Benha University

Benha Faculty of Engineering

Electric Engineering Department

An IoT-Based Smart Home Automation System

A project graduation report submitted in partial fulfilment for the award of BSc. Degree in Electrical Engineering, Benha Faculty of Engineering, Benha University

(Communication and Computer Engineering)

Presented by:

1. Ahmed Mohamed Abd elnabi Mohamed
2. Ahmed Medhat Hassan
3. Ahmed Younis Ibrahim
4. Asmaa Gamal Abd Elrahman

Supervised by:

Dr. Hossam Labib Zayed

Benha - Egypt

July 2022

ACKNOWLEDGEMENT

First, we would like to thank Allah for helping us to accomplish this project. We would like to thank our project supervisor, Dr. Hossam Labib Abdel Fattah Zayed for the guidance, support, and invaluable assistance that he gave us during this project. His comments have been of inestimable value throughout our work. Also, we would like to extend our thanks to all faculty members in the electrical engineering department at Benha University for their effort, help, and support during the period we spent studying in the college of engineering.

Finally, we would like to express our sincere thanks to our parents for their patience, encouragement, and countless grants gratis.

Table of Contents

List Of Tables

List of Figures

List of abbreviations

HTTP	Hypertext Transfer Protocol
UART	Universal asynchronous receiver-transmitter
GSM	Global System for Mobile Communications
IOT	Internet of Things
LTE	Long-Term Evolution
5G	5G is the fifth-generation technology standard for broadband cellular networks
MQTT	MQ Telemetry Transport

Abstract

Internet of Things (IoT) is an emerging technology that is making our world smarter. The idea of connected world cannot be imagined without IoT. IoT is a system that uses computers or mobile devices to control basic home functions and features automatically through internet from anywhere around the world, an automated home is sometimes called a smart home. It is meant to save the electric power and human energy and safety. The home automation system differs from other system by allowing the user to operate the system from anywhere around the world through internet connection. This project aims to develop a voice and gesture controlled smart home using a WI-FI and IOT, which is being remotely controlled by any Android OS smart phone or home microphones or cameras. The home automation becomes vital, as it gives the user the comfort and a trouble free system for using home devices, which made life better and easier and more safe.

1 Introduction

Home automation has achieved a lot of popularity in recent years, as day-to-day life is getting simpler due to the rapid growth of technology. Almost everything has become digitalized and automatic. a system for interconnecting sensors, actuators, and other data sources with the purpose of multiple home automations is proposed. The system works by leveraging the power of a flexible and powerful Application Programming Interface (API), which represents the foundation of a simple and common communication scheme. The devices are usually sensors or actuators with an upstream network connection implementing the API. Most devices are based on ESP8266 chips and on Raspberry Pi boards. A smartphone application has been developed that allows users to control a series of home appliances and sensors. The smart home system is user friendly, flexible, and can be further developed by using different devices and add-ons.

1.1 Iot value chain

An IoT solution is formed of several building blocks or components, and each of these building blocks forms part of the IoT value chain.1 The IoT value chain illustrates how the different components, in combination with one another or separately, add value to the overall IoT solution and, in turn, for the end user. Furthermore, each component is developed by a range of companies, some of which play several roles in the IoT value chain. The following components form part of the IoT value chain[1].

1.2 Devices.

This category includes existing devices such as smart meters or vehicles in which the connectivity component has been integrated into the product design. This could also include new devices that would not have existed without IoT, such as pet trackers. Such a device must have a sensor and an actuator, as well as communications hardware (described in more detail below), but it will also have other elements (for example, a power source such as a battery or mainselectricity). In addition, depending on the type of device, it may have a screen and other ways for the user to interact with it directly.

1.2.1 Sensor and actuators

are connected to the device. Sensors are able to capture data from the environment (for example, temperature). Actuators respond to instructions and make changes in the device (for example, adjusting the temperature on a thermostat). The instructions for an actuator can come from sensors on the same device, or from other sources (for example, a thermostat can be activated by mobile phone while the homeowner is on their way home). A device can have sensors, actuators, or both.

1.2.2 Communications hardware

enables the device to connect to the network to send the data from the sensors to the backend systems. This can include hardware for connecting wirelessly via BlueTooth, Wi-Fi, ZigBee, LoRa, cellular (for example GSM, 5G, NBIoT, LTE-M) or a number of proprietary technologies, or over a fixed network. Some devices will have hardware to connect to multiple types of network[1].

1.3 The connectivity network

which can be cellular, fixed or satellite, delivers the data from the sensors over the internet or a private network connection to the user's backend systems.

1.4 Backend systems

include the servers to collect and analyse the data coming from the sensors and from other sources (for example, weather forecast data). These backend systems can be found in the public or private cloud, or on on-premises hardware. For very simple systems, the backend can be a standard PC.

1.5 Software platforms

such as device management, security and data analytics ensure that IoT devices are functioning correctly and have not been compromised. Such platforms also include data analytics software to make sense of the data and improve business processes, as well as data bases to store the data

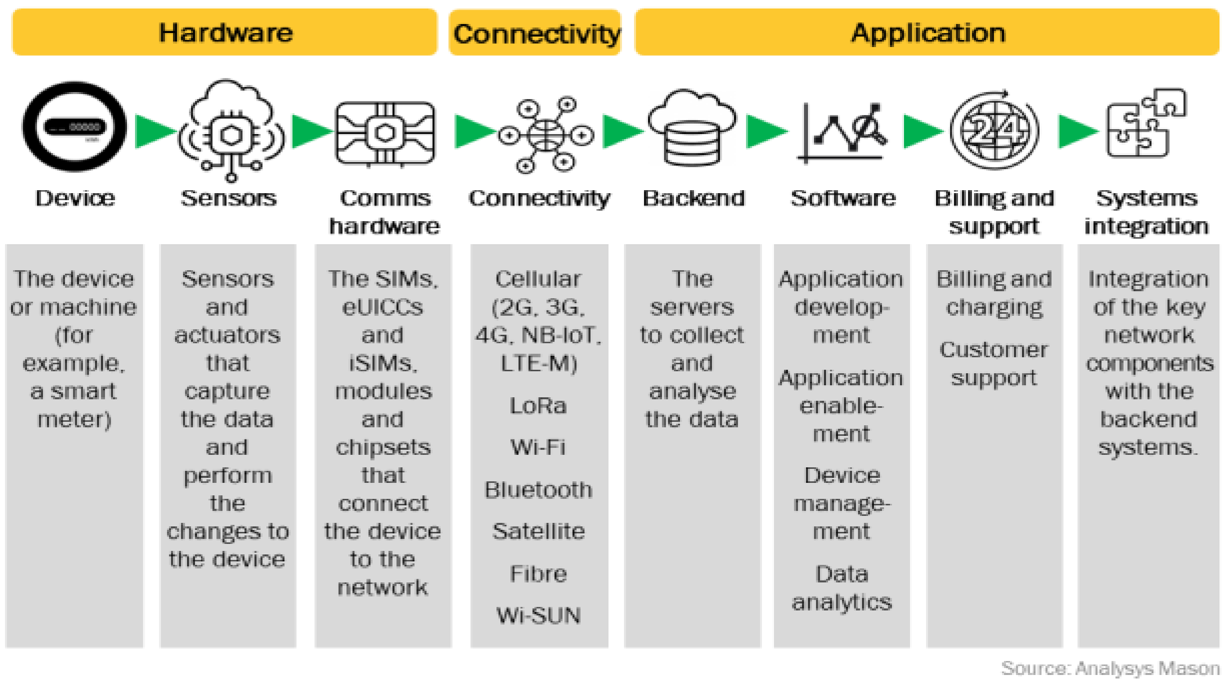

Hardware
Connectivity
Application
Device
Sensors
Comms hardware
Connectivity
Backend
Software
Billing and support
Systems integration
The device or machine (for example, a smart meter)
Sensors and actuators that capture the data and perform the changes to the device
The SIMs, eUICCs and iSIMs, modules and chipsets that connect the device to the network
Cellular (2G, 3G, 4G, NB-IoT, LTE-M)
LoRa
Wi-Fi
Bluetooth
Satellite
Fibre
Wi-SUN
The servers to collect and analyse the data
Application develop-ment
Application enable-ment
Device manage-ment
Data analytics
Billing and charging
Customer support
Integration of the key network components with the backend systems.
Source: Analysys Mason

Figure 1 Software platforms

1.6 What type of connectivity is best for IoT

Enterprises can choose from a range of IoT connectivity options to support their IoT deployment. Their choice will depend on use case requirements and will also be governed by factors such as cost, ecosystem support and coverage requirements (local, national or global). Some examples include[1]:

1.6.1 mobile assets

which can be supported by Wi-Fi or cellular in a building (for example, a factory). If they require wide area coverage, they will use cellular networks, or possibly even satellite in remote places.

1.6.2 static assets

which can be supported by fixed connectivity, Wi-Fi or cellular. For example, connected intruder alarms in a residential building may use a fixed line connection (such as PSTN), a broadband/Wi-Fi connection or a cellular connection. However, static assets that are geographically dispersed (such as smart meters) may use the cellular network or alternative technologies such as Wi-SUN

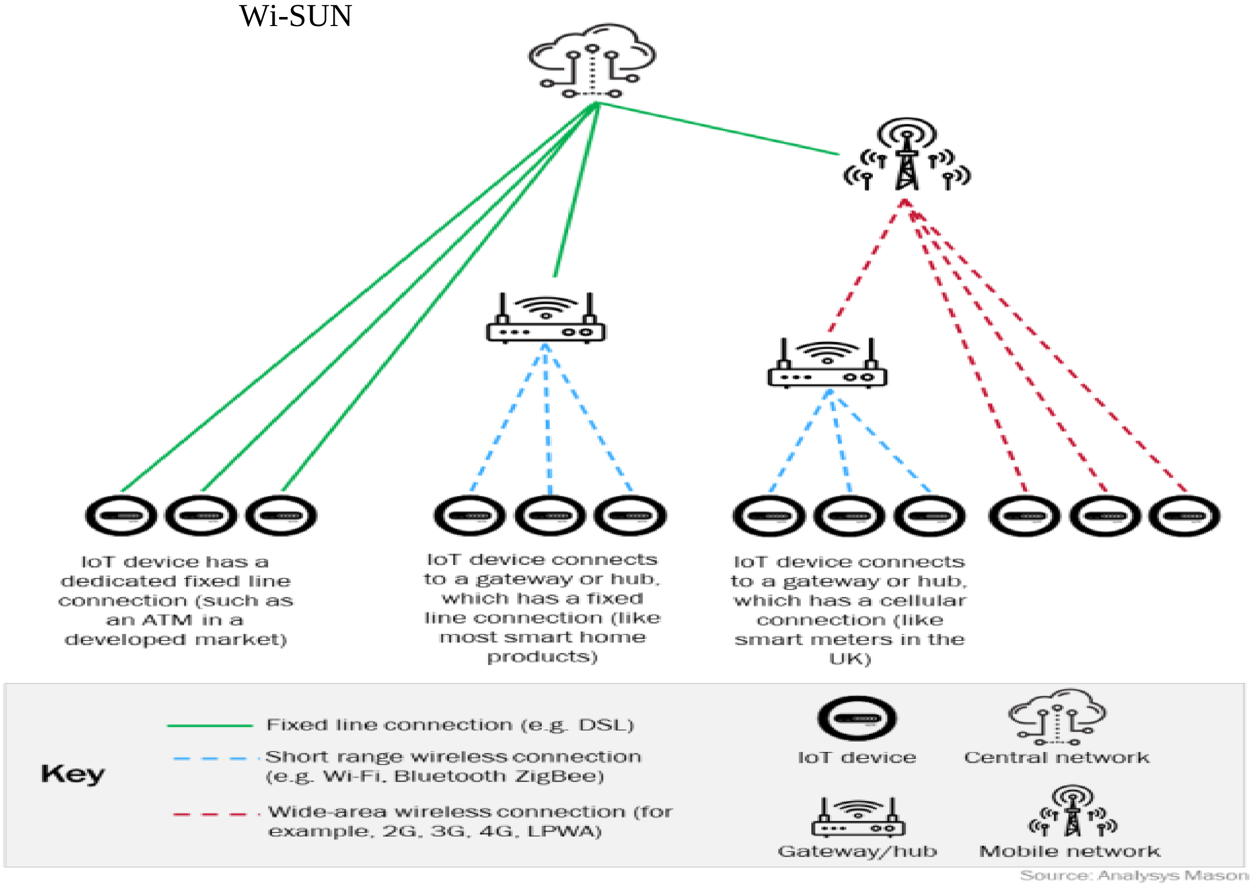

IoT device has a dedicated fixed line connection (such as an ATM in a developed market)
IoT device connects to a gateway or hub, which has a fixed line connection (like most smart home products)
IoT device connects to a gateway or hub, which has a cellular connection (like smart meters in the UK)
Key
Fixed line connection (e.g. DSL)
Short range wireless connection (e.g. Wi-Fi, Bluetooth ZigBee)
Wide-area wireless connection (for example, 2G, 3G, 4G, LPWA)
IoT device
Central network
Gateway/hub
Mobile network
Source: Analysys Mason

Figure 2 static assets

2 Home automation (previous work)

In the last years, the IoT concept has had a strong evolution, being currently used in various domains such as smart homes, telemedicine, industrial environments, etc. Wireless sensor network technologies integrated into the IoT enable a global interconnection of smart devices with advanced functionalities. A wireless home automation network, composed of sensors and actuators that share resources and are interconnected to each other, is the key technology to making intelligent homes. A “smart home” is a part of the IoT paradigm and aims to integrate home automation. Allowing objects and devices in a home to be connected to the Internet enables users to remotely monitor and control them . These include light switches that can be turned on and off by using a smartphone or by voice command, thermostats that will adjust the indoor temperatures and generate reports about energy usage, or smart irrigation systems that will start at a specific time of a day, on a custom monthly schedule, and thus will control water waste. Smart home solutions have become very popular in the last years. Figure 1 shows an example of a smart home that uses different IoT-connected utilities. One of the greatest advantages of home automation systems is their easy management and control using different devices, including smartphones, laptops and desktops, tablets, smart watches, or voice assistants. Home automation systems offer a series of benefits; they add safety through appliance and lighting control, secure the home through automated door locks, increase awareness through security cameras, increase convenience through temperature adjustment, save precious time, give control, and save money[2].

Several home automation systems involved with IoT have been proposed by academic researchers in the literature in the last decade. In wireless-based home automation systems, different technologies have been used, each of them with their pros and cons. For example, Bluetooth-based automation is low cost, fast, and easy to be installed, but it is limited to short distances. GSM and ZigBee are widely used wireless technologies as well. GSM provides long-range communication at the cost of a mobile plan of the service provider that operates in the area. ZigBee is a wireless mesh network standard that is designed to be low-cost and with low power consumption, targeted at battery-powered devices in wireless control and monitoring applications. However, it has a low data speed, low transmission, as well as low network stability, and has a high maintenance cost. Wi-Fi technology is used in . The advantages of Wi-Fi technology over ZigBee or Z-Wave are related to price, complexity (meaning simplicity), and accessibility. First, Wi-Fi-enabled smart devices are usually cheap. In addition, it is easier to find do-it-yourself devices that use Wi-Fi, resulting a less expensive option. Second, Wi-Fi is already a necessity and it is in most homes, so it is easier to buy devices that are already Wi-Fi-enabled. Finally, Wi-Fi is characterized by simplicity, meaning that a user must connect only a minimal number of devices for a home automation setup. Since it is very common, the investment on extra hardware is avoided; a user only needs the basic setup for a home automation system. However, Wi-Fi is not designed to create mesh networks, it consumes ten times more energy than similar devices using ZigBee, Z-Wave, or Bluetooth for example, and many Wi-Fi routers can only allow up to thirty devices connected at once. As compared to Ethernet, Wi-Fi brings several advantages, including the easy connection and access of multiple devices, the expandability (adding new devices without the hassle of additional wiring), lower cost, or single access point requirement. The cons include limited distance to cover (a Wi-Fi network with standard equipment can be limited in range through walls and other obstructions in a standard home), the number of devices can be limited, there is interference and complex propagation effects, obstacles can block the Wi-Fi signal and affect the devices connected to it, and there are connection speed (the fastest speed of Wi-Fi is much slower than a wired network), Internet security, and privacy issues. Low-cost, open source hardware components, such as Arduino and Raspberry Pi microcontroller unit (MCU) boards, and a combination of sensors have been very used in the home automation domain. Home automations using Arduino boards are proposed in . Arduino is highly flexible, open source, not expensive, and easy to program . In addition, the existence of a large and active community of users is a great plus. However, Arduino is not designed to handle the large complexity that comes with advanced projects. For more advanced and real-time projects, Raspberry Pi is a better option. Raspberry Pi is an exciting technological development that is much cheaper than any desktop computer or mobile device[3] . Most of the software and projects done on Raspberry Pi are open source and are maintained by online user communities, which are always excited about new projects. When developing software on Raspberry Pi, Python is the language of choice, since it is relatively simple (fewer lines and less complexity) compared to other programming languages. In addition to its low price, Raspberry Pi is energy efficient and does not require any cooling systems. Smart home automations with Raspberry Pi . ESP8266 chips are low-price Wi-Fi modules that are perfectly suited for projects in the IoT field. ESP8266 is a single core processor that runs at 80 MHz ESP8266 chips were used for home automations-related projects[2] .

Table 1 : A features comparison for home automation system published in scientific papers, in the last ten years, is presented in

Home Automation System	Communication	Controller	User Interface	Applications
	Bluetooth	PIC	mobile app	control indoor appliances
	Bluetooth	Arduino	mobile app	control appliances indoor and outdoor, within short range
	Bluetooth, GSM	PIC	mobile app	control appliances

			indoor and outdoor
ZigBee, Ethernet	Arduino MEGA	mobile app	control appliances indoor
X10, Serial, EIB, ZigBee, Bluetooth,	32-bit ARM microcontroller	Control panel (touch pad), desktop based	indoor automation solution
Wi-Fi, ZigBee	Raspberry PI, NodeMCU		controlling humidity, temperature, luminosity, movement, and current
ZigBee	Laptop/PC server	mobile app	control of indoor appliances but not actually implemented
ZigBee, Wi-Fi	Linux board	GUI interface	control HVAC appliances
ZigBee, Wi-Fi, Ethernet	Raspberry PI	web-based, mobile app	remote control of appliances (IP cams, smart plugs)
Wi-Fi	TI-CC3200 MCU	mobile app	control indoor appliances, monitor the soil moisture
Wi-Fi	NodeMCU	web-based	control indoor appliances
Bluetooth, Wi-Fi	Raspberry PI	mobile app	control indoor appliances
Wi-Fi	Arduino mega	web-based, mobile app	control of indoor appliances
Wi-Fi	PC server	mobile app	security, energy management
Wi-Fi, IR	PC server	mobile app	control of indoor appliances
Wi-Fi	Arduino	mobile app	control indoor appliances, video

			surveillance
Bluetooth	Arduino	mobile app	control indoor appliances, energy management
Wi-Fi	Arduino, ESP8266	web-based, mobile app	control indoor appliances
Bluetooth, Wi-Fi	Arduino mega	web-based	indoor and outdoor control, monitoring, energy management, safety, security
Ethernet	Arduino mega	web-based	control of indoor appliances
Ethernet	Raspberry PI	unspecified	control home appliances, surveillance
ZigBee, Z-wave, Wi-Fi	Raspberry PI	web-based, mobile app	light automation and physical intrusion detection
Wi-Fi	NodeMCU	unspecified	control indoor appliances (luminosity sensor, LED, buzzer)
Wi-Fi	ESP8266	mobile app	testing modules in a smart home system, related to indoor appliances control, surveillance, energy management
Wi-Fi	Arduino, ESP8266	web-based, mobile app	control of switches
Wi-Fi	Node MCU	web-based, mobile app	control of appliances indoor and outdoor, safety, security, energy management,

			monitoring
Ethernet	Galileo board	web-based	indoor and outdoor control, energy management, security
GSM, Wi-Fi	PC serve	web-based	safety, monitoring (gas, temperature, fire sensors)
GSM	8051 MCU	web-based	indoor and outdoor control
GSM	Arduino	LabVIEW PDA Module	control of indoor appliances, safety, energy management
ZigBee, Wi-Fi, GSM/GPR	PC	web app	remote monitoring and control system for intelligent buildings
ZigBee	PC		power outlet control
Wi-Fi	Raspberry PI, ESP 8266	web-based, mobile app	multiple home automations indoor and outdoor, irrigations, security, monitoring, power and energy management (including solar energy), Google assistant compatible

Another category of home automation systems is represented by commercial platforms, such as Qivicon, Domintell, Loxone, or HomeSeer. They offer a wide range of smart home devices, from multiple vendors, different communications protocols for wired (Domintell) and wireless (Qivicon) transmissions, or both (HomeSeer and Loxone), and multiple automations such as a locking system, controlling temperature, lightning system, environmental system, video surveillance (only Qivicon and HomeSeer), or anti-intrusion. All solutions provide a mobile app for controlling the systems.

Pricewise, it depends on the size of the house, the number of devices to be installed, and the needs of the user, the minimum cost comes between 1800 and 2600 euros. Currently, a great variety of open source home automation systems exist.

OpenHAB and Home Assistant are two of the strongest players in the open source home automation community, sharing a similar vision and integrating many devices. However, openHAB requires knowledge regarding how to insert commands to integrate devices; it is complex and time consuming. Home Assistant, on the other hand, is more user friendly, but it requires a significant configuration effort. Mobile apps seem less flexible and quite complicated and complex, especially for beginners. Domoticz delivers a decent number of features; its configuration is mostly done through the web interface, and plugins are used to extend its functionality. Unfortunately, the interface itself is not extremely intuitive. Domoticz is quite limited in terms of supported devices and configurations.

Calaos and Jeedom are two French players in the open source home automation community. Unfortunately, the communities and forums are predominantly French, which can be a barrier to worldwide adoption. A feature comparison of the most relevant open-source home automation platforms is presented in Table 2. The platforms can be differentiated, among others, in terms of the development language, the API, the amount

of implemented protocols and plugins, and the amount and type of documentations. Of course, these are not the only options available. The authors of present a detailed comparison of fifteen open-source platforms.

Table 2: Comparison of the most relevant open-source home automation platforms[2].

System	Development Language	API	Other Features
OpenHAB	Java	Representational state transfer (REST)	web interface, many protocols, many plugins, MQTT, EPL v1

			license, extensive documentation
HomeAssistant	Python	REST/Python/Web socket APIs	web interface, many protocols, many plugins, MQTT, Apache 2.0 license, extensive documentation
Domoticz	C++	JSON based	web interface, many protocols, many plugins, MQTT, GPL v3 license, extensive documentation
Calaos	C++	JSON based	web interface, a few protocols, under development plugins, MQTT, GPL v3 license, extensive documentation (in French)
Jeedom	PHP	JSON RPC and HTTP based	web interface, many protocols, many plugins, MQTT, GPL v2 license, extensive documentation (in French)
Fhem	Perl	ASCCII commands	web interface, many protocols, many plugins, MQTT, GPL v2 license, extensive documentation (in German)
qToogle	Python	JSON based REST	web interface, many protocols, a few plugins (undergoing continuous development), Hypertext Transfer Protocol (HTTP) based messaging, Apache 2.0 license, extensive documentation

3 System Architectural

the model of smart home system based on Raspberry PI, esp8266 and Android device. The system is designed to be scalable and easy to setup and extend. It is based on powerful Raspberry PI microcomputer. It includes sensors for listening of the environment and appliances that are controlled via Android device.

