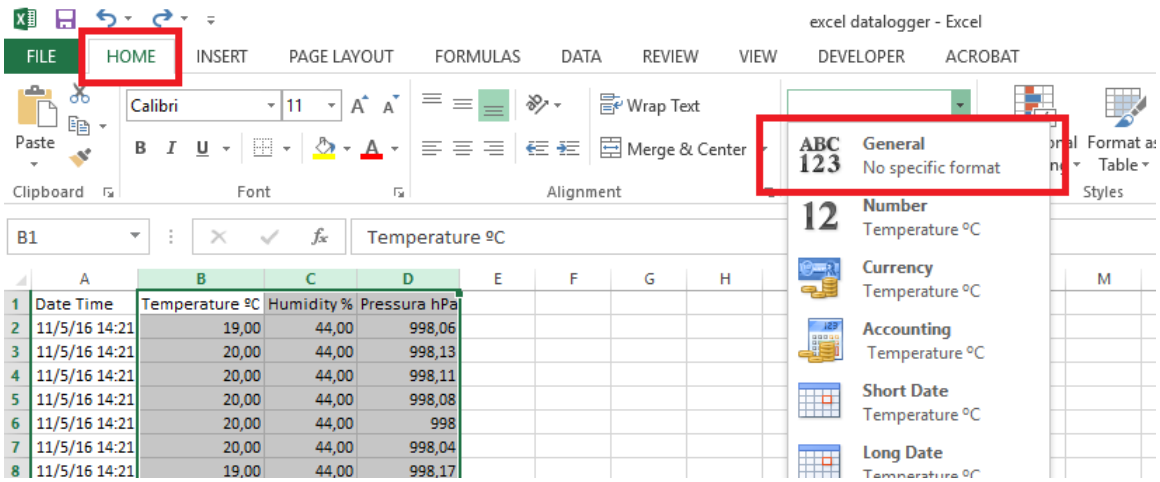
1) Select the first column, and in the Home Tab, go to "More Number Formats".



2) Select Date and choose one of those two data formats accordingly to the time format you prefer: time in 12-hour format or 24-hour format. Press the OK button.



3) For the temperature, humidity and pressure columns, choose the General option format as shown below.



Note: it is important that you know if your Excel uses dots (.) or comma (,) to separate the decimal numbers.

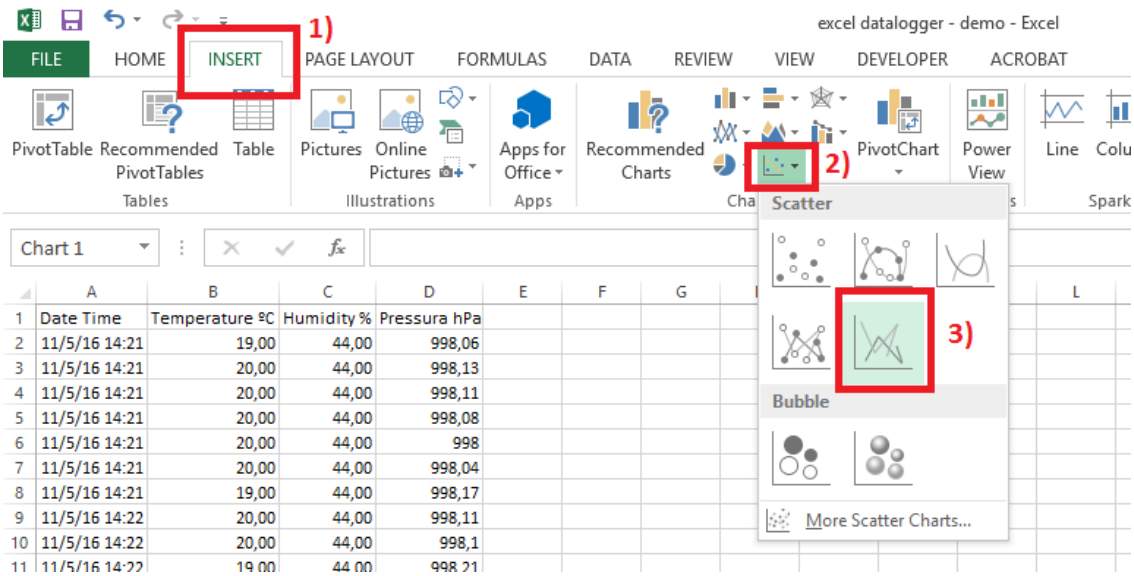
The data collected from the data logger uses points as the decimal separator. If your Excel uses commas, press CRTL+F and replace all dots with commas.

Now, your data is ready to create charts.

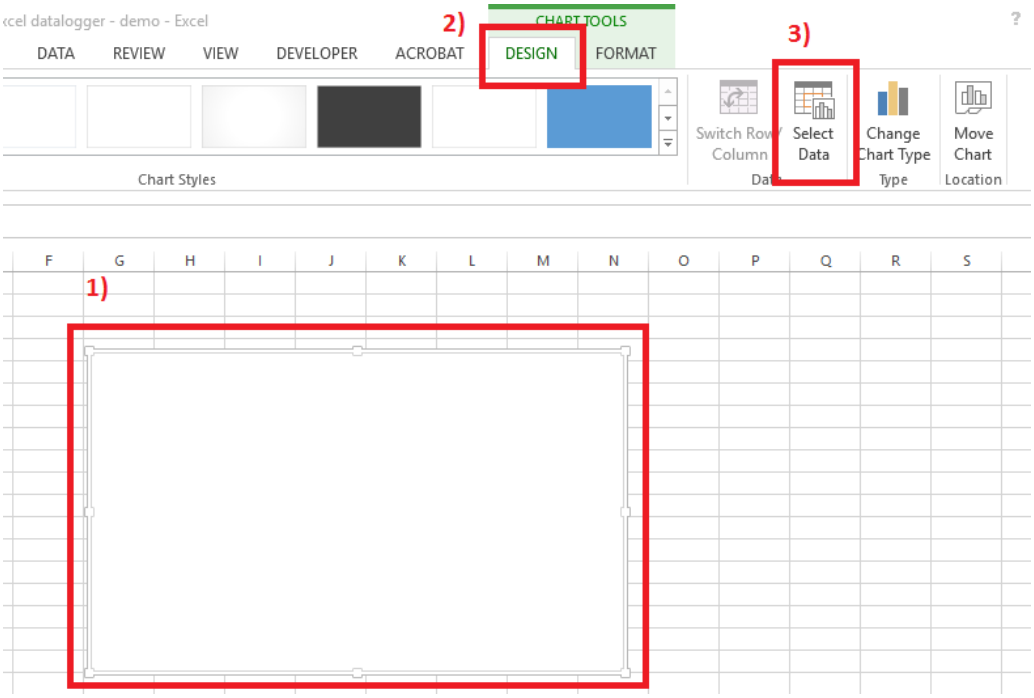
Charts

To create charts with your data do the following:

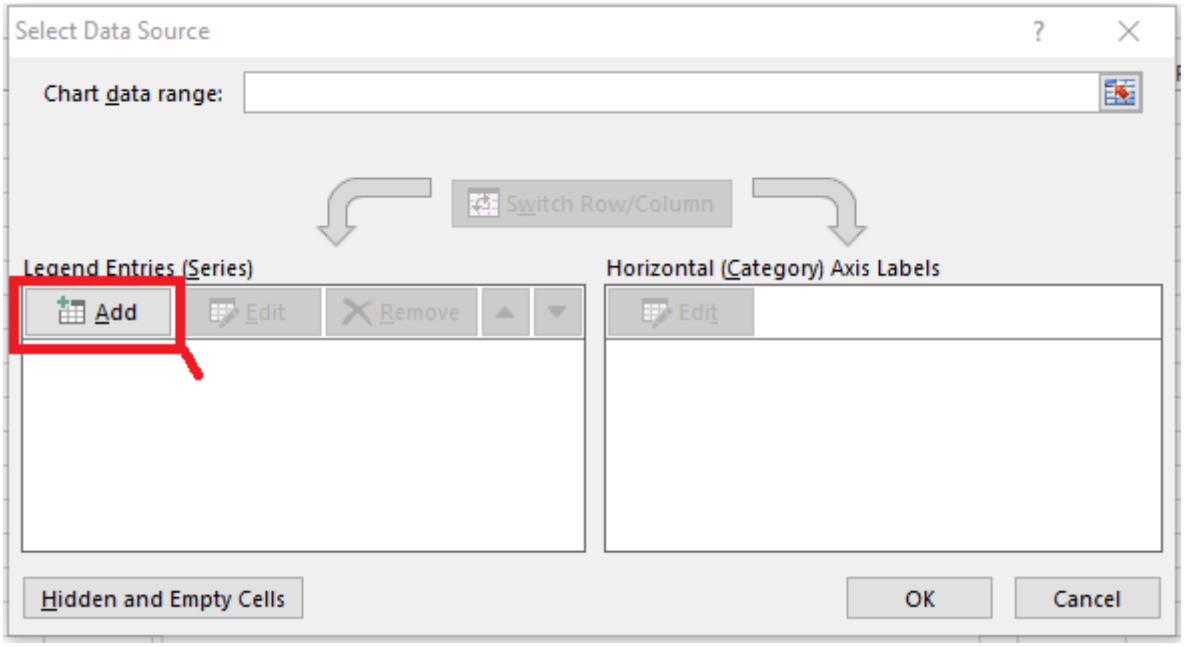
1) Go to Insert tab, Charts and select Scatter with Straight Lines or other format that you like most.



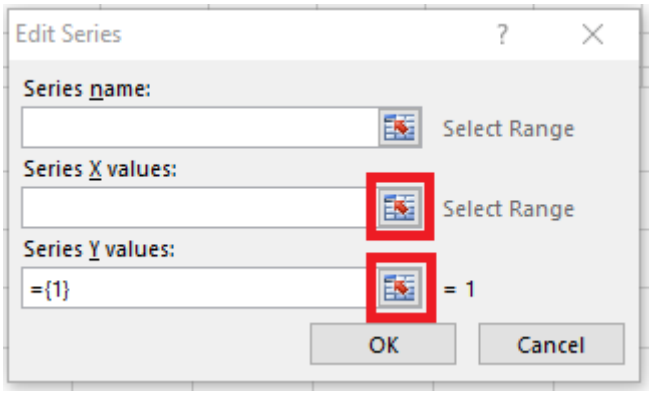
2) A blank rectangle appears – that's your chart. Select the rectangle, open the Design tab and go to Select Data.



3) Then, the following menu pops-up. Press Add.



4) A new window opens:

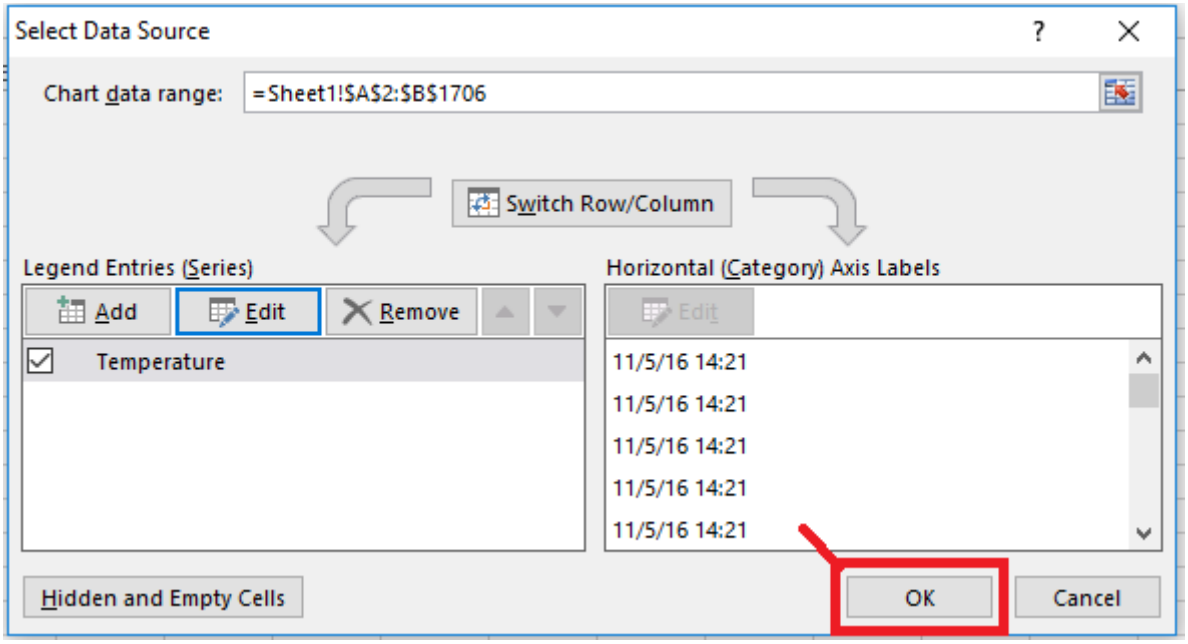


5) In the Series name, you write what you want your series to be called. For example, if you're going to plot the temperature, you can write Temperature.

Then, click on the arrows highlighted in red. In the Series X values select the first column, because the X axis displays time.

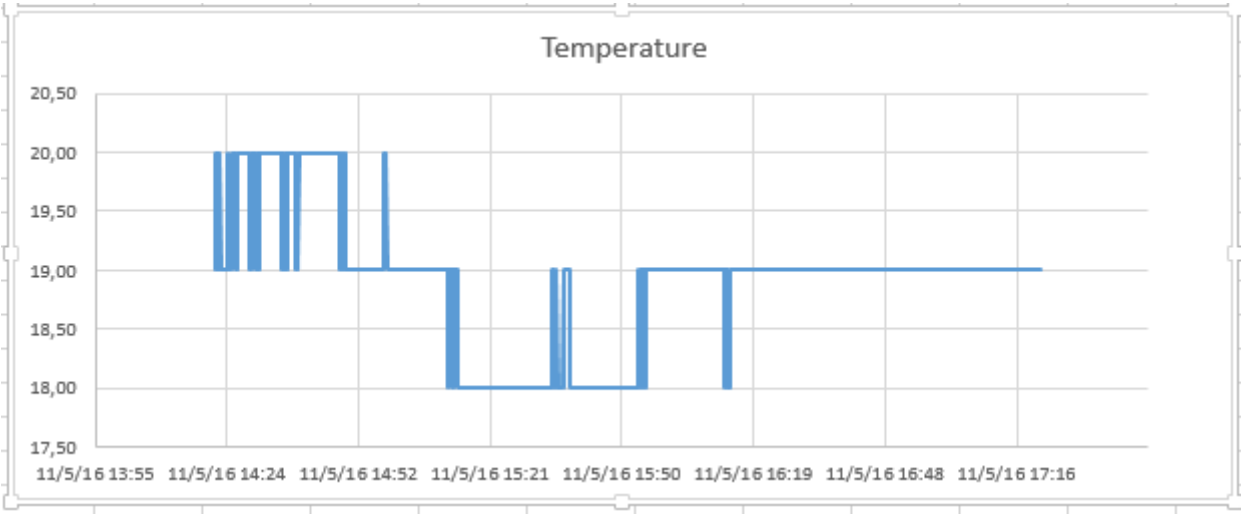
On the Y values you select the parameter you want to display. For example, if you want to plot the temperature, select the column where the temperature values are.

6) Then, press OK and OK again.



Visualizing the chart

Now, you have a chart displaying the temperature. Feel free to play around formatting the layout, axis and other features.



Do the same procedure to create charts with the humidity and pressure readings.

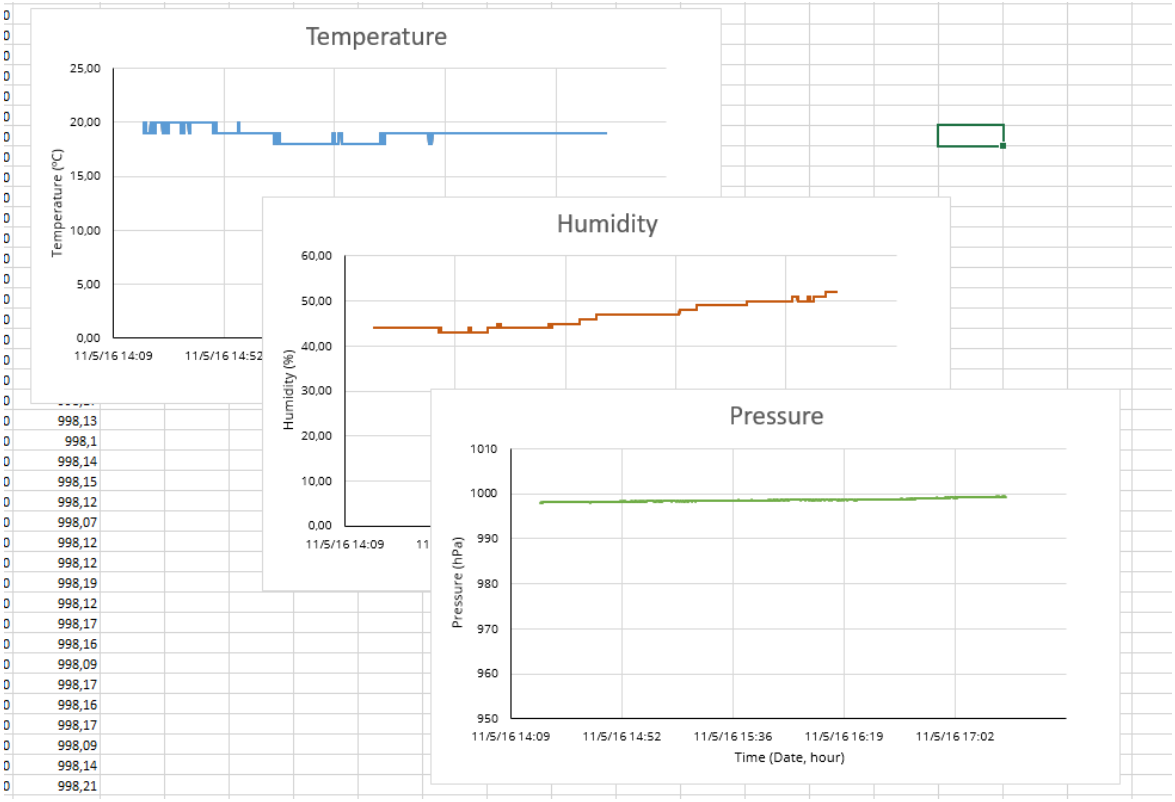
Download sample Excel Spreadsheet

If you prefer, you can download this sample Excel Spreadsheet and replace the values with your own data.

Click here to download the Excel Spreadsheet.

Demonstration

Here's the processed data we've collected during a 3 hour-sampling period.



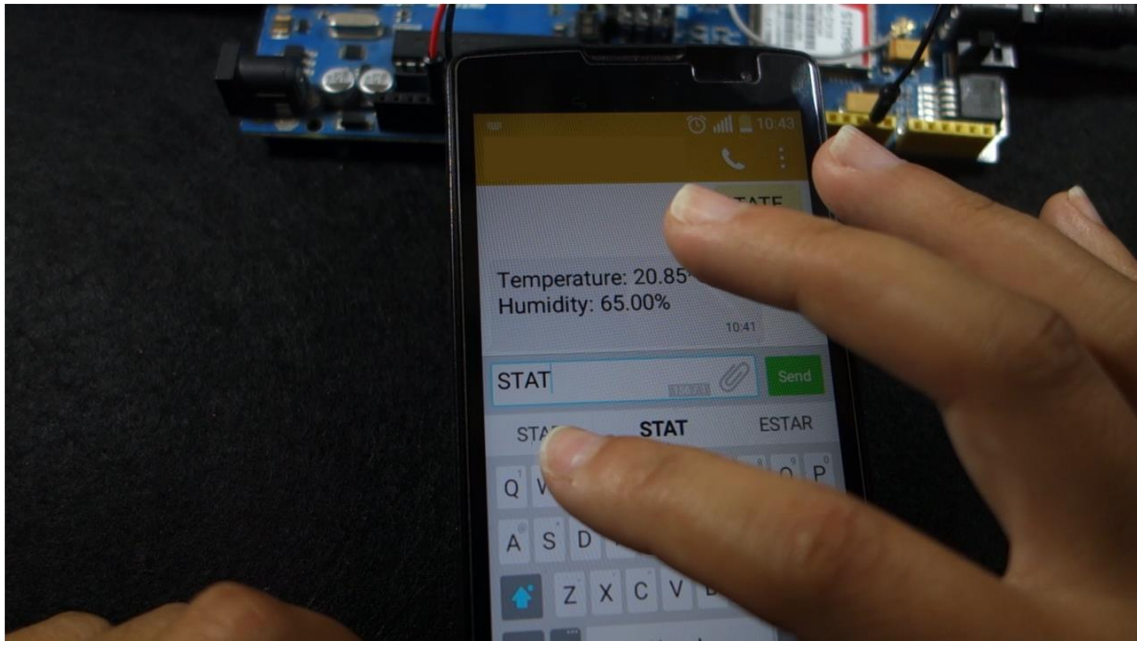
Wrapping up

In this project you've learned how to collected data on an SD card using a data logging shield.

This is a great project to get you started with data logging. There are other ways of logging your data on an SD card like using an SD card module.

Feel free to use this shield with other sensors that are better suited for your project.

Request Sensor Data via SMS



Request Sensor Data via SMS

Level: Intermediate – Time: 1h15 min

In this project you're going to request the temperature and humidity from a DHT11 sensor via SMS.

You're going to use the SIM900 GSM shield to send and receive SMS with the Arduino.

When you send an SMS to the Arduino with the message "STATE", it replies with the latest temperature and humidity readings.

SIM900 GSM Shield

There are several modules you can use to send and receive SMS with the Arduino. We did this project using the SIM900 GSM shield and that's the shield we recommend you to get.

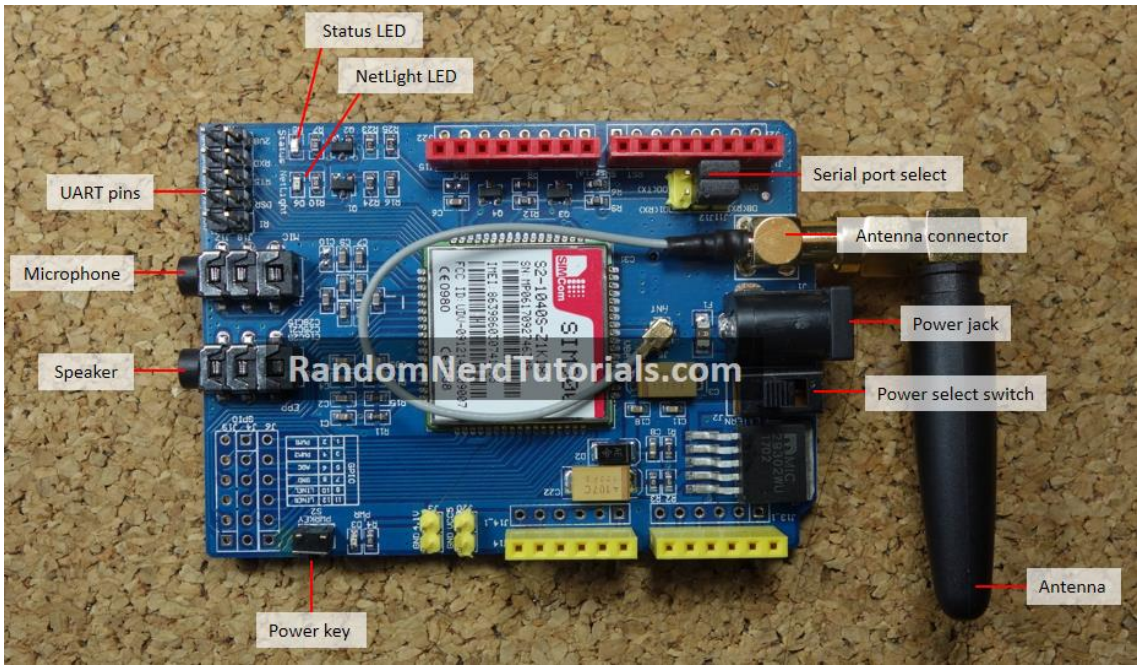
The SIM900 GSM shield is shown in figure below:

Here's the main features of the shield:

- Compatible with Arduino and clones
- Based on SIM900 module from SIMCOM
- Allows you to send SMS, MMS, GPRS and Audio via UART using AT commands.
- It has 12 GPIOs, 2 PWMs and built-in ADC of the SIM900 module

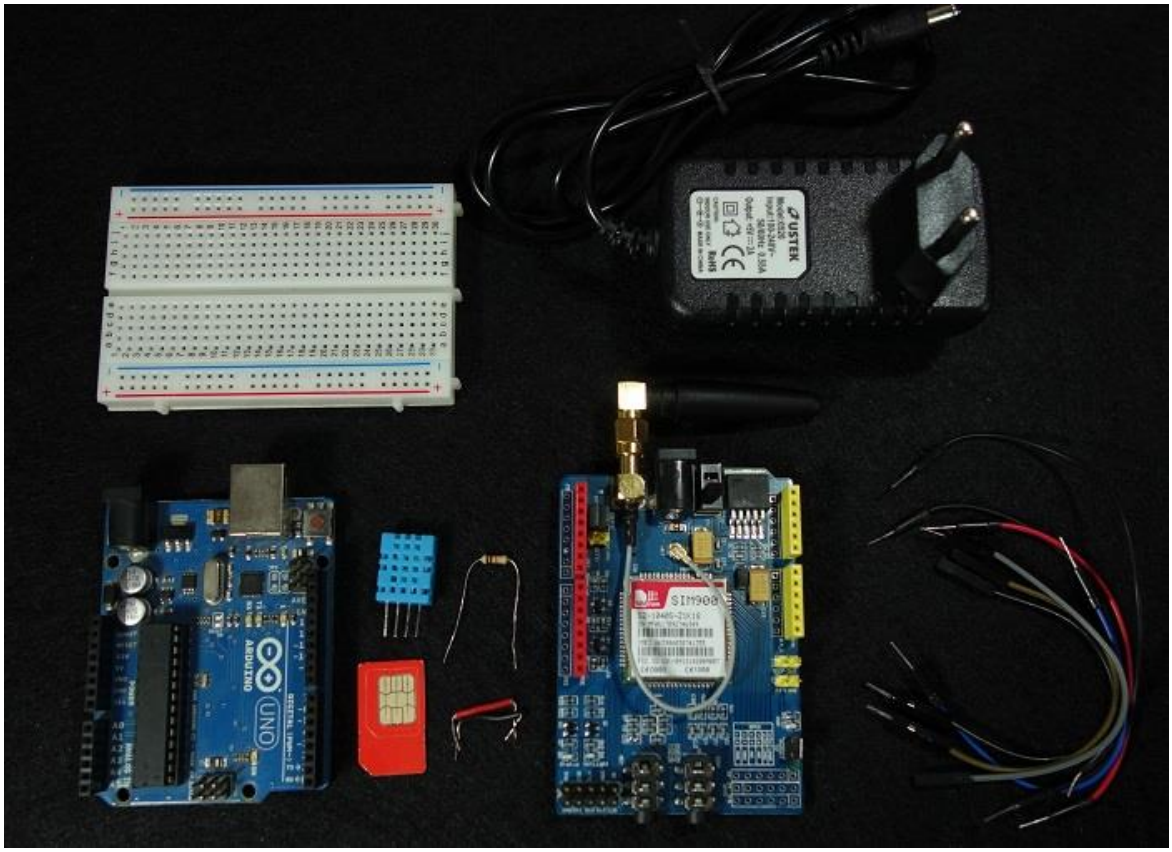
- Quad Band: 850; 900; 1800 and 1900 MHZ, so it should work on GSM networks in all countries
- Control via AT commands
- Supports RTC (real time clock) – it has a holder for a 3V CR1220 battery at the back
- Has microphone and headphone jacks for phone calls

The figure below shows the shield most important components on the board that you need to pay attention to.

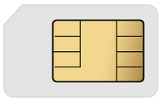


The figure below shows the back of the shield. It has a holder for the SIM card and for a 3V CR1220 battery for the RTC (real time clock).

Parts required



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Arduino UNO	http://ebay.to/1SQda0R
	SIM900 GSM Shield	http://ebay.to/2vPOzSF
	5V 2A Power Adaptor	http://ebay.to/2wLRhJS
	FTDI programmer (optional)	http://ebay.to/2w48aih
	SIM card	

	DHT11 temperature and humidity sensor	http://ebay.to/2ddDgLv
	1x 10kOhm Resistor	http://ebay.to/1KsMYFP
	1x Breadboard	http://ebay.to/21bEojM
	Jumper Wires	http://ebay.to/1PXeaJz

Preliminary steps

Before getting started with your SIM900 GSM module, you need to consider some aspects about the SIM card and the shield power supply.

Prepaid SIM card

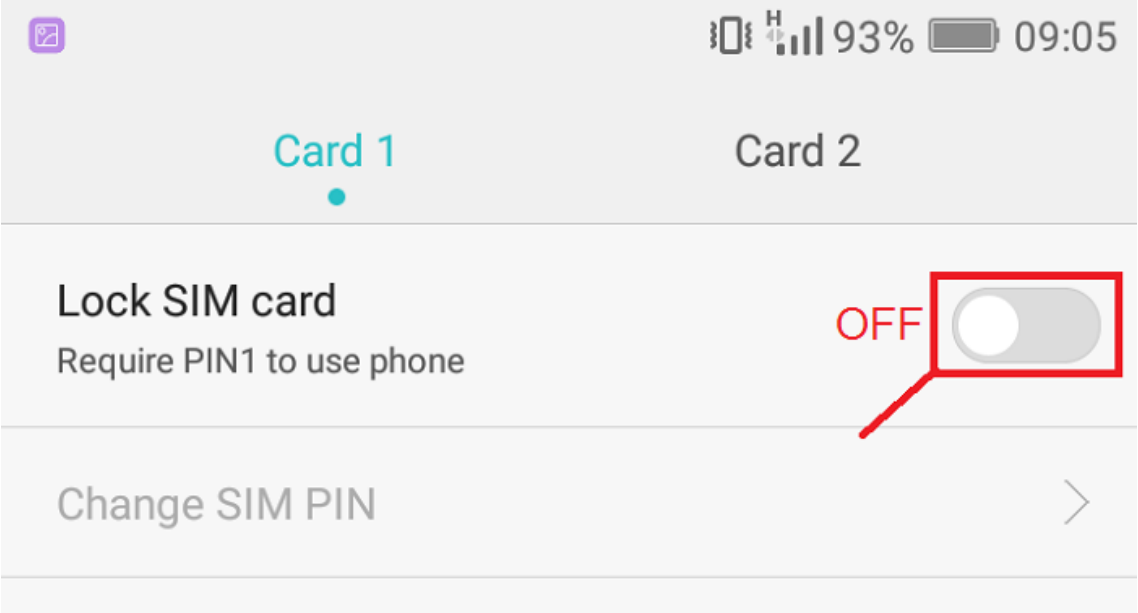
We recommend that you use a prepaid plan or a plan with unlimited SMS for testing purposes. Otherwise, if something goes wrong, you may need to pay a huge bill for hundreds of SMS text messages sent by mistake. In this tutorial we're using a prepaid plan with unlimited SMS.

The shield uses the original SIM card size, not micro or nano. If you have micro or nano you may consider getting a SIM card size adapter.

Turn off the PIN lock

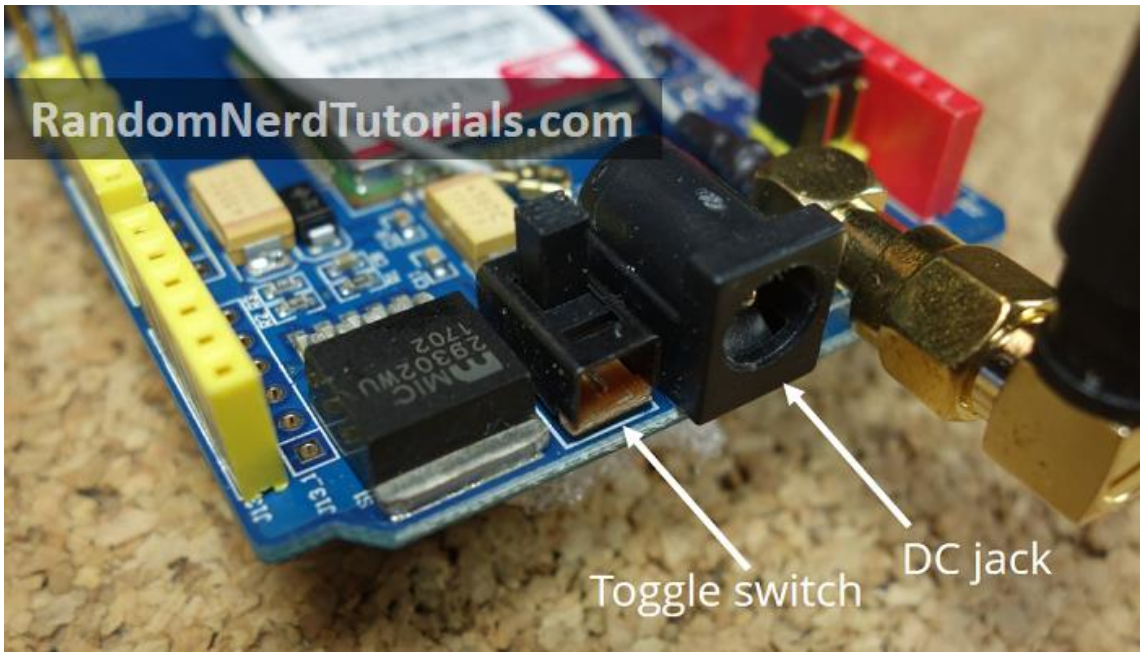
To use the SIM card with the shield, you need to turn off the pin lock. The easiest way to do this, is to insert the SIM card in your smartphone and turn off the pin lock in the phone security settings.

In my case, I need to go through: **Settings** > **Advanced Settings** > **Security** > **SIM lock** and turn off the lock sim card with pin.



Getting the right power supply

The shield has a DC jack for power as shown in figure below.



Next to the power jack there is a toggle switch to select the power source. Next to the toggle switch on the board, there is an arrow indicating the toggle position to use an external power supply – move the toggle switch to use the external power supply as shown above.

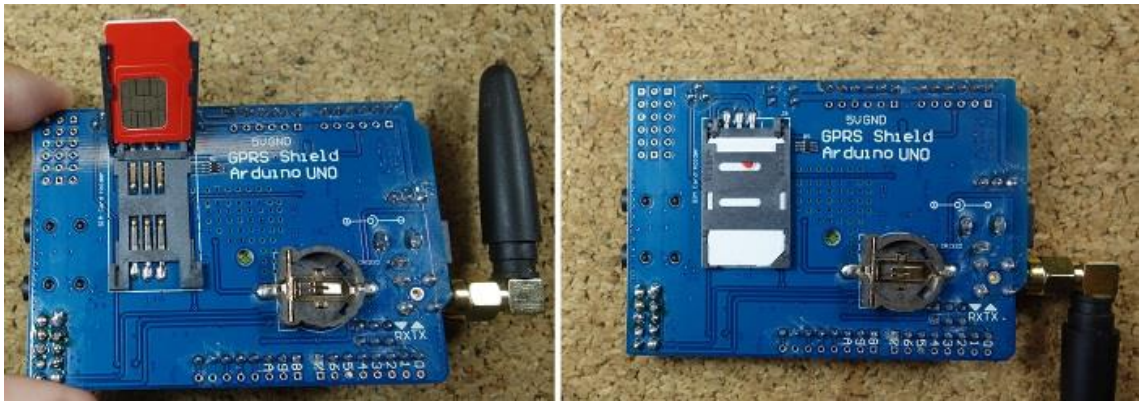
To power up the shield, it is advisable to use a 5V power supply that can provide 2A as the one shown below.

You can find the right power adapter for this shield here. Make sure you select the model with 5V and 2A.

Setting up the SIM900 GSM Shield

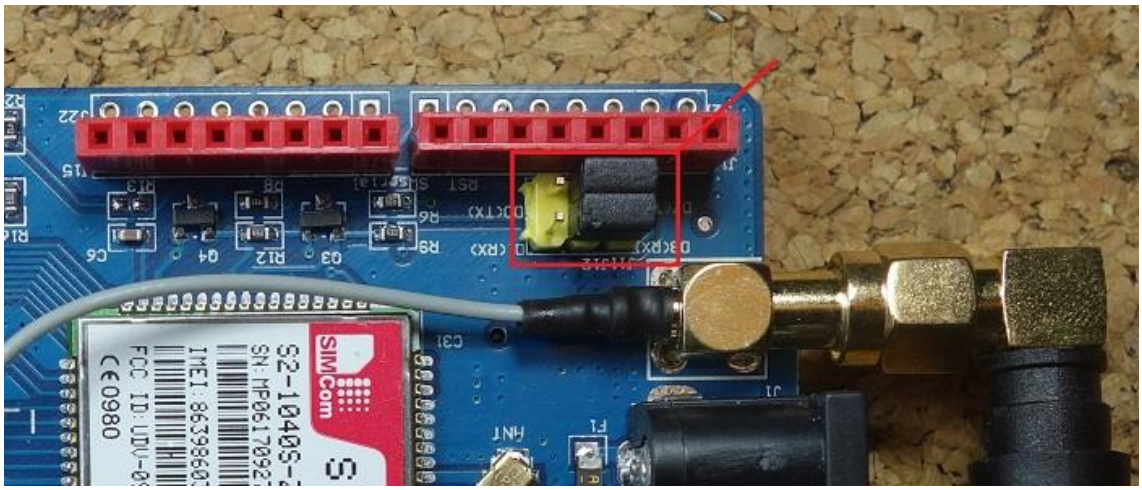
The following steps show you how to set up the SIM900 GSM shield.

1) Insert the SIM card into the SIM card holder. You need a SIM card with the standard size. The shield is not compatible with micro or nano SIM cards. If you need, you may get a sim card size adapter. Also, it is advisable to use a SIM card with a prepaid plan or unlimited SMS.

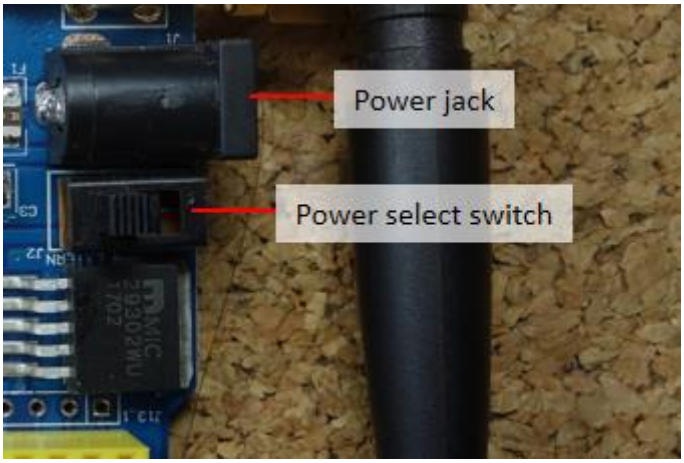


2) Confirm the antenna is well connected.

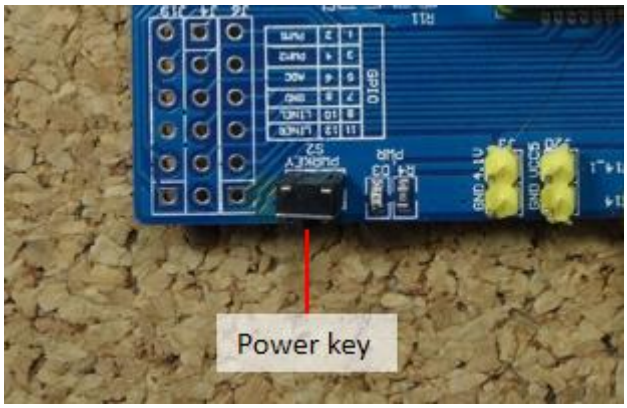
3) On the serial port select, make sure the jumper cap is connected as shown in figure below to use software serial.



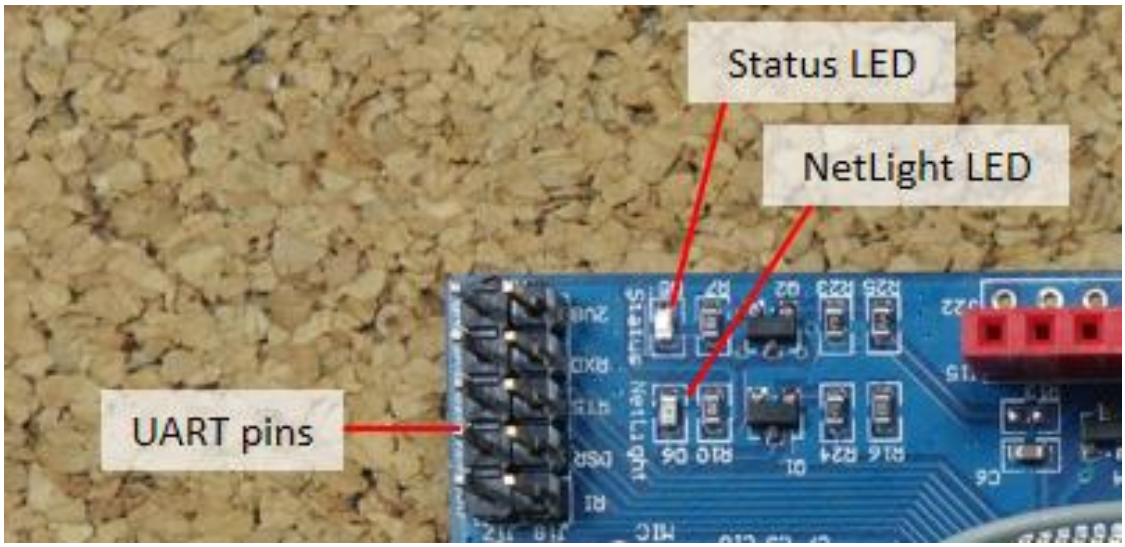
4) Power the shield using an external 5V power supply. Double-check that you have the external power source selected as we've mentioned earlier.



5) To power up/down the shield press the power key for about 2 seconds.



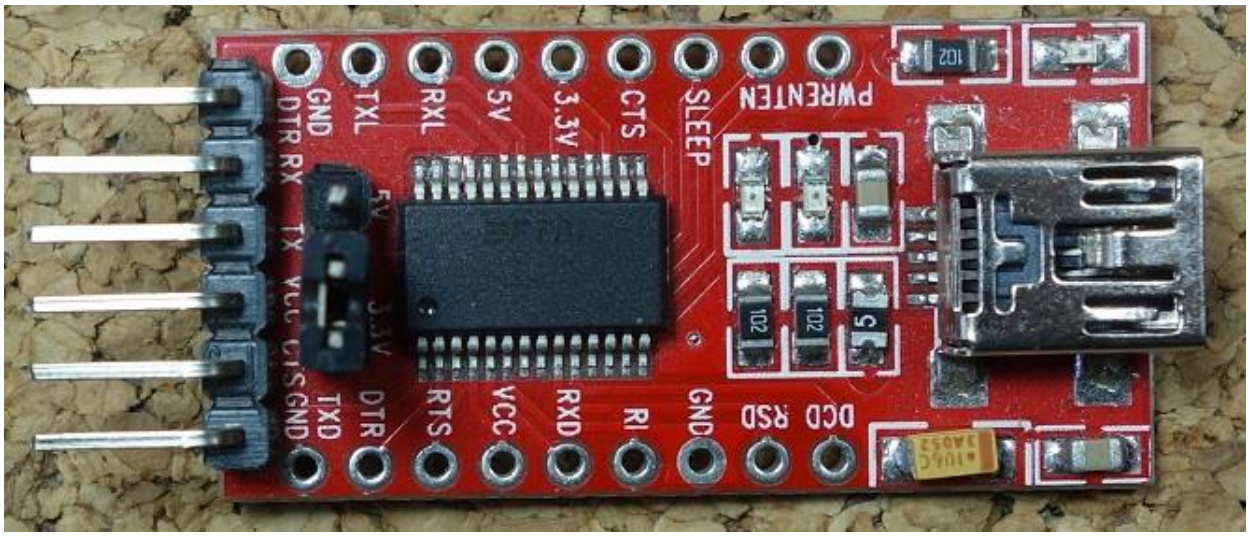
6) Then, the Status LED will light up and the NetLight LED will blink every 800 ms until it finds the network. When it finds the network the NetLight LED will start blinking every three seconds.



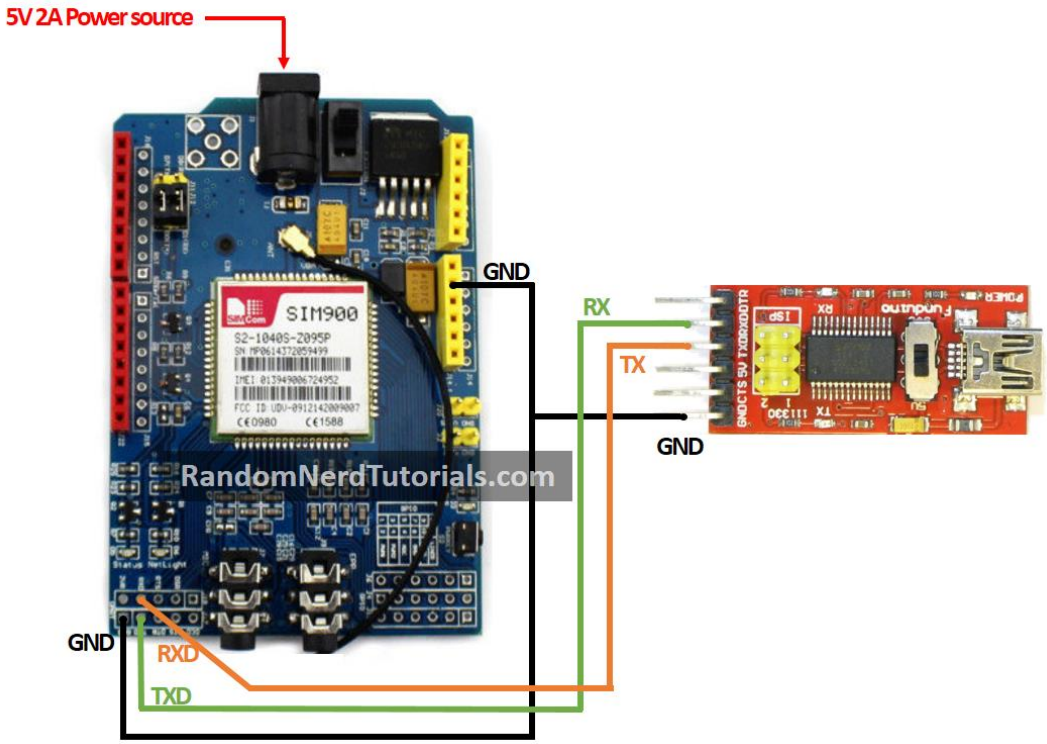
7) You can test if the shield is working properly by sending AT commands from the Arduino IDE using an FTDI programmer – as we'll show below.

Testing the Shield with FTDI programmer

You don't need to do this step to get the shield working properly. This is an extra step to ensure that you can communicate with your GSM shield and send AT commands from the Arduino IDE serial monitor. For that, you need an FTDI programmer as the one shown in figure below.



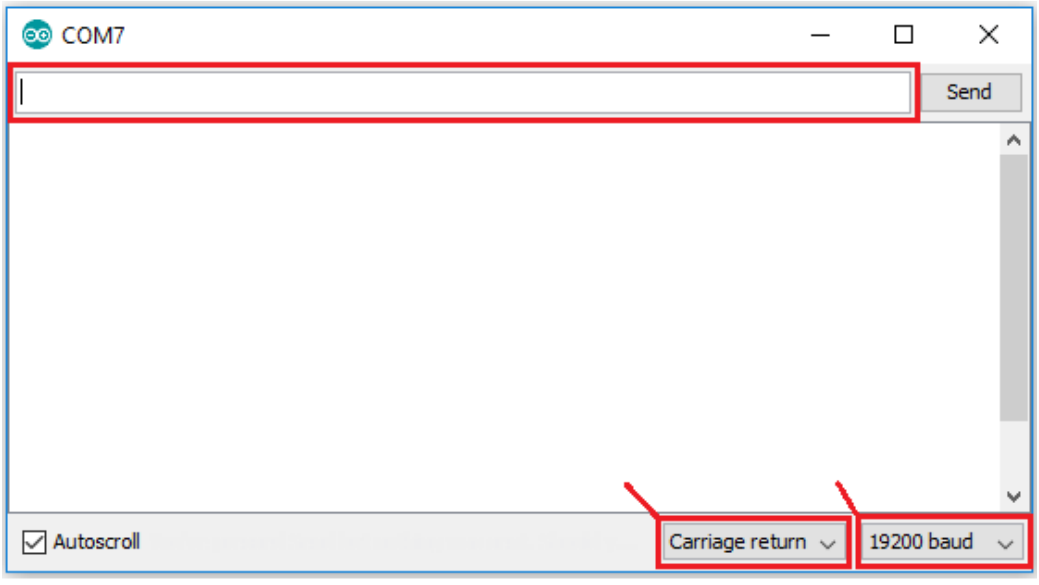
1) Connect the FTDI programmer to the GSM shield as shown in figure below.



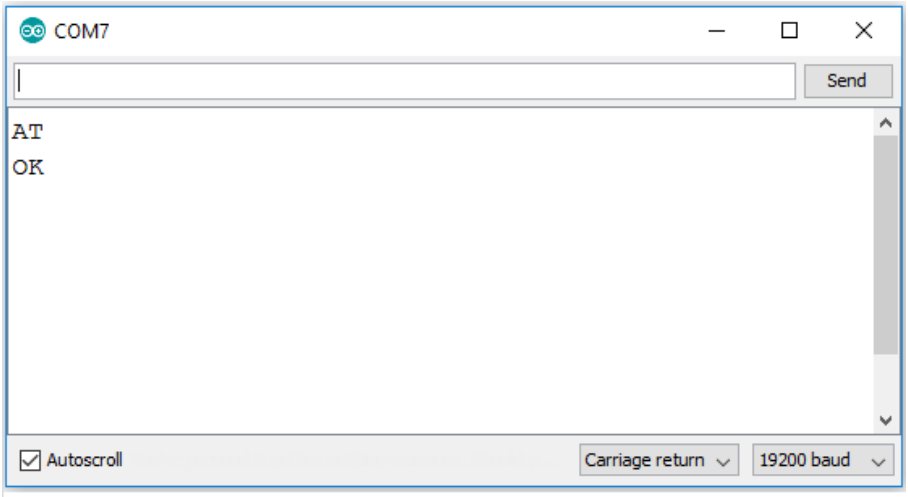
2) Open the Arduino IDE and select the right COM port.

3) Open the Serial monitor.

4) Select **19200** baud rate – the shield default setting is 19200 – and **Carriage return**. Write **AT** at the box highlighted in red and then press **ENTER**. See figure below.



