KMBB College of engineering and technology Dept. Of Electronics and Communication Engineering

AN IOT BASED POJECT REPORT On

" SMART LOCKING DOOR SYSTEM "

An IOT project report submitted in partial fulfillment of the requirements for the 7th Semester degree of Bachelor of Technology in ELECTRONICS AND COMMUNICATION ENGINEERING of Biju Patnaik University of Technology, Rourkela

Submitted by

OMM PRAKASHA MAHARANA(2201329094)
PRIYA SARATHI SAMNTRAY(2201329095)
DIBYANSHU DHAL(2201329091)
ATISHRANJAN SAHU(2201329088)

BRANCH - ELECTRONICS AND COMMUNICATION ENGINEERING
KMBB COLLEGE OF ENGINEERING AND TECHNOLOGY,
Daleiput, KHURDA (2025-2026)

Principal
DR. AMAR NATH NAYAK

HOD OF ECE
PROF. RABINDRA KU. PRADHAN

BIJU PATNAIK UNIVERSITY OF TECHNOLOGY

ROURKELA, ODISHA-769004

Batch ; 2022-26

DECLARATION

We, the undersigned group members, hereby declare that the project report titled "**SMART LOCKING DOOR SYSTEM**" submitted to KMBB College of engineering and technology for the partial fulfillment of the B.Tech is our original work, completed under the supervision of Prof. Rabindra Kumar Pradhan.

We further declare that this work is authentic and has been submitted for this B.Tech degree or award. To the best of our knowledge, this report does not contain any material previously published or written by other individuals without proper acknowledgment. All data, text, images, and diagrams borrowed from other sources have been properly cited and are listed in the references.

We acknowledge that we are jointly responsible for the authenticity and integrity of this project.

OMM PRAKASHA MAHARANA(2201329094)
PRIYA SARATHI SAMNTRAY(2201329093)
DIBYANSHU DHAL(2201329091)
ATISHRANJAN SAHU(2201329088)

Date:
Place: Daleiput, Khordha

ACKNOWLEDGMENT

We would like to express our heartfelt gratitude to all those who have helped us directly or indirectly in the successful completion of our project, "SMART LOCKING DOOR SYSTEM"
First and foremost, we would like to thank our Project Guide, Mr. Rabindra Kumar Pradhan, for his constant guidance, valuable suggestions, and encouragement throughout the project. His support and supervision have been instrumental in shaping our work and helping us overcome various challenges.
We would also like to thank our Principal, Prof. Amar Nath Nayak, for extending necessary facilities and encouragement to carry out this project successfully.
Last but not least, we express our gratitude to our families and friends for their continuous motivation, understanding, and moral support throughout the completion of this project.

1. Omm Prakasha Maharana
2. Priya Sarathi Samantray
3. Dibyanshu Dhal
4. Atishranjan Sahu

KMBB
College of Engineering & Technology

CERTIFICATE

Certified that the Report on IoT based Smart lock door system has been successfully designed and presented at KMBB College of Engineering and Technology by us for the fulfillment of the requirement for the 7th Semester in Bachelor degree of technology in Department of ECE of KMBB College of Engineering and Technology, under BPUT, Odisha during academic year 2025-2026. It is certified that all corrections/suggestions indicated for internal assessments have been incorporated in the report . The project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Under Supervision of

Prof. Rabindra Ku. Pradhan

Submitted to

Prof. Rabindra Ku. Pradhan

INDEX

INTRODUCTION

The **ESP8266-based IoT Smart Door Lock System** is a modern security project that combines **embedded electronics** and **IoT technology** to create an intelligent access control solution. The system uses an **ESP8266 Wi-Fi module** as the main controller, a **4x4 membrane matrix keypad** for password entry, and a **solenoid lock** to physically secure the door.

When a **correct password** is entered on the keypad, the ESP8266 activates the relay to **unlock the solenoid lock** for a short duration and then automatically locks it again for safety. The system is also connected to the **Blynk IoT platform**, allowing users to **control and monitor** the door **remotely via a smartphone**. Through the Blynk app, users can lock or unlock the door, check its status, and even receive **real-time notifications** about access events.

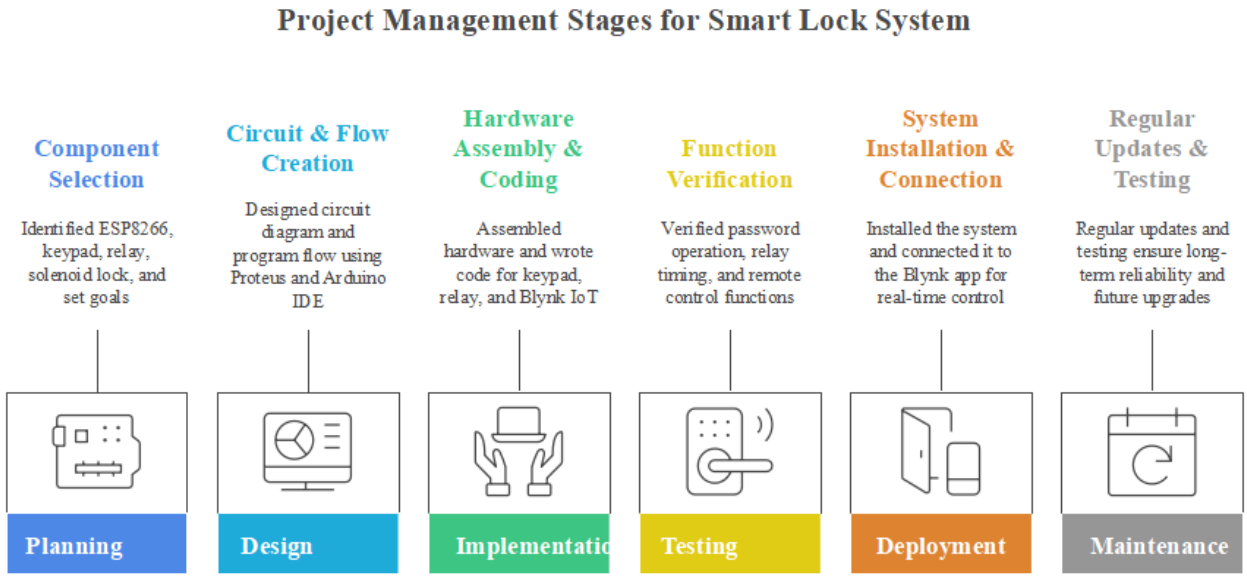
This IoT-based approach eliminates the need for traditional keys and provides a **smart, reliable, and user-friendly** way to manage home or office security. It represents an innovative step toward **smart home automation**, combining **hardware, software, and cloud connectivity** for efficient and secure door control.

PROJECT OBJECTIVES

A. **To develop a smart door lock system** using the **ESP8266 Wi-Fi module** that provides secure and efficient door access control.
B. **To replace traditional key-based locks** with a **password-protected and IoT-enabled system**, reducing the risk of key loss or unauthorized duplication.
C. **To integrate a 4x4 membrane matrix keypad** for **manual password entry**, ensuring local access with internet connectivity.
D. **To control the solenoid lock** automatically through a **relay module**, enabling safe electronic locking and unlocking operations.
E. **To connect the system with the Blynk IoT platform**, allowing users to **remotely lock/unlock the door** and **monitor access status** via smartphone.
F. **To enhance security** by automatically locking the door after a fixed duration and by using password verification.
G. **To design an affordable, user-friendly, and scalable smart lock system** suitable for **homes, offices, or institutional buildings**.

PROJECT MANAGEMENT

The project was managed in several stages to ensure smooth development and implementation.