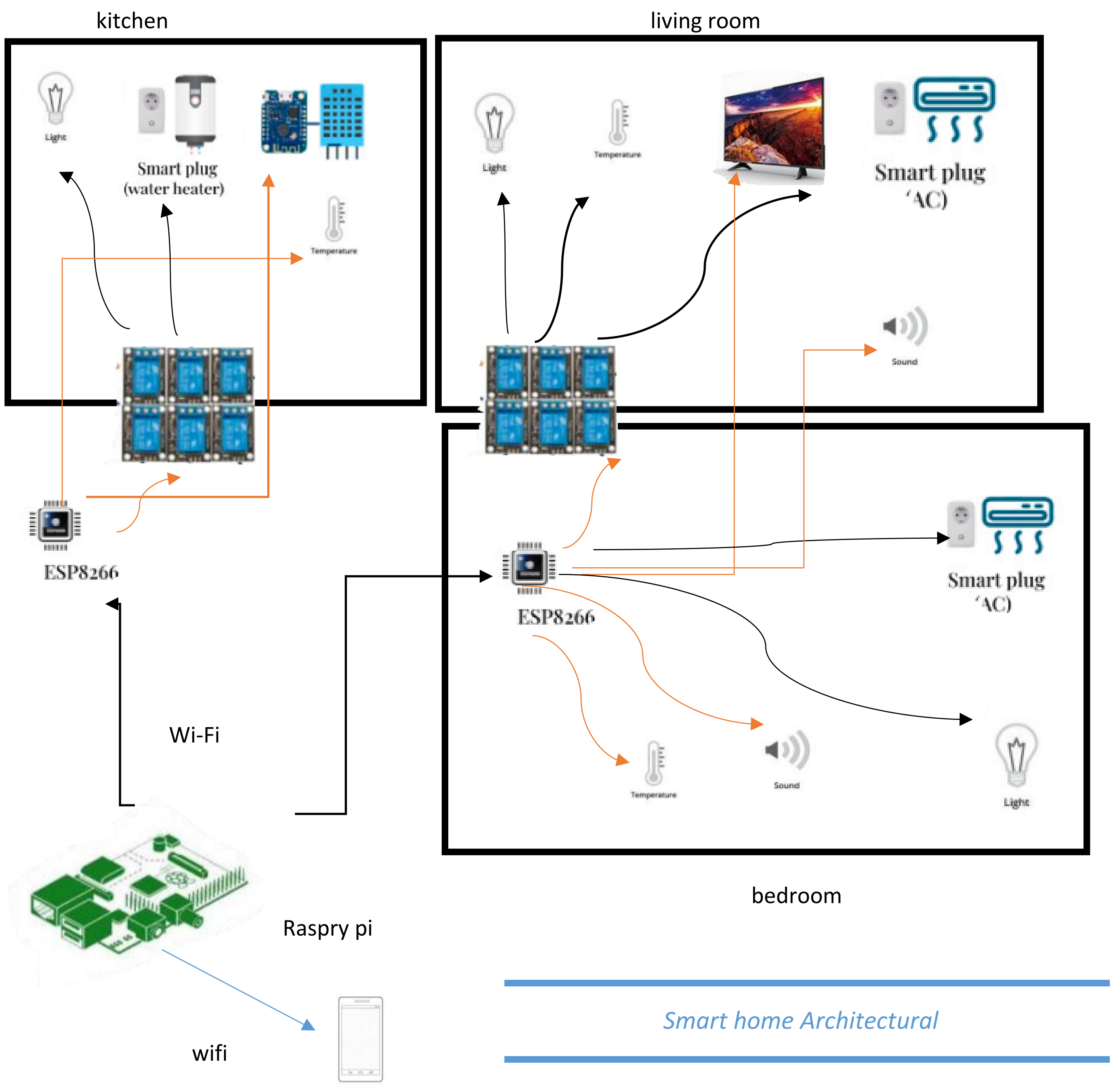


Figure 3 Smart Home Architectural

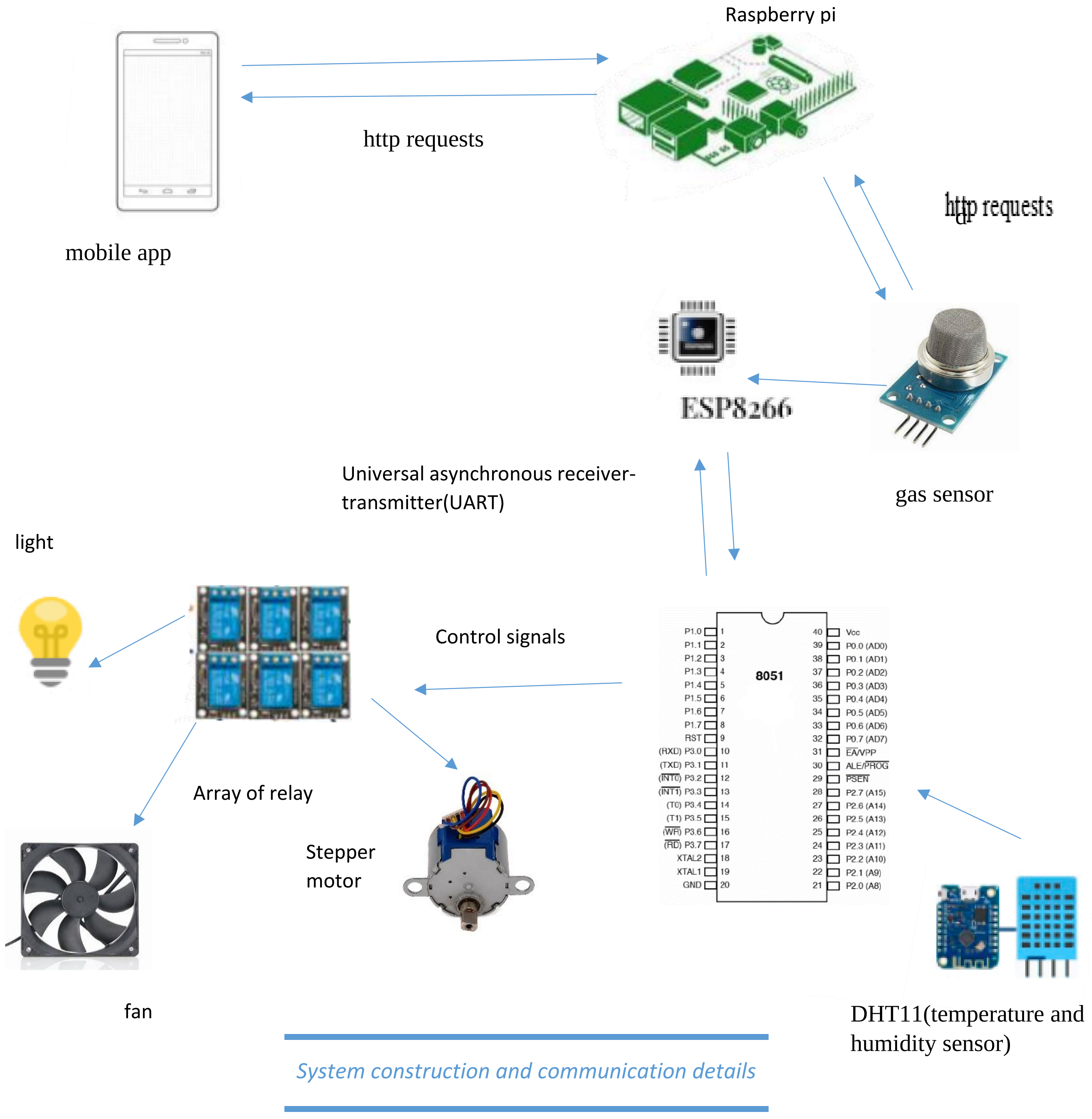

Raspberry pi
http requests
mobile app
http requests
ESP8266
gas sensor
Universal asynchronous receiver-transmitter(UART)
light
Control signals
8051
Array of relay
Stepper motor
fan
DHT11(temperature and humidity sensor)
System construction and communication details

Figure 4 System Construction And Communication Details

3.1 System communication steps:

Raspberry bi and two nodes made its server

1. raspberry save in its server devises states and sensor read.
2. mobile application send GET request to read devises states and sensor read.
3. mobile can change devices state by sending get request that has new state.
4. raspberry send new state to node and node send it to 8051 Microcontroller to make the action.
5. node has sensor send every 3 seconds new read to raspberry.
6. app send request to raspberry to read the new sensor read.

3.2 Implementation

We build a model of our smart home (livingroom-bedroom-kitchen-bathroom-door open with stepper motor-fan-some sensors).

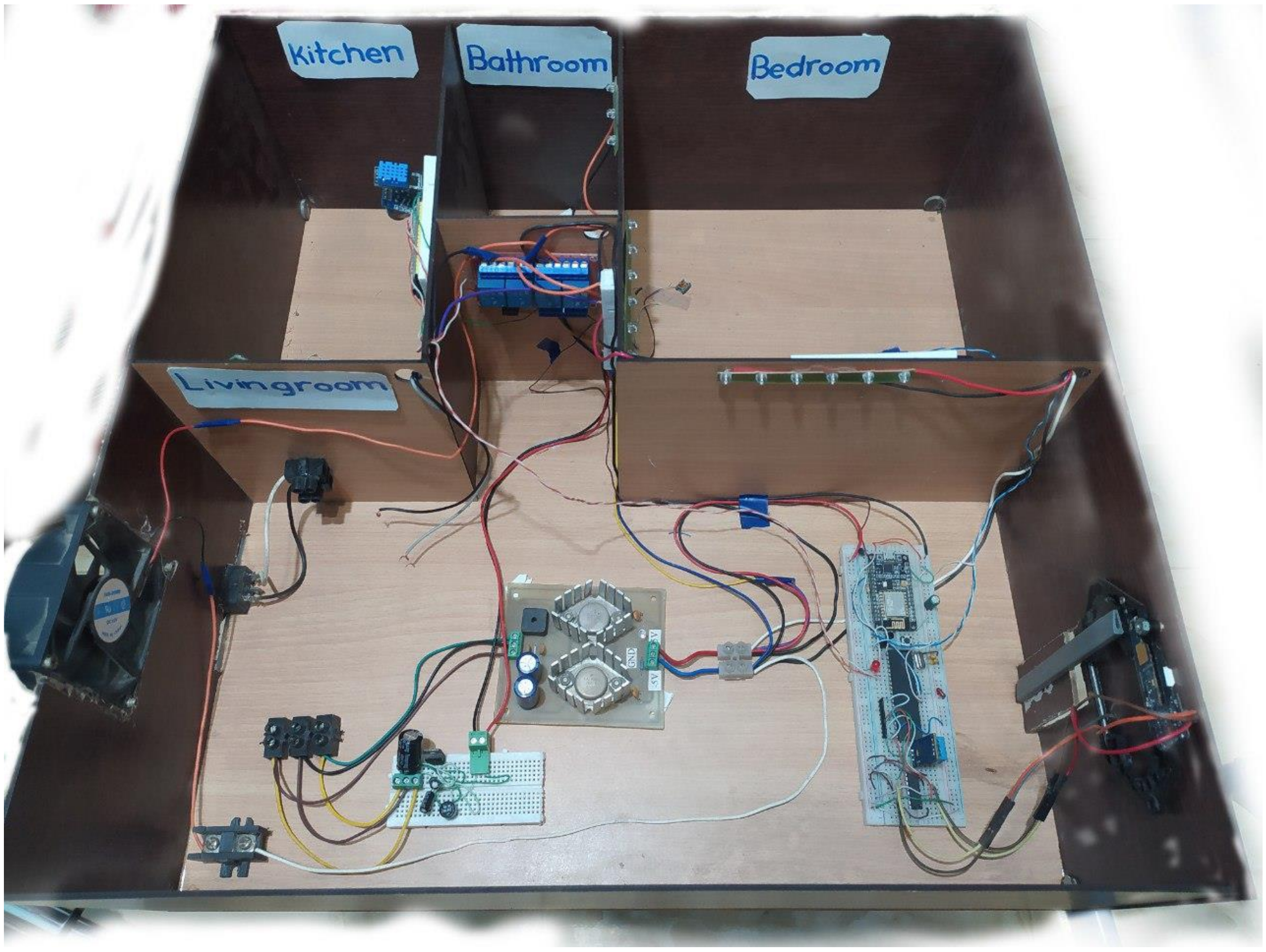


Figure 5 Model Of Our Smart Home

we can control home by three methods with mobile application and voice and with hand signs, Application also can show me sensors reads and can automatically control fan with temperature sensor read.

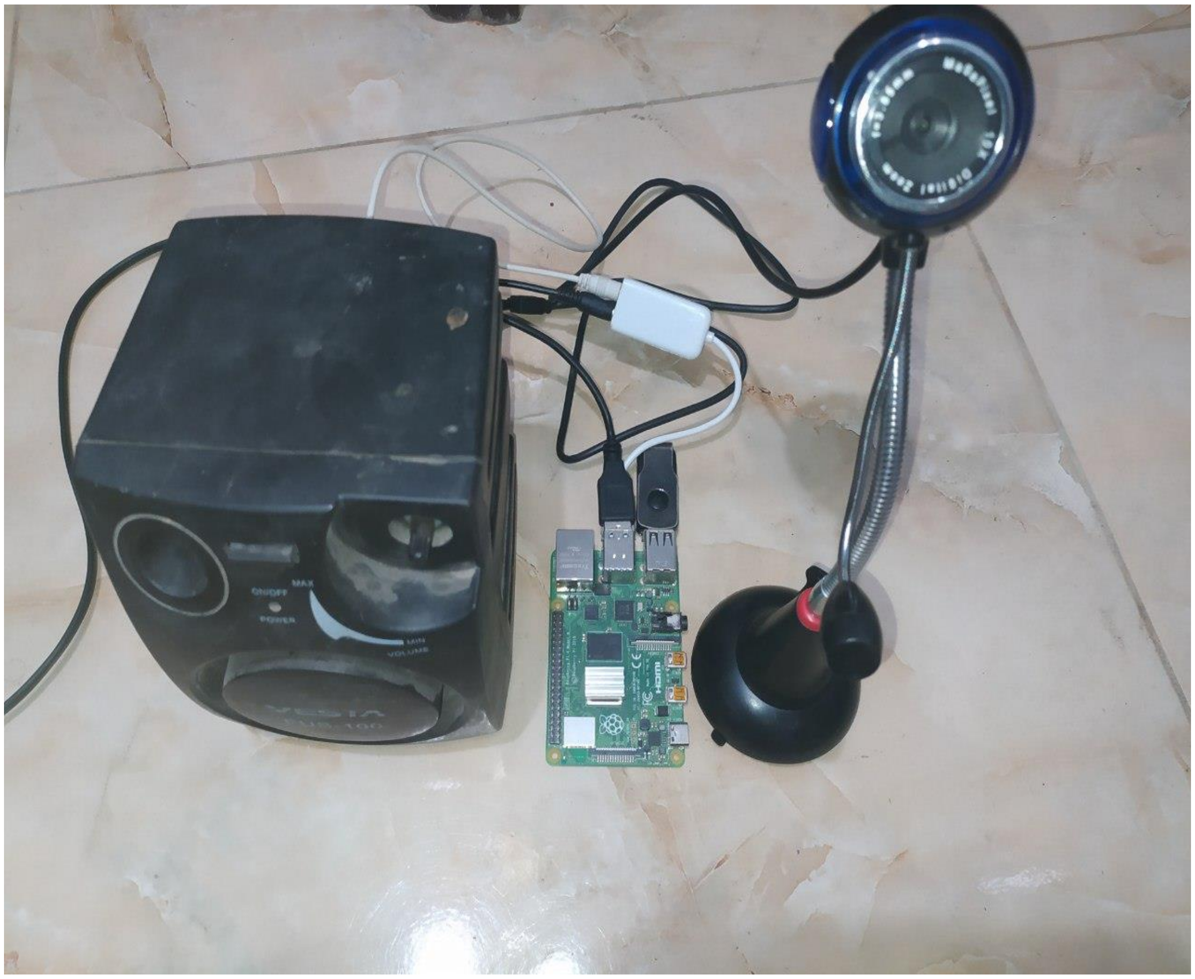


Figure 6 Voice And Face Recognition

This camera and microphone used to make voice recognition and face recognition and also gesture recognition which all used to control home, these various control ways to comfort human life and save time and power.

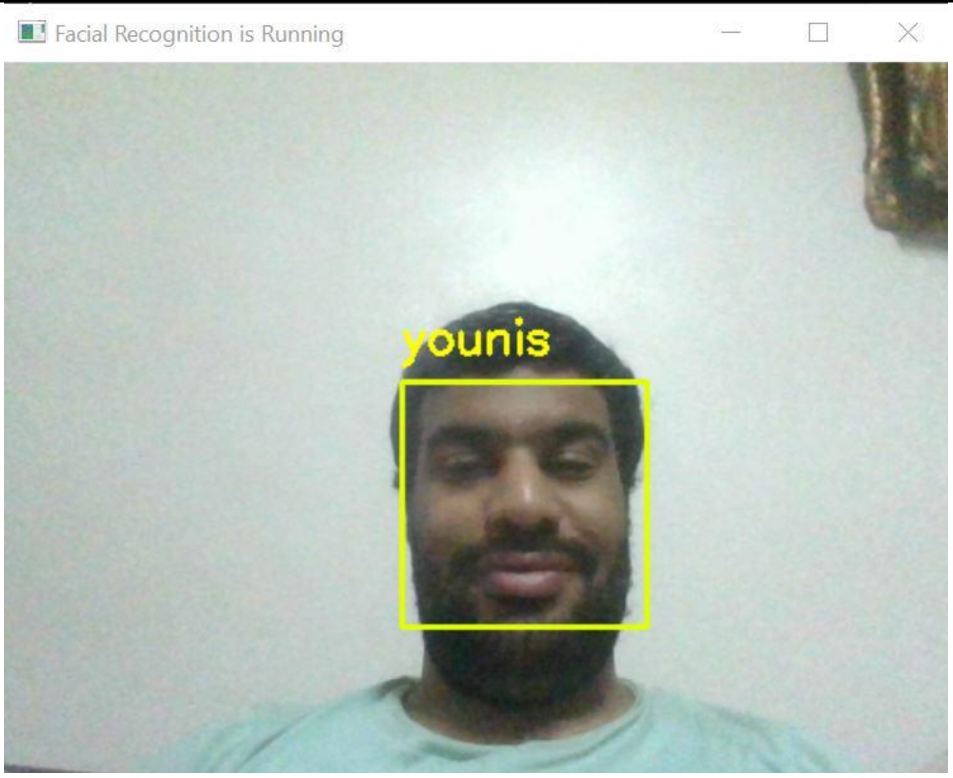


Figure 7 Camera Can Known User Like Younis

In this image our camera can know user or home owner and open door to him.

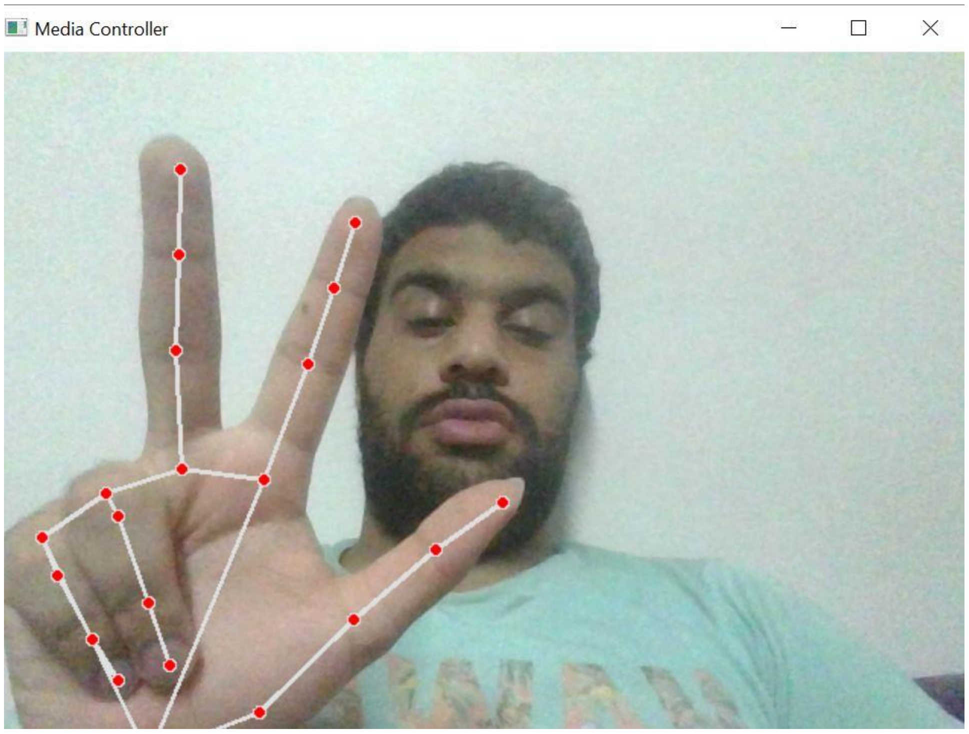


Figure 8 Hand Signs Using Camera

Some times we con not use application or our voice so we add feature to control devices with our hand signs using cameras.

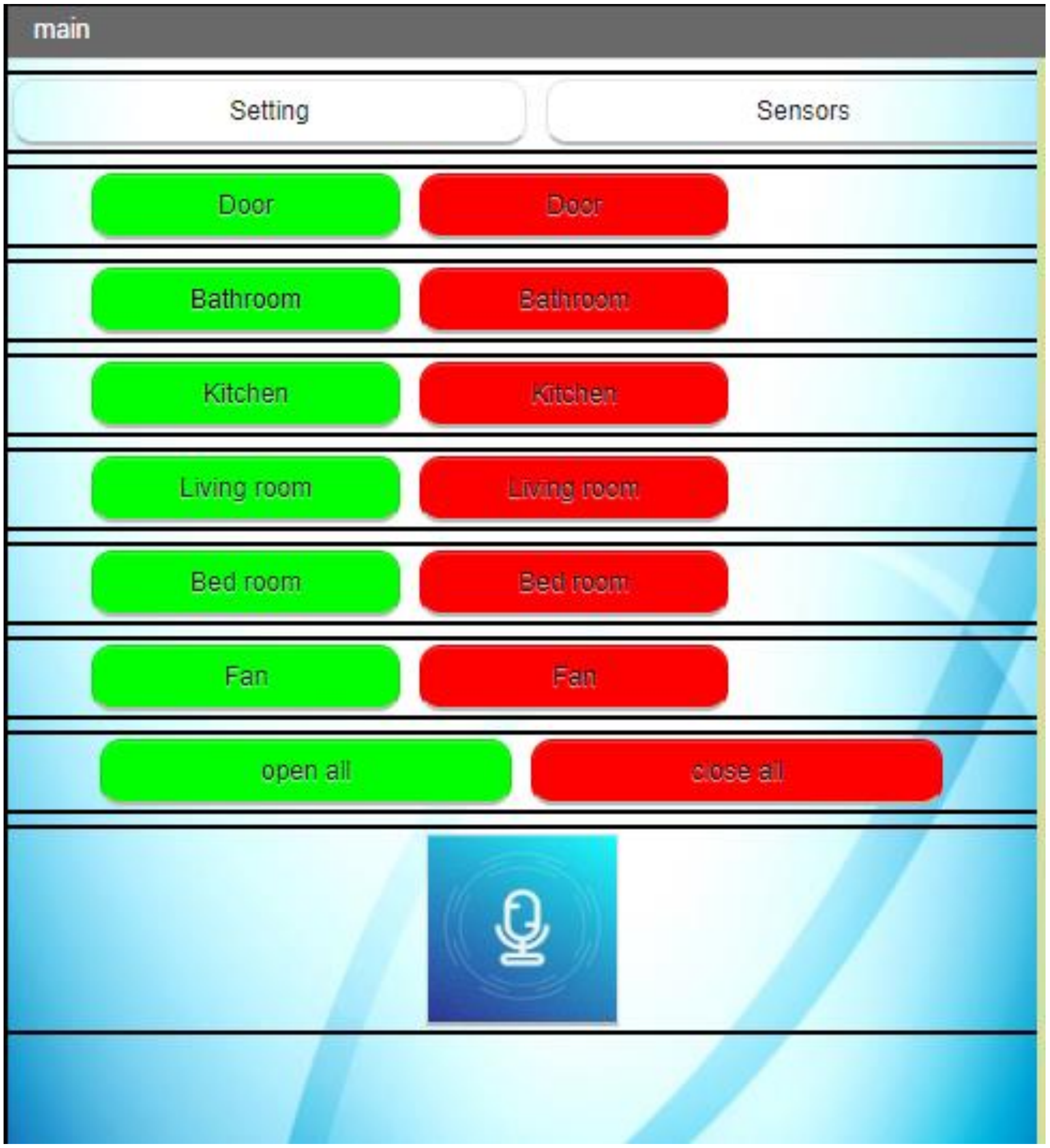
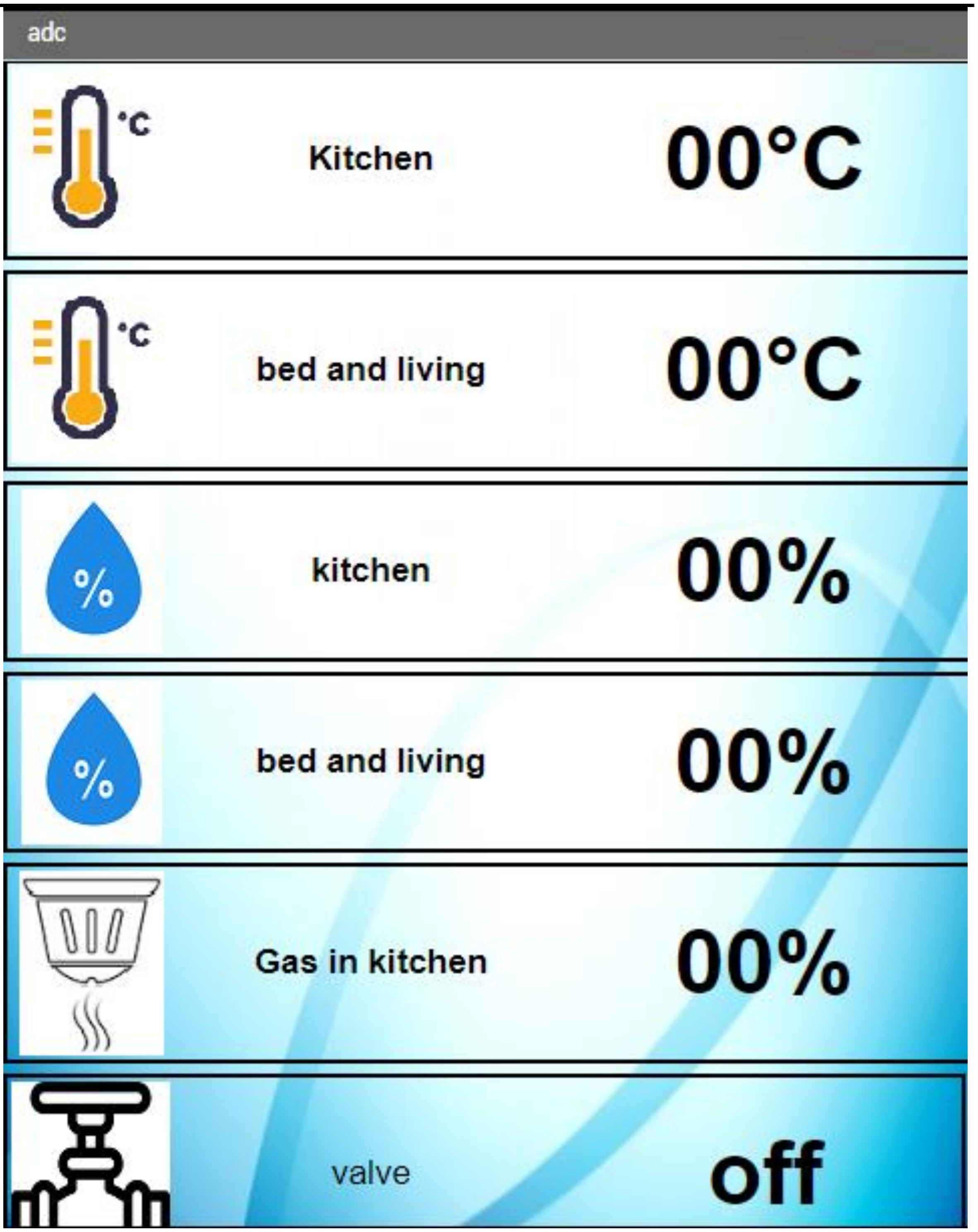


Figure 9 Our Mobile Application

This page in application can control home devices and lights, application implementation in page



adc

°C	Kitchen	00°C
°C	bed and living	00°C
%	kitchen	00%
%	bed and living	00%
	Gas in kitchen	00%
	valve	off

Figure 10 Sensors Reads And Some Devices States

In this page we can follow sensors reads and some devices states.

