A Project report on

Arduino Fingerprint Security Lock

Submitted by

O161636
O161817
O160793
O161881
O161724

Submitted to

Under the Supervision of
Mr.V.Leeladhar

Assistant Professor

As a part of
Partial fulfilment of the degree of Bachelor of Technology in Electronics and Communication Engineering
Date:

Certificate

This is to certify that the report entitled “Arduino Fingerprint Security Lock” submitted by K.Kamal Hanesha,K.Ganapathi Reddy,G.Naveen,M.Swetha and V.Jenifer bearing ID numbers O161636, O161817,O160793,O181881 and O161724 respectively in partial fulfilment of the requirements for the award of Bachelor of Technology in Electronics and Communication Engineering is a bona fide work carried by them under my supervision and guidance.

Head of the Department

P.Janardhana Reddy

Project Internal Guide,

V.Leeladhar

Acknowledgement

I would like to express my sincere gratitude to **V.Leeladhar**, my project guide for valuable suggestions and keen interest throughout the progress of my course and research.

I am grateful to **P.Janardhana Reddy**, HOD of **Electronics & Communication Engineering**, for providing excellent computing facilities and a congenial atmosphere for progressing with my project.

At the outset, I would like to thank **Rajiv Gandhi University of Knowledge Technologies, R.K Valley** for providing all the necessary resources for the successful completion of my course work. At last, but not the least I thank my teammates and other students for their physical and moral support.

With Sincere Regards,
K.Kamal Hanesha
K.Ganapathi Reddy
G.Naveen
M. Swetha
V. Jenifer

Abstract

In this project we are designing a fingerprint security lock using arduino.The implementation of this project is very easy.It is cost-efficient also.Here in this project first step is to enroll the fingerprints through a fingerprint scanner and it should be saved for the further authentication purpose.Next step is to use the fingerprint to open the lock.

The whole system works on the simple principle called matching which is used to compare previously stored templates of fingerprints against user's fingerprints for authentication purposes.If the user's fingerprint is matched with the stored templates of fingerprints then the lock is unlocked.If the user's fingerfrint is not matched with the previously stored templates of fingerprints then the lock won't be unlocked.

The simple fingerprint security lock using arduino can be very useful for door security,forensics,crime investigation,personal identification and attendance system and much more.In future,there could be manymore applications like fingerprint based driving licenses,bank account operations like money transactions and so on.

Introduction

Recently, there has been recorded tremendous increase in the crime rate everywhere in the world. . To get away with this problem, we decided to take help from technology and there this project “Arduino Fingerprint Sensor Lock” developed.The project helps us to implement the fact.

The old practice of using a simple key to unlock a door is time consuming as well as less secure. Replacing those methods with fingerprints, we get access inside a house/room just by placing the correct finger on the sensor. However, only authorized people can open the door because of the special fingerprint technique.

In this project we are going to make a Fingerprint security lock with Arduino. we will explain to you how to make that “**Arduino fingerprint”** lock step by step. This is the most trustable biometric these days.Today, fingerprint project is linked with security and major task, later it may be employed as fingerprint based driving license, bank accounts operation and so on. ‘Matching Algorithm’ is the main principle of this project where specified templates of fingerprints are initially stored. Then, the fingerprint of user is compared with the pre-stored templates of fingerprints.

For instances, there are numerous different applications like Fingerprint entryway lock. It is modest and you can utilize it into your home just as your office moreover.The fingerprint sensor can save up to 137 fingerprints itself. the fingerprint sensor is r307 which has the memory itself. so no need to insert any memory card with it and there are two programs to one is to enroll the fingerprint and another is to verify the fingerprint.

Approach

1. It can only be opened when an authorized user is present , so password or pin is not remembering in it.Most secure and accurate than traditional password
2. Password based security system has always the threat of being stolen and accesed by the unauthorized user, but it can only opened by only authorized user

Required Components

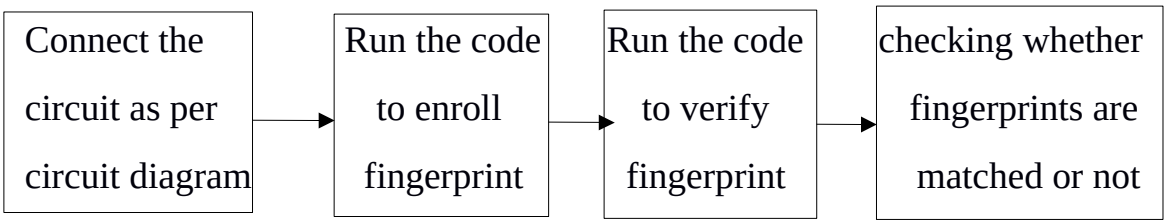
Hardware required :-

Slno	Name of the component	Number of pieces	Cost of the component
1	Breadboard	1	76Rs/-
2	Arduino UNO R3	1	384Rs/-
3	Linear solenoid 12vdc	1	315Rs/-
4	Adafruit R307 fingerprint sensor	1	795Rs/-
5	12v power supply	1	104Rs/-
6	Wooden block	1	--
7	Screw	1	50Rs/-
8	Jumper wires	1 set	65Rs/-
9	Single channel relay	1	60Rs/-
10	resistors-220Ω and leds	2	20Rs/-
11	USB Cable	1	40Rs/-
12	PCB	1	40Rs/-
TOTAL COST			1949Rs/-

Software required :-

Arduino IDE with required libraries like adafruit fingerprint sensor library.

Block Diagram



Explanation :-

1. Connecting Circuit :- Here,First we collect all the components required and then connect all the components according to the circuit diagram.
2. Run the code to enroll fingerprint:- First we need to install arduino software,required boards like arduino Uno R3 and libraries like Adafruit finger print sensor.Then we write the code in the software then run to enroll the fingerprint into arduino through a fingerprint sensor.
3. Run the code to verify fingerprint:- We write the code to verify the fingerprint.Then we run the code and user's fingerprint is scanned with the help of the fingerprint scanner.
4. Checking Whether fingerprints are matched or not:- Then we check the user's fingerprint with the fingerprints which are already enrolled before.If the fingerprints are matched then the lock opens ,if not the solenoid will be in locked state only.

Arduino Uno R3

1. The Arduino Uno(one) is a open source microcontroller board based on the Microchip (its a manufacturer company) ATmega328P(its a single chip microcontroller) and developed by Arduiono.cc .
2. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards and other circuits.
3. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable.
4. It can be powered by the USB cable or by an external 9-volt battery , though it accepts voltages between 7 and 20 volts.

Basic terminology to get familiar with arduino Uno R3

1. **FLASH MEMORY (Program Space):** it is where the Arduino sketch is stored.
2. **SRAM :** (static random accesss memory) is where the sketch creates and manipulates variables when it runs.
3. **EEPROM** : it is memory space where programmers can use to store long term information.
4. **Bootloader :** it is the little program that runs when you turn the Arduino on, or press the reset button. Its main function is to wait for the Arduino software on your computer to send it a new program for the Arduino , which it then writes to the memory on the Arduino.
5. **PWM :** Pulse Width Modulation

Arduino Uno R3 Specifications

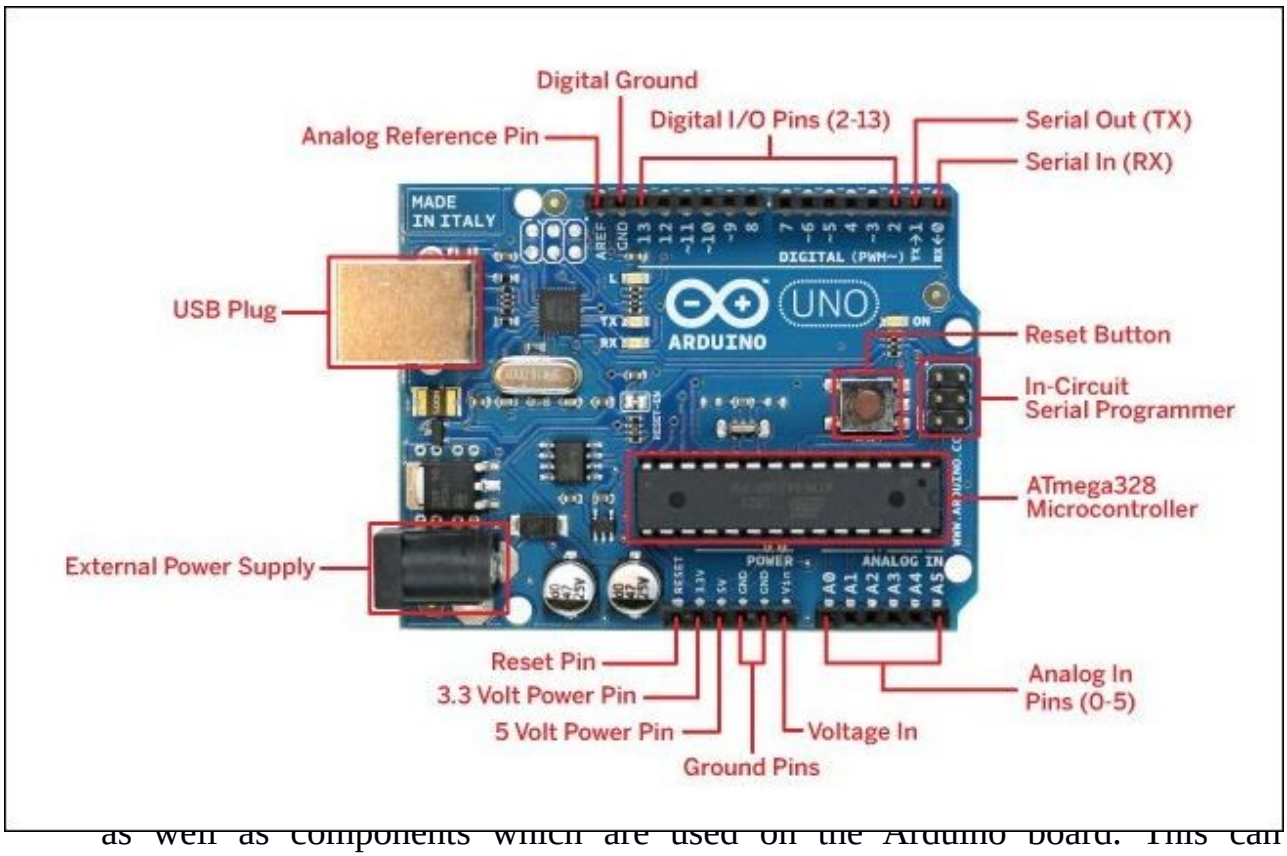
1. It is an ATmega328P based Microcontroller.
2. The Operating Voltage of the Arduino is 5V.
3. The recommended input voltage ranges from 7V to 12V.
4. The i/p voltage (limit) is 6V to 20V.
5. Digital input and output pins-14

   -- i. Digital input and output pins (PWM) -6

ii. Analog input pins are 6.

