

CMR INSTITUTE OF TECHNOLOGY

UGC AUTONOMOUS

Approved by AICTE, Permanent Affiliation to JNTUH & NBA Accredited

Kandlakoya (V), Medchal Dist-501 401

www.cmritonline.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

IOT WITH CLOUD COMPUTING LAB MANUAL



Prepared by

J.MANIKANDAN

Assistant Professor, Dept. of CSE(DS)

H.VENKATA SUBBAIAH

Assistant Professor, Dept. of CSE(DS)



CMR INSTITUTE OF TECHNOLOGY

Kandlakoya(V), Medchal Road, Hyderabad – 501 401

Ph. No. 08418-222042, 22106 Fax No. 08418-222106

(2023-24)

CMR INSTITUTE OF TECHNOLOGY

Vision: To create world class technocrats for societal needs.

Mission: Achieve global quality technical education by assessing learning environment through

- Innovative Research & Development
- Eco-system for better Industry institute interaction
- Capacity building among stakeholders

Quality Policy: Strive for global professional excellence in pursuit of key-stakeholders.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Vision: Develop competent software professionals, researchers and entrepreneurs to serve global society.

Mission: The department of **Computer Science and Engineering (Data Science)** is committed to

- create technocrats with proficiency in design and code for software development
- adapt contemporary technologies by lifelong learning and face challenges in IT and ITES sectors
- quench the thirst of knowledge in higher education, employment, R&D and entrepreneurship

I. Programme Educational Objectives (PEOs): Engineering Graduates will

1. Pursue successful professional career in IT and IT-enabled sectors.
2. Pursue lifelong learning skills to solve complex problems through multidisciplinary-research.
3. Exhibits professionalism, ethics and inter-personal skills to develop leadership qualities.

II. Programme Outcomes (POs): Engineering Graduates will be able to

1. Apply mathematics, science, engineering fundamentals to solve complex engineering problems.
2. Identify, formulate and analyze complex engineering problems to reach substantiated conclusions.
3. Design and develop a component/system/process to solve complex societal engineering

problems.

4. Design and conduct experiments to analyze, interpret and synthesize data for valid conclusions.
5. Create, select and apply modern tools, skills, resources to solve complex engineering problems.
6. Apply contextual engineering knowledge to solve societal issues.
7. Adapt modern engineering practices with environmental safety and sustainable development.
8. Apply professional code of ethics, responsibilities and norms in engineering practices.
9. Compete as an individual and/or as a leader in collaborative cross cultural teams.
10. Communicate effectively through technical reports, designs, documentations and presentations.
11. Endorse cognitive management skills to prepare project report using modern tools and finance.
12. Engage in independent and life-long learning in the broad context of technological changes.

III. Programme Specific Outcomes (PSOs): Engineering Graduates will be able to

1. Design and develop Computer-Based-Systems using Algorithms, Networks, Security, Gaming, Full Stack, Golang, IoT, Cloud, Data Science and AI&ML.
 2. Apply data analytics to solve real world problems.
-

CONTENTS

Sl. No.	Particulars	Page No.
1	Syllabus	6
2	Student Entry Behavior or Pre-requisites	7
3	Course Outcomes	8
4	Mapping of Course with PEOs-PSOs-POs	9-10
5	Mapping Of Course Outcomes with PEOs	11
6	Mapping Of Course Outcomes with PSOs	12
7	Direct Course Assessment	13
8	Indirect Course Assessment	14
9	Programs	17-87

IOT with Cloud Computing Lab

Course	B.Tech. VI-Sem	L	T	P	C
Subject Code	20-CS-PC-324	-	-	3	1.5

Course Outcomes(COs) & CO - PO Mapping(3-Strong; 2-Medium; 1-Weak Correlation)

COs	Upon completion of course the students will be able to	PO4	PO5	PO14
CO1	Identify various IOT Devices	3	3	3
CO2	Use IOT devices in various applications	3	3	3
CO3	Develop automation work-flow in IOT enabled cloud environment	3	3	3
CO4	Take part in practicing and monitoring remotely	3	3	3
CO5	Make use of various IOT protocols in cloud	3	3	3

IOT with Cloud Computing Lab

III-B.Tech.-II-Sem.**Subject Code: 20-CS-PC-324**

L	T	P	C
0	0	3	1.5

Course Outcomes: Upon completion of the course, the students will be able to

1. Identify various IoT devices.
2. Use IoT Devices in various applications
3. Develop automation work-flow in IoT enabled cloud environment
4. Take part in practicing and monitoring remotely
5. Make use of various IoT Protocols in Cloud

List of Experiments (Minimum **10** experiments to be conducted)

1. Install necessary software for Arduino and Raspberry Pi.
2. Familiarization with Arduino and Raspberry Pi boards.
3. Write a program to transfer sensor data to smart phone using Bluetooth on Arduino.
4. Write a program to implement RFID using Arduino.
5. Write a program to monitor temperature and humidity using Arduino and Raspberry Pi.
6. Write a program to interface IR Sensor with Arduino using IoT cloud application.
7. Write a program to upload temperature and humidity data to the cloud using Arduino or Raspberry Pi.
8. Write a program to retrieve temperature and humidity data to the cloud using Arduino or Raspberry Pi.
9. Write a program to create TCP Server on cloud using Arduino and Respond with humidity data to TCP Client when requested.
10. Write a program to create UDP Server on cloud using Arduino and Respond with humidity data to UDP Client when requested.

Reference:

1. Internet of Things Lab Manual, Department of CSE, CMRIT, Hyd.

Micro-Projects: Student must submit a report on one of the following Micro–Projects before commencement of second internal examination.

1. Air Pollution Meter
2. Smart Garbage Collector
3. IoT Based Smart street light
4. Humidity & Temperature Monitoring
5. Baggage Tracker
6. IoT based Gas leakage Monitoring
7. Circuit Breakage Detection
8. IoT based Smart irrigation system
9. Anti-Theft Flooring System
10. IoT based water level Monitoring system

2. STUDENT ENTRY BEHAVIOR OR PRE-REQUISITES

- IOT Devices
- Arduino and Raspberry pi

3. COURSE OUTCOMES

Course Outcome	Course Outcome Statements
CO - 1	Identify various IOT Devices
CO – 2	Use IOT devices in various applications
CO – 3	Develop automation work flow in IOT enabled cloud Environment
CO – 4	Take part in practicing and monitoring remotely
CO – 5	Make use of various IOT protocols in Cloud.

Co-Po Mapping

POs	PO4	PO5	P014
CO1	3	3	3
CO2	3	3	3
CO3	3	3	3
CO4	3	3	3
CO5	3	3	3

4. MAPPING OF COURSE WITH PEOS-PSOS-POS

Program Educational Objectives (PEOs)

Sl. No.	PEOs Name	Program Education Objective Statements
1	PEO – 1	Impart profound knowledge in humanities and basic sciences along with core engineering concepts for practical understanding & project development.[PO's: 1,2,3,4,5,7,8,9,10,11 and 12] [PSO's: 1 and 2]
2	PEO – 2	Enrich analytical skills and Industry-based modern technical skills in core and interdisciplinary areas for accomplishing research, higher education, entrepreneurship and to succeed in various engineering positions globally. [PO's: 1,2,3,4,5,6,7,8,9,10 and 12] [PSO's: 1, 2 and 3]
3	PEO – 3	Infuse life-long learning, professional ethics, responsibilities and adaptation to innovation along with effective communication skills with a sense of social awareness. [PO's: 1,2,3,4,5,6,7,8,9,10,11 and 12] [PSO's: 2 and 3]

Program Specific Objectives (PSOs)

Sl. No.	PSOs Name	Program Specific Objective Statements
1	PSO – 1	Use mathematical abstractions and Algorithmic design along with open source programming tools to solve complexities involved in efficient programming.[PO:1,2,3,4 and 5] & [PEO:1 and 2]
2	PSO – 2	Ensure programming & documentation skills for each individual student in relevant subjects i.e., C, C++, Java, DBMS, Web Technologies (Development), Linux, Data Warehousing & Data Mining and on Testing Tools.[PO:1,2,3,4,5,10 and 11] & [PEO:1,2 and 3]
3	PSO – 3	Ensure employability and career development skills through Industry oriented mini & major projects, internship, industry visits, seminars and workshops. [PO:6,7,8,9,10,11 and 12] & [PEO:1,2 and 3]

Program Outcomes (POs)

PO Name	Graduate Attributes	PO Statements
PO1	Engineering knowledge	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 2	Problem analysis	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 3	Design/development of solutions	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 4	Conduct investigations of complex problems	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 5	Modern tool usage	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 6	The engineer and society	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. [PEO's: 2 and 3]
PO 7	Environment and sustainability	Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. [PEO's: 1,2 and 3]
PO 8	Ethics	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. [PEO's: 1,2 and 3] [PSO's: 2 and 3]
PO 9	Individual and team work	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. [PEO's: 1,2 and 3] [PSO's: 3]
PO 10	Communication	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. [PEO's: 1,2 and 3] [PSO's: 2 and 3]
PO 11	Project management and finance	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. [PEO's: 1 and 3] [PSO's: 2 and 3]
PO 12	Life-long learning	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]

5. Mapping Of Course Outcomes with PEOs**(CORRELATION - TABLES)**

No	Course Outcomes	PEO1	PEO2	PEO3
1	CO - 1	3	3	3
2	CO – 2	3	3	3
3	CO – 3	3	3	3
4	CO – 4	3	3	3
5	CO – 5	3	3	3

6. DIRECT COURSE ASSESSMENT

(As mentioned in following table of 10 parameters, of which consider only the parameters required for this courses)

No	Description	Targeted Performance	Actual Performance	Remarks	Course Attainment
1	Internal Marks(25)	90% of Students(153 Students) should Secure 60% of Internal Marks i.e., 15 Marks			
2	External Marks(50)	80% of Students(136 Students) should Secure 70% of External Marks i.e., 35 Marks			
3	Clearing of Subject	A minimum of 95% of Students(161 Students) should clear this course in first attempt			
4	Getting First Class	90% of Students(153 Students) should Secure I Class Marks i.e., 45 Marks in my course			
5	Distinction	80% of Students (136 Students) should secure First Class With Distinction i.e., 53 Marks in my course			
6	Outstanding Performance	50% of Students (85 Students) should secure 80% and above Marks. i.e., 60 Marks in my course			

7. INDIRECT COURSE ASSESSMENT

(As mentioned-strong (3), moderate (2), weak (1) & no comment (0))

Mission Statement of CSE

- Impart fundamentals through state of art technologies for research and career in Computer Science & Engineering.
- Create value-based, socially committed professionals for anticipating and satisfying fast changing societal requirements.
- Foster continuous self learning abilities through regular interaction with various stake holders for holistic development.

Correlation of Mission Elements with Mission Statement of CSE Department related to the Course (only Ticking given by faculty)

No	Mission Elements	Strong	Moderate	Weak	No Comment
M-1	Impart Fundamentals	√			
M-2	State Of Art Technologies	√			
M-3	Research & Career Development	√			
M-4	Value based Socially Committed Professional	√			
M-5	Anticipating & Satisfying Industry Trends		√		
M-6	Changing Societal Requirements		√		
M-7	Foster Continuous Learning	√			
M-8	Self Learning Abilities	√			
M-9	Interaction with stakeholders	√			
M-10	Holistic Development	√			

Indirect Course Assessment through Student Satisfaction Survey

(Note for *: Parameters used for course teaching like

a: Classroom teaching b: Simulations c:labs d: Mini_Projects
 e: Major Projects f: Conferences g: professional activities
 h: Technical Clubs i: Guest Lectures j: Workshops k: Technical Fests l:Tutorials
 m:NPTLs n:Digital Library o: Industrial Visits
 p: software Tools q: Internship/training r:Technical Seminars
 s: NSS t: NSS u: sports etc.

Further assume other parameters if any)

No	Question Based on PEO/PO/PSO/CO	Parameters (a /b /c.../)*	Strong (3)	Moderate (2)	Weak (1)	No comment (0)
1	Did the course impart fundamentals through interactive learning and contribute to core competence?	a,b,c,d,e,h,i,l,m,p,q				
2	Did the course provide the required knowledge to foster continuous learning?	a,b,c,d,e,h,i,l,m,p,q				
3	Whether the syllabus content anticipates & satisfies the industry and societal needs?	a,b,c,d,e,h,i,l,m,p,q				
4	Whether the course focuses on value based education to be a socially committed professional?	a,b,c,d,e,h,i,l,m,p,q				
5	Rate the role of the facilitator in mentoring and promoting the self learning abilities to excel academically and professionally?	a,b,c,d,e,h,i,l,m,p,q				
6	Rate the methodology adopted and techniques used in teaching learning processes?	a,b,c,d,e,h,i,l,m,p,q				
7	Rate the course in applying sciences & engineering fundamentals in providing research based conclusions with the help of modern tools?	a,b,c,d,e,h,i,l,m,p,q				
8	Did the course have any scope to design, develop and test a system or component?	a,b,c,d,e,h,i,l,m,p,q				
9	Rate the scope of this course in addressing cultural, legal, health, environment and safety issues?	a,b,c,d,e,h,i,l,m,p,q				
10	Scope of applying management fundamentals to demonstrate effective technical project presentations & report writing?	a,b,c,d,e,h,i,l,m,p,q				

Average		
----------------	--	--

8. OVERALL COURSE ASSESSMENT

(80% Direct + 20% Indirect, if any)

No	Assessment Type	Weightage	Attainment Level
1	Direct-Assignment, Quiz, Subjective, University Exams, Results, Bench Marks	0.80	
2	Indirect-Surveys-Questionnaire	0.20	
	Overall	1.00	

Course Attainment level:

WEEK-1**Arduino Installation**

1. **Aim:** Install necessary software for Arduino and Raspberry Pi.

Arduino:

- Arduino is a platform that makes it easy for you to build projects using electronics.
- IoT is a way of using electronics - to make electronic modules talk to each other remotely and wirelessly (often using a Cloud) to solve problems.
- Now, Arduino can also help you easily build IoT projects in two ways: Using traditional Arduino boards and attaching communication breakout modules (like nRF Bluetooth, WiFi, LoRA, GSM, etc) to them.
- Arduino is a micro controller that can be connected to one or more sensors and help you capture the data or information and then pass it on to processor. If you know the full stack of IoT then you should also look at Raspberry.
- RaspPi is a microprocessor so the basic difference between Arduino and RasPi is that RaspPi is controller plus processor and Arduino is just a micro controller.
- They suit the need for different use cases. You can easily read online about this both.

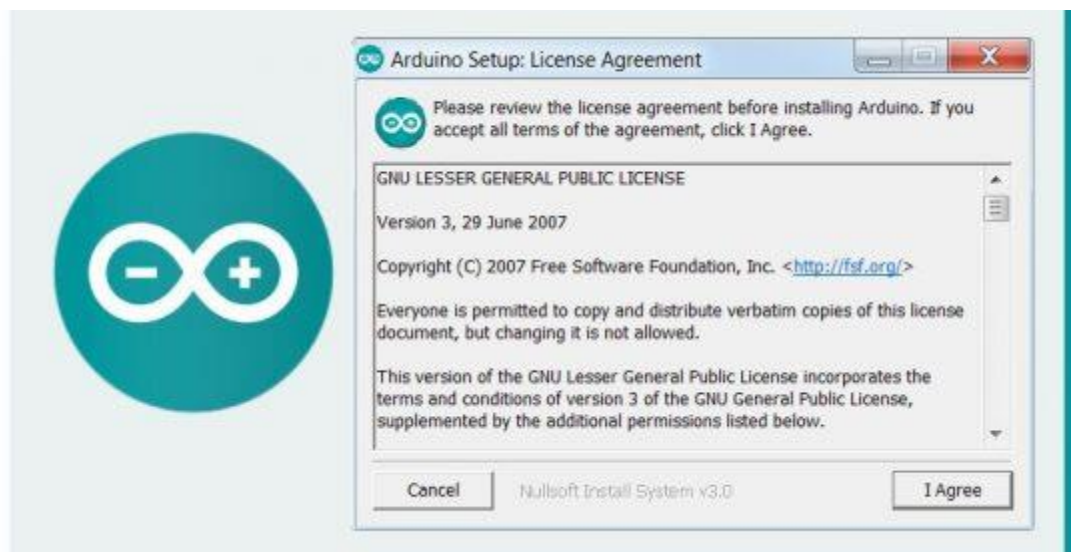
Download and install the Arduino software (Arduino IDE 1.8.5)

- Go to the [Arduino website](#) and click the download link to go to the download page.
- After downloading, locate the downloaded file on the computer and extract the folder from the downloaded zipped file. Copy the folder to a suitable place such as your desktop.