1. **Planning:** Identified components like ESP8266, keypad, relay, and solenoid lock, and set project goals.
2. **Design:** Created the circuit diagram and program flow using Proteus and Arduino IDE.
3. **Implementation:** Assembled hardware and wrote code for keypad input, relay control, and Blynk IoT connectivity.
4. **Testing:** Verified password operation, relay timing, and remote control functions.
5. **Deployment:** Installed the system and connected it to the Blynk app for real-time control.
6. **Maintenance:** Regular updates and testing ensure long-term reliability and future feature upgrades.

IOT (INTERNET OF THINGS)

IOT as a term has evolved long way as a result of convergence of multiple technologies, machine learning, embedded systems and commodity sensors. IOT is a system of interconnected devices assigned a UIDS, enabling data transfer and control of devices over a network. It reduced the necessity of actual interaction in order to control a device. IOT is an advanced automation and analytics system which exploits networking, sensing, big data, and artificial intelligence technology to deliver complete systems for a product or service. These systems allow greater transparency, control, and performance when applied to any industry or system.

Features of IOT

Intelligence

IOT comes with the combination of algorithms and computation, software & hardware that makes it smart. Ambient intelligence in IOT enhances its capabilities which facilitate the things to respond in an intelligent way to a particular situation and supports them in carrying out specific tasks.

Connectivity

Connectivity empowers the Internet of Things by bringing together everyday objects. Connectivity of these objects is pivotal because simple object level interactions contribute towards collective intelligence in the IOT network. It enables network accessibility and compatibility in the things. With this connectivity, new market opportunities for the Internet of things can be created by the networking of smart things and applications.

Dynamic Nature

The primary activity of Internet of Things is to collect data from its environment, this is achieved with the dynamic changes that take place around the devices. The state of these devices changes dynamically, example sleeping and waking up, connected and/or disconnected as well as the context of devices temperature, location and speed. In addition to the state of the device, the number of devices also changes dynamically with a person, place and time.

Security

IOT devices are naturally vulnerable to security threats. As we gain efficiencies, novel experiences, and other benefits from the IOT, it would be a mistake to forget about security concerns associated with it. There is a high level of transparency and privacy issues with IOT. It is important to secure the endpoints, the networks, and the data that is transferred across all of it means creating a security paradigm.

Advantages of IOT

Communication

IOT encourages the communication between devices, also famously known as Machine-to-Machine (M2M) communication. Because of this, the physical devices are able to stay connected and hence the total transparency is available with lesser inefficiencies and greater quality.

Automation and Control

Due to physical objects getting connected and controlled digitally and centrally with wireless infrastructure, there is a large amount of automation and control in the workings. Without human intervention, the machines are able to communicate with each other leading to faster and timely output.

Information

It is obvious that having more information helps making better decisions. Whether it is mundane decisions as needing to know what to buy at the grocery store or if your company has enough widgets and supplies, knowledge is power and more knowledge is better.

Monitor

The second most obvious advantage of IOT is monitoring. Knowing the exact quantity of supplies or the air quality in your home, can further provide more information that could not have previously been collected easily.

another trip to the store in the near future.

Saves Time

As hinted in the previous examples, the amount of time saved because of IOT could be quite large. And in today's modern life, we all could use more time.

Saving Money

The biggest advantage of IOT is saving money. If the price of the tagging and monitoring equipment is less than the amount of money saved, then the Internet of Things will be very widely adopted

Disadvantages of IOT

Compatibility

Currently, there is no international standard of compatibility for the tagging and monitoring equipment. I believe this disadvantage is the most easy to overcome. The manufacturing companies of these equipment just need to agree to a standard, such as Bluetooth, USB, etc. This is nothing new or innovative needed.

Complexity

As with all complex systems, there are more opportunities of failure. With the Internet of Things, failures could sky rocket. For instance, let's say that both you and your spouse each get a message saying that your milk has expired, and both of you stop at a store on your way home, and you both purchase milk. As a result, you and your spouse have purchased twice the amount that you both need. Or maybe a bug in the software ends up automatically ordering a new ink cartridge for your printer each and every hour for a few days, or at least after each power failure, when you only need a single replacement.

Privacy/Security

With all of this IOT data being transmitted, the risk of losing privacy increases. For instance, how well encrypted will the data be kept and transmitted with? Do you want your neighbors or employers to know what medications that you are taking or your financial situation.

HARDWARE DESCRIPTION FOR SMART LOCK DOOR SYSTEM

- **ESP8266 12E Node MCU**
- **Single channel Realy**
- **12v Adopter**
- **4X4 Matrix keybord**
- **Jumper Wire**
- **Bread Board**

ESP8266 12-E Node MCU

Digital GPIO Pins

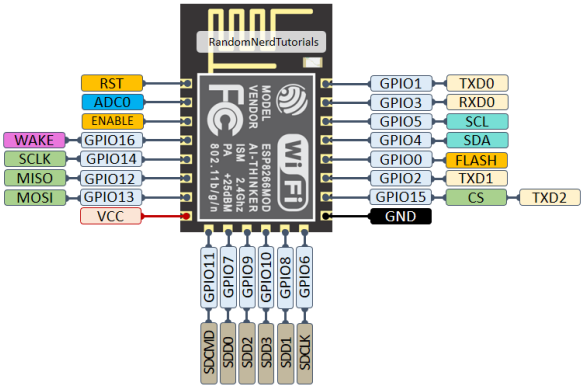
Pin	GPIO No.	Notes / Functions
D0	GPIO16	Can wake from deep sleep; no PWM, no I^2C, no SPI
D1	GPIO5	I^2C SCL, can be used as digital I/O
D2	GPIO4	I^2C SDA, can be used as digital I/O
D3	GPIO0	Boot mode selection; must be HIGH at boot for normal operation
D4	GPIO2	Built-in LED (active LOW), must be HIGH at boot
D5	GPIO14	SPI SCK; digital I/O, PWM supported
D6	GPIO12	SPI MISO; digital I/O, PWM supported
D7	GPIO13	SPI MOSI; digital I/O, PWM supported
D8	GPIO15	SPI CS/SS; must be LOW at boot
RX	GPIO3	UART RX, can be used for serial communication
TX	GPIO1	UART TX, can be used for serial communication

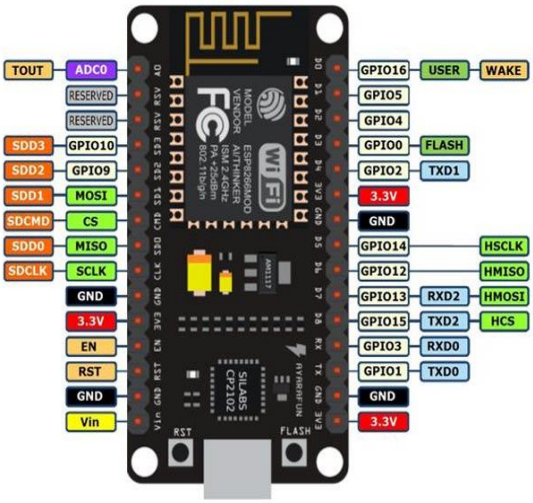
Analog Pin
A0: Analog input pin (0–3.3V). Used to read sensors like LDR, temperature sensors, etc.

Special Pin Notes
Boot Mode Conditions:
GPIO0 → HIGH
GPIO2 → HIGH
GPIO15 → LOW for normal boot.
PWM: Available on most digital GPIO pins except D0.
SPI & I^2C:- SPI: D5 (SCK)
D6 (MISO)
D7 (MOSI)
D8 (CS)- I^2C: D1 (SCL)
D2 (SDA)

Power Pins

3V3: Provides 3.3V regulated output; also used to power the board externally.
VIN: Input voltage (5–10V) to power the board through external source.
GND: Ground reference for the board.
EN / CH_PD: Chip enable pin. Must be HIGH for the ESP8266 to operate.
RST: Reset pin; pulling it LOW restarts the micro-controller.