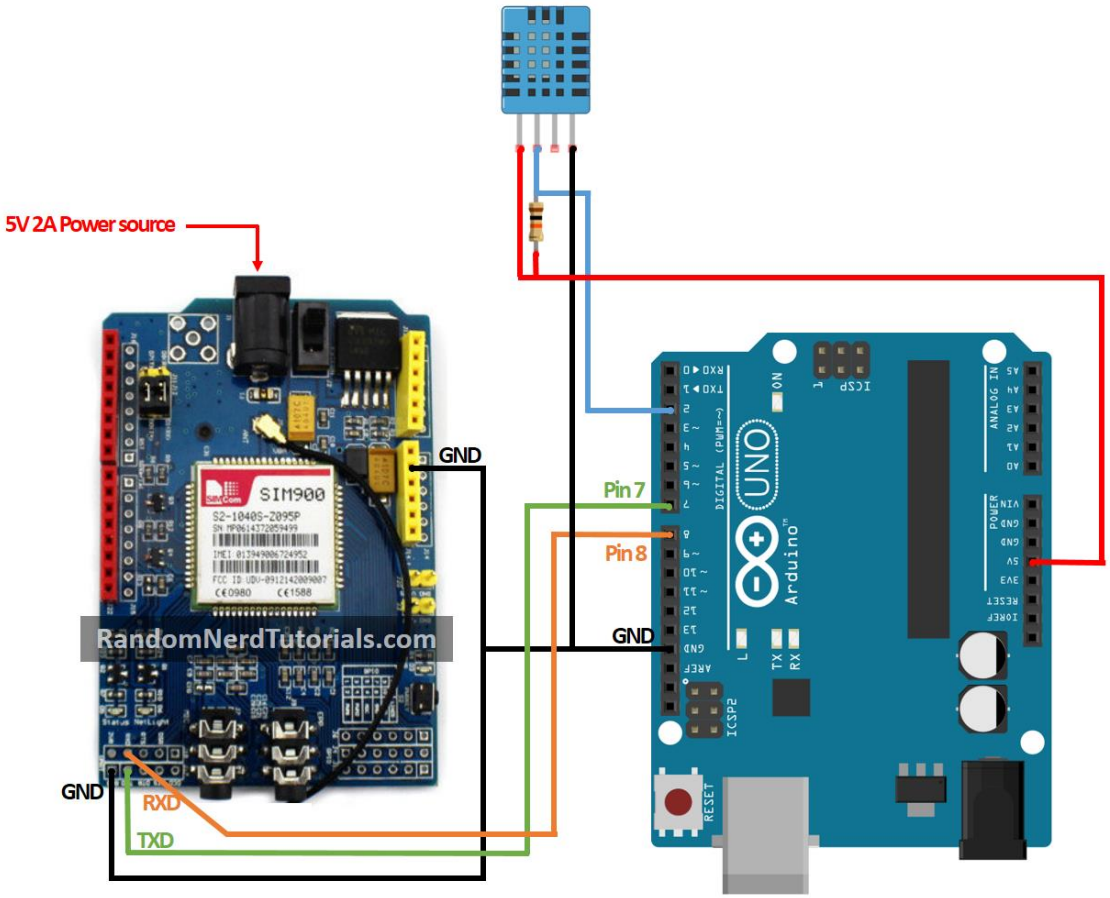
5) The shield will respond with OK, if everything is working properly.



Now that you know the shield is working properly, you are ready to start building your project.

Schematics

The figure below shows the circuit schematics for this project. You have to connect the SIM900 GSM shield and the DHT11 temperature and humidity sensor to the Arduino as shown in the figure below.



Installing the DHT library

To read from the DHT sensor, you must have the DHT library installed. If you don't have the DHT library installed, follow the instructions below:

1. Click here to download the DHT-sensor-library. You should have a *.zip* folder in your Downloads folder
2. Unzip the *.zip* folder and you should get **DHT-sensor-library-master** folder
3. Rename your folder from ~~**DHT-sensor-library-master**~~ to **DHT**
4. Move the **DHT** folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

Code

The following code reads the temperature and humidity from the DHT sensor and sends it via SMS when you send an SMS to the Arduino with the message "STATE".

You need to modify the code provided with the phone number your Arduino should reply the readings to. The code is well commented for you to understand the purpose of each line of code. Don't upload the code now. Scroll down and read the explanation below the code.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */


// Include DHT library
#include "DHT.h"
//Include Software Serial library to communicate with GSM
#include <SoftwareSerial.h>

// Pin DHT is connected to
#define DHTPIN 2

// Uncomment whatever type of sensor you're using
#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT22   // DHT 22  (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

// Initialize DHT sensor for normal 16mhz Arduino
DHT dht(DHTPIN, DHTTYPE);

// Create global varibales to store temperature and humidity
float t; // temperature in celcius
float f; // temperature in fahrenheit
float h; // humidity

// Configure software serial port
```

```
SoftwareSerial SIM900(7, 8);

// Create variable to store incoming SMS characters
char incomingChar;

void setup() {
  dht.begin();

  Serial.begin(19200);
  SIM900.begin(19200);

  // Give time to your GSM shield log on to network
  delay(20000);
  Serial.print("SIM900 ready...");

  // AT command to set SIM900 to SMS mode
  SIM900.print("AT+CMGF=1\r");
  delay(100);
  // Set module to send SMS data to serial out upon receipt
  SIM900.print("AT+CNMI=2,2,0,0,0\r");
  delay(100);
}

void loop(){
  if (SMSRequest()){
    if(readData()){
      delay(10);
      // REPLACE THE X's WITH THE RECIPIENT'S MOBILE NUMBER
      // USE INTERNATIONAL FORMAT CODE FOR MOBILE NUMBERS
      SIM900.println("AT + CMGS = \"+XXXXXXXXXX\"");
      delay(100);
      // REPLACE WITH YOUR OWN SMS MESSAGE CONTENT
      String dataMessage = ("Temperature: " + String(t) + "*C " + "
Humidity: " + String(h) + "%");
      // Uncomment to change message with farenheit temperature
      // String dataMessage = ("Temperature: " + String(f) + "*F "
+ " Humidity: " + String(h) + "%");

      // Send the SMS text message
      SIM900.print(dataMessage);
      delay(100);
      // End AT command with a ^Z, ASCII code 26
      SIM900.println((char)26);
      delay(100);
      SIM900.println();
      // Give module time to send SMS
      delay(5000);
    }
  }
  delay(10);
}

boolean readData() {
  //Read humidity
  h = dht.readHumidity();
  // Read temperature as Celsius
  t = dht.readTemperature();
  // Read temperature as Fahrenheit
```

```
  f = dht.readTemperature(true);

  // Compute temperature values in Celcius
  t = dht.computeHeatIndex(t,h,false);

  // Uncomment to compute temperature values in Fahrenheit
  //f = dht.computeHeatIndex(f,h,false);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return false;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  //Uncomment to print temperature in Farenheit
  //Serial.print(f);
  //Serial.print(" *F\t");
  return true;
}

boolean SMSRequest() {
  if(SIM900.available() >0) {
    incomingChar=SIM900.read();
    if(incomingChar=='S') {
      delay(10);
      Serial.print(incomingChar);
      incomingChar=SIM900.read();
      if(incomingChar =='T') {
        delay(10);
        Serial.print(incomingChar);
        incomingChar=SIM900.read();
        if(incomingChar=='A') {
          delay(10);
          Serial.print(incomingChar);
          incomingChar=SIM900.read();
          if(incomingChar=='T') {
            delay(10);
            Serial.print(incomingChar);
            incomingChar=SIM900.read();
            if(incomingChar=='E') {
              delay(10);
              Serial.print(incomingChar);
              Serial.print("...Request Received \n");
              return true;
            }
          }
        }
      }
    }
  }
  return false;
}
```

Source Code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/24_Data_request_via_SMS.ino

Importing libraries

First, you include the libraries needed for this project: the **DHT** libary to read from the DHT sensor and the SoftwareSerial library to communicate with the SIM900 GSM module.

```
#include "DHT.h"
#include <SoftwareSerial.h>
```

DHT sensor

Then, you tell the Arduino that the DHT data pin is connected to pin 2, select the DHT sensor type and create a dht instance. The code is compatible with other DHT sensors as long as you define the one you're using in the code.

```
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

You also create float variables to store the temperature and humidity values.

```
float t; // temperature in celcius
float f; // temperature in fahrenheit
float h; // humidity
```

GSM shield

The following line configures the software serial on pins 7 and 8. Pin 7 is being configure as RX and pin 8 as TX.

```
SoftwareSerial SIM900(7, 8);
```

You also create a char variable to store the incoming SMS characters.

```
char incomingChar;
```

setup()

In the **setup()**, you begin the DHT and the SIM900 shield. The SIM900 shield is set to text mode and you also set the module to send the SMS data to the serial monitor when it receives it. This is done with the following two lines, respectively:

```
SIM900.print("AT+CMGF=1\r");
SIM900.print("AT+CNMI=2,2,0,0,0\r");
```

Functions

We create a function to read the temperature and humidity called **readData()**. This function stores the values on the **t** and **h** variables. The code uses temperature in Celsius, but it is prepared if you want Fahrenheit instead – the code is commented on where you should make the changes.

We also create a function that checks if the incoming message is equal to STATE – the **SMSRequest()** function. This functions returns true if the Arduino receives a message with the text STATE and false if not. You read the SMS incoming characters using:

```
incomingChar=SIM900.read();
```

loop()

In the **loop()**, you check if there was an SMS request with the **SMSRequest()** function – you check if the Arduino received a STATE message. If **true**, it will read the temperature and humidity and send it via SMS to you.

The number the Arduino answers to is set at the following line:

```
SIM900.println("AT + CMGS = \"XXXXXXXXXXXX\"");
```

Replace the **XXXXXXXXXXXX** with the recipient's phone number.

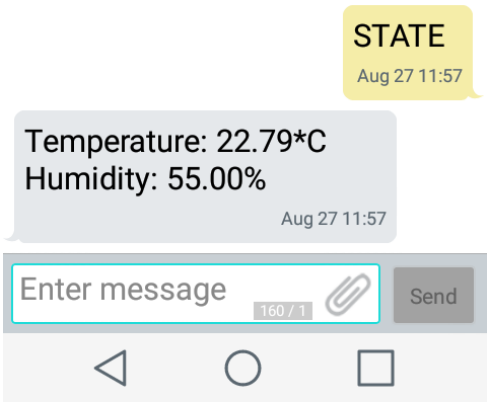
***Note**: you must add the number according to the international phone number format. For example, in Portugal the number is preceded by +351XXXXXXXXX.*

Then, you store the message you want to send in the **dataMessage** variable. Finally you send the SMS text message using:

```
SIM900.print(dataMessage);
```

Demonstration

When you send the STATE message to the Arduino, it replies with the sensor data.



Wrapping up

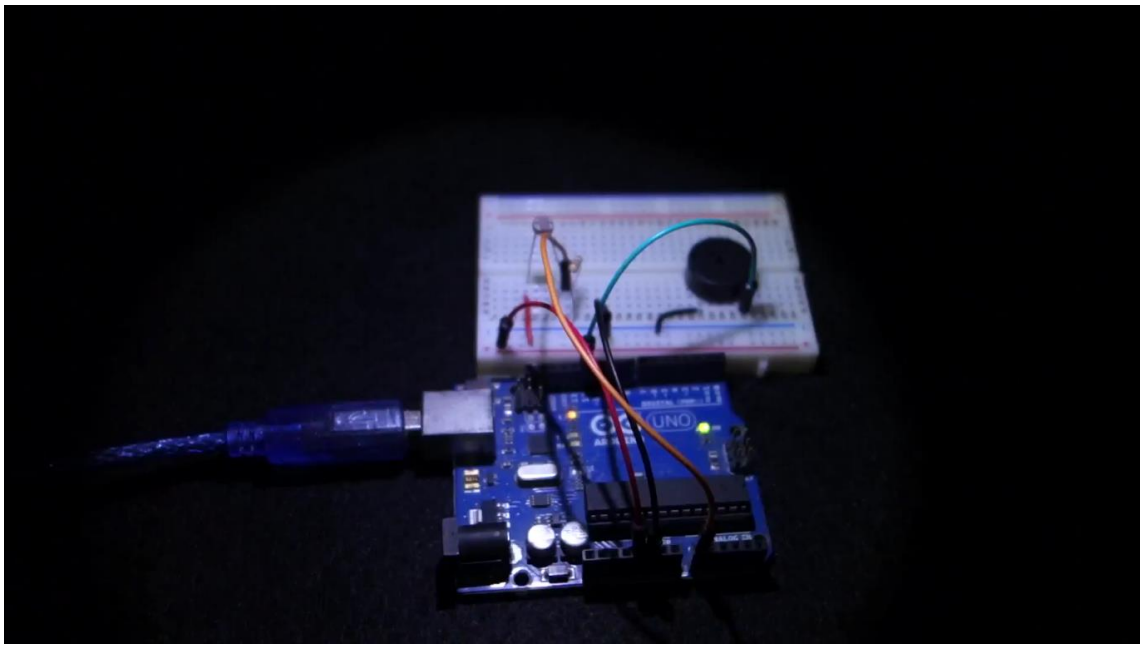
This is a great project to get you started with the SIM900 GSM shield. You've learned how to read and send SMS text messages with the Arduino.

You can apply the concepts learned in pretty much any project. Here's some ideas of projects:

- Surveillance system that sends an SMS when it detects movement
- Control a relay via SMS
- Request specific sensor data from a collection of sensors by adding more conditions to the code

The possibilities are endless. Have fun!

Pseudo-Theremin

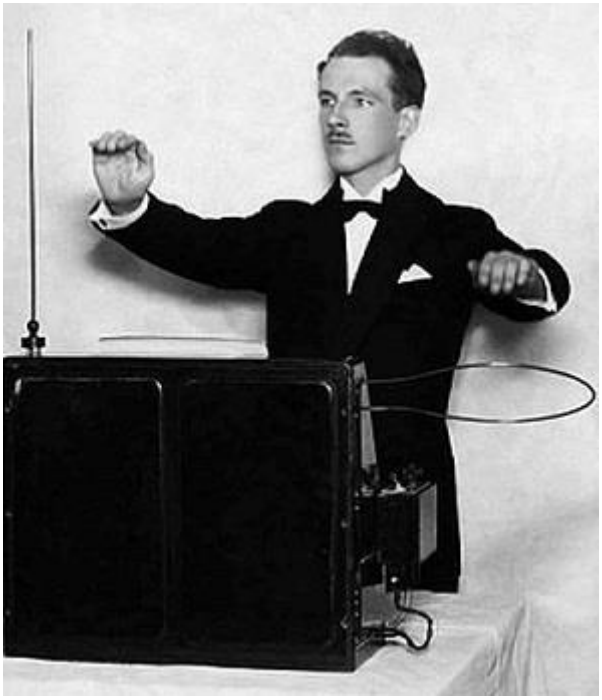


Pseudo-Theremin

Level: Beginner – Time: 15 min

In this project, you're going to build a pseudo-theremin. This is a great beginner project to get you started with the buzzer and photoresistor.

A theremin is a musical instrument controlled without physical contact between the instrument and the musician. The instrument makes different sounds as the player waves his hands in front of it. It was used in the theme music for the Start Trek Series.



Due to its weird sound, its music are often associated with eerie situations.

In this project, you'll use a piezo buzzer to produce the sound and a photoresistor that tells the Arduino how far your hands are (based on the luminosity).

Photoresistor

The photoresistor is the component in the figure below.

It is a light-dependent resistor. This means that the resistance across the photoresistor varies accordingly to the light intensity. Here's how the resistance varies:

- with increasing light, the less the resistance (more current flowing)
- with decreasing light, the more the resistance (less current flowing)

When connected to your Arduino, it gives analog values between 0 and 1023.

In other words:

- when you hand is near the photoresistor, the light intensity diminishes and the analog value read by the Arduino is lower
- when your hand is far away from the photoresistor, the light intensity increases and the analog value read by the Arduino is higher

Piezo buzzer

The piezo buzzer receives a digital output from the Arduino at a certain frequency and converts it into sound waves that we can hear.

For example, the middle C tone is defined at a frequency of 261 Hz. If you turn a digital output on and off 261 every second, the output will be middle C.

The function used to control the buzzer is the `tone()` function. It accepts three parameters `tone(pin, frequency, duration)`.

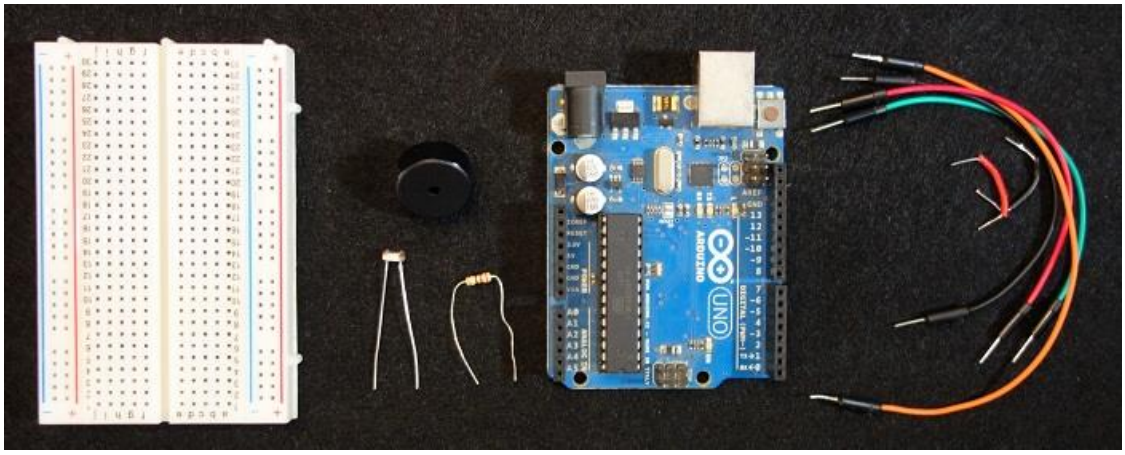
The `pin` represents the pin to which the buzzer is connected to.

The `frequency` corresponds to the frequency of the sound. This is what makes the buzzer produce different sounds.

The `duration` tells for how longer the buzzer makes the sound. If you don't enter any duration, it will keep playing the tone forever.

There's also another useful function the `noTone()` that makes the buzzer stop.

Parts required

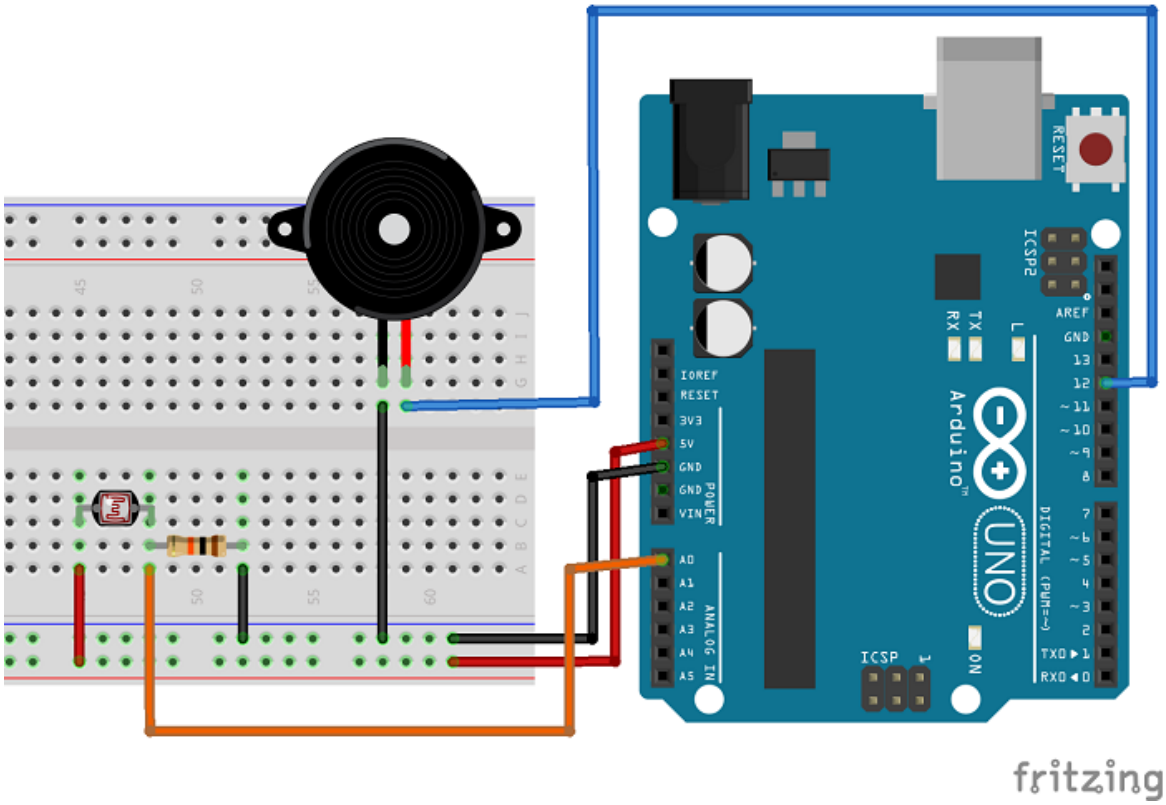


Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	1x Breadboard	http://ebay.to/21bEojM
	Arduino UNO	http://ebay.to/1SQda0R
	1 x Buzzer	http://ebay.to/1KsMYFP
	1x Photoresistors	http://ebay.to/2bMWpSV
	1x 10kOhm Resistor	http://ebay.to/1KsMYFP
	Jumper Wires	http://ebay.to/1PXeaJz

Schematics

Assemble all the parts by following the schematics below.



Note: check if your buzzer has a (+) (longer leg) and a (-) (shorter leg) sign next to its pins. If it has, connect the (+) side to the pin 12 and the (-) side to the GND pin.

Code

The code for this project is pretty straightforward. You don't need to install any libraries. Download or copy the following code to your Arduino IDE, and upload it to your Arduino Board. Make sure that you have the right board and COM port selected.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

int speakerPin = 12;
int photocellPin = A0;

void setup(){
  pinMode(speakerPin, OUTPUT);
}

void loop(){
  int reading = analogRead(photocellPin);
```

```
  int pitch = 200 + reading / 2;
  tone(speakerPin, pitch);
}
```

You can change the `pitch` variable to produce different tones.

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/8_pseudo_theremin.ino

Demonstration

Now, wave your hands next to the LDR to make different sounds.

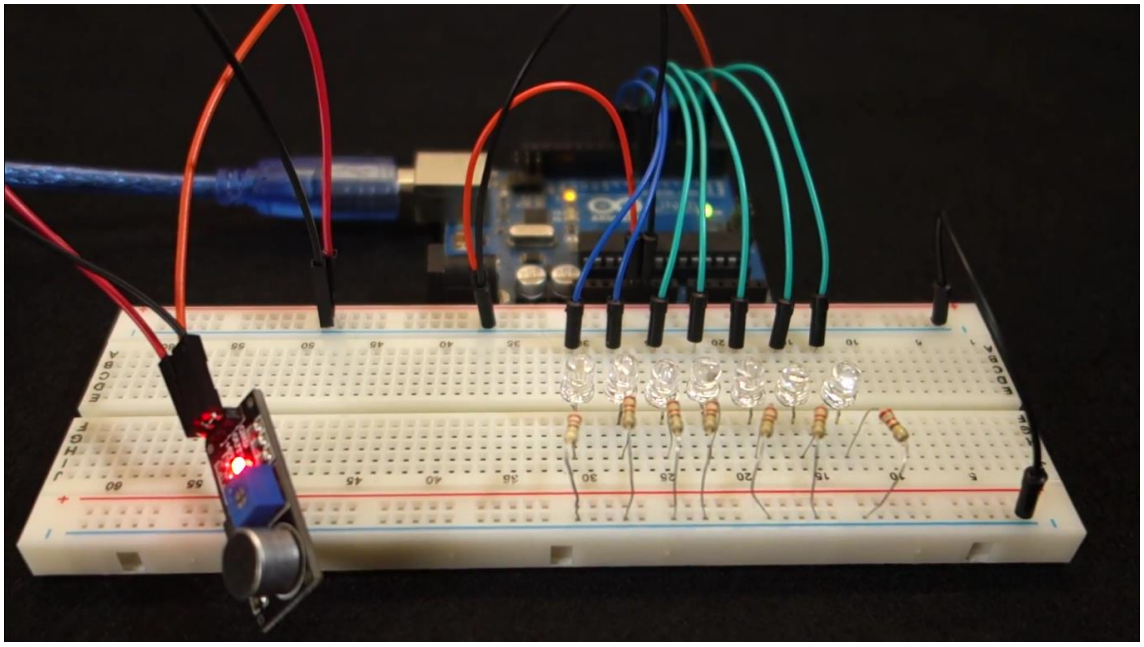
Tip: you can use a flashlight to make different sounds!

Wrapping up

This is a very simple project to introduce you to buzzers and photoresistors.

Now, you are able to build more advanced projects with sound!

Sound Sensitive Lights



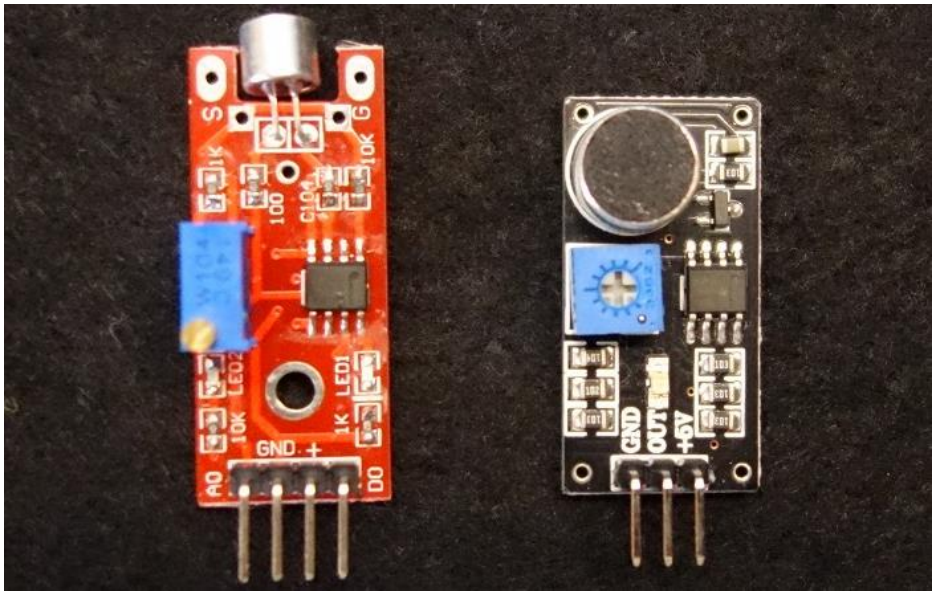
Sound Sensitive Lights

Level: Beginner – Time: 45 minutes

In this project, you're going to light up several LEDs accordingly to the beat of a music. For that, you need a sound sensor module. This project is very simple, but the final outcome is pretty cool!

Sound sensor

There are different versions of sound sensor modules for the Arduino. In the following figure you can see the most common sound sensor modules used with the Arduino.

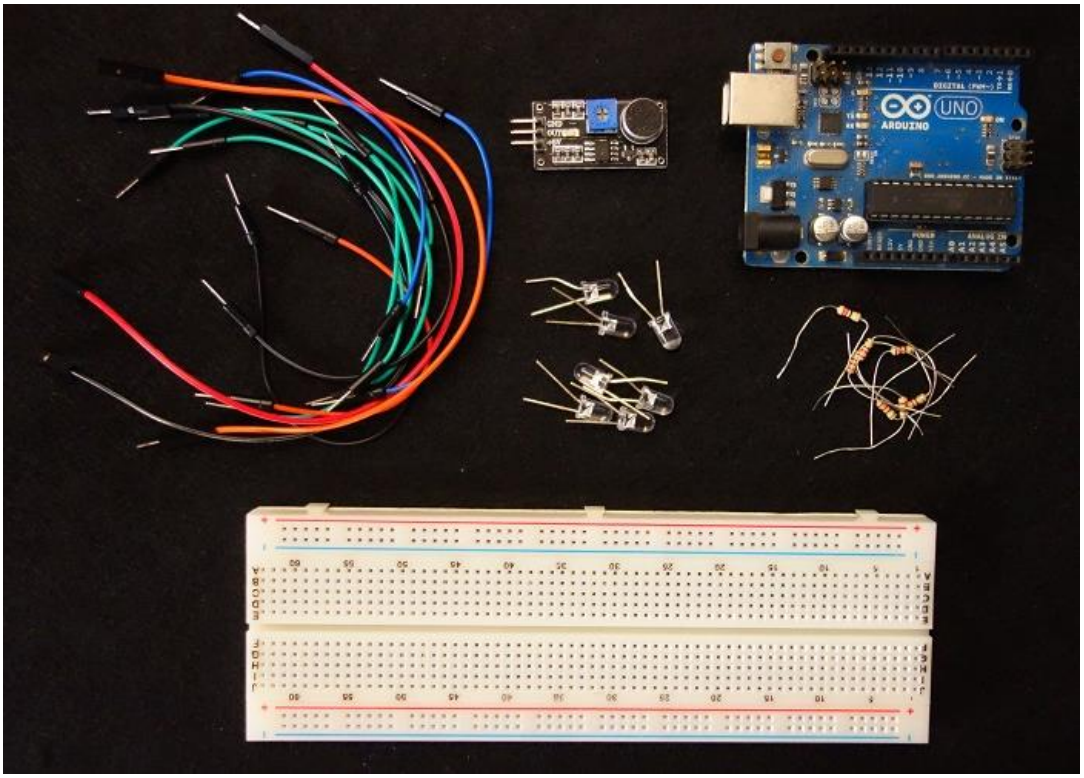


The sound sensor module has a microphone that captures the sound, a potentiometer that adjusts the sensor sensitivity and pins to connect it to your Arduino.

Some sound sensors come with both analog and digital output pins while others have just a digital output pin.

In this project, you're going to use just the digital output pin, so any version of the sensor should work just fine.

Parts required

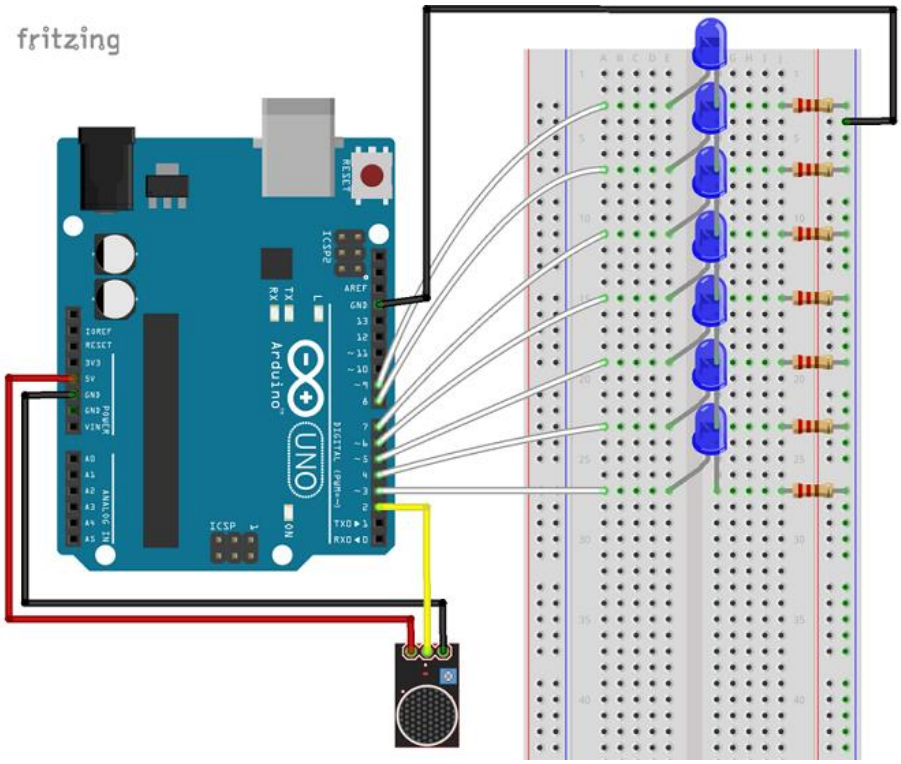


Grab all the components needed for this project:

Figure	Name	eBay
	Breadboard	http://ebay.to/21bEojM
	Arduino UNO	http://ebay.to/1SQda0R
	1x Sound Sensor Module	http://ebay.to/2dNejXG
	7x LED	http://ebay.to/20H2Oyy
	7x 220Ohm Resistor	http://ebay.to/1KsMYFP
	Jumper Wires	http://ebay.to/1PXeaJz

Schematics

Assemble all the parts by following the schematics below.

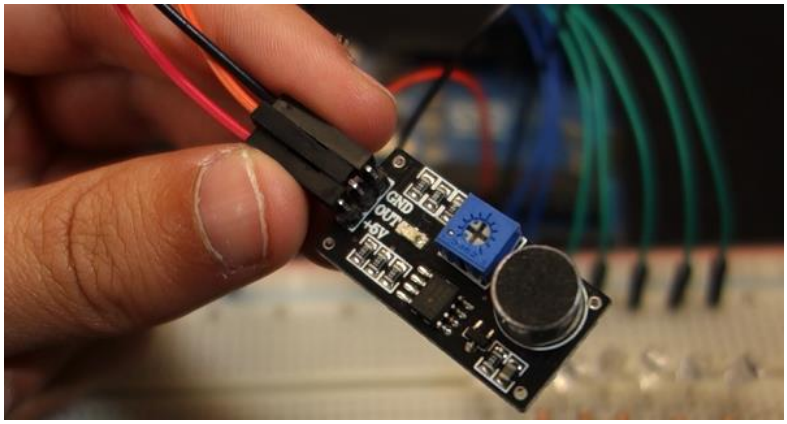


Connecting the sensor to your Arduino is pretty straightforward. You just need to follow these next instructions:

- connect the digital output pin to the Arduino digital pin 2
- VCC (or +) to 5V
- GND (or -) to GND

The rest of the circuit has 7 LEDs controlled by the Arduino digital pins.

Tip: it will be more handy if you connect the sound sensor module to your Arduino with long male to female jumper wires, so that later you can move your sensor freely.



Code

Here's some notes about this project that helps you understand the code:

- when the sound goes above a certain level, the Arduino reads a HIGH signal from the sensor
- when the sound is below a certain threshold, the Arduino reads a LOW signal from the sensor
- when the Arduino reads a LOW signal, all LEDs are off
- when the Arduino reads a HIGH signal, all LEDS are on

You don't need to install any libraries. So, you just have to download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure that you have the right board and COM port selected.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

int led1 = 3 ;// define LED
int led2 = 4;
int led3 = 5;
int led4 = 6;
int led5 = 7;
int led6 = 8;
int led7 = 9;

int sensorPin = 2;
int val = 0; // define numeric variables val

void setup (){
  pinMode(led1, OUTPUT) ;// define LED as output
  pinMode(led2, OUTPUT) ;// define LED as output
  pinMode(led3, OUTPUT) ;// define LED as output
  pinMode(led4, OUTPUT) ;// define LED as output
  pinMode(led5, OUTPUT) ;// define LED as output
  pinMode(led6, OUTPUT) ;// define LED as output
  pinMode(led7, OUTPUT) ;// define LED as output
  pinMode(sensorPin, INPUT) ;// output interface D0 is defined
sensor
  Serial.begin(9600);
}

void loop (){
  val = digitalRead(sensorPin);
  Serial.println(val);
  // When the sound detection module detects a signal, LED flashes
  if(val== HIGH){
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
```

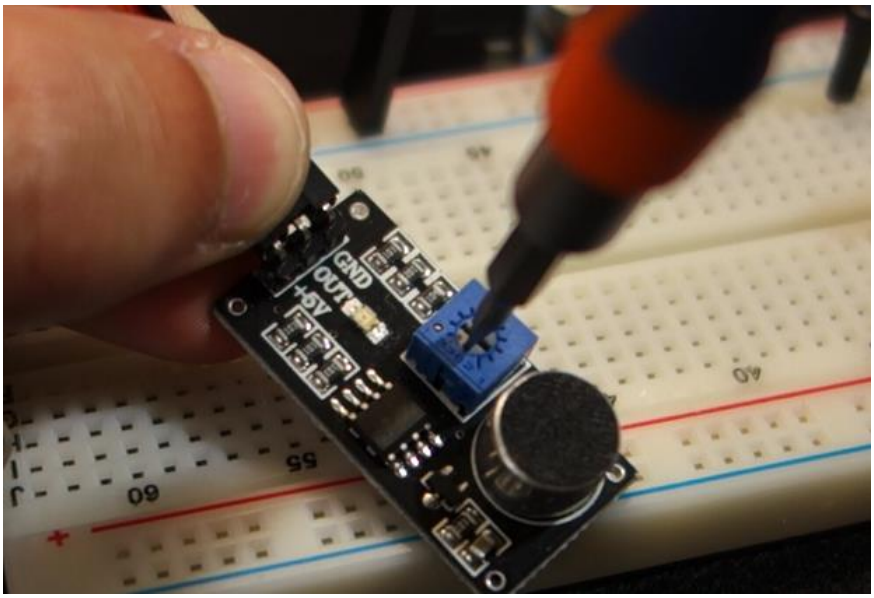
```
    digitalWrite(led5, HIGH);
    digitalWrite(led6, HIGH);
    digitalWrite(led7, HIGH);
  }
  else{
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    digitalWrite(led5, LOW);
    digitalWrite(led6, LOW);
    digitalWrite(led7, LOW);
  }
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/9_sound_sensitive_lights.ino

Adjusting the sensibility

After assembling the circuit and uploading the code, it's time to test the circuit. If you play a song and the LEDs don't follow the beat, you need to adjust the sensitivity of the sensor by rotating the built-in potentiometer.



If you want your LEDs to follow the beat of the music, here's what you should do:

- play your favorite song
- place the sensor next to the audio speakers
- if the lights are following the beat music, it's working properly
- if not, rotate the potentiometer to one side or to another, until you get the desired effect

Note: the potentiometer is very sensitive, so you need to be patient.

Demonstration

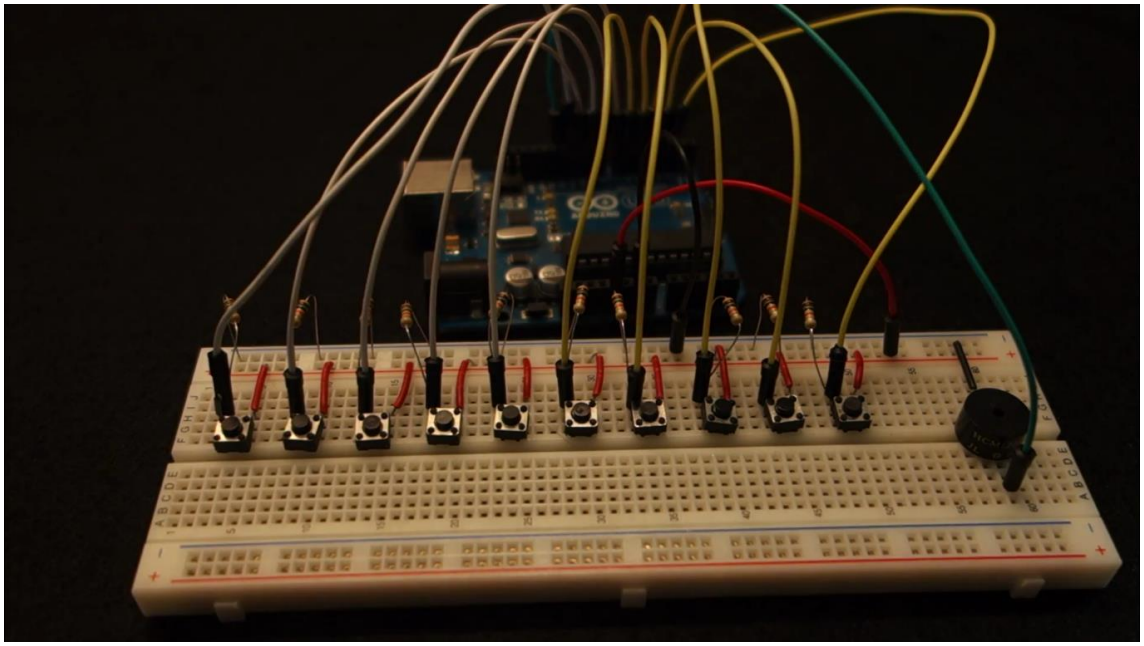
In the end you have a pretty cool effect! I encourage you to try it with your favorite songs.

Wrapping up

In this project you learned how to use the sound sensor module.

You can upgrade this project by adding more LEDs or LEDs with different colors.

Piano Keyboard



Piano Keyboard

Level: Beginner – Time: 45 minutes

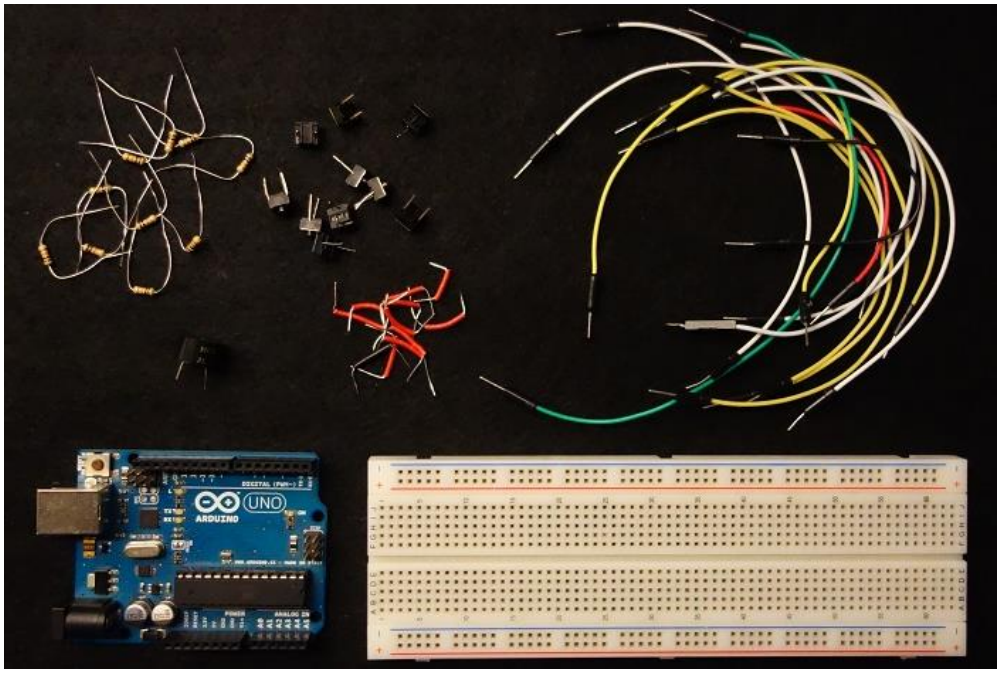
In this project, you're going to build a piano keyboard with your Arduino.

The piano keys are pushbuttons and the piezo buzzer makes the tones. Each pushbutton is associated with a tone with a certain frequency.

In this example we have 10 tones, but you can add more, as long as you know their frequency.

This is a very fun project, and in the end you can actually play music with the piano.

Parts required



Grab all the needed components for this project. If you don't have any of them, just click on the link to buy it on eBay.

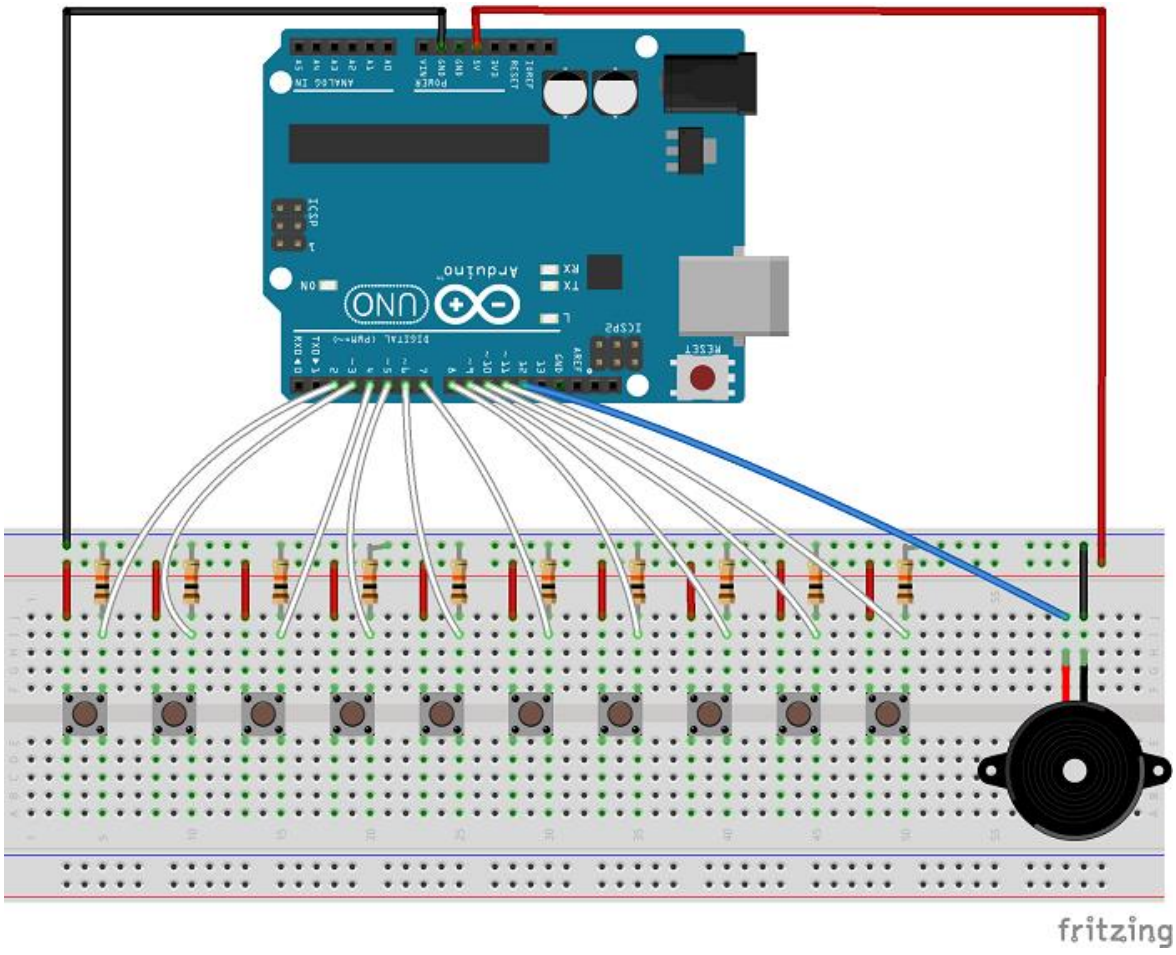
Figure	Name	eBay
	Breadboard	http://ebay.to/21bEojM
	Arduino UNO	http://ebay.to/1SQda0R

	10x pushbutton	http://ebay.to/211vcRv
	1x buzzer	http://ebay.to/2dcur1W
	10x 10kOhm Resistor	http://ebay.to/1KsMYFP
	Jumper Wires	http://ebay.to/1PXeaJz

Schematics

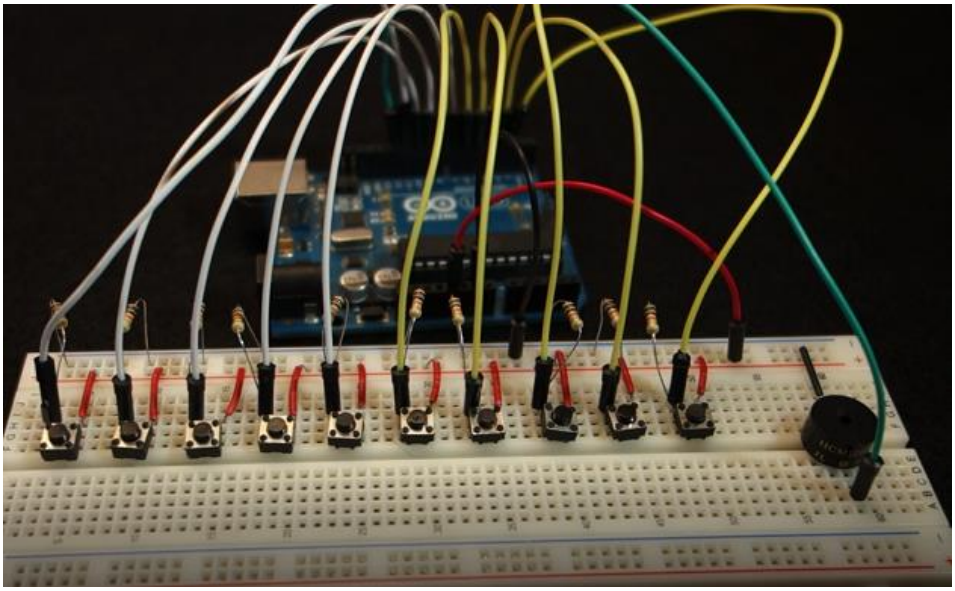
This circuit is very straightforward, you just have to connect 10 pushbuttons and a buzzer.

Assemble all the components following the schematics below.



In this project, I used pushbuttons with two legs. I used those, because they fit better in the breadboard than the four leg ones. However, you should use the type of pushbutton that is more handy for you.

In the end, your circuit looks like this:



You should follow the schematics provided so that the musical tones are in the correct order.

These are the tones of this piano: Middle C, C#, D#, D, E, F, F#, G, G# and A.

Code

This code is very simple. You don't need to install any library. Here's a quick explanation of how the code works:

- the code starts by creating an array that contains all the frequency tones that we want
- the code checks if any of the pushbuttons was pressed
- if any of them was pressed, it checks which one was pressed and attributes a tone with certain frequency to that button
- finally, the buzzer plays the corresponding tone
- if no button was pressed, the buzzer doesn't beep

To see this project in action download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure you have the right board and COM port selected.

