

Smart Bank locker using with Fingerprint scanning based on IOT

Submitted in Partial fulfilment of the Requirements for the Award of
Bachelor of Engineering in Electronics and Communication Engineering.

By

B. NANDA KUMAR (38130036)

B. MOKSHAGNA (38130035)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

May 2022



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING
BONAFIDE CERTIFICATE**

This is to certify that Project Report is the bonafide work of B. Nanda Kumar, B. Mokshagna who carried out the Project entitled **“Smart Bank Locker using with Fingerprint Scanning based on IoT”**. Under the guidance of **Mrs. F.V. Jaya Sudha**, during the period JAN 2022 to MARCH 2022.

INTERNAL GUIDE

Mrs.F. V Jaya Sudha

HEAD OF THE DEPARTMENT

Dr.T. RAVI, M.E., Ph.D.,

Submitted for Viva Voce held on

Internal Examiner

External Examiner

DECLARATION

We (B. Nandakumar and B. Mokshagna) Hereby declare that the Project report entitled **Smart Bank Locker using with Fingerprint Scanning based on IOT** is done under the guidance of **Mrs.F.V. Jaya Sudha** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in **ELECTRONICS AND COMMUNICATION ENGINEERING.**

DATE:

PLACE: CHENNAI

Signature 1

Signature 2

ACKNOWLEDGEMENT

I pleased to acknowledge my sincere thanks to board management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY for their kind encouragement in doing this internship and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. N.M. NANDHITHA, M.E., Ph.D.**, Dean, school of Electrical and Electronics engineering and **Dr. T. RAVI, M.E., Ph.D.**, Head of the Department, Electronics and Communication engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs.F.V. Jaya Sudha** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my Project.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

- This paper will focus on effective recognizing and controlling system for Bank locker room which is fully self-determining. In cases of robberies, its commonly happen that the banned entrance in the locker room area which can be detected by our security system. If the robbery take place the banks are not be capable to recognize the robber due to absence of the proof by using the current human operated security system.
- Development of various sensors has enabled systems to have preventive and corrective measures in this regard significantly. In order to deliver a concrete security solution for critically important and confidential documents and goods, we proposed an Automated Safety Vault with Double Layered Defense Mechanism.
- The solution comprised of an Electronic Lock driven by password verification and a Biometric authentication for users using a Fingerprint scanning and sensing tool. Both of these two layers ensured the authenticity of the user by preventing any unauthorized access to the Vault.
- The system was then implemented in a prototype scope for testing and validation of the proposals. The implemented system and testing data showed that the Automated Safety Vault with all its security features had successful operation. The specification of the whole system as well as the results is observed and verified.

Contents

<u>Topic</u>	<u>Page-Numbers</u>
INTRODUCTION	6
LITERATURE SURVEY	11
EXISTING SYSTEM	13
PROPOSED SYSTEM	14
MATERIALS AND METHOD USED	15
STEPS FOR ASSEMBLING	34
CODE	42
CONCLUSION	54
APPLICATIONS	55
FUTURE SCOPE	55
REFERENCES	55

INTRODUCTION:

Internet of Things (IoT):

The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

Internet of Things (IoT) is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment. In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established.

Over 9 billion 'Things' (physical objects) are currently connected to the Internet, as of now. In the near future, this number is expected to rise to a whopping 20 billion.

There are two ways of building IoT:

1. Form a separate internetwork including only physical objects.
2. Make the Internet ever more expansive, but this requires hard-core technologies such as rigorous cloud computing and rapid big data storage (expensive).

Characteristics of IoT:

- Massively scalable and efficient
- IP-based addressing will no longer be suitable in the upcoming future.
- An abundance of physical objects is present that do not use IP, so IoT is made possible.
- Devices typically consume less power. When not in use, they should be automatically programmed to sleep.
- A device that is connected to another device right now may not be connected in another instant of time.
- Intermittent connectivity – IoT devices aren't always connected. In order to save bandwidth and battery consumption, devices will be powered off periodically when not in use. Otherwise, connections might turn unreliable and thus prove to be inefficient.

Needs for setting up IoT environment for basic applications

1. *Choosing a platform for IoT development*, which provides a powerful toolkit for IoT development and end-to-end management that connects devices, smart sensors, and IoT gateways to the cloud.

- **AWS IoT:** (Amazon Web Services)
 0. A cloud platform designed for IoT apps with the facility of assuring millions of device connectivity and acts as a data sea.
 1. It supports all SDKs like *Embedded C, Python, and Java, etc.*
- **Microsoft Azure IoT:**

0. Azure cloud platform uses Microsoft Visual Studio SDK.
1. Collect and analyse real-time device data using a pre-configured remote monitoring system.

2. Choosing IoT hardware processor:

(a) Arduino

1. Open-source electronics prototyping platform
2. The simplest and the beginner's choice.
3. To create interactively (IoT) electronic applications
4. It is the first microcontroller-based development board
5. Easy to program for beginners by Arduino IDE

Set up – procedure:

1. It itself has **0.5KB of the boot loader** that makes the program be burned into the circuit.
2. All we have to play with Arduino is to download the Arduino software and start the code.
3. The Arduino programs are called sketches
4. **Basic Arduino language: C/C++**

Advantages:

1. Inexpensive
2. Cross-platform/Multiplatform
3. Flexible and easy prototyping
4. Provides pre-wiring and free code libraries.

Introduction

- The Bank which is used to place that indicates the very high-level security. In day-to-day life every person is involved in banking transaction. Because of high level security, we use bank lockers to secure our important documents, expensive jeweler, or cash etc., Hence it has become a very important part for every common human being. To suffer in this world and for a continuous development; the banking sector needs to accommodate a very hinge rise security. As we know new branches are opening by considering the public interest. Hence more security for every sector is required. Because of development current system and services becomes autonomous and banking service is not so far from that. Various researches show that there is accountability in devices and technologies in security system.
- In automatic security systems generally passwords, identification cards and PIN verification techniques are being used but the disadvantage is that the passwords could be hacked and a card may be stolen or lost. The most secured system is fingerprint recognition because a fingerprint of one person never matches the other. Biometrics studies commonly include fingerprint, face, iris, voice, signature, and hand geometry recognition and

verification. Many other modalities are in various stages of development and assessment.

- Among these available biometric traits fingerprint proves to be one of the best traits providing good mismatch ratio, high accurate in terms of security and also reliable. That's why we are motivated to do this paper. Therefore, the study shows all the approaches intent to solve the dire problem of security to critical systems of an institution with proper authorization. Such systems are only accessible by the designated users and not by the mass. Moreover, the solution must ensure the obstruction of all possible ways of violation of security within the periphery of the secured area. It is also expected to regulate the access to certain users divided in different using capacity groups.

LITERATURE SURVEY:

Subhash H. Jadhav, S. S. Agrawal worked on Smart Bank Locker Security System Using Biometric Fingerprint and GSM Technology. The main goal of this project is to design and implement a highly secured and reliable smart bank locker security system based on RFID, Biometric fingerprint, password and GSM technology. This can be organized in bank, offices (treasury), schools and homes. In this system only the authentic person can open the lock and collect the important documents, jewelry or money from the lockers. In this security system RFID, biometric fingerprint, password and GSM technology systems are used. In our proposed system first, the user will enroll his user's name, password and his mobile number, then the person will put finger on finger print module and finger print will be scanned and stored with fingerprint id.

Arvasu Chikara, Pallavi Choudekar, Ruchira, Divya Asija worked on Smart Bank Locker Using Fingerprint Scanning and Image Processing. The main feature of this work is it keeps track of time, date and number of accesses of locker by a user in the bank. The smart lock program will compare your image and fingerprint with the data already stored in the database. After checking the authenticity of the user, the microcontroller (Arduino) will give signal to the lock and it will open. It also gives a message when

the number of permissible access turns increases in a given duration.

Lokesh M. Giripunje, Suchita Sudke, Pradnya Wadkar, Krishna Ambure worked on IOT BASED SMART LOCKER SECURITY SYSTEM. This project will focus on effective recognizing and controlling system for Bank locker room which is fully self-determining. In cases of robberies, its commonly happen that the banned entrance in the locker room area which can be detected by our security system. If the robbery take place the banks are not be capable to recognize the robber due to absence of the proof by using the current human operated security system. The system will design in effective way by recognizing and controlling illegal person to access the locker for the safety of bank locker room. In this, we proposed a three-phase conformation of procedure for smart locker, by providing User Name and Password, using Fingerprints and OTP which check out the user.

J. Thirumalai, Gokul. R, Gunasekaran. P, Manellore Murali. M, Jackson Jubilance Joseph. L worked on An IoT based Bank Locker Security System. Our proposed provides a highly secure, valid and easy to operate for both the customers who has a locker in a Bank and the head of the branch who responsible for all the

operations connected to the safety lockers. Our work does verification process by user fingerprint, legitimate login credentials for authorized user. The key objective of the proposed solution is to obtain the result of collective decision making based on the data provided by multiple sensors. To attain this performance, a controller is employed to analyze the sensor data and to generate desired controlled variables in order to ensure the security of the vault.

EXISTING SYSTEM:

- As in our Local and City banks they are using manual key-based locker systems. In this system two keys are given for bank locker as one key is with the bank manager and another key with the account holder. So, there is no level of high security if the key is lost or someone come to bank with the locker key they will allow.
- Some banks are using with RFID tags. In this system along with RFID tag, heat sensor and GSM is used. Initially locker holder has to swipe his RFID tag which consists of information about bank locker holder like locker number and other details.
- Once the tag is recognized as valid tag, bank manager will provide respective locker through conveyer setup. Apart

from this if somebody tries to open the locker through machine or any instrument, then that theft is detected by heat sensor.

- As burglar will use some sort of instrument then temperature will be increase, and this rise in temperature will be sensed and an alarm will trigger.

Drawbacks:

- Key is lost or stolen.
- RFID Tags can be easily hack by the hackers.

PROPOSED SYSTEM:

- In this proposed, the design and implementation of a prototype of an automated vault door locking system is presented which warrants double layer of security. It ensures the proper user of the vault by securing the door with numeric password and biometric authentication.
- It monitors the conditions of operation of the vault from both the inside and the outside by employing several sensors which are continuously feeding information to the controller of the proposed system to confirm the robustness in terms of rightful access and security of the contents within the vault.