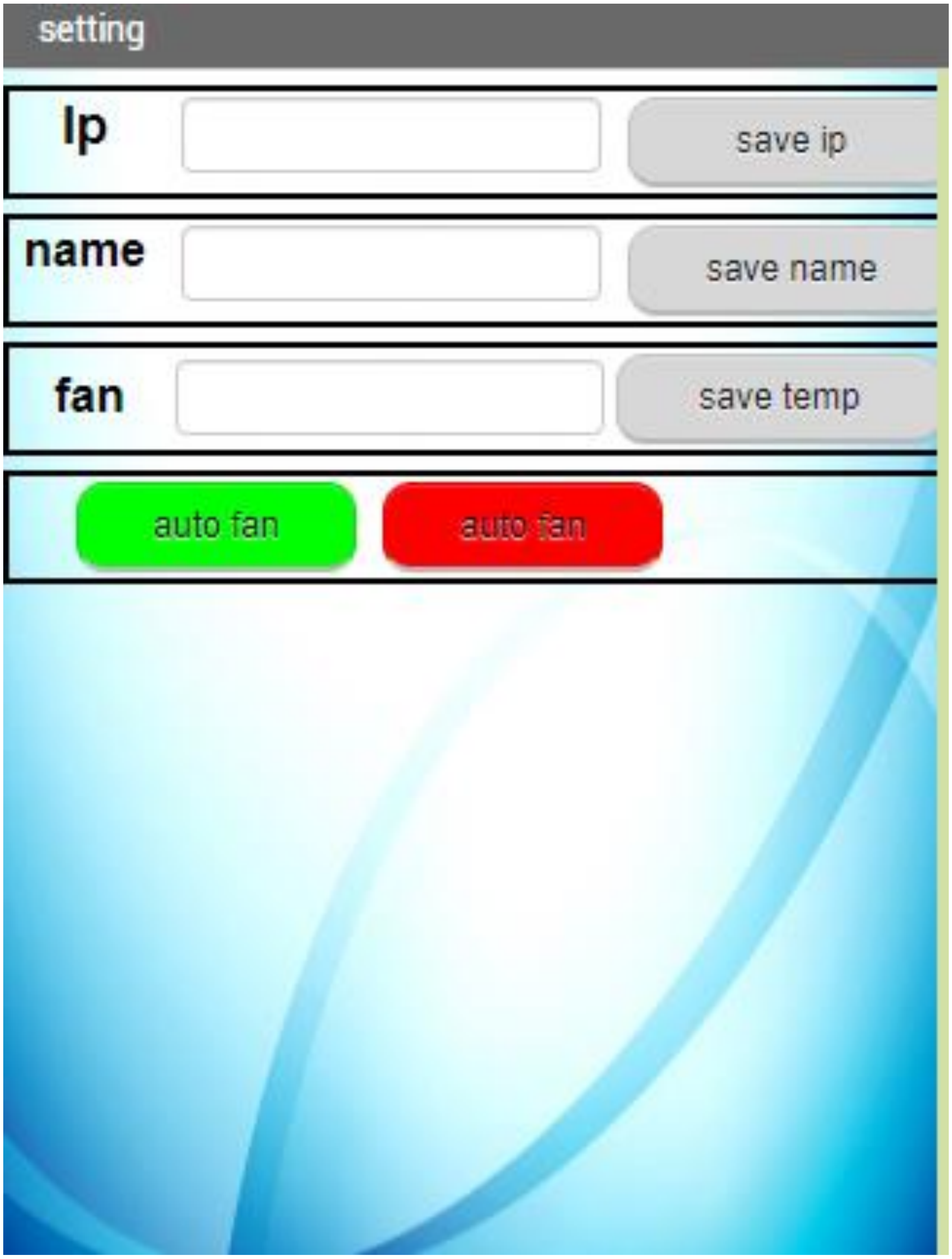


Figure 11 Server Ip And User Name

In this page we set server ip and user name and some settings to match our need.

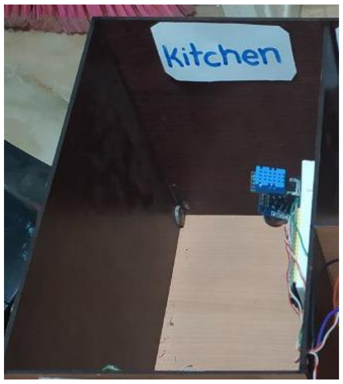


Figure 12 Model Of Kitchen

Temperature sensor and gas sensor in kitchen to provide safety in it.

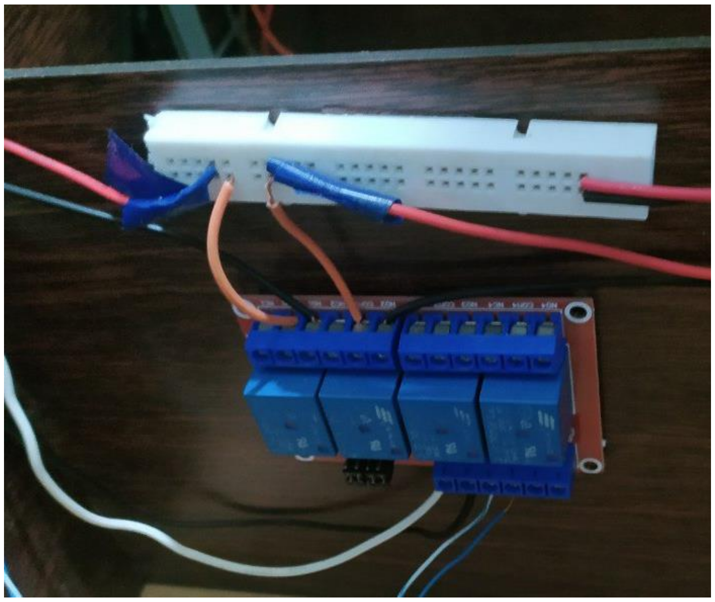


Figure 13 Power Circuit

Power circuit is completely isolated from control circuit using relays and opt coupler.

In the ever-changing technology trends a few components are being used in an attempt to make a more efficient, powerful and user-friendly smart home system. The technologies used for development of this smart home system are:

1. Wi-Fi.
2. Hypertext Transfer Protocol(HTTP).
3. Universal asynchronous receiver-transmitter(UART).

3.3 Wi-Fi

Wi-Fi is a family of wireless network protocols, based on the IEEE 802.11 family of standards, which are commonly used for local area networking of devices and Internet access, allowing nearby digital devices to exchange data by radio waves. These are the most widely used computer networks in the world, used globally in home and small office networks to link desktop and laptop computers, tablet computers, smartphones, smart TVs, printers, and smart speakers together and to a wireless router to connect them to the Internet, and in wireless access points in public places like coffee shops, hotels, libraries and airports to provide the public Internet access for mobile devices.

3.4 Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is an application layer protocol in the Internet protocol suite model for distributed, collaborative, hypermedia information systems.[1] HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example by a mouse click or by tapping the screen in a web browser[4].

Development of HTTP was initiated by Tim Berners-Lee at CERN in 1989 and summarized in a simple document describing the behavior of a client and a server using the first HTTP protocol version that was named 0.9.

Figure 14 HTTP

3.4.1 Technical overview

HTTP functions as a request–response protocol in the client–server model. A web browser, for example, may be the client whereas a process, named web server, running on a computer hosting one or more websites may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body[4].

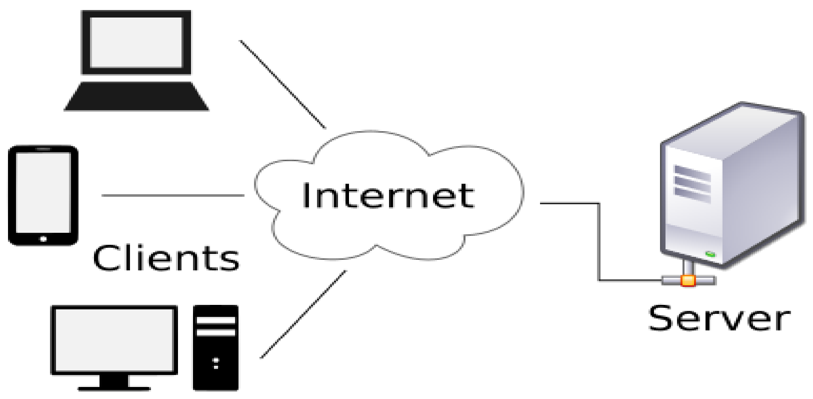


Figure 15 **client–server model**

A web browser is an example of a user agent (UA). Other types of user agent include the indexing software used by search providers (web crawlers), voice browsers, mobile apps, and other software that accesses, consumes, or displays web content.

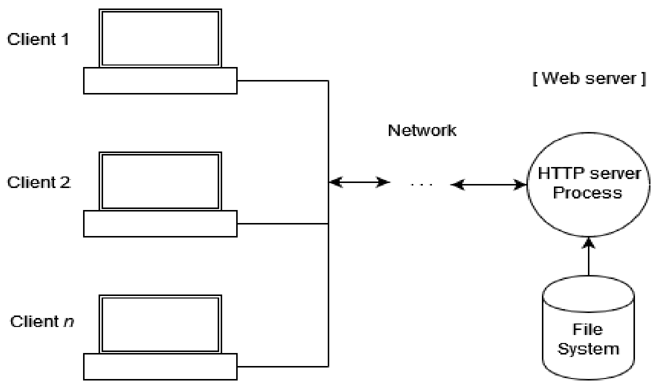


Figure 16 **user agent**

HTTP is designed to permit intermediate network elements to improve or enable communications between clients and servers. High-traffic websites often benefit from web cache servers that deliver content on behalf of upstream servers to improve response time. Web browsers cache previously accessed web resources and reuse them, whenever possible, to reduce network traffic. HTTP proxy servers at private network boundaries can facilitate communication for clients without a globally routable address, by relaying messages with external servers[4].

To allow intermediate HTTP nodes (proxy servers, web caches, etc.) to accomplish their functions, some of the HTTP headers (found in HTTP requests/responses) are managed hop-by-hop whereas other HTTP headers are managed end-to-end (managed only by the source client and by the target web server).

HTTP is an application layer protocol designed within the framework of the Internet protocol suite. Its definition presumes an underlying and reliable transport layer protocol,[5] thus Transmission Control Protocol (TCP) is commonly used. However, HTTP can be adapted to use unreliable protocols such as the User Datagram Protocol (UDP), for example in HTTPU and Simple Service Discovery Protocol (SSDP)[4].

3.4.2 Request syntax

A client sends request messages to the server, which consist of:

3.4.2.1 request line

consisting of the case-sensitive request method, a space, the requested URL, another space, the protocol version, a carriage return, and a line feed, example:

GET /images/logo.png HTTP/1.1

3.4.2.2 Request methods

HTTP defines methods (sometimes referred to as verbs, but nowhere in the specification does it mention verb) to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server. The HTTP/1.0 specification[50] defined the GET, HEAD, and POST methods, and the HTTP/1.1 specification[51] added five new methods: PUT, DELETE, CONNECT, OPTIONS, and TRACE. Any client can use any method and the server can be configured to support any combination of methods. If a method is unknown to an intermediate, it will be treated as an unsafe and non-idempotent method. There is no limit to the number of methods that can be defined, which allows for future methods to be specified without breaking existing infrastructure. For example, WebDAV defined seven new methods and RFC 5789 specified the PATCH method[4].

- GET

The GET method requests that the target resource transfer a representation of its state. GET requests should only retrieve data and should have no other effect. (This is also true of some other HTTP methods.) For retrieving resources without making changes, GET is preferred over POST, as they can be addressed through a URL. This enables bookmarking and sharing and makes GET responses eligible for caching, which can save bandwidth. The W3C has published guidance principles on this distinction, saying, "Web application design should be informed by the above principles, but also by the relevant limitations. “See safe methods below.

- HEAD

The HEAD method requests that the target resource transfer a representation of its state, as for a GET request, but without the representation data enclosed in the response body. This is useful for retrieving the representation metadata in the response header, without having to transfer the entire representation. Uses include checking whether a page is available through the status code and quickly finding the size of a file (Content-Length).

- POST

The POST method requests that the target resource process the representation enclosed in the request according to the semantics of the target resource. For example, it is used for posting a message to an Internet forum, subscribing to a mailing list, or completing an online shopping transaction.

- DELETE

The DELETE method requests that the target resource delete its state.

- PUT

The PUT method requests that the target resource create or update its state with the state defined by the representation enclosed in the request. A distinction from POST is that the client specifies the target location on the server.

- CONNECT

The CONNECT method requests that the intermediary establish a TCP/IP tunnel to the origin server identified by the request target. It is often used to secure connections through one or more HTTP proxies with TLS.

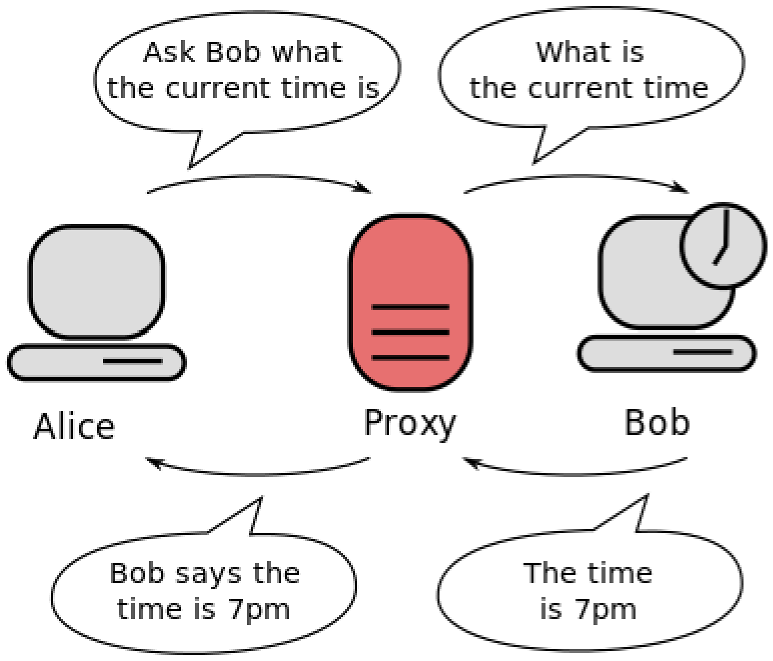


Figure 17Connection Figure

- OPTIONS

The OPTIONS method requests that the target resource transfer the HTTP methods that it supports. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.

- TRACE

The TRACE method requests that the target resource transfer the received request in the response body. That way a client can see what (if any) changes or additions have been made by intermediaries.

- PATCH

The PATCH method requests that the target resource modify its state according to the partial update defined in the representation enclosed in the request. This can save bandwidth by updating a part of a file or document without having to transfer it entirely.

3.5 Universal asynchronous receiver-transmitter

A universal asynchronous receiver-transmitter (UART) is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. It sends data bits one by one, from the least significant to the most significant, framed by start and stop bits so that precise timing is handled by the communication channel. The electric signaling levels are handled by a driver circuit external to the UART. Two common signal levels are RS-232, a 12-volt system, and RS-485, a 5-volt system. Early teletypewriters used current loops[6].

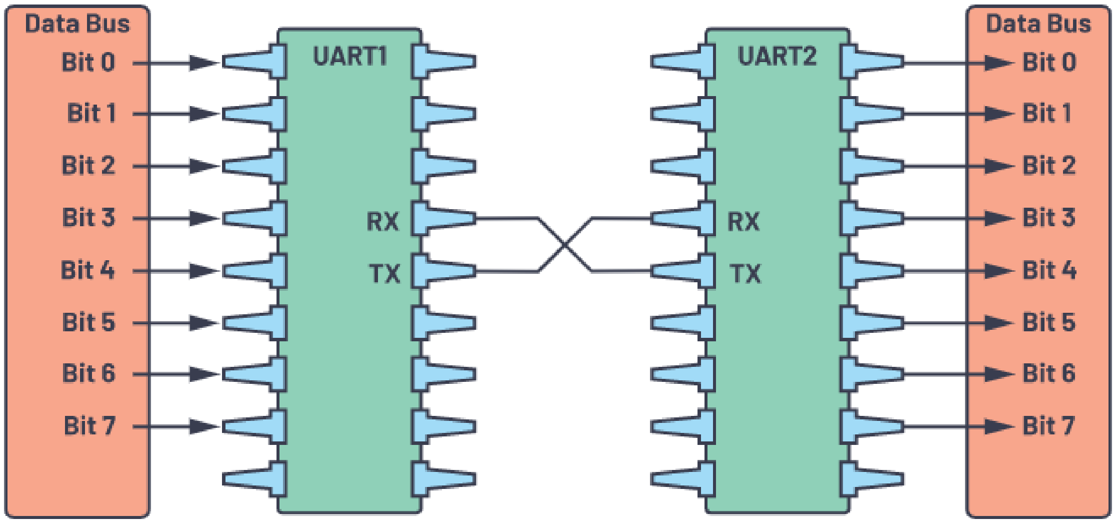


Figure 18Ic Connection

It was one of the earliest computer communication devices, used to attach teletypewriters for an operator console. It was also an early hardware system for the Internet.

A UART is usually an individual (or part of an) integrated circuit (IC) used for serial communications over a computer or peripheral device serial port. One or more UART peripherals are commonly integrated in microcontroller chips. Specialised UARTs are used for automobiles, smart cards and SIMs[7].

3.5.1 Transmitting and receiving serial data

The universal asynchronous receiver-transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion.[1] At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires[7].

The UART usually does not directly generate or receive the external signals used between different items of equipment. Separate interface devices are used to convert the logic level signals of the UART to and from the external signalling levels, which may be standardized voltage levels, current levels, or other signals[7].

3.5.2 Communication may be 3 modes:

- simplex (in one direction only, with no provision for the receiving device to send information back to the transmitting device)
- full duplex (both devices send and receive at the same time)
- half duplex (devices take turns transmitting and receiving)

3.5.2.1 Receiver

All operations of the UART hardware are controlled by an internal clock signal which runs at a multiple of the data rate, typically 8 or 16 times the bit rate. The receiver tests the state of the incoming signal on each clock pulse, looking for the beginning of the start bit. If the apparent start bit lasts at least one-half of the bit time, it is valid and signals the start of a new character. If not, it is considered a spurious pulse and is ignored. After waiting a further bit time, the state of the line is again sampled and the resulting level clocked into a shift register. After the required number of bit periods for the character length (5 to 8 bits, typically) have elapsed, the contents of the shift register are made available (in parallel fashion) to the receiving system. The UART will set a flag indicating new data is available, and may also generate a processor interrupt to request that the host processor transfers the received data[7].

Communicating UARTs have no shared timing system apart from the communication signal. Typically, UARTs resynchronize their internal clocks on each change of the data line that is not considered a spurious pulse. Obtaining timing information in this manner, they reliably receive when the transmitter is sending at a slightly different speed than it should. Simplistic UARTs do not do this; instead they resynchronize on the falling edge of the start bit only, and then read the center of each expected data bit, and this system works if the broadcast data rate is accurate enough to allow the stop bits to be sampled reliably.

It is a standard feature for a UART to store the most recent character while receiving the next. This "double buffering" gives a receiving computer an entire character transmission time to fetch a received character. Many UARTs have a small first-in, first-out (FIFO) buffer memory between the receiver shift register and the host system interface. This allows the host processor even more time to handle an interrupt from the UART and prevents loss of received data at high rates.

3.5.2.2 Transmitter

Transmission operation is simpler as the timing does not have to be determined from the line state, nor is it bound to any fixed timing intervals. As soon as the sending system deposits a character in the shift register (after completion of the previous character), the UART generates a start bit, shifts the required number of data bits out to the line, generates and sends the parity bit (if used), and sends the stop bits. Since full-duplex operation requires characters to be sent and received at the same time, UARTs use two different shift registers for transmitted and received characters. High performance UARTs could contain a transmit FIFO (first in first out) buffer to allow a CPU or DMA controller to deposit multiple characters in a burst into the FIFO rather than have to deposit one character at a time into the shift register. Since transmission of a single or multiple characters may take a long time relative to CPU speeds, a UART maintains a flag showing busy status so that the host system knows if there is at least one character in the transmit buffer or shift register; "ready for next character(s)" may also be signaled with an interrupt[7].

3.5.2.3 Application

Transmitting and receiving UARTs must be set for the same bit speed, character length, parity, and stop bits for proper operation. The receiving UART may detect some mismatched settings and set a "framing error" flag bit for the host system; in exceptional cases, the receiving UART will produce an erratic stream of mutilated characters and transfer them to the host system[7].

Typical serial ports used with personal computers connected to modems use eight data bits, no parity, and one stop bit; for this configuration, the number of ASCII characters per second equals the bit rate divided by 10.

Some very low-cost home computers or embedded systems dispense with a UART and use the CPU to sample the state of an input port or directly manipulate an output port for data transmission. While very CPU-intensive (since the CPU timing is critical), the UART chip can thus be omitted, saving money and space. The technique is known as bit-banging[7].

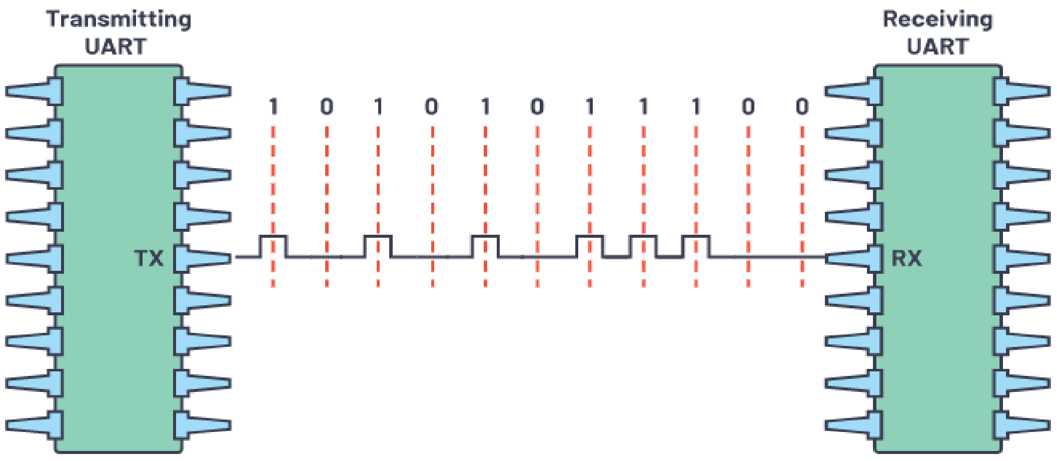


Figure 19 Transmitting and Receiving UART

4 COMPONENT

There are six main parts of the system (as shown on Fig.):

1. Group of connected sensors.
2. Raspberry Pi device that acts as a server system.
3. Android device as a remote client.
4. ESP8266.
5. Smart home appliances.
6. 8051 Microcontroller

4.1 Raspberry Pi

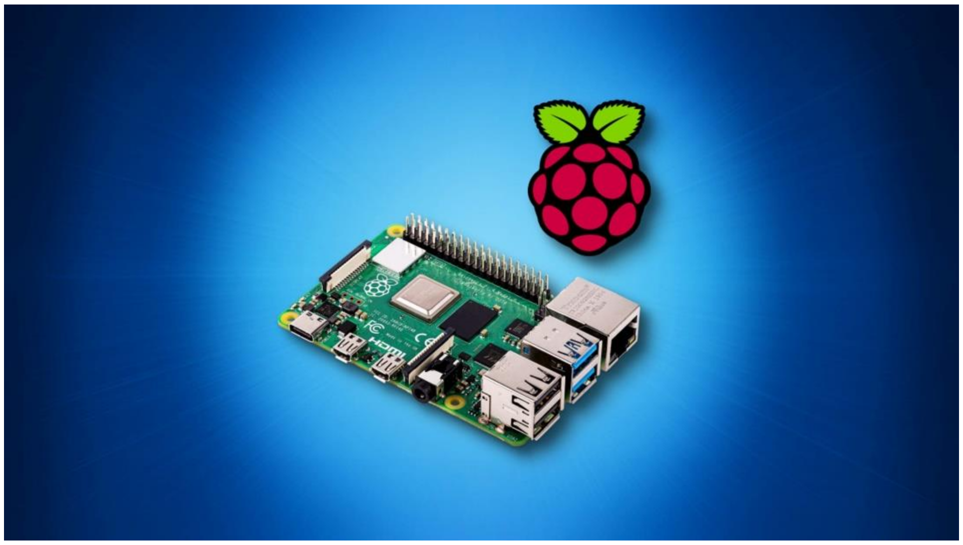


Figure 20Raspberry Pi

4.1.1 What is a Raspberry Pi?

Raspberry Pi is the name of a series of single-board computers made by the Raspberry Pi Foundation, a UK charity that aims to educate people in computing and create easier access to computing education[8].

The Raspberry Pi launched in 2012, and there have been several iterations and variations released since then. The original Pi had a single-core 700MHz CPU and just 256MB RAM, and the latest model has a quad-core CPU clocking in at over 1.5GHz, and 4GB RAM. The price point for Raspberry Pi has always been under $100 (usually around $35 USD), most notably the Pi Zero, which costs just $5.

All over the world, people use the Raspberry Pi to learn programming skills, build hardware projects, do home automation, implement Kubernetes clusters and Edge computing, and even use them in industrial applications.

The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and explore the Internet of Things (IoT).

4.1.2 What Raspberry Pi models have been released?

There have been many generations of the Raspberry Pi line: from Pi 1 to 4, and even a Pi 400. There has generally been a Model A and a Model B of most generations. Model A has been a less expensive variant, and tends to have reduced RAM and fewer ports (such as USB and Ethernet)[8]. The Pi Zero is a spinoff of the original (Pi 1) generation, made even smaller and cheaper. Here's the lineup so far:

Pi 1 Model B (2012)

Pi 1 Model A (2013)

Pi 1 Model B+ (2014)

Pi 1 Model A+ (2014)

Pi 2 Model B (2015)

Pi Zero (2015)

Pi 3 Model B (2016)

Pi Zero W (2017)

Pi 3 Model B+ (2018)

Pi 3 Model A+ (2019)

Pi 4 Model A (2019)

Pi 4 Model B (2020)

Pi 400 (2021)

4.1.3 What's the Raspberry Pi Foundation?

The Raspberry Pi Foundation works to put the power of computing and digital making into the hands of people all over the world. It does this by providing low-cost, high-performance computers that people use to learn, solve problems, and have fun. It provides outreach and education to help more people access computing and digital making—it develops free resources to help people learn about computing and making things with computers and also trains educators who can guide other people to learn[8].

Code Club and CoderDojo are part of the Raspberry Pi Foundation, although these programs are platform-independent (they're not tied to Raspberry Pi hardware). The Raspberry Pi Foundation promotes these clubs and helps grow the network around the world in order to ensure every child has access to learning about computing. Similarly, Raspberry Jams are Raspberry Pi-focused events for people of all ages to come together to learn about Raspberry Pi and share ideas and projects[8].

