



# Building Leaders for Advancing Science and Technology (BLAST)

## Internet of Things Hands-on Activities with Arduino

**Dr. Murat Kuzlu**  
**Department of Engineering Technology**

# Outline

- Internet
  - Internet of Things (IoT)
  - Arduino
  - ThingSpeak
  - Hands-on Activities



<https://medium.com/@otavioguastamacchia/creating-a-simple-iot-case-8102f22908a7>

# What is the Internet?



<https://www.connexusuk.com/high-speed-internet/>



- The Internet is a global **web of computers connected to each by communication lines** (mostly phone lines).

- If you look at a map of big cities, smaller towns, and scattered houses, each is connected together with roads, railways, etc. This is similar to the Internet, except with the Internet, wires connect computers.

The Internet is a superhighway.

<http://mediatechnologyeducation.pbworks.com/w/page/20693030/The%20Information%20Superhighway>

# Some ways to use the Internet

- Surfing
- E-mail
- Social media
- Shopping
- News
- Games
- **REMOTE MONITORING and CONTROL**



# Why Internet is Important

- Data, data, data!
- Modern organizations rely on the efficient transmission of data
- Enables distributed systems, **real-time communication**, electronic commerce, social media, and the Web

<https://makeawebsitehub.com/social-media-sites/>

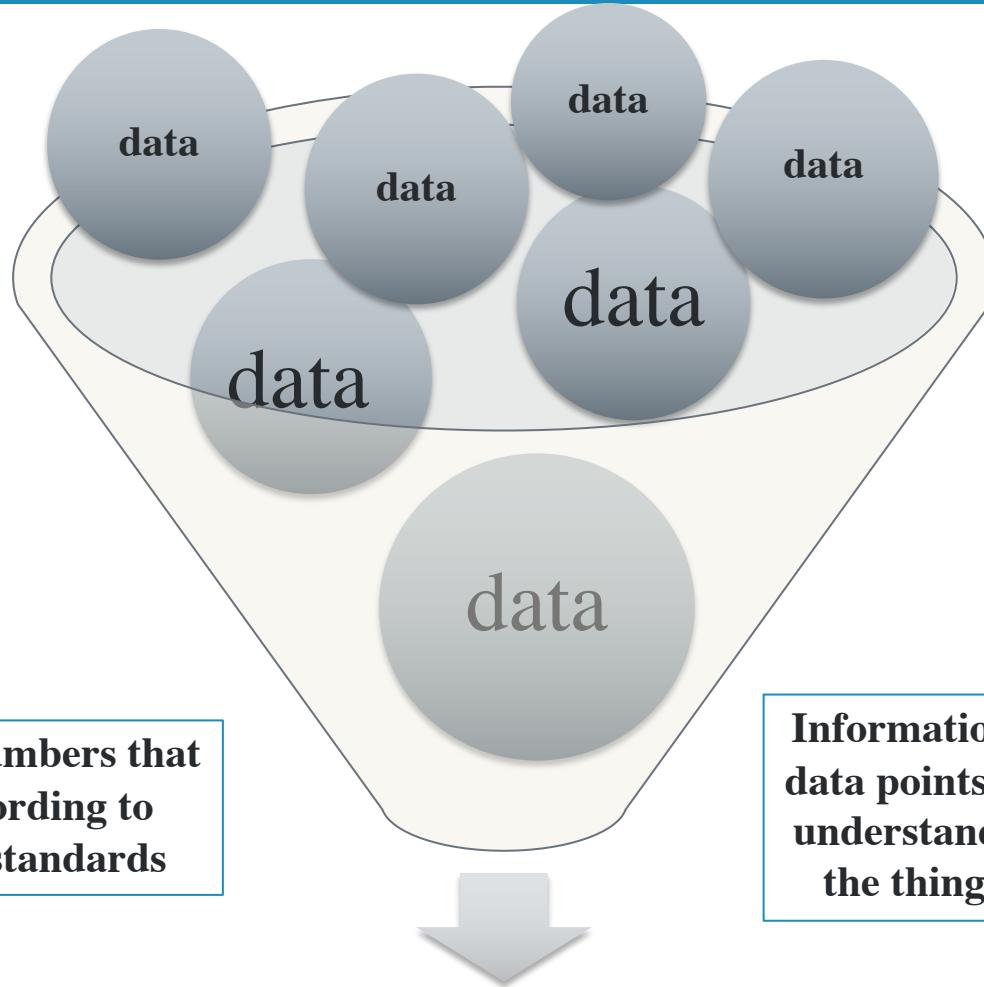


<https://www.edx.org/course/social-media-how-media-got-social>



Ref: Taylor M. Wells: College of Business Administration, California State University, Sacramento  
Copyright © 2015 John Wiley & Sons, Inc. All rights reserved

# Data vs. Information



**Data are the facts or details from which information is derived**

# Trends – WoT and IoT

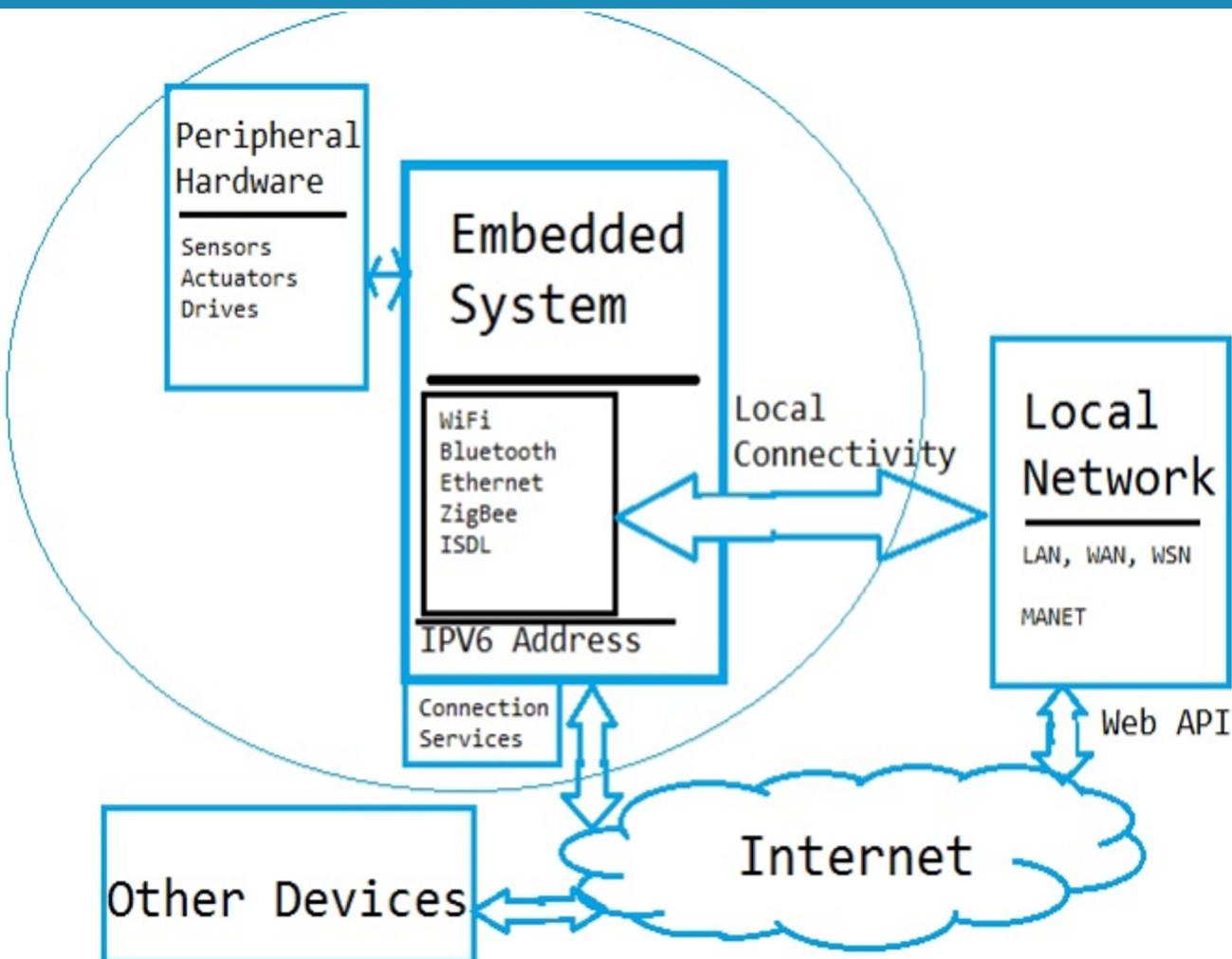
- The Web of Things
  - Everything connects to the network!
    - e.g., cars, refrigerators, thermostats, shoes, doors, etc.
  - Networks need to support the increased demands of these devices
- The Internet of Thing (IoT)
  - *The network of physical objects*—devices, vehicles, buildings and other items-- *embedded with electronics, software, sensors, and network connectivity*—that enables these objects *to collect and exchange data.*"



# The Internet of Thing (IoT)

*IoT*

- *Network of physical objects*
- *Embedded System*
- *Network connectivity*
- *Collect and exchange data*



# IoT Examples

Examples of objects that can fall into the scope of Internet of Things include connected security systems, sensors, thermostats, cars, electronic appliances, light in the household and commercial environments, alarm clocks, speaker systems, vending machines and more.



<https://www.expressvpn.com/blog/what-is-the-internet-of-things-iot/>

# Internet-connected devices

John Romkey's Toaster (1990, Ethernet)

Ambient Orb (2002, via pager network)

**iPod (2001), iTunes Store (2003, via USB/PC)**

Nike+ iPod (2006), Bracelet (2008 via USB/PC)

Rafi Haladjian's Nabaztag (2006, Wifi)

Rob Faludi's Botanicalls (2006, Ethernet)

Schulze&Webb Availabot (2006, via USB/PC)

**iPhone (2007, GSM)**

Amazon Kindle (2007, 3G)

Wafaa Bilal's Shoot an Iraqi (2007, ?)

Withings BodyScale (2008, Wifi)

Vitality GlowCap (2008, Wifi; 2011, 3G)

BakerTweet (2009, 3G)

Adrian McEwen's Bubblino (2009, Ethernet)

David Bowen's Telepresent Water (2011, ?)

**Nest Thermostat (2011, Wifi)**

BERG's Little Printer (2011, ?)

Supermechanical's Twine (2012, Wifi)

Olly & Polly (2012, via USB/PC)

Koubachi Sensor (2012, Wifi)

Descriptive Camera (2012, Ethernet)

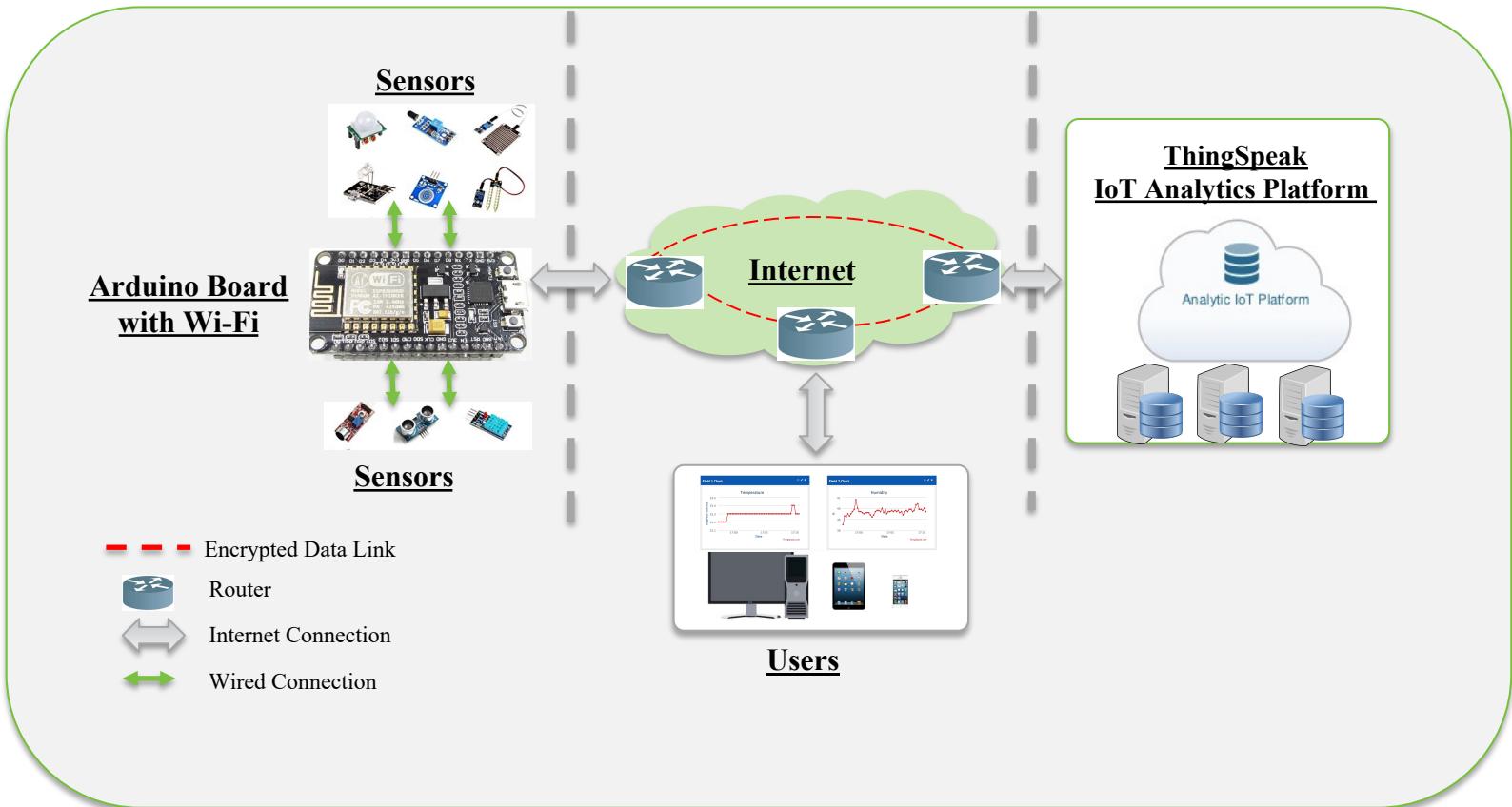
.....

# The Future of IoT

As far as the reach of the IoT, there are more than 12 billion devices that can currently connect to the Internet, and it is expected that by 2020 there will be 26 times more connected things than people.



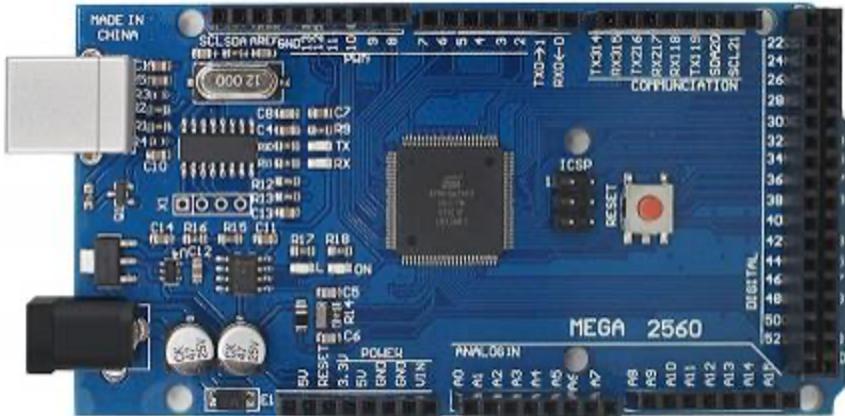
# Architecture of an IoT-based System



# Why Arduino ?

- It is an **open-source project**, software/hardware is extremely **accessible** and very flexible to be customized and extended
- It is **flexible**, offers a variety of digital and analog inputs, *SPI* and serial interface and digital and *PWM* outputs
- It is **easy to use**, connects to computer via USB and communicates using standard serial protocol, runs in standalone mode and as interface connected to PC/Macintosh computers
- It is **inexpensive**, and comes with free authoring software
- Arduino is backed up by a growing **online community**, lots of source code is already available and we can share and post our examples for others to use, too.

# Arduino

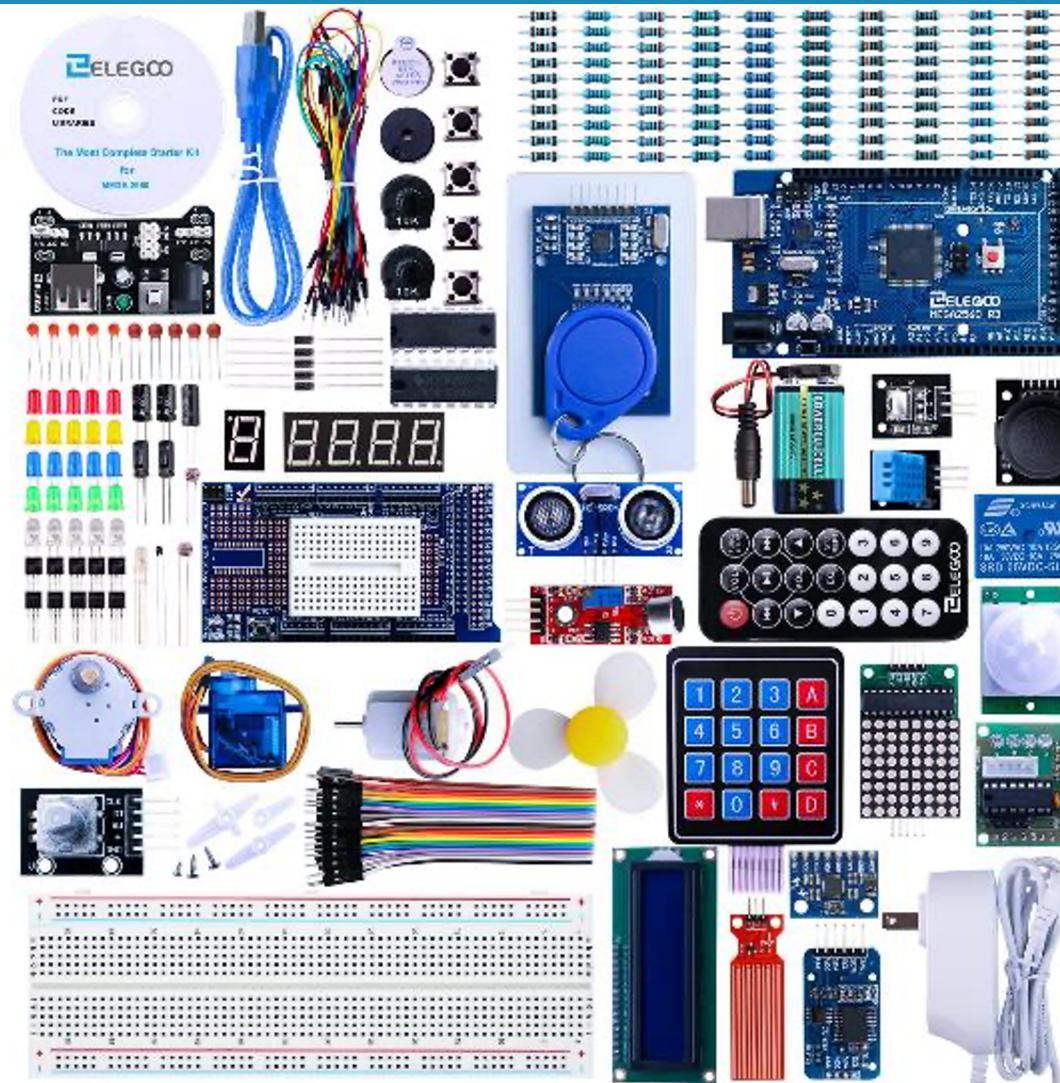


Mega2560 R3 ATmega2560-16AU  
CH340 Development Board



ESP8266 NodeMCU LUA  
CP2102 ESP-12E

# Arduino



Elegoo EL-KIT-008 Mega 2560 Project  
The Most Complete Ultimate Starter Kit

# Arduino

Servo Motor  
(SG90)  
1PC

ULN2003 Stepper  
Motor Driver  
Board  
1PC

Power Supply  
Module  
1PC

Mega 2560  
Controller Board  
1PCS

GY-521  
Module  
1PC

LCD 1602 Module  
(with pin header)  
1PC

IR Receiver  
Module  
1PC

Joystick Module  
1PC

DHT11 Temperature  
and Humidity  
Module  
1PC

Rc522  
RFID Module  
1PC

Stepper Motor  
1PC

Prototype Expansion  
Board  
1PC

Rotary Encoder  
Module  
1PC

Ultrasonic Sensor  
1PC

Ds3231 RTC  
Module  
1PC

Red LED  
5PCS

Yellow LED  
5PCS

Blue LED  
5PCS

HC-SR04  
Motion Sensor  
1PC

Sound Sensor  
Module  
1PC

Water Lever Detection  
Sensor Module  
1PC

Photoresistor  
(Photocell)  
2PCS

Green LED  
5PCS

RGB LED  
1PC

White LED  
5PCS

22pF Ceramic  
Capacitor  
5PCS

104pF Ceramic  
Capacitor  
5PCS

# Installing IDE

- The Arduino Integrated Development Environment



## ARDUINO 1.8.0

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions.