- The entire system can be easily managed with all the status updates being reeled by the controller to the administrators eliminating the unforced reasons of human errors. The future enhancement to this work could be done by adding some more aspects.
- Therefore, it improved the reliability of bank locker and unauthorized access will be minimized. The enhancement could be further applied to identify the illegal entrance.

MATERIALS AND METHOD USED:

- **NodeMCU**
- **Finger print module**
- **Arduino Nano**
- **Motor**
- **Keyboard**

NodeMCU:

- NodeMCU is an open-source Lua based firmware and development board specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266

Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

- The NodeMCU ESP8266 development board comes with the ESP-12E module containing ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.
- NodeMCU can be powered using Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.

History:

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. NodeMCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the gerber file of an ESP8266 board, named devkit v0.9. Later that month, Tuan PM ported MQTT client library

from Contiki to the ESP8266 SoC platform, and committed to NodeMCU project, then NodeMCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to the NodeMCU project, enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays. In the summer of 2015, the original creators abandoned the



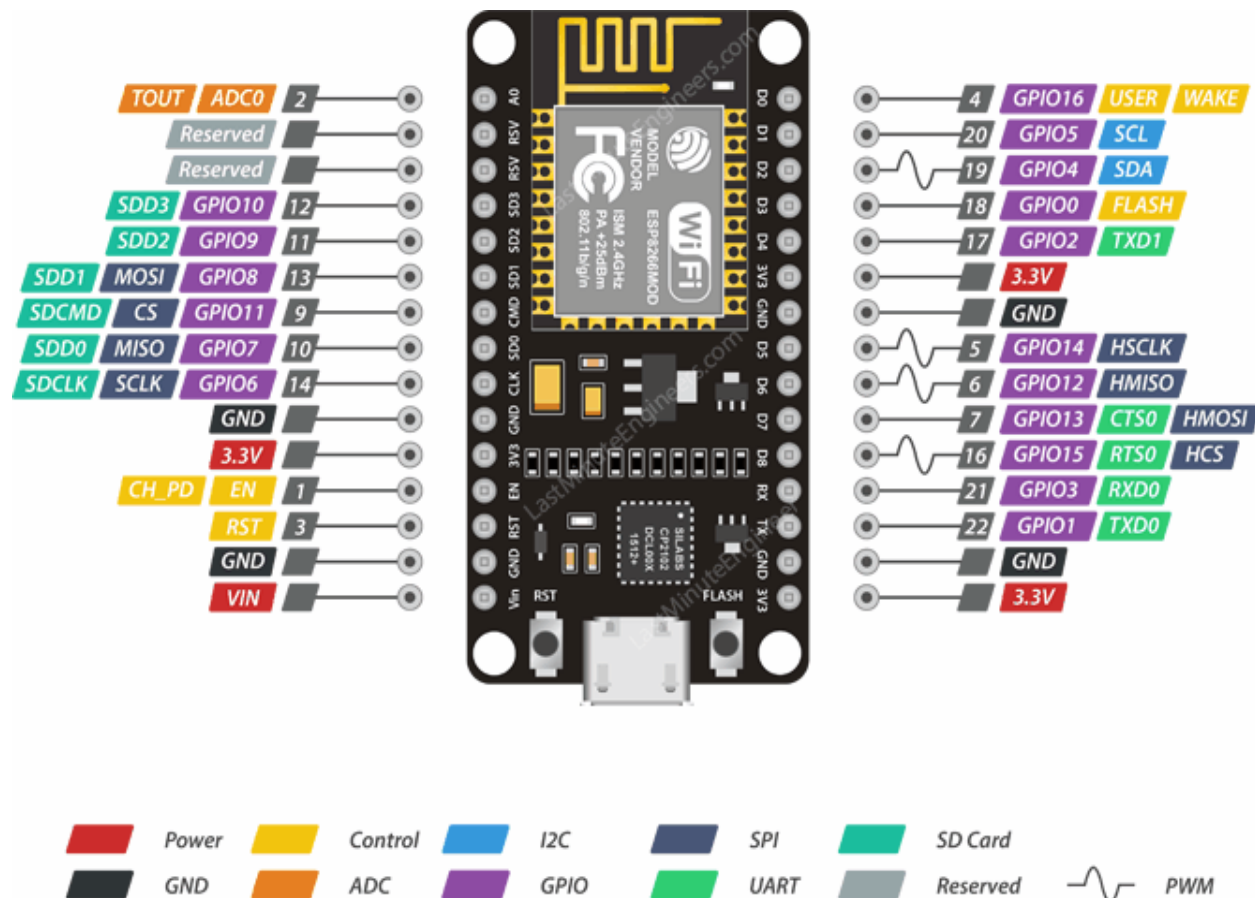
firmware project and a group of independent contributors took over. By the summer of 2016 the NodeMCU included more than 40 different modules.

Types:

There are two available versions of NodeMCU as version 0.9 & 1.0 where the version 0.9 contains **ESP-12** and version 1.0 contains **ESP-12E** where E stands for "Enhanced".

ESP8266 NodeMCU Pinout:

The ESP8266 NodeMCU has total 30 pins that interface it to the outside world. The connections are as follows:



Power Pins

There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used to directly supply the ESP8266 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pins are the output of an on-board voltage regulator. These pins can be used to supply power to external components.

GND

It is a ground pin of ESP8266 NodeMCU development board.

I2C Pins

These are used to hook up all sorts of I2C sensors and peripherals in your project. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

GPIO Pins

ESP8266 NodeMCU has 17 GPIO pins which can be assigned to various functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

ADC Channel

The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC viz. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

UART Pins

ESP8266 NodeMCU has 2 UART interfaces, i.e., UART0 and UART1, which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. It supports flow control. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

SPI Pins

ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks of 80 MHz
- Up to 64-Byte FIFO

SDIO Pins

ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

PWM Pins

The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μ s to 10000 μ s, i.e., between 100 Hz and 1 kHz.

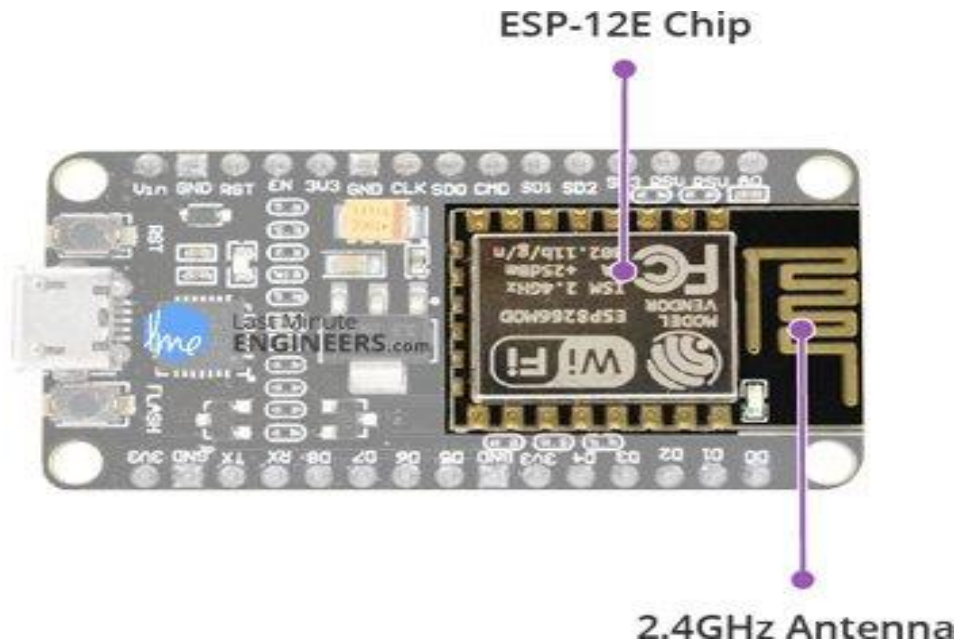
Control Pins

These are used to control ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

- EN pin – The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- RST pin – RST pin is used to reset the ESP8266 chip.
- WAKE pin – Wake pin is used to wake the chip from deep-sleep.

ESP-12E 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.

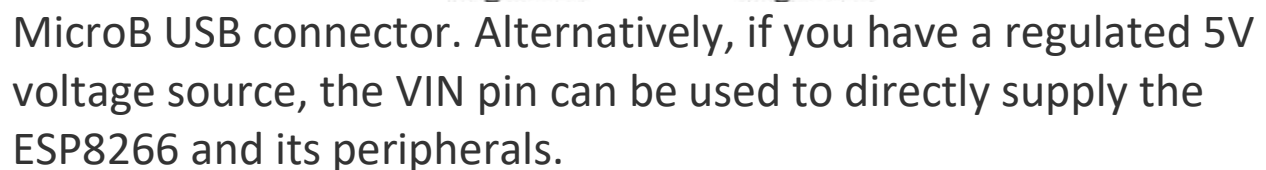
There's also 128 KB RAM and 4MB of Flash memory (for program and data storage) just enough to cope with the large strings that make up web pages, JSON/XML data, and everything we throw at IoT devices nowadays.

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. This makes the ESP8266 NodeMCU even more versatile.