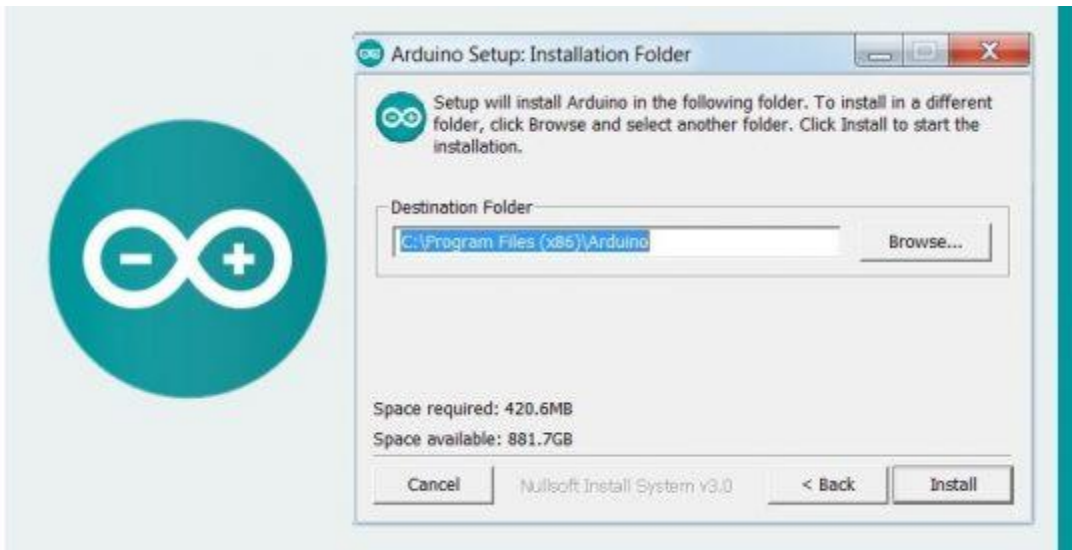
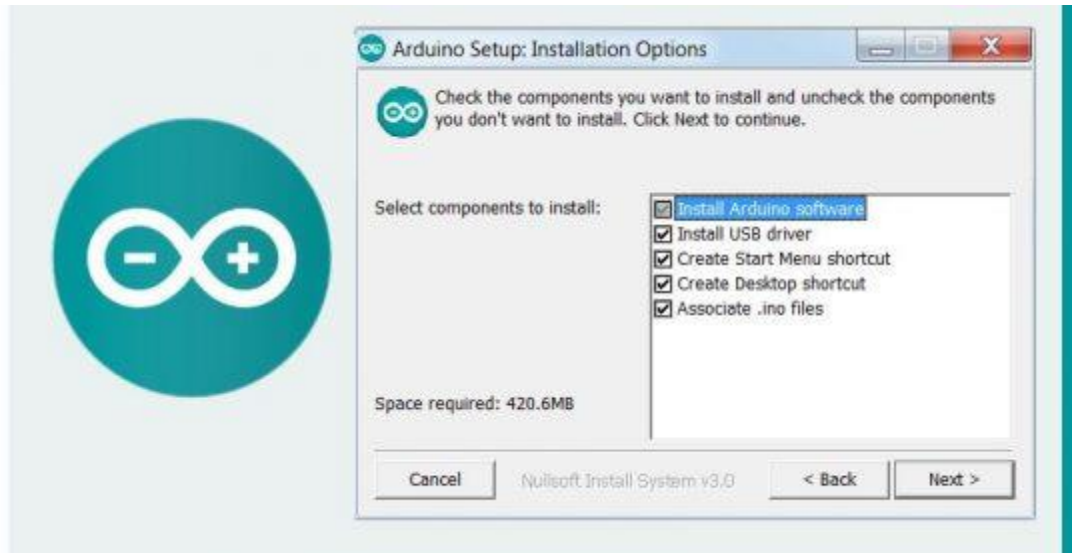
Download the Arduino IDE



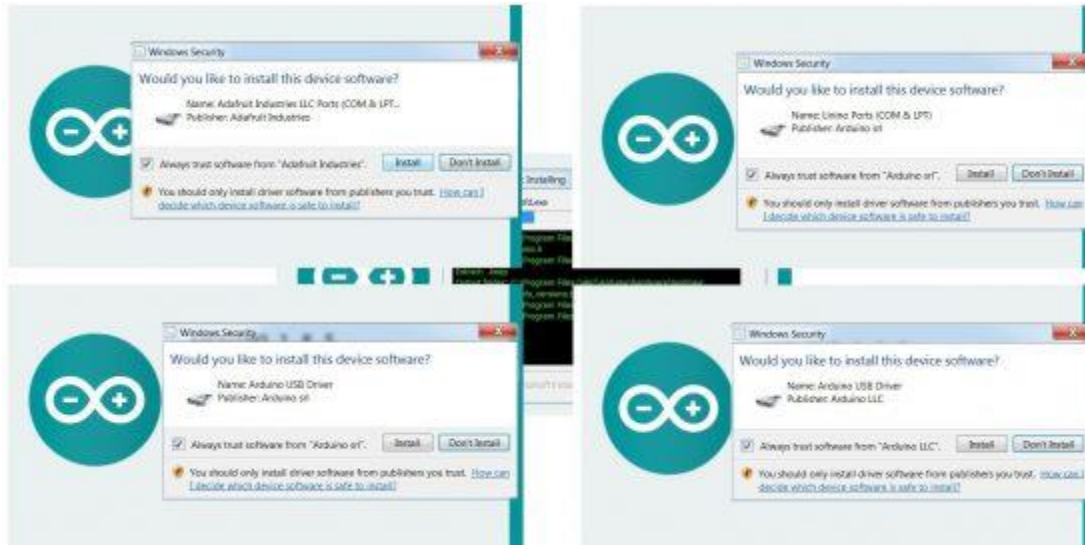
The image shows the Arduino 1.8.5 download page. On the left is the Arduino logo. To its right, the text reads: **ARDUINO 1.8.5**. Below this, it says: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment 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." On the right side of the page, there are links for different operating systems: "Windows Installer, for Windows XP and up", "Windows ZIP file for non-admin install", "Windows app Requires Win 8.1 or 10" (with a "Get" button), "Mac OS X 10.7 Lion or newer", "Linux 32 bits", "Linux 64 bits", and "Linux ARM". At the bottom right, there are links for "Release Notes", "Source Code", and "Checksums (sha512)".



Read the Arduino License agreement and click the “I Agree” button.



The Arduino software will start to install.



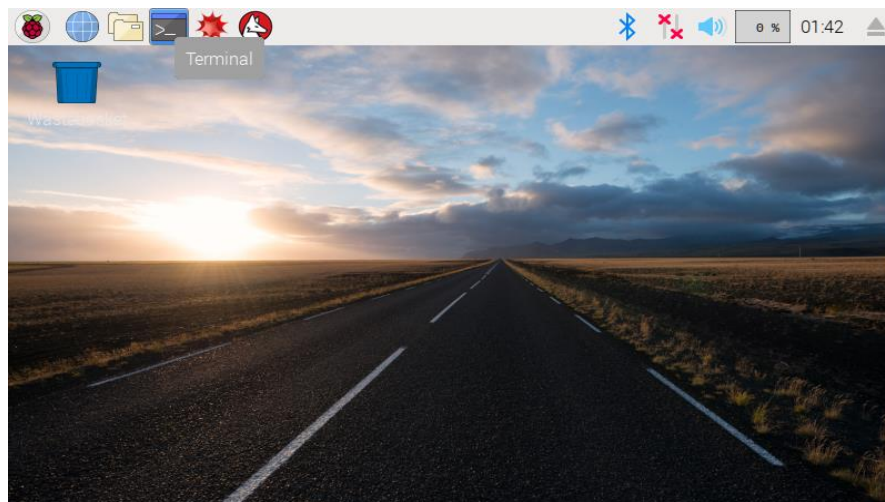
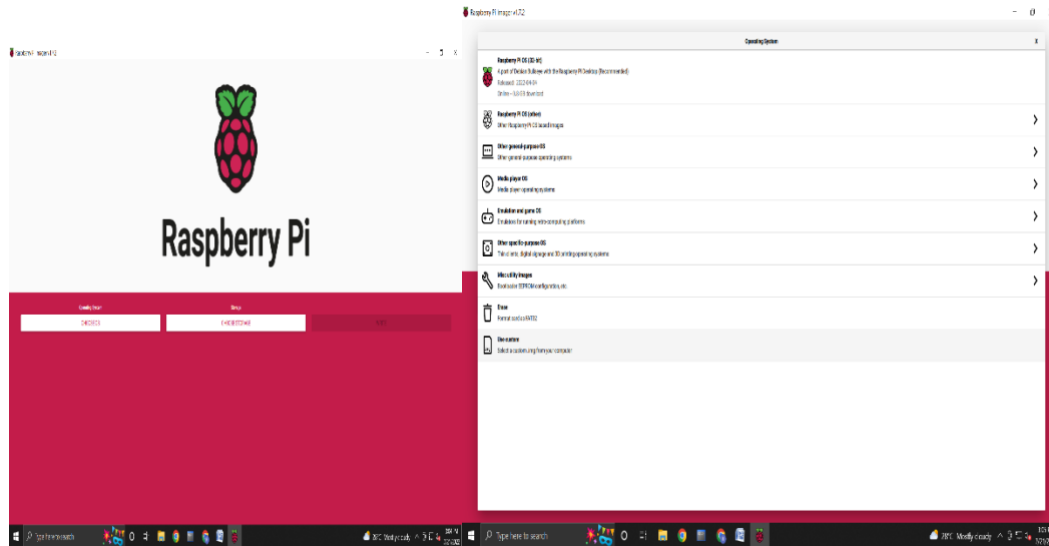
Running the Arduino IDE Software



Raspberry Pi

The screenshot displays the Raspberry Pi OS website in a web browser. The page title is "Install Raspberry Pi OS using Raspberry Pi Imager". The main content describes the Raspberry Pi Imager as a quick and easy way to install the OS and other operating systems to a microSD card. It includes a link to a 45-second video for installation instructions. Below the text, there are download links for Windows, macOS, and Ubuntu for x86. A terminal window snippet shows the command to install the imager: `sudo apt install rpi-imager`.

Below the website screenshot, a terminal window is shown with the Raspberry Pi Imager application running. The terminal displays the "Welcome to Raspberry Pi Imager Setup" dialog box, which prompts the user to select the operating system and storage. The user has selected "Raspberry Pi OS" and "SD Card" as the storage device. The terminal also shows the "Completing Raspberry Pi Imager Setup" dialog box, indicating that the setup is complete.

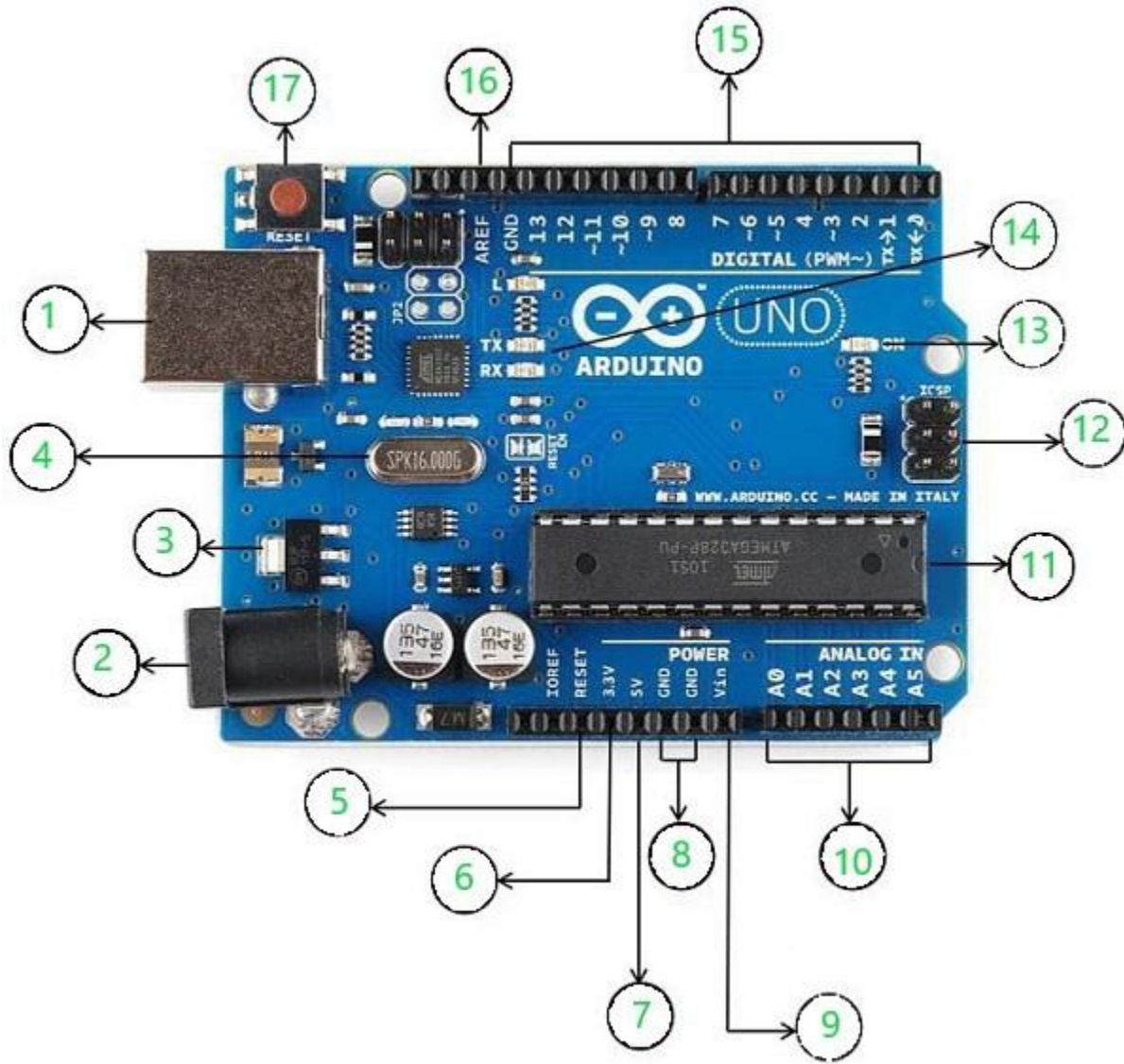


WEEK-2

AIM: Familiarization with Arduino and Raspberry Pi boards.

Arduino is an open source hardware and software company. Project, and user community that designs and manufactures single board micro controllers and micro controller kits for building digital devices.

. Arduino board designs a variety of micro-processors and controllers .The board are equipped with set of digital and analog input/output pins that may be interfaced to various expansion boards .



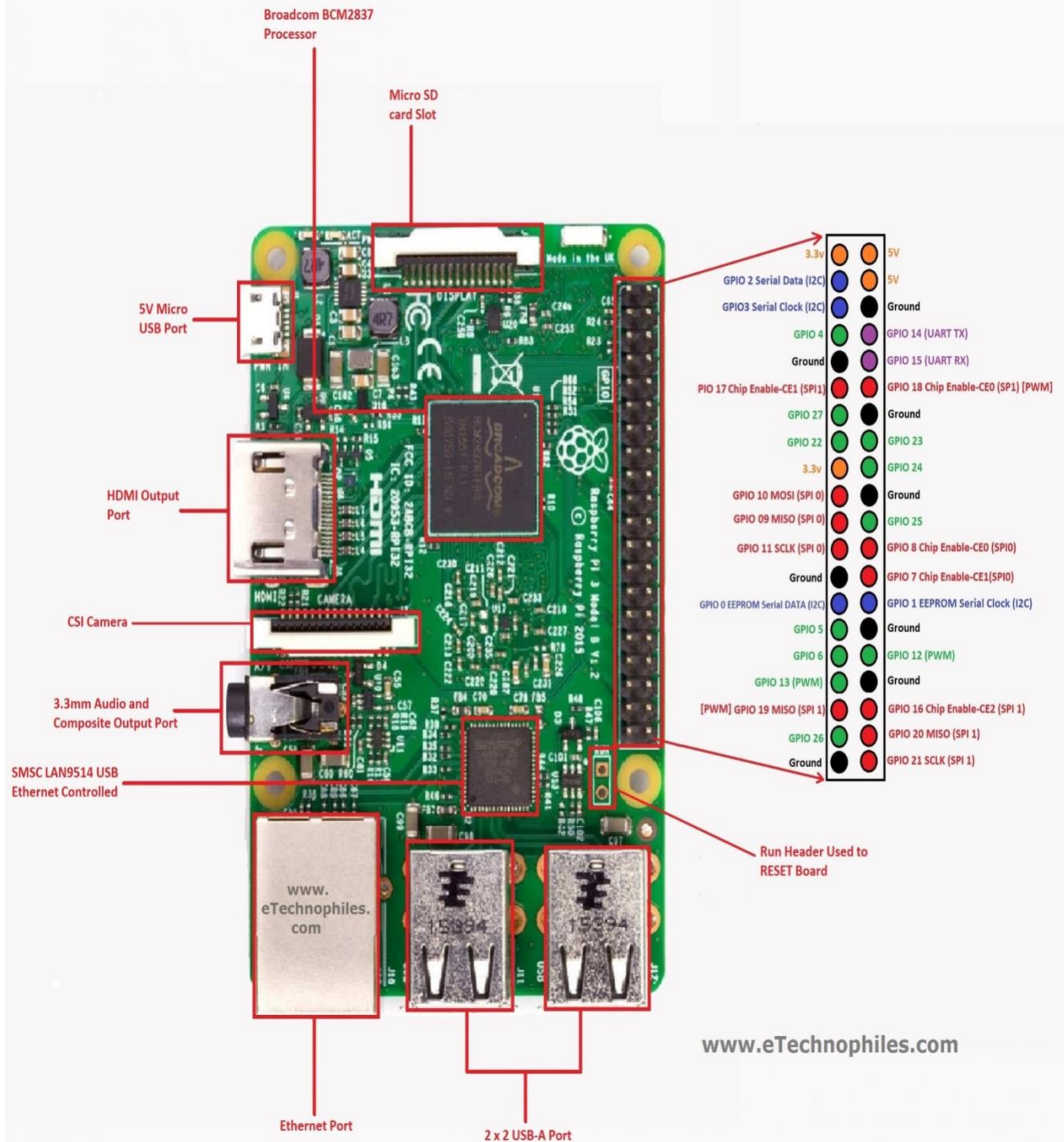
1.USB: can be used for both power and communication with the IDE

2.Barrel Jack: used for power supply

3.Voltage Regulator: regulates and stabilizes the input and output voltages

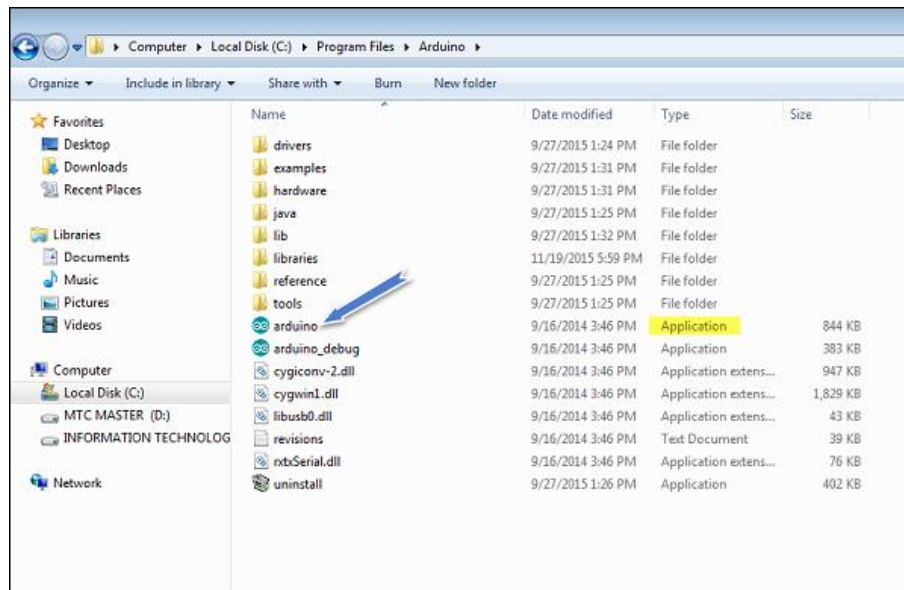
4.Crystal Oscillator: keeps track of time and regulates processor frequency

- 5.Reset Pin:** can be used to reset the Arduino Uno
- 6. 3.3V pin:** can be used as a 3.3V output
- 7.5V pin:** can be used as a 5V output
- 8. GND pin:** can be used to ground the circuit.
- 9. Vin pin:** can be used to supply power to the board
- 10. Analog pins(A0-A5):** can be used to read analog signals to the board
- 11. Microcontroller(ATMega328):** the processing and logical unit of the board
- 12. ICSP pin:** a programming header on the board also called SPI
- 13. Power indicator LED:** indicates the power status of the board
- 14. RX and TX LEDs:** receive(RX) and transmit(TX) LEDs, blink when sending or receiving serial data respectively
- 15. Digital I/O pins:** 14 pins capable of reading and outputting digital signals 6 of these pins are also capable of PWM
- 16. AREF pins:** can be used to set an external reference voltage as the upper limit for the analog pins
- 17. Reset button:** can be used to reset the board



Launch Arduino IDE.