```
/*
* Rui Santos
* Complete Project Details http://randomnerdtutorials.com
*/
```

```
int button_midC = 2;
int button_Csharp = 3;
int button_D = 4;
int button_Dsharp = 5;
int button_E = 6;
int button_F = 7;
int button_Fsharp = 8;
int button_G = 9;
int button_Gsharp = 10;
int button_A = 11;

int speaker = 12;

int buttonstate_midC = 0;
int buttonstate_Csharp = 0;
int buttonstate_D = 0;
int buttonstate_Dsharp = 0;
int buttonstate_E = 0;
int buttonstate_F = 0;
int buttonstate_Fsharp = 0;
int buttonstate_G = 0;
int buttonstate_Gsharp = 0;
int buttonstate_A = 0;

//NOTES          midC  C#    D     D#    E     F     F#    G     G#    A
int tones[] = { 261, 277, 294, 311, 330, 349, 370, 392, 415, 440 };
//freq
int playedNote = 0;

void setup(){
  pinMode(button_midC, INPUT);
  pinMode(button_Csharp, INPUT);
  pinMode(button_D, INPUT);
  pinMode(button_Dsharp, INPUT);
  pinMode(button_E, INPUT);
  pinMode(button_F, INPUT);
  pinMode(button_Fsharp, INPUT);
  pinMode(button_G, INPUT);
  pinMode(button_Gsharp, INPUT);
  pinMode(button_A, INPUT);
  pinMode(speaker, OUTPUT);
}

void loop(){
  buttonstate_midC = digitalRead(button_midC);
  buttonstate_Csharp = digitalRead(button_Dsharp);
  buttonstate_D = digitalRead(button_Dsharp);
  buttonstate_Dsharp = digitalRead(button_D);
  buttonstate_E = digitalRead(button_E);
  buttonstate_F = digitalRead(button_F);
  buttonstate_Fsharp = digitalRead(button_Fsharp);
  buttonstate_G = digitalRead(button_G);
  buttonstate_Gsharp = digitalRead(button_Gsharp);
  buttonstate_A = digitalRead(button_A);

  if(( buttonstate_midC == HIGH) || (buttonstate_Csharp == HIGH) ||
    (buttonstate_Dsharp == HIGH) || (buttonstate_D == HIGH) ||
```

```
    (buttonstate_E == HIGH) || (buttonstate_F == HIGH) ||
    (buttonstate_Fsharp == HIGH) || (buttonstate_G == HIGH)||
    (buttonstate_Gsharp ==HIGH) || (buttonstate_A ==HIGH)) {
    if (buttonstate_midC == HIGH){
      playedNote = tones[0];
    }
    if (buttonstate_Csharp == HIGH){
      playedNote = tones[1];
    }
    if (buttonstate_Dsharp == HIGH){
      playedNote = tones[2];
    }
    if (buttonstate_D == HIGH){
      playedNote = tones[3];
    }
    if (buttonstate_E == HIGH){
      playedNote = tones[4];
    }
    if (buttonstate_F == HIGH){
      playedNote = tones[5];
    }
    if (buttonstate_Fsharp == HIGH){
      playedNote = tones[6];
    }
    if (buttonstate_G == HIGH){
      playedNote = tones[7];
    }
    if (buttonstate_Gsharp == HIGH){
      playedNote = tones[8];
    }
    if (buttonstate_A == HIGH){
      playedNote = tones[9];
    }
    digitalWrite(speaker, HIGH);
    delayMicroseconds(playedNote);
    digitalWrite(speaker, LOW);
    delayMicroseconds(playedNote);
  }
  //in case no button is pressed, the buzzer doesn't beep
  else {
    digitalWrite(speaker, LOW);
  }
}
```

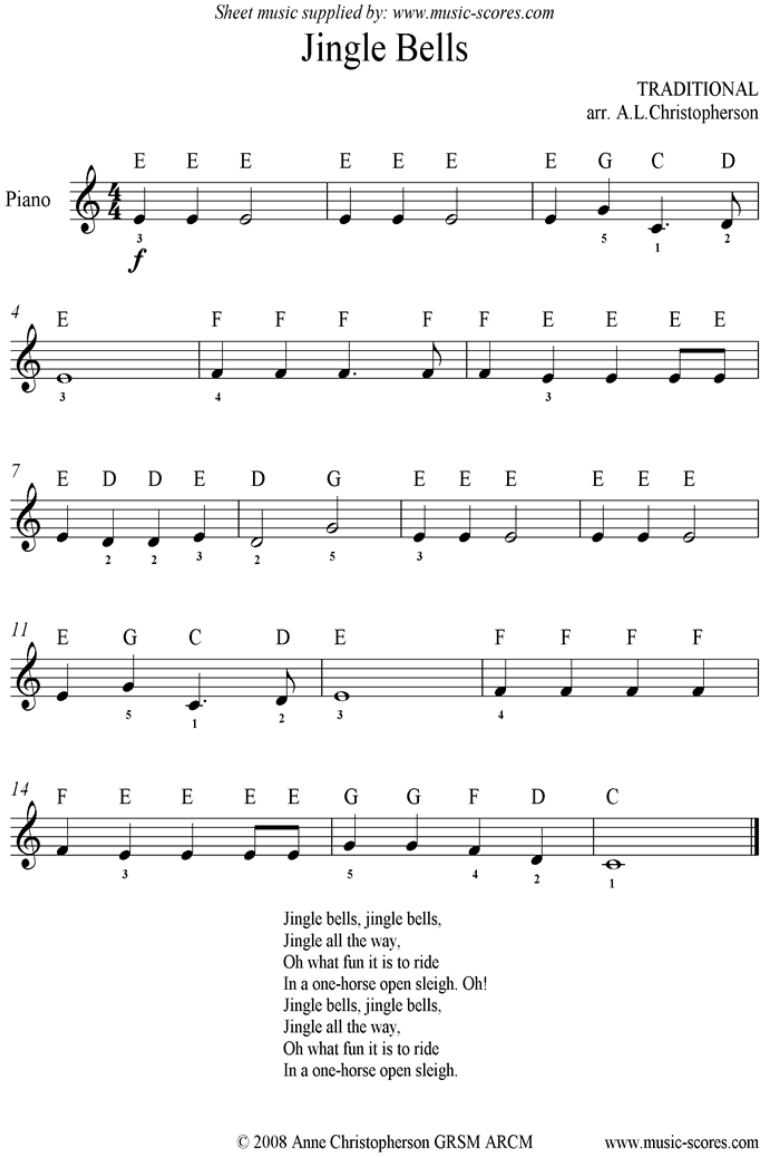
Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/10_piano_keyboard.ino

Demonstration

In the end, you can actually play songs.

Here's the sheet music for the Jingle Bells song – it's easy to play.



Now, it's time to play your favorite songs!

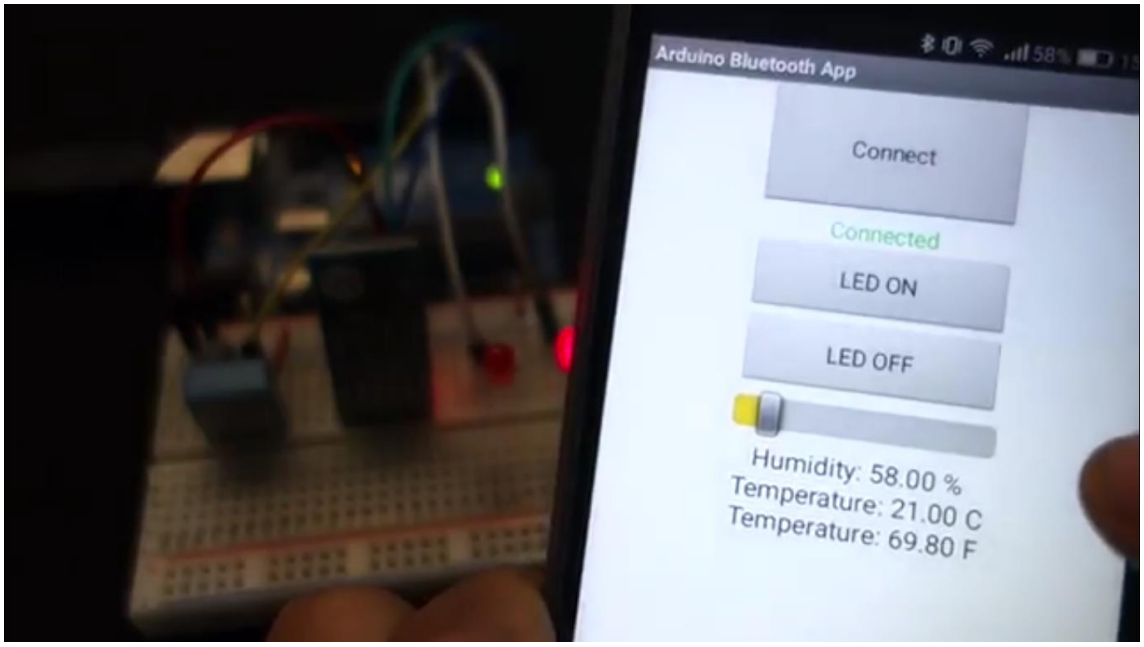
Wrapping up

I hope you've had fun building this project.

Here's some ideas on how you can improve this project:

- add more pushbuttons, so that you have more tones;
- add an led to each pushbutton, so that every time you play a tone, the corresponding led lights up.

Bluetooth Android App



Arduino Bluetooth Android App

Level: Intermediate – Time: 1h

In this project you're going to build an Android app that controls outputs and reads sensors.

The Android app is capable of:

- turn an LED on and off
- control the LED brightness
- display the temperature and humidity

This is a great project to get you started with the bluetooth technology and to learn how to develop simple Android apps that can easily integrate with your Arduino board.

Bluetooth module HC-05

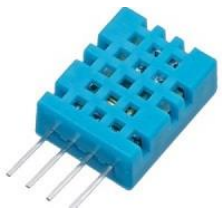
To establish the bluetooth communication between your smartphone and your Arduino, you need a bluetooth module. This project uses the HC-05 bluetooth module (as shown in the figure below).

This bluetooth module works with serial data. This means that the Arduino sends information and the bluetooth module receives it via serial (and vice-versa).

By default the HC-05 bluetooth module operates at a baud rate of 9600.

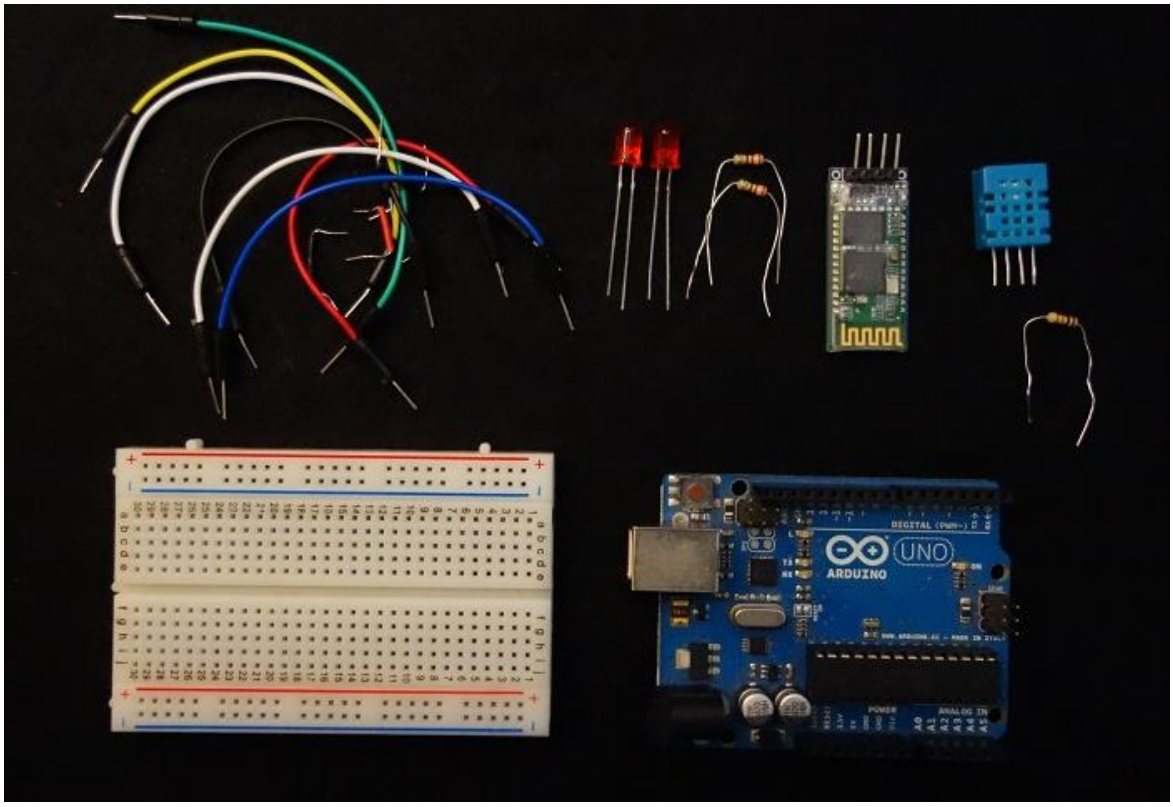
Measuring temperature and humidity

In this project, the temperature and humidity are measured with the DHT11 temperature and humidity sensor.



If you've followed the previous projects, you should already be familiar with this sensor. This project should work just fine with other sensors, as long as you do a couple of changes to the Arduino sketch.

Parts required



Grab all the components needed for this project. If you don't have them, just visit the links in the table below.

Figure	Name	eBay
	Arduino UNO	http://ebay.to/1SQda0R
	HC-04 or HC-05 or HC-06 Bluetooth module	http://ebay.to/2dt1b6M
	2x LEDs	http://ebay.to/20H2Oyy
	Breadboard	http://ebay.to/21bEojM

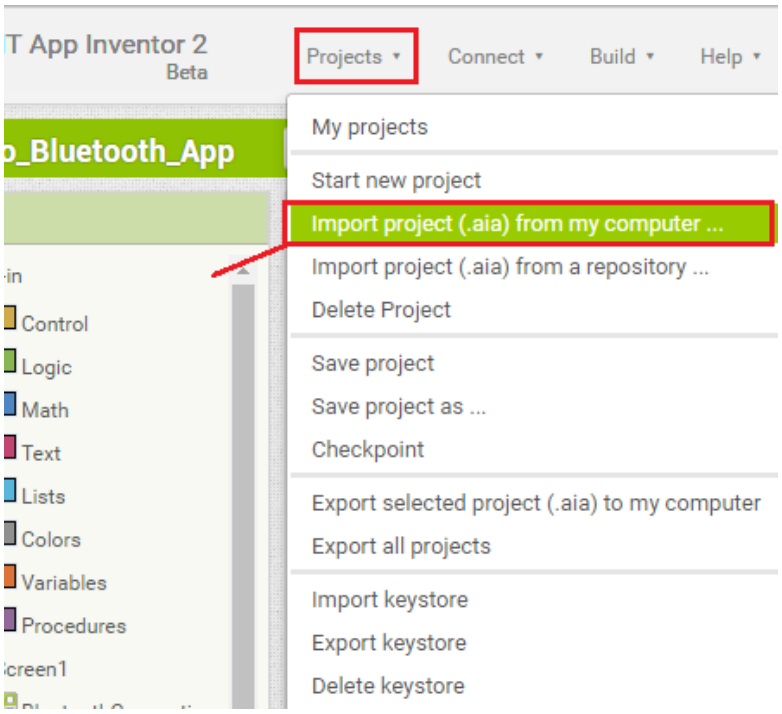
	DHT11 temperature and humidity sensor	http://ebay.to/2ddDgLv
	1x 10kOhm Resistor 2 x 220Ohm Resistor	http://ebay.to/1KsMYFP
	Jumper Wires	http://ebay.to/1PXeaJz

Creating your Android App

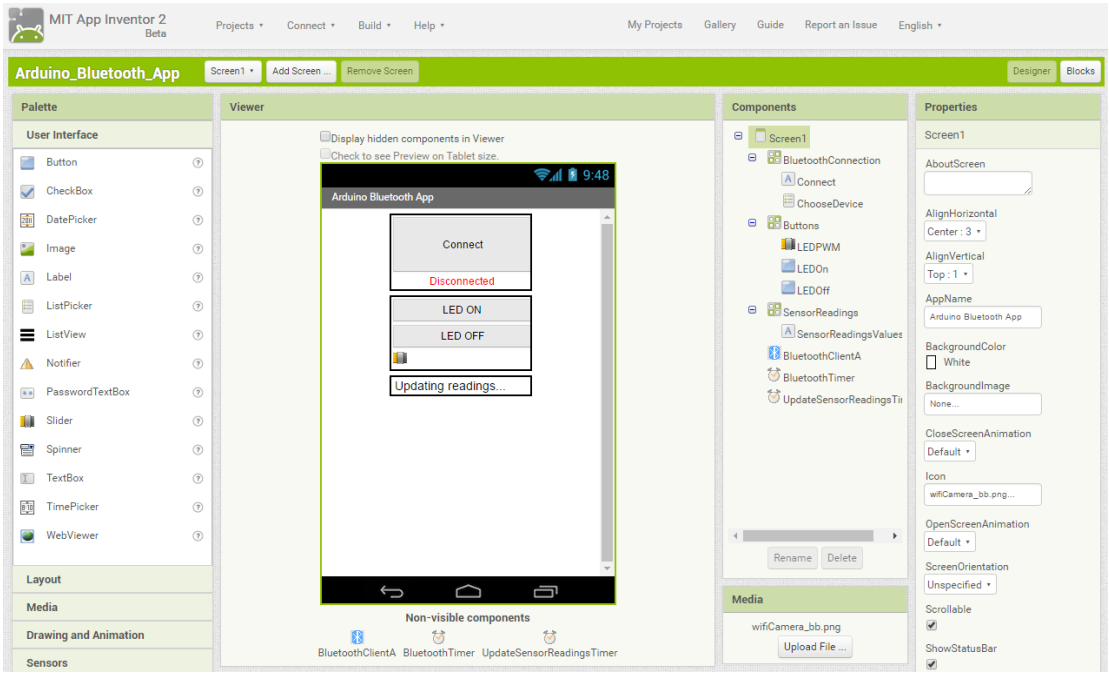
The Android App will be created using a free web application called MIT App Inventor. MIT App Inventor is a great place to get started with Android development, because it allows you to build simple apps with drag-n-drop.

You need a Google account to sign up for MIT App Inventor and here's the login page: http://ai2.appinventor.mit.edu.

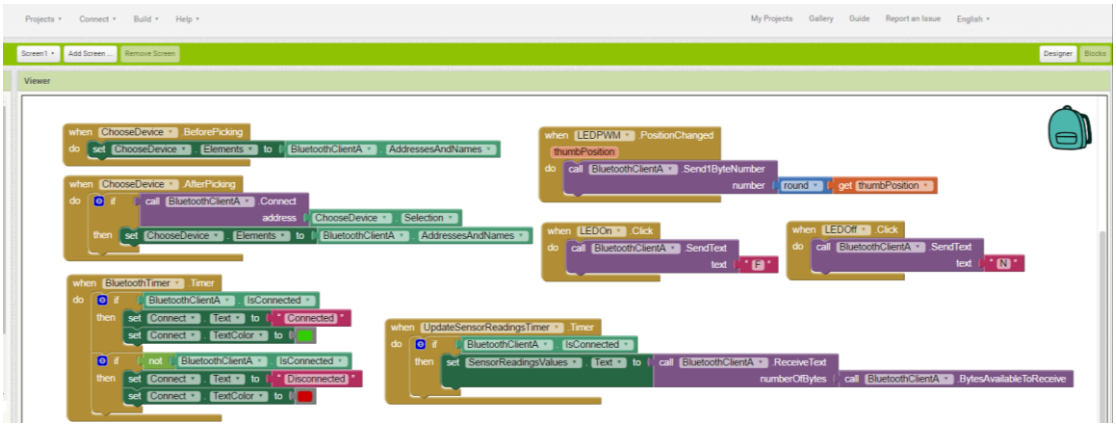
Click here to download the .aia file.



With MIT App Inventor you have 2 main sections: designer and blocks. The designer is what gives you the ability to add buttons, add text, add screens and edit the overall app look.



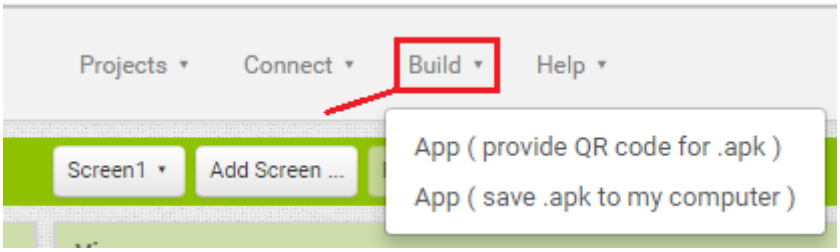
The blocks sections is what allows to create custom functionality for your app, so when you press the buttons it actually does something with that event.



I recommend that you start by following this project and using the app without modifying it.

If want to make any changes to the app, when you're done and you want to install the app in your smartphone, go to the Build tab.

You can either generate a QR code that you can scan with your smartphone and automatically install the app in your smartphone or you can download the .apk file, connect your smartphone to your computer and move the .apk file to the phone.



Simply follow the installation wizard to install the App and it's done!

Code

In this example, you need to have the DHT library installed. If you don't have it, just follow the instructions:

1. Click here to download the DHT-sensor-library. You should have a .zip folder in your Downloads folder
2. Unzip the .zip folder and you should get DHT-sensor-library-master folder
3. Rename your folder from ~~DHT-sensor-library-master~~ to DHT
4. Move the DHT folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

Important: before uploading the code, make sure you have the TX and RX pins disconnected from the bluetooth module!

Then, you can upload the code below:

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

#include <DHT.h>

#define DHTPIN 4     // what pin we're connected to
#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);

int ledOn = 7;
int ledPWM = 6;

int state;
int flag = 0;       //makes sure that the serial only prints once
the state
int stateStop = 0;

unsigned long previousMillis = 0;
const long interval = 5000;

void displayTempHumid() {
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very
```

```
slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.print(" ");
    Serial.println("Failed to read");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.println(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" C");
  Serial.print("Temperature: ");
  Serial.print(f);
  Serial.println(" F");
}

void setup() {
  // sets the pins as outputs:
  pinMode(ledOn, OUTPUT);
  pinMode(ledPWM, OUTPUT);

  digitalWrite(ledOn, LOW);
  analogWrite(ledPWM, LOW);

  dht.begin(); // initialize dht

  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

void loop() {
  unsigned long currentMillis = millis();

  //if some date is sent, reads it and saves in state
  if (Serial.available() > 0) {
    state = Serial.read();
    flag = 0;
  }
  // if the state is ON the LED is ON
  if (state == 'F' && flag == 0) {
    digitalWrite(ledOn, HIGH);
    flag = 1;
  }
  // if the state is OFF the LED is OFF
  else if (state == 'N' && flag == 0) {
    digitalWrite(ledOn, LOW);
    flag = 1;
  }
  else if (state != 'N' && state != 'F' && flag == 0) {
    analogWrite(ledPWM, state);
```

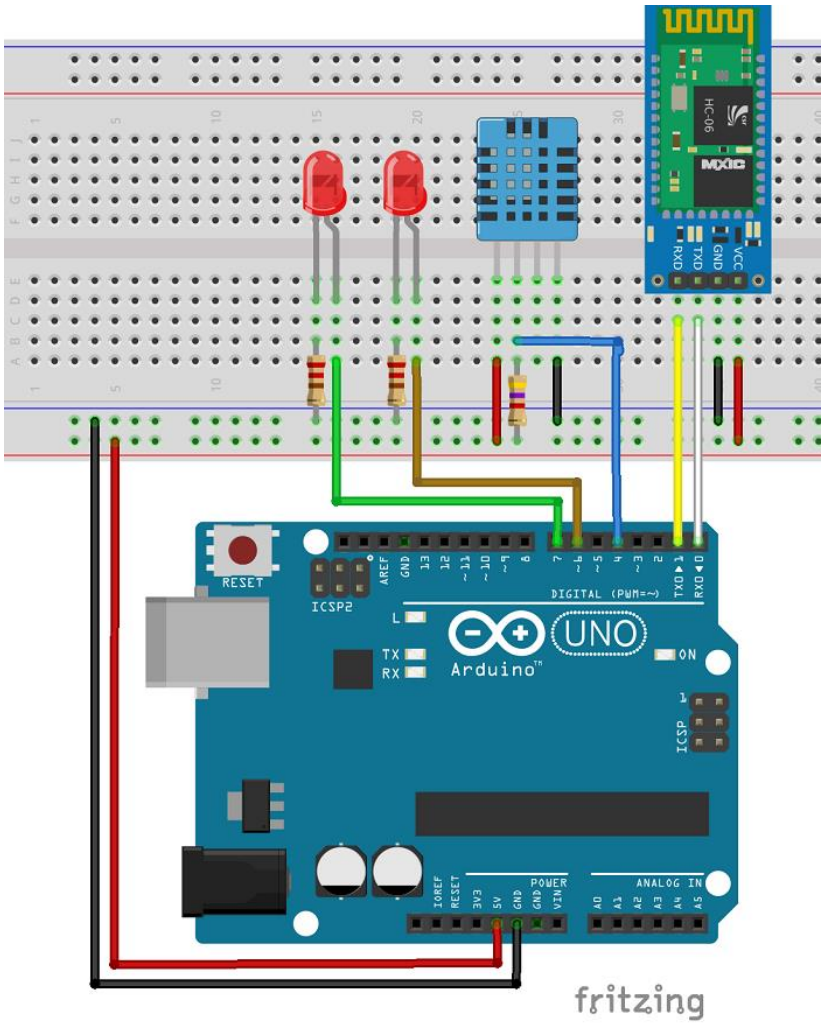
```
    flag = 1;
  }
  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
    displayTempHumid();
  }
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/7_arduino_bluetooth_android_app

Schematics

Assemble all the parts by following the schematics below.

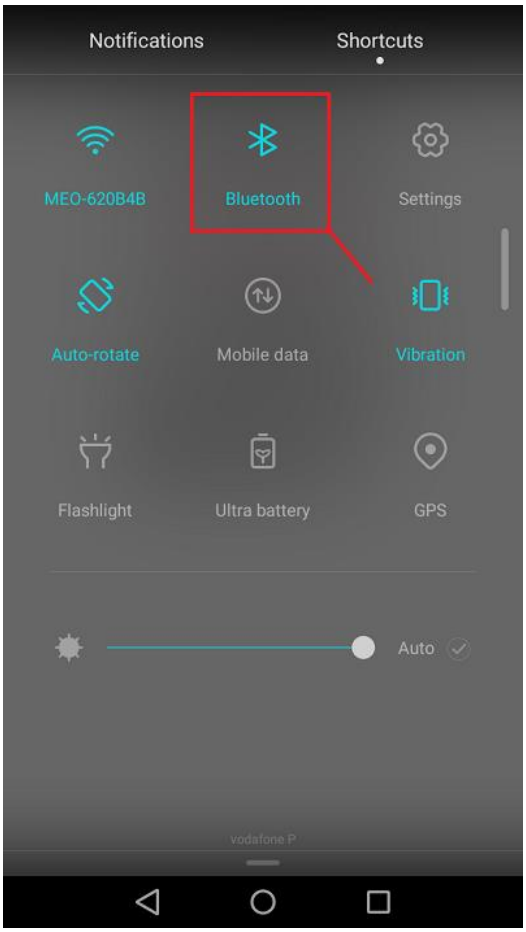


Important: you must disconnect the TX and RX pins before uploading the code to your Arduino.

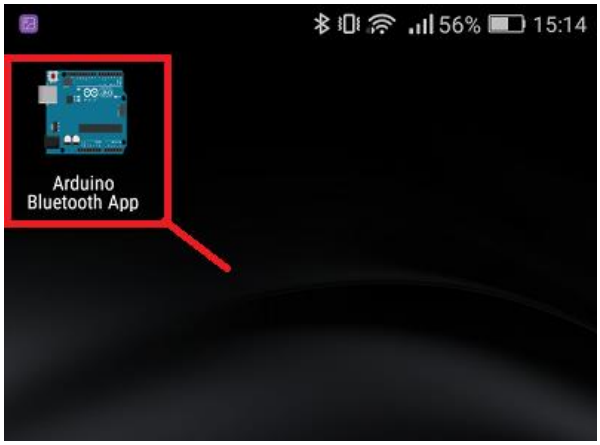
Launching your App

If you haven't generated the .apk file in a previous step, you can click here to download the .apk file (which is the Android App installation file). Move that file to your smartphone and open it. Follow the installation wizard to install the app.

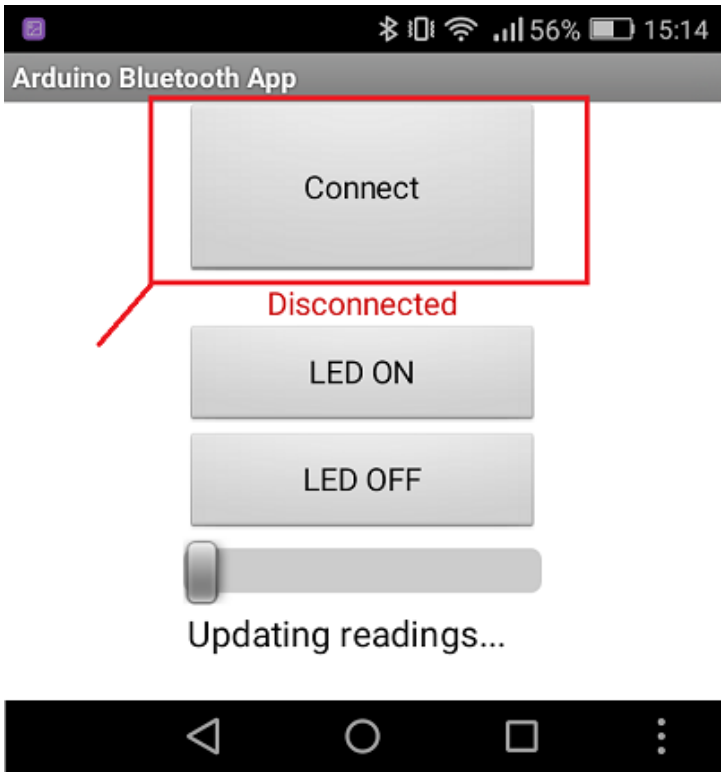
Turn on your smartphone's Bluetooth.



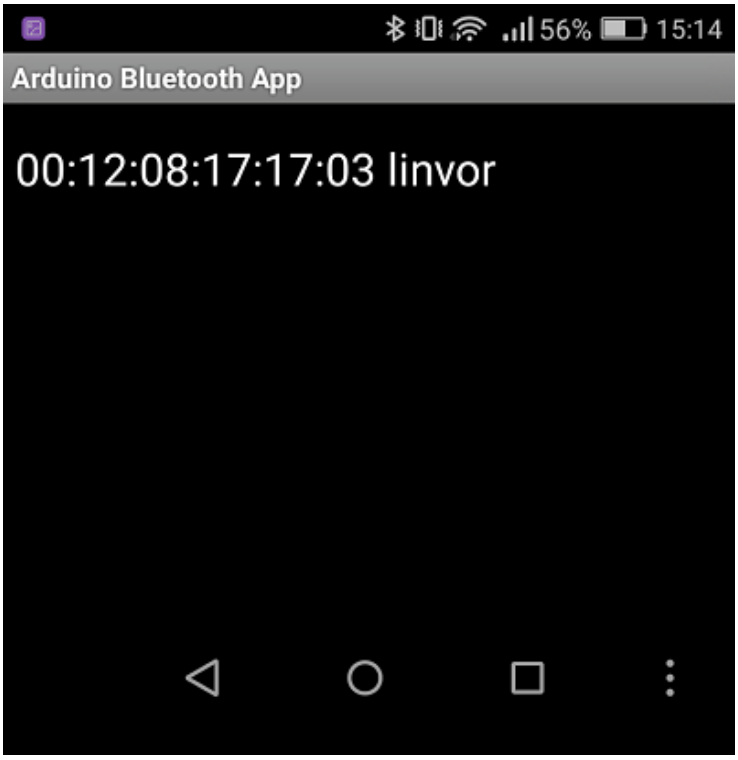
Tap on the Arduino Bluetooth App icon.



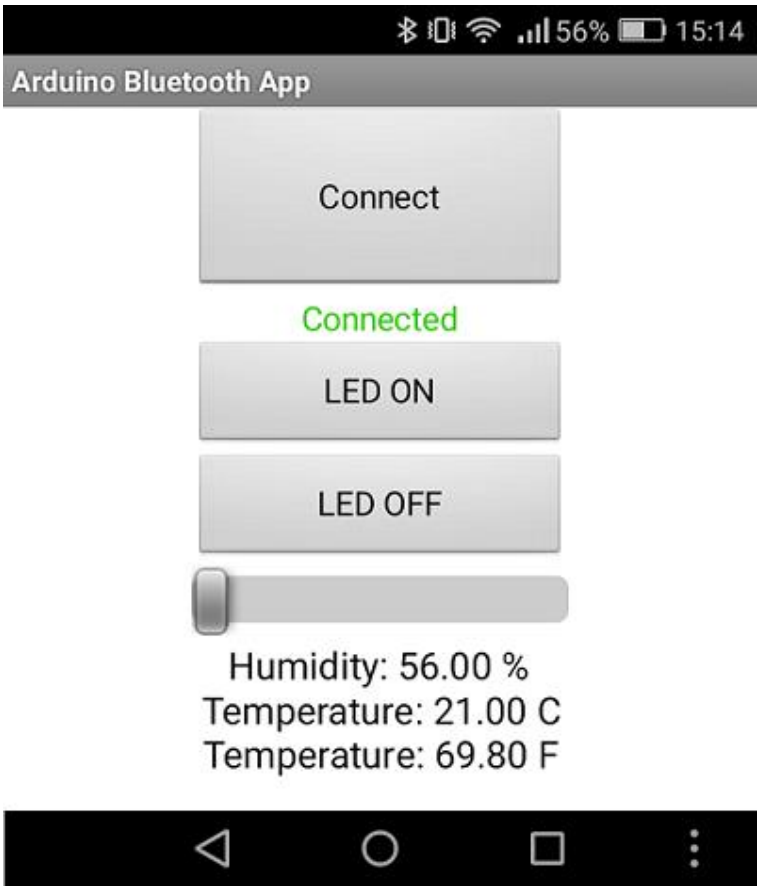
Press the "Connect" button to connect your application to your Arduino Bluetooth module.



Select your Bluetooth module (it should be named linvor).

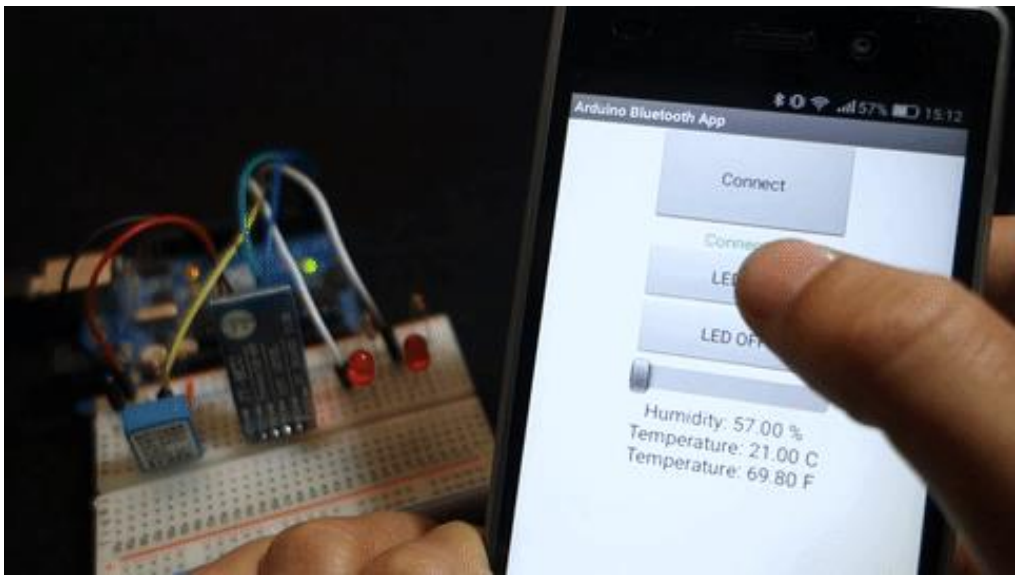


Now, it is ready to use!



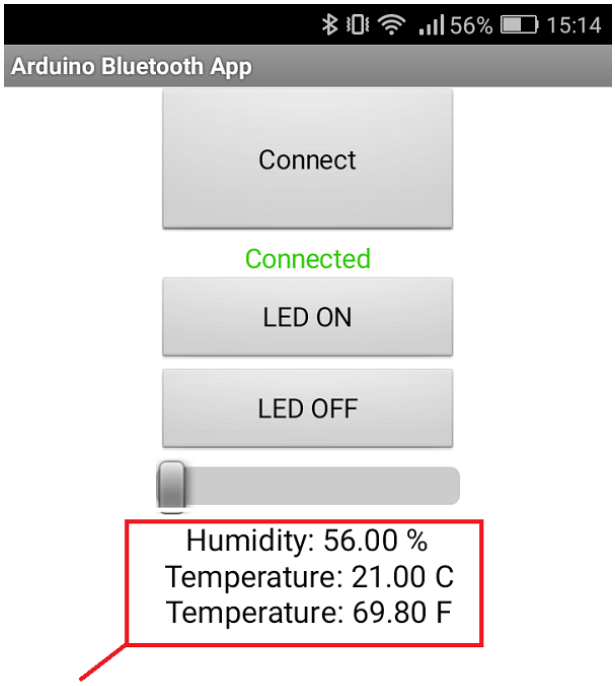
Demonstration

Here's what you have in the end. The app turns the LED on and off.



It controls the LED brightness.

And it displays the temperature and humidity



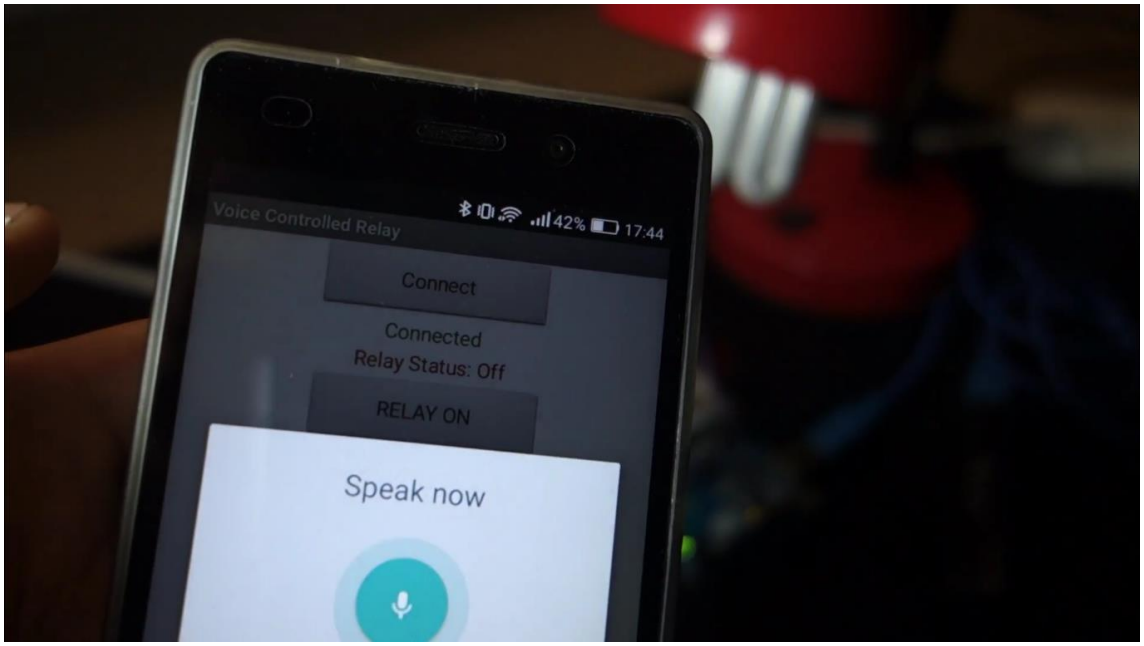
Wrapping up

In this project you've learned how to use the bluetooth module and how you can build a simple App using the MIT App Inventor.

This project gives you the knowledge to control other devices than the LEDs and read several sensors using your smartphone.

Now, you are able to grab this concept, modify it and build your own app!

Voice Controlled Relay App



Voice Controlled Relay App

Level: Advanced – Time: 1h 15 minutes

In this project you're going to build an Android app that controls a relay module with voice commands. The relay is connected to an AC desktop lamp, but you can connect any AC electronic device you want.

This is a simple project to get you familiar relay modules.

SAFETY WARNING!

When you are making projects that are connected to mains voltage, you really need to know what you are doing, otherwise **you may shock yourself.** This is a serious topic and I want you to be safe. If you are not 100% sure what you are doing, do yourself a favor and don't touch anything. Ask someone who knows!

You'll use two voice commands:

- **ON** – activates the relay and lights up the lamp
- **OFF** – deactivates the relay and turns off the lamp

Besides voice commands, you'll also be able to manually control the relay with buttons.

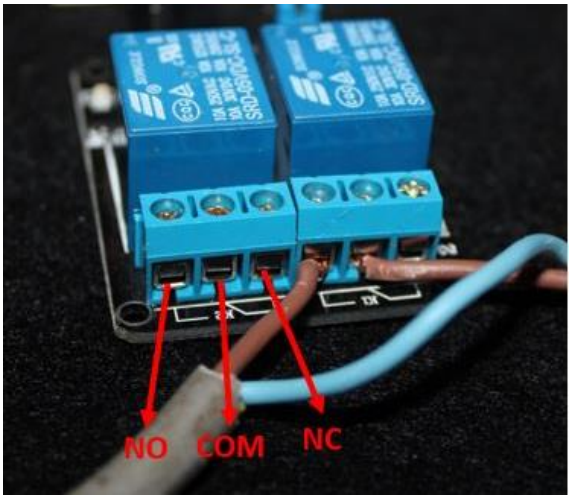
Note: if you're not comfortable dealing with mains voltage, but you still want to try to do the project, you can replace the relay module with an LED, for example. The code and the schematic are similar.

Relay module

A relay is an electrically operated switch. It means that it can be turned on or off, letting the current going through or not. The relay module is the one in the figure below.

This particular relay module comes with two relays (those blue cubes).

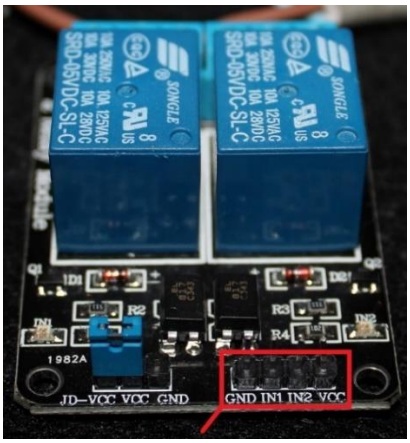
About mains voltage, relays have 3 possible connections:



- COM: common pin
- NO: normally open – there is no contact between the common pin and the normally open pin. So, when you trigger the relay, it connects to the COM pin and power is provided to the load (a desktop lamp, in our case).
- NC: normally closed – there is contact between the common pin and the normally closed pin. There is always between the COM and NC pins, even when the relay is turned off. When you trigger the relay, the circuit is opened and there is no power provided to the load.

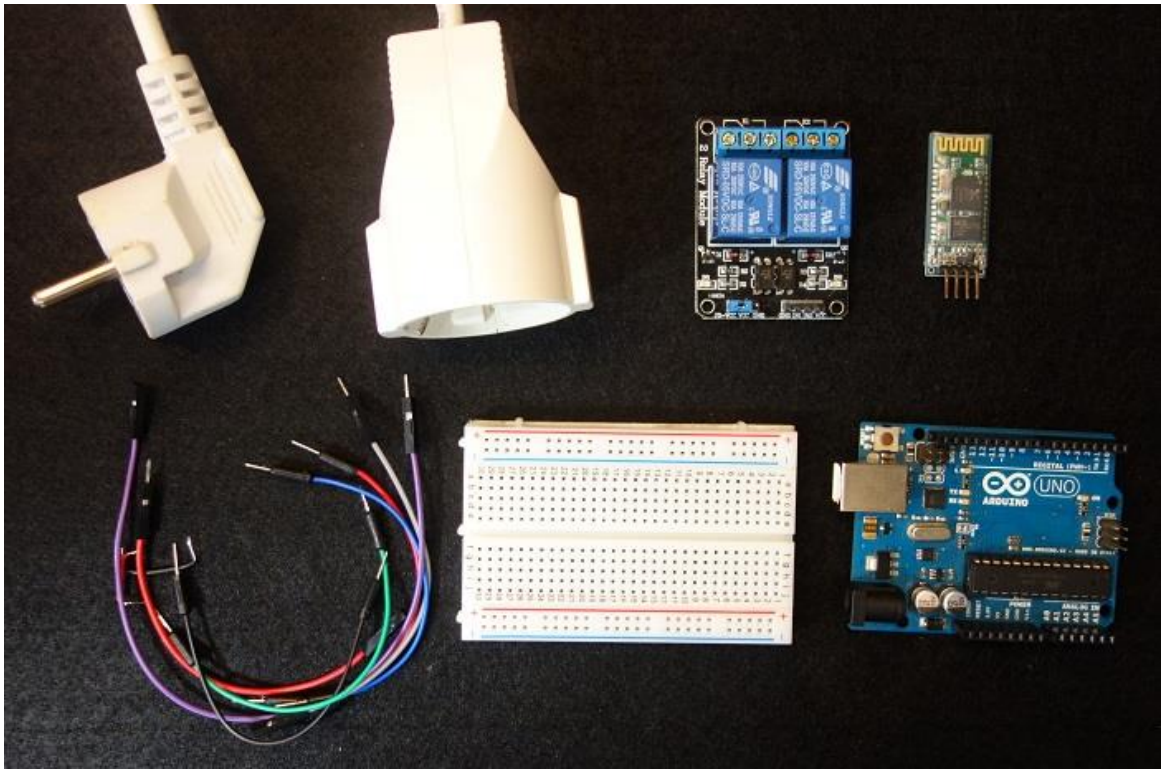
Relating this project, it is better to use a normally open circuit, because we just want to light up the lamp occasionally.

The connections between the relay and the Arduino are really simple:



- GND: goes to ground
- IN1: controls the first relay. Should be connected to an Arduino digital pin
- IN2: controls the second relay. Should be connected to an Arduino digital pin
- VCC: goes to 5V

Parts required



Grab all the components needed for this project. Note that you need a female and a male socket. I didn't add these items to the table because they are different accordingly to your country. Of course, you also need an AC wire. It can be one from an old electronics appliance, as long as the wire is still in good shape.

Figure	Name	eBay
	Arduino Uno	http://ebay.to/1SQda0R
	Relay Module	http://ebay.to/2dRv8lh
	HC-04 or HC-05 or HC-06 Bluetooth module	http://ebay.to/2dt1b6M
	Breadboard	http://ebay.to/21bEojM
	Jumper Wires	http://ebay.to/1PXeaJz

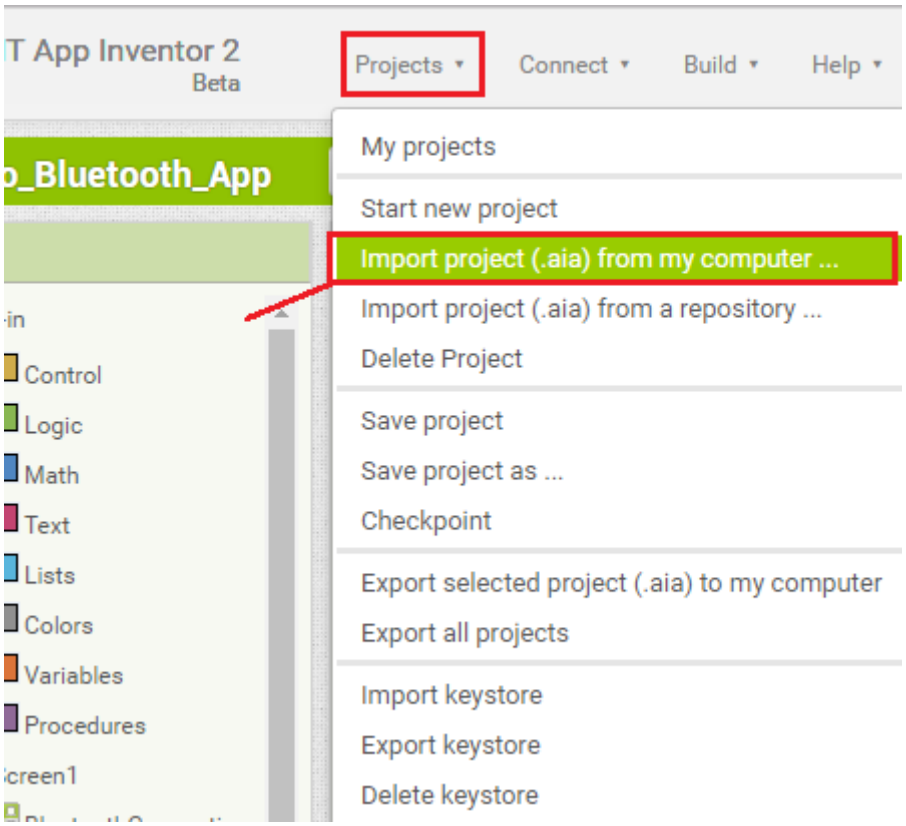
Creating your Android App

The Android App will be created using a free web application called MIT App Inventor. MIT App Inventor is a great place to get started with Android development, because it allows you to build simple apps with drag-n-drop.

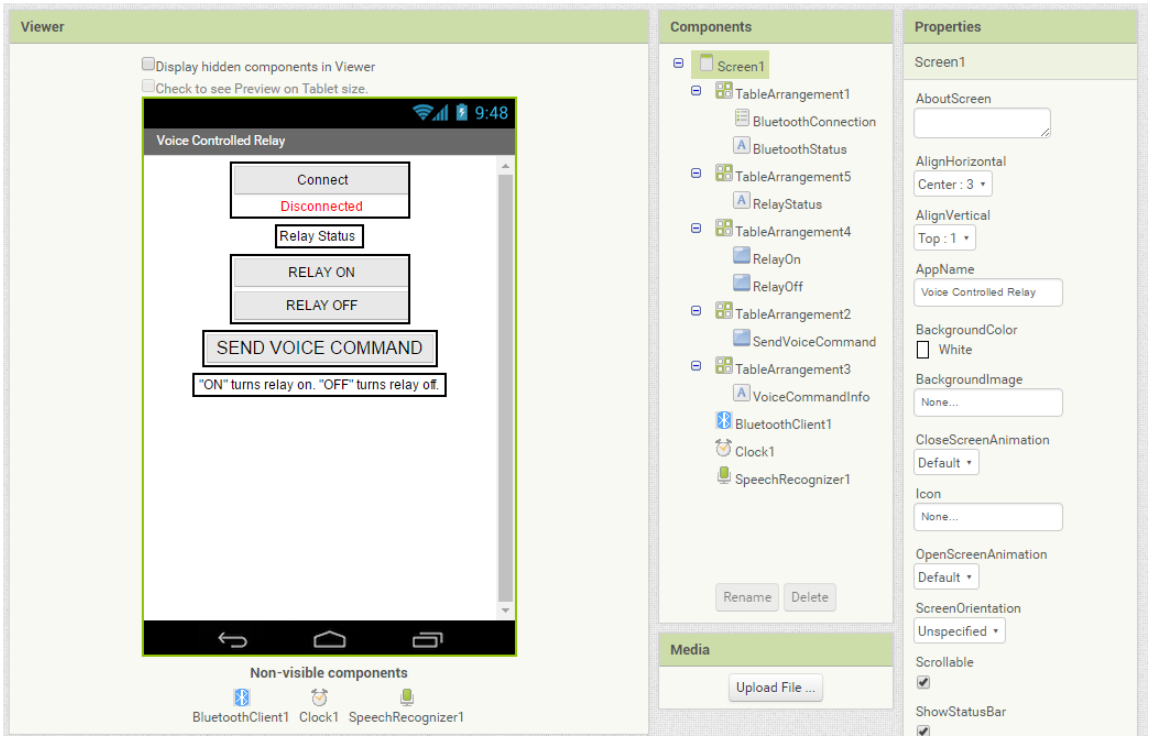
You need a Google account to sign up for MIT App Inventor and here's the login page: http://ai2.appinventor.mit.edu.

Click here to download the .aia file

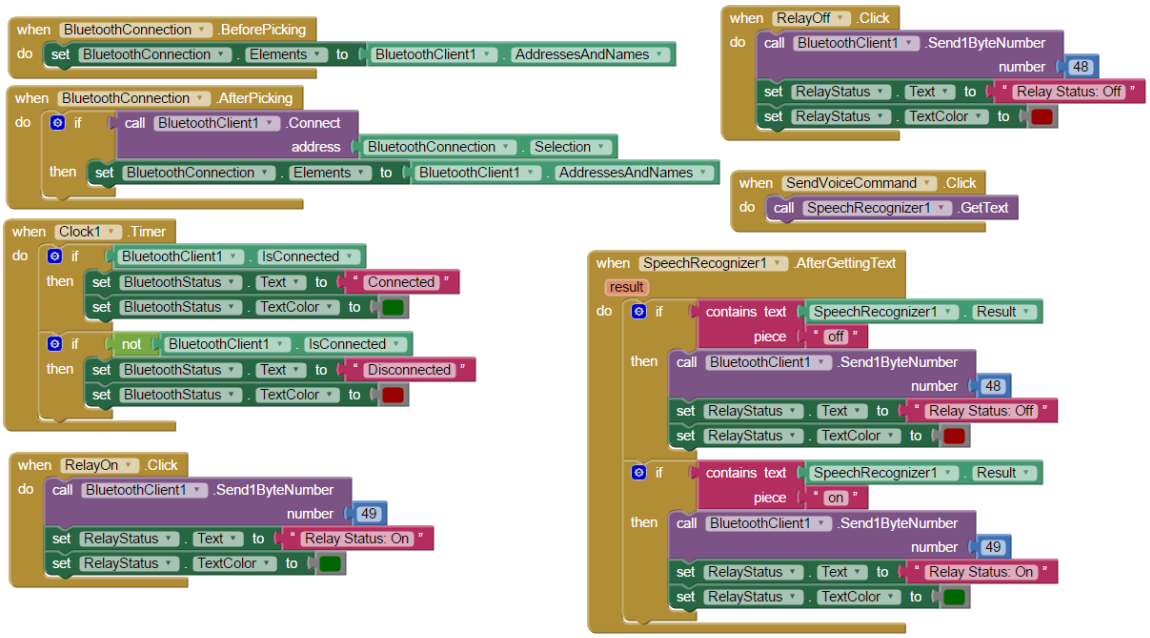
If you go to the Projects tab, you can upload the .aia file for this project.



With MIT App Inventor you have 2 main sections: designer and blocks. The designer is what gives you the ability to add buttons, add text, add screens and edit the overall app look.



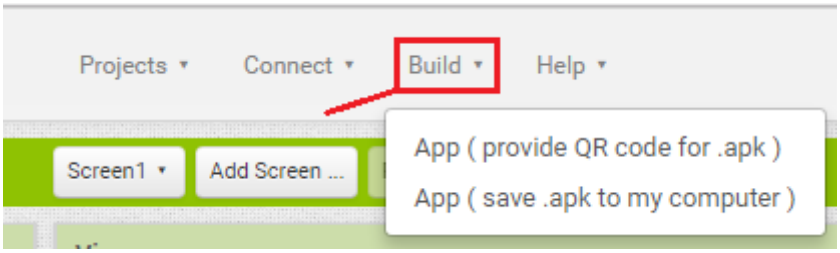
The blocks sections is what allows to create custom functionality for your app, so when you press the buttons it actually does something with that information.



I recommend that you start by following this project and using the app without modifying it.

If want to make any changes to the app, when you're done and you want to install the app in your smartphone, go to the Build tab.

You can either generate a QR code that you can scan with your smartphone and automatically install the app in your smartphone, or you can download the .apk file, connect your smartphone to your computer and move the .apk file to the phone.



Simply follow the installation wizard to install the App and it's done!

Code

For this project, you don't need to install any Arduino libraries. So, you just have to download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure that you have the right Board and COM port selected.

Warning: do not upload a new code to your Arduino board while your lamp is connected to the mains voltage. You should unplug the lamp from mains voltage, before upload a new sketch to your Arduino.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

int relay = 11;      // pin Digital 11
int state;           // saves the state
int flag=0;          // makes sure that the serial only prints once
the state

void setup() {
    // sets the Relay as output:
    pinMode(relay, OUTPUT);
    digitalWrite(relay, 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 relay will turn OFF
    if (state == '0') {
        digitalWrite(relay, HIGH);
        if(flag == 0){
```

```
            Serial.println("Relay Off!");
            flag=1;
        }
    }

    // if the state is '1' the relay will turn ON
    else if (state == '1') {
        digitalWrite(relay, LOW);
        if(flag == 0){
            Serial.println("Relay On!");
            flag=1;
        }
    }
    //Uncomment For debugging purpose
    //Serial.println(state);
}
```