6. DC Current for each I/O Pin is 20mA.
7. DC Current used for 3.3V Pin is 50mA.
8. Flash Memory -32KB and 0.5 KB memory is used by the boot loader.
9. SRAM is 2KB and EEPROM is 1 KB.
10. In Built LED and speed of the CLK is 16 MHZ.
11. The weight of the Arduino board is 25g.

Arduino Uno R3 Pin Diagram



as well as components which are used on the Arduino board. This can approach from the input voltage through a regulator.3V3 : A 3.3 supply voltage can be generated with the onboard regulator and the highest draw current will be 50mA.

3. **GND :** GND (ground) pins.

Input and Output :

1) We knows that an arguing Uno R3 includes 14 -digital pins which can be used as an

2) Input otherwise output by using the functions like pin Mode(), digital Read (), and digital Write().These pins can operate with 5V, and every digital pin can give or recieve 20mA,& includes a 20k to 50k ohm pull up resistor. The maximum current on any pin is 40mA which cannot surpass for avoiding the microcontroller from the damage.

Some of the pins of an Arduino include specific functions Serial Pins :

1. The serial pins of an Arduino board are TX(1) and RX(0) pins and these pins can be used to transfer the TTL serial data. The connection of these pins can be done with the equivalent pins of the Atmega8 U2 USB to TTL chip.
2. **External Interrupt Pins :** The external interrupt pins of the board are 2 and 3 and these pins can be arranged to activate an interrupt on a rising otherwise falling edge , a low -value otherwise a modify in value.
3. **PWM Pins :** The PWM pins of Arduino are 3,5,6,9,10,&11. and gives an output of an 8-bit PWM with the function analog Write ().
4. **SPI (Serial Peripheral Interface) Pins :** The SPI pins are 10,11,12,13 namely SS,MOSI,MISO,SCK and these will maintain the SPI communication with the help of the SPI library.
5. **LED Pin:** An arguing board is inbuilt with a LED using digital pin-13. Whenever the digital pin is high the LED will glow otherwise it will not glow.
6. **TWI (2-Wire Interface) Pins :** The TWI pins are SDA or A4 ,& SCL or A5. Which can support the communicator of TWI with the help of Wire library.
7. **AREF (Analog Reference) Pin :** An analog reference pin is the reference voltage to the inputs of an analog i/ps using the function like analog Reference().
8. **Reset (RST) Pin :** The pin brings a low line for resetting the microcontroller, and it is very useful for using an RST button toward shields which can block the one over the Arduino R3 board.

Communication :

The communication protocols of an Arduino Uno include SPI,12C, and UART serial communication.

1. **UART :** An Arduino Uno uses the two functions like the transmitter digital pin1 and the reciever digital pin0. These pins are mainly used in UART TTL serial communication.
2. **I2C :** An Arduino UNO board employs SDA pin otherwise A4 Pin & A5 pin otherwise SCL pin is used for I2C communication with wire library. In this both the SCL and CLK signal and data signal.
3. **SPI Pins :** The SPI communication includes MOSI,MISO and SCK.
4. **MOSI (Pin11) :** This is the master out slave inthe pin. Used to transmit the data to the devices.

Uses :

1. It is an open-source project, software/hardware is extremely accessible and very flexible to be customized and extended.
2. It is flexible, offers a variety of digital and analog inputs, SPI and serial interface and digital and PWM outputs
3. It is easy to use, connects to computer via USB and communicates using standard serial protocol, runs in standalone mode and as interface connected to PC/Macintosh computers
4. It is inexpensive, around 30 euro per board and comes with free authoring software
5. Arduino is backed up by a growing online community, lots of source code is already available and we can share and post our examples for others to use, too!

What can we do with Arduino ?

Arduino is a great tool for developing interactive objects, taking inputs from a variety of switches or sensors and controlling a variety of lights, motors and other outputs. Arduino projects can be stand-alone or they can be connected to a computer using USB. The Arduino will be seen by the computer as a standard serial interface . There are serial communication APIs on most programming languages so interfacing Arduino with a software program running on the computer should be pretty straightforward.

The Arduino board is a microcontroller board, which is a small circuit (the board) that contains a whole computer on a small chip (the microcontroller). There are different versions of the Arduino board: they are different in components, aim

and size, etc. Some examples of Arduino boards are: Arduino Diecimila, Arduino Duemilanove, Freeduino, Arduino NG and lot more. Arduino schematics are distribute using an open licese so anyone is free to build his own Arduino compatible board. The Arduino name is a registered trademark so you won't be able to call your hacked board Arduino.

Arduino software

what is Arudino Software ?

1. Arduino Software is simply, set of instructions, data or programs used to operate arduino and execute specific tasks.
2. Arduino Software (IDE) – contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Writing Sketches

Programs wriiten using Arduino Software (IDE) are called “sketches”. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays error. The console displays text output by the Arudion Software (IDE),including complete error messages and other information. The bottom righthand corner of the window displays the configured boaerd and serial port. The toolbar buttons allow you to verify and upload programs, create open and save sketches and open the serial monitor.Additonal commands are found within the five menus: **File,Edit,Sketch,Tools,Help**. The menus are context sensitive which means only those items relevant to the work currently being carried out are available.

File :-

1. **New :** creates a new instance of the editor,with the bare minimum structure of a sketch already in place.
2. **Open :** Allows to load a sketch file browsing through the computer drives and folders.
3. **Sketchbook :** Shows the current sketches within the sketchbook folder structure;clicking on any name opens the corresponding sketch in a new editor instance.

4. **Print :** Sends the current sketch to the printer according to the settins defined in Page Setup.

Edit :

1. Undo/Redo

   Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.

2. Cut

   Removes the selected text from the editor and places it into the clipboard.

3. Copy

   Duplicates the selected text in the editor and places it into the clipboard.

4. Copy for Forum

   Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.

5. Copy as HTML

   Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages. Etc.,

Sketch :

1. Verify/Compile

   Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

2. Upload

   Compiles and loads the binary file onto the configured board through the configured Port.

3. Upload Using Programmer

   This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a Tools -> Burn Bootloader command must be executed.

4. Export Compiled Binary

Saves a .hex file that may be kept as archive or sent to the board using other tools.

5. Show Sketch Folder

   Opens the current sketch folder.

6. Include Library

   Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see librariesbelow. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

7. Add File...

   Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side o the toolbar

Tools :

1. Auto Format
2. This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
3. Archive Sketch

   Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

4. Fix Encoding & Reload

   Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

5. Serial Monitor

   Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

6. Board

   Select the board that you're using. See below for descriptions of the various boards

7. Port

   This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

8. Programmer

   For selecting a harware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

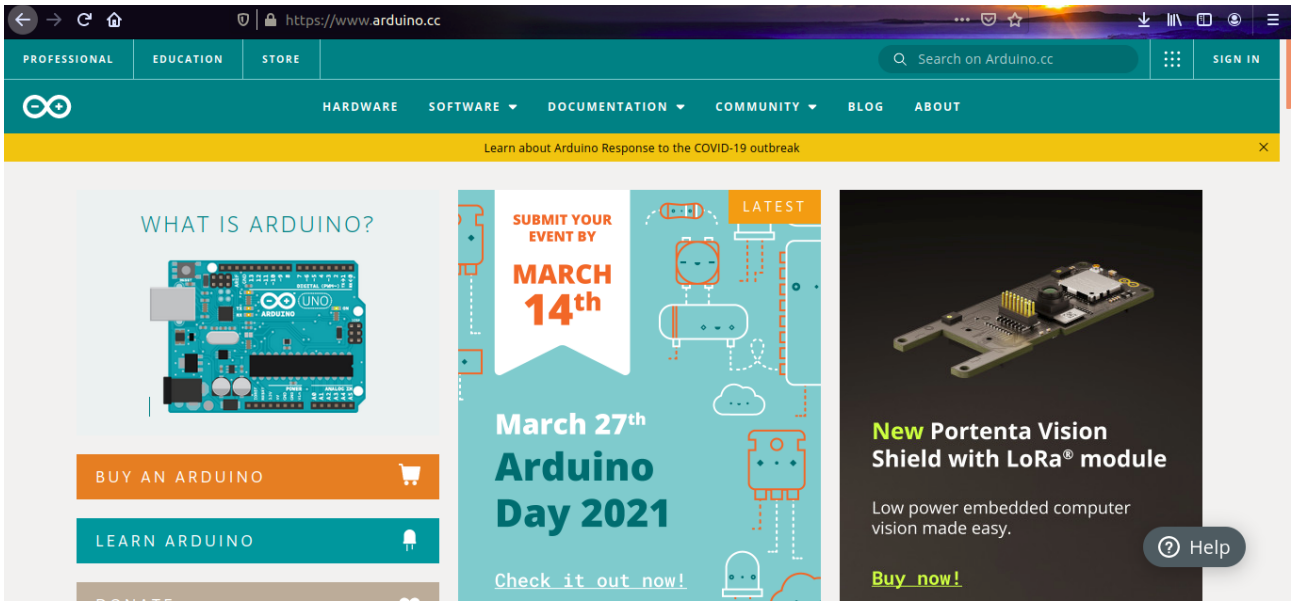
Burn Bootloader bootloader The items in this menu allow you to burn onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

Installation of software and boards

Steps to Install arduino software :-

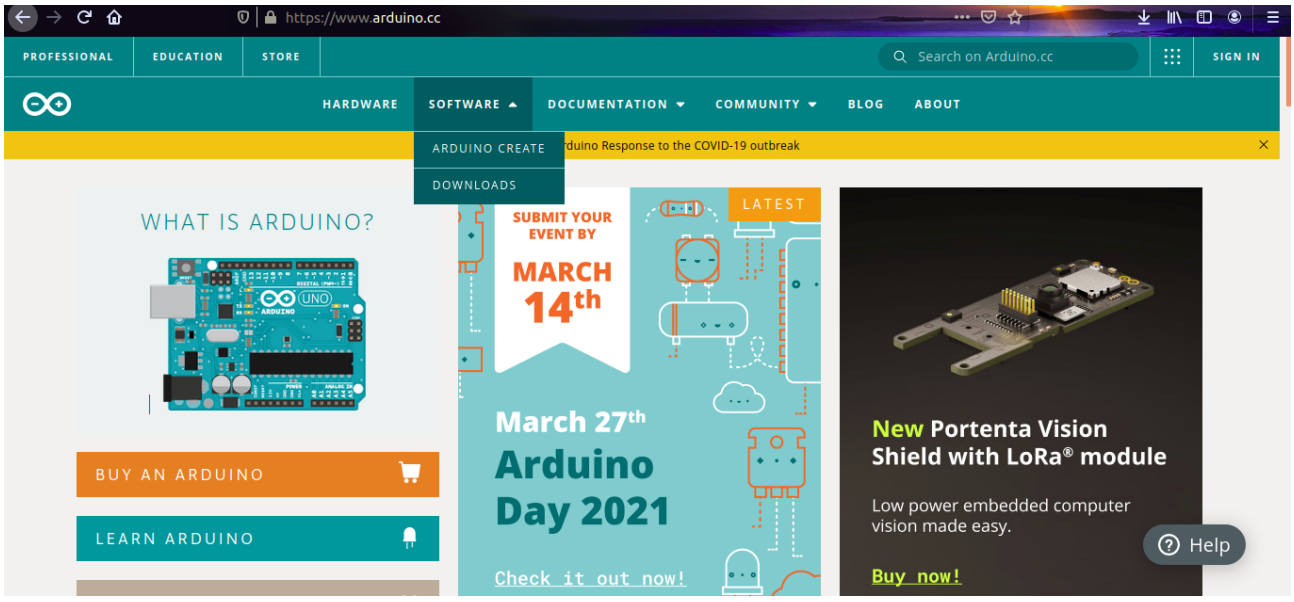
Step 1 :-

Go to https://www.arduino.cc/ then you will see a website like this.



