



AKHBAR ELYOM ACADEMY



Automated & Smart Home
[Arduino, Raspberry Pi, Android]

Dr/ Shereen Aly Taie



SMART HOME

Team work:

- ABD ELRAHMAN ABD ELATY
- MAHMOUD ABDALLAH NOBY
- OMAR GAMAL
- MOSTAFA HESHAM
- ISLAM ALI
- AHMED HESHAM ELZINY
- AYA AHMED
- BASSANT MOHAMED
- NOHA MOHAMED

Under supervised by:

Dr.Shereen Taie

{Table of content}

Acknowledgements	6
Abstract	7
CHAPTER 1 INTRODUCTION	8
1.1 What is embedded systems.....	9
1.2 Design.....	10
1.3 The reason for choosing this project.....	13
1.4 Project definition.....	14
1.5 Recent home's technologies.....	15
1.6 Statistics about smart home's over the world.....	18
CHAPTER 2 IOT APPLICATION	24
2.1 What is IOT.....	25
2.2 ESP-WIFI NODE-MCU module.....	26
2.3 IR (infrared) transfer sensor.....	33
2.4 TSOP IR Receiver.....	36
2.5 DHT11 Sensor temperature.....	41
2.6 OLED LCD.....	46
2.7 Fingerprint Recognition.....	49
2.8 Solenoid lock.....	53
2.9 TIP122 Transistor.....	55
2.10 Relay Module 5Vdc.....	58
2.11 EasyVR 3.....	60
2.12 Other Components.....	64

2.13 Arduino.....	66
2.14 Arduino IDE.....	75
2.15 Arduino IDE Libraries.....	81

CHAPTER 3 Hardware Systems84

3.1 Automation AC Temperature Controller.....	85
3.2 Fingerprint Door Lock.....	100
3.3 Light System.....	110

CHAPTER 4 FACE RECOGNITION.....135

4.1 What is Image Processing?	136
4.2 Fundamental steps of image processing	137
4.3 What is OpenCV?	144
4.4 Background On Face Recognition.....	147
4.5 Theory of opencv face recognition.....	150
4.6 Eigenface face recognizer.....	152
4.7 FISHERFACES FACE RECOGNIZER.....	159
4.8 HOW TO FIX THIS ISSUE?.....	163
4.9 LOCAL BINARY PATTERNS HISTOGRAMS.....	165
4.10 CODING FACE RECOGNITION USING OPENCV	172
4.11 DATA PREPARATION FOR FACE RECOGNITION	177
4.12 principle component analysis (PCA).....	179
4.13 Face-recognition library based on Dlib.....	182
4.14 Face-recognition.....	184
4.15 Feature.....	185
4.16 Controlling facial recognition (hardware).....	189

4.17 Introduction for raspberry pi.....	191
4.18 Implementation.....	195
CHAPTER 5 ANDROID	204
5.1 Preface.....	205
5.2 Design phase.....	210
5.3 Function phase.....	236
CHAPTER 6 SYSTEM ANALYSIS & DESIGN.....	292
6.1 Objective.....	293
6.2 Analysis Vs Design.....	295
6.3 Analysis and design are not top-down or bottom-up.....	296
6.4 Analysis and design workflow.....	297
6.5 What Is a Use-Case realization?.....	301
6.6 Analysis and Design in an Iterative process.....	303
6.7 Functional Requirements.....	304
6.8 Nonfunctional Requirements.....	305
6.9 Systems Flow Chart.....	305
6.10 SWOT ANALYSIS.....	308
6.11 Use Case For Face Recognition Process.....	311
6.12 Flow Chart For Face Recognition.....	313
References.....	314

Acknowledgements

We would like to take this opportunity to express our heartiest gratitude to our supervisor

Dr. Shereen Aly Taie.

Who has always provided every possible help and support over this time that we have utilized to develop this project of ours we have tried to develop a project that can easily be used and can be related to by general people and we had a vision in our mind to implement such a project and our batch mates have also helped us to work as a source of inspiration through ideas about components wherever we struggled.

Abstract

Home Automation system achieved great popularity in the last decades and it increases the comfort and quality of life. In this paper, an overview of current and emerging home automation systems is discussed. Nowadays most home automation systems consist of a smartphone and microcontroller. In addition, it is a way to have things around your home happen automatically. The first thing that comes to mind when folks think of home automation are robots, flashing lights, complicated electronics and a general feeling that their home is less of a warm home and more of a cold science experiment. However, in most homes today, you can easily find some simple forms of automation such as: security door lock, air conditioner controller, motion activated lights , security systems, in addition a smart phone application is used to control and monitor the home appliances using different type of communication techniques such as: Wi-Fi, Bluetooth, etc...

Therefore, the users can choose their own choice of technology to build home automation system.

CHAPTER 1

{Introduction}



1.1 What is Embedded Systems

An embedded system is a combination of computer hardware and software, either fixed in capability or programmable, designed for a specific function or functions within a larger system. Industrial machines, agricultural and process industry devices, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys, as well as mobile devices, are possible locations for an embedded system.

Embedded systems are computing systems, but they can range from having no user interface (UI) -- for example, on devices in which the system is designed to perform a single task -- to complex graphical user interfaces (GUIs), such as in mobile devices. User interfaces can include buttons, LEDs, touchscreen sensing and more. Some systems use remote user interfaces as well.

1.2 Design

Wireless System: Uses a cellular chip or broadband for Connectivity rather than Hardwires.

Video Monitoring: Offers live feeds of what is happening in the home.

Motion Detector: Detects movement in the house.

Door and Window Sensor: Detects when a door or window opened.

The most basic embedded systems contain two main elements:

- **Embedded systems hardware:** As with any electronic system, an embedded system requires a hardware platform on which to run. The hardware will be based around a microprocessor or microcontroller. The embedded system hardware will also contain other elements including memory, input output (I/O) interfaces as well as the user interface, and the display.

When using an embedded system there is a choice between the use of a microcontroller or a microprocessor:

- **Microcontroller based systems:** A microcontroller is essentially a CPU, central processor unit, or processor with integrated memory or peripheral devices. As fewer external components needed, embedded system using microcontrollers tend to be more widely used.
- **Microprocessor based systems:** Microprocessors contain a CPU but use external chips for memory and peripheral interfaces. As they require more devices on the board, but they allow more expansion and selection of exact peripherals, etc. This approach tends to be used for the larger embedded systems.

Whatever type of processor is used in the embedded system, it may be a very general-purpose type of one of the many highly specialized processors intended for a particular application.

- **Sensors:** Devices, which detect intrusions. Sensors may be placed at the perimeter of the protected area, within it, or both. Sensors can detect intruders by a variety of methods, such as monitoring doors and windows for opening, or by monitoring unoccupied interiors for motions, sound, vibration, or other disturbances.

- **Interconnections:** between components. This may consist of direct wiring to the control unit, or wireless links with local power supplies.
- **Security devices:** Devices to detect unauthorized entry or movements such as spotlights, cameras & lasers.
- **Embedded systems software:** The embedded system software is written to perform a particular function. It is typically written in a high-level format and then compiled down to provide code that can be lodged within a non-volatile memory within the hardware.

1.3 The reason for choosing this project

The Main reasons for choosing this project are:

- **Safety:** Many home automation technologies fall under the umbrella of home security. Consumers purchase these devices because they want to make their homes safer and more secure.
 - Security through automated door locks there must have been times when you or your kids rushed out of the house in a hurry and forgot to lock the door. With automated door locks, you can lock your doors with just one touch on your smart device from any place.
 - Security cameras make your home safer. You cannot be home or monitor everything happening in and around the house at all times, but you can automate the security system to provide the kind of security you desire. You can record clips, detect movements and view the activities and detected stranger people.
- **Temperature:** One of the most unpleasant things is coming to a house that is too warm or too cold. By bring the temperature sensors and programing it to your desired level and it will be controlling the AC temperature automatically to make it balanced.

- **Light controlling:** To make it easier to control lights. Like, you did not need to make move to turn ON/OFF your lights, it just make it from your smart phone or your own voice.

1.4 Project definition

We need made a completely automation home with nontraditional way, so we choose to get a high secure method, we made home door locked by fingerprint and door will not open until we put the right fingerprint, also security camera inside the home with face recognition system to detected the strangers, in addition we made AC system work automatically by programed temperature sensors to reach to the desired level of temperature, finally also we made light controlling system to make it easier to control lights, all of this systems can be controlled by user mobile application.

• Goals and challenges

- Make home completely secure.
- User can controller all entire home by mobile application.
- Our goal to make the user feel comfortable.

• System features

- **Domain:** The ability to create logically independent multiple sections within a partition, with each domain running its own operating system.
- **Reliability:** A function of the care with which the hardware and software design was executed, the quality of the components selected, and the quality of the manufacturing process.
- **Availability:** Give him availability to control his home by manually and by mobile application.

1.5 Recent Home's Technologies

Robot vacuum cleaners: Floor cleaning and vacuuming are easier with a range of Deebot from Ecovacs, a pioneer in the field of engineering robotic vacuum cleaners. D77, the latest Deebot is a 3-dimensional home cleaning solution that has smart technology to detect and navigate obstacles. It has the capability to automatically empty its dustbin. It has different modes for cleaning all types of flooring. Even when you are not at home, you can pre-set it to clean your floors with its intelligent time scheduling feature. This smart device will ease your cleaning woes.

Clocky robotic alarm: Have you felt the need for an alarm that can outsmart and wake you instantaneously? Clocky will make you run around the room before you can turn it off. This smart alarm clock runs away and hides as it continues to beep until you get off your bed. You can no longer snooze and go back to bed. Clocky will ensure you never oversleep again.

Smart Faucet: This environment friendly faucet saves up to 15,000 gallons per unit per year. You can save water with this innovative technology and help conserve water sources. It also conserves energy with its intelligent design. By conserving water and energy, you can leave behind a reduced carbon footprint. Smart Faucet is hygienic and contamination free, as there is no need to touch the faucet valves. It is well suited for children, the elderly, and the disabled. It is an inexpensive means to conserve water and preserve our environment.

Wireless speakers: Stand-alone Wi-Fi home speakers enhance your music pleasure. These speakers can be controlled with a smartphone app. The apps provide access to your iTunes library and other streaming services offering great flexibility. You can play your music from any device that is loaded with the app. The compact design conserves space and provides a classy look to your décor.

The audio quality is exemplary rendering crisp and clear high frequency response. If you enjoy music, wireless speakers are a must-have at your home.

Book Light: Are you worried that your habit of reading books before you hit the bed will disturb your partner? Book Light is an integrated LED display encased in a plastic body that provides discreet lighting to suit your reading needs. You can clip the Book Light to your book easily. You can also adjust the brightness and viewing angle. The product is suited for long distance travel. You can read your favorite book while you are in an airplane or a train with this light-weight product.

Shower meter: Amphiyo A1 is a self-powered energy and water meter for your shower that helps conserve resources. You can conveniently save 440 kWh of energy and 8,500 liters of water every year with this smart device. The device does not require any battery or electricity to run. It gets charged from the energy generated by water flow and is truly energy efficient. It displays real-time information of the water temperature, volume of water used and a climate animation. You can install this device easily, as it does not require any tools.

Eco Dish Cleaner: The Eco cleaner uses ultrasonic waves to clean dishes by ionizing the food particles. This

new technology converts food waste on plates into reusable compost for plants. An eco-friendly gadget provides healthy soil for your plants from your food waste. The Electrolux Eco Cleaner is a portable and compact gadget that will revolutionize the family's dining ritual. You will be able to meet the demands of a modern life with this gadget. It uses solar technology for charging the battery, making it easily sustainable for urban lifestyle.

1.6 Statistics about smart home's over the world

How Many Smart Home Devices Exist?

There are between 6.4 billion and 13 billion smart home devices in use (depending on who is counting). That is up to two devices for every person on the planet!

With each of these devices connecting to the Internet, we are currently in the midst of a rapid expansion of the “Internet of Things.” As purchases of individual smart-home devices grows, so too grows the number of households using the technology.

In 2016, only 13% of U.S. homes had smart technology. As of 2018, it's about 20%, and it's predicted that by 2021,

28% of U.S. households will have at least one smart home device.

In fact, the growing number of smart homes seems inevitable as manufacturers are beginning to phase out the production of devices that do not connect to the Internet. As of 2015, many major television manufacturers began limiting their middle-end to high-end TV models to solely Internet-enabled smart TVs. A subset of smart home technology covers home automation devices that allow the user to “set it and forget it.” Examples include motion sensor triggered cameras, and programmable thermostats. Younger homeowners are more likely to add these home automation devices to their households. For those under 35 years old, 40% are likely to do a renovation that involves home automation in the next three years. In comparison, when looking at all age groups, this drops down to 30%.

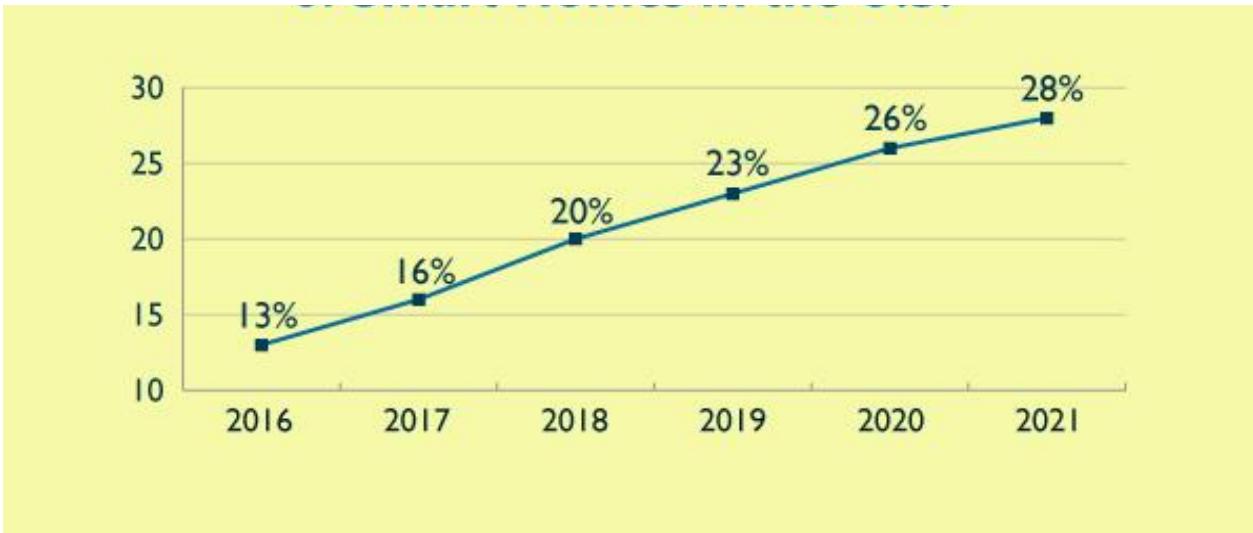


Figure 1 2016 – 2021 statistics

Which Smart Home Products are Popular, and Why?

While having an Internet-enabled lawnmower or BBQ grill may sound exciting, the more popular smart home products are ones that have been on the market for a while. For example, smart televisions account for 41% of the smart home products in use today, and Samsung's first smart TV was released a decade ago in 2008.

Most Common Smart Home Products

TVs:	41%
Thermostats:	23%
Home security:	17%
Refrigerators:	16%
Audio systems:	13%

The reasons for smart home technology adoption vary as widely as the devices that are available. However, about one third of users purchasing smart-home devices are early adopters who seek to be on the cutting edge of technology.

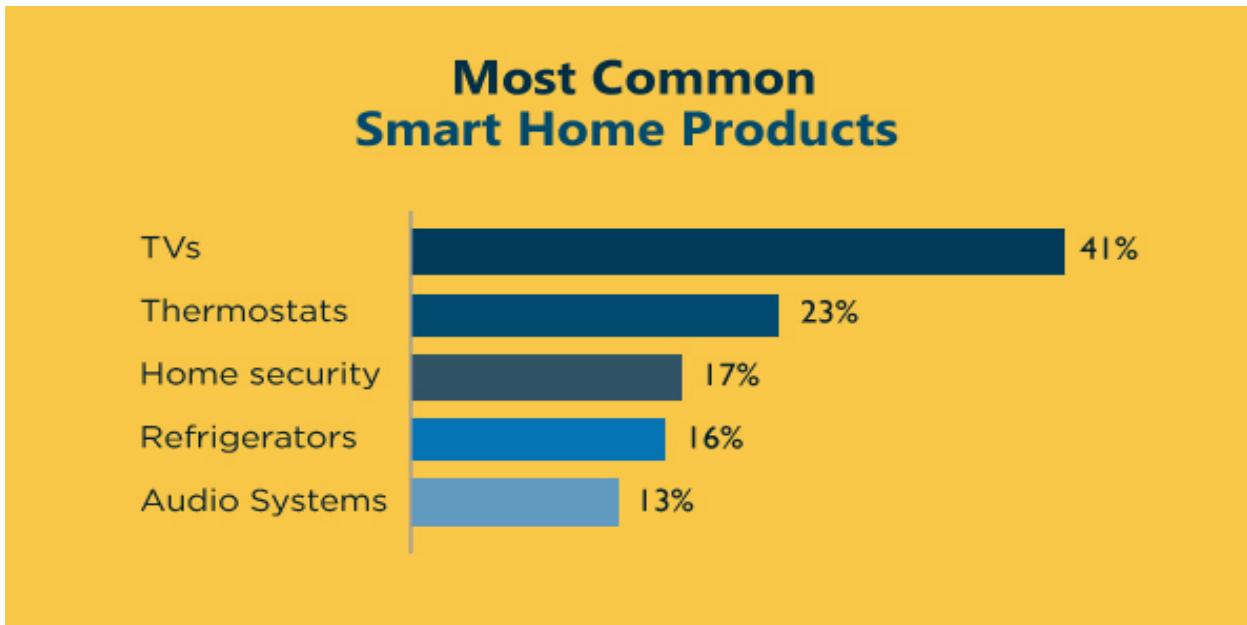


Figure 2

Why Are People Interested in Smart Home Technology?

- 30%: To be on the cutting edge of technology
- 27%: To increase their family's safety
- 24%: To save money by using less energy
- 14%: To be more eco-friendly
- 5%: Received as gifts/don't want them or use them.

Integrating smart-home technology into your house or apartment increases its value and makes it more marketable. Whether the implemented technology is used for convenience, home security, or to have a greener home, a smart home is desired by two-thirds of consumers.

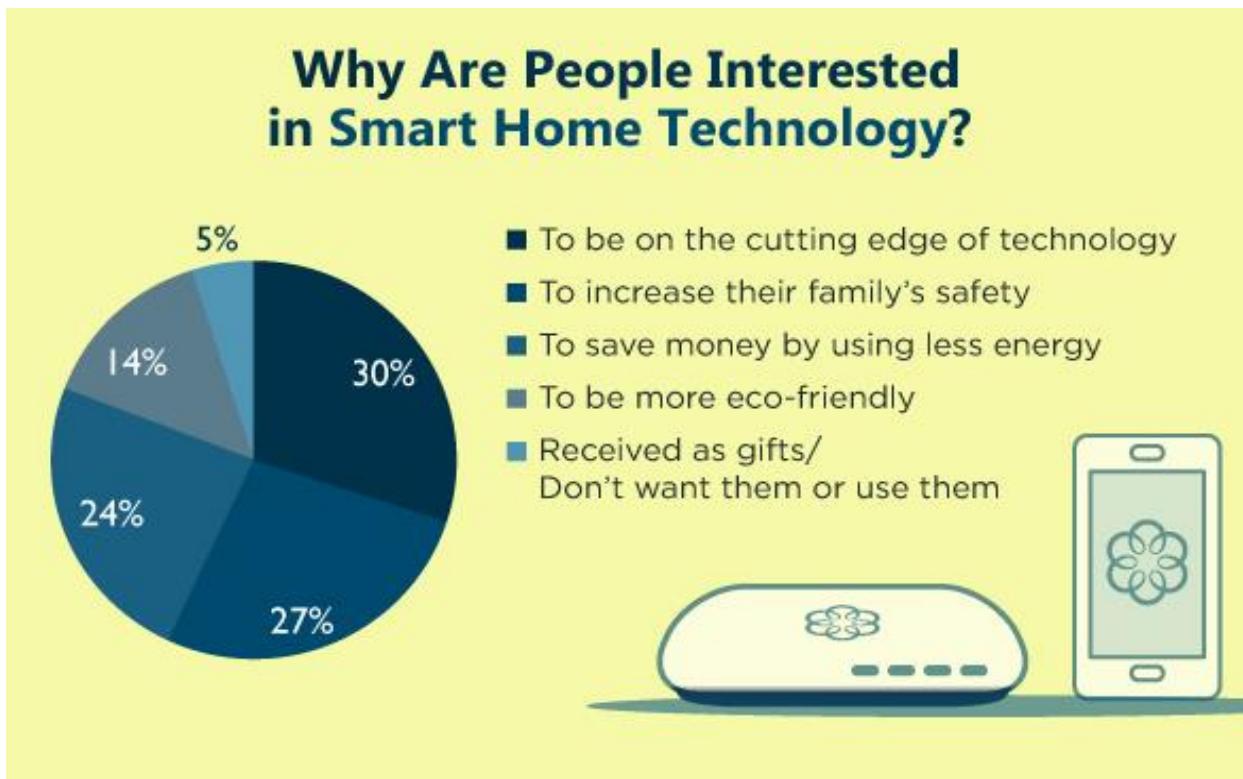


Figure 3

Cities with the Most Smart Homes

The citizens living in these cities will not be left behind with antiquated rotary wall phones and flip cell phones. Instead, many households and their neighbors are adopting smart home technology.

It is easy to assume that these smart cities would be located in the tech center of Silicon Valley or in finance hubs like NYC, however you might be surprised by the cities with the highest number of smart homes per capita.

Top 10 Cities with the Highest Concentration of Smart Homes



Figure 4

CHAPTER 2

{IOT APPLICATION}

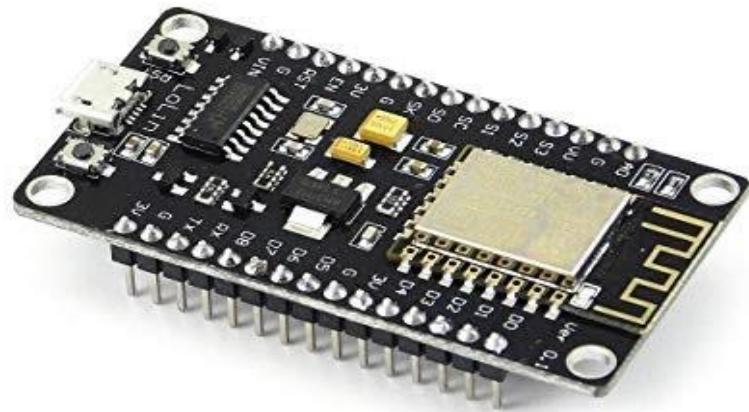


2.1 What is IOT?

Internet of Things (IOT) describes an emerging trend where a large number of embedded devices (things) are connected to the Internet. These connected devices communicate with people and other things and often provide sensor data to cloud storage and cloud computing resources where the data is processed and analyzed to gain important insights. Cheap cloud computing power and increased device connectivity is enabling this trend.

IOT solutions are built for many vertical applications such as environmental monitoring and control, health monitoring, vehicle fleet monitoring, industrial monitoring and control, and home automation.

2.2 ESP-WIFI NODE-MCU module



General Description:

Node-MCU V3 is an open-source firmware and development kit that plays a vital role in designing an IOT product using a few script lines. Multiple GPIO pins on the board allow us to connect the board with other peripherals and are capable of generating PWM, I2C, SPI, and UART serial communications. The interface of the module is mainly divided into two parts including both Firmware and Hardware where former runs on the ESP8266 Wi-Fi SOC and later is based on the ESP-12 module. The firmware is based on LUA – A scripting language that is easy to learn, giving a simple programming environment layered with a fast scripting language that connects you with a well-known developer community.

Features:

- Uses CH340G instead of CP2102.
- Node-MCU has built-in USB-TTL serial with super reliable industrial strength CH340G for superior stability on all supported platforms.
- Communication interface voltage: 3.3V.
- Antenna type: Built-in PCB antenna is available.
- Wireless 802.11 b/g/n standard
- Wi-Fi at 2.4GHz, support WPA / WPA2 security mode
- Support STA/AP/STA + AP three operating modes
- Built-in TCP/IP protocol stack to support multiple TCP Client connections (5 MAX)
- D0 ~ D8, SD1 ~ SD3: used as GPIO, PWM, IIC, etc., port driver capability 15mA
- AD0: 1 channel ADC
- Power input: 4.5V ~ 9V (10VMAX), USB-powered
- Transfer rate: 110-460800bps
- Support UART / GPIO data communication interface

- Remote firmware upgrade (OTA)
- Support Smart Link Smart Networking
- Working temperature: -40 ~ + 125
- Drive Type: Dual high-power H-bridge driver
- ESP8266 has IO Pin
- Don't need to download resetting
- A great set of tools to develop ESP8266
- Flash size: 4MByte

Hardware Interfaces Configuration:

The ESP8266EX integrates a Tensilica L106 32-bit RISC processor, which achieves extra low power consumption and reaches a maximum clock speed of 160 MHZ. The Real-Time Operating System (RTOS) and Wi-Fi stack allow 80% of the processing power to be available for user application programming and development. The CPU includes the interfaces as below:

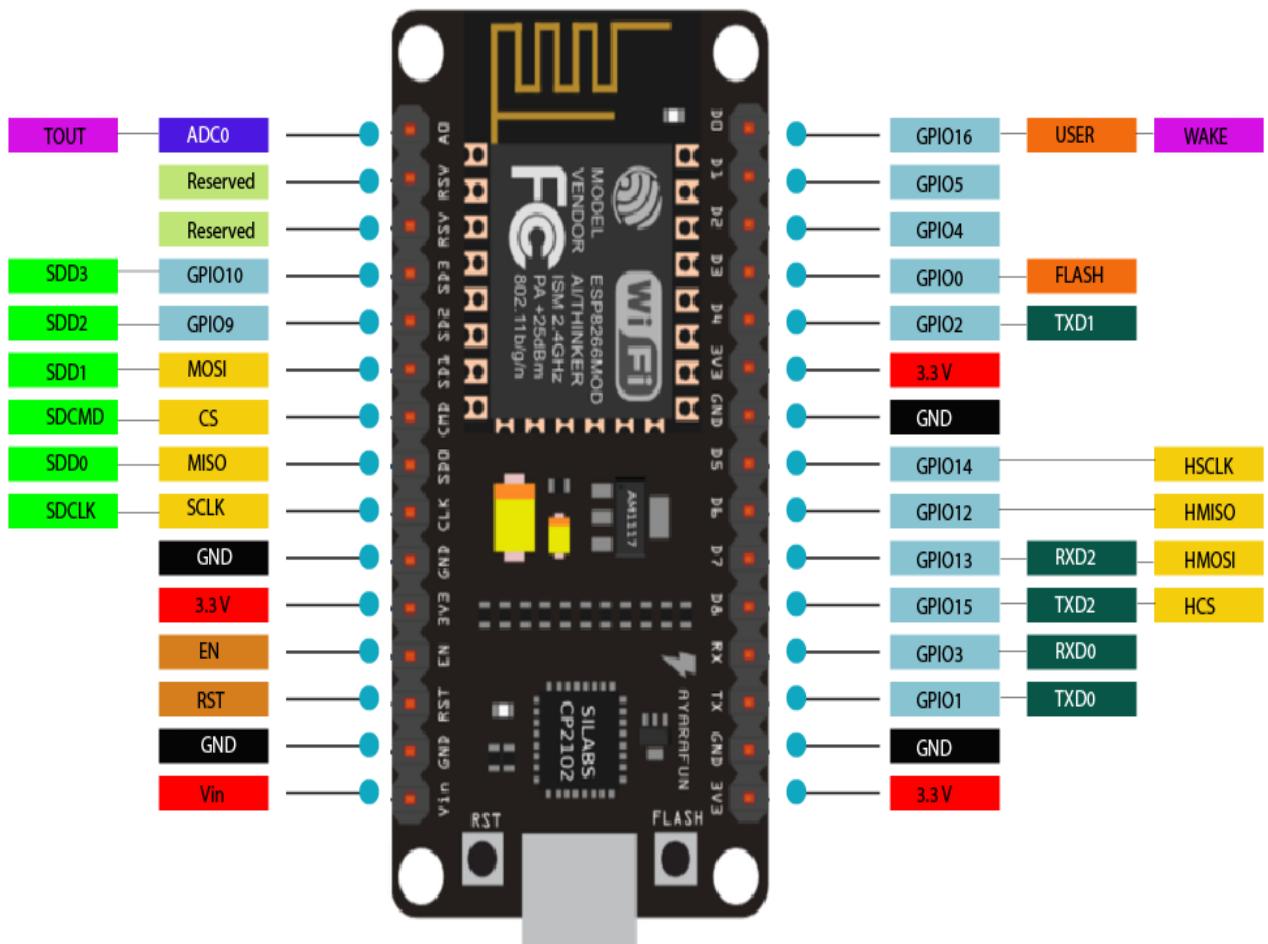
- Programmable RAM/ROM interfaces (iBus), which can be connected with memory controller, and can be used to visit flash.
- Data RAM interface (dBus), which can be connected with memory controller.
- AHB interface which can be used to visit the register.

Performance Specification:

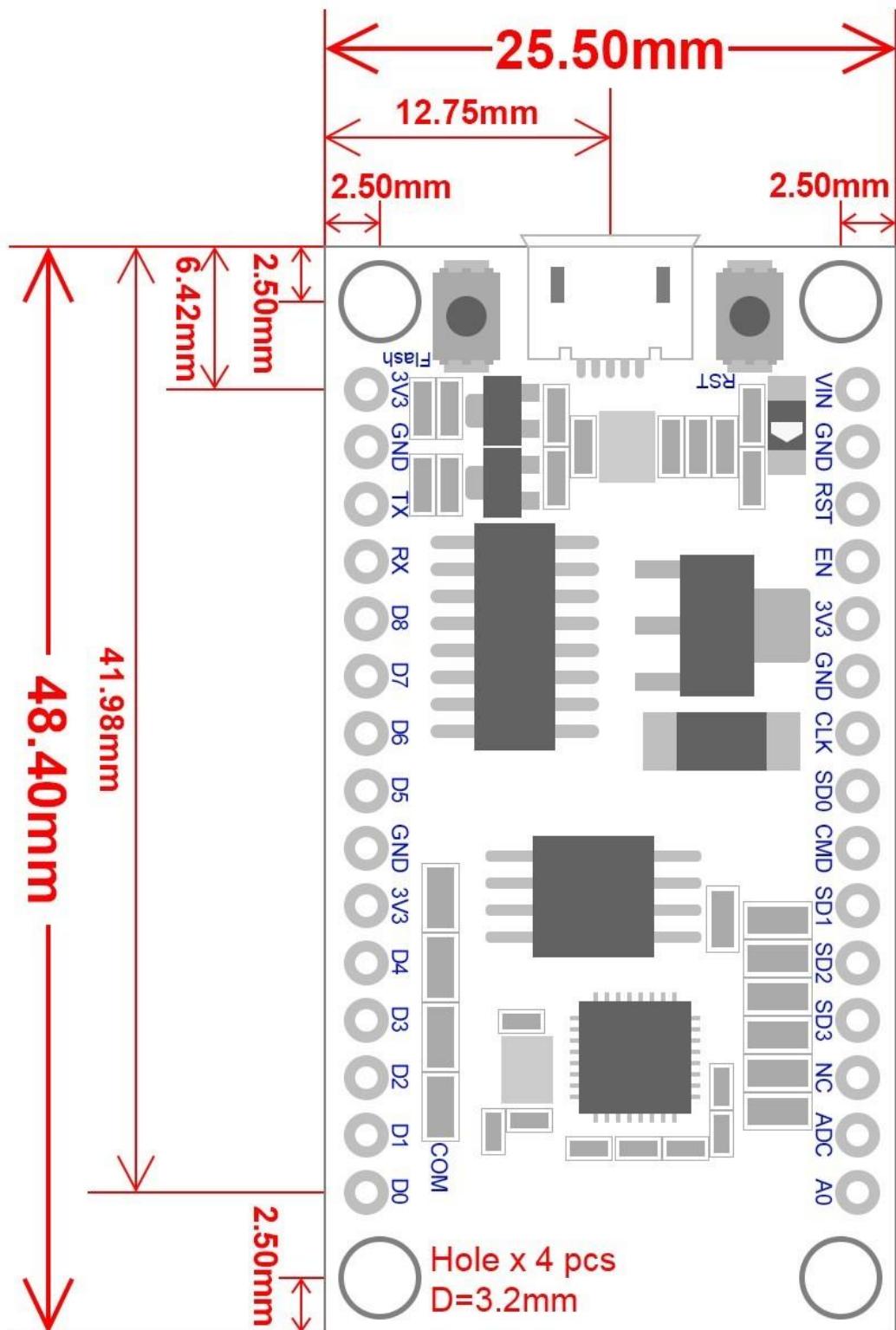
Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n (HT20)
	Frequency Range	2.4G ~ 2.5G (2400M ~ 2483.5M)
	TX Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
		802.11 g: -75 dbm (54 Mbps)
		802.11 n: -72 dbm (MCS7)
Hardware	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
	CPU	Tensilica L106 32-bit processor
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5V ~ 3.6V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	-40°C ~ 125°C
	Package Size	QFN32-pin (5 mm x 5 mm)
Software	External Interface	-
	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Pin Descriptions:

There are altogether 30 pin counts, the definitions of which are described.



Packaging and Dimension:



2.3 IR (Infrared) Transfer Sensor



General Description:

An **infrared sensor** is an electronic device, which emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measures only infrared radiation, rather than emitting it that is called as a passive IR sensor. Usually in the infrared spectrum, all the objects radiate some form of thermal radiations. These types of radiations are invisible to our eyes, which can be detected by an infrared sensor. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode which is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, the resistances and these output voltages, change in proportion to the magnitude of the IR light received.

Features:

- High Reliability
- Excessive radiant intensity
- Forward voltage is low
- Having lead spacing of 2.54mm
- Maximum wavelength is 940nm
- Pb free
- RoHS certified
- Easy to use with breadboard or perf board
- Package type is T-1 ¾

Specifications:

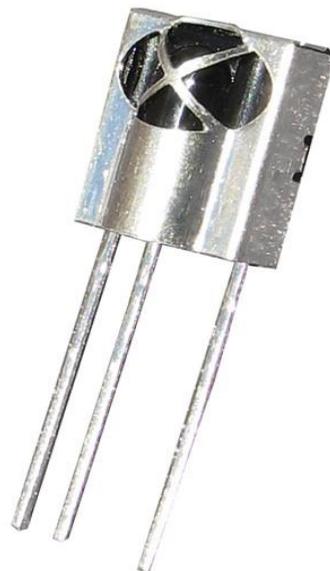
- Working voltage: 3 - 5V DC
- Output type: Digital switching output (0 and 1)
- 3mm screw holes for easy mounting

Hardware Interfaces Configuration:

An infrared sensor circuit is one of the basic and popular sensor module in an electronic device. This sensor is analogous to human's visionary senses, which can be used to detect obstacles and it is one of the common applications in real time. In this project, the transmitter section includes an IR sensor, which transmits continuous

IR rays to be received by an IR receiver module. An IR output terminal of the receiver varies depending upon its receiving of IR rays. Since this variation cannot be analyzed as such, therefore this output can be fed to a comparator circuit. Here an operational amplifier (op-amp) of LM 339 is used as comparator circuit. When the IR receiver does not receive a signal, the potential at the inverting input goes higher than that non-inverting input of the comparator IC (LM339). Thus, the output of the comparator goes low, but the LED does not glow. When the IR receiver module receives signal to the potential at the inverting input goes low. Thus, the output of the comparator (LM 339) goes high and the LED starts glowing. Resistor R1 (100), R2 (10k) and R3 (330) are used to ensure that minimum 10 mA current passes through the IR LED Devices like Photodiode and normal LEDs respectively. Resistor VR2 (preset=5k) is used to adjust the output terminals. Resistor VR1 (preset=10k) is used to set the sensitivity of the circuit Diagram. Read more about IR sensors.

2.4 TSOP IR Receiver



General Description:

The **TSOP1738 series** are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter. The demodulated output signal can directly be decoded by a microprocessor. TSOP1738 is the standard IR remote control receiver series, supporting all major transmission codes. The sensor can be used for Proximity detection application along with an IR led in robotics and security systems. It receives IR signal at 38 kHz frequency.

Features:

- Photo detector and preamplifier in one package
- Internal filter for PCM frequency
- Improved shielding against electrical field disturbance
- TTL and CMOS compatibility
- Output active low
- Low power consumption
- High immunity against ambient light
- Continuous data transmission possible (up to 2400 bps)
- Suitable burst length 10 cycles/burst

Specifications:

- Supply Voltage: 5 V
- Power consumption: 0.4 to 1.0 mA
- Min irradiation: 0.35 MW/m² typ.
- Angle of detection: 90
- Dimensions of the casing (mm): 12.5 x 10 x Thickness 5.8
- Temperature range: -25 C to +85 C

Applications:

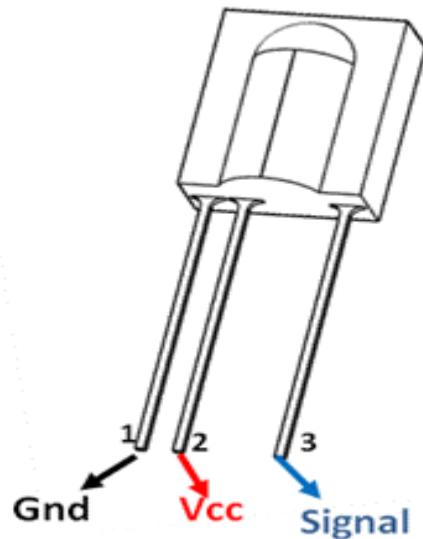
- Receive IR signals
- Decode Remote signals
- Analyses, re-create or duplicate remote Signals
- Wireless control applications
- Receiver circuit for IR remote controls
- IR Remote tester circuits

Hardware Interfaces Configuration:

TSOP 1738 is an IR Receiver for IR (Infrared) remote controls. It consists of Photo Detector, Gain Control, Band Pass Filter, Demodulator and a Pre – amplifier in a single package. TSOP17XX series of IR Receivers are very common for all types of transmission codes supporting different modulating techniques like RC6, RC5, NEC, Sony, etc. TSOP 1738 in particular supports a carrier frequency of 38 KHz and the output of this IR Receiver can be directly connected to a microcontroller or a microprocessor. TSOP 1738 has three pins namely GND, VS and OUT. The following image shows the pin out of TSOP 1738. Some of the other characteristics of TSOP 1738 are high immunity to ambient light, active low output, low power consumption, continuous data transmission and compatibility with TTL as well as CMOS. A typical

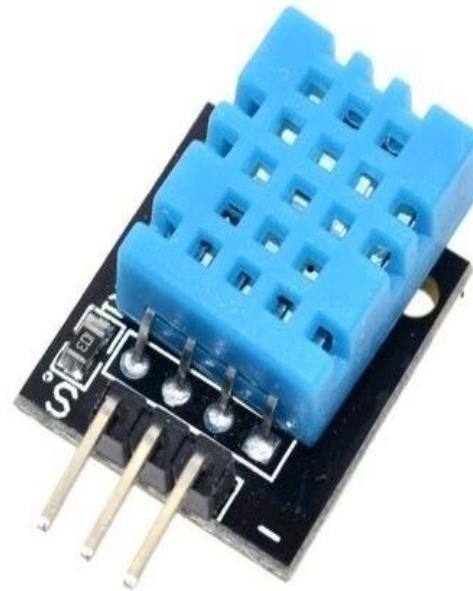
application circuit of TSOP 1738 IR Receiver (or any IR Receiver for that matter) will consists of an IR Transmitter (IR LED), IR Receiver (TSOP 1738) and a microcontroller. The following image describes the block diagram of the IR Transmitter and Receiver. In the image, the IR Transmitter emits Infrared light modulated at a particular frequency. The IR Receiver will receive this modulated signal, demodulates it, and sends it to the Microcontroller. From the image, it can be observed that the output of the IR Receiver is directly connected to the Microcontroller and there is no need for any extra interface. In addition, the Resistor R and Capacitor C are only used to suppress the power supply disturbances.

Pin Descriptions:



Pin Number	Pin Name	Description
1	Ground	Connected to the Ground of circuit
2	Vcc	Typically connect to +5V, maximum of 6V can be given
3	Signal	The signal pin gives out the sequence based on the IR signal detected

2.5 DHT11 SENSOR TEMPERATURE



General Description:

This **DHT11** Temperature and Humidity Sensor features a calibrated digital signal output with the temperature and humidity sensor capability. It is integrated with a high-performance 8-bit microcontroller. Its technology ensures the high reliability and excellent long-term stability. This sensor includes a resistive element and a sensor for wet NTC temperature measuring devices. It has excellent quality, fast response, anti-interference ability and high performance.

Each DHT11 sensors features extremely accurate calibration of humidity calibration chamber. The calibration coefficients stored in the OTP program memory, internal sensors detect signals in the process, we should call these calibration coefficients. The single-wire serial interface system is integrated to become quick and easy. Small size, low power, signal transmission distance up to 20 meters, enabling a variety of applications and even the most demanding ones. The product is 4-pin single row pin package. Convenient connection, special packages can be provided according to users need.

Features:

- Full range temperature compensated
- Relative humidity and temperature measurement
- Calibrated digital signal
- Outstanding long-term stability
- Extra components not needed
- Long transmission distance
- Low power consumption
- 3 pins packaged and fully interchangeable

Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: $\pm 1^{\circ}\text{C}$ and $\pm 1\%$

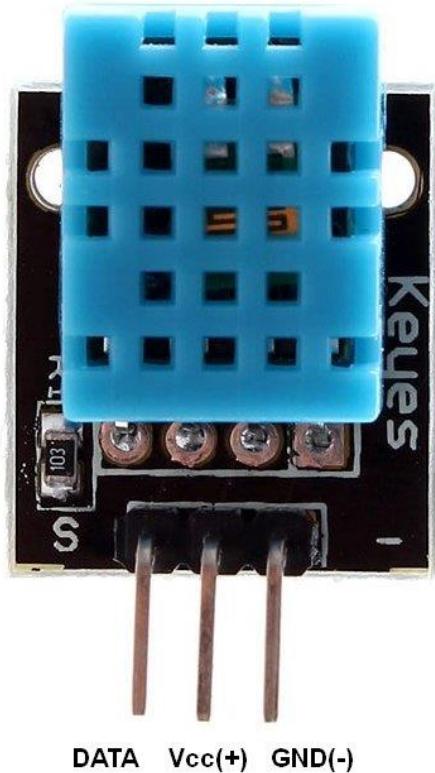
Applications:

- Measure temperature and humidity
- Local Weather station
- Automatic climate control
- Environment monitoring

Hardware Interfaces Configuration:

The DHT11 Sensor is factory calibrated and outputs serial data and hence it is highly easy to set it up. The connection diagram for this sensor is shown below. As you can see the data, pins are connected to an I/O pin of the MCU and a 5K pull-up resistor is used. This data pin outputs the value of both temperature and humidity as serial data. If you are trying to interface DHT11 with Arduino then there are ready-made libraries for it, which will give you a quick start. If you are trying to interface it with some other MCU then the datasheet given below will come in handy. The output given out by the data pin will be in the order of 8-bit humidity integer data + 8bit the Humidity decimal data +8 bit temperature integer data + 8bit fractional temperature data +8 bit parity bit. To request the DHT11 module to send these data the I/O pin has to be momentarily made low and then held high as shown in the timing diagram below. The duration of each host signal is explained in the DHT11 datasheet, with neat steps and illustrative timing diagrams.

Pin Configuration:



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

2.6 OLED LCD



General Description:

The **OLED displays** are one of the most attractive displays available for a microcontroller. It has a good view angle and pixel density, which makes it reliable for displaying small level graphics. Either interfacing this IC with MCU can be done using IIC or using SPI hence helps to save some pins as well. Therefore, if you are looking for a slim, attractive and efficient display module to make your project look cool with graphics then this module might be the right choice for you.

Features:

- Monochrome 7-pin SSD1306 0.96" OLED display.
- 128×64 pixel resolution with 160° viewing angle.
- Supply voltage 3V – 5V (supports both 5V and 3.31v logic devices).
- Uses SSD1306 for interfacing hence can communicate through SPI or IIC.
- Multiple SPI or IIC devices are supported
- Can be easily interfaced with Arduino (Library available).
- Supports decent graphics of bitmap images.
- Available in different colors and sizes as discussed below.

Applications:

- Used in consumer electronics.
- Used for Smartwatch, mobile phone, and MP3 displays.
- Small level gaming displays.
- Wide range of viewing angle enable to be used in low light.

Pin Configuration:

Pin No:	Pin Name:	Description
1	Ground (Gnd)	Connected to the ground of the circuit
2	Supply (Vdd,Vcc,5V)	Can be powered by either 3.3V or 5V
3	SCK (D0,SCL,CLK)	The display supports both IIC and SPI, for which clock is supplied through this pin
4	SDA (D1,MOSI)	This is the data pin of the both, it can either be used for IIC or for SPI
5	RES(RST,RESET)	When held to ground momentarily this pin resets the module
6	DC (A0)	This is command pin, can either be used for SPI or for IIC
7	Chip Select (CS)	Normally held low, used only when more than one SPI device is connected to MCU

2.7 Fingerprint Recognition



General Description:

R307 Fingerprint Module consists of an optical fingerprint sensor, high-speed DSP processor, high-performance fingerprint alignment algorithm, high-capacity FLASH chips and other hardware and software composition, stable performance, simple structure, with fingerprint entry, image processing, fingerprint matching, search and template storage and other functions. The R307 fingerprint module has two interface TTL UART and USB2.0, USB2.0 interface can be connected to the computer; RS232 interface is a TTL level, the default baud rate is 57600, can be changed, refer to a communication

protocol; can And microcontrollers, such as ARM, DSP and other serial devices with a connection, 3.3V 5V microcontroller can be connected directly. Needs to connect the computer level conversion, level conversion note, embodiments such as a MAX232 circuit.