Source code

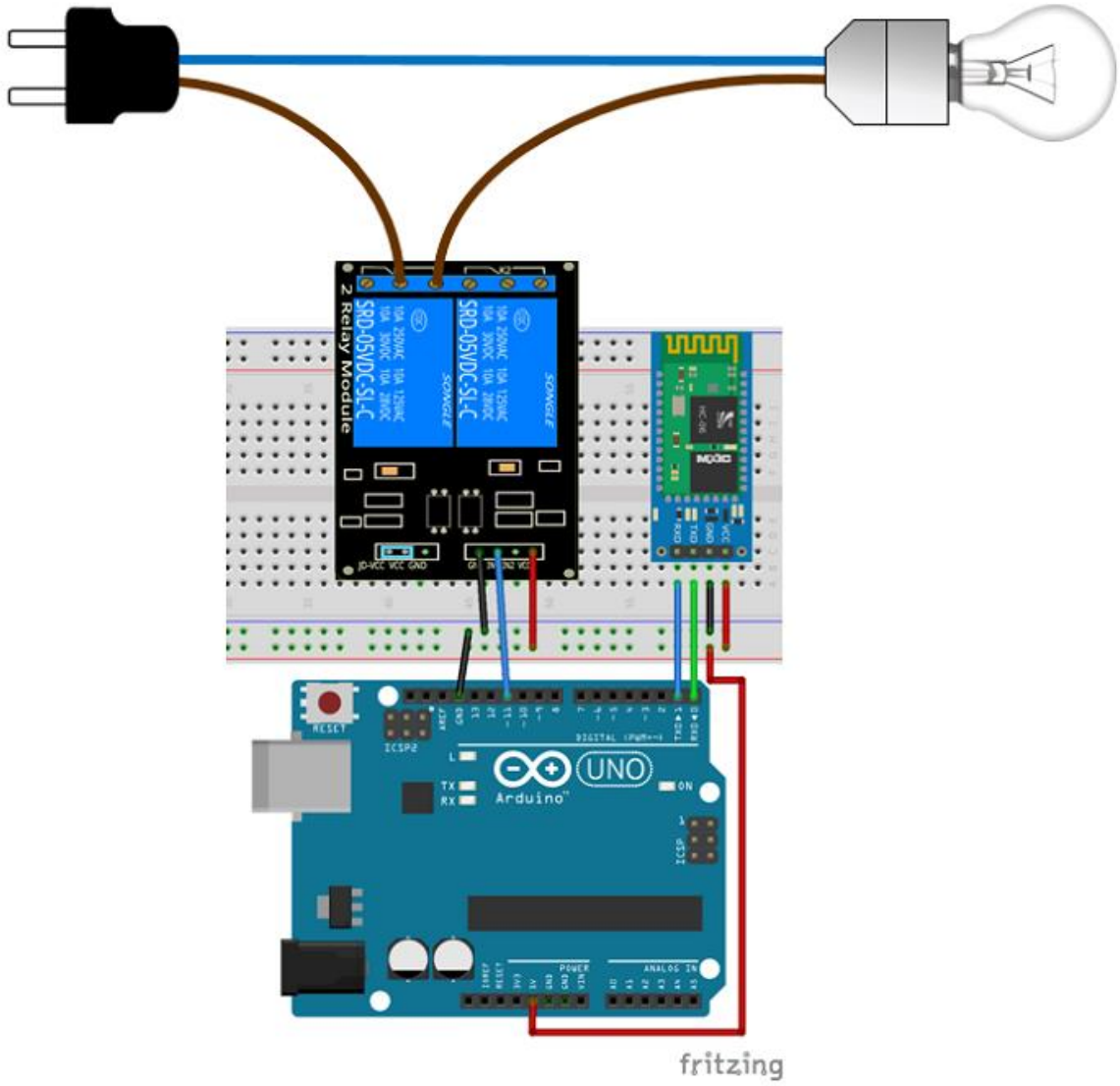
https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/14_voice_controlled_relay.ino

Note: before uploading the code, make sure you have the TX and RX pins disconnected from the bluetooth module!

Schematics

Here's the schematics for this project. Be very careful with all the connections, mainly with the mains voltage connections.

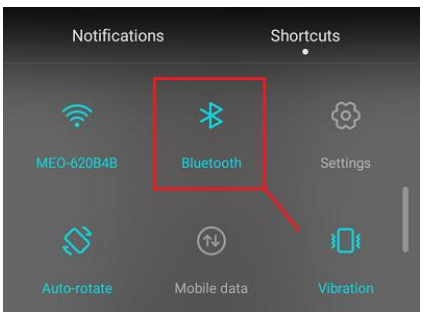
Warning: I can't stress this enough, but do not touch the relay or any live wires while they are connected to the mains voltage. This can be dangerous, if you're not comfortable dealing with mains voltage, do yourself a favor and don't touch anything.



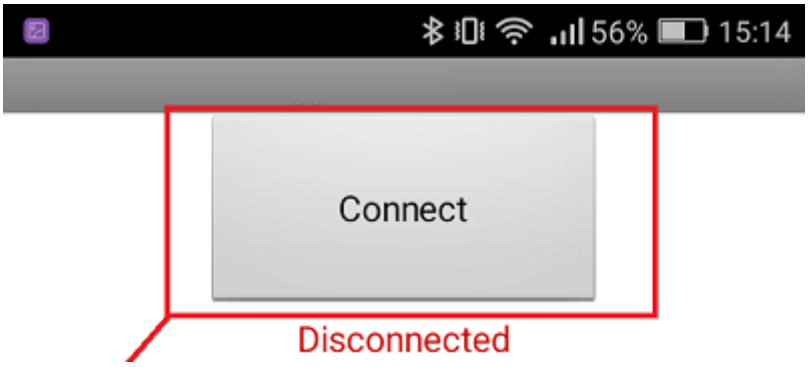
Launching your App

If you haven't generated the .apk file in a previous step, you can click here to download the .apk file (which is the Android App installation file). Move that file to your smartphone and open it. Follow the installation wizard to install the app.

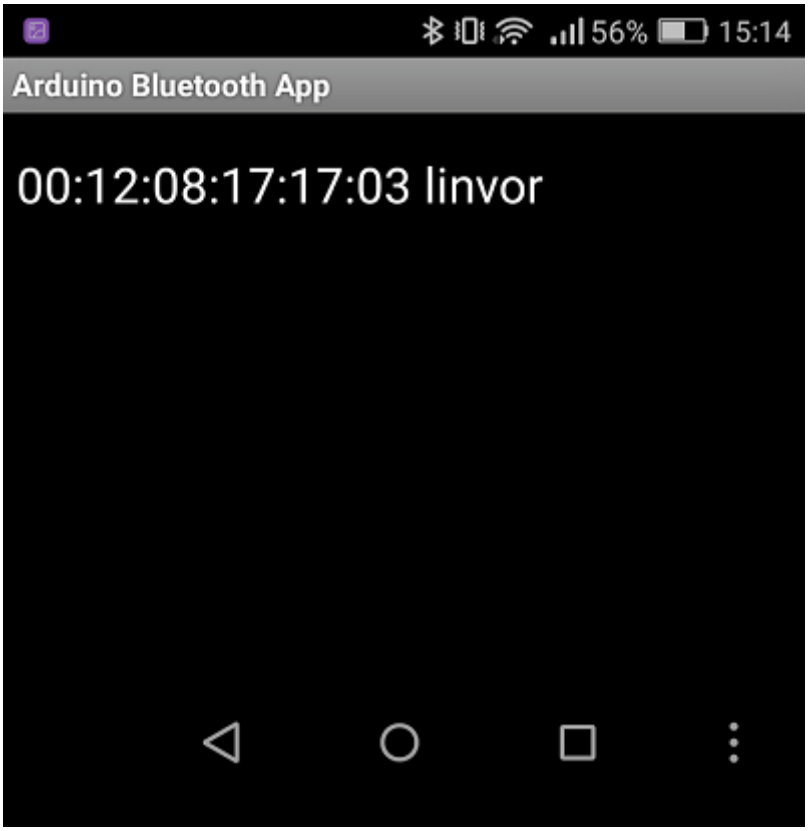
Turn on your smartphone's Bluetooth.



Tap on the newly installed app. Press the “Connect” button to connect your application to your Arduino Bluetooth module.



Select your Bluetooth module (it should be named linvor).



Now, it is ready to use!

Demonstration

Here’s your app in action.

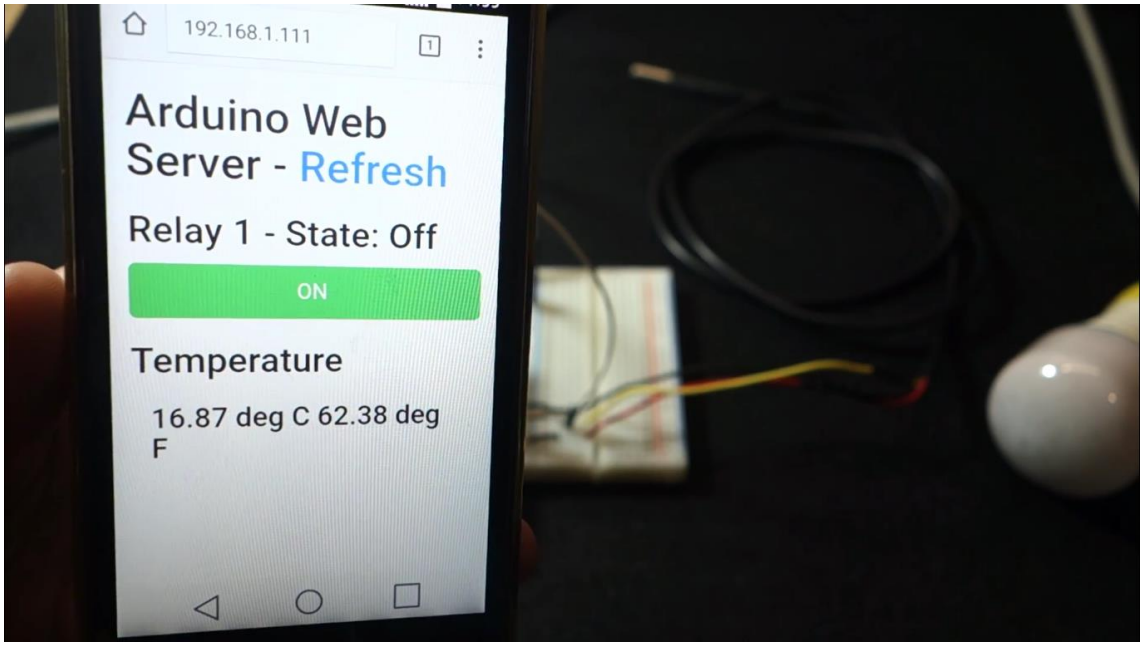
You are able to control the relay with “ON” and “OFF” voice commands. You can also manually control the relay with buttons.

Wrapping up

In this project you learned how to control an output with voice commands with an Android app.

You can replace the desktop lamp with any AC electronics appliance you want.

Arduino Ethernet Web Server



Ethernet Web Server (Relay + Temperature)

Level: Advanced – Time: 1 hour

In this project you're going to build a password protected web server with a relay module and a temperature sensor. You can access your web server with any device that has a browser and it's connected to the same network.

Here are the main features of the web server:

- is accessible with any device in your network that has a browser
- is username/password protected
- controls a relay module that is attached to a lamp
- displays temperature in Celsius and Fahrenheit

This project uses the following components:

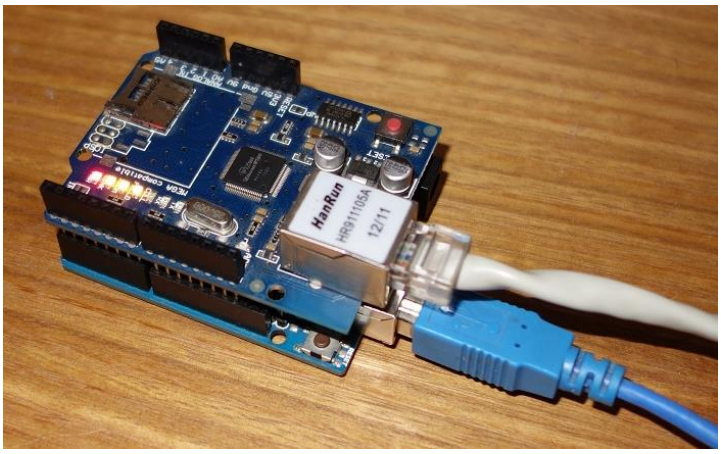
- Arduino with Ethernet shield
- Relay module with lamp
- DS18B20 temperature sensor

Note: if you're not comfortable dealing with mains voltage, but you still want to try to do the project, you can replace the relay module with an LED, for example. The code and the schematics are very similar.

Ethernet shield

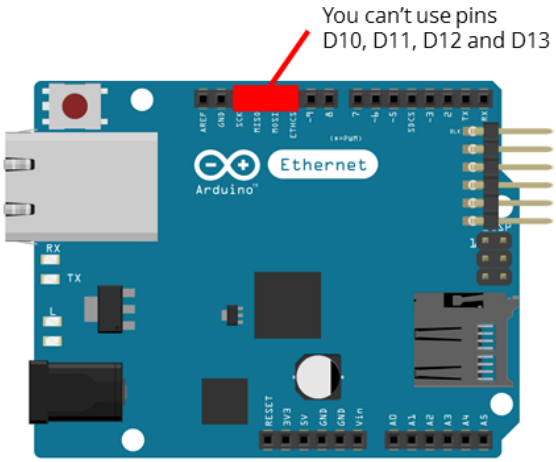
The Arduino Ethernet shield connects your Arduino to the internet in a simple way. Just mount this module onto your Arduino board, connect it to your network with an RJ45 cable and follow a few simple steps to start controlling your projects through the web.

Note: you must connect an Ethernet cable from your router to your Ethernet shield as shown in the following figure.



Pin usage

When the Arduino is connected to an Ethernet shield, you can't use Digital pins from 10 to 13, because they are being used in order to establish a communication between the Arduino and the Ethernet shield.

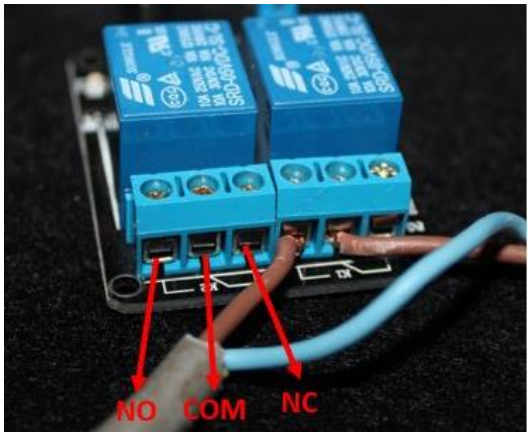


Relay module

A relay is an electrically operated switch. It means that it can be turned on or off, letting the current going through or not. The relay module is the one in the figure below.

This particular relay module comes with two relays (those blue cubes).

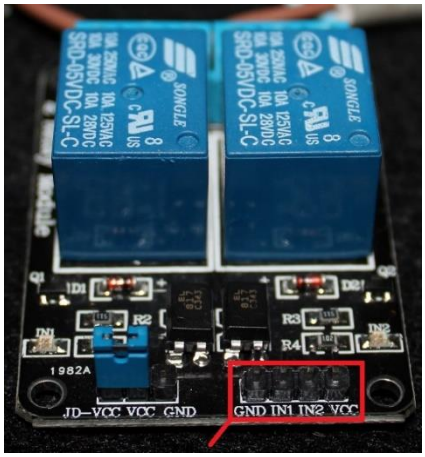
About mains voltage, relays have 3 possible connections:



- COM: common pin
- NO: normally open – there is no contact between the common pin and the normally open pin. So, when you trigger the relay, it connects to the COM pin and power is provided to the load (a lamp, in our case).
- NC: normally closed – there is contact between the common pin and the normally closed pin. There is always contact between the COM and NC pins, even when the relay is turned off. When you trigger the relay, the circuit is opened and there is no power provided to the load.

Relating this project, it is better to use a normally open circuit, because we want to light up the lamp occasionally.

The connections between the relay and the Arduino are really simple:

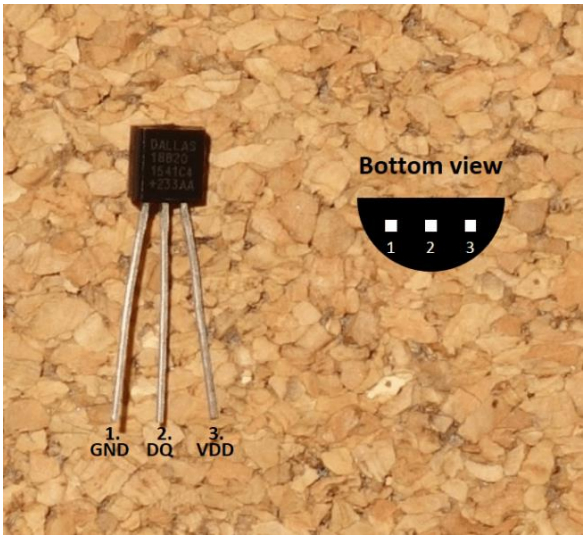


- GND: goes to ground
- IN1: controls the first relay. Should be connected to an Arduino digital pin
- IN2: controls the second relay. Should be connected to an Arduino digital pin
- VCC: goes to 5V

DS18B20 temperature sensor

The DS18B20 temperature sensor is a 1-wire digital temperature sensor. This means that you can read the temperature with a very simple circuit setup. It communicates on common bus, which means that you can connect several devices and read their values using one Arduino digital pin.

The sensor has just three pins as you can see in the following figure:

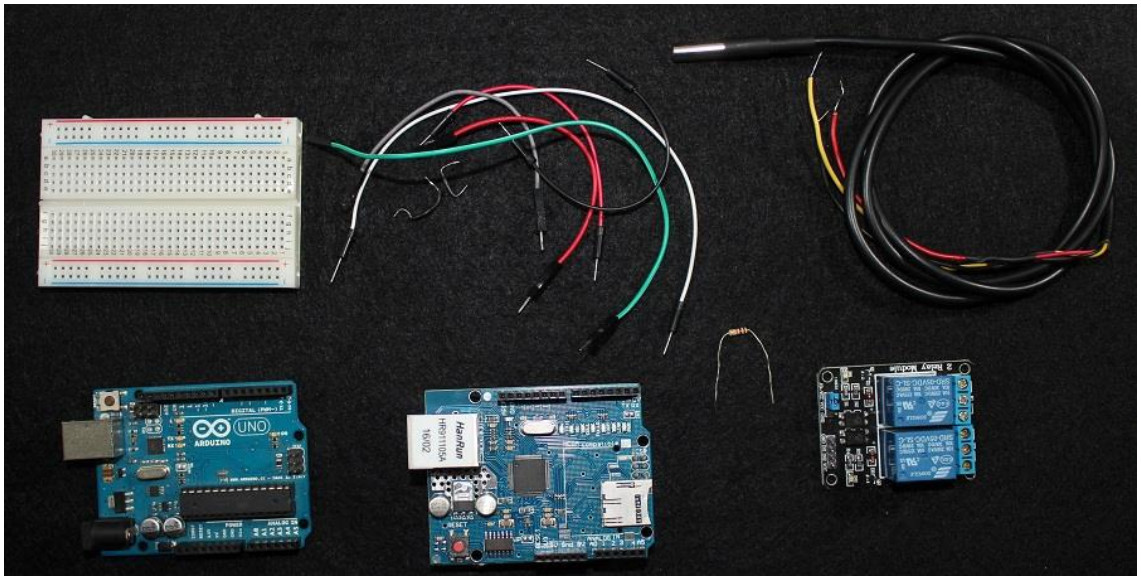


The DS18B20 is also available in waterproof version:

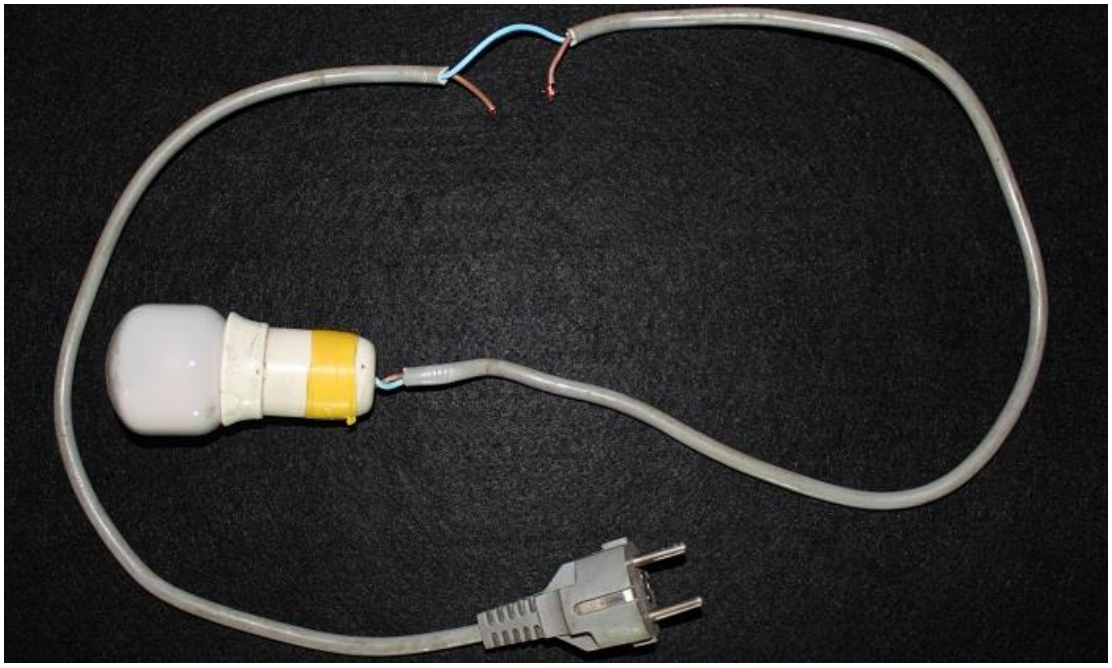
Here's some features of the DS18B20 temperature sensor:

- Communicates over 1-wire bus communication
- Operating range temperature: -55ºC to 125ºC
- Accuracy +/-0.5 ºC (between the range -10ºC to 85ºC)

Parts required



Besides these electronics components, you'll also need an AC male socket, an AC wire and a lamp bulb holder (a lamp cord set).



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Arduino Uno	http://ebay.to/1SQda0R

	Ethernet Shield (WIZnet W5100)	http://ebay.to/1WWpuhv
	DS18B20	http://ebay.to/2aRm6DC
	10kOhm Resistor	http://ebay.to/1KsMYFP
	Relay Module	http://ebay.to/2dRv8lh
	Lamp Cord Set	http://ebay.to/2feHELc
	Breadboard	http://ebay.to/21bEojM
	Jumper Wires	http://ebay.to/1PXeaJz

Code

For this project, you need to install two Arduino libraries: OneWire library and DallasTemperature library

Installing the OneWire library

- Click here to download the OneWire library. You should have a .zip folder in your Downloads
- Unzip the .zip folder and you should get OneWire-master folder
- Rename your folder from ~~OneWire-master~~ to OneWire
- Move the OneWire folder to your Arduino IDE installation libraries folder
- Finally, re-open your Arduino IDE

Installing the DallasTemperature library

- Click here to download the DallasTemperature library. You should have a .zip folder in your Downloads

- Unzip the .zip folder and you should get Arduino-Temperature-Control-Library-master folder
- Rename your folder from ~~Arduino-Temperature-Control-Library-master~~ to DallasTemperature
- Move the DallasTemperature folder to your Arduino IDE installation libraries folder
- Finally, re-open your Arduino IDE

Uploading code

Copy the following code to your Arduino IDE and before uploading it to your Arduino board make sure you read these two extra steps described below:

A) Configuring your network

B) Encoding your username/password

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
*/

#include <SPI.h>
#include <Ethernet.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is conntec to the Arduino digital pin 2
#define ONE_WIRE_BUS 2

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1, 111);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

// Relay state and pin
String relay1State = "Off";
const int relay = 7;

// Setup a oneWire instance to communicate with any OneWire devices
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature sensor
DallasTemperature sensors(&oneWire);

// Client variables
char linebuf[80];
int charcount=0;
```

```
boolean authentificated=false;

void setup() {
  // Start up the sensors library
  sensors.begin();
  // Relay module prepared
  pinMode(relay, OUTPUT);
  digitalWrite(relay, HIGH);

  // Open serial communication at a baud rate of 9600
  Serial.begin(9600);

  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}

// Display dashboard page with on/off button for relay
// It also print Temperature in C and F
void dashboardPage(EthernetClient &client) {
  client.println("<!DOCTYPE HTML><html><head>");
  client.println("<meta name=\"viewport\" content=\"width=device-
width, initial-scale=1\">");
  client.println("<link rel=\"stylesheet\"
href=\"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstra
p.min.css\">");

  client.println("<script
src=\"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.
min.js\"></script>");

  client.println("</head><div class=\"container\">");
  client.println("<h1>Arduino Web Server - <a
href=\"/\">Refresh</a></h1>");
  // Generates buttons to control the relay
  client.println("<h2>Relay 1 - State: " + relay1State + "</h2><div
class=\"row\">");
  // If relay is off, it shows the button to turn the output
on
  if(relay1State == "Off"){
    client.println("<div class=\"col-md-2\"><a href=\"/relay1on\"
class=\"btn btn-block btn-lg btn-success\"
role=\"button\">ON</a></div></div>");
  }
  // If relay is on, it shows the button to turn the output
off
  else if(relay1State == "On"){
    client.println("<div class=\"col-md-2\"><a href=\"/relay1off\"
class=\"btn btn-block btn-lg btn-danger\"
role=\"button\">OFF</a></div></div>");

  }
  // Displays temperature in C and F
  client.println("<h2>Temperature</h2><div class=\"col-md-
2\"><h3>");
  sensors.requestTemperatures();
```

```
  client.print(sensors.getTempCByIndex(0));
  client.print(" deg C ");
  client.print(sensors.getTempFByIndex(0));
  client.print(" deg F</h3></div>");
  client.println("</div></body></html>");
}

// If login fails, display 401 page
void failedLoginPage(EthernetClient &client){
  client.println("HTTP/1.1 401 Authorization Required");
  client.println("WWW-Authenticate: Basic realm=\"Secure Area\"");
  client.println("Content-Type: text/html");
  client.println("Connnection: close");
  client.println();
  client.println("<!DOCTYPE html><html>");
  client.println("<head><title>Login Failed</title></head>");
  client.println("<body><h1>Login Failed.</h1></body></html>");
}

void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    memset(linebuf,0,sizeof(linebuf));
    charcount=0;
    authentificated=false;
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
       char c = client.read();
       //read char by char HTTP request
        linebuf[charcount]=c;
        if (charcount<sizeof(linebuf)-1) charcount++;
        // if you've gotten to the end of the line (received a
newline
        // character) and the line is blank, the http request has
ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          if (authentificated)
            dashboardPage(client);
          else
            failedLoginPage(client);
          break;
        }
        if (c == '\n') {
          if (strstr(linebuf,"GET /relay1off") > 0){
            digitalWrite(relay, HIGH);
            relay1State = "Off";
          }
          else if (strstr(linebuf,"GET /relay1on") > 0){
            digitalWrite(relay, LOW);
            relay1State = "On";
          }
          // you're starting a new line
          currentLineIsBlank = true;
```

```
          // checking if header is valid and contains the right
username/password combination
          // dXNlcjpwYXNz = 'user:pass' (user:pass) base64 encode
          if (strstr(linebuf,"dXNlcjpwYXNz")>0 &&
strstr(linebuf,"Authorization: Basic")>0)
            authentificated=true;
          memset(linebuf,0,sizeof(linebuf));
          charcount=0;
        }
        else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
    // give the web browser time to receive the data
    delay(1);
    // close the connection:
    client.stop();
    Serial.println("client disonnected");
  }
}
```

Warning: do not upload a new code to your Arduino board while your lamp is connected to the mains voltage. You should unplug the lamp from mains voltage, before upload a new sketch to your Arduino.

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/19_arduino_web_server.ino

A) Configuring your network

Take a look at the configuring your network code snippet:

```
byte mac[] = {
0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,XXX);
```

Important: you actually might need to replace that variable highlighted in red with appropriate values that are suitable for your network, otherwise your Arduino will not establish a connection with your network.

Replace the following line with an IP that is available and suitable for your network:

```
IPAddress ip(X, X, X, X);
```

In my case, my IP range is **192.168.1.X** and with the software Angry IP Scanner I know that the IP **192.168.1.111** is available in my network, because it doesn't have any active device in my network with that exact same IP address:

```
IPAddress ip(192, 168, 1, 111);
```

B) Encoding your username/password

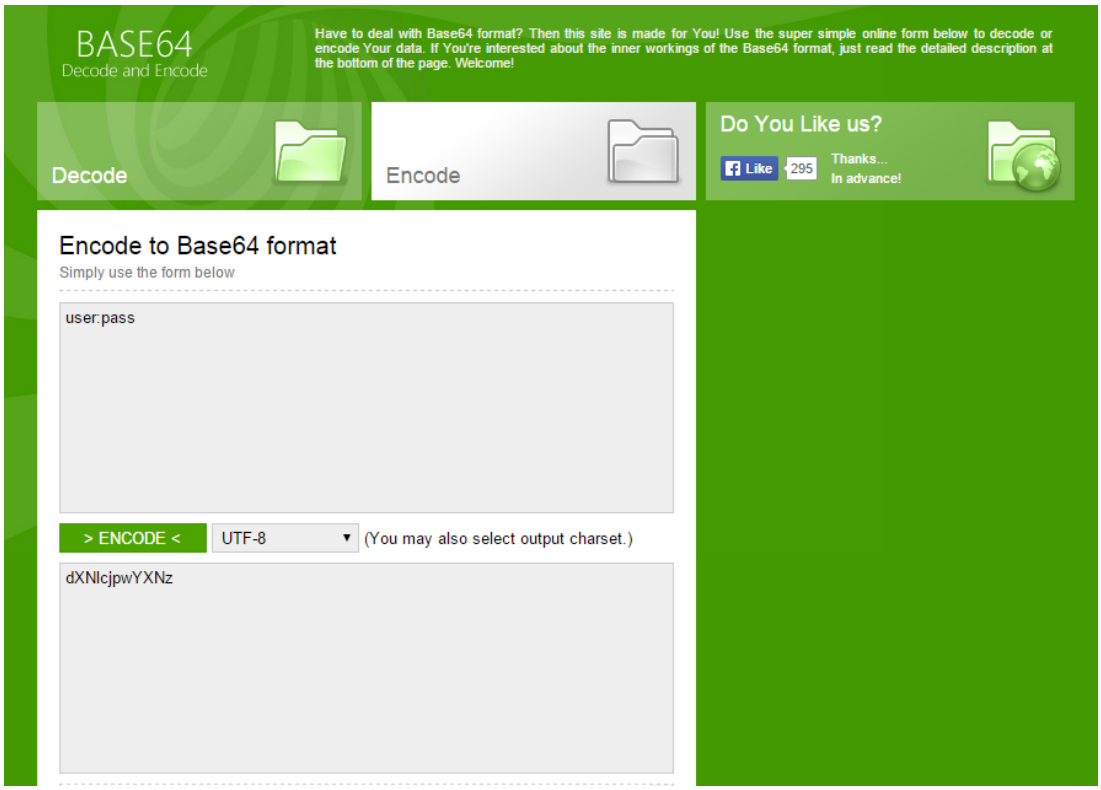
At this point if you upload the code created in the preceding section, your username is user and your password is pass. I'm sure you want to change and customize this example with your own credentials.

Go to the following URL: https://www.base64encode.org. In the first field, type the following:

your_username:your_password

Note: you actually need to type the ":" between your username and your password.

In my example, I've entered **user:pass** (as you can see in the Figure below):



Then Press the green "Encode" button to generate your base64 encoded string. In my example is **dXNlcjpwYXNz**.

Copy your string and replace it in this line of the sketch that you downloaded.

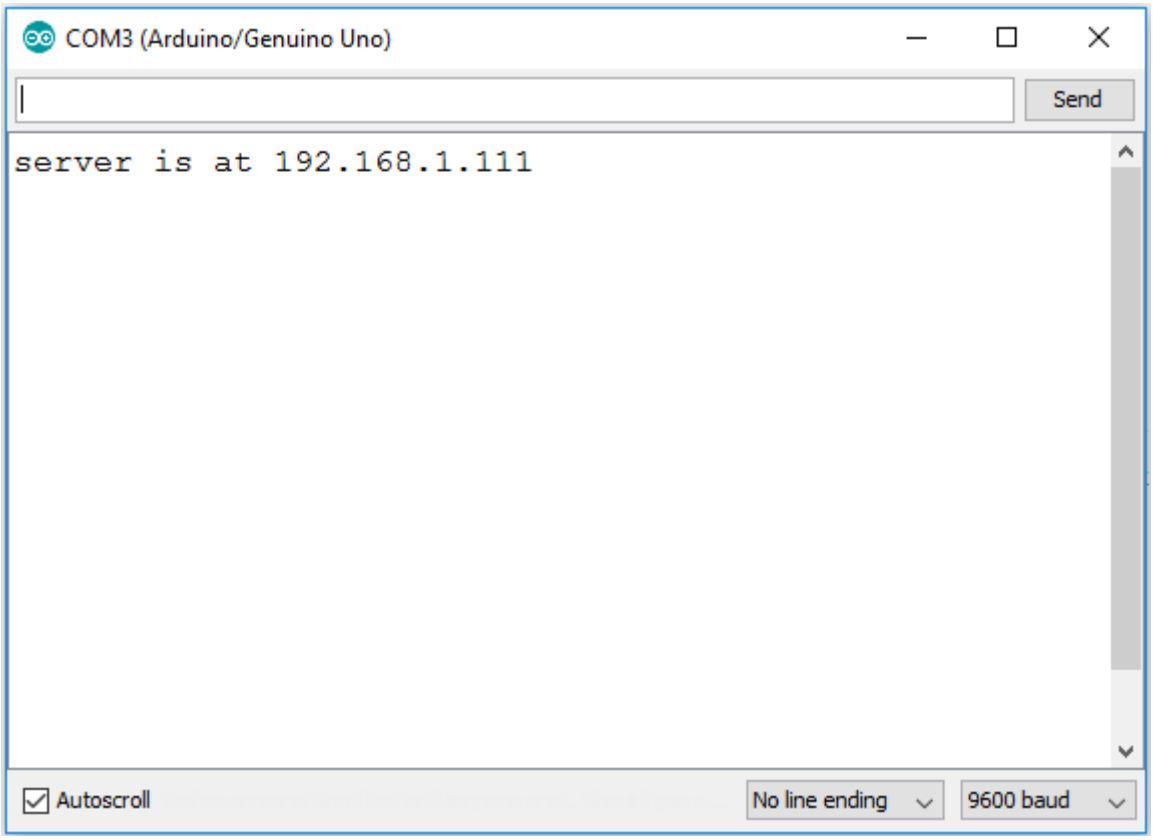
```
if (strstr(linebuf,"dXNlcjpwYXNz")>0 && strstr(linebuf,"Authorization: Basic")>0)
```

Discovering your web server IP address

Open the Arduino IDE serial monitor at a baud rate of 9600 and press the reset button in your Ethenet shield.

Your Arduino should print a message saying “server is at 192.168.1.111“.

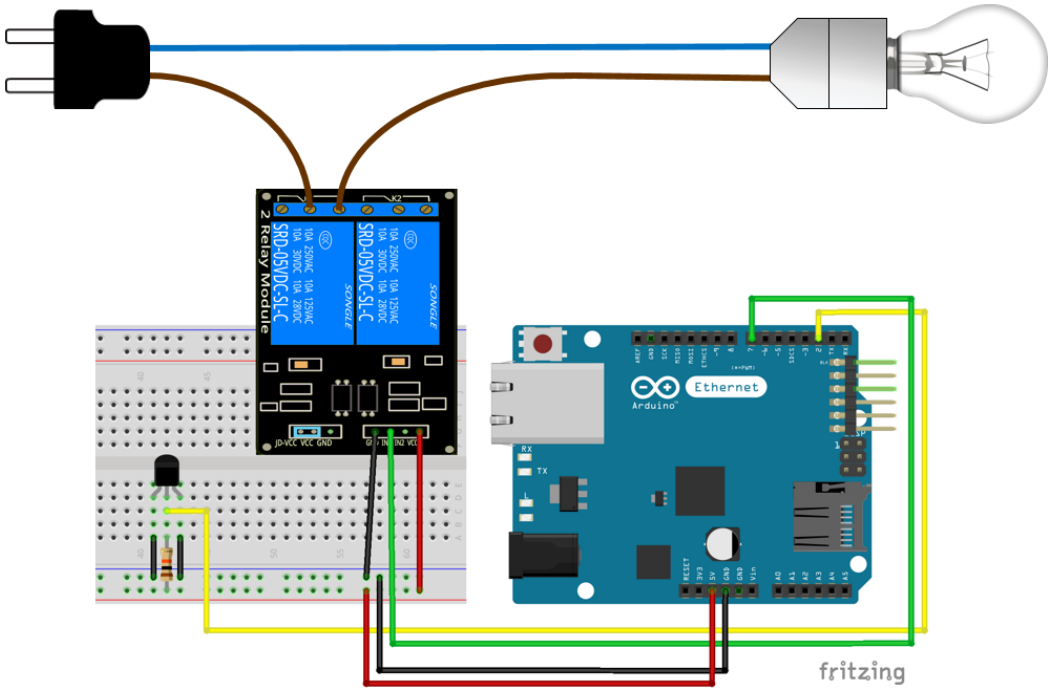
192.168.1.111 is your IP address, save that number because you’ll need it later.



Schematics

Here’s the schematics for this project. Be very careful with all the connections, mainly with the mains voltage connections.

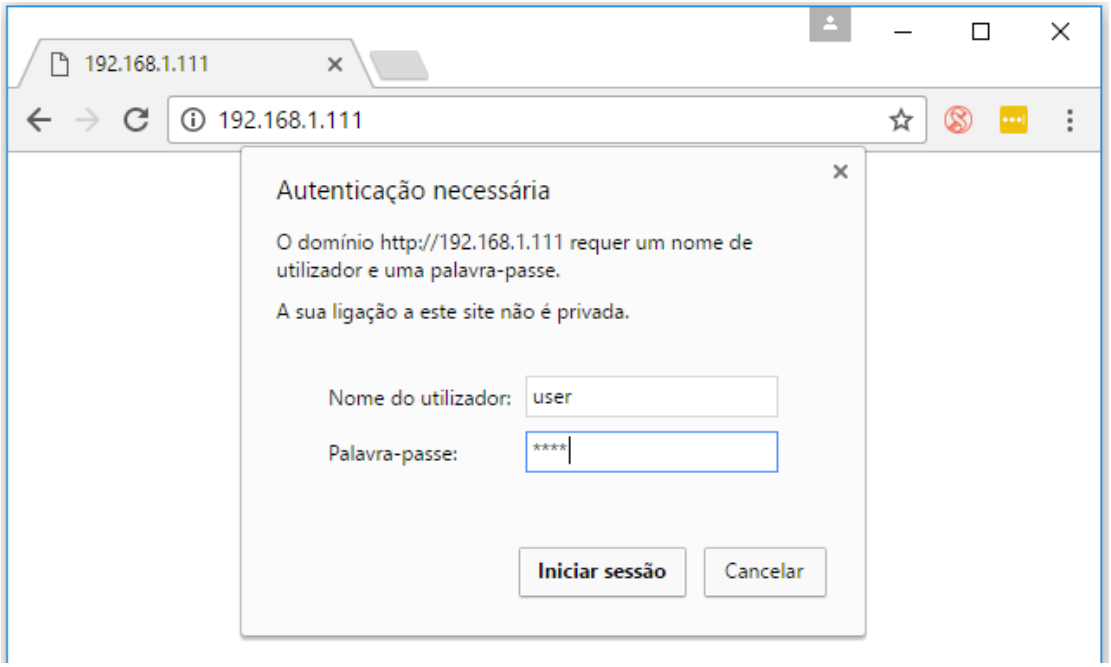
Warning: I can’t stress this enough, but do not touch the relay or any live wires while they are connected to the mains voltage. This can be dangerous, if you’re not comfortable dealing with mains voltage, do yourself a favor and don’t touch anything.



Accessing your web server

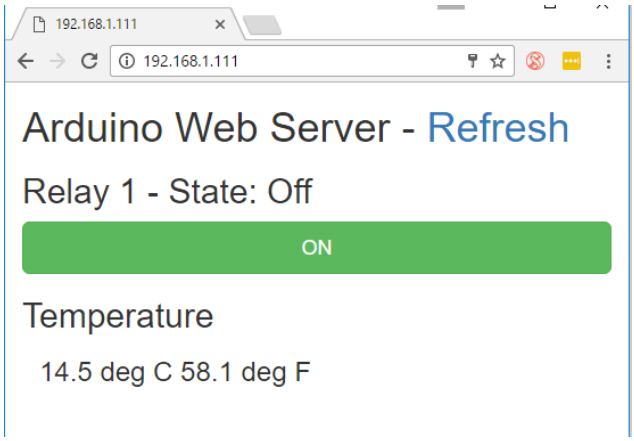
Now follow the next instructions before accessing your web server:

1. Open a browser
2. Type the IP address that you've previously saved in the URL bar (in my case: **http://192.168.1.111**)



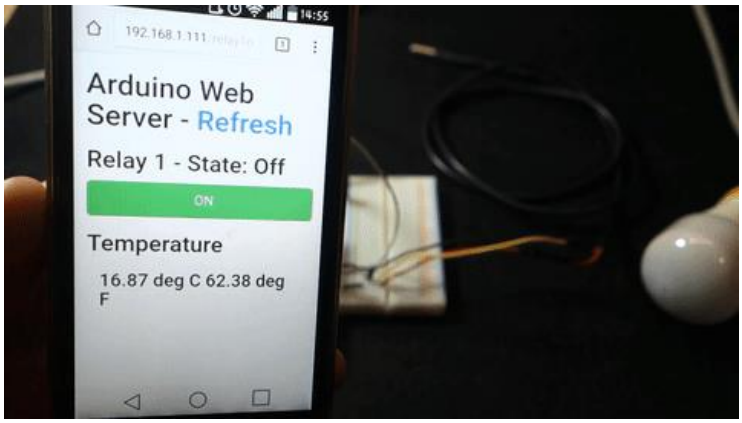
It should require that you enter your username and password in order to open your web server.

3. Enter your username (user) and password (pass).
4. Press Login.
5. A web page like the one below loads.

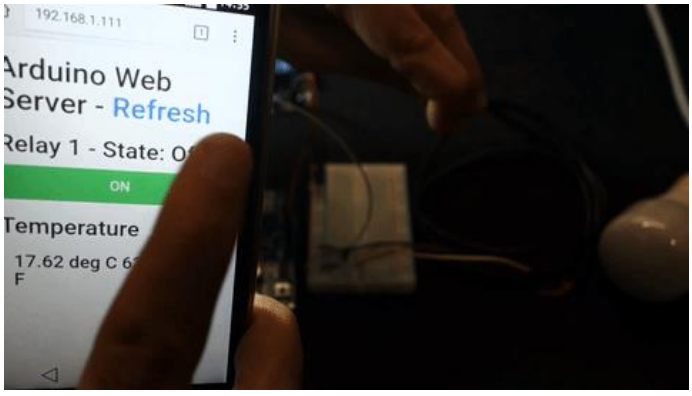


Demonstration

Here's your circuit in action. When you press the ON button the lamp lights up!



When you click the Refresh link, it updates the temperature in Celsius and Fahrenheit with latest readings:

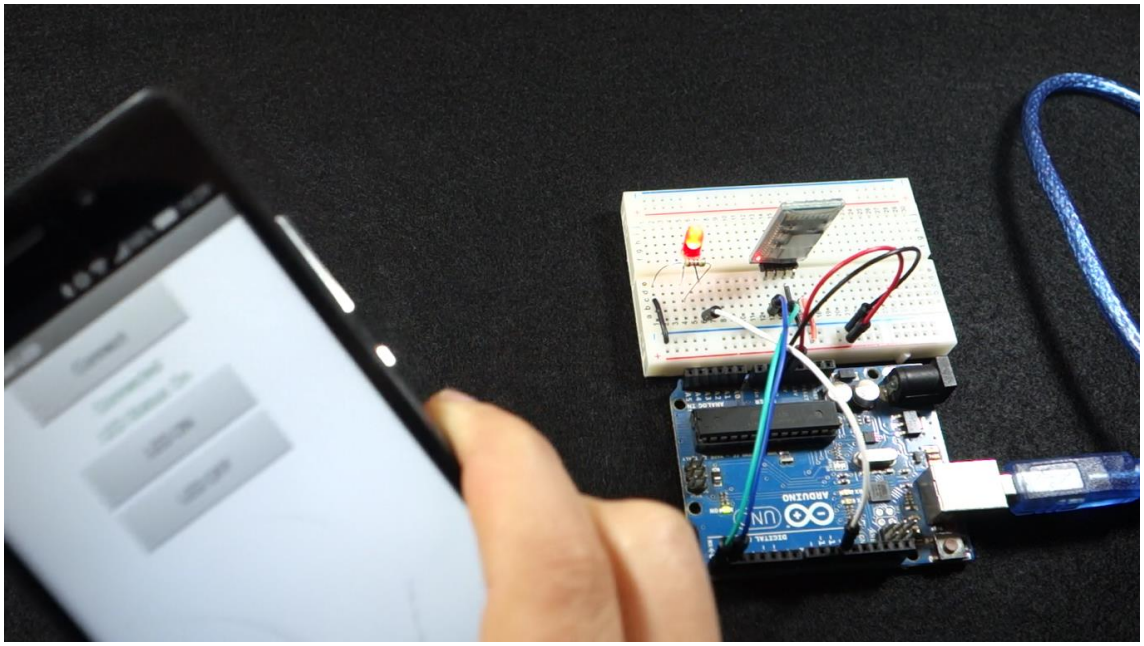


Wrapping up

In this project you've built a powerful web server that can be used to control house appliances or monitor your home.

With just a few changes you can modify this project to work with more sensors or control more outputs.

Shake Controlled LED



Shake Controlled LED (Android App)

Level: Intermediate – Time: 45 minutes

In this project you're going to build an Android app that controls an LED on and off when you shake your phone.

The app we're going to build also gives you the possibility to control the LED by taping the app buttons.

The smartphone will communicate with the Arduino using bluetooth.

Accelerometer

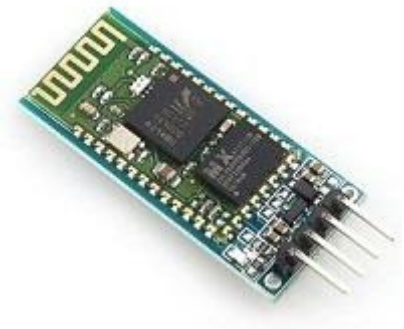
Recent smartphones have an accelerometer sensor. An accelerometer measures the linear acceleration of the device allowing you to measure changes in velocity and changes in position. It measures the acceleration across an XYZ axis.

The accelerometer has microscopic crystals that go under stress when vibration occurs. From that stress, results a voltage that allows you to measure the acceleration.

For this project, the accelerometer allows you to know if the smartphone was shake or not.

Bluetooth module HC-05

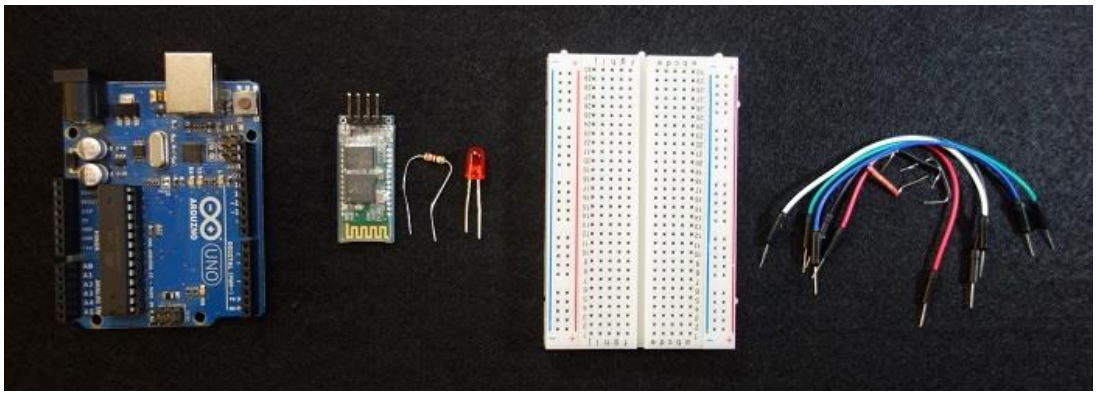
To establish the bluetooth communication between your smartphone and your Arduino, you need a bluetooth module. This project uses the HC-05 bluetooth module (as shown in the figure below).



This bluetooth module works with serial data. This means that the Arduino sends information and the bluetooth module receives it via serial (and vice-versa).

By default the HC-05 bluetooth module operates at a baud rate of 9600.

Parts required



Grab all the components needed for this project.

Figure	Name	eBay
	Arduino UNO	http://ebay.to/1SQda0R
	LED	http://ebay.to/20H2Oyy

	220Ohm Resistor	http://ebay.to/1KsMYFP
	HC-04 or HC-05 or HC-06 Bluetooth module	http://ebay.to/2dt1b6M
	Breadboard	http://ebay.to/21bEojM
	Jumper Wires	http://ebay.to/1PXeaJz

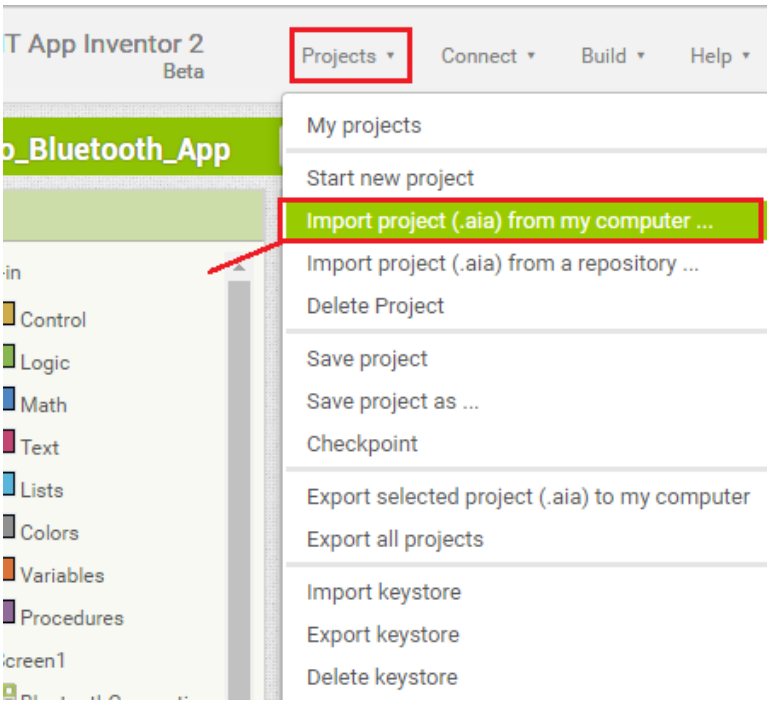
Creating your Android App

The Android App will be created using a free web application called MIT App Inventor. MIT App Inventor is a great place to get started with Android development, because it allows you to build simple apps with drag-n-drop.

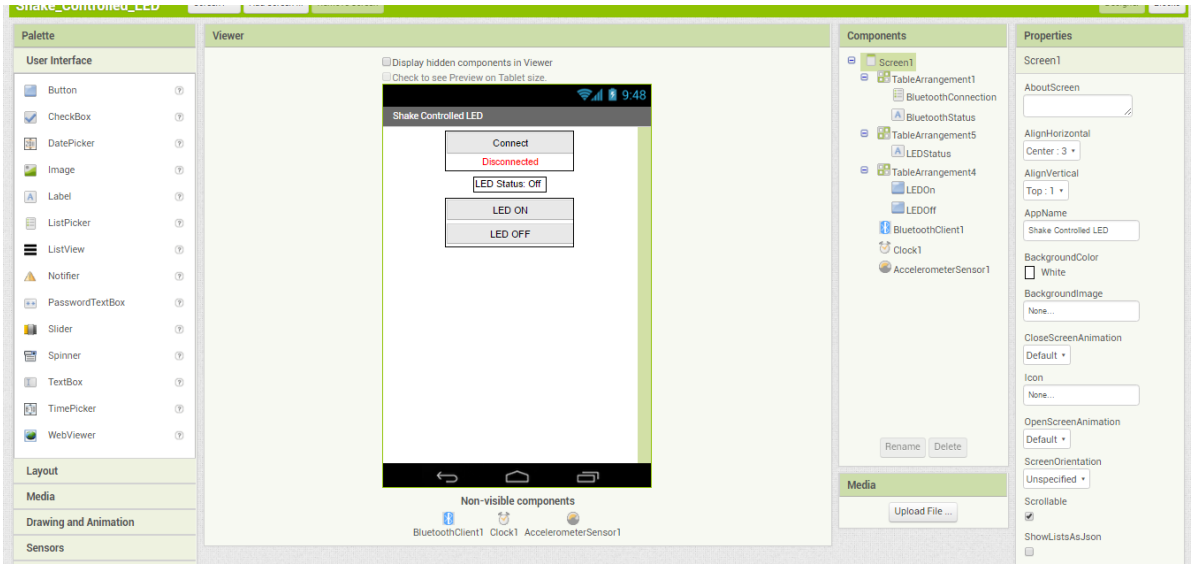
You need a Google account to sign up for MIT App Inventor and here's the login page: http://ai2.appinventor.mit.edu.

Click here to download the .aia file.

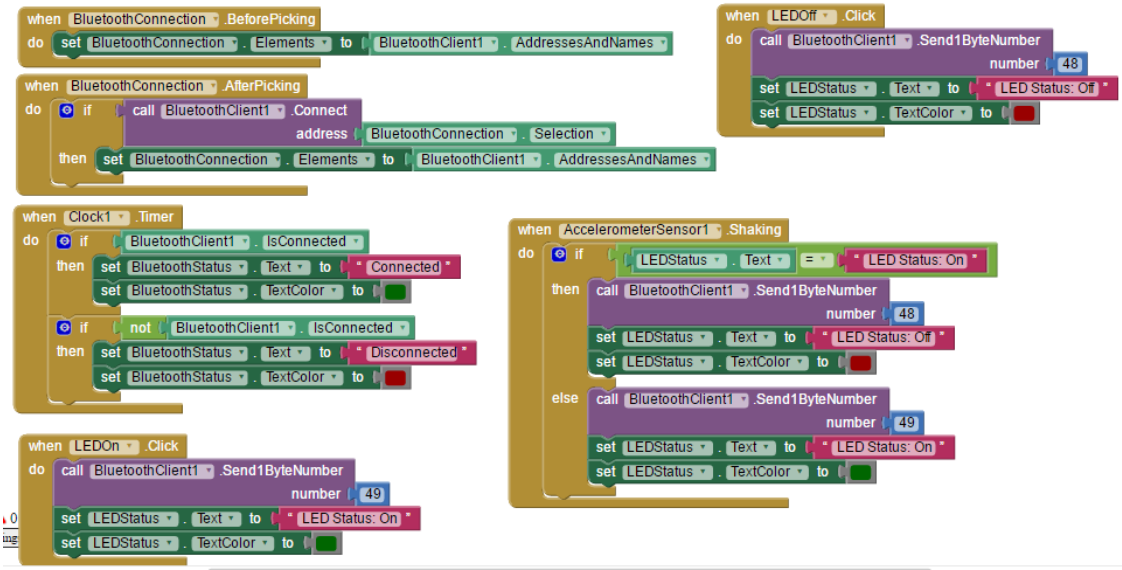
If you go to the Projects tab, you can upload the .aia file for this project.



With MIT App Inventor you have 2 main sections: designer and blocks. The designer is what gives you the ability to add buttons, add text, add screens and edit the overall app look.



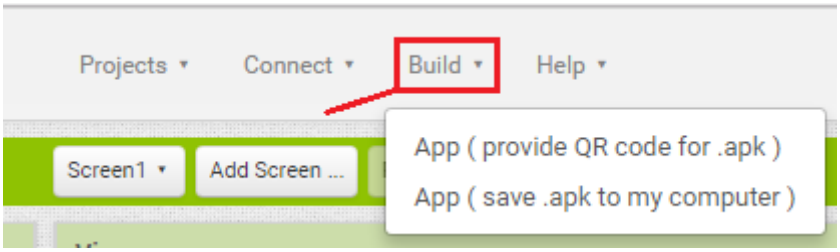
The blocks sections is what allows to create custom functionality for your app, so when you press the buttons it actually does something with that information.



I recommend that you start by following this project and using the app without modifying it.

If want to make any changes to the app, when you're done and you want to install the app in your smartphone, go to the Build tab.

You can either generate a QR code that you can scan with your smartphone and automatically install the app in your smartphone, or you can download the .apk file, connect your smartphone to your computer and move the .apk file to the phone.



Simply follow the installation wizard to install the App and it's done!

Code

For this project, you don't need to install any Arduino libraries. So, you just have to download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure that you have the right Board and COM port selected.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

int led = 11;      // pin Digital 11
int state;           // saves the state
int flag = 0;          // makes sure that the serial only prints
once the state

void setup() {
    // sets the Relay as output:
    pinMode(led, OUTPUT);
    digitalWrite(led, LOW);

    // 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 LED will turn OFF
    if (state == '0') {
        digitalWrite(led, LOW);
        if(flag == 0){
          Serial.println("LED Off!");
          flag=1;
        }
    }

    // if the state is '1' the LED will turn ON
    else if (state == '1') {
        digitalWrite(led, HIGH);
        if(flag == 0){
          Serial.println("LED On!");
          flag=1;
```

```
        }
    }
    //Uncomment For debugging purpose
    //Serial.println(state);
}
```

