

Institute of Engineering and Technology
Devi Ahilya Vishwavidyalaya, Indore
Department of Electronics & Telecommunication



INTERNET OF THINGS
(ETR6E1)

Session January-April 2025

Submitted To:

DR. Uma Bhatt Mam

Submitted By:

(22T6010)-ANAND JATAV
(22T6036)-KRISHNA NIRMAL
(22T6037)-KULDEEP NAGAWA
(23T6092)-PAWAN PATIDAR

SIGNATURE

ETC-A (3RD YEAR)

INDEX

S.NO	TOPIC	SIGNATURE
1.	Assignment-1 INSTALLATION AND INTRODUCTION TO MATLAB AND THINGSPEAK CLOUD FOR WINDOWS	
2.	Assignment-2 Sensors Based Experiments on ARDUINO UNO Board with Details	
3.	Assignment-3 Hardware Project Description (ALL DETAILS)	

ASSIGNMENT-1

Introduction to Matlab & Thingspeak

MATLAB offline Installation Guidelines

Prerequisites.

1. Enrolling the MATLAB portal with your University Email ID.
2. Download the installer ISO file, mount it in your PC and open the installer file.

Step 1: Double click on the MATLAB icon (the binary file which we down loaded earlier). After clicking the icon, a pop-up will ask for the installer to run, click on the **Run**. The MathWorks Installer window will pop-up on the screen.

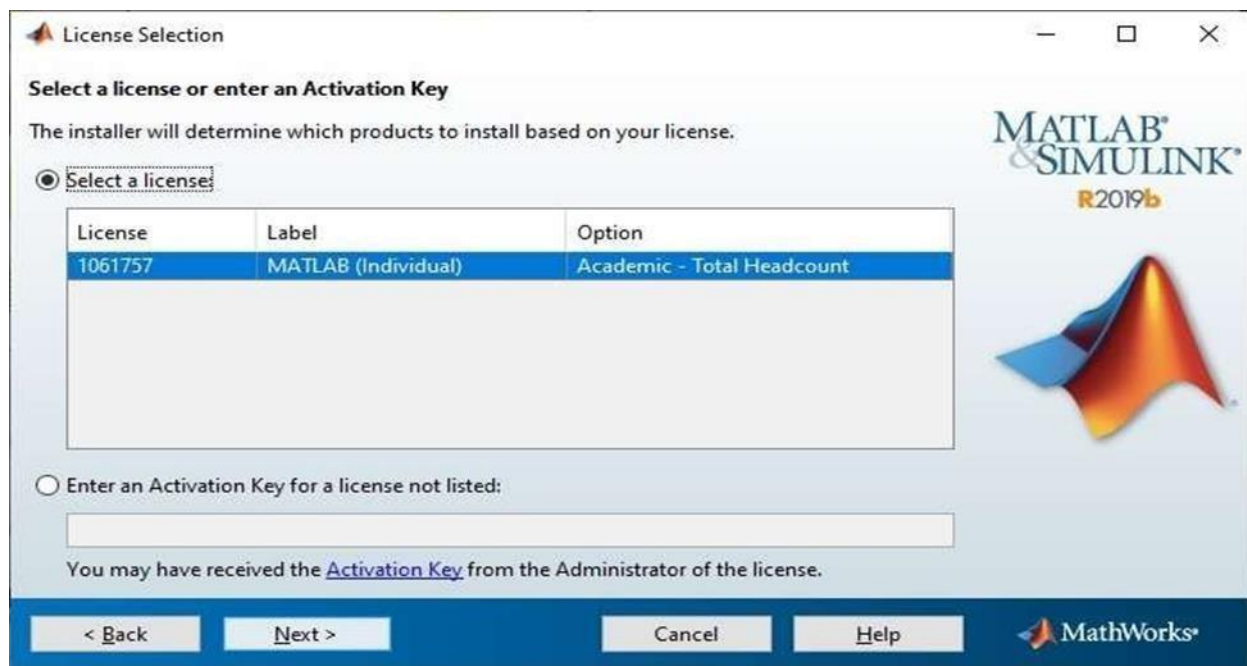
Step 2: By default, the first option, i.e., Log in with a MathWorks Account, is selected, proceed with this option and do remember to check your internet connection for proper installation of the MATLAB environment. Click on Next in the bottom of the window.



Step 3: Accept the license terms by selecting **Yes** in the next page and again click on the **Next** button.

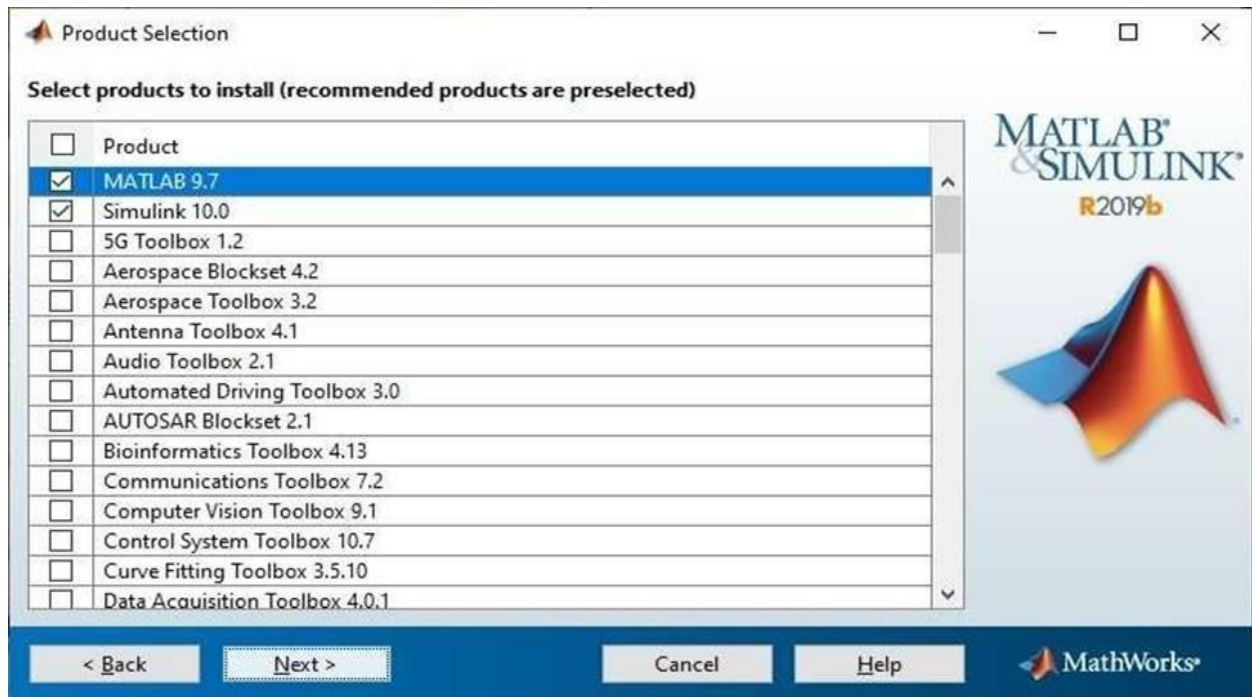
Step 4: A new page appears, by default, the first option is selected, Log in to your MathWorks Account. Enter here your **email id** and **password** that we created during the creation of our account with MathWorks. Refer to the image below and click on **next**.

Step 5: A license Selection window will appear, a preselected license id will be highlighted with a blue background. Select "Select a license:", select license 1061757 then click Next.