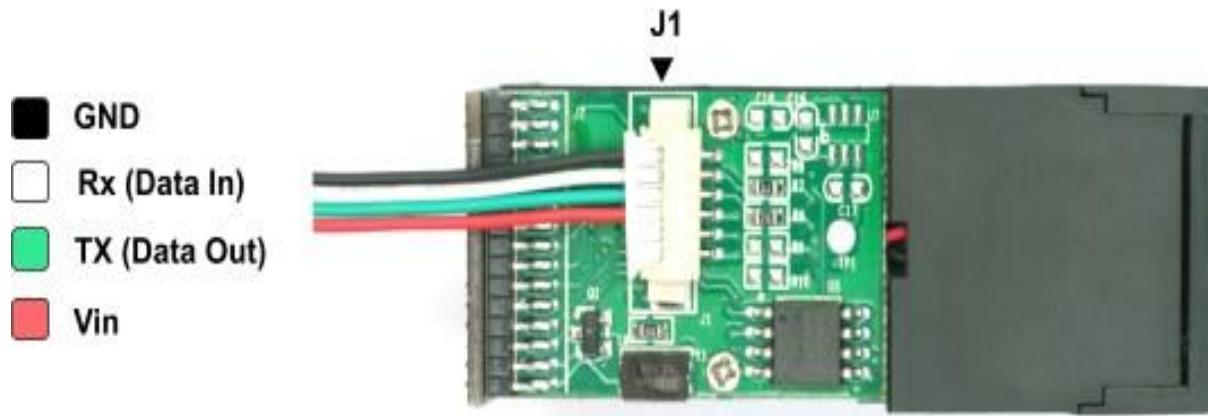
Features:

- Independent fingerprint collection, fingerprint registration, fingerprint comparison and fingerprint search function.
- A low power consumption of the product
- A strong anti-static ability, the anti-static index reached 15KV above.
- Professional optical technology, precise module manufacturing techniques
- Suitable for different applications, security levels can be set by the user to adjust.

Specifications:

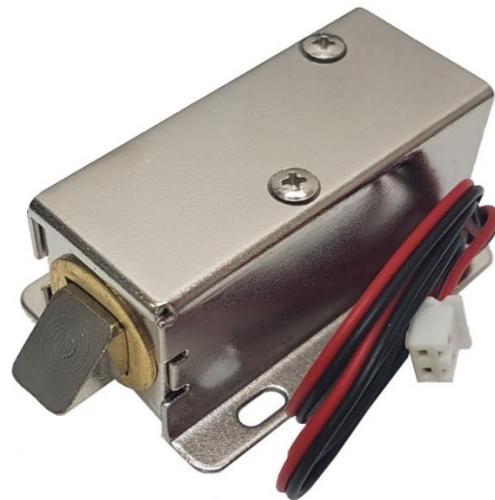
- Fingerprint sensor type: Optical
- Sensor Life: 100 million times
- Static indicators: 15KVBacklight: bright green
- Interface: USB1.1/UART(TTL logical level)
- RS232 communication baud rate: 4800BPS~115200BPS changeable
- Dimension: 44.1*20*23.5mm
- Verification Speed: 0.3 sec
- Scanning Speed: 0.5 sec
- Character file size: 256 bytes
- Template size: 512 bytes
- Storage capacity: 1000
- Security level: 5 (1,2,3,4,5(highest))
- False Acceptance Rate (FAR) :0.0001%
- False Rejection Rate (FRR): 0.1%
- Resolution 500 DPI
- Working current: Typical 50 mA, Peak 80mA
- Matching Method: 1: N

Pin Descriptions:



Pin #	Pin Name	Details
1	5V	Regulated 5V DC
2	GND	Common Ground
3	TXD	Data output - Connect to MCU RX
4	RXD	Data Input - Connect to MCU TX

2.8 Solenoid lock



General Description:

12V Solenoid lock has a slug with a slanted cut and a good mounting bracket. It is an electronic lock, designed for a basic cabinet, safe or door. When 9-12VDC is applied, the slug pulls in so it does not stick out and the door can be opened. It does not use any power in this state. It is very easy to install for automatic door lock systems like electric door lock with the mounting board. This solenoid in particular is nice and strong.

Features:

- Slim design, security and stability, low power consumption
- Applied to the cabinet lock, locker locks, file cabinet locks, luggage locks, electric locks, door locks, solenoid locks, drawer, newspaper boxes lock, sauna lock, locker electromagnetic locks, electric locks, newspaper boxes, sauna Electronics lock
- Designed with the open frame type and mount board, high power.
- Easy to install for the electric door lock or other automatic door lock systems with the mounting board.

Specifications:

- Operating voltage : 12VDC
- Draws 650mA at 12V, 500 mA at 9V when activated
- Designed for 1-10 seconds long activation time
- Wire length: 222.25mm

2.9 TIP122 Transistor



General Description:

The **TIP122** is a Darlington pair NPN transistor. It functions like a normal NPN transistor, but since it has a Darlington pair inside it has a good collector current rating of about 5A and a gain of about 1000. It can also withstand about 100V across its collector.

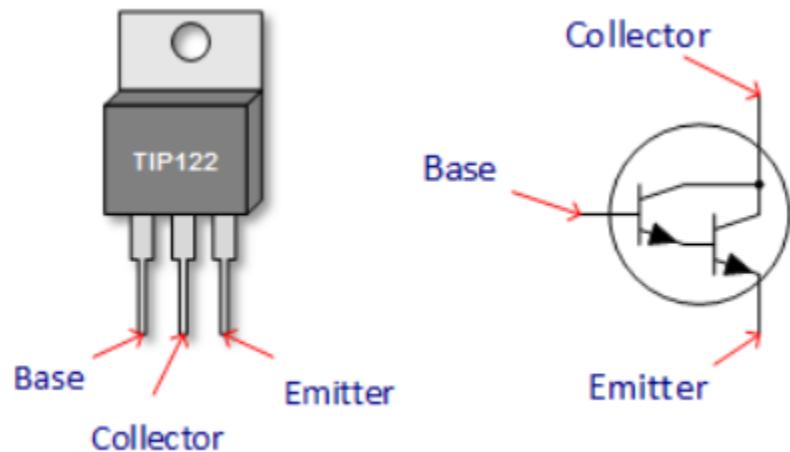
Features:

- Darlington Medium-power NPN Transistor
- High DC Current Gain (hFE), typically 1000
- Continuous Collector current (I_C) is 5A
- Collector-Emitter voltage (V_{CE}) is 100 V
- Collector-Base voltage (V_{CB}) is 100V
- Emitter Base Voltage (V_{BE}) is 5V
- Base Current(I_B) is 120mA
- Available in To-220 Package

Applications:

- Can be used to switch high current (up to 5A) loads
- Can be used as medium Power switches
- Used where high amplification is needed
- Speed control of Motors
- Inverter and other rectifier circuits

Pin Descriptions:



Pin Number	Pin Name	Description
1	Base	Controls the biasing of transistor, Used to turn ON or OFF the transistor
2	Collector	Current flows in through collector, normally connected to load
3	Emitter	Current Drains out through emitter, normally connected to ground

2.10 Relay Module 5Vdc



General Description:

- 30A relay module, normally open interface maximum load: AC 250V/30A, DV 30V/30A;
- Optical coupling isolation, driving ability is strong, stable performance Trigger current is 5mA;
- Module can be set high level or low level trigger by the jumper;
- Power indicator light (green), relay status indicator light (red);
- Module Size:50mm*33mm*24mm;

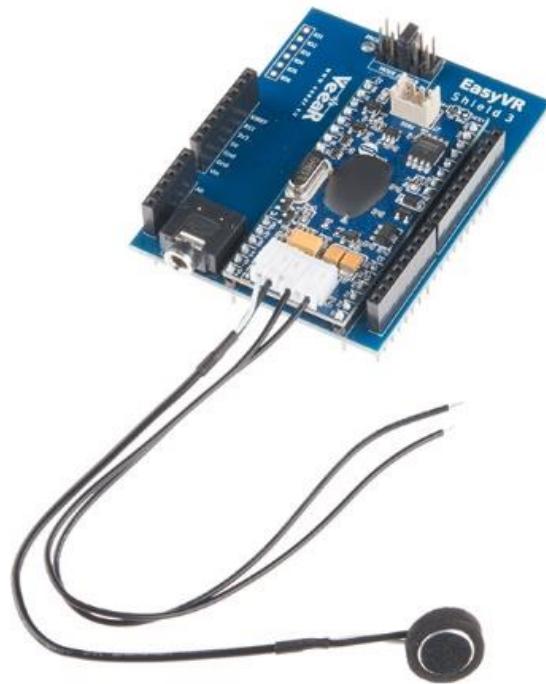
Technical parameters:

- Voltage version: 5V
- The static current: 5mA
- Working current: 190mA
- Trigger current: 2-4mA

Module interface specifications:

- DC+ : DC power supply positive pole
- DC- :DC power supply negative pole
- IN : signal triggering pin
- JD+ : relay control voltage positive
- JD- : relay control voltage negative
- DC+ and JD+ shorted by jumper cap, DC- and JD- shorted with jumper cap, it is the same voltage between trigger terminal and relay control terminal
- High and low level trigger mode selection. Jumper and L pin connection, IN pin is low level trigger Jumper and H pin connection, IN pin is high level trigger
- Normally closed pin (NC) : relay normally closed pin
- Common pin (COM) : relay common pin
- Normally opened pin (NO) : relay normally opened pin

2.11 EasyVR 3



General Description:

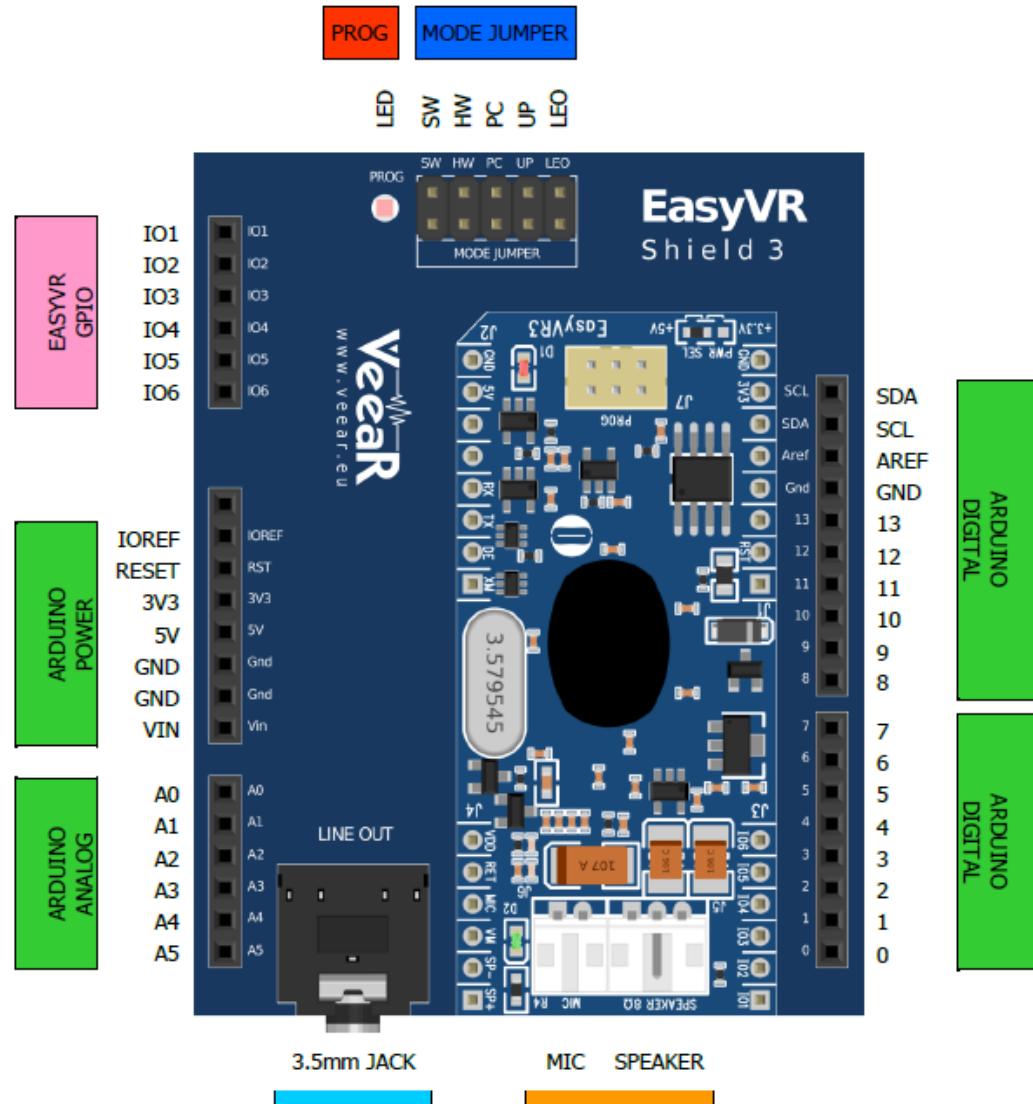
The EasyVR Shield 3 is an adapter board for the EasyVR 3 Module, designed to simplify its use among the Arduino Community. The Shield is compatible with any Arduino board using UNO-R3 Shield headers, running at either 3.3V or 5V levels, by using the IOREF pin to select the EasyVR operating voltage. It is also backward compatible with earlier Arduino boards that do not have the IOREF pin, which are using 5V I/O levels by default. If your board does not have the IOREF pin but it is running at 3.3V, you can still operate the EasyVR Shield 3 correctly if you

manually connect pins IOREF and 3V3 together, for example with a jumper wire. The board comes with separate Arduino stackable headers for the Shield interface. The EasyVR 3 module

Features:

- Compatible with Arduino boards that have the 1.0 Shield interface (UNO R3) including.
- Supports 5V and 3.3V main boards through the IOREF pin (defaults to 5V if this pin is absent).
- Supports direct connection to the PC on main boards with a separate USB/Serial chip and a special
- Software-driven “bridge mode” on boards with only native USB interface, for easy access.
- Configuration with the EasyVR Commander.
- Enables different modes of serial connection and flash updates to the embedded EasyVR.
- module (through the Mode Jumper)
- Supports remapping of serial pins used by the Shield (in SW mode).
- Provides a 3.5mm audio output jack suitable for headphones or as a line out.

• Pin Descriptions:



(Detail - Bottom View)

Group	Pin	Description
ARDUINO HEADERS	-	Arduino UNO-R3 Shield interface, pass-through connectors (Pins 0-1 are in use when J12 is set to UP, PC, HW or LEO) (Pins 12-13 or 8-9 are in use when J12 is set to SW)
EASYVR AUDIO	-	Audio cables connectors of the EasyVR 3 module (microphone and speaker)
LINE OUT	-	3.5mm stereo/mono jack (16Ω - 32Ω headphones or line-level output)
MODE JUMPER	SW	Arduino Software Serial (connected to pins 12-13 or 8-9)
	HW	Arduino Hardware Serial (connected to pins 0-1)
	PC	PC Mode (Arduino disabled, EasyVR in command mode)
	UP	Update Mode (Arduino disabled, EasyVR in boot mode)
	LEO	Leonardo Update (Arduino enabled, EasyVR in boot mode)
PROG	-	Red light indicator for Flash programming modes (UP and LEO)
SW SERIAL PINS	RX	Use resistor to select Software Serial RX pin: 12 or 8
	TX	Use resistor to select Software Serial TX pin: 13 or 9
EASYVR GPIO	I01	General purpose I/O as found on the embedded EasyVR 3 module (referenced at the internal VDD logic level - see note below)
	I02	
	I03	
	I04	
	I05	
	I06	

2.12 Other Components

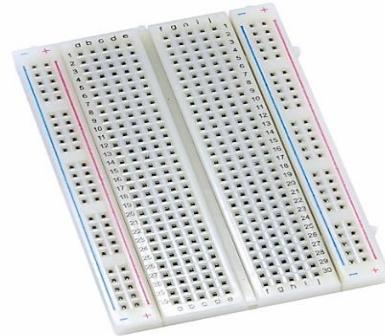
- **Batteries (12V ,9V)**



- **Connection wires**



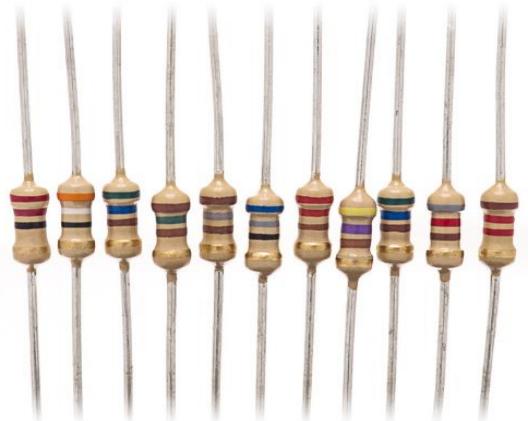
- **Breadboards**



- **leds**



- **resistors**



2.13 Arduino

What is Arduino?

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. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years, Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students

without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IOT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the

projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step-by-step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than \$50
- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

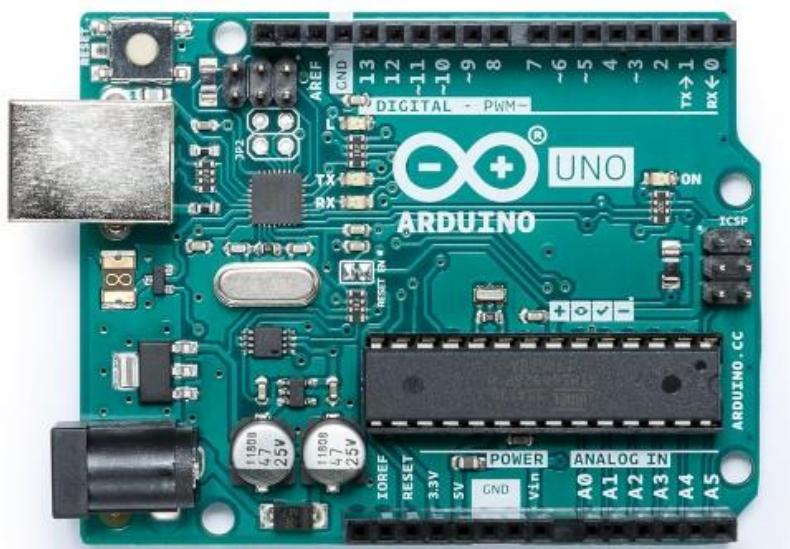
- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it is conveniently based on the Processing-programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it is based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Types of Arduino:

- Arduino UNO
- Arduino MEGA
- Arduino NANO
- Arduino MICRO

In our project we used Arduino (UNO, MEGA)

Arduino UNO



General Description:

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP

header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

Technical Specifications:

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Arduino Mega



General Description:

The Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards.

Technical Specifications:

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

2.14 Arduino IDE

The Arduino is a fantastic single-board microcontroller solution for many projects, and, in this blog, we will look at the Integrated Development Environment, or IDE, that is used to program it!



ARDUINO 1.8.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software

Getting Started with Arduino

WELCOME TO ARDUINO! BEFORE YOU START CONTROLLING THE WORLD AROUND YOU, YOU'LL NEED TO SETUP THE SOFTWARE TO PROGRAM YOUR BOARD

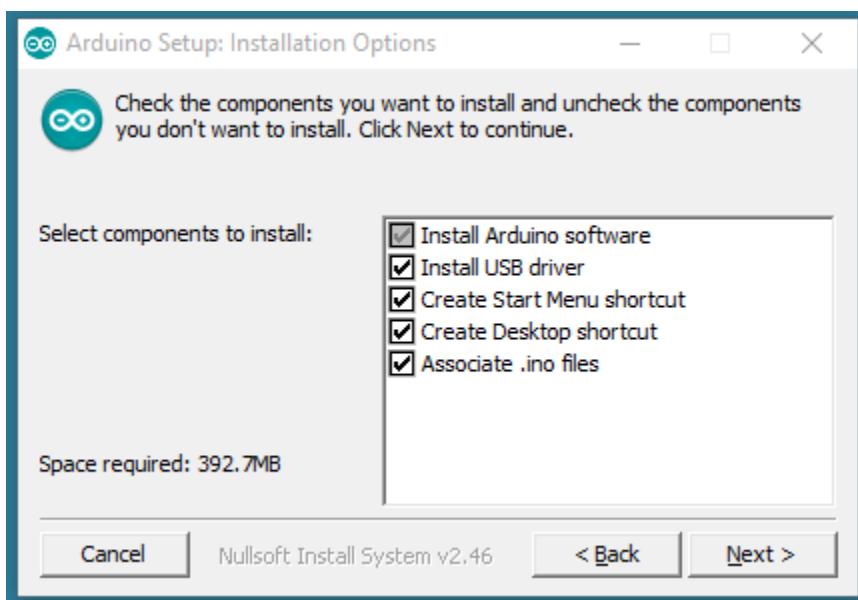
The Arduino Software (IDE) allows you to write programs and upload them to your board, you will find two options:

1. If you have a reliable Internet connection, you should use the online IDE (Arduino Web Editor). It will allow you to save your sketches in the cloud, having them available from any device and backed up. You will always have the most up-to-date version of the IDE without the need to install updates or community generated libraries.
2. If you would rather work offline, you should use the latest version of the desktop IDE

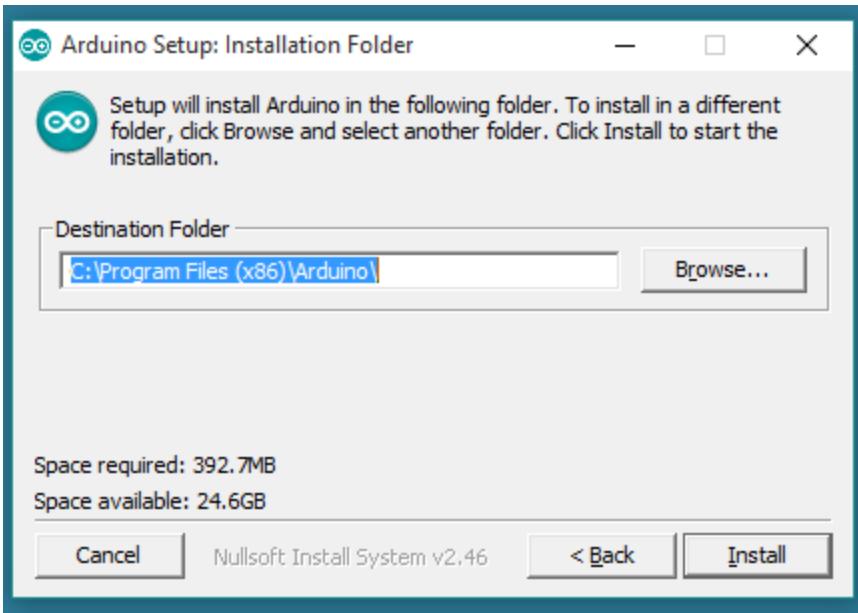
Install the Arduino Software (IDE) on Windows PCs

Get the latest version from the [download page](#). You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package, you need to install the drivers manually. The Zip file is also useful if you want to create a [portable installation](#).

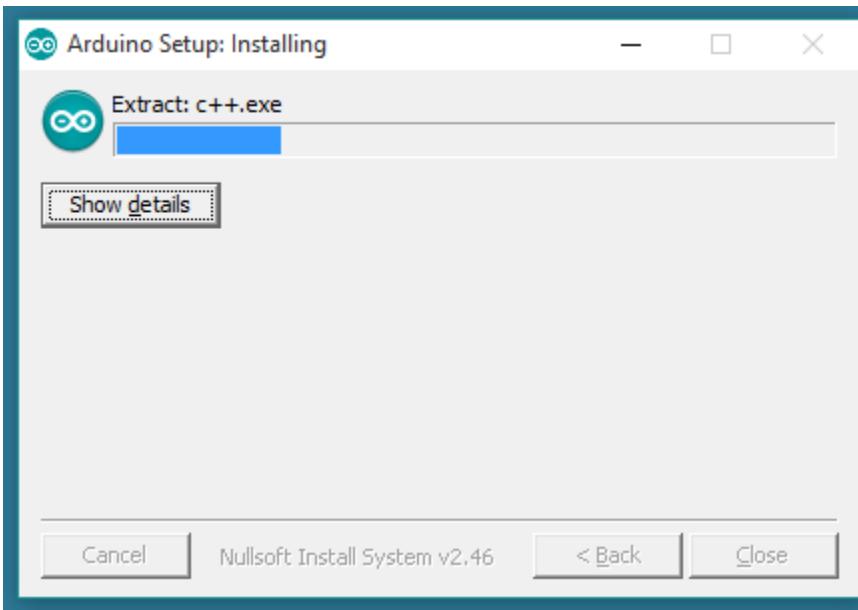
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



Choose the components to install



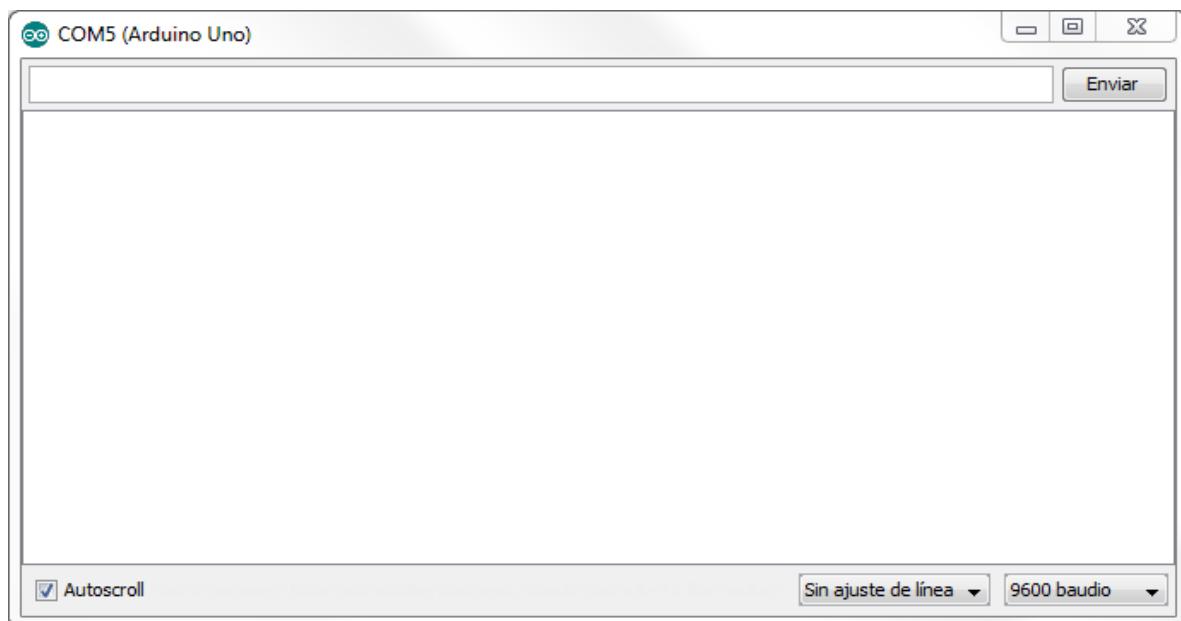
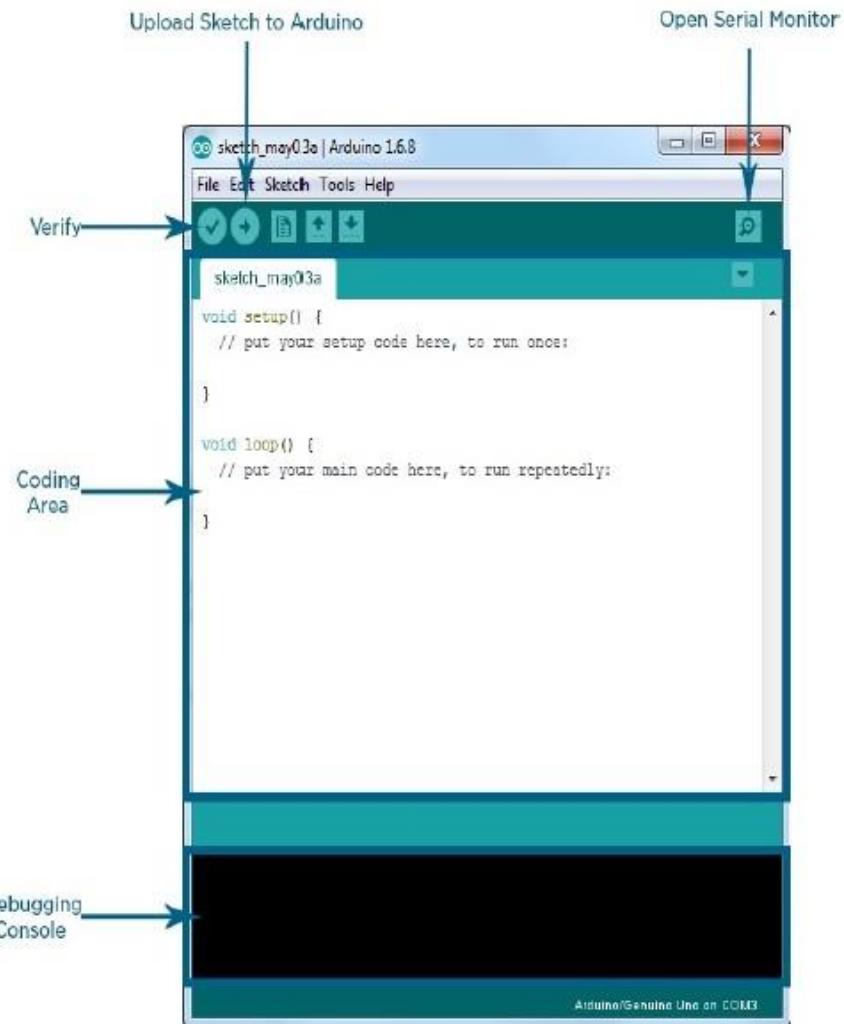
Choose the installation directory (we suggest to keep the default one)



The process will extract and install all the required files to execute properly the Arduino Software (IDE)

How to used It

The Arduino IDE is incredibly minimalistic, yet it provides a near-complete environment for most Arduino-based projects. The top menu bar has the standard options, including “File” (new, load save, etc.), “Edit” (font, copy, paste, etc.), “Sketch” (for compiling and programming), “Tools” (useful options for testing projects), and “Help”. The middle section of the IDE is a simple text editor that where you can enter the program code. The bottom section of the IDE is dedicated to an output window that is used to see the status of the compilation, how much memory has been used, any errors that were found in the program, and various other useful messages.



2.15 Arduino IDE Libraries

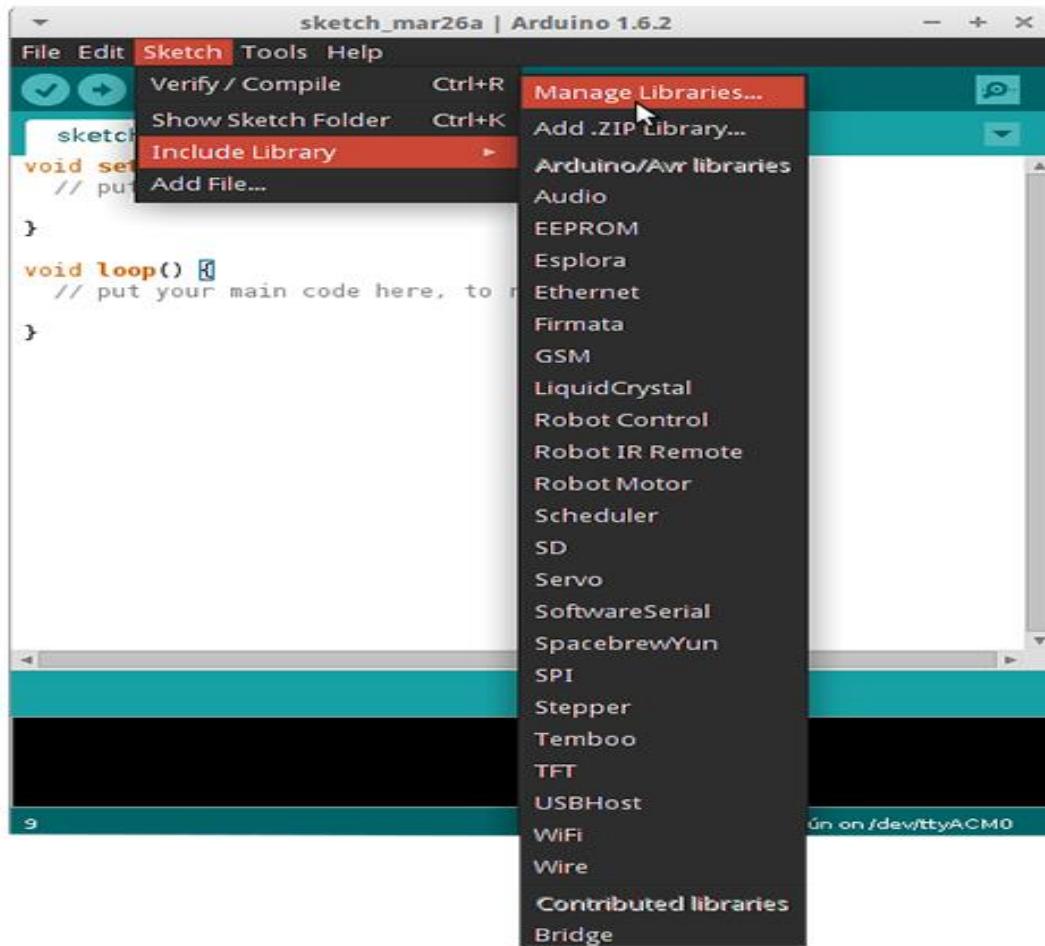
what are Libraries?

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in Liquid Crystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in the reference. To use the additional libraries, you will need to install them.

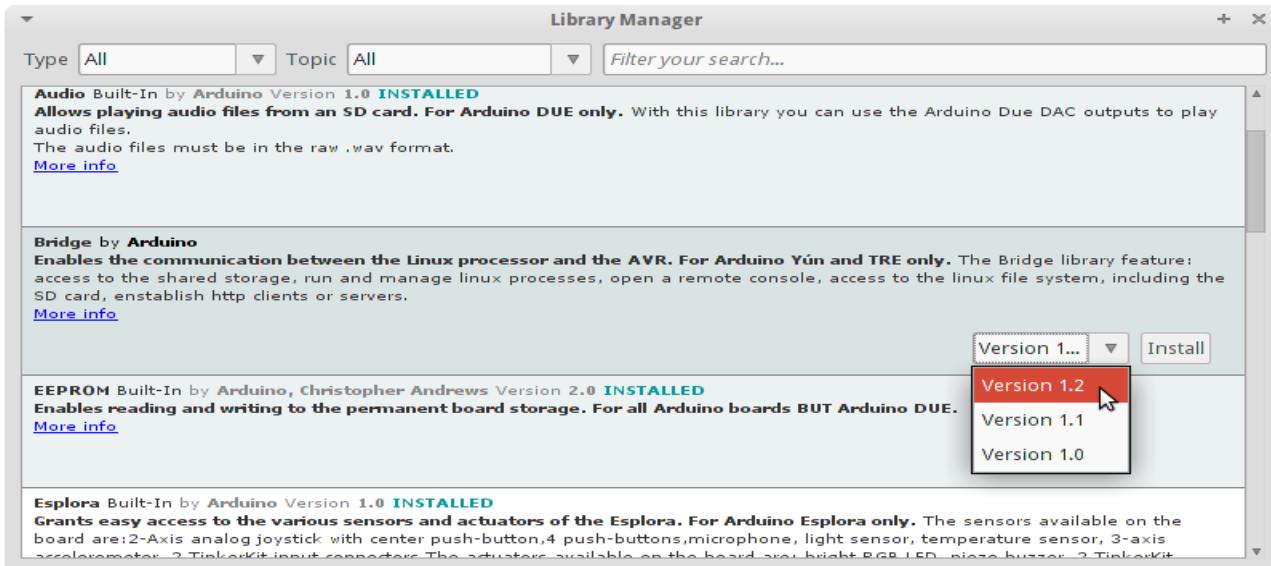
How to Install a Library:

Using the Library Manager.

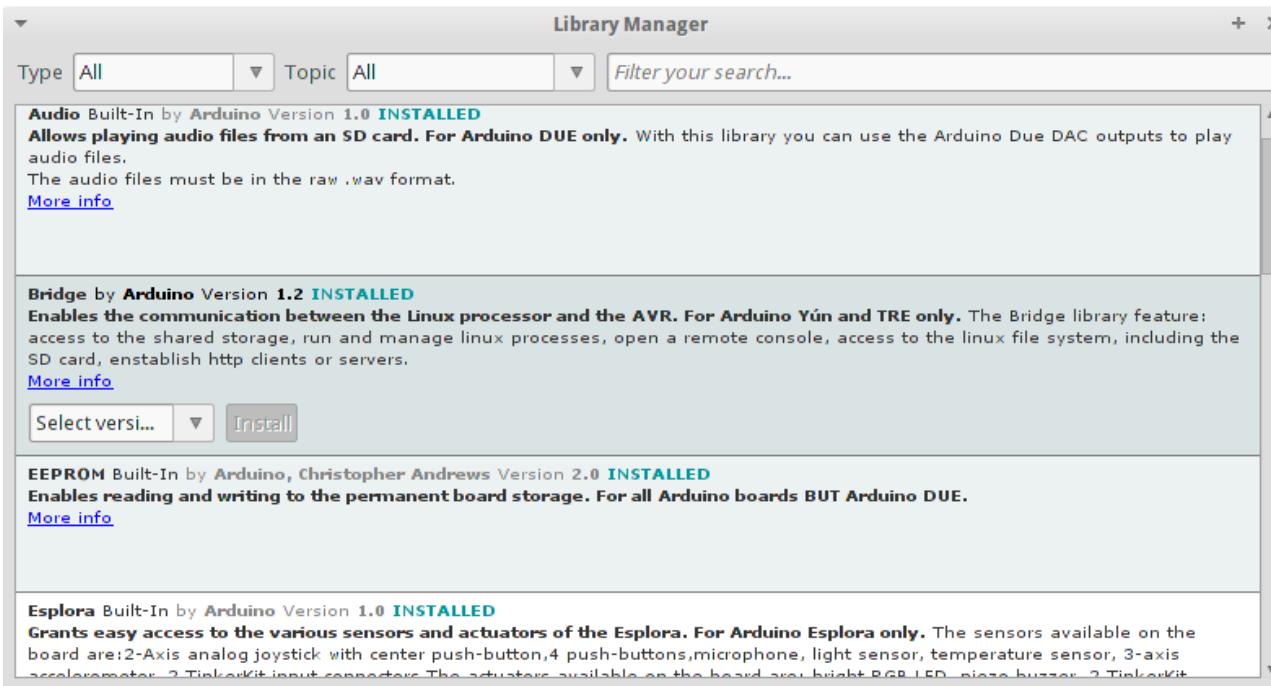
To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.6.2). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



Then the Library Manager will open and you will find a list of libraries that are already installed or ready for installation. In this example, we will install the Bridge library. Scroll the list to find it, click on it, then select the version of the library you want to install. Sometimes only one version of the library is available. If the version selection menu does not appear, do not worry: it is normal.



Finally click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager.



CHAPTER 3

{Hardware Systems}



3.1 Automatic AC Temperature Controller

An AC (Air Conditioner) which was once considered to be a luxury item and was only to be found in big hotels, movie halls, restaurants etc... However, now almost everyone has an AC in our home to beat out the summer/winter and those who have it, worry about one common thing. That is their high electricity consumption and chargers due to it. We are going to make a small **Automatic Temperature Control Circuit** that could minimize the electricity chargers by **varying the AC temperature automatically based on the Rooms temperature**. By varying the set temperature periodically, we can avoid making the AC to work for lower temperature values for a long time and thus making it consume less power.

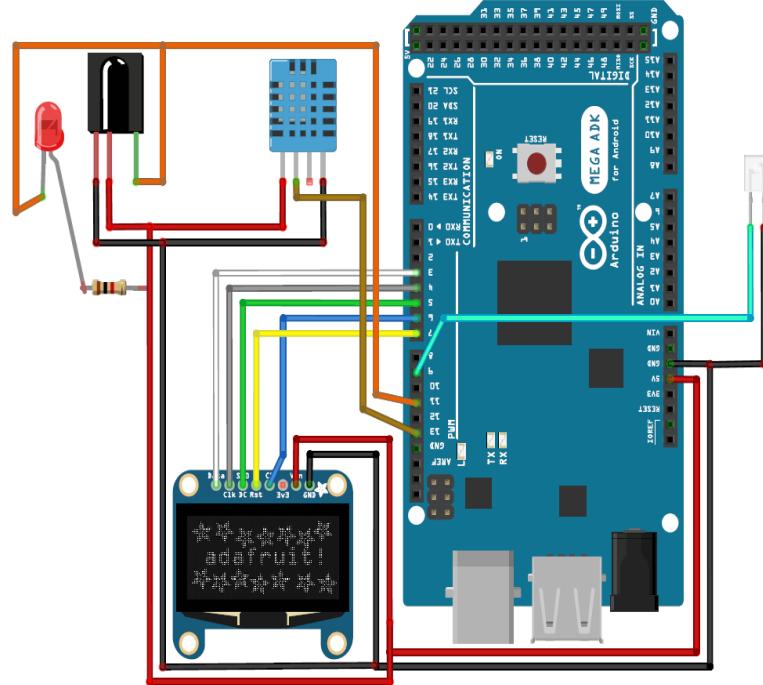
Most of us would have experienced a situation where we have to change the Air Conditioner's set temperature to different values during different times of the day, to keep us comfortable throughout. To automate a process this project uses a **Temperature sensor (DHT11)** which reads the present temperature of the room and based on that value it will send commands to the AC through an **IR blaster** similar to the AC's Remote. The AC will react to these commands as if it is reacting to its Remote and thus adjust the temperature. As your room's temperature changes, the **Arduino** will also adjust your AC's set

temperature to maintain your temperature in just the way you want it to be. Sounds cool right? Let us see how to build one.

Materials Required:

- Arduino Mega 2560
- TSOP1738 (HS0038)
- IR Led
- DHT11 Temperature/Humidity Sensor
- Any Color LED and 1K Resistor(optional)
- Breadboard
- Connecting Wires

Circuit Diagram:



Tables of pins:

The following table can also be used to verify your connections.

S.No:	Component Pin	Arduino Pin
1	OLED – Vcc	5V
2	OLED – Gnd	Gnd
3	OLED- SCK, D0,SCL,CLK	4
4	OLED- SDA, D1,MOSI, Data	3
5	OLED- RES, RST,RESET	7
6	OLED- DC, A0	5
7	OLED- CS, Chip Select	6
8	DHT11 – Vcc	5V
9	DHT11 – Gnd	Gnd
10	DHT11 – Signal	13
11	TSOP – Vcc	5V
12	TSOP – Gnd	Gnd
13	IR Led – Anode	9
14	IR Led – Cathode	Gnd

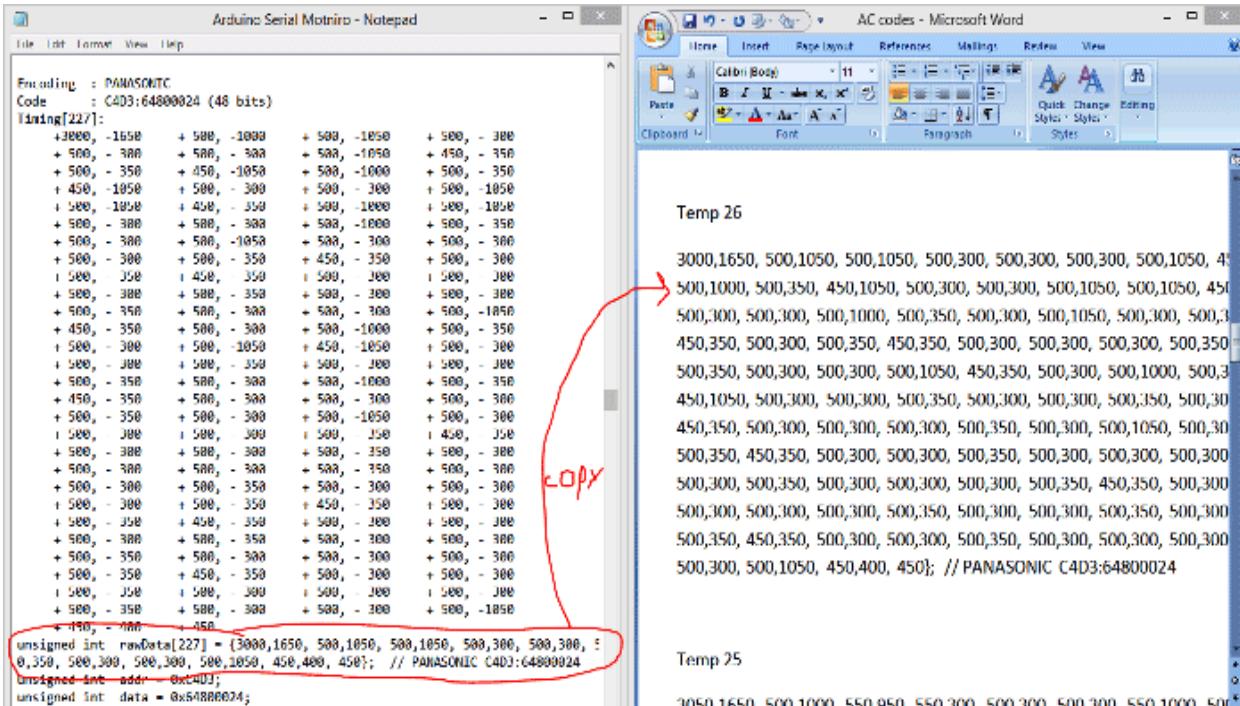
Libraries:

- [IR Remote Library](#) for TSOP and IR Blaster
- [Adafruit Library](#) for OLED
- [GFX Graphics Library](#) for OLED
- [DHT11 Sensor Library](#) for Temperature sensor

Decoding your AC Remote Signals:

The first step to control your AC is to use TSOP1738 to **decode AC Remote Control IR Codes**. Make all the connections as shown in the circuit diagram and make sure you have installed all the mentioned libraries. Now open the example program “*IRrecvDumpV2*” which can be found at *File -> Examples -> IRremote -> IRrecvDumpV2*. Upload the program to your Arduino Mega and open the Serial Monitor.

Point your Remote towards TSOP and press any button, for each button you press its respective Signal will be read by the TSOP1738, decoded by Arduino and displayed in the Serial Monitor. For every change in temperature on your Remote, you will get a different Data. Save this Data for we will be using it in our main program. Your serial monitor will look something like this; I have also shown the Word file on which I have saved the copied data.