Parts of Node MCU Development Board

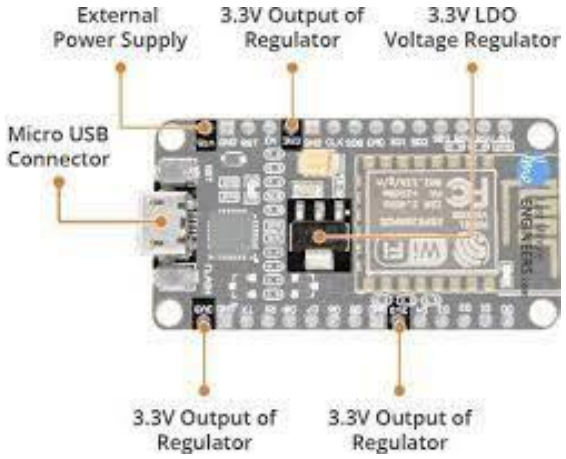
ESP 12-E Module

- The development board equips the ESP-12E module containing ESP8266 chip having Tensilica Xtensa® 32-bit LX106 RISC microprocessor which operates at 80 to 160 MHz adjustable clock frequency and supports RTOS.
- There's also 128 KB RAM and 4MB of Flash memory (for program and data storage) just enough to cope with the large strings that make up web pages, JSON/XML data, and everything we throw at IOT devices nowadays.
- The ESP8266 Integrates 802.11b/g/n HT40 Wi-Fi transceiver, so it can not only connect to a Wi-Fi network and interact with the Internet, but it can also set up a network of its own, allowing other devices to connect directly to it.

- Tensilica Xtensa® 32-bit LX106
- 80 to 160 MHz clock frequency
- 128 kb internal RAM
- 4 MB external flash
- 802.11b/g/n HT40 Wi-Fi transceiver

Power Requirements

As the operating voltage range of ESP8266 is 3V to 3.6V, the board comes with a LDO voltage regulator to keep the voltage steady at 3.3V. It can reliably supply up to 600mA, which should be more than enough when ESP8266 pulls as much as 80mA during RF transmissions. The output of the regulator is also broken out to one of the sides of the board and labelled as 3V3. This pin can be used to supply power to external components.
Power to the ESP8266 Node MCU is supplied via the on-board Micro B USB connector. Alternatively, if you have a regulated 5V voltage source, the VIN pin can be used to directly supply the ESP8266 and its peripherals.



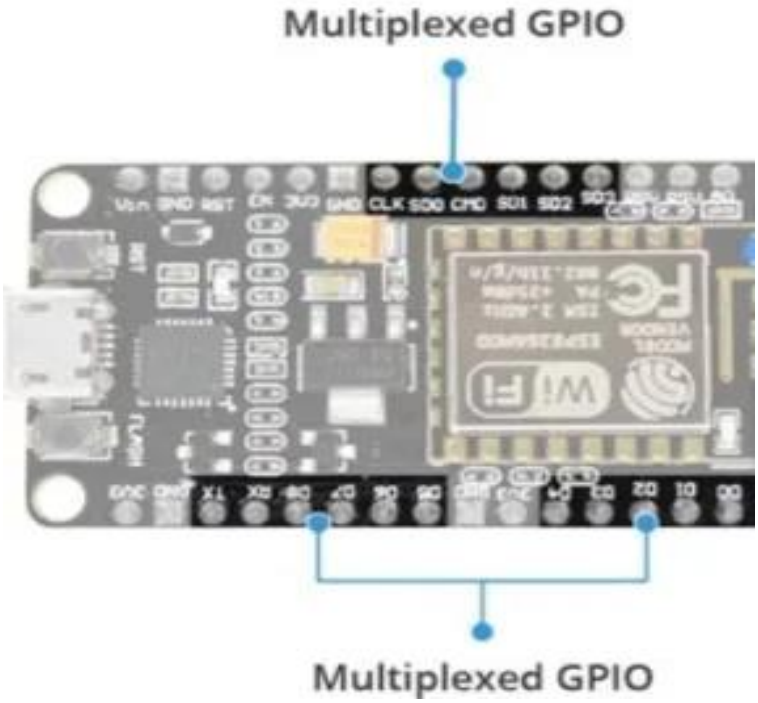
- Operating voltage 2.5V to 3.6V
- On-board 3.6V 600mA regulator
- 80 mA operating current
- 20 μA during sleep mode

Peripheral I/O

The ESP8266 Node MCU has total 17 GPIO pins broken out to the pin headers on both sides of the development board. These pins can be assigned to all sorts of peripheral duties, including:

- ADC channel – A 10-bit ADC channel.
- UART interface – UART interface is used to load code serially.
- PWM outputs – PWM pins for dimming LED s or controlling motors.
- SPI, I2C & I2S interface – SPI and I2C interface to hook up all sorts of sensors and peripherals.
- I2S interface – I2S interface if you want to add sound to your project.

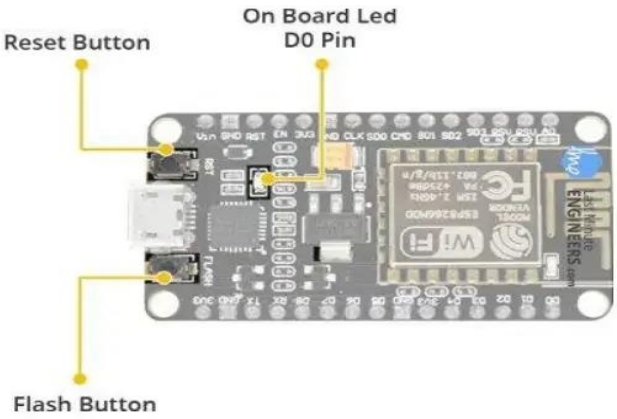
As a result of the pin multiplexing feature (Multiple peripherals multiplexed on a single GPIO pin), a single GPIO pin can act as PWM/UART/SPI.



GPIO pins on Node MCU development board

On Board Switches and LED Indicators

The ESP8266 Node MCU features two buttons. One marked as RST located on the top left corner is the Reset button, used of course to reset the ESP8266 chip. The other FLASH button on the bottom left corner is the download button used while upgrading firmware. The board also has a LED indicator which is user programmable and is connected to the D0 pin of the board.

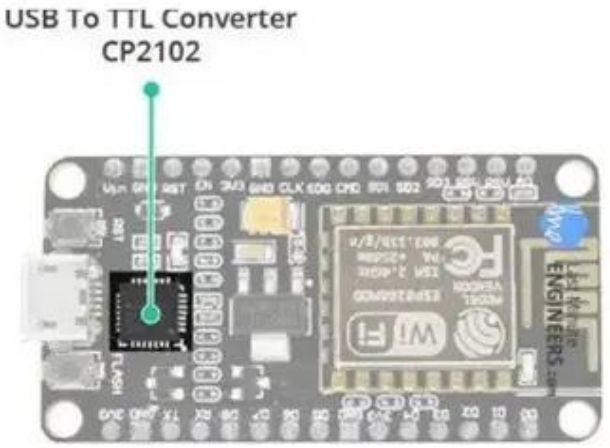


- RST: Reset the ESP8266 chip
- FLASH: Download new programs
- Blue LED: User programmable

ON board switches and LED indicators on Node MCU development board.

Serial Communication

The board includes CP2102 USB-to-UART Bridge Controller from Silicon Labs, which converts USB signal to serial and allows your computer to program and communicate with the ESP8266 chip.



- CP2120 USB-to-UART converter
- 4.5 Mbps communication speed
- Flow control support

CP2120 on Node MCU development board.