4.1.4 Pin Configuration of Raspberry Pi

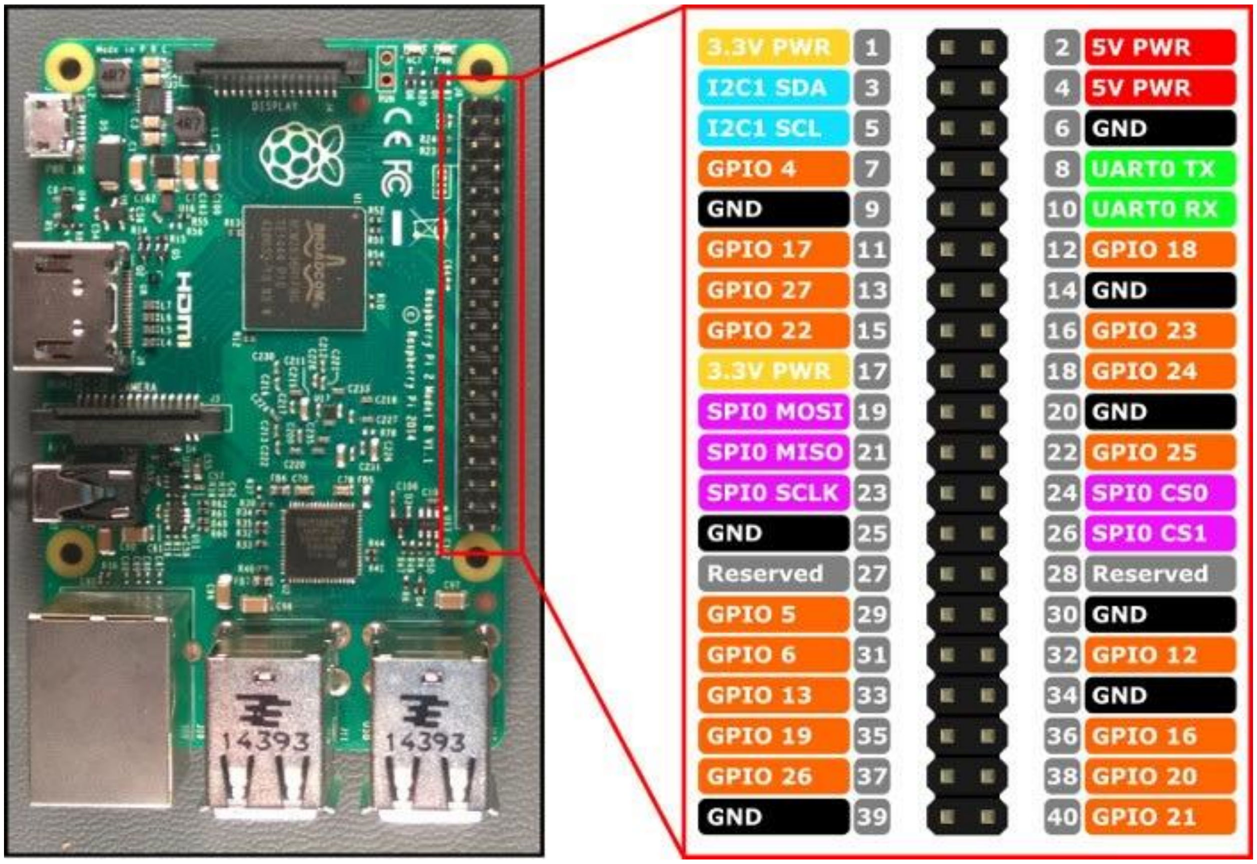


Figure 21 Pin Configuration

4.1.5 Is the Raspberry Pi open source?

The Raspberry Pi operates in the open source ecosystem: it runs Linux (a variety of distributions), and its main supported operating system, Pi OS, is open source and runs a suite of open source software. The Raspberry Pi Foundation contributes to the Linux kernel and various other open source projects as well as releasing much of its own software as open source[8].

The Raspberry Pi's schematics are regularly released as documentation, but the board is not open hardware.

4.2 ESP8266

4.2.1 What is the ESP8266?

The ESP8266 module enables microcontrollers to connect to 2.4 GHz Wi-Fi, using IEEE 802.11 bgn. It can be used with ESP-AT firmware to provide Wi-Fi connectivity to external host MCUs, or it can be used as a self-sufficient MCU by running an RTOS-based SDK. The module has a full TCP/IP stack and provides the ability for data processing, reads and controls of GPIOs[9].

4.2.2 Low-power, highly-integrated Wi-Fi solution

The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much WiFi-ability as a WiFi Shield offers (and that’s just out of the box)! The ESP8266 module is an extremely cost effective board with a huge, and ever growing, community[9].

4.2.3 Technical Specifications

1. Processor: L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz.
2. 32 KiB instruction RAM
3. 32 KiB instruction cache RAM
4. 80 KiB user-data RAM
5. 16 KiB ETS system-data RAM
6. External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
7. IEEE 802.11 b/g/n Wi-Fi
8. Integrated TR switch, balun, LNA, power amplifier and matching network
9. WEP or WPA/WPA2 authentication, or open networks

   16 GPIO pins

10. SPI
11. I²C (software implementation)
12. I²S interfaces with DMA (sharing pins with GPIO)
13. UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2

4.2.4 ESP8266 Functions

ESP8266 has many applications when it comes to the IoT. Here are just some of the functions [9] the chip is used for:

- Networking: The module's Wi-Fi antenna enables embedded devices to connect to routers and transmit data
- Data Processing: Includes processing basic inputs from analog and digital sensors for far more complex calculations with an RTOS or Non-OS SDK
- P2P Connectivity: Create direct communication between ESPs and other devices using IoT P2P connectivity
- Web Server: Access pages written in HTML or development languages.

4.2.5 Where to Use/Applications of ESP8266 Wi-Fi Module

The applications of the ESP8266 Wi-Fi module are given below

- Access points portals
- IoT projects:
  - Smart security devices, including surveillance cameras and smart locks
  - Smart energy devices, including HVACs and thermostats
  - Smart industrial devices, including Programmable Logic Controllers (PLCs)
  - Smart medical devices, including wearable health monitors
- Wireless data logging
- Used in learning the networking fundamentals
- Sockets and smart bulbs
- Smart home automation systems

The ESP32 is an alternative ESP8266 Wi-Fi module. It is a standalone and most powerful module.

Thus, this is all about an overview of the ESP8266 Wi-Fi module datasheet – definition, pin configuration, specifications, circuit diagram/How to use, where to use/applications, and its alternatives. The ESP8266 Wi-Fi module is a User-friendly module because it can be programmed with the help of Arduino IDE. This module can also be used to build ESP8266 wifi module projects. The other standalone modules like ESP-12 and ESP-32 are also commonly used for IoT applications development and to achieve internet connection to the project[9].

4.2.6 Advantage of ESP8266 over other competitors like Arduino

The main advantage is the embedded wireless technology that is web friendly with no use of shields or any peripherals, as is required for Arduinos. The price and size are the USP of the module with the added advantage of good speed and processing power. Arduinos are approximately 10 times more expensive and much larger than ESP with comparable processing powers.

4.2.7 Circuit Diagram/How to Use?

There are several techniques and IDEs are available by using ESP8266 Wi-Fi modules. The Arduino IDE is the most commonly used technique. Now, let’s learn the working of the Arduino IDE using the ESP8266 Wi-Fi module. The circuit diagram/how to use the Arduino IDE or FTDI device is illustrated in the below figure.

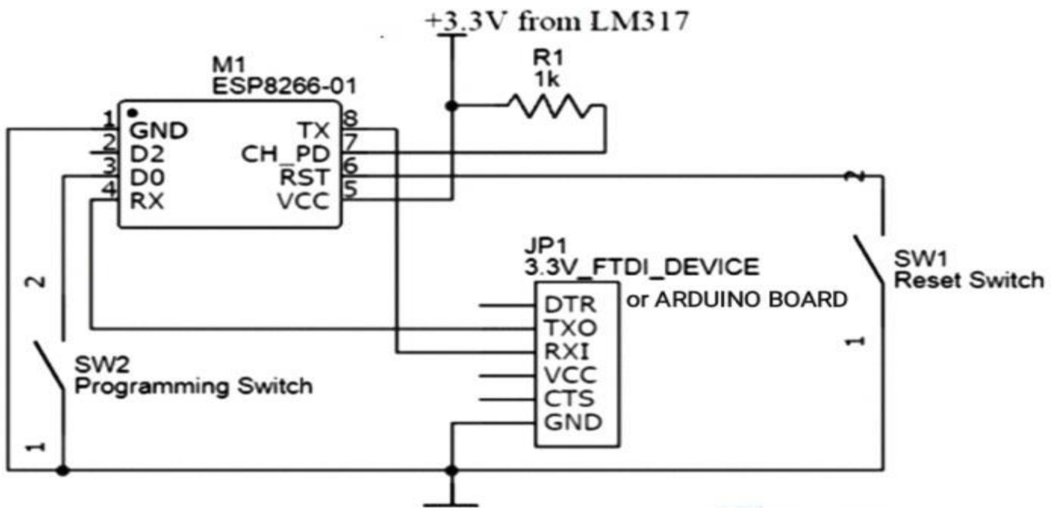


Figure 22 Circuit Diagram of ESP8266 Module

The power supply required for the ESP8266 module is only 3.3 Volts. If it is more than 3.7 Volts, then the module gets damaged, and this leads to circuit failure. Hence it is necessary to program the ESP-01 Wi-Fi module by using either Arduino board or FTDI device, which supports the programming 3.3 Volts supply. It is recommended for the user to buy either one FTDI device or an Arduino board.

The most common issue with the ESP-01 module is the powering up issue. The 3.3 Volts pin on the Arduino board is used to power up this module or simply we can use the potential divider. So, to provide a minimum current of 500 mA, the voltage regulator that supports 3.3 Volts is mandatory. The LM317 voltage regulator does this work very easily and effectively[9].

The programming switch SW2 is pressed to connect the GPIO-0 pin to the GND (Ground). This is the programming mode to upload the code by the user. After uploading the code, the switch is released.

4.2.8 Pin Configuration/Pin Diagram

The ESP8266 Wi-Fi module pin configuration/pin diagram is shown in the figure below. The ESP8266-01 Wi-Fi module runs in two modes[9]. They are;

Flash Mode: When GPIO-0 and GPIO-1 pins are active high, then the module runs the program, which is uploaded into it.

UART Mode: When the GPIO-0 is active low and GPIO-1 is active high, then the module works in programming mode with the help of either serial communication or Arduino board.

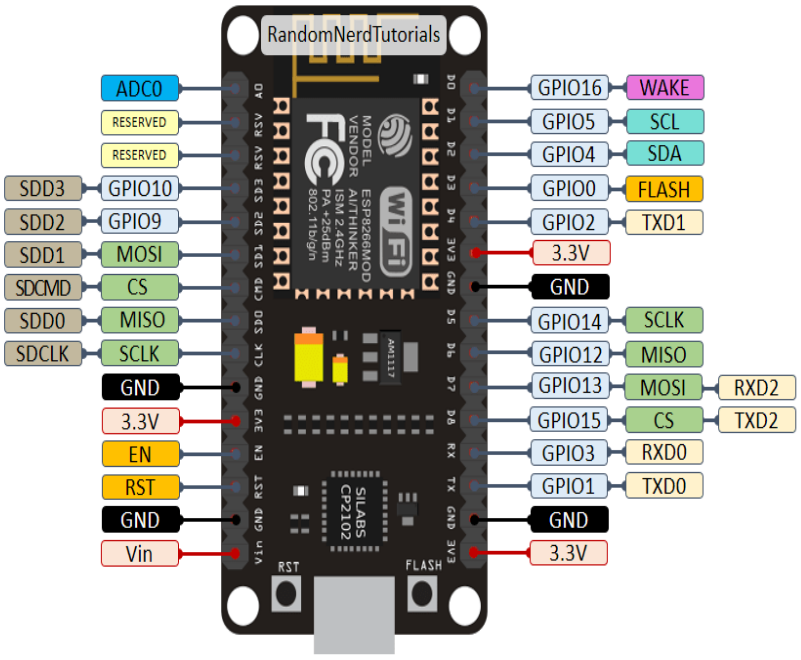


Figure 23 Pin Diagram

4.3 DHT11 Humidity & Temperature Sensor

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness[10].

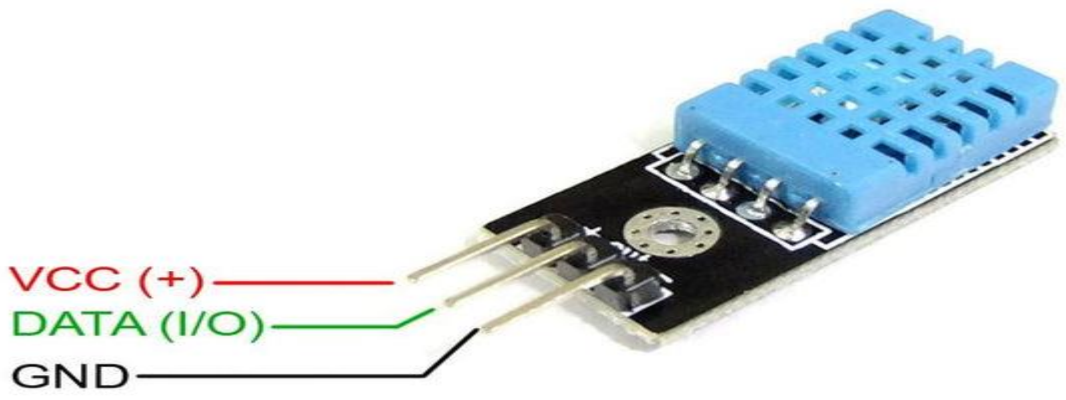


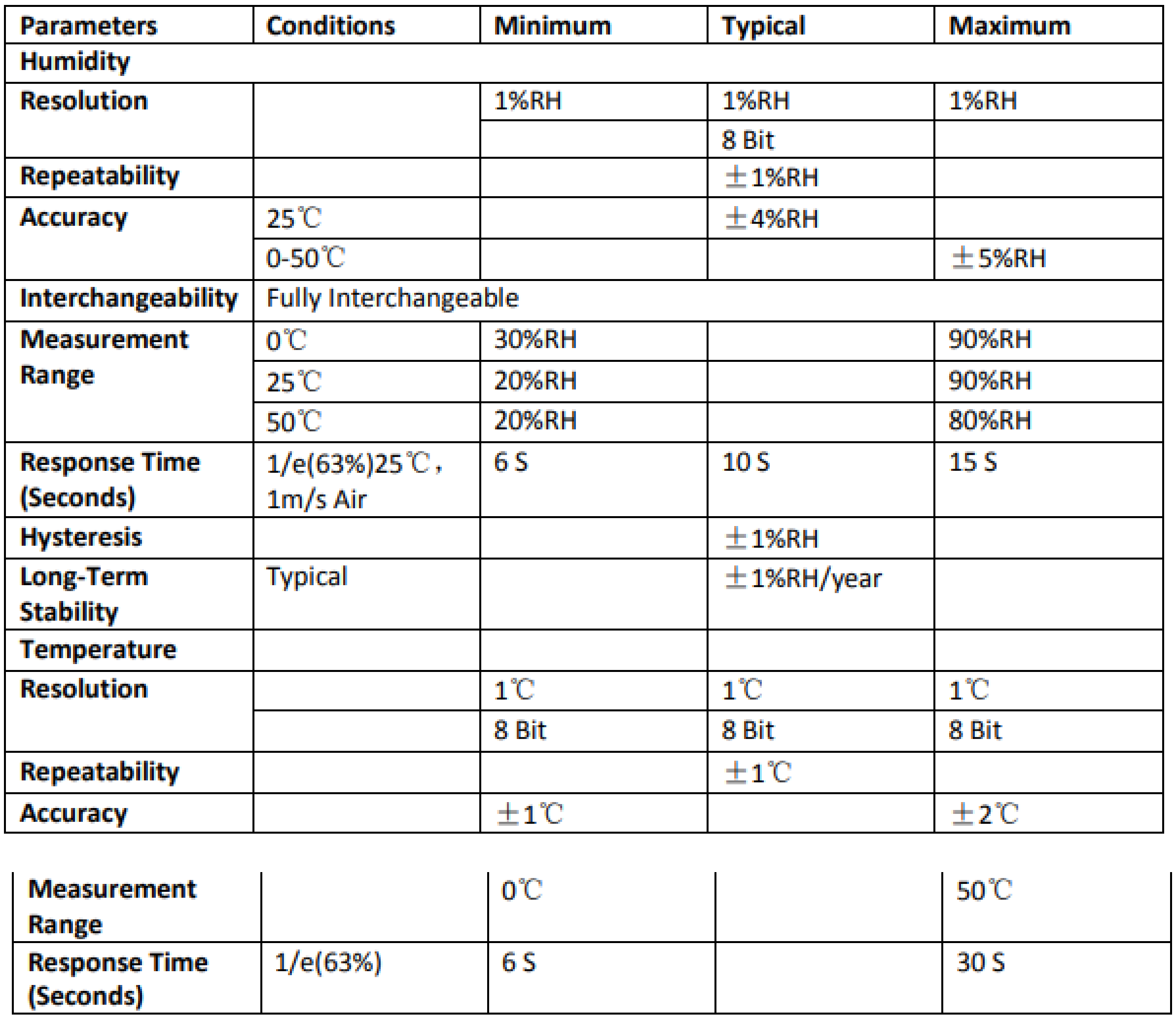
Figure 24 DHT11 Temperature & Humidity Sensor

Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmers in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connect and special packages can be provided according to users' request[10].

Table 3 : Technical Specifications of DHT11:

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Table 4 :Detailed Specifications of DHT11:



Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃，1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

4.3.1 Power and Pin

DHT11's power supply is 3-5.5V DC. When power is supplied to the sensor, do not send any instruction to the sensor in within one second in order to pass the unstable status. One capacitor valued 100nF can be added between VDD and GND for power filtering[11].

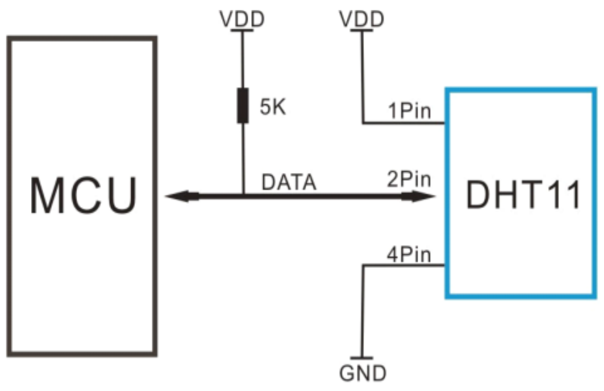


Figure 25 DHT11 Connection

4.3.2 Communication Process:

Serial Interface (Single-Wire Two-Way) Single-bus data format is used for communication and synchronization between MCU and DHT11 sensor. One communication process is about 4ms. Data consists of decimal and integral parts. A complete data transmission is 40bit, and the sensor sends higher data bit first. Data format: 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum. If the data transmission is right, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

4.3.2.1 Overall Communication Process:

When MCU sends a start signal, DHT11 changes from the low-power-consumption mode to the running-mode, waiting for MCU completing the start signal. Once it is completed, DHT11 sends a response signal of 40-bit data that include the relative humidity and temperature information to MCU. Users can choose to collect (read) some data. Without the start signal from MCU, DHT11 will not give the response signal to MCU. Once data is collected, DHT11 will change to the low power-consumption mode until it receives a start signal from MCU again[11].

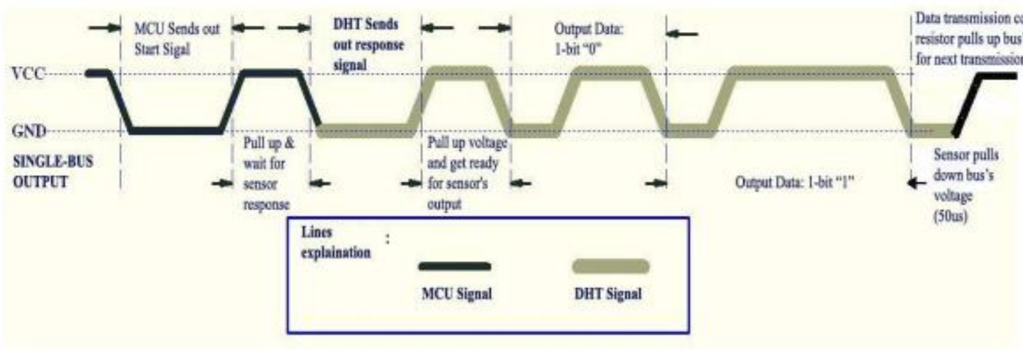
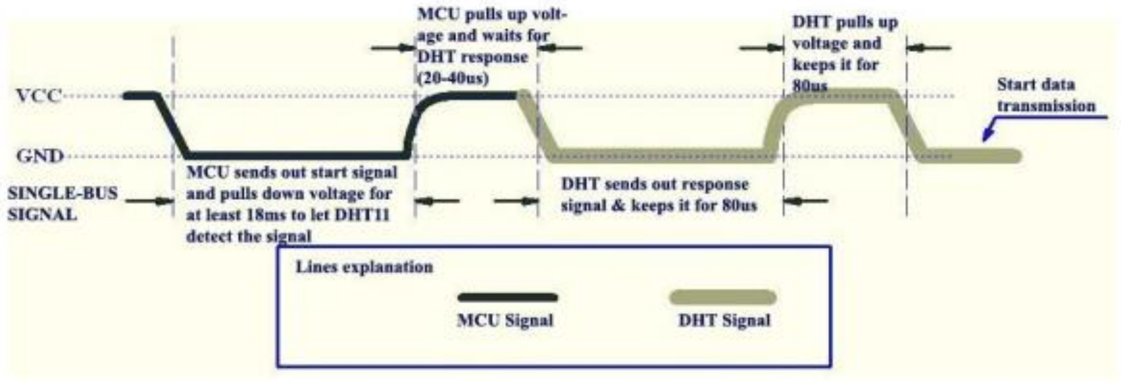

MCU Sends out Start Sigal
DHT Sends out response signal
Output Data: 1-bit "0"
Data transmission completed; resistor pulls up bus's voltage for next transmission
VCC
GND
SINGLE-BUS OUTPUT
Pull up & wait for sensor response
Pull up voltage and get ready for sensor's output
Output Data: 1-bit "1"
Sensor pulls down bus's voltage (50us)
Lines explaination :
MCU Signal
DHT Signal

Figure 26 Communication Process

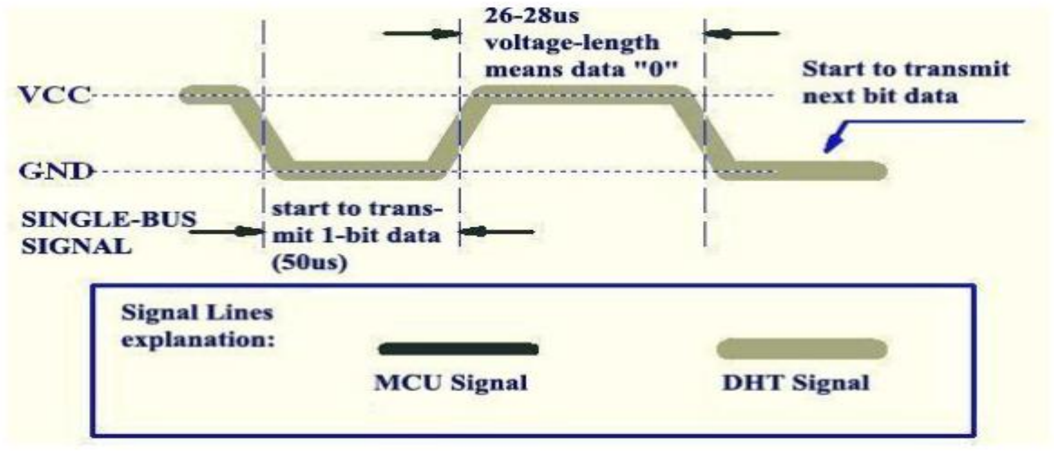
5.2 MCU Sends out Start Signal to DHT

Data Single-bus free status is at high voltage level. When the communication between MCU and DHT11 begins, the programme of MCU will set Data Single-bus voltage level from high to low and this process must take at least 18ms to ensure DHT's detection of MCU's signal, then MCU will pull up voltage and wait 20-40us for DHT's response[11].


MCU pulls up voltage and waits for DHT response (20-40us)
DHT pulls up voltage and keeps it for 80us
VCC
GND
Start data transmission
SINGLE-BUS SIGNAL
MCU sends out start signal and pulls down voltage for at least 18ms to let DHT11 detect the signal
DHT sends out response signal & keeps it for 80us
Lines explanation
MCU Signal
DHT Signal

4.3.3 DHT Responses to MCU

Once DHT detects the start signal, it will send out a low-voltage-level response signal, which lasts 80us. Then the programmer of DHT sets Data Single-bus voltage level from low to high and keeps it for 80us for DHT's preparation for sending data. When DATA Single-Bus is at the low voltage level, this means that DHT is sending the response signal. Once DHT sent out the response signal, it pulls up voltage and keeps it for 80us and prepares for data transmission. When DHT is sending data to MCU, every bit of data begins with the 50us low-voltage-level and the length of the following high-voltage-level signal determines whether data bit is "0" or "1"[11].


26-28us voltage-length means data "0"
VCC
GND
Start to transmit next bit data
SINGLE-BUS SIGNAL
start to transmit 1-bit data (50us)
Signal Lines explanation:
MCU Signal
DHT Signal

4.3.4 Electrical Characteristics

VDD=5V, T = 25°C (unless otherwise stated)

Table 5: Electrical Characteristics

	Conditions	Minimum	Typical	Maximum
Power Supply	DC	3V	5V	5.5V
Current Supply	Measuring	0.5mA		2.5mA
	Average	0.2mA		1mA
	Standby	100uA		150uA
Sampling period	Second	1		

4.4 Gas Sensors

Gas sensors are generally understood as providing a measurement of the concentration of some analyte of interest, such as CO, CO_2, NO_x, SO_2, without at this point dwelling on the plethora of underlying approaches such as optical absorption, electrical conductivity, electrochemical (EC), and catalytic bead (see Section 3). However, and as discussed in Section 2, many other gas sensors measure a physical property of the environment around them, such as simple temperature, pressure, flow, thermal conductivity, and specific heat, or more complex properties such as heating value, super compressibility, and octane number for gaseous fuels. The latter may require capital-intensive (engines) or destructive testing, for example, via combustion, or involve the measurement of a number of parameters to serve as inputs to a correlation with the complex property of interest[12].