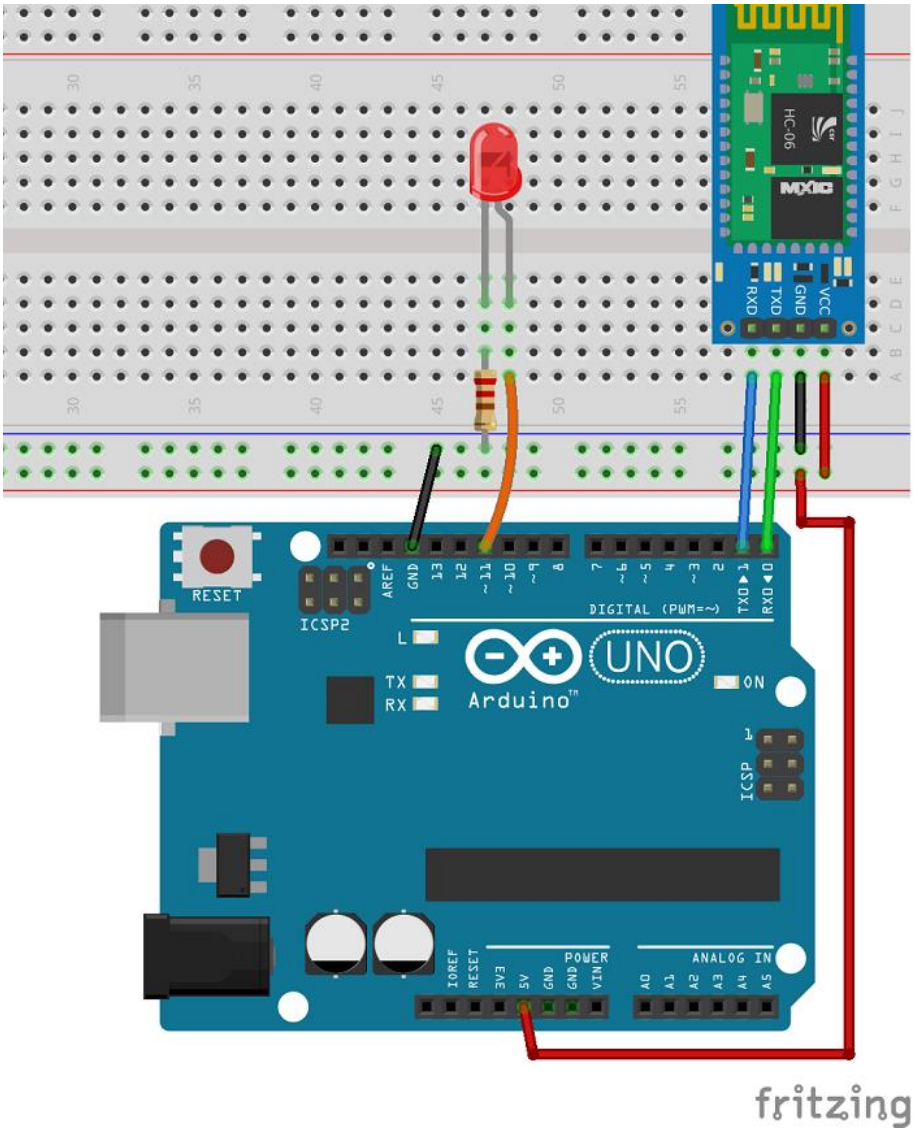
Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/22_Shake_Controlled_LED.ino

Note: before uploading the code, make sure you have the TX and RX pins disconnected from the bluetooth module!

Schematics

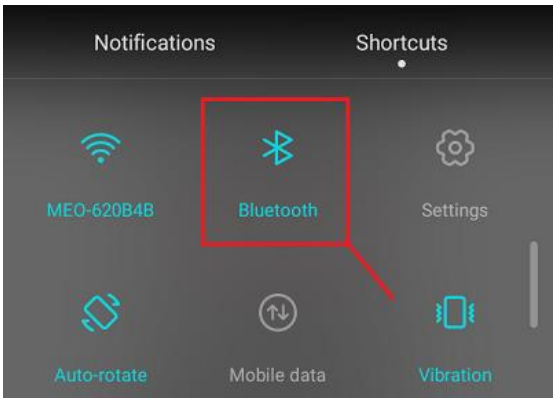
Here's the schematics for this project. It is very simple, you just need an LED and a bluetooth module.



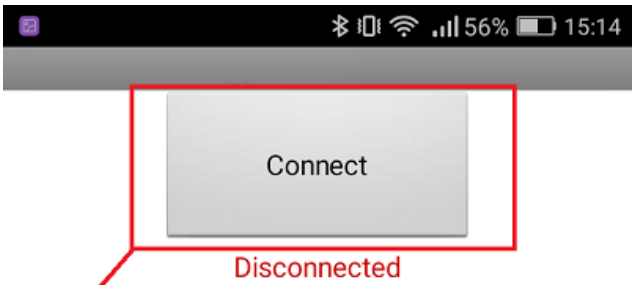
Launching your App

If you haven't generated the .apk file in a previous step, you can click here to download the .apk file (which is the Android App installation file). Move that file to your smartphone and open it. Follow the installation wizard to install the app.

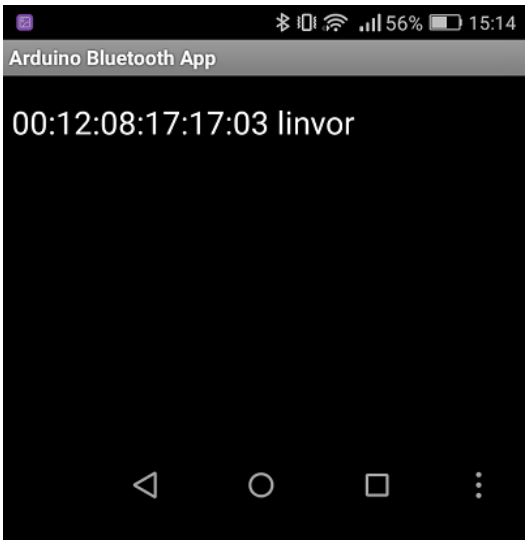
Turn on your smartphone's Bluetooth.



Tap on the newly installed app. Press the "Connect" button to connect your application to your Arduino Bluetooth module.



Select your Bluetooth module (it should be named linvor).

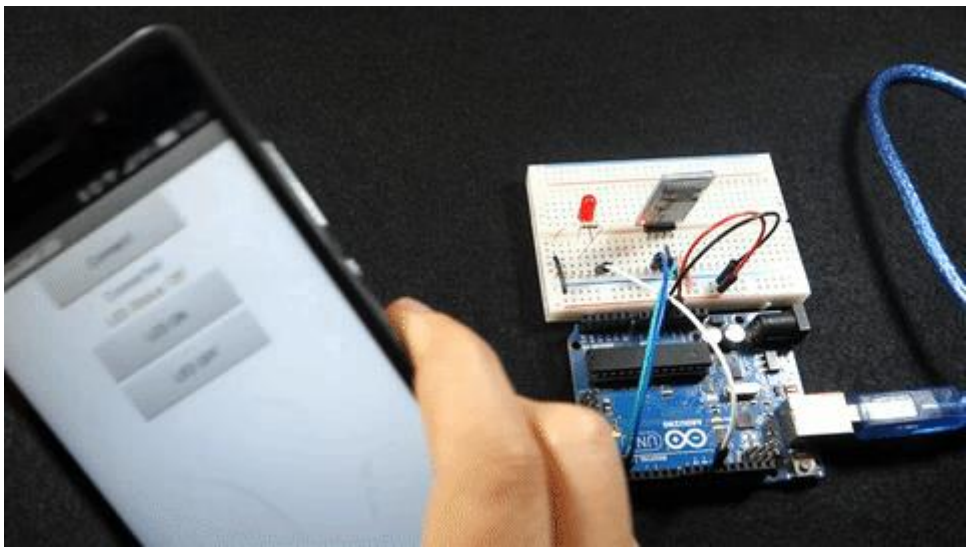


Now, it is ready to use!

Demonstration

Here's your app in action. When you shake your phone, the LED turns on.

If you shake your phone again, it goes off.



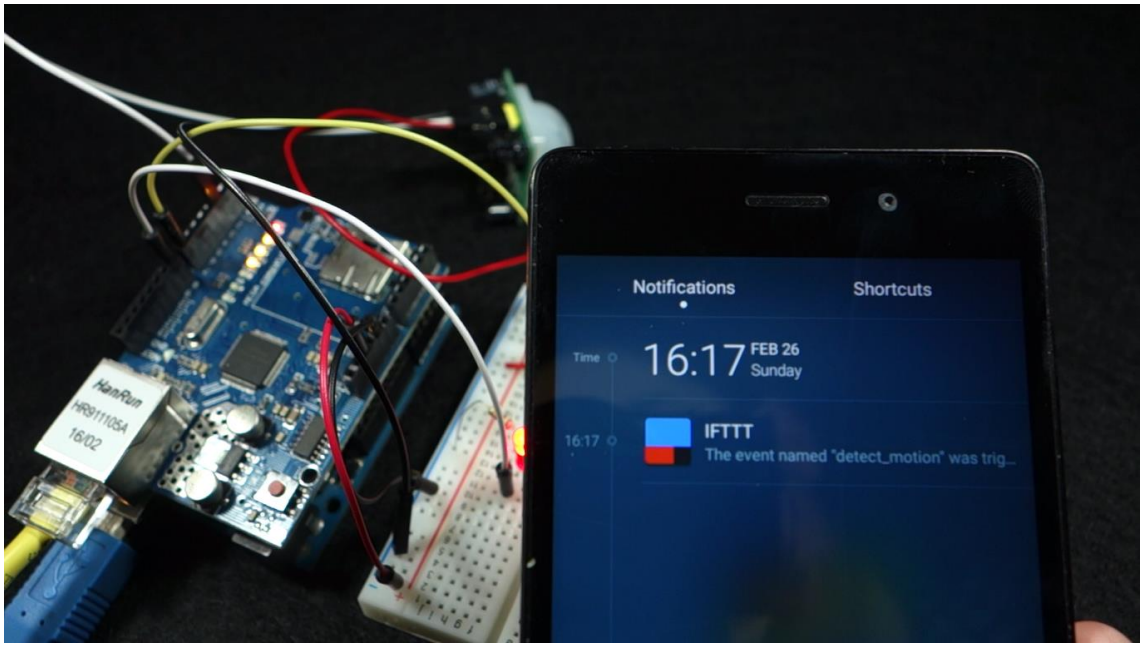
Impress your friends with this app! Have fun!

Wrapping up

In this project you learned how to control an LED by shaking your phone.

Now, the idea is to apply this feature to your own project!

Intruder Detector with Notifications



Intruder Detector with Notifications

Level: Intermediate – Time: 45 minutes

In this project you're going to create an intruder detector with notifications. You'll receive a notification in your smartphone when a PIR motion sensor detects movement.

You can use this project to detect whether someone enters in your office when you're out.

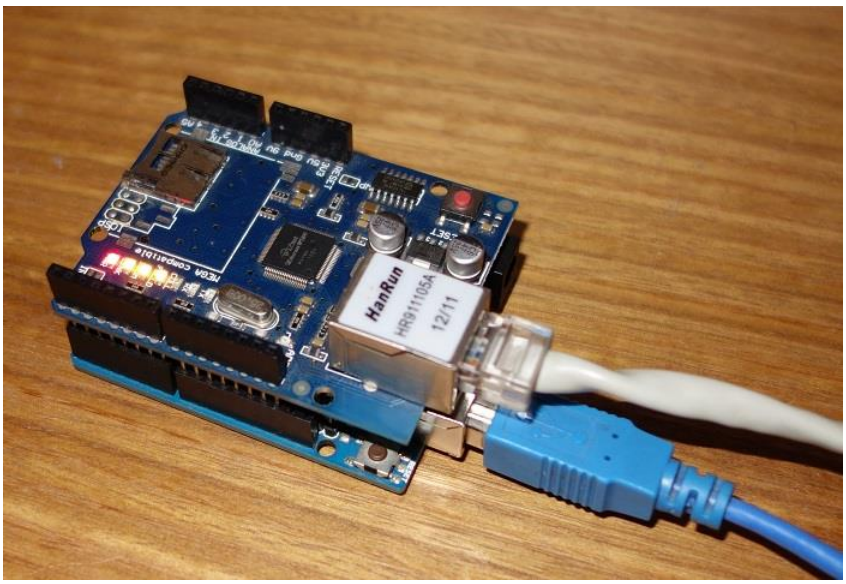
For this project:

- you need a PIR motion sensor to detect movement
- you need to use a service called IFTTT to send the notifications
- you need an Ethernet Shield to make web requests to the IFTTT service
- you'll have a button to activate and deactivate the sensor, plus an LED to show its current state

Ethernet shield

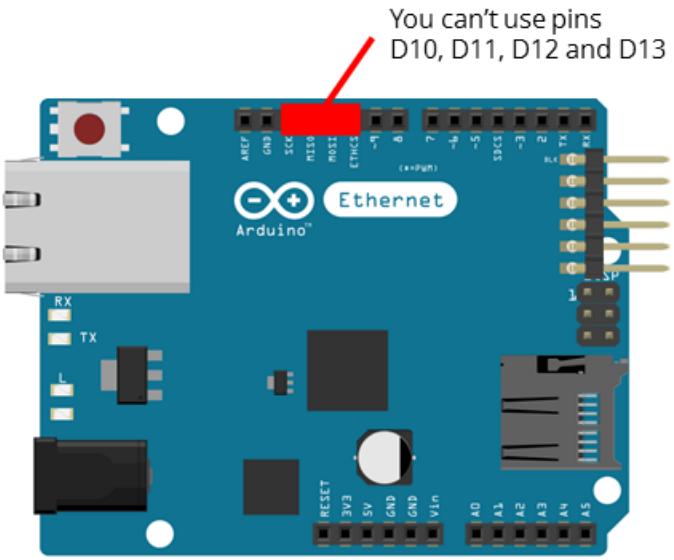
The Arduino Ethernet shield connects your Arduino to the internet in a simple way. Just mount this module onto your Arduino board, connect it to your network with an RJ45 cable and follow a few simple steps to start controlling your projects through the web.

Note: you must connect an Ethernet cable from your router to your Ethernet shield as shown in the following figure.



Pin usage

When the Arduino is connected to an Ethernet shield, you can't use Digital pins from 10 to 13, because they are being used in order to establish a communication between the Arduino and the Ethernet shield.



IFTTT

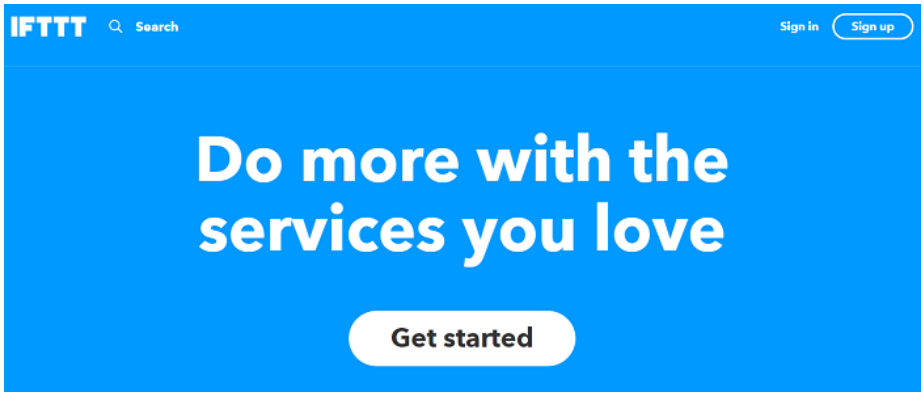
IFTTT is a platform that gives you creative control over dozens of products and apps.

You can make apps work together. For example when you send a request to IFTTT, it triggers an Applet that sends you an email alert, smartphone notifications and much more!

Creating your IFTTT account

Creating an account on IFTTT is free!

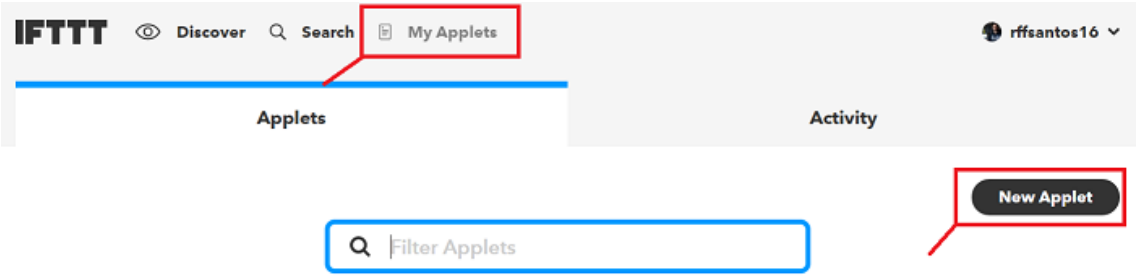
Go the official site: https://ifttt.com and click the "Get Started" button in the middle of the page.



Complete the form with your personal information and create your account.

Creating a new Applet

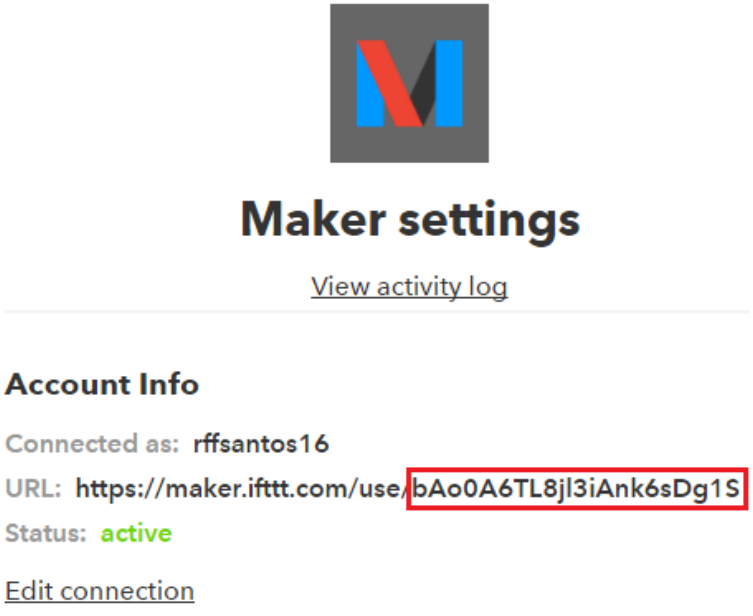
Create a new applet by selecting “My Applets” tab and then by clicking the “New Applet” button.



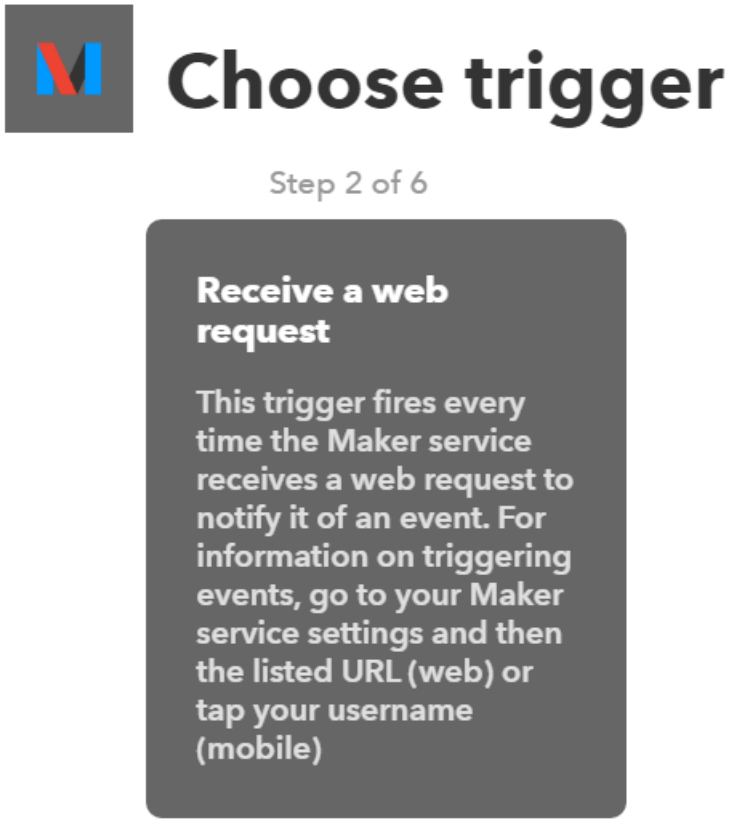
Then, you’ll be presented with the screen shown in the following figure. Click on “this” and search for the “Maker Channel”.

Establish a communication with the Maker Channel. You’ll also find your Maker Channel API Key, in my case: bAo0A6TL8jl3iAnk6sDg1S.

Copy the API Key to a safe place, because you’ll need it later in this project.

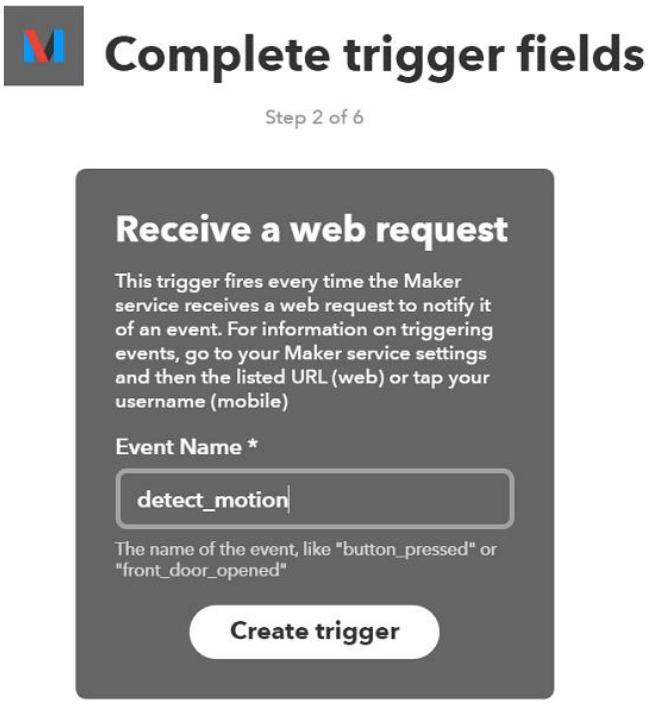


As a trigger, choose the “Receive a web request” option.



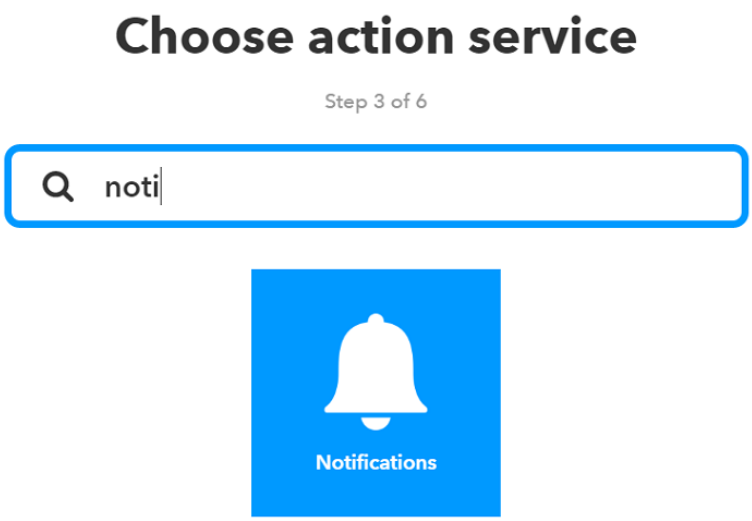
Set the Event Name field to `detect_motion` and press the “Create trigger” button.

Note: the Event Name field is case sensitive, so `detect_motion` is different from `Detect_Motion`.

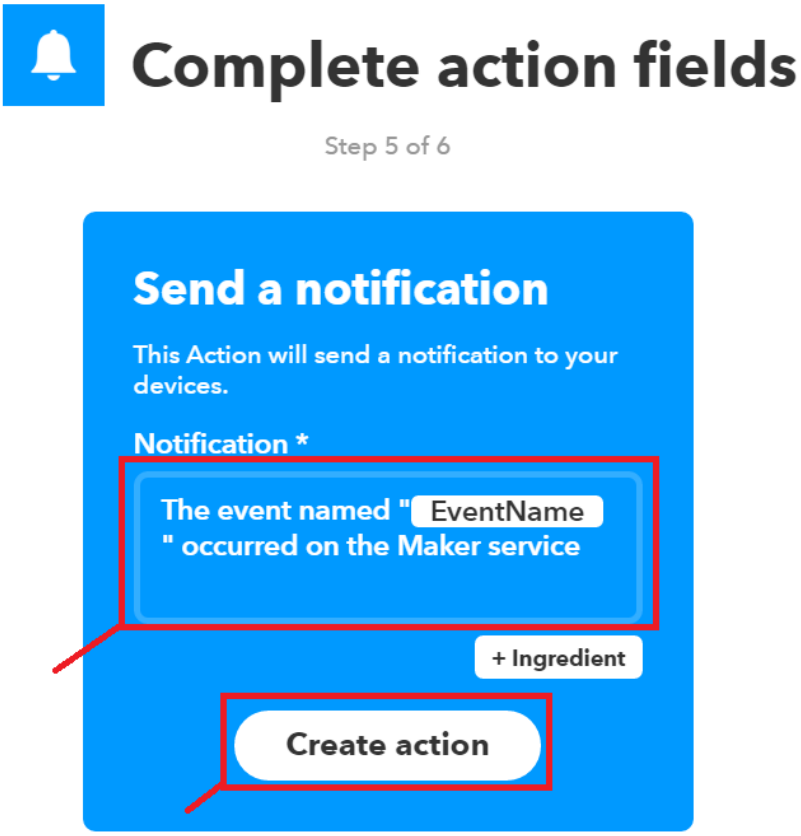


Then, press the “that” icon.

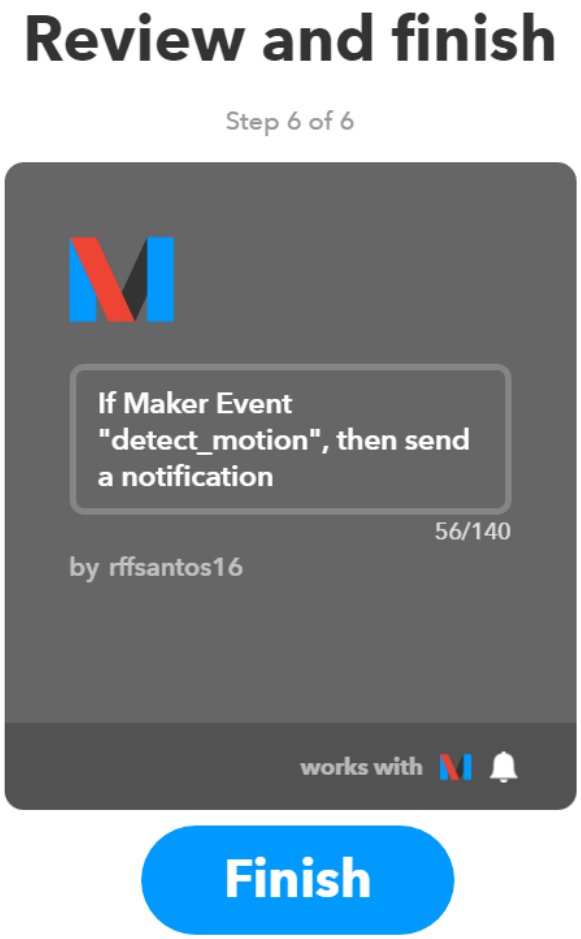
Search for “Notifications” and select the Notifications service:



You can leave the Notification field as it is or you can customize the message that you’ll receive. Press the “Create action” button.



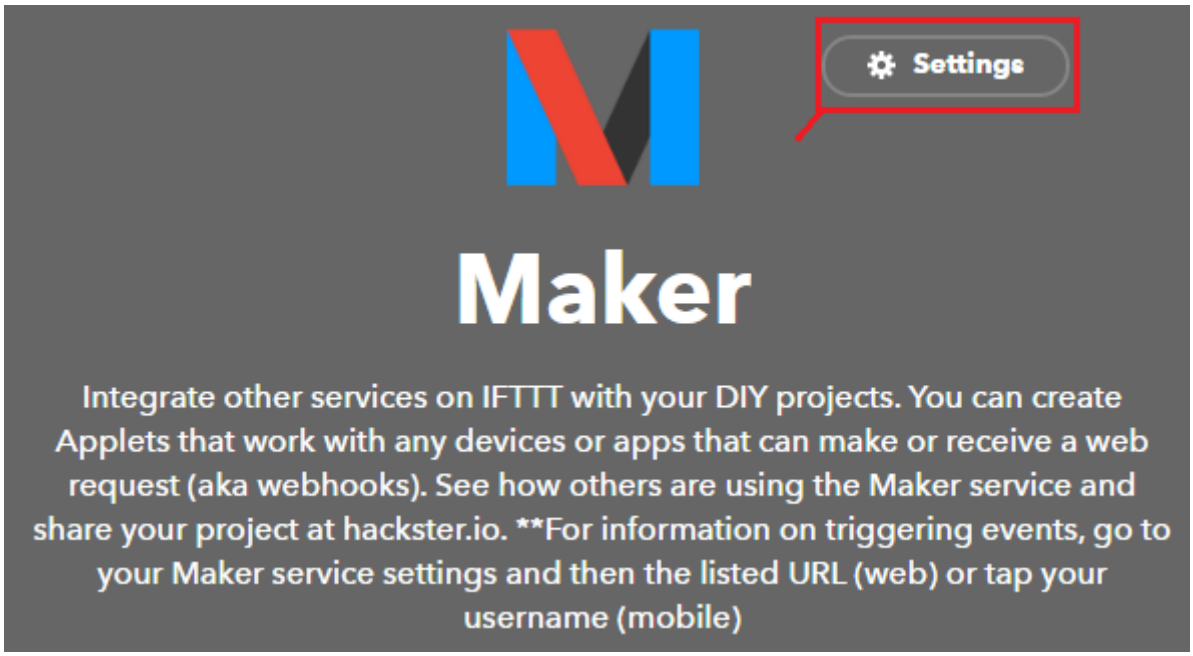
Finally, click the “Finish” button to complete the Applet creation.



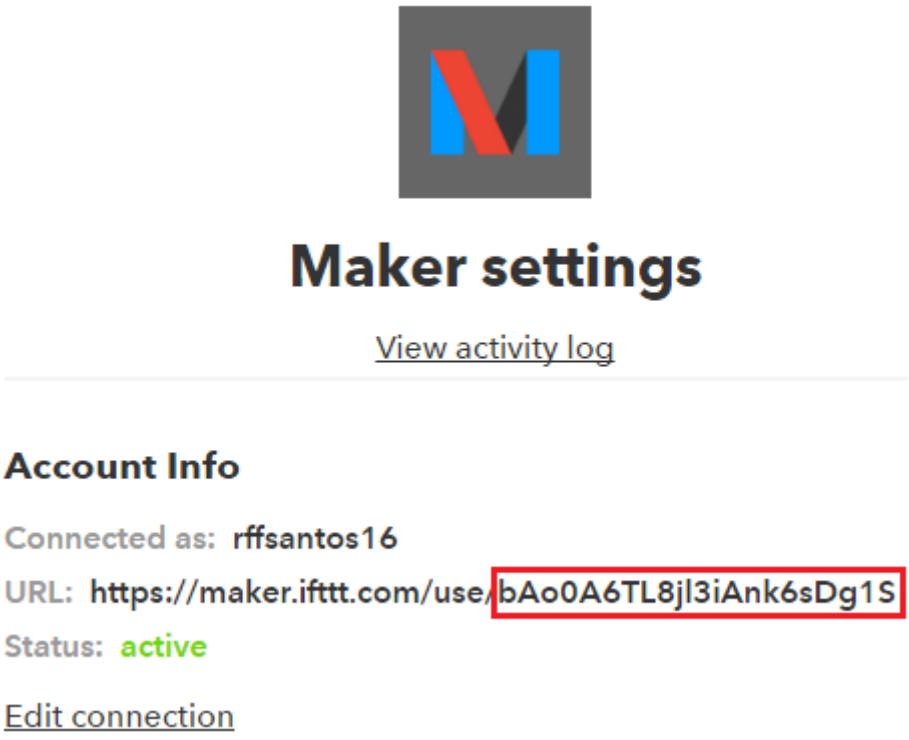
Make sure that your Applet is turned on:

Retrieve your Maker Channel API Key

In case you didn't find the API Key in the previous steps. While logged in, go to this link: https://ifttt.com/maker and select the Settings option:

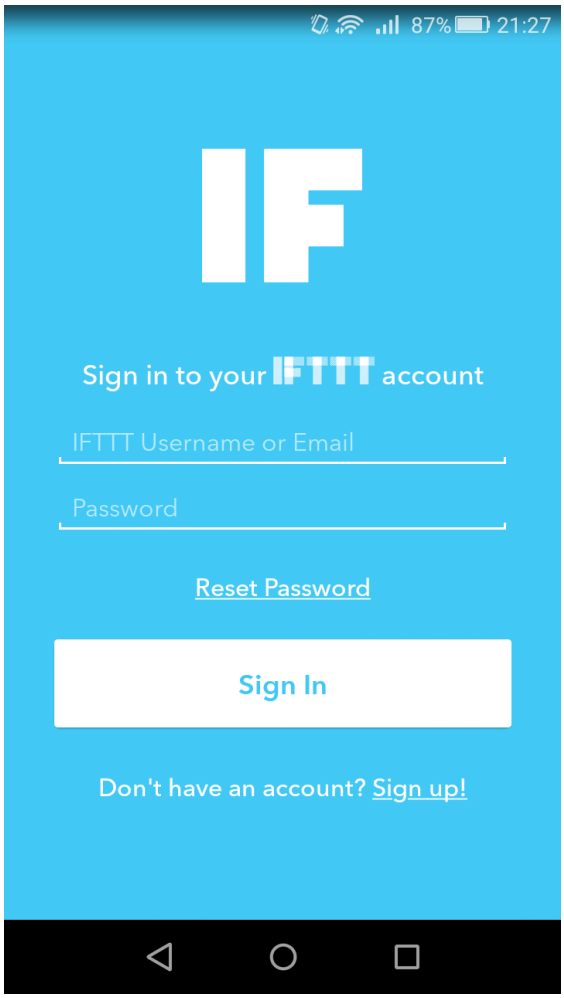


Copy your Maker Channel API Key to a safe place, in my case: **bAo0A6TL8jl3iAnk6sDg1S**.



Installing the IFTTT mobile app

Go to the Google Play Store or App Store, install the IFTTT app in your smartphone and sign in with your account.



Testing your Applet

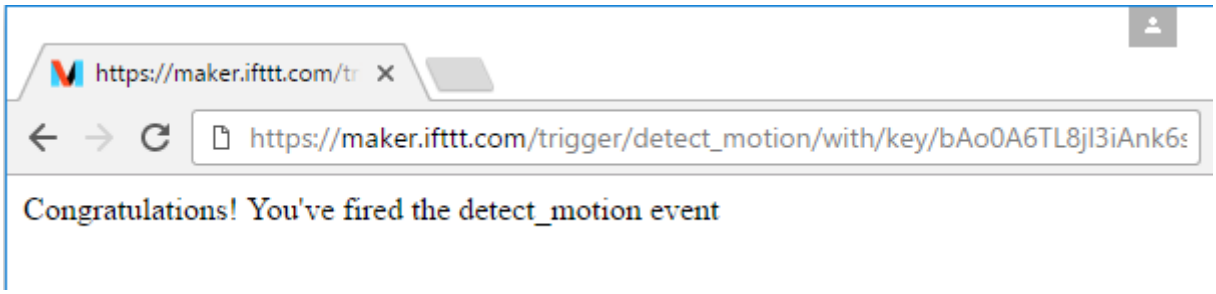
Let's test if your request is working properly. Replace YOUR_API_KEY from the following URL:

https://maker.ifttt.com/trigger/detect_motion/with/key/YOUR_API_KEY

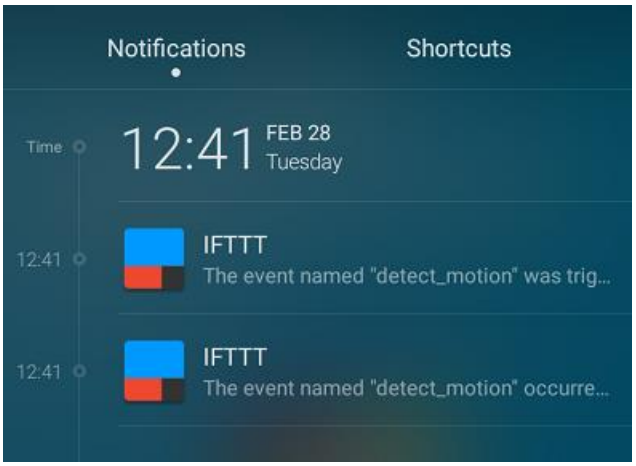
With your API KEY:

https://maker.ifttt.com/trigger/detect_motion/with/key/bAo0A6TL8jl3iAnk6sDg1S

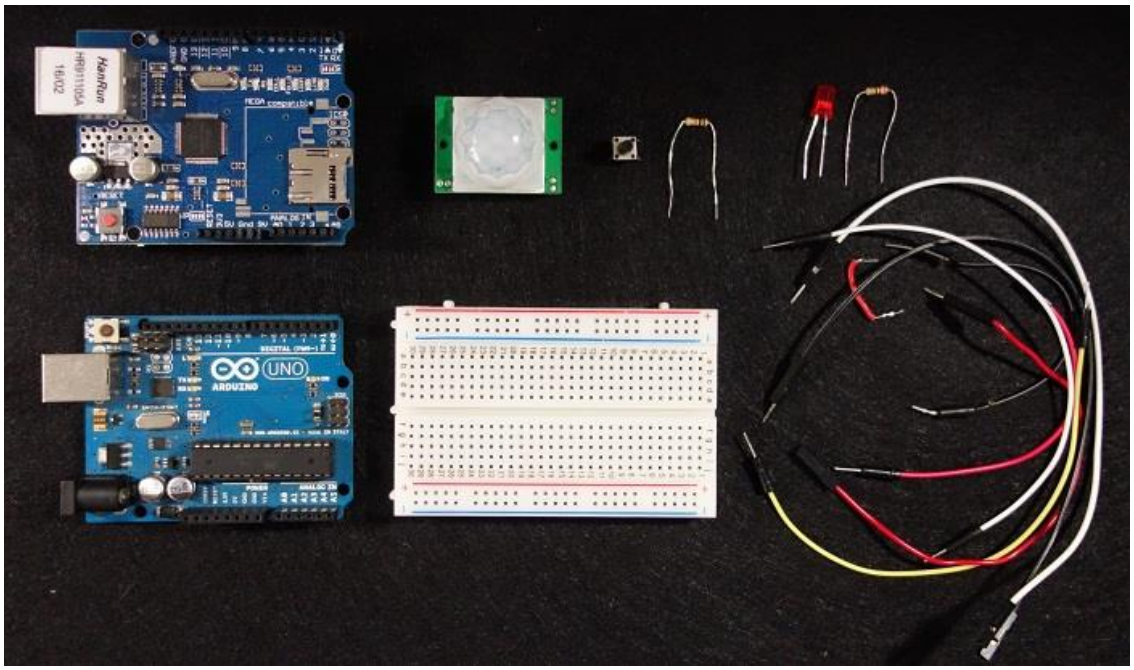
Open your URL with your API KEY in your browser.