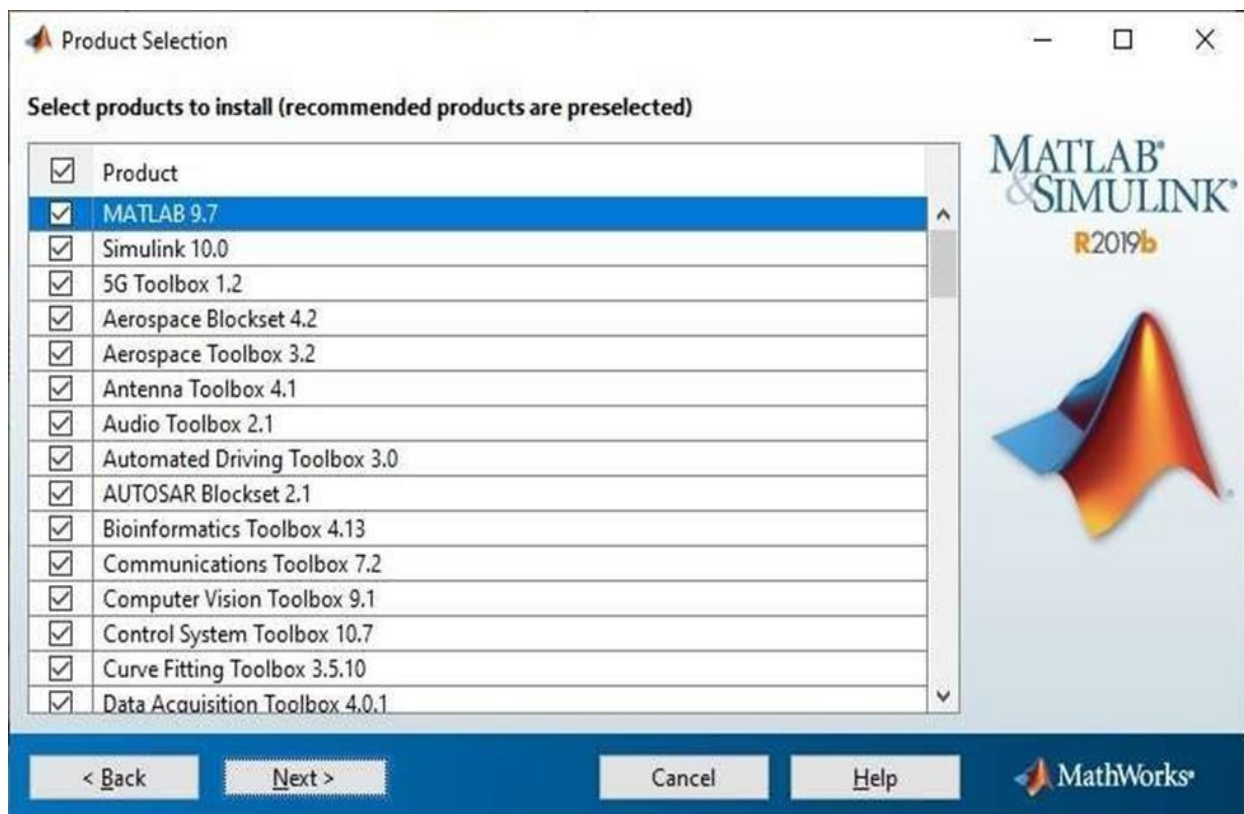


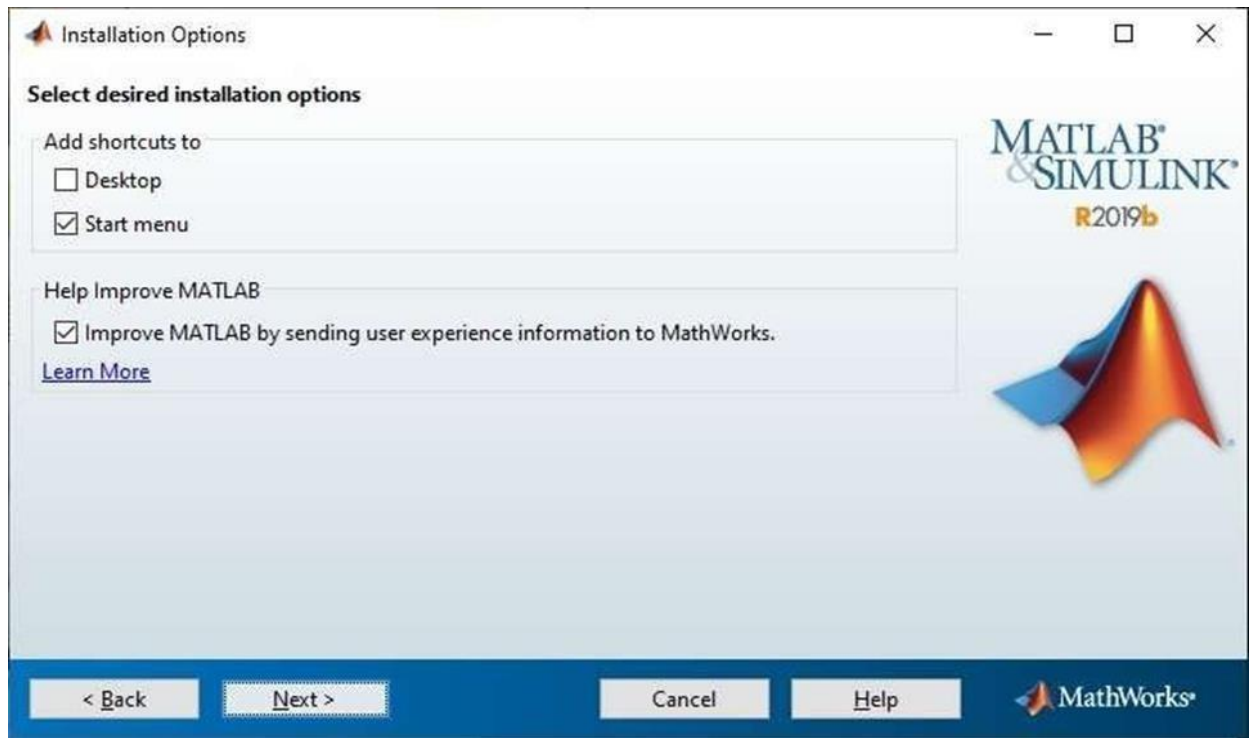
Step 6: A new Folder Selection window appears, no need to change the folder location for installation of MATLAB, click on Next.





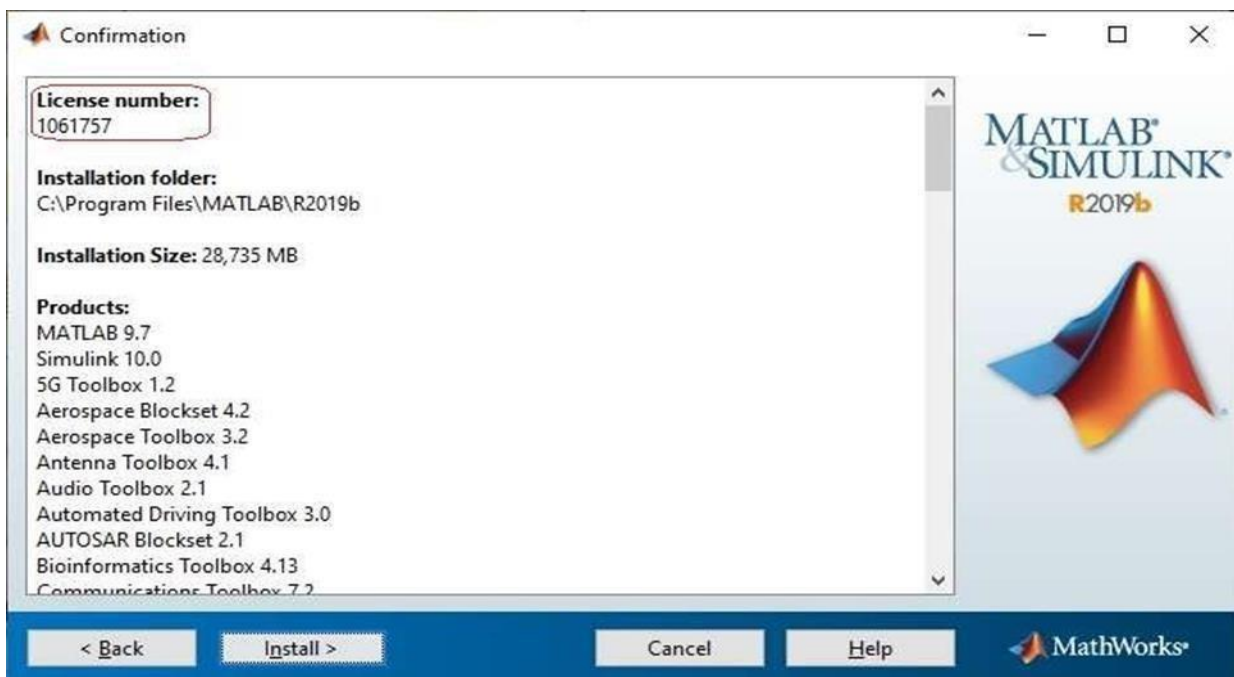
Step 7: Next is Product Selection window, the first product is MATLAB9.7, this is mandatory to select because it is the MATLAB environment, and from other products, you can choose as many of your choices and click on **Next**.





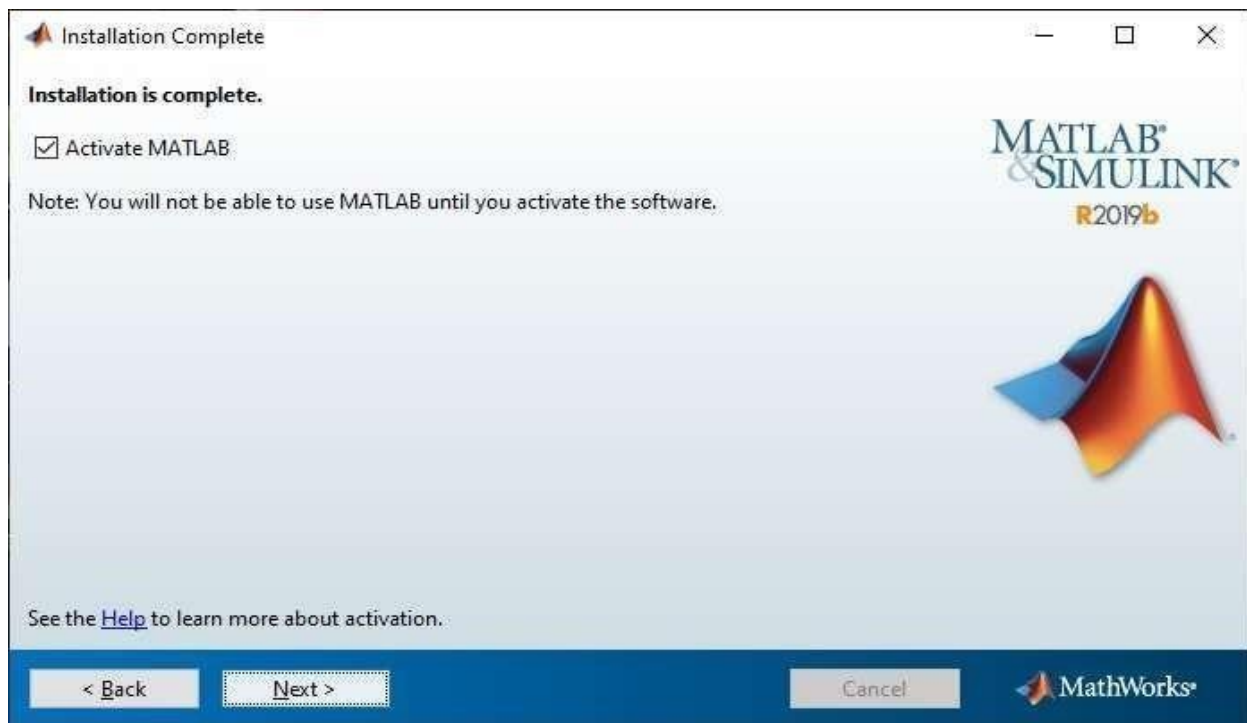
Step 8: Optional: Configure your additional shortcuts for MatLab, then click Next

Step 9: Next is the *Installation Options* window, select options as per your choice. Anytime you feel something to change, you can go back to the previous step by clicking on the Back button.