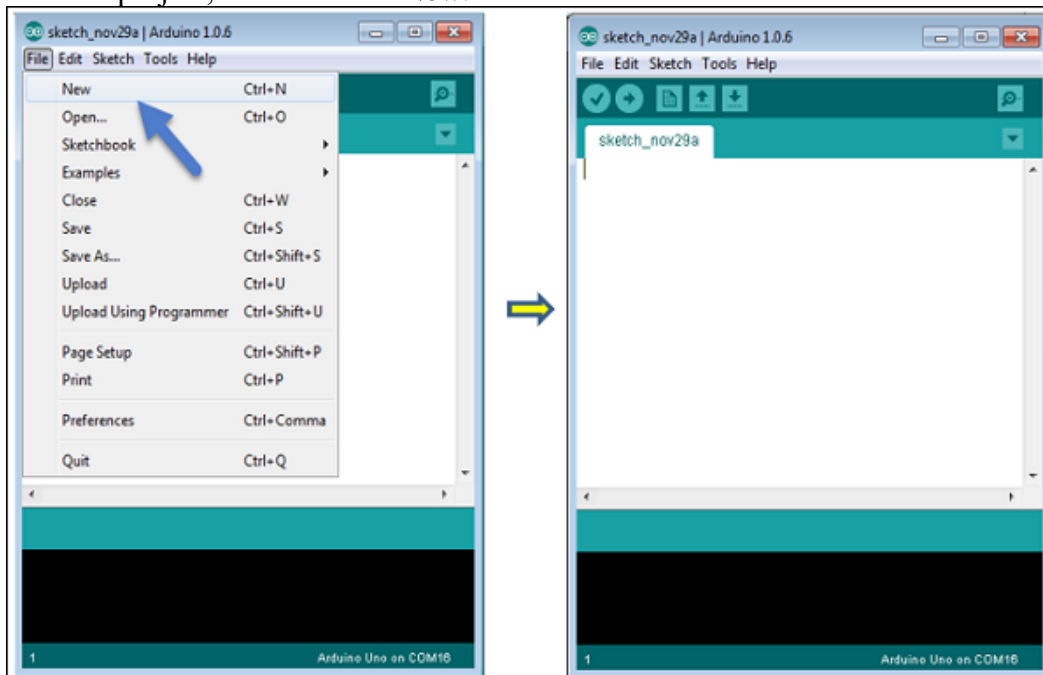
After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

**Step 5 – Open your first project.**

Once the software starts, you have two options –
Create a new project.

Open an existing project example.

To create a new project, select File → New.



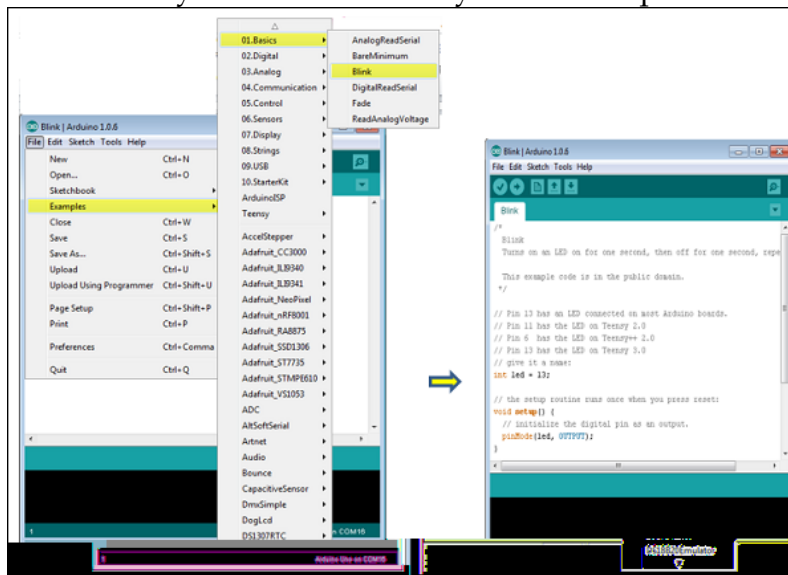
To open an existing project example,
select File

→ Example

→ Basics

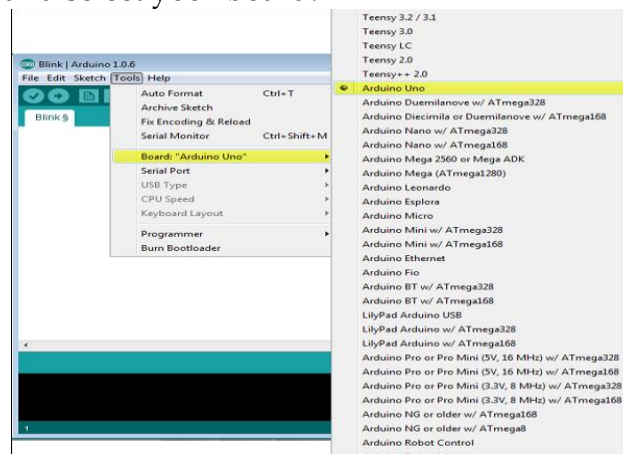
→ Blink.

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.



Step 6 – Select your Arduino board.

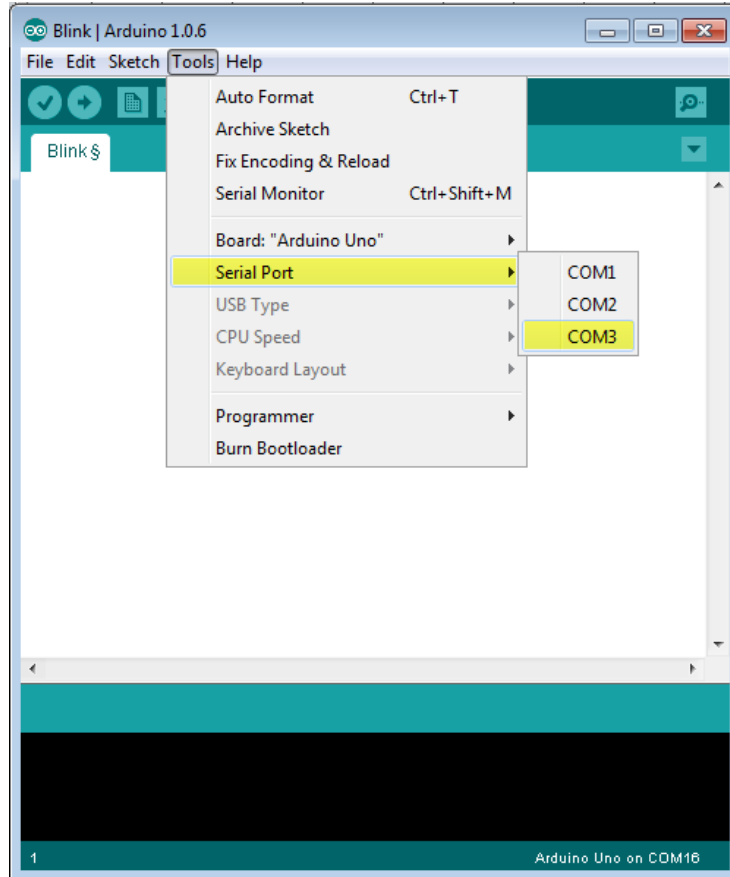
To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer. Go to Tools → Board and select your board.



Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

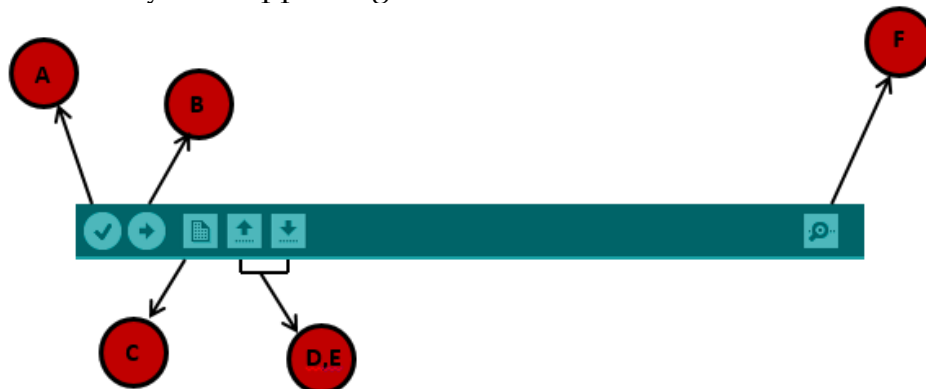
Step 7 – Select your serial port.

Select the serial device of the Arduino board. Go to **Tools** → **Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 – Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



- A – Used to check if there is any compilation error.
- B – Used to upload a program to the Arduino board.

C – Shortcut used to create a new sketch.

D – Used to directly open one of the example sketch.

E – Used to save your sketch.

F – Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Note – If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

Arduino - Program Structure

In this chapter, we will study in depth, the Arduino program structure and we will learn more new terminologies used in the Arduino world. The Arduino software is open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.

Sketch – The first new terminology is the Arduino program called “**sketch**”.

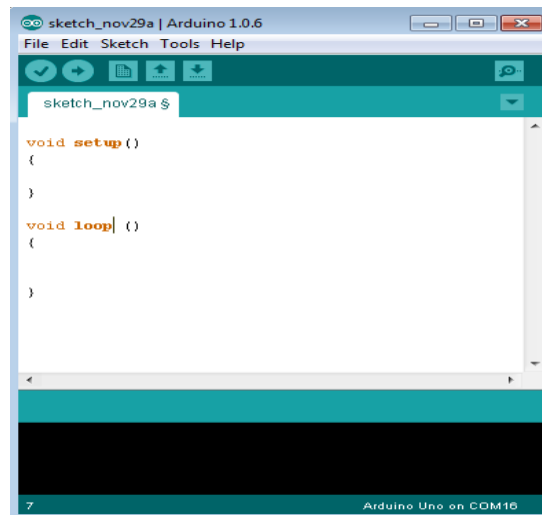
Structure

Arduino programs can be divided in three main parts: **Structure**, **Values** (variables and constants), and **Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the **Structure**. Software structure consist of two main functions –

Setup() function

Loop() function



Void setup () {

}

PURPOSE – The **setup()** function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

INPUT – -

OUTPUT – -

RETURN – -

```
Void Loop ( ) {  
}
```

PURPOSE – After creating a **setup()** function, which initializes and sets the initial values, the **loop()** function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

INPUT – -

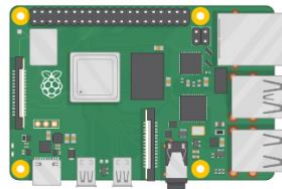
OUTPUT – -

RETURN – -

RASPBERRY PI

There are several models of Raspberry pi and for most people Raspberry Pi 3 Model B is the one to choose. Raspberry Pi 3 Model B is the newest, fastest, and easiest to use.

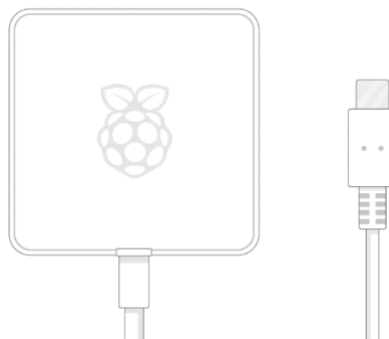
Raspberry Pi 3 comes with either 2GB or 4GB of RAM. For most educational purposes and hobbyist projects, and for use as a desktop computer, 2GB is enough.



Raspberry Pi Zero and Zero W are smaller and require less power, so they're useful for portable projects such as robots. It's generally easier to start a project with Raspberry Pi 3, and to move to Pi Zero when you have a working prototype that a smaller Pi would be useful for.

A power supply

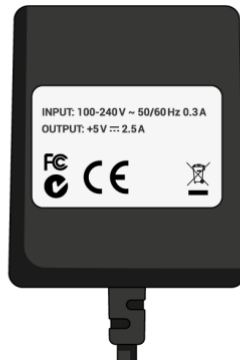
To connect to a power socket, all Raspberry Pi models have a USB port (the same found on many mobile phones): either USB-C for Raspberry Pi 3, or micro USB for Raspberry Pi 3, 2 and 1.



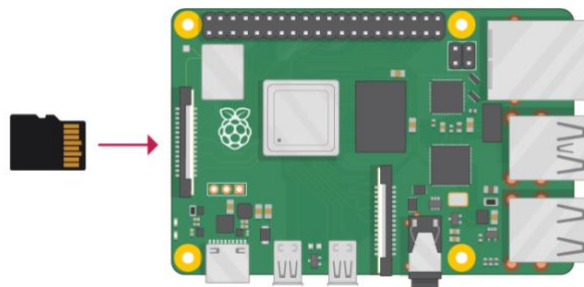
You need a power supply that provides:
At least 3.0 amps for Raspberry Pi 3



At least 2.5 amps for Raspberry Pi 3



- We recommend using [our official Universal Power Supply](#).
- A microSD card
- Your Raspberry Pi needs an SD card to store all its files and the Raspbian operating system.



- You need a microSD card with a capacity of at least 8 GB.
- Many sellers supply SD cards for Raspberry Pi that are already set up with Raspbian and ready to go.

A keyboard and a mouse

To start using your Raspberry Pi, you need a USB keyboard and a USB mouse. Once you've set your Pi up, you can use a Bluetooth keyboard and mouse, but you'll need a USB keyboard and mouse for the first setup.

An Ethernet cable

The large Raspberry Pi models (but not Pi Zero/Zero W) have a standard Ethernet port to connect them to the internet; to connect Pi Zero to the internet, you need a USB-to-Ethernet adaptor.

Raspberry Pi 3, Pi 3B+, and Pi Zero W can also be wirelessly connected to the internet.

Set up your SD card

If you have an SD card that doesn't have the Raspbian operating system on it yet, or if you want to reset your Raspberry Pi, you can easily install Raspbian yourself. To do so, you need a computer that has an SD card port — most laptop and desktop computers have one.

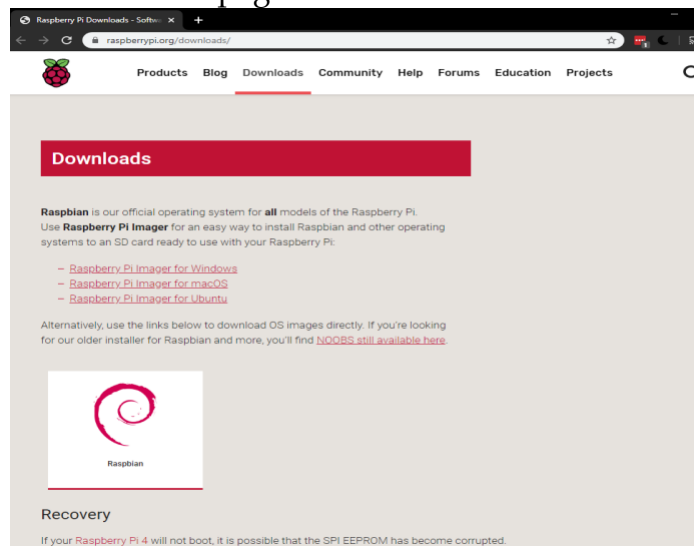
The Raspbian operating system via the Raspberry Pi Imager

Using the Raspberry Pi Imager is the easiest way to install Raspbian on your SD card.

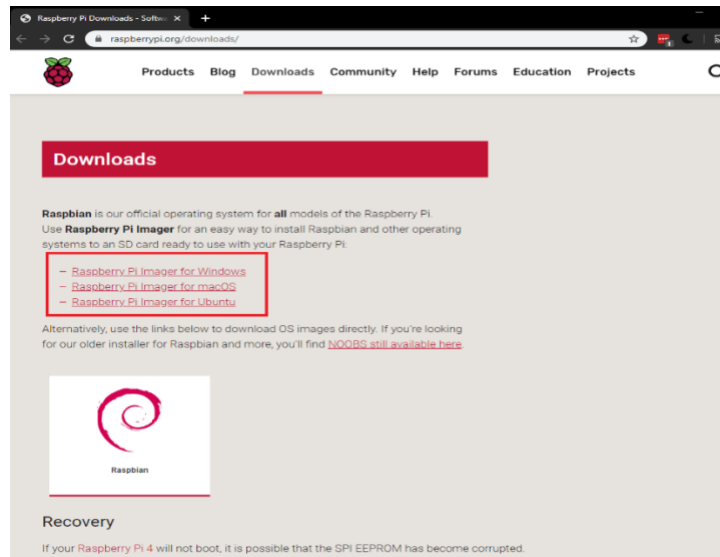
Note: More advanced users looking to install a particular operating system should use this guide to installing operating system images.

Download and launch the Raspberry Pi Imager

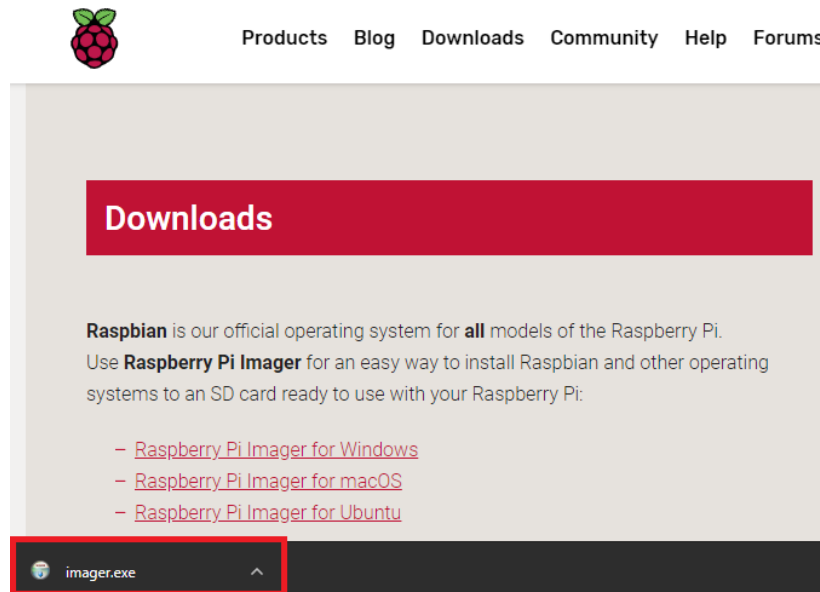
Visit the Raspberry Pi downloads page



Click on the link for the Raspberry Pi Imager that matches your operating system



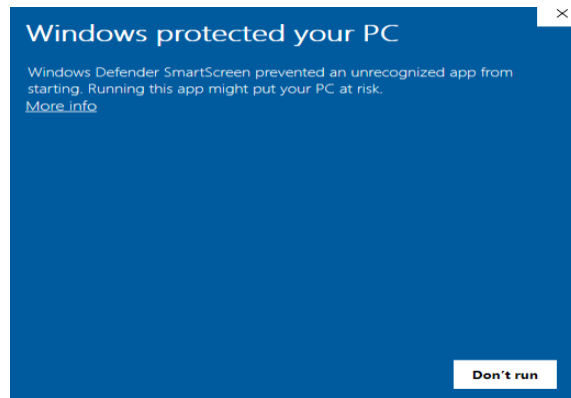
When the download finishes, click it to launch the installer



Using the Raspberry Pi Imager

Anything that's stored on the SD card will be overwritten during formatting. If your SD card currently has any files on it, e.g. from an older version of Raspbian, you may wish to back up these files first to prevent you from permanently losing them.