Driver installation for hardware interfacing

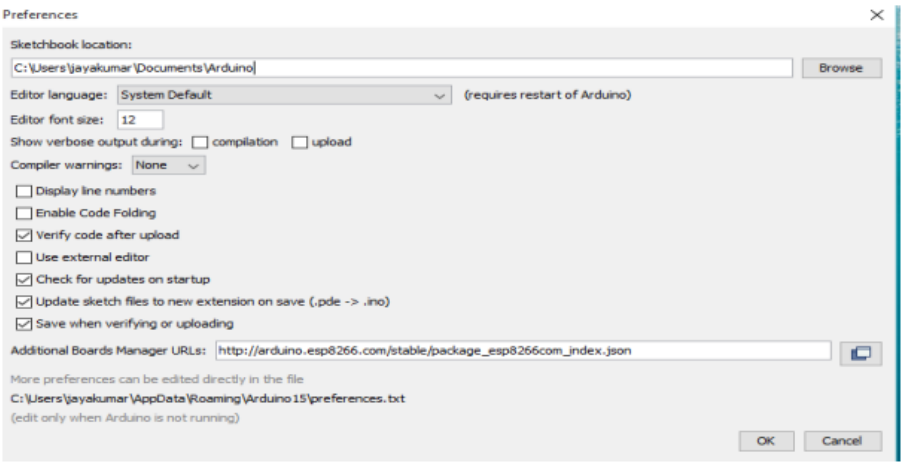
Mostly these days devices download and install drivers on their own, automatically. Windows doesn't know how to talk to the USB driver on the Node MCU so it can't figure out that the board is a Node MCU and proceed normally.

1. Node MCU Amica is an ESP8266 Wi-Fi module based development board. It has got Micro USB slot that can directly be connected to the computer or other USB host devices. Ti has got 15X2 header pins and a Micro USB slot, the headers can be mounted on a breadboard and Micro USB slot is to establish connection to USB host device. It has CP2120 USB to serial converter. In order to install CP2120 (USB to serial converter), user is needed to download the driver for the same.
2. Once user downloads drivers as per its respective operating system, the system establishes connection to Node MCU.
3. The user needs to node down the COM post allotted to newly connected USB device (Node MCU) from device manager of the system. This com port number will be required while using Node MCU Amica.

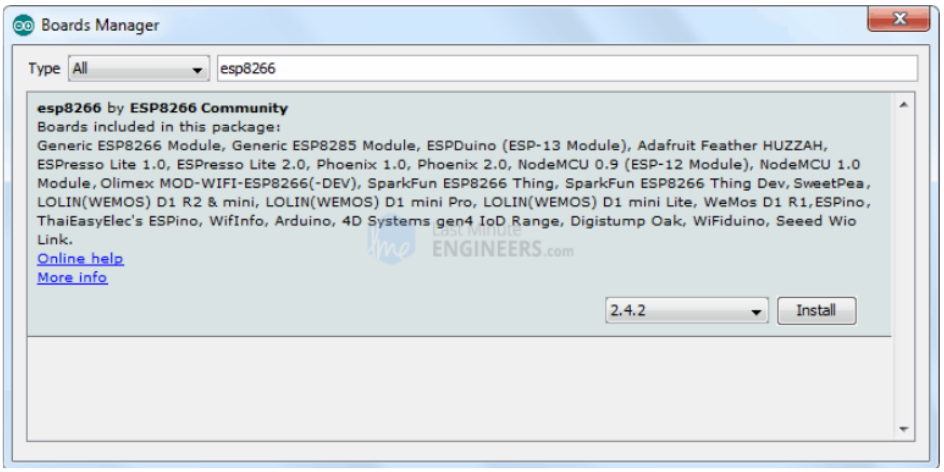
Interfacing Node MCU with Arduino IDE

To begin with the latest Arduino IDE version, we'll need to update the board manager with a custom URL. Open up Arduino IDE and go to File > Preferences. Then, copy below URL into the Additional Board Manager URLs text box situated on the bottom of the window:

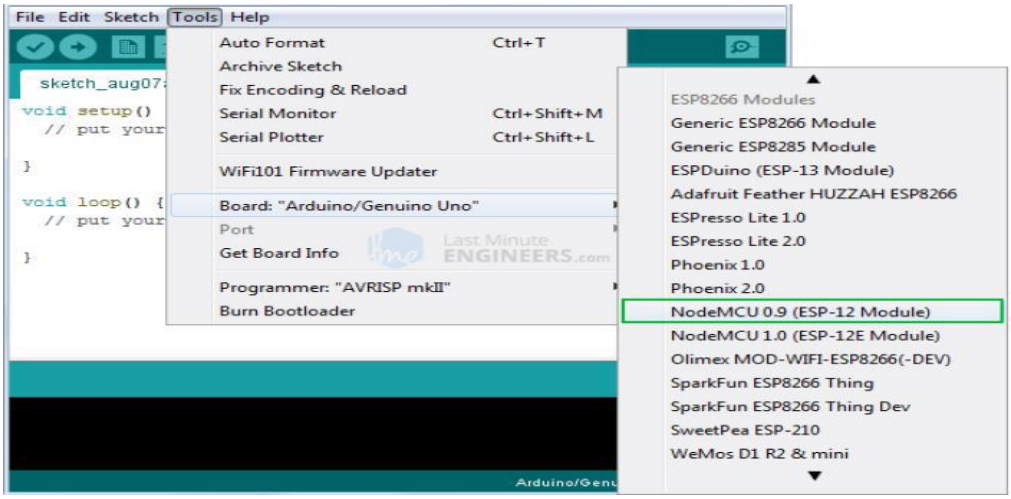
http://arduino.esp8266.com/stable/package_esp8266com_index.json



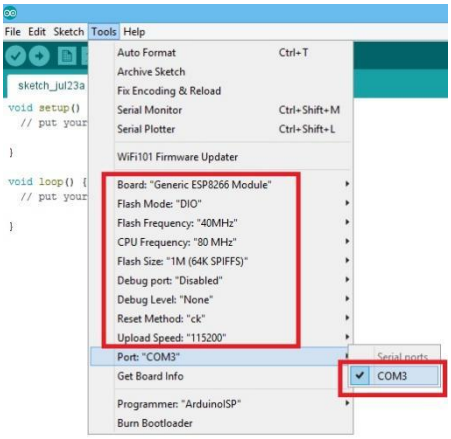
OK. Then navigate to the Board Manager by going to Tools > Boards > Boards Manager. There should be a couple new entries in addition to the standard Arduino boards. Filter your search by typing esp8266. Click on that entry and select Install.



Before we get to uploading sketch & playing with LED, we need to make sure that the board is selected properly in Arduino IDE. Open Arduino IDE and select Node MCU 0.9 (ESP-12 Module) option under your Arduino IDE > Tools > Board menu.



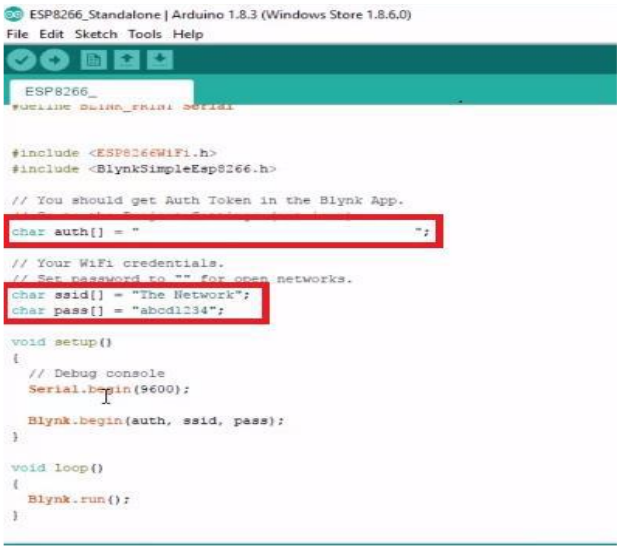
Now, plug your ESP8266 NodeMCU into your computer via micro-B USB cable. Once the board is plugged in, it should be assigned a unique COM port. On Windows machines, this will be something like COM#, and on Mac/Linux computers it will come in the form of /dev/tty.usbserial-XXXXXX. Select this serial port under the Arduino IDE > Tools > Port menu. Also select the Upload Speed: 115200



Uploading code to Node MCU

- NodeMCU is connected to PC using a USB cable.
- Now, we'll set up the Arduino IDE by changing some settings. So, open up the Arduino IDE. Select Tools > Board and select 'NodeMCU 1.0 (ESP-12E Module)' as the board. And that's all the settings we need to change. So now we begin writing the code.
- Select Files > Examples > Blynk > Boards_WIFI > ESP8266_Standalone. A new file with some prewritten code opens. The following changes to the code are made.