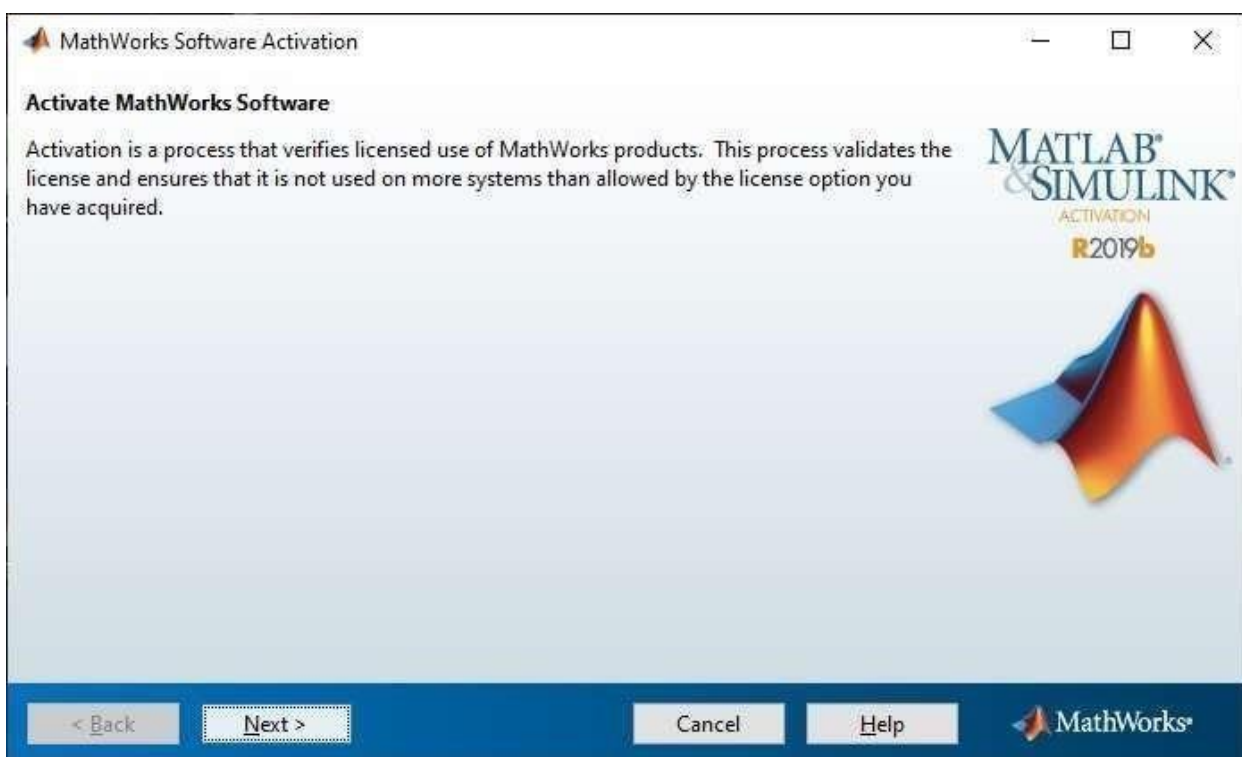


Step 10: Next is the Confirmation window, here you no need to do anything, confirm what you are going to download in the process of the installation of MATLAB, its other Add-on products, and what is the size of the downloads; and click on Install.

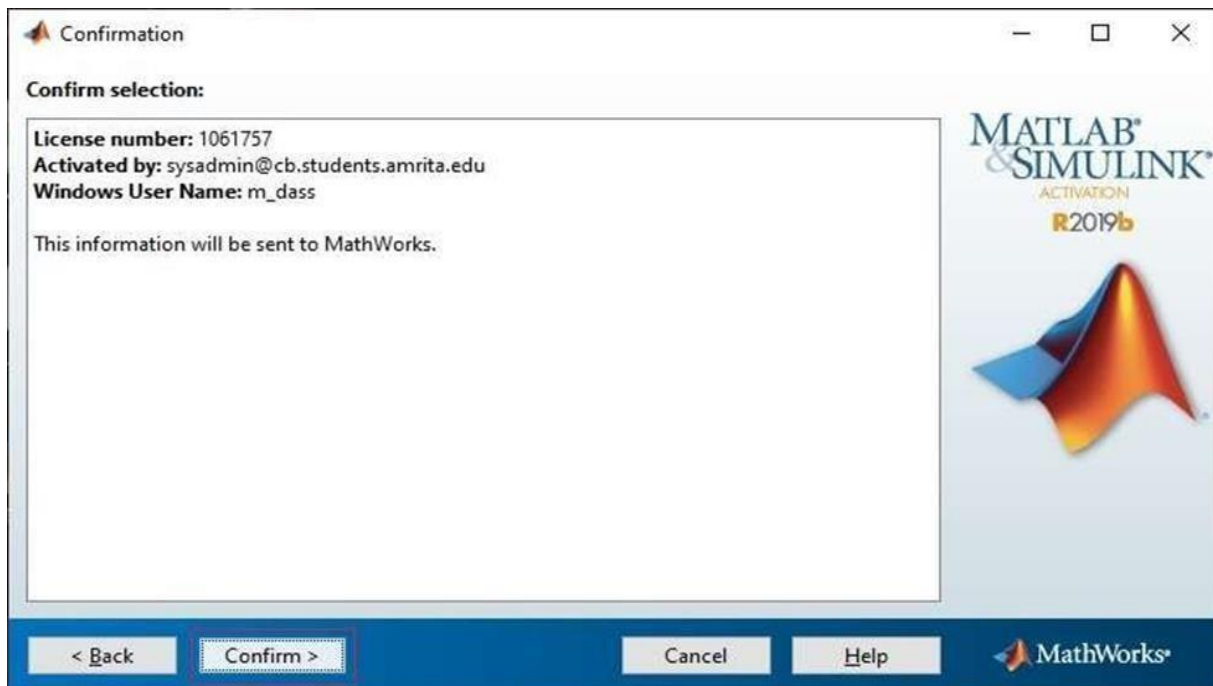
Step 11: After completion of the installation, a window appears that says to Activate the MATLAB, no need to do anything, click on the **Next** button.



Step 12: After clicking on **Next**, a new window appears that says about what is the meaning of activation. Proceed by clicking on *Next*.



Step 13: Make sure you enter the correct user name, then click Next.



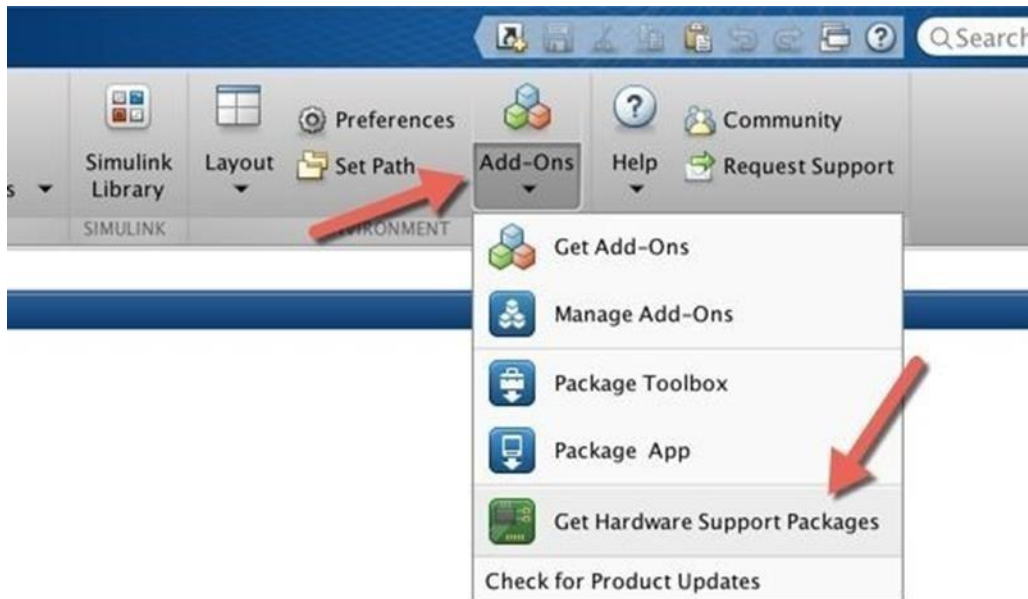
Step 14: Again, a new window appears displaying your **email id** and your products **license id**, proceed by clicking on **Confirm** button.

The installation process of MATLAB and other selected components have been completed successfully. Now click on the **Finish** button. Your MATLAB offline installation Is ready to use.

Setting up Arduino for MATLAB

In this section, we try to set up Arduino development board for MATLAB. You can configure MATLAB Support Package for Arduino hardware using MATLAB2014a or later. We also need internet connection to download this package.

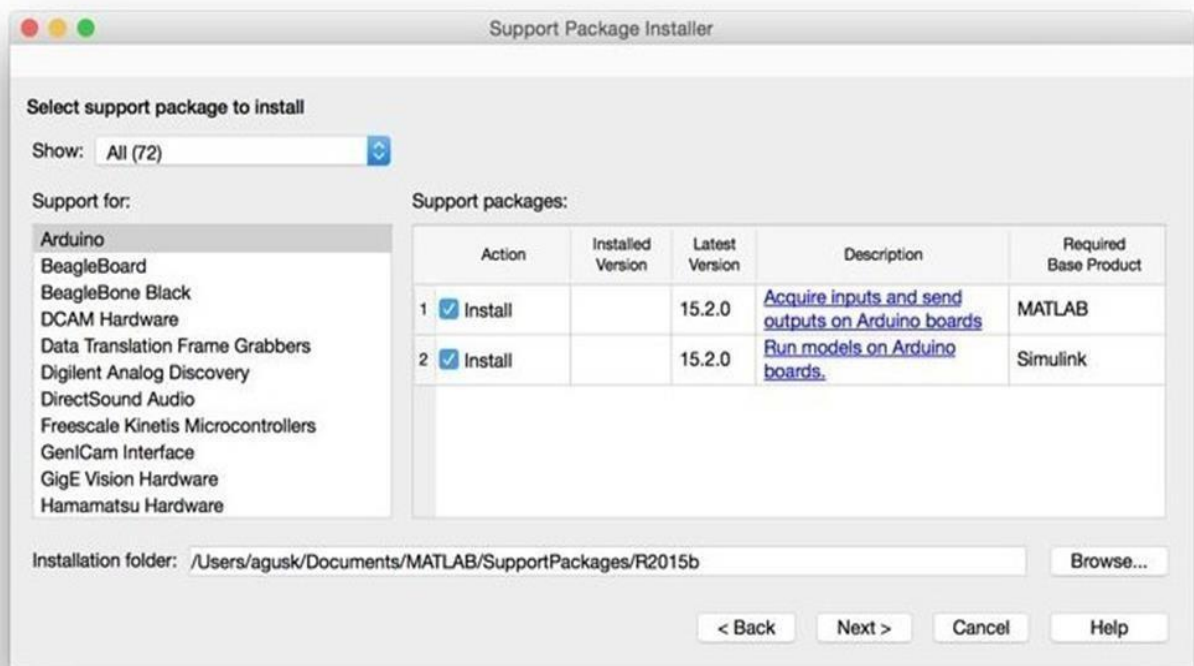
Run MATLAB application. Click **Get Hardware Support Packages** on **Add-Ons** icon on tool box.