**Windows Installer**  
**Windows ZIP file for non admin install**

**Windows app**

**Mac OS X 10.7 Lion or newer**

**Linux 32 bits**  
**Linux 64 bits**  
**Linux ARM**

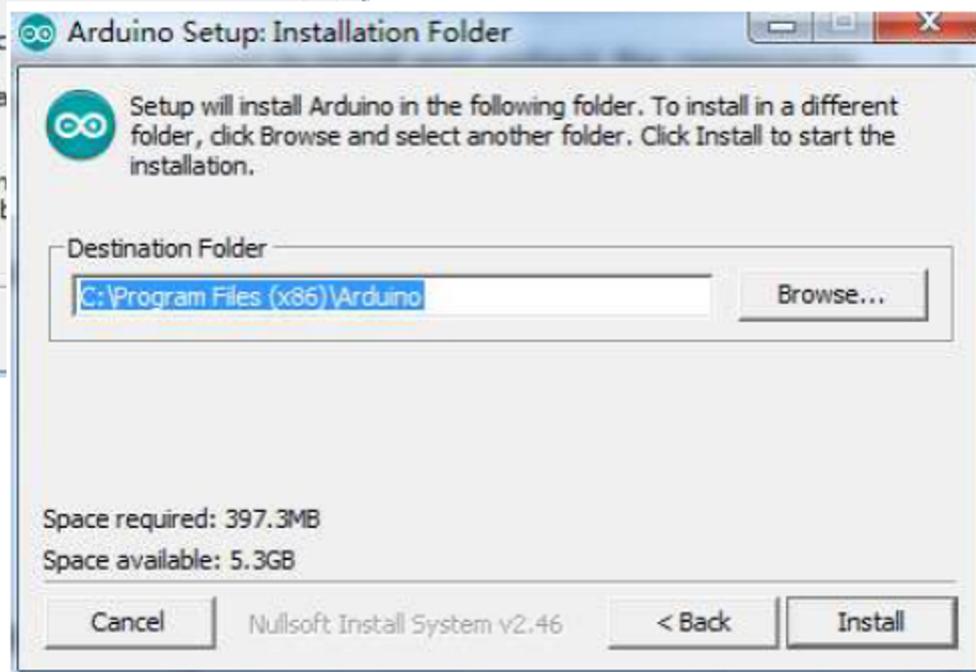
[Release Notes](#)  
[Source Code](#)  
[Checksums \(sha512\)](#)

**The version available at this website is usually the latest version, and the actual version may be newer than the version in the picture.**

# Installing Arduino (Windows)



Click Agree and Install...

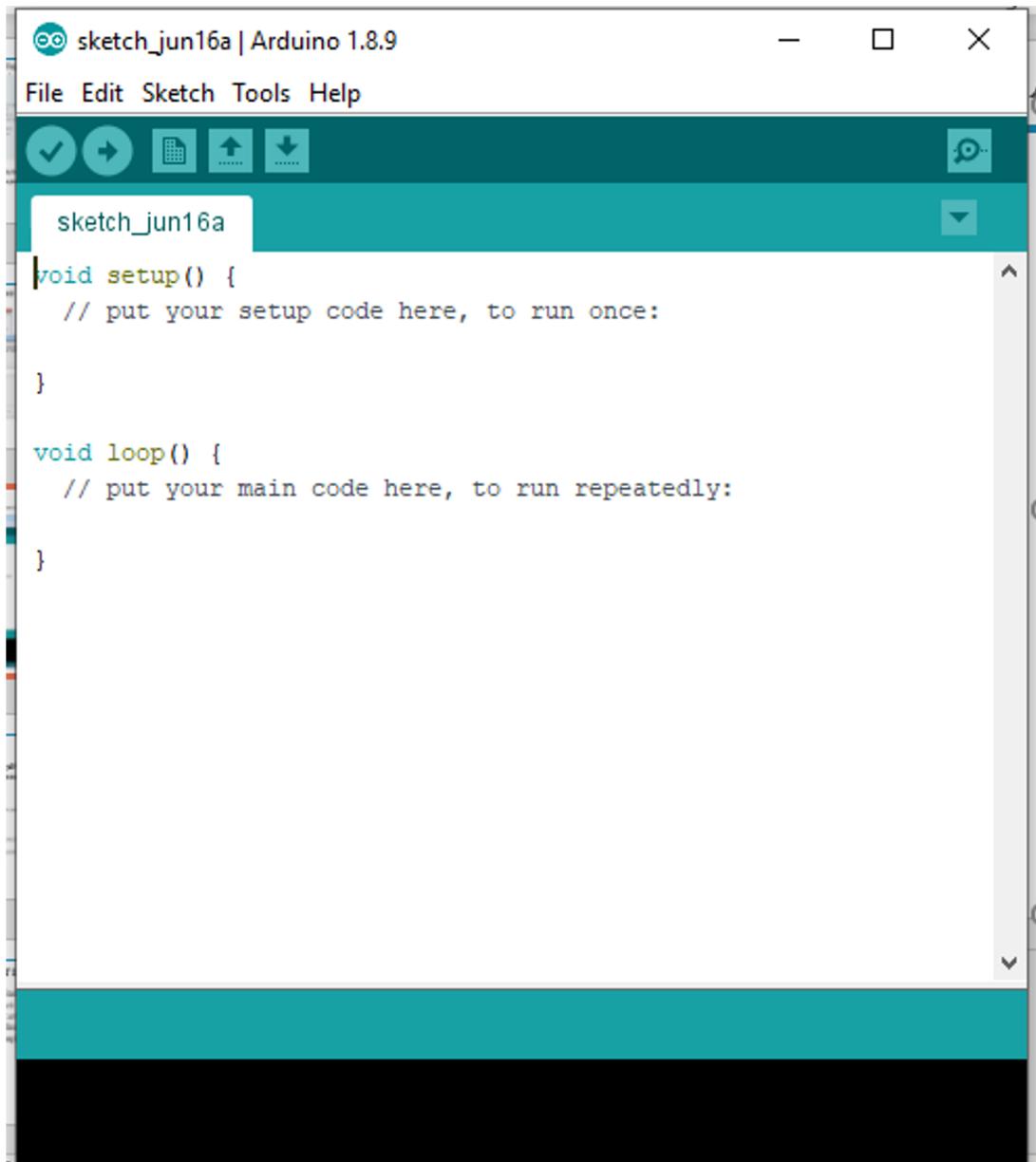


The following Icon appears on the desktop



# Arduino IDE

Double-click to enter the  
desired development  
environment



# Add Libraries and Open Serial Monitor

## What are Libraries?

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in the reference. To use the additional libraries, you will need to install them.

## Arduino Serial Monitor (Windows, Mac, Linux)

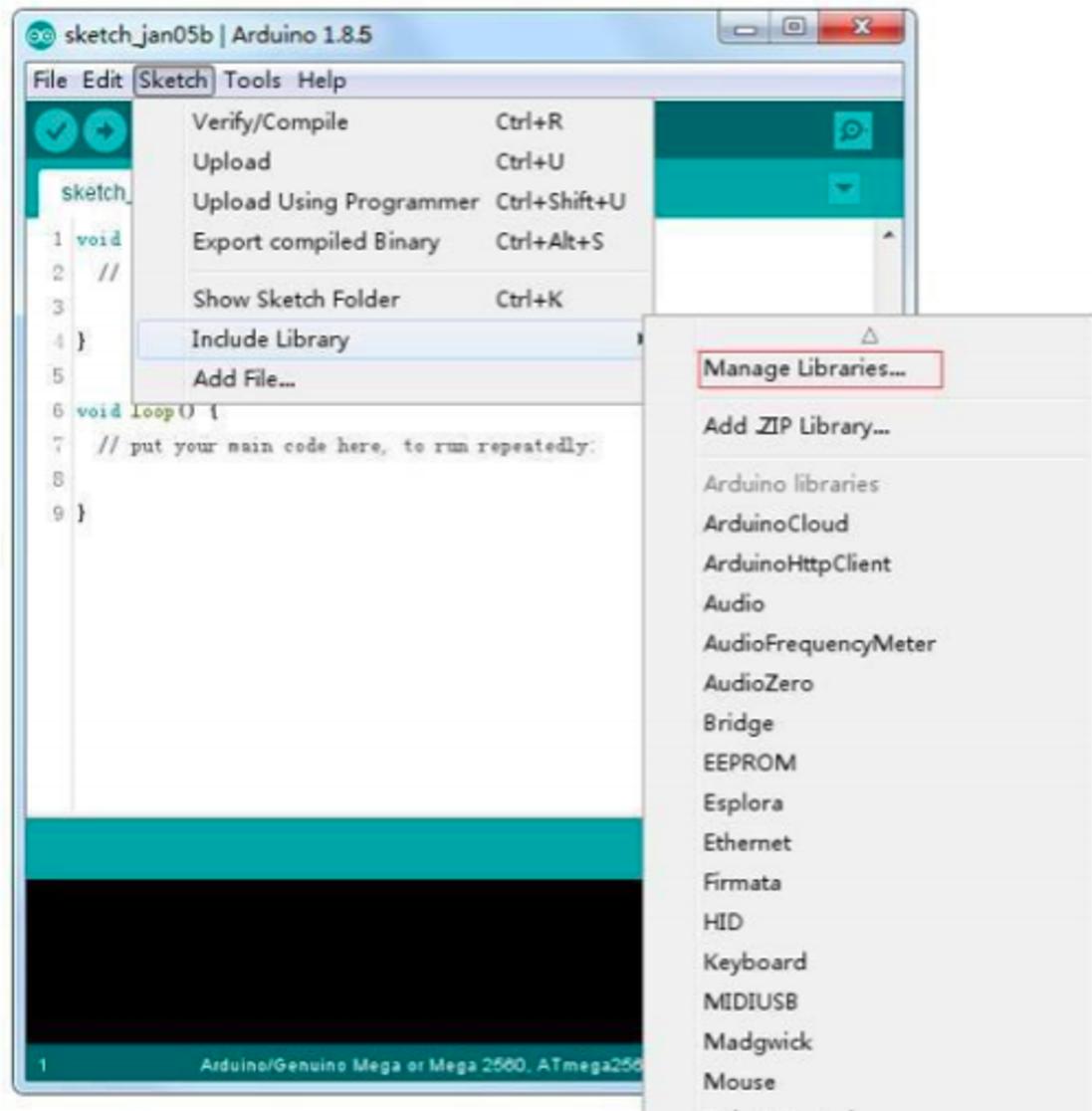
The Arduino Integrated Development Environment (IDE) is the software side of the Arduino platform. And, because using a terminal is such a big part of working with Arduinos and other microcontrollers, they decided to include a serial terminal with the software. Within the Arduino environment, this is called the Serial Monitor.

# Add Libraries and Open Serial Monitor

## How to Install a Library?

Using the Library Manager  
To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.0).

Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



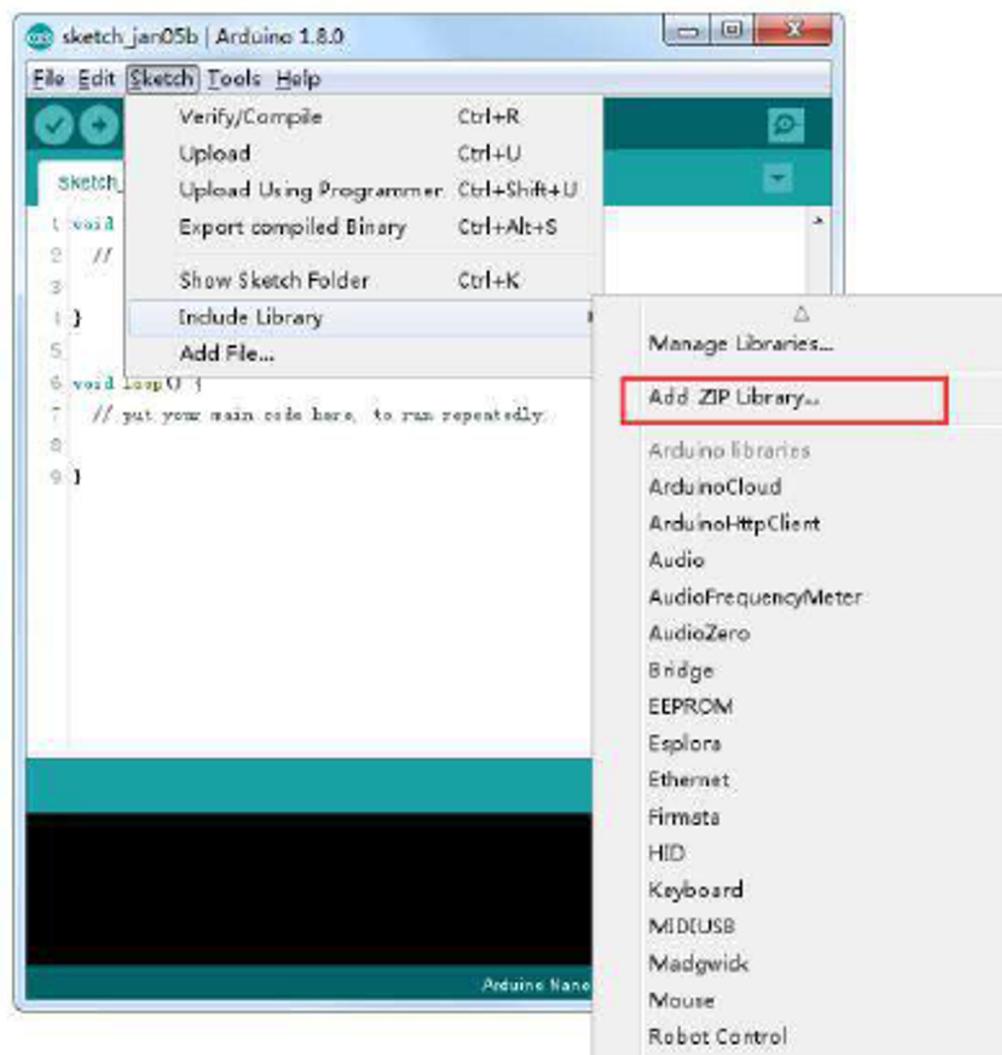
# Add Libraries and Open Serial Monitor

## How to Install a Library?

### Importing a .zip Library.

In the Arduino IDE, navigate to Sketch > Include Library.

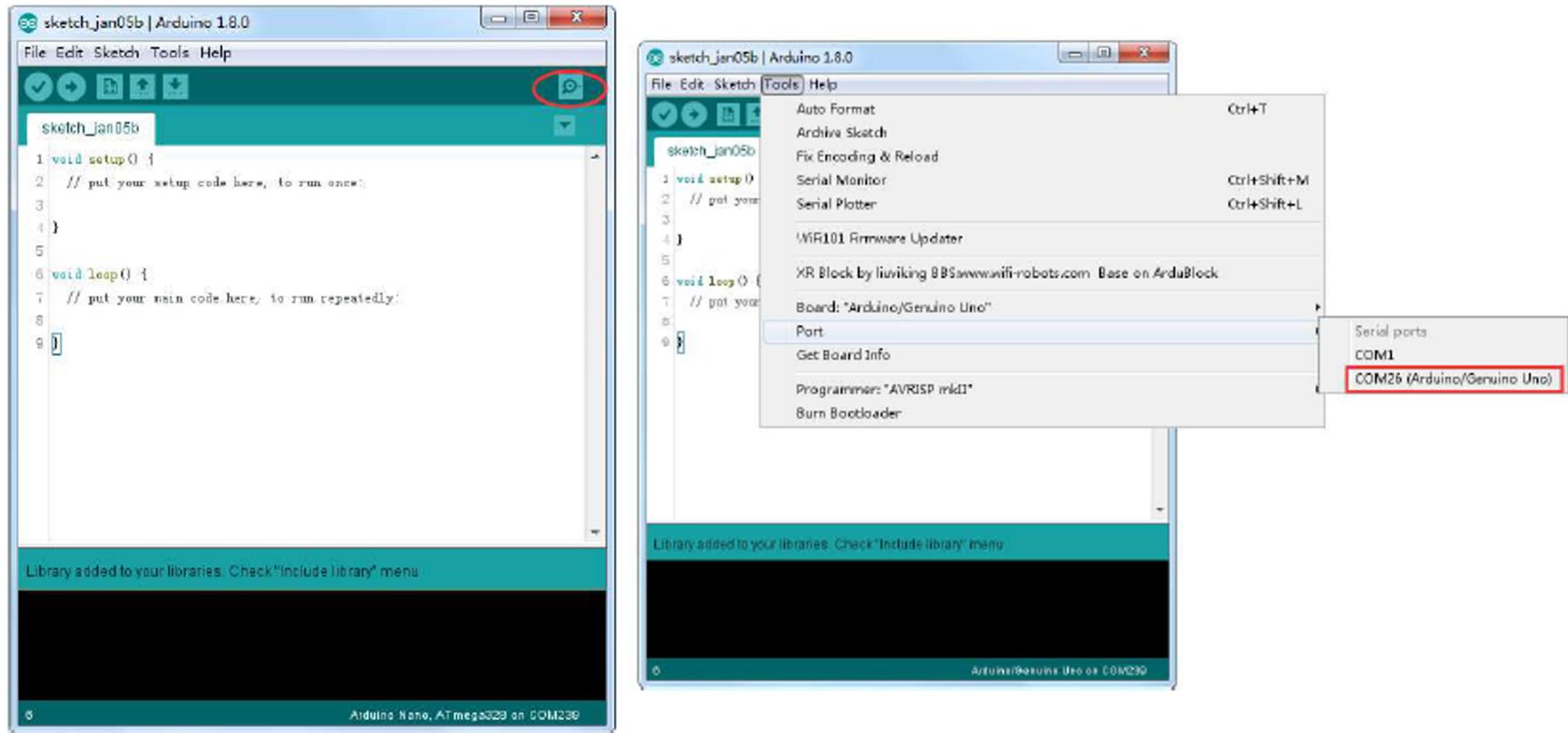
At the top of the drop-down list, select the option to "Add ZIP Library".



# Add Libraries and Open Serial Monitor

## Arduino Serial Monitor (Windows, Mac, Linux) and Make a Serial Connection

The Arduino Integrated Development Environment (IDE) is the software side of the Arduino platform. And, because using a terminal is such a big part of working with Arduinos and other microcontrollers, they decided to include a serial terminal with the software. Within the Arduino environment, this is called the Serial Monitor.