Code:

```
#include <IRremote.h> //Lib for IT Blaster and TSOP
#include <SPI.h> // Inbuilt Lib
#include <Wire.h> //Inbuilt Lib
#include <Adafruit_GFX.h> //Lib for OLED
#include <Adafruit_SSD1306.h> //Lib for OLED
#include <dht.h> //Library for dht11 Temperature and Humidity sensor
//(Download from Link in article)
// Assign pins for OLED (Software config.)
#define OLED_MOSI 3
#define OLED_CLK 4
#define OLED_DC 5
```

```
#define OLED_CS 6
#define OLED_RESET 7

Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC,
OLED_RESET, OLED_CS);

#define SSD1306_LCDHEIGHT 64 //Change if you are using a Different
OLED

#define DHT11_PIN 13 //Sensor output pin is connected to pin 13
dht DHT; //Sensor object named as DHT

#define Desired_temperature 27 //The desired temperature is 27*C at
any time

//Decoded Remote Signals For my AC ##CHANGE IT FOR YOUR REMOTE
unsigned int ACoff[] = {4600,2600, 350,400, 350,950, 400,900, 400,350,
400,950, 350,400, 350,400, 350,350, 400,950, 350,400,
350,400, 350,950, 400,350, 400,350, 350,950, 400,350,
400,350, 400,900, 400,950, 350,400, 350,400, 350,400, 350,950,
400,350,
400,350, 350,400, 350,400, 350,400, 350,350, 400,350, 400,350,
400,350, 400,350, 400,350, 400,900, 400,350, 400,350, 400,350,
400,350, 400,350, 400,350, 400,900}; // UNKNOWN 5B04E3C0

unsigned int ACon[] = {4550,2600, 400,350, 400,950, 350,950,
400,350, 400,900, 400,350, 400,350, 400,350, 400,350, 400,900,
400,350,
```

```
400,350, 400,350, 400,950, 350,350, 400,350, 400,950, 400,900,  
400,950, 350,350, 400,950, 400,350, 350,400, 350,400, 350,950,  
400,350,  
  
400,350, 400,350, 400,350, 350,400, 350,400, 350,350, 400,350,  
400,350, 400,350, 400,350, 400,950, 350,400, 350,350, 400,350,  
400,350,  
  
400,350, 400,350, 400,350, 400,900, 400,350, 400,350, 400,350,  
400,950 }; // UNKNOWN EA45DAC8  
  
unsigned int Temp22[] = {4650,2550, 400,350, 400,900, 400,950,  
350,400, 350,950, 400,350, 400,350, 350,400, 350,400, 350,950,  
400,350,  
  
400,350, 400,350, 350,950, 400,350, 400,350, 400,350, 400,400,  
350,350, 350,400, 350,400, 350,950, 400,350, 400,350, 400,900,  
400,350,  
  
400,350, 400,350, 400,350, 400,350, 400,350, 350,400, 350,400,  
350,400, 350,350, 400,350, 400,950, 350,400, 350,400, 350,400,  
350,350,  
  
400,350, 400,350, 400,350, 400,950, 350,400, 350,350, 400,350,  
400,350 }; // UNKNOWN A7997CC1  
  
unsigned int Temp23[] = {4550,2600, 400,350, 400,950, 350,950,  
400,350, 400,900, 400,350, 400,350, 400,350, 400,350, 400,900,  
400,400,  
  
350,350, 400,350, 400,950, 350,350, 400,350, 400,350, 400,350,  
400,350, 400,350, 400,350, 400,900, 400,350, 400,350, 400,950,  
350,350,
```

```
400,350, 400,350, 400,350, 400,350, 400,350, 400,350, 400,350,  
400,350, 400,350, 400,350, 350,950, 400,350, 400,350, 400,350,  
400,350,  
  
400,350, 350,400, 350,400, 350,950, 400,350, 400,350, 400,350,  
350,950 }; // UNKNOWN A7997CC2  
  
unsigned int Temp24[] = {4600,2600, 350,350, 400,950, 400,900,  
400,350, 400,950, 350,350, 400,350, 400,350, 400,350, 400,950,  
350,400,  
  
350,350, 400,350, 400,950, 350,400, 350,400, 350,950, 400,350,  
400,350, 400,350, 350,400, 350,950, 400,350, 400,400, 350,900,  
400,350,  
  
400,350, 400,350, 400,350, 400,350, 400,350, 400,350, 350,400,  
350,400, 350,400, 350,350, 400,950, 400,350, 350,400, 350,400,  
350,400,  
  
350,350, 400,350, 400,350, 400,950, 350,400, 350,400, 350,350,  
400,350 }; // UNKNOWN 36F17307  
  
unsigned int Temp25[] = {4600,2600, 350,400, 350,950, 400,900,  
400,350, 400,950, 350,400, 350,400, 350,350, 400,400, 350,950,  
400,350,  
  
350,400, 350,400, 350,950, 400,350, 400,350, 400,900, 400,950,  
350,350, 400,350, 400,350, 400,950, 350,400, 350,400, 350,950,  
400,350,  
  
400,350, 400,350, 350,400, 350,400, 350,350, 400,350, 400,350,  
400,350, 400,350, 400,350, 400,950, 350,350, 400,350, 400,350,  
400,350,  
  
350,400, 400,350, 400,350, 400,900, 400,350, 400,350, 350,400,  
400,900 }; // UNKNOWN 97DD0400
```

```
unsigned int Temp26[] = {4600,2600, 400,350, 350,950, 400,950,  
350,400, 350,950, 400,350, 350,400, 350,400, 350,350, 400,950,  
400,350,  
400,350, 350,400, 350,950, 350,450, 350,350, 350,400, 400,350,  
350,950, 400,350, 400,350, 350,950, 400,350, 400,350, 400,950,  
350,400,  
350,350, 400,350, 400,350, 400,350, 350,400, 350,400,  
350,400, 350,400, 350,400, 350,950, 400,350, 400,350, 350,400,  
350,400,  
350,400, 350,400, 350,400, 350,950, 350,400, 350,400, 350,400,  
350,400 }; // UNKNOWN 2186470A
```

```
unsigned int Temp27[] = {4600,2600, 350,400, 350,950, 400,900,  
400,350, 400,950, 350,400, 350,400, 350,350, 400,350, 400,950,  
350,400,  
350,400, 350,400, 350,950, 400,350, 400,350, 350,400, 350,400,  
350,950, 400,350, 350,400, 400,900, 400,350, 400,350, 350,950,  
400,350,  
400,350, 400,350, 400,350, 350,400, 350,450, 300,400, 350,400,  
350,400, 350,400, 350,400, 350,950, 350,400, 350,400, 350,400,  
350,400,  
350,400, 350,400, 350,400, 350,950, 350,400, 350,400, 350,400,  
350,950 }; // UNKNOWN 40C2F924
```

```
unsigned int Temp28[] = {4550,2600, 400,350, 400,950, 350,950,  
400,400, 350,900, 400,350, 400,350, 400,350, 400,350, 400,900,  
400,350,
```

```
400,350, 400,350, 400,950, 350,350, 400,350, 400,950, 350,400,  
350,950, 400,350, 400,350, 400,900, 400,350, 400,350, 400,950,  
350,350,  
  
400,350, 400,350, 400,350, 400,350, 400,400, 350,350, 400,350,  
400,350, 400,350, 350,400, 350,950, 400,350, 400,350, 400,350,  
400,350,  
  
350,400, 350,400, 350,400, 350,950, 400,350, 400,350, 350,400,  
350,400 }; // UNKNOWN F1D819D9  
  
unsigned int Temp29[] = {4600,2600, 400,350, 400,900, 400,950,  
400,350, 350,950, 400,350, 400,350, 400,350, 400,350, 400,900,  
400,350,  
  
400,350, 400,350, 400,900, 400,350, 400,350, 400,350, 400,900,  
400,950, 400,350, 350,400, 350,950, 400,350, 400,350, 400,900,  
400,350,  
  
400,350, 400,350, 400,350, 400,350, 400,350, 400,350, 400,350,  
350,400, 350,400, 350,350, 400,950, 400,350, 350,400, 350,400,  
350,400,  
  
350,350, 400,350, 400,350, 400,950, 350,400, 350,400, 350,350,  
400,950 }; // UNKNOWN D6C4468C  
  
unsigned int Temp30[] = {4650,2550, 400,350, 400,950, 350,950,  
400,350, 400,900, 400,350, 400,350, 400,400, 350,350, 400,900,  
400,350,  
  
400,350, 400,400, 350,950, 350,350, 400,350, 400,350, 400,950,  
350,950, 400,350, 400,350, 400,900, 400,350, 400,400, 350,950,  
350,400,
```

```
350,350, 400,350, 400,350, 400,350, 400,350, 400,350, 400,350,  
400,350, 400,350, 350,400, 350,950, 400,350, 400,350, 400,350,  
400,350,  
350,400, 350,400, 350,400, 350,950, 400,350, 400,350, 350,400,  
350,400 }; // UNKNOWN D6C4468F  
  
IRsend irsend;  
  
int Measured_temp;  
  
int Measured_Humi;  
  
int AC_Temp;  
  
char temp_error = 2;  
  
int Pev_value;  
  
boolean AC = false;  
  
int khz = 38; // 38kHz carrier frequency for the NEC protocol  
  
void setup()  
{  
    Serial.begin(9600);  
    display.begin(SSD1306_SWITCHCAPVCC);  
    display.clearDisplay();  
}  
  
void loop() {  
    DHT.read11(DHT11_PIN); //Read the Temp and Humidity  
    Measured_temp = DHT.temperature + temp_error;  
    Measured_Humi = DHT.humidity;
```

```
// text display tests

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0,0);

display.print("Temperature: ");

display.print(Measured_temp);display.println("C");

display.setCursor(0,10);

display.print("Humidity: ");

display.print(Measured_Humi);display.println("%");

display.setCursor(0,20);

display.print("AC Temperature: ");

display.print(AC_Temp);display.println("C");

display.display();

delay(500);

display.clearDisplay();

if ((Measured_temp <= (Desired_temperature-3)) && AC == true) //If
AC is turned on and temperature is less than 3 degree of Desired value
#24 turn off

{
    irsend.sendRaw(ACoff, sizeof(ACoff) / sizeof(ACoff[0]), khz);
    delay(2000);//Send signal to Turn Off the AC

    AC_Temp = 0; AC=false;
}
```

```

if ((Measured_temp >= Desired_temperature+4) && AC == false) //If
AC is off and measured Temp is greater than Desired Temp

{
    irsend.sendRaw(ACon, sizeof(ACon) / sizeof(ACon[0]), khz);
    delay(2000); //Send Signal to Turn On the AC

    delay(2000);

    irsend.sendRaw(Temp27, sizeof(Temp27) / sizeof(Temp27[0]), khz);
    //Send signal to set 27*C

    AC_Temp = 27; AC=true;

}

if ( Measured_temp != Pev_value) //Change the temperature only if the
measured voltage value changes

{
    if (Measured_temp == Desired_temperature+3) //If AC is ON and
measured temp is very very high than desired

    {
        irsend.sendRaw(Temp24, sizeof(Temp24) / sizeof(Temp24[0]), khz);
        delay(2000); //Send signal to set 24*C

        AC_Temp = 24;

    }

    if (Measured_temp == Desired_temperature+2) //If AC is ON and
measured temp is very high than desired

    {

```

```

    irsend.sendRaw(Temp25, sizeof(Temp25) / sizeof(Temp25[0]), khz);
delay(2000); //Send signal to set 25*C

    AC_Temp = 25;

}

if (Measured_temp == Desired_temperature+1) //If AC is ON and
measured temp is very high than desired

{
    irsend.sendRaw(Temp26, sizeof(Temp26) / sizeof(Temp26[0]), khz);
delay(2000); //Send signal to set 26*C

    AC_Temp = 26;

}

if (Measured_temp == 27 ) //If AC is ON and measured temp is desired
value

{
    irsend.sendRaw(Temp27, sizeof(Temp27) / sizeof(Temp27[0]), khz);
//Send signal to set 27*C

    AC_Temp = 27;

}

if (Measured_temp == Desired_temperature-1) //If AC is ON and
measured temp is low than desired value

{
    irsend.sendRaw(Temp28, sizeof(Temp28) / sizeof(Temp28[0]), khz);
delay(2000); //Send signal to set 28*C

    AC_Temp = 28;

```

```
}

if (Measured_temp == Desired_temperature-2 ) //If AC is ON and
measured temp is very low than desired value

{

    irsend.sendRaw(Temp29, sizeof(Temp29) / sizeof(Temp29[0]), khz);
delay(2000); //Send signal to set 29*C

    AC_Temp = 29;

}

if (Measured_temp == Desired_temperature-3 ) //If AC is ON and
measured temp is very very low desired value

{

    irsend.sendRaw(Temp30, sizeof(Temp30) / sizeof(Temp30[0]), khz);
delay(2000); //Send signal to set 30*C

    AC_Temp = 30;

}

Pev_value = Measured_temp;

}
```

3.2 Fingerprint Door Lock

Fingerprint locks operate by scanning and converting your fingerprint data into a numerical template. Once you place your finger onto the scanner for the first time the conversion into numerical data takes place, and the fingerprint template saved. This process then repeated every time you want to grant someone access. The next time someone places his/her finger on the sensor, it matches the data obtained through the finger with the pre-saved values. If a match is found, access is granted and the door opens. On the other hand, if it is someone else trying to get through, access is not allowed and the door remains locked.

Materials Required:

- Arduino UNO
- R307 Fingerprint
- 12V Solenoid lock
- TIP122
- Battery 12V
- Breadboards
- Connecting Wires

Tables of pins:

S.No:	Component Pin	Arduino Pin	Batterer pin
1	Finger-VSS	5v	
2	Finger-GND	GND	
3	Finger-RX	Pin 2	
4	Finger-TX	Pin 3	
5	Lock- Anode		+
6	Lock- Cathode		
7	TIP-Base	GND	-
8	TIP-Collector	Lock-cathode	
9	TIP-Emitter	Pin 8	

Libraries:

- [Adafruit_Fingerprint.h](#)
- [SoftwareSerial.h](#)

Code:

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
int getFingerprintIDez();
SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
void setup()
{
    Serial.begin(9600);
    pinMode(8,OUTPUT);
    finger.begin(57600);
}
void loop()          // run over and over again
{
    getFingerprintIDez();
    delay(50);      //don't ned to run this at full speed.
}
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;
Serial.print("Found ID #"); Serial.print(finger.fingerID);
Serial.print(" with confidence of "); Serial.println(finger.confidence);
if(finger.fingerID==2){
    digitalWrite(8,HIGH);
    delay(3000);
    digitalWrite(8,LOW);
}
return finger.fingerID;
}
```

Another method for control by mobile application

Materials Required:

- 12V Solenoid lock
- Battery 12V
- Relay
- Node-MCU V3
- Breadboards
- Connecting Wires

Libraries:

- [ESP8266WiFi.h](#)
- [WiFiClient.h](#)
- [ESP8266WebServer.h](#)

Tables of pins:

S.No:	Component Pin	Batterer	Node-MCU	Relay
1	Lock- Anode	+		
2	Lock- Cathode	-		
3	Relay- NO	+	NO	
4	Node- GND	-		
5	Node- 3V			VSS
6	Node- GND			GND
7	Lock- Anode		D1	OUT
8	Lock- Cathode	-	GND	COM

Code:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
const char *ssid = "popos";
const char *password = "123456789";
int stateLED = HIGH;
ESP8266WebServer server(80);
void handleRoot() {
    response();
}
void handleLedOn() {
    stateLED = LOW;
    digitalWrite(LED_BUILTIN, stateLED);
    delay(3000);
    stateLED = HIGH;
    digitalWrite(LED_BUILTIN, stateLED);
    response();
}
void handleLedOff() {
    stateLED = HIGH;
```

```

digitalWrite(LED_BUILTIN, stateLED);

response();

}

const String HtmlHtml = "<html><head>

<meta name=\"viewport\" content=\"width=device-width, initial-
scale=1\" /></head>";

const String HtmlHtmlClose = "</html>";

const String HtmlTitle = "<h1>Arduino-er: ESP8266 AP WebServer
exercise</h1><br/>\n";

const String HtmlLedStateLow = "<big>LED is now
<b>ON</b></big><br/>\n";

const String HtmlLedStateHigh = "<big>LED is now
<b>OFF</b></big><br/>\n";

const String HtmlButtons =
"<a href=\"opendoor\"><button style=\"display: block; width:
100%;\">ON</button></a><br/>

<a href=\"LEDOFF\"><button style=\"display: block; width:
100%;\">OFF</button></a><br/>";

void response(){

String htmlRes = HtmlHtml + HtmlTitle;

if(stateLED == HIGH){

    htmlRes += HtmlLedStateLow;

}else{

```

```
htmlRes += HtmlLedStateHigh;  
}  
  
htmlRes += HtmlButtons;  
htmlRes += HtmlHtmlClose;  
server.send(200, "text/html", htmlRes);  
}  
  
void setup() {  
    delay(1000);  
    Serial.begin(9600);  
    Serial.println();  
    WiFi.softAP(ssid, password);  
    IPAddress apip = WiFi.softAPIP();  
    Serial.print("visit: \n");  
    Serial.println(apip);  
    server.on("/", handleRoot);  
    server.on("/opendoor", handleLedOn);  
    //delay(2000);  
    //stateLED = LOW;  
    //digitalWrite(LED_BUILTIN, stateLED);  
    //server.on("/LEDOff", handleLedOff);  
    server.begin();  
    Serial.println("HTTP server beginned");
```

```
pinMode(LED_BUILTIN, OUTPUT);
digitalWrite(LED_BUILTIN, stateLED);
}

void loop() {
    server.handleClient();
}
```

3.3 Light System

The idea was to proof that you can control your home lighting system using your voice commands, our system helps you to be more comfortable and easy to control lights(ON,OFF) in your room ,kitchen and garage....etc.

Materials Required:

- Arduino Uno
- EasyVR 3 Shield
- Led
- Connecting Wire

Libraries:

- [Arduino.h](#)
- [EasyVR.h](#)

Tables of pins:

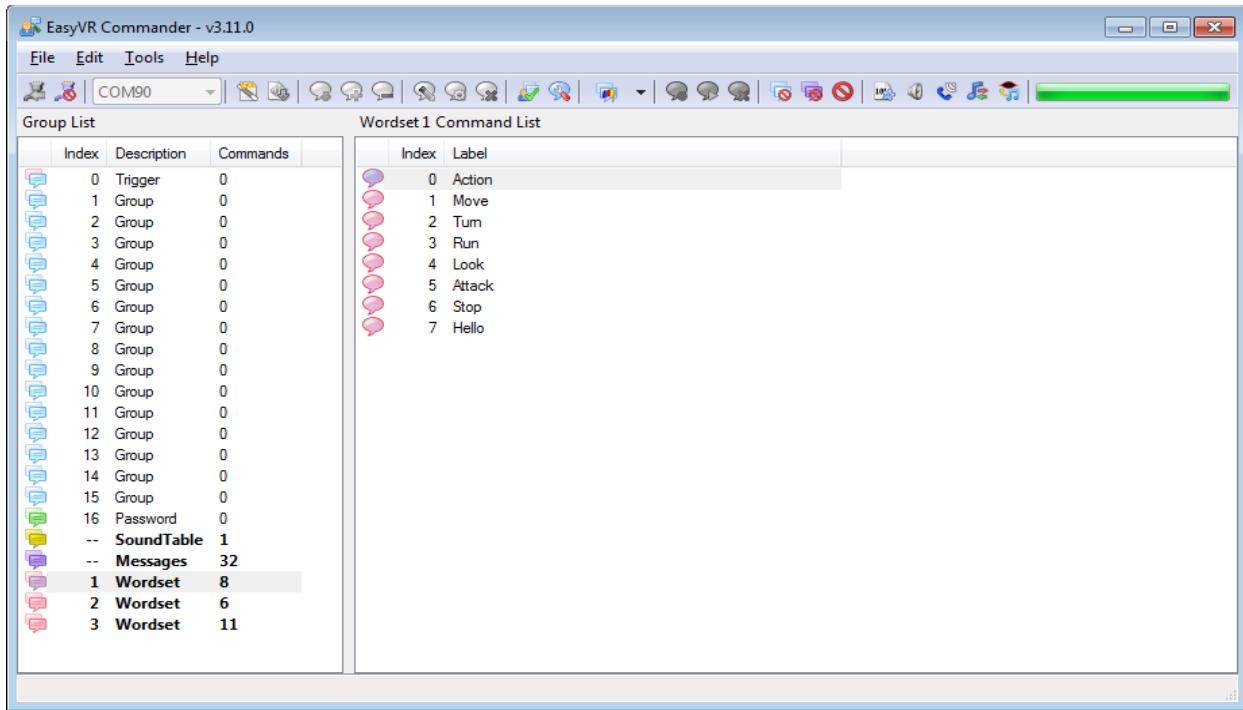
S.No:	Component Pin	EasyVR
1	Led-Anode	Pin 11
2	Led- Cathode	GND

EasyVR Commander

The EasyVR Commander software can be used to easily configure your EasyVR module connected to your PC through a Quick USB cable, an adapter board, or by using the microcontroller host board with the provided “bridge” program (available for ROBONOVA controller board, Arduino, Parallax Basic Stamp). You can define groups of commands or passwords and generate a basic code template to handle them. It is required to edit the generated code to implement the application logic, but the template contains all the functions or subroutines to handle the speech recognition tasks.

Getting Started

Connect the Quick USB cable, an adapter board or a microcontroller host board with a running “bridge” program⁶ to your PC, and then check that all devices are properly turned on and start the EasyVR Commander. Select the serial port to use from the toolbar or the “File” menu, and then go with the “Connect” command.



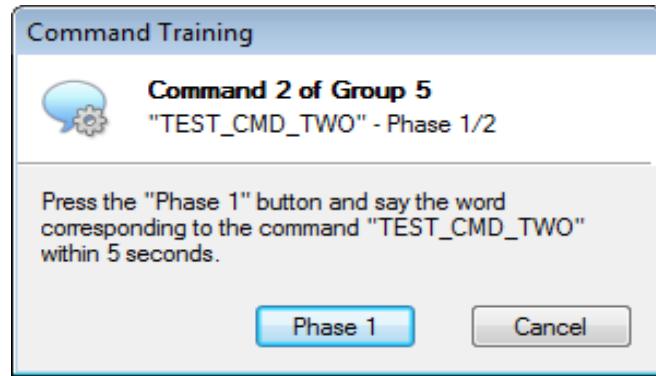
There are five kinds of commands in the software (see Figure 3 and Figure 7):

- .Trigger - special group where you have the built-in SI trigger word "Robot" and you may add one
- User-defined SD trigger word. Trigger words are used to start the recognition process
- Group - where you may add user-defined SD commands, organized in subsets
- Password – special group of "voice passwords" (up to 5), using Speaker Verification (SV) technology
- Word set - built-in set of SI commands (for instance in Figure 3 above, the Word set 1 is selected)

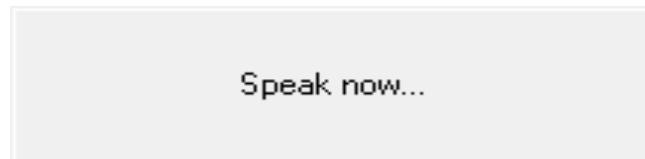
- Grammar – custom set of SI commands (created with QuickT2SITM Lite software).
- There are also two other categories of data shown:
- Sound Table – list of compressed audio samples (prompts, sounds, etc.) loaded into the module
- Messages – run-time sound recordings stored on internal memory

Speech Recognition

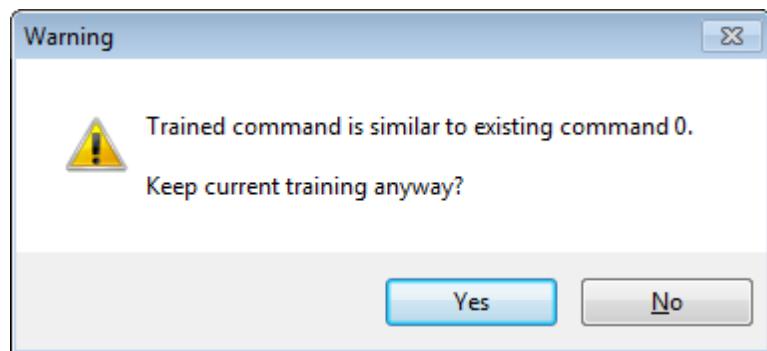
The recognition function of the EasyVR works on a single group at a time, so that users need to group together all the commands that they want to be able to use at the same time. When EasyVR Commander connects to the module, it reads back all the user-defined commands and groups, which are stored into the EasyVR module non-volatile memory. You can add a new command by first selecting the group in which the command needs to be created and then using the toolbar icons or the “Edit” menu. A command should be given a label and then it should be trained twice with the user's voice: the user will be guided throughout this process (see Figure 5) when the “Train Command” action is invoked.



After clicking on “Phase 1” or “Phase 2” buttons, remember you have to start speaking only when you see this little window:

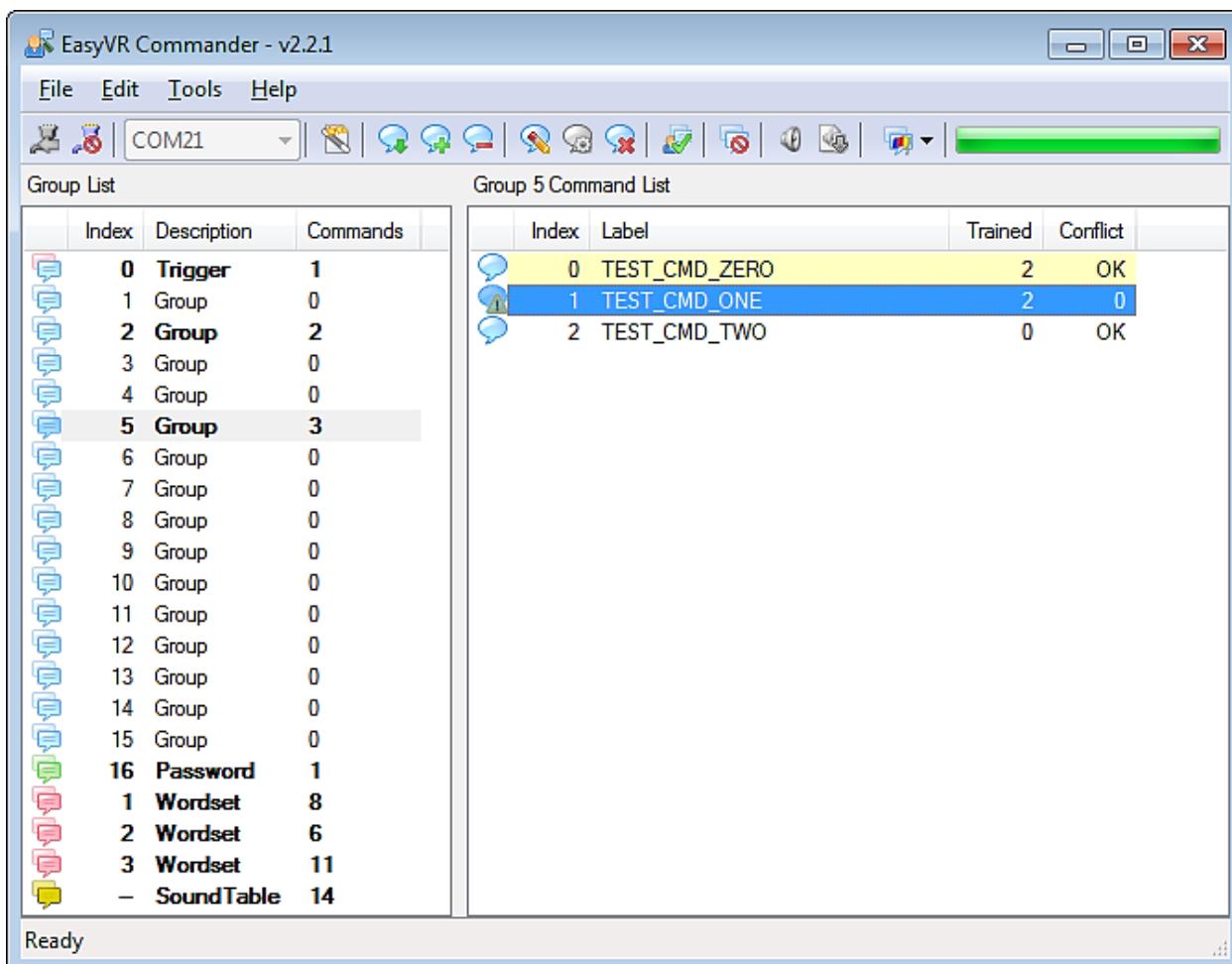


If any error happens, command training will be cancelled. Errors may happen when the user’s voice is not heard correctly, there is too much background noise or when the second word heard is too different from the first one.



The software will also alert if a command is too similar to an existing one by specifying the index of the conflicting command in the "Conflict" column. For example, in the

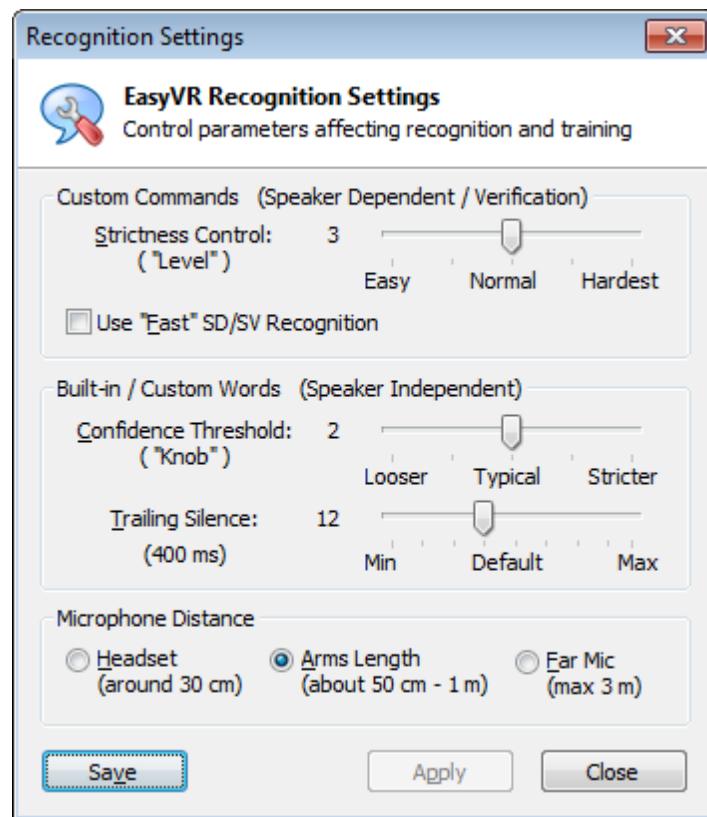
following Figure 7 the command "TEST_CMD_ONE" sounds too similar to "TEST_CMD_ZERO" (i.e. they have been trained with a similar pronunciation).



The status is displayed in the EasyVR Commander list-view where groups that already contain commands are highlighted in bold. The selected group of commands can also be tested, by using the icon on the toolbar or the "Tools" menu, to make sure the trained commands can be recognized successfully.

Recognition Settings

The module comes programmed with some default settings that can affect voice recognition. These parameters can be altered in those cases where the default values do not offer the best performance.



The “Level” and “Knob” parameters affect the way recognition results are evaluated and reported, each one for a different kind of voice recognition algorithm (Speaker Dependent/Verification and Speaker Independent, respectively). Both these values are used for a sort of acceptance threshold: each word or command

recognized is assigned a score by the algorithm, which is compared to the threshold. In some situations, the algorithm may flag a correct result as an error or a low confidence result. In those cases, you may try to lower the threshold and allow more results to be reported as correct. The drawback is that even words that were correctly refused before now might also be accepted. The vice-versa is also true: you can increase the threshold to avoid some incorrect words to be reported as good, but then you may also lose a few correct results. Therefore, in the end, you need to find the best compromise. Ticking “Use “Fast” SD/SV Recognition” will use a faster algorithm for SD and SV recognition. In a similar way, faster SI recognition can be obtained by lowering the Trailing Silence (default is 400 ms). The last parameter affects the internal microphone pre-amplifier and AGC (Automatic Gain Control) stages and is an indication of the expected operating distance of the microphone from the speaker’s mouth. To change the recognition settings of the currently connected EasyVR device press the “Apply” button. The window is non-modal, so you can test the effects of your changes while leaving it open. The “Save” button makes the EasyVR Commander remember your settings and automatically apply them to every connected device. The module itself does not store any option.

Code:

```
#include "Arduino.h"

#if !defined(SERIAL_PORT_MONITOR)

    #error "Arduino version not supported. Please update your IDE to the
latest version."

#endif

#if defined(__SAMD21G18A__)

    // Shield Jumper on HW (for Zero, use Programming Port)

    #define port SERIAL_PORT_HARDWARE

    #define pcSerial SERIAL_PORT_MONITOR

#elif defined(SERIAL_PORT_USBVIRTUAL)

    // Shield Jumper on HW (for Leonardo and Due, use Native Port)

    #define port SERIAL_PORT_HARDWARE

    #define pcSerial SERIAL_PORT_USBVIRTUAL

#else

    // Shield Jumper on SW (using pins 12/13 or 8/9 as RX/TX)

    #include "SoftwareSerial.h"

    SoftwareSerial port(12, 13);

    #define pcSerial SERIAL_PORT_MONITOR

#endif

#include "EasyVR.h"
```

```
EasyVR easyvr(port);

//Groups and Commands

enum Groups
{
    GROUP_0 = 0,
    GROUP_1 = 1,
};

enum Group0
{
    G0_LED = 0,
};

enum Group1
{
    G1_ON = 0,
    G1_OFF = 1,
};

//Grammars and Words

enum Wordsets
{
    SET_1 = -1,
    SET_2 = -2,
    SET_3 = -3,
```

```
};
```

```
enum Wordset1
```

```
{
```

```
    S1_ACTION = 0,
```

```
    S1_MOVE = 1,
```

```
    S1_TURN = 2,
```

```
    S1_RUN = 3,
```

```
    S1_LOOK = 4,
```

```
    S1_ATTACK = 5,
```

```
    S1_STOP = 6,
```

```
    S1_HELLO = 7,
```

```
};
```

```
enum Wordset2
```

```
{
```

```
    S2_LEFT = 0,
```

```
    S2_RIGHT = 1,
```

```
    S2_UP = 2,
```

```
    S2_DOWN = 3,
```

```
    S2_FORWARD = 4,
```

```
    S2_BACKWARD = 5,
```

```
};
```

```
enum Wordset3
```

```
{  
    S3_ZERO = 0,  
    S3_ONE = 1,  
    S3_TWO = 2,  
    S3_THREE = 3,  
    S3_FOUR = 4,  
    S3_FIVE = 5,  
    S3_SIX = 6,  
    S3_SEVEN = 7,  
    S3_EIGHT = 8,  
    S3_NINE = 9,  
    S3_TEN = 10,  
};  
  
// use negative group for wordsets  
int8_t group, idx;  
  
void setup()  
{  
    // setup PC serial port  
    pcSerial.begin(9600);  
  
    bridge:  
    // bridge mode?  
  
    int mode = easyvr.bridgeRequested(pcSerial);
```

```
switch (mode)
{
    case EasyVR::BRIDGE_NONE:
        // setup EasyVR serial port
        port.begin(9600);
        // run normally
        pcSerial.println(F("Bridge not requested, run normally"));
        pcSerial.println(F("---"));
        break;

    case EasyVR::BRIDGE_NORMAL:
        // setup EasyVR serial port (low speed)
        port.begin(9600);
        // soft-connect the two serial ports (PC and EasyVR)
        easyvr.bridgeLoop(pcSerial);
        // resume normally if aborted
        pcSerial.println(F("Bridge connection aborted"));
        pcSerial.println(F("---"));
        break;

    case EasyVR::BRIDGE_BOOT:
        // setup EasyVR serial port (high speed)
        port.begin(115200);
```

```
pcSerial.end();

pcSerial.begin(115200);

// soft-connect the two serial ports (PC and EasyVR)
easyvr.bridgeLoop(pcSerial);

// resume normally if aborted
pcSerial.println(F("Bridge connection aborted"));

pcSerial.println(F("---"));

break;

}

// initialize EasyVR

while (!easyvr.detect())

{

    pcSerial.println(F("EasyVR not detected!"));

    for (int i = 0; i < 10; ++i)

    {

        if (pcSerial.read() == '?')

            goto bridge;

        delay(100);

    }

}

pcSerial.print(F("EasyVR detected, version "));
```

```

pcSerial.print(easyvr.getID());

if (easyvr.getID() < EasyVR::EASYVR3)

    easyvr.setPinOutput(EasyVR::IO1, LOW); // Shield 2.0 LED off

if (easyvr.getID() < EasyVR::EASYVR)

    pcSerial.print(F(" = VRbot module"));

else if (easyvr.getID() < EasyVR::EASYVR2)

    pcSerial.print(F(" = EasyVR module"));

else if (easyvr.getID() < EasyVR::EASYVR3)

    pcSerial.print(F(" = EasyVR 2 module"));

else

    pcSerial.print(F(" = EasyVR 3 module"));

    pcSerial.print(F(", FW Rev."));

    pcSerial.println(easyvr.getID() & 7);

easyvr.setDelay(0); // speed-up replies

easyvr.setTimeout(5);

    easyvr.setLanguage(0); //<-- same language set on EasyVR
Commander when code was generated

group = EasyVR::TRIGGER; //<-- start group (customize)

}

void loop()

{

if (easyvr.getID() < EasyVR::EASYVR3)

```

```
easyvr.setPinOutput(EasyVR::IO1, HIGH); // LED on (listening)

if (group < 0) // SI wordset/grammar
{
    pcSerial.print("Say a word in Wordset ");
    pcSerial.println(-group);
    easyvr.recognizeWord(-group);
}

else // SD group
{
    pcSerial.print("Say a command in Group ");
    pcSerial.println(group);
    easyvr.recognizeCommand(group);
}

do
{
    // allows Commander to request bridge on Zero (may interfere with
    user protocol)

    if (pcSerial.read() == '?')
    {
        setup();
        return;
    }
}
```

```
}

// <<-- can do some processing here, while the module is busy

}

while (!easyvr.hasFinished());

if (easyvr.getID() < EasyVR::EASYVR3)

    easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off

idx = easyvr.getWord();

if (idx == 0 && group == EasyVR::TRIGGER)

{

    // beep

    easyvr.playSound(0, EasyVR::VOL_FULL);

    // print debug message

    pcSerial.println("Word: ROBOT");

    // write your action code here

    // group = GROUP_X\SET_X; <-- jump to another group or wordset

    return;

}

else if (idx >= 0)

{

    // beep

    easyvr.playSound(0, EasyVR::VOL_FULL);

    // print debug message
```

```
uint8_t flags = 0, num = 0;
char name[32];
pcSerial.print("Word: ");
pcSerial.print(idx);
if (easyvr.dumpGrammar(-group, flags, num))
{
    for (uint8_t pos = 0; pos < num; ++pos)
    {
        if (!easyvr.getNextWordLabel(name))
            break;
        if (pos != idx)
            continue;
        pcSerial.print(F(" = "));
        pcSerial.println(name);
        break;
    }
}
// perform some action
action();
return;
}
idx = easyvr.getCommand();
```

```
if (idx >= 0)
{
    // beep
    easyvr.playSound(0, EasyVR::VOL_FULL);
    // print debug message
    uint8_t train = 0;
    char name[32];
    pcSerial.print("Command: ");
    pcSerial.print(idx);
    if (easyvr.dumpCommand(group, idx, name, train))
    {
        pcSerial.print(" = ");
        pcSerial.println(name);
    }
    else
        pcSerial.println();
    // perform some action
    action();
}
else // errors or timeout
{
    if (easyvr.isTimeout())
```

```
pcSerial.println("Timed out, try again...");

int16_t err = easyvr.getError();

if (err >= 0)

{

    pcSerial.print("Error ");

    pcSerial.println(err, HEX);

}

}

}

void action()

{

    switch (group)

    {

        case GROUP_0:

            switch (idx)

            {

                case G0_LED:

                    // write your action code here

                    group = GROUP_1; // group = GROUP_X\SET_X; <-- or jump to
another group or wordset for composite commands

                    break;

            }


```

```
break;

case GROUP_1:
    switch (idx)
    {
        case G1_ON:
            // write your action code here
            digitalWrite(11, HIGH);

            // group = GROUP_X\SET_X; <-- or jump to another group or
            wordset for composite commands

            break;
        case G1_OFF:
            // write your action code here
            digitalWrite(11, LOW);

            // group = GROUP_X\SET_X; <-- or jump to another group or
            wordset for composite commands

            break;
    }
    break;
}
```

Another method for control by mobile application

Materials Required:

- Node-MCU V3
- Breadboards
- led
- Connecting Wires

Libraries:

- [ESP8266WiFi.h](#)
- [WiFiClient.h](#)
- [ESP8266WebServer.h](#)

Tables of pins:

S.No:	Component Pin	Node-MCU
1	Led-Anode	D1
2	Led- Cathode	GND

Code:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
const char *ssid = "popos";
const char *password = "123456789";
int stateLED = LOW;
ESP8266WebServer server(80);
void handleRoot() {
    response();
}
void handleLedOn() {
    stateLED = HIGH;
    digitalWrite(D1, stateLED);
    response();
}
void handleLedOff() {
    stateLED = LOW;
    digitalWrite(D1, stateLED);
    response();
}
const String HtmlHtml = "<html><head>"
```

```
"<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\" /></head>";

const String HtmlHtmlClose = "</html>";

const String HtmlTitle = "<h1>Arduino-er: ESP8266 AP WebServer exercise</h1><br/>\n";

const String HtmlLedStateLow = "<big>LED is now <b>ON</b></big><br/>\n";

const String HtmlLedStateHigh = "<big>LED is now <b>OFF</b></big><br/>\n";

const String HtmlButtons = "<a href=\"LEDOn\"><button style=\"display: block; width: 100%;\">ON</button></a><br/>" + "<a href=\"LEDOFF\"><button style=\"display: block; width: 100%;\">OFF</button></a><br/>";

void response(){

    String htmlRes = HtmlHtml + HtmlTitle;

    if(stateLED == HIGH){

        htmlRes += HtmlLedStateLow;

    }else{

        htmlRes += HtmlLedStateHigh;