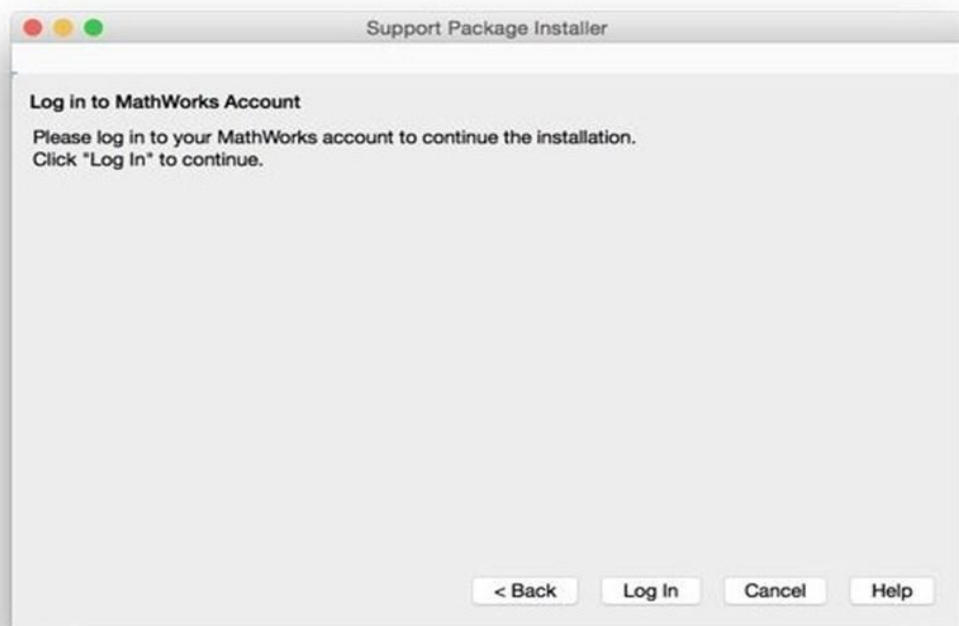
Then, you get a dialog. Select **Install from Internet**. If done, click **Next>** button.



Click on Arduino option and select both support packages. If done, click **Next>** button.



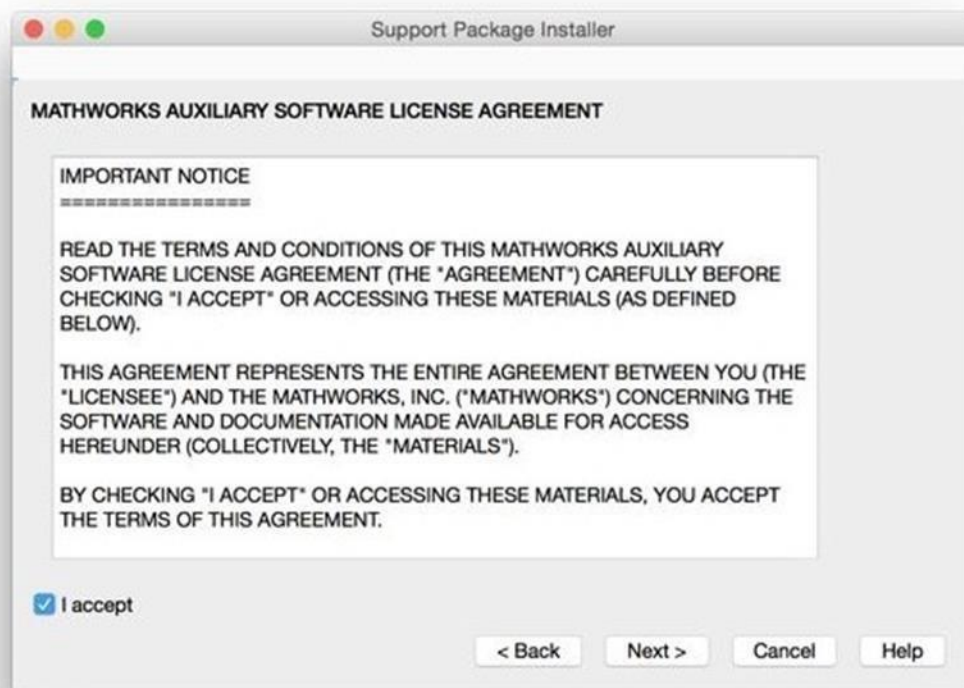
You will be asked to log on with your MATLAB account. You should have MATLAB license. Click **Log In** button.



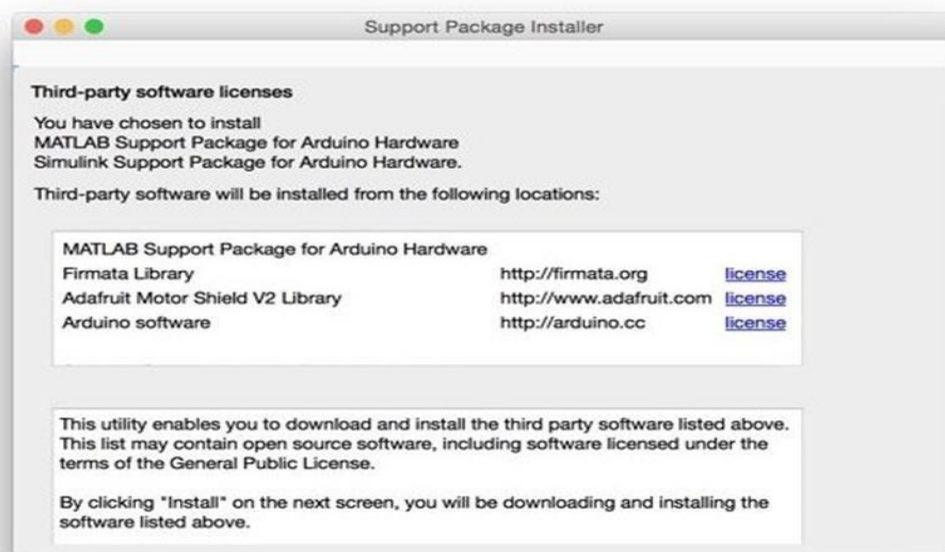
You will see the authentication dialog box. Fill in your login credentials, then click **Log In** button.



If success, you should get a software license agreement. Checkin **I accept** and then click **Next>**button.



You will get confirmation. Click **Next>** button.



Click **Install** button to start installation.



After done, you will be asked to configure Arduino board. Select Arduino and then, click **Next>** button.



Confirmation form will be shown. Click **Continue>** button.



The program will check if your platform needs Arduino driver or not. If you're working on Linux and Mac, you don't need a driver. You need to install Arduino driver if you're working on Windows platform. Click **Next >** button if done.

Installation is over. Click **Finish** button to close installation.

Connecting Arduino Board to Computer

Now you can connect Arduino board to computer. Then, verify which serial port is used for Arduino board. On Mac platform, you type this command.

```
$ls/dev/cu*
```

On Linux platform, you type this command.

```
$ls/dev/tty*
```

You can use Device Manager on Windows platform.

After that, you should see serial port of Arduino board which is attached on the computer. My OSX detected my Arduino board used/dev/cu.usb modem 1421 serial port.

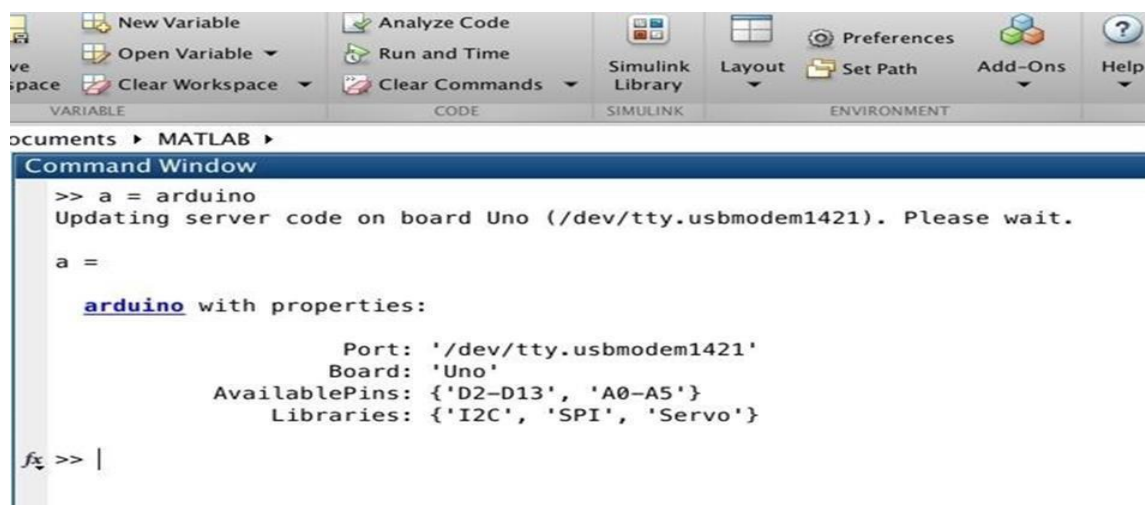


```
agusk$ ls /dev/cu*  
/dev/cu.Bluetooth-Incoming-Port /dev/cu.usbmodem1421  
/dev/cu.Bluetooth-Modem  
agusk$
```

On MATLAB command Windows, type this command

```
>>a = arduino
```

MATLAB will detect your Arduino board. You should detect Arduino board information on Matlab Command Window.