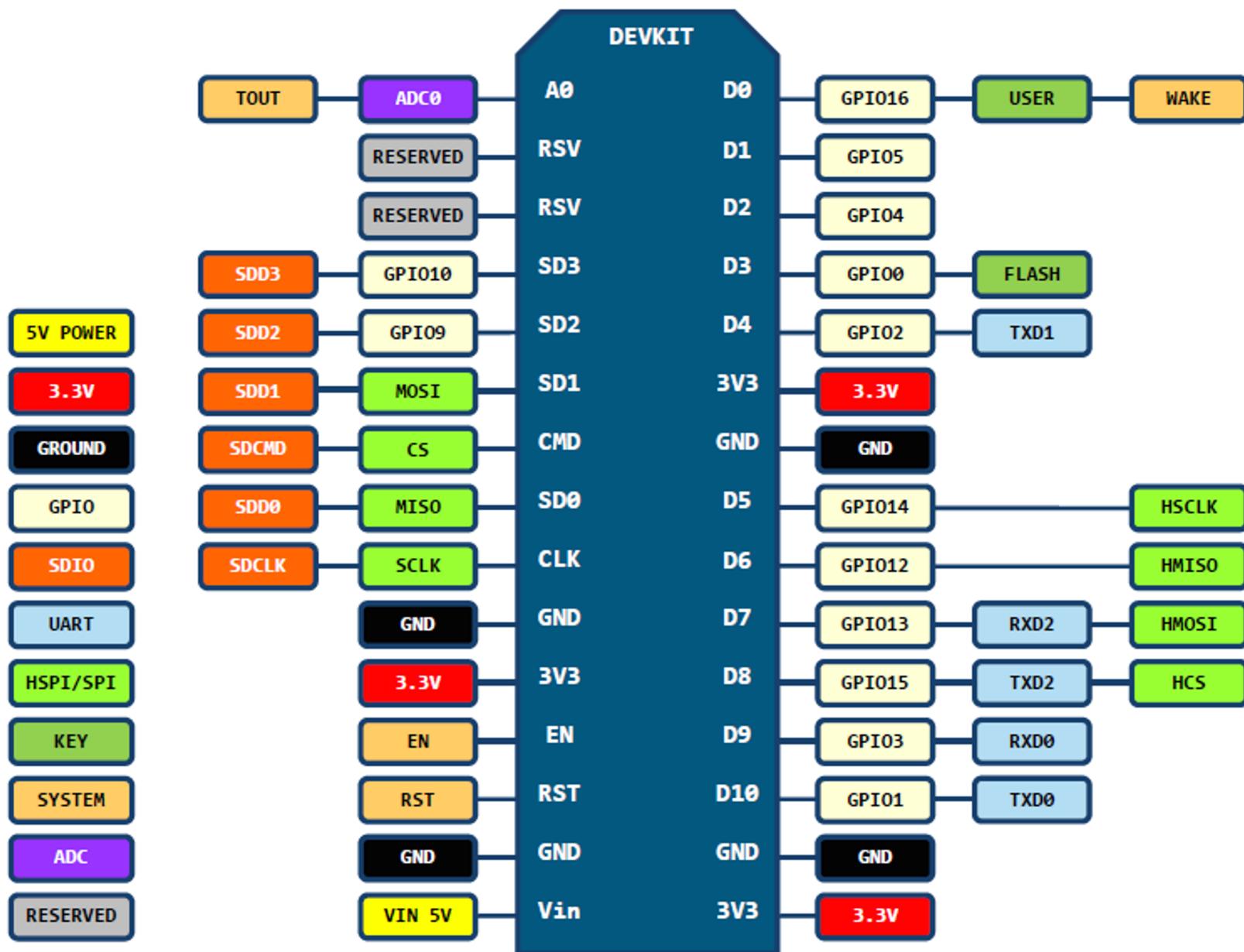


# NodeMCU ESP8266 ESP-12E



- Voltage: 3.3V.
- Wi-Fi Direct (P2P), soft-AP.
- Integrated TCP/IP protocol stack.
- GPIOs: 17 (multiplexed with other functions).
- Analog to Digital: 1 input with 1024 step resolution.
- 802.11 support: b/g/n.
- Maximum concurrent TCP connections: 5
- Good tutorial:  
[https://www.handsontec.com/pdf\\_learn/esp8266-V10.pdf](https://www.handsontec.com/pdf_learn/esp8266-V10.pdf)

# NodeMCU ESP8266 ESP-12E Pinout



# NodeMCU setup

- *Additional Board Manager URL:*
  - [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

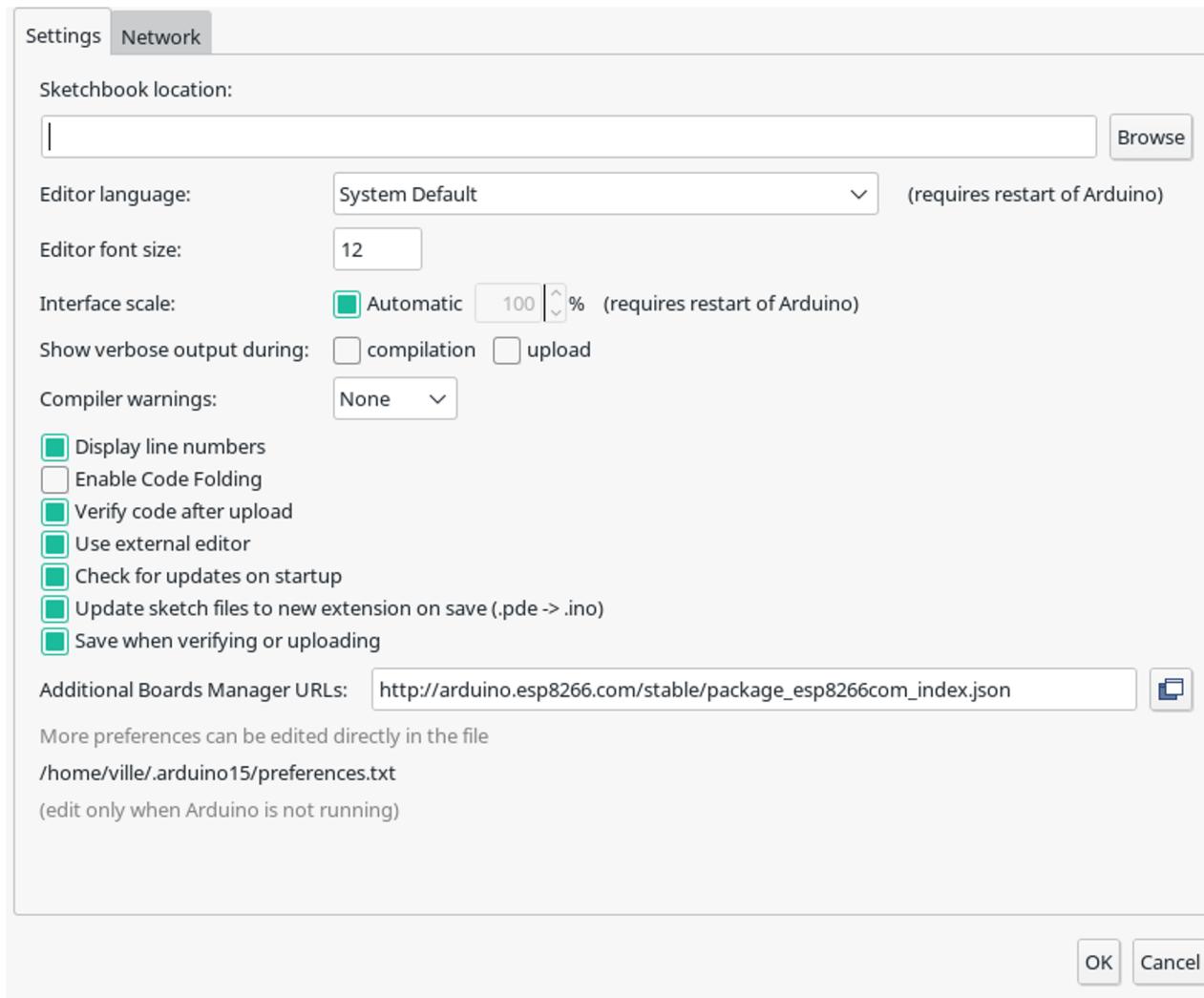
## Installing with Boards Manager

Starting with 1.6.4, Arduino allows installation of third-party platform packages using Boards Manager. We have packages available for Windows, Mac OS, and Linux (32 and 64 bit).

- Install Arduino 1.6.8 from the [Arduino website](#).
- Start Arduino and open Preferences window.
- Enter `http://arduino.esp8266.com/stable/package_esp8266com_index.json` into *Additional Board Manager URLs* field.  
You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and install *esp8266* platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation).

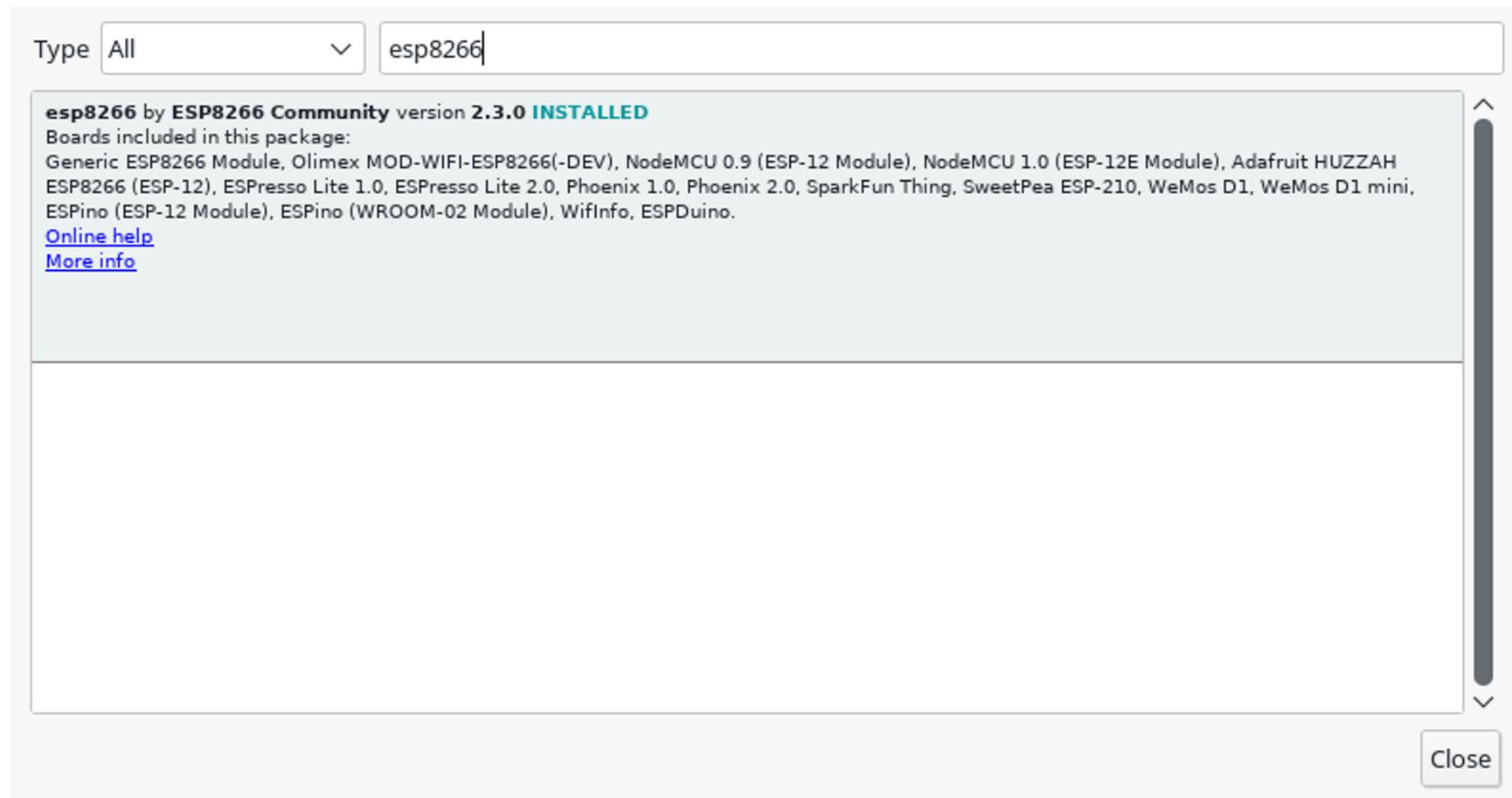
# NodeMCU setup

- Add the link to the *Additional Board Manager URLs*



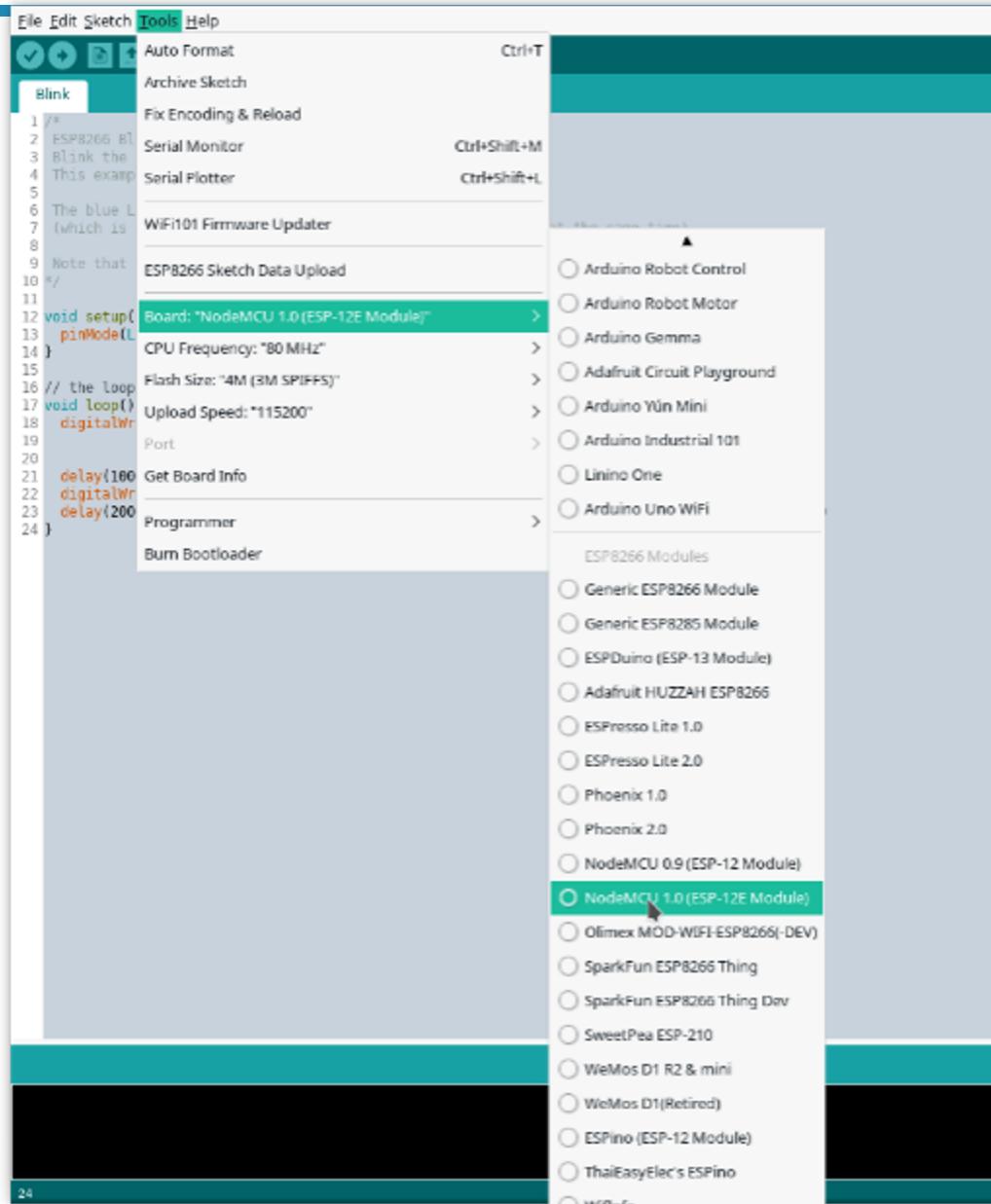
# NodeMCU setup

- Search the board manager for *esp8266*
- Install the library



# NodeMCU setup

- Select the NodeMCU 2.0 as your board
- Additional settings appear under the board menu
- They can be left as they are
- Higher Upload Speed reduces your upload times



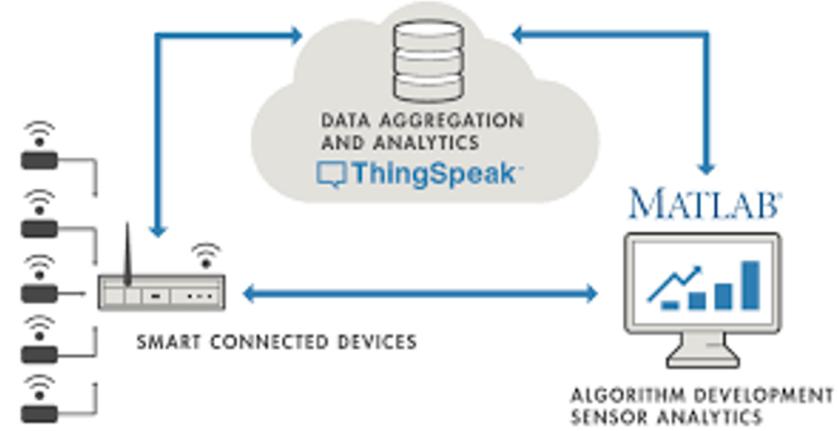
# NodeMCU setup

---

- If the device doesn't appear in the port menu after installing the library and connecting the board you can try installing the USB to Serial chip drivers.
- The chip is the ch340g
- Drivers can be found on the NodeMCU git:  
<https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers>

# ThingSpeak

- ThingSpeak is a Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications.
- It works with Arduino, Raspberry Pi and MATLAB (premade libraries and APIs exists).
- But it should work with all kind of Programming Languages, since it uses a REST API and HTTP.



- <https://thingspeak.com>

# What is ThingSpeak?

- **What is ThingSpeak:**  
ThingSpeak is an IoT analytics platform service that lets you collect and store sensor data in the cloud and develop Internet of Things applications.
- **ThingSpeak Channel:**  
'ThingSpeak Channel' is the core element of the ThingSpeak platform. This channel is used to store the real-time data, or the data transferred through various sensors and embedded systems. Data stored at the channel is further used for analysis and visualization.
- Software Requirement: Internet
- Hardware Requirement: Arduino.



The screenshot shows the ThingSpeak homepage. At the top, there's a navigation bar with links for 'Channels', 'Apps', 'Community', 'Support', 'How to Buy', 'Account', and 'Sign Out'. Below the navigation is a large banner with the text 'Understand Your Things' and 'The open IoT platform with MATLAB analytics.' It features a city skyline background with network connections. Below the banner are three main sections: 'Collect' (Send sensor data privately to the cloud), 'Analyze' (Analyze and visualize your data with MATLAB), and 'Act' (Trigger a reaction). At the bottom, there's a section titled 'ThingSpeak Features' with a list of items like 'Collect data in private channels', 'Share data with public channels', 'RESTful and MQTT APIs', 'MATLAB® analytics and visualizations', 'Alerts', 'Event scheduling', 'App integrations', and 'Worldwide community'. Another section titled 'Works With' lists compatible platforms such as Arduino®, Particle Photon and Electron, ESP8266 WiFi Module, Raspberry Pi®, Mobile and web apps, Twitter®, Twilio®, and MATLAB®.