When you launch the installer, your operating system may try to block you from running it. For example, on Windows I receive the following message:

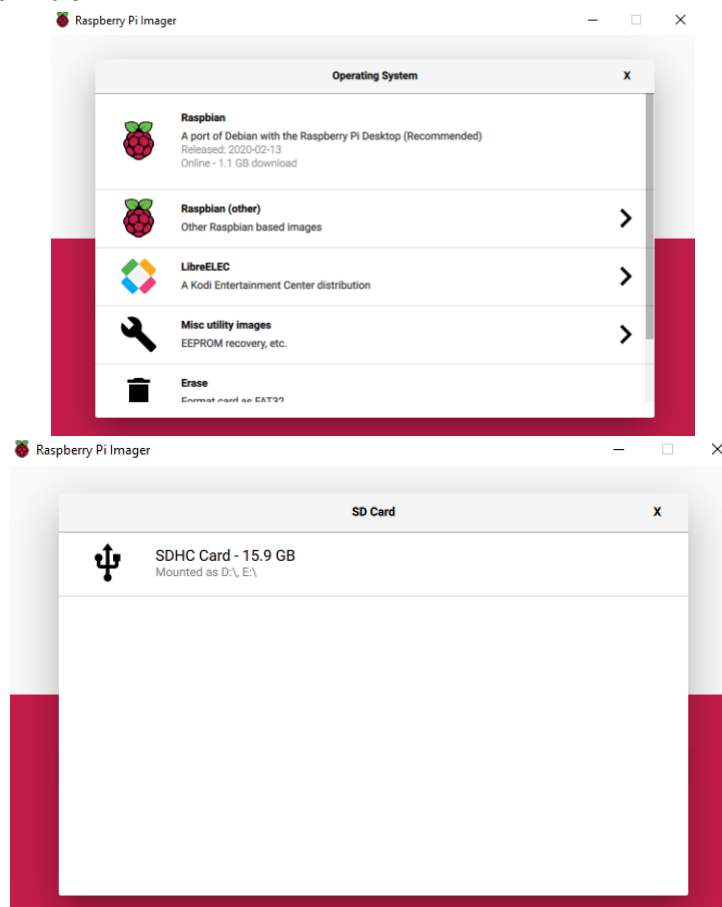


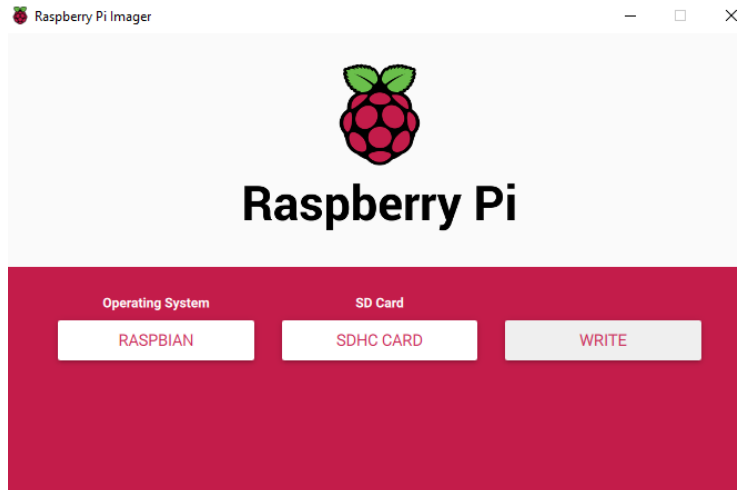
If this pops up, click on **More info** and then **Run anyway**

Follow the instructions to install and run the Raspberry Pi Imager

Insert your SD card into the computer or laptop SD card slot

In the Raspberry Pi Imager, select the OS that you want to install and the SD card you would like to install it on

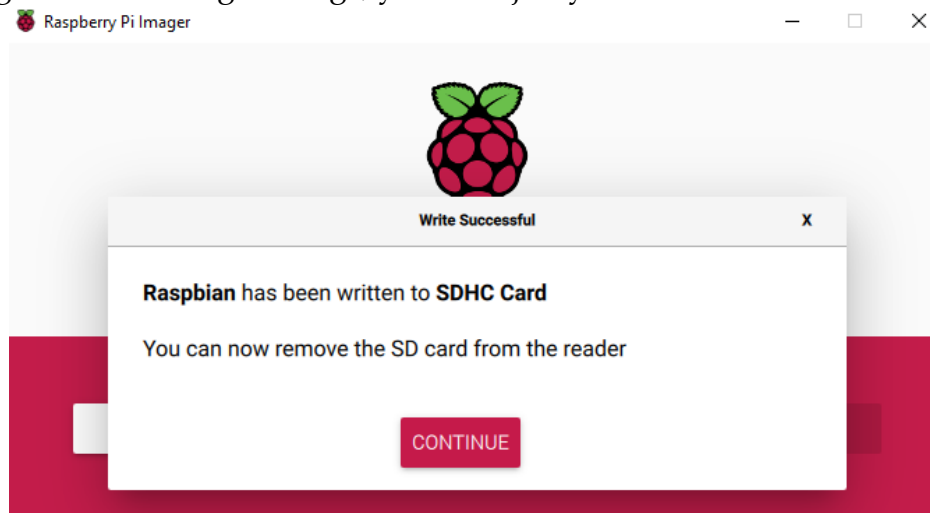




Then simply click the **WRITE** button

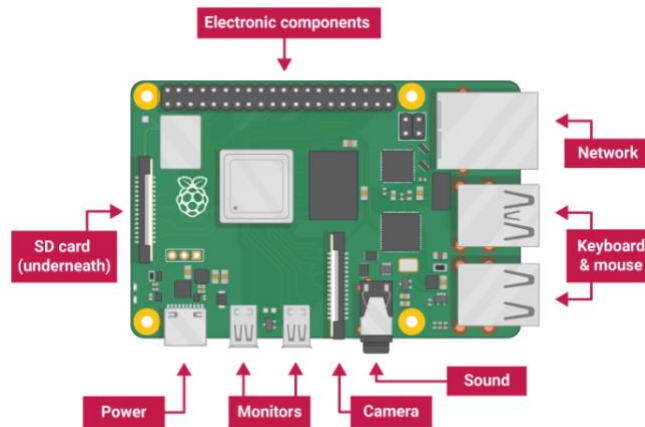
Wait for the Raspberry Pi Imager to finish writing

Once you get the following message, you can eject your SD card

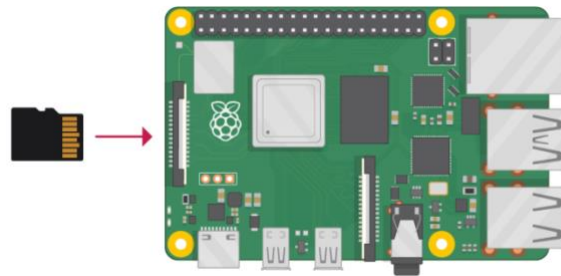


Connect your Raspberry Pi

Now get everything connected to your Raspberry Pi. It's important to do this in the right order, so that all your components are safe.



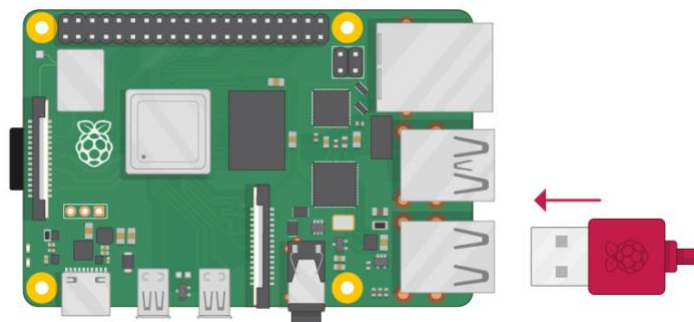
Insert the SD card you've set up with Raspbian (via NOOBS) into the microSD card slot on the underside of your Raspberry Pi.



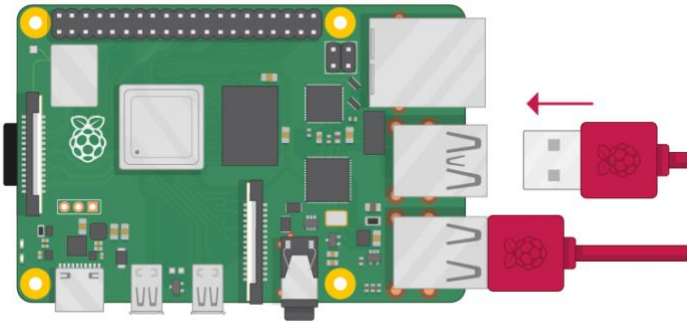
Note: Many microSD cards come inside a larger adapter — you can slide the smaller card out using the lip at the bottom.



Find the USB connector end of your mouse's cable, and connect the mouse to a USB port on Raspberry Pi (it doesn't matter which port you use).



Connect the keyboard in the same way.

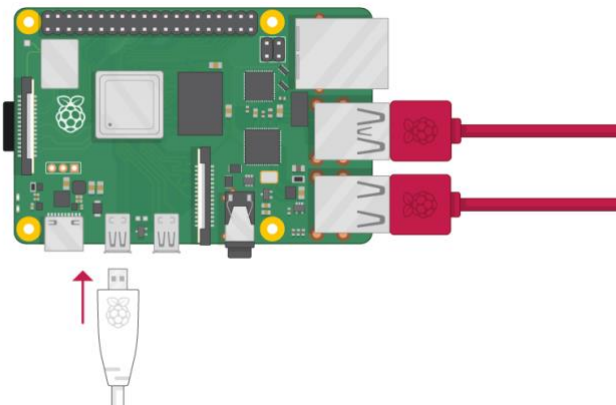


Make sure your screen is plugged into a wall socket and switched on.

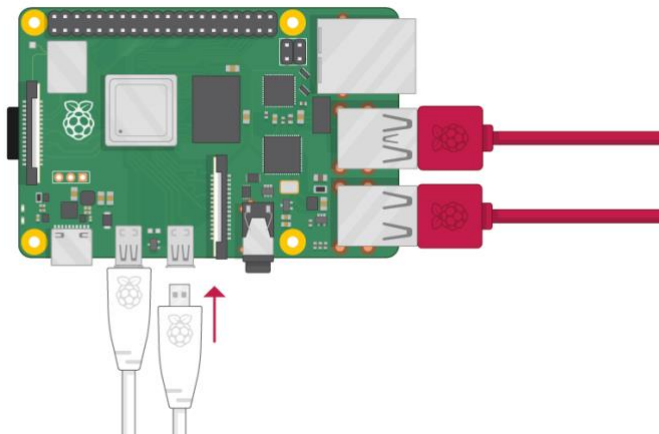
Look at the HDMI port(s) on the Raspberry Pi – notice that they have a flat side on top. Use a cable to connect the screen to Raspberry Pi's HDMI port – use an adapter if necessary.

Raspberry Pi 3

Connect your screen to the first of Raspberry Pi 3's HDMI ports, labelled HDMI0.

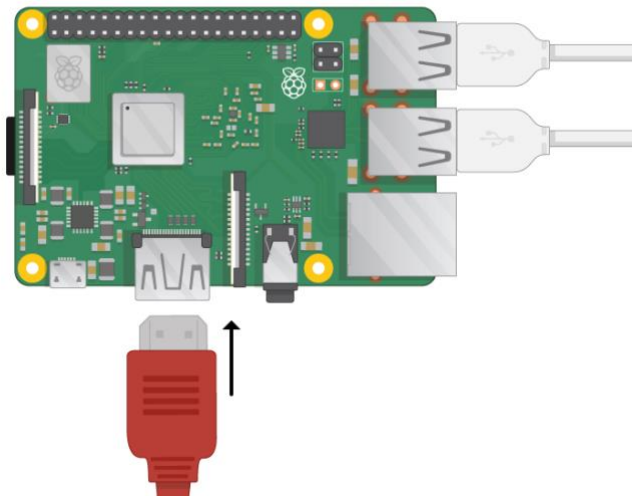


You can connect an optional second screen in the same way.

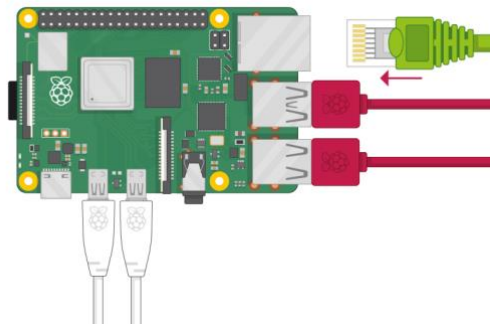


Raspberry Pi 1, 2, 3

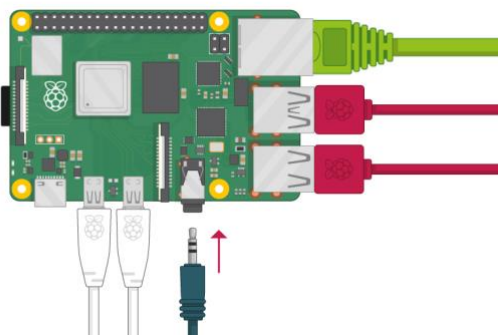
Connect your screen to the single HDMI port.



Note: nothing will display on the screen, because your Raspberry Pi is not running yet. If you want to connect your Raspberry Pi to the internet via Ethernet, use an Ethernet cable to connect the Ethernet port on Raspberry Pi to an Ethernet socket on the wall or on your internet router. You don't need to do this if you want to use wireless connectivity, or if you don't want to connect to the internet.



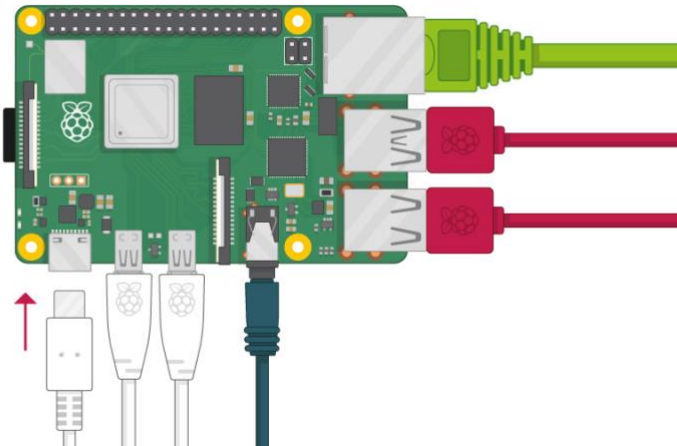
If the screen you are using has speakers, sound will play through those. Alternatively, connect headphones or speakers to the audio port if you prefer.



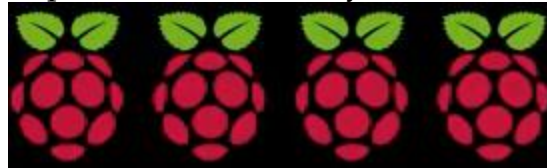
Start up your Raspberry Pi

Your Raspberry Pi doesn't have a power switch: as soon as you connect it to a power outlet, it will turn on.

Plug the USB power supply into a socket and connect it to your Raspberry Pi's power port.



You should see a red LED light up on the Raspberry Pi, which indicates that Raspberry Pi is connected to power. As it starts up (this is also called booting), you will see raspberries appear in the top left-hand corner of your screen.



First-time startup with NOOBS

After a few seconds the Raspbian Desktop will appear.



Finish the setup

When you start your Raspberry Pi for the first time, the Welcome to Raspberry Pi application will pop up and guide you through the initial setup.

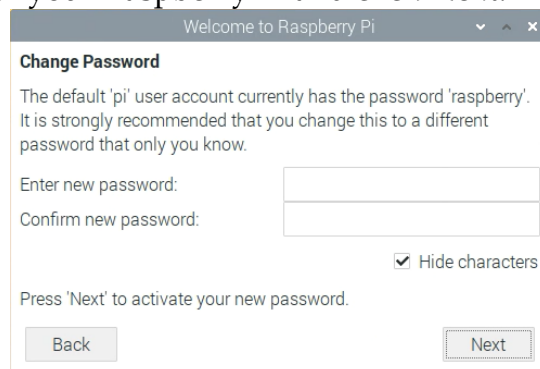


Click Next to start the setup.

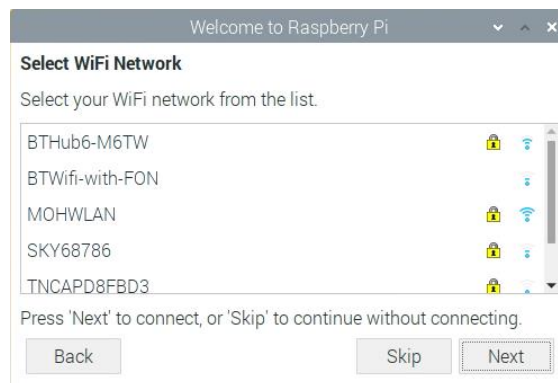
Set your Country, Language, and Timezone, then click Next again.



Enter a new password for your Raspberry Pi and click Next.

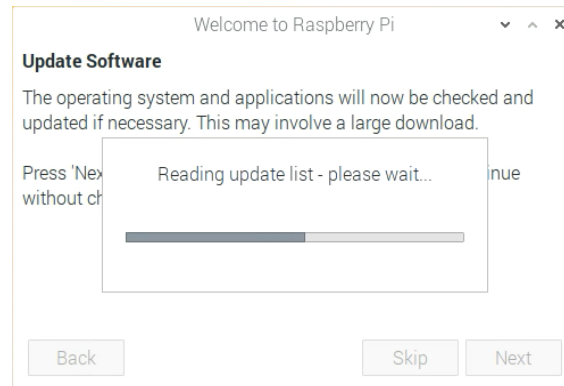


Connect to your WiFi network by selecting its name, entering the password, and clicking Next.



Note: if your Raspberry Pi model doesn't have wireless connectivity, you won't see this screen.

Click Next let the wizard check for updates to Raspbian and install them (this might take a little while).



Click Done or Reboot to finish the setup.

Note: you will only need to reboot if that's necessary to complete an update.



WEEK-3

Aim: Write a program to transfer sensor data to smart phone using Bluetooth on Arduino.

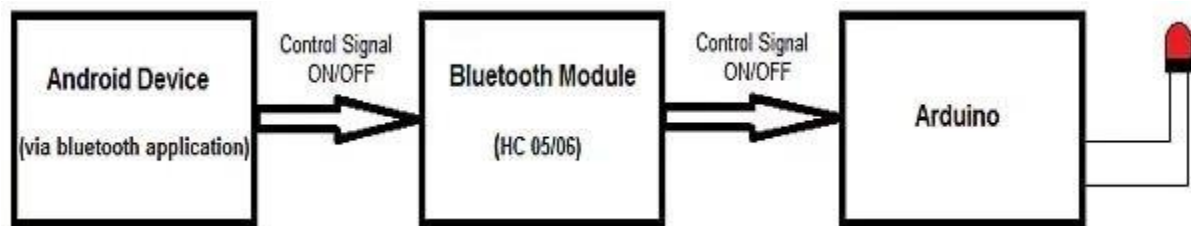
Hardware Requirements:

1. Arduino UNO
2. Android Smartphone that has Bluetooth.
3. **HC-05** Bluetooth Module
4. Android Studio (To develop the required Android app)
5. USB cable for programming and powering the Arduino

Procedure:

There are three main parts to this project.

- Smartphone
- Bluetooth transceiver
- Arduino.