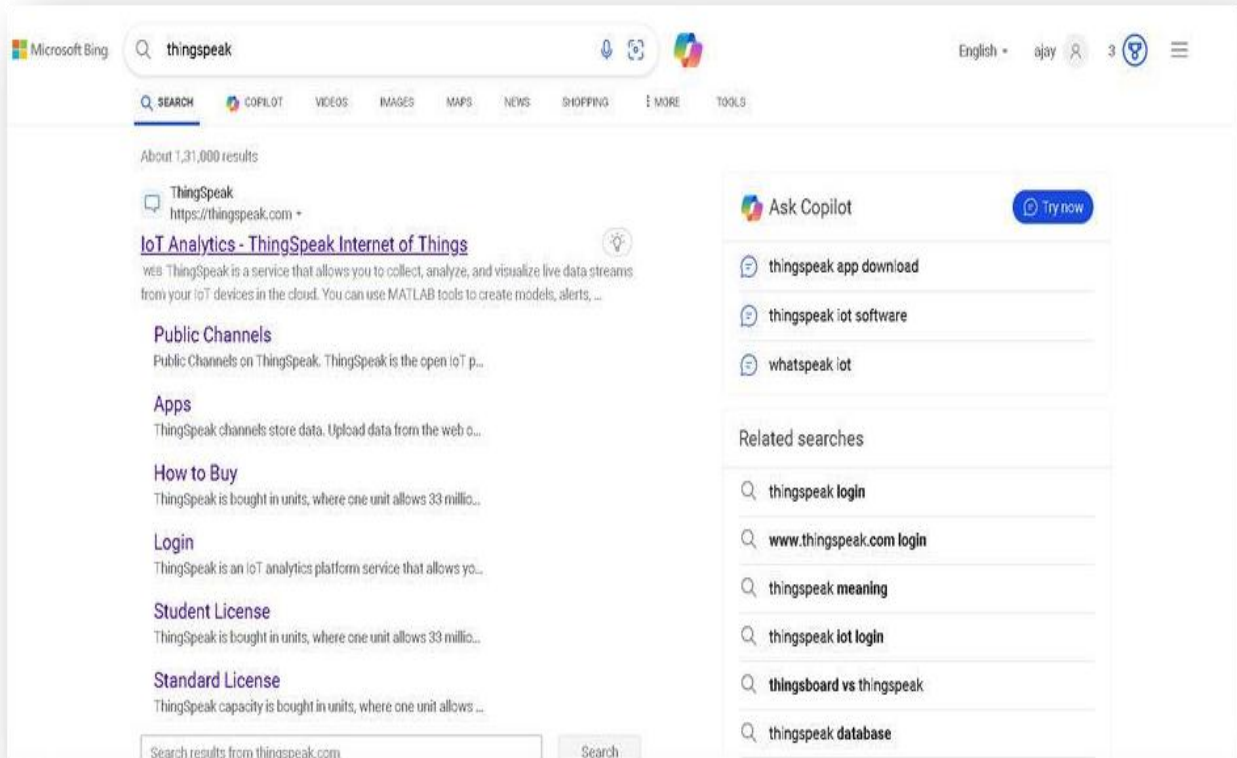


```
>> a = arduino  
Updating server code on board Uno (/dev/tty.usbmodem1421). Please wait.  
  
a =  
  
    arduino with properties:  
        Port: '/dev/tty.usbmodem1421'  
        Board: 'Uno'  
        AvailablePins: {'D2-D13', 'A0-A5'}  
        Libraries: {'I2C', 'SPI', 'Servo'}  
  
fx >> |
```


Thingspeak

How to use Thing speak and create channels (stepwise method):

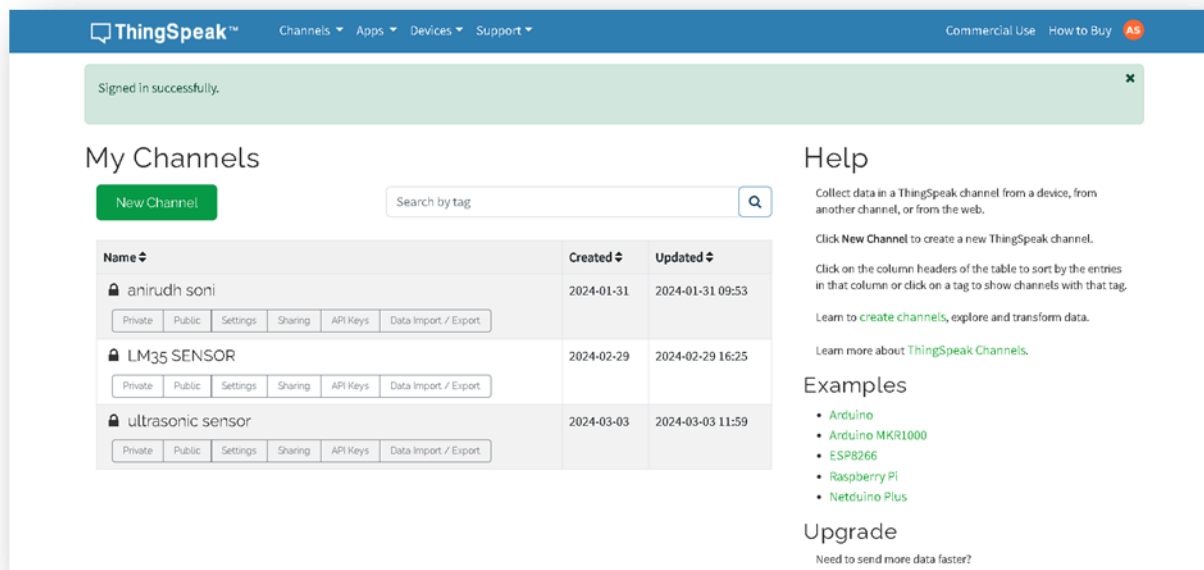
1. Open your Internet Browser and search Thingspeak.



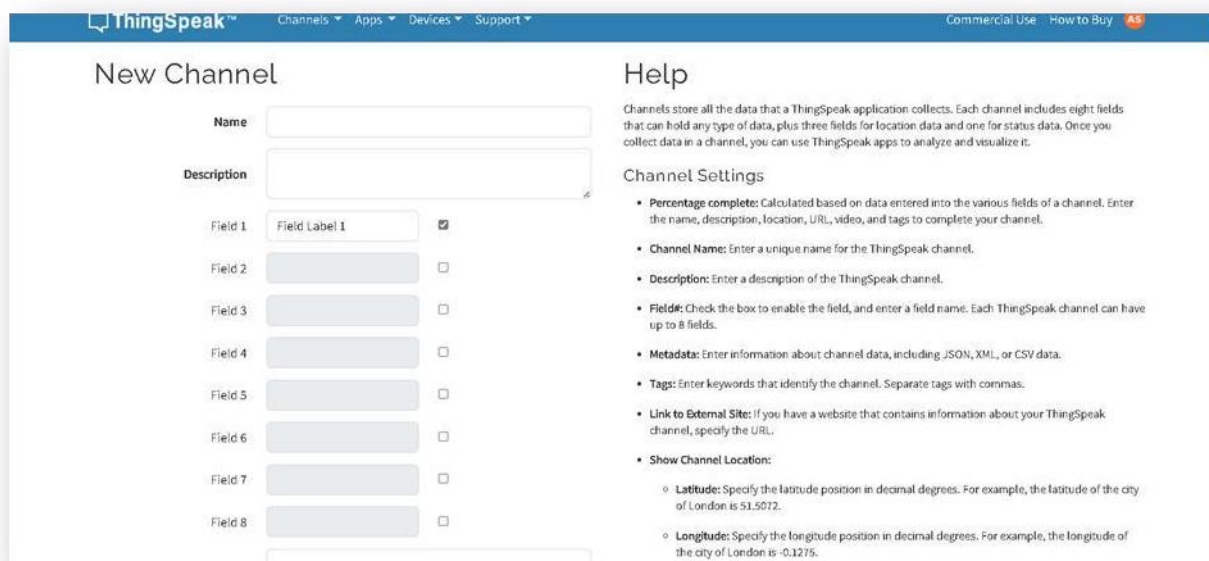
2. Click on the first link Thingspeak.
3. The user interface will look like as follows:



- Log in with your college email ID.
- After logging in the user interface will look like this:



- Click on New Channel.
- The following window will open:

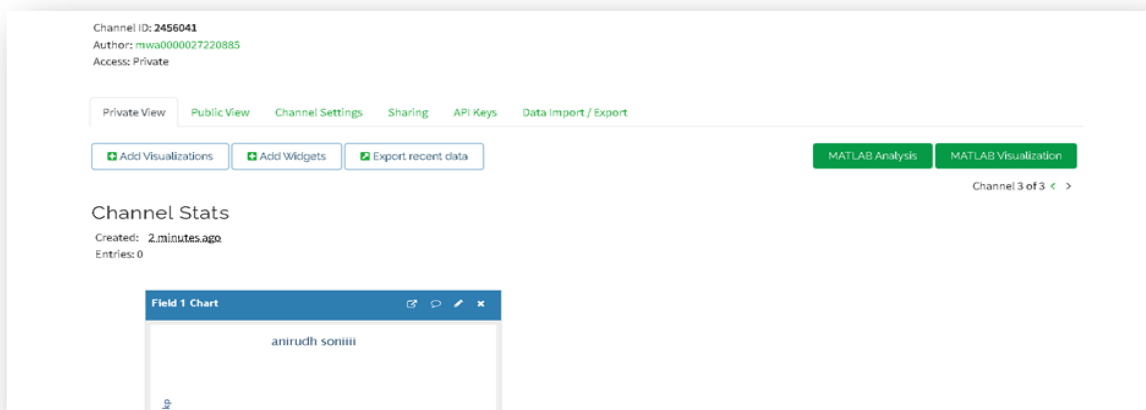


- Now give the name of the channel, description and the fields. The field column will contain the variable which is to be measured and can contain up to 8 variables, variables may include temperature, voltage, current, distance etc.

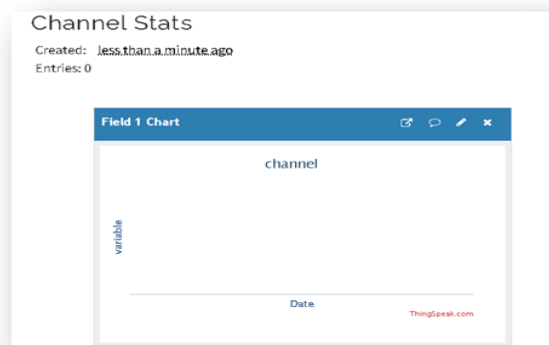
9. Click on save channel at the bottom after filling all the details.

The screenshot shows the 'Create Channel' form on the ThingSpeak website. The form includes fields for 'Link to GitHub' (with a placeholder 'https://github.com/'), 'Elevation', 'Show Channel Location' (checkbox), 'Latitude' (0.0), 'Longitude' (0.0), 'Show Video' (checkbox), 'Video URL' (http://), and 'Show Status' (checkbox). A green 'Save Channel' button is at the bottom. To the right, there is a 'Using the Channel' section with text explaining how to get data into a channel and a 'Learn More' link.

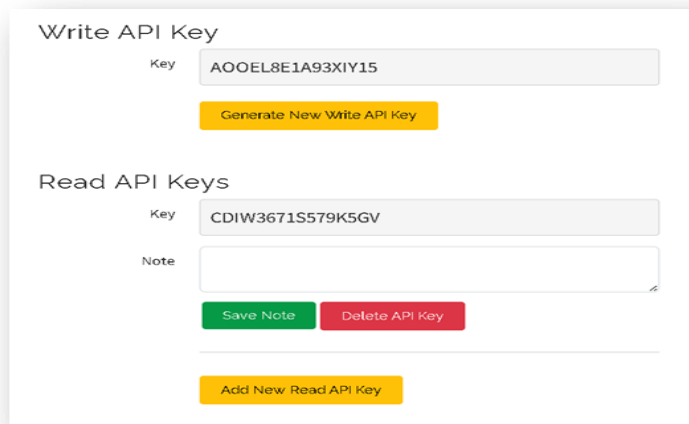
10. Now the following window will open:



11. The name of the channel and the variable which is to be measured is shown with date on the x-axis. Details of the channel is specified under the name of the channel.