# Creating A Channel

- Before creating a channel, you need to sign into things speak. You can easily sign in either using your either ThingSpeak account or mathswork account, or create a new mathswork account via following link:
- [https://thingspeak.com/users/sign\\_up](https://thingspeak.com/users/sign_up)
- Login Page
- Email: [mkuzlu@hotmail.com](mailto:mkuzlu@hotmail.com)
- Password: ODU\_Blast2000

Create MathWorks Account

Email Address

Missing required information

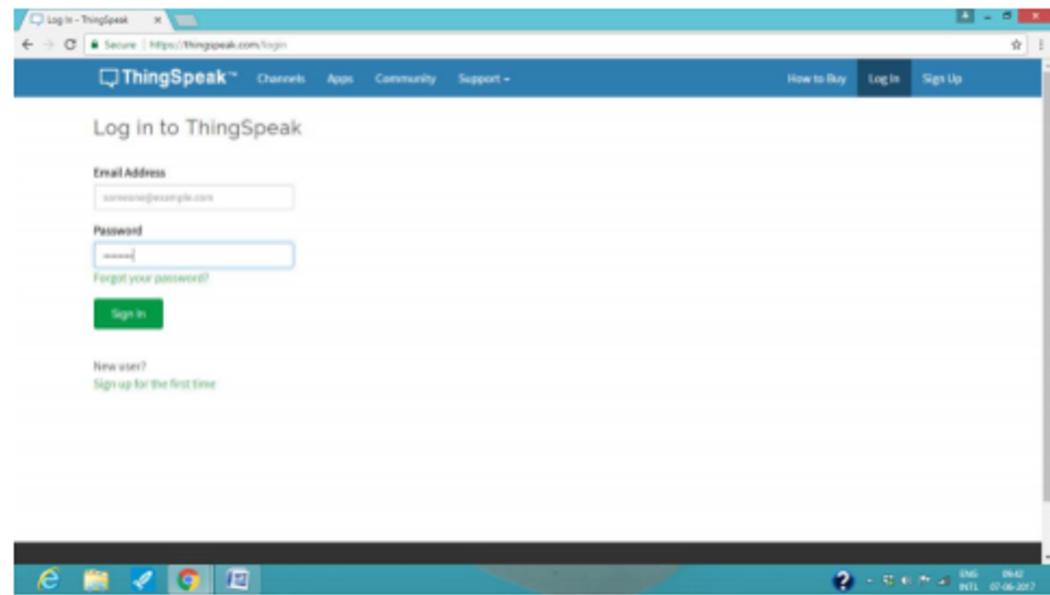
To access your organization's MATLAB license, use your school or work email.

Location

▼

First Name

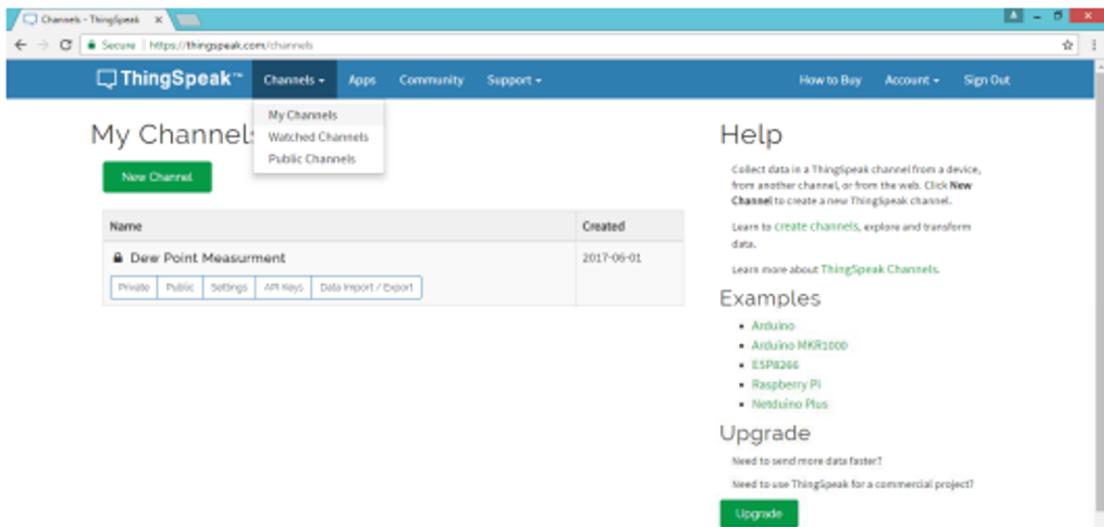
Last Name

The screenshot shows a web browser window titled "Log In - ThingSpeak" with the URL "https://thingspeak.com/login". The page has a blue header with the "ThingSpeak" logo and navigation links for "Channels", "Apps", "Community", and "Support". On the right side of the header, there are buttons for "How to Buy", "Log In", and "Sign Up". Below the header, the main content area is titled "Log in to ThingSpeak". It contains two input fields: "Email Address" with the value "someone@example.com" and "Password" with the value "secret". Below the password field is a link "Forgot your password?". At the bottom of the form is a green "Sign In" button. Further down, there is a "New user?" section with a "Sign up for the first time" link. The browser's taskbar at the bottom shows various pinned icons, and the status bar indicates the date and time as "09-06-2017".

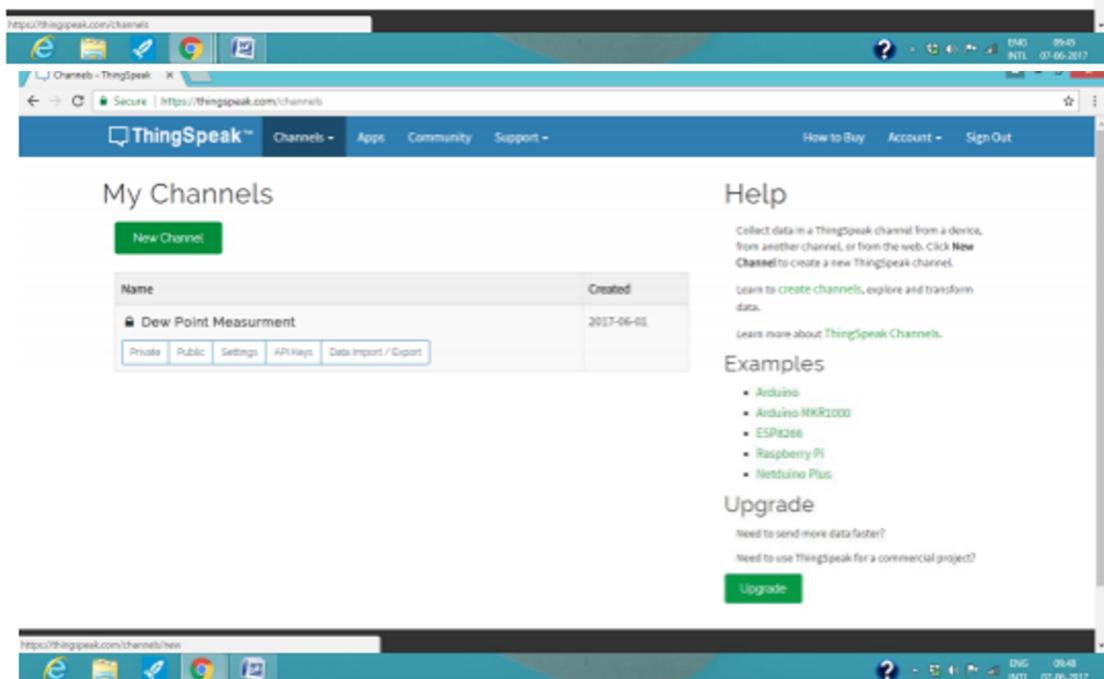
# Creating A Channel

1. Click on the menu bar  
**Channels> My Channels.**



The screenshot shows a web browser window for 'Channels - ThingSpeak'. The address bar shows 'https://thingspeak.com/channels'. The navigation bar includes 'ThingSpeak™', 'Channels', 'Apps', 'Community', 'Support', 'How to Buy', 'Account', and 'Sign Out'. A dropdown menu from the 'Channels' button shows options: 'My Channels', 'Watched Channels', and 'Public Channels'. The main content area is titled 'My Channel' and contains a table with one row. The table columns are 'Name' and 'Created'. The row shows 'Dew Point Measurement' and '2017-06-01'. Below the table are buttons for 'Private', 'Public', 'Settings', 'API Keys', and 'Data Import / Export'. To the right, there's a 'Help' section with links to collect data, create channels, and explore transforms. It also lists examples like Arduino, Raspberry Pi, etc., and a 'Upgrade' section with a green 'Upgrade' button.

2. Now on the channels page click on the button '**New Channel**'.



This screenshot is identical to the one above, showing the 'My Channels' page with a single channel entry for 'Dew Point Measurement'. The interface, including the navigation bar, dropdown menu, and right-hand sidebar with help, examples, and upgrade information, remains the same.

# Creating A Channel

3. New channel page have various text box fields showing the settings of the channel

- a. **Name** -- provide a unique name to your channel.
- b. **Fields** -- Click the check boxes next to the field and then enter the field name.
- c. To make your channel public check the '**Make Public**' check box
- d. Similarly, you can also add the location to your channel by clicking the '**Show Location**' check box
- e. Check the '**Show video**' check box to make the video visible uploaded by you.
- f. Now click the '**Save channel**' button to save your channel

The image displays two screenshots of the ThingSpeak 'New Channel' creation interface. The top screenshot shows the 'Fields' section, where users can define up to eight data fields. In this example, 'Field 3: dew point' is checked, and the field name is 'dew point'. The bottom screenshot shows the 'Location' and 'Video' sections. Under 'Location', 'Show Location' is checked, and the 'Latitude' and 'Longitude' fields are both set to 0.0. Under 'Video', 'Show Video' is checked, and the 'YouTube' and 'Vimeo' options are available. Both screenshots include a 'Save Channel' button at the bottom.

# Creating A Channel

4. Now, the channel page opens with the following tabs:

- a. **Private View**-- It displays the information about your channel that is only visible to you

The screenshot shows a web browser window titled "weather calculation - ThingSpeak". The URL is https://thingspeak.com/channels/283988/private\_show. The page has a blue header with the ThingSpeak logo and navigation links: Channels, Apps, Community, Support, How to Buy, Account, and Sign Out. Below the header, there are tabs: Private View (selected), Public View, Channel Settings, API Keys, and Data Import / Export. There are also buttons for Add Visualizations and Data Export. On the right, there are links for MATLAB Analysis and MATLAB Visualization. The main content area is titled "Channel Stats" and shows the following information:

- Created: 27 minutes ago
- Updated: 27 minutes ago
- Entries: 0

Below this are two charts labeled "Field 1 Chart" and "Field 2 Chart", both titled "weather calculation". The charts have a single data series named "temperature" and "humidity" respectively. At the bottom of the page, there is a toolbar with icons for Internet Explorer, File, Edit, Insert, Refresh, Stop, Home, and Favorites, along with system status indicators like battery level, signal strength, and date/time.

- b. **Public View**- if you have chosen to make your channel publicly visible then it will display the selected fields and information

The screenshot shows a web browser window titled "weather calculation - ThingSpeak". The URL is https://thingspeak.com/channels/283988. The page layout is identical to the Private View screenshot above, with the same header, tabs, and sidebar links. The "Public View" tab is selected. The "Channel Stats" section shows the same information as the Private View. Below the stats are three charts labeled "Field 1 Chart", "Field 2 Chart", and "Field 3 Chart", all titled "weather calculation". The "Field 1 Chart" shows "temperature" and "humidity" data. The "Field 2 Chart" shows "humidity" data. The "Field 3 Chart" shows "temperature" data. To the right of the charts is a map titled "Channel Location" showing the location of the channel in Nigeria. The map includes a red marker indicating the location. The bottom of the page features the same toolbar and system status indicators as the Private View screenshot.

# Creating A Channel

c. **Channel Settings-** it will show all the options that are available during the channel creation

The screenshot shows the 'Channel Settings' tab of a ThingSpeak channel. The channel ID is 283988. The name is set to 'weather calculation'. The description field is empty. There are six data fields labeled Field 1 through Field 6, each containing a placeholder value: 'temperature', 'humidity', 'dew point', 'Field 4', 'Field 5', and 'Field 6'. The help section provides a detailed explanation of the fields and their purposes.

d. **API Keys-** in this tab you will have two API Keys -- Read API Key (to read from your channel), write API Key (to write to your channel)

The screenshot shows the 'API Keys' tab of a ThingSpeak channel. It contains two main sections: 'Write API Key' and 'Read API Keys'. In the 'Write API Key' section, there is a key field with the value 'EHOFXHw6LISM1YL1' and a button to 'Generate New Write API Key'. In the 'Read API Keys' section, there is a key field with the value 'XYUT6QR5CW98QG4J', a note field, and buttons for 'Save Note' and 'Delete API Key'. The help section provides instructions for using API keys to interact with the channel.

# Creating A Channel

- e. **Data import/export-** it enables you to import and export the channel data

The screenshot shows the 'Data Import / Export - Ti' page on ThingSpeak. At the top, there are tabs for 'Private View', 'Public View', 'Channel Settings', 'API Keys', and 'Data Import / Export'. The 'Data Import / Export' tab is selected. The page is divided into two main sections: 'Import' and 'Export'. The 'Import' section contains a 'Choose File' input field with 'No file chosen', a 'Time Zone' dropdown set to '(GMT+00:00) UTC', and a green 'Upload' button. To the right of the import section is a 'Help' section with instructions on importing CSV files and a link to 'Learn more'. The 'Export' section contains a 'Download' button. On the right side of the page, under 'API Requests', there are several examples of API endpoints with their methods and URLs.

## My Channels

- f. In future your channel will be available to you just by clicking '**Channels> My Channels**'

Name	Created	Updated
LED Status	2019-03-30	2019-03-30 04:46
DHT11	2019-03-30	2019-06-16 00:15

# Chart Options

Field 1 Chart

ODU\_Blast2019

DHT11\_Temperature

Date

ThingSpeak.com



Field 1 Chart Options

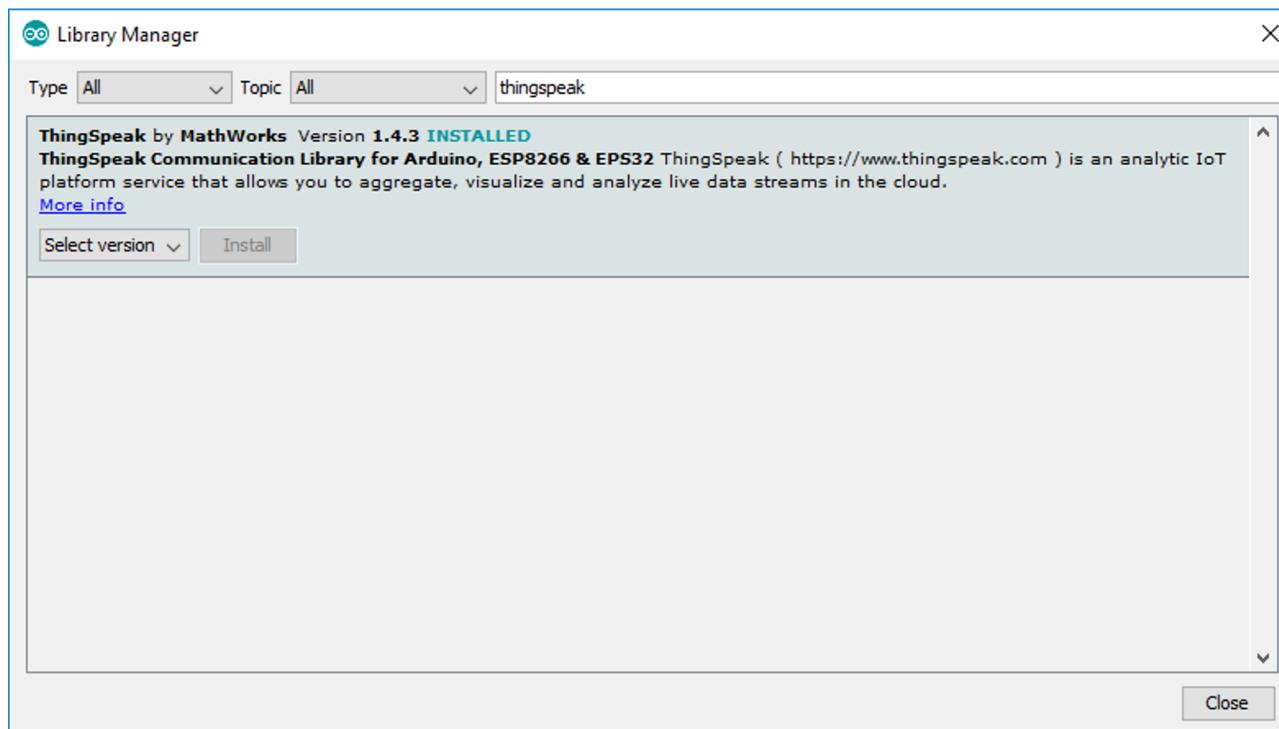
X

Title:		Timescale:	▼
X-Axis:		Average:	▼
Y-Axis:		Median:	▼
Color:	#d62020	Sum:	▼
Background:	#ffffff	Rounding:	
Type:	line	Data Min:	
Dynamic?:	true	Data Max:	
Days:		Y-Axis Min:	
Results:	60	Y-Axis Max:	

Save Cancel

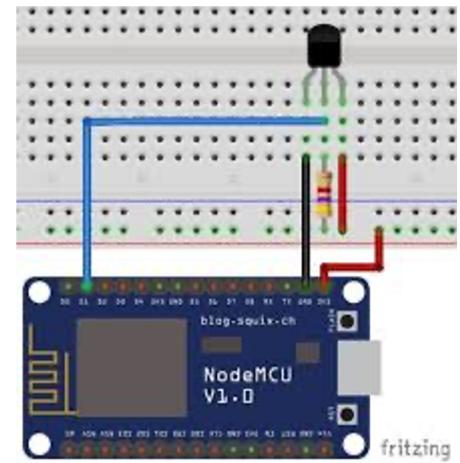
# ThingSpeak Library for Arduino

- The Arduino IDE needs to have the ThingSpeak library installed in order for your devices to know how to send data to ThingSpeak. In the Arduino IDE, choose Sketch, Include Library, and Manage Libraries. Search for “thingspeak” and click Install.



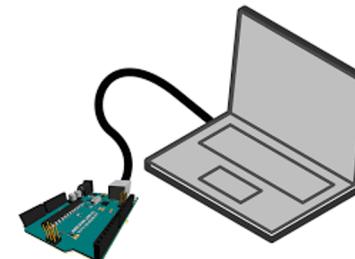
# Topics of this workshop

- **Getting started**  
(setup and programming of IoT hardware)
- **Reading and Writing**  
(physical computing: sensors and actuators)
- **Connecting your device to the Internet**  
(IoT: monitoring sensors, controlling actuators)



# IoT hardware

Any Internet-connected computer with an **interface to the real world** (sensors, actuators)

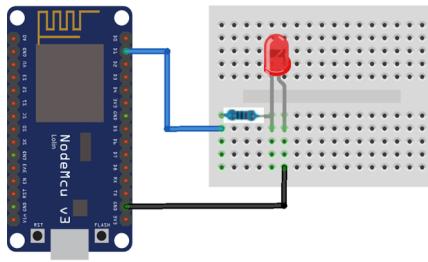


Small => can be **embedded into things**



Small computer = **microcontroller** (or **board**), e.g.  
Arduino, Netduino Plus, BeagleBone, ...