Power Requirement:

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

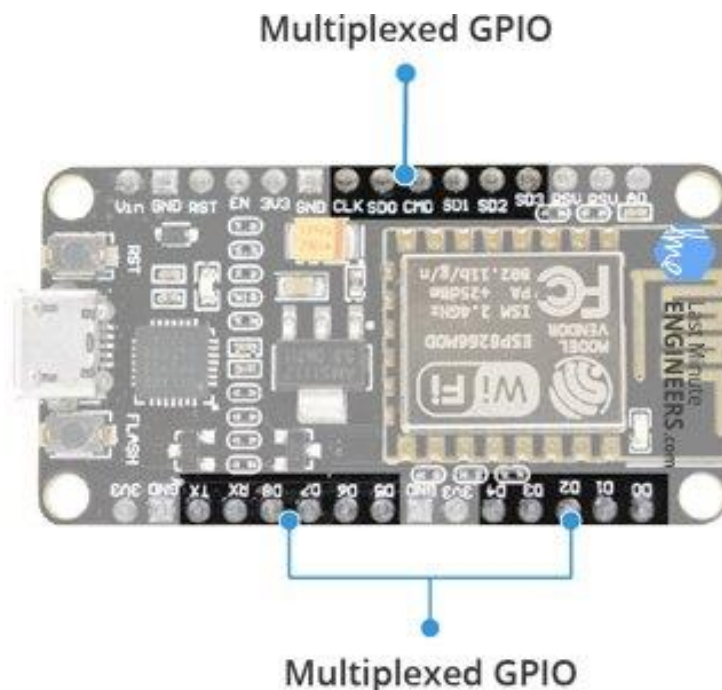
Power to the ESP8266 NodeMCU is supplied via the on-board



The ESP8266 NodeMCU 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:

- 23

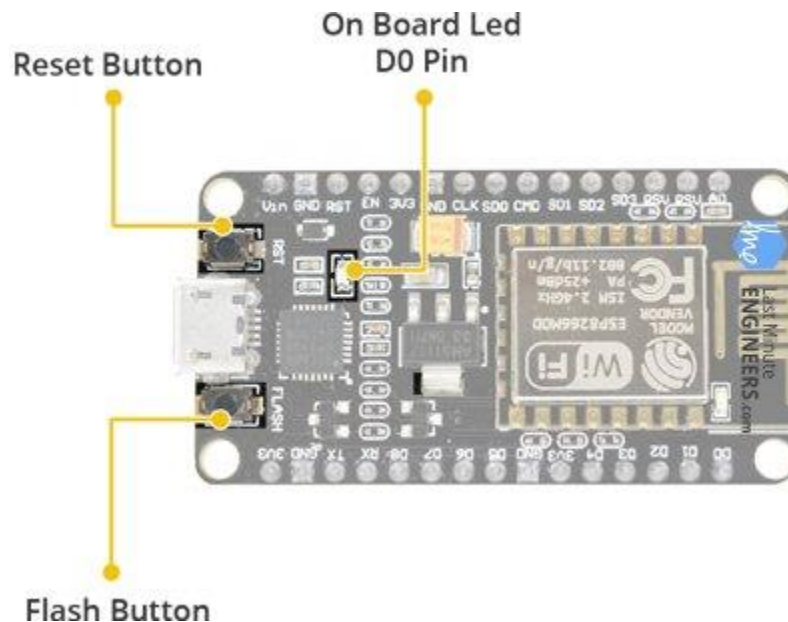
- UART interface – UART interface is used to load code serially.
- PWM outputs – PWM pins for dimming LEDs or controlling motors.



- 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.
- Thanks to the ESP8266's pin multiplexing feature (Multiple peripherals multiplexed on a single GPIO pin). Meaning a single GPIO pin can act as PWM/UART/SPI.

On-board Switches & LED Indicator:

- The ESP8266 NodeMCU 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.

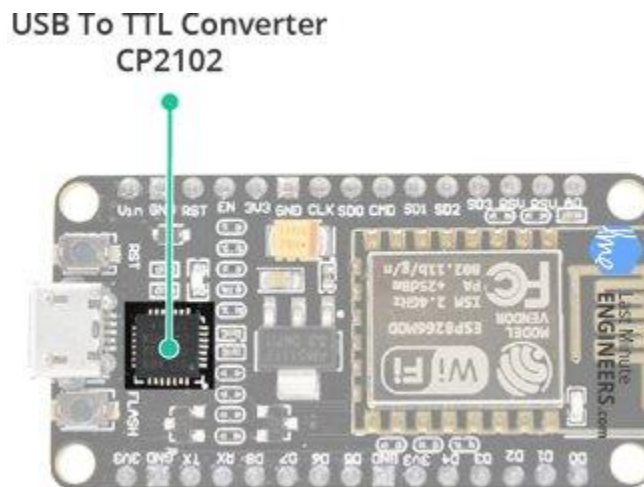


- The other FLASH button on the bottom left corner is the download button used while upgrading firmware.
- The board also has a LED indicator which is user programmable and is connected to the D0 pin of the board.

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.



ESP8266 Arduino Core:

As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate toolchains to allow Arduino C/C++ to be compiled for these new processors. They did this with the introduction of the Board Manager and the SAM Core. A "core" is the collection of software components required by the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file for the target MCU's machine language. Some ESP8266 enthusiasts developed an Arduino core for the ESP8266 Wi-Fi SoC, popularly called the "ESP8266 Core for the Arduino IDE". This has become a leading software development platform for the

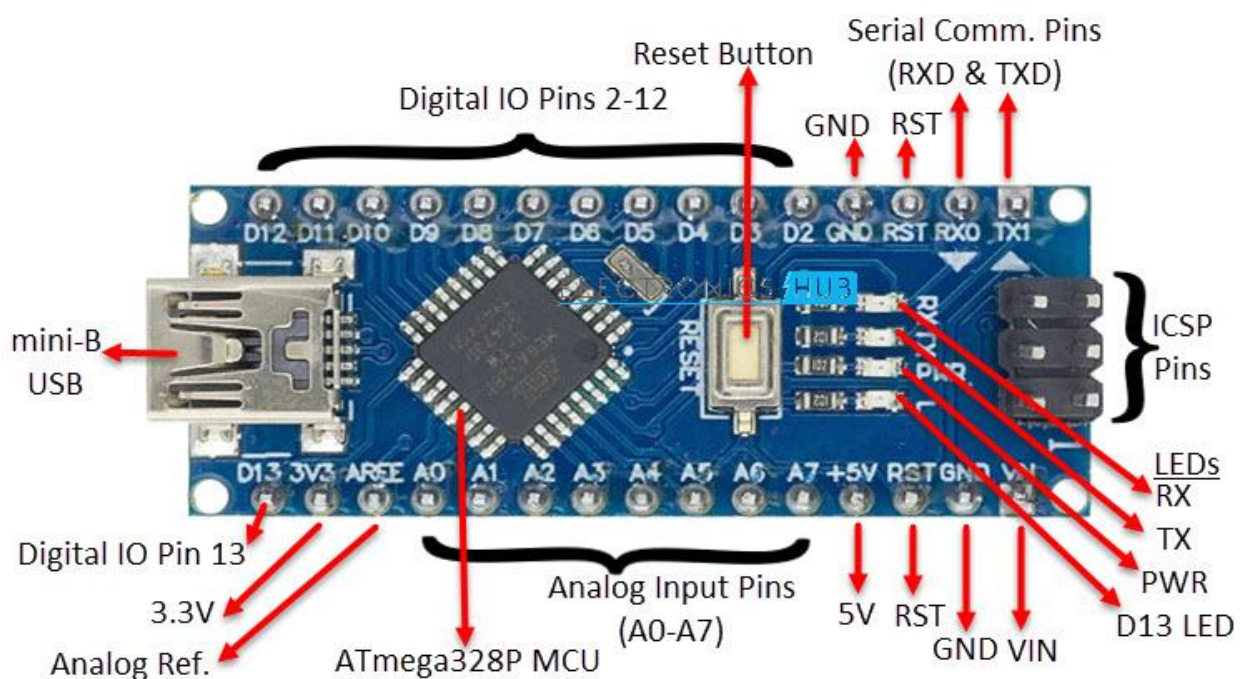
various ESP8266-based modules and development boards, including NodeMCUs.

Arduino Nano:

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one.

Arduino Nano comes with a crystal oscillator of frequency 16 MHz. It is used to produce a clock of precise frequency using constant voltage.

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter



message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

Arduino IDE Installation (Windows):

Step 1 – First you must have your Arduino board (you can choose your favourite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug).

In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.



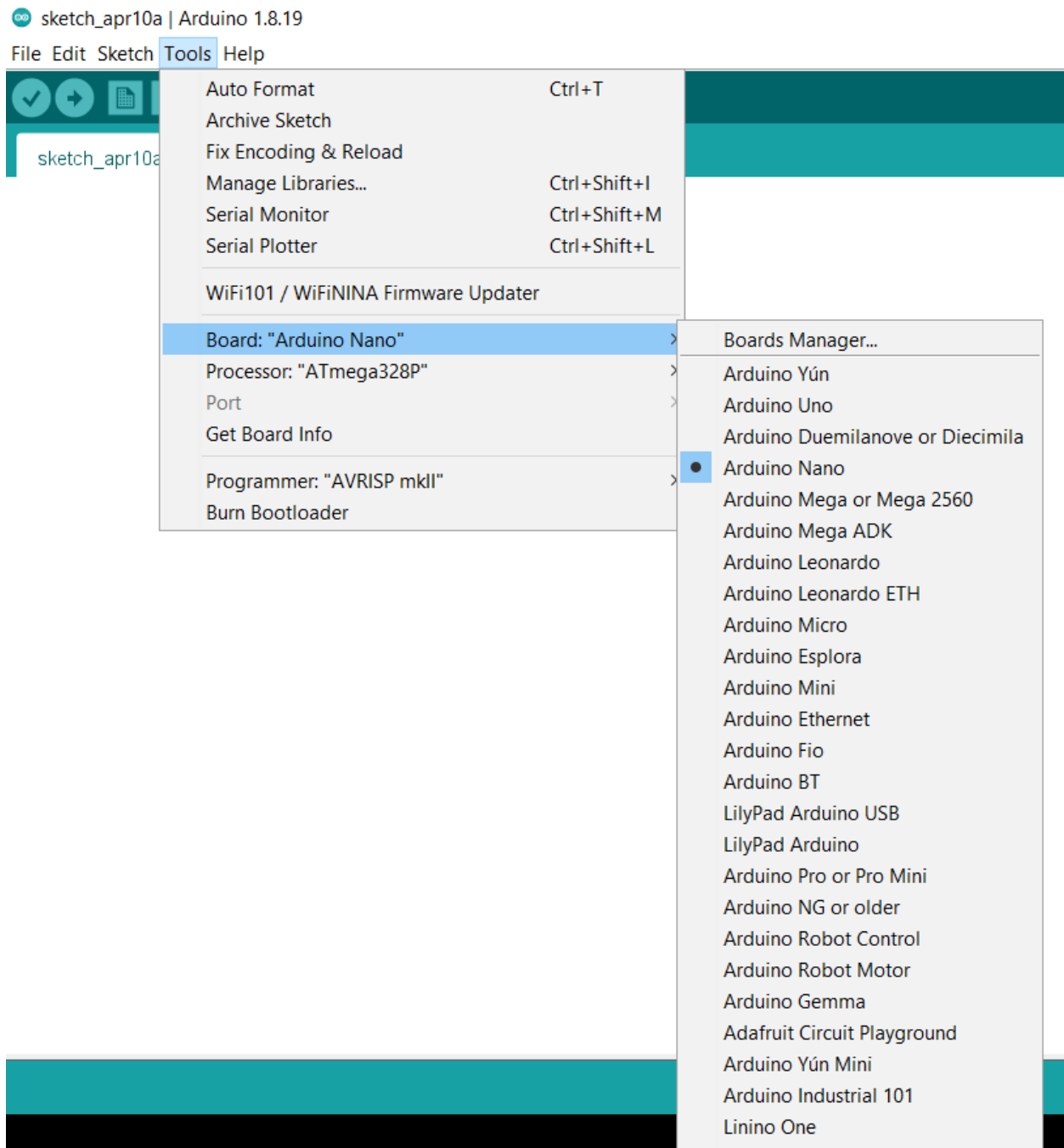
Step 2 –

1. Visit <http://www.arduino.cc/en/main/software> to download the latest Arduino IDE version for your computer's operating system. There are versions for Windows, Mac, and Linux systems. At the download page, click on the "Windows Installer" option for the easiest installation.