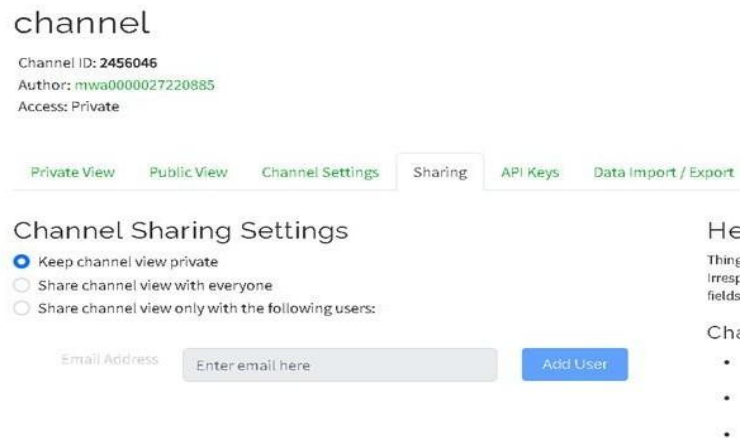
12. Now we take a look at the API keys.



The screenshot shows the 'Write API Key' and 'Read API Keys' sections of the Thingspeak interface. The 'Write API Key' section has a text input field containing 'A00EL8E1A93XIY15' and a yellow 'Generate New Write API Key' button. The 'Read API Keys' section has a text input field containing 'CDIW3671S579K5GV', a 'Note' text area, a green 'Save Note' button, a red 'Delete API Key' button, and a yellow 'Add New Read API Key' button at the bottom.

13. The following keys are used to communicate with the Thingspeak to send or write data to or from the cloud respectively.

14. In sharing options, you can choose who you want to share your channel with, either with a particular or sharing it with everyone by keeping your channel public or keeping your channel private.



The screenshot shows the 'channel' page with the 'Sharing' tab selected. The channel details are: Channel ID: 2456046, Author: mwa0000027220885, Access: Private. The 'Channel Sharing Settings' section has three radio button options: 'Keep channel view private' (selected), 'Share channel view with everyone', and 'Share channel view only with the following users:'. Below these is an 'Email Address' input field with the placeholder 'Enter email here' and an 'Add User' button. On the right side, there is a sidebar with a search bar and a list of channel items.

Connecting Matlab with Thingspeak

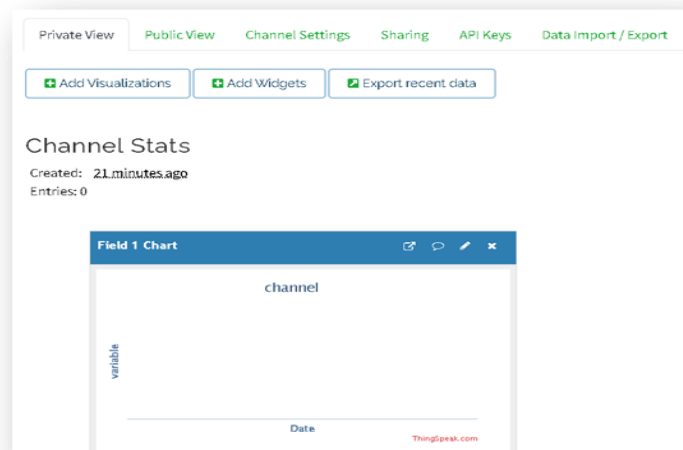
1. Open matlab command window.
2. In the command window type the following command:

```
ChannelID = 2454257;  
writeAPIkey = 'NP6QWGVKYOK1UPDI';  
readAPIkey = 'L5MTWQ20ODNFH2RG';
```
3. The following are the details of the channel that you have created.

```
writeData = thingSpeakWrite(ChannelID, variable, 'WriteKey', writeAPIKey);  
readData = thingSpeakRead(ChannelID, 'ReadKey', readAPIKey);
```

The following command will read the variable from the sensor and will publish it to your channel created in Thingspeak.

4. In Thingspeak you can view your data by clicking on private view in Thingspeak.



ASSIGNMENT-2

Sensors Based Experiments

Sensors step wise interfacing with Arduino using MATLAB

Interfacing LM35 with Arduino using MATLAB:

1. Pin Connection of LM35 with Arduino

LM35	Arduino
Vcc	5 volts
Ground	Ground
Aout	A0

Buzzer	Arduino
Positive Node	D12
Negative Node	Ground

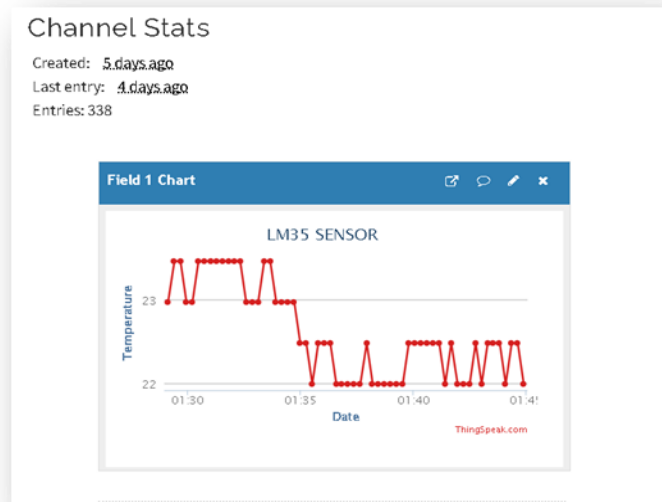
2. Code for interfacing LM35 with Arduino and displaying data on Thingspeak:

```
clear;
a = arduino();
ChannelID = 2423758;
writeAPIKey = 'I4ZMNIS1KEZRJF8B';
readAPIKey = 'MJIX26NSU3OHHITE';

while (1)
    voltage = readVoltage(a, A0);
    temp = voltage*100;
    writeData = thingSpeakWrite(ChannelID, temp, 'WriteKey', writeAPIKey);
    readData = thingSpeakRead(ChannelID, 'ReadKey', readAPIKey);
    pause(15);

    if readData >= 50
        writeDigitalPin(a, 'D12', 1);
    else
        writeDigitalPin(a, 'D12', 0);
    end
end
```


3. Graph of Temperature vs date graph on thingspeak:



Interfacing Flame Sensor with Arduino and displaying the voltage on Thingspeak:

1. Pin Connections

Flame Sensor	Arduino
Vcc	5 V
GND	GND
A0	A0

2. Code for interfacing Flame Sensor with Arduino and displaying data on Thingspeak:

```
clear;
a = arduino();
ChannelID = 2457997;
writeAPIKey = '5XKMOND3TFW7AL1W';
readAPIKey = 'MJIX26NSU3OHHITE';

while(1)
    voltage = readVoltage(a,'A0');
    flame = (5-voltage)*10;
    writeData = thingSpeakWrite(ChannelID, flame, 'WriteKey', writeAPIKey);
    readData = thingSpeakRead(ChannelID, 'ReadKey', readAPIKey);
    pause(15);

    if(readData>=20)
```

```

        writeDigitalPin(a,'D9',1);
    else
        writeDigitalPin(a,'D9',0);
    end
end
end

```

Interfacing ultrasonic sensor and a buzzer with Arduino using MATLAB and sending the data to Thingspeak:

1. Pin connections

Ultrasonic sensor	Arduino
Vcc	5V
GND	GND
Echo	D11
Trigger	D12

Buzzer	Arduino
Positive terminal	D9
Negative terminal	GND

2. Code for interfacing Ultrasonic Sensor with Arduino:

Before Writing the code for ultrasonic sensor you have to go to add-on in the MATLAB window and click on manage add-ons and program the Arduino by adding ultrasonic libraries.

```

clear;
a = arduino();
ChannelID = 2454257;
writeAPIkey = 'NP6QWGVKYOK1UPDI';
readAPIkey = 'L5MTWQ20ODNFH2RG';
ultrasonicObj = ultrasonic(a, 'D12', 'D11');

while(1)
    distance = readDistance(ultrasonicObj);
    writeData = thingSpeakWrite(ChannelID, distance, 'WriteKey', writeAPIKey);
    readData = thingSpeakRead(ChannelID, 'ReadKey', readAPIKey);
    pause(10);
end

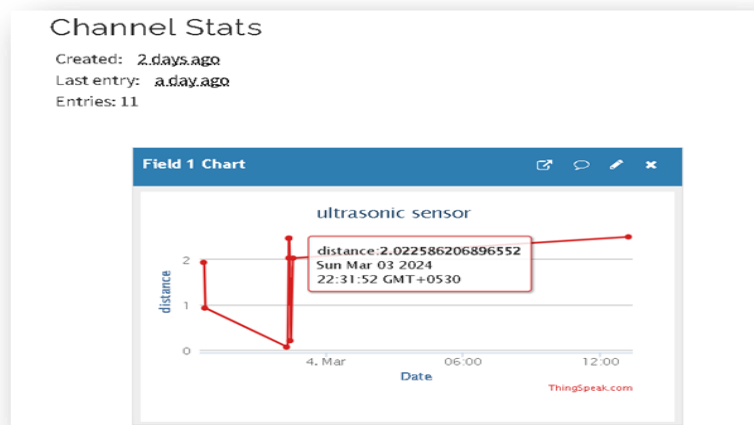
```

```

if (readData >= 0.5)
    writeDigitalPin(a, 'D9', 1);
    pause(5);
    writeDigitalPin(a, 'D9', 0);
else
    writeDigitalPin(a, 'D9', 0);
end
end
end

```

3. Graph of distance on y-axis with date on x-axis :



Interfacing IR sensor, LDR sensor, with Arduino and displaying data on Thingspeak:

1. Pin connections

IR sensor	Arduino
Vcc	5 V
GND	GND
OUT	D2

LDR sensor	Arduino
One leg	5V
Other Leg	Junction of 10K resistor & A0 of Arduino
10kohm other leg	Ground of Arduino

2. Code for interfacing IR & LDR Sensor with Arduino:

```
clear;
a = arduino();
ChannelID = 2456439;
writeAPIkey = 'LB0IJ3ZCZFAF5G23';
readAPIkey = '7U436XWS4D68S9ER';

while(1)
    IRValue = readDigitalPin(a, 'D2');
    writeData = thingSpeakWrite(ChannelID, IRValue, 'WriteKey', writeAPIKey);
    readData = thingSpeakRead(ChannelID, 'ReadKey', readAPIKey);
    pause(15);
end
```

This part includes the command used to interface LDR sensor with Arduino :

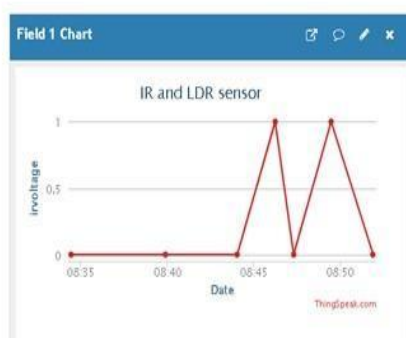
```
a = arduino();
channelID = 123456;
writeAPIKey = 'LB0IJ3ZCZFAF5G23';
readAPIKey = '7U436XWS4D68S9ER';

while(1)
    ldrValue = readVoltage(a, A0);
    thingSpeakWrite(channelID, ldrValue, 'WriteKey', writeAPIKey);
    pause(15);
end
```

3. Graph of Voltage and light intensity vs date:

Channel Stats

Created: about 18 hours ago
Last entry: about 17 hours ago
Entries: 7



ASSIGNMENT -3

HARDWARE PROJECT DISCRIPTION

TOPIC - IOT Enabled Street Light Fault Detection System

Basic Project Idea- Street lights evolving from oil lamps, gas lightings, to CFLs (Compact Fluorescent Lights), to Sodium Vapor Lights and now LEDs (Light Emitting Di- odes) are one of the most prominent source for illumination of streets, localities and campuses. However, faults in street lights left undetected results in accidents robbery/thefts etc. Here, we have developed a system to automatically detect the fault in street lights with the use of sensors and send the data to the concerned authorities through IoT networks resulting in faster and easier repairs.