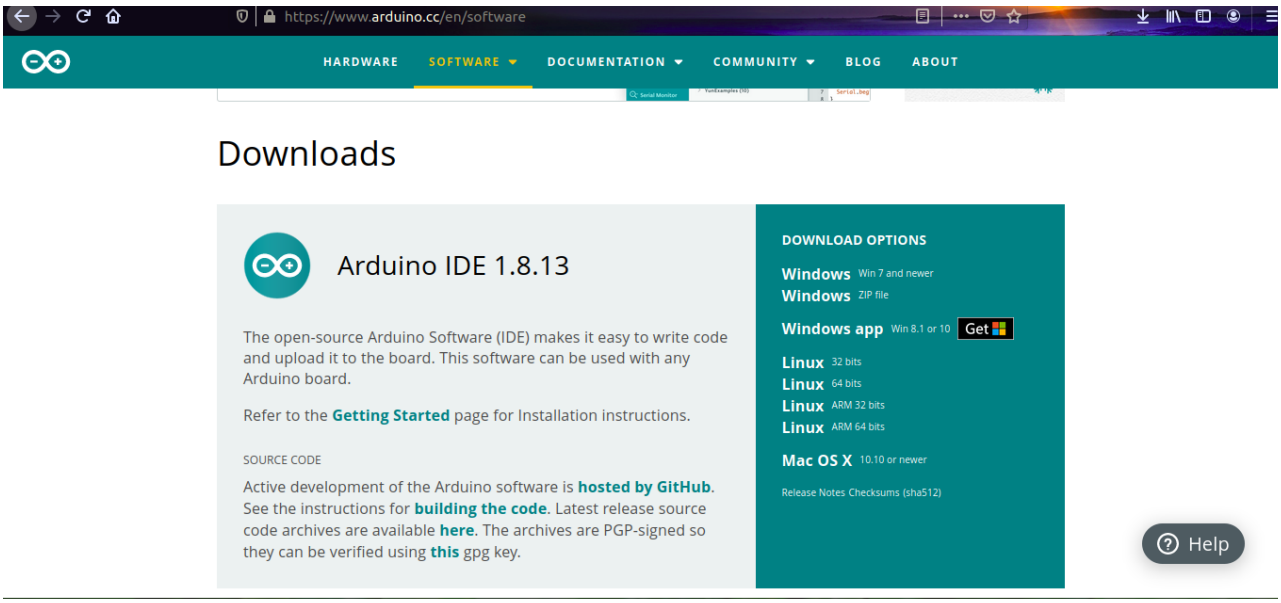
Step 2 :-

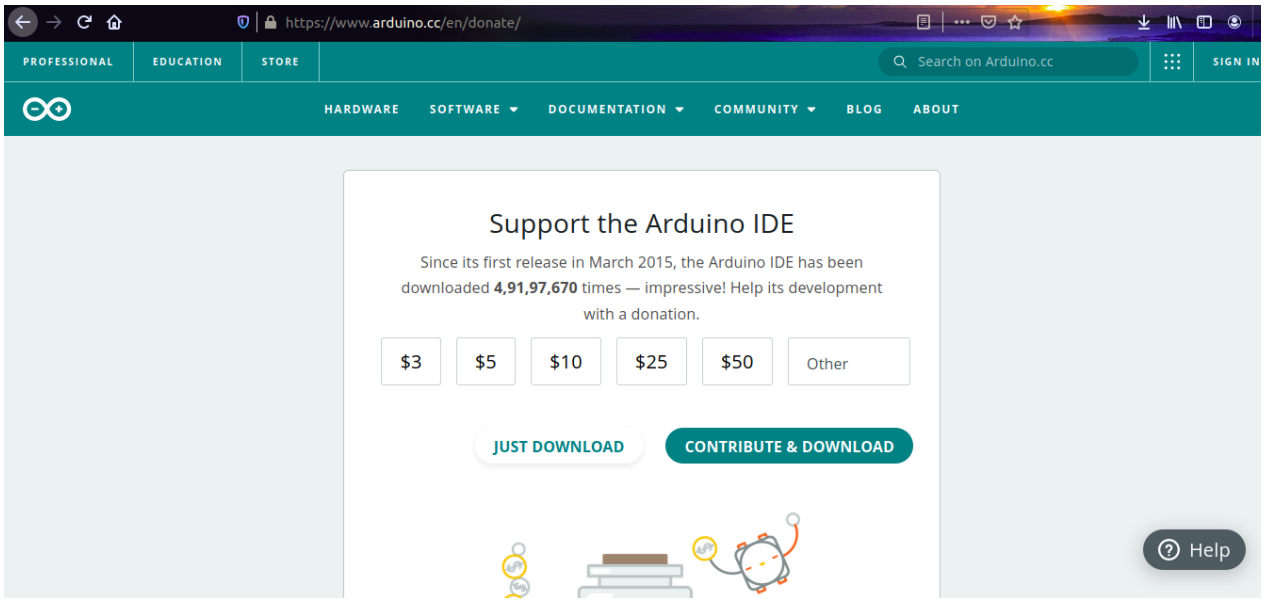
Go to **“SOFTWARE”** option in the menu bar and select the **“DOWNLOADS”** option



Step 3 :-

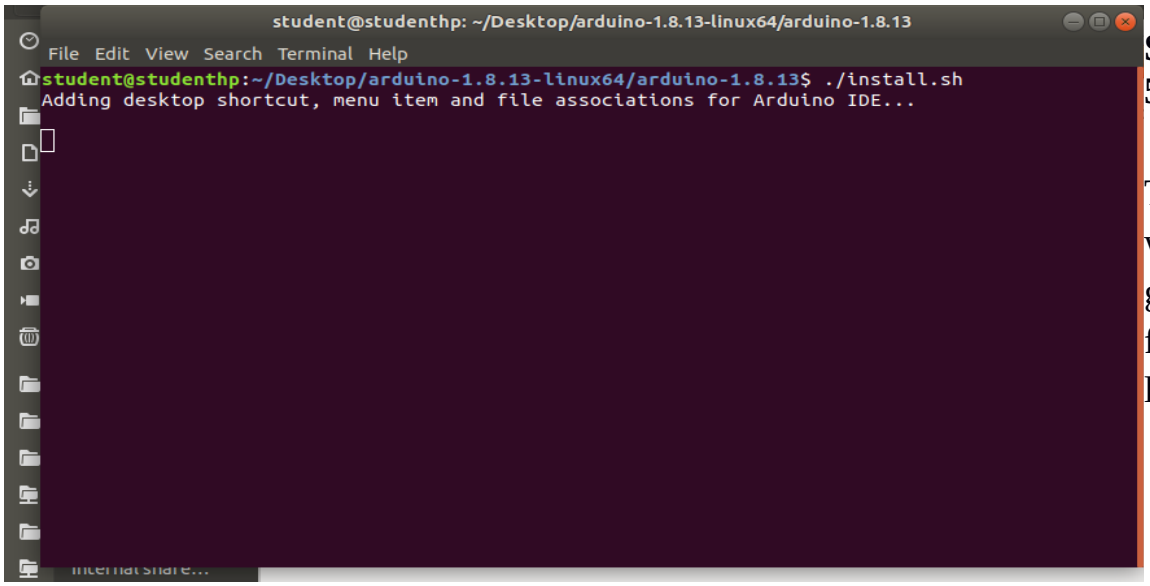
select the “**Linux 64bits”** option and click on just download then extract the file and open it.





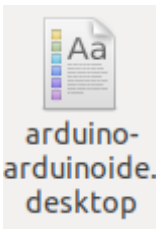
Step 4:-

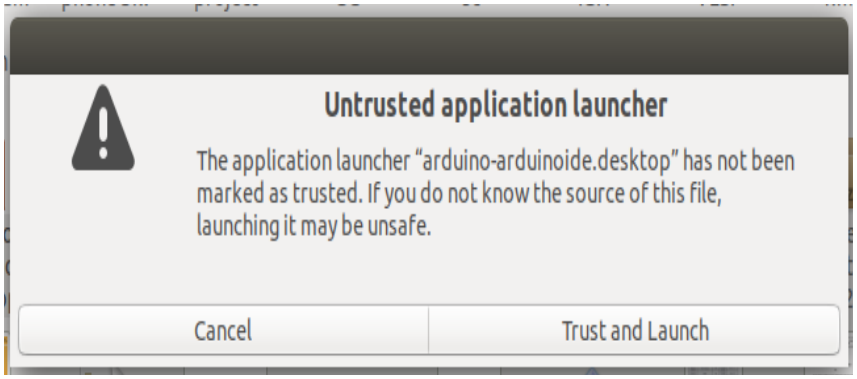
Open the extracted file in the terminal (open the extracted file and the give a right click and select “**open in terminal** ”option.)Then type “**./install.sh”**