2. Save the .exe file to your hard drive.
3. Open the .exe file.
4. Click the button to agree to the licensing agreement.
5. Decide which components to install, then click “Next”.
6. Select which folder to install the program to, then click “Install”.
7. Wait for the program to finish installing, then click “Close”.
8. Now find the Arduino shortcut on your Desktop and click on it. The IDE will open up and you’ll see the code editor.



The next thing to do is to make sure the software is set up for your particular Arduino board. Go to the “Tools” drop-down menu, and find “Board”. Another menu will appear, where you can select from a list of Arduino models. I have the Arduino Nano, so I chose “Arduino Nano”.



Fingerprint Module:



This is a finger print sensor module with TTL UART interface. The user can store the finger print data in the module and can configure it in 1:1 or 1: N mode for identifying the person. The finger print module can directly interface with 3v3 or 5v Microcontroller.

MG90S Micro Servo Motor:

The MG90S Micro Servo is a 13g servo motor that is great for applications in low-cost robotics and automation. The MG90S can be powered directly from any 5.0V Arduino board, and can be controlled using the servo library included in most Arduino IDEs.



A servomotor (or servo motor) is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback.

Keypad:

The 4 x 4 Matrix Keypad Module is a non-encoded matrix keypad consisting of 16 keys in parallel. The keys of each row and column are connected through the pins outside – pin 1-4 as

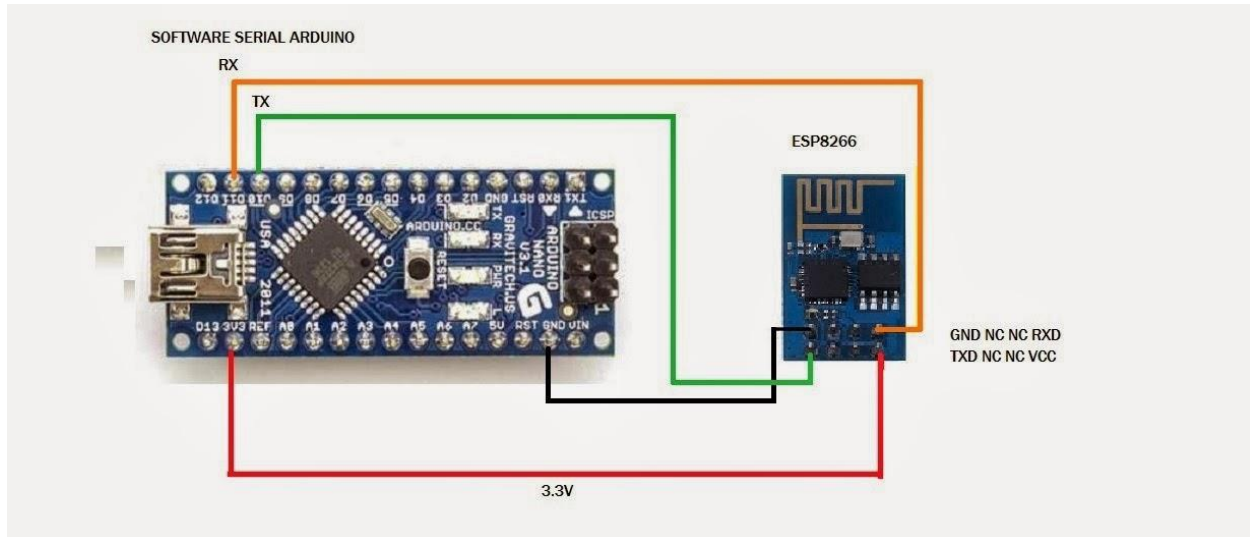
labelled beside control the rows, when pin 5-8, the columns. A 4×4 matrix keypad consisting of microswitch buttons.



STEPS FOR ASSEMBLING:

Step 1:

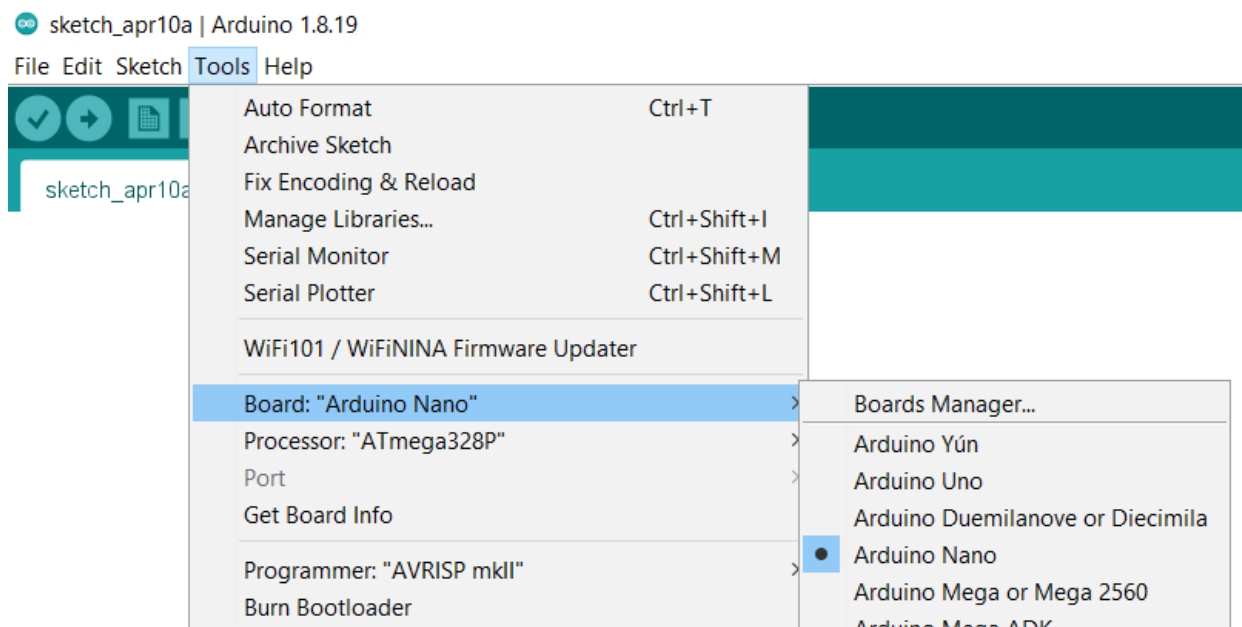
Take a wooden cardboard attach and connect the NodeMCU to



Arduino nano.

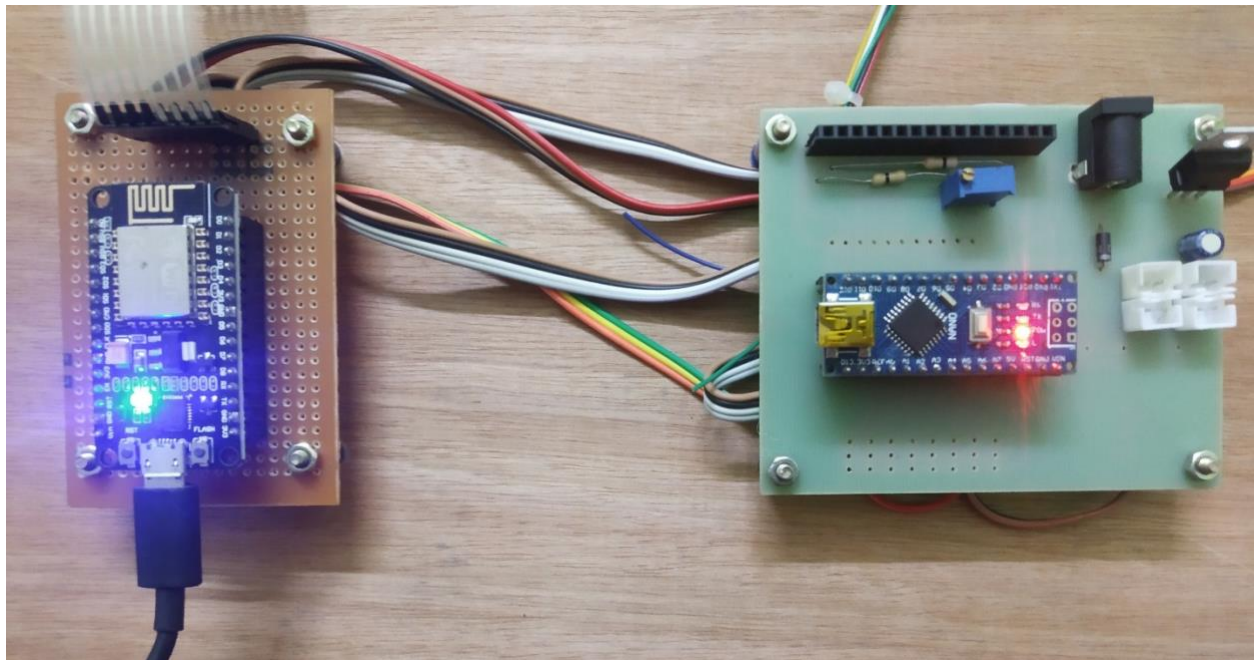
Install the Arduino IDE and compile the code in the code editor.

The next thing to do is to make sure the software is set up for your particular Arduino board. Go to the “Tools” drop-down



menu, and find “Board”. Another menu will appear, and select from a list of Arduino models.

Connect your NodeMCU to the laptop/computer using USB cable. You will see the blue onboard LED flicker on NodeMCU and red LED flicker on Arduino nano when powered up.