The fault in a street light (if any) is detected with the use of LDR sensors, placed in direct vicinity of the LED of the street light, which senses whether the LED is glowing during the dark or not. If the surrounding is dark and the LED still doesn't glow up it causes a change in the voltage drop of the LDR and the fault is detected. This information is then uploaded to a cloud (THINGSPEAK) which can be accessed by the authorities/personnel responsible for the maintenance of the lights.

Components Required:

1. Zero PCB Board
2. ESP-8266 Microcontroller Module
3. L298N Motor Driver
4. BO
5. BO-Motor Wheels
6. 12v Power Supply

1. Zero PCB Board

The **Zero PCB (Printed Circuit Board)** is a general-purpose board used for prototyping electronic circuits. It comes with a grid of pre-drilled holes and copper pads that allow easy mounting and soldering of components. It doesn't have pre-designed tracks, so you manually connect the components with jumper wires or solder.

2. ESP8266 Microcontroller Module

The **ESP8266** is a low-cost Wi-Fi-enabled microcontroller developed by Espressif Systems. It features:

- A 32-bit processor (Tensilica L106),
- Support for 802.11 b/g/n Wi-Fi,
- GPIO pins for interfacing with sensors and actuators,
- Built-in flash memory,

- Ideal for IoT and automation projects.

Most commonly used in the **NodeMCU** development board format.

3. L298N Motor Driver

The **L298N** is a dual H-Bridge motor driver IC that allows you to control the direction and speed of **two DC motors** or **one stepper motor**. Features include:

- Supports motors powered up to 46V and 2A current,
- Built-in heat sink for cooling,
- Control via logic-level inputs (IN1, IN2, etc.),
- Onboard 5V regulator to power logic circuits.

4. BO (Battery Operated) Motor

BO Motors are small, low-torque, DC geared motors commonly used in robotics. These are:

- Operated typically at 3V–12V,
- Lightweight and compact,
- Easy to mount on chassis,
- Suitable for driving lightweight robots or cars.

5. BO Motor Wheels

These are wheels designed to fit snugly onto the shaft of BO motors. They provide mobility to the robot or device and are:

- Made of plastic or rubber,
- Easy to attach/detach,
- Come in various sizes (typically 65mm diameter for basic projects).

6. 12V Power Supply

This provides the required voltage and current to run all electronic modules and motors.

- It powers components like the L298N and motors,
- May be a battery pack, adapter, or power brick,
- Needs to be regulated to prevent over-voltage to sensitive components like the ESP8266 (which runs on 3.3V).

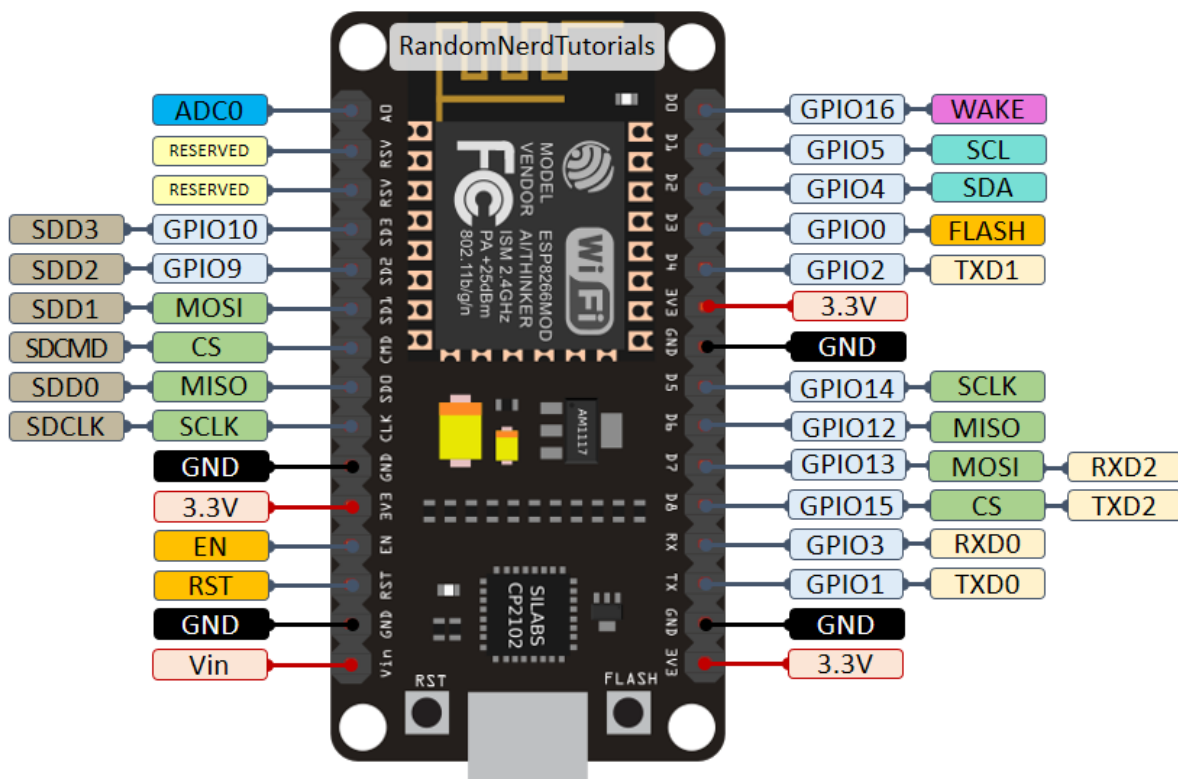
ESP8266 MODULE

ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability, designed by Espressif Systems and built using 40 nm technology.

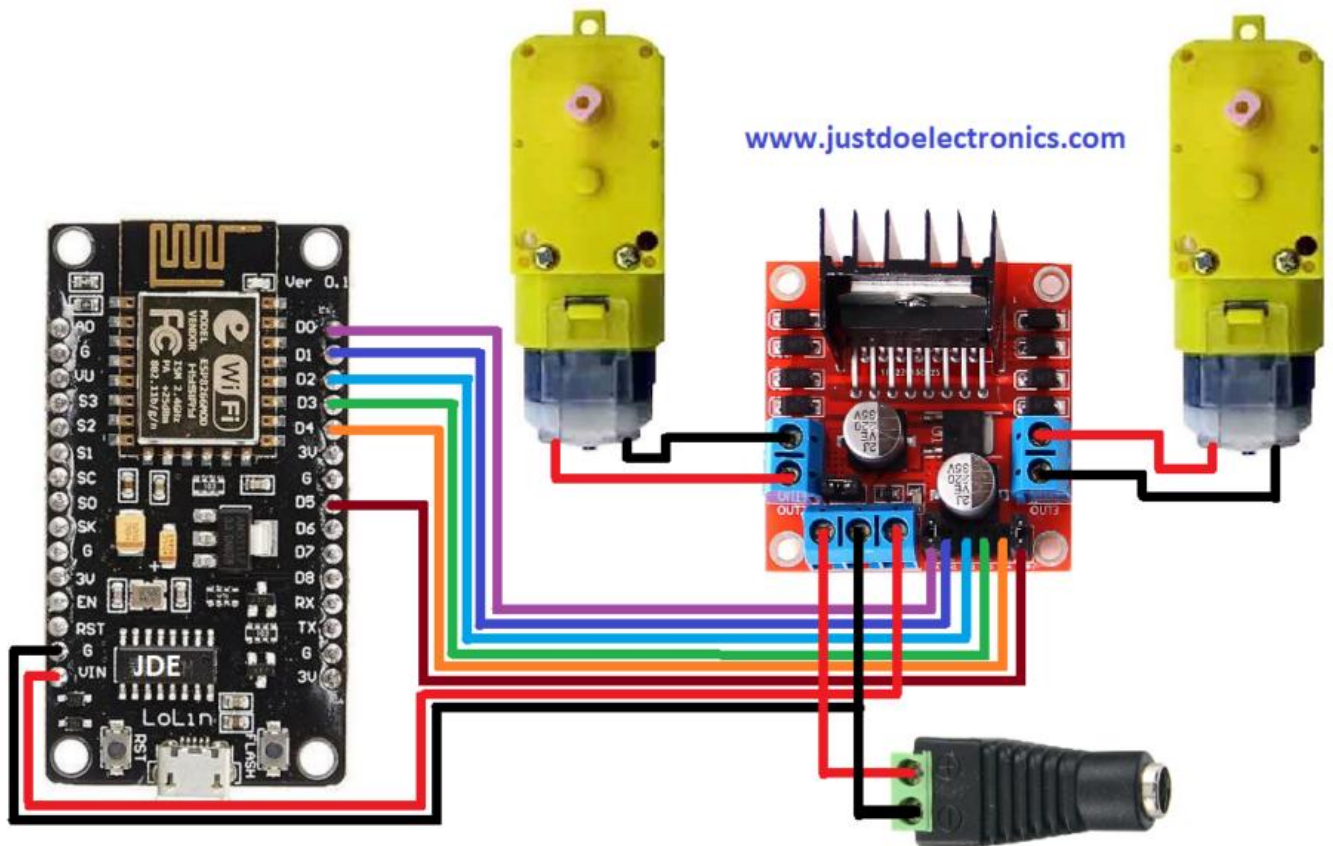
- **32-bit Tensilica L106 Microprocessor**, clocked at up to 80 MHz (can be overclocked to 160 MHz).
- **160 KB of SRAM, 64 KB of instruction RAM, and 96 KB of data RAM.**
- **Flash memory** support (typically 4 MB external flash on NodeMCU boards).
- **802.11 b/g/n Wi-Fi connectivity** with speeds up to 72.2 Mbps (20 MHz channel width).
- **No built-in Bluetooth support.**
- **Up to 17 GPIO pins** (some are shared with other functions).