    }

    htmlRes += HtmlButtons;

    htmlRes += HtmlHtmlClose;

    server.send(200, "text/html", htmlRes);
}
```

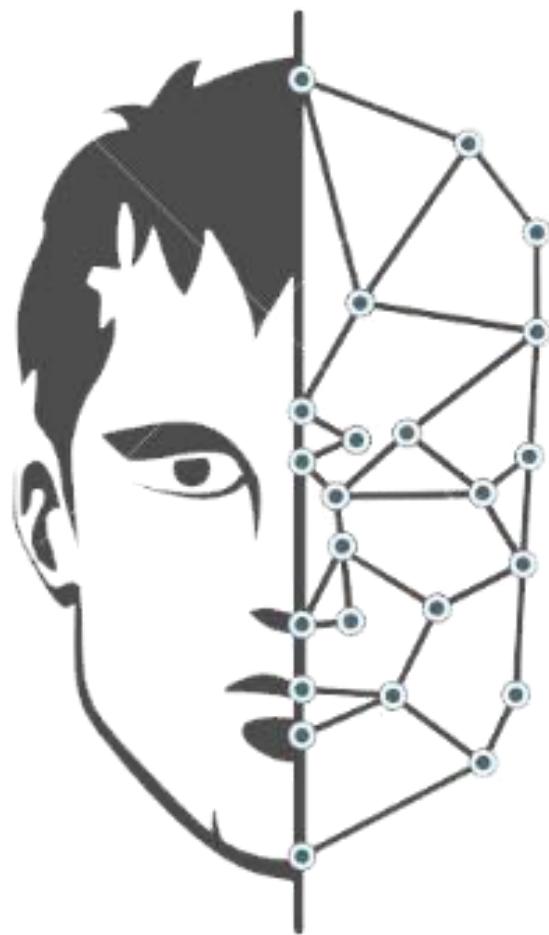
```
}

void setup() {
    delay(1000);
    Serial.begin(9600);
    Serial.println();
    WiFi.softAP(ssid, password);
    IPAddress apip = WiFi.softAPIP();
    Serial.print("visit: \n");
    Serial.println(apip);
    server.on("/", handleRoot);
    server.on("/LEDOn", handleLedOn);
    server.on("/LEDOFF", handleLedOff);
    server.begin();
    Serial.println("HTTP server beginnned");
    pinMode(D1, OUTPUT);
    digitalWrite(D1, stateLED);
}

void loop() {
    server.handleClient();
}
```

CHAPTER 4

{Face Recognition}

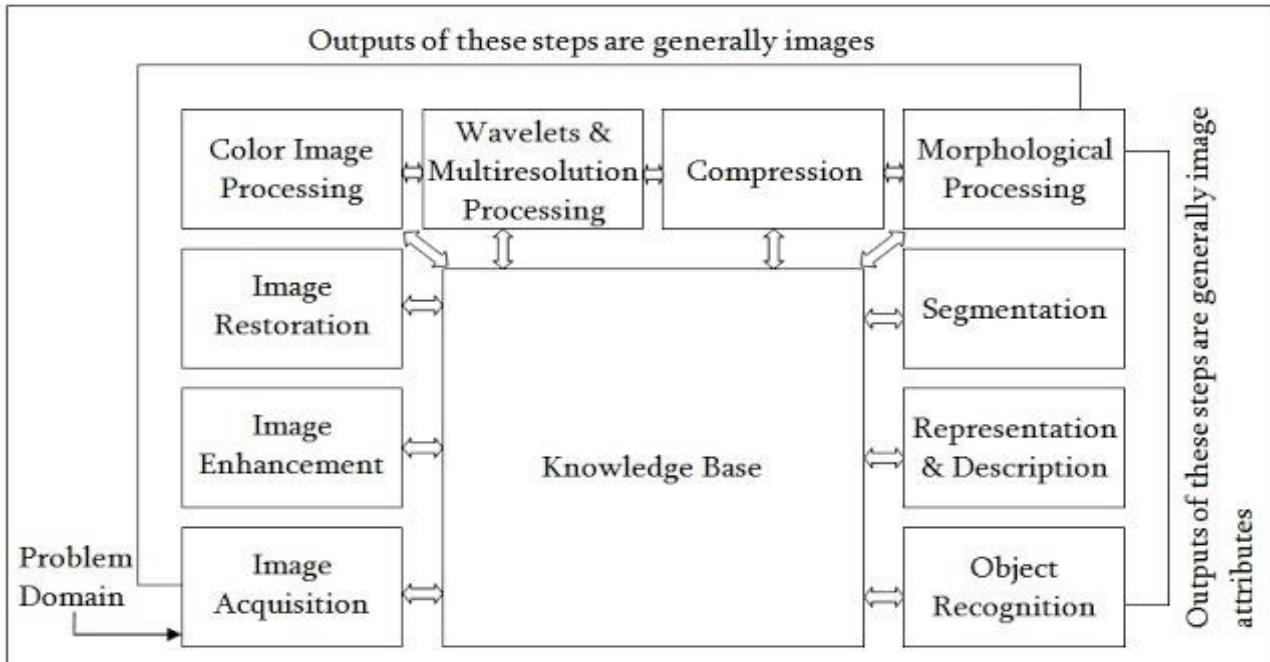


4.1 What is Image Processing?

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.

It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

4.2 Fundamental steps of Image Processing: -



- **Image Acquisition:**

When you are going to digital image processing, you have to preprocess the image in the digital format. It is also known as image acquisition. Impost to realize. it is the first fundaments steps in digital image processing. Here, we need to perform the preprocessing of the digital image as like scaling. When it will complete the image acquisition, it needs to function the image enhancement.

- **Image Enhancement:**

As the images need to use on the digital image processing, it needs to enhance the quality. The reasons for image enhancement is highlighting certain part which is essential. At the same time, it will remove or blur the unnecessary parts of the image. In the *digital image processing*, you need the actual parts which are needed for you. The most appealing parts for image enhancements are changing brightness & contrast etc. That means it needs to keep the parts which are necessary for your work.

- **Image Restoration:**

Image Restoration is another system of image enhancement. Here you need to upgrade the quality of the image. When you try to make the unnecessary parts disappear on the image enhancement, here you need to ensure making appear on the image from damaged or lost.

That means some of the images may be damaged or lost some of the parts, for doing the task successfully you need to restore them. It will help to do the work simultaneously. The most used techniques for image restoration is Point Spread Function (PSF). The process helps to restore the blur or damaged images in the quality one.

- **Color Image Processing:**

The use of the digital image is increasing. Human eyes can read thousands of color and it helps to identify the real objects quickly. With the same process, digital image processing is going forward to gain the significant amount when processing. Generally, a color image is chosen by the properties of brightness, hue, and saturation. In the digital color system, you will see only three primary colors including Cyan, Yellow, and Magenta. With the combination of three primary colors, you will get secondary colors on the color image processing.

- **Wavelets and Multiresolution Processing:**

When you want to represent the images in various degrees or resolution wavelets and multiresolution process works as the foundation. In particular, data compression and for pyramidal representation works to understand the subdivision of images. On the other hand, it needs to figure out the degrees and resolution. With this in mind, you can discover the subdivision of the resolution of the digital images.

- **Compression:**

Compression technology is used to reduce the file size. In the online, it is very popular to share any images or documents. Compression helps to reduce the size of the real documents. With this in mind, compression will support to reduce the use of bandwidth. It will work fast

and you can transfer the data or information as well as the digital images firstly.

- **Morphological Processing:**

Morphology works based on shapes and it is basically an image processing operations. The most used morphology operations are dilation and erosion. To clarify, it is the extracting procedures of the image components. It is necessary to show the description of shape and very useful for representation.

- **Segmentation:**

The primary use of segmentation of an image are partitioning the digital images into multiple segments. That means the main image will be divided into constituent parts or objects. You can ask why segmentation is used? Basically, it helps to make the image and every parts of the image more meaningful. In

general, segmentation tries to find out the objects and boundaries, lines, curves, etc. from the image. The whole process helps to identify the images and its parts individually. Important to realize, autonomous segmentation is a tough work to finish the task beautifully. On the other hand, a rugged segmentation procedure can handle the task of segmentation very carefully.

- **Representation and Description:**

In the fundamental steps in Digital Image processing, it is the output parts of segmentation process. It consists the raw pixel's data. It selects if the segmentation will be one part or the whole parts. It will transform the raw data into computer processing units. The total transform works to represent or output. On the other hand, descriptions parts keep the data or attributes to

differentiate one class of objects from another. It helps to identify the objects easily.

- **Object recognition:**

tell it the introduction of the object. All the objects are different in nature and they have different label. With this in mind, it has the different name and identity according to the descriptions. Suppose, the image is telling it is “Tree”, that means it has the leaf or the shape as like the tree. In the digital image processing system, it will save the description among the image.

- **Knowledge Base:**

Knowledgebase is the helping area to find out or doing the work easily. Here you can find out the genre of an image. You can search the group for information or data from here also. On the other hand, you can filter your search to seek the desired image. At the same time, it needs to get

the image information in a place. After all, you can use for following the steps of image processing.

4.3 What is OpenCV?

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects,

extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitzer, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together,

detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written

natively in C++ and has a templated interface that works seamlessly with STL containers.

4.4 Background On Face Recognition:

When you look at an apple, your mind immediately tells you: that is an apple. This process is recognition in the simplest of terms. So, what's facial recognition? The same, but for faces, obviously.

But, the real question is:

How can a computer recognize a face?

Take a real life example:

When you meet someone for the first time, you don't know who that person is at once, right? While he's talking to you or shaking your hand, you're looking at his face: eyes, nose, mouth, skin tone... This process is your mind gathering data and training for face recognition.

Next, that person tells you that his name is Kirill (yes, our All-Star Data Science Mentor). So, your brain has already gotten the face data, and now it has learned that this data belongs to Kirill.

The next time you see Kirill or see a picture of his face, your mind will follow this exact process:

Face Detection:

Look at the picture and find a face in it.

Data Gathering:

Extract unique characteristics of Kirill's face that it can use to differentiate him from another person, like eyes, mouth, nose, etc.

Data Comparison:

Despite variations in light or expression, it will compare those unique features to all the features of all the people you know.

Face Recognition:

It will determine “Hey, that’s my boy Kirill!”



Our mind's Face Recognition Process

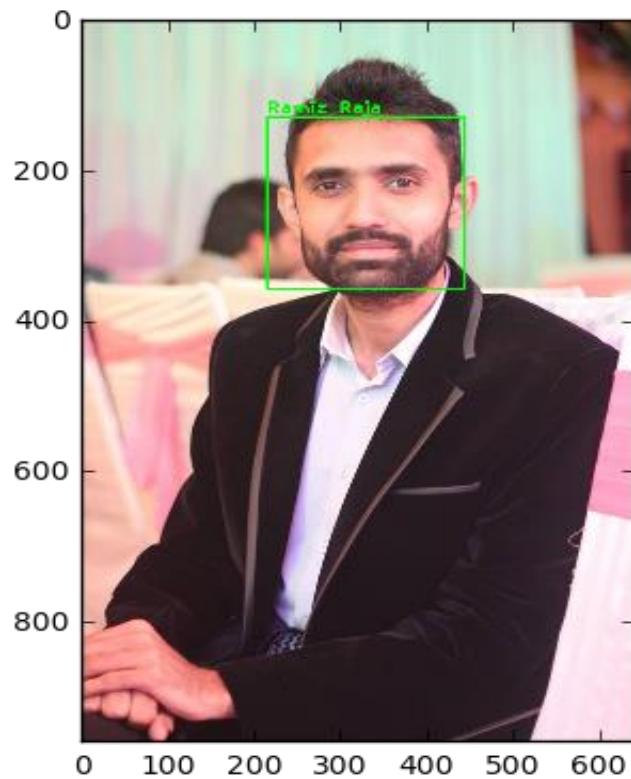
Then, the more you meet Kirill, the more data you will collect about him, and the quicker your mind will be able to recognize him.

4.5 Theory Of OpenCV Face Recognizers:

Thanks to OpenCV, coding facial recognition is now easier than ever. There are three easy steps to computer coding facial recognition, which are similar to the steps that our brains use for recognizing faces. These steps are:

- 1. Data Gathering:** Gather face data (face images in this case) of the persons you want to identify.
- 2. Train the Recognizer:** Feed that face data and respective names of each face to the recognizer so that it can learn.
- 3. Recognition:** Feed new faces of that people and see if the face recognizer you just trained recognizes them. It is that simple!

Moreover, this is how our Face Recognizer will look once we finish coding it:



Hello, it is me.

OpenCV has three built-in face recognizers and thanks to its clean coding, you can use any of them just by changing a single line of code. Here are the names of those face recognizers and their OpenCV calls:

- **EigenFaces** – cv2.face.createEigenFaceRecognizer()

- **FisherFaces** – cv2.face.createFisherFaceRecognizer()
- **Local Binary Patterns Histograms (LBPH)** – cv2.face.createLBPHFaceRecognizer()

You might be wondering:

“Which Face Recognizer should I use and when?”

Here is a summary of each one that will answer that question. Let us rock!

4.6 EIGENFACES FACE RECOGNIZER:

We can write this mathematically as,

$$F = F_m + \sum_{i=1}^n \alpha_i F_i$$

Where,

F Is a new face.

F_m Is the mean or the average face,

F_i Is an EigenFace.

α_i Are scalar multipliers we can choose to create new faces these can be positive or negative.

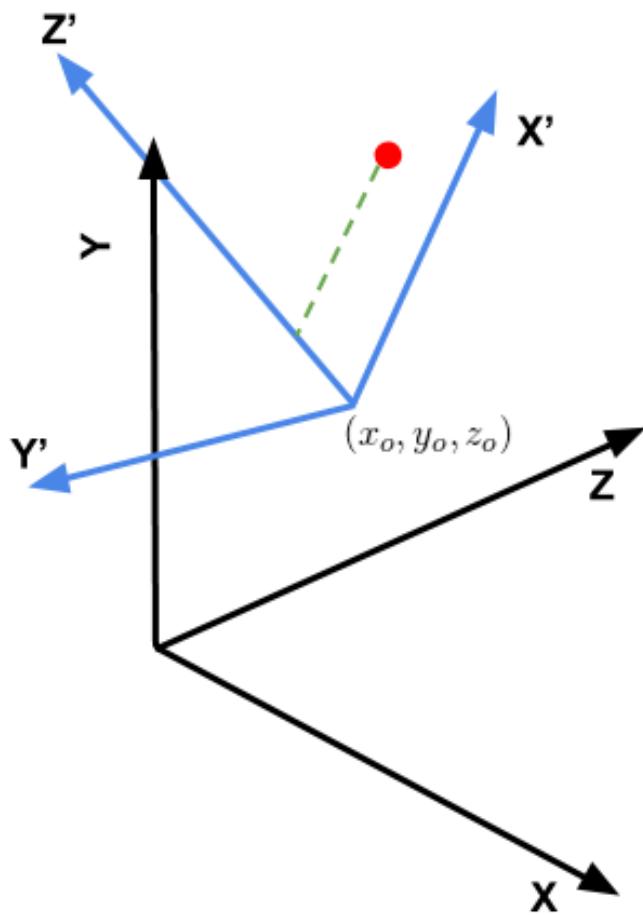
In our previous post, we explained how to calculate the EigenFaces F_i , how to interpret them and how to create new faces by changing weights. α_i

Now suppose, we are given a new facial photo as shown in Figure 1. How can we reconstruct the photo using EigenFaces F_i ? In other words, how do we find the weights α_i that when used in the above equation will produce the facial image as an output? This is exactly the question covered in this post but before we attempt to do that, we need a little background in linear algebra.

Change of coordinates:

Consider a 3D coordinate system X, Y, Z as shown using black in Figure 2. You can imagine another set of

perpendicular axes that is rotated and translated (shifted) by (x_o, y_o, z_o) with respect to the original X, Y and Z frame. In Figure 2, we show the axes of this rotated and translated coordinate system $X'Y'Z'$ in blue. Let us consider a point (shown using the red dot) whose coordinates in the XYZ coordinates is (x, y, z) .



How do we find the coordinates (x', y', z') of the point in the $X'Y'Z'$ coordinate system? This can be done in two steps:

1. Translate: First, we can remove the translation

component from (x, y, z) by subtracting the origin (x_o, y_o, z_o) of the new coordinate system.

Therefore, we have a new vector $(x - x_o, y - y_o, z - z_o)$.

2. Project: Next, we need to

project $(x - x_o, y - y_o, z - z_o)$ onto X', Y', Z' which is nothing but the dot product

of $(x - x_o, y - y_o, z - z_o)$ with the direction X' , Y' and Z' respectively. The green line in Figure 2 shows the projection of the point onto the Z' axis.

This algorithm considers the fact that not all parts of a face are equally important or useful for face recognition. Indeed, when you look at someone, you recognize that person by his distinct features, like the eyes, nose,

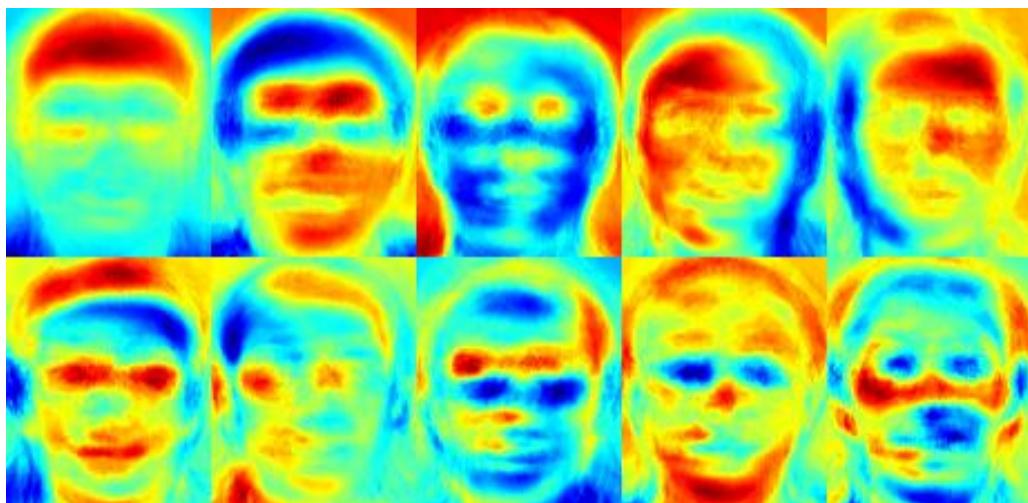
cheeks or forehead; and how they vary respect to each other.

In that sense, you are focusing on the areas of maximum change. For example, from the eyes to the nose there is a significant change, and same applies from the nose to the mouth. When you look at multiple faces, you compare them by looking at these areas, because by catching the maximum variation among faces, they help you differentiate one face from the other.

In this way, is how EigenFaces recognizer works? It looks at all the training images of all the people as a whole and tries to extract the components which are relevant and useful and discards the rest. These important features are called principal components.

Note: We will use the terms: principal components, variance, areas of high change and useful features indistinctly as they all mean the same.

Below is an image showing the variance extracted from a list of faces.



EigenFaces Face Recognizer Principal Components.

Source: [docsopencv.org](https://docs.opencv.org)

You can see that the useful features represent faces which receive the name of Eigen Faces. I mean, how else do you think the algorithm got its name?

So, EigenFaces recognizer trains itself by extracting principal components, but it also **keeps a record** of which ones belong to which person. Thus, whenever you introduce a new image to the algorithm, it repeats the same process as follows:

1. Extract the principal components from the new picture.
2. Compare those features with the list of elements stored during training.
3. Find the ones with the best match.
4. Return the ‘person’ label associated with that best match component.

In simple words, it’s a game of matching.

However, one thing to note in above image is that EigenFaces algorithm also considers illumination as an important feature. In consequence, lights and

shadows are picked up by EigenFaces, which classifies them as representing a ‘face.’

Face recognition picks up on human things, dominated by shapes and shadows: two eyes, a nose, a mouth.

4.7 FISHERFACES FACE RECOGNIZER:

Algorithmic Description:

Let X be a random vector with samples drawn from c classes:

$$\begin{aligned} X &= \{X_1, X_2, \dots, X_c\} \\ X_i &= \{x_1, x_2, \dots, x_n\} \end{aligned}$$

The scatter matrices S_B and S_W are calculated as:

$$\begin{aligned} S_B &= \sum_{i=1}^c N_i (\mu_i - \mu)(\mu_i - \mu)^T \\ S_W &= \sum_{i=1}^c \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T \end{aligned}$$

, where μ is the total mean:

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i$$

And μ_i is the mean of class $i \in \{1, \dots, c\}$:

$$\mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j$$

Fisher's classic algorithm now looks for a projection W that maximizes the class separability criterion:

$$W_{\text{opt}} = \arg \max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

Following, a solution for this optimization problem is given by solving the General Eigenvalue Problem:

$$\begin{aligned} S_B v_i &= \lambda_i S_w v_i \\ S_W^{-1} S_B v_i &= \lambda_i v_i \end{aligned}$$

There's one problem left to solve: The rank of S_w is at most $(N - c)$, with N samples and c classes. In pattern

recognition problems the number of samples N is almost always smaller than the dimension of the input data (the number of pixels), so the scatter matrix S_w becomes singular .this was solved by performing a Principal Component Analysis on the data and projecting the samples into the $(N - c)$ -dimensional space. A Linear Discriminant Analysis was then performed on the reduced data, because S_w is not singular anymore.

The optimization problem can then be rewritten as:

$$W_{\text{pca}} = \arg \max_W |W^T S_T W|$$

$$W_{\text{fld}} = \arg \max_W \frac{|W^T W_{\text{pca}}^T S_B W_{\text{pca}} W|}{|W^T W_{\text{pca}}^T S_W W_{\text{pca}} W|}$$

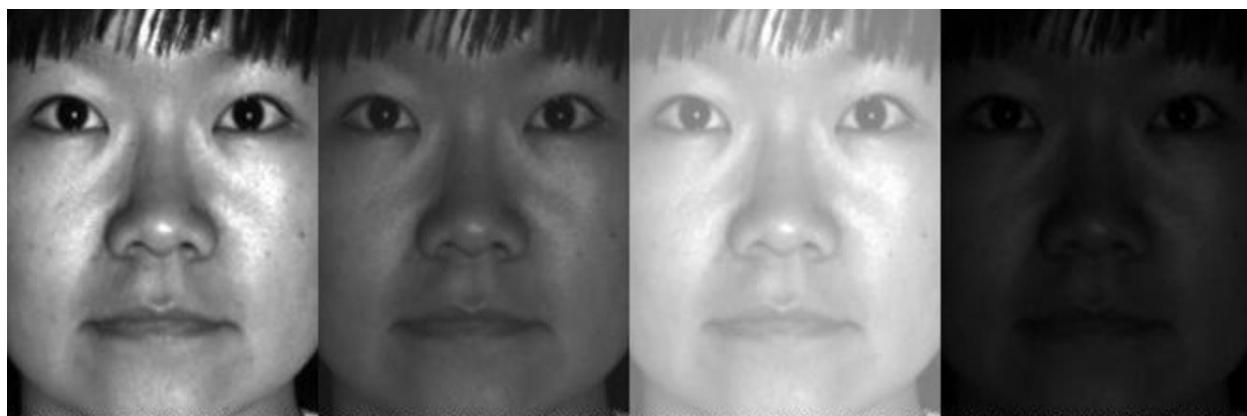
The transformation matrix W that projects a sample into the $(c - 1)$ -dimensional space is then given by:

$$W = W_{\text{fld}}^T W_{\text{pca}}^T$$

This algorithm is an improved version of the last one. As we just saw, **EigenFaces** looks at all the training faces of all the people at once and finds principal components from all of them combined. By doing that, it does not focus on the features that discriminate one individual from another. Instead, it concentrates on the ones that represent all the faces of all the people in the training data, as a whole.

However, here is the kicker:

Consider the lighting changes in following images:



Since EigenFaces also finds illumination as a useful component, it will find this variation very relevant for

face recognition and may discard the features of the other people's faces, considering them less useful. In the end, the variance that EigenFaces has extracted represents just one individual's facial features.

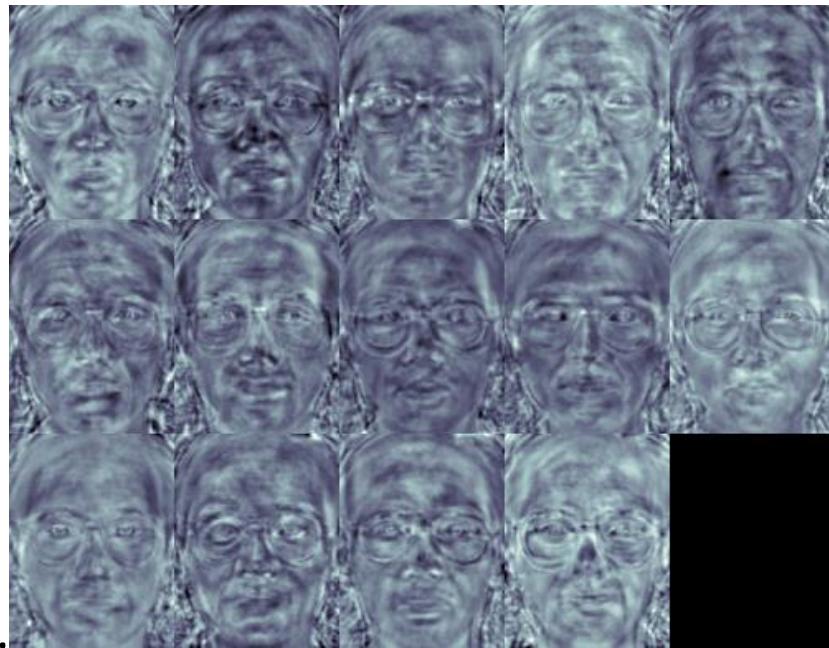
4.8 HOW TO FIX THIS ISSUE?

We can do it by tuning EigenFaces so that it extracts useful features from the faces of each person separately instead of extracting them from all the faces combined.

In this way, even if one person has high illumination changes, it will not affect the other people's features extraction process.

Precisely, **FisherFaces** face recognizer algorithm extracts principal components that differentiate one person from the others. In that sense, an individual's components do not dominate (become more useful) over the others.

Below is an image of principal components using FisherFaces,



FisherFaces Face Recognizer Principal Components.

Source: docs.opencv.org

You can see that the features represent faces which receive the name of Fisher Faces. Are you noticing a theme with the names of the algorithms?

One thing to note here is that FisherFaces only prevents features of one person from becoming dominant, but it

still considers illumination changes as a useful feature. We know that light variation is not a useful feature to extract as it is not part of the actual face.

Then, how to get rid of this problem?

Here is where our next face recognizer comes in.

4.9 LOCAL BINARY PATTERNS HISTOGRAMS (LBPH) FACE RECOGNIZER:

Algorithmic Description:

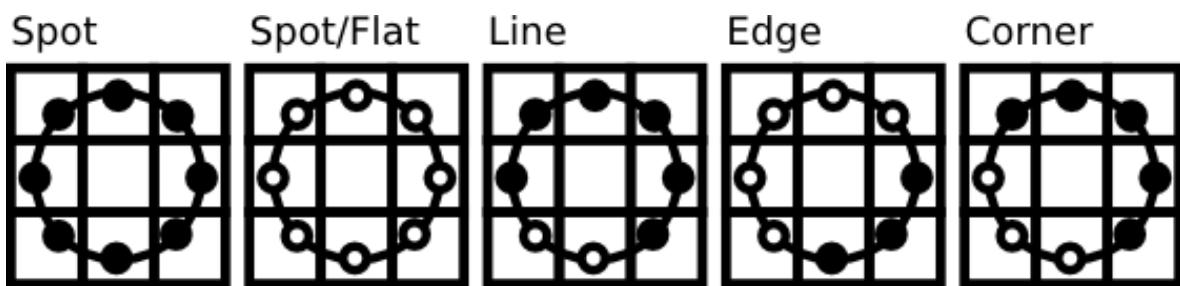
A more formal description of the LBP operator can be given as:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

, with (x_c, y_c) as central pixel with intensity i_c , and i_n being the intensity of the neighbor pixels is the sign function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases} \quad (1)$$

This description enables you to capture very fine grained details in images. In fact, the authors were able to compete with state of the art results for texture classification. Soon after the operator was published it was noted, that a fixed neighborhood fails to encode details differing in scale. So the operator was extended to use a variable neighborhood in [AHP04]. The idea is to align an arbitrary number of neighbors on a circle with a variable radius, which enables to capture the following neighborhoods:



For a given Point (x_c, y_c) the position of the neighbor can

$$(x_p, y_p), p \in P$$

be calculated by:

$$\begin{aligned} x_p &= x_c + R \cos\left(\frac{2\pi p}{P}\right) \\ y_p &= y_c - R \sin\left(\frac{2\pi p}{P}\right) \end{aligned}$$

Where R is the radius of the circle and P is the number of sample points.

The operator is an extension to the original LBP codes, so it is sometimes called *Extended LBP* (also referred to as *Circular LBP*). If a point's coordinate on the circle does not correspond to image coordinates, the point gets interpolated. Computer science has a bunch of clever interpolation schemes; the OpenCV implementation does a bilinear interpolation:

$$f(x, y) \approx [1-x \ x] \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}.$$

I wrote a detailed explanation of Local Binary Patterns Histograms in my previous article on face-detection, which I am sure you have read by now.

Therefore, here I will just give a brief overview of how it works.

We know that Eigenfaces and Fisherfaces are both affected by light and, in real life; we cannot guarantee perfect light conditions. **LBPH** face recognizer is an improvement to overcome this drawback.

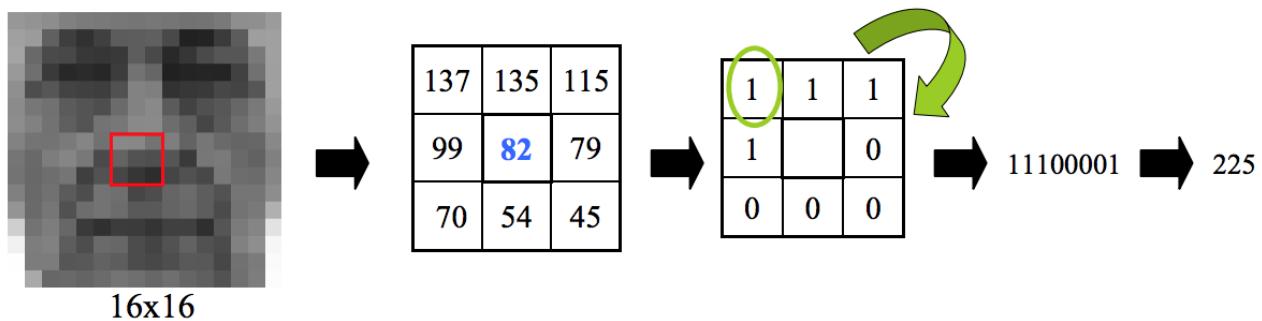
The idea with LBPH is not to look at the image as a whole, but instead, try to find its local structure by comparing each pixel to the neighboring pixels.

THE LBPH FACE RECOGNIZER PROCESS:

Take a 3×3 window and move it across one image. At each move (each local part of the picture), compare the pixel at the center, with its surrounding pixels. Denote the

neighbors with intensity value less than or equal to the center pixel by 1 and the rest by 0.

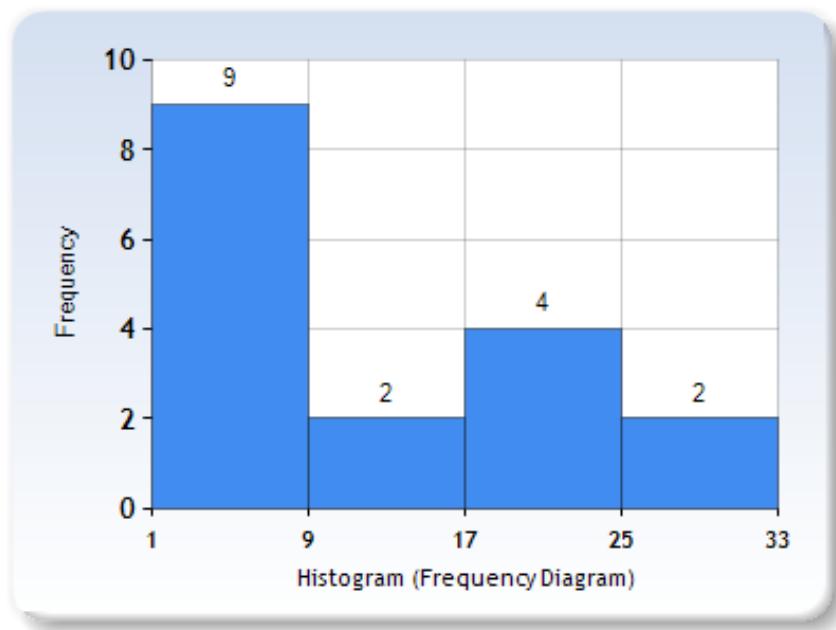
After you read these 0/1 values under the 3×3 window in a clockwise order, you will have a binary pattern like 11100011 that is local to a particular area of the picture. When you finish doing this on the whole image, you will have a list of **local binary patterns**.



LBP conversion to binary. Source: Lopez & Ruiz; Local Binary Patterns applied to Face Detection and Recognition.

Now, after you get a list of **local binary patterns**, you convert each one into a decimal number using binary to

decimal conversion (as shown in above image) and then you make a histogram of all of those decimal values. A sample histogram looks like this:



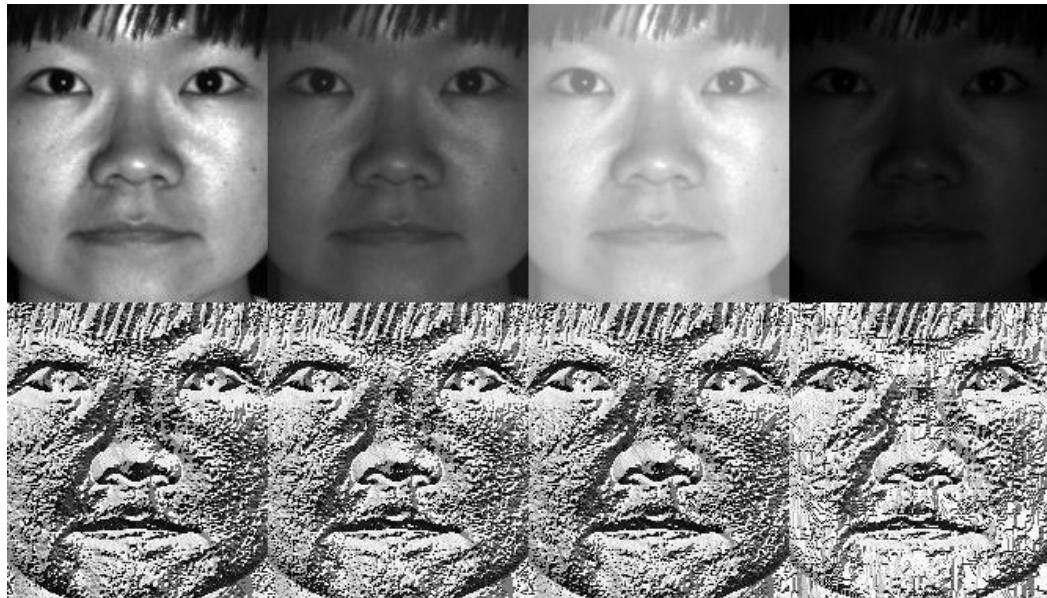
Histogram Sample.

In the end, you will have one histogram for each face in the training data set. That means that if there were 100 images in the training data set then LBPH will extract 100 histograms after training and store them for later recognition. Remember, **the algorithm also keeps track of which histogram belongs to which person.**

Later during recognition, the process is as follows:

1. Feed a new image to the recognizer for face recognition.
2. The recognizer generates a histogram for that new picture.
3. It then compares that histogram with the histograms it already has.
4. Finally, it finds the best match and returns the person label associated with that best match.

Below is a group of faces and their respective local binary patterns images. You can see that the LBP **faces are not affected by changes in light conditions**:



LBPH Face Recognizer Principal Components.

Source: docs.opencv.org

The theory part is over, and now it is time to unmask this moron.

Brace yourself. Here comes the coding!

4.10 CODING FACE RECOGNITION USING PYTHON AND OPENCV:

We are going to divide the Face Recognition process in this tutorial into three steps:

1. Prepare Training Data: Read training images for each person/subject along with their labels, detect faces from each image and assign each detected face an integer label of the person it belongs.

2. Train Face Recognizer: Train OpenCV's LBPH recognizer by feeding it the data we prepared in step 1.

3. Prediction: Introduce some test images to face recognizer and see if it predicts them correctly.

To detect faces, I will use the code from my previous article on face detection.

Before we start the actual coding, we need to install the **Code Dependencies** and import the **Required Modules**:

CODE DEPENDENCIES

Install the following dependencies:

1. OpenCV 3.2.0

2. Python v3.5
3. NumPy that makes computing in Python easy. It contains a powerful implementation of N-dimensional arrays which we will use for feeding data as input to OpenCV functions.

REQUIRED MODULES

Import the following modules:

- cv2: This is the OpenCV module for Python used for face detection and face recognition.
- Os: We will use this Python module to read our training directories and file names.
- Numpy: This module converts Python lists to numpy arrays, as OpenCV face recognizer needs them for the face recognition process.

• PREPARE TRAINING DATA:

The premise here is simple:

The more images used in training, the better.

Being thorough with this principle is important because it is the only way for training a face recognizer so it can learn the different ‘faces’ of the same person; for example: with glasses, without glasses, laughing, sad, happy, crying, with a beard, without a beard, etc.

So, our training data consists of total two people with 12 images of each one. All training data is inside the folder: training-data.

This folder contains one subfolder for every individual, named with the format: slabel (e.g. s1, s2) where the label is the integer assigned to that person. For example, the subfolder called s1 means that it contains images for person 1.

With that in mind, the directory structure tree for training data is as follows:

```
training-data
|----- s1
|       |-- 1.jpg
|       |-- ...
|       |-- 12.jpg
|----- s2
|       |-- 1.jpg
|       |-- ...
|       |-- 12.jpg
```

On the other hand, the folder test-data contains images that we will use to test our face recognition program after we have trained it successfully.

Considering that the OpenCV face recognizer only accepts labels as integers, we need to define a mapping between integer tags and the person's actual name.

So, below I am defining the mapping of a person's integer labels and their respective names.

I have a sneaking suspicion that my follower thief is none other than Elvis Presley. Why else do I keep listening to 1950s rock and roll?

4.11 DATA PREPARATION FOR FACE RECOGNITION:

Perhaps you are thinking:

Why are we talking about preparing data?

Well, to know which face belongs to which person, OpenCV face recognizer accepts information in a particular format. In fact, it receives two vectors:

- One is the faces of all the people.
- The second is the integer labels for each face.

For example, if we had two individuals and two images for each one:

PERSON-1	PERSON-2
img1	img1
img2	img2

Then, the data preparation step will produce following face and label vectors:

FACES	LABELS
person1_img1_face	1
person1_img2_face	1
person2_img1_face	2
person2_img2_face	2

In detail, we can further divide this step into the following sub-steps:

1. Read all the sub folders names provided in the folder training-data. In this tutorial, we have folder names: s1, s2.
2. Extract label number. Remember that all the sub folders containing images of a person following the format: sLabel where Label is an integer representing each person. So for example, folder name: s1 means that the person has label 1, s2 means the person's label is 2, and so on. We will assign the integer

extracted in this step to every face detected in the next one.

3. Read all the images of the person, and apply face detection to each one.
4. Add each face to face vectors with the corresponding person label (extracted in above step)

4.12 Principle component analysis (PCA):

PCA has two useful properties when used in face recognition. The first is that can use to reduce the dimensionality of the feature vector. This dimensionality reduction can be performed in either a lossy or a lossless manor. When applied in lossy manor, basis vector is truncated from the front or back of the transformation matrix. It is assumed that these vector correspond to not useful information such as lighting variation (When dropped from the front) or noise (when dropped from the back).

If none of the basis vector are dropped, it is lossless transformation for training data based on the compressed feature vector.

The second useful property is that PCA eliminates all of the statistical covariance in the transformed feature vectors. This means that covariance matrix for the transformed (training) feature vector will always be diagonal. This property is exploited for some distance measures such as L1, MahAngle and Bayesian based classification.

And always PCA used with Eigenfaces.

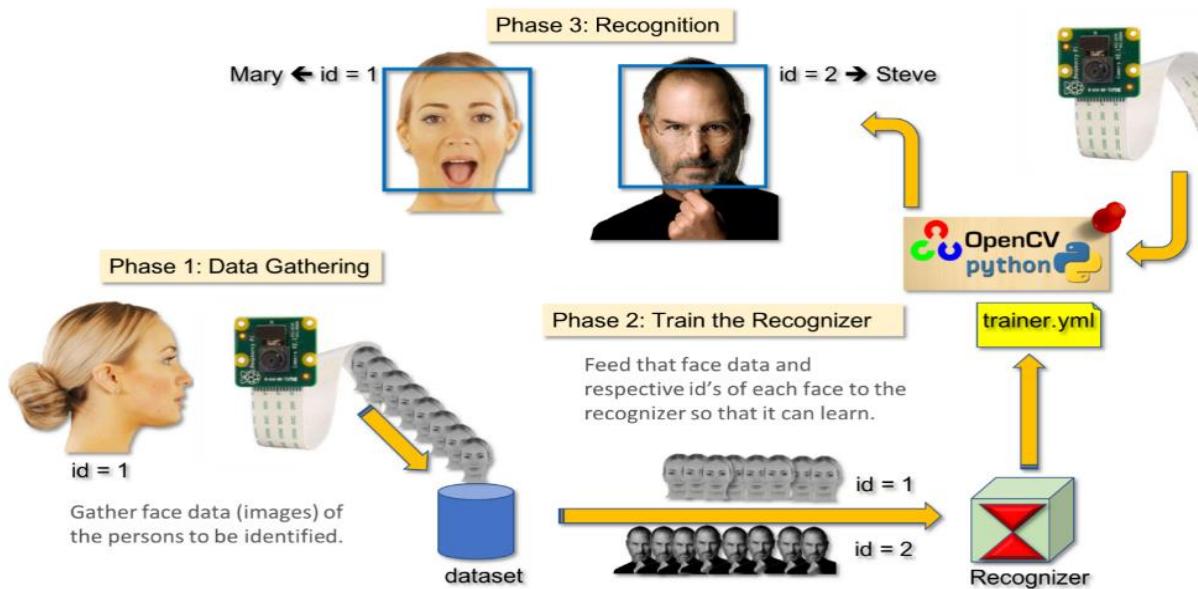
And there are a lot of algorithms that can be used for facial recognition that gave more accuracy such as (artificial neural network, tensorflow, support vector machine (SVM), SURF and SIFT algorithms).

And we will see in implementation chapter some example to use this algorithms and classifiers.

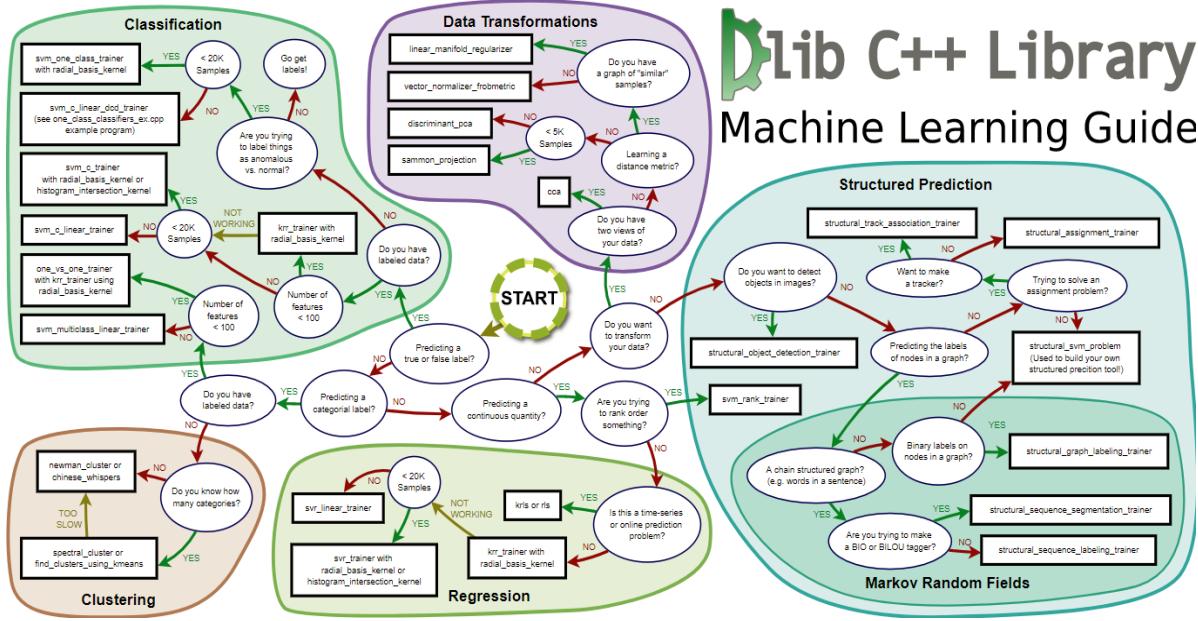
All this algorithms need steps to recognize the face in the picture and compare it with dataset that we had discussed before:

1. Face Detection and Data Gathering
2. Train the Recognizer
3. Face Recognition

The below block diagram resumes those phases:



4.13 Face-recognition library based on Dlib:



What is Dlib?



Dlib is a general purpose cross platform software library written in the programming language C++. Its design is heavily influenced by ideas

from design by contract and component-based software engineering. Thus it is, first and foremost, a set of independent software components. It is open source software released under a Boost Software License.

Since development began in 2002, Dlib has grown to include a wide variety of tools. As of 2016, it contains software components for dealing with networking, threads, graphical user interfaces, data structures, linear algebra, machine learning, image processing, data mining, XML and text parsing, numerical optimization, Bayesian networks, and many other tasks. In recent years, much of the development has been focused on creating a broad set of statistical machine learning tools and in 2009 Dlib was published in the Journal of Machine Learning Research.

4.14 Face-recognition:

Now let us talk about face-recognition library that we already use it in our project: -

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library.

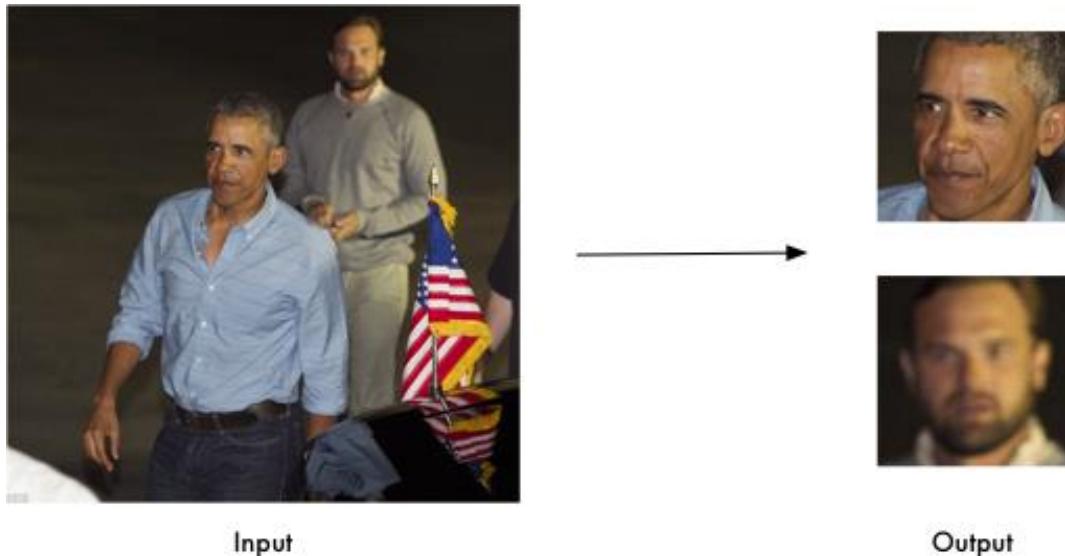
Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.

This also provides a simple face_recognition command line tool that lets you do face recognition on a folder of images from the command line!

This reason why we use it because it is accuracy.

4.15 Feature:

1-Find all the faces that appear in a picture:



```
Import face_recognition
```

```
Image = face_recognition.load_image_file
```

```
("your_file.jpg")
```

```
face_locations = face_recognition.face_locations (image)
```

2-Find and manipulate facial feature in picture:-



Input



Output

Import face_recognition

```
Image =face_recognition.load_image_file("your_file.jpg")
```

```
face_landmarks_list =face_recognition.face_landmarks  
(image)
```

Get the locations and outlines of each person's eyes,
nose, mouth and chin.

3-Identify faces in picture: -

Recognize who appears in each photo.



Input



Picture contains
“Joe Biden”

Output

Import face_recognition