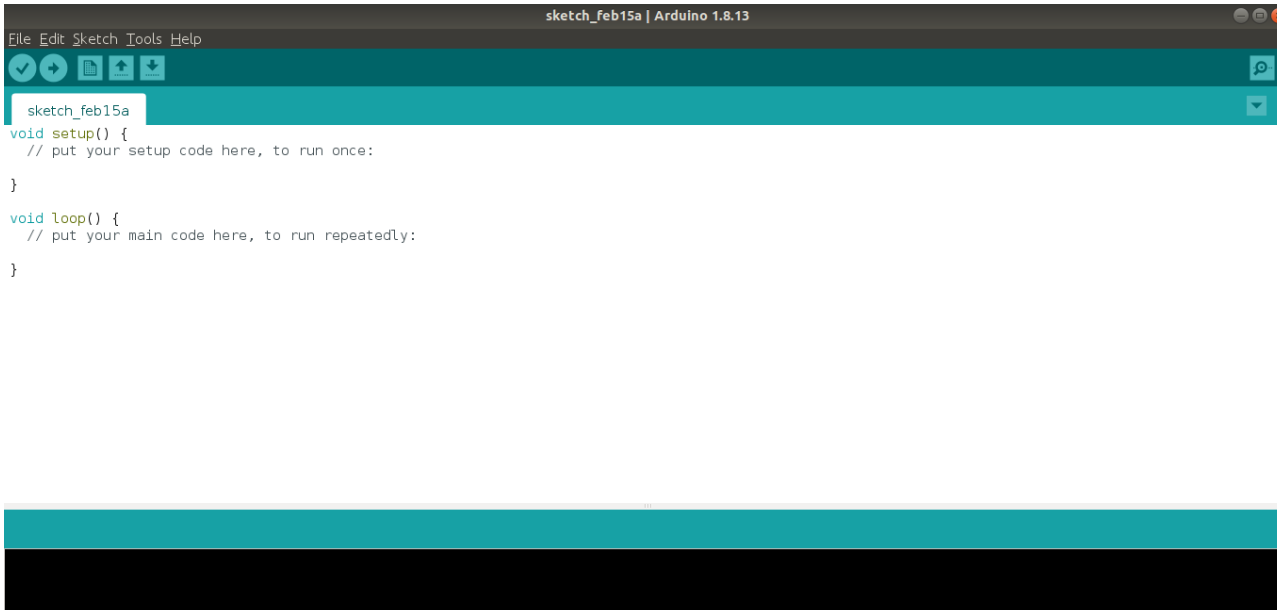


Step 5:-

Then we get a file like shown below in the figure.Select the **"Trust and Launch"** option.Then the software will be opened if we click it.





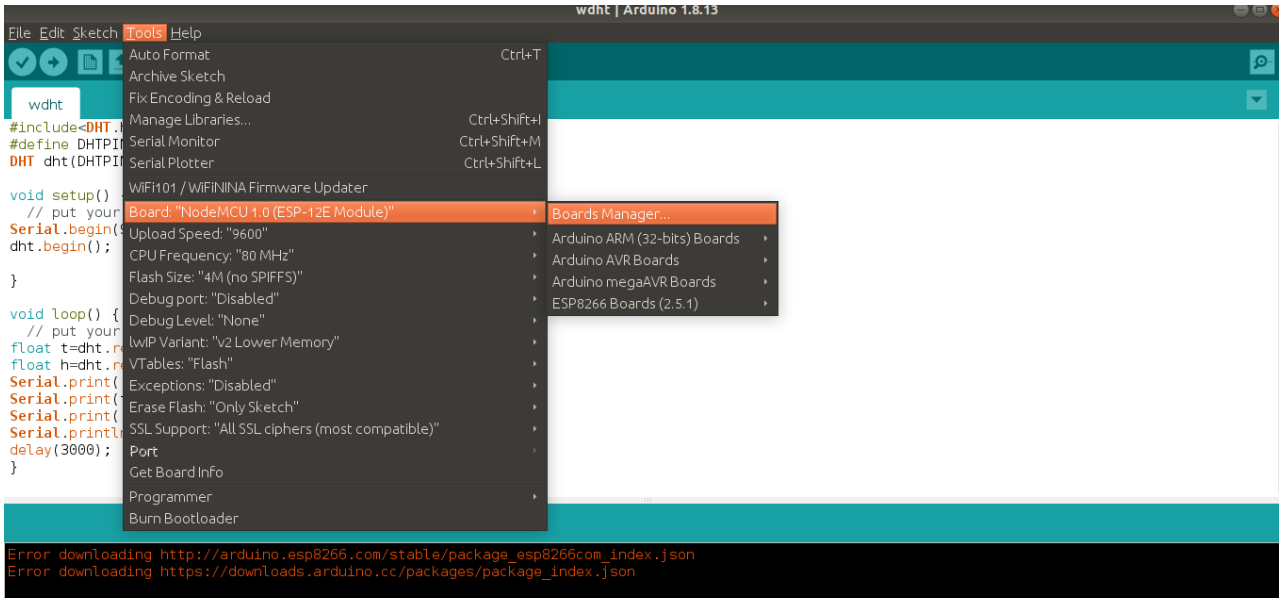


Installing Boards in the arduino software

Steps involved in the process:-

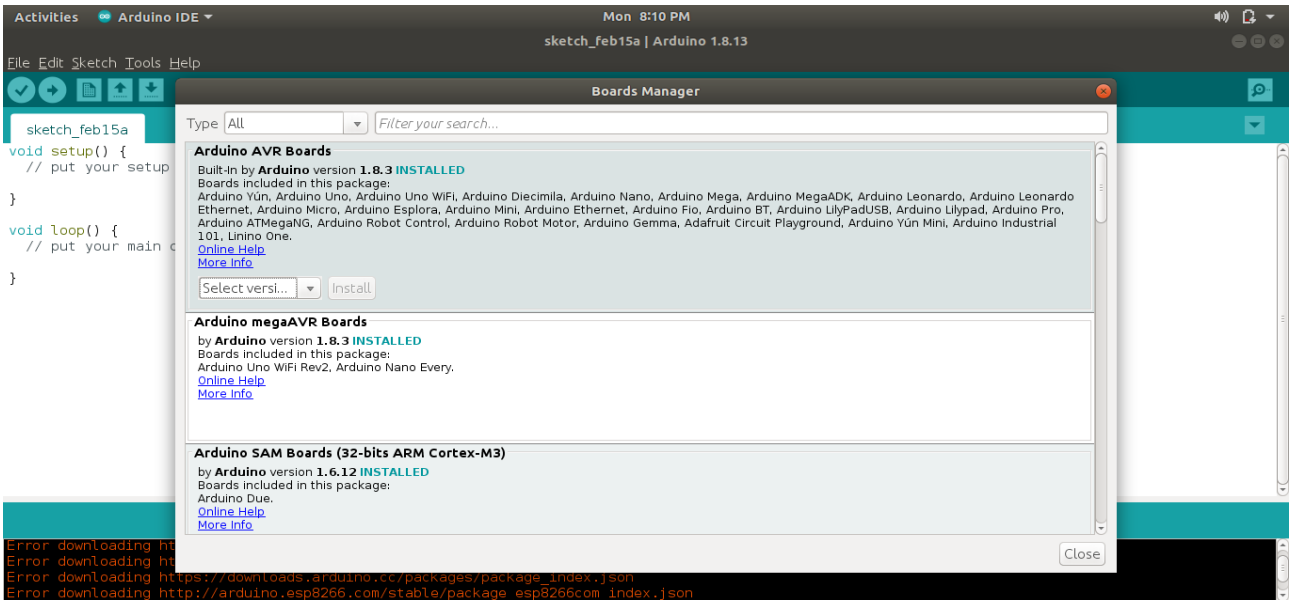
Step 1:-

Select the required options as shown in figure.



Step2:-

Then select the **“1.83”**version at **“select version ”**drop down symbol.Then install it with sufficient Data network.

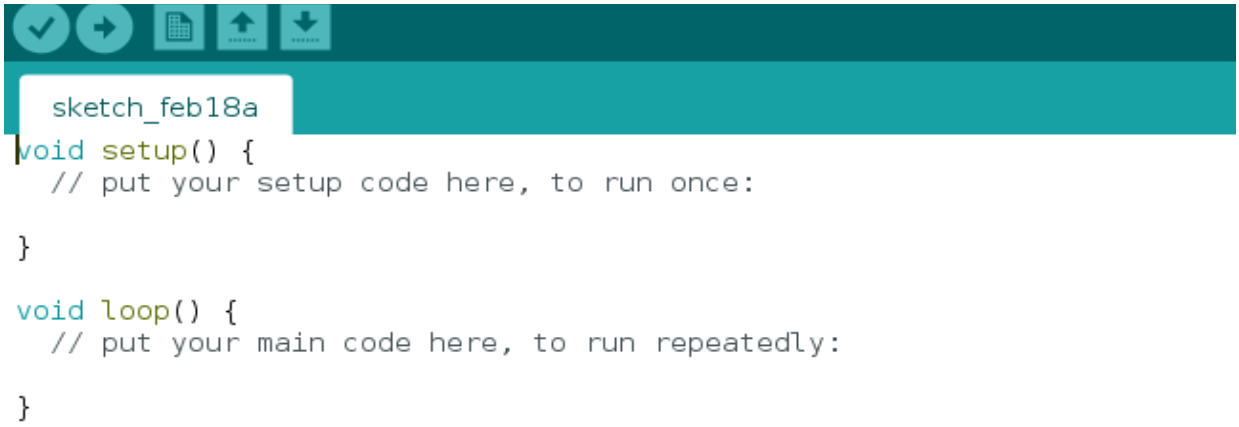


We install boards by these steps.

Useful Functions

Before going to know about the functions ,there are some prerequistes to know. Those are

1. The internal time in an arduino is 1milli second if we apply delay of 1000 then the delay is 1 sec.
2. There are two functions in the arduino software if you observe the picture below.



There are two required functions in an Arduino sketch, setup() and loop(). Other functions must be created outside the brackets of those two functions.

3. Those are **"void setup()"** loop and "**void loop()"** .Setup is used when we need to execute the code only for a single time .
4. Loop function is used to execute the code continuously.

Let us see the functions useful to our project.

1) Define function

#define is a useful C++ component that allows the programmer to give a name to a constant value before the program is compiled. Defined constants in arduino don't take up any program memory space on the chip. The compiler will replace references to these constants with the defined value at compile time.This can have some unwanted side effects though, if for example, a constant name that had been #defined is included in some other constant or variable name. In that case the text would be replaced by the #defined number (or text).

In general, the const keyword is preferred for defining constants and should be used instead of #define.

Syntax :- #define constantName value

Parameters :-constantName: the name of the macro to define.
value: the value to assign to the macro.

Example Code

#define ledPin 3 // The compiler will replace any mention of ledPin with the value 3 at compile time.
Notes and Warnings :-There is no semicolon after the #define statement. If you include one, the compiler will throw cryptic errors further down the page.

#define ledPin 3; // this is an error

Similarly, including an equal sign after the #define statement will also generate a cryptic compiler error further down the page.