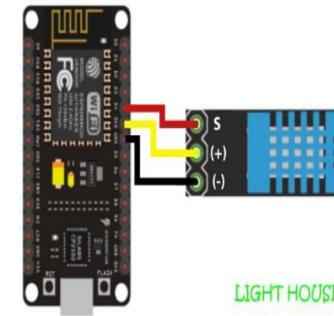
# Hands-on Activities



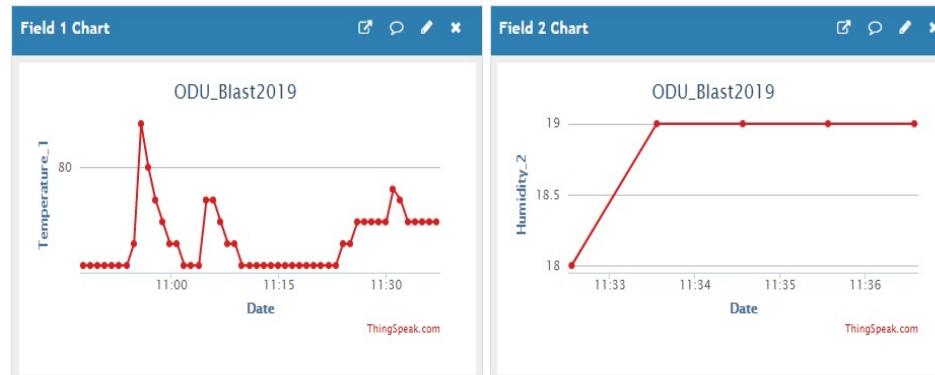
```
#define LED D1 // Led in NodeMCU at pin GPIO5 (D1).
C
void setup()
{
pinMode(LED, OUTPUT); // set the digital pin as output.

}

void loop()
{
digitalWrite(LED, HIGH); // turn the LED off.
delay(1000); // wait for 1 second.
digitalWrite(LED, LOW); // turn the LED on.
delay(1000); // wait for 1 second.
}
```



LIGHT HOUSE



# Hands on Activities and Source Code

---

**Copy&Paste or Download** examples, from GitHub:

[https://github.com/muratkuzlu/ODU\\_BLAST2021](https://github.com/muratkuzlu/ODU_BLAST2021)

Focus on **end-to-end results, not details**

# Getting started

The **IDE** (Integrated Development Environment) allows you to **program** your board, i.e., “make it do something new”

You **edit** a program on your computer, then **upload** it to your board where it’s stored in the program memory (flash) and **executed** in RAM

Please Note: Once it has been programmed, your board can run on its own, without another computer



# Reading and Writing

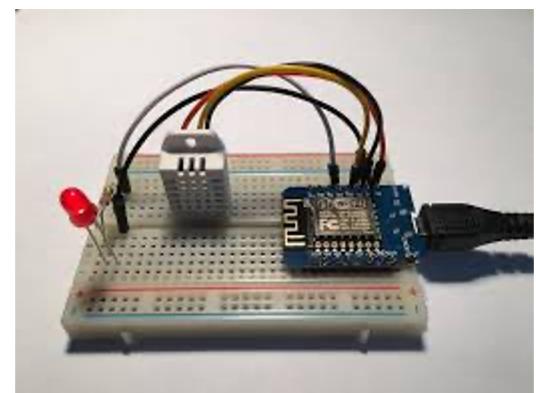
IoT hardware has an **interface to the real world**

**GPIO (General Purpose Input/Output) pins**

Measure: **read** sensor value from **input** pin

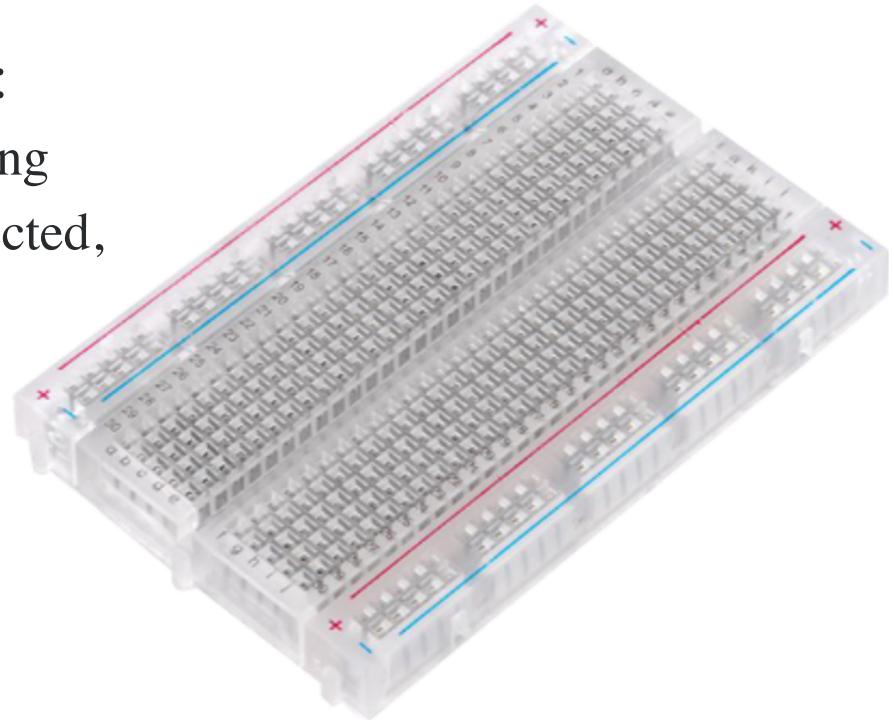
Manipulate: **write** actuator value to **output** pin

Inputs and outputs can be **digital or analog**



# Prototyping Circuits Solderless Breadboard

- One of the most useful tools in an engineer or Maker's toolkit.
- The three most important things:
  - A breadboard is easier than soldering
  - A lot of those little holes are connected, which ones?
  - Sometimes breadboards break



# Wiring a LED with Arduino

## Hardware

- NodeMCU

**NodeMCU ESP8266**

**GPIO LIMITATIONS**

Best Pins for Input (best to worst)	
Board Label	Raw Pin Number
D1	5
D2	4
D5	14
D6	12
D7	13
D0	16
SD2	9
SD3	10
RX	3

Best Pins for Output (best to worst)	
Board Label	Raw Pin Number
D1	5
D2	4
D5	14
D6	12
D7	13
D8	15

**\*Pin is high on boot**

**\*Boot failure if pulled low**

**\*Boot failure if pulled high**

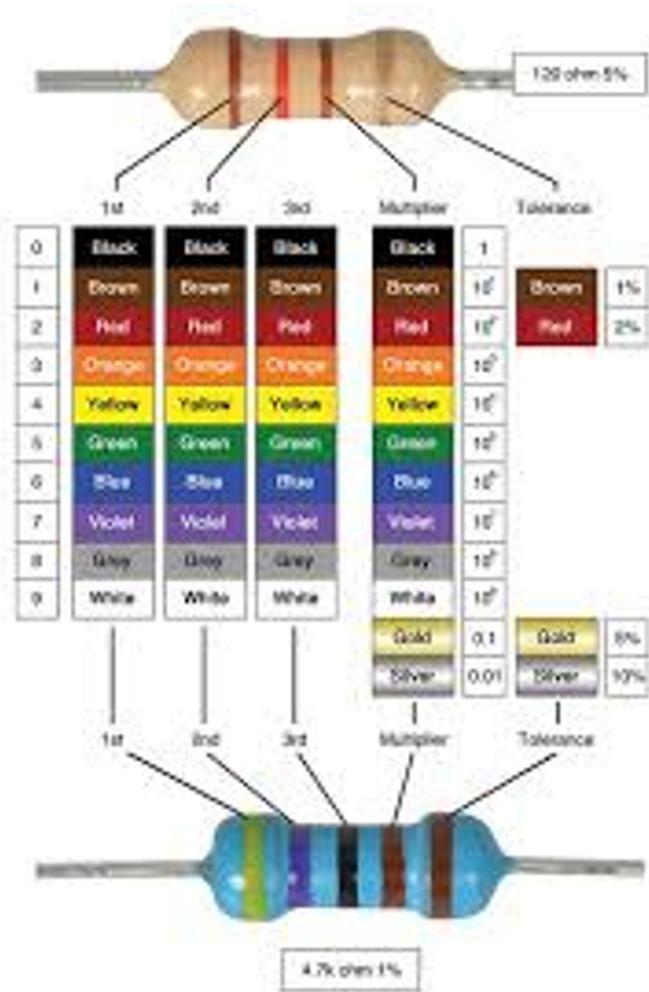
Made by: [www.youtubee.com/cTheHooKUp](http://www.youtubee.com/cTheHooKUp)

# The resistor

Resistors are the **workhorse** of electronics

Resistance is **measured** in  $\Omega$  (Ohm) and adds up in series; a resistors orientation doesn't matter

A resistors  $\Omega$  value is **color-coded** right on it

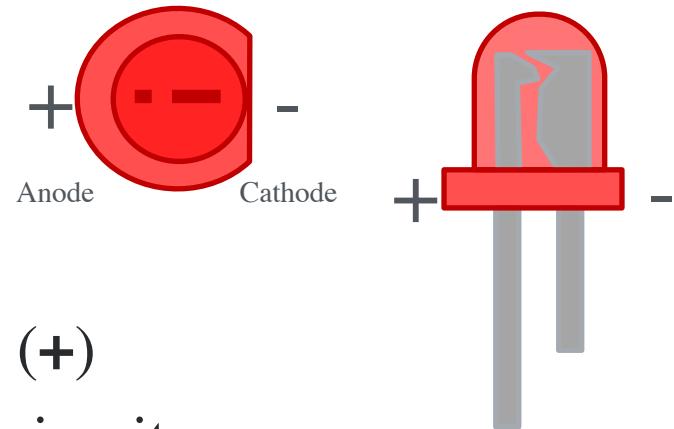


# The LED

The **LED** (Light Emitting Diode)  
is a simple, digital **actuator**

LEDs have a **short leg (-)** and a **long leg (+)**  
and it matters how they are oriented in a circuit

To prevent damage, LEDs are used together with a  $1K\Omega$   
**resistor** (or anything from  $300\Omega$  to  $2K\Omega$ )



---

# Hands-on Activity - I

## NodeMCU

# Wiring a LED with Arduino

---

We will introduce how to blink the on-board LED and how to blink a external LED.

## Hardware

- NodeMCU x 1
- LED x 1
- 1K ohm resistor x 1
- Micro USB cable x 1
- PC x 1

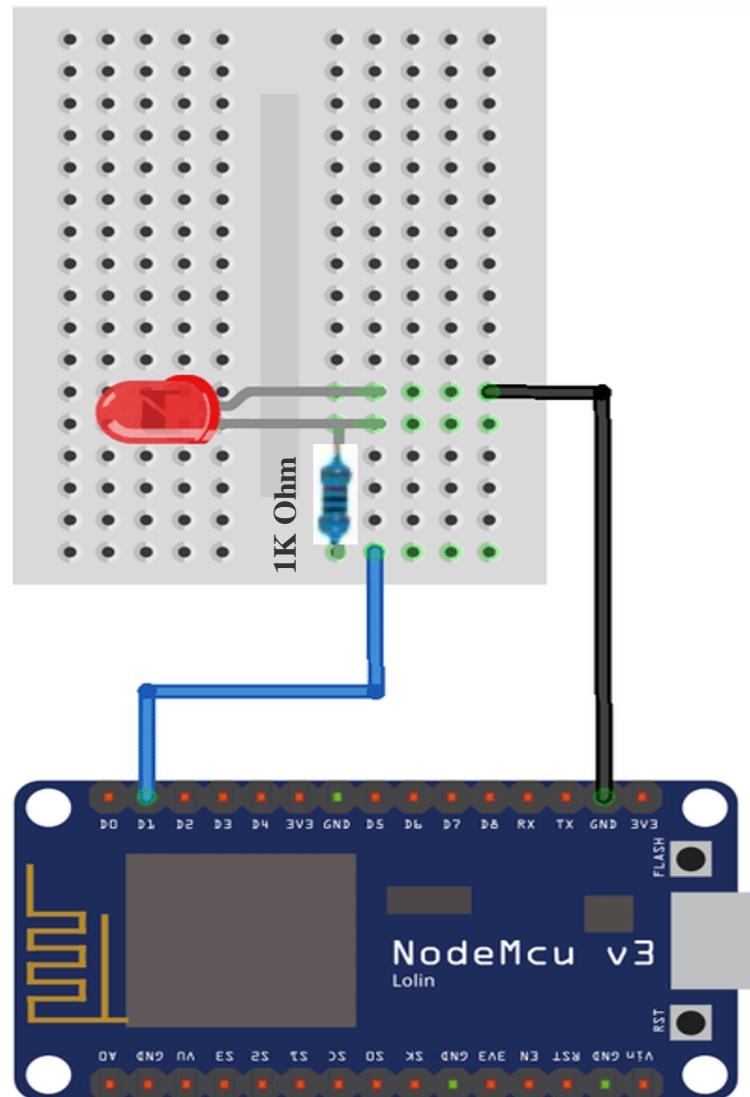
## Software

- Arduino IDE(version 1.6.4+)

# Wiring a LED with Arduino

## *Set up*

- Connect the long leg of the LED (the positive leg, called the anode) to the other end of the resistor (1K ohm).
- Connect the short leg of the LED (the negative leg, called the cathode) to the GND.
- In the diagram below we show a NodeMCU that has D1 as the `LED_BUILTIN` value.



# Digital output with Arduino

## Code

→ Copy the following code to the IDE

```
#define LED D1 // Led in NodeMCU at pin GPIO5 (D1).
```

**Specifying what pin is going to be used**

```
void setup()
{
pinMode(LED, OUTPUT); // set the digital pin as output.
}
```

**Initialize variables.  
Runs once**

```
void loop()
{
digitalWrite(LED, HIGH); // turn the LED off.
delay(1000); // wait for 1 second.
digitalWrite(LED, LOW); // turn the LED on.
delay(1000); // wait for 1 second.
}
```

**Used to actively control the Arduino board. Run repeatedly**

Note that LOW is the voltage level but actually the LED is on; this is because it is active low on the ESP8266.

→ Upload

# Digital output with Arduino and IoT - I

Now we are going to connect to IoT

→ Copy the following code to a new IDE sketch

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>

const char *ssid = "ssid"; // replace with your wifi ssid and wpa2 key
const char *pass = "password";
const char* server = "api.thingspeak.com";
// Enter your Write API key from ThingSpeak
const char * myWriteAPIKey = "3M0SBN71PI6UD1A4";
unsigned long myChannelNumber = 803487;
uint8_t LED_Status, l=0;

WiFiClient client;

#define LED D1 // Led in NodeMCU at pin GPIO5 (D1).
```

Private View Public View Channel Settings API Keys Data Import / Export

Write API Key

Key 3M0SBN71PI6UD1A4

Generate New Write API Key

Help

API keys enable you to write data to feeds and charts. Click Generate New Write API Key

API Keys Settings

- Write API Key: Use this key to write data to feeds and charts. Click Generate New Write API Key
- Read API Keys: Use this key to read data from feeds and charts. Click Generate New Read API Key

Private View Public View Channel Settings Share

Channel Settings

Percentage complete 50%

Channel ID 803487

Name ODU\_Blast2019

Description Hands-on IOT Activities

Field 1 Temperature

Field 2 Humidity

Field 3 LED\_Status

# Digital output with Arduino and IoT - II

```
void setup()
{
    Serial.begin(115200);
    delay(10);

    pinMode(LED, OUTPUT); // LED D1 pin as output.

    Serial.println("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    // Print the IP address
    Serial.println(WiFi.localIP());
    ThingSpeak.begin(client);
}

56 }
```

# Digital output with Arduino and IoT - III

```
void loop() {  
    LED_Status = HIGH;  
  
    digitalWrite(LED, LED_Status); // turn the LED off.(Note that LOW is the voltage level  
    but actually  
        //the LED is on; this is because it is active low on the ESP8266.  
  
    Serial.print("LED Status is :");  
    Serial.print(LED_Status);  
  
    ThingSpeak.writeField(myChannelNumber, 3, LED_Status, myWriteAPIKey);  
    delay(30000); // ThingSpeak will only accept updates every 15 seconds.  
  
  
    LED_Status = LOW;  
    digitalWrite(LED, LED_Status); // turn the LED on.  
  
    Serial.print("LED Status is :");  
    Serial.print(LED_Status);  
  
    ThingSpeak.writeField(myChannelNumber, 3, LED_Status, myWriteAPIKey);  
    delay(30000); // ThingSpeak will only accept updates every 15 seconds. → Upload  
}
```

# Digital output with Arduino and IoT -IV

→ Check ThingSpeak

ODU\_Blast2019

Channel ID: 803487  
Author: mkuzlu123  
Access: Public

Hands-on IOT Activities

Private View    Public View    Channel Settings    Sharing    API Keys    Data Import / Export

Add Visualizations

Add Widgets

Export recent data

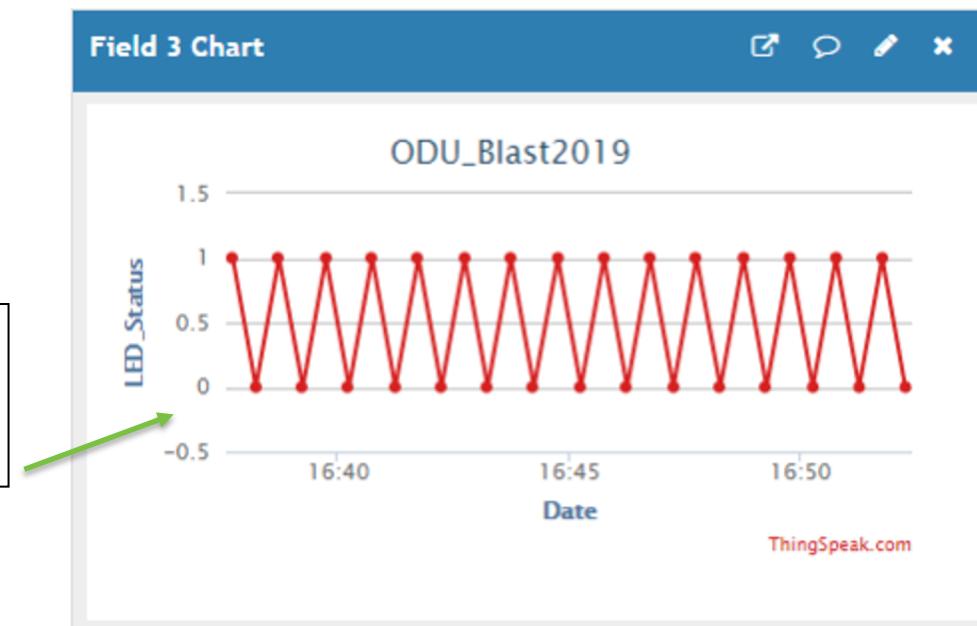
MATLAB Analysis

MATLAB Visualization

Channel Stats

Created: about 24 hours ago  
Last entry: 5 minutes ago  
Entries: 64

Changes in LED  
ON & OFF every  
second



---

# Hands-on Activity - II

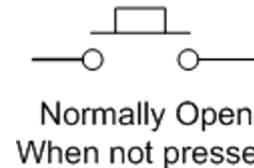
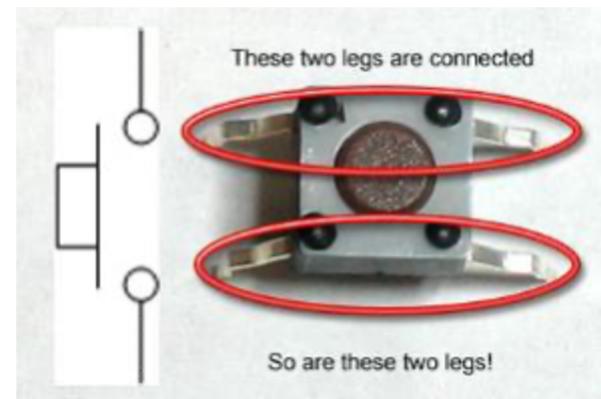
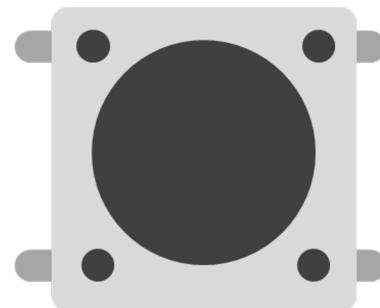
## NodeMCU

# The switch

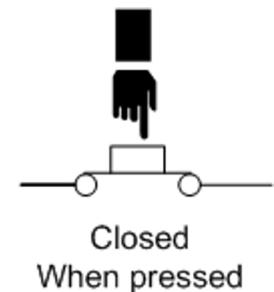
A switch is a simple, digital **sensor**

Switches come in different forms, but all of them in some way **open** or **close** a gap in a wire

The **pushbutton** switch has four legs for easier mounting, but only two of them are needed



Normally Open  
When not pressed



Closed  
When pressed

# Wiring a switch with Arduino

---

We will introduce how to blink the on-board LED and how to blink a external LED.

## Hardware

- NodeMCU x1
- Push Button x1
- LED x1
- 10 K ohm Resistor x1
- 200-ohm Resistor x1
- Bread Board x1
- PC x1

## Software

- [Arduino IDE\(version 1.6.4+\)](#)

# Digital input with Arduino

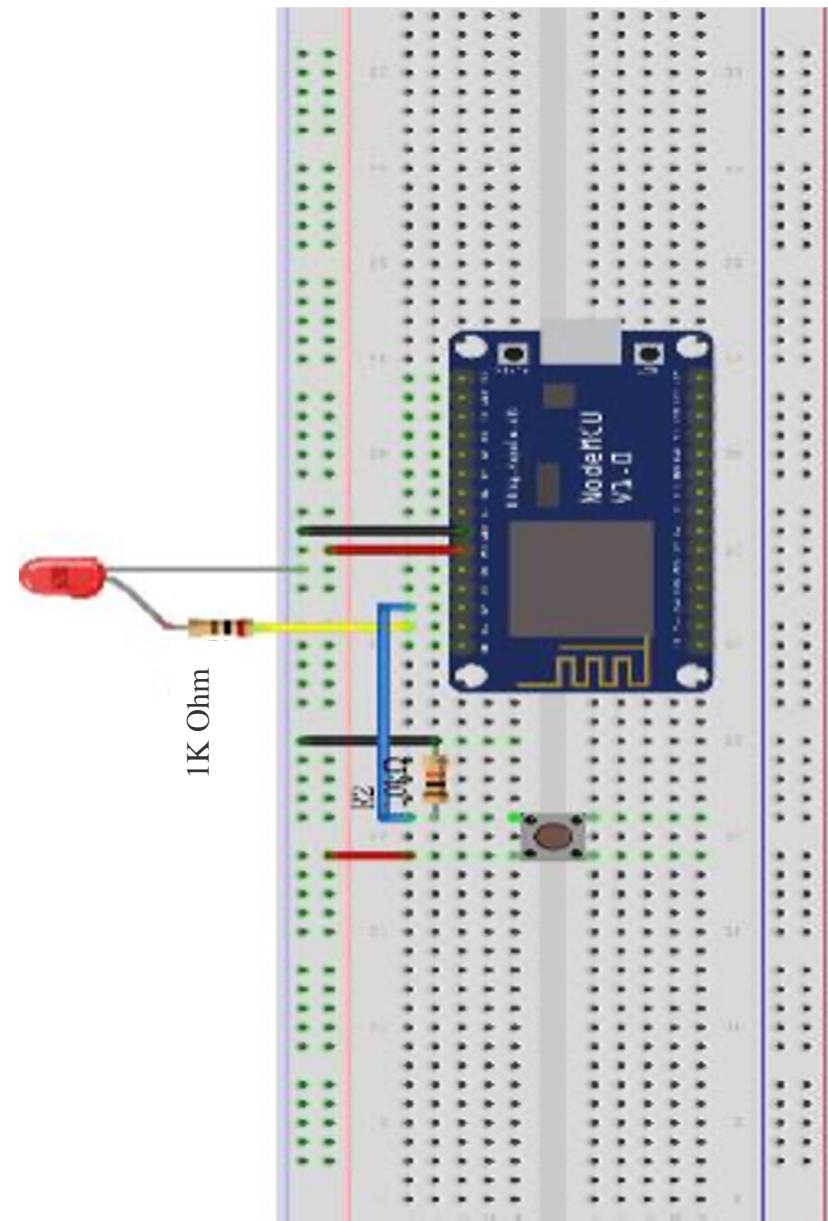
## Set up

### **Push Button connections :**

- The first pin goes from one leg of the pushbutton through a pull-up resistor(here 10K Ohms) to Ground (**GND**).
- The **second pin** goes from the corresponding leg of the pushbutton to the **5v supply pin**.
- The **third pin** connects to a Digital I/O pin (here pin **D2**) which reads the button's state.

### **LED connections:**

- LED Anode is connected to Digital I/O pin (here pin **D1**) and Cathode to ground (**GND**) pin.



# Digital input with Arduino

## Code

- Copy the following code to the IDE