1. The line which says 'char auth[] = "YourAuthToken", replace YourAuthToken part with your Blynk's authentication token that was generated by the Blynk server.
2. The line which says *char ssid[] = "YourNetworkName"*, replace *YourNetworkName* part with the name of Wi-Fi network that the Node MCU must connect to.
3. The line where it says *char pass[] = "YourPassword*" and replace the *YourPassword* part with the password of the Wi-Fi network.



The code is ready to be uploaded to the hardware. On clicking upload button, the code is uploaded to Node MCU and the next time it's powered on, it automatically connects to the assigned Wi-Fi network.

Solenoid Lock

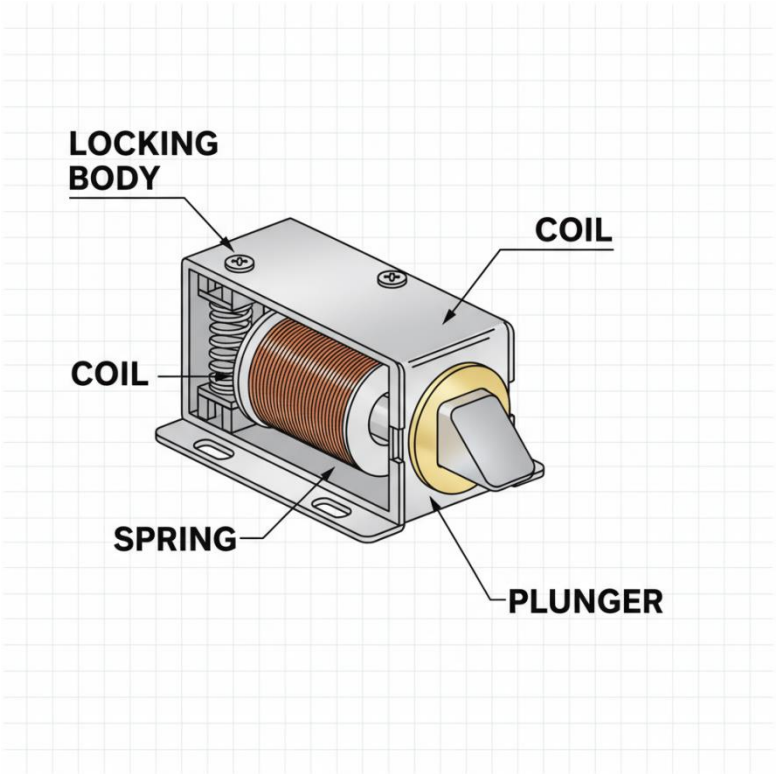
A **solenoid lock** is an **electromechanical locking device** that uses a **solenoid (coil of wire)** to control the locking mechanism. It operates on the principle of **electromagnetism**, converting electrical energy into linear motion.

The **main parts** of a solenoid lock include:

a. **Solenoid Coil:** A copper wire coil that generates a magnetic field when current flows through it.
b. **Metal Plunger (Rod):** A movable iron or steel rod that moves in or out when the solenoid is energized.
c. **Spring Mechanism:** Returns the plunger to its original position when power is removed.
d. **Locking Body:** The mechanical casing that holds the components and interfaces with the door latch.

Working Principle:
When an electric current passes through the solenoid coil, it creates a **magnetic field** that pulls the **metal plunger** inside the coil. This movement either **releases or locks** the door latch depending on the design (normally locked or normally unlocked). When the power is cut off, the **spring pushes the plunger back**, returning the lock to its default state.

In the **Smart Solenoid Lock System**, the solenoid is powered through a **relay module** controlled by the **ESP8266 Node MCU**. When the correct password is entered or a Blynk command is received, the relay energizes the solenoid, **unlocking the door** for a set time, and then automatically **locks** it again.



Single-Channel Relay

A single-channel relay is an electromechanical or solid-state switching device that allows a low-power control signal from a microcontroller (like ESP8266 or Arduino) to control a high-power load such as a lamp, motor, or solenoid lock.
It has one control channel, meaning it can switch one device on or off. The module typically has:
- VCC: Power supply (usually 5V)
- GND: Ground
- IN: Control signal from the microcontroller
- COM, NO, NC: Output terminals for connecting the load
  - COM → Normally Open (NO): Closed when relay is activated
  - COM → Normally Closed (NC): Closed when relay is inactive

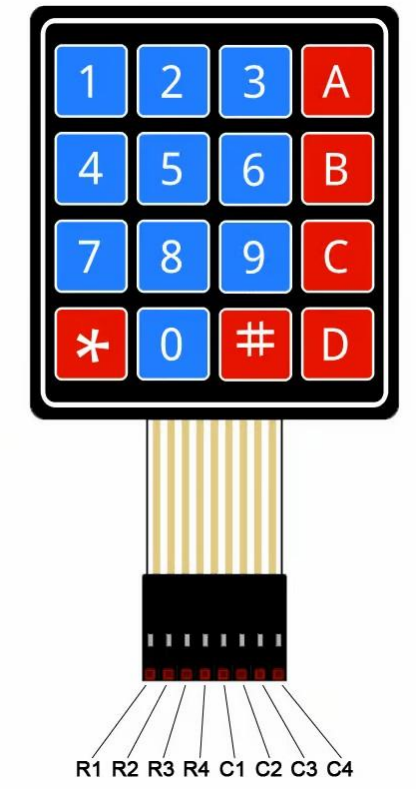
Single-channel relays are widely used in home automation, IoT projects, and safety control systems to safely isolate the microcontroller from high-voltage circuits.

12V adapter

A **12V adapter** is an **electrical device** that converts **AC mains voltage** into a **stable 12-volt DC output**, commonly used to power electronic devices such as routers, CCTV cameras, LED strips, small appliances, and microcontroller projects, providing **regulated voltage** to ensure safe and reliable operation, protecting connected devices from overvoltage, short circuits, and current fluctuations, enabling efficient energy transfer, supporting continuous use, and serving as a convenient and essential power supply solution for low- to medium-power electronic applications.

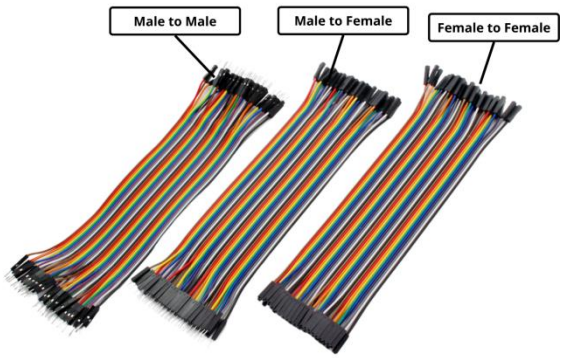
4x4 matrix keyboard

A **4x4 matrix keyboard** is a **compact electronic input device** consisting of sixteen keys arranged in four rows and four columns, which allows users to enter numerical, alphabetical, or special function commands efficiently, and it is widely used in embedded systems, microcontroller projects, security systems, digital locks, calculators, and other electronic applications because it reduces the number of input pins required, supports scanning techniques to detect key presses accurately, and enables convenient, reliable, and cost-effective user interaction.