#define ledPin = 3 // this is also an error

2) SoftwareSerial(rxPin, txPin, inverse_logic)

Description

SoftwareSerial is used to create an instance of a SoftwareSerial object, whose name you need to provide as in the example below. The inverse_logic argument is optional and defaults to false. See below for more details about what it does. Multiple SoftwareSerial objects may be created, however only one can be active at a given moment.

You need to call SoftwareSerial.begin() to enable communication.

Parameters

rxPin: the pin on which to receive serial data

txPin: the pin on which to transmit serial data

inverse_logic: is used to invert the sense of incoming bits (the default is normal logic). If set, SoftwareSerial treats a LOW (0 volts on the pin, normally) on the Rx pin as a 1-bit (the idle state) and a HIGH (5 volts on the pin, normally) as a 0-bit. It also affects the way that it writes to the Tx pin. Default value is false.

Warning: You should not connect devices which output serial data outside the range that the Arduino can handle, normally 0V to 5V, for a board running at 5V, and 0V to 3.3V for a board running at 3.3V.

3) UINT8_T :-uint8_t is the same as a byte.

its shorthand for: a type of unsigned integer of length 8 bits

4) Serial.println()

Description

Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\ n'). This command takes the same forms as Serial.print().

Syntax

Serial.println(val)
Serial.println(val, format)

Parameters

val: the value to print.

Allowed data types: any data type.
format: specifies the number base (for integral data types) or number of decimal places (for floating point types).

Returns

println() returns the number of bytes written,though reading that number is optional.Data type :size t.

5) While

Description

A while loop will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. Something must change the tested variable, or the while loop will never exit. This could be in your code, such as an incremented variable, or an external condition, such as testing a sensor.

Syntax while (condition) {

```
  // statement(s)
}
```

condition: a boolean expression that evaluates to true or false.

Example Code

```
var = 0;
while (var < 200) {
  // do something repetitive 200 times
  var++;
}
```

6) Serial.available()

Description

Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes).Serial.available() inherits from the Stream utility class.

Syntax :-*Serial*.available()

Parameters :-*Serial*: serial port object**.**

Returns:-The number of bytes available to read**.**

7) Serial.parseInt()

Description

Looks for the next valid integer in the incoming serial. The function terminates if it times out.

Serial.parseInt() inherits from the Stream utility class.In particular:

1. Parsing stops when no characters have been read for a configurable time-out value, or a non-digit is read;
2. If no valid digits were read when the time-out (see Serial.setTimeout()) occurs, 0 is returned;

Syntax

Serial.parseInt()
Serial.parseInt(lookahead)
Serial.parseInt(lookahead, ignore)

Parameters

Serial: serial port object.
lookahead: the mode used to look ahead in the stream for an integer. Allowed data types: LookaheadMode. Allowed lookahead values:

1. SKIP_ALL: all characters other than digits or a minus sign are ignored when scanning the stream for an integer. This is the default mode.
2. SKIP_NONE: Nothing is skipped, and the stream is not touched unless the first waiting character is valid.
3. SKIP_WHITESPACE: Only tabs, spaces, line feeds, and carriage returns are skipped.

ignore: used to skip the indicated char in the search. Used for example to skip thousands divider. Allowed data types: char

Returns :-The next valid integer with data type ‘long’

8) Return

Description

Terminate a function and return a value from a function to the calling function, if desired.

Syntax

return;
return value;

Parameters :-value: Allowed data types: any variable or constant type.

9) Serial.print()

Description

Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as is. For example-

Serial.print(78) gives "78"

- *Serial*.print(1.23456) gives "1.23"
- *Serial*.print('N') gives "N"
- *Serial*.print("Hello world.") gives "Hello world."

An optional second parameter specifies the base (format) to use; permitted values are BIN(binary, or base 2), OCT(octal, or base 8), DEC(decimal, or base 10), HEX(hexadecimal, or base 16). For floating point numbers, this parameter specifies the number of decimal places to use. For example-

- *Serial*.print(78, BIN) gives "1001110"
- *Serial*.print(78, OCT) gives "116"
- *Serial*.print(78, DEC) gives "78"
- *Serial*.print(78, HEX) gives "4E"
- *Serial*.print(1.23456, 0) gives "1"

- *Serial*.print(1.23456, 2) gives "1.23"
- *Serial*.print(1.23456, 4) gives "1.2345"

You can pass flash-memory based strings to *Serial*.print() by wrapping them with F(). For example:

Serial.print(F("Hello World"))

To send data without conversion to its representation as characters, use Serial.write()

Syntax

Serial.print(val)
Serial.print(val, format)

Parameters

Serial: serial port object.
val: the value to print. Allowed dataReturns

print() returns the number of bytes written, though reading that number is optional. Data type: size_t.

types: any data type.

10) PinMode()

Description

Configures the specified pin to behave either as an input or an output.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

Syntax:-pinMode(pin, mode)

Parameters

pin: the Arduino pin number to set the mode of.
mode: INPUT, OUTPUT, or INPUT_PULLUP.

Returns:-Nothing

Notes and Warnings:-The analog input pins can be used as digital pins, referred to as A0, A1, etc.

11) DigitalWrite()

Description

Write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin. It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor.

If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim. Without explicitly setting pinMode(), digitalWrite() will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

Syntax:-digitalWrite(pin, value)

Parameters

pin: the Arduino pin number.
value: HIGH or LOW.

Returns:-Nothing

Notes and Warnings

The analog input pins can be used as digital pins, referred to as A0, A1, etc. The exception is the Arduino Nano, Pro Mini, and Mini's A6 and A7 pins, which can only be used as analog inputs.

12) AnalogWrite()

Description

Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite()) on the same pin.

Board	PWM Pins	PWM Frequency
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev2, Nano Every	3, 5, 6, 9, 10	976 Hz
MKR boards *	0 - 8, 10, A3, A4	732 Hz
MKR1000 WiFi *	0 - 8, 10, 11, A3, A4	732 Hz
Zero *	3 - 13, A0, A1	732 Hz
Nano 33 IoT *	2, 3, 5, 6, 9 - 12, A2, A3, A5	732 Hz
Nano 33 BLE/BLE Sense	1 - 13, A0 - A7	500 Hz
Due **	2-13	1000 Hz
101	3, 5, 6, 9	pins 3 and 9: 490 Hz, pins 5 and 6: 980 Hz

* In addition to PWM capabilities on the pins noted above, the MKR, Nano 33 IoT, and Zero boards have true analog output when using analogWrite() on the DAC0 (A0)pin.
* In addition to PWM capabilities on the pins noted above, the Due has true analog output when using analogWrite() on pins DAC0 and DAC1.

You do not need to call pinMode() to set the pin as an output before calling analogWrite().
The analogWrite function has nothing to do with the analog pins or the analogRead function.

Syntax:-analogWrite(pin, value)

Parameters

pin: the Arduino pin to write to.

Allowed data types: int.
value: the duty cycle: between 0 (always off) and 255 (always on).
Allowed data types: int.

Returns :-Nothing

Notes and Warnings

The PWM outputs generated on pins 5 and 6 will have higher-than-expected duty cycles. This is because of interactions with the millis() and delay() functions, which share the same internal timer used to generate those PWM outputs. This will be noticed mostly on low duty-cycle settings (e.g. 0 - 10) and may result in a value of 0 not fully turning off the output on pins 5 and 6.

13) getTemplateCount()

uint8_t Adafruit_Fingerprint::getTemplateCount (void)

Ask the sensor for the number of templates stored in memory. The number is stored in **templateCount** on success.

Returns
FINGERPRINT_OK on success
FINGERPRINT_PACKETRECIEVEERR on communication error

Useful Devices

- Adafruit R307 fingerprint sensor
- Single channel relay
- Linear solenoid 12vdc

Adafruit R307 fingerprint sensor

Introduction:

Ada fingerprint module is a finger print sensor with TTL UART interface.The user can store the fingerprint data in the module and can configure it in 1:1 or1:n mode for identifying the person.The FP module can directly interface with 3.3 or 5v microcontroller.

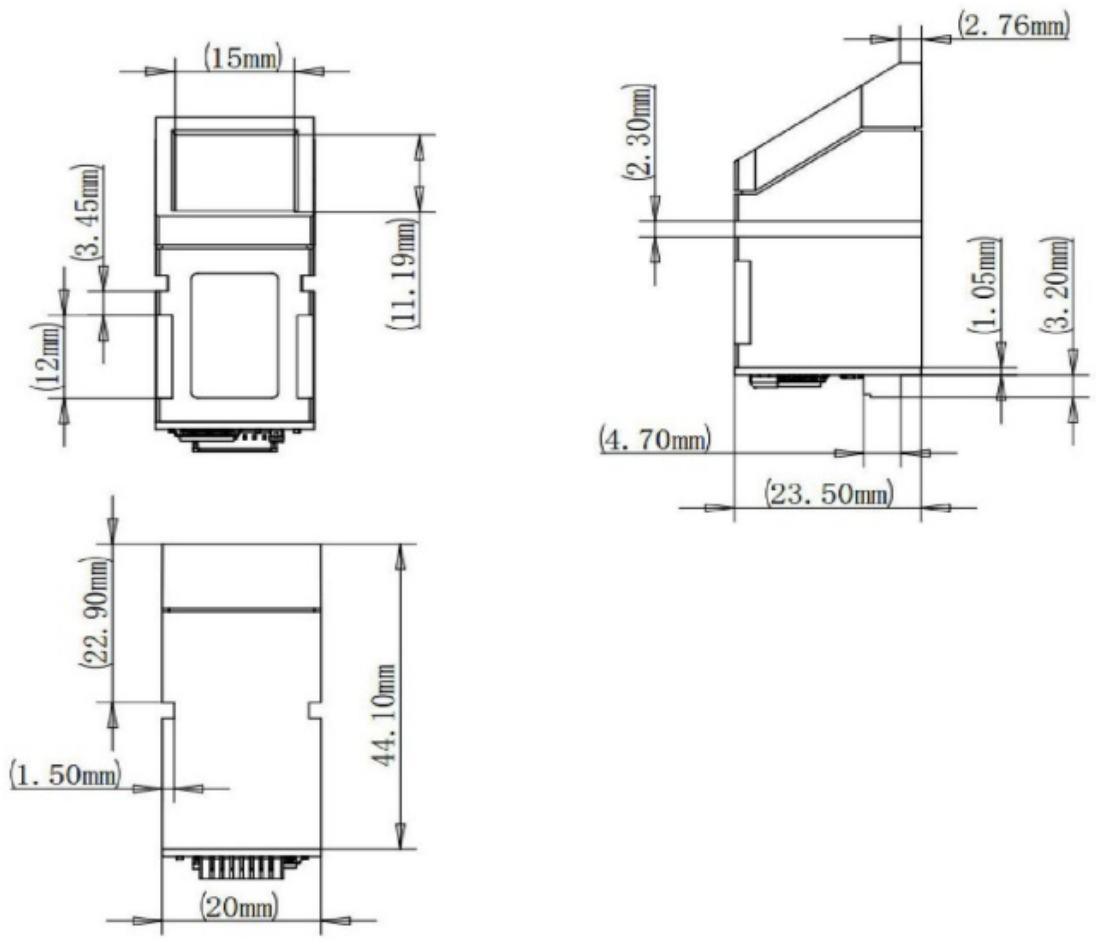
The adafruit R307 fingerprint sensor, secure your project with biometrics this all in one optical fingerprint sensor will make adding fingerprint detection and verification super simple.These modules are typically used in safes there's a high powered DSP chip that does the image rendering,calculation,feature-finding and

searching.It uses UART interface to communicate with external devices.Connect to any microcontroller or system with TTL serial,and send packets of data to take photos,detect prints,hash and search.You can also enroll new fingures directly up to 162 finger prints can be stored in the onboard flash memory**.**