1. HC 05/06 works on serial communication.
2. The Android app is designed to send serial data to the Arduino Bluetooth module when a button is pressed on the app.
3. The Arduino Bluetooth module at the other end receives the data and sends it to the Arduino through the TX pin of the Bluetooth module (connected to RX pin of Arduino).
4. The code uploaded to the Arduino checks the received data and compares it. If the received data is 1, the LED turns ON.
5. The LED turns OFF when the received data is 0. You can open the serial monitor and watch the received data while connecting.

Source Code:

```
const int irPin=10;
void setup() {

    // put your setup code here, to run once:
    pinMode(irPin,INPUT);
    Serial.begin(9600);

}

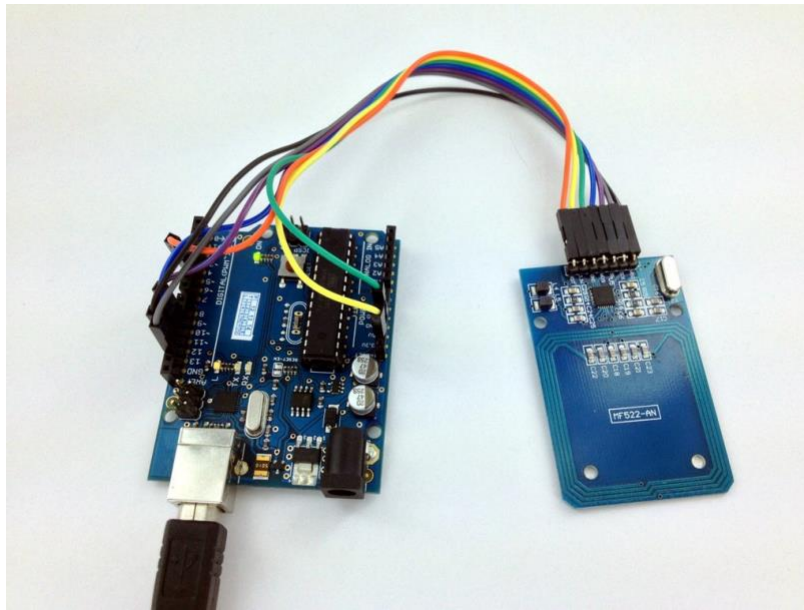
void loop() {
    // put your main code here, to run repeatedly:
    int val;
    val=digitalRead(irPin);
    Serial.println(val);
    if(val==0)
        Serial.println("Obstacle is detected ");
    else
        Serial.println("no obstacle");
    delay(1000);

}
```

OUTPUT:**0 Obstacle detected****1 no obstacle**

WEEK-4

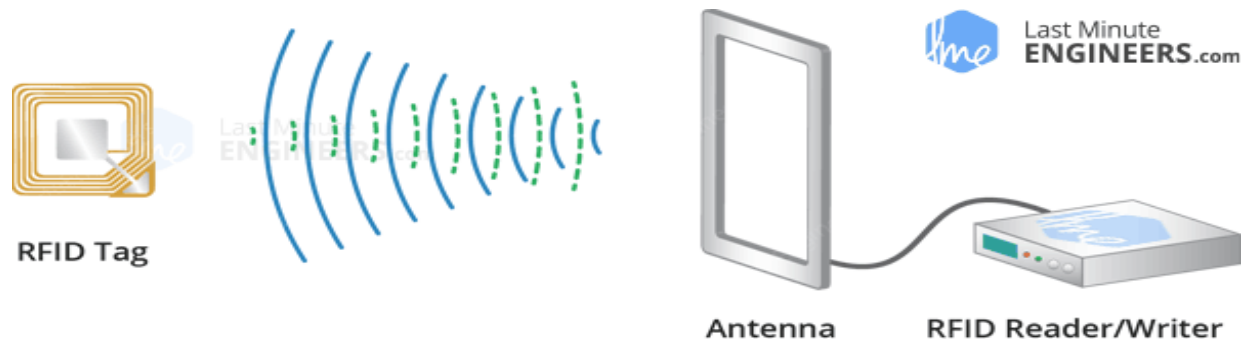
AIM: Write a program to implement RFID using Arduino.



What is RFID technology and how does it work?

An [RFID](#) or radio frequency identification system consists of two main components, a tag attached to the object to be identified, and a reader that reads the tag.

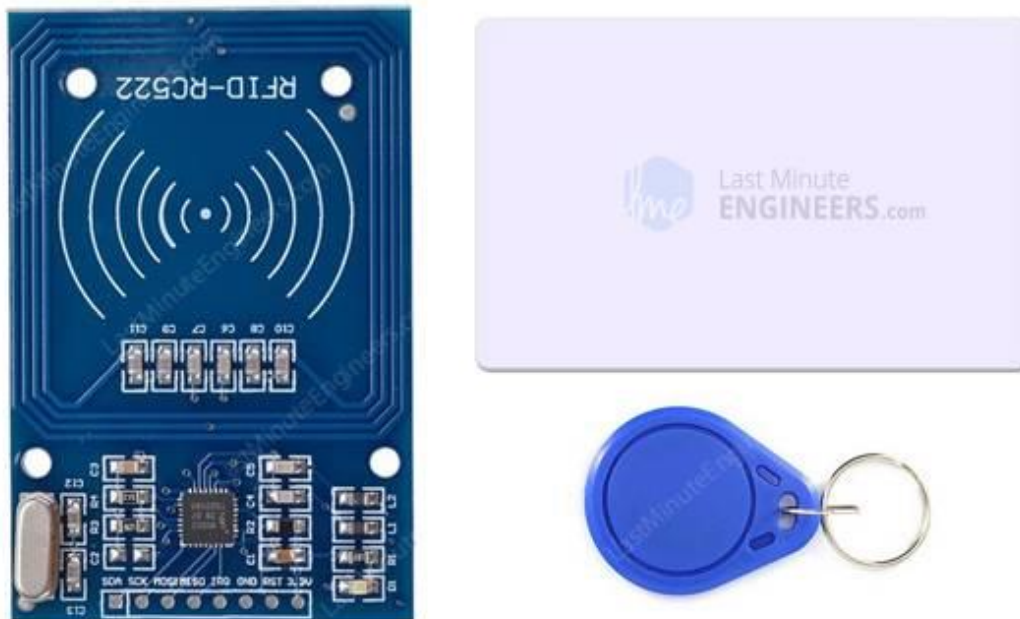
A reader consists of a radio frequency module and an antenna that generates a high frequency electromagnetic field. Whereas the tag is usually a passive device (it does not have a battery). It consists of a microchip that stores and processes information, and an antenna for receiving and transmitting a signal.



When the tag is brought close to the reader, the reader generates an electromagnetic field. This causes electrons to move through the tag's antenna and subsequently powers the chip.

Hardware Overview

The RC522 RFID module based on the MFRC522 IC is one of the cheapest RFID options you can get online for less than four dollars. It usually comes with an RFID card tag and a key fob tag with 1KB of memory. And the best part is that it can write a tag that means you can store any message in it.



The RC522 RFID reader module is designed to create a 13.56MHz electromagnetic field and communicate with RFID tags (ISO 14443A standard tags).

The reader can communicate with a microcontroller over a 4-pin SPI with a maximum data rate of 10 Mbps. It also supports communication over I2C and UART protocols.

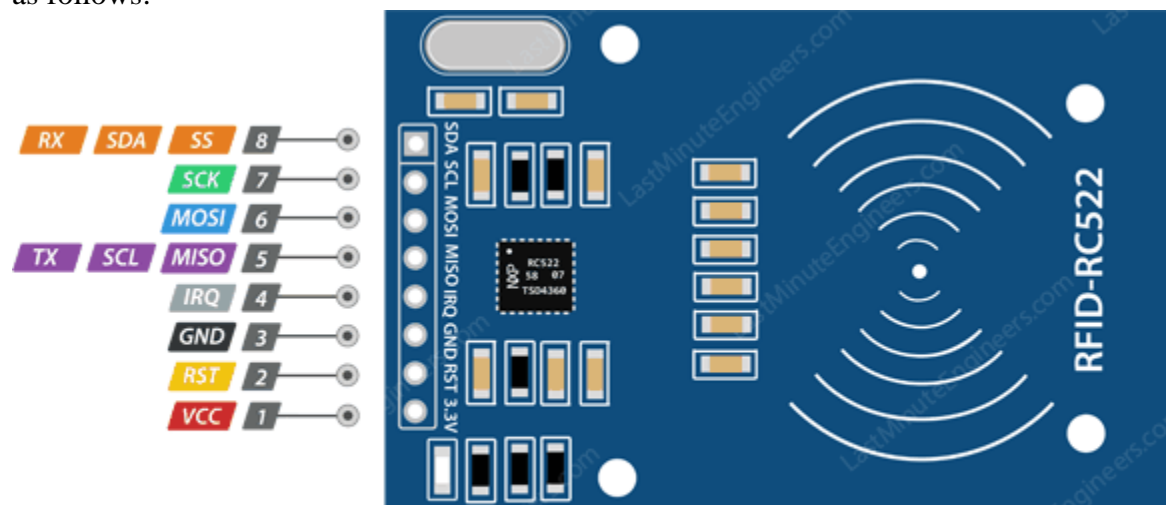
The RC522 RFID module can be programmed to generate an interrupt, allowing the module to alert us when a tag approaches it, instead of constantly asking the module "Is there a card nearby?".

The module's operating voltage ranges from 2.5 to 3.3V, but the good news is that the logic pins are 5-volt tolerant, so we can easily connect it to an Arduino or any 5V logic microcontroller without using a logic level converter.

Technical Specifications

Frequency Range	13.56 MHz ISM Band
Host Interface	SPI / I2C / UART
Operating Supply Voltage	2.5 V to 3.3 V
Max. Operating Current	13-26mA
Min. Current(Power down)	10 μ A
Logic Inputs	5V Tolerant
Read Range	5 cm

The RC522 module has a total of 8 pins that connect it to the outside world. The connections are as follows:



RC522 Pinout



VCC supplies power to the module. This can be anywhere from 2.5 to 3.3 volts. You can connect it to the 3.3V output from your Arduino. But remember that connecting it to the 5V pin will probably destroy your module!

RST is an input for reset and power-down. When this pin goes low the module enters power-down mode. In which the oscillator is turned off and the input pins are disconnected from the outside world. Whereas the module is reset on the rising edge of the signal.

GND is the ground pin and needs to be connected to the GND pin on the Arduino.

IRQ is an interrupt pin that alerts the microcontroller when an RFID tag is in the vicinity.

MISO / SCL / Tx pin acts as master-in-slave-out when SPI interface is enabled, as serial clock when I2C interface is enabled and as serial data output when the UART interface is enabled.

MOSI (Master Out Slave In) is the SPI input to the RC522 module.

SCK (Serial Clock) accepts the clock pulses provided by the SPI bus master i.e. Arduino.

SS / SDA / Rx pin acts as a signal input when the SPI interface is enabled, as serial data when the I2C interface is enabled and as a serial data input when the UART interface is enabled. This pin is usually marked by encasing the pin in a square so that it can be used as a reference to identify other pins.

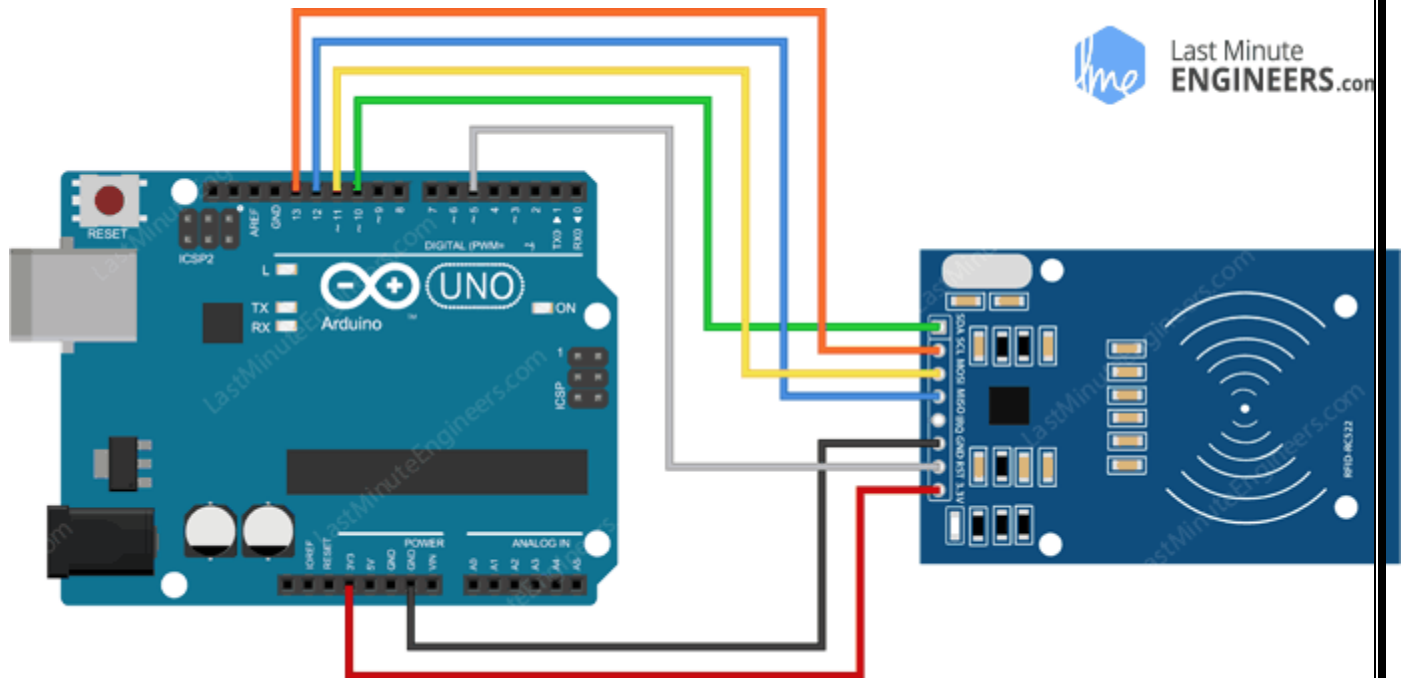
First connect the VCC pin on the module to 3.3V and the GND pin to ground on the Arduino. Pin RST can be connected to any digital pin on the Arduino. In our case, it is connected to digital pin #5. The IRQ pin is left unconnected because the Arduino library we are going to use does not support it.

Now we are left with the pins that are used for SPI communication. Since RC522 modules require a lot of data transfer, they will give the best performance when connected to the hardware SPI pins on the microcontroller.

Note that each Arduino board has different SPI pins that must be connected accordingly. Check the table below for quick understanding.

	MOSI	MISO	SCK	CS
Arduino Uno	11	12	13	10
Arduino Nano	11	12	13	10
Arduino Mega	51	50	52	53

If you are using a different Arduino than the boards mentioned above, please check the Arduino's official [documentation](#) before proceeding.



Wiring RC522 RFID Reader Writer Module with Arduino UNO

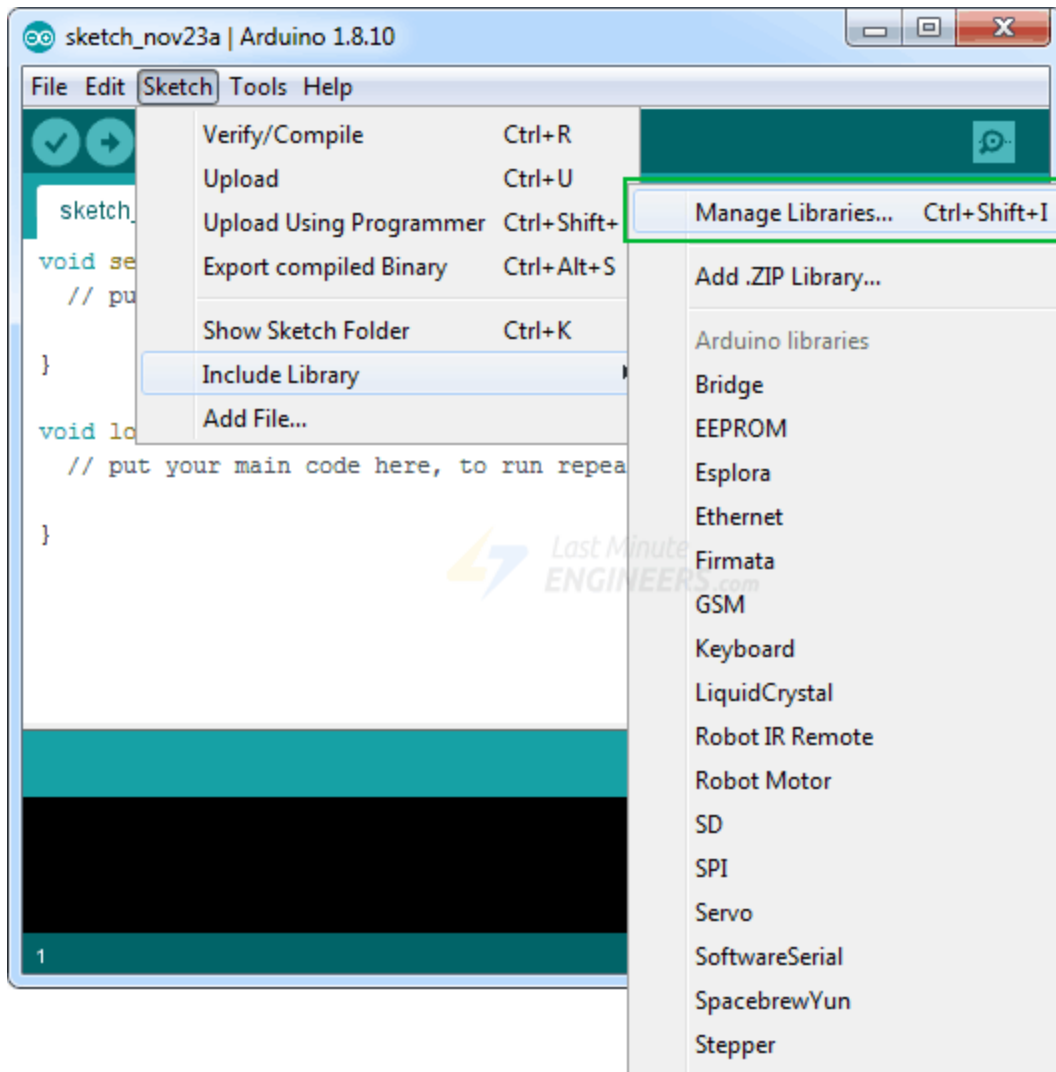
Once you have connected everything you are ready to go!