```
known_image =face_recognition.load_image_file  
("biden.jpg")
```

```
unknown_image=face_recognition.load_image_file  
("unknown.jpg")
```

```
biden_encoding=face_recognition.face_encodings  
(known_image) [0]
```

```
unknown_encoding=face_recognition.face_encodings  
(unknown_image) [0]
```

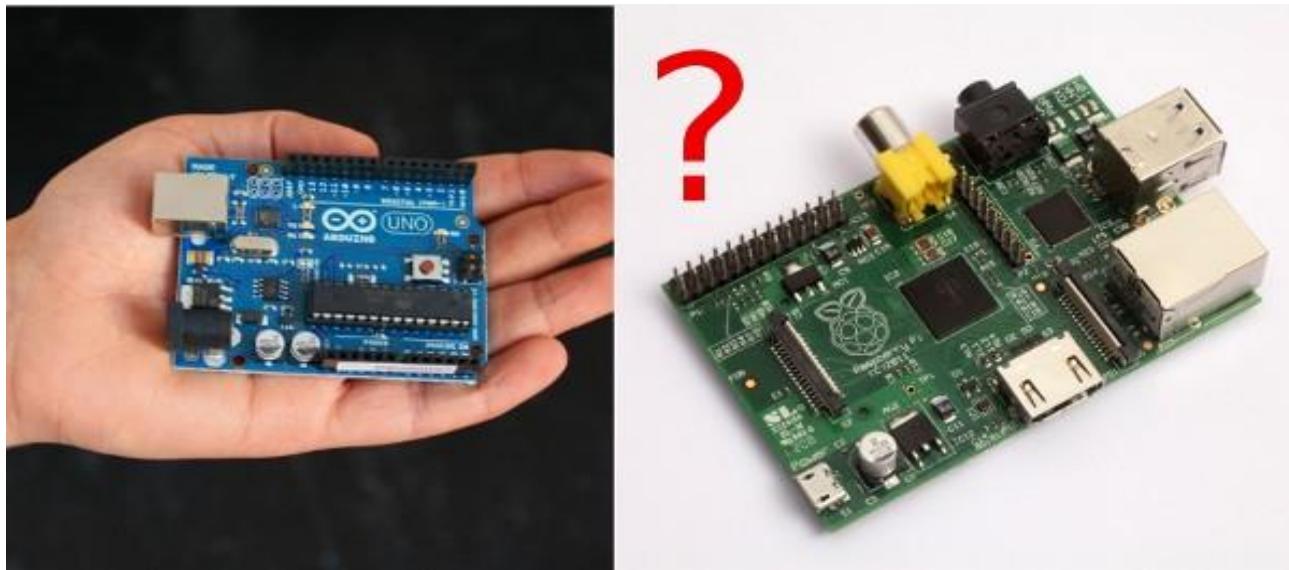
```
Results = face_recognition.compare_faces  
([biden_encoding],unknown_encoding)
```

And we will explain how to use it in implementation chapter

Now let us talk about the component and hardware that used to recognize faces. Moreover, control the DC motor. The basic idea that when camera recognize face of car owner the motor can turn on how it done!

4.16 Controlling facial recognition **(hardware):**

What is different between Raspberry Pi and Arduino?



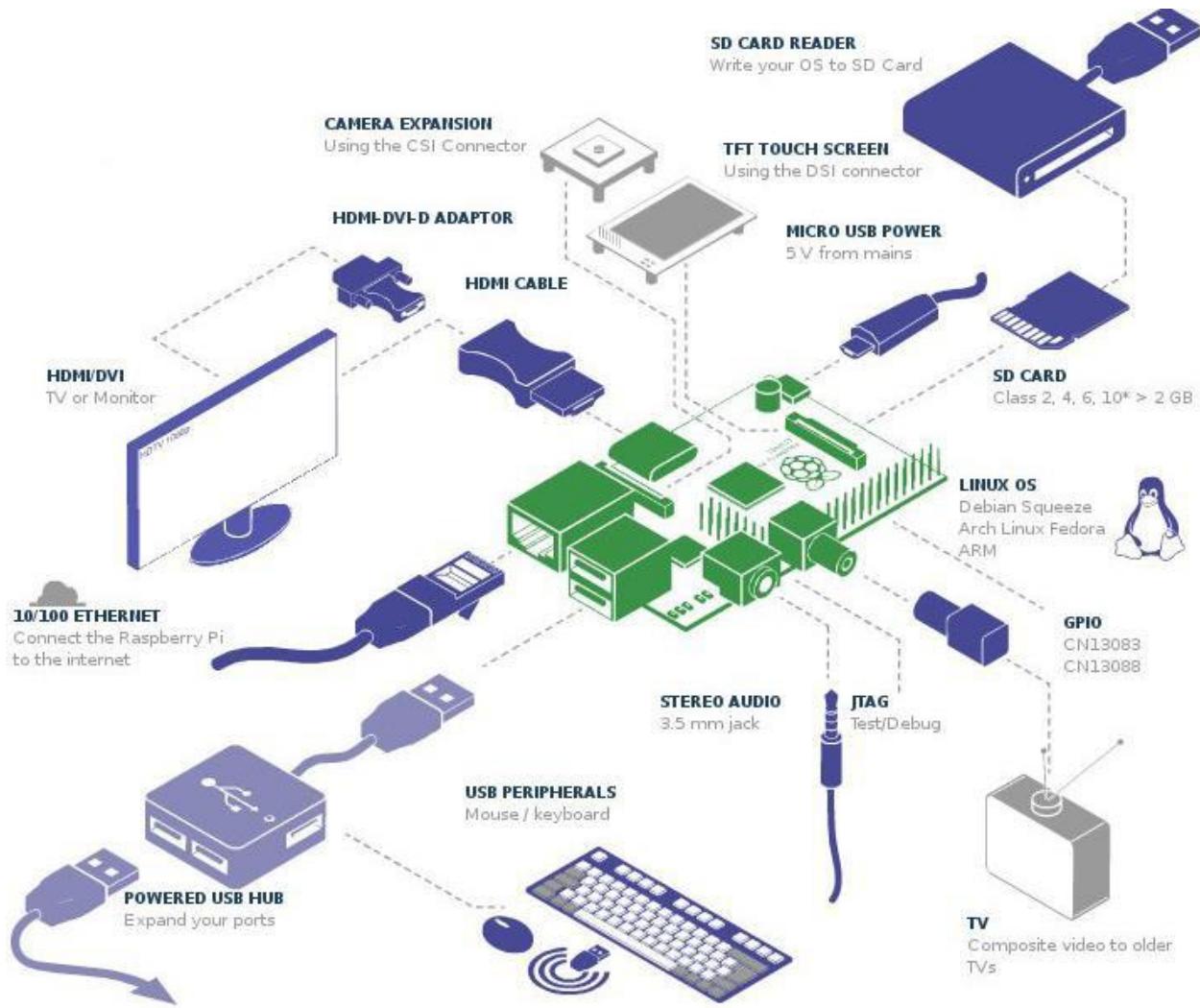
The Arduino Uno (or Mega or any of the other models -- except the Due) is a microcontroller based physical computing platform.

The Raspberry Pi is a microprocessor based single-board computer (SBC). The Arduino Due and the Raspberry Pi are more directly comparable in terms of form and function.

The Arduino employs an 8-bit [ATmega series microcontroller](#) whereas the Raspberry Pi is based around a 32-bit [ARM](#) processor, and the Arduino is typically clocked at between 8-16MHz and with 2-8kB of RAM available, and in contrast, the Raspberry Pi can be clocked at up to 1GHz and may have up to 512MB of RAM. On top of which the Pi has a GPU and video outputs, Ethernet as standard and USB host ports.

So that we decided to use raspberry pi cause, the image processing and facial recognition need a lot of processing with a high speed of run time.

4.17 Introduction for raspberry pi:



This figure can explain in simple way the powerful of raspberry pi

The operating system that used in raspberry pi is Linux (raspbian)

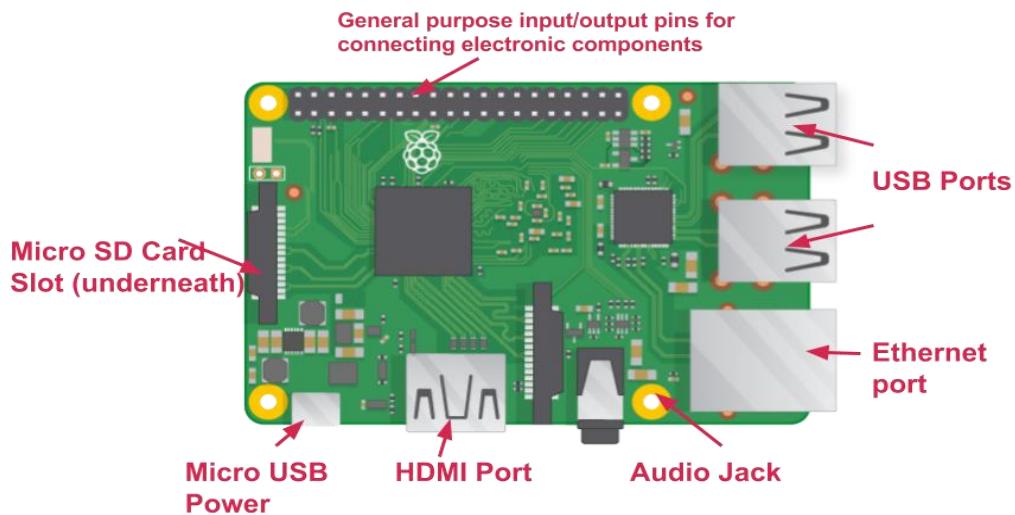
The future that can used in raspberry pi:

- **USB ports** — these are used to connect a mouse and keyboard. You can also connect other components, such as a USB drive.
- **SD card slot** — you can slot the SD card in here. This is where the operating system software and your files are stored.
- **Ethernet port** — this is used to connect the Raspberry Pi to a network with a cable. The Raspberry Pi can also connect to a network via wireless LAN.
- **Audio jack** — you can connect headphones or speakers here.
- **HDMI port** — this is where you connect the monitor (or projector) that you are using to display the output from the Raspberry Pi. If your monitor has speakers, you can also use them to hear sound.

- **Micro USB power connector** — this is where you connect a power supply. You should always do this last, after you have connected all your other components.
- **GPIO ports** — these allow you to connect electronic components such as LEDs and buttons to the Raspberry Pi.

In this project, we used camera to detect and recognize the face

And GPIO to control motors that will be turn on if camera recognize the face and we will explain in briefly how it work.



It also comes with Python a professional programming language used by YouTube, Google, and Industrial Light & Magic (the special

Effects gurus for the Star Wars films), among many others language that can be used to program with it such C/C++.

It has a General Purpose Input/output (GPIO) port on it that you can use to connect up your own circuits to the Raspberry Pi, so you can use your Raspberry Pi to control other devices and to receive and interpret signals from them

4.18 Implementation

1-Face-recognition Dlib code:-

Using webcam :

```
File Edit Format Run Options Window Help
import face_recognition
import cv2
import numpy as np

# Get a reference to webcam #0 (the default one)
video_capture = cv2.VideoCapture(0)
video_capture.set(3,640) # set Width
video_capture.set(4,480) # set Height

# Load a sample picture and learn how to recognize it.
print("Loading known face image(s)")
obama_image = face_recognition.load_image_file("obama.jpg")
obama_face_encoding = face_recognition.face_encodings(obama_image)[0]

# Load a second sample picture and learn how to recognize it.
biden_image = face_recognition.load_image_file("biden.jpg")
biden_face_encoding = face_recognition.face_encodings(biden_image)[0]

# Create arrays of known face encodings and their names
known_face_encodings = [
    obama_face_encoding,
    biden_face_encoding
]
known_face_names = [
    "Barack Obama",
    "Joe Biden"
]

# Initialize some variables
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True
print("Capturing image.")

while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()
```

File Edit Format Run Options Window Help

```
# Initialize some variables
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True
print("Capturing image.")

while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()

    # Resize frame of video to 1/4 size for faster face recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]

    # Only process every other frame of video to save time
    if process_this_frame:
        # Find all the faces and face encodings in the current frame of video
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

        face_names = []
        for face_encoding in face_encodings:
            # See if the face is a match for the known face(s)
            matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
            name = "Unknown"

            # If a match was found in known_face_encodings, just use the first one.
            if True in matches:
                # first_match_index = matches.index(True)
                # name = known_face_names[first_match_index]

            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]
```

```

File Edit Format Run Options Window Help
#     name = known_face_names[first_match_index]

# Or instead, use the known face with the smallest distance to the new face
face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
best_match_index = np.argmin(face_distances)
if matches[best_match_index]:
    name = known_face_names[best_match_index]

face_names.append(name)

process_this_frame = not process_this_frame

# Display the results
for (top, right, bottom, left), name in zip(face_locations, face_names):
    # Scale back up face locations since the frame we detected in was scaled to 1/4 size
    top *= 4
    right *= 4
    bottom *= 4
    left *= 4

    # Draw a box around the face
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

    # Draw a label with a name below the face
    cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

# Display the resulting image
cv2.imshow('Video', frame)

# Hit 'q' on the keyboard to quit!
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

# Release handle to the webcam
video_capture.release()
cv2.destroyAllWindows()

```

2-Live Stream for more control and security:

RPi Cam Web Interface is a web interface for the Raspberry Pi Camera module. It can be used for a wide variety of applications including surveillance, dvr recording and time lapse photography. It is highly configurable and can be extended with the use of macro scripts. It can be opened on any browser (smartphones included) and contains the following features:

- View, stop and restart a live-preview with low latency and high framerate. Full sensor area available.
- Control camera settings like brightness, contrast, ... live
- Record full-HD videos and save them on the sd-card packed into mp4 container while the live-preview continues
- Do timed or continuous video recording with splitting into fixed length segments

- Take single or multiple (timelapse) full-res pictures and save them on the sd-card (live-preview holds on for a short moment)
- Preview, download and delete the saved videos and pictures, zip-download for multiple files
- Trigger captures by motion detection using internal or external detection processes.
- Trigger captures by many scheduling-possibilities
- Circular buffer to capture actions leading up to motion detection
- Control Pan-Tilt or Pi-Light
- Shutdown/Reboot your Pi from the web interface
- Show annotations (e.g. timestamp) on live-preview and taken images/videos

- Supports selection from 2 cameras when used with a compute module

IMPORTANT NOTE: This is for the Raspberry Pi camera only. It does NOT support USB cameras.

Basic Installation:

Step 1: Install Raspbian on your RPi

Step 2: Attach camera to RPi

Step 3: Important : Enable camera

support (<http://www.raspberrypi.org/camera>)

Step 4: Update your RPi with the following commands:

```
sudo apt-get update  
sudo apt-get dist-upgrade
```

Occasionally if camera core software updates have been done then a sudo rpi-update may be used to benefit from these before they become available as standard.

Step 4:

For Jessie Lite run sudo apt-get install git

Clone the code from github and enable and run the install script with the following commands:

```
git clone  
https://github.com/silvanmelchior/RPi\_Cam\_Web\_Interface.git  
cd RPi_Cam_Web_Interface  
. ./install.sh
```

Older versions needed the scripts to be made executable with chmod u+x *.sh If you get permission denied while trying to run the install scripts then try that step

6 separate scripts are provided to do separate installation and maintenance functions.

The scripts are

- install.sh main installation as used in step 4 above
- update.sh check for updates and then run main installation
- start.sh starts the software. If already running it restarts.
- stop.sh stops the software
- remove.sh removes the software
- debug.sh is same as start but allows raspimjpeg output to console for debugging
- To run these scripts make sure you are in the RPi_Cam_Web_Interface folder then precede the script with a
- E.g. To update an existing installation ./update.sh
- E.g. To start the camera software ./start.sh
- E.g. To stop the camera software ./stop.sh

3-Alert system:

Why I should go to live stream and when?!

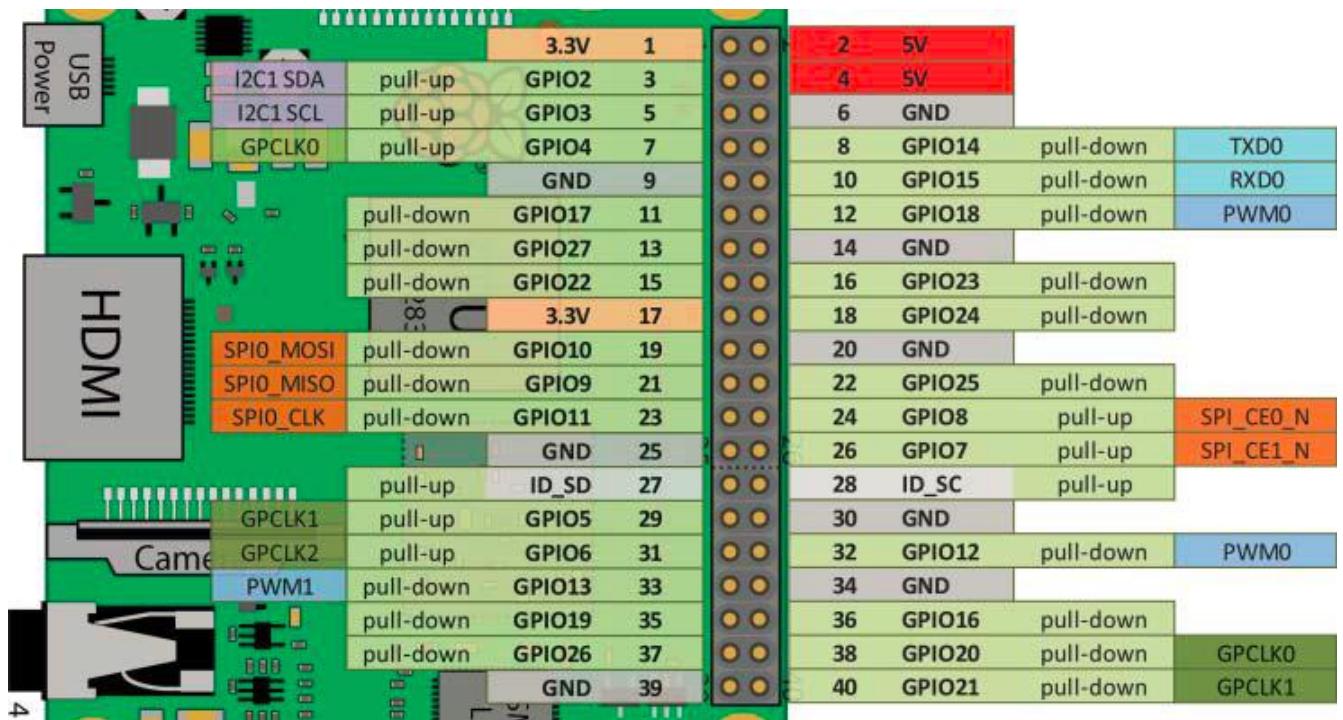
When alert system warns me that there is an unknown person then I can watch everything and record it also.

We use GPIO pins in raspberry pi to construct the system.

GPIO pins:

To know how use it and know the function of every pin.

There are two modes of GPIO (BCM or BOARD) this mode is control in the number of pins position. Such as in this picture:



File Edit Format Run Options Window Help

```
#import the GPIO and time package
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(11, GPIO.OUT)
# loop through 50 times, on/off for 1 second
for i in range(3):
    GPIO.output(11,1)
    time.sleep(1)
    GPIO.output(11,0)
    time.sleep(1)
GPIO.cleanup()
```

CHAPTER 5

{Android}



5.1 Preface

- Introduction:**

The era of mobile technology opens the windows to the android applications, the mobile phones are emerging.

The incredible increase in technology in the field of Mobile phones, allows us to develop Mobile Applications that has more capabilities.

As the Android operating system is getting more popular, the application based on Android SDK (Software Development Kit) attracts much more attention.

It is time to start considering using mobile apps more, which has become a part of our daily routine.

- Smart Home Application:**

Our Mobile Application is important in many ways to the Smart Home Project, since it adds many functionality and other controls to the hardware systems in the project.

Smart Home Application is developed based on Java and Android SDK (Software Development Kit) using Android Studio.

The Smart Home Application, basically has connection with three systems in the project, and they are (Light System, Door System, Security System)

- **Importance of Smart Home Application to the systems:**

1- Light System:

The Smart Home Application, adds another control method for turning on/off the lights in the house, Rather than using the way provided by the Hardware System, You can use the Mobile Application and press buttons to control the light.

This other way is important in many cases, and can be used when there are issues or problems in the primary way of controlling the Light.

2- Door System:

The Smart Home Application, adds another control method for opening the Doors in the house, Rather than using the way provided by the Hardware System, You can use the Mobile Application and press button to open the door.

This other way is important in many cases, and can be used when there are issues or problems in the primary way of opening the Door.

3- Security System:

The Smart Home Application, adds another functionality to the Security System, it is complementary to the Security System, Rather than only knowing when there is a breakthrough or someone is trying to access the Home illegally, When hearing the Alarm You can use the Mobile Application to see the video live stream of the Security System Camera.

• Operating Environment:

The Smart Home Application will only be available for the Android Operating Systems.

The Application shall only be used with compatible android devices.

The user shall use this application on Android OS 4.2 (API 17) (Jelly Bean) or any later versions of the Android OS.

• Preparation:

In order to be able to develop the required application, there are some important skills you need to acquire.

- 1- Solid skills in java programming language.

2-great background in and how to use Android Studio.

- **Courses:**

Courses we learned from are

1-Udemy Complete Java Masterclass.

2-Udemy - Android Java Masterclass - Become an App Developer.

3-Android Developer Documentation.

- **Terminology and Definitions:**

- 1) Java:**

Java is a popular programming language, it is owned by Oracle, and more than 3 billion devices run Java, It is used for:

- Mobile applications (specially Android apps)
- Desktop applications
- Web applications

We will focus on using Java in developing Mobile application (our Smart Home Application)

2) Android:

Android is a mobile operating system developed by Google.

And is designed primarily for touch screen mobile devices such as smart phones and tablets.

3) Android SDK:

The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform.

The Android SDK includes the following:

- Required libraries
- Debugger

4) Android Studio:

Android Studio provides the fastest tools for building apps on every type of Android device.

Android Studio features:

- Visual layout editor (Constraint Layout)
- Fast Emulator
- Intelligent code editor

5.2 Design phase

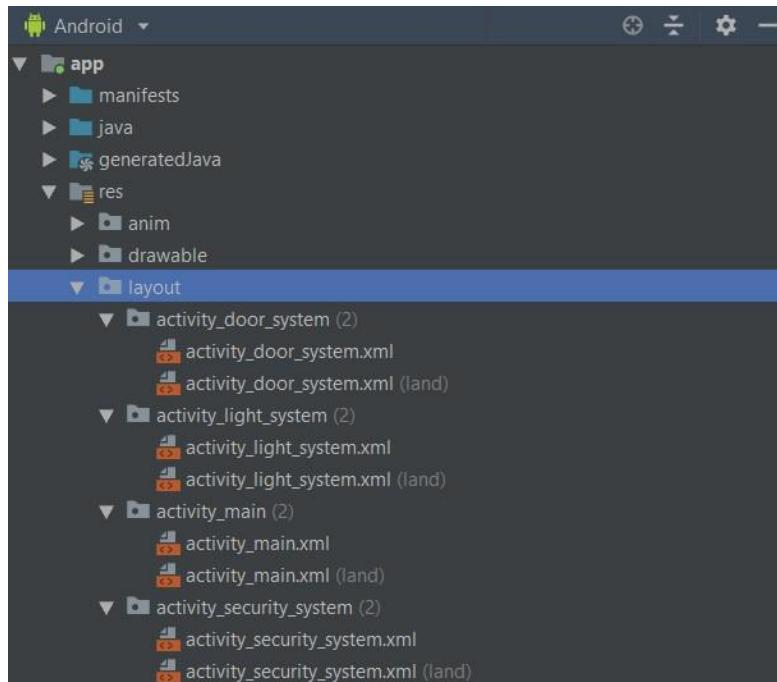
- **Introduction:**

In the Design phase, we will review the Smart Home Application Design, and go through the design phases and explain the steps in order to make them.

- **Smart Home Application:**

The Smart Home Application is divided into 4 Activities; each of them contains two layouts for different orientations (Normal layout – Landscape layout).

First thing first, we will go through each of them and review the Widgets or Views or Components, which we used in each of them.



- Fig0 Smart Home App Activities Layout

1) Main Activity:

- The Main Activity is the first Panel or Activity that loads when the Application is opened, we can think of it as a landing page and a Navigation page that works as a gate or a portal to more specific activities that are used in controlling the Hardware devices.
- So, in a nutshell The Main Activity, main function is only to make the user job very easy in order to choose what the Activity he wants to use.

Main Activity Layouts:



Fig1.1 Main Activity Normal Layout

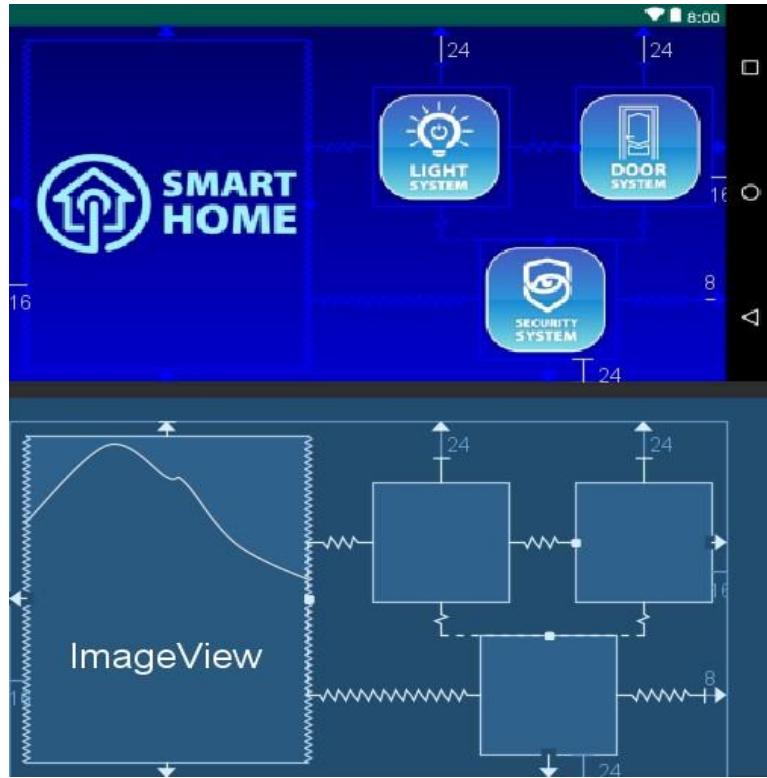


Fig1.2 Main Activity Landscape Layout

- In (Fig1.1), this is **The Main Activity Normal Layout** we can see how the Widgets or Views or Components are laid out and how they are connected together by using **Constraint Layout**, which is a feature in Android Studio that allows us to connect the Views in an easier way, rather than coding the statements using XML language.
- In (Fig1.2), this is **The Main Activity Landscape Layout** we can see how the Widgets or Views or Components are laid out and how they are connected together by using **Constraint Layout**, which is a feature in Android Studio that allows us to connect the Views in an easier way, rather than coding the statements using XML language.

Main Activity Widgets or Views or Components:

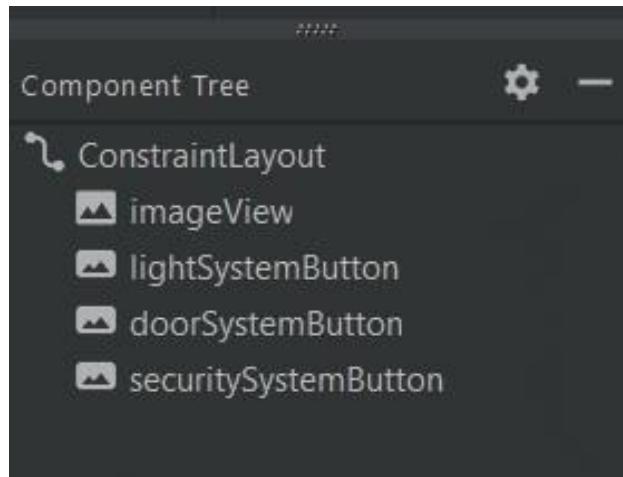


Fig1.3 Main Activity Components

As we can see in (Fig1.3) in the Main Activity component tree, we can see that we used (4) components and they are:

- 1-Image View of the Smart Home logo. (Fig1.4)
- 2-Button of light System. (Fig1.5)
- 3-Button of door system. (Fig1.6)
- 4-Button of security system. (Fig1.7)



Fig1.4 the Smart Home logo



Fig1.5 light System button



Fig1.6 door System button



Fig1.7 security System button

2) Light System Activity:

- The Light System Activity is the Panel or Activity that is responsible for controlling the Hardware Light System, we can think of it as a page that enables us to turn on/off the light by only pressing a button rather than using the primary way that the Hardware Light System provides, and also a Navigation page that works as a gate or a portal to more specific activities that are used in controlling the other Hardware devices.
- So, in a nutshell The Light System Activity, main function is to make the user job very easy in order to control the light system in the home, and has a secondary function that it acts as a Navigation page to make the user job very easy in order to choose what the Activity he wants to use.

Light System Activity Layouts:



Fig2.1 Light System Activity Normal Layout

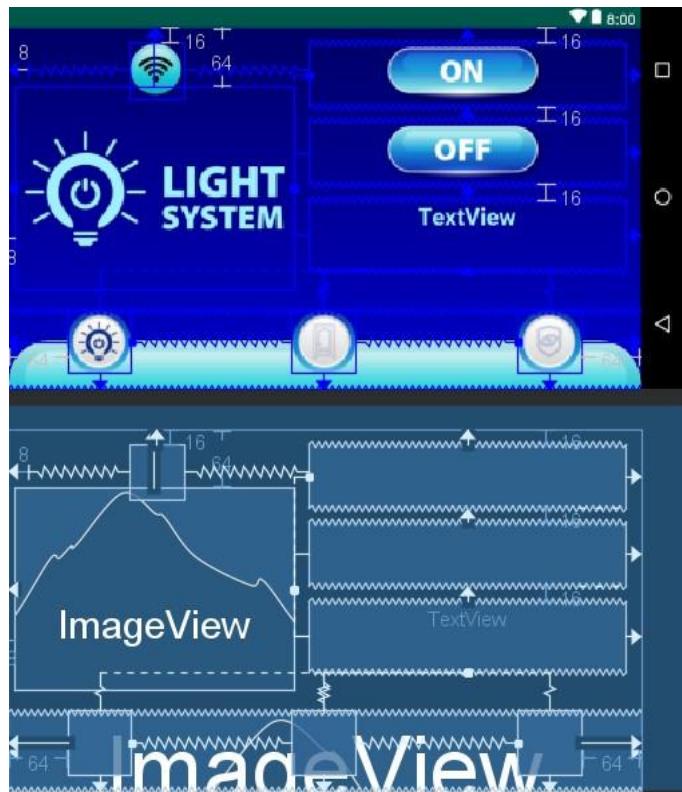


Fig2.2 Light System Activity Layout Layout

- In (Fig2.1), this is **The Light System Activity Normal Layout** we can see how the Widgets or Views or Components are laid out and how they are connected together by using **Constraint Layout**, which is a feature in Android Studio that allows us to connect the Views in an easier way, rather than coding the statements using XML language.
- In (Fig2.2), this is **The Light System Activity Landscape Layout** we can see how the Widgets or Views or Components are laid out and how they are connected together by using **Constraint Layout**, which is a feature in Android Studio that allows us to connect the Views in an easier way, rather than coding the statements using XML language.

Light System Activity Widgets or Views or Components:

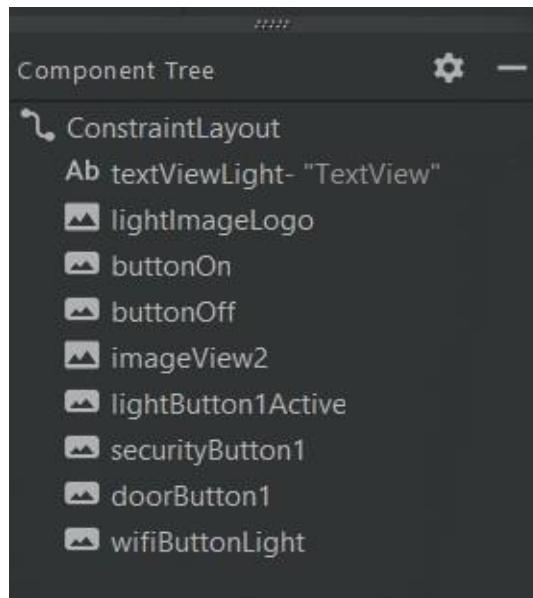


Fig2.3 Light System Activity Components

As we can see in (Fig2.3) in the Light System Activity component tree, we can see that we used 9 components and they are:

- 1-Image View of the Light System logo. (Fig2.4)
- 2-ON Button of light System. (Fig2.5)
- 3-OFF Button of light system. (Fig2.6)
- 4-Wi-Fi connection Button. (Fig2.7)
- 5-Text View to show a text message.
- 6-Image View of the underline bar of which the buttons placed on. (Fig2.8)
- 7-Active Button of Light System. (Fig2.9)
- 8-Inactive Button of door system. (Fig2.10)
- 9-Inactive Button of security system.(Fig2.11)



Fig2.4 the Light System logo



Fig2.5 ON Button of Light System



Fig2.6 OFF Button of Light System

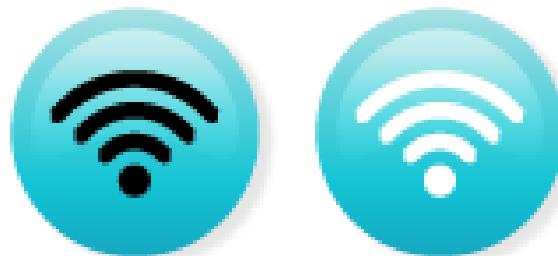


Fig2.7 Wi-Fi connection Button



Fig2.8 the underline bar



Fig2.9 Active Button of Light System



Fig2.10 Inactive Button of door System



Fig2.11 Inactive Button of security System

3) Door System Activity:

- The Door System Activity is the Panel or Activity that is responsible for controlling the Hardware Door System, we can think of it as a page that enables us to open the door by only pressing a button rather than using the primary way that the Hardware Light System provides, and also a Navigation page that works as a gate or a portal to more specific activities that are used in controlling the other Hardware devices.
- So, in a nutshell The Door System Activity, main function is to make the user job very easy in order to control the door system in the home, and has a secondary function that it acts as a Navigation page to make the user job very easy in order to choose what the Activity he wants to use.

Door System Activity Layouts:



Fig3.1 Door System Activity Normal Layout



Fig3.2 Door System Activity Landscape Layout

- In (Fig3.1), this is **The Door System Activity Normal Layout** we can see how the Widgets or Views or Components are laid out and how they are connected together by using **Constraint Layout**, which is a feature in Android Studio that allows us to connect the Views in an easier way, rather than coding the statements using XML language.
- In (Fig3.2), this is The Door System Activity Landscape Layout we can see how the Widgets or Views or Components are laid out and how they are connected together by using **Constraint Layout**, which is a feature in Android Studio that allows us to connect the Views in an easier way, rather than coding the statements using XML language.

Door System Activity Widgets or Views or Components:

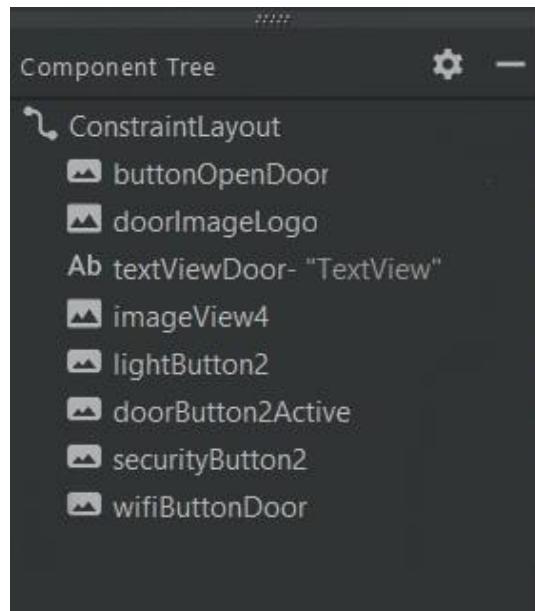


Fig3.3 Door System Activity Components

As we can see in (Fig3.3) in the Door System Activity component tree, we can see that we used 8 components and they are:

- 1-Image View of the Door System logo. (Fig3.4)
- 2-OPEN Button of Door System. (Fig3.5)
- 3-Wi-Fi connection Button. (Fig3.6)
- 4-Text View to show a text message.
- 5-Image View of the underline bar of which the buttons placed on. (Fig3.7)
- 6-Active Button of door System. (Fig3.8)
- 7-Inactive Button of light system. (Fig3.9)
- 8-Inactive Button of security system.(Fig3.10)

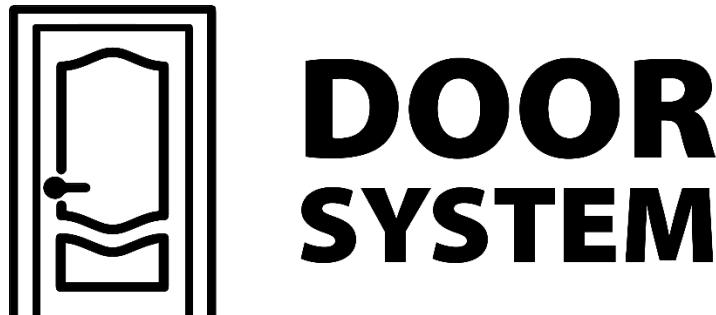


Fig3.4 the Door System logo



Fig3.5 OPEN Button of Door System

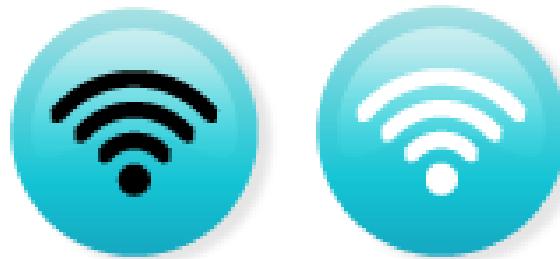


Fig3.6 WiFi connection Button



Fig3.7 the underline bar



Fig3.8 Active Button of door System



Fig3.9 Inactive Button of light System



Fig3.10 Inactive Button of security System

4) Security System Activity:

- The Security System Activity is the Panel or Activity that is responsible for controlling the Hardware Security System, we can think of it as a page that enables us to see the video live stream provided by the camera of the Security System, and also a Navigation page that works as a gate or a portal to more specific activities that are used in controlling the other Hardware devices.
- So, in a nutshell The Security System Activity, main function is to make the user job very easy in order to see the video live stream provided by the camera of the Security System in the home, and has a secondary function that it acts as a Navigation page to make the user job very easy in order to choose what the Activity he wants to use.

Security System Activity Layouts:



Fig4.1 Security System Activity Normal Layout



Fig4.2 Security System Activity Landscape Layout

- In (Fig4.1), this is **The Security System Activity Normal Layout** we can see how the Widgets or Views or Components are laid out and how they are connected together by using **Constraint Layout**, which is a feature in Android Studio that allows us to connect the Views in an easier way, rather than coding the statements using XML language.
- In (Fig4.2), this is **The Security System Activity Landscape Layout** we can see how the Widgets or Views or Components are laid out and how they are connected together by using **Constraint Layout**, which is a feature in Android Studio that allows us to connect the Views in an easier way, rather than coding the statements using XML language.

Security System Activity Widgets or Views or Components:

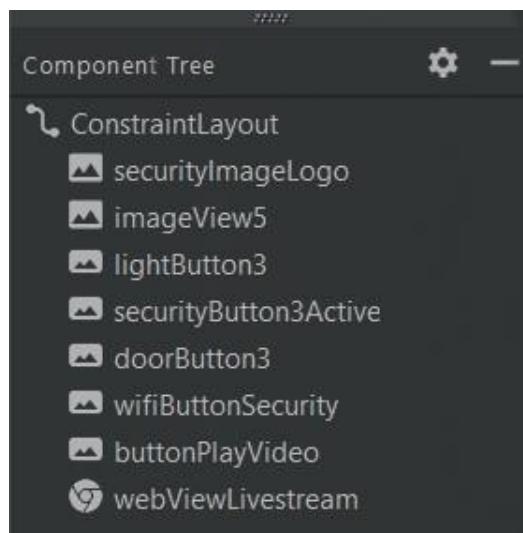


Fig4.3 Security System Activity Components

As we can see in (Fig4.3) in the Door System Activity component tree, we can see that we used 5 components and they are:

- 1-Image View of the Security System logo. (Fig4.4)
- 2-Image View of the underline bar of which the buttons placed on. (Fig4.5)
- 3-Active Button of security System. (Fig4.6)
- 4-Inactive Button of light system. (Fig4.7)
- 5-Inactive Button of door system. (Fig4.8)
- 6-Wi-Fi connection Button. (Fig4.9)
- 7-PLAY Button of Security System. (Fig4.10)
- 8-WebView. (Fig4.11)



Fig4.4 the Security System logo



Fig4.5 the underline bar



Fig4.6 Active Button of security System



Fig4.7 Inactive Button of light System



Fig4.8 Inactive Button of door System

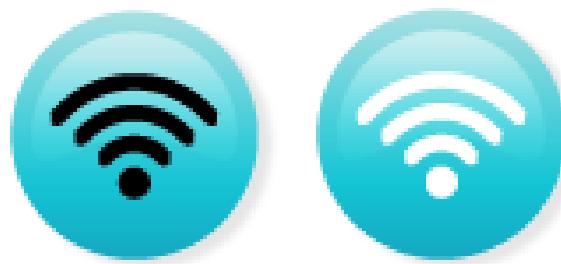


Fig4.9 Wi-Fi connection Button



Fig4.10 PLAY Button of Security System

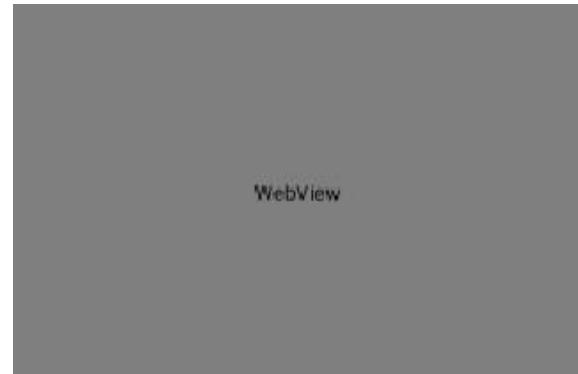


Fig4.11 WebView

5) Smart Home Application Gradient Background:

- In the Smart Home Application we set the background color as a gradient rather than a single color background, in other words we specified two colors and made a gradient background with them, and made it the background for the whole app, hence the background of the 4 Activities.
- We simply done this by creating a new drawable xml file. (Fig5.1)

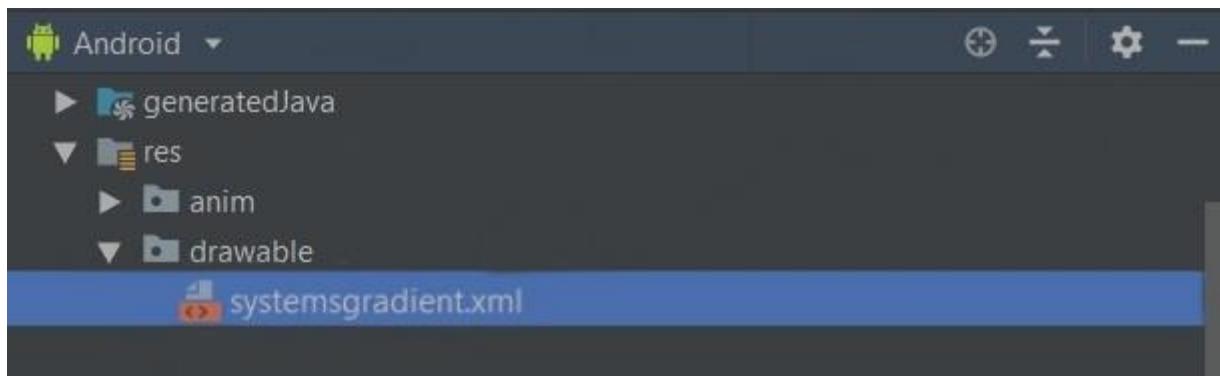
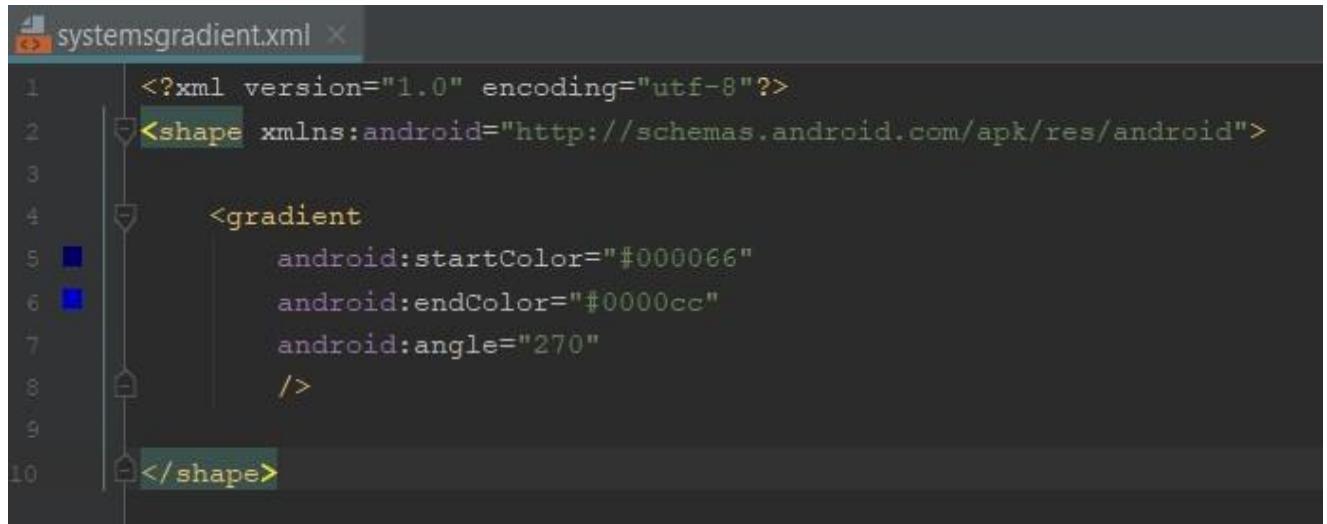


Fig5.1 gradient background xml drawable file

- We simply identify a shape object and then identify the start color and the end color hex values and the angle of the gradient, which in our case is 270°. (Fig5.2)