Features:

- ➔ Supply voltage:3.6-6.0vdc
- ➔ Operating current:120m
- ➔ peak current:150mA max
- ➔ Fingerprint imaging time:<1.0seconds
- ➔ Window area:14mm x 18mm
- ➔ signature file: 256 bytes
- ➔ template file:512 bytes
- ➔ storage capacity:162 templates
- ➔ safety ratings (1-5 low to high safety)
- ➔ false reject rate:<0.001%
- ➔ false reject rate:<1.0%
- ➔ interface:TTL serial
- ➔ Baud rate:9600,19200,28800,38400,57600(default is 57600)
- ➔ working temperature rating:-20c to +50c
- ➔ working humidy:40%-85%RH
- ➔ full dimensions:56x20x21.5mm
- ➔ Exposed dimensions:21mm x21mmx21mm triangular
- ➔ weight:20 grams

Dimensions:



Uses:

- ➔ You can use fingerprint gestures to launch an app,control music payback,change ringing modes and more.
- ➔ Most new android phones have a fingerprint scanner now.
- ➔ Normally , you should protect access to your device with fingerprint ,passcode or pattern.

Single Channel Relay

Introduction:

Relay is an electromechanical device that uses an electric current to open or close the contacts of a switch.The single channel relay module is much more than just a plain relay,it comprises of components that make switching and connection

easier and act as indicators to show if the module is powered and if the relay is active or not.

Single channel relay module is used to control high voltage,high current load such as motor,solenoid values,lamps and AC loads.

The single channel relay module is a convenient board which can be used to control high voltage,high current load such as motor,solenoid valves,lamps and AC load.It is designed to interface with microcontroller such as arduino,PIC and etc. It also comes with a Led to indicate the status of relay.Components present on a single channel relay module is 5v relay,transistor,diode,LEDs,resistors,male header pins,3-pin screw type terminal connector,etc.

Specifications:

- ➔ Supply voltage-3.75v to 6v
- ➔ Quiescent current:2mA
- ➔ Current when the relay is active:70mA
- ➔ Relay maximum contact voltage-250VAC or 30VDC
- ➔ Relay maximum current-10A

Pin description:

Pin Number	Pin Name	Description
1	Relay Trigger	Input to active the relay
2	Ground	0V reference
3	VCC	Supply input for powering the relay coil
4	Normally open	Normally open terminal of the relay
5	Common	Common terminal of the relay
6	Normally closed	Normally closed contact of the relay

Uses:

- ➔ Triggering the relay operates the normally open or normally closed contacts.
- ➔ It is frequently used in an automatic control circuit.

➔ To put it simply ,it is an automatic switch to control a high-current circuit with a low-current signal.

Applications:

➔ Mains switching

➔ High current switching

➔ Isolated power delivery

➔ Home automation

Limitations:

➔ Relays are bulkier than transistors for switching small currents.

➔ Relay cannot switch rapidly,transistors can switch many times per second.

➔ Relay use more power due to the current flowing through their coil.

Linear solenoid

Introduction:

Another type of electromagnetic actuator that converts an electrical signal into a magnetic field producing a linear motion is called the linear solenoid .A linear solenoid is an electromagnetic device that converts electrical energy into a mechanical pushing or pulling force or motion.They can also be switchied and controlled using bipolar transistor or MOSFET's.Linear solenoid's basically consist of an electrical coil wound around a cylindrical tube with a ferro magnetic actuator or plunger that is free to move or slide IN and OUT of the coils body.

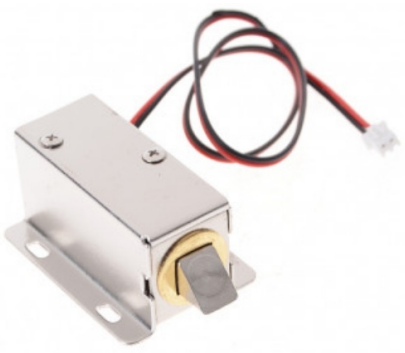
Linear solenoids are available in two basic configurations called a pull- type as it pulls the connected load towards itself when energised,and the push -type that act in the opposite direction pushing it away from itself when energised.Both push and pull types are generally constructed the same with the difference being in the location of the return spring and design of the plunger.

Solenoid converts electrical network into mechanical work.

Characteristics:

➔ Stoke up to 30mm(depending on type).

➔ Holding force up to 2,000N(in end position,energized).



➔ Can be used as push-type or pull-type linear solenoid.

➔ High lifetime:up to 50 million cycles and more.

➔ Fast response time:switching time<5ms possible(depending on size).

➔ Standara sizes from 11mm to 874mm diameter.

Limitations:

➔ They have only two positions:fully open and fully closed.

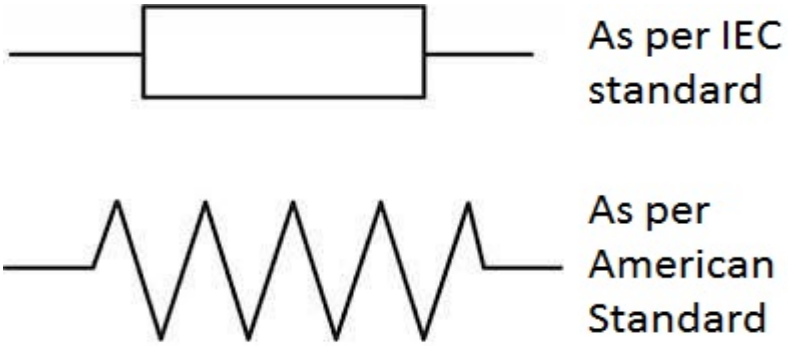
➔ They don’t produce much force,so they usually only operate relatively small valves.

Components

***Resistors*:**

A resistor is a passive electrical component with the primary function to limit the flow of electric current.The SI unit of the resistor is Ohm.

The standard symbols are:



Types of resistors:

1.CarbonComposition Resistor
2.Thermistor
3.Wire Wound
4.Resistor
5.Metal Firm Resistor
6.Corbon Firm Resistor
7.Variable Resistor
8.Varistor
9. Light Dependent Resistor

Functions of a resistor:

A resistor is responsible for dissipating power in the form of heat.

In electronics, a resister is often used to control the current. Resistors are needed to prevent short circuits.

Advantages:

1. Resistors are very small.Hence,it is very is easy to carry them from one place to another place.
2. Resistors are very cheap.Hence,it is easy to replace them.

Disadvantages:

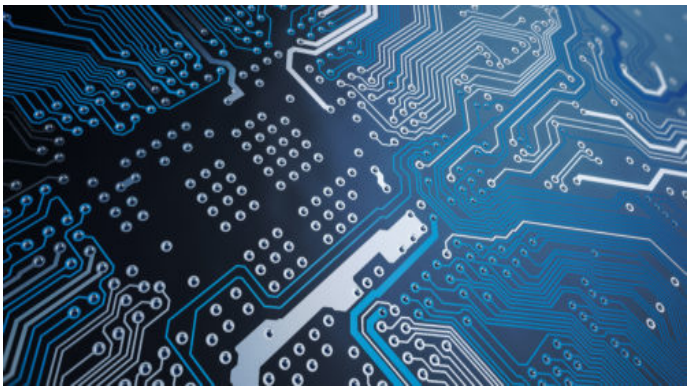
1. Resistors with high resistance will oppose large amount of electrice current.Hence large amoutn of energy is wasted in form of heat.

Printed Circuit Board:(PCB)

PCB's are widely used in elctronics in order to mount and creat circuits on it. A PCB is use to incorrect all the electrical and electronic components together on a common board. In PCB all the components are connected internally so wire are not used.

TYPES OF PCB:

There are several types of PCB available for the circuite.Out of these type of PCB, we have to choose the appropriate type of PCB according to our applications.



1.Singel-layer PCB
2.Double-layer PCB
3.Multi-layer PCB
4.Flexible PCB
5.Aluminium back PCB
6.Flex-rigid PCB

Funtions of PCB:

PCB is used to mechanically support and electrically connect electronic components using conductive pathways, tracks or signal traces.

Advantages:

1. Are cost-effective and highly reliable.
2. Economical for high volume production.
3. Have color codes for different connections and are therefore easy to install.

Disadvantages:

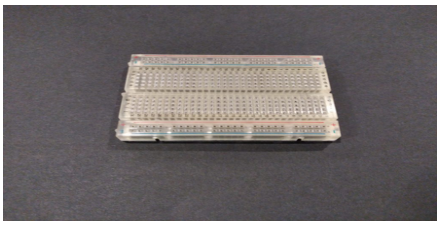
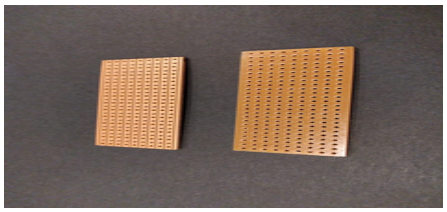
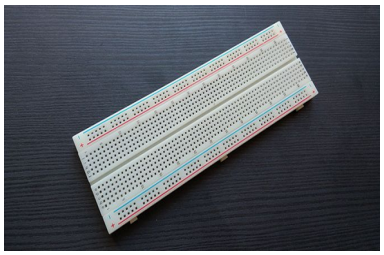
1. More layout effort.
2. Can not be updated once it gets printed.
3. Etching of circuit boards generates chimecals which are dangerous for environment.

Bread Board:

A breadboard is a simple device designed to let you creat circuits without the need of soldering.They come in various sizes and the design can vary.Thin plastic board used to hold the electronic components(transistor,resistors,chips,etc..).

Types of breadboard:

1. Solderless
2. Solder

Advantages:

1. Easy to adjust
2. It is flexible
3. No drilling holes
4. It can be debugges easily
5. cheap and connections can be changed.