When the sensor provides a multiplicity of outputs, as with optical or mass spectrometers (MSs), we refer to it as a gas analyzer. Gas chromatography (GC), differential thermal analysis (DTA), ion mobility, and nuclear magnetic resonance (NMR) are additional examples, some of which will be detailed in Section 4. Such analyzers, preferred by the author, should not be confused with sensor arrays, in which different sensing materials (typically polymers and metal oxides) are used on each element of the array, which then needs to conform to difficult-to-achieve stability requirements.

The performance of all of the above-mentioned sensors and analyzers may be characterized by their signal-to-noise (S/N) ratio, minimum detectable limit (MDL), selectivity, and response time. Increasingly, power consumption, size, and weight are becoming more important as interest and demand increases for handheld, battery-powered sensors, with or without wireless capability. These specifications may be viewed as simple performance parameters, because they are relatively simple to quantify[12].

Self-calibration, drift, S/N, and false alarm rate (FAR) (mainly for composition sensors or analyzers) require more sophisticated approaches, but are of increasing importance in all applications such as for medical, industrial, environmental, security, and first-responder use. Section 5 goes into the details of this subject.

Another classification of gas sensors and analyzers could be based on their sampling method: by diffusion, pumped transport, or via remote optical sampling to induce fluorescence, absorption, or scattering.

Researchers, designers, and planners continually face the need to make development or fabrication decisions before all the facts are available. Therefore, there is a perennial need to generate estimates about the performance and sensitivity of devices, structure, and also sensor systems. This is where mathematical modeling, be it simple or complex or *ad hoc* (to simulate a specific sensor) or multipurpose (such as ANSYS, FLUENT to simulate heat transfer or flow of a sensor within a given programmatic framework, which is adapted to individual geometries and conditions) can be of tremendous help. Rather than dedicating a section specifically to this subject, several examples will be discussed throughout this chapter[12].

The intent of this chapter is not to provide an exhaustive review of the world of gas sensors, nor a history of their development, but to highlight and share selected gas-sensing approaches that impressed the author in meeting modern expectations for performance, features, and cost.

The issue of cost merits additional comments. Contrary to initial negative reactions one might have about it, because of its potential commercialism, the author subscribes to the view that cost just adds a tough professional challenge (conservation, sustainable development, affordability) to all the others related to achieving generic and useful sensor performance attributes mentioned above. In fact, many elegant sensing approaches are withering on the shelf because few potential users could afford to implement them.

4.4.1 Different Types of Gas sensors

Gas sensors are typically classified into various types based on the type of the sensing element it is built with. Below is the classification of the various types of gas sensors based on the sensing element that are generally used in various applications:

- Metal Oxide based gas Sensor.
- Optical gas Sensor.
- Electrochemical gas Sensor.
- Capacitance-based gas Sensor.
- Calorimetric gas Sensor.

- Acoustic based gas Sensor.

4.4.2 Gas Sensor Construction

Of all the above-listed types, the most commonly used gas sensor is the Metal oxide semiconductor based gas sensor. All Gas sensors will consist of a sensing element which comprises of the following parts.

1. Gas sensing layer
2. Heater Coil
3. Electrode line
4. Tubular ceramic
5. Electrode

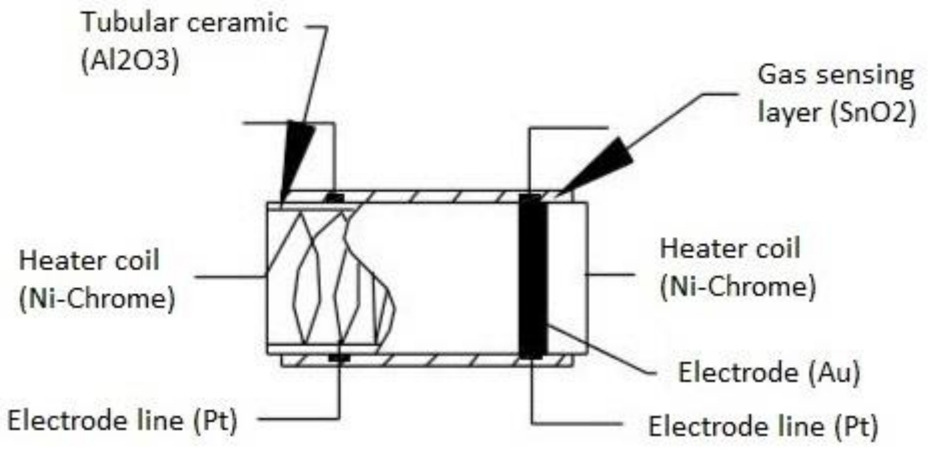


Figure 27 Gas Sensing

The purpose of each of these elements is as below:

4.4.2.1 *Gas sensing layer:*

It is the main component in the sensor which can be used to sense the variation in the concentration of the gases and generate the change in electrical resistance. The gas sensing layer is basically a chemiresistor which changes its resistance value based on the

The concentration of particular gas in the environment. Here the sensing element is made up of a Tin Dioxide (SnO_2) which is, in general, has excess electrons (donor element). So whenever toxic gases are being detected the resistance of the element changes and the current flown through it varies which represents the change in concentration of the gases[12].

4.4.2.2 Heater coil:

The purpose of the heater coil is to burn-in the sensing element so that the sensitivity and efficiency of the sensing element increases. It is made of Nickel-Chromium which has a high melting point so that it can stay heated up without getting melted.

4.4.2.3 Electrode line:

As the sensing element produces a very small current when the gas is detected it is more important to maintain the efficiency of carrying those small currents. So Platinum wires come into play where it helps in moving the electrons efficiently.

4.4.2.4 Electrode:

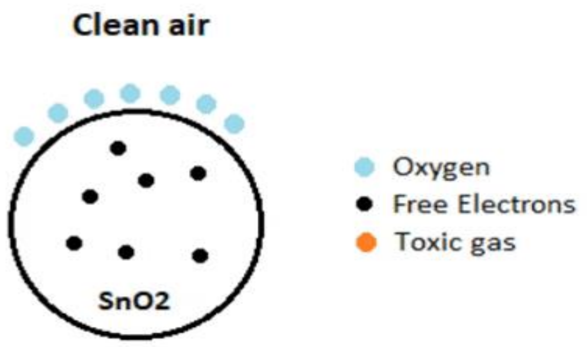
It is a junction where the output of the sensing layer is connected to the Electrode line. So that the output current can flow to the required terminal. An electrode here is made of Gold (Au – Aurum) which is a very good conductor.

4.4.2.5 Tubular ceramic:

In between the Heater coil and Gas sensing layer, the tubular ceramic exists which is made of Aluminum oxide (Al_2O_3). As it has high melting point, it helps in maintaining the burn-in (preheating) of the sensing layer which gives the high sensitivity for the sensing layer to get efficient output current.

4.4.3 Gas Sensor Working

The ability of a Gas sensor to detect gases depends on the chemiresister to conduct current. The most commonly used chemiresistor is Tin Dioxide (SnO_2) which is an n-type semiconductor that has free electrons (also called as donor). Normally the atmosphere will contain more oxygen than combustible gases. The oxygen particles attract the free electrons present in SnO_2 which pushes them to the surface of the SnO_2. As there are no free electrons available output current will be zero. The below gif shown the oxygen molecules (blue color) attracting the free electrons (black color) inside the SnO_2 and preventing it from having free electrons to conduct current.



When the sensor is placed in the toxic or combustible gases environment, this reducing gas (orange color) reacts with the adsorbed oxygen particles and breaks the chemical bond between oxygen and free electrons thus releasing the free electrons. As the free electrons are back to its initial position they can now conduct current, this conduction will be proportional the amount of free electrons available in SnO2, if the gas is highly toxic more free electrons will be available.

4.4.4 How to use a Gas sensor?

.A basic gas sensor has 6 terminals in which 4 terminals (A, A, B, B) acts input or output and the remaining 2 terminals (H, H) are for heating the coil. Of these 4 terminals, 2 terminals from each side can be used as either input or output (these terminals are reversible as shown in the circuit diagram) and vice versa[13].

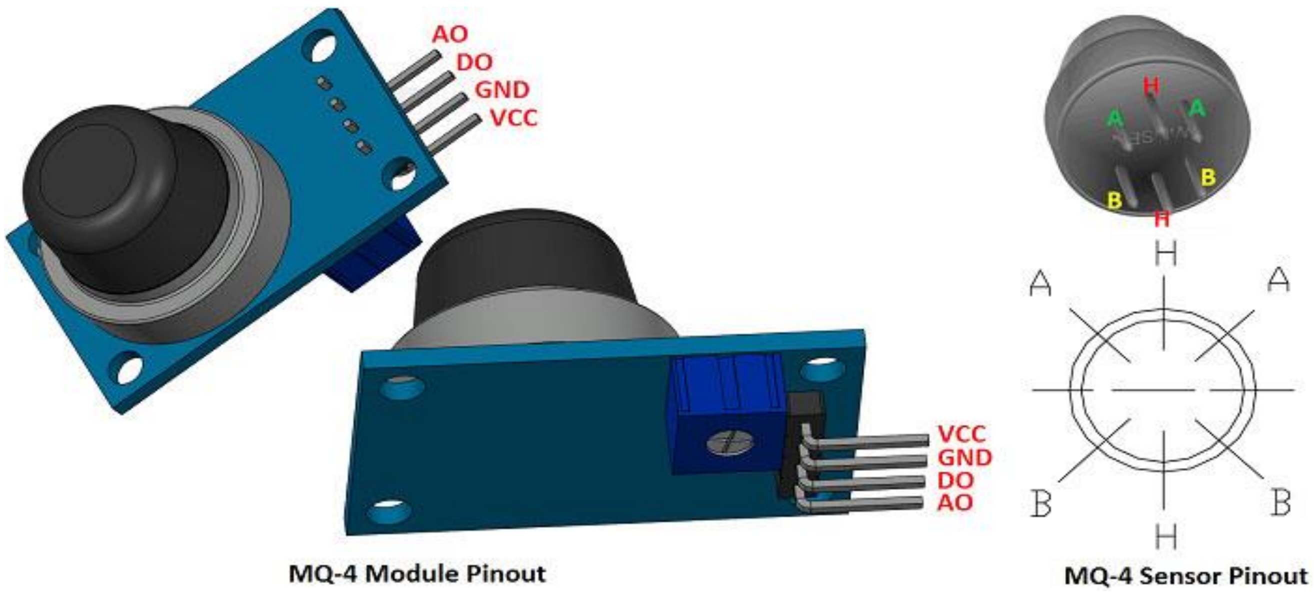


Figure 28 Gas Sensor Connection

These sensors are normally available as modules (shown right), these modules consist of the gas sensor and a comparator IC. Now let's see the pin description of the gas sensor module which we will generally use with an Arduino. The gas sensor module basically consists of 4 terminals

- Vcc – Power supply
- GND – Power supply
- Digital output – This pin gives an output either in logical high or logical low (0 or 1) that means it displays the presence of any toxic or combustible gases near the sensor.
- Analog output – This pin gives an output continuous in voltage which varies based on the concentration of gas that is applied to the gas sensor.

As discussed earlier the output of a gas sensor alone will be very small (in mV) so an external circuit has to be used in order to get a digital high low output from the sensor. For this purpose, a comparator (LM393), adjustable potentiometer, some resistors and capacitors are used[13].

The purpose of LM393 is to get the output from the sensor, compare it with a reference voltage and display whether the output is logically high or not. Whereas the purpose of the potentiometer is to set the required threshold value of the gas above which the digital output pin should go high[13].

The below diagram shows the basic circuit diagram of a gas sensor in a gas sensor module

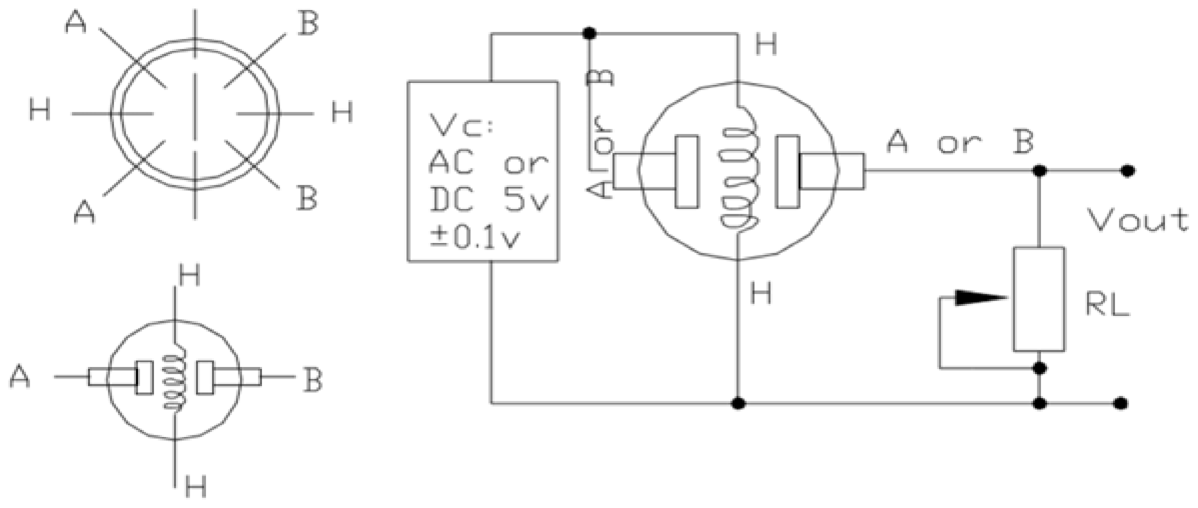


Figure 29 circuit diagram of a gas sensor

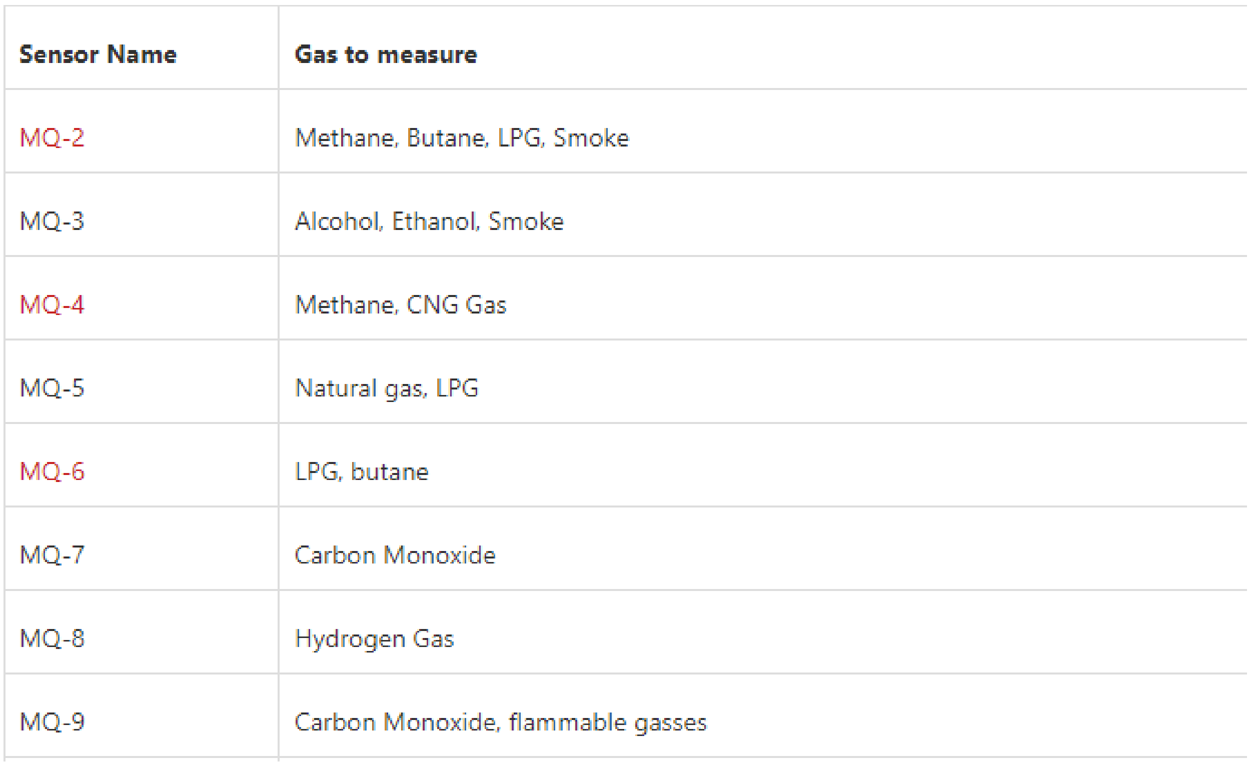
Here A and B are the input and output terminals (these are reversible - means any of the paired terminals can be used as input or output) and H is the Heater coil terminal. The purpose of the variable resistor is to adjust the output voltage and to maintain high sensitivity[13].

If no input voltage is applied to the heater coil, then the output current will be very less (which is negligible or approximately 0). When sufficient voltage is applied to the input terminal and heater coil, the sensing layer wakes up and is ready to sense any combustible gases nearby it. Initially let's assume that there is no toxic gas near the sensor, so the resistance of the layer doesn't change and the output current and voltage are also unchanged and are negligible (approximately 0).

Now let's assume that there is some toxic gas nearby. As the heater coil is pre-heated it is now easy to detect any combustible gases. When the sensing layer interacts with the gases, the resistance of the material varies and the current flowing through the circuit also varies. This change in variation can be then observed at the load resistance (RL).

The value of load resistance (RL) can be anywhere from 10KΩ to 47KΩ. The exact value of the load resistance can be selected by calibrating with the known concentration of the gas. If low load resistance is selected then the circuit has less sensitivity and if high load resistance is selected then the circuit has high sensitivity[13].

Table 6:List of Different Types of Gas Sensors and What Gases They Sense



Sensor Name	Gas to measure
MQ-2	Methane, Butane, LPG, Smoke
MQ-3	Alcohol, Ethanol, Smoke
MQ-4	Methane, CNG Gas
MQ-5	Natural gas, LPG
MQ-6	LPG, butane
MQ-7	Carbon Monoxide
MQ-8	Hydrogen Gas
MQ-9	Carbon Monoxide, flammable gasses

4.4.5 Applications of Gas Sensors

- Used in industries to monitor the concentration of the toxic gases.
- Used in households to detect an emergency incidents.
- Used at oil rig locations to monitor the concentration of the gases those are released.
- Used at hotels to avoid customers from smoking.
- Used in air quality check at offices.
- Used in air conditioners to monitor the CO2 levels.
- Used in detecting fire.
- Used to check concentration of gases in mines.
- Breath analyzer.

4.5 Stepper Motor

Stepper motors are DC motors that move in discrete steps. They have multiple coils that are organized in groups called "phases". By energizing each phase in sequence, the motor will rotate, one step at a time[14].

4.5.1 What are stepper motors good for?

With a computer controlled stepping you can achieve very precise positioning and/or speed control. For this reason, stepper motors are the motor of choice for many precision motion control applications.

Stepper motors come in many different sizes and styles and electrical characteristics. This guide details what you need to know to pick the right motor for the job[14].

What are their limitations?

- Positioning – Since steppers move in precise repeatable steps, they excel in applications requiring precise positioning such as 3D printers, CNC, Camera platforms and X,Y Plotters. Some disk drives also use stepper motors to position the read/write head[14].
- Speed Control – Precise increments of movement also allow for excellent control of rotational speed for process automation and robotics.
- Low Speed Torque - Normal DC motors don't have very much torque at low speeds. A Stepper motor has maximum torque at low speeds, so they are a good choice for applications requiring low speed with high precision.

- Low Efficiency – Unlike DC motors, stepper motor current consumption is independent of load. They draw the most current when they are doing no work at all. Because of this, they tend to run hot.
- Limited High Speed Torque - In general, stepper motors have less torque at high speeds than at low speeds. Some steppers are optimized for better high-speed performance, but they need to be paired with an appropriate driver to achieve that performance.
- No Feedback – Unlike servo motors, most steppers do not have integral feedback for position. Although great precision can be achieved running ‘open loop’. Limit switches or ‘home’ detectors are typically required for safety and/or to establish a reference position.

- Low Efficiency – Unlike DC motors, stepper motor current consumption is independent of load. They draw the most current when they are doing no work at all. Because of this, they tend to run hot.
- Limited High Speed Torque - In general, stepper motors have less torque at high speeds than at low speeds. Some steppers are optimized for better high-speed performance, but they need to be paired with an appropriate driver to achieve that performance.
- No Feedback – Unlike servo motors, most steppers do not have integral feedback for position. Although great precision can be achieved running ‘open loop’. Limit switches or ‘home’ detectors are typically required for safety and/or to establish a reference position.
- Low Efficiency – Unlike DC motors, stepper motor current consumption is independent of load. They draw the most current when they are doing no work at all. Because of this, they tend to run hot.

- Limited High Speed Torque - In general, stepper motors have less torque at high speeds than at low speeds. Some steppers are optimized for better high-speed performance, but they need to be paired with an appropriate driver to achieve that performance.

- No Feedback – Unlike servo motors, most steppers do not have integral feedback for position. Although great precision can be achieved running ‘open loop’. Limit switches or ‘home’ detectors are typically required for safety and/or to establish a reference position.

- Low Efficiency – Unlike DC motors, stepper motor current consumption is independent of load. They draw the most current when they are doing no work at all. Because of this, they tend to run hot.

- Limited High Speed Torque - In general, stepper motors have less torque at high speeds than at low speeds. Some steppers are optimized for better high-speed performance, but they need to be paired with an appropriate driver to achieve that performance.

- No Feedback – Unlike servo motors, most steppers do not have integral feedback for position. Although great precision can be achieved running ‘open loop’. Limit switches or ‘home’ detectors are typically required for safety and/or to establish a reference position.

- Low Efficiency – Unlike DC motors, stepper motor current consumption is independent of load. They draw the most current when they are doing no work at all. Because of this, they tend to run hot.

- Limited High Speed Torque - In general, stepper motors have less torque at high speeds than at low speeds. Some steppers are optimized for better high-speed performance, but they need to be paired with an appropriate driver to achieve that performance.

- No Feedback – Unlike servo motors, most steppers do not have integral feedback for position. Although great precision can be achieved running ‘open loop’. Limit switches or ‘home’ detectors are typically required for safety and/or to establish a reference position.

- Low Efficiency – Unlike DC motors, stepper motor current consumption is independent of load. They draw the most current when they are doing no work at all. Because of this, they tend to run hot.

- Limited High Speed Torque - In general, stepper motors have less torque at high speeds than at low speeds. Some steppers are optimized for better high-speed performance, but they need to be paired with an appropriate driver to achieve that performance.

- No Feedback – Unlike servo motors, most steppers do not have integral feedback for position. Although great precision can be achieved running ‘open loop’. Limit switches or ‘home’ detectors are typically required for safety and/or to establish a reference position.

- Low Efficiency – Unlike DC motors, stepper motor current consumption is independent of load. They draw the most current when they are doing no work at all. Because of this, they tend to run hot.

- Limited High Speed Torque - In general, stepper motors have less torque at high speeds than at low speeds. Some steppers are optimized for better high-speed performance, but they need to be paired with an appropriate driver to achieve that performance.

- No Feedback – Unlike servo motors, most steppers do not have integral feedback for position. Although great precision can be achieved running 'open loop'. Limit switches or 'home' detectors are typically required for safety and/or to establish a reference position.

- Low Efficiency – Unlike DC motors, stepper motor current consumption is independent of load. They draw the most current when they are doing no work at all. Because of this, they tend to run hot.

- Limited High Speed Torque - In general, stepper motors have less torque at high speeds than at low speeds. Some steppers are optimized for better high-speed performance, but they need to be paired with an appropriate driver to achieve that performance.

- No Feedback – Unlike servo motors, most steppers do not have integral feedback for position. Although great precision can be achieved running 'open loop'. Limit switches or 'home' detectors are typically required for safety and/or to establish a reference position.

4.5.2 Types of Steppers

There are a wide variety of stepper types, some of which require very specialized drivers. For our purposes, we will focus on stepper motors that can be driven with commonly available drivers. These are: Permanent Magnet or Hybrid steppers, either 2-phase bipolar, or 4-phase unipolar[14].