You should see something similar to the preceding figure. Then, go to your smartphone and a new notification should be there.



Parts required



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Arduino Uno	http://ebay.to/1SQda0R
	Ethernet Shield (WIZnet W5100)	http://ebay.to/1WWpuhv
	PIR Motion Sensor	http://ebay.to/1Wqh9jL
	1x LED	http://ebay.to/20H2Oyy
	220Ohm Resistor	http://ebay.to/1KsMYFP
	1x pushbutton	http://ebay.to/211vcRv
	10kOhm Resistor	http://ebay.to/1KsMYFP
	Breadboard	http://ebay.to/21bEojM
	Jumper Wires	http://ebay.to/1PXeaJz

Code

Copy the following code to your Arduino IDE. Before uploading the code to your Arduino board make sure you read these two extra steps described below:

A) Configuring your network

B) Adding your IFTTT Maker Channel API Key

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
*/
```

```
#include <SPI.h>
#include <Ethernet.h>

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1, 111);

// REPLACE THE NEXT VARIABLE WITH YOUR IFTTT MAKER CHANNEL API KEY
String iftttApiKey = "YOUR_MAKER_CHANNEL_API_KEY";

// name address for IFTTT
char server[] = "maker.ifttt.com";

// Initialize the Ethernet client library
// with the IP address and port of the server
// that you want to connect to (port 80 is default for HTTP):
EthernetClient client;

// the number of the pushbutton pin
const int button = 7;
int lastButtonState = LOW;   // the previous reading from the input
pin

// the number of the LED pin
const int led = 8;

// PIR Motion Sensor is connected to D2.
int PIRInterrupt = 2;
int PIRArmed = LOW;

// Timer Variables
long lastDebounceTime = 0;
long debounceDelay = 10000;
long lastButtonDebounceTime = 0;
long debounceButtonDelay = 50;

void setup() {
  // button pin configuration
  pinMode(button, INPUT);
  // led pin configuration
  pinMode(led, OUTPUT);
  // PIR motion sensor set as an input
  pinMode(PIRInterrupt, INPUT);
  // Triggers detectMotion function on rising mode to turn the
relay on, if the condition is met
  attachInterrupt(digitalPinToInterrupt(PIRInterrupt),
detectMotion, RISING);
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB
port only
  }

  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0) {
```

```
    Serial.println("Failed to configure Ethernet using DHCP");
    // try to congifure using IP address instead of DHCP:
    Ethernet.begin(mac, ip);
  }
  // give the Ethernet shield a second to initialize:
  delay(1000);

}

void loop() {
   // read the state of the switch into a variable
  int reading = digitalRead(button);

  if ((millis() - lastButtonDebounceTime) > debounceButtonDelay) {
    // whatever the reading is at, it's been there for longer
    // than the debounce delay, so take it as the actual current
state:

    // if the button state has changed:
    if (reading != lastButtonState) {
      lastButtonState = reading;
      if(lastButtonState == HIGH) {
        PIRArmed = !PIRArmed;
      }
      digitalWrite(led, PIRArmed);
    }
    lastButtonDebounceTime = millis();
  }
}


void detectMotion() {
  Serial.println("Motion detected");
  if((millis() - lastDebounceTime) > debounceDelay && PIRArmed){
    // stop previous client connection, if still running
    client.stop();
    if (client.connect(server, 80)) {
      Serial.println("connected");
      // Make a HTTP request:
      client.print("GET /trigger/detect_motion/with/key/");
      client.print(iftttApiKey);
      client.println(" HTTP/1.1");
      client.println("Host: maker.ifttt.com");
      client.println("Connection: close");
      client.println();
    }
    else {
      // if you didn't get a connection to the server:
      Serial.println("connection failed");
    }
    Serial.println("Notification sent");
    lastDebounceTime = millis();
  }
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/23_Intruder_Detector_with_Notifications.ino

A) Configuring your network

Take a look at the configuring your network code snippet:

```
byte mac[] = {
0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,XXX);
```

Important: you actually might need to replace that variable highlighted in red with appropriate values that are suitable for your network, otherwise your Arduino will not establish a connection with your network.

Replace the following line with an IP that is available and suitable for your network:

```
IPAddress ip(X, X, X, X);
```

In my case, my IP range is **192.168.1.X** and with the software Angry IP Scanner I know that the IP **192.168.1.111** is available in my network, because it doesn't have any active device in my network with that exact same IP address:

```
IPAddress ip(192, 168, 1, 111);
```

B) Adding your IFTTT Maker Channel API Key

Replace line 15 with your Maker Channel API Key:

```
String iftttApiKey = "YOUR_MAKER_CHANNEL_API_KEY";
```

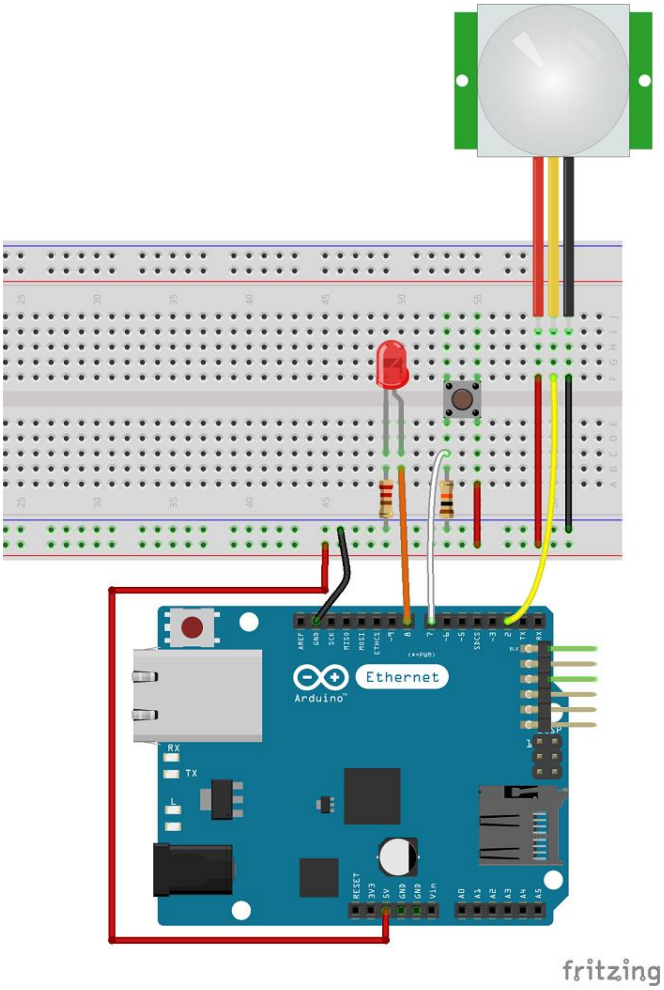
In my case, it looks like this:

```
String iftttApiKey = "bAo0A6TL8jl3iAnk6sDg1S";
```

Now, you can upload the code to your Arduino board.

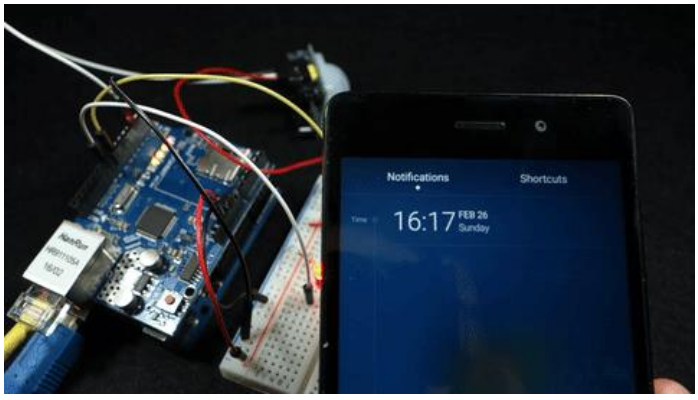
Schematics

Here's the schematics for this project.



Demonstration

Here's your project in action. When it detects movement, your smartphone receives a notification.

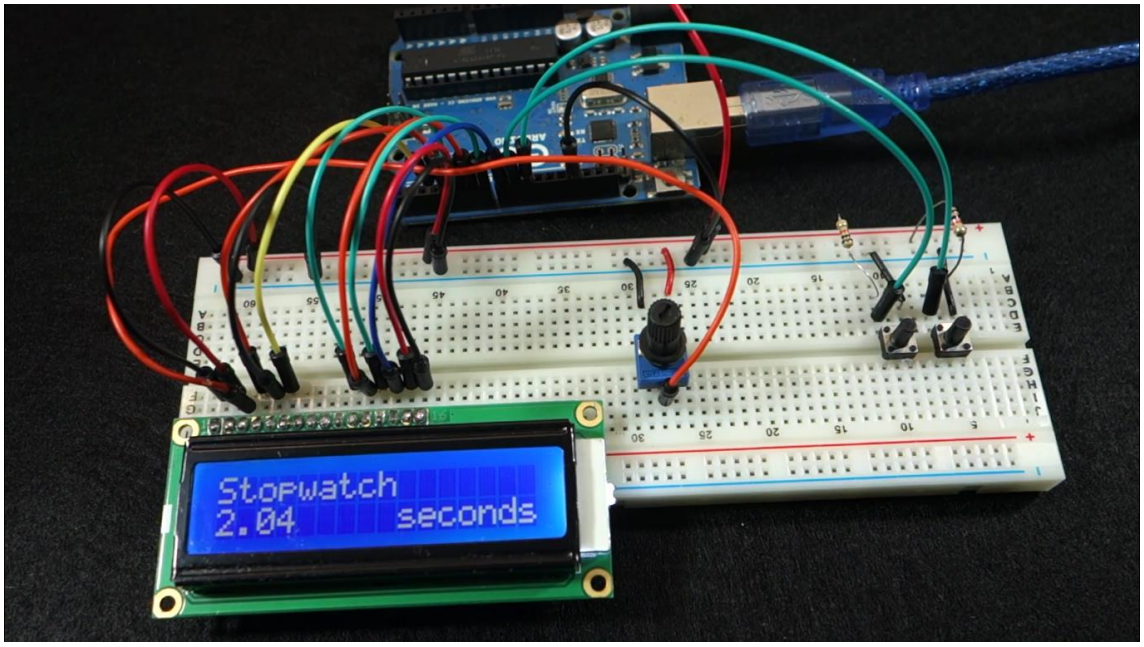


Wrapping up

In this project you learned how to send notifications to your smartphone.

Now, you could replace the PIR motion sensor with a smoke sensor or a temperature sensor.

Arduino Stopwatch



Arduino Stopwatch

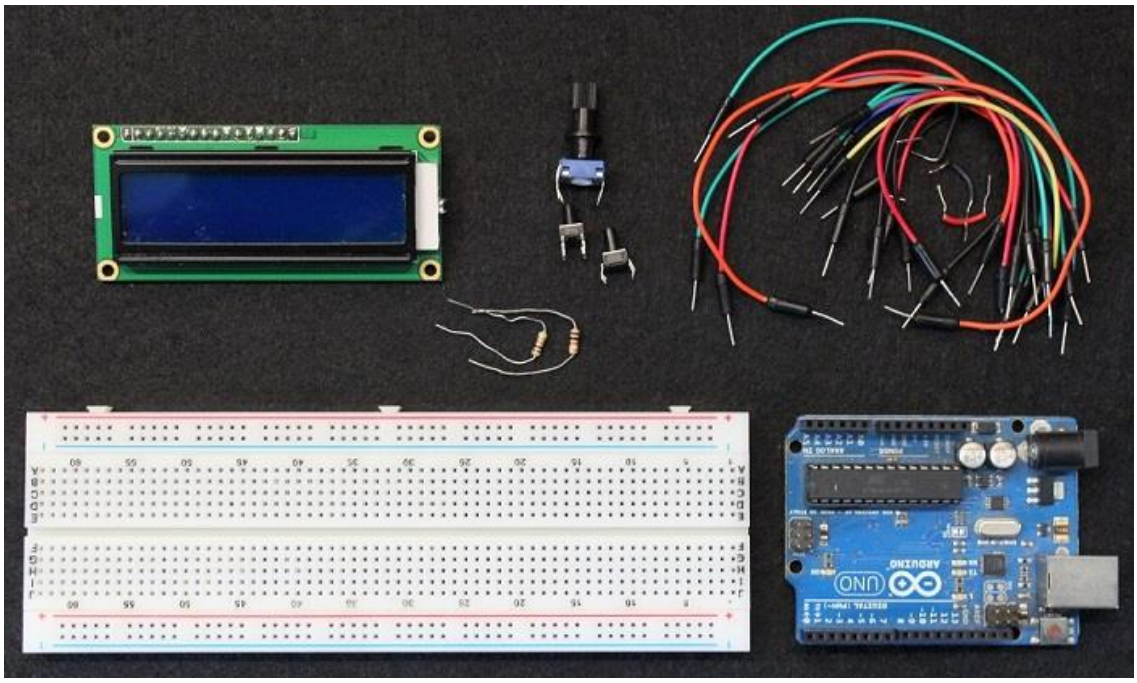
Level: Beginner – Time: 25 minutes

In this project you're going to build an Arduino stopwatch. This is a very simple – you just need a few components – yet very useful project.

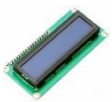
The time is counted using the `millis()` Arduino function and it is displayed on a 16×2 LCD.

You need two pushbuttons: one to start the counter and other to stop it.

Parts required



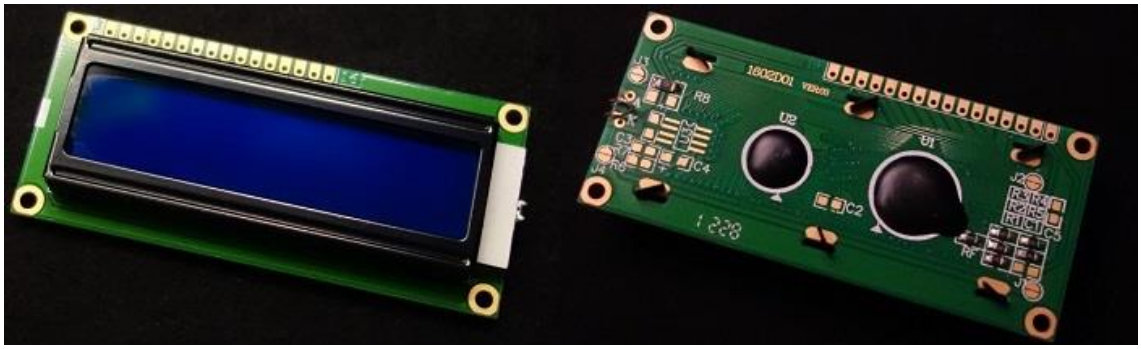
Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Arduino UNO	http://ebay.to/1SQda0R
	16x2 LCD display (Hitachi HD44780 driver)	http://ebay.to/2cgnKeW

	2x 10kOhm Resistor	http://ebay.to/1KsMYFP
	2x Pushbuttons	http://ebay.to/211vcRv
	1x 10kOhm Potentiometer	http://ebay.to/1PUefOb
	Jumper Cables	http://ebay.to/1PXeaJz
	Breadboard	http://ebay.to/21bEojM

16×2 Liquid Crystal Display (LCD)

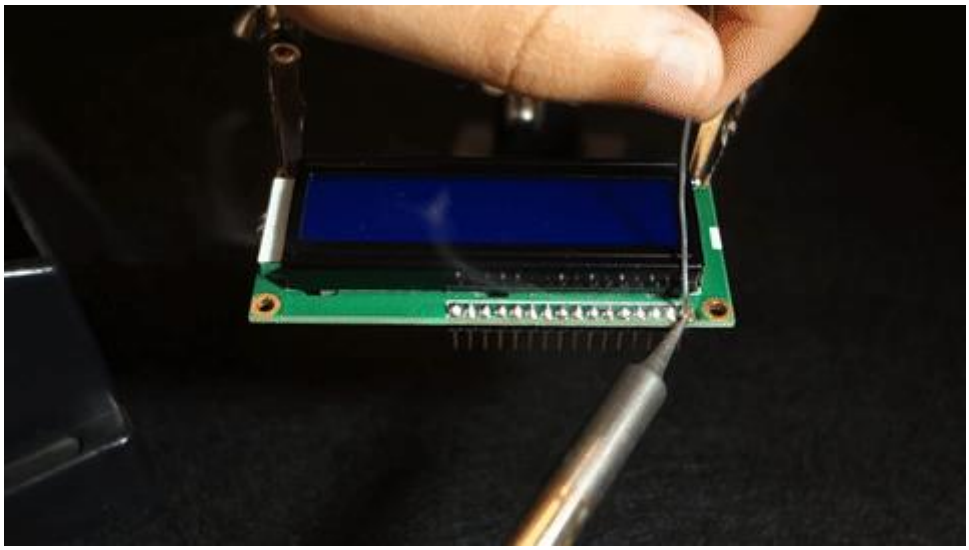
The 16×2 LCD is the one in the following figure (front and the back view).



The 16×2 LCD is commonly used in several circuits and devices. This LCD has 2 rows, and each row can display 16 characters. It also has LED backlight that can be adjusted accordingly to the contrast between the background and the characters that you want.

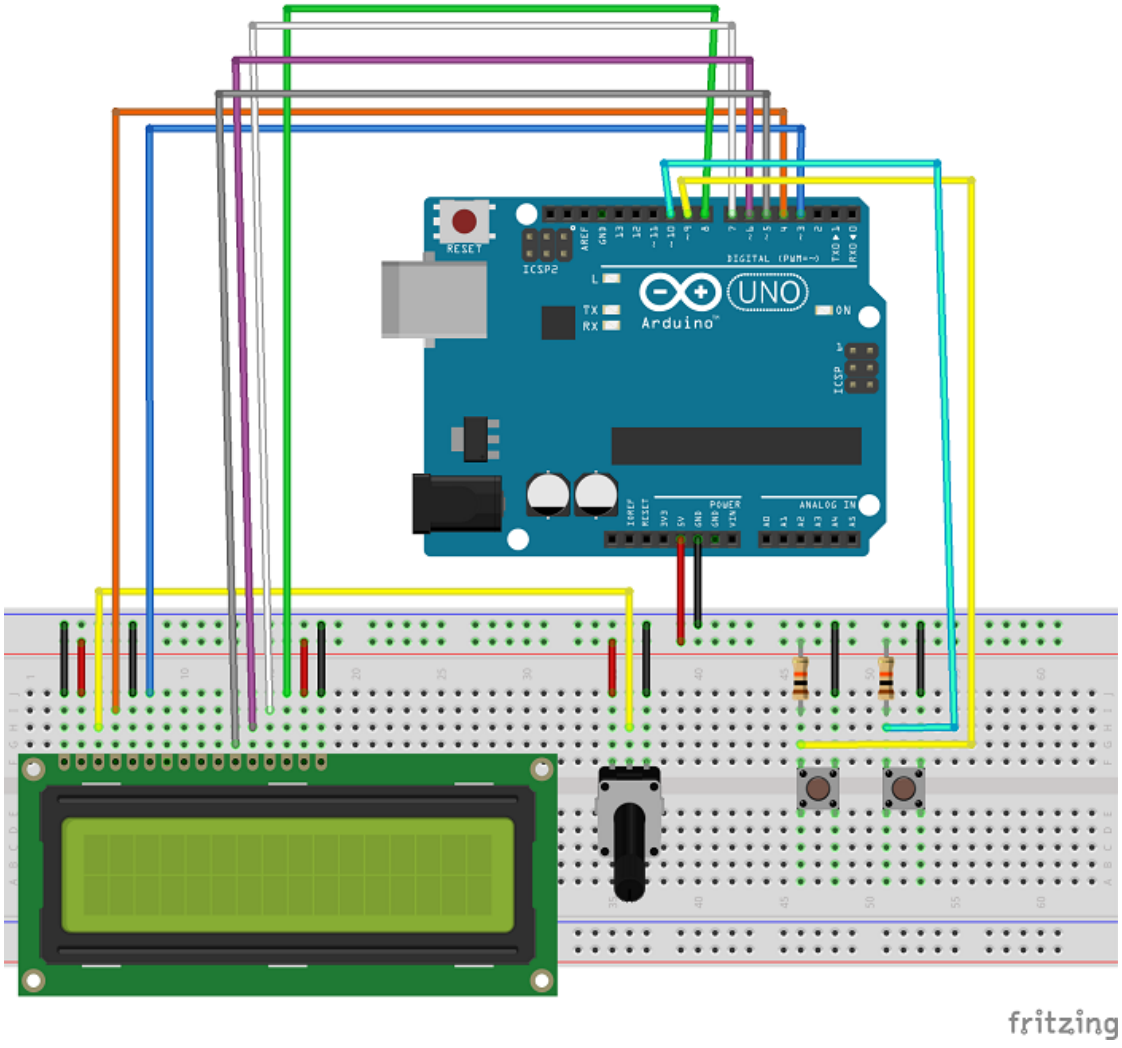
When you buy a 16×2 LCD, usually it doesn't come with breadboard friendly pins. So, you may need some headers.

Solder the headers to you LCD, and it is ready to use.



Schematics

Assemble all the parts by following the schematics below:



Code

To use the LCD with the Arduino, you don't need to install any library. The `LiquidCrystal.h` is installed in the Arduino IDE by default.

Download or copy the following code to your Arduino IDE, and upload it to your Arduino board. Make sure that you have the right Board and COM port selected.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

#include <LiquidCrystal.h>

// Preparing LCD connections
LiquidCrystal lcd(4, 3, 5, 6, 7, 8);

// Start and pause buttons
int start = 9;
int pause = 10;

// Variables to store start time, current time and time passed in
seconds
double startTime = millis();
double currentTime = 0;
double timeInSeconds = 0;

void setup(){
  // Starting a 16x2 character LCD
  lcd.begin(16, 2);
  lcd.clear();

  // Welcome message
  lcd.setCursor(0, 0);
  lcd.print("Stopwatch");
  lcd.setCursor(0, 1);
  lcd.print("Press play!");

  //Serial.begin(9600);

  // Preparing buttons as INPUTs
  pinMode(start, INPUT);
  pinMode(pause, INPUT);
}

void loop(){
  // When start button is pressed, it starts counting time
  if (digitalRead(start) == LOW){
    lcd.clear();
    startTime = millis();
    // While pause button is not pressed it keeps counting
    while (digitalRead(pause) == HIGH){
      currentTime = millis();
      timeInSeconds = (currentTime - startTime) / 1000;
      lcd.setCursor(0, 0);
```

```
      lcd.print("Stopwatch");
      lcd.setCursor(0, 1);
      lcd.print(timeInSeconds);
      lcd.setCursor(9, 1);
      lcd.print("seconds");
      // Serial.println(timeInSeconds);
      delay(100);
    }
    // When pause button is pressed, it pauses timer
    if (digitalRead(pause) == LOW){
      lcd.setCursor(0, 0);
      lcd.print("Stopwatch");
      lcd.setCursor(0, 1);
      lcd.print(timeInSeconds);
      lcd.setCursor(9, 1);
      lcd.print("seconds");
      delay(100);
    }
  }
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/20_Arduino_stopwatch.ino

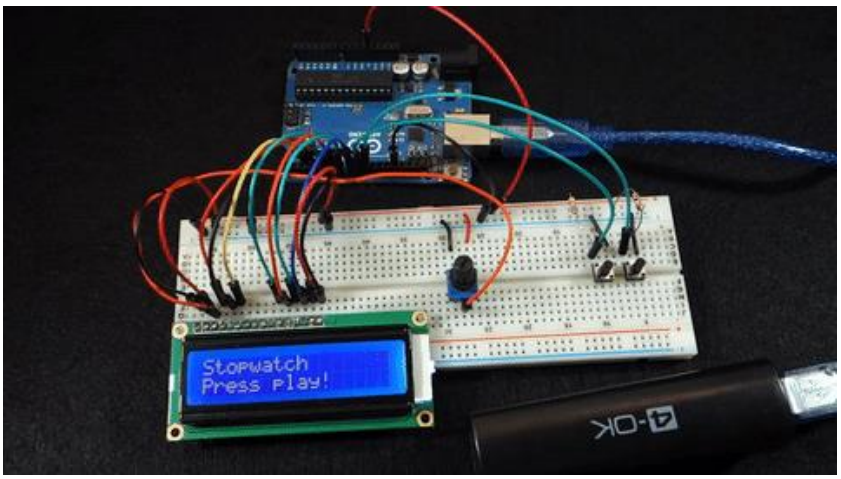
millis() function

The `millis()` function is essential in this code. In this example, you use it to count the time since the start button is pressed until the stop button is pressed.

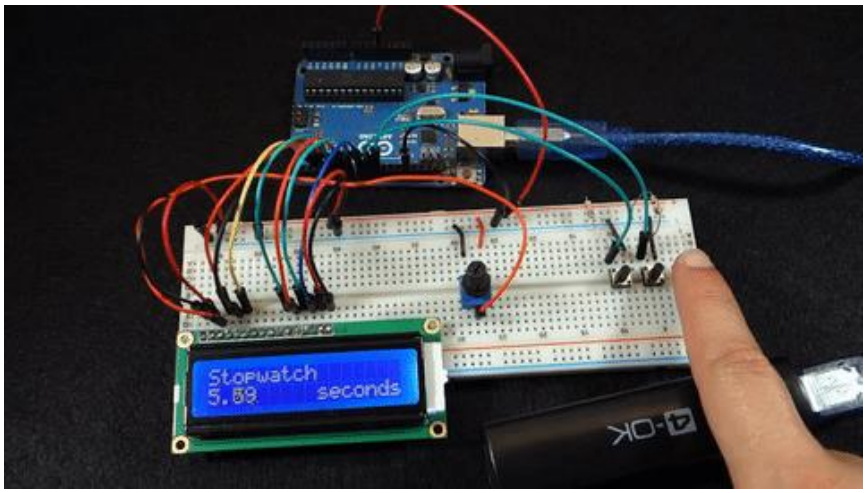
The `millis()` function returns the number of milliseconds since the program started. To know how much time has passed since you pressed the start button, you subtract the `startTime` to the `currentTime`.

Demonstration

When you press the start button, the time starts counting.



To stop the time, you just need to press the stop button.

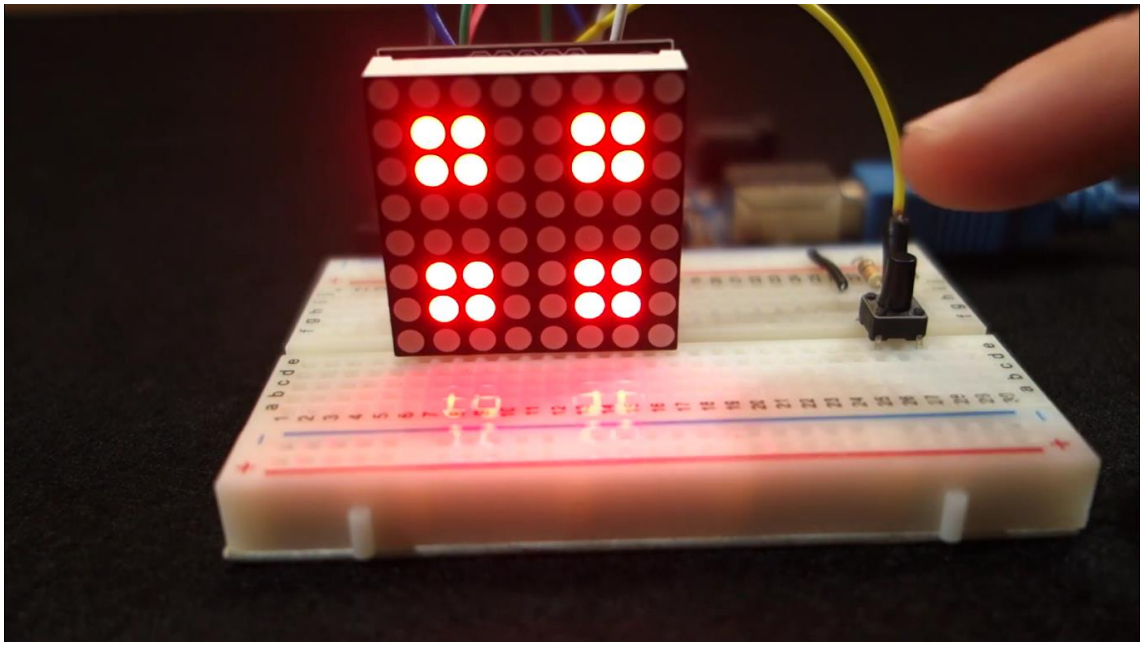


Wrapping up

The stopwatch is a good project to learn about the `millis()` function and the 16×2 LCD.

You can build a more complex stopwatch by adding a pushbutton that when pressed saves laps time.

Electronic Dice Roller



Electronic Dice Roller

Level: Beginner – Time: 25 minutes

In this project you're going to create an electronic dice roller using a dot matrix and a pushbutton.

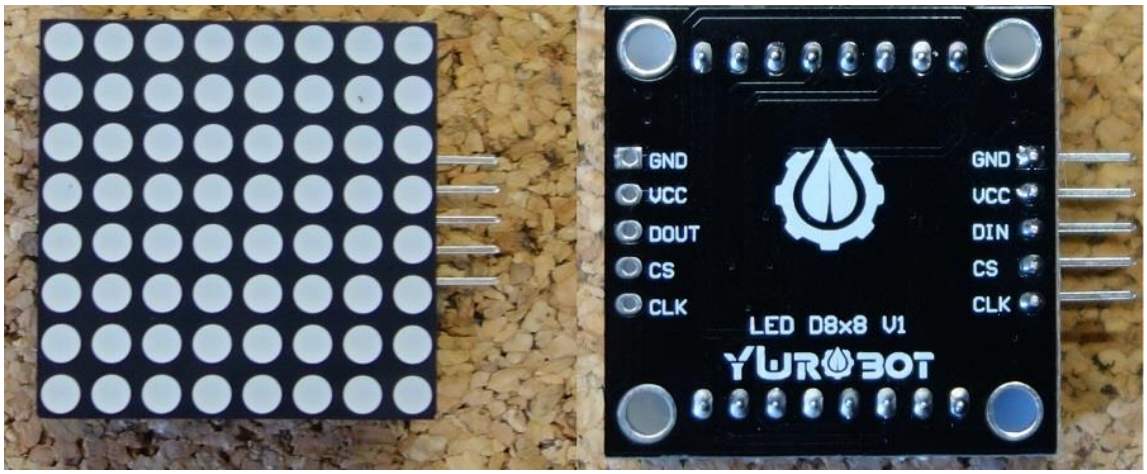
Every time you press the pushbutton a random number between 1 and 6 is generated and the corresponding dice face is displayed on the dot matrix.

8×8 Dot Matrix MAX7219

The dot matrix that we're going to use in this project is an 8×8 matrix which means that it has 8 columns and 8 rows, containing a total of 64 LEDs.

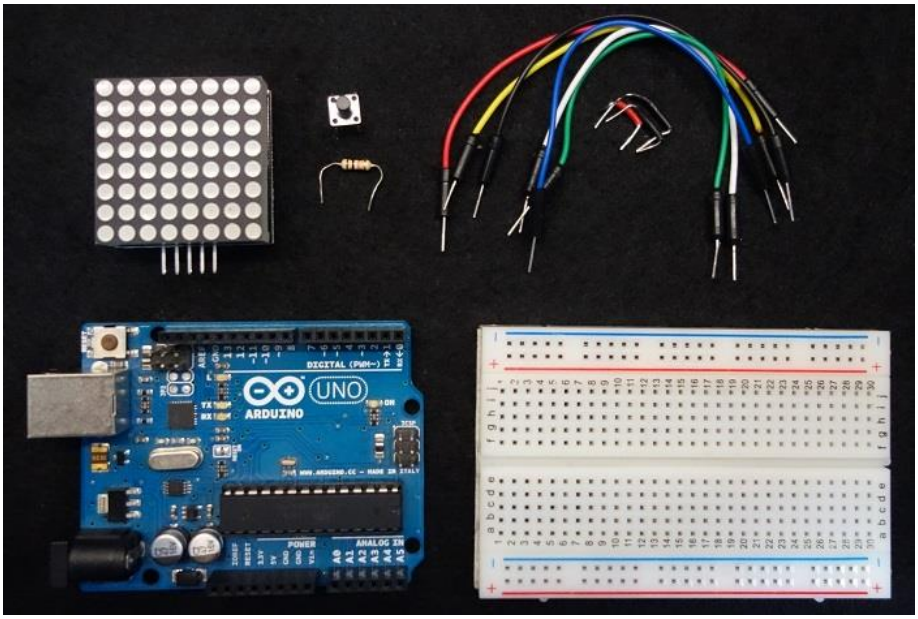
This matrix module comes with the MAX7219 microcontroller. This microcontroller makes it easy to control the dot matrix by using just 3 Arduino digital pins.

The figure below shows the dot matrix that you're going to use (front and back view).



You can control multiple dot matrixes at a time with the same Arduino digital pins, because these displays are controlled with SPI communication and can be connected in series.

Parts required



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Breadboard	http://ebay.to/21bEojM
	Arduino UNO	http://ebay.to/1SQda0R
	1 x 8x8 Dot Matrix MAX7219	http://ebay.to/29VvS3H
	1x pushbutton	http://ebay.to/211vcRv
	1x 10kOhm Resistor	http://ebay.to/1KsMYFP
	Jumper Cables	http://ebay.to/1PXeaJz

Note: the dot matrix must have the MAX7219 microcontroller.

Installing the LedControl library

You need to download and install the LedControl library. To install the library follow these next steps:

- Click here to download the LedControl library. You should have a .zip folder in your Downloads folder
- Unzip the .zip folder and you should get LedControl-master folder
- Rename your folder from ~~LedControl-master~~ to LedControl
- Move the LedControl folder to your Arduino IDE installation libraries folder
- Finally, re-open your Arduino IDE

Using the LedControl library functions

The easiest way to display something on the dot matrix is by using the functions `setLed()`, `setRow()` or `setColumn()`. These functions allow you to control one single LED, one row or one column at a time, respectively.

Here's the parameters for each function:

```
setLed(addr, row, col, state)
```

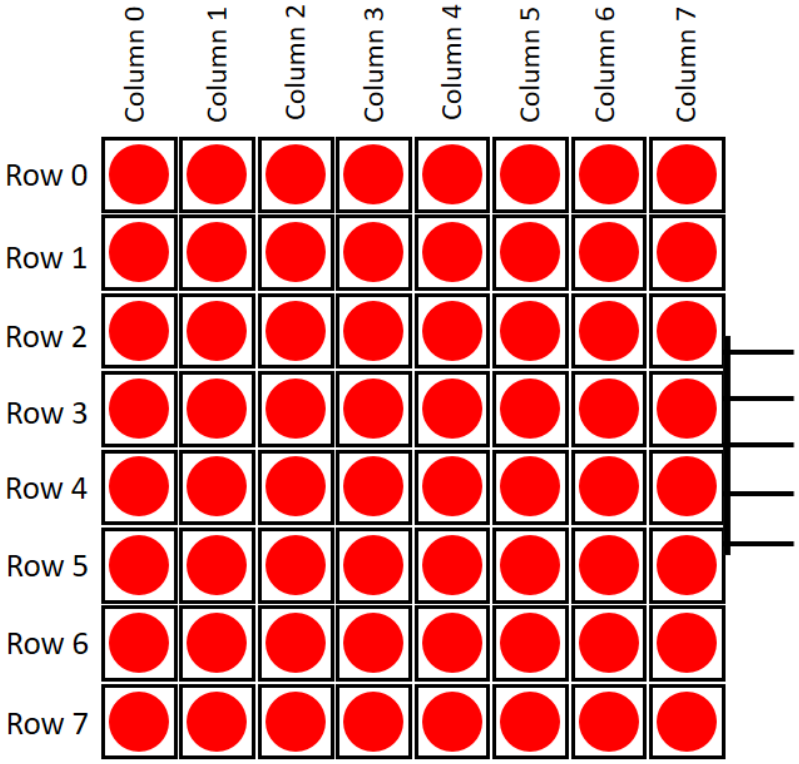
- `addr` is the address of your matrix, for example, if you have just 1 matrix, the int `addr` is 0
- `row` is the row where the LED is located
- `col` is the column where the LED is located
- `state`
    - It's true or 1 if you want to turn the LED on
    - It's false or 0 if you want to switch it off

```
setRow(addr, row, value)
```

```
setCol(addr, column, value)
```

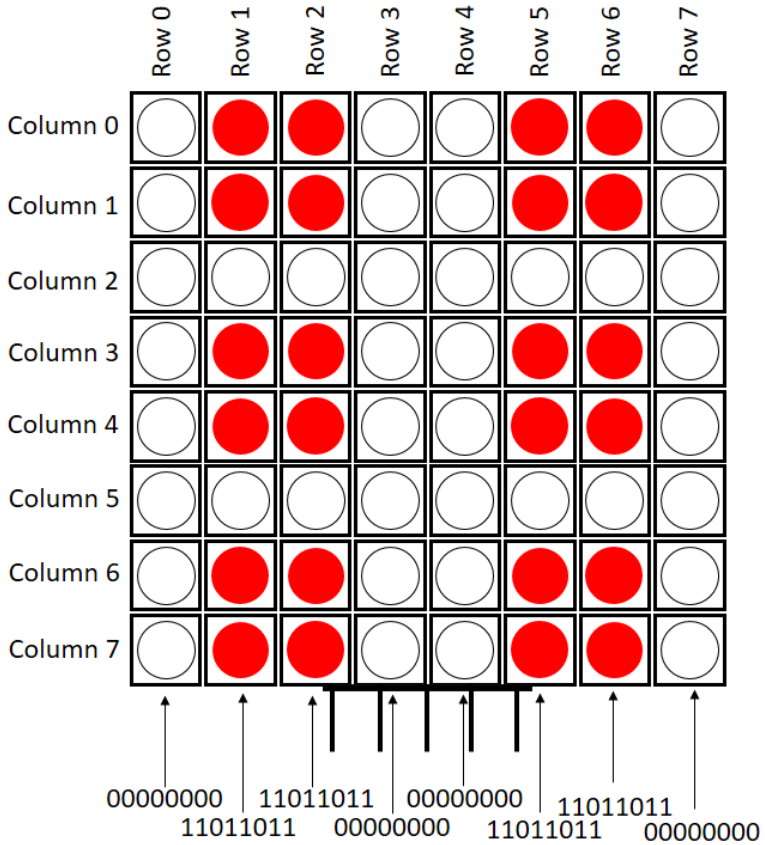
Index

As previously stated, this matrix has 8 columns and 8 rows. Each one is indexed from 0 to 7. Here's a figure for better understanding:



The easiest way to display something on the dot matrix is by creating a binary sequence for each row that represents the LEDs that are on and off.

For example, if you want to display the dice face corresponding to number six, you have the following:

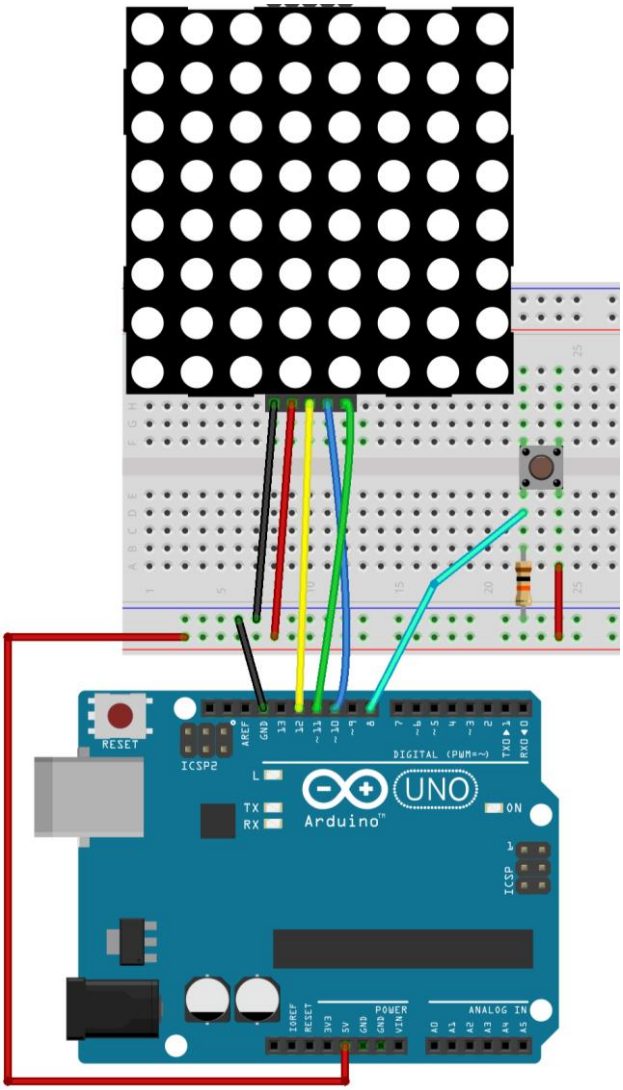


Here's the binary variable:

```
byte six[]={
B00000000,
B11011011,
B11011011,
B00000000,
B00000000,
B11011011,
B11011011,
B00000000};
```

Schematics

Assemble all the parts by following the schematics below:



Code

Upload the following code to your Arduino board. Make sure you have the right Board and COM Port selected.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

#include "LedControl.h"
#include "binary.h"

/*
 * Define pins for the LED matrix
 * DIN connects to pin 12
 * CLK connects to pin 11
 * CS connects to pin 10
 */
LedControl lc=LedControl(12,11,10,1);

//defining binary patterns for each number
byte zero[] = {
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B00000000};

byte one[] = {
B00000000,
B00000000,
B00000000,
B00011000,
B00011000,
B00000000,
B00000000,
B00000000};

byte two[] ={
B00000000,
B01100000,
B01100000,
B00000000,
B00000000,
B00000110,
B00000110,
B00000000};

byte three[]={
B11000000,
B11000000,
B00000000,
B00011000,
B00011000,
B00000000,
```

```
B00000011,
B00000011};

byte four[]={
B00000000,
B01100110,
B01100110,
B00000000,
B00000000,
B01100110,
B01100110,
B00000000};

byte five[]={
B11000011,
B11000011,
B00000000,
B00011000,
B00011000,
B00000000,
B11000011,
B11000011};

byte six[]={
B00000000,
B11011011,
B11011011,
B00000000,
B00000000,
B11011011,
B11011011,
B00000000};

int buttonPin = 8;

int result = 0;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
  lc.shutdown(0,false);
  // Set brightness to a medium value
  lc.setIntensity(0,8);
  // Clear the display
  lc.clearDisplay(0);
}

void loop() {
  if (digitalRead(buttonPin)){
    rollDice();
  }
  delay(50);
}

void rollDice(){
  int result = 0;
  int timeDiceRolls = random(10, 15);
  for(int i=0; i<timeDiceRolls; i++){
    // result variable will be 1 to 6
    result = random(1, 7);
    printResult(result);
```

```
    delay(100 + i * 10);
  }
  for(int i=0; i<2; i++){
    printResult(0);
    delay(500);
    printResult(result);
    delay(500);
  }
}

void printResult(int result){
  if(result==0){
    lc.setRow(0,0,zero[0]);
    lc.setRow(0,1,zero[1]);
    lc.setRow(0,2,zero[2]);
    lc.setRow(0,3,zero[3]);
    lc.setRow(0,4,zero[4]);
    lc.setRow(0,5,zero[5]);
    lc.setRow(0,6,zero[6]);
    lc.setRow(0,7,zero[7]);
  }
  else if(result==1){
    lc.setRow(0,0,one[0]);
    lc.setRow(0,1,one[1]);
    lc.setRow(0,2,one[2]);
    lc.setRow(0,3,one[3]);
    lc.setRow(0,4,one[4]);
    lc.setRow(0,5,one[5]);
    lc.setRow(0,6,one[6]);
    lc.setRow(0,7,one[7]);
  }
  else if(result==2){
    lc.setRow(0,0,two[0]);
    lc.setRow(0,1,two[1]);
    lc.setRow(0,2,two[2]);
    lc.setRow(0,3,two[3]);
    lc.setRow(0,4,two[4]);
    lc.setRow(0,5,two[5]);
    lc.setRow(0,6,two[6]);
    lc.setRow(0,7,two[7]);
  }
  else if(result==3){
    lc.setRow(0,0,three[0]);
    lc.setRow(0,1,three[1]);
    lc.setRow(0,2,three[2]);
    lc.setRow(0,3,three[3]);
    lc.setRow(0,4,three[4]);
    lc.setRow(0,5,three[5]);
    lc.setRow(0,6,three[6]);
    lc.setRow(0,7,three[7]);
  }
  else if(result==4){
    lc.setRow(0,0,four[0]);
    lc.setRow(0,1,four[1]);
    lc.setRow(0,2,four[2]);
    lc.setRow(0,3,four[3]);
    lc.setRow(0,4,four[4]);
    lc.setRow(0,5,four[5]);
    lc.setRow(0,6,four[6]);
    lc.setRow(0,7,four[7]);
  }
```

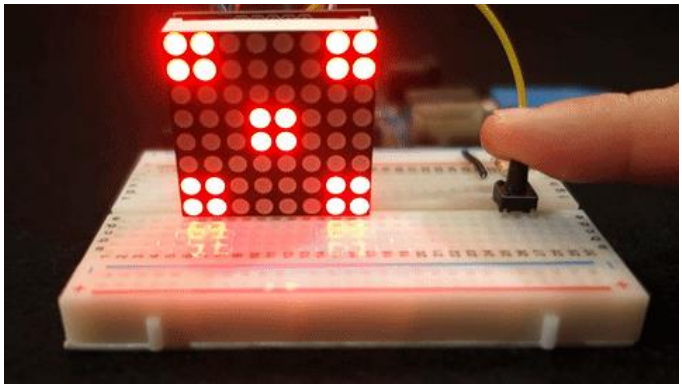
```
  else if(result==5){
    lc.setRow(0,0,five[0]);
    lc.setRow(0,1,five[1]);
    lc.setRow(0,2,five[2]);
    lc.setRow(0,3,five[3]);
    lc.setRow(0,4,five[4]);
    lc.setRow(0,5,five[5]);
    lc.setRow(0,6,five[6]);
    lc.setRow(0,7,five[7]);
  }
  else if(result==6){
    lc.setRow(0,0,six[0]);
    lc.setRow(0,1,six[1]);
    lc.setRow(0,2,six[2]);
    lc.setRow(0,3,six[3]);
    lc.setRow(0,4,six[4]);
    lc.setRow(0,5,six[5]);
    lc.setRow(0,6,six[6]);
    lc.setRow(0,7,six[7]);
  }
  Serial.print(result);
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/15_electronic_dice_roller.ino

Demonstration

Here's your dice roller in action.

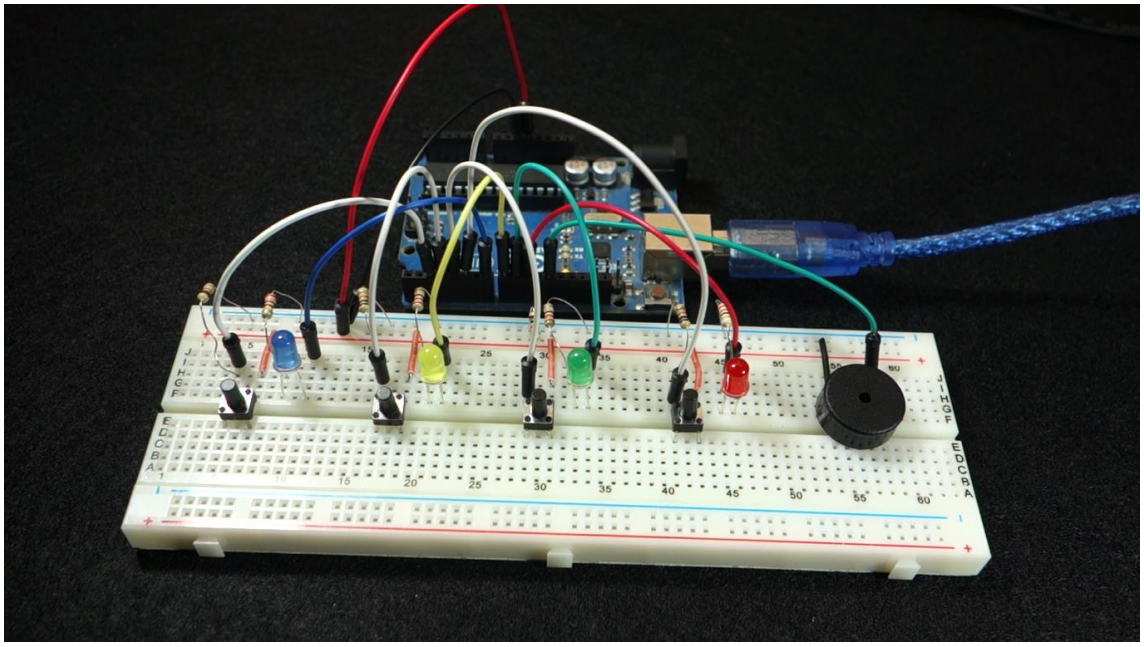


Wrapping up

In this project you created a dice roller that you can actually use with your friends when playing board games.

You can easily upgrade this project to simulate two dices rolling by adding another dot matrix and a few more lines of code.