Disadvantages:

1. It is unreliable
2. Difficult to replicate
3. It is temporary
4. Not good for high applications
5. It works very poorly for high-speed design.

Jumperwires:

Jumper wires are simply wires that have connecter pins to each end,allowing them to be used to connect two points to each other without soldering.Jumper wire are typically used with breadboard and other protyping tools inorder to make it easy to change a circuit as needed.

Types:

Jumper wires are typically three versions:

1. Male-to-male
2. Male-to female.
3. Female-to female



Universal Serial Bus:

It is an industry standard that establishes specifications for cables,connectors and protocols for connection and supply between computers, peripherals and other computers.

Types of USB:

1. USB-A
2. USB-B
3. Mini-USB
4. Micro-USB
5. USB-C
6. USB-3


Functions:

The main objectives of these cabels is to effectively,quickly and properly gather or transfer the data from one device to another.

Advantages:

1. Portability
2. Storage capacity
3. Read and Write
4. Compatible
5. Durable
6. Data security.

Disadvantages:

1. Easy to loss
2. Rewrite limitation
3. Corruption Issue.

LED:

A light -emmiting diode is a semiconductur device that emits light when an electric current is passed throught it.Light is produced when the particles that carry the current combine together within the semiconducture.

Types:

1. Dimmer Switches
2. LED Lighting tubes
3. SMD LED
4. COB LED
5. Graphene Light.

Working principle:Electroluminescence.

Pin configuration: Anode and Cathode.

Advantages:

1. Long life
2. Energy efficiency
3. High brightness and intensity
4. Exceptional color change
5. Low radiated heat
6. Reliability

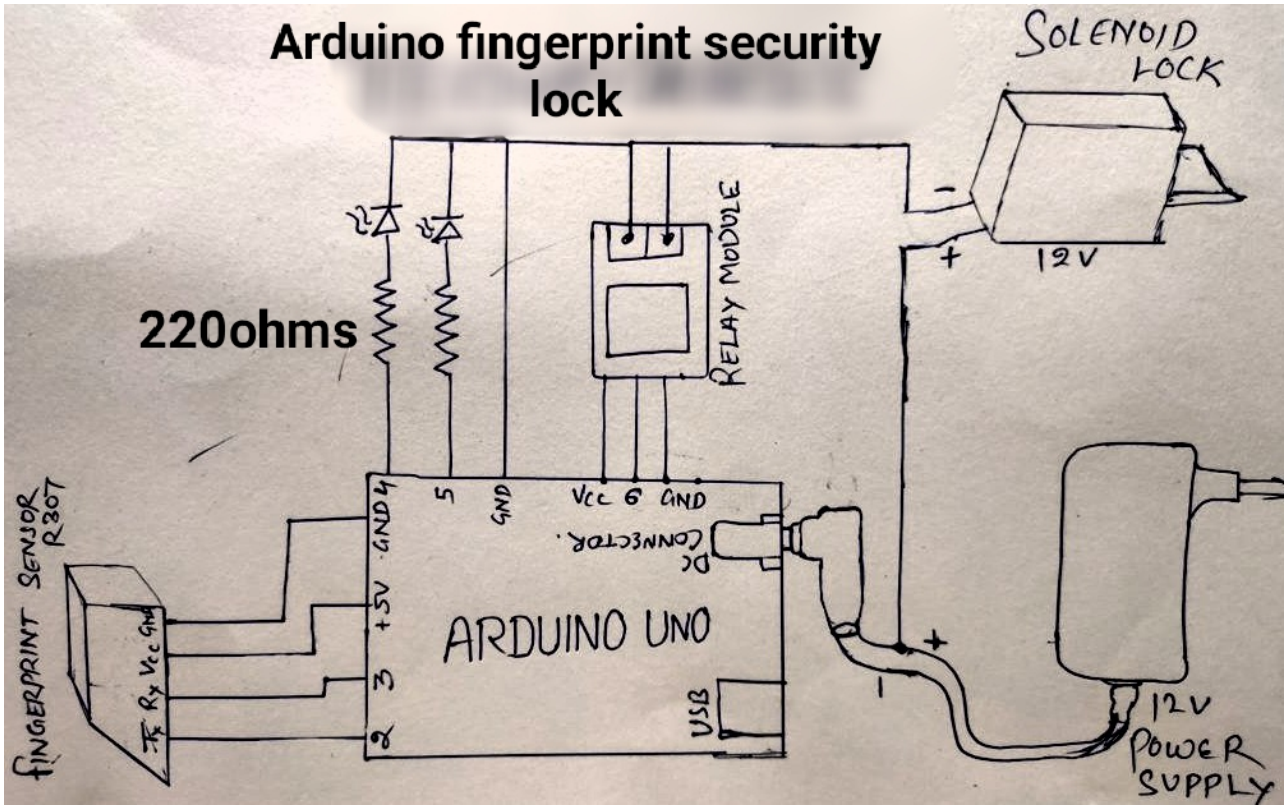
Disadvantages:

1. High up-front costs
2. Transformer capability

3. Overheating can cause reduced lamp life
4. Potential color shift over lamp life.

Circuit of Arduino Fingerprint Sensor Lock

12V power is the main source of energy supply required for this system, which is given to the V_{IN} pin of arduino board. The solenoid electric lock itself consumes 12V supply, however Arduino microcontroller (MCU) requires only 5V which can be easily supplied from the inbuilt 5V regulator from the Arduino Uno Board. And, the other common 12V supply is externally supplied to the system.



Implementation

(a). Arduino Uno MCU board Arduino Fingerprint Sensor Lock

Arduino Uno MCU board is acts like a CPU of the system "Arduino Fingerprint Sensor Lock". This board comprises multiple features. There are 14 digital input/output (I/O) pins, six analogue inputs, 32k flash memory, 16MHz crystal oscillator, a USB connection, power jack, ICSP header and reset button. We can use any of its features through Arduino IDE software through proper programming.

(b). Fingerprint sensor module Arduino Fingerprint Sensor Lock The $R_X$and $T_X$pin of fingerprint sensor module R305 is connected across D_3 and D_2pin of arduino board respectively. Since this module is constructed using UART

technology, it is easy to interface sensor directly with the MCU or also to the PC using max232/USB serial adaptor. The information collected from the fingerprint can be collected in the module. During the process of identification, the data can be configured in either 1:1 or 1:N module. In order to ensure serial communication, two pins of R305 sensor; TX and RX are connected across digital pins 2 and 3 of Arduino Uno.

- The V_{cc} of the fingerprint sensor R307 is connected to the +5v pin of arduino uno R3 board.The ground pin of fingerprint sensor R307 is connected to the GND pin of arduino uno R3 board.
- 12V power supply is connected to the DC connector port in arduino uno board .We can also supply the power through USB cable also.
- Relay Module output connected to D_6 pin of arduino uno R3 board.The V_{cc} and Gnd of relay module connected to V_{cc} and Gnd pins of arduino board respectively.
- Resistors 220Ω is connected with LED in series with it .Two 220Ω resistors with two leds connected in parallel with each other and the two resistors are connected to D_4 and D_5 pins respectively and their negative terminals are connected to GND of arduino board.
- 12V solenoid positive terminal is connected to the 12V power supply and negative terminal is connected to other side of relay module.Now the connections are complete.

Precautions

1. Make sure that circuit connections are correct.
2. We should not supply high voltages to avoid damage
3. Avoid loose connections.
4. Check out the connections before switching on power supply.
5. Before using components ,check whether the components are working properly or not.

Code Implementation

By using the functions discussed earlier we implement the code.We require two types of codes one is to enroll fingerprint and the other is to verify the fingerprint.

Fingerprint enroll code:-

These displays use TTL Serial to communicate, 2 pins are required to interface

For UNO and others without hardware serial, we must use software serial.

pin #2 is IN from sensor (GREEN wire) and pin #3 is OUT from arduino (WHITE wire)

```
#include <Adafruit_Fingerprint.h>

#define mySerial Serial1


// comment these two lines if using hardware serial

SoftwareSerial mySerial(2, 3);

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

uint8_t id;

void setup()

{

Serial.begin(9600);

while (!Serial); // For Yun/Leo/Micro/Zero/...

delay(100);

Serial.println("\n\nAdafruit Fingerprint sensor enrollment");

// set the data rate for the sensor serial port

finger.begin(57600);

if (finger.verifyPassword()) {

Serial.println("Found fingerprint sensor!");

} else {

Serial.println("Did not find fingerprint sensor :(");

while (1) { delay(1); }

}

}

uint8_t readnumber(void) {
```

```
uint8_t num = 0;
while (num == 0) {
while (! Serial.available());
num = Serial.parseInt();
}
return num;
}

void loop()
{
Serial.println("Ready to enroll a fingerprint!");
Serial.println("Please type in the ID # (from 1 to 127) you want to save this finger as...");
id = readnumber();
if (id == 0) {// ID #0 not allowed, try again!
return;
}
Serial.print("Enrolling ID #");
Serial.println(id);
while (! getFingerprintEnroll() );
}
uint8_t getFingerprintEnroll() {
int p = -1;
Serial.print("Waiting for valid finger to enroll as #");
Serial.println(id);
while (p != FINGERPRINT_OK) {
p = finger.getImage();
switch (p) {
```

```
case FINGERPRINT_OK:
Serial.println("Image taken");
break;
case FINGERPRINT_NOFINGER:
Serial.println(".");
break;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
break;
case FINGERPRINT_IMAGEFAIL:
Serial.println("Imaging error");
break;
default:
Serial.println("Unknown error");
break;
}
}
// OK success!
p = finger.image2Tz(1);
switch (p) {
case FINGERPRINT_OK:
Serial.println("Image converted");
break;
case FINGERPRINT_IMAGEMESS:
Serial.println("Image too messy");
return p;
case FINGERPRINT_PACKETRECIEVEERR:
```

```
Serial.println("Communication error");
return p;
case FINGERPRINT_FEATUREFAIL:
Serial.println("Could not find fingerprint features");
return p;
case FINGERPRINT_INVALIDIMAGE:
Serial.println("Could not find fingerprint features");
return p;
default:
Serial.println("Unknown error");
return p;
}
Serial.println("Remove finger");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
p = finger.getImage();
}
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
while (p != FINGERPRINT_OK) {
p = finger.getImage();
switch (p) {
case FINGERPRINT_OK:
Serial.println("Image taken");
break;
```

```
case FINGERPRINT_NOFINGER:
Serial.print(".");
break;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
break;
case FINGERPRINT_IMAGEFAIL:
Serial.println("Imaging error");
break;
default:
Serial.println("Unknown error");
break;
}
}
// OK success!
p = finger.image2Tz(2);
switch (p) {
case FINGERPRINT_OK:
Serial.println("Image converted");
break;
case FINGERPRINT_IMAGEMESS:
Serial.println("Image too messy");
return p;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
return p;
case FINGERPRINT_FEATUREFAIL:
```

```
Serial.println("Could not find fingerprint features");
return p;
case FINGERPRINT_INVALIDIMAGE:
Serial.println("Could not find fingerprint features");
return p;
default:
Serial.println("Unknown error");
return p;
}
// OK converted!
Serial.print("Creating model for #"); Serial.println(id);
p = finger.createModel();
if (p == FINGERPRINT_OK) {
Serial.println("Prints matched!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
Serial.println("Communication error");
return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
Serial.println("Fingerprints did not match");
return p;
} else {
Serial.println("Unknown error");
return p;
}
Serial.print("ID "); Serial.println(id);
p = finger.storeModel(id);
if (p == FINGERPRINT_OK) {
```

```
Serial.println("Stored!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
Serial.println("Communication error");
return p;
} else if (p == FINGERPRINT_BADLOCATION) {
Serial.println("Could not store in that location");
return p;
} else if (p == FINGERPRINT_FLASHERR) {
Serial.println("Error writing to flash");
return p;
} else {
Serial.println("Unknown error");
return p;
}
}
```