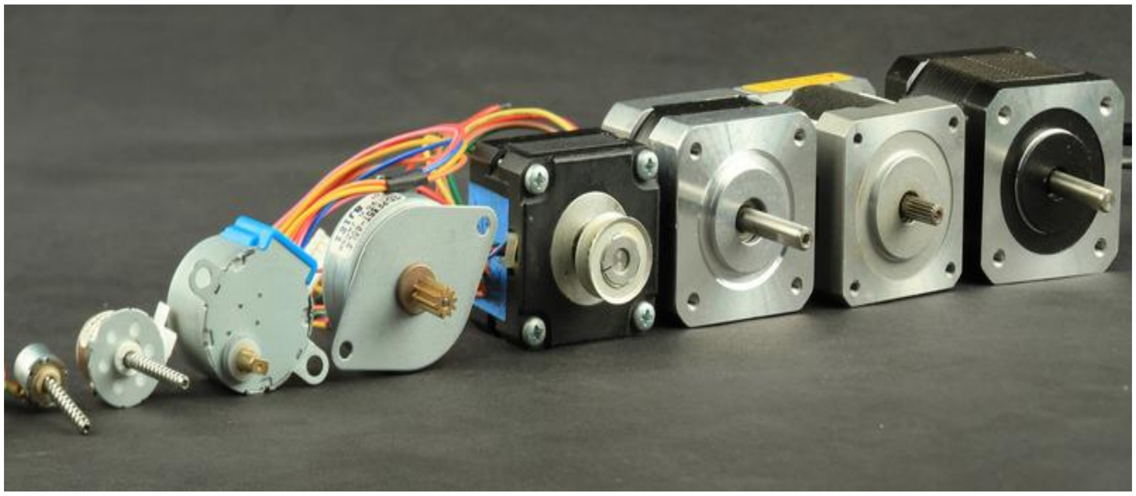


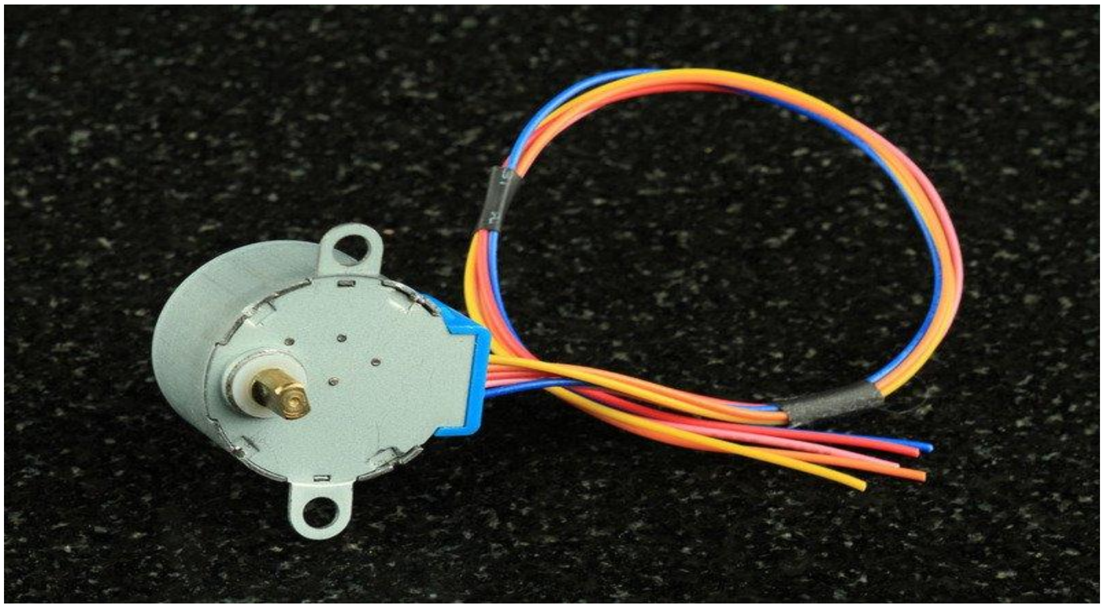
Figure 30 Stepper Motor

4.5.3 Motor Size

One of the first things to consider is the work that the motor has to do. As you might expect, larger motors are capable of delivering more power. Stepper motors come in sizes ranging from smaller than a peanut to big NEMA 57 monsters. Most motors have torque ratings. This is what you need to look at to decide if the motor has the strength to do what you want.NEMA 17 is a common size used in 3D printers and smaller CNC mills. Smaller motors find applications in many robotic and animatronic applications. The larger NEMA frames are common in CNC machines and industrial applications. The NEMA numbers define standard faceplate dimensions for mounting the motor. They do not define the other characteristics of a motor. Two different NEMA 17 motors may have entirely different electrical or mechanical specifications and are not necessarily interchangeable[14].

4.5.4 Step Count

The next thing to consider is the positioning resolution you require. The number of steps per revolution ranges from 4 to 400. Commonly available step counts are 24, 48 and 200. Resolution is often expressed as degrees per step. A 1.8° motor is the same as a 200 step/revolution motor.The trade-off for high resolution is speed and torque. High step count motors top-out at lower RPMs than similar size. And the higher step-rates needed to turn these motors results in lower torque than a similar size low-step-count motor at similar speeds.



4.5.5 Coils and Phases

A stepper motor may have any number of coils. But these are connected in groups called "phases". All the coils in a phase are energized together.

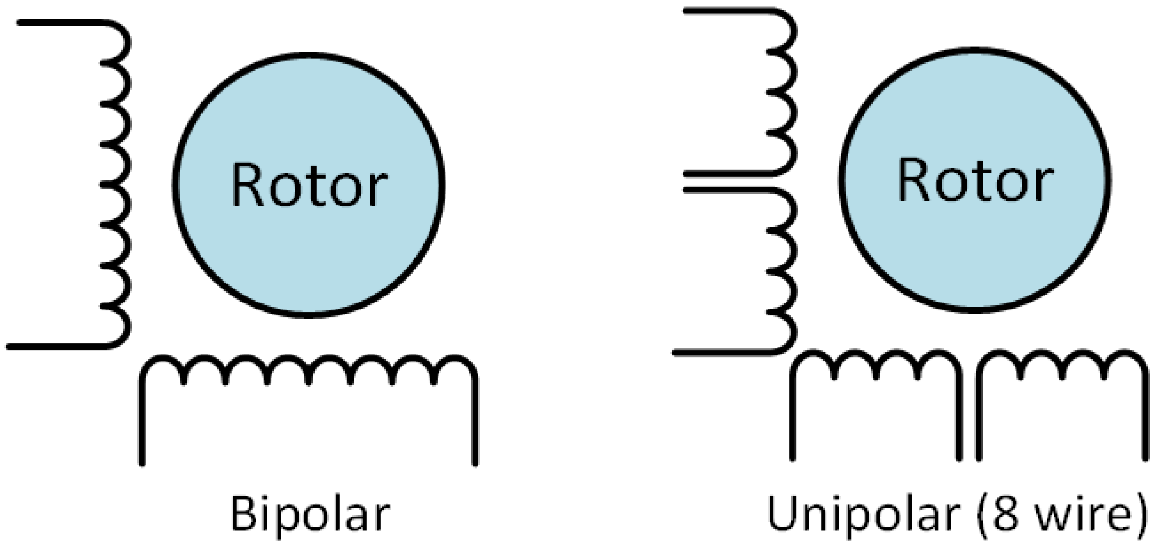


Figure 31 Bipolar and Unipolar

4.5.6 Unipolar vs. Bipolar

Unipolar drivers, always energize the phases in the same way. One lead, the "common" lead, will always be negative. The other lead will always be positive. Unipolar drivers can be implemented with simple transistor circuitry. The disadvantage is that there is less available torque because only half of the coils can be energized at a time. Bipolar drivers use H-bridge circuitry to actually reverse the current flow through the phases. By energizing the phases with alternating the polarity, all the coils can be put to work turning the motor. A two phase bipolar motor has 2 groups of coils. A 4 phase unipolar motor has 4. A 2-phase bipolar motor will have 4 wires - 2 for each phase. Some motors come with flexible wiring that allows you to run the motor as either bipolar or unipolar.

4.6 8051 Microcontroller

The first microprocessor 4004 was invented by Intel Corporation. 8085 and 8086 microprocessors were also invented by Intel. In 1981, Intel introduced an 8-bit microcontroller called the 8051. It was referred as system on a chip because it had 128 bytes of RAM, 4K byte of on-chip ROM, two timers, one serial port, and 4 ports (8-bit wide), all on a single chip. When it became widely popular, Intel allowed other manufacturers to make and market different flavors of 8051 with its code compatible with 8051. It means that if you write your program for one flavor of 8051, it will run on other flavors too, regardless of the manufacturer. This has led to several versions with different speeds and amounts of on-chip RAM[15].

4.6.1 Pin diagram:

8051 microcontroller is a 40 pin Dual Inline Package (DIP). These 40 pins serve different functions like read, write, I/O operations, interrupts etc. 8051 has four I/O ports wherein each port has 8 pins which can be configured as input or output depending upon the logic state of the pins. Therefore, 32 out of these 40 pins are dedicated to I/O ports. The rest of the pins are dedicated to VCC, GND, XTAL1, XTAL2, RST, ALE, EA' and PSEN'[15].

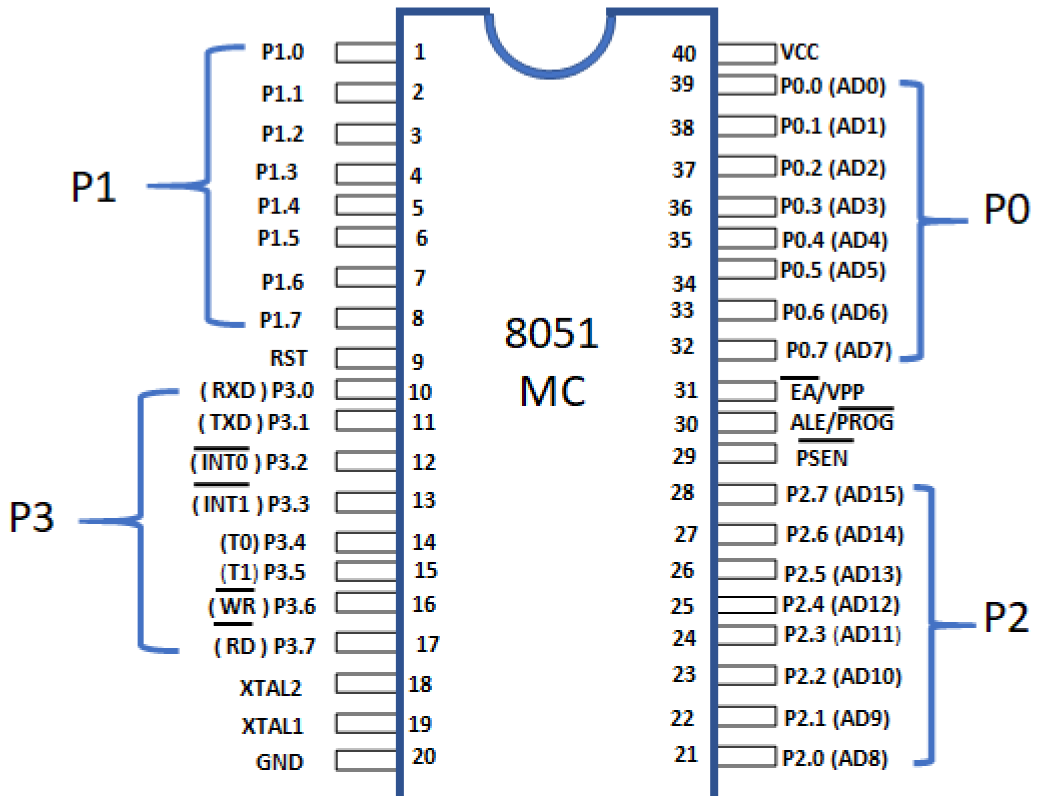


Figure 32 Pin Diagram Of 8051

4.6.2 Description of the Pins :

1. Pin 1 to Pin 8 (Port 1)

Pin 1 to Pin 8 are assigned to Port 1 for simple I/O operations. They can be configured as input or output pins depending on the logic control i.e. if logic zero (0) is applied to the I/O port it will act as an output pin and if logic one (1) is applied the pin will act as an input pin. These pins are also referred to as P1.0 to P1.7 (where P1 indicates that it is a pin in port 1 and the number after '.'

tells the pin number i.e. 0 indicates first pin of the port. So, P1.0 means first pin of port 1, P1.1 means second pin of the port 1 and so on). These pins are bidirectional pins.

2. Pin 9 (RST)

Reset pin. It is an active-high, input pin. Therefore if the RST pin is high for a minimum of 2 machine cycles, the microcontroller will reset i.e. it will close and terminate all activities. It is often referred as “power-on-reset” pin because it is used to reset the microcontroller to it’s initial values when power is on (high).

3. Pin 10 to Pin 17 (Port 3)

Pin 10 to pin 17 are port 3 pins which are also referred to as P3.0 to P3.7. These pins are similar to port 1 and can be used as universal input or output pins. These pins are bidirectional pins.

These pins also have some additional functions which are as follows:

4. P3.0 (RXD)

10th pin is RXD (serial data receive pin) which is for serial input. Through this input signal microcontroller receives data for serial communication.

5. P3.1 (TXD)

11th pin is TXD (serial data transmit pin) which is serial output pin. Through this output signal microcontroller transmits data for serial communication.

6. P3.2 and P3.3 (INT0’, INT1’)

12th and 13th pins are for External Hardware Interrupt 0 and Interrupt 1 respectively. When this interrupt is activated(i.e. when it is low), 8051 gets interrupted in whatever it is doing and jumps to the vector value of the interrupt (0003H for INT0 and 0013H for INT1) and starts performing Interrupt Service Routine (ISR) from that vector location[15].

7. P3.4 and P3.5 (T0 and T1)

14th and 15th pin are for Timer 0 and Timer 1 external input. They can be connected with 16 bit timer/counter.

8. P3.6 (WR’)

16th pin is for external memory write i.e. writing data to the external memory.

9. P3.7 (RD’)

17th pin is for external memory read i.e. reading data from external memory.

10. Pin 18 and Pin 19 (XTAL2 And XTAL1)

These pins are connected to an external oscillator which is generally a quartz crystal oscillator. They are used to provide an external clock frequency of 4MHz to 30MHz.

11. Pin 20 (GND)

This pin is connected to the ground. It has to be provided with 0V power supply. Hence it is connected to the negative terminal of the power supply.

12. Pin 21 to Pin 28 (Port 2)

Pin 21 to pin 28 are port 2 pins also referred to as P2.0 to P2.7. When additional external memory is interfaced with the 8051 microcontroller, pins of port 2 act as higher-order address bytes. These pins are bidirectional.

13. Pin 29 (PSEN)

PSEN stands for Program Store Enable. It is output, active-low pin. This is used to read external memory. In 8031 based system where external ROM holds the program code, this pin is connected to the OE pin of the ROM[15].

14. Pin 30 (ALE/ PROG)

stands for Address Latch Enable. It is input, active-high pin. This pin is used to distinguish between memory chips when multiple memory chips are used. It is also used to de-multiplex the multiplexed address and data signals available at port 0.

During flash programming i.e. Programming of EPROM, this pin acts as program pulse input (PROG).

15. Pin 31 (EA/ VPP)

EA stands for External Access input. It is used to enable/disable external memory interfacing. In 8051, EA is connected to Vcc as it comes with on-chip ROM to store programs. For other family members such as 8031 and 8032 in which there is no on-chip ROM, the EA pin is connected to the GND.

16. Pin 32 to Pin 39 (Port 0)

Pin 32 to pin 39 are port 0 pins also referred to as P0.0 to P0.7. They are bidirectional input/output pins. They don’t have any internal pull-ups. Hence, 10 K? pull-up registers are used as external pull-ups. Port 0 is also designated as AD0-AD7 because 8051 multiplexes address and data through port 0 to save pins.

17. Pin 40 (VCC)

This pin provides power supply voltage i.e. +5 Volts to the circuit.

4.6.3 Block-Diagram-of-the-Intel-8051-Microcontroller:

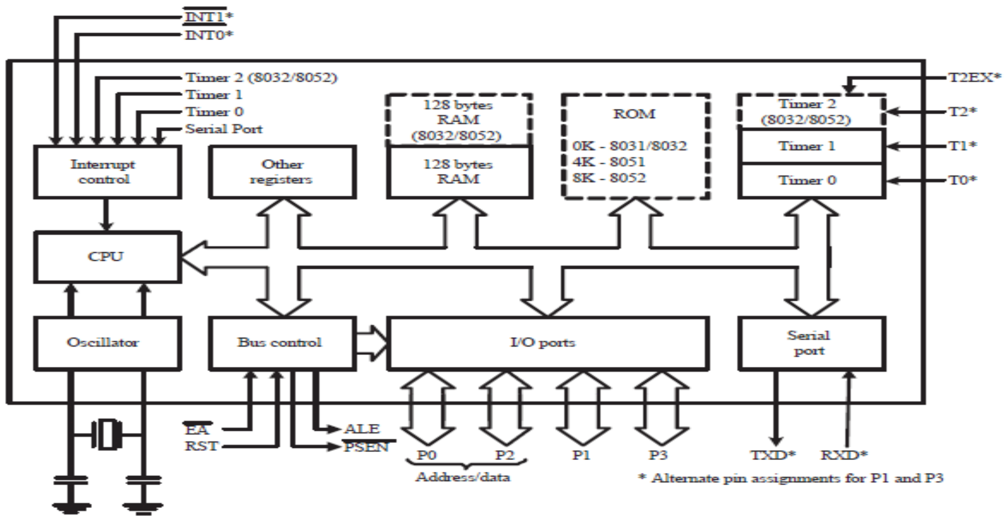


Figure 33 Block-Diagram-of-the-Intel-8051-Microcontroller

4.6.4 Comparison between 8051 Family Members

Table 7:compares the features available in 8051, 8052, and 8031.

Feature	8051	8052	8031
ROM(bytes)	4K	8K	0K
RAM(bytes)	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

4.6.5 Features of 8051 Microcontroller

An 8051 microcontroller comes bundled with the following features –

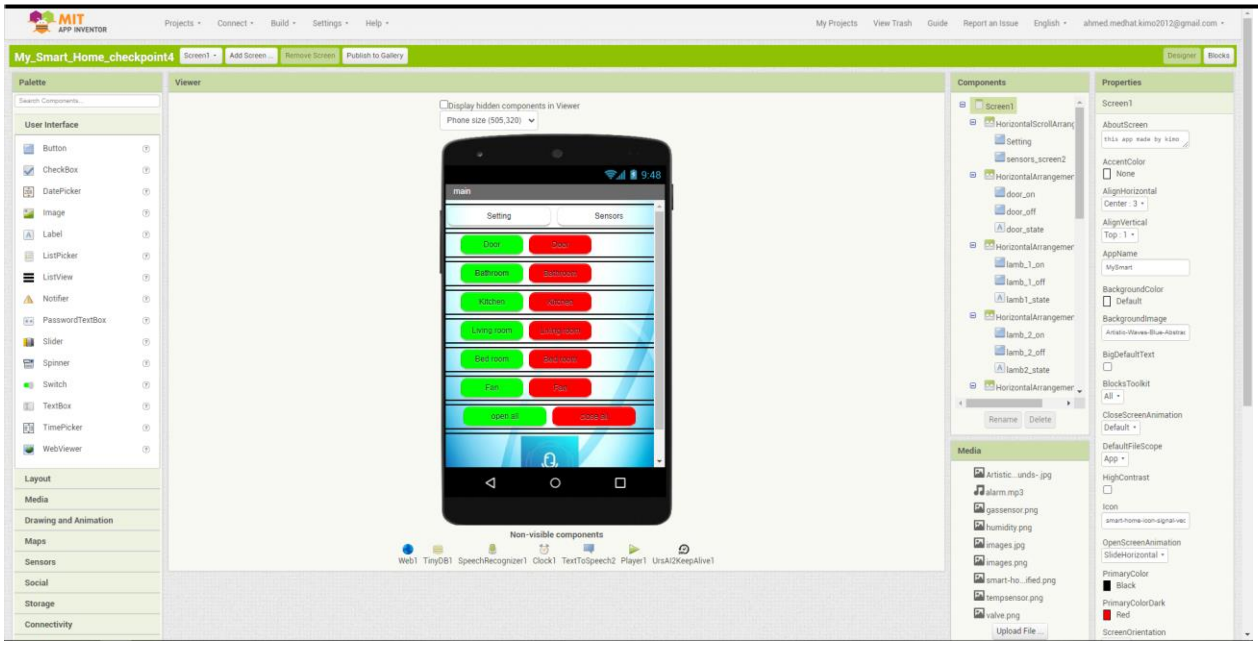
- 4KB bytes on-chip program memory (ROM)
- 128 bytes on-chip data memory (RAM)
- Four register banks
- 128 user defined software flags
- 8-bit bidirectional data bus
- 16-bit unidirectional address bus
- 32 general purpose registers each of 8-bit
- 16 bit Timers (usually 2, but may have more or less)
- Three internal and two external Interrupts
- Four 8-bit ports,(short model have two 8-bit ports)
- 16-bit program counter and data pointer
- 8051 may also have a number of special features such as UARTs, ADC, Op-amp, etc.

5 Mobile app (MIT App Inventor)

The smartphone is an information nexus in today's digital age, with access to a nearly infinite supply of content on the web, coupled with rich sensors and personal data. However, people have difficulty harnessing the full power of these ubiquitous devices for themselves and their communities. Most smartphone users consume technology without being able to produce it, even though local problems can often be solved with mobile devices. How then might they learn to leverage smartphone capabilities to solve real-world, everyday problems? MIT App Inventor is designed to democratize this technology and is used as a tool for learning computational thinking in a variety of educational contexts, teaching people to build apps to solve problems in their communities. MIT App Inventor is an online development platform that anyone can leverage to solve real-world problems. It provides a web-based "What you see is what you get" (WYSIWYG) editor for building mobile phone applications targeting the Android and iOS operating systems. It uses a block-based programming language built on Google Blockly [16] and inspired by languages such as StarLogo TNG [16] empowering anyone to build a mobile phone app to meet a need. To date, 6.8 million people in over 190 countries have used App Inventor to build over 24 million apps. We offer the interface in more than a dozen languages. People around the world use App Inventor to provide mobile solutions to real problems in their families, communities, and the world. The platform has also been adapted to serve requirements of more specific populations, such as building apps for emergency/first responders and robotics [16] In this chapter, we describe the goals of MIT App Inventor and how they have influenced our design and development—from the program's inception at Google in 2008, through the migration to MIT, to the present day. We discuss the pedagogical value of MIT App Inventor and its use as a tool to teach and encourage people of all ages to think and act computationally. We also describe three applications developed by students in different parts of the world to solve real issues in their communities. We conclude by discussing the limitations and benefits of tools such as App Inventor and proposing new directions for research.

5.1 MIT App Inventor Overview

The MIT App Inventor user interface includes two main editors: the design editor and the blocks editor. The design editor, or designer (see Fig. 3.1), is a drag and drop interface to lay out the elements of the application's user interface (UI). The blocks editor (see Fig. 3.2) is an environment in which app inventors can visually lay out the logic of their apps using color-coded blocks that snap together like puzzle pieces to describe the program. To aid in development and testing, App Inventor provides a mobile app called the App Inventor Companion (or just "the Companion") that developers can use to test and adjust the behavior of their apps in real time. In this way, anyone can quickly build a mobile app and immediately begin to iterate and test

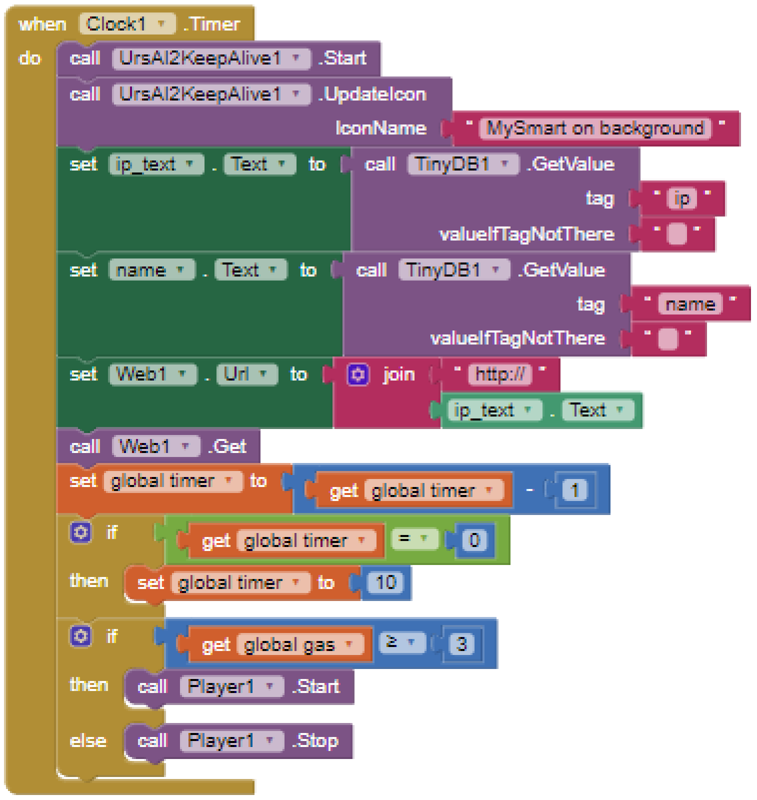


AboutScreen: this app made by kimo
AccentColor: None
AlignHorizontal: Center : 3
AlignVertical: Top : 1
AppName: MySmart
BackgroundColor: Default
BackgroundImage: Artistic-Waves-Blue-Abstrac
BlocksToolkit: All
CloseScreenAnimation: Default
DefaultFileScope: App
Icon: smart-home-icon-signal-vec
OpenScreenAnimation: SlideHorizontal
PrimaryColor: Black
PrimaryColorDark: Red

Figure 34 MIT App Inventor

5.2 MIT App Inventor Design Goals

In the design of MIT App Inventor, introducing mobile app development in educational contexts was a central goal. Prior to its release, most development environments for mobile applications were clunky, only accessible with expertise in systems level or embedded programming, or both. Even with Google's Android operating system and the Java programming language, designing the user interface was a complex task. Further, use of the platform required familiarity with Java syntax and semantics, and the ability to debug Java compilation errors (e.g., misspelled variables or misplaced semicolons) for success. These challenges presented barriers to entry for individuals not versed in computer science, App Inventor's target demographic. We briefly highlight and discuss design goals for the App Inventor project, specifically, the use of components to abstract some of the complexity of platform behavior, and the use of blocks to eliminate complexity of the underlying programming language. These goals can be further explained as aligning the visual language to the mental models of young developers and enabling exploration through fast, iterative design.



```
when Clock1 .Timer
do  call UrsAI2KeepAlive1 .Start
    call UrsAI2KeepAlive1 .UpdateIcon
                          IconName " MySmart on background "
    set ip_text . Text to   call TinyDB1 .GetValue
                                          tag " ip "
                            valueIfTagNotThere " "
    set name . Text to   call TinyDB1 .GetValue
                                       tag " name "
                         valueIfTagNotThere " "
    set Web1 . Url to   join " http:// "
                             ip_text . Text
    call Web1 .Get
    set global timer to   get global timer - 1
    if   get global timer = 0
    then set global timer to 10
    if   get global gas ≥ 3
    then call Player1 .Start
    else call Player1 .Stop
```

Figure 35 MIT App Inventor Design

5.2.1 Component Abstraction for Platform Behavior

Components are core abstractions in MIT App Inventor. Components reduce the complexity of managing interactions with platform-specific application programming interfaces (APIs) and details concerning state management of device hardware. This allows the user to think about the problem at hand rather than the minutia typically required of application developers. For example, someone planning to use MIT App Inventor to build an app to use the global positioning system (GPS) to track movement need not be concerned with application lifecycle management, GPS software and hardware locks, or network connectivity (in case location detection falls back to network-based location). Instead, the app developer adds a location sensor component that abstracts away this complexity and provides an API for enabling and processing location updates. More concretely, this implementation reduces 629 lines of Java code to 23 blocks, of which only two are required to accomplish location tracking. This reduction in complexity enables app inventors to focus on the problem at hand and quickly accomplish a goal. Components are made up of three major elements: properties, methods, and events. Properties control the state of the component and are readable and/or writable by the app developer. For example, the enabled property of the location sensor includes the functionality required to configure the GPS receiver and to manage its state while the app is in use. Methods operate on multiple inputs and possibly return a result. Events respond to changes in the device or app state based on external factors. For example, when the app user changes their location, the location changed event allows the app logic to respond to the change.