- **1 × 10-bit ADC channel** (used for analog input).
- **Serial Connectivity includes:**
 - 2 × SPI (1 usable),
 - 1 × I2C (software-implemented),
 - 1 × UART (plus 1 for flashing/debugging).
- **No native Ethernet MAC** (Wi-Fi only).
- **PWM support on all digital pins.**
- **Supports secure connections with SSL/TLS.**
- **Basic cryptographic capabilities via software libraries.**

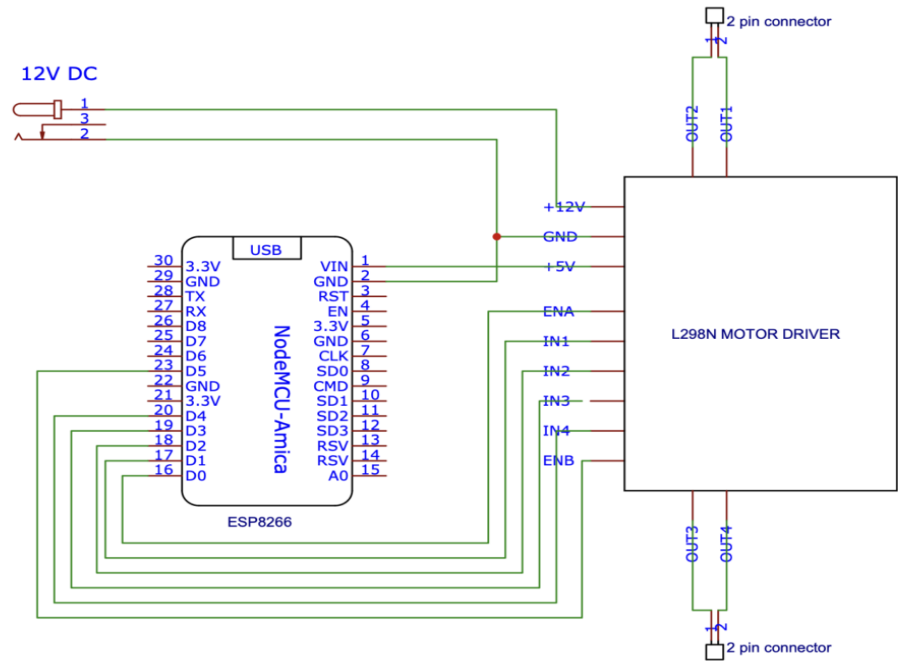
- **PINLAYOUT STRUCTURE OF ESP-8266 MODULE:**



Circuit Diagram



CIRCUIT DIAGRAM(PROTEUS)



RELATED CODE FOR PROJECT:

```
#define BLYNK_TEMPLATE_ID "TMPL3woQQcdAe"
#define BLYNK_TEMPLATE_NAME "WIFI CAR"
```

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
```

```
// Motor Driver Pins
```

```
#define ENA D2
#define IN1 D6
#define IN2 D7
#define ENB D8
#define IN3 D3
#define IN4 D4
```

```
// WiFi Credentials
```

```
char auth[] = "HVAhbU8fEwWV1nSm9fo1DjLzfpJjnaMn";
char ssid[] = "Redmi Note 9 Pro";
```

```

char pass[] = "kpnagawa";

int Speed = 0; // Default Speed
bool forward = 0, backward = 0, left = 0, right = 0;

void setup() {
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);

  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

BLYNK_WRITE(V1) { forward = param.asInt(); moveCar(); } // Forward
BLYNK_WRITE(V4) { backward = param.asInt(); moveCar(); } // Backward
BLYNK_WRITE(V2) { left = param.asInt(); moveCar(); } // Left
BLYNK_WRITE(V3) { right = param.asInt(); moveCar(); } // Right
BLYNK_WRITE(V0) { Speed = param.asInt(); } // Speed Control

void moveCar() {
  if (forward) {
    carForward();
  } else if (backward) {
    carBackward();
  } else if (left) {
    carLeft();
  } else if (right) {
    carRight();
  } else {
    carStop();
  }
}

void carForward() {

```

```

    analogWrite(ENA, Speed);
    analogWrite(ENB, Speed);
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
}

void carBackward() {
    analogWrite(ENA, Speed);
    analogWrite(ENB, Speed);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
}

void carRight() {
    analogWrite(ENA, Speed / 2);
    analogWrite(ENB, Speed);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
}

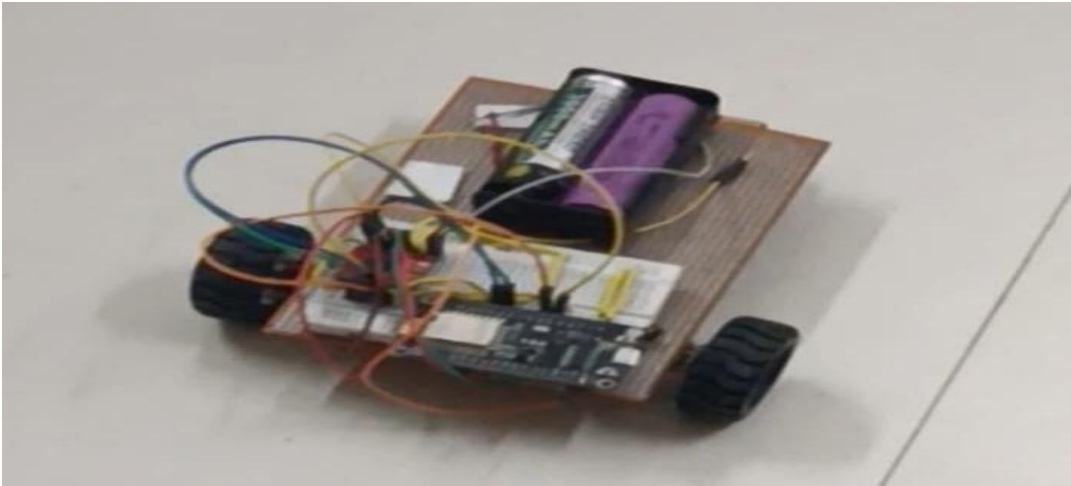
void carLeft() {
    analogWrite(ENA, Speed);
    analogWrite(ENB, Speed / 2);
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
}

void carStop() {
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);}

```

```
void loop() {  
    Blynk.run();  
}
```

RESPECTIVE OUTPUT OF THE PROGRAM:



Viva Questions

1. What is the role of the ESP8266 NodeMCU in this Wi-Fi car project?

The ESP8266 NodeMCU acts as the main controller and Wi-Fi module in the project. It performs two main tasks:

- It creates a Wi-Fi network (Access Point mode) so the user can connect to it using a mobile phone.
- It receives commands (like Forward, Backward, Left, Right) from a web interface and sends control signals to the motor driver (L298N) to move the car accordingly.

It runs on 3.3V logic and has built-in GPIO pins and flash memory to store and run the control program (Arduino code).

2. Why do we use an L298N motor driver module with the NodeMCU?

The L298N motor driver is used because NodeMCU cannot directly drive motors due to low current output from its GPIO pins. L298N can:

- Control two DC motors independently.
- Handle higher voltage (up to 46V) and current (up to 2A per channel).
- Control both speed (via PWM) and direction (via logic inputs) of the motors. It also has an onboard 5V regulator which can supply power to NodeMCU.

3. Which GPIO pins of NodeMCU are connected to L298N and what are their functions?

In the project, the NodeMCU controls the L298N using the following GPIO pins:

- D5 (GPIO14) → ENA (enables Motor A)
- D6 (GPIO12) → ENB (enables Motor B)
- D7 (GPIO13) → IN1, D8 (GPIO15) → IN2 (control Motor A direction)
- D3 (GPIO0) → IN3, D4 (GPIO2) → IN4 (control Motor B direction)

ENA/ENB pins control motor speed using PWM signals.

IN1–IN4 control the direction of motor rotation.

4. How is Wi-Fi connectivity established and used to control the car?

The NodeMCU runs in Access Point (AP) mode, creating its own Wi-Fi hotspot (e.g., “WiFi_Car”).

- The user connects their smartphone to this Wi-Fi.
- NodeMCU serves a simple web page interface (HTML code stored in flash).
- When a button (e.g., "Forward") is pressed, it sends an HTTP request to the NodeMCU.
- NodeMCU reads the request and sends signals to the motor driver accordingly.

No internet is required—this is a local Wi-Fi connection.

5. Is any app like Blynk used for control? If not, how is the car controlled?

No, the project does not use Blynk or any external app. Instead:

- The NodeMCU hosts its own control web page.

- The web page has buttons for car directions.
 - The user controls the car directly via a browser on their phone by tapping the buttons.
- This method avoids third-party apps and works even without internet.

6. Why is a 12V power supply used and how is voltage managed?

The 12V power supply is required to run the DC motors through the L298N motor driver. Motors need more voltage and current than the

NodeMCU can provide.

- The L298N regulates this 12V and also gives a 5V output.
- This 5V is used to safely power the NodeMCU via its VIN pin. So, a single 12V source can power both the motors and the controller with the help of the motor driver's voltage regulation.

7. What are BO motors and why are they suitable for this project?

BO (Battery Operated) motors are small DC gear motors used in DIY robotic projects.

They are suitable for this car because:

- They run on low voltage (3V–12V),
- Provide enough torque to move a light-weight car,
- Are compact, lightweight, and easy to mount,
- Have shafts that easily fit standard robot wheels.

Their low power needs match well with L298N motor driver and battery setup.

8. How does the car move when a control button is pressed on the web page?

When a user presses a button (e.g., “Forward”) on the web page:

1. The browser sends an HTTP GET request to the ESP8266's IP.
2. NodeMCU reads the command using its program code.
3. It sets the GPIO pins HIGH/LOW based on the command.
4. The L298N receives these signals and drives the motors accordingly.
 - For example, Forward = Motor A & B both rotate forward.
 - Left = Motor A forward, Motor B stop or reverse.

This system gives real-time control of the car wirelessly.