Library Installation

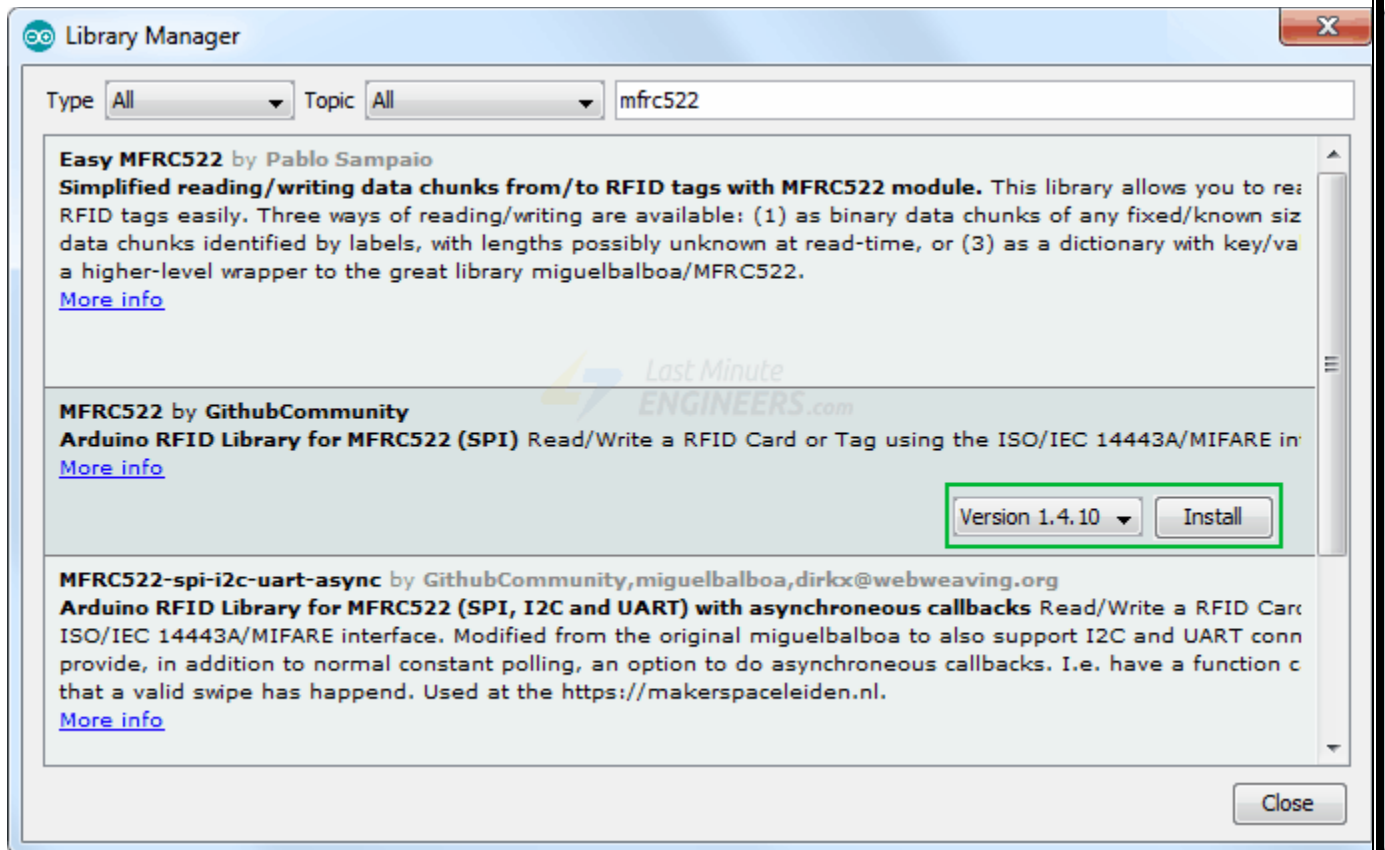
Communicating with an RC522 RFID module is a lot of work, but luckily for us there is a library called the [MFRC522 library](#) that makes reading and writing RFID tags simple.

This library is not included in the Arduino IDE, so you will need to install it first.

To install the library navigate to Sketch > Include Libraries > Manage Libraries... Wait for Library Manager to download the library index and update the list of installed libraries.

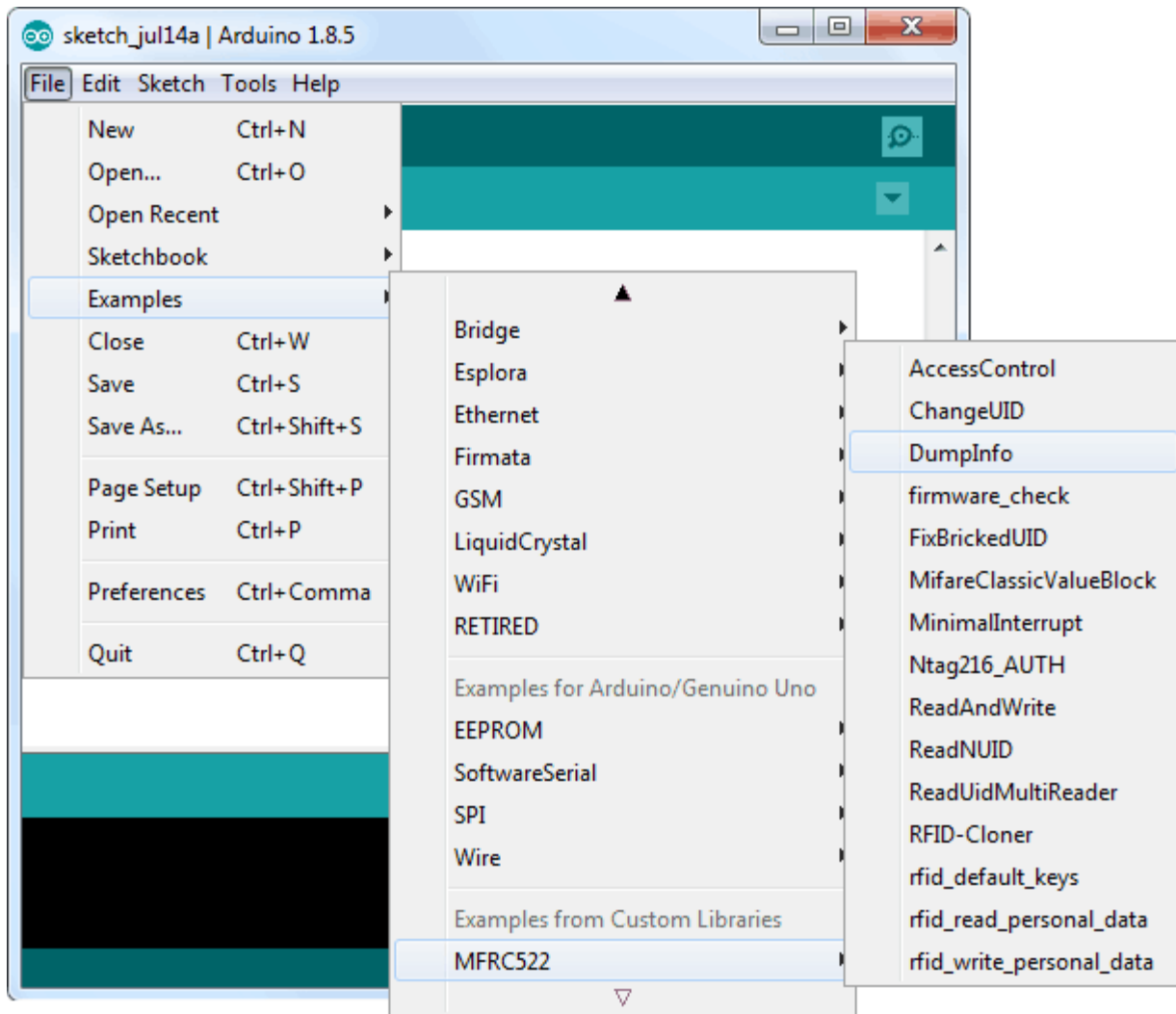


Filter your search by typing 'mfrc522'. Look for the library by GithubCommunity. Click on that entry, and then select Install.



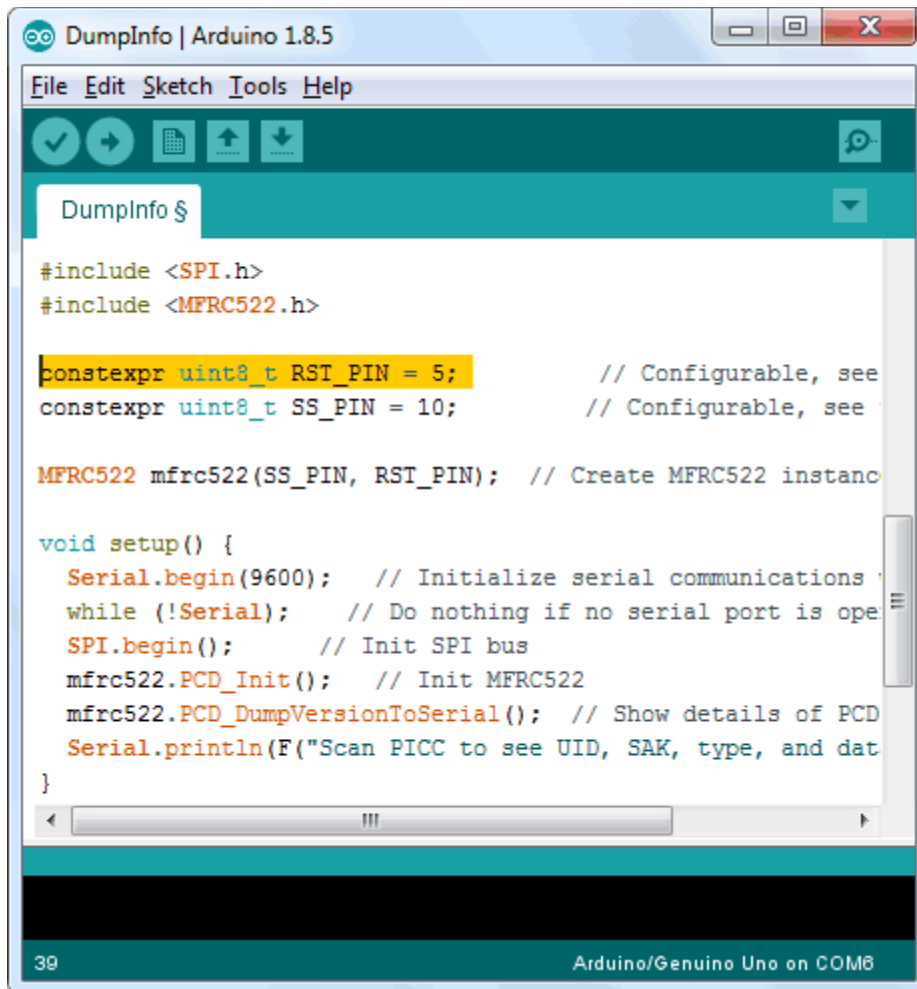
Arduino Code – Reading an RFID Tag

Once you have installed the library, open the Examples submenu and choose MFRC522 > DumpInfo example sketch.



This sketch just reads the tag and displays the information stored in it. This sketch can be very handy before trying out any new tags!

Go to the beginning of the sketch and make sure RST_PIN is initialized correctly, in our case we are using digital pin #5 so change it to 5.



```
DumpInfo | Arduino 1.8.5
File Edit Sketch Tools Help

DumpInfo $

#include <SPI.h>
#include <MFRC522.h>

constexpr uint8_t RST_PIN = 5;           // Configurable, see
constexpr uint8_t SS_PIN = 10;          // Configurable, see

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance

void setup() {
  Serial.begin(9600); // Initialize serial communications
  while (!Serial);   // Do nothing if no serial port is open
  SPI.begin();        // Init SPI bus
  mfrc522.PCD_Init(); // Init MFRC522
  mfrc522.PCD_DumpVersionToSerial(); // Show details of PCD
  Serial.println(F("Scan PICC to see UID, SAK, type, and data..."));
}

//
```

Now upload the sketch and open Serial Monitor. As you bring the tag closer to the module, you'll get something like the following. Do not move the tag until all the information is displayed.

Source code:

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN A5

MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class

MFRC522::MIFARE_Key key;

// Init array that will store new NUID
byte nuidPICC[4];

void setup() {
  Serial.begin(9600);
  SPI.begin(); // Init SPI bus
  rfid.PCD_Init(); // Init MFRC522

  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }

}

void loop() {

  // Look for new cards
  if ( ! rfid.PICC_IsNewCardPresent())
    return;

  // Verify if the NUID has been readed
  if ( ! rfid.PICC_ReadCardSerial())
    return;

  for (byte i = 0; i < 4; i++) {
    nuidPICC[i] = rfid.uid.uidByte[i];
  }

  printHex(rfid.uid.uidByte, rfid.uid.size);
  Serial.println();
  rfid.PICC_HaltA();

  rfid.PCD_StopCrypto1();
}
```

```
void printHex(byte *buffer, byte bufferSize) {  
    for (byte i = 0; i < bufferSize; i++) {  
        Serial.print(buffer[i] < 0x10 ? " 0" : "");  
        Serial.print(buffer[i], HEX);  
    }  
}
```

Output:

6D 41 2F 31

AC 9C 01 32

WEEK-5

AIM: Write a program to monitor temperature and humidity using Arduino .

Hardware Requirements:

1. Arduino UNO Board
2. DHT11 Temperature and Humidity Sensor(3 pins)
3. Jumper Wires
4. Bread Board

Procedure:

1. Connect pin 1 (on the left) of the sensor to +5V.

NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1 // to 3.3V instead of 5V!

2. Connect pin 2 of the sensor to whatever your DHTPIN is
3. Connect pin 4 (on the right) of the sensor to GROUND
4. Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor.

+ - 5v

Out- digital pin2

- - GND

Sketch->include library->manage library ->dht->dht sensor library by adafruit then install.

Goto file->Examples ->dht sensor library ->open dht tester

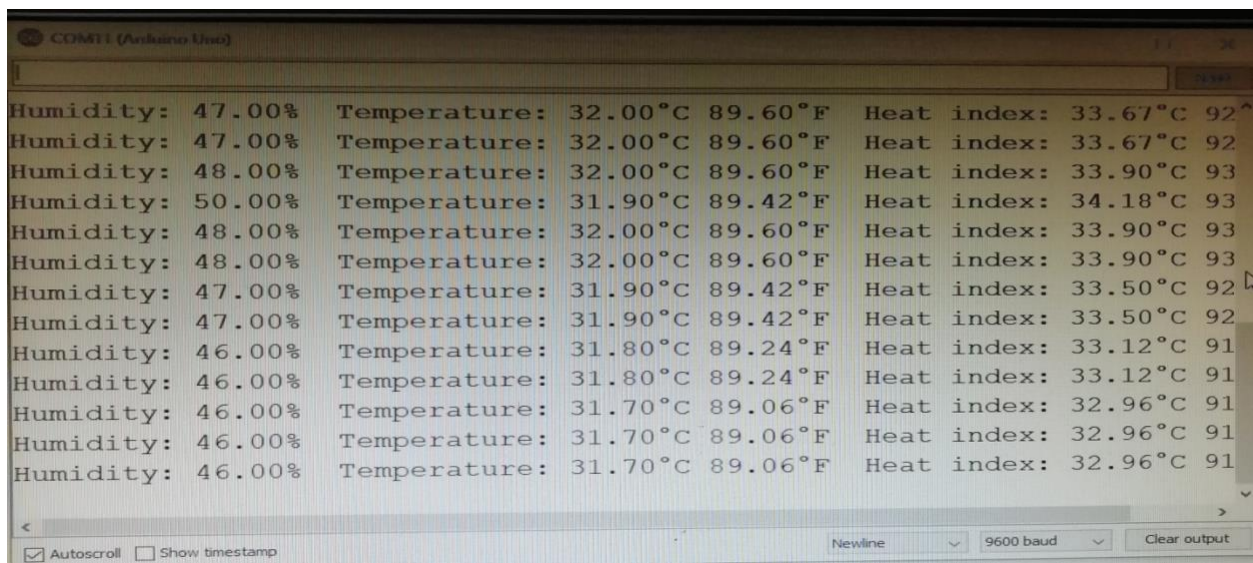
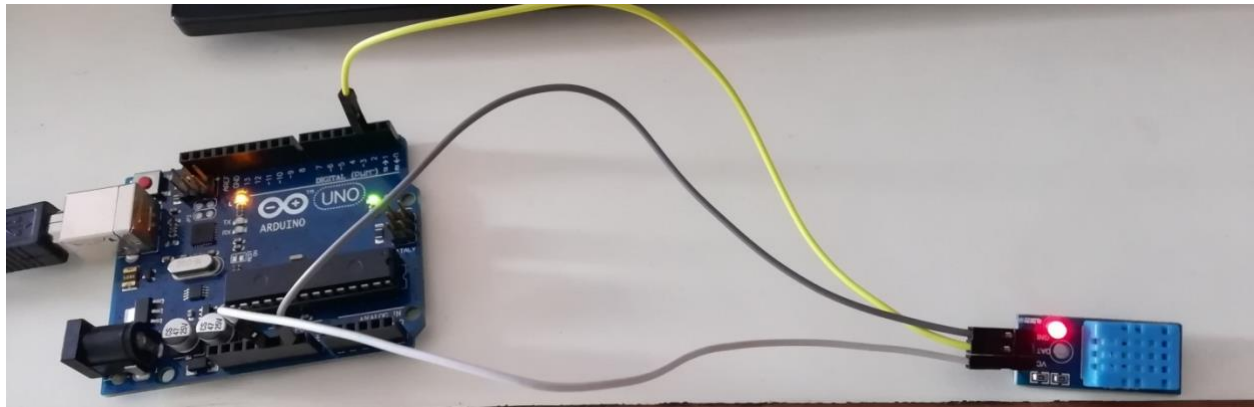
Source code :

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
// #define DHTTYPE DHT21 // DHT 21 (AM2301)
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Serial.println(F("DHTxx test!"));
  dht.begin();
```



```
}  
void loop() {  
  // Wait a few seconds between measurements.  
  delay(2000);  
  // Reading temperature or humidity takes about 250 milliseconds!  
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)  
  float h = dht.readHumidity();  
  // Read temperature as Celsius (the default)  
  float t = dht.readTemperature();  
  // Read temperature as Fahrenheit (isFahrenheit = true)  
  float f = dht.readTemperature(true);  
  // Check if any reads failed and exit early (to try again).  
  if (isnan(h) || isnan(t) || isnan(f)) {  
    Serial.println(F("Failed to read from DHT sensor!"));  
    return;  
  }  
  // Compute heat index in Fahrenheit (the default)  
  float hif = dht.computeHeatIndex(f, h);  
  // Compute heat index in Celsius (isFahreheit = false)  
  float hic = dht.computeHeatIndex(t, h, false);  
  Serial.print(F("Humidity: "));  
  Serial.print(h);  
  Serial.print(F("% Temperature: "));  
  Serial.print(t);  
  Serial.print(F("°C "));  
  Serial.print(f);  
  Serial.print(F("°F Heat index: "));  
  Serial.print(hic);  
  Serial.print(F("°C "));  
  Serial.print(hif);  
  Serial.println(F("°F"));  
}
```

OUTPUT:



Experiment 5(b): Write a program to monitor temperature and humidity using Raspberry Pi.