```
#define LED 5 // D1(gpio5)
#define BUTTON 4 //D2(gpio4)

int buttonState=0;
int switchState = 0; // actual read value from pin4
int oldSwitchState = 0; // last read value from pin4
int lightsOn = 0; // is the switch on = 1 or off = 0

void setup() {
pinMode(BUTTON, INPUT); // push button
pinMode(LED, OUTPUT); // anything you want to control using a switch e.g. a Led
}

void loop() {
switchState = digitalRead(BUTTON); // read the pushButton State
if (switchState != oldSwitchState) // catch change
{
oldSwitchState = switchState;
if (switchState == HIGH)
{
// toggle
lightsOn = !lightsOn;
}
}
if(lightsOn)
{
digitalWrite(LED, HIGH); // set the LED on
} else {
digitalWrite(LED, LOW); // set the LED off
}
}
```

→ Upload

# Digital input with Arduino and IoT - I

Now we are going to connect to IoT

- Copy the following code to a new IDE sketch

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>
const char *ssid = "AS2L-Room"; // replace with your wifi ssid and wpa2 key
const char *pass = "as2l214c";
const char* server = "api.thingspeak.com";
const char * myWriteAPIKey = "3M0SBN71PI6UD1A4"; // Enter your Write API key from
ThingSpeak

unsigned long myChannelNumber = 803487;

WiFiClient client;

#define LED 5 // D1(gpio5)
#define BUTTON 4 //D2(gpio4)

int buttonState=0;

int switchState = 0; // actual read value from pin4
int oldSwitchState = 0; // last read value from pin4
int lightsOn = 0; // is the switch on = 1 or off = 0
```

# Digital input with Arduino and IoT - II

```
void setup()
{
    Serial.begin(115200);
    delay(10);

    pinMode(BUTTON, INPUT); // push button
    pinMode(LED, OUTPUT); // anything you want to control using a switch e.g. a Led

    Serial.println("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, pass);

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");

    // Print the IP address
    Serial.println(WiFi.localIP());
    ThingSpeak.begin(client);
}
```

# Digital input with Arduino and IoT - III

```
void loop() {
switchState = digitalRead(BUTTON); // read the pushButton State
if (switchState != oldSwitchState) // catch change
{
oldSwitchState = switchState;
if (switchState == HIGH)
{
// toggle
lightsOn = !lightsOn;
}
}
if(lightsOn)
{
digitalWrite(LED, HIGH); // set the LED on
buttonState = HIGH;
} else {
digitalWrite(LED, LOW); // set the LED off
buttonState = LOW;
}
Serial.print("Button Status is :");
Serial.print(buttonState);
ThingSpeak.writeField(myChannelNumber, 4, buttonState,
myWriteAPIKey);
delay(1000);
}
```

→ Upload

# Digital output with Arduino and IoT -IV

→ Check ThingSpeak

ODU\_Blast2019

Channel ID: 803487

Author: mkuzlu123

Access: Public

Hands-on IOT Activities

Private View

Public View

Channel Settings

Sharing

API Keys

Data Import / Export

Add Visualizations

Add Widgets

Export recent data

MATLAB Analysis

MATLAB Visualization

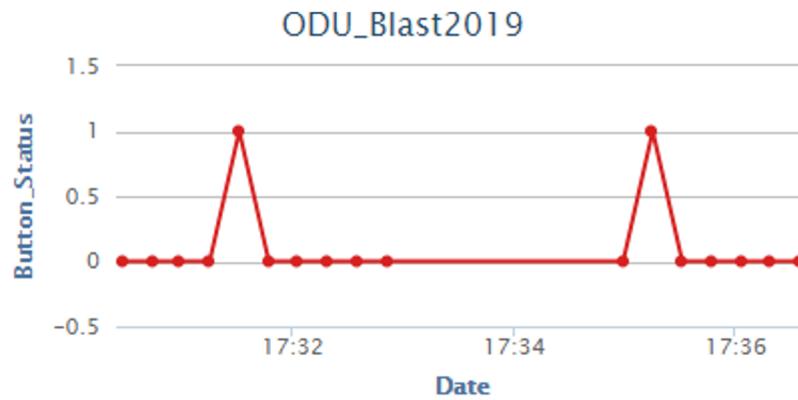
## Channel Stats

Created: about 24 hours ago

Last entry: 5 minutes ago

Entries: 64

**Changes in Button Status**



ThingSpeak.com

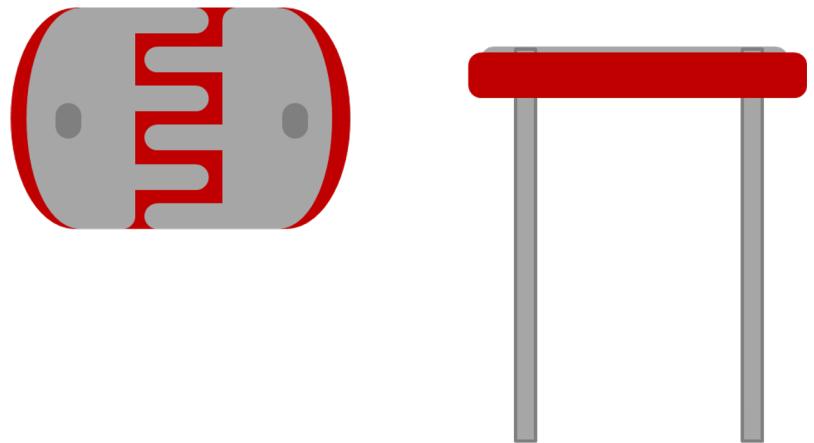
---

# Hands-on Activity - III

## NodeMCU

# Photoresistor (LDR)

A photoresistor or **LDR** (light dependent resistor) is a resistor whose resistance depends on light intensity



An LDR can be used as a simple, **analog sensor**

The orientation of an LDR does not matter

# Wiring a LED with Arduino

---

We will introduce how to blink the on-board LED and how to blink a external LED.

## Hardware

- NodeMCU
- LDR / photoresistor
- 10k ohm resistor
- Breadboard
- Micro USB cable
- Connecting Wires

## Software

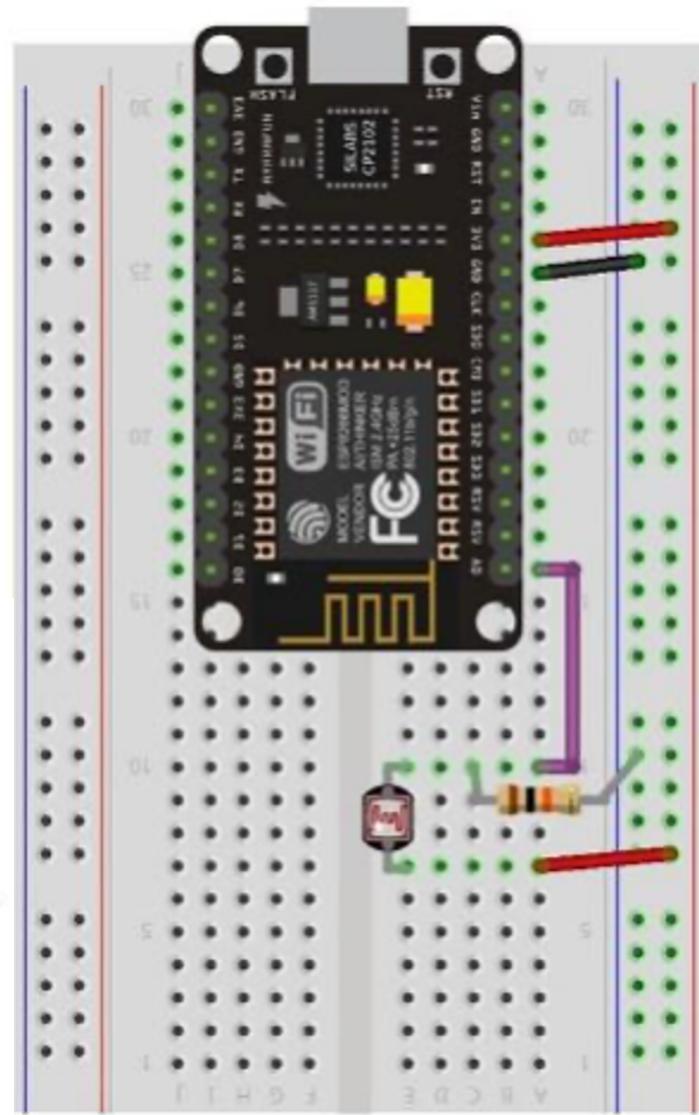
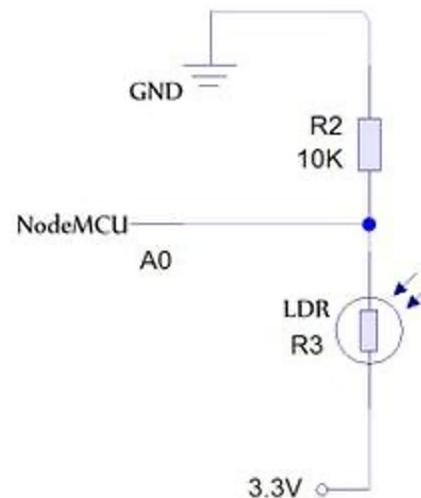
- Arduino IDE(version 1.6.4+)

# Wiring a LED with Arduino

## Set up

The LDR output is actually analog in nature, so it gets connected to the A0 pin of the NodeMCU.

**Note:** this setup is a *voltage-divider*, as the total voltage is divided between LDR and resistor to keep  $0V < A0 < 2.5V$



# Analog input with Arduino

## Code

→ Copy the following code to the IDE

```
void setup()
{
    Serial.begin(115200);
    delay(10);
}

void loop() {
    int sensorValue = analogRead(A0); // read the input on analog pin 0

    float voltage = sensorValue * (5.0 / 1023.0); // Convert the analog
    reading (which goes from 0 - 1023) to a voltage (0 - 5V)

    Serial.println(voltage); // print out the value you read
}
```

→ Upload

# Analog input with Arduino and IoT - I

Now we are going to connect to IoT

- Copy the following code to a new IDE sketch

```
#include "ThingSpeak.h"  
#include <ESP8266WiFi.h>  
  
const char *ssid = "AS2L-Room"; // replace with your wifi ssid and wpa2 key  
const char *pass = "as2l214c";  
const char* server = "api.thingspeak.com";  
const char * myWriteAPIKey = "3M0SBN71PI6UD1A4"; // Enter your Write API key from ThingSpeak  
  
unsigned long myChannelNumber = 803487;  
  
WiFiClient client;
```

# Analog input with Arduino and IoT - II

```
void setup()
{
    Serial.begin(115200);
    delay(10);

    Serial.println("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");

    // Print the IP address
    Serial.println(WiFi.localIP());
    ThingSpeak.begin(client);
}
```

# Analog input with Arduino and IoT - III

```
void loop() {  
  
    int sensorValue = analogRead(A0); // read the input on analog pin 0  
  
    float voltage = sensorValue * (5.0 / 1023.0); // Convert the analog reading (which goes from  
0 - 1023) to a voltage (0 - 5V)  
  
    Serial.println(voltage); // print out the value you read  
    Serial.print("Photoresistor value is :");  
    Serial.print(voltage);  
    ThingSpeak.writeField(myChannelNumber, 5, voltage, myWriteAPIKey);  
    delay(30000); // ThingSpeak will only accept updates every 15 seconds.  
}
```

→ Upload

# Digital output with Arduino and IoT -IV

→ Check ThingSpeak

ODU\_Blast2019

Channel ID: 803487

Author: mkuzlu123

Access: Public

Hands-on IOT Activities

Private View

Public View

Channel Settings

Sharing

API Keys

Data Import / Export

Add Visualizations

Add Widgets

Export recent data

MATLAB Analysis

MATLAB Visualization

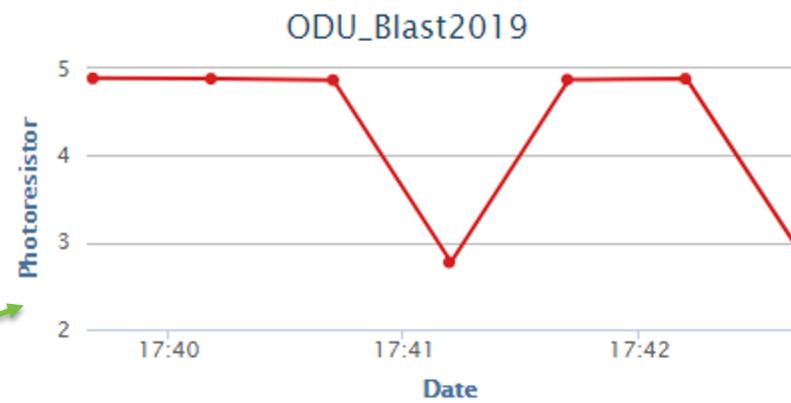
## Channel Stats

Created: about 24 hours ago

Last entry: 5 minutes ago

Entries: 64

**Changes in  
Photoresistor**



ThingSpeak.com

---

# Hands-on Activity - IV

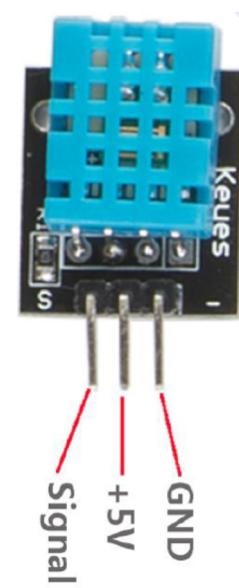
## NodeMCU

# Temperature & Humidity Sensor DHT11

The DHT11 sensor can detect temperature (C and F) & humidity.

The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface.

It has everything it requires built into it, so it will work very well with the NodeMCU. This sensor is used in conjunction with the DHT11 Library.



# Wiring a switch with Arduino

---

We will learn how to set up the DHT11 Humidity and Temperature sensor on your NodeMCU. And learn about how the Humidity sensor works, and how to check output readings from the Serial monitor.

## Hardware

- NodeMCU
- DHT11 Humidity and Temperature sensor
- Breadboard
- Jumper Wires (Optional)
- Micro USB Cable

## Software

- [Arduino IDE\(version 1.6.4+\)](#)

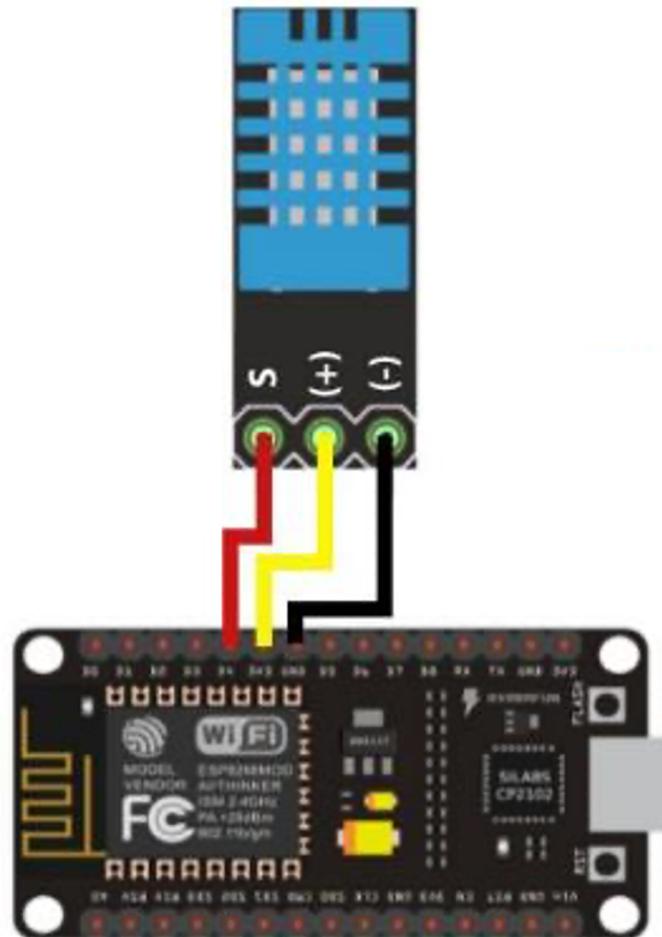
# Digital input with Arduino

## *Set up*

Wiring the **DHT11** to the NodeMCU is really easy, but the connections are different depending on which type you have either 3-pins or 4-pins

The **wiring connections** are made as follows:

- **Pin 1** of the DHT11 goes into Digital Pin **D4** of the NodeMCU.
- **Pin 2** of the DHT11 goes into **+3.3v** of the NodeMCU.
- **Pin 3** of the DHT11 goes into Ground Pin (**GND**) of the NodeMCU.