```
systemsgradient.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <shape xmlns:android="http://schemas.android.com/apk/res/android">
3
4     <gradient
5         android:startColor="#000066"
6         android:endColor="#0000cc"
7         android:angle="270"
8     />
9
10    </shape>
```

Fig5.2 gradient background xml file

6) Create app icon with Image Asset Studio:

- In order to create an icon for the Smart Home Application we used Image Asset Studio
- **To start Image Asset Studio, follow these steps:**

1- In the Project window, select the Android view. (Fig6.1)

2- Right-click the res folder and select New > Image Asset. (Fig6.1)

3- Since our app supports Android 8.0, then we create adaptive and legacy launcher icons. (Fig6.3)

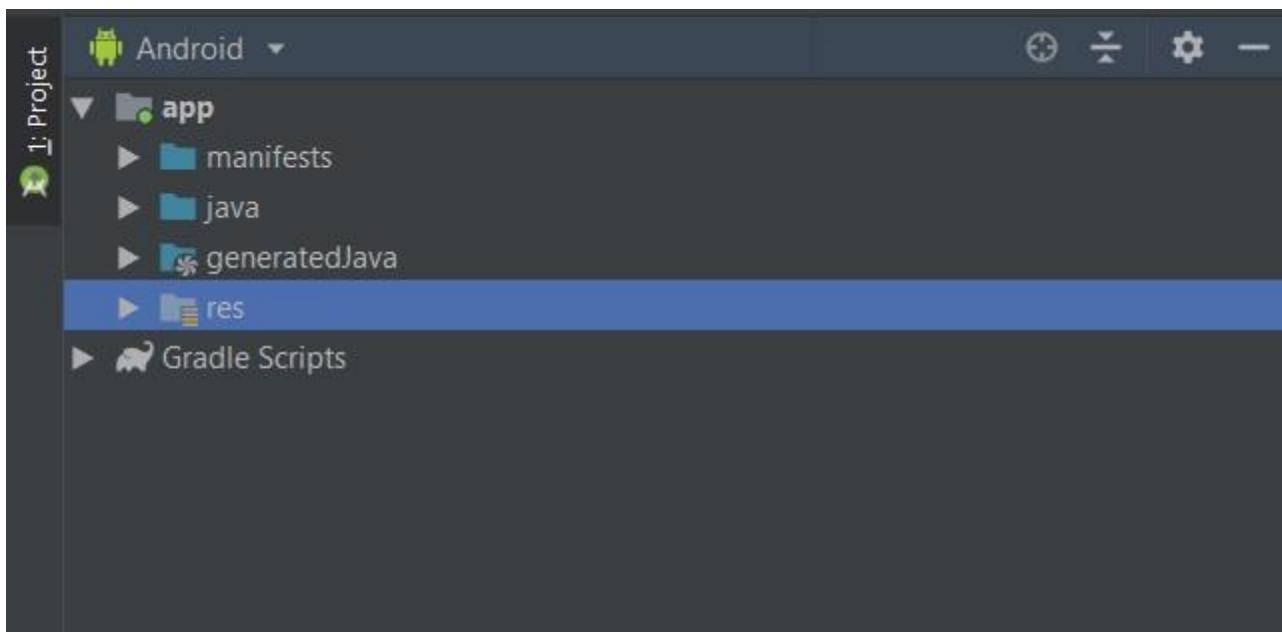


Fig6.1 res folder in Android View



Fig6.2 smart home app icon

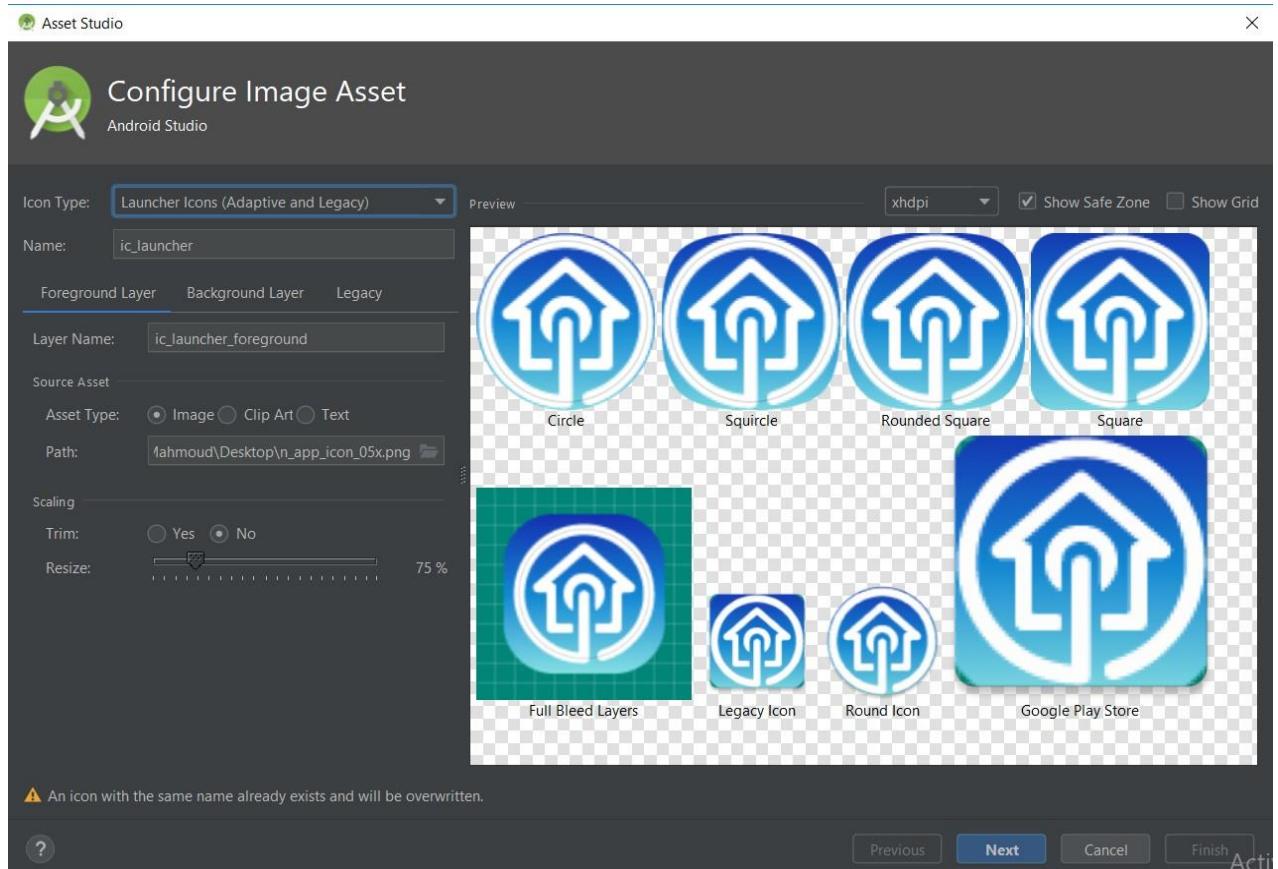


Fig6.3 Image Asset panel to create adaptive and legacy launcher icons

5.3 Functionality phase

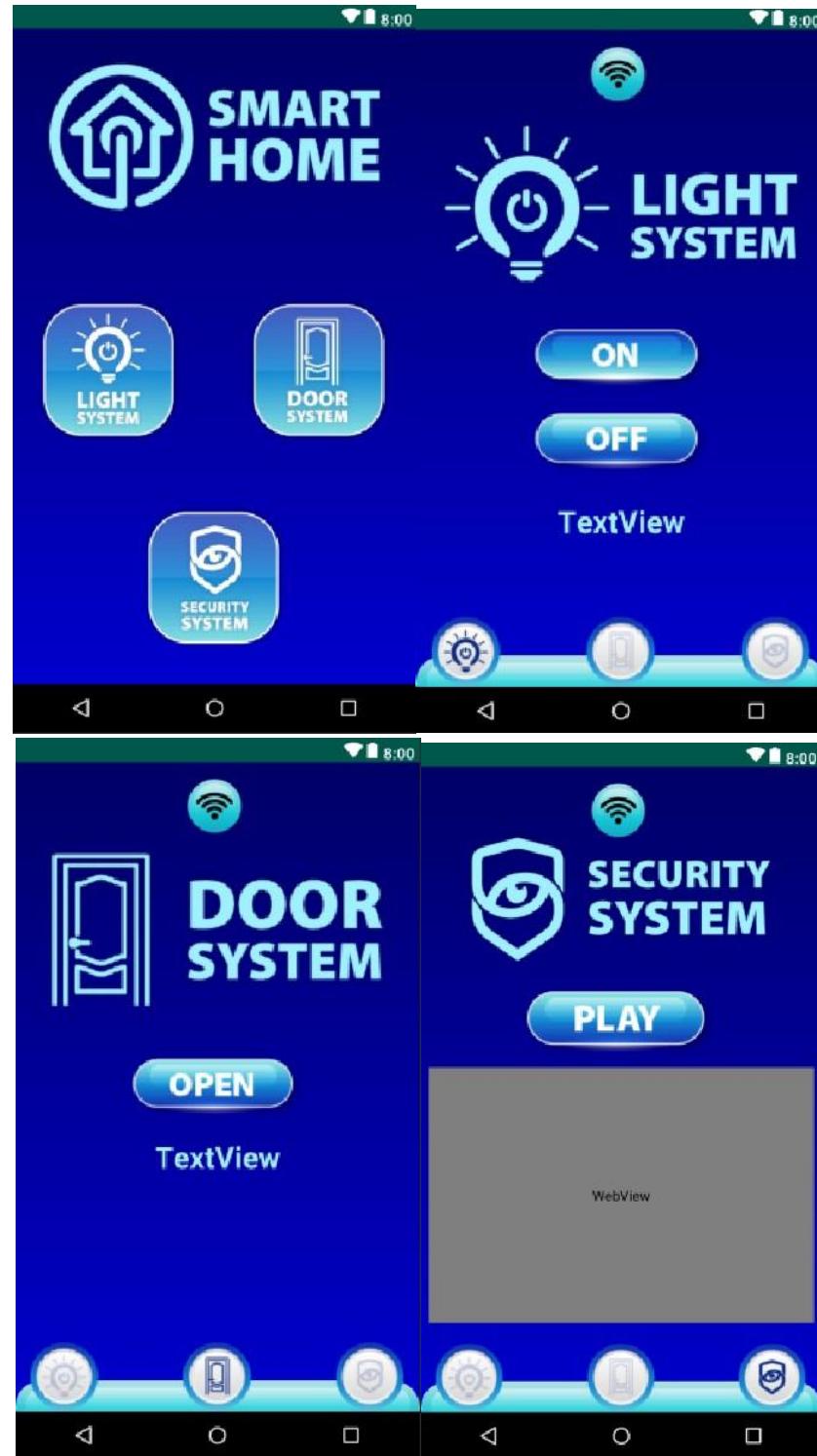
- **Introduction:**

In the Functionality phase, we will review the Smart Home Application Functionality, and go through the Functionality phases and explain the steps in order to make them.

We will break the different functionality of the app into several phases and we will review each of them, showing what the importance of each of them is and how each of them is done.

- **Smart Home Application:**

The Smart Home Application is divided into 4 Activities; each of them contains two layouts for different orientations (Normal layout – Landscape layout).



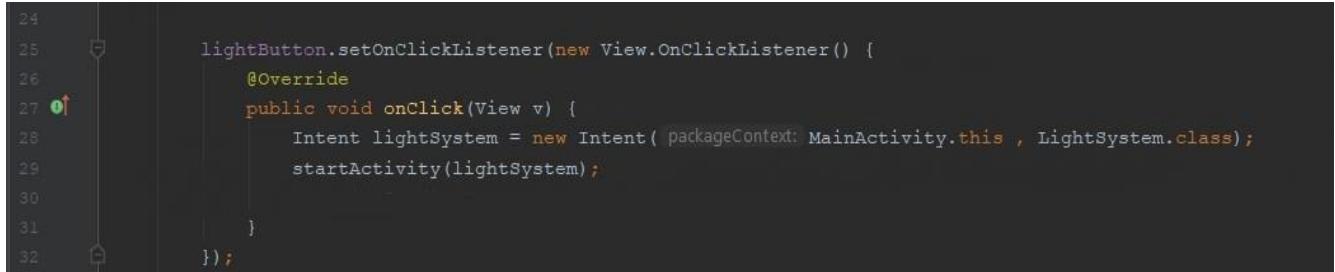
• Fig0 Smart Home App 4 Activities Normal Layout

1) Linking Activities:

- First thing after designing the layout of the whole app, in other words designing the application Activities Layouts, is Figuring out how to link those Activities together and putting a logical Navigation System between those Activities that syncs perfectly with the Activities Layouts.
- So, in a nutshell the first functionality we will be discussing is the method we applied to link or connect the Activities with each other using the buttons from the Layouts of the Activities.
- So, what we are trying to achieve is linking different Activities of our application by opening them using button clicks.
- We will be using **setOnTouchListener** method of **Buttons** and we will create a new **Intent** and pass it to the **startActivity** method.
- **Button:** A user interface element the user can tap or click to perform an action.
- **SetonClickListener:** Interface definition for a callback to be invoked when a view (button) is clicked.
- **Intent:** Intent is an abstract description of an operation to be performed, it can be used with **startActivity(Intent)** to launch an Activity, we can say that it is the description of the activity to start.

- **startActivity**: method used to Launch a new activity.
- **Code Reference:**

- In the following we will see how the previous description about linking the Activities together is implemented in **Java code** in **Android Studio**.
- The following code reference will be linking the 3 buttons in the Main Activity (**Fig0**) each of them to open the required Activity (Light, Door, and Security).
- In here, we can see that we used what we described above in order to be able to open **Light System Activity** when clicking the **light button** in the **Main Activity**. (**Fig1.1**)



```

24
25
26
27     lightButton.setOnClickListener(new View.OnClickListener() {
28         @Override
29         public void onClick(View v) {
30             Intent lightSystem = new Intent( mContext: MainActivity.this , LightSystem.class);
31             startActivity(lightSystem);
32         }
33     });

```

- **Fig1.1 lightButton onClick method in Main Activity.**
- In here, we can see that we used what we described above in order to be able to open **Door System Activity** when clicking the **door button** in the **Main Activity**. (**Fig1.2**)

```
34     doorButton.setOnClickListener(new View.OnClickListener() {  
35         @Override  
36         public void onClick(View v) {  
37             Intent doorSystem = new Intent( mContext, DoorSystem.class );  
38             startActivity(doorSystem);  
39         }  
40     });  
41 }
```

- Fig1.2 doorButton onClick method in Main Activity.

- In here we can see that we used what we described above in order to be able to open **Security System Activity** when clicking the **security button** in the **Main Activity**. (Fig1.3)

```
43     securityButton.setOnClickListener(new View.OnClickListener() {  
44         @Override  
45         public void onClick(View v) {  
46             Intent securitySystem = new Intent( mContext, SecuritySystem.class );  
47             startActivity(securitySystem);  
48         }  
49     });  
50 }
```

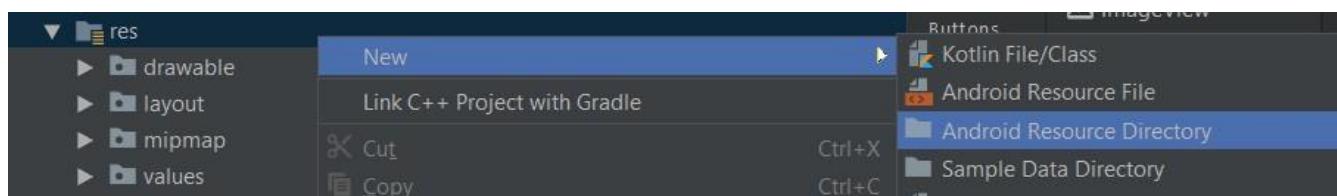
- Fig1.3 securityButton onClick method in Main Activity.

- **Note: Linking the rest Activities in the application will be implemented using the same above method we described.**

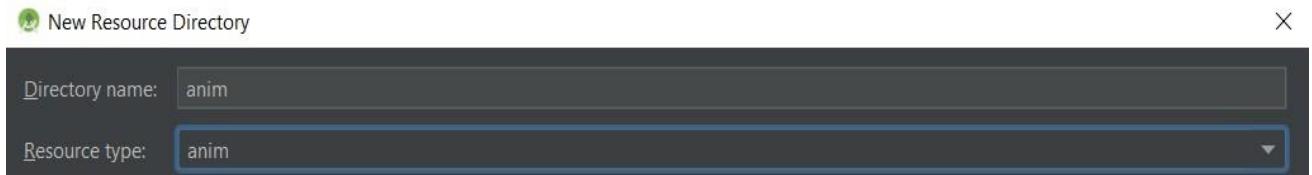
2) Slide Animation Between Activities:

- Second thing we will discuss is changing the default transition that occurs when moving from one Activity to another one.
- So, in a nutshell the Second functionality we will be discussing, is how we can add a slide transition between activities.
- So, what we are trying to achieve is changing the default transition animation into more good and practical one which is slide animation.
- We will be using several **anim files** and will be using **overridePendingTransition** method in two different places, one would be after starting an **intent** in any button that uses intent to start new activity, and the second is also after calling the **finish** method.
- **Intent:** Intent is an abstract description of an operation to be performed, it can be used with **startActivity(Intent)** to launch an Activity, we can say that it is the description of the activity to start.
- **Anim files:** anim refers to animation, and anim files is used to describe the animation properties that is written in xml language, and xml anim files are very important because they define the animation files that we will refer to and use in code.

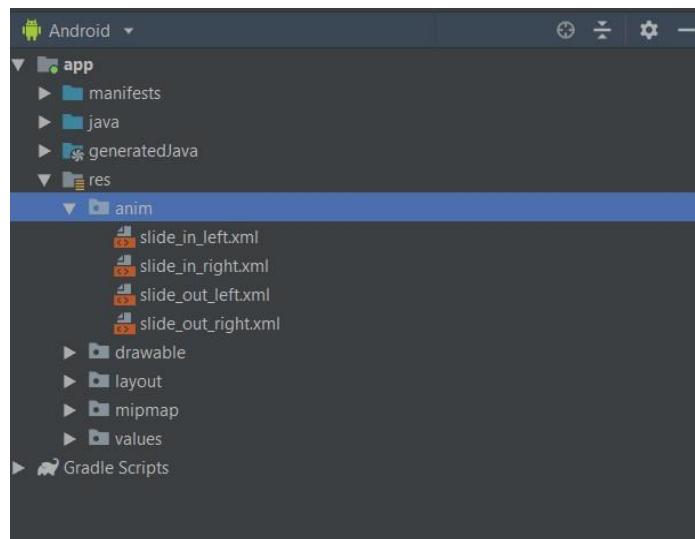
- **OverridePendingTransition**: Called immediately after one of the **startActivity(Intent)** or **finish()** to specify an explicit transition animation to perform next, and must have two parameters, **enterAnim** and **exitAnim**.
- **EnterAnim**: determines how views in an activity enter the scene.
- **ExitAnim**: determines how views in an activity exit the scene.
- **Finish**: is a method that is used when your activity is done and should be closed, in other words we will be using it to implement the slide animation to the back button that is in the mobile phones.
- **Code Reference:**
- In the following we will see how the previous description about changing the default transition that occurs when moving from one Activity to another one, and how we can add a slide transition between activities is implemented in **xml & Java** code in **Android Studio**.
- **Step1**: First step would be creating an Android Resource Directory (**anim**) in which we can put anim xml files in it. **(Fig2.1) (Fig2.2)**



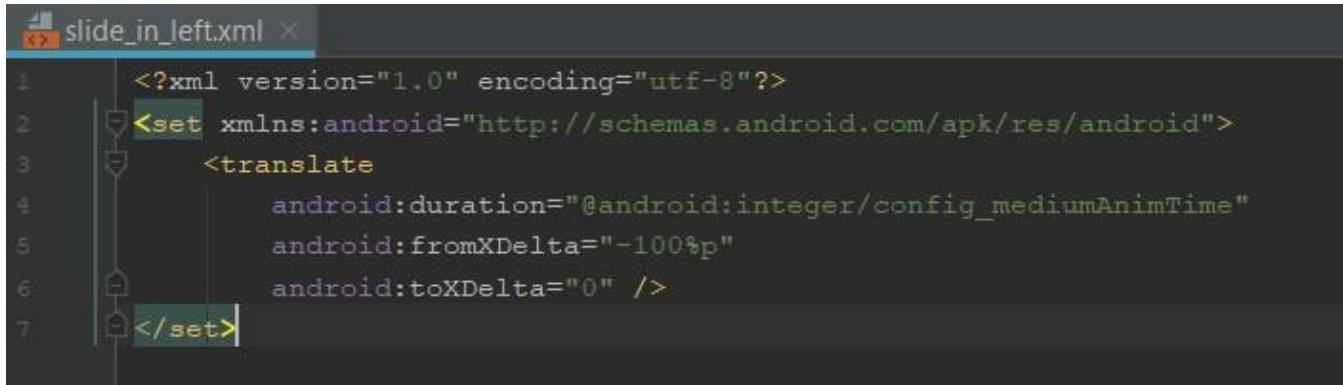
- **Fig2.1 creating a new Android Resource Directory.**



- **Fig2.2** naming the new Directory (anim) and choosing (anim) as its type
- **Step2:** Second step is creating (4) anim files that will be used to describe the slide animation transition elements. (**Fig2.3**)

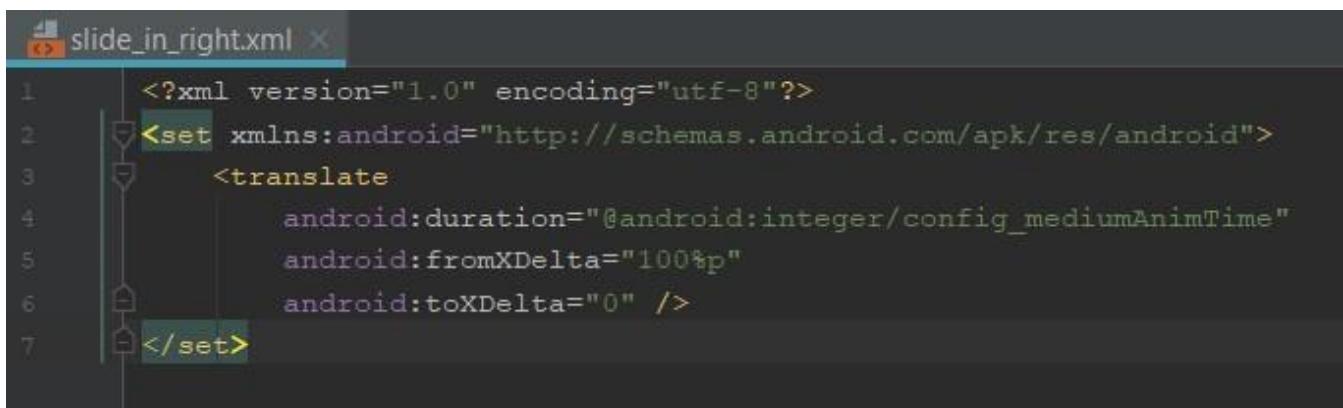


- **Fig2.3** anim files in anim directory.
- **Step3:** Third step is coding the anim files to specify the slide transition animation effect we would like to have using xml language.
- We will create (2) anim files that act as **enterAnim**, and another two that act as **exitAnim**.



```
slide_in_left.xml X
1  <?xml version="1.0" encoding="utf-8"?>
2  <set xmlns:android="http://schemas.android.com/apk/res/android">
3      <translate
4          android:duration="@android:integer/config_mediumAnimTime"
5          android:fromXDelta="-100%p"
6          android:toXDelta="0" />
7      </set>
```

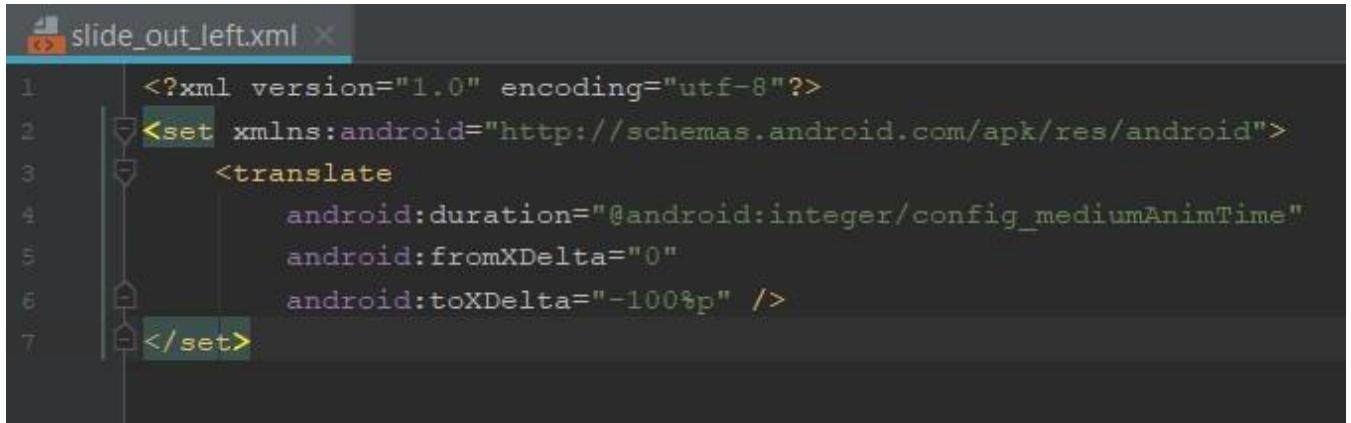
- Fig2.4 slide_in_left enterAnim file in anim directory.



```
slide_in_right.xml X
1  <?xml version="1.0" encoding="utf-8"?>
2  <set xmlns:android="http://schemas.android.com/apk/res/android">
3      <translate
4          android:duration="@android:integer/config_mediumAnimTime"
5          android:fromXDelta="100%p"
6          android:toXDelta="0" />
7      </set>
```

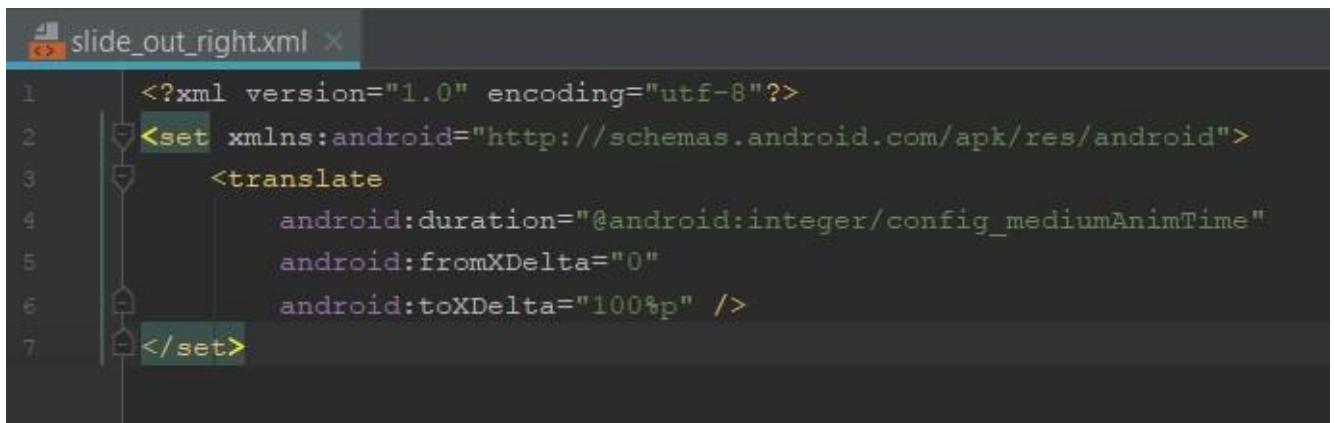
- Fig2.5 slide_in_right enterAnim file in anim directory.

- In here we can see that we created two enterAnim files (slide_in_left, slide_in_right), and in them we stated and defined the properties that is required in order to achieve the slide animation transition between Activities.
- Properties defined in enterAnim files are: the duration that the enterAnim file take to execute, and the direction of which the views will use to enter the scene. **(Fig2.4) (Fig2.5)**



```
slide_out_left.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <set xmlns:android="http://schemas.android.com/apk/res/android">
3      <translate
4          android:duration="@android:integer/config_mediumAnimTime"
5          android:fromXDelta="0"
6          android:toXDelta="-100%p" />
7      </set>
```

- Fig2.6 slide_out_left exitAnim file in anim directory.

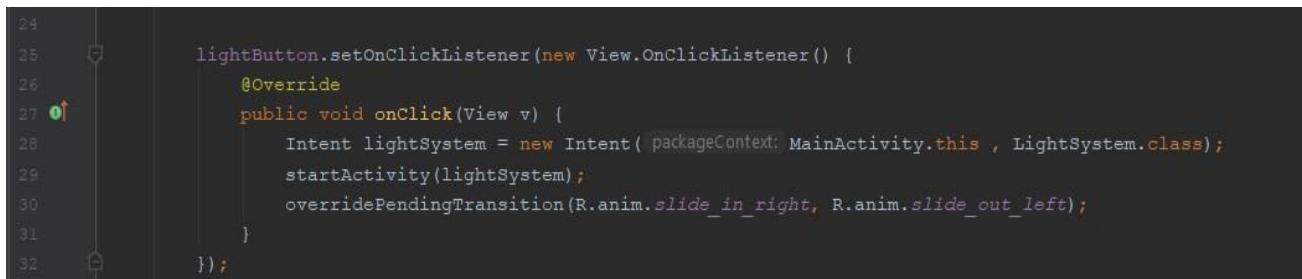


```
slide_out_right.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <set xmlns:android="http://schemas.android.com/apk/res/android">
3      <translate
4          android:duration="@android:integer/config_mediumAnimTime"
5          android:fromXDelta="0"
6          android:toXDelta="100%p" />
7      </set>
```

- Fig2.7 slide_out_right exitAnim file in anim directory.

- In here we can see that we created two exitAnim files (slide_out_left, slide_out_right), and in them we stated and defined the properties that is required in order to achieve the slide animation transition between Activities.
- Properties defined in exitAnim files are: the duration that the exitAnim file take to execute, and also the direction of which the views will use to exit the scene. **(Fig2.6) (Fig2.7)**

- **Step4:** Fourth step is applying the anim files we created in previous steps to specific activities.
- We will implement that by using the **overridePendingTransition** method after starting an intent and after calling the **finish** method.
- The following code reference will describe the implementation of the slide animation transition by using **overridePendingTransition** after the intent inside the **SetonClickListener** method of the (3) buttons in the Main Activity (**Fig0**) each of them is used to open the required Activity.
- In here, we can see that we used what we described above in order to be able to implement the slide transition animation when moving between Activities, from the **Main Activity** to the **Light Activity**, when clicking the light button. (**Fig2.8**)



```

24
25
26     lightButton.setOnClickListener(new View.OnClickListener() {
27         @Override
28         public void onClick(View v) {
29             Intent lightSystem = new Intent(getApplicationContext(), LightSystem.class);
30             startActivity(lightSystem);
31             overridePendingTransition(R.anim.slide_in_right, R.anim.slide_out_left);
32         }
33     });

```

- **Fig2.8 using overridePendingTransition in Main Activity.**
- In here, we can see that we used what we described above in order to be able to implement the slide transition animation when moving between Activities,

from the **Main Activity** to the **Door Activity**, when clicking the door button. (**Fig2.9**)

```
34     doorButton.setOnClickListener(new View.OnClickListener() {  
35         @Override  
36         public void onClick(View v) {  
37             Intent doorSystem = new Intent(getApplicationContext(), DoorSystem.class);  
38             startActivity(doorSystem);  
39             overridePendingTransition(R.anim.slide_in_right, R.anim.slide_out_left);  
40         }  
41     });
```

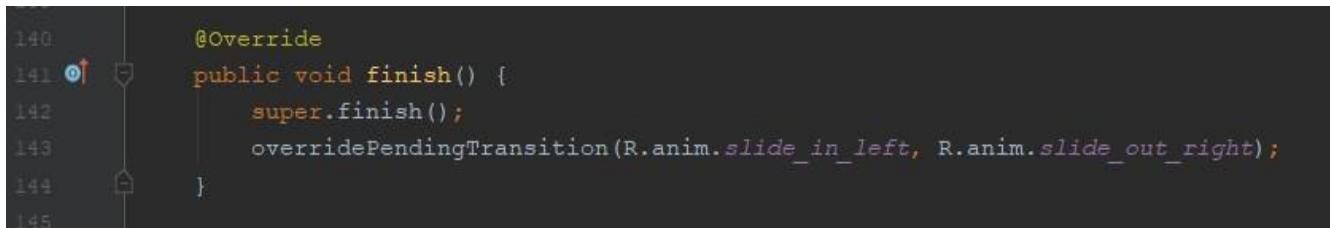
- Fig2.9 using `overridePendingTransition` in Main Activity.
- In here, we can see that we used what we described above in order to be able to implement the slide transition animation when moving between Activities, from the **Main Activity** to the **Security Activity**, when clicking the security button. (**Fig2.10**)

```
43     securityButton.setOnClickListener(new View.OnClickListener() {  
44         @Override  
45         public void onClick(View v) {  
46             Intent securitySystem = new Intent(getApplicationContext(), SecuritySystem.class);  
47             startActivity(securitySystem);  
48             overridePendingTransition(R.anim.slide_in_right, R.anim.slide_out_left);  
49         }  
50     });
```

- Fig2.10 using `overridePendingTransition` in Main Activity.
- **Note: Implementing slide transition animation in the rest Activities in the application will be implemented using the same above method we described.**
- The following code reference will describe the implementation of the slide animation transition by using `overridePendingTransition` after calling the `finish`

method, In the (Light, Door, and Security) Activities (**Fig0**) in order to implement the slide transition animation to the back button that is in the mobile phones.

- That is done in order to convey or mimic the slide animation when clicking the back button.
- In here we can see that we used what we described above in order to be able to implement the slide transition animation when clicking the back button. (**Fig2.11**)



```
140
141     @Override
142     public void finish() {
143         super.finish();
144         overridePendingTransition(R.anim.slide_in_left, R.anim.slide_out_right);
145     }

```

A screenshot of the Android Studio code editor. The code shown is Java code for overriding the finish() method. It includes imports at the top, followed by the @Override annotation and the method definition. Inside the method, it calls super.finish() and then uses overridePendingTransition to set up a slide transition from left to right. Line numbers 140 through 145 are visible on the left side of the code area.

- **Fig2.11** using `overridePendingTransition` with `finish` method in (Light, Door, and Security) Activities.

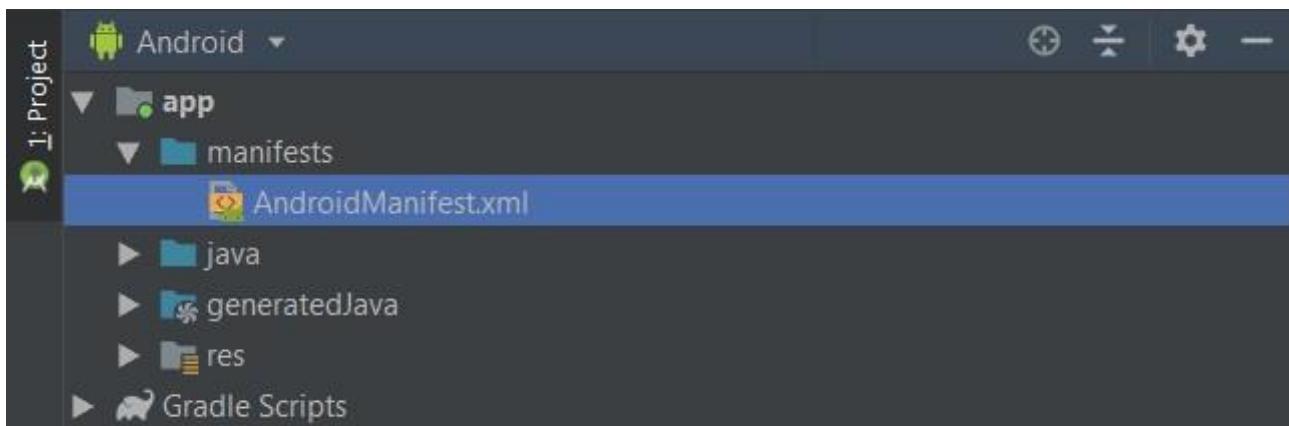
3) Connecting to the Light and Door Systems:

- Third thing we will discuss is connecting to the Light and Door Hardware Systems through Wi-Fi Connection.
- So, in a nutshell the Third functionality we will be discussing, is how we can connect our mobile phone to the Wi-Fi Module of the Light and Door Hardware Systems from our Smart Home Application.
- So, what we are trying to achieve is connecting our mobile phone device with the Light and Door Hardware Systems, because that is critical in order to give them commands to control them later.
- We will be using **WifiManager** and will be using **App Manifest** and also will be using **Permissions**.
- **WifiManager**: This class provides the primary API for managing all aspects of Wi-Fi connectivity, It deals with several categories of items:
 1. The list of configured networks.
 2. The currently active Wi-Fi network, if any.
Connectivity can be established or torn down.
 3. Results of access point scans, containing enough information to make decisions about what access point to connect to.

- **App Manifest:** Every app project must have an **AndroidManifest** xml file (with precisely that name) at the root of the project source set, The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play, the manifest file is required to declare the following:
 1. The app's package name.
 2. The components of the app, which include all activities, services.
 3. The permissions that the app needs.
 4. The hardware and software features the app requires.
- **Permissions:** The permissions that the app needs in order to access protected parts of the system or other apps. It also declares any permissions that other apps must have if they want to access content from this app, Android apps must request permission to access certain system features (such as the camera and internet access), Each permission is identified by a unique label.

- **Code Reference:-**

- In the following we will see how the previous description about connecting to the Light and Door Hardware Systems through Wi-Fi Connection, and how we can achieve that by using **WifiManager** and **Permissions**, in **xml & Java** code in **Android Studio**.
- **Step1:** First step would be adding the required permissions in the App Manifest file, which will help us to access certain system features (such as internet access and manipulating Wi-Fi state). (**Fig3.1**) (**Fig3.2**)



- **Fig3.1 App Manifest File.**

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.smarthome">
4          <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
5          <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
6          <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
7          <uses-permission android:name="android.permission.INTERNET"/>
```

- **Fig3.2 Required permissions in App Manifest file.**

- **Step2:** Second step would be creating an instance of the **WifiManager** class in order to use it when creating the connection method. (**Fig3.3**) (**Fig3.4**)

34

```
private WifiManager wifiManager;
```

- Fig3.3 WifiManager instance.

59
60
61
62

```
wifiManager = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SERVICE);  
wifiManager.setWifiEnabled(true);
```

- Fig3.4 WifiManager instance.

- **Step3:** Third step would be creating our **connectToWiFi** method that we will be passing two parameters (network ssid, network password) into it from our WiFi Button in order to connect to the Wi-Fi Module in the Light and Door Hardware Systems. (**Fig3.5**)

166
167
168
169
170
171
172
173
174
175
176
177

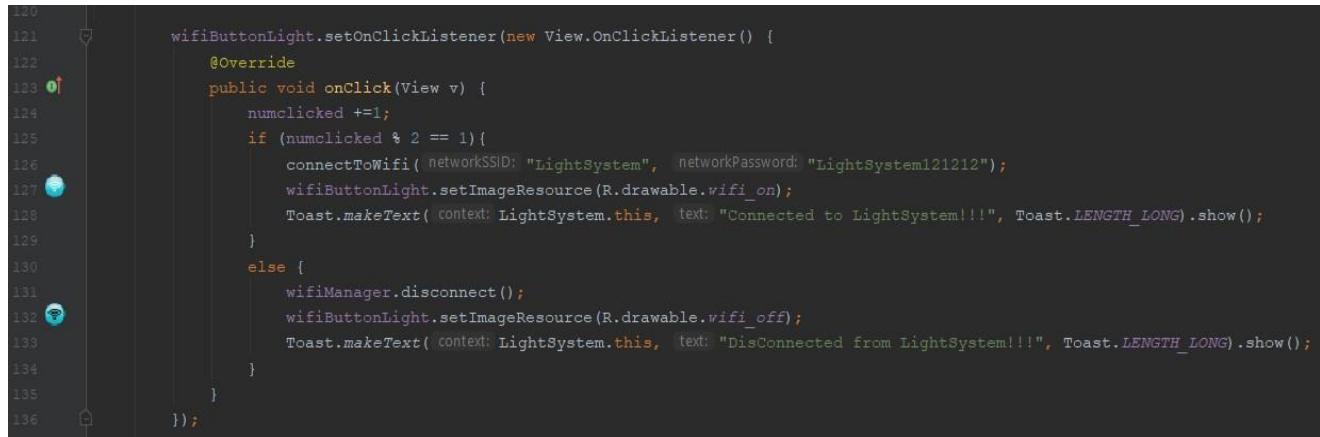
```
public void connectToWifi(String networkSSID, String networkPassword) {  
  
    WifiConfiguration conf = new WifiConfiguration();  
    conf.SSID = String.format("%s\"", networkSSID);  
    conf.preSharedKey = String.format("%s\"", networkPassword);  
  
    int netId = wifiManager.addNetwork(conf);  
    wifiManager.disconnect();  
    wifiManager.enableNetwork(netId, attemptConnect: true);  
    wifiManager.reconnect();  
}
```

- Fig3.5 connectToWiFi method.

- **Note:** Line 169 creates a WifiConfiguration object which represents a configured Wifi network, including the security configurations for the Wifi network.
- Line 170 and 171 are public calls to the SSID and the preSharedKey variables to set the Wifi SSID and password to the configurations. Both the SSID and the password need to be enclosed around quotes, hence the use of the String.format() method to do so.
- Line 173 adds the network configurations to the WifiManager, which then returns the network id which is used to specify the network to be acted upon.
- Line 174 disassociates from the currently active access point.
- Line 175 does the actual connection to the Wifi network by calling enableNetwork() and passing in the network id we got previously, and passing in true to let the WifiManager know to disable every other network.
- Line 176 is a safe check in case the initial connection failed, What the reconnect() method does is

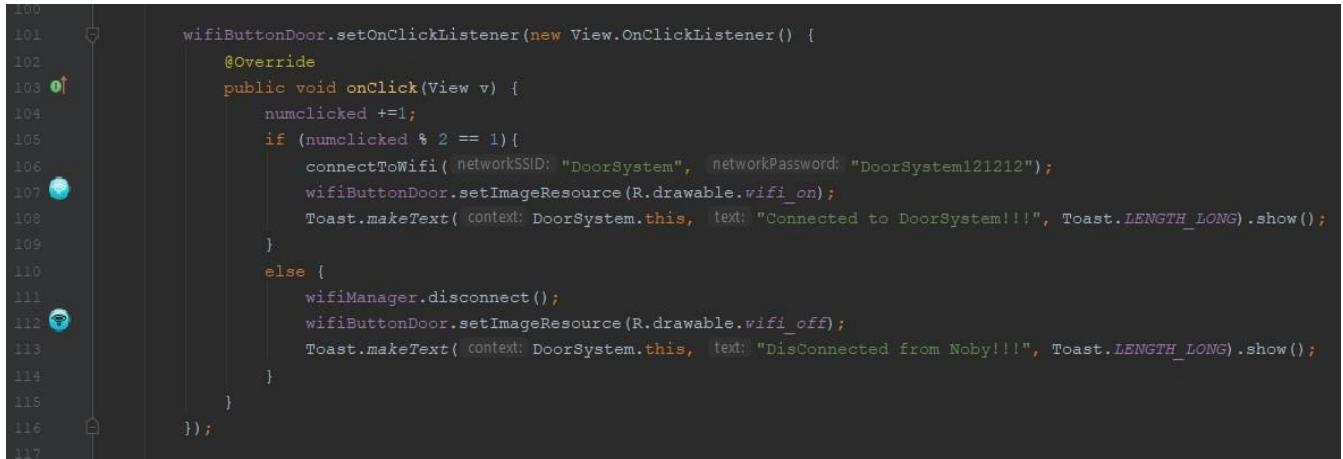
Reconnect to the currently active access point, if we are currently disconnected.

- And that's pretty much it, and we can now use our **connectToWiFi** method to connect to the Light and Door Hardware Systems.
- **Step4:** Fourth step would be using the **connectToWiFi** method that we created in the third step, in our WiFi Button onClick method, in Light System Activity and in Light System Activity (**Fig3.6**) (**Fig3.7**)



```
120
121     wifiButtonLight.setOnClickListener(new View.OnClickListener() {
122
123         @Override
124         public void onClick(View v) {
125             numclicked +=1;
126             if (numclicked % 2 == 1){
127                 connectToWiFi( networkSSID: "LightSystem",  networkPassword: "LightSystem121212");
128                 wifiButtonLight.setImageResource(R.drawable.wifi_on);
129                 Toast.makeText(context: LightSystem.this,  text: "Connected to LightSystem!!!",  Toast.LENGTH_LONG).show();
130             }
131             else {
132                 wifiManager.disconnect();
133                 wifiButtonLight.setImageResource(R.drawable.wifi_off);
134                 Toast.makeText(context: LightSystem.this,  text: "DisConnected from LightSystem!!!",  Toast.LENGTH_LONG).show();
135             }
136         });
137     });
138 }
```

- **Fig3.6 using connectToWiFi method in WiFiButtonLight onClick method in Light System Activity.**



```
100
101     wifiButtonDoor.setOnClickListener(new View.OnClickListener() {
102         @Override
103         public void onClick(View v) {
104             numclicked +=1;
105             if (numclicked % 2 == 1){
106                 connectToWifi( networkSSID: "DoorSystem", networkPassword: "DoorSystem121212");
107                 wifiButtonDoor.setImageResource(R.drawable.wifi_on);
108                 Toast.makeText( context: DoorSystem.this, text: "Connected to DoorSystem!!!", Toast.LENGTH_LONG).show();
109             }
110             else {
111                 wifiManager.disconnect();
112                 wifiButtonDoor.setImageResource(R.drawable.wifi_off);
113                 Toast.makeText( context: DoorSystem.this, text: "DisConnected from Noby!!!", Toast.LENGTH_LONG).show();
114             }
115         }
116     });
117 }
```

- Fig3.7 using `connectToWifi` method in `WiFiButtonDoor` `onClick` method in `Door System Activity`.
- Note: We only used WiFiManager in connecting with Light and Door Hardware Systems, because both Hardware systems have Wi-Fi module that we can connect to.
- After successfully connecting to the Light or Door a Toast message appears to ensure that the connection occurred.
- When clicking on the wifiButton again it disconnects the connection with the Wi-Fi module.
- In other words the variable (`numclicked`) is a counter of the number of how many clicks the wifiButton is clicked, in order to keep track of the connection situation (connected or disconnected).

4) Using the Bundle:

- Fourth thing we will discuss is using the Bundle to store some data in order to restore it when the activity is recreated again.
- So, in a nutshell the Fourth functionality we will be discussing, is how we can use bundles that are used by activities to pass data to themselves in the future.
- So, what we are trying to achieve is using the bundle to pass the current value of the variable numclicked which contains the number of times the wifi Button clicked in the Light Activity or Door Activity in the onSave method, and capturing that data from the bundle in the onRestore method.
- We will be using **Bundle** and will be using **onSaveInstanceState** method and also will be using **onRestoreInstanceState** method.
- **Bundle**: bundles are used by activities to pass data to themselves in the future, in scenarios such as When the screen rotates (orientation changes, from Normal to Landscape), or when another activity is started, usually when that happens android destroys the current Activity with some important data and we might not be

able to retain it in the future so that's why we use the Bundle.

- **onSaveInstanceState:** The `onSaveInstanceState()` method is a callback method that is not shown on most Activity lifecycle diagrams, this method is called by Android between the `onPause()` and `onStop()` methods, The `onSaveInstanceState()` method is passed a Bundle object as a parameter, Data you want saved for the activity should be placed in the Bundle.
- **onRestoreInstanceState:** the `onRestoreInstanceState()` method is used to extract saved state from the Bundle and restore it into new instances of the activity, The `onRestoreInstanceState()` method is another callback lifecycle method that is also rarely seen in the activity lifecycle diagrams, The `onRestoreInstanceState()` method is automatically invoked by Android between the `onStart()` and the `onResume()` lifecycle methods.
- **Code Reference:**
- In the following we will see how the previous description about saving data into a bundle and then restoring it again, and how we can achieve that by using `onSaveInstanceState` and `onRestoreInstanceState`, in Java code in **Android Studio**.

- **Step1:** First step would be adding the required data we want to save into the bundle in **onSaveInstanceState** method, in this case the variable numclicked. (**Fig4.1**)

```

145
146
147 @Override
148     protected void onSaveInstanceState(Bundle outState) {
149         outState.putString(TEXT_STATE, textViewState.getText().toString());
150         outState.putInt(numClickedString, numclicked);
151         super.onSaveInstanceState(outState);
152     }

```

- **Fig4.1** **onSaveInstanceState** method.

- **Step2:** Second step would be restoring that variable (numclicked) from **onRestoreInstanceState** method and checking if it was odd or even to identify the current wifiButton Background image value. (**Fig4.2**)

```

153
154 @Override
155     protected void onRestoreInstanceState(Bundle savedInstanceState) {
156         super.onRestoreInstanceState(savedInstanceState);
157         textViewState.setText(savedInstanceState.getString(TEXT_STATE));
158         numclicked = savedInstanceState.getInt(numClickedString);
159
160         if (numclicked % 2 == 1){
161             wifiButtonLight.setImageResource(R.drawable.wifi_on);
162         }else {
163             wifiButtonLight.setImageResource(R.drawable.wifi_off);
164         }
165     }

```

- **Fig4.2** **onRestoreInstanceState** method.

- **Note: With the previous way, we guarantee that the state of the WiFi connection and the state of the WiFi Button will be the same even if the orientation changed from Normal Layout to Landscape Layout.**

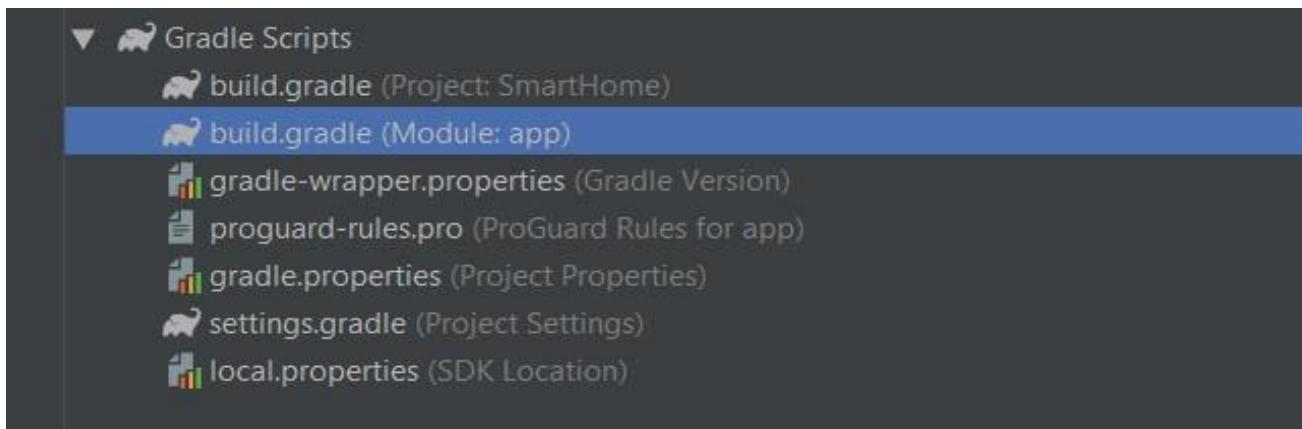
5) Using Retrofit in Light System Activity:

- Fifth thing we will discuss is using Retrofit in order to send an HTTP Post Request into the Server in the Light Hardware System, in order to be able to receive the data in that HTTP Post Request and to toggle light when receiving a particular message or in other words a particular string.
- So, in a nutshell the Fifth functionality we will be discussing, is how we can use Retrofit in order to be able to send a particular message or a particular string integrated in a HTTP Post Request.
- So, what we are trying to achieve is using Retrofit to pass a value to the Light Hardware System, that when received by the Light Hardware System, based on what is the message is the state of the Light will change (ON or OFF).