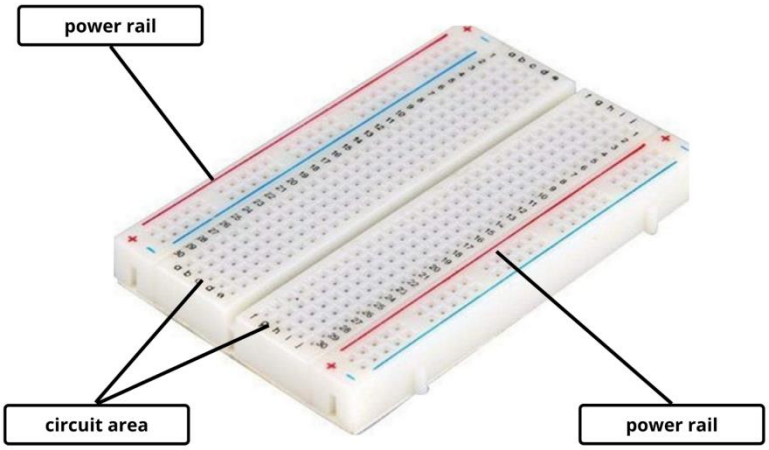
jumper wire

A **jumper wire** is a **short, flexible electrical wire** with connectors or pins at each end, used to **make temporary or permanent connections** between components on a breadboard, PCB, or other electronic circuits, enabling easy prototyping, testing, and debugging without soldering, supporting signal, voltage, or ground connections, available in male-to-male, male-to-female, or female-to-female types, widely used in microcontroller projects, robotics, IoT devices, and electronics experiments, allowing quick circuit modifications, reducing assembly time, and enhancing convenience and flexibility for electronics enthusiasts and engineers.



breadboard

A **breadboard** is a **solderless electronic prototyping board** with a grid of interconnected holes that allows engineers, students, and hobbyists to **easily build, test, and modify circuits** without permanent connections, supporting components like resistors, LEDs, ICs, and jumper wires, enabling rapid experimentation, troubleshooting, and learning in electronics and embedded systems projects, widely used in microcontroller interfacing, IoT devices, robotics, and circuit design, providing a reusable, convenient, and safe platform for assembling circuits while ensuring reliable connections and flexible layout adjustments



SOFTWARE DESCRIPTION FOR SMART LOCK DOOR SYSTEM

- **ARDINO IDE**
- **BLYNK IOT**

Arduino IDE –
The Arduino IDE (Integrated Development Environment) is a software platform used to write, compile, and upload programs (sketches) to microcontroller boards like Arduino, ESP8266, or ESP32. It supports C/C++ language and provides a simple interface for beginners to develop and test embedded systems or IoT projects.

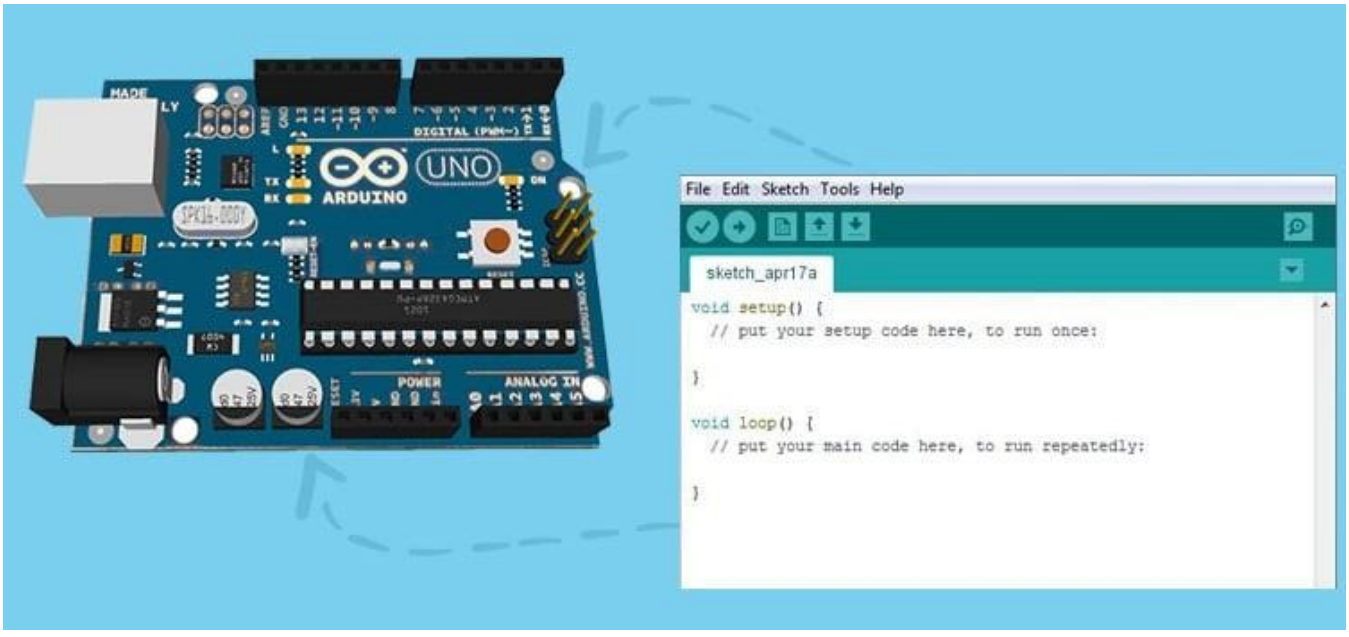
Installation Steps

1. Download the Arduino IDE: Go to the official site (https://www.arduino.cc/en/software) and choose the correct version for your operating system.
2. Install the Software: Run the downloaded setup file and follow the installation instructions. Allow drivers to install.
3. Launch the IDE: Open the Arduino IDE from your desktop or start menu. You'll see the coding area, toolbar, and serial monitor options.

Setup for ESP8266 (Node MCU)

4. Open File → Preferences.
5. In 'Additional Boards Manager URLs', paste: https://arduino.esp8266.com/stable/package_esp8266com_index.json
6. Go to Tools → Board → Boards Manager, search for ESP8266, and click Install.
7. Select your board: Tools → Board → NodeMCU 1.0 (ESP-12E Module).
8. Connect the NodeMCU using a USB cable.
9. Choose the correct Port under Tools → Port.

Click Upload to send your program to the board

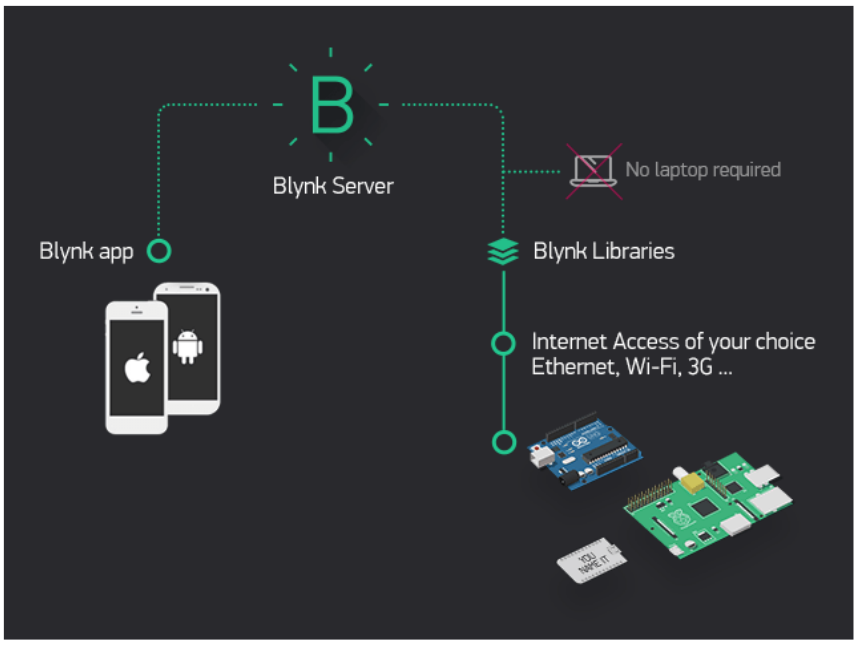


BLYNK APPLICATION

The Blynk application was designed for the primary purpose of Internet of Things. **Blynk** is a platform with IOS and Android apps to control Arduino, Raspberry Pi and the likes over the Internet. It's a digital dashboard where graphic interface for a prototype can be built by simply dragging and dropping widgets. It can control hardware remotely, it can display sensor data, can store and visualize data and possessed a lot more functionality. There are three major components in the platform:

- **Blynk Application:** allows to you create amazing interfaces for your projects using various widgets we provide.
- **Blynk Server:** responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally. It's an open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
- **Blynk Libraries:** for all the popular hardware platforms – enable communication with the server and process all the incoming and outgoing commands.

Every time a radio button is accessed in the Blynk application, the message travels to the Blynk Cloud, where it finds the specific hardware by the unique generated authentication token. It works in the same way for the opposite direction.



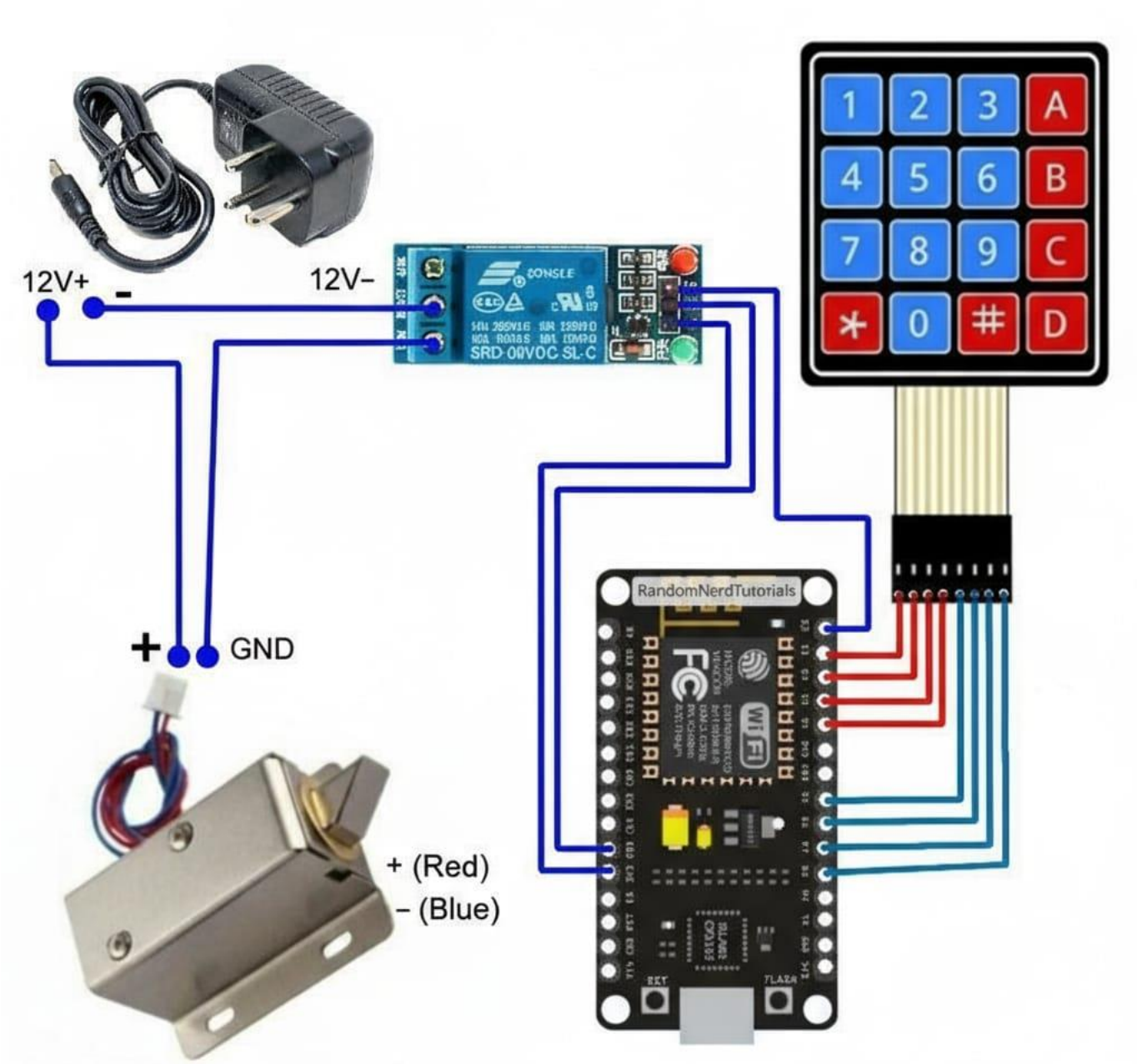
SMART LOCK DOOR SYSTEM CODE

```
// __define-ocg__ SMART SOLENOID LOCK with ESP8266 + Keypad + Blynk + Serial
// Author: Omm Prakasha

#define BLYNK_TEMPLATE_ID "TMPL3ZcVPHmr5"
#define BLYNK_TEMPLATE_NAME "SMART DOOR LOCK"
#define BLYNK_AUTH_TOKEN "qZeUXKcePv5DRvfiFnNV1K07kKuJepUF"

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


// ---------- WiFi Credentials ----------
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "POCO X6 Neo 5G";       //��Enter your WiFi name
char pass[] = "8895847315";    //��Enter your WiFi password

// ---------- Pin Configuration ----------
#define RELAY D0      // Relay module connected to D1
#define VPIN_SWITCH V1 // Blynk switch virtual pin
#define VPIN_LED V2    // Blynk LED virtual pin

// ---------- Relay Auto-Off Timer ----------
#define RELAY_ON_TIME 5000  // 5 seconds (5000 ms)

// ---------- Password Settings ----------
String correctPassword = "7890";  // You can change your password here
String enteredPassword = "";

// ---------- Keypad Setup ----------
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {D1, D2, D3, D4};
byte colPins[COLS] = {D5, D6, D7, D8};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// ---------- Variables ----------
bool relayState = false;
unsigned long relayStartTime = 0;

// ---------- Relay Control Functions ----------
void activateRelay() {
  digitalWrite(RELAY, HIGH);
```