Memory Game



Memory Game

Level: Intermediate – Time: 25 minutes

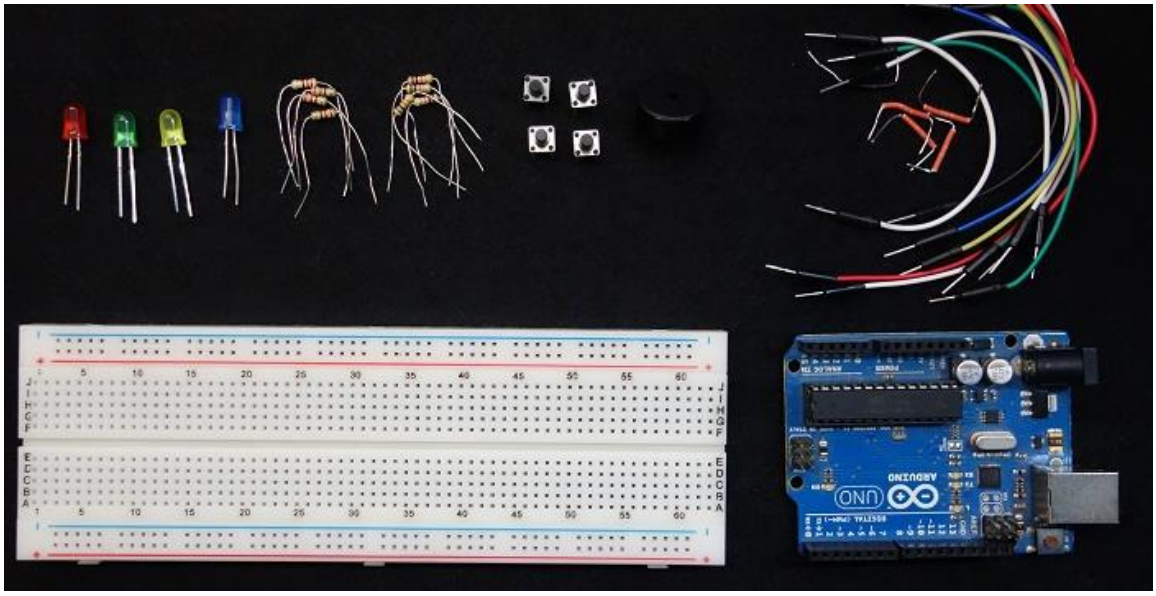
In this project you're going to build a memory game!

You'll have 4 LEDs and 4 pushbuttons – one associated with each LED.

Your Arduino will blink the LEDs in a specific random sequence and you'll have to copy that sequence by pressing the corresponding pushbuttons.

Additionally, you'll have a buzzer to make sound effects, making the game even more fun!

Parts required



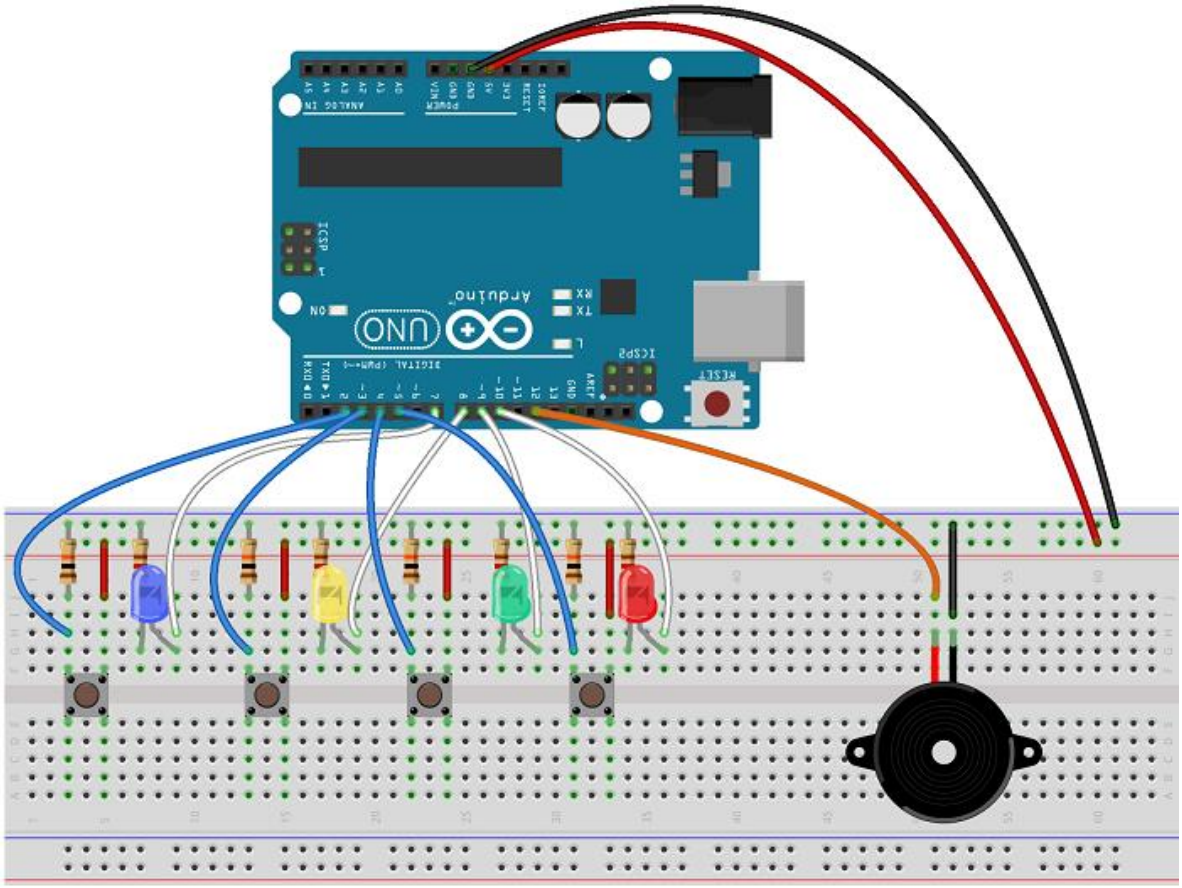
Grab all the components needed for this project. If you don't have them, just visit the links in the table below.

Figure	Name	eBay
	Arduino UNO	http://ebay.to/1SQda0R
	Breadboard	http://ebay.to/21bEojM
	4x 5mm LED (1x red, 1x yellow, 1x green, 1x blue)	http://ebay.to/20H2Oyy

	4x 220Ohm Resistor	http://ebay.to/1KsMYFP
	1x buzzer	http://ebay.to/2dcur1W
	4x 10kOhm Resistor	http://ebay.to/1KsMYFP
	4x pushbutton	http://ebay.to/211vcRv
	Jumper Wires	http://ebay.to/1PXeaJz

Schematics

Assemble all the parts by following the schematics below.



Code

Copy the next code to your Arduino IDE and upload it to your Arduino board. Make sure you have the right board and COM port selected.

```
/*
  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 leddelay = 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;
    }

}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/21_arduino_memory_game.ino

Demonstration

Now, test your memory with this game!

Light Following Robot

Light Following Robot

Level: intermediate – Time: 1h

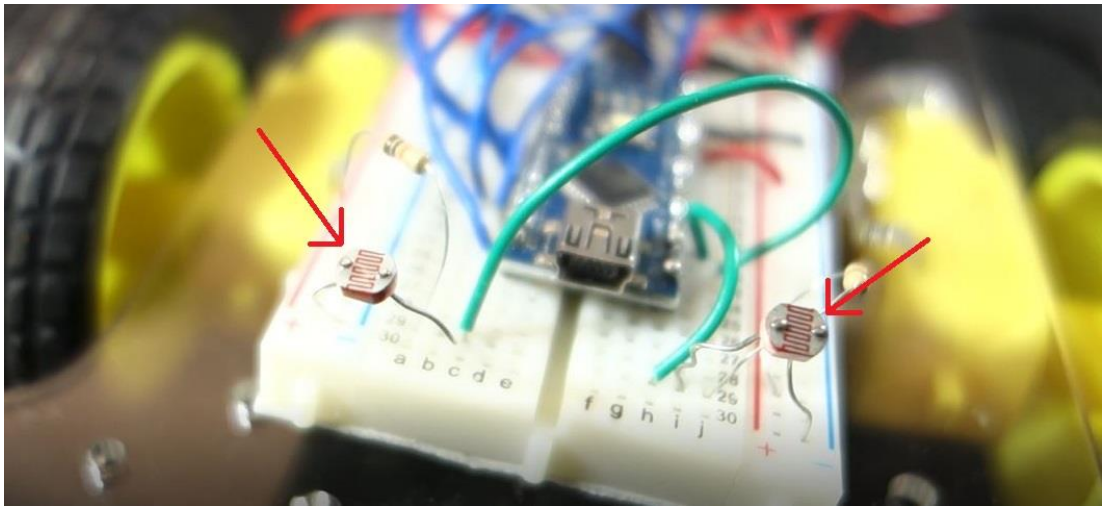
In this project you're going to build a light following car robot.

The car robot DC motors will be controlled accordingly to the light intensity on its surroundings.

Here are the main features of the robot:

- When it detects more light at the rightmost side, it turns right
- When it detects more light at the leftmost side, it turns left
- When the light detected from both sides is similar, it goes forward
- When it's dark, it rotates over itself trying to find light

The light intensity is measured using two photoresistors that are strategically positioned at the front of the robot.



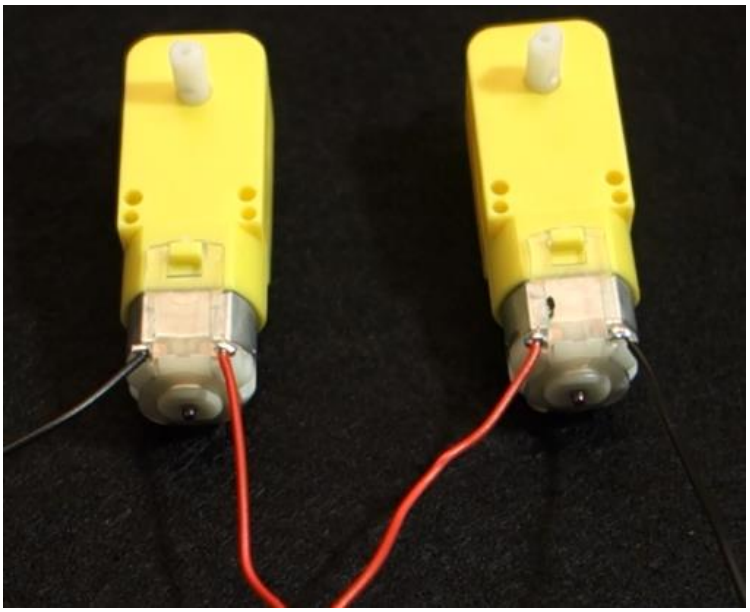
Assembling the robot chassis

You can use any robot chassis you want, or you can build one with materials that you might have laying around. However, I decided to buy one at eBay, it is affordable and it comes with 2 DC motors.

DC motors

After having your robot chassis ready, it's time to check the electronics components.

To get your robot moving, you'll use two DC motors as shown in the figure below.



Connecting the DC motors

You have two wires connected to your DC motors: one red and one black.

Note: usually, the motors in these kits have no polarity (unless they are specifically marked with a + or -. So, changing the way you wire the motor changes the direction of rotation.

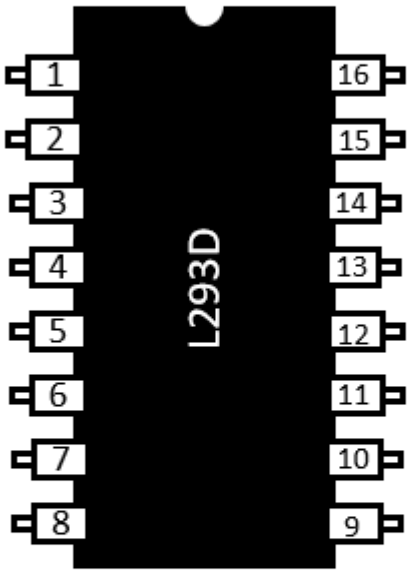
To get you familiar with the DC motor, connect the red wire to 5V and the black wire to the GND (or the other way around). Once they are connected, the shaft starts spinning continuously in one direction.

However, you don't want your robot to just move forward or backward, you want both and without having to switch the red and the black wire. For that, you need to use an H-bridge.

L293D IC (H-bridge)

An H-bridge is a circuit that enables a voltage to be applied across a load (the motors) in either direction. For that, you need an L293D IC.

The figure shows the L293D IC pinout.

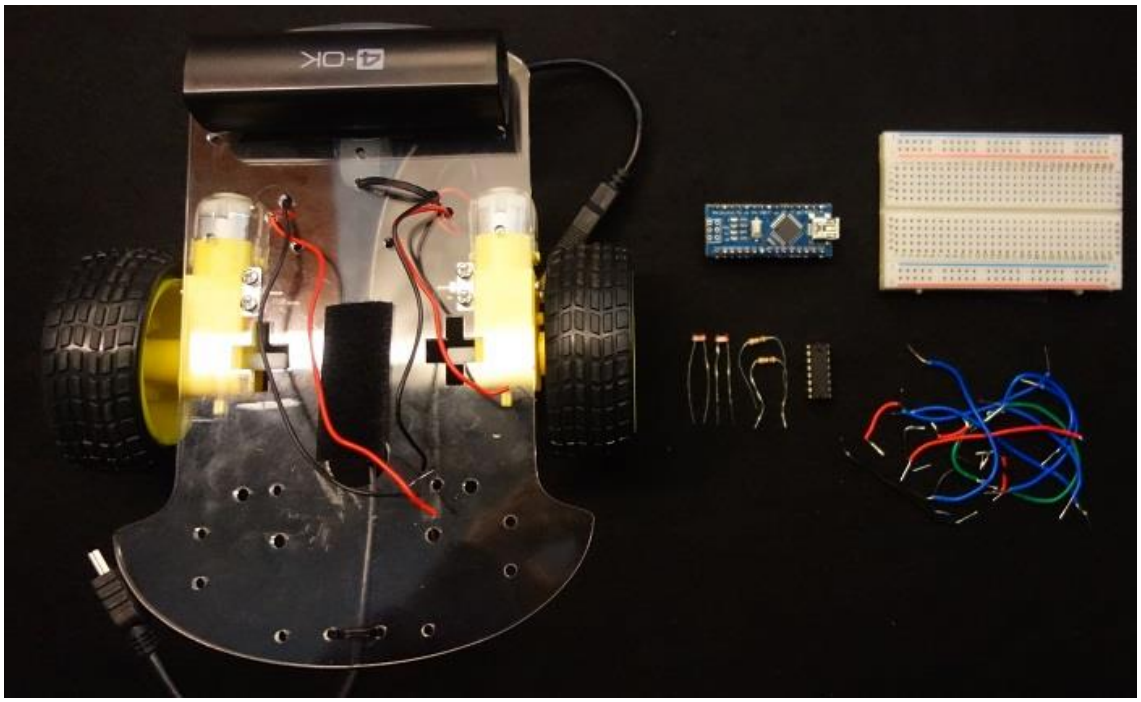


Pin	Description
1	Enable 1
2	Input 1
3	Output 1
4	GND
5	GND
6	Output 2
7	Input 2
8	VCC 1
9	Enable 2
10	Input 3
11	Output 3
12	GND
13	GND
14	Output 4
15	Input 4
16	VCC 2

- The output pins are the ones that your DC motor should be connected to. These are the pins that output the signal that tells the motor what to do
- The VCC pins power up the IC
- The input pins are controlled by the Arduino. They determine in which direction the DC motor should rotate
- The enable pins turn the motor on or off

Parts required

For this project, I'm using an Arduino nano, because it can be attached to a breadboard, making it easier to fix it to the Robot. Nonetheless, this project works using an Arduino uno. You just have to be careful the way you hook it up to the chassis.



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Arduino Nano	http://ebay.to/2eKDZ9a
	L293D IC	http://ebay.to/2e4AXt4
	2x Photoresistor	http://ebay.to/2bMWpSV
	2x 10kOhm resistor	http://ebay.to/1KsMYFP

	Robot Chassis	http://ebay.to/2exBEKO
	Power Bank	http://ebay.to/2dDhmRo
	Breadboard	http://ebay.to/21bEojM
	Jumper Wires	http://ebay.to/1PXeaJz

You don't need to buy a power bank, if you don't have one. You can use batteries to power up your robot, because a battery holder comes with the chassis kit.

In my opinion, a power bank is more handy and practical than batteries.

Code

Upload the following code to your Arduino board. Make sure you have the right board and COM Port selected.

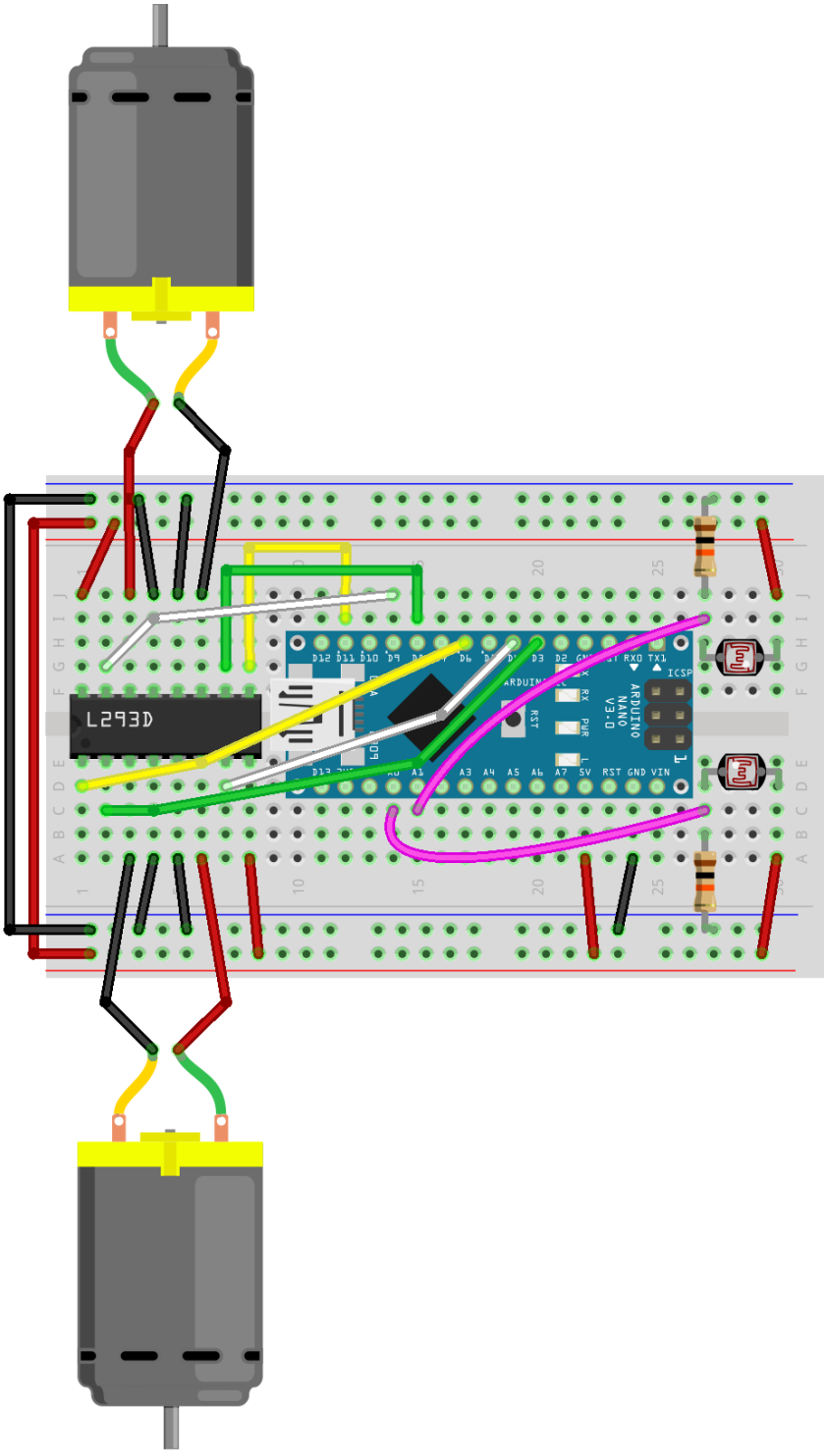
```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

const int motor1Pin1 = 3; // pin 2 on L293D IC
const int motor1Pin2 = 4; // pin 7 on L293D IC
const int enable1Pin = 6; // pin 1 on L293D IC
const int motor2Pin1 = 8; // pin 10 on L293D IC
const int motor2Pin2 = 9; // pin 15 on L293D IC
const int enable2Pin = 11; // pin 9 on L293D IC

// My right sensor might be YOUR left sensor
const int rightSensor = A0;
const int leftSensor = A1;

// Variable definitions
int leftSensorValue;
int rightSensorValue;
int sensorDifferenceValue;

// Threshold Value for the sensor difference
const int thresholdValueDifference = 75;

void setup() {
  // Setting outputs and inputs
```

```
  pinMode(enable2Pin, OUTPUT);
  pinMode(motor2Pin1, OUTPUT);
  pinMode(motor2Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(motor1Pin1, OUTPUT);
  pinMode(leftSensor, INPUT);
  pinMode(rightSensor, INPUT);
  // Serial debugging at 9600 baud rate
  Serial.begin(9600);
}

void loop() {
  // Reads LDR sesnsors and calculates the differences
  leftSensorValue = 1023-analogRead(leftSensor);
  rightSensorValue = 1023-analogRead(rightSensor);
  sensorDifferenceValue = abs(leftSensorValue-rightSensorValue);

  // Prints the current readings
  Serial.print("Left Sensor = ");
  Serial.print(leftSensorValue);
  Serial.print(" - Right Sensor = ");
  Serial.println(rightSensorValue);

  // Turns the robot Left, if the light is currently higher in the
Left LDR
  if (leftSensorValue > rightSensorValue &&
    sensorDifferenceValue > thresholdValueDifference){
    analogWrite(enable2Pin, 255);
    digitalWrite(motor2Pin1, LOW);
    digitalWrite(motor2Pin2, HIGH);
    analogWrite(enable1Pin, 255);
    digitalWrite( motor1Pin2, HIGH);
    digitalWrite( motor1Pin1, LOW);
    Serial.println("Left");
    delay(100);
  }
  // Turns the robot Right if the light is currently higher in the
Right LDR
  if (leftSensorValue < rightSensorValue &&
    sensorDifferenceValue > thresholdValueDifference){
    analogWrite(enable2Pin, 255);
    digitalWrite(motor2Pin1, HIGH);
    digitalWrite(motor2Pin2, LOW);
    analogWrite(enable1Pin, 255);
    digitalWrite(motor1Pin2, LOW);
    digitalWrite(motor1Pin1, HIGH);
    Serial.println("Right");
    delay(100);
  }
  // Goes forward if the difference between the two is not relevant
  else if (sensorDifferenceValue < thresholdValueDifference){
    analogWrite(enable2Pin, 255);
    digitalWrite(motor2Pin1, HIGH);
    digitalWrite(motor2Pin2, LOW);
    analogWrite(enable1Pin, 255);
    digitalWrite(motor1Pin2, HIGH);
    digitalWrite(motor1Pin1, LOW);
```

```
    Serial.println("Forward");
    delay(100);
  }
}
```

Schematics

Assemble the circuit by following the schematics below.

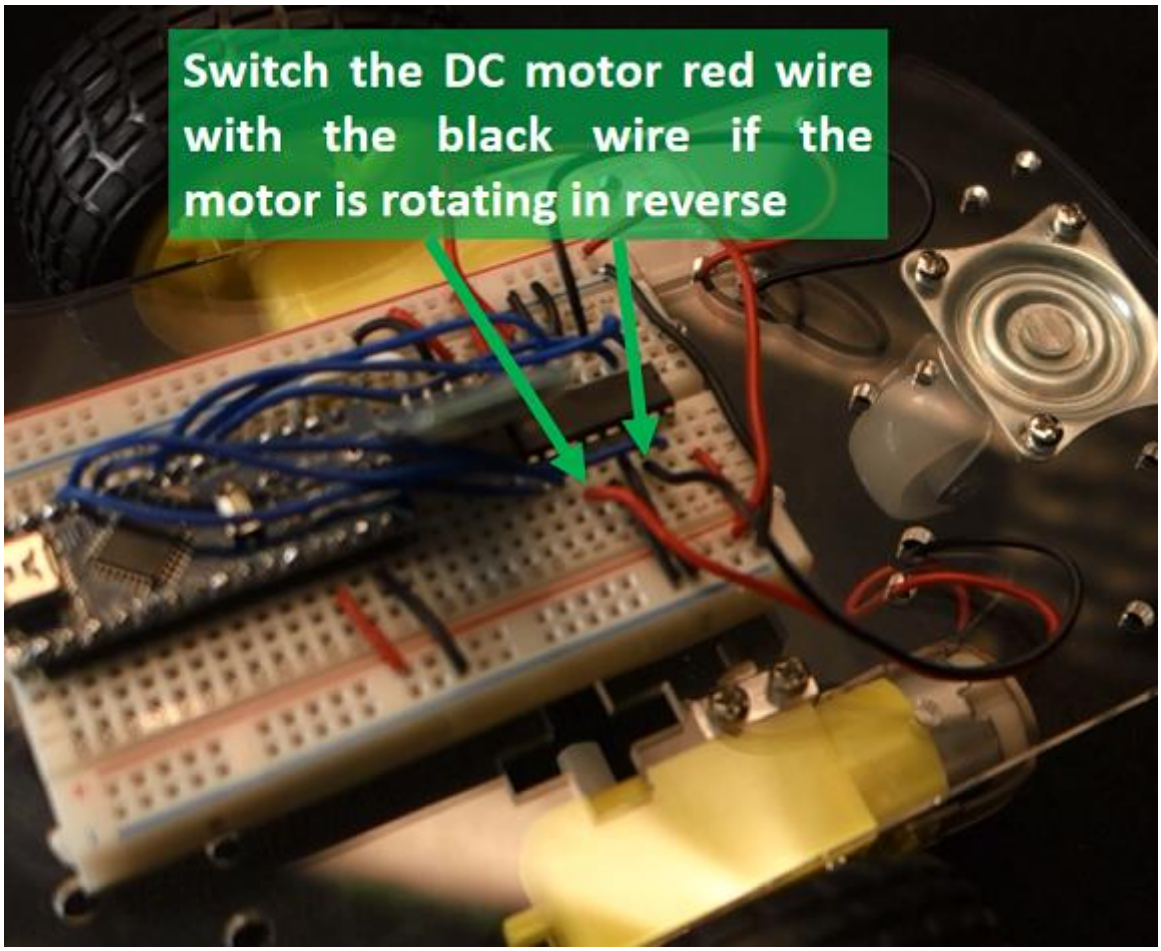


If the connections to the L293D seem confusing, take a look at the wiring table below. This wiring is also valid if you're using an Arduino Uno.

L293D	Wiring to
Pin 1	Digital Pin 6
Pin 2	Digital Pin 3
Pin 3	DC motor
Pin 4	GND
Pin 5	GND
Pin 6	DC motor
Pin 7	Digital Pin 4
Pin 8	5V
Pin 9	Digital Pin 11
Pin 10	Digital Pin 8
Pin 11	DC motor
Pin 12	GND
Pin 13	GND
Pin 14	DC motor
Pin 15	Digital Pin 9
Pin 16	5V

Troubleshooting

If one or both motors are rotating in the wrong direction, you should switch the DC motor red wire with the black wire. This should solve that problem.



Demonstration

Here's your light following robot in action. My robot is following a flashlight light.

Have fun!

Wrapping up

In this project, you've built a simple light following robot!

As you can see, building a robot is not difficult at all.

Now that you understand the main components of building a robot, you can customize it to do other tasks.

Remote Controlled Robot

Bluetooth Remote Controlled Robot

Level: intermediate – Time: 2h

In this project you're going to build a bluetooth remote controlled robot. You'll make an Android app with the MIT App Inventor that allows you to control 2 DC motors.

You will be able to control your robot using 5 states: forward, reverse, right, left and stop.

This is a great project to get you introduced to DC motors.

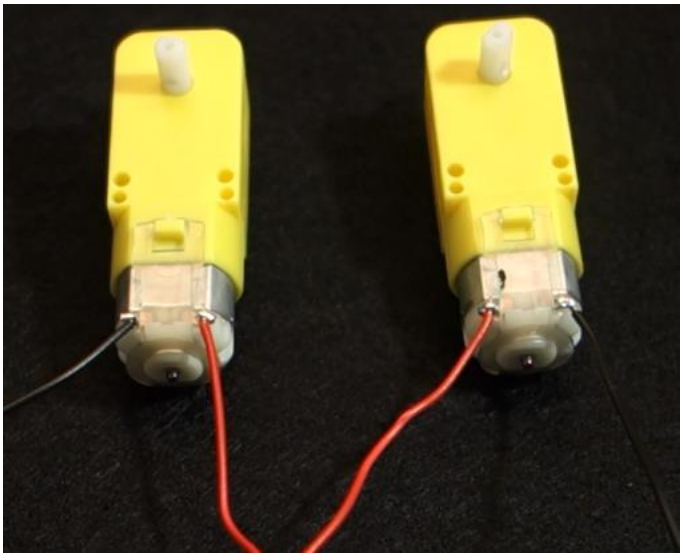
Assembling the robot chassis

You can use any robot chassis you want, or you can build one with materials that you might have laying around. However, I decided to buy one at eBay, it is affordable and it comes with 2 DC motors.

DC motors

After having your robot chassis ready, it's time to check the electronics components.

To get your robot moving, you'll use two DC motors as shown in the figure below.



Connecting the DC motors

You have two wires connected to your DC motors: one red and one black.

Note: usually, the motors in these kits have no polarity (unless they are specifically marked with a + or -. So, changing the way you wire the motor changes the direction of rotation.

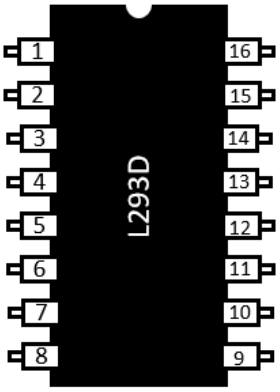
To get you familiar with the DC motor, connect the red wire to 5V and the black wire to the GND (or the other way around). Once they are connected, the shaft starts spinning continuously in one direction.

However, you don't want your robot to just move forward or backward, you want both and without having to switch the red and the black wire. For that, you need to use an H-bridge.

Introducing L293D IC (H-bridge)

An H-bridge is a circuit that enables a voltage to be applied across a load (the motors) in either direction. For that, you need an L293D IC.

The figure shows the L293D IC pinout.



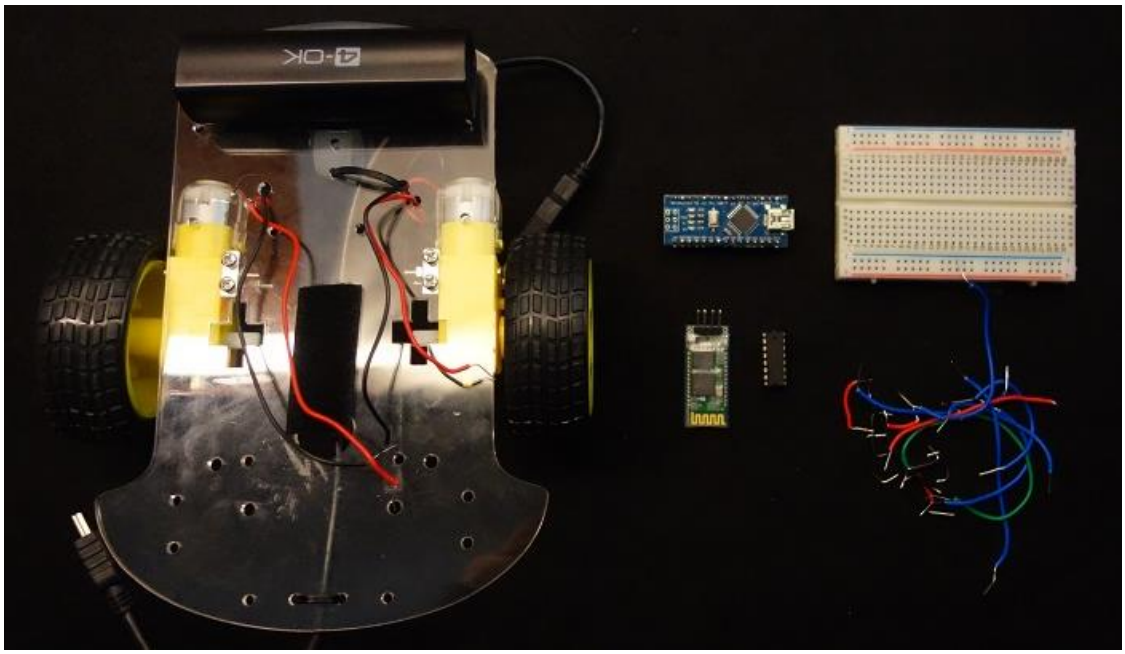
Pin	Description
1	Enable 1
2	Input 1
3	Output 1
4	GND
5	GND
6	Output 2
7	Input 2
8	VCC 1
9	Enable 2
10	Input 3
11	Output 3
12	GND
13	GND
14	Output 4
15	Input 4
16	VCC 2

The output pins are the ones that your DC motor should be connected to. These are the pins that output the signal that tells the motor what to do

- The VCC pins power up the IC
- The input pins are controlled by the Arduino. They determine in which direction the DC motor should rotate
- The enable pins turn the motor on or off

Parts required

For this project, I'm using an Arduino nano, because it can be attached to a breadboard, making it easier to fix it to the Robot. Nonetheless, this project works using an Arduino uno. You just have to be careful the way you hook it up to the chassis.



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

Figure	Name	eBay
	Arduino Nano	http://ebay.to/2eKDZ9a
	HC-04 or HC-05 or HC-06 Bluetooth module	http://ebay.to/2dt1b6M
	L293D IC	http://ebay.to/2e4AXt4
	Robot Chassis	http://ebay.to/2exBEKO
	Power Bank	http://ebay.to/2dDhmRo

	Breadboard	http://ebay.to/21bEojM
	Jumper Wires	http://ebay.to/1PXealz

You don't need to buy a power bank, if you don't have one. You can use batteries to power up your robot, because a battery holder comes with the chassis kit.

In my opinion, a power bank is more handy and practical than batteries.

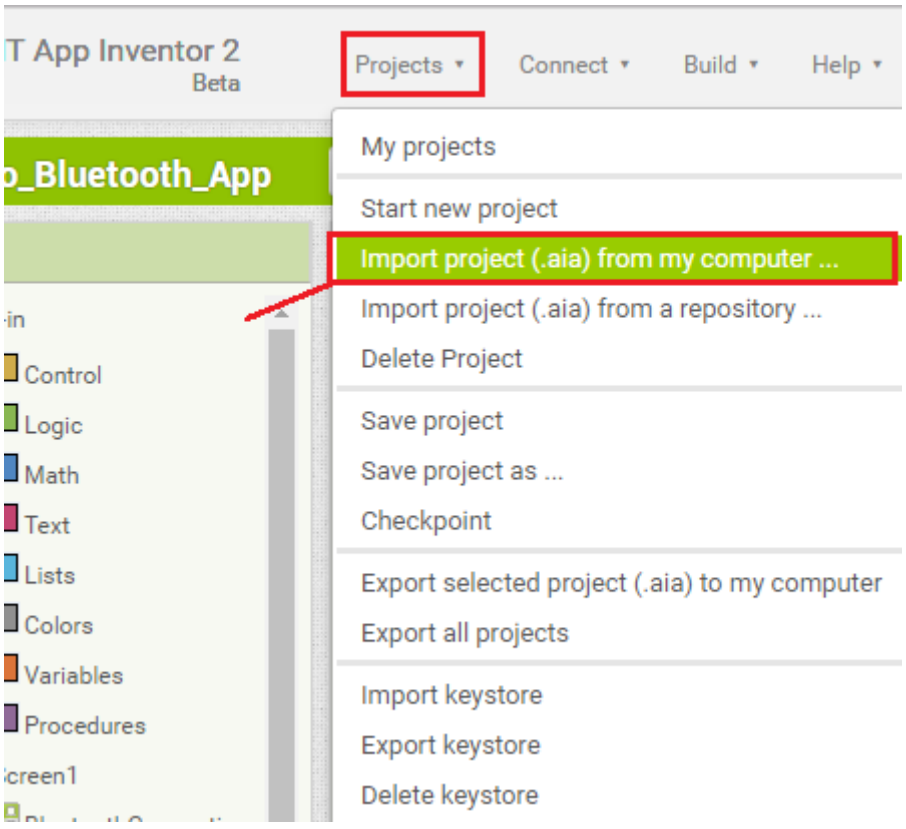
Creating your Android App

The Android App will be created using a free web application called MIT App Inventor. MIT App Inventor is a great place to get started with Android development, because it allows you to build simple apps with drag-n-drop.

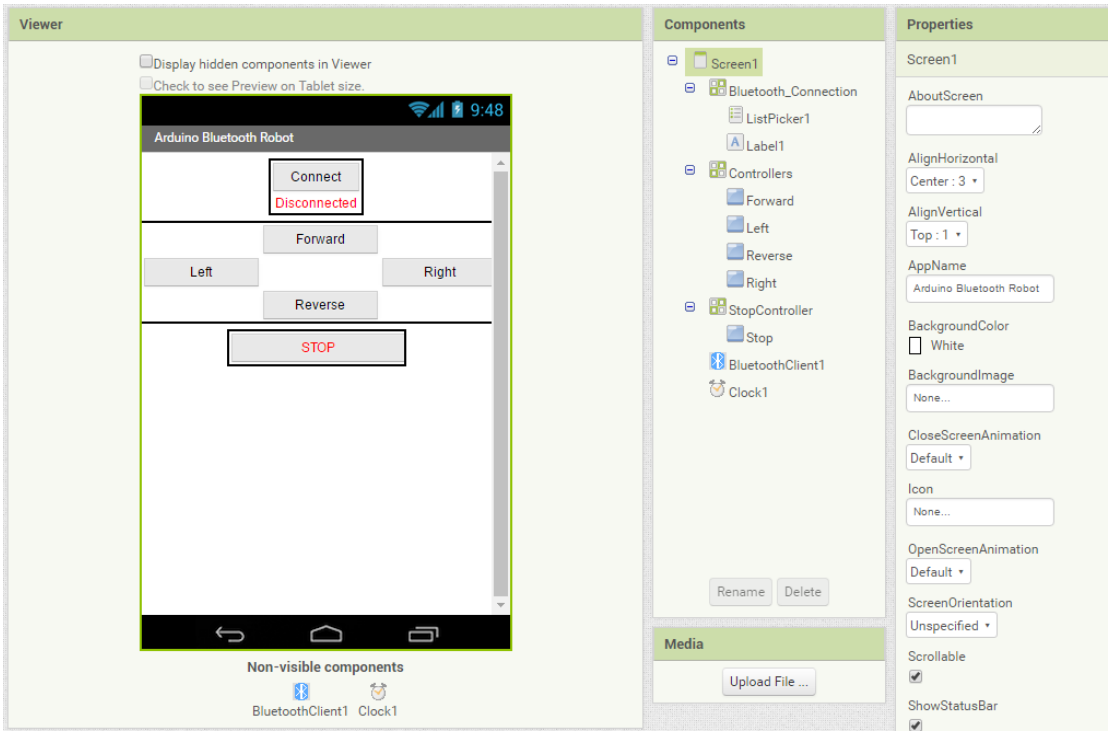
You need a Google account to sign up for MIT App Inventor and here's the login page: http://ai2.appinventor.mit.edu.

Click here to download the .aia file.

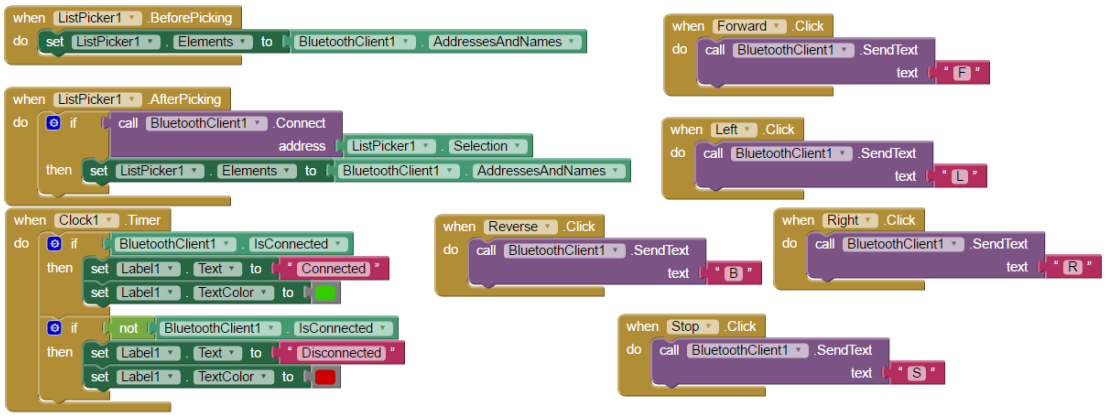
If you go to the Projects tab, you can upload the .aia file for this project.



With MIT App Inventor you have 2 main sections: designer and blocks. The designer is what gives you the ability to add buttons, add text, add screens and edit the overall app look.



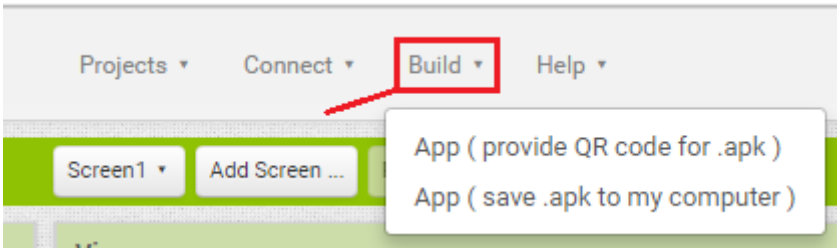
The blocks sections is what allows to create custom functionality for your app, so when you press the buttons it actually does something with that information.



I recommend that you start by following this project and using the app without modifying it.

If want to make any changes to the app, when you're done and you want to install the app in your smartphone, go to the Build tab.

You can either generate a QR code that you can scan with your smartphone and automatically install the app in your smartphone, or you can download the .apk file, connect your smartphone to your computer and move the .apk file to the phone.



Simply follow the installation wizard to install the App and it's done!

Code

Upload the following code to your Arduino. Make sure you have the right Board and COM Port selected.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

int motor1Pin1 = 3; // pin 2 on L293D IC
int motor1Pin2 = 4; // pin 7 on L293D IC
int enable1Pin = 6; // pin 1 on L293D IC
int motor2Pin1 = 8; // pin 10 on L293D IC
int motor2Pin2 = 9; // pin 15 on L293D IC
int enable2Pin = 11; // pin 9 on L293D IC
int state;
int flag=0;        //makes sure that the serial only prints once
the state
int stateStop=0;
void setup() {
    // sets the pins as outputs:
    pinMode(motor1Pin1, OUTPUT);
    pinMode(motor1Pin2, OUTPUT);
    pinMode(enable1Pin, OUTPUT);
    pinMode(motor2Pin1, OUTPUT);
    pinMode(motor2Pin2, OUTPUT);
    pinMode(enable2Pin, OUTPUT);
    // sets enable1Pin and enable2Pin high so that motor can turn
on:
    digitalWrite(enable1Pin, HIGH);
    digitalWrite(enable2Pin, 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 '1' the DC motor will go forward
    if (state == 'F') {
        digitalWrite(motor1Pin1, LOW);
        digitalWrite(motor1Pin2, HIGH);
```

```
        digitalWrite(motor2Pin1, HIGH);
        digitalWrite(motor2Pin2, LOW);
        if(flag == 0){
          Serial.println("Go Forward!");
          flag=1;
        }
    }

    // if the state is '2' the motor will turn left
    else if (state == 'R') {
        digitalWrite(motor1Pin1, LOW);
        digitalWrite(motor1Pin2, LOW);
        digitalWrite(motor2Pin1, HIGH);
        digitalWrite(motor2Pin2, LOW);
        if(flag == 0){
          Serial.println("Turn Right");
          flag=1;
        }
        delay(500);
        state=3;
        stateStop=1;
    }
    // if the state is '3' the motor will Stop
    else if (state == 'S' || stateStop == 1) {
        digitalWrite(motor1Pin1, LOW);
        digitalWrite(motor1Pin2, LOW);
        digitalWrite(motor2Pin1, LOW);
        digitalWrite(motor2Pin2, LOW);
        if(flag == 0){
          Serial.println("STOP!");
          flag=1;
        }
        stateStop=0;
    }
    // if the state is '4' the motor will turn right
    else if (state == 'L') {
        digitalWrite(motor1Pin1, LOW);
        digitalWrite(motor1Pin2, HIGH);
        digitalWrite(motor2Pin1, LOW);
        digitalWrite(motor2Pin2, LOW);
        if(flag == 0){
          Serial.println("Turn Left");
          flag=1;
        }
        delay(500);
        state=3;
        stateStop=1;
    }
    // if the state is '5' the motor will Reverse
    else if (state == 'B') {
        digitalWrite(motor1Pin1, HIGH);
        digitalWrite(motor1Pin2, LOW);
        digitalWrite(motor2Pin1, LOW);
        digitalWrite(motor2Pin2, HIGH);
        if(flag == 0){
          Serial.println("Reverse!");
          flag=1;
        }
```

```
    }
    //For debugging purpose
    //Serial.println(state);
}
```

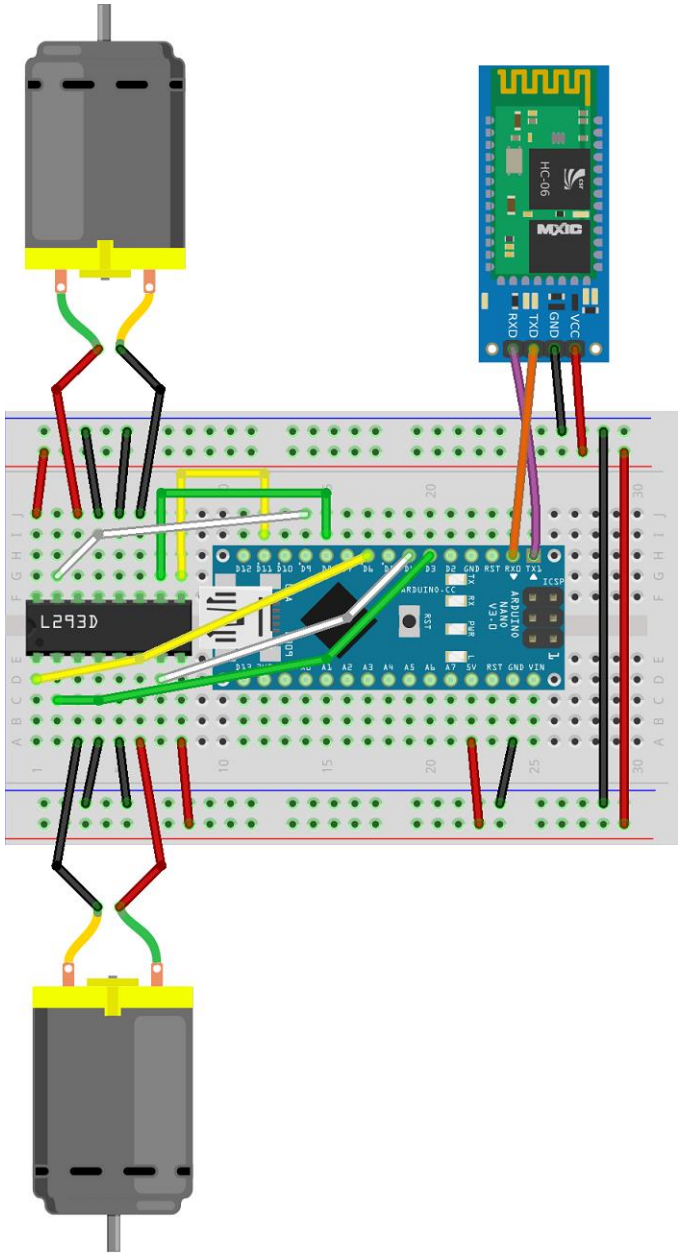
Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/12_bluetooth_remote_controlled_robot.ino

Note: before uploading the code, make sure you have the TX and RX pins disconnected from the bluetooth module!

Schematics

Assemble the circuit by following the schematics below.



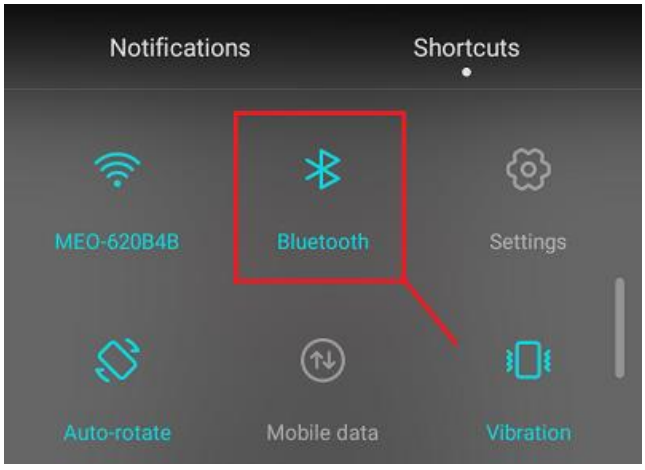
If the connections to the L293D seem confusing, take a look at the wiring table below. This wiring is also valid if you're using an Arduino Uno.

L293D	Wiring to
Pin 1	Digital Pin 6
Pin 2	Digital Pin 3
Pin 3	DC motor
Pin 4	GND
Pin 5	GND
Pin 6	DC motor
Pin 7	Digital Pin 4
Pin 8	5V
Pin 9	Digital Pin 11
Pin 10	Digital Pin 8
Pin 11	DC motor
Pin 12	GND
Pin 13	GND
Pin 14	DC motor
Pin 15	Digital Pin 9
Pin 16	5V

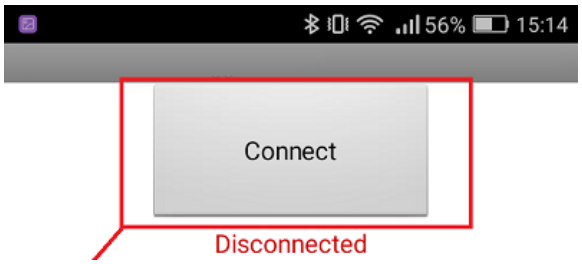
Launching your App

If you haven't generated the .apk file in a previous step, you can click here to download the .apk file (which is the Android App installation file). Move that file to your smartphone and open it. Follow the installation wizard to install the app.