Procedure:

DHT 22 sensor:

Pin connection

+ -> pin is connected to raspberry pi pin1

Data -> pin is connected to raspberry pi pin7

- -> Negative pin is connected to raspberry pi pin6

sudo apt-get update

```
sudo apt-get install build-essential python-dev python-openssl git
```

```
sudo pip3 install adafruit-circuitpython-dht
```

```
sudo apt-get install libgpiod2
```

source code:

```
import time
```

```
import board
```

```
import adafruit_dht
```

```
dhtDevice = adafruit_dht.DHT22(board.D4, use_pulseio=False)
```

```
while True:
```

```
    try:
```

```
        # Print the values to the serial port
```

```
        temperature_c = dhtDevice.temperature
```

```
        temperature_f = temperature_c * (9 / 5) + 32
```

```
        humidity = dhtDevice.humidity
```

```
        print(
```

```
            "Temp: {:.1f} F / {:.1f} C  Humidity: {}% ".format(
                temperature_f, temperature_c, humidity
```

```
            )
```

```
        )
```

```
    except RuntimeError as error:
```

```
        # Errors happen fairly often, DHT's are hard to read, just keep going
```

```
        print(error.args[0])
```

```
        time.sleep(2.0)
```

```
        continue
```

```
    except Exception as error:
```

```
        dhtDevice.exit()
```

```
        raise error
```

```
    time.sleep(2.0)
```

output:

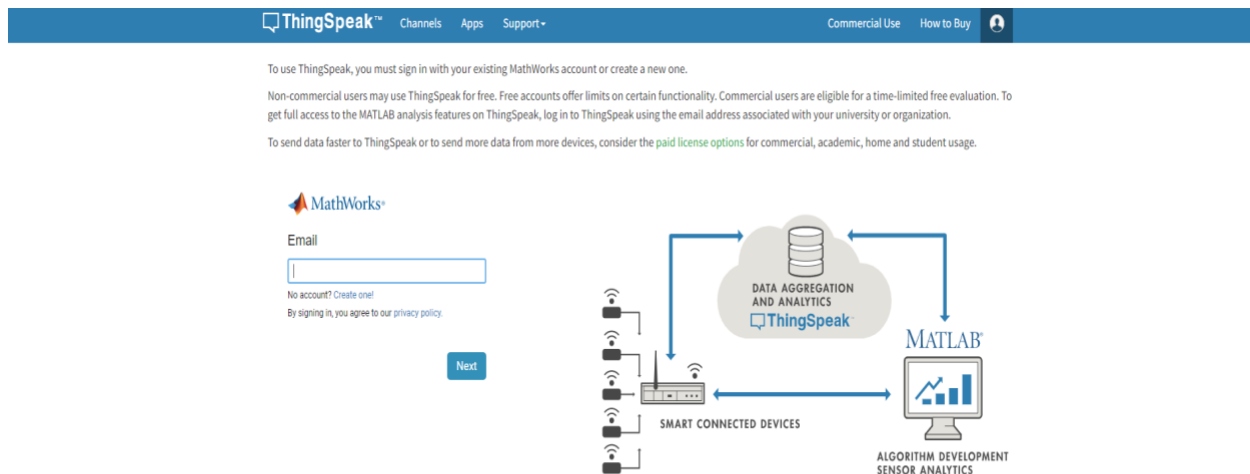
Temperature : 86.9F/30.5 C

Humidity: 65.7

WEEK-6

AIM: Write a program to interface IR Sensor with Arduino using IoT cloud application.

IOT-Cloud Application:

THINGSPEAK

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy VC

Signed in successfully. X

My Channels

New Channel Search by tag

Name	Created	Updated
Veeru example	2022-12-21	2022-12-21 08:54

Private Public Settings Sharing API Keys Data Import / Export

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to **create channels**, explore and transform data.

Learn more about **ThingSpeak Channels**.

Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

Upgrade

Need to send more data faster?

Need to use ThingSpeak for a commercial project?

Upgrade

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy VC

New Channel

Name

Description

Field 1 Field Label 1 ☒

Field 2 ☐

Field 3 ☐

Field 4 ☐

Field 5 ☐

Field 6 ☐

Field 7 ☐

Field 8 ☐

Metadata

Tags

(Tags are comma separated)

Link to External Site

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
 - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
 - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
 - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.
- **Video URL:** If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.
- **Link to GitHub:** If you store your ThingSpeak code on GitHub®, specify the GitHub repository URL.

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy VC

Veeru example

Channel ID: 1989368

Author: mwao000028692115

Access: Private

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: 3 days ago

Entries: 0

Field 1 Chart

Veeru example

Field Label 1

Date

ThingSpeak.com

Write API Key

Key: LC4THR1YB01G3UP7

Generate New Write API Key

Read API Keys

Key: GIV3TZLIV9RBEVSL

Note:

Save Note Delete API Key

Add New Read API Key

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

GET `https://api.thingspeak.com/update?api_key=LC4THR1YB01G3UP7&field=`

Read a Channel Feed

GET `https://api.thingspeak.com/channels/1989368/feeds.json?api_key=`

Read a Channel Field

GET `https://api.thingspeak.com/channels/1989368/feeds.json?api_key=`

Procedure:

1. file->preferences ->additional boards manager URLs arduino.esp8266 link is copied.
`http://arduino.esp8266.com/stable/package_esp8266com_index.json`
2. Sketch->include library->manage library->thing speak library install.
3. Tools->board node mcu1.0 ->board manager ->ESP8266 install.
4. File->Examples->Thingspeak->ESP8266->program board directly->write single field.
5. Tools->board node MCU1.0->board manager->node MCU1.0 is selected.
6. Tools->port is selected.

Pin connection Irsensor:

Vcc- vin

GND-GND

Data-D5

Source code:

```
#include <ESP8266WiFi.h>
#include "secrets.h"
#include "ThingSpeak.h" // always include thingspeak header file after other header
files and custom macros

char ssid[] = SECRET_SSID; // your network SSID (name)
char pass[] = SECRET_PASS; // your network password
int keyIndex = 0;          // your network key Index number (needed only for WEP)
WiFiClient client;

unsigned long myChannelNumber = SECRET_CH_ID;
const char * myWriteAPIKey = SECRET_WRITE_APIKEY;

int number = 0;
const int irPin=D5;

void setup() {
  Serial.begin(115200); // Initialize serial
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo native USB port only
  }

  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client); // Initialize ThingSpeak
}
```



```
void loop() {

    // Connect or reconnect to WiFi
    if(WiFi.status() != WL_CONNECTED){
        Serial.print("Attempting to connect to SSID: ");
        Serial.println(SECRET_SSID);
        while(WiFi.status() != WL_CONNECTED){
            WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if
using open or WEP network
            Serial.print(".");
            delay(5000);
        }
        Serial.println("\nConnected.");
    }

    // 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.
    int x = ThingSpeak.writeField(myChannelNumber, 1, number, myWriteAPIKey);
    if(x == 200){
        Serial.println("Channel update successful.");
    }
    else{
        Serial.println("Problem updating channel. HTTP error code " + String(x));
    }

    // change the value
```

```
number++;  
if(number > 99){  
    number = 0;  
}  
delay(20000); // Wait 20 seconds to update the channel again  
}  
// Use this file to store all of the private credentials  
// and connection details  
  
#define SECRET_SSID "Redmi Note 11 Pro+ 5G"      // replace MySSID with  
your WiFi network name  
#define SECRET_PASS "mani@123" // replace MyPassword with your WiFi  
password  
#define SECRET_CH_ID 2038959                    // replace 0000000 with your  
channel number  
#define SECRET_WRITE_APIKEY "7W61L6N4VY7ZFBZP" // replace XYZ  
with your channel write API Key
```

Output:

irsensor

Channel ID: 2058364

Author: mwa0000028732251

Access: Private

Private View

Public View

Channel Settings

Sharing

API Keys

Data Import / Export

+ Add Visualizations

+ Add Widgets

Export recent data

MATLAB Analysis

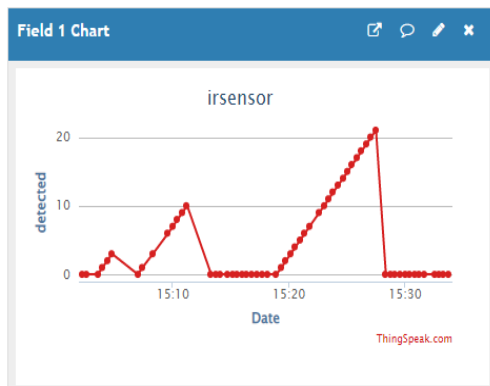
MATLAB Visualization

Channel 1 of 2 < >

Channel Stats

Created: about a month agoLast entry: about a month ago

Entries: 64



WEEK-7

Aim: Write a program Upload temperature and humidity data to the cloud using Arduino or Raspberry Pi.

Hardware Requirements:

1. Arduino UNO board
2. NodeMCU ESP8266 Breakout Board
3. DHT-11 temperature and humidity sensor
4. Jumper wires
5. Bread board
6. WIFI Network

Procedure:

1. Download esp8266 Zip file->go to libraries->add Zip file
2. Connect Node MCU, Go to tools->change board to Node MCU esp8266 and port number
3. Connect DHT-11 temperature and Humidity sensor to Node MCU
4. Sign up to cloud->open THINGSPEAK>create channels->copy API key to the source code
5. SSID and password of your WIFI connection should be given in source code
6. Compile and upload the program and verify the temperature and humidity readings in serial monitor
7. Go to cloud and verify the temperature and humidity values in graph.

DHT 11 sensor pin connection:

+ - vin

Out- D3

- - GND

Write multiple fields:

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

```
#include "secrets.h"
```

```
#include "ThingSpeak.h"
```

```
#include "DHT.h"

#define DHTTYPE DHT11
#define DHTPIN D3

// always include thingspeak header file after other header files and custom macros

char ssid[] = SECRET_SSID; // your network SSID (name)
char pass[] = SECRET_PASS; // your network password
int keyIndex = 0;
DHT dht(DHTPIN, DHTTYPE); // your network key Index number (needed only for
WEP)
WiFiClient client;

unsigned long myChannelNumber = SECRET_CH_ID;
const char * myWriteAPIKey = SECRET_WRITE_APIKEY;

// Initialize our values
int number1 = 0;
int number2 = random(0,100);
int number3 = random(0,100);
int number4 = random(0,100);
String myStatus = "";

void setup() {
  dht.begin();
  Serial.begin(115200); // Initialize serial
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo native USB port only
```

```
}

WiFi.mode(WIFI_STA);
ThingSpeak.begin(client); // Initialize ThingSpeak
}

void loop() {

    // Connect or reconnect to WiFi
    if(WiFi.status() != WL_CONNECTED){
        Serial.print("Attempting to connect to SSID: ");
        Serial.println(SECRET_SSID);
        while(WiFi.status() != WL_CONNECTED){
            WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if
using open or WEP network
            Serial.print(".");
            delay(5000);
        }
        Serial.println("\nConnected.");
    }

    // set the fields with the values
    ThingSpeak.setField(1, number1);
    ThingSpeak.setField(2, number2);
    ThingSpeak.setField(3, number3);
    ThingSpeak.setField(4, number4);
```

```
// figure out the status message
if(number1 > number2){
    myStatus = String("field1 is greater than field2");
}
else if(number1 < number2){
    myStatus = String("field1 is less than field2");
}
else{
    myStatus = String("field1 equals field2");
}

// set the status
ThingSpeak.setStatus(myStatus);

// write to the ThingSpeak channel
int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
if(x == 200){
    Serial.println("Channel update successful.");
}
else{
    Serial.println("Problem updating channel. HTTP error code " + String(x));
}

// change the values
number1++;
if(number1 > 99){
    number1 = 0;
```

```
}  
number2 = random(0,100);  
number3 = random(0,100);  
number4 = random(0,100);  
  
delay(20000); // Wait 20 seconds to update the channel again  
}
```

Secret.h

// Use this file to store all of the private credentials

// and connection details

```
#define SECRET_SSID "samsung on7 pro"           // replace MySSID with your  
WiFi network name
```

```
#define SECRET_PASS "Venkata123#"             // replace MyPassword with your WiFi  
password
```

```
#define SECRET_CH_ID 2040479                   // replace 0000000 with your  
channel number
```

```
#define SECRET_WRITE_APIKEY "0RNY24517GJ9LRWD" // replace XYZ  
with your channel write API Key
```


Output:

DHT11sensor

Channel ID: 2072605

Author: mwa0000028732251

Access: Private

[Private View](#) [Public View](#) [Channel Settings](#) [Sharing](#) [API Keys](#) [Data Import / Export](#)[+ Add Visualizations](#)[+ Add Widgets](#)[Export recent data](#)[MATLAB Analysis](#)[MATLAB Visualization](#)

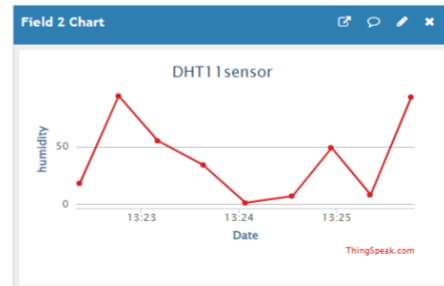
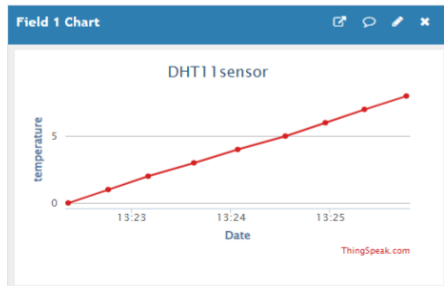
Channel 2 of 2 < >

Channel Stats

Created: about a month ago

Last entry: about a month ago

Entries: 9

Activate Windows
Go to Settings to activate Windows.

WEEK-8

Aim: Write a program on Arduino or Raspberry Pi to retrieve temperature and humidity data from the Cloud

Hardware Requirements:

1. Arduino UNO board
2. NodeMCU ESP8266 Breakout Board
3. DHT-11 temperature and humidity sensor
4. Jumper wires
5. Bread board

Procedure:

1. Open up the Arduino IDE and head over to the library manager.
2. Install the [DHT library](#) (You can also install it by going to Sketch > Include Library > Manage Libraries, and search for adafruit dht library).

DHT sensor with 3 pins:

1. Power supply 3.5V to 5.5V.
2. Data, Outputs both Temperature and Humidity through serial Data.
3. Ground, Connected to the ground of the circuit.

Set up in source code:

1. Set your Wi-Fi SSID and password.
2. Set the API Key
3. Save->Compile->upload->now us can visualize our data in cloud

Source code:

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>
const char ssid[] = "Redmi Note 11 Pro+ 5G"; // your network SSID (name)
const char pass[] = "mani@123"; // your network password
int statusCode = 0;
```

WiFiClient client;

//-----Channel Details-----//

unsigned long counterChannelNumber = 2062499; // Channel ID

const char * myCounterReadAPIKey = "W5564YQ3MJPU0S7V"; // Read API Key

const int FieldNumber1 = 1; // The field you wish to read

const int FieldNumber2 = 2; // The field you wish to read

//-----//

void setup()

{

Serial.begin(115200);

WiFi.mode(WIFI_STA);

ThingSpeak.begin(client);

}

void loop()

{

//----- Network -----//

if (WiFi.status() != WL_CONNECTED)

{

Serial.print("Connecting to ");

Serial.print(ssid);

Serial.println("");

while (WiFi.status() != WL_CONNECTED)

{

WiFi.begin(ssid, pass);

delay(5000);

}

Serial.println("Connected to Wi-Fi Succesfully.");

}

```
//----- End of Network connection-----//

//----- Channel 1 -----//
long    temp    =    ThingSpeak.readLongField(counterChannelNumber,    FieldNumber1,
myCounterReadAPIKey);
statusCode = ThingSpeak.getLastReadStatus();
if (statusCode == 200)
{
    Serial.print("Temperature: ");
    Serial.println(temp);
}
else
{
    Serial.println("Unable to read channel / No internet connection");
}
delay(100);
//----- End of Channel 1 -----//

//----- Channel 2 -----//
long    humidity    =    ThingSpeak.readLongField(counterChannelNumber,    FieldNumber2,
myCounterReadAPIKey);
statusCode = ThingSpeak.getLastReadStatus();
if (statusCode == 200)
{
    Serial.print("Humidity: ");
    Serial.println(humidity);
}
else
{
    Serial.println("Unable to read channel / No internet connection");
}
```