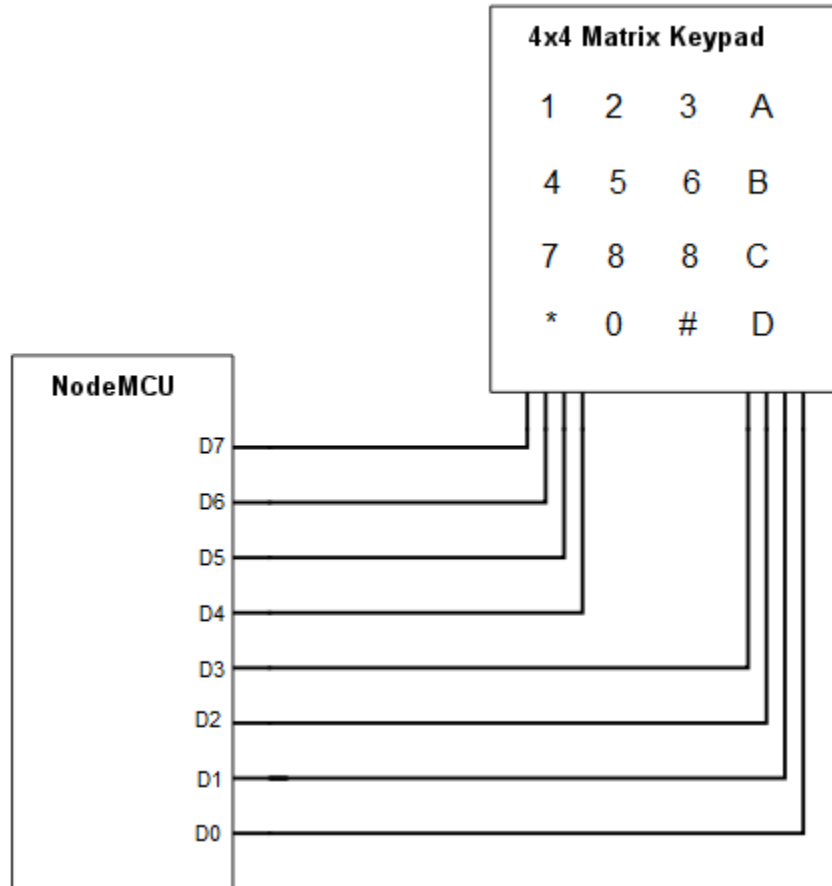


Step 2:

Connect the Keypad to the NodeMCU.

A 4×4 matrix only has 8 interface pins, which correspond to the columns and rows of the matrix.

So, to interface with the matrix keypad, we simply need to connect both its columns and rows to the pins of the NodeMCU.



Step 3:

Connection of Fingerprint sensor to Arduino nano.

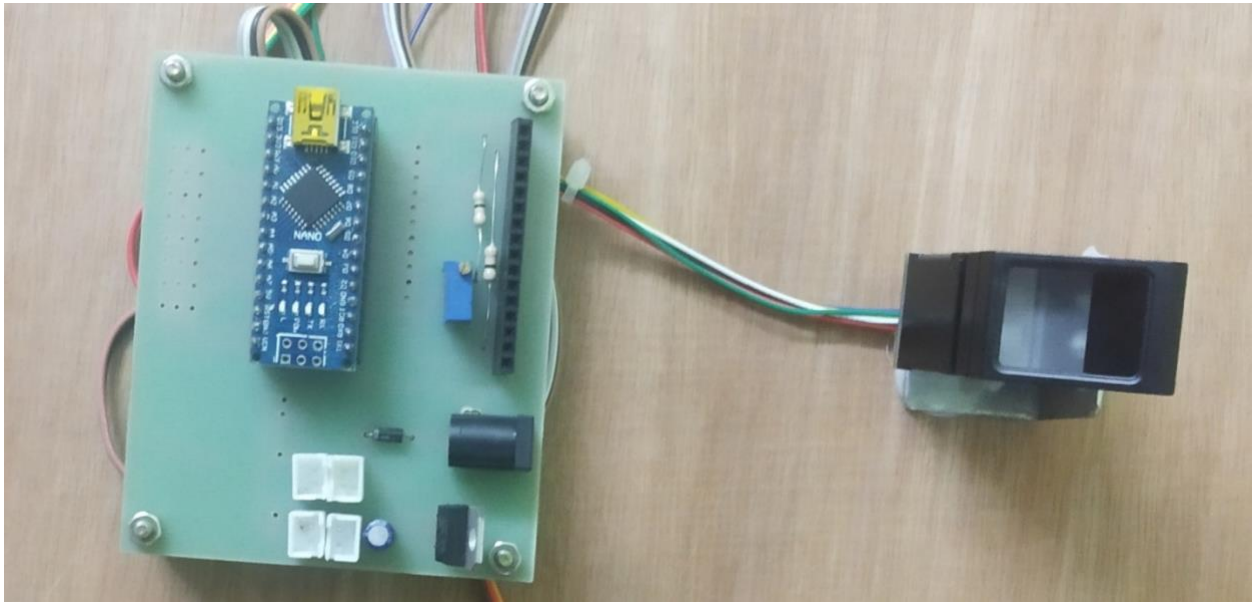
Fingerprint sensor includes pins like DNC, VCC, TX, RX, and GND. These pins are connected through different coloured connecting wires. Each colour wire is used to indicate each pin of the sensor.

- DNC pin is connected by a white wire
- VCC pin is connected by a red wire
- TX pin is connected by a blue wire

- RX pin is connected by a green wire
- GND pin is connected by a black wire

The connection of the fingerprint sensor module to an Arduino board can be done like the following.

- The black wire is connected to the GND pin of the Arduino
- The red wire is connected to the 5V of the Arduino
- Green wire is connected to the digital pin-2 of the Arduino
- The white wire is connected to the digital pin-3 of the



Arduino.

Step 4:

- In the Arduino IDE, go to File > Examples > Adafruit Fingerprint Sensor Library > Enroll.
- Upload the code, and open the serial monitor at a baud rate of 9600.

- For fingerprint enrollment we have to enter 1111 in keypad and it shows we have to enter ID for fingerprint in serial monitor.
- You should enter an ID for the fingerprint. Here we give ID as 5 in serial monitor.

```

COM3
Found fingerprint sensor!
Reading sensor parameters
Status: 0x0
Sys ID: 0x0
Capacity: 1000
Security level: 3
Device address: FFFFFFFF
Packet len: 128
Baud rate: 57600
Waiting for valid finger...
Sensor contains 7 templates
1111
Enroll Open
Ready to enroll a fingerprint!
Please type in the ID # (from 1 to 127) you want to save t

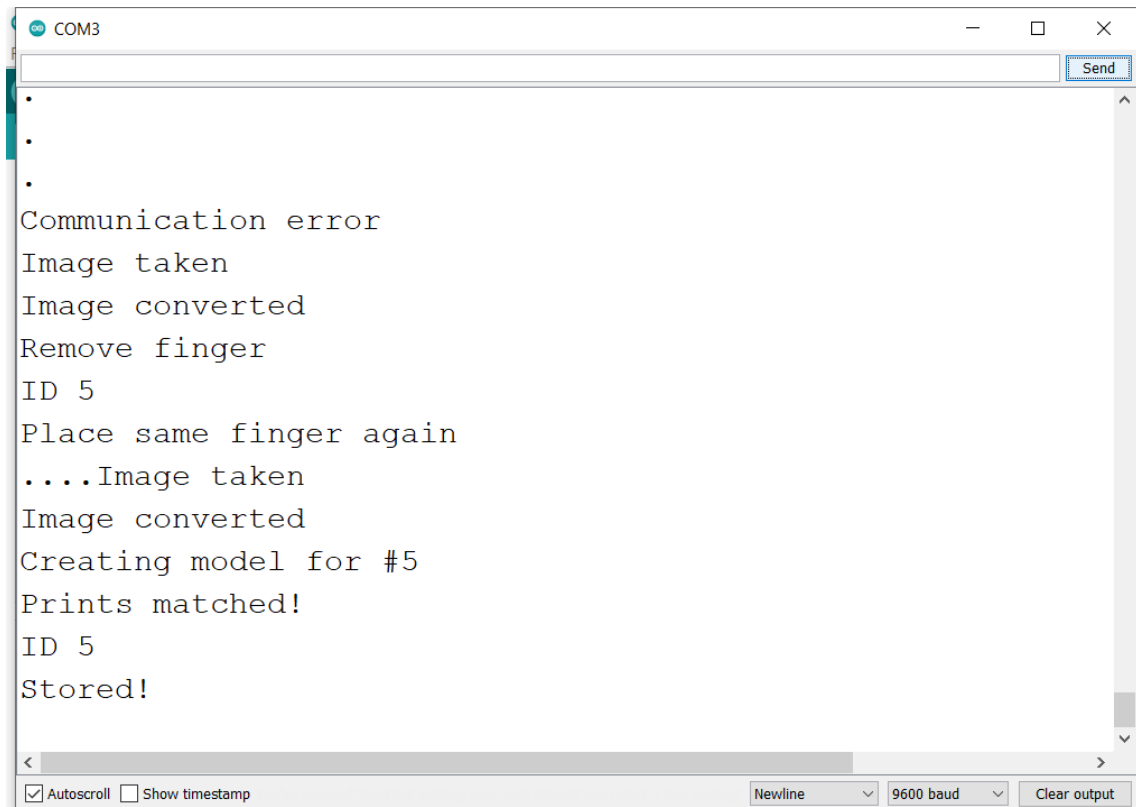
// creating struct to store customer data
struct customer_data {

```

Board at COM3 is not available
Board at COM3 is not available
Board at COM3 is not available

- Place your finger on the scanner and follow the instructions on the serial monitor.