5.2.2 Blocks as Logic

In MIT App Inventor, users code application behavior using a block-based programming language. There are two types of blocks in App Inventor: built-in blocks and component blocks. The built-in blocks library provides the basic atoms and operations generally available in other programming languages, such as Booleans, strings, numbers, lists, mathematical operators, comparison operators, and control flow operators. Developers use component blocks (properties, methods, and events) to respond to system and user events, interact with device hardware, and adjust the visual and behavioral aspects of components.

5.2.2.1 Top-Level Blocks

All program logic is built on three top-level block types: global variable definitions, procedure definitions, and component event handlers. Global variables provide named slots for storing program states. Procedures define common behaviors that can be called from multiple places in the code. When an event occurs on the device, it triggers the corresponding application behavior prescribed in the event block. The event handler block may reference global variables or procedures. By limiting the top-level block types, there are fewer entities to reason about

5.2.3 Mental Modeling

The development team for App Inventor considered a number of restrictions when designing the environment. We examine a few design decisions, the rationale behind them, and their effects on computational thinking within App Inventor.

5.2.3.1 What You See Is What You Get (WYSIWYG)

The design editor for App Inventor allows developers to see how the app will appear on the device screen and adjust the form factor of the visualized device (e.g., phone or tablet). Adjustments to properties of the visual components, for example, background color and size, are reflected in real time. Apps can also be run in a live development mode using the Companion, which we will be discussed in more detail below The App Inventor team recently added capability for creating map-based applications. The functionality allows app inventors to drag, drop, and edit markers, lines, polygons, rectangles, and circles in their maps, as well as integrate web-based data from geographic information systems (GIS) to build content-rich apps. This way, the user can move the content around easily to achieve great results without needing to provide most of the logic for this in code.

5.2.3.2 .Design Time Component Creation

Unlike many programming languages, App Inventor limits runtime creation of new entities. This provides multiple benefits. First, by explicitly positioning all components in the app, the user can visualize it clearly rather than having to reason about things that will not exist until a future time. Second, it reduces the chances of users introducing cyclic memory dependencies in the user interface that would eventually cause the app to run out of memory. This encourages app inventors to think about how to appropriately structure their applications and reuse components to avoid overloading the system or their end users

5.2.3.3 Natural Numbering

The number system in App Inventor assumes a starting value of 1, in line with children's counting skills[[16] .This is unlike most programming languages, which are more aligned with machine architecture and therefore start at 0.

5.2.4 Fast Iteration and Design Using the Companion

A key feature of MIT App Inventor is its live development environment for mobile applications. App Inventor provides this by means of a companion app installed on the user's mobile device. The App Inventor web interface sends code to the companion app, which interprets the code and displays the app in real time to the developer (Fig. 3.3). This way, the user can change the app's interface and behavior in real time. For example, a student making a game involving the ball component may want to bounce the ball off the edge of the play area. However, an initial implementation might have the ball collide with the wall and then stop. After discovering the Ball.EdgeReached event, the student can add the event and update the direction of the ball using the Ball.Bounce method. By testing the app and adjusting its programming in response to undesired behavior, students can explore more freely. The traditional build cycle for an Android app involves writing code in a text editor or integrated development environment, and rebuilding the application for testing may often take minutes, whereas making a change in the live development

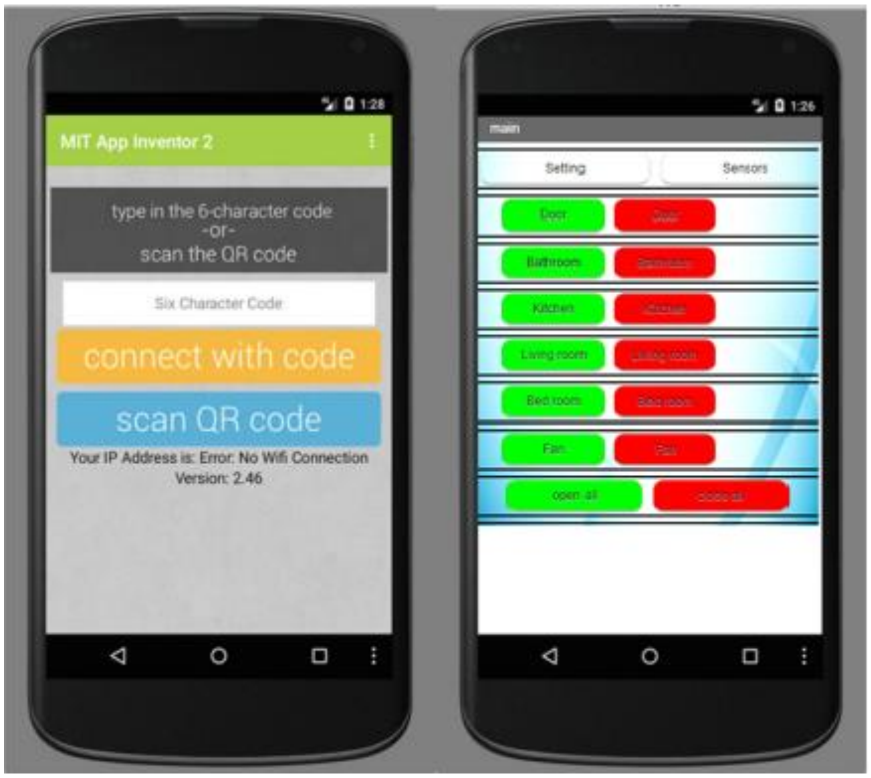


Figure 36 Fast Iteration and Design

environment typically takes effect in 1–2 s. Seeing changes reflected in the app quickly means that students can explore and even make mistakes while exploring, because the time cost of those mistakes is relatively small.

5.3 The History of MIT App Inventor

The App Inventor project began at Google in 2007 when Prof. Hal Abelson of MIT went on sabbatical at Google Labs. The project leads were inspired by increased interest in educational blocks programming languages, such as Scratch, and the release of the new Android operating system. This educational project was migrated to MIT when Google closed Google Labs in 2011. In this section, we briefly cover inception and early development of the App Inventor platform, first at Google, and then at MIT.

5.3.1 Inception at Google

Hal Abelson conceived the idea of App Inventor while on sabbatical at Google Labs in 2007. Abelson had previously taught a course at MIT on mobile programming, but at the time mobile app development required significant investment on the part of developers and development environments. Also in 2007, Google publicly announced the Android operating system. Abelson and Mark Friedman of Google began developing an intermediate language between the blocks language and Java APIs for Android, called Yet Another Intermediate Language (YAIL). The project was intended to help younger learners program for Android. Abelson and Friedman generated YAIL from a block-based language based on OpenBlocks [16], and the design of which was drawn from StarLogo TNG[[16] .The user interface and related components embodied Papert’s idea of “powerful ideas in mind-size bites” [16] The Google version of the project terminated at the end of 2011, but the educational technology was transferred to MIT so that development and educational aspects could continue [16]. Prof. Abelson joined Prof. Eric Klopfer of the Scheller Teacher Education Program lab and Prof. Mitch Resnick of the MIT Media Lab, forming a group called the MIT Center for Mobile Learning to carry on the App Inventor vision.

5.3.2 Educational Expansion at MIT

In late 2011, Google transferred stewardship of the App Inventor project to MIT. Much of the development focused on increasing capabilities to support educational goals of the project. At this time, the team developed additional curricula, making them freely available to teachers for computer science and computational thinking education. The MIT team also hosted a number of 1-day workshops, primarily around the northeast United States, training teachers in the pedagogy of App Inventor. We now focus on guided and open exploration in our materials rather than presenting students with step-by-step instructions in order to encourage self-guided learning. By making mistakes, students have the opportunity to practice more of the computational thinking principles, such as debugging,[16]. Technical development at MIT focused on development of new components including robotics (LEGO™ EV3), cloud-oriented data storage (CloudDB), and geographic visualization (Map). App Inventor team also developed Internet of Things related extensions so learners could interact with physical hardware external to their mobile devices, and to leverage the growing collection of small computer boards, such as Arduino, BBC micro:bit, and Raspberry Pi.

To this day, the team continues its work of development, creating complementary educational materials in parallel.

5.4 MIT App Inventor in Education

The primary aim of MIT App Inventor is providing anyone with an interest in building apps to solve problems with the tools necessary to do so. Instructional materials developed by the team are primarily oriented toward teachers and students at the middle- and high-school levels, but app inventors come in all ages from around the world. In this section, we describe a few of the key components of the MIT App Inventor educational strategy, including massively online open courses (MOOCs) focused on MIT App Inventor, the Master Trainer (MT) program, the extensions functionality of App Inventor that allows incorporation of new material for education, and research projects that have leveraged App Inventor as a platform for enabling domain-specific computing.

5.4.1 Massive Open Online Courses

A desire to learn computational thinking has driven a proliferation of online educational material that anyone can access to increase their knowledge and understanding. As we continue to integrate information technology into our daily lives, mobile devices, and other new technologies, we can observe that a deeper understanding of computing is necessary to be an effective member of society, and those who learn computational thinking will have an advantage in our knowledge-driven economy. Many massive open online courses have been developed wholely or in part using App Inventor. For example, an App Inventor EdX course closely integrates with the AP CS Principles course and incorporates many computational thinking elements. Students therefore can both build their own mobile apps and learn core competencies related to computation.

5.4.2 MIT Master Trainers Program

MIT provides special instruction to educators through the Master Trainers program.1 A prototype of the Master Trainers program began during a collaboration with the Verizon App Challenge in 2012. Skilled App Inventor educators were recruited and given a small amount of special training to help mentor and train teams who subsequently won the App Challenge. The current Master Trainers program was conceived in 2015, to "grow a global community of experts on mobile app development who are available to guide others through the exploration of mobile app creation…, thus providing a pathway into computer science, software development, and other disciplines relevant in today's digital world."In order to become a Master Trainer, one must demonstrate proficiency in App Inventor, for example, through taking the App Inventor EdX MOOC. The MOOC is highly integrated with computational thinking concepts, giving students a strong foundation in the concepts and practices associated with computational thinking. Aspiring

Master Trainers then complete a 10-week online reading course covering topics such as App Inventor's mission and philosophy, pedagogy of teaching children and adults, constructionism, and design thinking. Lastly, there is an on-site 3-day workshop at MIT where participants dive into App Inventor features and learn to use App Inventor in a classroom to foster creativity, collaboration, and problem-solving. At the time of writing, there were 57 master trainers in 19 countries.

5.4.3 Extensions

Anyone with Java and Android programming experience can write their own components for App Inventor using our extension mechanism. For example, MIT recently published a suite of Internet of things (IOT)-related extensions2 for interfacing with Arduino 101 and BBC micro:bit microcontrollers, with support for other platforms in development. Using these extensions, teachers can assemble custom curricula to leverage these technologies in the classroom and encourage their students to explore the interface between the world of software and the world of hardware. We foresee the development of extensions related to artificial intelligence technologies, including deep learning, device support for image recognition, sentiment analysis, natural language processing, and more. Ideally, these complex technologies could be leveraged by anyone looking to solve a problem with the smartphone as a platform.

5.4.4 Research Projects

In addition to its pedagogical applications, App Inventor offers excellent opportunities for research in education and other areas. Early work focused on understanding how to appropriately name components for educational use [16]. Usability in domain-specific contexts, such as humanitarian needs and educational settings [16], is also an area of interest. More recently, App Inventor has been used as a mechanism for data collection and visualization [16]. We are currently exploring expanding App Inventor's capabilities to include real-time collaboration between students, which should yield additional educational opportunities [16].

5.5 Empowerment Through Programming

By placing the output of student programming on mobile devices, App Inventor allows students to move their work out of traditional computer labs, and into their everyday lives and communities. This transition has powerful implications for what students create and how they envision themselves as digital creators. It allows students to shift their sense of themselves from individuals who "know how to code" to members of a community empowered to have a real impact in their lives and those of others. Below, we outline how App Inventor moves computing education from a focus on the theoretical to a focus on the practical, how we can reconceptualize computing education through a lens of computational action, and how we support students to engage in a broader community of digitally empowered creators.

5.5.1 From Theoretical to Practical

Traditional computer science curricula at the university level often focus on theory and include evaluation tools (e.g., Big-O notation of algorithms) and comprehension of the space and time complexity of data structures. Instead, App Inventor curricula focus on using a language practically to solve real-world problems. Rather than placing emphasis on learning concepts such as linked lists or key–value mappings, App Inventor hides the complexity of these data structures behind blocks so that students can spend more time designing apps that perform data collection and analysis, or integrate with a range of sensors and actuators interacting with external environments. This allows for a top-down, goal-based decomposition of the problem rather than a bottom-up approach, although App Inventor does not preclude such a strategy.

5.5.2 Computational Thinking

The concept of computational thinking was first used by Seymour Papert in his seminal book Mindstorms: Children, computers, and powerful ideas [16] however, it was largely brought into the mainstream consciousness by Jeannette Wing in 2006. For Wing, computational thinking is the ability to think like a computer scientist. In the decade since, many educational researchers have worked to integrate computational thinking into modern computing and STEM curricula [16]. However, the explosive growth of computational thinking has also resulted in a fragmentation of its meaning, with educational researchers, curriculum designers, and teachers using different definitions, educational approaches, and methods of assessments [16]. There have been attempts to reconcile these differences [16] and to bring leading researchers together to compare and contrast these perspectives [16]. For most educational practitioners and researchers, computational thinking is dominated by an epistemological focus on computational thinking, in which students learn programming concepts (such as loops, variables, and data handling) and the use of abstractions to formally represent relationships between computing and objects in the real world [16]. While this view has become the most prominent view of computational thinking, Papert critiqued mainstream schooling's emphasis on these "skills and facts" as a bias against ideas [16]. Papert went further, arguing that students should be encouraged to follow their own projects and that learning the necessary skills and knowledge would arise as students encountered new problems and needed to solve (or not solve) them. This position of computational thinking and computing education fits more naturally with the ways that professionals engage in computer science: in pursuit of finishing a project, problems naturally come up and computer scientists reach out to the community through sites like Stack Overflow, or search the web for tutorials or other support. This disconnect between how we teach computing and how it is practiced in the real world requires us to critically reexamine theoretical and practical approaches. Below, we argue for an approach to computing education, termed computational action, that we believe matches these broader ideals.

5.5.3 Computational Action

While the growth of computational thinking has brought new awareness to the importance of computing education, it has also created new challenges. Many educational initiatives focus solely on the programming aspects, such as variables, loops, conditionals, parallelism, operators, and data handling [16], divorcing computing from real-world contexts and applications. This decontextualization threatens to make learners believe that they do not need to learn computing, as they cannot envision a future in which they will need to use it, just as many see math and physics education as unnecessary [16]. This decontextualization of computing education from the actual lives of students is particularly problematic for students underrepresented in the fields of computing and engineering, such as women and other learners from nondominant groups. For these students, there is a need for their work to have an impact in their community and for it to help them develop a sense of fit and belonging [16]. [16] argue that a critical perspective for computing is essential for students to develop a critical consciousness around what they are learning and making, moving beyond simply programming, instead of asking the students what they are programming and why they are programming it. In response, the App Inventor team advocates for a new approach to computing education that we call computational action. The computational action perspective on computing argues that while learning about computing, young people should also have opportunities to create with computing which have direct impact on their lives and their communities. Through our work with App Inventor, we have developed two key dimensions for understanding and developing educational experiences that support students in engaging in computational action: (1) computational identity and (2) digital empowerment. Computational identity builds on prior research that showed the importance of young people’s development of scientific identity for future STEM growth [16]. We define computational identity as a person’s recognition that they can use computing to create change in their lives and potentially find a place in the larger community of computational problem-solvers. Digital empowerment involves instilling in them the belief that they can put their computational identity into action in authentic and meaningful ways. Computational action shares characteristics with other approaches for refocusing computing education toward student-driven problem-solving, most notably computational participation[16]. Both computational action and computational participation recognize the importance of creating artifacts that can be used by others. However, there is a slight distinction between the conceptualizations of community in the two approaches. In computational participation, community largely means the broader community of learners engaging in similar computing practices (e.g., the community of Scratch programmers that share, reuse, and remix their apps). While such a learning community may be very beneficial to learners taking part in a computational action curriculum, the community of greater importance is the one that uses or is impacted by the learners’ created products (e.g., their family, friends, and neighbors). This computational identity element of computational action acknowledges the importance of learners feeling a part of a computing community (i.e., those that build and solve problems with computing), but it is not a requirement that they actively engage with this larger community. A small group of young app builders, such as those described below, may develop significant applications and believe they are authentically part of the computing community, without having connected with or engaged with it in a deep or sustained way as would be expected in computational participation. Through students' use of App Inventor, we have seen this computational action approach produce amazing results. Students in the United States have developed apps to help a blind classmate navigate their school (Hello Navi3); students in Moldova developed an app to help people in their country crowdsource clean drinking water (Apa Pura4); and as part of the CoolThink@JC project, students in Hong Kong created an app, "Elderly Guardian Alarm," to help the elderly when they got lost. Across these projects, we see students engaging with and facilitating change in their communities, while simultaneously developing computational identities.

5.5.4 Supporting a Community Around Computation and App Creation

We started the App of the Month program in 2015 in order to encourage App Inventors to share their work with the community. Any user can submit their app to be judged in one of four categories: Most Creative, Best Design, Most Innovative, and Inventor. Submissions must be App Inventor Gallery links, so that any user can remix winning apps. Furthermore, apps are judged in two divisions: youth and adult. Now, 3 years after the program's inception, approximately 40 apps are submitted each month. More youth tend to submit than adults, and significantly more male users submit than female users, especially in the adult division. While submissions come in from all over the world, India and the USA are most highly represented. Themes of submitted apps vary widely. Many students submit "all-in-one" apps utilizing the Text to Speech and Speech Recognizer components. Adults often submit learning apps for small children. Classic games, such as Pong, also get submitted quite frequently. Teachers tend to submit apps that they use in their classrooms. Perhaps most importantly, students and adults alike submit apps designed to solve problems within their own lives or their communities. For example, a recent submitter noticed that the Greek bus system is subject to many slowdowns, so he built an app that tracks buses and their routes. Similarly, a student noticed that many of her peers were interested in reading books, but did not know how to find books they would like, so she built an app that categorizes and suggests popular books based on the Goodreads website. However, not all users fit the same mold. One student found that he enjoys logic and math-based games, and after submitting regularly for about a year, his skill improved tremendously. Hundreds of people have remixed his apps from the Gallery, and even downloaded them from the Google Play Store, encouraging the student to pursue a full-time career in game development. The App of the Month program, as a whole, encourages users to think of App Inventor as a tool they can use in their daily lives and off-the-screen communities. It also provides incentive to share their apps and recognition for their hard work. Users go to App Inventor to solve problems—which makes them App Inventors themselves.

5.6 Application Implementation

This photo is our gui for our user

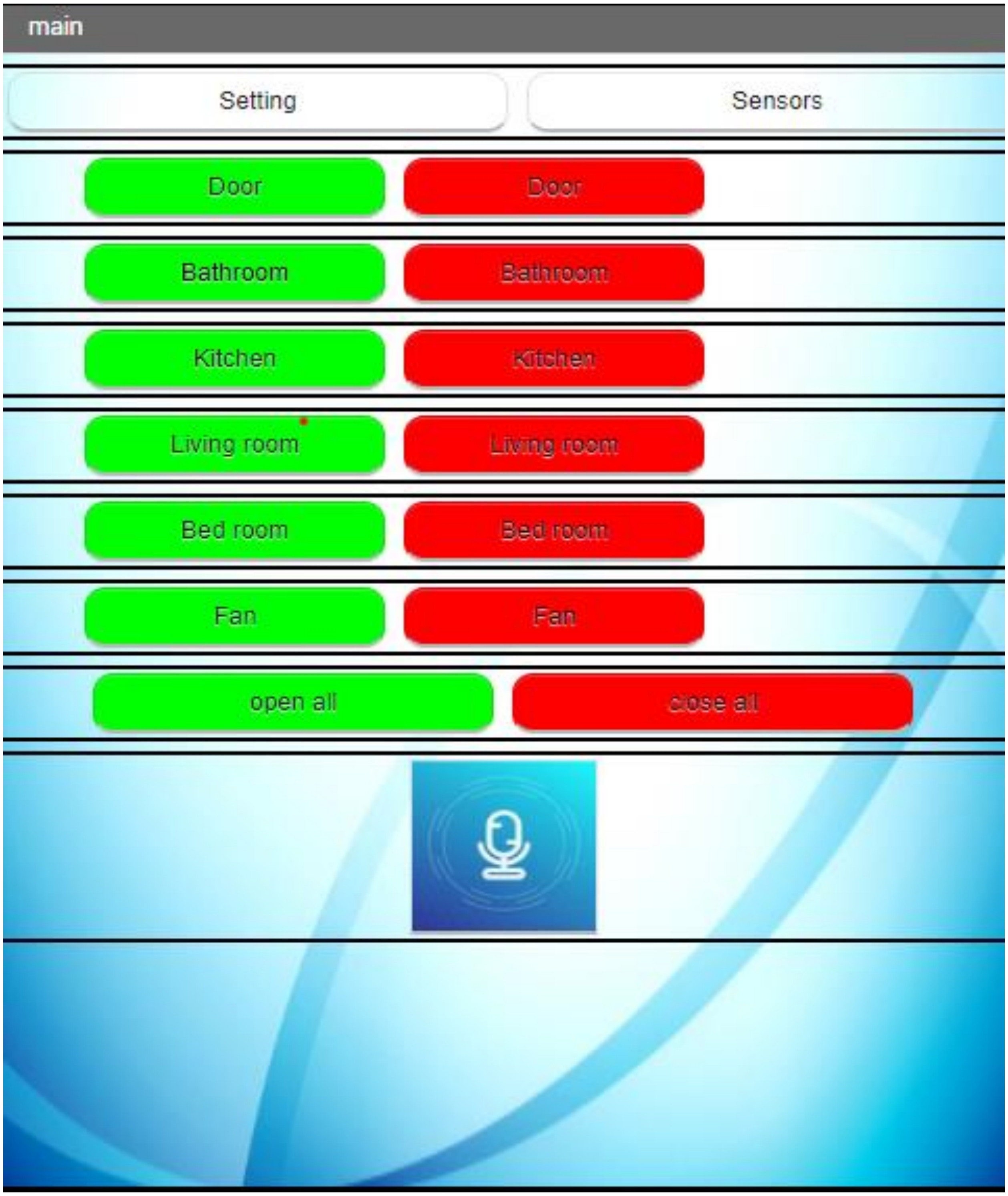


Figure 37 Home Page Of Application

5.6.1 Setting button

Sitting button will open a new page that have some setting user should fill before use this app
local or public ip
and user name
operation temperature value for fan
and make fan run automatically or manually

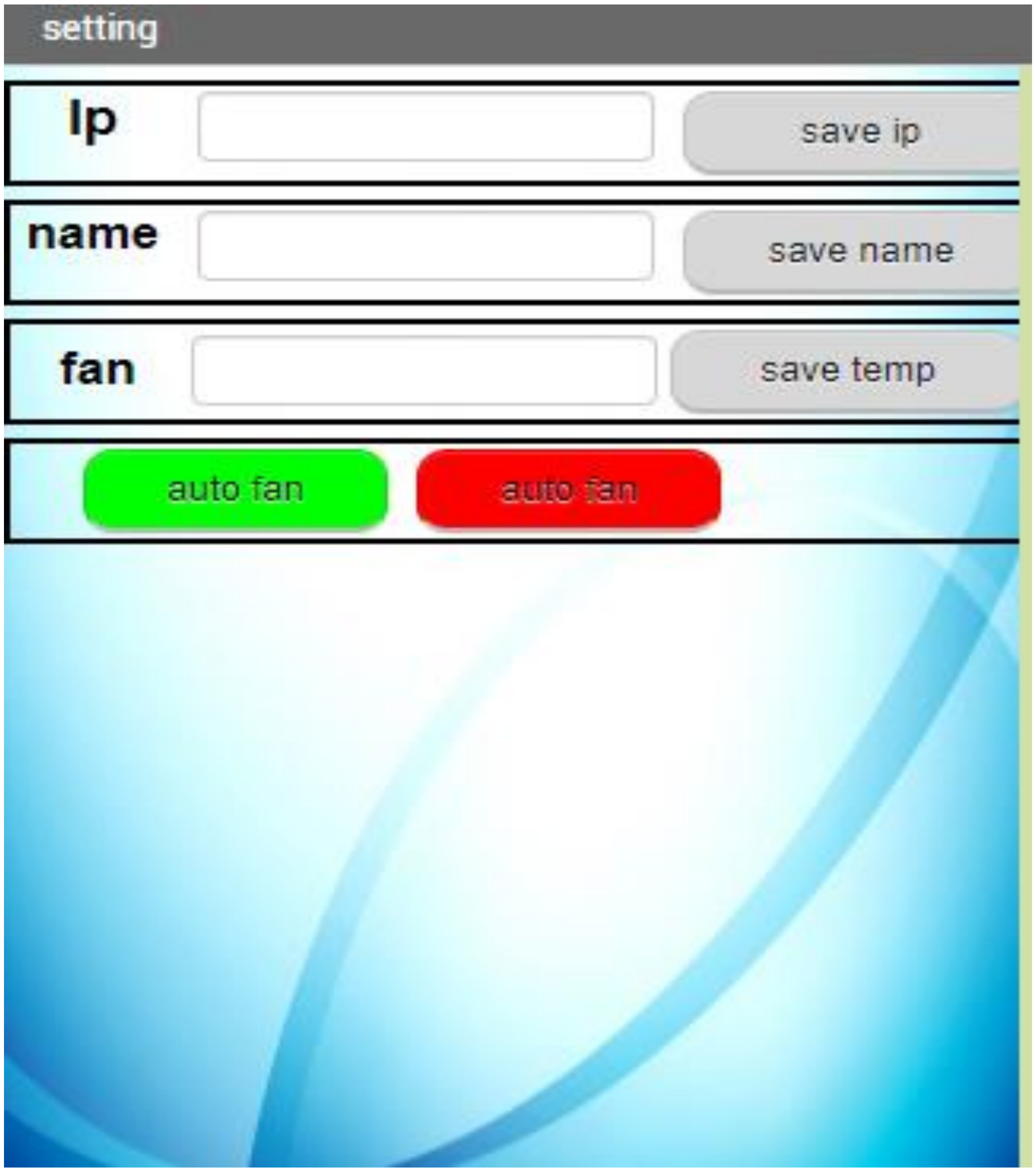
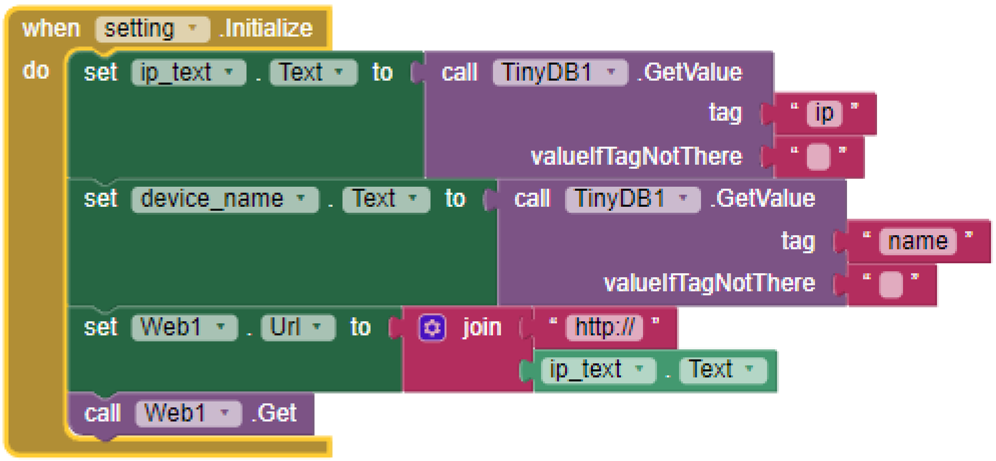