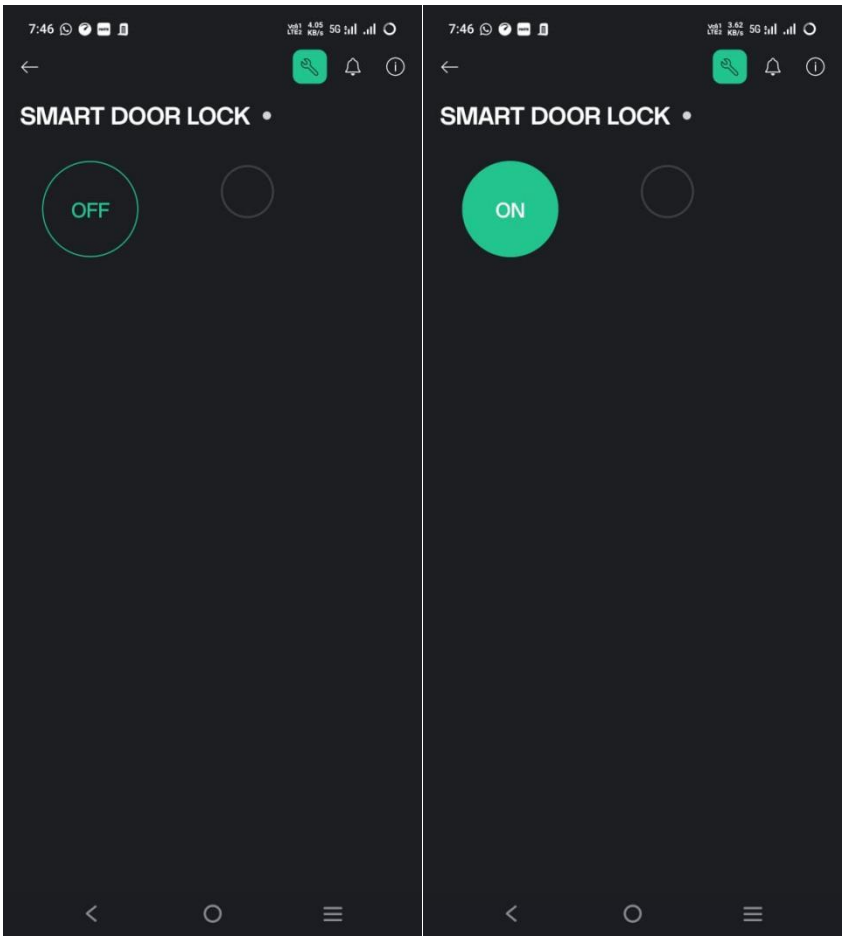
```
  Blynk.virtualWrite(VPIN_LED, 255);
  Blynk.virtualWrite(VPIN_SWITCH, 1);
  Serial.println("🔓 Solenoid Lock: UNLOCKED (Relay ON)");
  relayState = true;
  relayStartTime = millis();
}
void deactivateRelay() {
  digitalWrite(RELAY, LOW);
  Blynk.virtualWrite(VPIN_LED, 0);
  Blynk.virtualWrite(VPIN_SWITCH, 0);
  Serial.println("🔒 Solenoid Lock: LOCKED (Relay OFF)");
  relayState = false;
}

// ---------- Blynk App Control ----------
BLYNK_WRITE(VPIN_SWITCH) {
  int value = param.asInt();
  if (value == 1) activateRelay();
  else deactivateRelay();
}

// ---------- Setup ----------
void setup() {
  Serial.begin(9600);
  pinMode(RELAY, OUTPUT);
  digitalWrite(RELAY, LOW);

  Blynk.begin(auth, ssid, pass);
  Serial.println("\nSMART SOLENOID LOCK SYSTEM READY");
  Serial.println("Control by: Keypad | Serial Monitor | Blynk App");
  Serial.println("Serial Commands: Type ON or OFF");
}

// ---------- Main Loop ----------
void loop() {
  Blynk.run();

  // ---- Serial Control ----
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n');
    input.trim();
    input.toUpperCase();
    if (input == "ON") {
      activateRelay();
    } else if (input == "OFF") {
      deactivateRelay();
    }
  }

  // ---- Keypad Control ----
  char key = keypad.getKey();
  if (key) {
```

```
  Serial.print("Key Pressed: ");
  Serial.println(key);
  if (key == '#') { // Enter key
   if (enteredPassword == correctPassword) {
    Serial.println("✅Correct Password — Unlocking...");
    activateRelay();
   } else {
    Serial.println("❌Wrong Password — Access Denied");
   }
   enteredPassword = ""; // Reset password
  }
  else if (key == '*') { // Clear input
   enteredPassword = "";
   Serial.println("🔄Password Cleared");
  }
  else {
   enteredPassword += key;
   Serial.print("Entered: ");
   Serial.println(enteredPassword);
  }
 }

 // ---- Auto-Off after 5 seconds ----
 if (relayState && (millis() - relayStartTime >= RELAY_ON_TIME)) {
  deactivateRelay();
 }
}
```

Circuit diagram



Blynk IoT setup



RESULT

When the system is powered on, the **ESP8266 NodeMCU** initializes and connects to the specified **Wi-Fi network** and the **Blynk server**, making the smart lock ready for operation. The system can then be controlled through **three modes**:

Keypad Input: When a user enters the correct 4-digit password on the **4x4 keypad**, the **relay activates**, energizing the **solenoid lock**, and the door unlocks. If the wrong password is entered, the system denies access and displays a message on the serial monitor. The entered password can also be cleared using the * key.

Serial Monitor Commands: The lock can be operated using the serial monitor by typing **ON** to unlock or **OFF** to lock. The serial monitor also displays status messages for lock activation, deactivation, and password entry feedback.

Blynk App Control: A **virtual switch** on the Blynk app allows remote unlocking/locking, while a **virtual LED** indicates the current status of the solenoid lock in real-time.

After activation, the system automatically **locks the door after 5 seconds**, providing secure auto-lock functionality. All interactions are **synchronized**, so changes via keypad, serial monitor, or Blynk are reflected immediately across all interfaces.

The final result is a **fully functional smart solenoid lock system** that is secure, automated, and controllable both locally and remotely, suitable for **home automation and IoT-based security applications**.

LIMITATION

1. Wi-Fi Dependency: Remote control via Blynk will not work if the Wi-Fi network is unstable or disconnected.
2. Power Supply Dependence: A power outage can leave the lock unsecured unless a backup source is available.
3. Mechanical Wear: The relay and solenoid mechanism may wear out over time, reducing durability.
4. Password Vulnerability: Keypad passwords can be observed (shoulder surfing), compromising security.
5. · · Limited GPIO Pins: The ESP8266 has a limited number of pins, restricting additional devices or sensors.
6. Security Risks: Lack of encryption or proper network security may expose the system to hacking.
7. Short Range: Remote access is limited to the Wi-Fi network range without cloud or internet integratio

CONCLUSION

The **Smart Solenoid Lock System** demonstrates a compact, reliable, and user-friendly approach to modern security. By integrating an **ESP8266 NodeMCU**, a **4x4 keypad**, a **solenoid lock**, and the **Blynk IoT platform**, the system provides multiple control options—local via keypad, remote via mobile app, and serial commands. Its **auto-lock feature**, real-time status feedback, and Wi-Fi connectivity make it suitable for **home automation and smart security applications**. Overall, it effectively combines **automation, convenience, and security** for modern IoT environments.

FURTHER ENHANCEMENT AND FUTURE SCOPE

The **Smart Solenoid Lock System** can be enhanced for better security and convenience. Future improvements include **biometric authentication** such as fingerprints or facial recognition, **cloud-based control** for remote access, and **integration with smart home systems** like Google Home or Alexa. Features like **mobile notifications, entry logs, and multi-door control** can be added. Implementing **low-power modes** and **battery backup** ensures operation during outages. These enhancements make the system more **secure, intelligent, energy-efficient, and suitable for modern IoT smart environments**.

References

A. Espressif Systems, **"ESP8266EX Datasheet"**, Espressif, 2021. https://www.espressif.com/en/products/socs/esp8266
B. Arduino Official Documentation, **"Arduino IDE Guide"**, Arduino.cc. https://www.arduino.cc/en/Guide
C. Blynk IoT Platform, **"Blynk Documentation"**, Blynk Inc. https://docs.blynk.io/en/
D. Keypad Library for Arduino, **"Keypad Library Documentation"**, Arduino Libraries. https://playground.arduino.cc/Code/Keypad/
E. Relay Module Guide, **"Relay Module Basics"**, Electronics Hub. https://www.electronicshub.org/relay-module/
F. IoT and Smart Lock Concepts, **"Internet of Things for Smart Homes"**, ResearchGate. https://www.researchgate.net/publication/327813818

1.