This module can store up to **1000 fingerprints** in its memory and its false acceptance rate is less than 0.001% which makes it pretty secure.



```
COM3
.
.
.
Communication error
Image taken
Image converted
Remove finger
ID 5
Place same finger again
....Image taken
Image converted
Creating model for #5
Prints matched!
ID 5
Stored!
```

```
// creating struct to store customer data
struct customer_data {
```

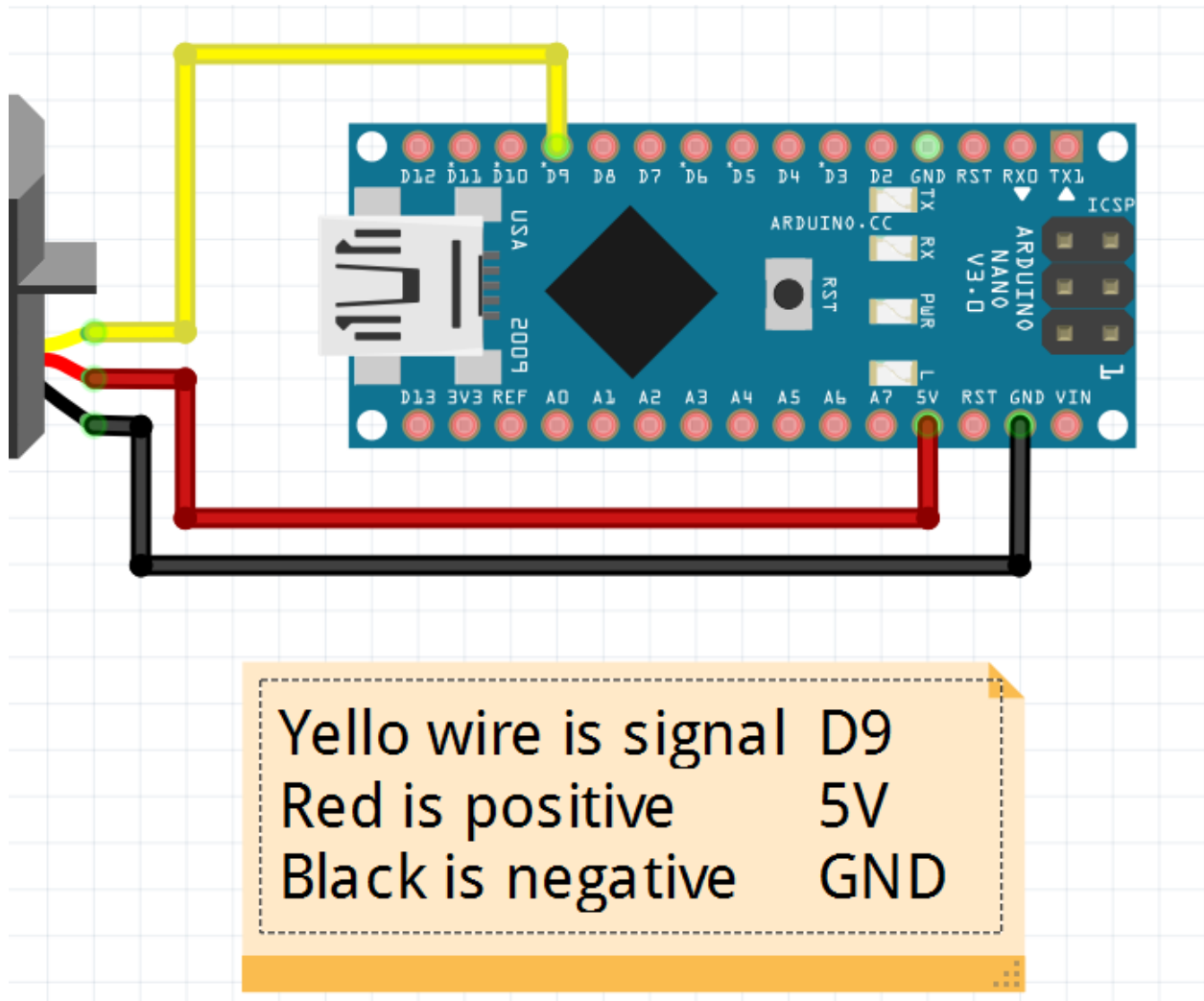
```
Board at COM3 is not available
Board at COM3 is not available
```

Step 5:

Connect the servo motor to the Arduino nano as follow.

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the board.

The signal pin is typically yellow or orange and should be connected to PWM pin on the board.

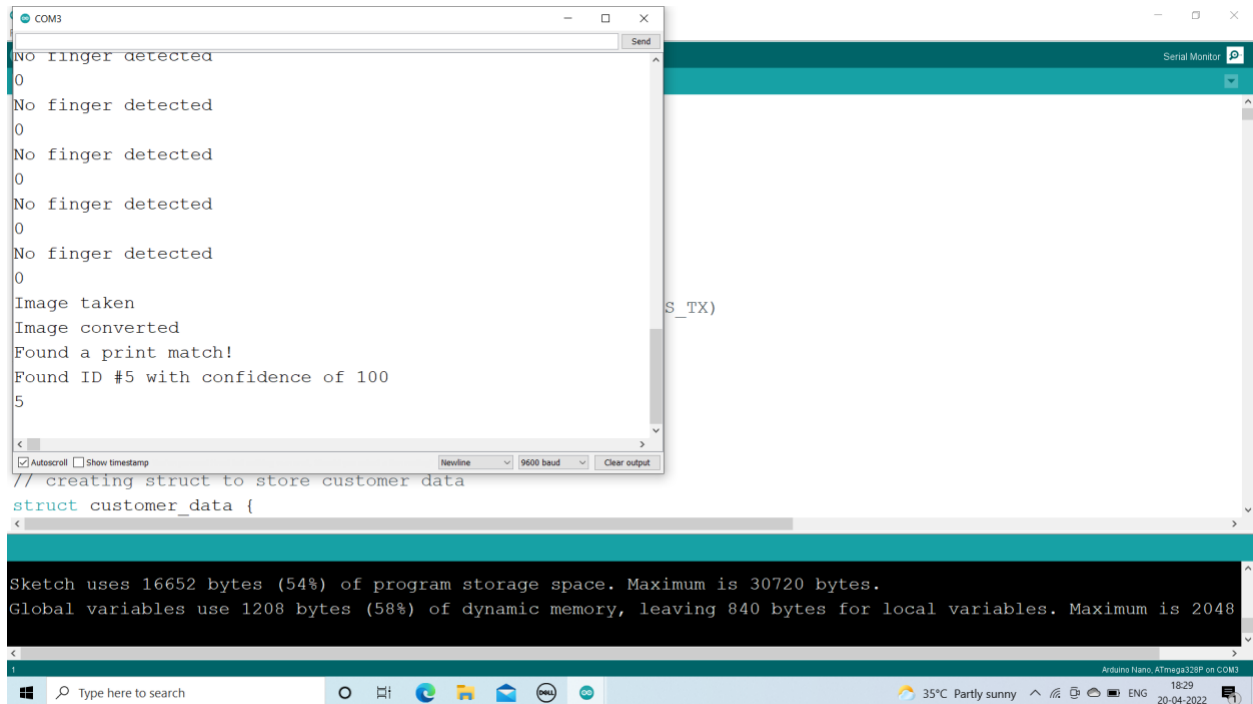


Step 6:

Through the USB cables give the current supply to Arduino boards. After compiling the code enter the pin number of the respective fingerprint ID.

If the fingerprint matches the servo motor rotates and the locker opens.

If the fingerprint doesn't match the servo motor doesn't rotate.

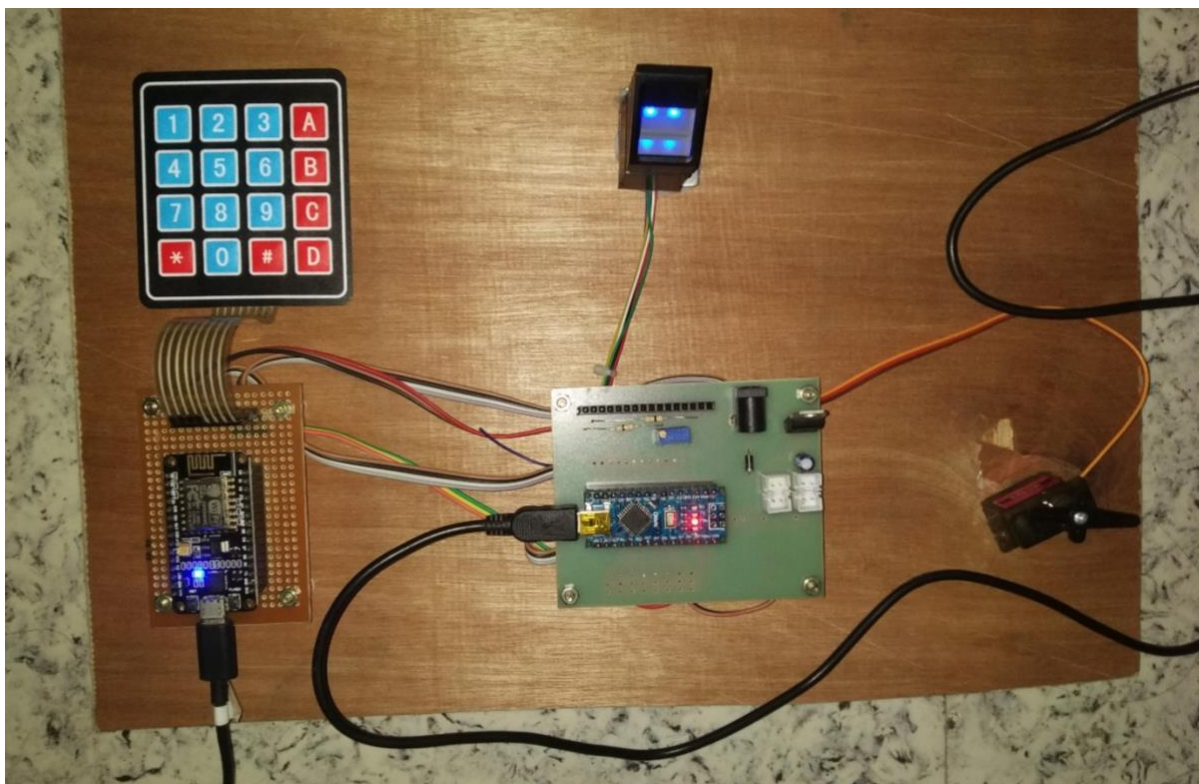


```
COM3
No finger detected
0
No finger detected
0
No finger detected
0
No finger detected
0
No finger detected
0
Image taken
Image converted
Found a print match!
Found ID #5 with confidence of 100
5

// creating struct to store customer data
struct customer_data {

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino IDE: Arduino Nano, ATmega328P on COM3
```



CODE:

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $
#include<Adafruit_Fingerprint.h>
#if (defined(__AVR__) || defined(ESP8266)) && !defined(__AVR_ATmega2560__)
SoftwareSerialmySerial(2,3);
#else
#definemySerialSerial1
#endif
#include<Servo.h>
ServomyServo;
Adafruit_Fingerprintfinger=Adafruit_Fingerprint(&mySerial);
uint8_tid;
#include<Keypad.h>
Stringone="1234";
Stringtwo="1235";
Stringthree="1236";
Stringfour="1237";
constbyteROWS=4;//fourrows
constbyteCOLS=4;//fourcolumns
charhexaKeys[ROWS][COLS]={

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino Nano, ATmega328P on COM3
38°C Sunny 17:02
20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $
constbyteCOLS=4;//fourcolumns
charhexaKeys[ROWS][COLS]={
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};
bytecolPins[COLS]={17,16,15,14};
byterowPins[ROWS]={5,4,19,18};
KeypadcustomKeypad=Keypad(makeKeymap(hexaKeys),rowPins,colPins,ROWS,COLS);
voidsetup(){
Serial.begin(9600);
while(!Serial); //ForYun/Leo/Micro/Zero...
delay(100);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
pinMode(9,OUTPUT);
pinMode(10,OUTPUT);

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino Nano, ATmega328P on COM3
38°C Sunny 17:03
20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
digitalWrite(7, LOW);
digitalWrite(8, LOW);
digitalWrite(9, LOW);
digitalWrite(10, LOW);
myservo.attach(6);
myservo.write(180);
Serial.println("\n\nAdafruitFingerprintsensorenrollment");
//setthedatarateforthesensorserialport
finger.begin(57600);
if(finger.verifyPassword()){
Serial.println("Foundfingerprintsensor!");
}else{
Serial.println("Didnotfindfingerprintsensor:");
while(1){delay(1);}
}
Serial.println(F("Readingsensorparameters"));

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino Nano, ATmega328P on COM3
38°C Sunny 17:04 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

while(1){delay(1);}
}
Serial.println(F("Readingsensorparameters"));
finger.getParameters();
Serial.print(F("Status:0x"));Serial.println(finger.status_reg,HEX);
Serial.print(F("SysID:0x"));Serial.println(finger.system_id,HEX);
Serial.print(F("Capacity:"));Serial.println(finger.capacity);
Serial.print(F("Securitylevel:"));Serial.println(finger.security_level);
Serial.print(F("Deviceaddress:"));Serial.println(finger.device_addr,HEX);
Serial.print(F("Packetlen:"));Serial.println(finger.packet_len);
Serial.print(F("Baudrate:"));Serial.println(finger.baud_rate);
finger.getTemplateCount();
if(finger.templateCount==0){
Serial.print("Sensordoesn'tcontainanyfingerprintdata.Pleaserunthe'enroll'example.");
}
else{
Serial.println("Waitingforvalidfinger...");
Serial.print("Sensor contains ");Serial.print(finger.templateCount);Serial.println("

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino Nano, ATmega328P on COM3
38°C Sunny 17:04 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

Serial.println("Waitingforvalidfinger...");
Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println("
templates");
}
}
uint8_treadnumber(void){
uint8_tnum=0;
while(num==0){
while(!Serial.available());
num=Serial.parseInt();
}
returnnum;
}
inti=0;
inttemp=0;
intidtemp;
chara[5]={'0','0','0','0','0'};
voidloop(){

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino Nano, ATmega328P on COM3
17:04
20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

chara[5]={'0','0','0','0','0'};
voidloop(){
charcustomKey=customKeypad.getKey();
if(customKey){
i++;
a[i]=customKey;
//Serial.println(customKey);
if(i==4){
i=0;
Strings=String(a[1])+String(a[2])+String(a[3])+String(a[4]);
Serial.println(s);
if(s.indexOf(one)!=-1){
Serial.println("Authorised1234");
while(1)
{
getFingerprintID();
Serial.println(idtemp);
if(idtemp>0){

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino Nano, ATmega328P on COM3
17:04
20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

Serial.println(idtemp);
if(idtemp>0){
  if(idtemp==5)
  {
    myservo.write(0);
    digitalWrite(7,HIGH);
    delay(5000);
    myservo.write(180);
    digitalWrite(7,LOW);
    idtemp=0;
  }
  break;
}
delay(100);
}
}
elseif(s.indexOf(two)!=-1){
  Serial.println("Authorised1235");

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Type here to search 38°C Sunny 17:04 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

elseif(s.indexOf(two)!=-1){
  Serial.println("Authorised1235");
  while(1)
  {
    getFingerprintID();
    Serial.println(idtemp);
    if(idtemp>0){
      if(idtemp==6)
      {
        myservo.write(0);
        digitalWrite(8,HIGH);
        delay(5000);
        myservo.write(180);
        digitalWrite(8,LOW);
        idtemp=0;
      }
      break;
    }
  }
}

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Type here to search 38°C Sunny 17:05 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

break;
}
delay(100);
}
}
elseif(s.indexOf(three)!=-1){
  Serial.println("Authorised1236");
  while(1)
  {
    getFingerprintID();
    Serial.println(idtemp);
    if(idtemp>0){
      if(idtemp==7)
      {
        myservo.write(0);
        digitalWrite(9,HIGH);
        delay(5000);
        myservo.write(180);
      }
    }
  }
}

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino Nano, ATmega328P on COM3
17:05
20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

delay(5000);
myservo.write(180);
digitalWrite(9,LOW);
idtemp=0;
}
break;
}
delay(100);
}
}
elseif(s.indexOf(four)!=-1){
  Serial.println("Authorised1237");
  while(1)
  {
    getFingerprintID();
    Serial.println(idtemp);
    if(idtemp>0){
      if(idtemp==9)
      {
        myservo.write(0);
        digitalWrite(9,HIGH);
        delay(5000);
        myservo.write(180);
      }
    }
  }
}

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino Nano, ATmega328P on COM3
17:05
20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

if (idtemp>0) {
  if (idtemp==8)
  {
    myservo.write(0);
    digitalWrite(10,HIGH);
    delay(5000);
    myservo.write(180);
    digitalWrite(10,LOW);
    idtemp=0;
  }
  break;
}
delay(100);
}
}
elseif(s.indexOf("1111")!=-1){
temp=1;
Serial.println("EnrollOpen");

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino Nano, ATmega328P on COM3
38°C Sunny 17:05 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

temp=1;
Serial.println("EnrollOpen");
Serial.println("Readytoenrollafingerprint!");
Serial.println("PleasetypeintheID#(from 1to127)youwanttosavethisfinger as...");
id=readnumber();
if (id==0) { //ID#0notallowed,tryagain!
return;
}
Serial.print("EnrollingID#");
Serial.println(id);
while(!getFingerprintEnroll());
}
else{
}
}
}
}

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

Arduino Nano, ATmega328P on COM3
38°C Sunny 17:05 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $
}
else{
}
}
}
}

uint8_tgetFingerprintID(){
uint8_tp=finger.getImage();
switch(p){
caseFINGERPRINT_OK:
Serial.println("Imagetaken");
break;
caseFINGERPRINT_NOFINGER:
Serial.println("Nofingerdetected");
returnp;
caseFINGERPRINT_PACKETRECEIVEERR:
Serial.println("Communicationerror");
returnp;

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

203 Arduino Nano, ATmega328P on COM3
Type here to search 38°C Sunny 17:06 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $
Serial.println("Communicationerror");
returnp;
caseFINGERPRINT_IMAGEFAIL:
Serial.println("Imagingerror");
returnp;
default:
Serial.println("Unknownererror");
returnp;
}
//OKsuccess!
p=finger.image2Tz();
switch(p){
caseFINGERPRINT_OK:
Serial.println("Imageconverted");
break;
caseFINGERPRINT_IMAGEMESS:
Serial.println("Imagetoomessy");
returnp;

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

203 Arduino Nano, ATmega328P on COM3
Type here to search 38°C Sunny 17:07 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $
Serial.println("Imagetoomessy");
returnp;
caseFINGERPRINT_PACKETRECEIVEERR:
Serial.println("Communicationerror");
returnp;
caseFINGERPRINT_FEATUREFAIL:
Serial.println("Couldnotfindfingerprintfeatures");
returnp;
caseFINGERPRINT_INVALIDIMAGE:
Serial.println("Couldnotfindfingerprintfeatures");
returnp;
default:
Serial.println("Unknownerror");
returnp;
}
//OKconverted!
p=finger.fingerSearch();
if(p==FINGERPRINT_OK){

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

203 Arduino Nano, ATmega328P on COM3
38°C Sunny 17:07 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $
//OKconverted!
p=finger.fingerSearch();
if(p==FINGERPRINT_OK){
Serial.println("Foundaprintmatch!");
}elseif(p==FINGERPRINT_PACKETRECEIVEERR){
Serial.println("Communicationerror");
returnp;
}elseif(p==FINGERPRINT_NOTFOUND){
Serial.println("Didnotfindamatch");
returnp;
}else{
Serial.println("Unknownerror");
returnp;
}
//foundamatch!
Serial.print("FoundID#");Serial.print(finger.fingerID);
Serial.print("withconfidenceof");Serial.println(finger.confidence);
idtemp=finger.fingerID;

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

203 Arduino Nano, ATmega328P on COM3
38°C Sunny 17:07 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

Serial.print("FoundID#");Serial.print(finger.fingerID);
Serial.print("withconfidenceof");Serial.println(finger.confidence);
idtemp=finger.fingerID;
returnfinger.fingerID;
}
uint8_tgetFingerprintEnroll(){
intp=-1;
Serial.print("Waitingforvalidfingertoenrollas#");Serial.println(id);
while(p!=FINGERPRINT_OK){
p=finger.getImage();
switch(p){
caseFINGERPRINT_OK:
Serial.println("Imagetaken");
break;
caseFINGERPRINT_NOFINGER:
Serial.println(".");
break;
caseFINGERPRINT_PACKETRECEIVEDERR:

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

251 Arduino Nano, ATmega328P on COM3
38°C Sunny 17:07 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

break;
caseFINGERPRINT_NOFINGER:
Serial.println(".");
break;
caseFINGERPRINT_PACKETRECEIVEDERR:
Serial.println("Communicationerror");
break;
caseFINGERPRINT_IMAGEFAIL:
Serial.println("Imagingerror");
break;
default:
Serial.println("Unknownerror");
break;
}
}
p=finger.image2Tz(1);
switch(p){
caseFINGERPRINT_OK:

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

251 Arduino Nano, ATmega328P on COM3
38°C Sunny 17:07 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $
p=finger.image2Tz(1);
switch(p){
caseFINGERPRINT_OK:
Serial.println("Imageconverted");
break;
caseFINGERPRINT_IMAGEMESS:
Serial.println("Imagetoomessy");
returnp;
caseFINGERPRINT_PACKETRECEIVEERR:
Serial.println("Communicationerror");
returnp;
caseFINGERPRINT_FEATUREFAIL:
Serial.println("Couldnotfindfingerprintfeatures");
returnp;
caseFINGERPRINT_INVALIDIMAGE:
Serial.println("Couldnotfindfingerprintfeatures");
returnp;
default:
}

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $
Serial.println("Couldnotfindfingerprintfeatures");
returnp;
default:
Serial.println("Unknownerror");
returnp;
}
Serial.println("Removefinger");
delay(2000);
p=0;
while(p!=FINGERPRINT_NOFINGER){
p=finger.getImage();
}
Serial.print("ID");Serial.println(id);
p=-1;
Serial.println("Placesamefingeragain");
while(p!=FINGERPRINT_OK){
p=finger.getImage();
}
switch(p){

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

p=finger.getImage();
switch(p){
caseFINGERPRINT_OK:
Serial.println("Imagetaken");
break;
caseFINGERPRINT_NOFINGER:
Serial.print(".");
break;
caseFINGERPRINT_PACKETRECEIVEERR:
Serial.println("Communicationerror");
break;
caseFINGERPRINT_IMAGEFAIL:
Serial.println("Imagingerror");
break;
default:
Serial.println("Unknownerror");
break;
}

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

251 Arduino Nano, ATmega328P on COM3
38°C Sunny 17:08 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

break;
}
}

p=finger.image2Tz(2);
switch(p){
caseFINGERPRINT_OK:
Serial.println("Imageconverted");
break;
caseFINGERPRINT_IMAGEMESS:
Serial.println("Imagetoomessy");
returnp;
caseFINGERPRINT_PACKETRECEIVEERR:
Serial.println("Communicationerror");
returnp;
caseFINGERPRINT_FEATUREFAIL:
Serial.println("Couldnotfindfingerprintfeatures");
returnp;
caseFINGERPRINT_INVALIDIMAGE:

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

251 Arduino Nano, ATmega328P on COM3
38°C Sunny 17:08 20-04-2022
```

```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

returnp;
caseFINGERPRINT_INVALIDIMAGE:
Serial.println("Couldnotfindfingerprintfeatures");
returnp;
default:
Serial.println("Unknownerror");
returnp;
}
//OKconverted!
Serial.print("Creatingmodelfor#"); Serial.println(id);
p=finger.createModel();
if(p==FINGERPRINT_OK){
Serial.println("Printsmatched!");
}elseif(p==FINGERPRINT_PACKETRECEIVEERR){
Serial.println("Communicationerror");
returnp;
}elseif(p==FINGERPRINT_ENROLLMISMATCH){
Serial.println("Fingerprintsdidnotmatch");
}

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

251 Arduino Nano: ATmega328P on COM3 38°C Sunny 17:08 20-04-2022
```

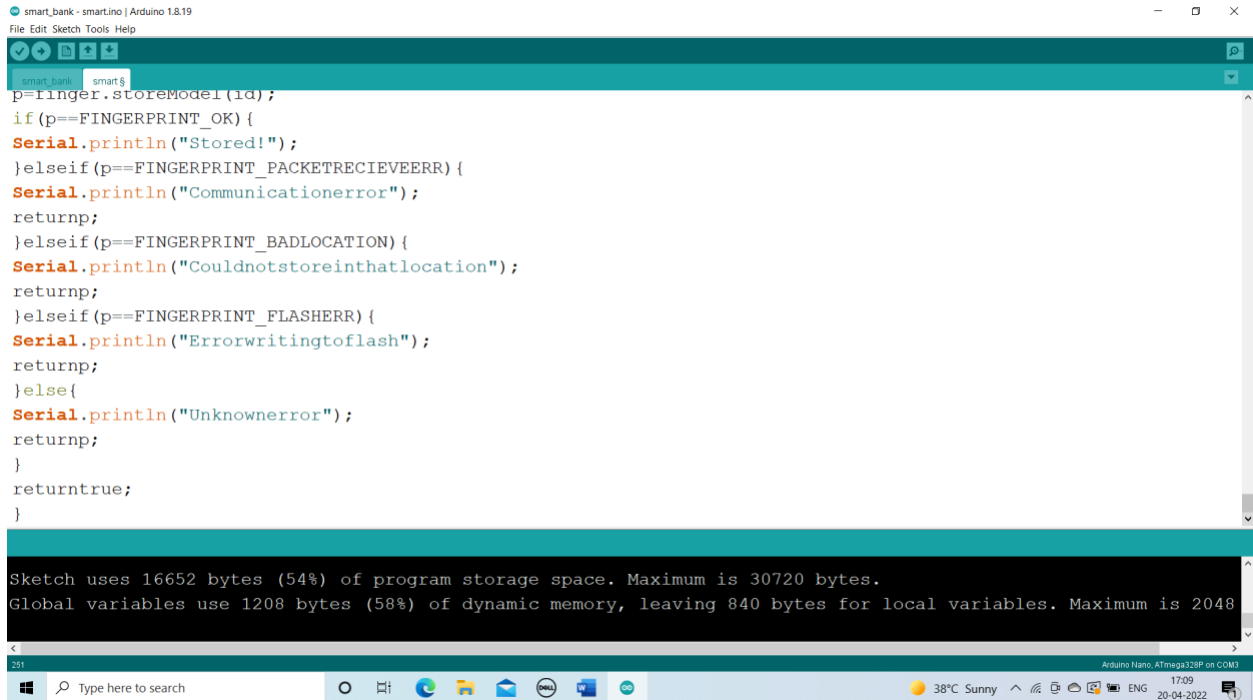
```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart $

Serial.println("Communicationerror");
returnp;
}elseif(p==FINGERPRINT_ENROLLMISMATCH){
Serial.println("Fingerprintsdidnotmatch");
returnp;
}else{
Serial.println("Unknownerror");
returnp;
}
Serial.print("ID");Serial.println(id);
p=finger.storeModel(id);
if(p==FINGERPRINT_OK){
Serial.println("Stored!");
}elseif(p==FINGERPRINT_PACKETRECEIVEERR){
Serial.println("Communicationerror");
returnp;
}elseif(p==FINGERPRINT_BADLOCATION){
Serial.println("Couldnotstoreinthatlocation");
}

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

251 Arduino Nano: ATmega328P on COM3 38°C Sunny 17:08 20-04-2022
```



```
smart_bank - smartino | Arduino 1.8.19
File Edit Sketch Tools Help

smart_bank smart_8
p=finger.storeModel(id);
if (p==FINGERPRINT_OK) {
  Serial.println("Stored!");
}elseif (p==FINGERPRINT_PACKETRECEIVEERR) {
  Serial.println("Communicationerror");
  returnp;
}elseif (p==FINGERPRINT_BADLOCATION) {
  Serial.println("Couldnotstoreinthatlocation");
  returnp;
}elseif (p==FINGERPRINT_FLASHERR) {
  Serial.println("Errorwritingtoflash");
  returnp;
}else{
  Serial.println("Unknownerror");
  returnp;
}
returntrue;
}
```

Sketch uses 16652 bytes (54%) of program storage space. Maximum is 30720 bytes.
Global variables use 1208 bytes (58%) of dynamic memory, leaving 840 bytes for local variables. Maximum is 2048

251

Arduino Nano, ATmega328P on COM3

38°C Sunny 17:09 20-04-2022

CONCLUSION:

The main objective of this project is to build a low cost doubled layer security system to improve the bank security locker. In this we use Fingerprint and password to the security locker for money, jewellery and any other security documents of every other citizen we can make at safe custody. Using this technology an authorized person can only open the lock and collect the money, jewellery and any other documents. This is a low-cost equipment, low power consumption, compact in size, reliable and stand-alone unique system. Hence this will help banks to automate their locker system.

APPLICATIONS:

This project is used in all bank for lockers. In house, school's treasury, college treasury and in industries, VIP vehicles, hospital, offices, Vehicle Security applications.

FUTURE SCOPE:

In addition to this the future scope of this project is to develop smart bank locker security system based on "FACE", "IRIS and RETINA" scanning for visual identification of the person.

REFERENCES:

1. Sanal Malhotra, Banking Locker System with Odor Identification & Security Question Using RFID GSM Technology. International Journal of Advances in Electronics Engineering IJAEE Volume 4: Issue 3
2. P. Sugapriya, K. Amsavalli, Smart Banking Security System Using Pattern Analyzer. International Journal of Innovative Research in Computer and Communication Engineering. An ISO 3297: 2007 Certified Organization Vol.3, Special Issue 8, October 2015
3. M.P. Manjunath, P.M. Ram Kumar, Pradeep Kumar, Nalajala Gopinath, Ms. Hari Priya M.E, NFC Based Bank

Locker System. International Journal of Engineering Trends and Technology (IJETT) Volume23 Number 1- May 2015

4. Peng-Loon Teh, Huo-Chong Ling, Soon- Nyeon Cheong, NFC Smartphone Based Access Control System Using Information Hiding, IEEE Conference on Open Systems (ICOS), December 2013.
5. Vaijanath R. Shintre, Mukesh D. Patil, Banking Security System Using PSoC. International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 7, July 2015.
6. Abhilasha A Sayar , Dr. Sunil N Pawar, “Review of Bank Locker System Using Embedded System” , International Journal of Advanced Research in Computer and Communication Engineering .,Vol. 5, Issue 2,February 2016
7. Prof R. Srinivasan, T. Mettilda, D.Surendhran, K.Gopinath, P.Sathishkumar, Advanced Locker Security System. International conference on Information Engineering, Management and Security
8. 8. Roshni Thomas, Sanjana Mathews, Sona Ojus, Sona Roseline Joseph, Bank Locker Security System Using Face Recognition. International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE) Vol 1, Issue 5, April 2015.

9. Ms. Geetha Hanumanthu, Mr.Dilip Chandra E, Wireless Identification Of RFID, Fingerprint & IRIS. International journal of innovative research and development.
10. Seshapu Prasad, D. Suneel, Proximity Sensor Based Security Lock and Theft Detection. International journal of Science Technology and Management, Vol. No.4, Issue No.01.