- And this Functionality is very important because it will give our Smart Home Application a great importance, because this functionality will make us be able to control the Light System Remotely from our Application, rather than using the primary way offered within the Light Hardware System.
- We will be using **Retrofit** and will be using **HTTP POST REQUEST** and will be using **Gradle dependencies** also **API Endpoint** and **REST API** and finally **JSON Converter**.
- **Retrofit:** is a type-safe HTTP client for Android and Java, Retrofit allows easy communication with a web service by abstracting the HTTP API into a Java interface, in other words it save us a lot of coding when we use it to send HTTP Post Requests to a Web Service or when we Receive HTTP Get Requests from a Web Service, which in our case the Light Hardware System Server.
- **HTTP POST REQUEST:** POST is one of the most common HTTP methods, POST is used to send data to a server to create/update a resource, and the data sent to the server with POST is stored in the request body of the HTTP request, or stored as parameters in the URL of the HTTP Request.

- **Gradle dependencies:** necessary Gradle dependencies in order to set up Retrofit and GSON Converter in Android Studio, in other words these dependencies are important and critical in order to be able to use Retrofit Functionality.
- **API:** An Application Programming Interface (API) allows two systems to communicate with one another, An API essentially provides the language and contract for how two systems interact, Each API has documentation and specifications which determine how information can be transferred, Just like a webpage is rendered, APIs can use HTTP requests to get information from a web application or web server.
- **REST API:** A REST API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.
- **API Endpoint:** Simply put, an endpoint is one end of a communication channel, When an API interacts with another system, the touch points of this communication are considered endpoints, For APIs, an endpoint can include a URL of a server or service, Each endpoint is the location from which APIs can access the resources they need to carry out their function.

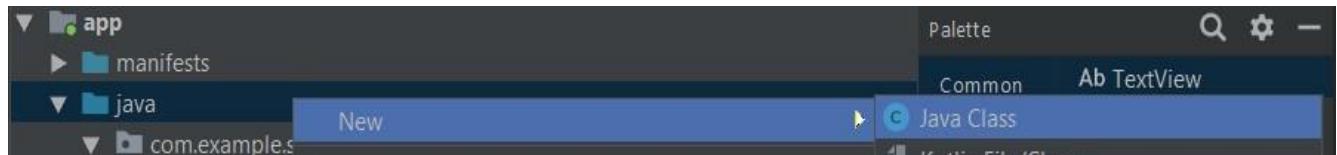
- **JSON**: is an open-source Java library to serialize and deserialize Java objects to and from JSON, in other words it help us converting data we would like to send to the appropriate form so that the server can understand it.
- [Code Reference](#):
- In the following, we will see how the previous description about sending HTTP POST Request, and how we can achieve that by using **Retrofit**, in **Java** code in **Android Studio**.
- **Step1**: First step would be adding necessary Gradle dependencies in order to set up Retrofit and GSON Converter in Android Studio, because these dependencies are important and critical in order to be able to use Retrofit Functionality. (**Fig5.1**) (**Fig5.2**)



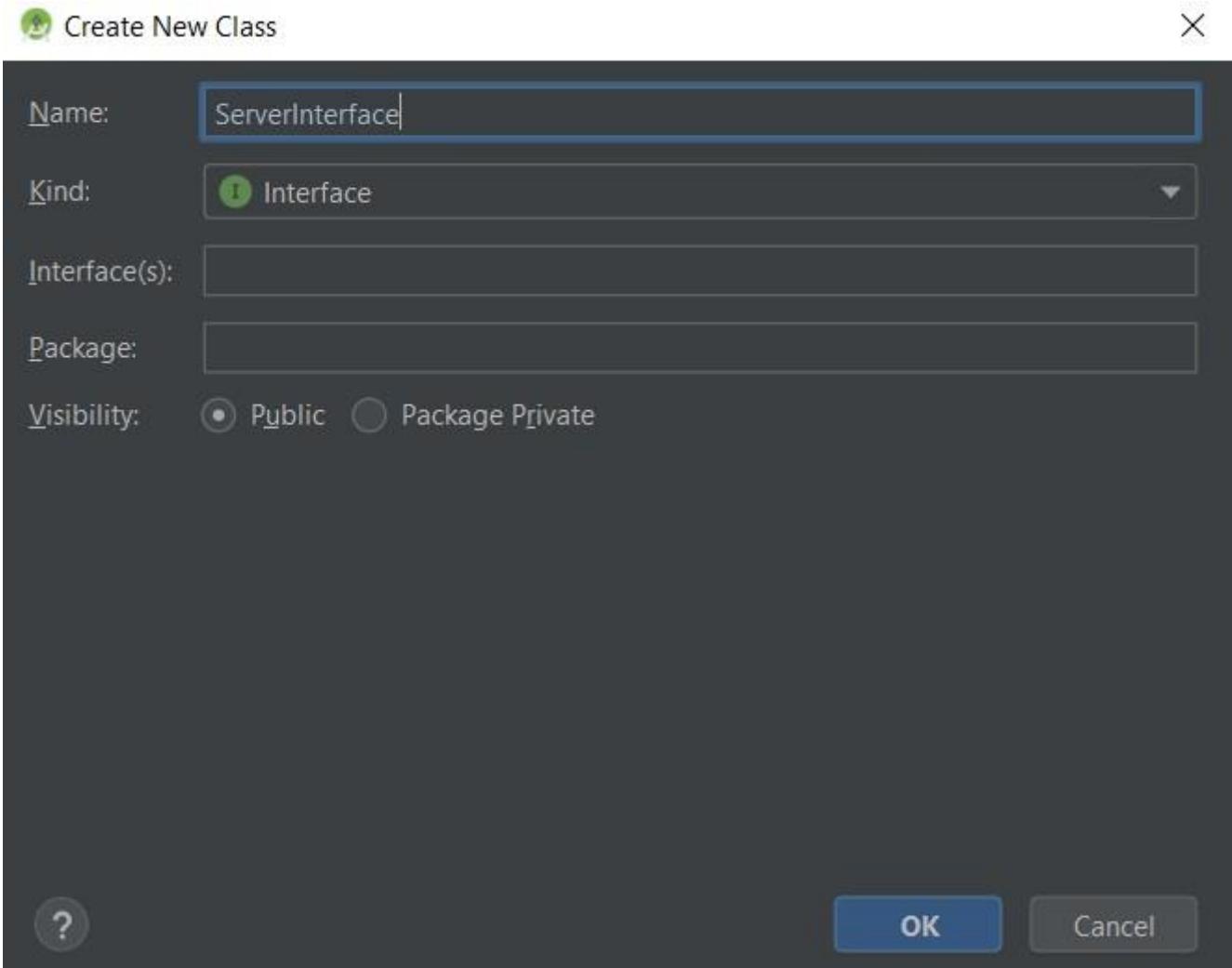
- **Fig5.1 gradle file.**

```
21     dependencies {  
22         implementation fileTree(dir: 'libs', include: ['*.jar'])  
23         implementation 'com.android.support:appcompat-v7:28.0.0'  
24         implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
25         implementation 'com.squareup.retrofit2:retrofit:2.5.0'  
26         implementation 'com.squareup.retrofit2:converter-gson:2.5.0'  
27         implementation 'com.google.code.gson:gson:2.8.5'  
28         testImplementation 'junit:junit:4.12'  
29         androidTestImplementation 'com.android.support.test:runner:1.0.2'  
30         androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
31     }
```

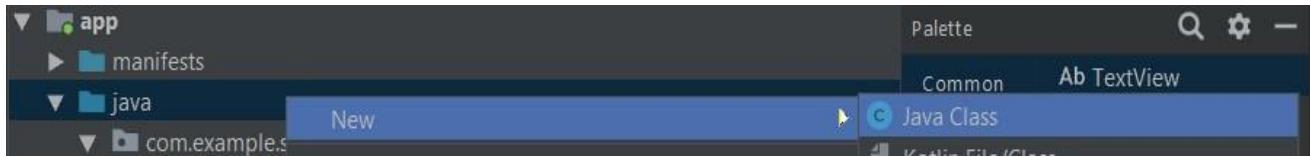
- Fig5.2 necessary Gradle dependencies.
- Note: The necessary dependencies we added are from line 25 to line 27.
- Step2: Second step would be creating an interface that would contain our (2) POST methods that we will be using to send data to the Light Hardware System, and that we annotate with @POST. (Fig5.3) (Fig5.4)



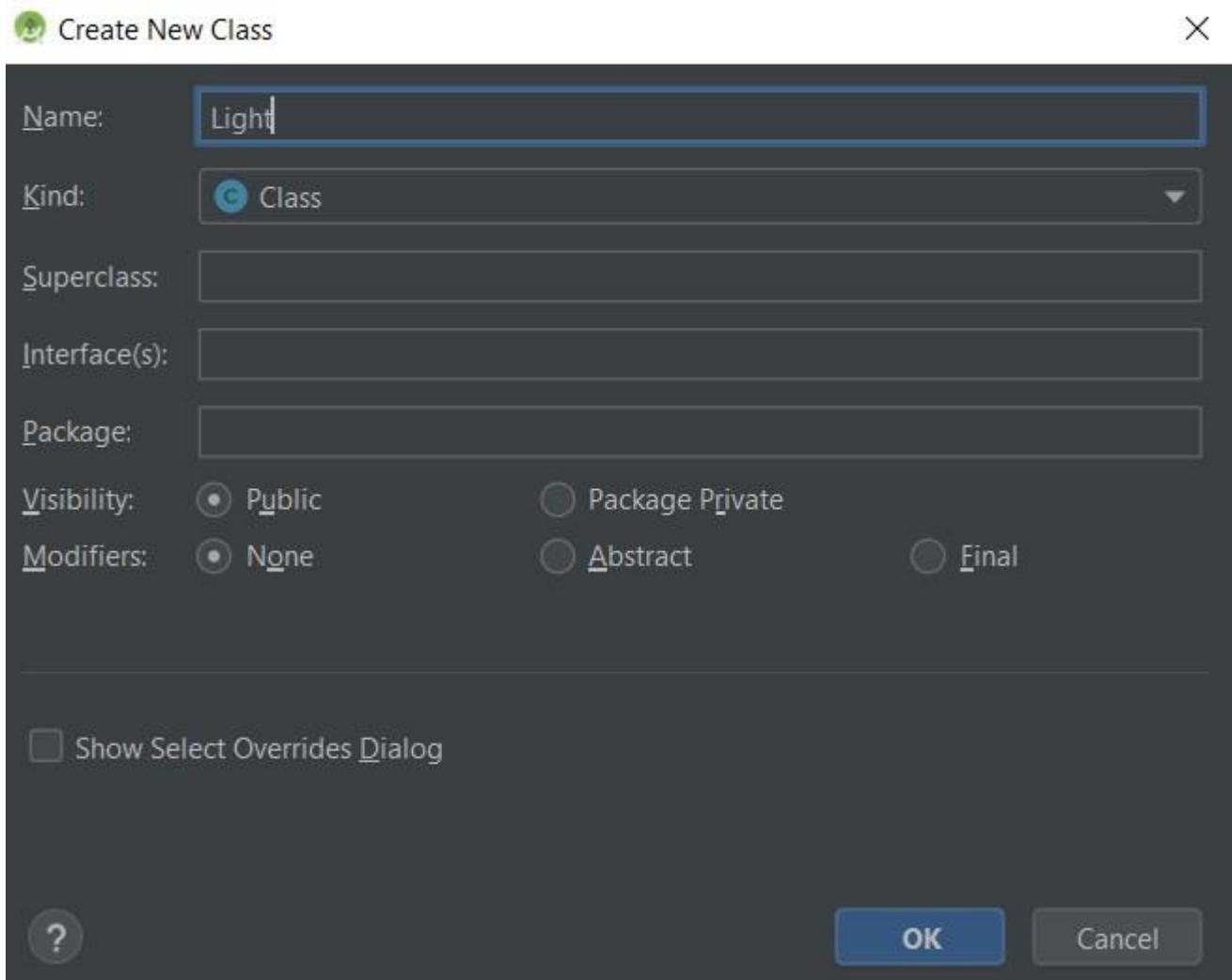
- Fig5.3 create an Interface class.



- Fig5.4 our interface called ServerInterface.
- **Step3:** Third step would be creating a Class that would contain our message details such as what is its parameter, and call it Light which represent the POST message we will be sending to the Server and create a constructor, so that we could create objects from that class. (Fig5.5) (Fig5.6)



- Fig5.5 create a class.



- Fig5.6 our POST Light Class.

- **Step4:** Fourth step would be adding code to the POST Light Class we created, adding the field of the Class and creating a constructor and a get method. (Fig5.7)

```
1 package com.example.smarthome;
2
3 import com.google.gson.annotations.Expose;
4 import com.google.gson.annotations.SerializedName;
5
6 public class Light {
7     @SerializedName("state")
8     @Expose
9     private String state;
10
11
12
13     public Light(String state) {
14         this.state = state;
15     }
16
17     public String getState() {
18         return state;
19     }
20
21 }
```

- Fig5.7 our POST Light Class.

- **Step5:** Fifth step would be adding code to the Interface Class we created, adding the 2 POST methods, (**Fig5.8**) (**Fig5.9**)

```
10     @FormUrlEncoded
11     @POST("/LEDOn")
12     Call<Light> createLightOn(
13         @Field("state") String state
14     );
```

- Fig5.8 Light on POST method.

```
16     @FormUrlEncoded  
17     @POST("/LEDOFF")  
18     Call<Light> createLightOff(  
19         @Field("state") String state  
20     );
```

- Fig5.9 Light off POST method.

- In (Fig5.8) (Fig5.9) we define one Post Request for Turning On Light and another one for Turning OFF Light, and create the 2 POST methods by the FormUrlEncoded way, which means that the POST Request contents would be sent to the server as parameters values.
- And we specify the API Endpoint that the POST Requests would be sent to.
- **Step6:** sixth step would be creating an instance from our Interface and an instance from Retrofit and Gson to the Light System Activity in order to use them. (Fig5.10) (Fig5.11) (Fig5.12)

```
37     private ServerInterface serverInterface;
```

- Fig5.10 instance of Interface class.

```
67     Gson gson = new GsonBuilder()  
68         .setLenient()  
69         .create();  
70  
71     OkHttpClient client = new OkHttpClient();
```

- Fig5.11 instance of Gson.

```
73     Retrofit retrofit = new Retrofit.Builder()
74         .baseUrl("http://192.168.4.1")
75         .client(client)
76         .addConverterFactory(GsonConverterFactory.create(gson))
77         .build();
78
79     serverInterface = retrofit.create(ServerInterface.class);
```

- **Fig5.12 instance of Retrofit class.**
- In (**Fig5.12**) we define the base URL of the Server, which indicates the address of which the POST Requests would be sent at.
- **Step7:** seventh step would be adding code to the Light System Activity in order to create the methods that will send Requests to the Server, and link that methods to the ON and OFF Buttons, so that when clicked a POST Request would be sent. (**Fig5.13**) (**Fig5.14**) (**Fig5.15**) (**Fig5.16**)

```
179     public void createState() {
180         Call<Light> call = serverInterface.createLightOn( state: "LEDON" );
181
182         call.enqueue(new Callback<Light>() {
183             @Override
184             ↑ public void onResponse(Call<Light> call, Response<Light> response) {
185                 if (!response.isSuccessful()) {
186                     Toast.makeText( context: LightSystem.this, text: "not connected" + response.code(), Toast.LENGTH_LONG).show();
187                     return;
188                 }
189                 Light lightResponse = response.body();
190                 String content = "";
191                 content += "Code: " + response.code() + "\n";
192                 content += "State: " + lightResponse.getState() + "\n";
193
194                 textViewState.setText(content);
195             }
196
197             @Override
198             public void onFailure(Call<Light> call, Throwable t) {
199                 textViewState.setText(t.getMessage());
200             }
201         });
202     };
203 }
```

- Fig5.13 createState method to send a POST Request that includes 'LEDON' string.

```
82     buttonOn.setOnClickListener(new View.OnClickListener() {
83
84         ↑     @Override
85         public void onClick(View v) {
86             textViewState.setText("");
87             String result = "The Light Is ON. \n";
88             textViewState.append(result);
89             createState();
90         }
91     });
92 }
```

- Fig5.14 calling the createState from the ON Button onClick method.

```
205     public void createState2() {
206         Call<Light> call = serverInterface.createLightOff( state: "LEDOFF");
207
208         call.enqueue(new Callback<Light>() {
209             @Override
210             public void onResponse(Call<Light> call, Response<Light> response) {
211                 if (!response.isSuccessful()) {
212                     Toast.makeText( context: LightSystem.this, text: "not connected" + response.code(), Toast.LENGTH_LONG).show();
213                     return;
214                 }
215                 Light lightResponse = response.body();
216                 String content = "";
217                 content += "Code: " + response.code() + "\n";
218                 content += "State: " + lightResponse.getState() + "\n";
219
220                 textViewState.setText(content);
221             }
222
223             @Override
224             public void onFailure(Call<Light> call, Throwable t) {
225                 textViewState.setText(t.getMessage());
226             }
227         });
228     };
229 }
```

- Fig5.15 createState2 method to send a POST Request that includes ‘LEDOFF’ string.

```
92     buttonOff.setOnClickListener(new View.OnClickListener() {
93
94         @Override
95         public void onClick(View v) {
96             textViewState.setText("");
97             String result = "The Light Is OFF. \n";
98             textViewState.append(result);
99             createState2();
100        }
101    );
102 }
```

- Fig5.16 calling the createState2 from the OFF Button onClick method.

● Note: From the previous steps, we described how we would be able to send POST Requests to the Light Hardware System Server, in order to be able to control the Light in the Light Hardware System.

6) Using Retrofit in Door System Activity:

- Sixth thing we will discuss is using Retrofit in order to send an HTTP Post Request into the Server in the Door Hardware System, in order to be able to receive the data in that HTTP Post Request and to open Door when receiving a particular message or in other words a particular string.
- So, in a nutshell the sixth functionality we will be discussing, is how we can use Retrofit in order to be able to send a particular message or a particular string integrated in a HTTP Post Request.
- So, what we are trying to achieve is using Retrofit to pass a value to the Door Hardware System, that when received by the Door Hardware System, based on what is the message is the door will open.
- And this Functionality is very important because it will give our Smart Home Application a great importance, because this functionality will make us be able to control the Door System Remotely from our Application, rather than using the primary way offered within the Door Hardware System.

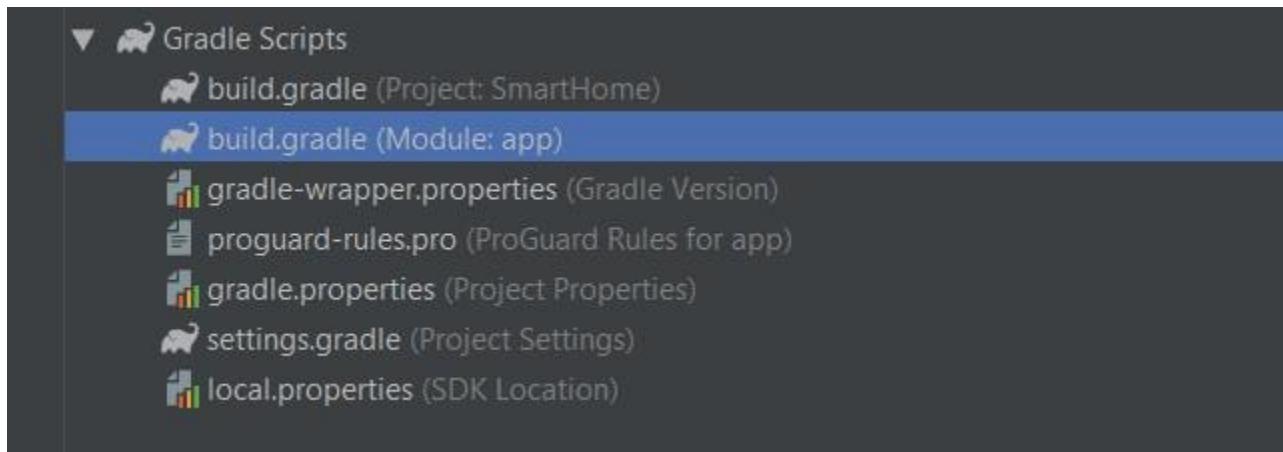
- We will be using **Retrofit** and will be using **HTTP POST REQUEST** and also will be using **Gradle dependencies** and also **API Endpoint** and **REST API** and finally **JSON Converter**.
- **Retrofit:** is a type-safe HTTP client for Android and Java, Retrofit allows easy communication with a web service by abstracting the HTTP API into a Java interface, in other words it save us a lot of coding when we use it to send HTTP Post Requests to a Web Service or when we Receive HTTP Get Requests from a Web Service, which in our case the Door Hardware System Server.
- **HTTP POST REQUEST:** POST is one of the most common HTTP methods, POST is used to send data to a server to create/update a resource, the data sent to the server with POST is stored in the request body of the HTTP request, or stored as parameters in the URL of the HTTP Request.
- **Gradle dependencies:** necessary Gradle dependencies in order to set up Retrofit and JSON Converter in Android Studio, in other words these dependencies are important and critical in order to be able to use Retrofit Functionality.

- **API**: An Application Programming Interface (API) allows two systems to communicate with one another, An API essentially provides the language and contract for how two systems interact, Each API has documentation and specifications which determine how information can be transferred, Just like a webpage is rendered, APIs can use HTTP requests to get information from a web application or web server.
- **REST API**: A REST API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.
- **API Endpoint**: Simply put, an endpoint is one end of a communication channel, When an API interacts with another system, the touch points of this communication are considered endpoints, For APIs, an endpoint can include a URL of a server or service, Each endpoint is the location from which APIs can access the resources they need to carry out their function.
- **JSON**: is an open-source Java library to serialize and deserialize Java objects to and from JSON, in other words it help us converting data we would like to send to the appropriate form so that the server can understand it.

- **Code Reference:-**

- In the following we will see how the previous description about sending HTTP POST Request, and how we can achieve that by using **Retrofit**, in **Java** code in **Android Studio**.
- **Step1:** First step similar to the previous functionality would be adding necessary Gradle dependencies in order to set up Retrofit and GSON Converter in Android Studio, because these dependencies are important and critical in order to be able to use Retrofit Functionality.

(Fig6.1) (Fig6.2)

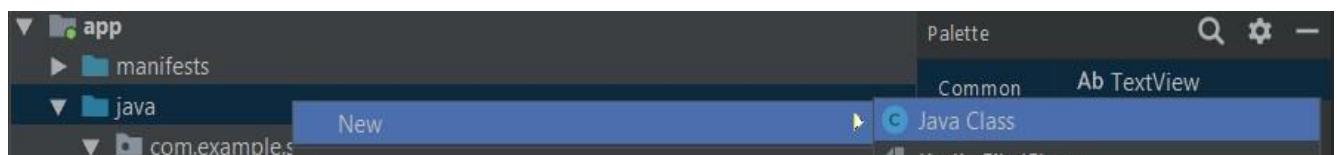


- **Fig6.1 gradle file.**

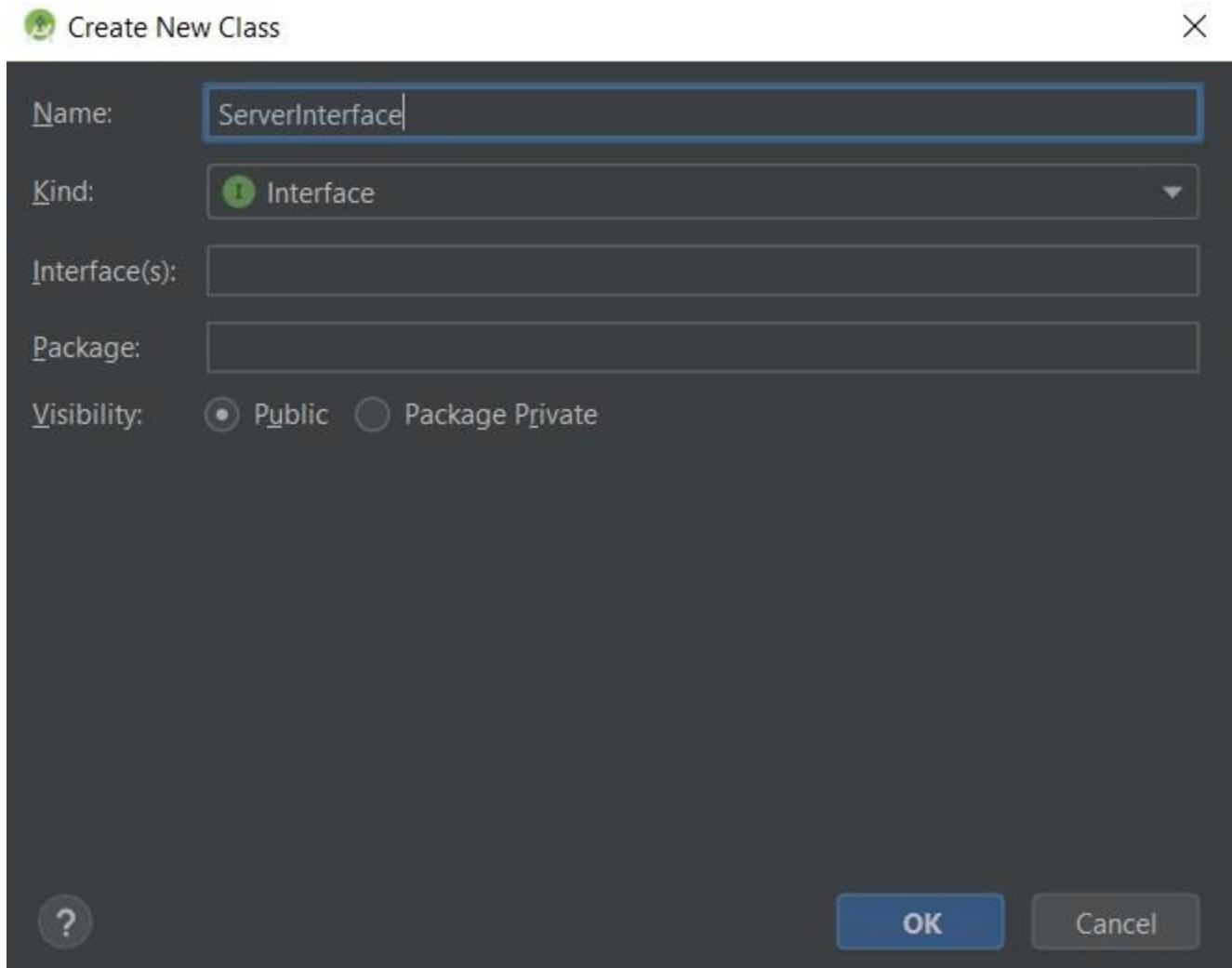
```
21     dependencies {  
22         implementation fileTree(dir: 'libs', include: ['*.jar'])  
23         implementation 'com.android.support:appcompat-v7:28.0.0'  
24         implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
25         implementation 'com.squareup.retrofit2:retrofit:2.5.0'  
26         implementation 'com.squareup.retrofit2:converter-gson:2.5.0'  
27         implementation 'com.google.code.gson:gson:2.8.5'  
28         testImplementation 'junit:junit:4.12'  
29         androidTestImplementation 'com.android.support.test:runner:1.0.2'  
30         androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
31     }  
32 }
```

- Fig6.2 necessary Gradle dependencies.

- Note: The necessary dependencies we added are from line 25 to 27.
- We already added them in the previous Functionality, this is only for enlightenment.
- Step2: Second step would be creating an interface that would contain our POST method that we will be using to send data to the Door Hardware System, and that we annotate with @POST. (Fig6.3) (Fig6.4)

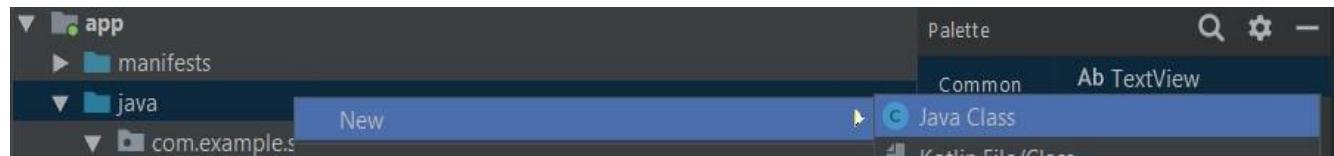


- Fig6.3 create an Interface class.

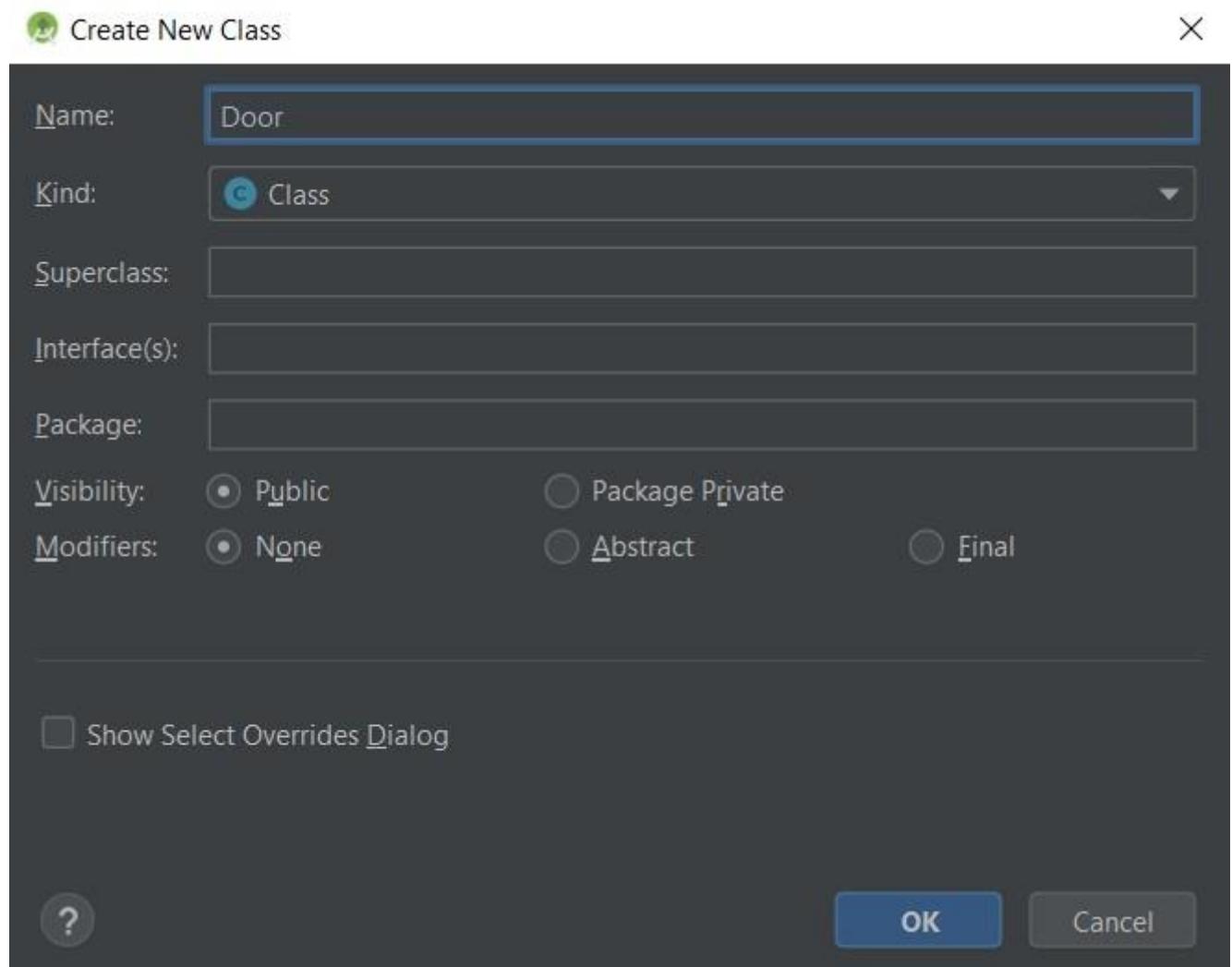


- Fig6.4 our interface called ServerInterface.
- **Note:** We already created the interface in the previous Functionality, this is only for enlightenment.
- **Step3:** Third step would be creating a Class that would contain our message details such as what is its parameter, and call it Door which represent the POST message we will be sending to the Server and create a

constructor, so that we could create objects from that class. (**Fig6.5**) (**Fig6.6**)



• **Fig6.5 create a class.**



• **Fig6.6 our POST Door Class.**

- **Step4:** Fourth step would be adding code to the POST Door Class we created, adding the field of the Class and creating a constructor and a get method. (**Fig6.7**)

```

1  package com.example.smarthome;
2
3  import com.google.gson.annotations.Expose;
4  import com.google.gson.annotations.SerializedName;
5
6  public class Door {
7
8      @SerializedName("DoorState")
9      @Expose
10     private String DoorState;
11
12     public Door(String state) {
13         this.DoorState = state;
14     }
15
16     public String getDoorState() {
17         return DoorState;
18     }
19
20 }
```

• Fig6.7 our POST Door Class.

- **Step5:** Fifth step would be adding code to the Interface Class we created in previous step, adding the POST method. (**Fig6.8**)

```

22     @FormUrlEncoded
23     @POST("/opendoor")
24     Call<Door> createDoor(
25         @Field("DoorState") String DoorState
26     );
```

• Fig6.8 open Door POST method.

- In (**Fig6.8**) we define one Post Request for opening the door, and create the POST method by the FormUrlEncoded way, which means that the POST Request contents would be sent to the server as parameters values, and we also specify the API Endpoint that the POST Requests would be sent to.
- **Step6:** Sixth step would be creating an instance from our Interface and an instance from Retrofit and Gson to the Door System Activity in order to use them. (**Fig6.9**) (**Fig6.10**) (**Fig6.11**)

37

```
private ServerInterface serverInterface;
```

- **Fig6.9 instance of Interface class.**

```
67:     Gson gson = new GsonBuilder()
68:         .setLenient()
69:         .create();
70:
71:     OkHttpClient client = new OkHttpClient();
```

- **Fig6.10 instance of Gson.**

```
73:     Retrofit retrofit = new Retrofit.Builder()
74:         .baseUrl("http://192.168.4.1")
75:         .client(client)
76:         .addConverterFactory(GsonConverterFactory.create(gson))
77:         .build();
78:
79:     serverInterface = retrofit.create(ServerInterface.class);
```

- **Fig6.11 instance of Retrofit class.**

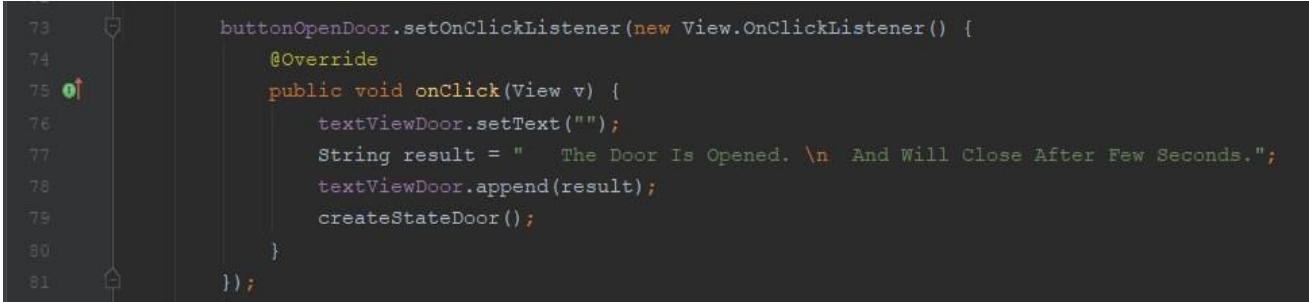
- In (**Fig6.11**) we define the base Url of the Server, which indicates the address of which the POST Requests would be sent at.
- **Step7:** seventh step would be adding code to the Door System Activity in order to create the method that will send Requests to the Server, and link that method to the OPEN door Button, so that when clicked a POST Request would be sent. (**Fig6.12**) (**Fig6.13**)

```

158
159
160
161
162
163
164 0↑
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179 0↑
180
181
182
183
    public void createStateDoor() {
        Call<Door> call = serverInterface.createDoor( DoorState: "opendoor");
        call.enqueue(new Callback<Door>() {
            @Override
            public void onResponse(Call<Door> call, Response<Door> response) {
                if (!response.isSuccessful()) {
                    Toast.makeText( context: DoorSystem.this, text: "not connected" + response.code(), Toast.LENGTH_LONG).show();
                    return;
                }
                Door doorResponse = response.body();
                String content = "";
                content += "Code: " + response.code() + "\n";
                content += "State: " + doorResponse.getDoorState() + "\n";
                textViewDoor.setText(content);
            }
            @Override
            public void onFailure(Call<Door> call, Throwable t) {
                textViewDoor.setText(t.getMessage());
            }
        });
    }

```

- **Fig6.12** `createStateDoor` method to send a POST Request that includes ‘`opendoor`’ string.



```
73 buttonOpenDoor.setOnClickListener(new View.OnClickListener() {
74     @Override
75     public void onClick(View v) {
76         textViewDoor.setText("");
77         String result = "    The Door Is Opened. \n    And Will Close After Few Seconds.";
78         textViewDoor.append(result);
79         createStateDoor();
80     }
81 }) ;
```

- Fig6.13 calling the createStateDoor from the OPEN Button onClick method.
- **Note: From the previous steps, we described how we would be able to send POST Requests to the Door Hardware System Server, in order to be able to control the Door in the Door System.**

7) Receiving the video from Security System and Stream it:

- Seventh thing we will discuss is using WebView in order to show the live stream of the Security System's Camera, in order to be able to know who is trying to get into the Home, without permission.
- So, in a nutshell the seventh functionality we will be discussing, is how we can use WebView in order to be able to show the live stream of the Security System's Camera.
- So, what we are trying to achieve is using WebView to load the Camera's URL (Static IP), which is provided by

the Security Hardware System on the same LAN Network.

- And this Functionality is very important because it will give our Smart Home Application a great importance, because this functionality will make us be able to see what happens in real time through the camera and hence we can make decisions based on what we see, if there is an intruder then we can take our precautions and call the police.
- We will be using **WebView** and will be using **WifiManager** and will be using **App Manifest** also **Permissions & Bundle**.
- **WebView:** WebView objects allow us to display web content as part of our activity layout, but lack some of the features of fully-developed browsers, A WebView is useful when we need increased control over the UI and advanced configuration options that will allow us to embed web pages in a specially-designed environment for our app.
- **WifiManager:** This class provides the primary API for managing all aspects of Wi-Fi connectivity, It deals with several categories of items:
 1. The list of configured networks.

2. The currently active Wi-Fi network, if any.
Connectivity can be established or torn down.
 3. Results of access point scans, containing enough information to make decisions about what access point to connect to.
- **App Manifest:** Every app project must have an **AndroidManifest** xml file (with precisely that name) at the root of the project source set, The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play, the manifest file is required to declare the following:
 1. The app's package name.
 2. The components of the app, which include all activities, services.
 3. The permissions that the app needs.
 4. The hardware and software features the app requires.
 - **Permissions:** The permissions that the app needs in order to access protected parts of the system or other apps. It also declares any permissions that other apps must have if they want to access content from this app, Android apps must request permission to access certain

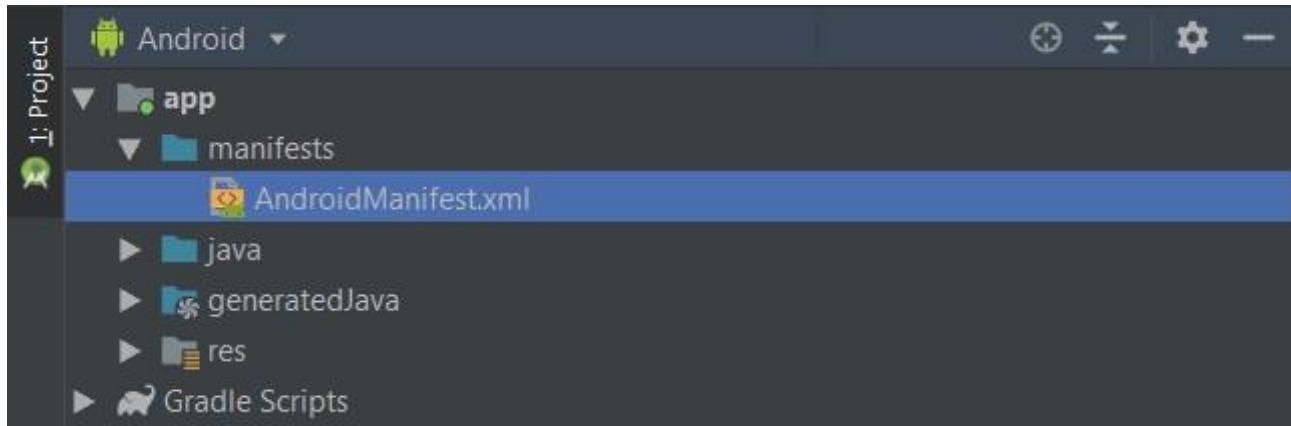
system features (such as the camera and internet access), Each permission is identified by a unique label.

- **Bundle:** bundles are used by activities to pass data to themselves in the future, in scenarios such as When the screen rotates (orientation changes, from Normal to Landscape), or when another activity is started, usually when that happens android destroys the current Activity with some important data and we might not be able to retain it in the future so that's why we use the Bundle.

- **Code Reference:**

- In the following we will see how the previous description about showing the live stream of the Security Hardware System's Camera, and how we can achieve that by using **WebView**, in **Java** code in **Android Studio**.
- **Step1:** First step similar to the Connecting to the Light and Door Systems previous functionality, hence using WifiManager in order to connect to the Same LAN provided by the Security Hardware System which includes the URL (Static IP) of the Camera, so first step would be adding the required permissions in the App Manifest file, which will help us to access certain system

features (such as internet access and manipulating Wi-Fi state). (Fig7.1) (Fig7.2)



- Fig7.1 App Manifest File.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.smarthome">
4          <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
5          <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
6          <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
7          <uses-permission android:name="android.permission.INTERNET"/>
```

- Fig7.2 Required permissions in App Manifest file.

- Note: We already added the required permissions in previous Functionality, this is only for enlightenment.
- Step2: Second step would be creating an instance of the **WifiManager** class in order to use it when creating the connection method. (Fig7.3) (Fig7.4)

```
34
```

```
private WifiManager wifiManager;
```

- Fig7.3 WifiManager instance.

```
59  
60  
61     wifiManager = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SERVICE);  
62     wifiManager.setWifiEnabled(true);
```

- Fig7.4 WiFiManager instance.

- **Step3:** Third step would be creating our **connectToWiFi** method that we will be passing two parameters (network ssid, network password) into it from our WiFi Button in order to connect to the LAN provided by the Security Hardware System. (**Fig7.5**)

```
166  
167     public void connectToWiFi(String networkSSID, String networkPassword) {  
168  
169         WifiConfiguration conf = new WifiConfiguration();  
170         conf.SSID = String.format("\"%s\"", networkSSID);  
171         conf.preSharedKey = String.format("\"%s\"", networkPassword);  
172  
173         int netId = wifiManager.addNetwork(conf);  
174         wifiManager.disconnect();  
175         wifiManager.enableNetwork(netId, attemptConnect: true);  
176         wifiManager.reconnect();  
177     }
```

- Fig7.5 connectToWiFi method.

- **Note:** Line 169 creates a **WifiConfiguration** object which represents a configured Wifi network, including the security configurations for the Wifi network.
- Line 170 and 171 are public calls to the **SSID** and the **preSharedKey** variables to set the Wifi SSID and password to the configurations. Both the SSID and the password need to be enclosed around quotes, hence the use of the **String.format()** method to do so.

- Line 173 adds the network configurations to the WifiManager, which then returns the network id which is used to specify the network to be acted upon.
 - Line 174 disassociates from the currently active access point.
 - Line 175 does the actual connection to the Wifi network by calling enableNetwork() and passing in the network id we got previously, and passing in true to let the WifiManager know to disable every other network.
 - Line 176 is a safe check in case the initial connection failed, What the reconnect() method does is Reconnect to the currently active access point, if we are currently disconnected.
 - And that's pretty much it, and we can now use our connectToWiFi method to connect to the LAN provided by the Security Hardware System.
-
- **Step4:** Fourth step would be using the **connectToWiFi** method that we created in the third step, in our WiFi Button onClick method, in Security System Activity (**Fig7.6**)

```
81     wifiButtonSecurity.setOnClickListener(new View.OnClickListener() {
82         @Override
83         public void onClick(View v) {
84             numclicked +=1;
85             if (numclicked % 2 == 1){
86                 connectToWifi( networkSSID: "SecuritySystem", networkPassword: "SecuritySystem121212");
87                 wifiButtonSecurity.setImageResource(R.drawable.wifi_on);
88                 Toast.makeText( context: SecuritySystem.this, text: "Connected to SecuritySystem!!!", Toast.LENGTH_LONG).show();
89             }
90             else {
91                 wifiManager.disconnect();
92                 wifiButtonSecurity.setImageResource(R.drawable.wifi_off);
93                 Toast.makeText( context: SecuritySystem.this, text: "DisConnected from SecuritySystem!!!", Toast.LENGTH_LONG).show();
94             }
95         }
96     });
}
```

- Fig7.6 using **connectToWifi** method in **WiFiButtonSecurity** **onClick** method in **Security System Activity**.
- Note: We used **WiFiManager** in connecting with the LAN provided by the Security Hardware System, because the Security Hardware systems can provide a LAN that we can connect to.
- After successfully connecting to the LAN a **Toast** message appears to ensure that the connection occurred.
- When clicking on the **wifiButton** again it disconnects the connection from the LAN.
- In other words the variable (**numclicked**) is a counter of the number of how many clicks the **wifiButton** is clicked, in order to keep track of the connection situation (connected or disconnected).
- Step5: Fifth step would be using the **WebView**, in our Play Button **onClick** method, in **Security System Activity** (Fig7.7) (Fig7.8)

```
29         private String webviewButtonClickString = "";
30
31         private String TEXT_URL = "";
```

- Fig7.7 creating TEXT_URL String variable.