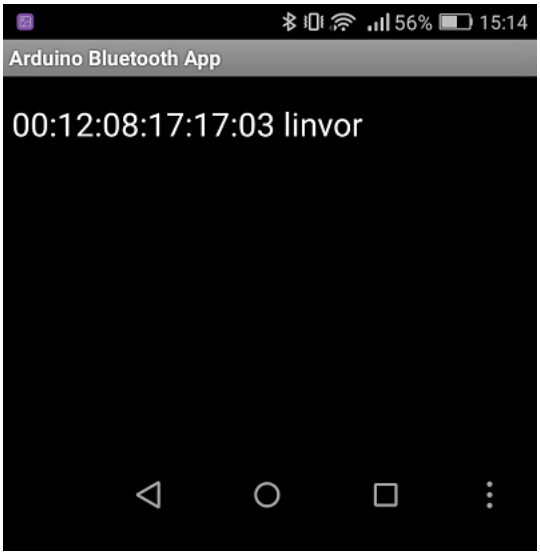
Turn on your smartphone's Bluetooth.



Tap on the newly installed app. Press the "Connect" button to connect your application to your Arduino Bluetooth module.



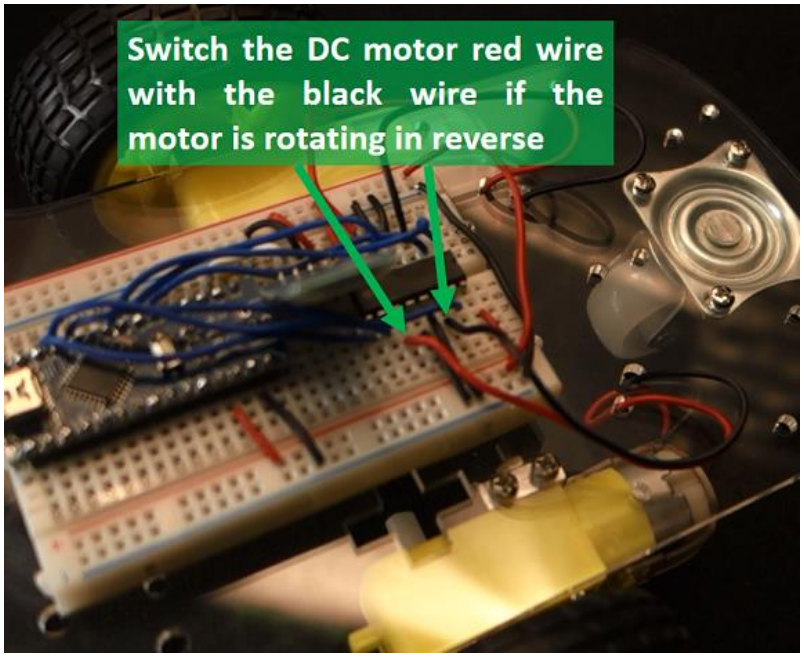
Select your Bluetooth module (it should be named linvor).



Now, it is ready to use!

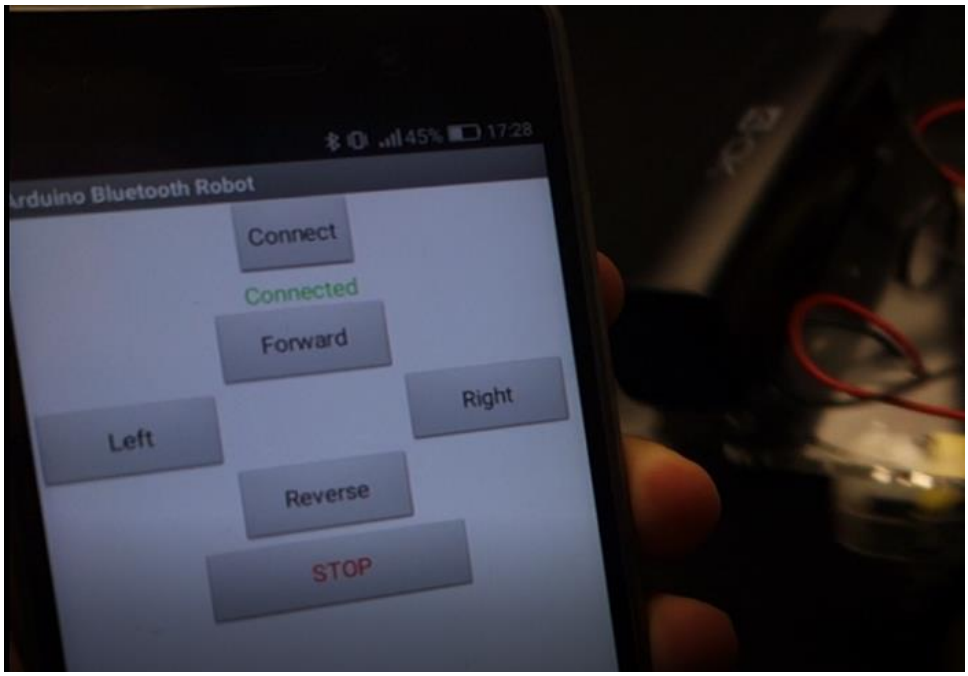
Troubleshooting

If one or both motors are rotating in the wrong direction, you should switch the DC motor red wire with the black wire. This should solve that problem.



Demonstration

This is how your app looks like:



And here's your bluetooth remote controlled robot.

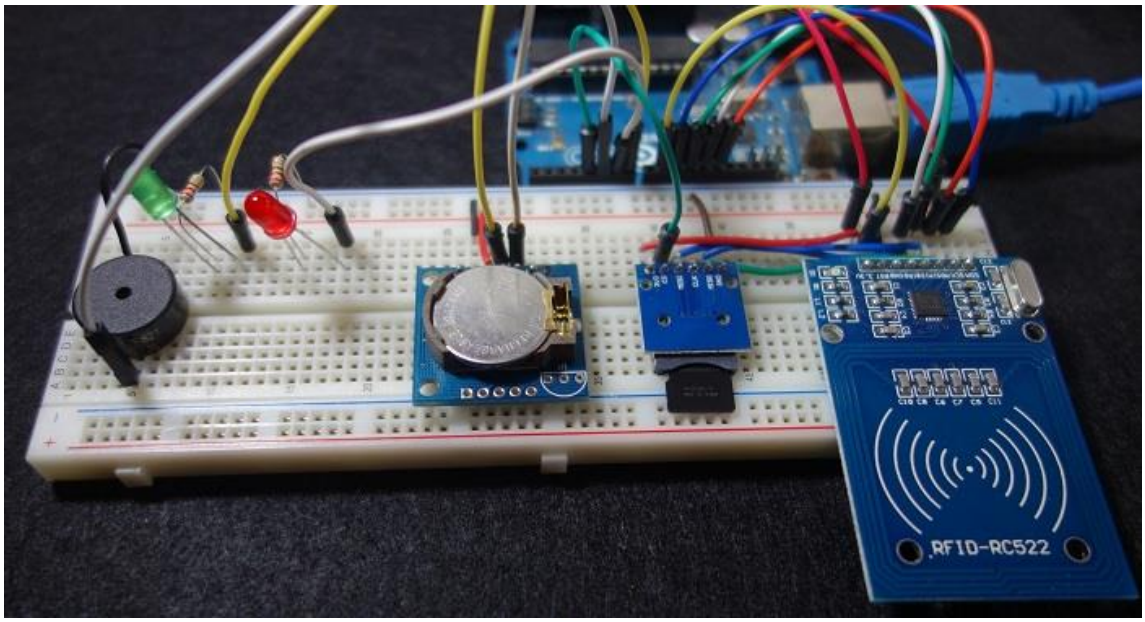
Have fun!

Wrapping up

In this project, you learned how to build a simple remote controlled robot.

You can add more features, such as LEDs that indicate the robot direction, automatic lights that light up when it's dark, and much more.

Time Attendance System with RFID

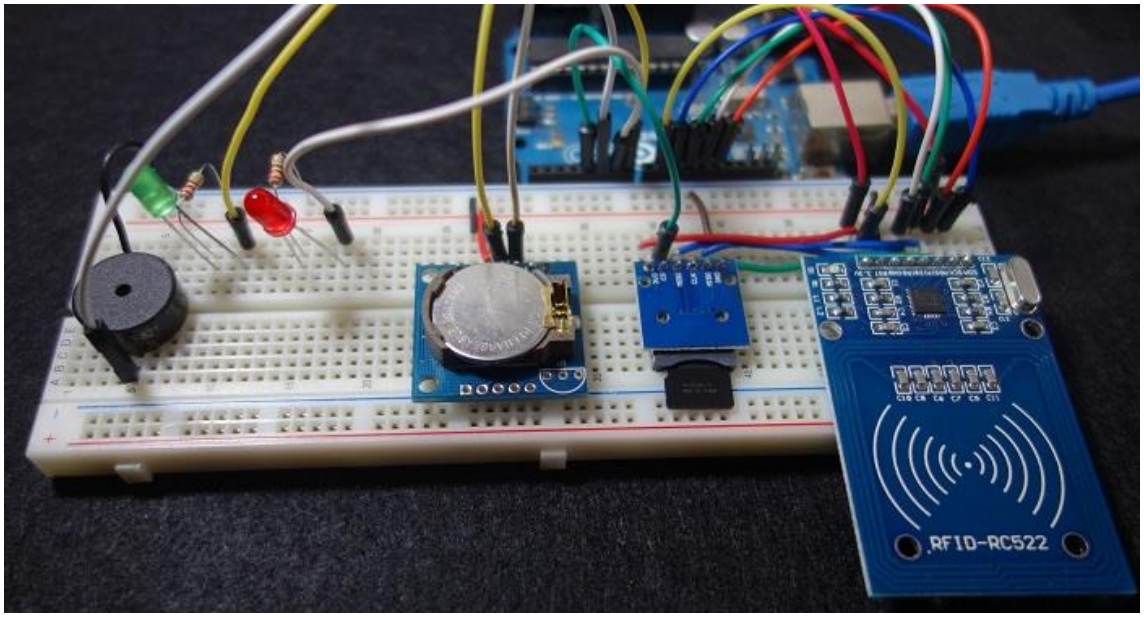


Time Attendance System with RFID

Level: Intermediate– Time: 1h15m

In this project you're going to build a time attendance system with RFID.

When you swipe an RFID tag next to the RFID reader, it saves the user UID and time in an SD card. It also shows if you are late or in time according to a preset hour and minute.



Project overview

Before getting started it's important to layout the project main features:

- Contains an RFID reader that reads RFID tags
- Has a real time clock module to keep track of time
- When the RFID reader reads an RFID tag, it saves the current time and the UID of the tag in an SD card
- The Arduino communicates with the SD card using an SD card module
- You can set a check in time to compare if you are in time or late
- If you are on time, a green LED lights up, if you are late, a red LED lights up
- The system also has a buzzer that beeps when a tag is read

MFRC522 RFID Reader

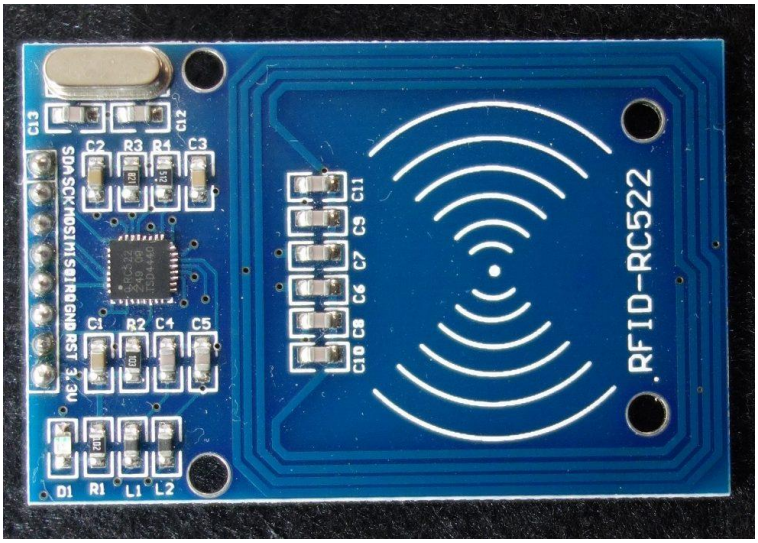
In this project we're using the MFRC522 RFID reader and that's the one we recommend you to get (although this project may also be compatible with other RFID readers).

RFID means radio-frequency identification. RFID uses electromagnetic fields to transfer data over short distances and is useful to identify people, to make transactions, etc.

An RFID system needs tags and a reader:

- **tags** are attached to the object to be identified, in this example we have a keychain and an electromagnetic card. Some stores also use RFID tags in their products' labels to identify them. Each tag has its own unique identification (UID).

- **reader** is a two-way radio transmitter-receiver that sends a signal to the tag and reads its response.



The MFRC522 RFID reader works at 3.3V and it can use SPI or I2C communication. However, the library we're going to use to control the RFID reader only supports SPI, so that's the communication protocol we're going to use.

Installing the MFRC522 library

This project uses the MFRC522.h library to control the RFID reader. This library doesn't come installed in Arduino IDE by default, so you need to install it. Go to **Sketch** > **Include library** > **Manage libraries** and search for **MFRC522** or follow the next steps:

1. Click here to download the MFRC522 library. You should have a *.zip* folder in your Downloads folder.
2. Unzip the *.zip* folder and you should get **RFID-master** folder
3. Rename your folder from ~~**RFID-master**~~ to **RFID**
4. Move the **RFID** folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

MFRC522 RFID Reader pinout

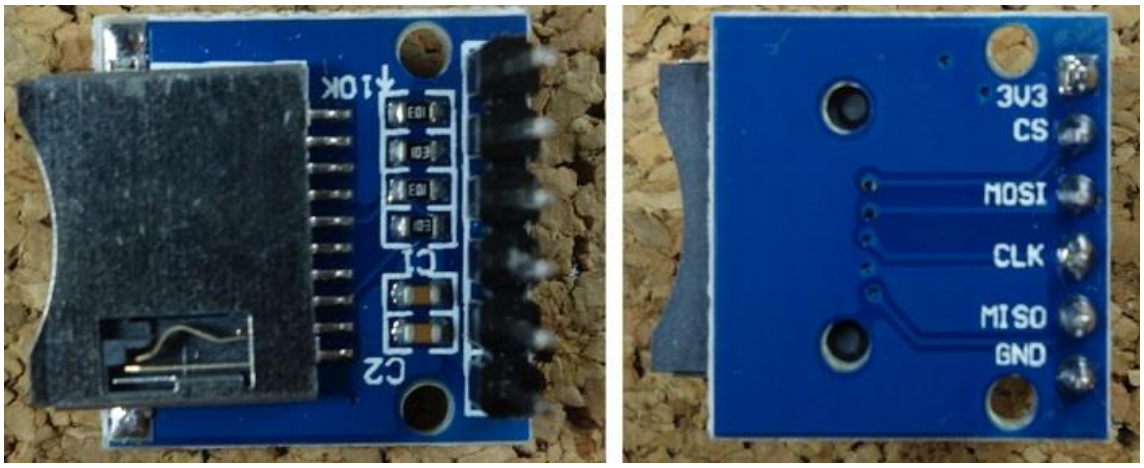
The following table shows the reader pinout for a future reference:

Pin	Wiring to Arduino Uno
SDA	Digital 10
SCK	Digital 13
MOSI	Digital 11
MISO	Digital 12
IRQ	Don't connect
GND	GND
RST	Digital 9
3.3V	3.3V

***Note**: different Arduino boards have different SPI pins. If you're using another Arduino board, check the Arduino documentation.*

SD Card module

When a tag is read, its UID and time are saved on an SD card so that you can keep track of check in's. There are different ways to use an SD card with the Arduino. In this project we're using the SD card module shown in figure below – it works with micro SD card.



There are different models from different suppliers, but they all work in a similar way, using the SPI communication protocol. To communicate with the SD card we're going to use a library called **SD.h**, that comes already installed in Arduino IDE by default.

SD card Module Pinout

The following table shows the SD card module pinout for a future reference:

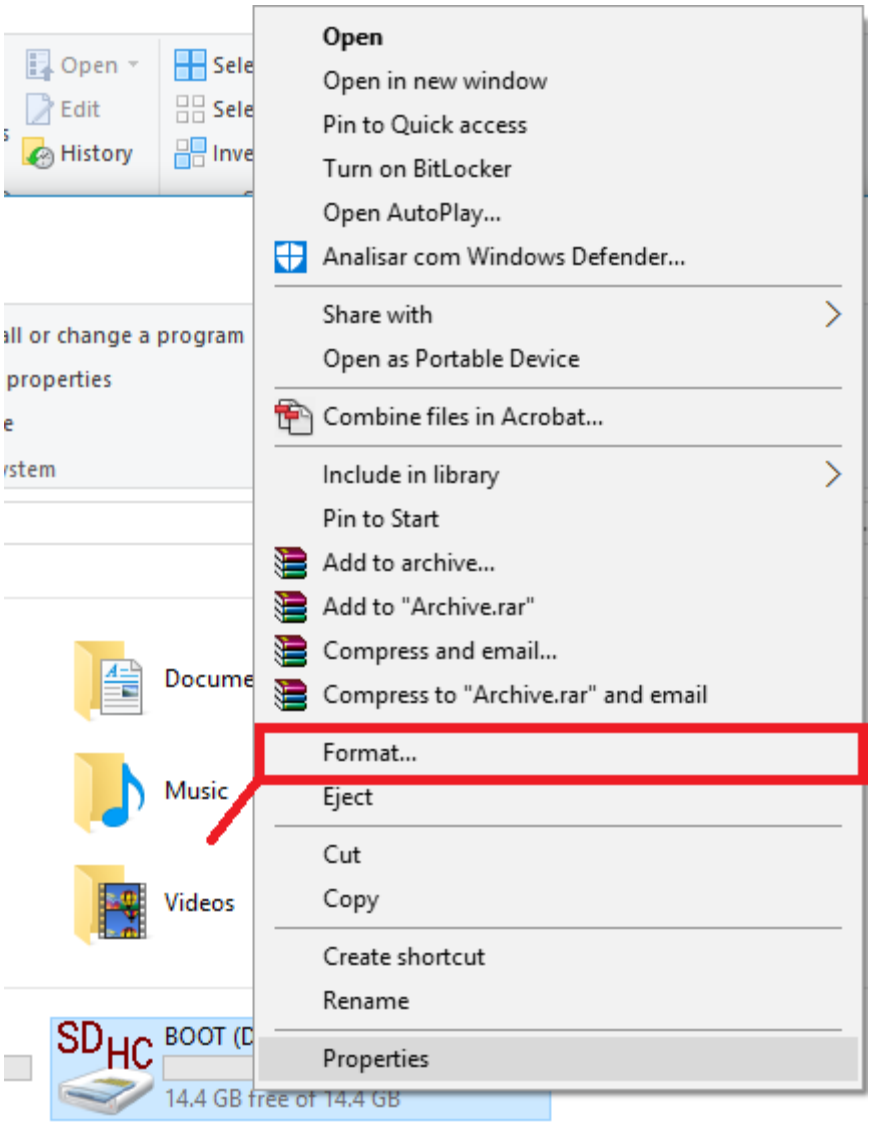
Pin	Wiring to Arduino Uno
VCC	3.3V
CS	Digital 4
MOSI	Digital 11
CLK	Digital 13
MISO	Digital 12
GND	GND

***Note**: different Arduino boards have different SPI pins. If you're using another Arduino board, check the Arduino documentation.*

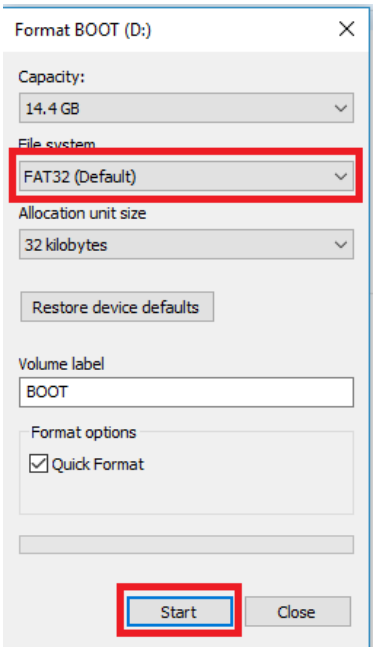
Preparing the SD card

The first step when using the SD card module with Arduino is formatting the SD card as FAT16 or FAT32. Follow the instructions below.

1) To format the SD card, insert it in your computer. Go to **My Computer** and right click on the **SD card**. Select Format as shown in figure below.



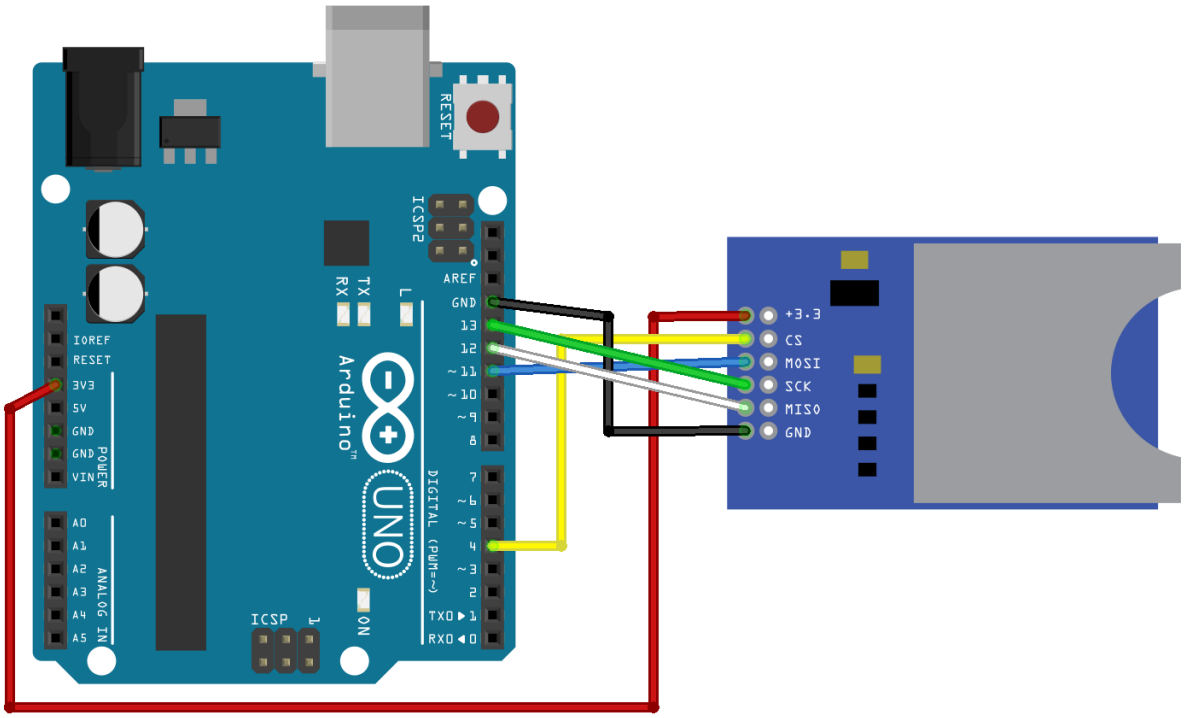
2) A new window pops up. Select FAT32, press Start to initialize the formatting process and follow the onscreen instructions.



Testing the SD card module

(This step is optional. This is an additional step to make sure the SD card module is working properly.)

Insert the formatted SD card in the SD card module. Connect the SD card module to the Arduino as shown in the circuit schematics below or check the pinout table.



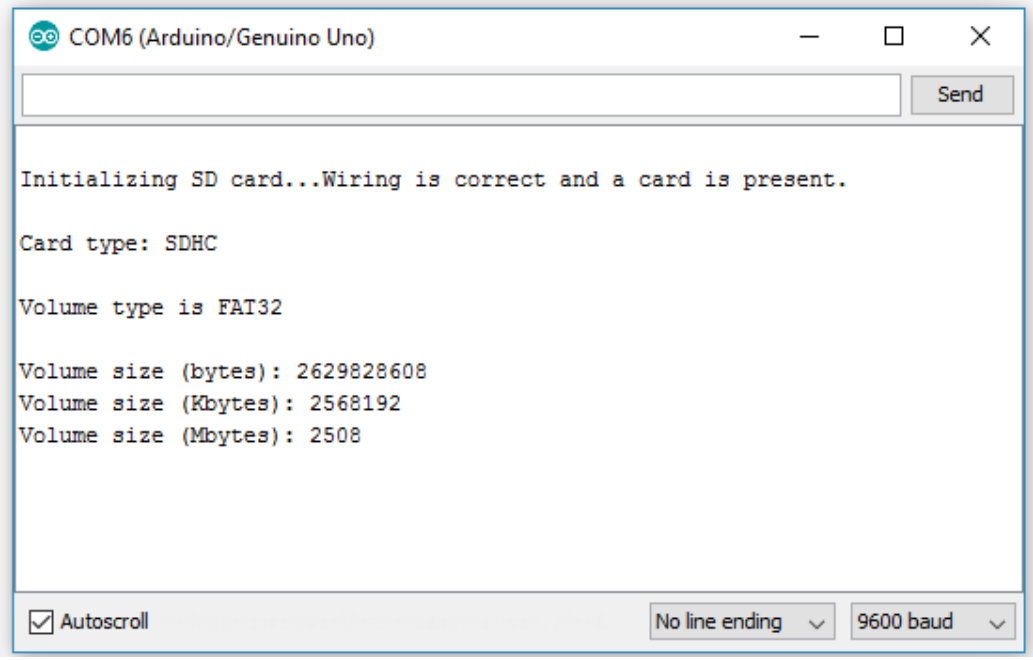
***Note**: depending on the module you're using, the pins may be in a different order.*

Uploading CardInfo sketch

To make sure everything is wired correctly and the SD card is working properly, in the Arduino IDE window go to **File**> **Examples** > **SD** > **CardInfo**.

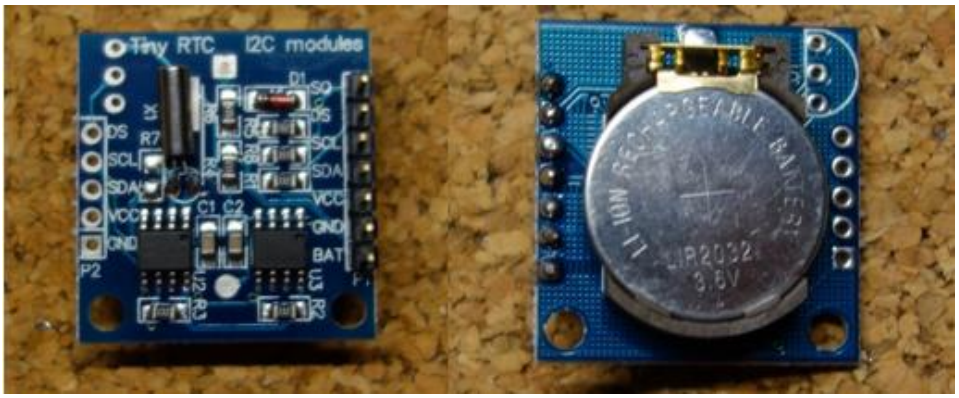
Upload the code to your Arduino board. Make sure you have the right board and COM port selected.

Open the Serial Monitor at a baud rate of **9600** and the SD card information will be displayed. If everything is working properly you'll see a similar message on the serial monitor.



RTC (Real Time Clock) module

To keep track of time, we're using the SD1307 RTC module. However, this project works just fine with the DS3231, which is very similar. One main difference between them is the accuracy. The DS3231 is much more accurate than the DS1307. The figure below shows the SD1307 model.



The module has a backup battery installed. This allows the module to retain the time, even when it's not being powered up.

This module uses I2C communication and we'll use the **RTCLib.h** library to read the time from the RTC.

RTC Module Pinout

The following table shows the RTC module pinout for a future reference:

Pin	Wiring to Arduino Uno
SCL	A5
SDA	A4
VCC	5 V (check your module datasheet)
GND	GND

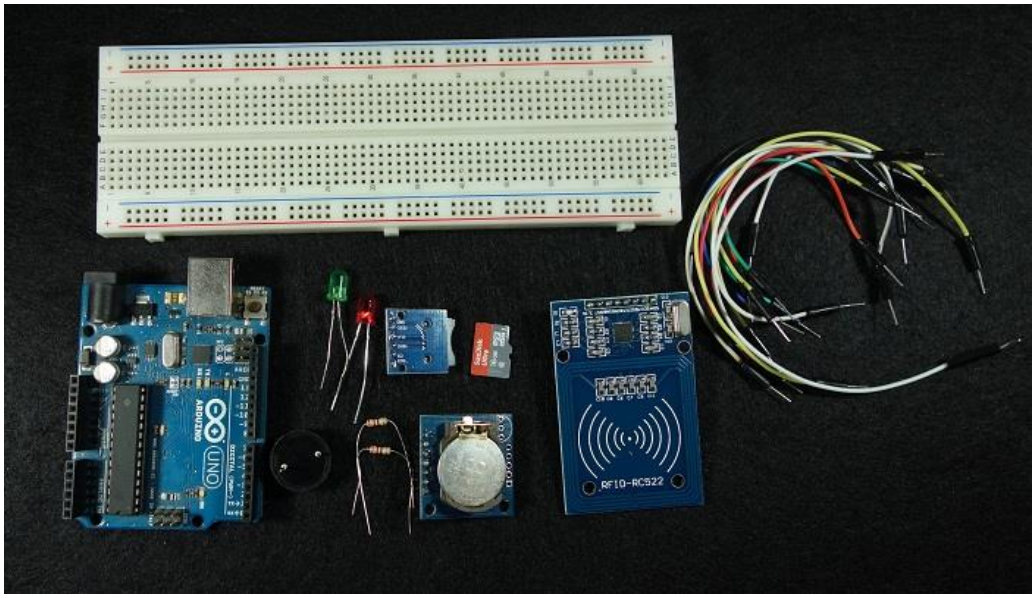
***Note**: different Arduino boards have different I2C pins. If you're using another Arduino board, check the Arduino documentation.*

Installing the RTCLib library

To install the **RTCLib.h** go to **Sketch** > I**nclude library** > **Manage libraries** and search for **RTCLib** or follow the next steps:

1. Click here to download the RTCLib library. You should have a *.zip* folder in your Downloads folder.
2. Unzip the *.zip* folder and you should get **RTCLib-master folder**
3. Rename your folder from ~~**RTCLib-master**~~ to **RTCLib**
4. Move the **RTCLib** folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

Parts required



Grab all the components needed for this project. If you don't have any of them, just click on the link to buy it on eBay.

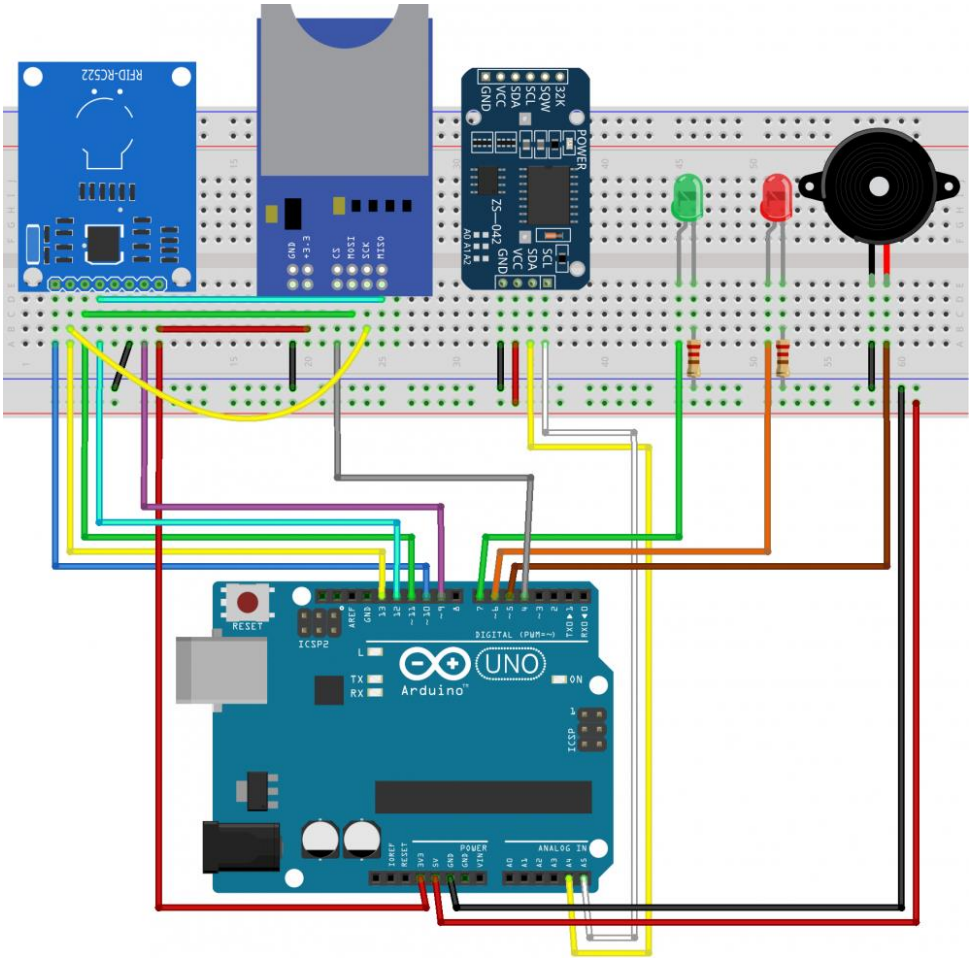
Figure	Name	eBay
	Arduino UNO	http://ebay.to/1SQda0R
	RFID reader + tags	http://ebay.to/1YwzoUB
	SD card module	http://ebay.to/2v5NF5D
	Micro SD Card	http://ebay.to/2wBwM3j
	SD1307 RTC module	http://ebay.to/2d3fGRl
	1x Red LED, 1x Green LED	http://ebay.to/20H2Oyy
	Buzzer	http://ebay.to/2dcur1W

	2x 220 Ohm Resistor	http://ebay.to/1KsMYFP
	Breadboard	http://ebay.to/21bEojM
	Jumper Wires	http://ebay.to/1PXeaJz

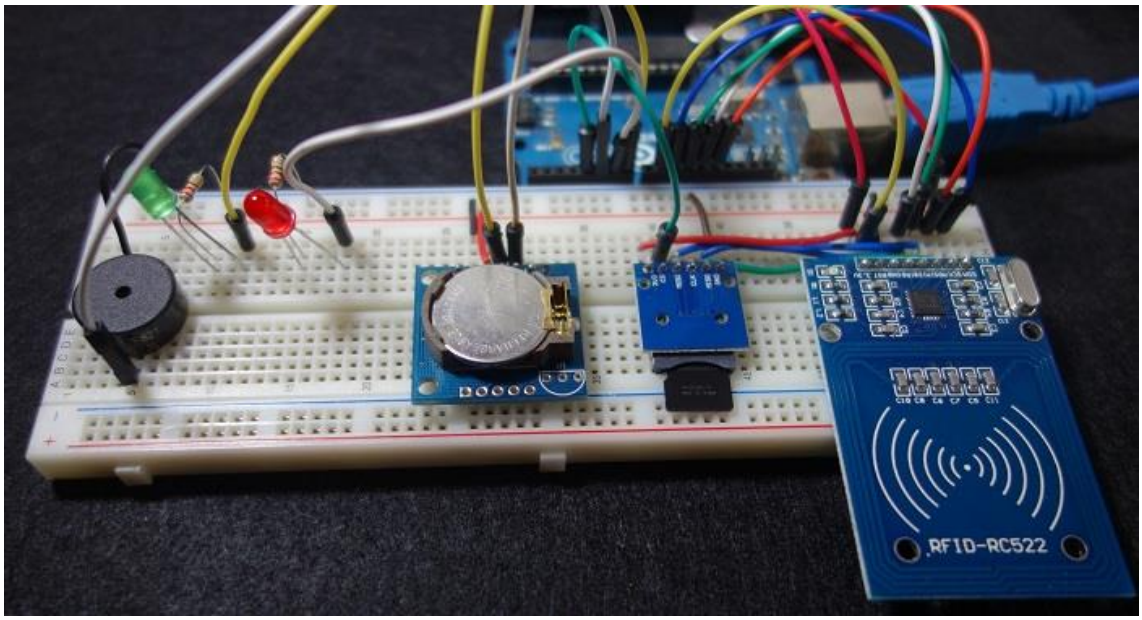
Schematics

The circuit for this project is shown in the circuit schematics below.

In this circuit there are 3.3V and 5V devices, make sure you wire them correctly. Also, if you're using different modules, check the recommend voltage before powering the circuit. Wire one module at a time and follow the pinout tables if needed.



Here's how your circuit should look like after assembling.



Code

Upload the following code to your Arduino. Make sure you have the right Board and COM Port selected.

Check if you have the needed libraries installed.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

#include <MFRC522.h> // for the RFID
#include <SPI.h> // for the RFID and SD card module
#include <SD.h> // for the SD card
#include <RTClib.h> // for the RTC

// define pins for RFID
#define CS_RFID 10
#define RST_RFID 9
// define select pin for SD card module
#define CS_SD 4

// Create a file to store the data
File myFile;

// Instance of the class for RFID
MFRC522 rfid(CS_RFID, RST_RFID);

// Variable to hold the tag's UID
String uidString;
```

```
// Instance of the class for RTC
RTC_DS1307 rtc;

// Define check in time
const int checkInHour = 9;
const int checkInMinute = 5;

//Variable to hold user check in
int userCheckInHour;
int userCheckInMinute;

// Pins for LEDs and buzzer
const int redLED = 6;
const int greenLED = 7;
const int buzzer = 5;

void setup() {

  // Set LEDs and buzzer as outputs
  pinMode(redLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(buzzer, OUTPUT);

  // Init Serial port
  Serial.begin(9600);
  while(!Serial); // for Leonardo/Micro/Zero

  // Init SPI bus
  SPI.begin();
  // Init MFRC522
  rfid.PCD_Init();

  // Setup for the SD card
  Serial.print("Initializing SD card...");
  if(!SD.begin(CS_SD)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");

  // Setup for the RTC
  if(!rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while(1);
  }
  else {
    // following line sets the RTC to the date & time this sketch
was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
  if(!rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
  }
}

void loop() {
  //look for new cards
  if(rfid.PICC_IsNewCardPresent()) {
```

```
    readRFID();
    logCard();
    verifyCheckIn();
  }
  delay(10);
}

void readRFID() {
  rfid.PICC_ReadCardSerial();
  Serial.print("Tag UID: ");
  uidString = String(rfid.uid.uidByte[0]) + " " +
String(rfid.uid.uidByte[1]) + " " +
    String(rfid.uid.uidByte[2]) + " " +
String(rfid.uid.uidByte[3]);
  Serial.println(uidString);

  // Sound the buzzer when a card is read
  tone(buzzer, 2000);
  delay(100);
  noTone(buzzer);

  delay(100);
}

void logCard() {
  // Enables SD card chip select pin
  digitalWrite(CS_SD,LOW);

  // Open file
  myFile=SD.open("DATA.txt", FILE_WRITE);

  // If the file opened ok, write to it
  if (myFile) {
    Serial.println("File opened ok");
    myFile.print(uidString);
    myFile.print(", ");

    // Save time on SD card
    DateTime now = rtc.now();
    myFile.print(now.year(), DEC);
    myFile.print('/');
    myFile.print(now.month(), DEC);
    myFile.print('/');
    myFile.print(now.day(), DEC);
    myFile.print(',');
    myFile.print(now.hour(), DEC);
    myFile.print(':');
    myFile.println(now.minute(), DEC);

    // Print time on Serial monitor
    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
```

```
    Serial.println(now.minute(), DEC);
    Serial.println("sucessfully written on SD card");
    myFile.close();

    // Save check in time;
    userCheckInHour = now.hour();
    userCheckInMinute = now.minute();
  }
  else {
    Serial.println("error opening data.txt");
  }
  // Disables SD card chip select pin
  digitalWrite(CS_SD,HIGH);
}

void verifyCheckIn(){
  if((userCheckInHour <
checkInHour)||((userCheckInHour==checkInHour) && (userCheckInMinute
<= checkInMinute))){
    digitalWrite(greenLED, HIGH);
    delay(2000);
    digitalWrite(greenLED,LOW);
    Serial.println("You're welcome!");
  }
  else{
    digitalWrite(redLED, HIGH);
    delay(2000);
    digitalWrite(redLED,LOW);
    Serial.println("You are late...");
  }
}
```

Source code

https://github.com/RuiSantosdotme/Arduino-Projects/blob/master/code/25_Time_Attendance_System_with_RFID.ino

Importing libraries

The code starts by importing the needed libraries. The **MFRC522** for the RFID reader, the **SD** for the SD card module and the **RTClib** for the RTC. You also include the **SPI** library for SPI communication with the RFID and SD card module.

```
#include <MFRC522.h> // for the RFID
#include <SPI.h> // for the RFID and SD card module
#include <SD.h> // for the SD card
#include <RTClib.h> // for the RTC
```

Preparing RFID reader, SD card and RTC

Then, you define the pins for the RFID reader and the SD card module. For the RFID, the SCK pin (**CS_RFID**) is connected to pin 10 and the RST pin (**RST_RFID**) is connected to pin 9. For the SD card module, the Chip Select pin (**CS_SD**) is connected to pin 4.

```
// define pins for RFID
#define CS_RFID 10
#define RST_RFID 9
// define chip select pin for SD card module
#define CS_SD 4
```

You create a File called **myFile** to store your data.

```
File myFile;
```

Then, you create an instance for the RFID and for the RTC:

```
// Instance of the class for RFID
MFRC522 rfid(CS_RFID, RST_RFID);
// Instance of the class for RTC
RTC_DS1307 rtc;
```

Variables

You create a string variable **uidString** that holds the UID tags.

```
String uidString;
```

The following lines create variables to define the check in time hour and minute. In this case, we're defining the check in hour to 9h05m AM. You can change the check in time by changing these values:

```
// Define check in time
const int checkInHour = 9;
const int checkInMinute = 5;
```

You also need to create variables to hold the user's check in hour. These variables will save the hour a certain UID tag was read. The following variables hold the check in hour and the check in minute.

```
//Variable to hold user check in
int userCheckInHour;
int userCheckInMinute;
```

Finally, you attribute the pin numbers to the LEDs and buzzer.

```
// Pins for LEDs and buzzer
const int redLED = 6;
const int greenLED = 7;
const int buzzer = 5;
```

setup()

Next, in the **setup()** you set the LEDs and buzzer as outputs.

```
// Set LEDs and buzzer as outputs
pinMode(redLED, OUTPUT);
pinMode(greenLED, OUTPUT);
pinMode(buzzer, OUTPUT);
```

After that, each module is initialed.

Functions

In this code you create 3 functions: **readRFID()**, **logCard()** and **verifyCheckIn()**.

The **readRFID()** function reads the tag UID, saves it in the **uidString** variable and displays it on the serial monitor. Also, when it reads the tag, the buzzer makes a beep sound.

The **logCard()** function creates a file on your SD card called **DATA.txt**. You can edit the name of the file, if you want, on the following line.

```
myFile=SD.open("DATA.txt", FILE_WRITE);
```

Then, it saves the **uidString** (that holds the UID of the tag) on the SD card and the current time.

```
myFile.print(uidString);
// Save time on SD card
```

```
DateTime now = rtc.now();
myFile.print(now.year(), DEC);
myFile.print('/');
myFile.print(now.month(), DEC);
myFile.print('/');
myFile.print(now.day(), DEC);
myFile.print(',');
myFile.print(now.hour(), DEC);
myFile.print(':');
myFile.print(now.minute(), DEC);
```

It also saves the user check in hour and minute in the following variables for further comparison with the predefined check in time.

```
userCheckInHour = now.hour();
userCheckInMinute = now.minute();
```

The **verifyCheckIn()** function simply compares the user check in time with the predefined check in hour and gives feedback accordingly. If the user is late, the red LED lights up; if the user is on time, the green LED lights up.

loop()

After studying the created functions, the **loop()** is pretty straightforward to understand.

First, the code checks if an RFID tag was swiped. If yes, it will read the RFID UID, log the UID and the time into the SD card, and then it will give feedback to the user by lighting up one of the LEDs.

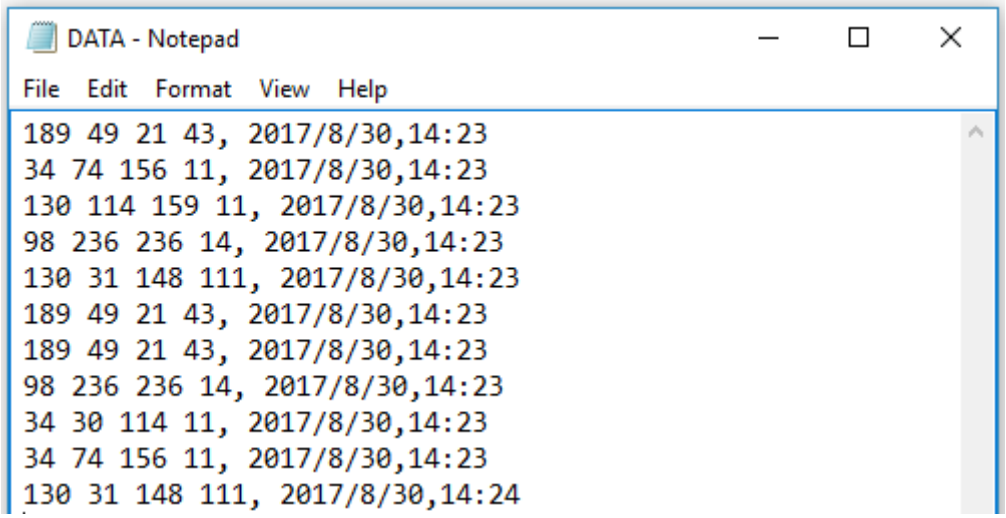
Grabbing the SD Card data

To check the data saved on the SD card, remove it from the SD card module and insert it on your computer.

Open the SD card folder and you should have a file called DATA.txt.

DATA

Open the file using a text editor. You'll have something as follows:



Notice that each value is separated by commas. This makes it easier if you want to import this data to Excel or other data processing software.

Wrapping up

In this project you learned how to use an RFID card reader and the SD card module with Arduino. You can modify this project to your own needs and you can use the functions created here in other projects that require data logging or reading RFID tags.

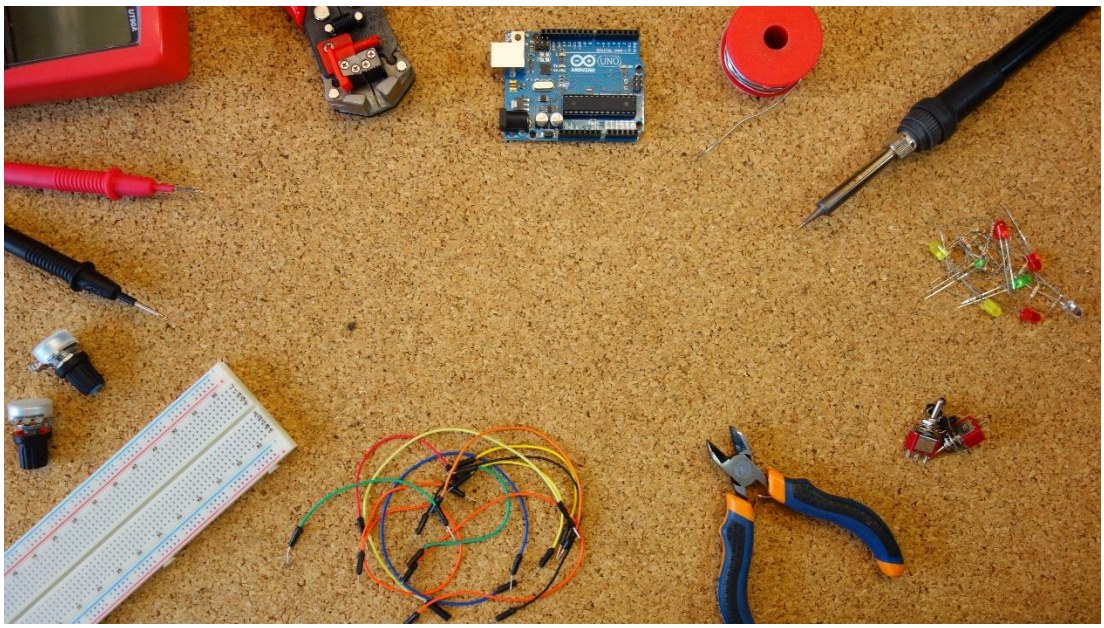
You can take this project further and add a display to give extra feedback to the user. You can associate a name with each UID and display the user name when the tag is read.

Have fun with your projects!

Final Thoughts

Congratulations for completing this course!

If you followed all the projects presented in this eBook you are practically an Arduino master.



Let's see the most important things that you've accomplished. You know how to:

- Control outputs: LEDs, relays, buzzer, etc. ;
- Read sensors: LDR, ultrasonic sensor, sound sensor, temperature and humidity sensor, barometric sensor, etc. ;
- Use the OLED display and the LCD;
- Add awesome features to the Arduino by using the Ethernet shield, data logging shield, GSM shield, RFID reader and bluetooth module;
- And much more!

Now feel free to change the projects and build up on the snippets of code presented in this eBook to fit your own projects!

We hope you had fun following all these projects! If you have something that you would like to share let us know in the Facebook group (Join the Facebook group here).

Good luck with all your projects,

Rui Santos and Sara Santos

Download other RNT products

Random Nerd Tutorials is an online resource with electronics projects, tutorials and reviews.

Creating and posting new projects takes a lot of time. At this moment, Random Nerd Tutorials has nearly 100 free blog posts with complete tutorials using open-source hardware that anyone can read, remix and apply to their own projects: http://randomnerdtutorials.com

To keep free tutorials coming, there's also paid content or as we like to call "Premium Content".

To support Random Nerd Tutorials you can download Premium content here. If you enjoyed this eBook make sure you check all the others.

Android Apps for Arduino with MIT App Inventor 2

The Android Apps for Arduino with MIT App Inventor 2 is a practical course in which you're going to build 8 Android Apps to interact with the Arduino. Read full product description.

Home Automation Using ESP8266 (2nd Edition)

This eBook is my step-by-step guide designed to help you get started with this amazing $4 WiFi module called ESP8266.

If you're new to the world of ESP8266, this eBook is perfect for you. If you already used the ESP8266 before, I'm sure you'll also learn something new.

This eBook contains the information you need to get up to speed quickly and start your own venture with the ESP8266 applied to Home Automation! Read full product description.

Build a Home Automation System for $100

Learn **Raspberry Pi**, **ESP8266**, **Arduino** and **Node-RED**.

This is a premium step-by-step course to get you building a real world home automation system using open-source hardware and software. Read full product description.

Thanks for taking the time to read our work!

Good luck with all your projects,

-Rui Santos

-Sara Santos

P.S. Click here for more Courses and eBooks like this one.