# Digital input with Arduino

## Code

→ Copy the following code to the IDE

```
#include "DHTesp.h"

int temperature, humidity, k=0, l=0;

#define DHTPIN D4      //pin where the dht11 is connected
DHTesp dht;

void setup()
{
    Serial.begin(115200);
    delay(10);
    //dht.begin();
    dht.setup(DHTPIN, DHTesp::DHT11); // data pin 4
}

void loop()
{
    static boolean data_state = false;
    float humidity = dht.getHumidity();
    float temperature = dht.getTemperature();
    temperature = CelsiusToFahrenheit(temperature);
    Serial.print("Temperature Value is :");
    Serial.print(temperature);
    Serial.println("F");
    Serial.print("Humidity Value is :");
    Serial.print(humidity);
    Serial.println("%");
    delay(5000);
}
```

```
//Functions  
  
float FahrenheitToCelsius(float fahrenheit)  
{  
    float celsius;  
    celsius = (fahrenheit - 32.0) * 5.0 / 9.0;  
    return celsius;  
}
```

```
float CelsiusToFahrenheit(float celsius)  
{  
    float fahrenheit;  
    fahrenheit = (celsius * 9.0) / 5.0 + 32;  
    return fahrenheit;  
}
```

#### NOTE:

When you check the serial monitor make sure the baud rate and the serialbegin number in your code is the same.

→ Upload

# Digital input with Arduino and IoT - I

Now we are going to connect to IoT

- Copy the following code to a new IDE sketch

```
#include "DHTesp.h"
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>

const char *ssid = "AS2L-Room";    // replace with your wifi ssid and wpa2 key
const char *pass = "as2l214c";
const char* server = "api.thingspeak.com";
const char * myWriteAPIKey = "3M0SBN71PI6UD1A4"; // Enter your Write API key from ThingSpeak
unsigned long myChannelNumber = 803487;
uint8_t temperature, humidity, k=0, l=0;

#define DHTPIN D4      //pin where the dht11 is connected
DHTesp dht;

WiFiClient client;
```

# Digital input with Arduino and IoT - II

```
void setup()
{
    Serial.begin(115200);
    delay(10);
    //dht.begin();
    dht.setup(DHTPIN, DHTesp::DHT11); // data pin 2
    Serial.println("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");

    // Print the IP address
    Serial.println(WiFi.localIP());
    ThingSpeak.begin(client);
}
```

# Digital input with Arduino and IoT - III

```
void loop()
{
    static boolean data_state = false;
    float humidity = dht.getHumidity();
    float temperature = dht.getTemperature();
    temperature = CelsiusToFahrenheit(temperature);
    Serial.print("Temperature Value is :");
    Serial.print(temperature);
    Serial.println("F");
    Serial.print("Humidity Value is :");
    Serial.print(humidity);
    Serial.println("%");
    // Write to ThingSpeak. There are up to 8 fields in a channel, allowing you to store up to 8 different pieces of information in a channel.
    Here, we write to field 1.
    if(temperature < 255)
    {
        k=temperature;
    }
    if(humidity < 255)
    {
        l=humidity;
    }
    if( data_state )
    {
        ThingSpeak.writeField(myChannelNumber, 1, k, myWriteAPIKey);
        data_state = false;
    }
    else
    {
        ThingSpeak.writeField(myChannelNumber, 2, l, myWriteAPIKey);
        data_state = true;
    }
    delay(30000); // ThingSpeak will only accept updates every 15 seconds.
}
```

→ Upload

85 `delay(30000);` // ThingSpeak will only accept updates every 15 seconds.

# Digital output with Arduino and IoT -IV

→ Check ThingSpeak

ODU\_Blast2019

Channel ID: 803487

Author: mkuzlu123

Access: Public

Hands-on IOT Activities

Private View

Public View

Channel Settings

Sharing

API Keys

Data Import / Export

+ Add Visualizations

+ Add Widgets

Export recent data

MATLAB Analysis

MATLAB Visualization

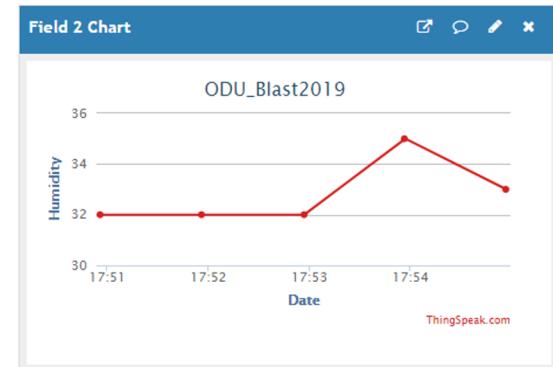
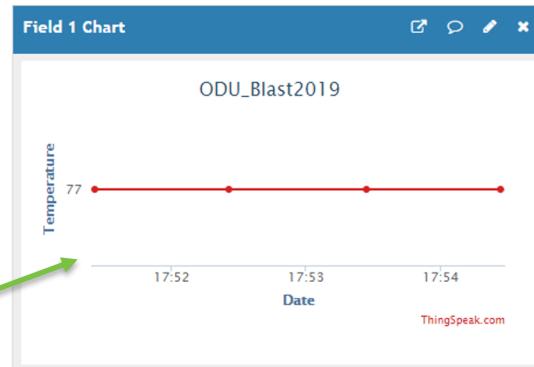
## Channel Stats

Created: about 24 hours ago

Last entry: 5 minutes ago

Entries: 64

Changes in  
Temperature and  
Humidity



---

# Hands-on Activity - V

## NodeMCU

# Interfacing a 4x4 Keypad with Arduino

We will introduce how to use a 4x4 matrix keypad with the NodeMCU ESP8266. We will then monitor the inputs of the keypad through IoT by using ThingSpeak.

## Hardware

- NodeMCU x 1
- Breadboard x 1
- Micro USB cable x 1
- PC x 1
- Membrane Switch Module Keypad
- Male to Male Wires x 8



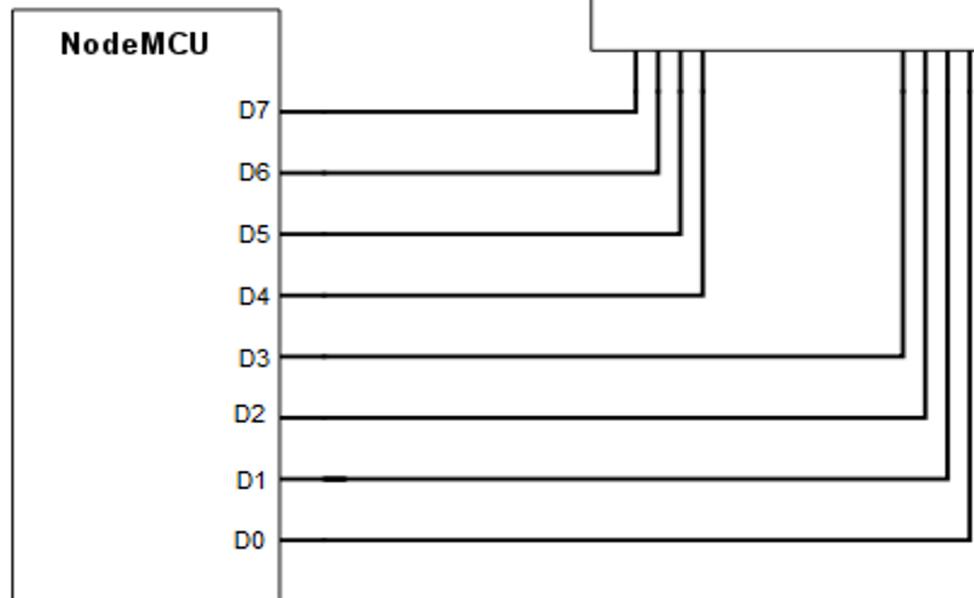
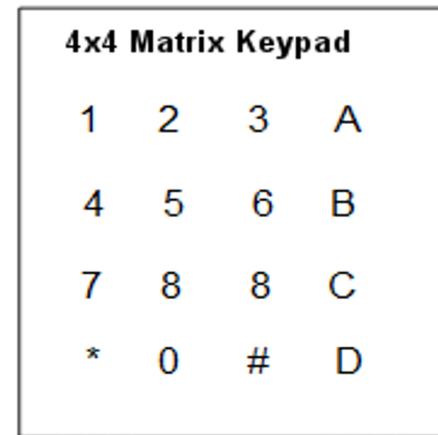
## Software

- Arduino IDE(version 1.6.4+)

# Wiring a 4x4 Keypad with Arduino

## Set up

- With the screen of Membrane Switch Module facing you, start from left to right when wiring each of the eight male to male wires
- The first wire connects to pin (D7)
- The second wire connects to pin (D6)
- The third wire connects to pin (D5)
- The fourth wire connects to pin (D4)
- The fifth wire connects to pin (D3)
- The sixth wire connects to pin (D2)
- The seventh wire connects to pin (D1)
- The eighth wire connects to pin (D0)

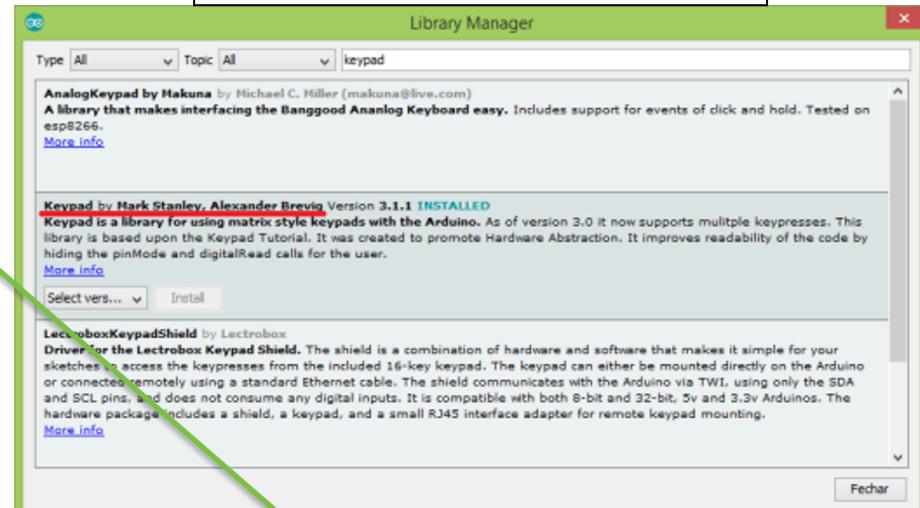


# Digital output with Arduino - I

## Code

```
#include <Keypad.h> ←  
  
const byte n_rows = 4;  
const byte n_cols = 4;  
  
char keys[n_rows][n_cols] = [  
    {'1', '2', '3', 'A'},  
    {'4', '5', '6', 'B'},  
    {'7', '8', '9', 'C'},  
    {'*', '0', '#', 'D'}  
};  
  
byte colPins[n_rows] = {D3, D2, D1, D0};  
byte rowPins[n_cols] = {D7, D6, D5, D4};
```

Add the keypad.h library



Global  
Variables

Keypad Mappings

Array Declaration

# Digital output with Arduino - II

```
Keypad myKeypad = Keypad( makeKeymap(keys), rowPins, colPins, n_rows, n_cols);  
  
void setup(){  
    Serial.begin(115200);  
}  
  
void loop(){  
    char myKey = myKeypad.getKey();  
  
    if (myKey != NULL){  
        Serial.print("Key pressed: ");  
        Serial.println(myKey);  
    }  
}
```

Passing the matrix of keys to a macro that will cast it to a char array

Initialize the Serial Connection

Obtain the key that is being pressed

# Digital output with Arduino and IoT - I

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>
#include <Keypad.h>

const char *ssid = "AS2L-Room";      // replace with your wifi ssid
const char *pass = "as2l214c";        // replace with your wifi password

const char* server = "api.thingspeak.com";
const char * myWriteAPIKey = "3M0SBN71PI6UD1A4";           ←
// Enter your Write API key from ThingSpeak
unsigned long myChannelNumber = 803487;

WiFiClient client;

const byte n_rows = 4;
const byte n_cols = 4;

char keys[n_rows][n_cols] = {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}
};

byte colPins[n_rows] = {D3, D2, D1, D0};
byte rowPins[n_cols] = {D7, D6, D5, D4};
```

Now we are going to connect to IoT

The screenshot shows two pages from the ThingSpeak website. The top page is titled 'Write API Key' and contains a text input field with the value '3M0SBN71PI6UD1A4'. A green arrow points from the 'myWriteAPIKey' variable in the code above to this input field. The bottom page is titled 'Channel Settings' and lists details for channel ID 803487. The channel name is 'ODU\_Blast2019' and the description is 'Hands-on IOT Activities'. Three fields are selected: 'Temperature', 'Humidity', and 'LED\_Status'. Each field has a checked checkbox next to it.

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Help

API keys enable you to write data to: keys are auto-generated when you ci

API Keys Settings

Write API Key

Key 3M0SBN71PI6UD1A4

Generate New Write API Key

Private View Public View Channel Settings Share

Percentage complete 50%

Channel ID 803487

Name ODU\_Blast2019

Description Hands-on IOT Activities

Field 1 Temperature

Field 2 Humidity

Field 3 LED\_Status

# Digital output with Arduino and IoT - II

```
Keypad myKeypad = Keypad( makeKeymap(keys), rowPins, colPins, n_rows, n_cols);

void setup()
{
    Serial.begin(115200);
    delay(10);

    Serial.println("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, pass);

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");

    // Print the IP address
    Serial.println(WiFi.localIP());
    ThingSpeak.begin(client);

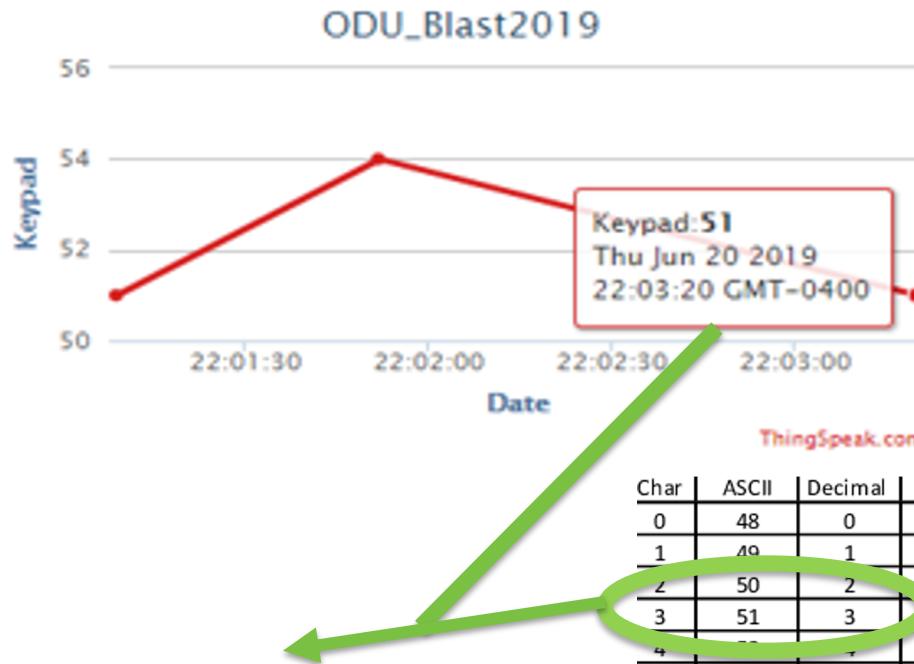
}
```

# Digital output with Arduino and IoT - III

```
void loop() {  
    char myKey = myKeypad.getKey();  
  