```
52             buttonPlayVideo.setOnClickListener(new View.OnClickListener() {
53                 @Override
54                 public void onClick(View v) {
55
56                     TEXT_URL += "A";
57
58                     webview.setWebViewClient(new WebViewClient());
59                     webview.getSettings().setJavaScriptEnabled(true);
60                     webview.getSettings().setBuiltInZoomControls(true);
61                     webview.getSettings().setUseWideViewPort(true);
62                     webview.loadUrl("http://192.168.1.7/SecurityCamera/");
63                 }
64             });

```

- Fig7.8 using WebView in the PLAYVideo Button onClick method.

- **Step6:** Sixth step would be adding the required data we want to save into the bundle in **onSaveInstanceState** method, in this case the variable (numclicked) which indicates how many times the WiFi button in Security Activity clicked, and the variable (TEXT_URL) which indicates how many times the Play button clicked, since each time the PLAY button is clicked an ‘A’ is added to the String. (Fig7.9)

```
108     @Override  
109     protected void onSaveInstanceState(Bundle outState) {  
110         outState.putInt(numClickedString ,numclicked);  
111         outState.putString(webviewButtonClickString,TEXT_URL);  
112         super.onSaveInstanceState(outState);  
113     }
```

- Fig7.9 onSaveInstanceState method.

- **Step7:** Seventh step would be restoring that variable (numclicked) from **onRestoreInstanceState method** and checking if it was odd or even to identify the current wifiButton Background image value, and also would be restoring that variable (TEXT_URL) from **onRestoreInstanceState method** and checking if it was NOT NULL, and also checking if there is a wifi connection or not, and we do that to ensure that the state of the WebView stays the same after orientation changes (**Fig7.10**)

```
115     @Override  
116     protected void onRestoreInstanceState(Bundle savedInstanceState) {  
117         super.onRestoreInstanceState(savedInstanceState);  
118         numclicked = savedInstanceState.getInt(numClickedString);  
119         if (numclicked % 2 == 1){  
120             wifiButtonSecurity.setImageResource(R.drawable.wifi_on);  
121         }else {  
122             wifiButtonSecurity.setImageResource(R.drawable.wifi_off);  
123         }  
124  
125         TEXT_URL = savedInstanceState.getString(webviewButtonClickString);  
126         if (TEXT_URL != "" && wifiManager.isWifiEnabled()){  
127             webview.setWebViewClient(new WebViewClient());  
128             webview.getSettings().setJavaScriptEnabled(true);  
129             webview.getSettings().setBuiltInZoomControls(true);  
130             webview.getSettings().setUseWideViewPort(true);  
131             webview.loadUrl("http://192.168.1.7/SecurityCamera/");  
132         }  
133     }  
134 }
```

- Fig7.10 onRestoreInstanceState method.

- Note: With the previous way, we guarantee that the state of the Wi-Fi connection and the state of the Wi-Fi Button will be the same even if the orientation changed from Normal Layout to Landscape Layout.
- Also we guarantee that the state of the WebView will be the same even if the orientation changed from Normal Layout to Landscape Layout.

CHAPTER 6

{System Analysis & Design}



6.1 objective

- Review the key Analysis and design terms and concepts.
- Introduce the analysis and design process, including roles, artifacts and workflow.
- Explain the difference between analysis and design.

Analysis and design in context:

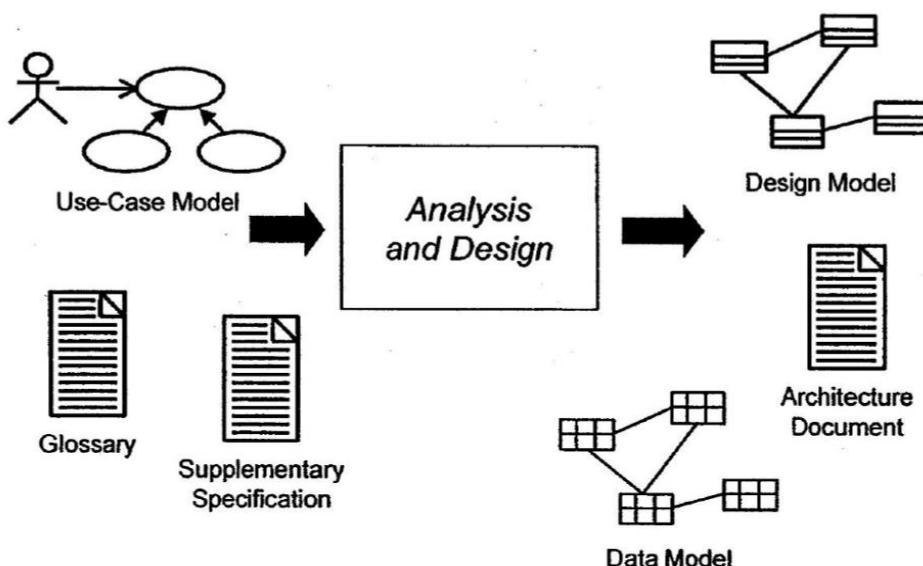
- The requirements discipline provides the primary input for analysis and design.
- **The purposes of analysis and design are to:**
 - Transform the requirements into a system design.
 - Evolve a robust architecture for the system.
 - Adapt the design to match the implementation environment, designing it for performance.

Analysis and design in overview:

- The result of analysis and design is a design model that serves as an abstraction of the source code: that is, the design model acts as a blueprint of how the source code is structured and written. The design model consists of design classes structured into design packages: it also contains descriptions of how

objects of these design classes collaborate to perform use cases (use-case realizations.)

- The design activities are centered on the notion of architecture. The production and validation of this architecture is the focus of early design iterations.
- A number of architectural views that capture the major structural design decisions represents architecture. In essence, architectural vies are abstractions or simplifications of the entire design, in which important characteristics are made more visible by leaving details aside. The architecture is an important vehicle not only for developing a good design model, but also for increasing the quality of any model built during system development. The architecture is documented in the architecture document.



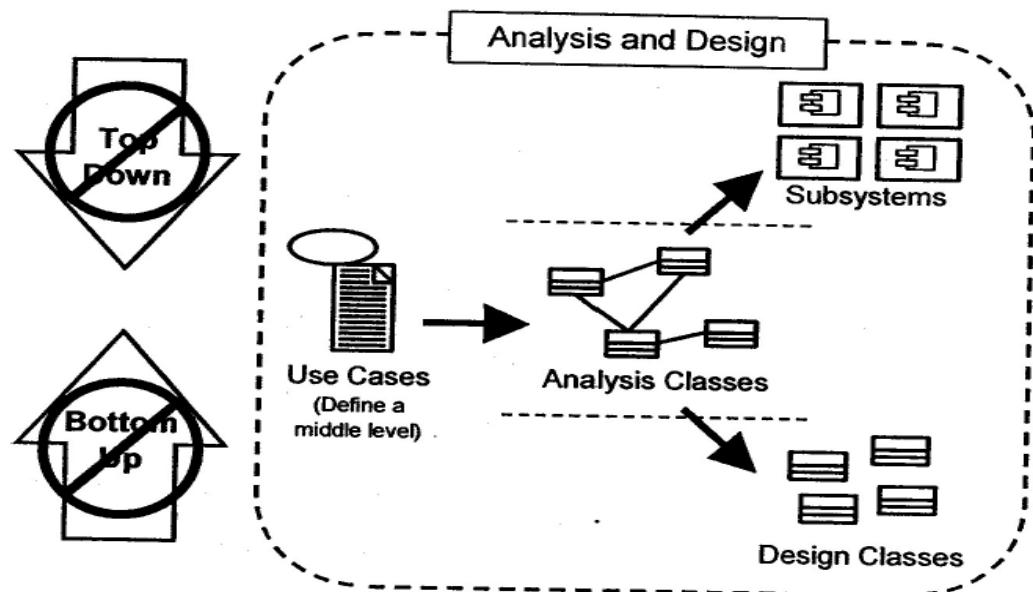
6.2 Analysis Versus Design:

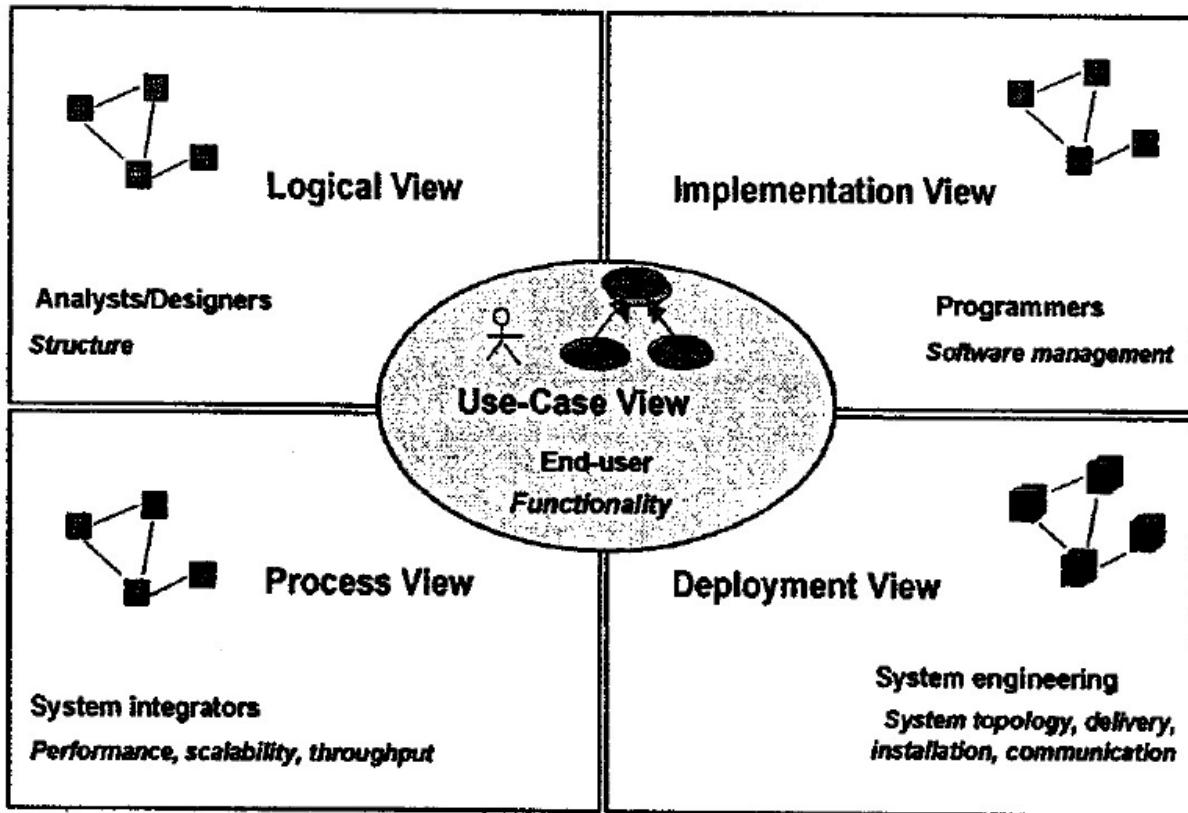
- The goal in analysis is to understand the problem and to begin to develop a visual model of what you are trying to build, independent of implementation and technology concerns, analysis focuses on translating the functional requirements into software concepts. The idea is to get a rough cut at the objects that comprise our system, but focusing on behavior (and therefore encapsulation). We then move very quickly, nearly seamlessly, into ‘Design’ and the other concerns.
- A goal of design is to refine the model with the intention of developing a design model that will allow a seamless transition to the coding phase. In design, we adapt to the implementation and the deployment environment. The implementation environments is the “developer” environment, which is a software superset and a hardware subset of the deployment environment.

Analysis	Design
<ul style="list-style-type: none"> • Focus on understanding the problem. • Idealized design. • Behavior. • System structure. • Functional requirement. • A small model. 	<ul style="list-style-type: none"> • Focus on understanding the solution. • Operations and attributes. • Performance. • Close to real code. • Object lifecycles. • Nonfunctional requirements. • A large model.

6.3 Analysis and design are not top-down or bottom-up:

Analysis is both top-to-middle, middle-up, middle-down and bottom-to-middle. There is no way of saying that one path is more important than another, you have to travel on all paths to get the system right. All of these four paths are equally important. That is why the bottom-up and top-down question cannot be solved.

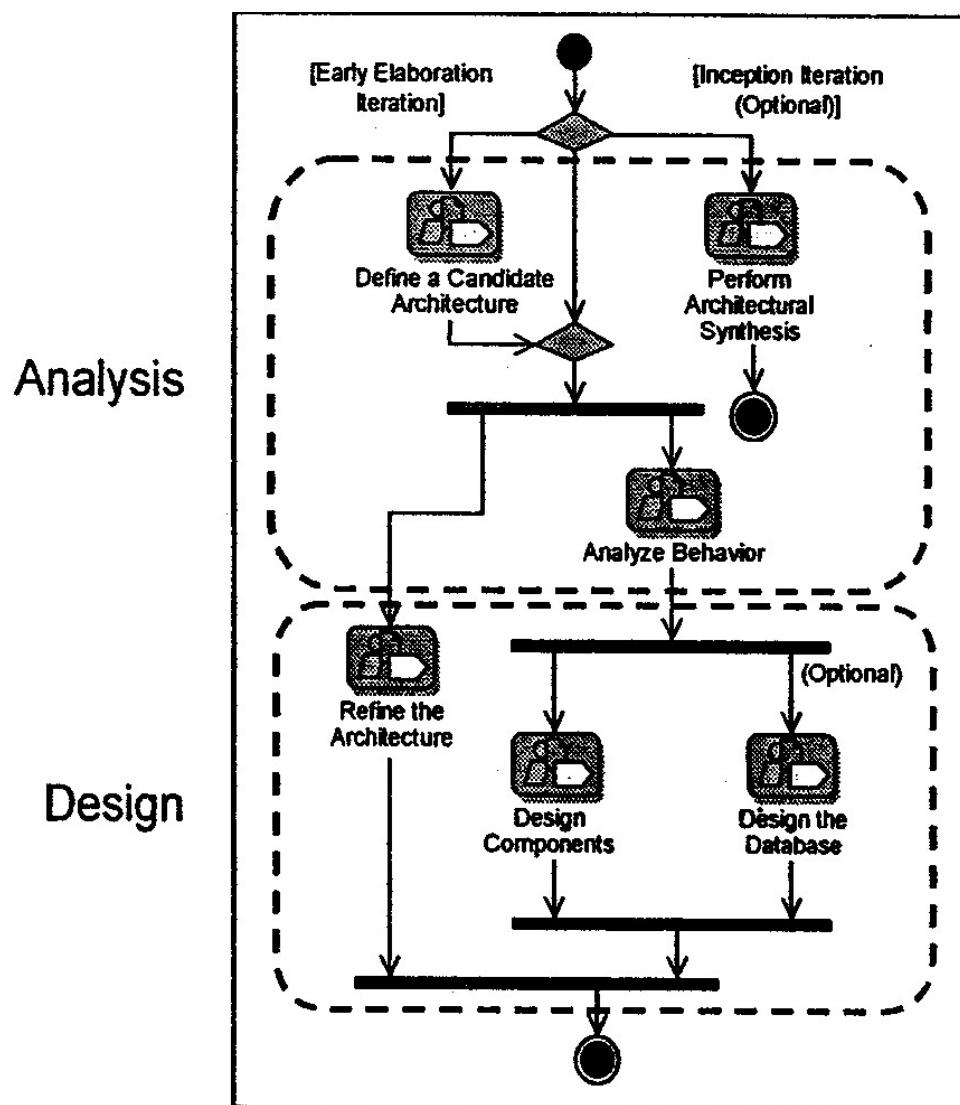




6.4 Analysis and design workflow:

- The early elaboration phase focuses on creating an initial architecture for the system (define a candidate architecture) to provide a starting point for the main analysis work. If the architecture already exists (because it was produced in previous iteration, or projects, or is obtained from an application framework), the focus of the work changes to refining the architecture (refine the architecture), analyzing behavior, and creating an initial set of elements that provide the appropriate behavior (analyze behavior).

- After the initial elements are identified, they are further refined. Design components produce a set of components that provide the appropriate behavior so satisfy the requirement on the system. In parallel with these activities, persistence issues are handling in design the database. The result is an initial set of components that are further refined in the implementation discipline.

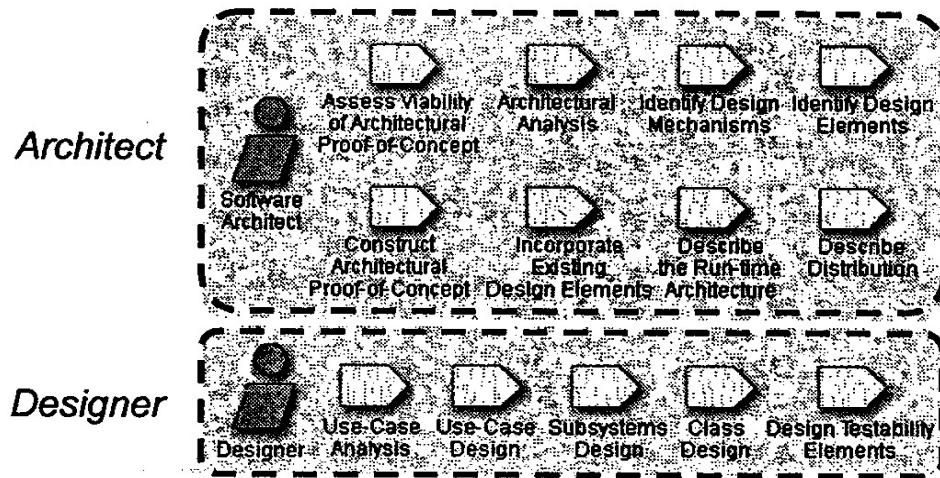


Analysis and Design Activity Overview:

The designs activities are centered on the notion of architecture.

The production and validation of this architecture are the main focal points of early design iterations.

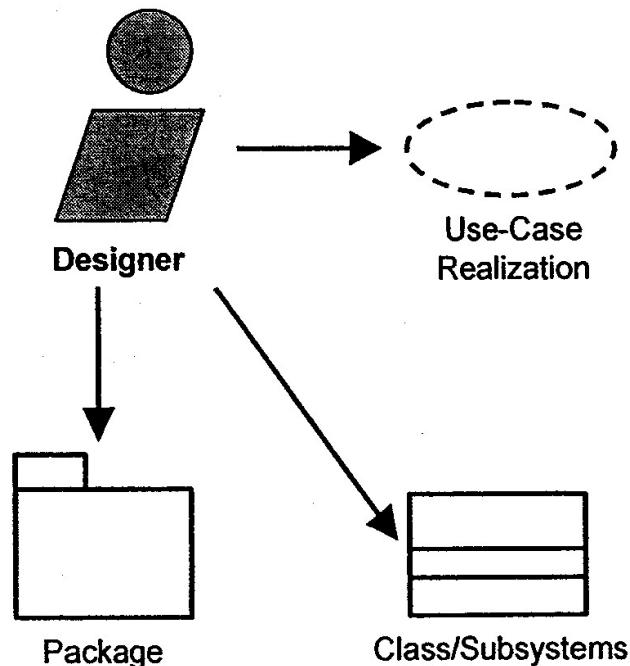
The focus of this course is on activities of the designer.



Designer Responsibilities:

The designer role defines the responsibilities, operations, attributes, and relationships of one or several classes, and determines how they are adjusted to the implementation environment. In addition, the designer role may have responsibility for one or more classes, including analysis, design, subsystems or testability.

- ◆ The designer must know use-case modeling techniques, system requirements, and software design techniques.



The designer must have a solid working knowledge of:

- Use-case modeling techniques.
- System requirements.
- Software design techniques, including object-oriented.
- Analysis and Design techniques and the Unified Modeling Language.
- Technologies with which the system will be implemented.
- Understand the architecture of the system. As represented in the software Architecture Document.

Analysis and Design is Use-Case Driven:

- Use cases defined for a system are the basis for the entire development process
- **Benefits of use cases:**
 - Concise, simple, and understandable by a wide range of stakeholders.
 - Help synchronize the content of different models.

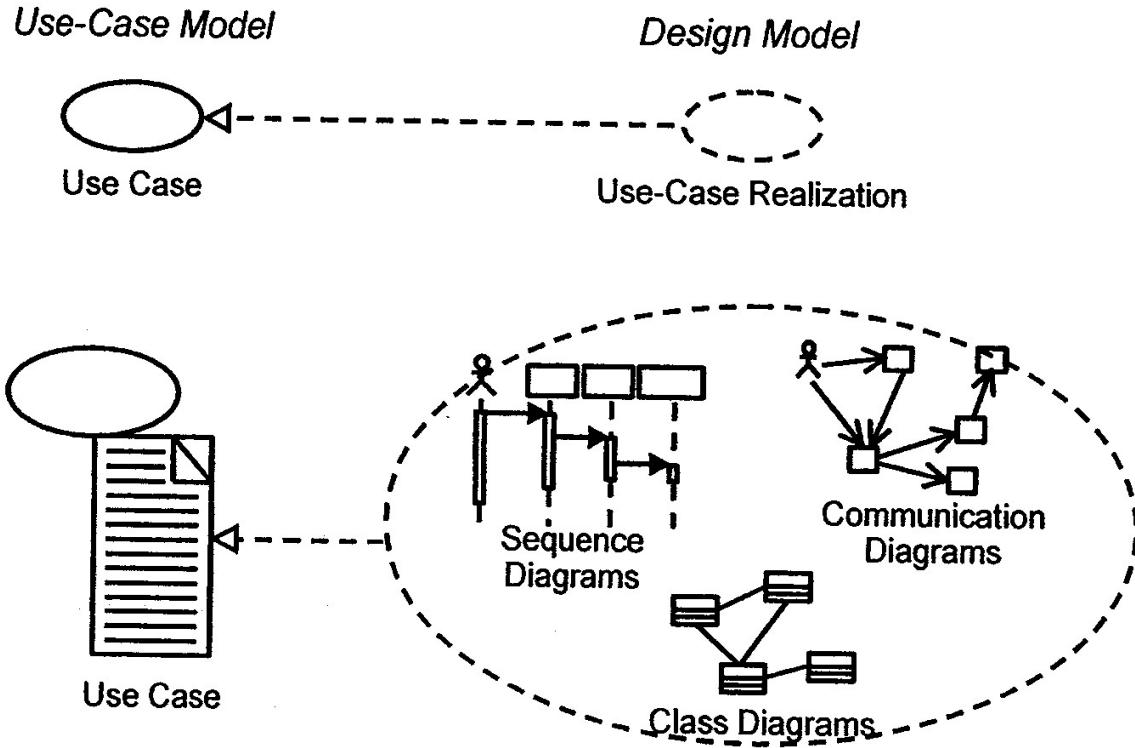
6.5 What Is a Use-Case realization?

- A use-case realization describes how a particular use case is realized within the design Model, in terms of collaboration objects.

A use-case realization ties together the use cases from the Use-Case Model with the classes and relationships of the Design Model. A use-case realization specifies what classes must be built to implement each use case.

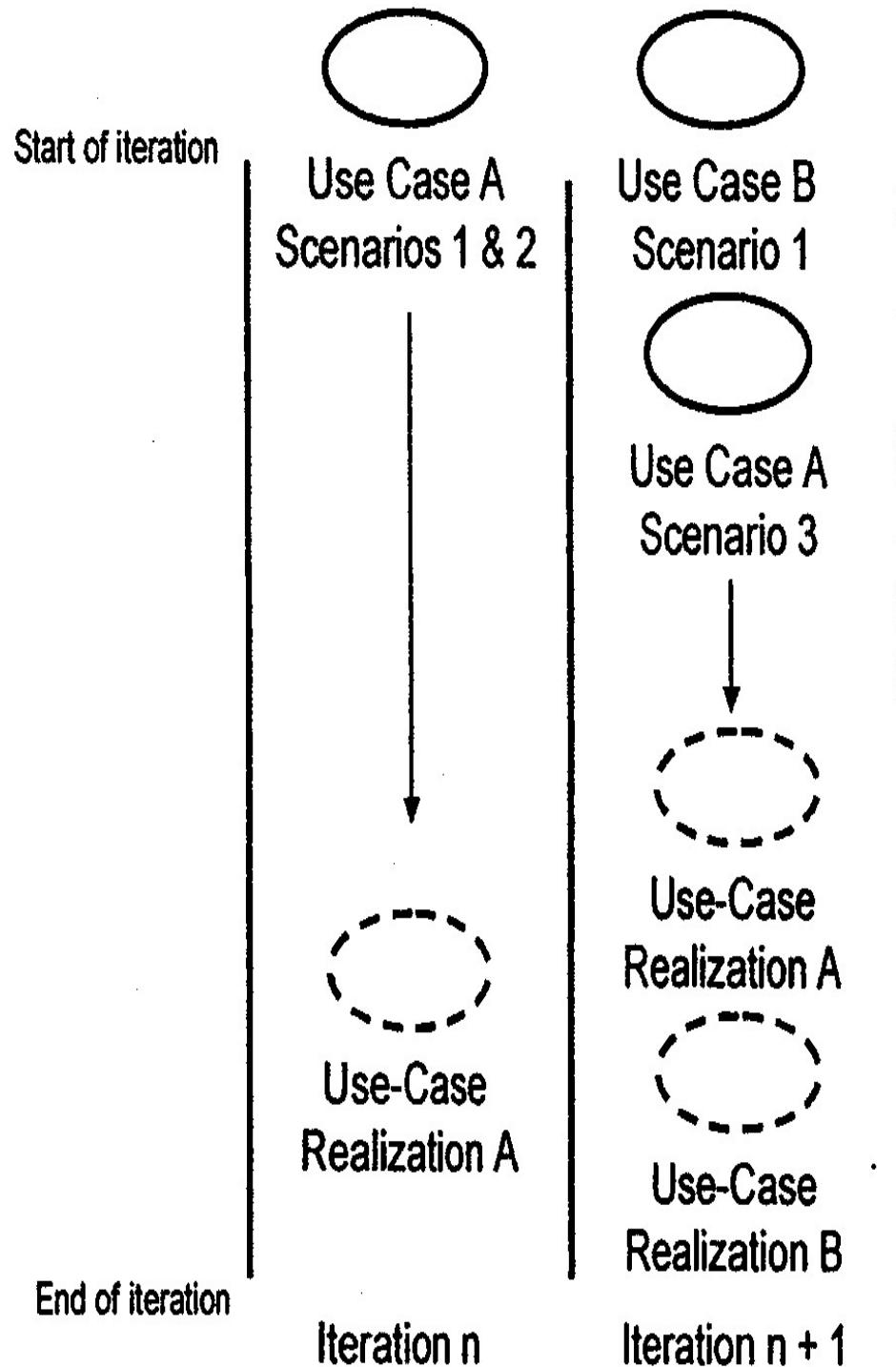
- In the UML, use-case realizations are stereotyped collaborations.

The symbol for collaborations is an ellipsis containing the name of the collaborations. The symbol for a use-case realization is a dotted line version of the collaboration symbol.



- The number and types of the diagrams that are used depend on what is needed to provide a complete picture of the collaboration and the guidelines developed for the project under development.

6.6 Analysis and Design in an Iterative process:



6.7 Functional Requirements:

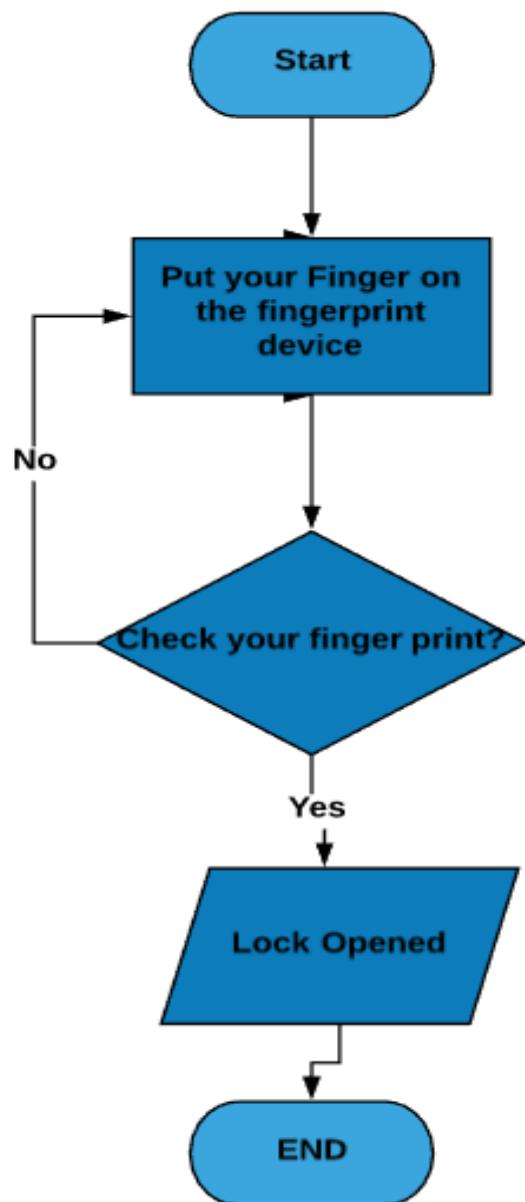
- Temperature sensor: must keep sensor active to get the measurement of the temperature of the room.
- Tsop sensor: should be active to get the remote signals from user.
- Infrared: must be available in the home to send signals to the air conditioner to make it [On/Off] or change the Ac Degrees.
- Finger Print Device: it must be available in the home to take the fingerprint from user and check if he/she the right person or not and that for security.
- Solenoid Lock: it must be in the home to [Close/Open] the door.
- Transistor: It must be active to Regulates voltage between circuit and the power source.
- EasyVR 3 Voice Recognition Shield: it is very important to get the user voice commands and make the Leds [On/Off].
- Easy VR Commander Software: the user to record his /her own voice commands uses it.

6.8 Nonfunctional Requirements:

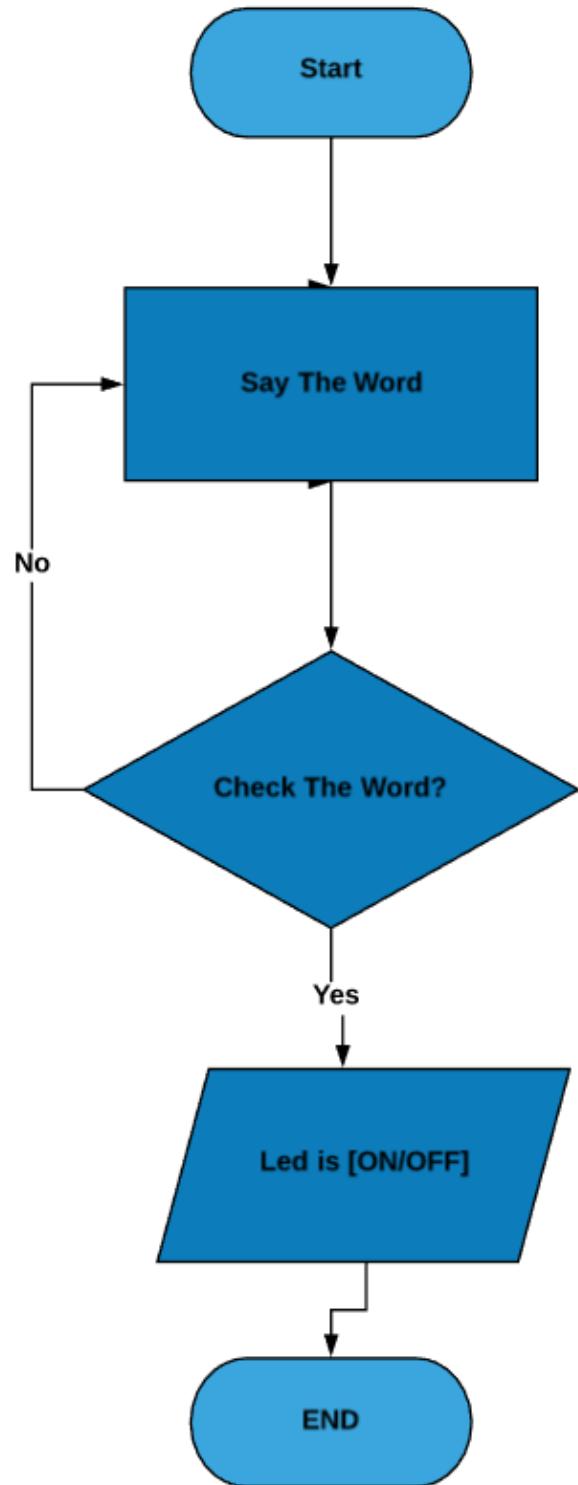
- LCD in Ac System because we LCD in our air conditioner.

6.9 Systems Flow Chart:

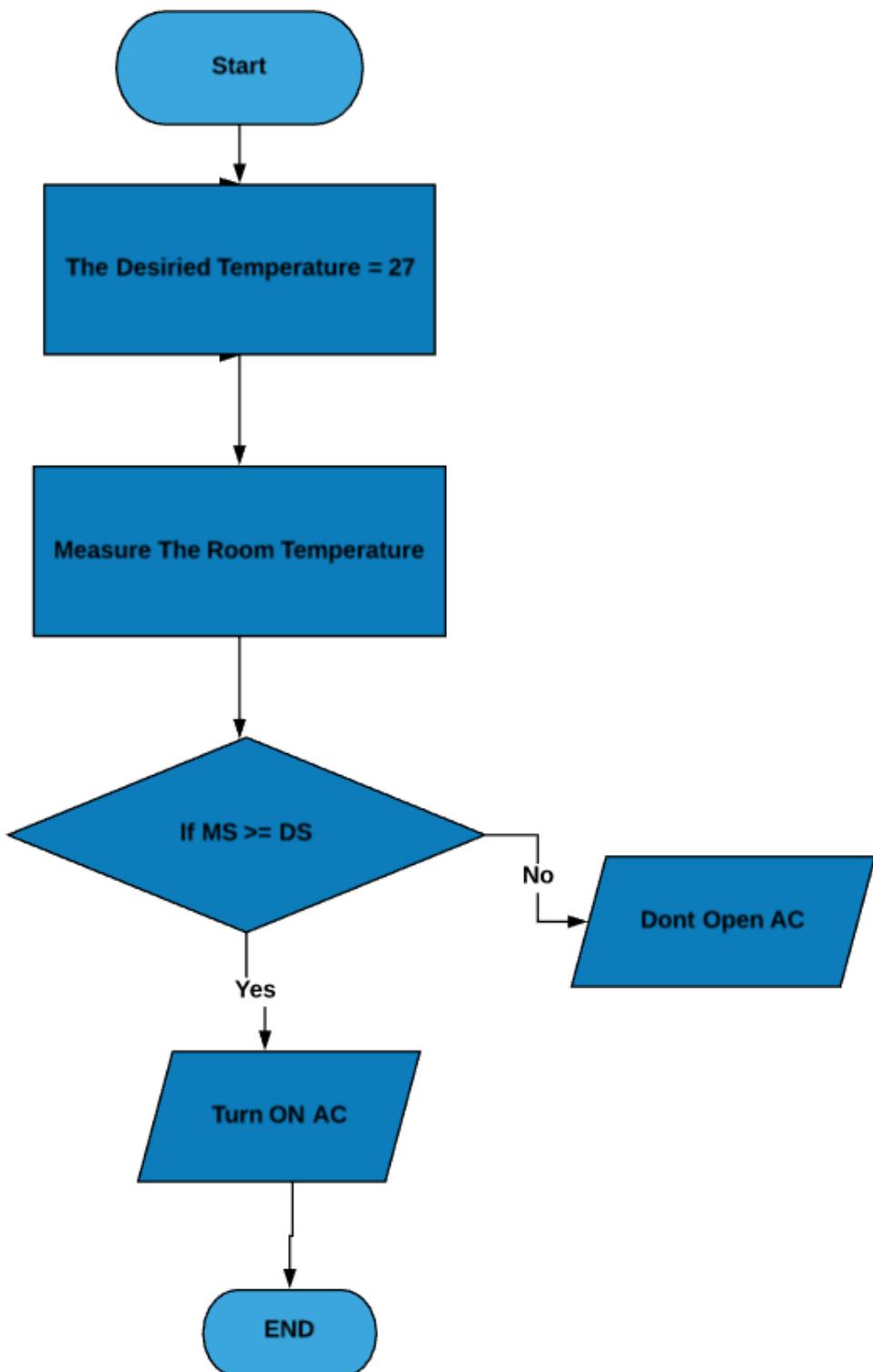
1. Door lock



2. Light



3. Air Conditioner



6.10 SWOT ANALYSIS:

SWOT analysis is a process that identifies an organization's strengths, weaknesses opportunities and threats. Specifically, SWOT is basic, analytical framework that assesses what an entity (usually a business, though it can be used for a place, industry or product) can and cannot do, for factors both internal (the strengths and weaknesses) as well as external (the potential opportunities and threats).



Strengths:

- High Security.
- Cost will be low compare to other smart home projects.
- Control lights easily with my voice.
- Regulate the temperature of the room without me.



Weaknesses:

- Sensors might break.
- Camera cannot detect your face in some conditions (low brightness, position of your face ...etc.).
- Fingerprint Device cannot detect your fingerprint in some conditions (position of your finger).
- In light system, if you say different word rather than the right word the lights will not [On/Off].



Opportunities:

- The highly rate of stolen homes can create our chance to sell our project.
- Low error percentage of our project make our project completely secure.

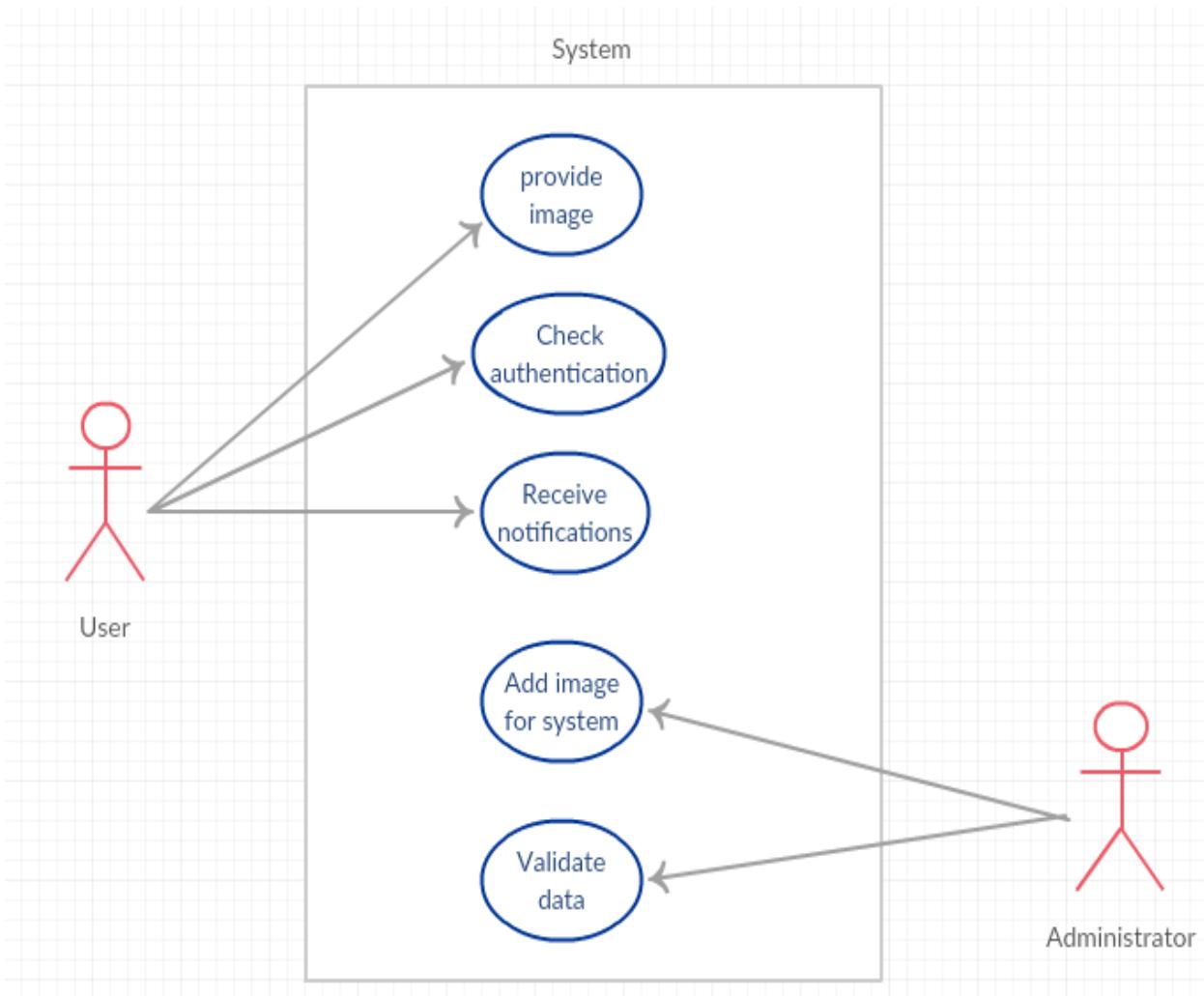


Threats:

- Camera can be stolen or broken.
- Sensors can damaged in any situation.
- In light system the quality of the mic. Can be low so you cannot turn [On/Off] your lights.



6.11 USE CASE FOR FACE RECOGNITION PROCESS



Primary Actor: enter my face photo

Goal in context: camera recognize my face

Precondition:

- the camera catches my face.
- camera recognize my face.

Scenario:

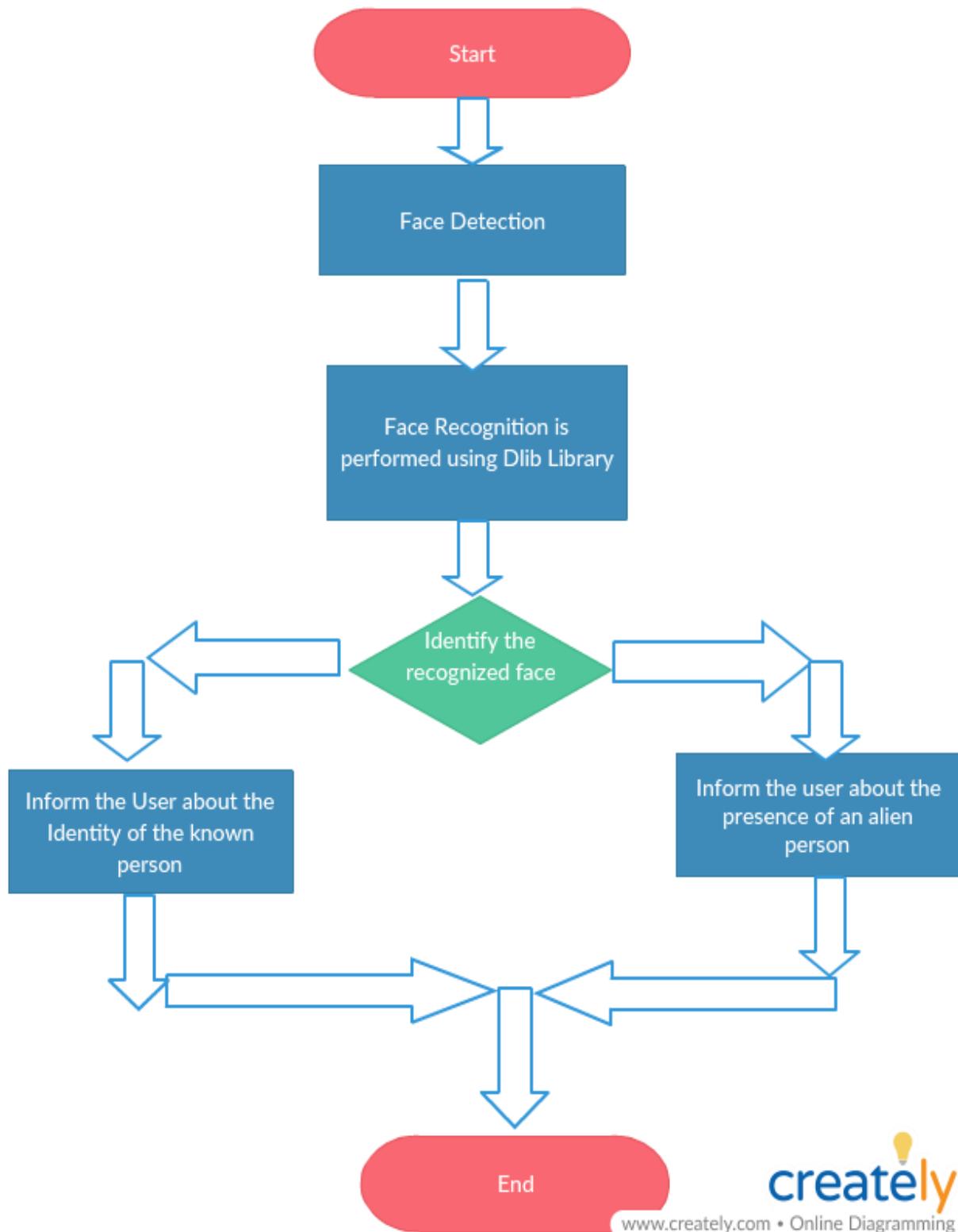
- User put his face front the camera.
- Camera catch his face photo.
- Camera recognize his face.

Exception: camera cannot catch face.

Priority: Essential, must be implemented

Available: First increment.

6.12 FLOW CHART FOR FACE RECOGNITION



{References}

Face recognition & raspberry with live stream:

- <https://www.superdatascience.com/opencv-face-recognition/>
- <http://www.instructables.com/id/Real-time-Face-Recognition-an-End-to-end-Project/>
- https://pythonhosted.org/face_recognition/
- https://pythonhosted.org/face_recognition/
- https://github.com/ageitgey/face_recognition/blob/master/README.md#installation
- <https://www.raspberrypi.org/documentation/usage/gpio/README.md>
- <https://business.tutsplus.com/tutorials/controlling-dc-motors-using-python-with-a-raspberry-pi--cms-20051>
- <https://maker.pro/raspberry-pi/tutorial/how-to-control-a-dc-motor-with-an-l298-controller-and-raspberry-pi>
- <https://elinux.org/RPi-Cam-Web-Interface>

IOT & Hardware:

- https://github.com/codergartenllc/tiny_gps_plus
- <https://github.com/mathworks/thingspeak-arduino>
- http://www.handsontec.com/pdf_learn/esp8266-V10.pdf
- http://www.handsontec.com/pdf_learn/esp8266-V10.pdf
- <http://silicontechnolabs.in/upload/IR%20Proximity%20Sensor%20data%20sheet.pdf>

Android:

- 1)<https://developer.android.com/>
- 2)[Udemy Complete Java Masterclass](#)
- 3)[Udemy - Android Java Masterclass - Become an App Developer](#)
- 4)<https://codinginflow.com/>
- 5)<https://square.github.io/retrofit/>



**SMART
HOME**