```
delay(100);  
//----- End of Channel 2 -----//  
}
```

Output:

Connected to wifi Successfully

Humidity :63

Temperature : 9

WEEK-9

AIM: Write a program to create TCP Server on cloud using Arduino and Respond with humidity data to TCP Client when requested.

An IoT device can be made to communicate with a cloud or server using TCP/IP protocol without any hassle of network programming and network administration. In this project, an IoT device will be designed that could transmit sensor data to ThingSpeak Platform using the TCP/IP protocol.

The IoT device designed in this project is built using Arduino UNO. The Arduino is just a microcontroller board and cannot connect to an internet network on its own. For internet connectivity, the Arduino UNO is interfaced with ESP8266 module. The ESP8266 Wi-Fi Module is a self contained SOC with integrated TCP/IP protocol stack that can access to a Wi-Fi network. The ESP module allows the Arduino board to connect with a router and access internet network. The Arduino is programmed to communicate with the cloud platform ThingSpeak over TCP/IP protocol. The Arduino can implement TCP/IP protocol by passing AT commands serially to the ESP8266 module.

The IoT device designed is a visitor counter as well as temperature and humidity monitor. For working as visitor counter, the IR sensors and Photodiodes are interfaced with the Arduino board. For working as temperature and humidity monitor, a DHT-11 sensor is interfaced with the Arduino board. The Arduino reads data from the sensors and send it to the ThingSpeak platform. A 0.6 inch 128 X 64 OLED is also interfaced with the Arduino which receives serial data from the board on I2C protocol and display the current temperature and humidity readings. The user can monitor the number of occupants in the house, temperature and humidity values from anywhere by accessing the ThingSpeak platform.

The Arduino board controls all the functionalities of the IoT device like counting visitors, reading temperature and humidity values from DHT-11 sensor, displaying temperature and humidity data on OLED, implementing TCP/IP protocol, connecting with ThingSpeak platform and sending data to the cloud server.

For this, the Arduino code is written and compiled using Arduino IDE.

Software Required –

- ThingSpeak server
- Arduino IDE

The IoT device that communicates with the ThingSpeak Cloud is built on Arduino UNO. The DHT-11 Sensor, IR sensor, Photodiodes, ESP8266 module and OLED module are interfaced with the Arduino board to make the IoT device.

Arduino: It is an Atmega 328 based controller board which has 14 GPIO pins, 6 PWM pins, 6 Analog inputs and on board UART, SPI and TWI interfaces. The Atmega 328 is the sitting MCU on the Arduino board. There are two GPIO pins (external interrupt pins INT0 and INT1) of Arduino board used to interface photodiodes, TX and RX pins (pins 3 and 2 respectively) are used

interface ESP module, SDA and SCL (pins 27 and 28 respectively) pins used to interface the OLED module.

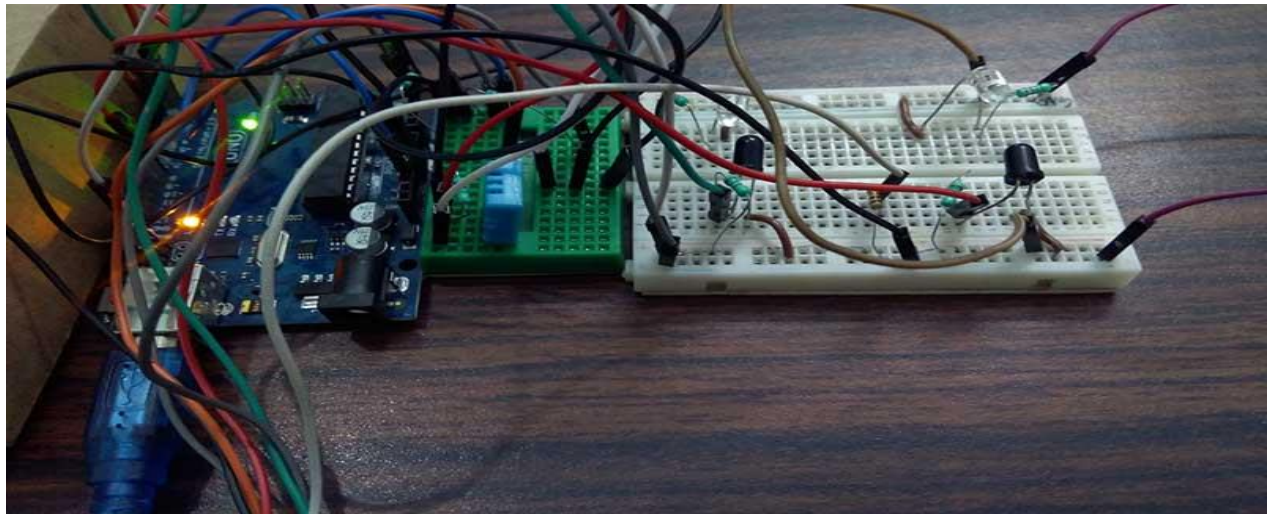
OLED Module – The OLED module is used to display the temperature and humidity information as well as the count of the visitors (occupants) in the house. The module communicates with the Arduino board using I2C protocol. This is a two wire protocol. The module has four pins – SDA, SCL, VCC and GND. The VCC and Ground are connected to 5V DC and common ground respectively. The 5V DC can be supplied through a battery via 7805 voltage regulator IC. The SDA and SCL pins of the OLED module are connected to the SDA and SCL pins ((pins 27 and 28 respectively) of the Arduino board. The OLED are made up of carbon based organic materials. So unlike LCD displays, they do not require backlight and filters.

ESP8266 Module – The ESP8266 Wi-Fi Module is a self contained SOC with integrated TCP/IP protocol stack that can access to a Wi-Fi network. The ESP8266 is capable of either hosting an application or off loading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware. The Chip Enable and VCC pins of the module are connected to the 3.3 V DC while Ground pin is connected to the common ground. The chip enable pin is connected to VCC via a 10K pull up resistor. The RESET pin is left not connected. The Tx and Rx pins of the module are connected to the RX and TX pins of the Arduino UNO. The GPIO-0 pin of the module is connected to VCC through a 10K pull up resistor.

DHT-11 Sensor – DHT-11 is a temperature and humidity sensor. The DHT11 sensor consists of two main components – one is Humidity sensing component and other is NTC temperature sensor (or Thermistor). The Thermistor is actually a variable resistor that changes its resistance with change in temperature. They both sense the temperature and humidity of area and give the output to the IC (which is placed on back side of sensor). The sensor has four pins – VCC, Ground, data Out and NC. The VCC and Ground pins are connected to the common VCC and Ground respectively. The Data Out pin of the sensor is connected to PD7 pin of the Arduino board via 10K pull-up resistor.

ThingSpeak Server – The ThingSpeak server is used to visualize the data received from the IoT device. The data is displayed in the form of graphs on the platform. The ThingSpeak generates the read and write API key. The Write API key is used to write the data to the channel and read API channel is used to allow other people to view private channel feeds and charts. The data can also be saved on the platform for future reference.

In order to configure the ThingSpeak platform to access the data on it, first an **account** must be created on the platform. Then a channel must be created for the data on that account. It can be done by navigating to **channel window** and creating a new channel. The required information must be filled in the given form at the website so that the needed fields are created. For this project, there must be created three fields – Total Persons, Temperature and Humidity. These fields can then be checked live on the server. After saving the channel settings, a Write API key is generated which must be noted down. This Write API key is used in the firmware code of the Arduino to access the private channel created on the ThingSpeak account.



As the circuit is powered on the Arduino board starts reading data from the IR receivers and the DHT-11 sensor. The IR receivers are connected in series with variable resistors between VCC and ground in reverse bias configuration forming a voltage divider circuit. The output from the IR receivers (photodiodes) are drawn from the junction of cathode terminals of the IR receiver. Under normal conditions, the light emitted by the IR transmitters is continuously received by the photodiodes. This keeps the digital logic output of the photodiodes to HIGH. When a person enters the house, the light from the IR transmitters is blocked and the logical output from the photodiodes is switched to LOW. The IR receivers are connected at the external interrupt pins of the Arduino, so an interrupt is generated at the INT0 and INT1 pin of the Arduino when a person enters or exits the house. There are two pairs of IR transmitter and receiver used. The sequence in which the interrupts are generated at the INT0 and INT1 pins indicate whether a person has entered or exit from the house. Accordingly, the count of the current occupants of the house is increased or decreased.

The DHT11 Temperature and Humidity Sensor is a digital sensor with inbuilt capacitive humidity sensor and Thermistor. It relays a real-time temperature and humidity reading every 2 seconds. The sensor operates on 3.5 to 5.5 V supply and can read temperature between 0° C and 50° C and relative humidity between 20% and 95%. The DHT11 detects water vapors by measuring the electrical resistance between the two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapors are absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes.

Pin connection :

DHT11 sensor

GND-GND

DATA PIN-D4

VCC-Vin

Source code:

```
#include "ESP8266WiFi.h"
#include "DHT.h"
#define DHTTYPE DHT11
const char* ssid = "Redmi Note 11 Pro+ 5G";
const char* password = "mani@123";
WiFiServer wifiServer(9000);
DHT dht(D4, DHT11);

void setup() {
  Serial.begin(115200);
  delay(1000);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting..");
  }
  Serial.print("Connected to WiFi. IP:");
  Serial.println(WiFi.localIP());
  wifiServer.begin();
  dht.begin();
}

void loop() {
  WiFiClient client = wifiServer.available();
  if (client) {
```

```
while (client.connected()) {  
  while (client.available()>0) {  
    int h = dht.readHumidity();  
    client.print("humidity :");  
    client.println(h);  
    //Serial.println(sensor_value);  
    delay(2000);  
  }  
}  
client.stop();  
Serial.println("Client disconnected");  
}  
}
```

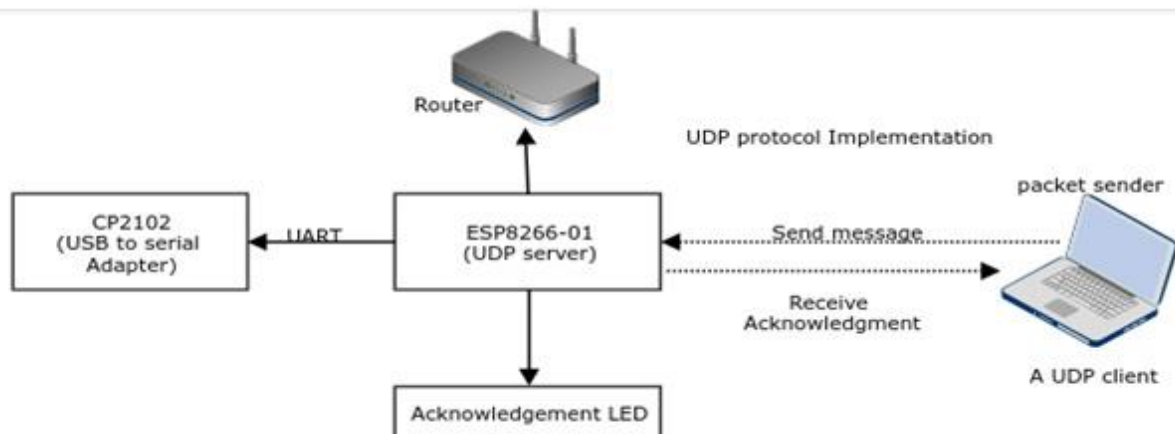
Output:**Connected to wifi IP: 192.168.102.45****TCP Client Terminal****Port :9000****Humidity:49****Humidity:55****Humidity:56****Humidity : 58**

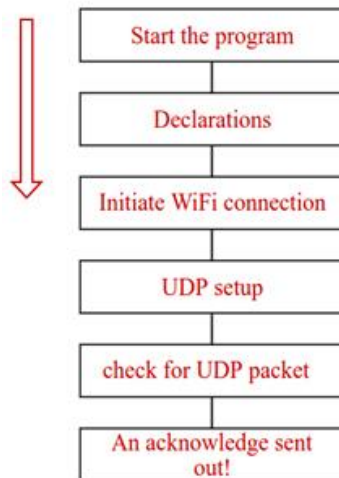
WEEK-10

AIM: Write a program to create UDP Server on cloud using Arduino and Respond with humidity data to UDP Client when requested.

The UDP protocol has a small overhead of 8 bytes which makes it more suitable for use in the Internet of Things. In this project, the application of UDP protocol in IoT will be demonstrated. In this project, an ESP8266 Wi-Fi modem will be configured as UDP server and a laptop will be used as UDP Client. Both Client and server will be co-located communicating through same Wi-Fi router so, the ESP board will act as a local server. The ESP module working as server checks for the UDP packet received from the client on a particular port. When a valid packet is arrived, an acknowledge packet is sent back to the client to the same port it has been sent out. In response to receiving packet and sending acknowledgement, the ESP modem switches on an LED as visual indication of successful Client-Server Communication. This application is very useful as it demonstrates server/client communication over UDP protocol.

It can be said that the IOT device designed in this experiment is a simple UDP server with LED indicator. It is designed by interfacing an LED with the ESP-8266 Wi-Fi module. The Wi-Fi module as well as LED light are powered continuously with the help of a USB to Serial Converter. The Wi-Fi module needs to be loaded with a firmware that could receive data over UDP protocol and respond with an acknowledgement. The Arduino UNO is used to flash the firmware code on the ESP8266 module. The ESP module can also be flashed with code using a FTDI converter like CP2102. The firmware itself is written in the Arduino IDE.



**Software Required –**

- ThingSpeak server
- Arduino IDE

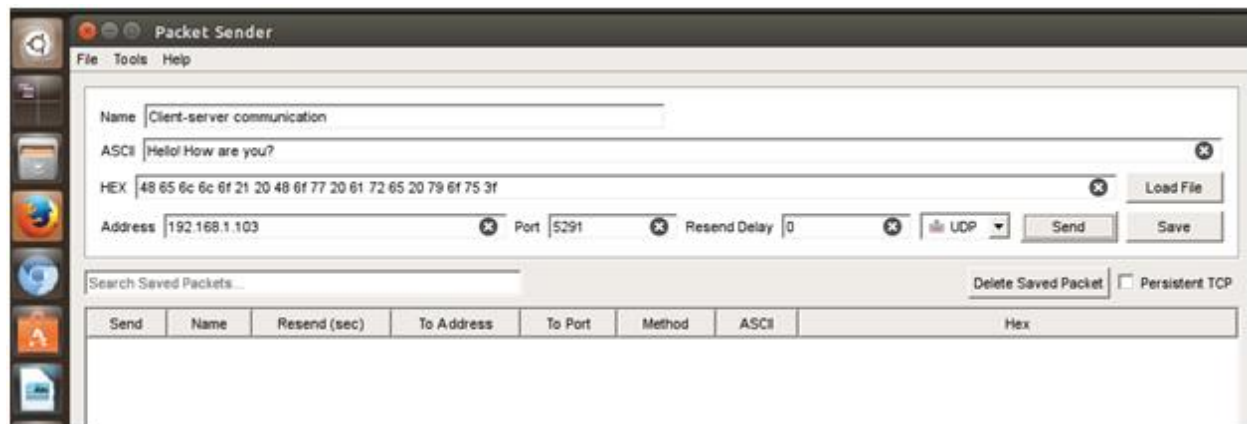
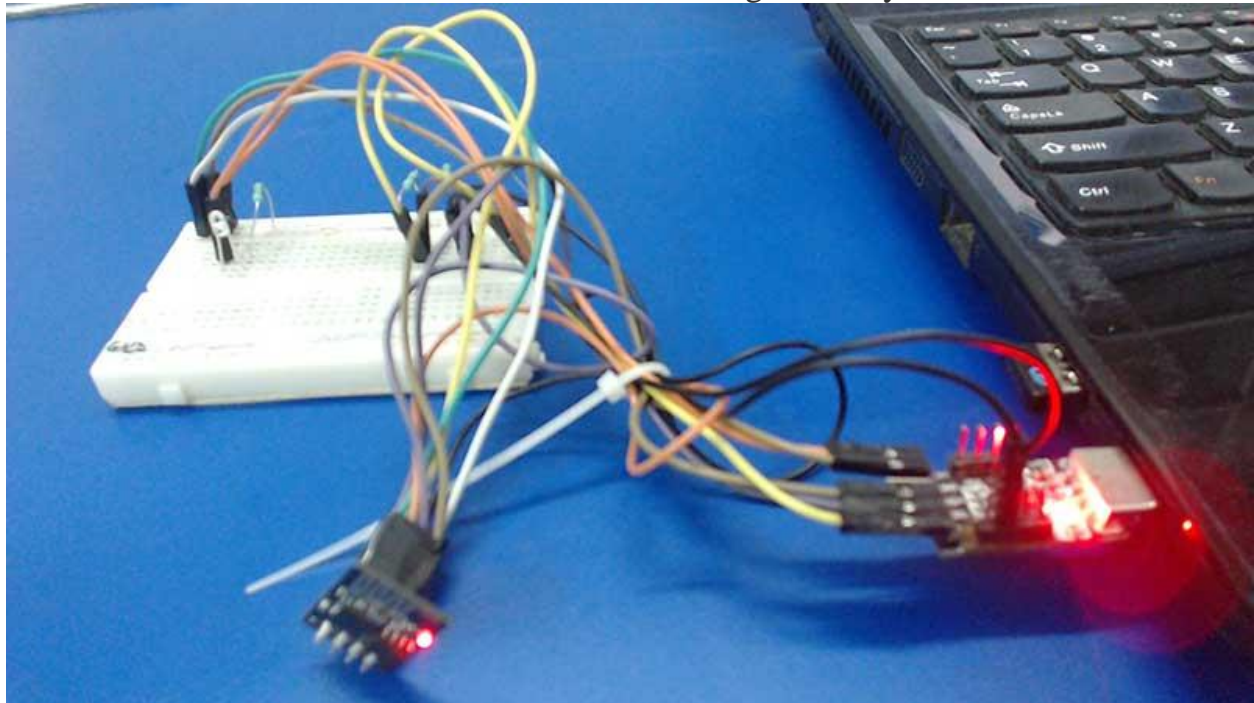
ESP8266 board needs to be loaded with the firmware code. In this tutorial, the firmware code is written using Arduino IDE. It is loaded to the ESP8266 board using the Arduino UNO. A generic ESP8266 board is used in this project. This board does not have any bootstrapping resistors, no voltage regulator, no reset circuit and no USB-serial adapter. The ESP8266 module operates on 3.3 V power supply with current greater than or equal to 250 mA. So, CP2102 USB to serial adapter is used to provide 3.3 V voltage with enough current to run ESP8266 reliably in every situation.

The ESP8266 Wi-Fi Module is a self contained SOC with integrated TCP/IP protocol stack that can access to a Wi-Fi network. The ESP8266 is capable of either hosting an application or off loading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware.

The Chip Enable and VCC pins of the module are connected to the 3.3 V DC while Ground pin is connected to the common ground. The chip enable pin is connected to VCC via a 10K pull up resistor. The RESET pin is connected to the ground via a tactile switch where the pin is supplied VCC through a 10K pull up resistor by default. The Tx and Rx pins of the module are connected to the RX and TX pins of the Arduino UNO. The GPIO-0 pin of the module is connected to ground via a tactile switch where the pin is supplied VCC through a 10K pull up resistor by default.

Write the firmware code in the Arduino IDE and connect the Arduino board with the PC via USB cable. Open Arduino IDE and go to Tools->Port and select the Arduino board (Arduino UNO). It may look like /dev/ttyABM0 (Arduino/Genuino Uno). Select the correct port name. The port name can be different in different IDE setups. Then Open Serial monitor in the Arduino IDE by navigating to Tools->Serial Monitor and set the baud rate to 115200 bauds per second. Pass 'AT' and 'AT+GMR' commands to

test the connection between the Arduino and ESP-01 module. Try different settings for the 'Line ending' option of the serial monitor like Both NL & CR. Try different combinations until the ESP module starts interacting correctly with the serial monitor.



The ESP modem act as UDP server in this setup. It is loaded with a firmware that can receive data on UDP protocol and in response send back an acknowledgement on the same port. The Laptop which is acting as UDP Client sends the datagram using packet sender application. The firmware code on the ESP modem light up an LED after successful communication with the UDP Client.

In the firmware code of the ESP8266 based UDP server, the Wi-Fi connection is initiated with the available router and the UDP connection is setup using standard library functions. A library named WifiUDP.h is imported in the ESP code for

managing the UDP connection and receiving data over it. Once the UDP packet is successfully received, the code sends an acknowledgement on the same port and set the LED connected pin to HIGH.

Pin connection:

GND-GND

DATA- D5

VCC-VIN

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

```
#include <WiFiUdp.h>
```

```
#include "DHT.h"
```

```
#define DHTTYPE DHT11
```

```
// Set WiFi credentials
```

```
#define WIFI_SSID "333-3"
```

```
#define WIFI_PASS "123456789"
```

```
#define UDP_PORT 4210
```

```
DHT dht(D5, DHT11);
```

```
// UDP
```

```
WiFiUDP UDP;
```

```
char packet[255];
```

```
char reply[] = "Packet received!";
```

```
void setup() {
```

```
    // Setup serial port
```

```
    Serial.begin(115200);
```

```
    Serial.println();
```

```
    Serial.println(F("DHTxx test!"));
```

```
    dht.begin();
```

```
    // Begin WiFi
```

```
WiFi.begin(WIFI_SSID, WIFI_PASS);

// Connecting to WiFi...
Serial.print("Connecting to ");
Serial.print(WIFI_SSID);
// Loop continuously while WiFi is not connected
while (WiFi.status() != WL_CONNECTED)
{
    delay(2000);
    Serial.print(".");
}

// Connected to WiFi
Serial.println();
Serial.print("Connected! IP address: ");
Serial.println(WiFi.localIP());

// Begin listening to UDP port
UDP.begin(UDP_PORT);
Serial.print("Listening on UDP port ");
Serial.println(UDP_PORT);

}

void loop() {
    int h = dht.readHumidity();
    //float h = dht.readHumidity();
    delay(2000);
    // Send return packet
    UDP.beginPacket(UDP.remoteIP(), UDP.remotePort());
    UDP.write(reply);
```

```
// client.print("humidity :");  
    // client.println(h);  
    UDP.println(h);  
    UDP.endPacket();  
    Serial.println(F("Humidity:\n "));  
    Serial.println(h);  
  
}
```

Output:**Connected IP Address :192.168.137.191****Listening on UDP port : 4210****Humidity : 32****Receive at 192.168.137.69 : 4210****[192.168.137.69 : 4210]:31****33****Packet received.**