    if (myKey != NULL) {  
        Serial.print("Key pressed: ");  
        Serial.println(myKey);  
        ThingSpeak.writeField(myChannelNumber, 6, myKey, myWriteAPIKey);  
        delay(3000); // ThingSpeak will only accept updates every 15 seconds.  
    }  
}
```

→ Upload

# Digital output with Arduino and IoT -IV



Char	ASCII	Decimal	Bits	Char	ASCII	Decimal	Bits	Char	ASCII	Decimal	Bits
0	48	0	000000	F	70	22	010110	d	100	44	101100
1	49	1	000001	G	71	23	010111	e	101	45	101101
2	50	2	000010	H	72	24	011000	f	102	46	101110
3	51	3	000011	I	73	25	011001	g	103	47	101111
4	52	4	000100	J	74	26	011010	h	104	48	110000
5	53	5	000101	K	75	27	011011	i	105	49	110001
6	54	6	000110	L	76	28	011100	j	106	50	110010
7	55	7	000111	M	77	29	011101	k	107	51	110011
8	56	8	001000	N	78	30	011110	l	108	52	110100
9	57	9	001001	O	79	31	011111	m	109	53	110101
:	58	10	001010	P	80	32	100000	n	110	54	110110
;	59	11	001011	Q	81	33	100001	o	111	55	110111
<	60	12	001100	R	82	34	100010	p	112	56	111000
=	61	13	001101	S	83	35	100011	q	113	57	111001
>	62	14	001110	T	84	36	100100	r	114	58	111010
?	63	15	001111	U	85	37	100101	s	115	59	111011
@	64	16	010000	V	86	38	100110	t	116	60	111100
A	65	17	010001	W	87	39	100111	u	117	61	111101
B	66	18	010010	'	96	40	101000	v	118	62	111110
C	67	19	010011	a	97	41	101001	w	119	63	111111
D	68	20	010100	b	98	42	101010				
E	69	21	010101	c	99	43	101011				

- The number three was pressed on the keypad and the last entry recorded on ThingSpeak was 51. If you refer to the ASCII table shown to the right, the number 51 represents the decimal number 3.

# More

---

- Google & Youtube - search for projects, solutions to occurring problems and data sheets for components
- <http://www.blynk.cc/> - Homepage of the Blynk software, getting started, community forums
- <http://www.esp8266.com/> - Everything on ESP8266, wiki
- <https://www.adafruit.com/> - Learning materials, guides, example projects, forums, store
- <https://github.com/> - Largest code host, lots of projects and sample code
- <http://allaboutee.com/> - good ESP8266 tutorials
- <https://nurdspace.nl/ESP8266/> - ESP8266 info and basic list of AT commands



**Dr. Murat Kuzlu**  
[mkuzlu@odu.edu](mailto:mkuzlu@odu.edu)

---

# Backup Slides

---

# Hands-on Activity - I

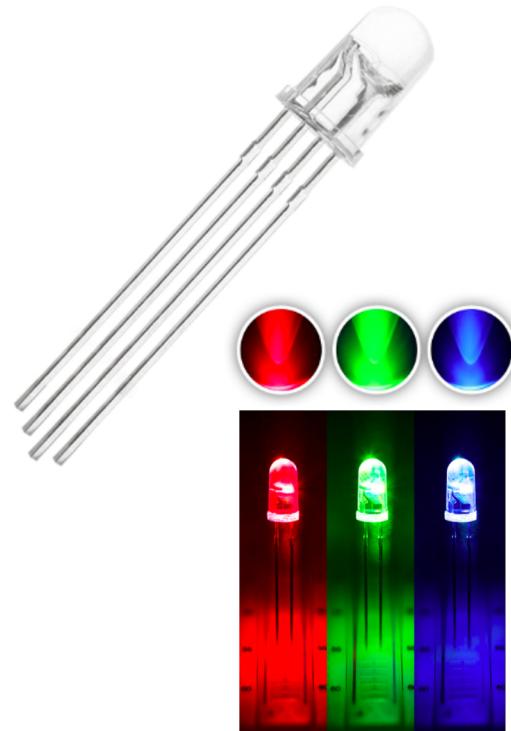
## MEGA2560

# Wiring a RGB LED with Arduino

We will introduce how to use a RGB LED with the MEGA2560

## Hardware

- (1) x Elegoo Mega 2560 R3
- (1) x 830 Tie Points Breadboard
- (4) x M-M wires (Male to Male jumperwires)
- (1) x RGB LED
- (3) x 220 ohm resistors



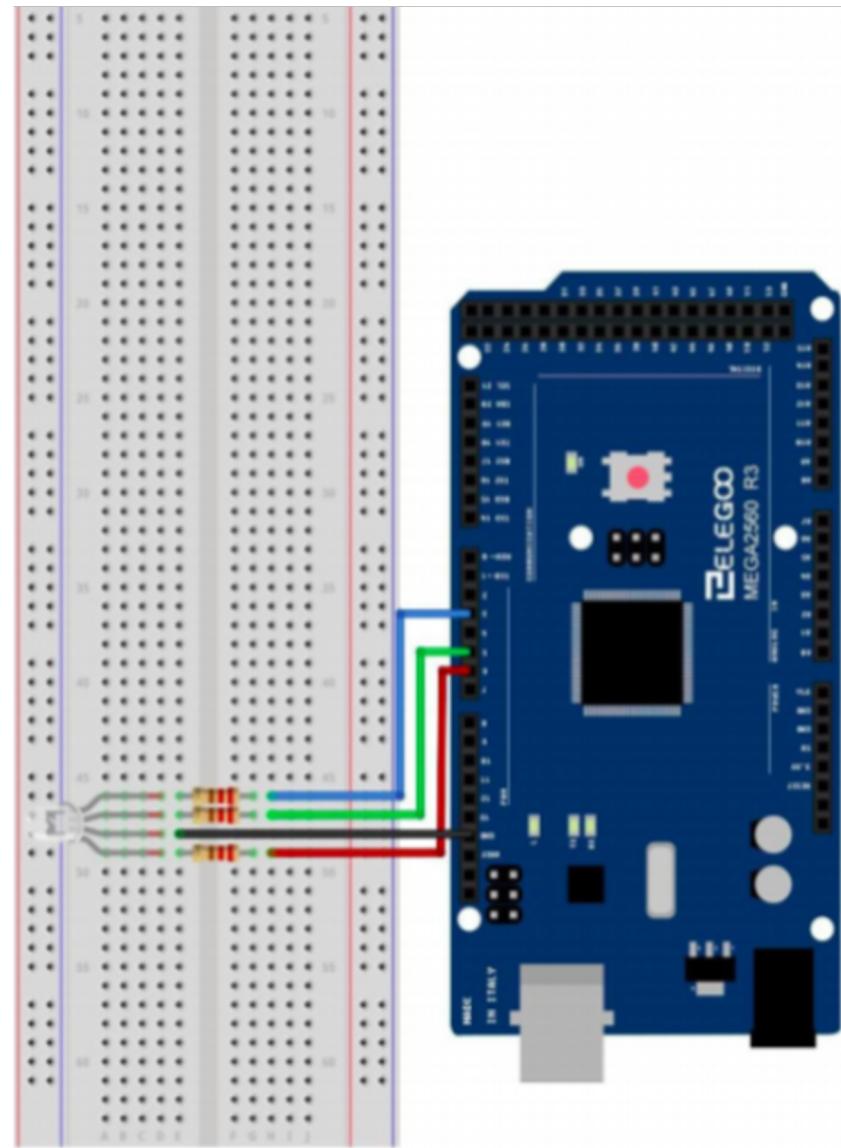
## Software

- Arduino IDE(version 1.6.4+)

# Wiring a RGB LED with Arduino

## *Set up*

- Connect the longest lead(common cathode) of the RGB to ground.
- Connect each of the other three leads to a  $220\Omega$  resistor
- These three positive leads of the LEDs (one red, one green and one blue) are connected to MEGA 2560 output pins using these resistors.



# Digital output with Arduino

## Code

→ Copy the following code to the IDE

```
// Define Pins  
#define BLUE 3  
#define GREEN 5  
#define RED 6  
  
void setup()  
{  
pinMode(RED, OUTPUT);  
pinMode(GREEN, OUTPUT);  
pinMode(BLUE, OUTPUT);  
digitalWrite(RED, HIGH);  
digitalWrite(GREEN, LOW);  
digitalWrite(BLUE, LOW);  
}  
  
// define variables  
int redValue;  
int greenValue;  
int blueValue;
```

# Digital output with Arduino

```
// main loop
void loop()
{
#define delayTime 20 // fading time between colors

redValue = 255; // choose a value between 1 and 255 to change
the color.
greenValue = 0;
blueValue = 0;

for(int i = 0; i < 255; i += 1) // fades out red bring green full when
i=255
{
redValue -= 1;
greenValue += 1;
// The following was reversed, counting in the wrong directions
// analogWrite(RED, 255 - redValue);
// analogWrite(GREEN, 255 - greenValue);
analogWrite(RED, redValue);
analogWrite(GREEN, greenValue);
delay(delayTime);
}
```

# Digital output with Arduino

```
redValue = 0;  
greenValue = 255;  
blueValue = 0;
```

```
for(int i = 0; i < 255; i += 1) // fades out green bring blue full when i=255  
{  
    greenValue -= 1;  
    blueValue += 1;  
    analogWrite(GREEN, greenValue);  
    analogWrite(BLUE, blueValue);  
    delay(delayTime);  
}
```

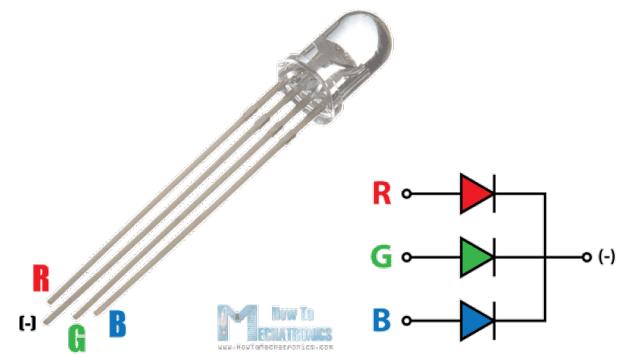
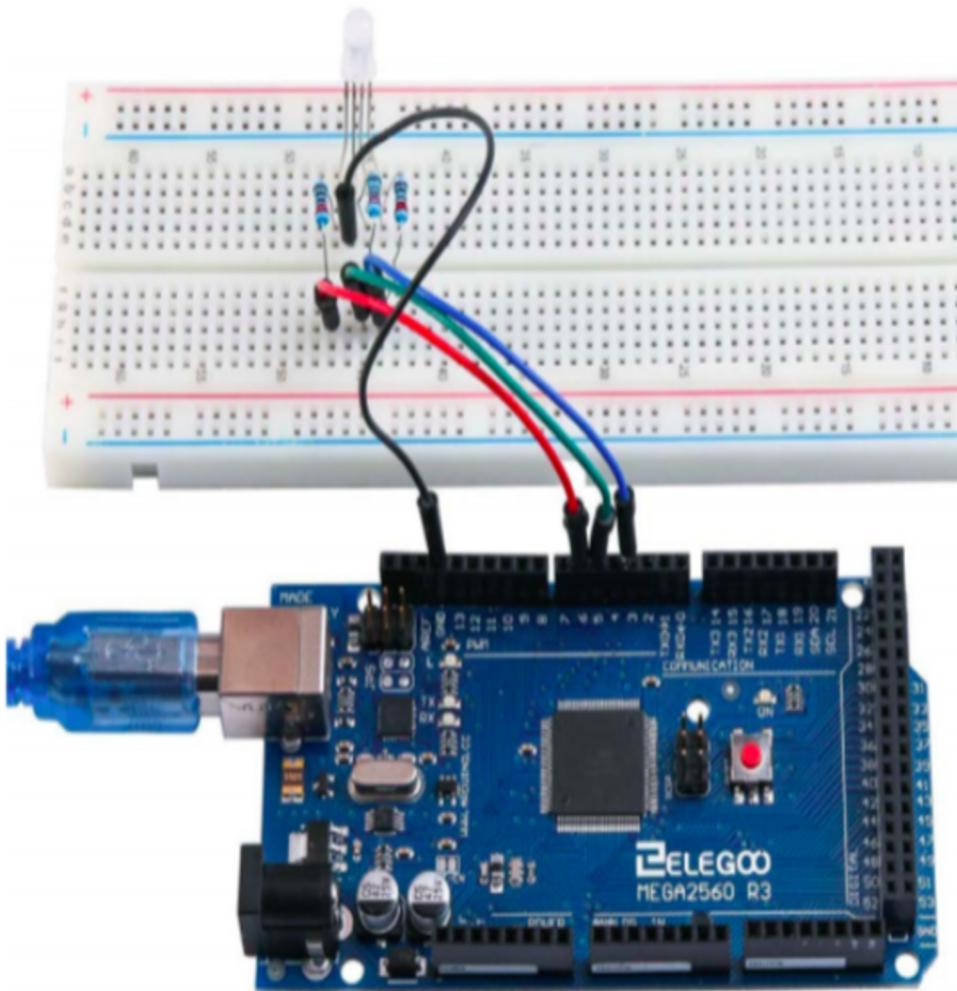
```
redValue = 0;  
greenValue = 0;  
blueValue = 255;  
}  
for(int i = 0; i < 255; i += 1) // fades out blue bring red full when i=255  
{
```

```
    blueValue -= 1;  
    redValue += 1;
```

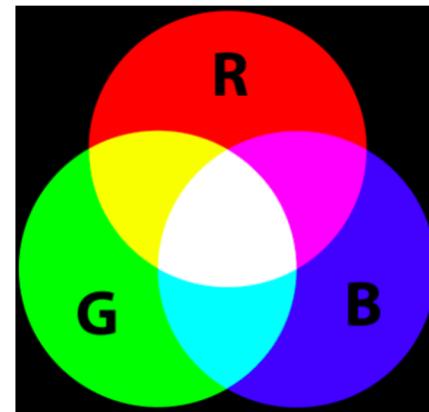
```
    analogWrite(BLUE, blueValue);  
    analogWrite(RED, redValue);  
    delay(delayTime);  
}
```

→ Upload

# Digital output with Arduino



The RGB LED they are like three regular LEDs in one; Red, Green, Blue.



Combines these colors to produce the orders

---

# Hands-on Activity - II

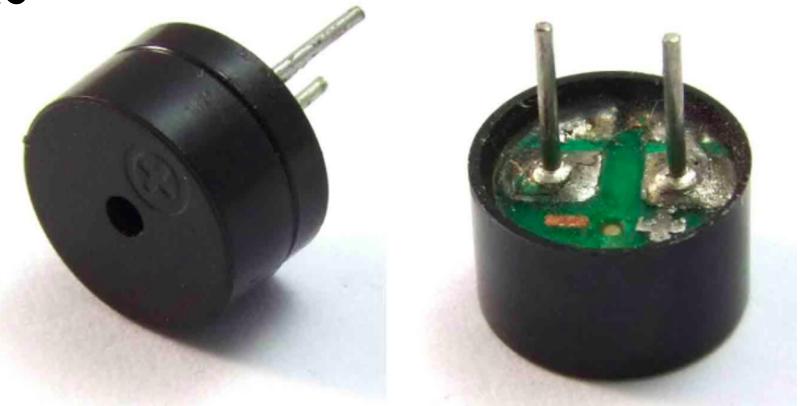
## MEGA2560

# Wiring a Buzzer with Arduino

We will introduce how to use a Passive Buzzer with the MEGA2560. You will generate eight different sounds, from Alto Do, Re, Mi, Fa, So, La, Si, to Treble Do.

## Hardware

- (1) x Elegoo Mega 2560 R3
- (1) x Passive buzzer
- (2) x F-M wires (Female to Male DuPont wires)



## Software

- [Arduino IDE\(version 1.6.4+\)](#)

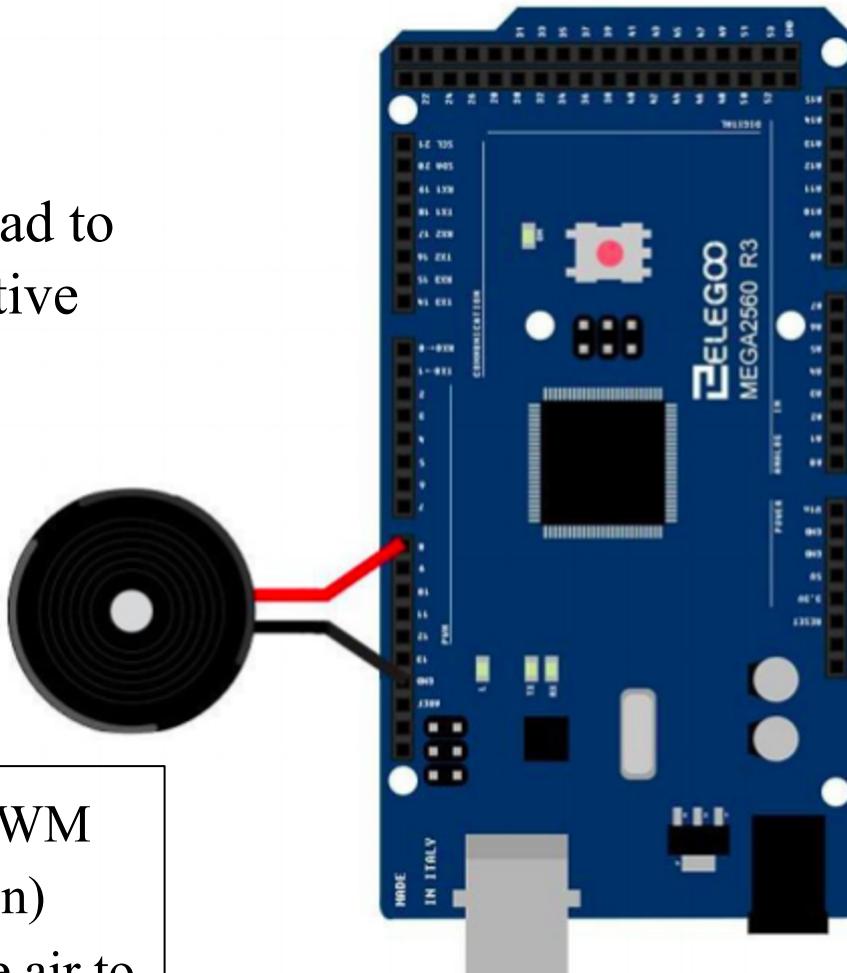
# Wiring a RGB LED with Arduino

## *Set up*

- Connect the positive lead to the pin 8, and the negative lead to the GND.

That's it!

The passive buzzer uses PWM (Pulse-Width Modulation) generating audio to make the air to vibrate. It works through pulses



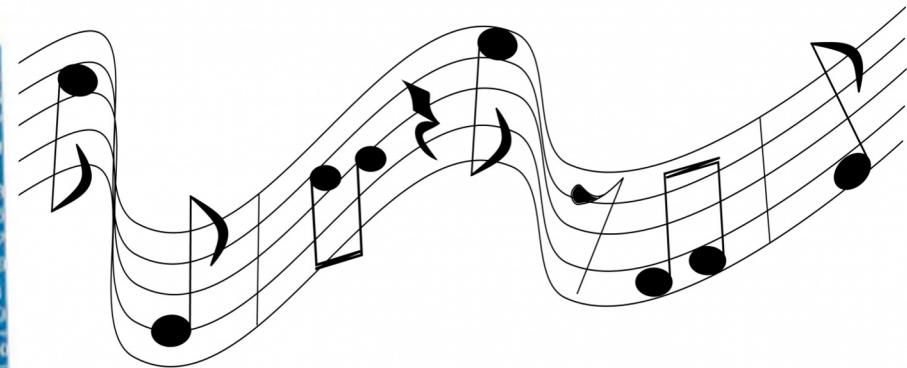
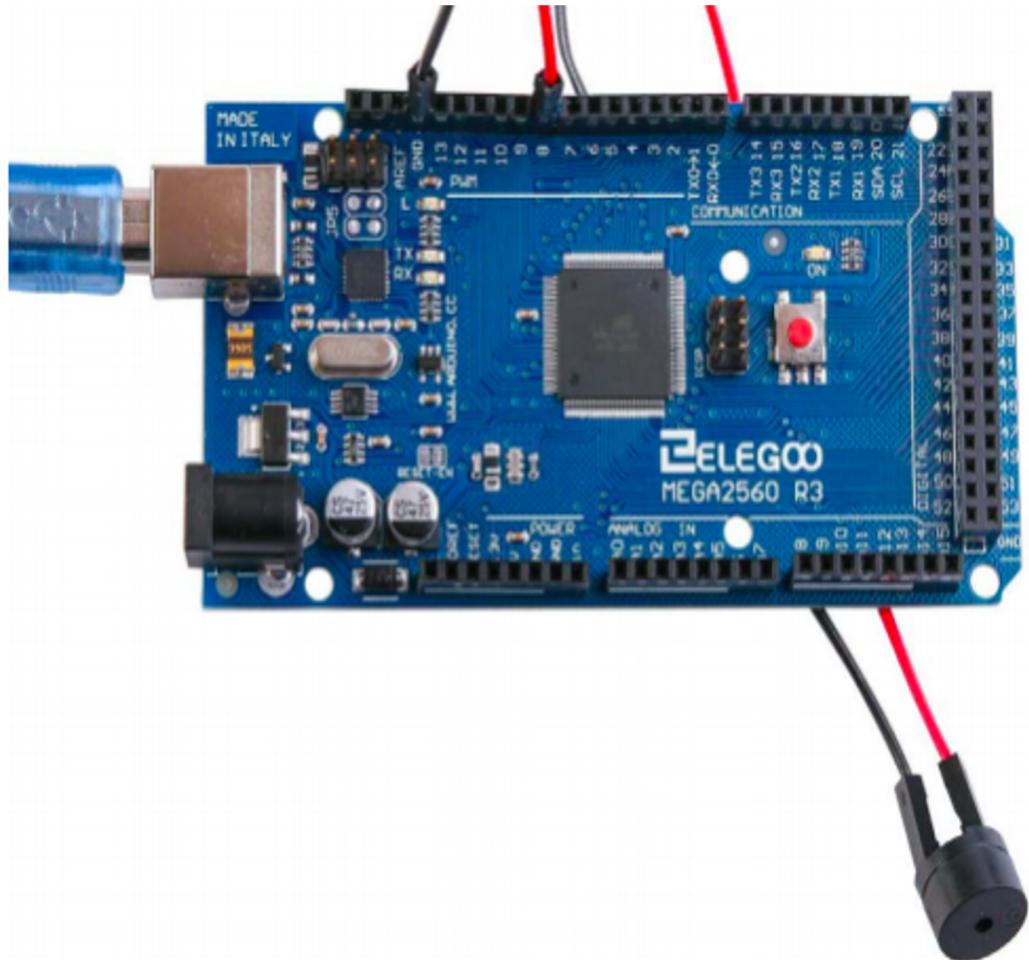
# Digital output with Arduino

## Code

→ Copy the following code to the IDE

```
#include "pitches.h"  
  
// notes in the melody:  
int melody[] = {  
    NOTE_C5, NOTE_D5, NOTE_E5, NOTE_F5,  
    NOTE_G5, NOTE_A5, NOTE_B5, NOTE_C6};  
int duration = 500; // 500 miliseconds  
void setup() {  
}  
void loop() {  
    for (int thisNote = 0; thisNote < 8; thisNote++) {  
        // pin8 output the voice, every scale is 0.5 sencond  
        tone(8, melody[thisNote], duration);  
        delay(500);  
    }  
    // restart after two seconds  
    delay(2000);  
}
```

Include Library  
Pitches.h



Frequency of vibrations  
change sound/note

*Example:*

*Alto Do (523Hz), Re (587Hz), Mi (659Hz), Fa (698Hz).....*