Here in this code first we establish connection between the fingerprint sensor and the arduino,next we check whether it is ready to enroll the fingerprint or not.It asks for the number for our fingerprint as it is capable of saving 137 fingerprints.After it is ready we enroll the fingerprint and wait for the acknowledgement.In this waiting process it checks whether the fingerprint scanned is clear or not by checking all the necessary conditions for it .Then we get “image taken” or any error occurs.The enrolled fingerprint will be saved.

Fingerprint verify code:-

These displays use TTL Serial to communicate, 2 pins are required to interface For UNO and others without hardware serial, we must use software serial.

Pin #2 is IN from sensor (GREEN wire) andpin #3 is OUT from arduino (WHITE wire)

```
#include <Adafruit_Fingerprint.h>
#define mySerial Serial1
```

```
// comment these two lines if using hardware serial
SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
void setup()
{
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);
pinMode(13, OUTPUT);
digitalWrite(13, HIGH);
digitalWrite(4, HIGH);
Serial.begin(9600);
while (!Serial);
delay(100);
Serial.println("\n\nAdafruit finger detect test");// set the data rate for the sensor serial port
finger.begin(57600);
delay(5);
if (finger.verifyPassword()) {
Serial.println("Found fingerprint sensor!");
} else {
Serial.println("Did not find fingerprint sensor :(");
while (1) { delay(1); }
}
finger.getTemplateCount();
Serial.print("Sensor contains "); Serial.print(finger.templateCount);
Serial.println(" templates");
Serial.println("Waiting for valid finger...");
}
```

```
void loop() // run over and over again
{
getFingerprintIDez();
delay(50); //don't ned to run this at full speed.
}
uint8_t getFingerprintID() {
uint8_t p = finger.getImage();
switch (p) {
case FINGERPRINT_OK:
Serial.println("Image taken");
break;
case FINGERPRINT_NOFINGER:
Serial.println("No finger detected");
return p;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
return p;
case FINGERPRINT_IMAGEFAIL:
Serial.println("Imaging error");
return p;
default:
Serial.println("Unknown error");
return p;
}
// OK success!
p = finger.image2Tz();
switch (p) {
```

```
case FINGERPRINT_OK:
Serial.println("Image converted");
break;
case FINGERPRINT_IMAGEMESS:
Serial.println("Image too messy");
return p;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
return p;
case FINGERPRINT_FEATUREFAIL:
Serial.println("Could not find fingerprint features");
return p;
case FINGERPRINT_INVALIDIMAGE:
Serial.println("Could not find fingerprint features");
return p;
default:
Serial.println("Unknown error");
return p;
}
// OK converted!
p = finger.fingerFastSearch();
if (p == FINGERPRINT_OK) {
Serial.println("Found a print match!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
Serial.println("Communication error");
return p;
} else if (p == FINGERPRINT_NOTFOUND) {
```

```
Serial.println("Did not find a match");
return p;
} else {
Serial.println("Unknown error");
return p;
}
// found a match!
Serial.print("Found ID #"); Serial.print(finger.fingerID);
Serial.print(" with confidence of "); Serial.println(finger.confidence);
return finger.fingerID;
}
// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {
uint8_t p = finger.getImage();
if (p != FINGERPRINT_OK) return -1;
p = finger.image2Tz();
if (p != FINGERPRINT_OK) return -1;
p = finger.fingerFastSearch();
if (p != FINGERPRINT_OK) return -1;
if( finger.fingerID==5 )
{
digitalWrite(13, LOW);
digitalWrite(5, HIGH);
delay(2000);
digitalWrite(13, HIGH);
digitalWrite(5, LOW);
}
```

```
// found a match!
Serial.print("Found ID #"); Serial.print(finger.fingerID);
Serial.print(" with confidence of "); Serial.println(finger.confidence);
return finger.fingerID;
}
```

Here in this code first we have defined output pins then we establish connection with fingerprint sensor.We should give our fingerprint template count for valid fingerprint.Then wait for acknowledgment whether fingerprint is ready to take input or not,If it does we give our fingerprint then wait for complete process of taking a fingerprint.After it has received sucessfully,we search for the matched fingerprint templates given earlier and check it with the present enrolled fingerprint.If it is not matched then we get “not matched”.

Working and Results

Working

Arduino based fingerprint door lock working is very simple and easy. Fingerprint sensor is interfaced with the arduino Uno and first we have to enroll the fingerprint to the fingerprint sensor and saved our fingerprint sensor It save your fingerprint data into the inbuit memory and then we compare this saved file with the every finger scanned on the scanner. for example if i save my finger into the data base then the system will compare every single fingerprint with mine. and if it get the same fingerprint then it will send the command to open the lock otherwise the lock will remain close.

Fingerprint sensor capture the image of the fingerprint and make the pattern inside the memory of the fingerprint. the shape of the pattern will break into the binary code and then save into the memory of the fingerprint. for each and every fingerprint it will save the different pattern according to the fingerprint because as we know we all have the different fingerprint even in our hand each fingerprint having their own unique fingerprint. it will never match the other finger and according to the research the accuracy of the fingerprint is near about the 98% which is good enough to secure any system.

1. 12V Power supply is given to the arduino uno R3 board,from the arduino board power is supplied to fingerprint scanner and single channel relay
2. Leds are used for indication.
3. Whenever we are enrolling the fingerprint both the relay and solenoid will be in off state
4. The relay and solenoid will be in off state until the fingerprint scanned be matched with previous stored templates when they get matched the relay module controls the high current used by solenoid and the solenoid converts the electric current into mechanical work.
5. As a result of this, the solenoid will be unlocked.

Results

Hence we have designed a arduino fingerprint security lock with a good security which can be accessed by either individual or a particular group.It can be implemented in banks for transactions ,for attendance in offices etc.

Conclusion and References

Conclusion

The objective of this research is to create a securable environment in the society to decrease tremendous increase of crime rate. we mainly improve and optimize the user privacy, which can be damaged by unauthorized users.

References

1) Introduction

- ✓ https://techatronic.com/fingerprint-arduino-security-lock-diy/
- ✓ http://bestelectronicprojects.com/

2) Arduino uno R3 and Arduino software

- ✓ https://www.arduino.cc/en/guide/environment#libraries
- ✓ https://www.arduino.cc/en/guide/environment#boards
- ✓ https://www.arduino.cc/en/Tutorial/Bootloader
- ✓ https://www.arduino.cc/en/Hacking/Bootloader

3) Installation of arduino software and boards

- ✓ https://www.arduino.cc/
- ✓ https://www.arduino.cc/en/guide/linux

4) Useful functions

- ✓ https://www.arduino.cc/reference/en/language/structure/further-syntax/define/
- ✓ https://www.arduino.cc/en/Reference/SoftwareSerialConstructor
- ✓ https://forum.arduino.cc/index.php?topic=41590.0
- ✓ https://www.arduino.cc/reference/en/language/functions/communication/serial/println/
- ✓ https://www.arduino.cc/en/Tutorial/BuiltInExamples/WhileStatementConditional
- ✓ https://www.arduino.cc/reference/en/language/structure/control-structure/while/
- ✓ https://www.arduino.cc/reference/en/language/functions/communication/serial/available/
- ✓ https://www.arduino.cc/reference/en/language/functions/communication/serial/parseint/
- ✓ https://www.arduino.cc/reference/en/language/structure/control-structure/return/
- ✓ https://www.arduino.cc/reference/en/language/functions/communication/serial/print/
- ✓ https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/
- ✓ https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/
- ✓ https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/
- ✓ https://adafruit.github.io/Adafruit-Fingerprint-Sensor-Library/html/class_adafruit___fingerprint.html

5) Resistors

- ✓ https://www.electrical4u.com/types-of-resistor/
- ✓ http://electrotopic.com/what-are-the-functions-of-a-resistor/
- ✓ https://www.physics-and-radio-electronics.com/electronic-devices-and-circuits/passive-components/resistors/resistors.html

6) PCB

- ✓ https://circuitdigest.com/tutorial/basics-of-pcb
- ✓ https://medium.com/@chinapcbmanufacturers/advantage-and-disadvantage-of-printed-circuit-board-c276adc68cbb

7) Breadboard

- ✓ https://electronicguidebook.com/what-are-the-two-basic-types-of-breadboard/
- ✓ https://www.ecstuff4u.com/2019/08/advantage-disadvantage-breadboard.html

✓ https://www.muo.com/tag/what-is-breadboard/

8) Jumper wires

✓ http://blog.sparkfuneducation.com/what-is-jumper-wire

9) Usb

✓ https://en.wikipedia.org/wiki/USB
✓ https://www.l-com.com/frequently-asked-questions/what-is-a-usb-cable https://www.samsung.com/uk/support/mobile-devices/what-are-the-different-types-of-usb-cables/https://bytebitebit.com/1137/usb-flash-drives-advantages-disadvantages/

10) Led

✓ https://www.ledsmagazine.com/leds-ssl-design/materials/article/16701292/what-is-an-led
✓ https://www.renewableenergyhub.co.uk/main/led-lighting-information/types-of-led-lighting/
✓ https://www.thelightbulb.co.uk/resources/ultimate-guide-led-lights-advantages-leds/

11) Code implementation

✓ http://www.adafruit.com/products/751

✓ https://techatronic.com/fingerprint-arduino-security-lock-diy/

12) Working

✓ https://techatronic.com/fingerprint-arduino-security-lock-diy/