Figure 38 Main Data

When we this open page we get the saved ip and our user name and get data for raspberry pi

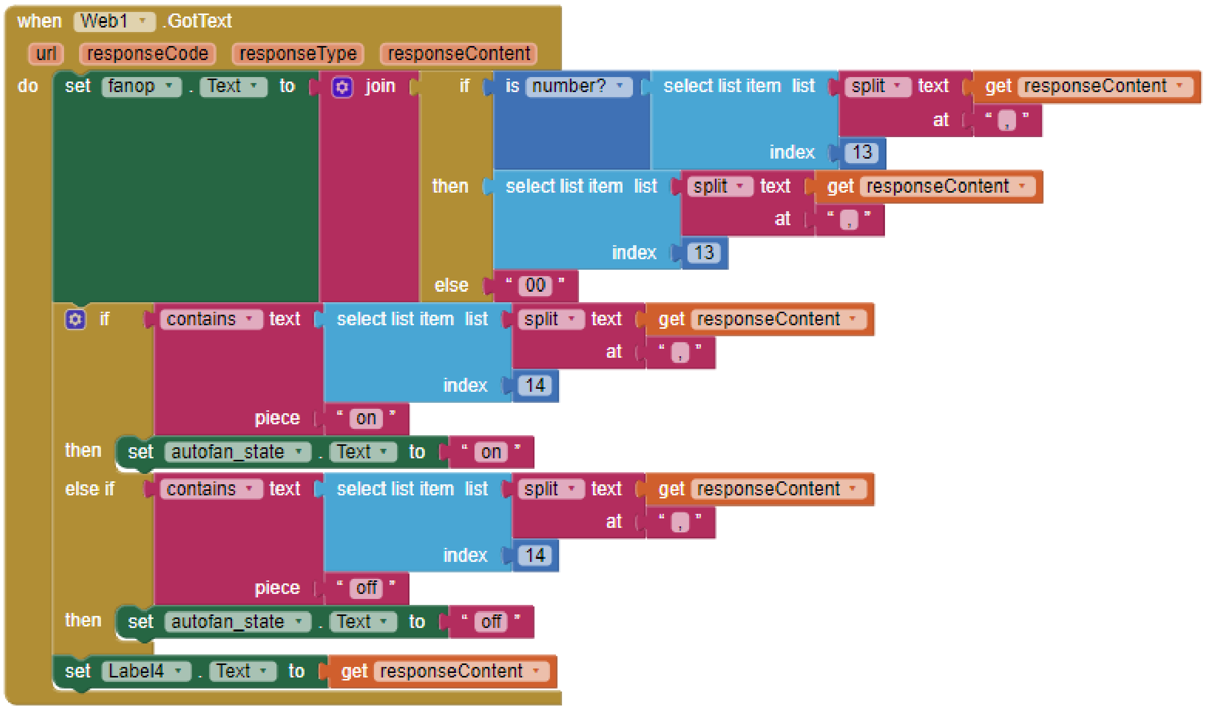


```
when setting .Initialize
do  set ip_text . Text to  call TinyDB1 .GetValue
                                tag " ip "
                                valueIfTagNotThere "  "
    set device_name . Text to  call TinyDB1 .GetValue
                                tag " name "
                                valueIfTagNotThere "  "
    set Web1 . Url to  join " http:// "
                            ip_text . Text
    call Web1 .Get
```

Figure 39 Getting Data From Ras Pi

When app get data from raspberry pi we split data and get fan operation value

And automatic fan state



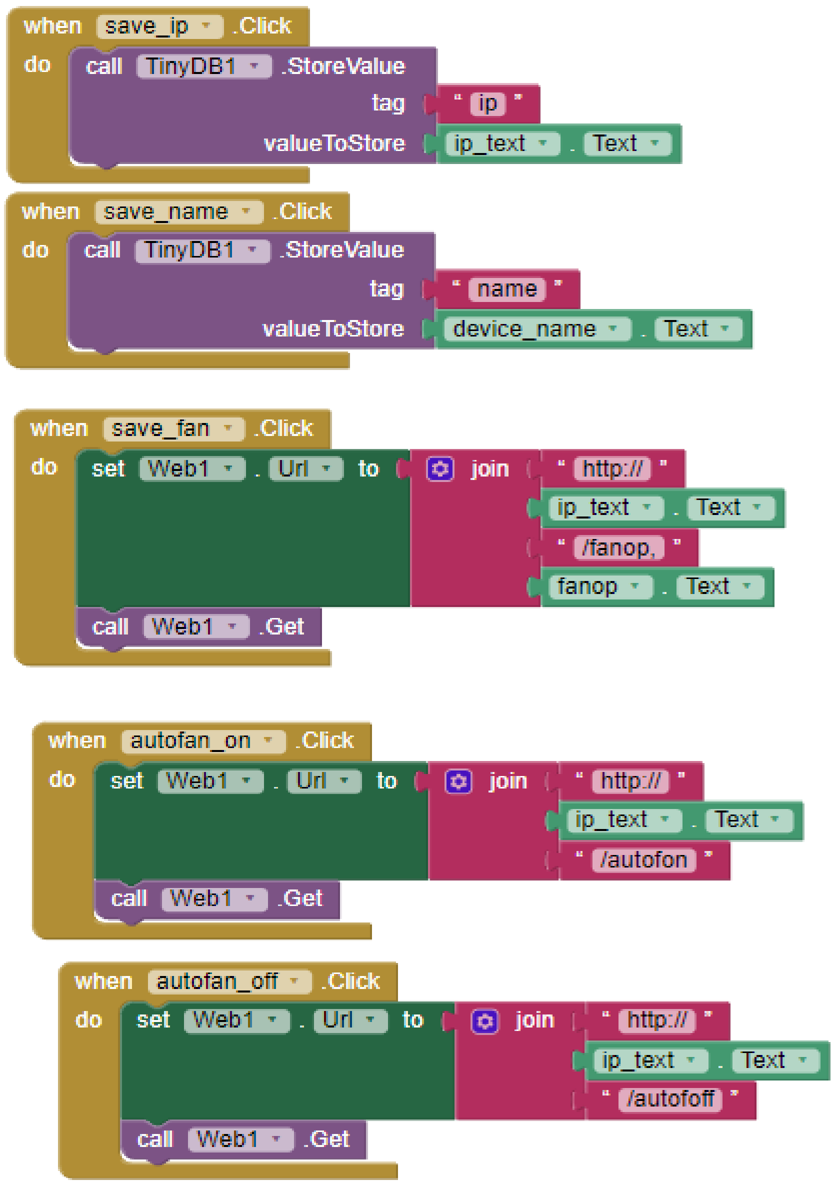
```
when Web1 .GotText
 url  responseCode  responseType  responseContent
do  set fanop . Text to  join  if  is number?  select list item list  split text  get responseContent
                                                                            at " , "
                                                                index 13
                               then  select list item list  split text  get responseContent
                                                                  at " , "
                                                      index 13
                               else " 00 "
    if  contains text  select list item list  split text  get responseContent
                                                     at " , "
                                         index 14
                 piece " on "
    then  set autofan_state . Text to " on "
    else if  contains text  select list item list  split text  get responseContent
                                                          at " , "
                                              index 14
                      piece " off "
    then  set autofan_state . Text to " off "
    set Label4 . Text to  get responseContent
```

Figure 40 Get Fan Operation

When you press on save ip button the value which is written is save in the phone

And When you press on save name button the value which is written is save in the phone

And button When you press on save temp a request will be sent from app to rasp to save value of operation temperature



```
when save_ip .Click
do  call TinyDB1 .StoreValue
                        tag  " ip "
               valueToStore  ip_text . Text

when save_name .Click
do  call TinyDB1 .StoreValue
                        tag  " name "
               valueToStore  device_name . Text

when save_fan .Click
do  set Web1 . Url to  join  " http:// "
                             ip_text . Text
                             " /fanop, "
                             fanop . Text
    call Web1 .Get

when autofan_on .Click
do  set Web1 . Url to  join  " http:// "
                             ip_text . Text
                             " /autofon "
    call Web1 .Get

when autofan_off .Click
do  set Web1 . Url to  join  " http:// "
                             ip_text . Text
                             " /autofoff "
    call Web1 .Get
```

Figure 41 Mit Blocks

5.6.2 sensors button

sensors button will open this page which has temperature sensors and humidity sensors

and gas sensor values and valve state

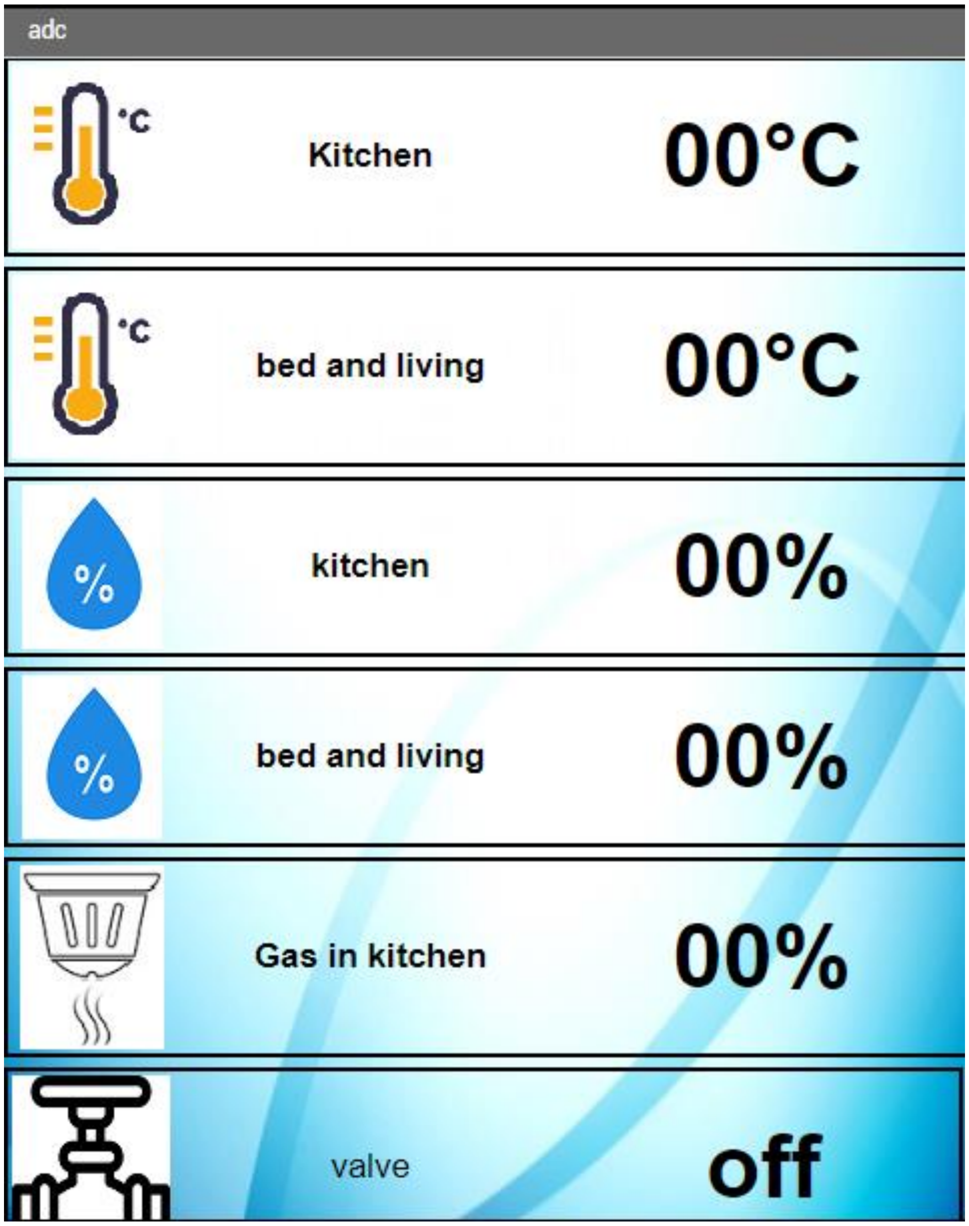
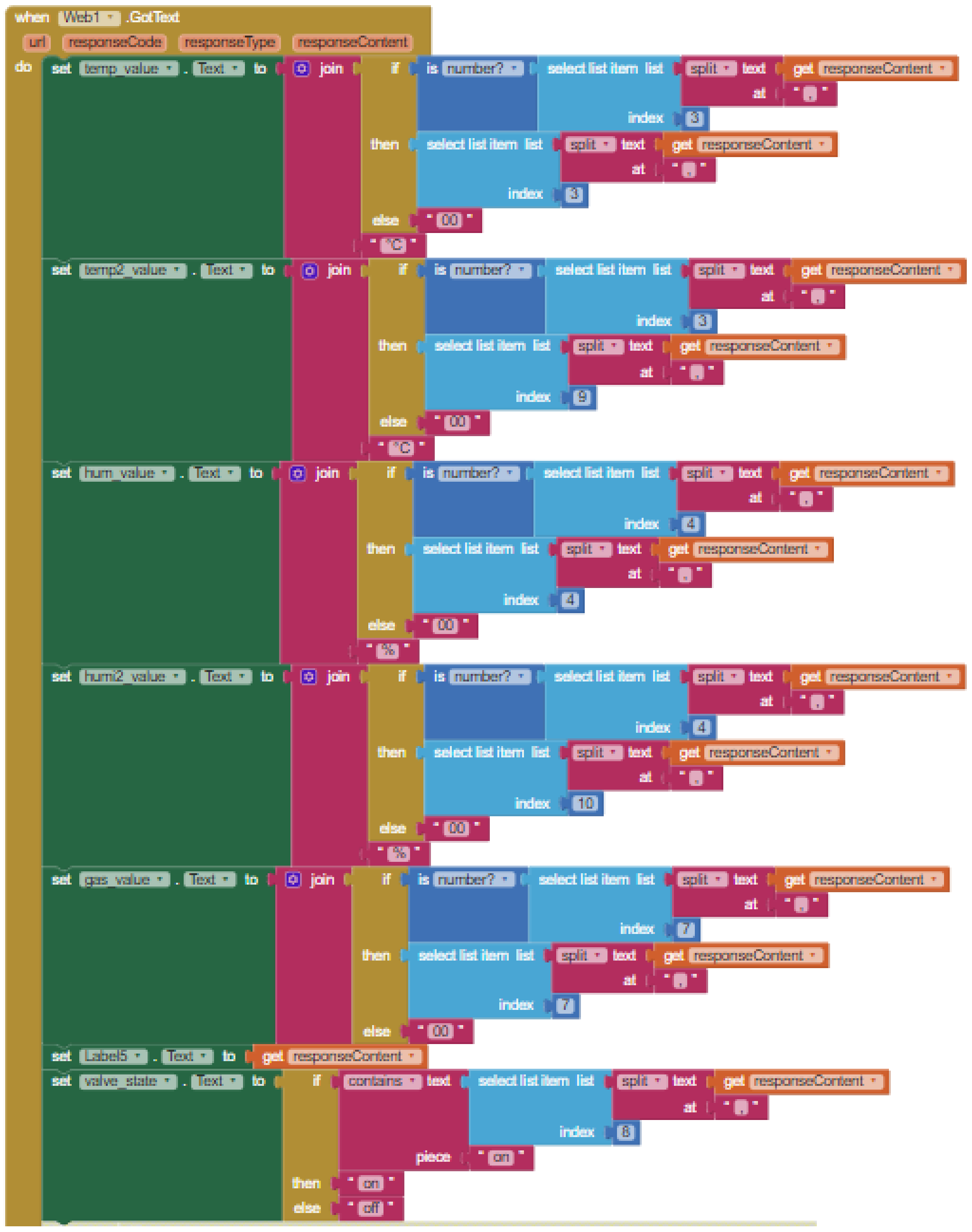


Figure 42 Sensor Button

This is what happening in the background
every second the app send get request to the main server and do its process on these data and display it on the gui



```
when Web1 .GotText
  url  responseCode  responseType  responseContent
do set temp_value . Text to join
       if is number? select list item list split text get responseContent
                                                  at ","
                                       index 3
       then select list item list split text get responseContent
                                         at ","
                              index 3
       else "00"
       "°C"
   set temp2_value . Text to join
       if is number? select list item list split text get responseContent
                                                  at ","
                                       index 3
       then select list item list split text get responseContent
                                         at ","
                              index 9
       else "00"
       "°C"
   set hum_value . Text to join
       if is number? select list item list split text get responseContent
                                                  at ","
                                       index 4
       then select list item list split text get responseContent
                                         at ","
                              index 4
       else "00"
       "%"
   set humi2_value . Text to join
       if is number? select list item list split text get responseContent
                                                  at ","
                                       index 4
       then select list item list split text get responseContent
                                         at ","
                              index 10
       else "00"
       "%"
   set gas_value . Text to join
       if is number? select list item list split text get responseContent
                                                  at ","
                                       index 7
       then select list item list split text get responseContent
                                         at ","
                              index 7
       else "00"
   set Label5 . Text to get responseContent
   set valve_state . Text to
       if contains text select list item list split text get responseContent
                                                     at ","
                                          index 8
                   piece "on"
       then "on"
       else "off"
```

Figure 43 Mit Blocks 2

5.6.3 main page

We have here control button that open or close door and led of bathroom, kitchen, bedroom and living room and we can open all leds together and close it
and control fan if automatic fan in setting is off

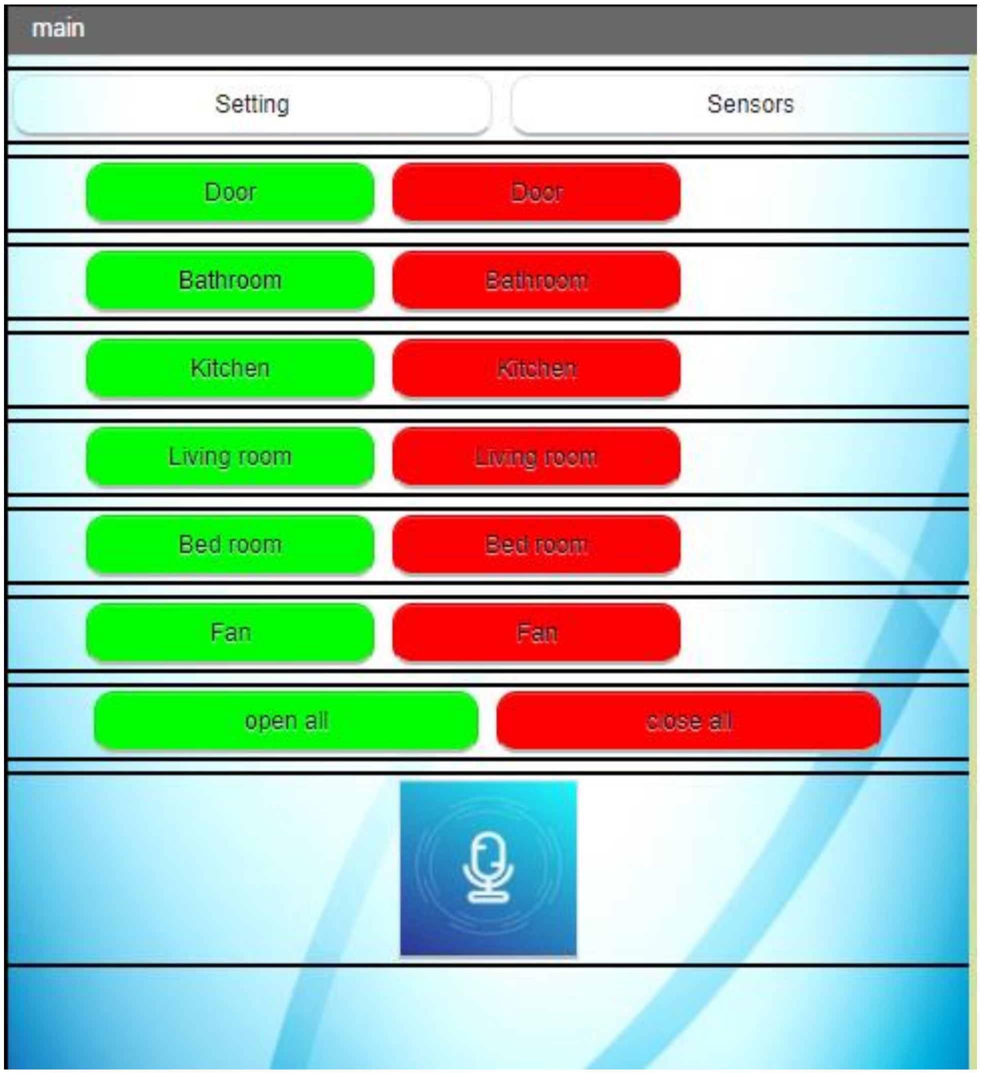


Figure 44 Main Page

6 Speech recognition

Performance evaluation of cloud-based speech recognition systems under different network conditions has received much less attention than other streaming systems. Although Apple Siri and Google Speech Recognition (GSR) are very popular applications that help users to interact with search engines using voice commands, an experimental evaluation of these applications is noticeably missing. 1A brief experimental study on Siri and Google Speech Recognition is reported in ”Impact of the network performance on cloud-based speech recognition systems” in which, a solution that uses network coding to improve the performance of cloudbased speech recognition applications has been proposed. The aforementioned paper is published in ICCCN 2015 [17]. In this paper, we design and implement an extensive experimental evaluation of Apple Siri and Google Speech Recognition under different network conditions to compare the performance of these applications under different conditions.

Delay and accuracy of the voice recognition process is an important parameter that affects the quality a user’s experience with cloud-based speech recognition applications. Streaming voice from the client to the server and converting it to text are two phases of this process and should have the minimum possible delay in order to satisfy the quality of a user’s experience. Delays of this process should also be consistent under all different network conditions. To date, there has not been an extensive evaluation of how Siri and GSR perform under different network conditions. In this paper, we design and implement an experimental evaluation of Siri and GSR. We evaluate these applications under different packet loss and jitter values and measure the delay of each under difficult network conditions. Specifically, we employ two models to evaluate the effects of packet loss and jitter, respectively. Each model is designed to evaluate two factors (jitter or packet loss) with one blocking variable on the response variable - delay. The blocking variable is the application (GSR and Siri), for both of the experiments. An ANOVA test is used to evaluate effects of packet loss and jitter for each experiment respectively. Results of our study show that delays in both applications are affected by packet loss and jitter. The remainder of this paper is organized as follows. In Section II we explore related work. In Section III we describe our experimental methods. In Section IV we describe overall results. In Section V we describe our experimental design and the mathematical model used to analyze experimental data. Section VI discusses results. Finally, in Section VII we discusses threats to validity of our experiment and conclude in Section VIII.

6.1 Related Work

A measurement study on Google+, iChat, and Skype was performed by Yang Xu et al. [17]. They explored the architectural features of these applications. Using passive and active experiments, the authors unveiled some performance details of these applications such as video generation and adaption techniques, packet loss recovery solutions, and end-to-end delays. Based on their experiments the server location had a significant impact on user performance and also loss recovery in server-based applications. They also argued that using batched re-transmissions was a good alternative for real time applications instead of using Forward Error Correction (FEC) –an error control technique in streaming over unreliable network connections. Te-Yuan Huang et al. did a measurement study on the performance of Skype's FEC mechanism [17]. They studied the amount of the redundancy added by the FEC mechanism and the trade-offs between the quality of the users' experience and also the resulting redundancy due to FEC. They tried to find an optimal level of redundancy to achieve the maximum quality of the users' experience. Te-Yuan Huang et al. also performed a study on voice rate adaption of Skype under different network conditions [17]. Results of this study showed that using public domain codecs was not an ideal choice for users' satisfaction. In this study, they considered different levels of packet loss to run their experiment and came up with a model to control the redundancy under different packet loss conditions. Kuan-Ta Chen et al. proposed a framework for users' QoE measurement [17]. Their proposed framework was called OneClick, and provided a dedicated key that could be pressed by users whenever they felt unsatisfied by the network conditions with streaming media. OneClick was implemented on two applications –instant messaging applications, and shooter games. Another framework that quantified the quality of a user's experience was proposed by Kuan-Ta Chen et al [17]. The proposed system was able to verify participants' inputs, so it supported crowd-sourcing. Participation is made easy in this framework, and it also generates interval-scale scores. They argue that researchers can use this framework for measuring the quality of a users' experience without affecting quality of the results and achieve a higher level of diversity in users' participation while also keeping a cost low. A delayed-based congestion control is proposed and developed by Lukasz Budzisz et al. [17]. The proposed system offers low standing queues and delay in homogeneous networks, and balanced delay-based and lossbased flows in heterogeneous networks. They argue that this system can achieve these properties under different loss values, and outperform TCP flows. Using experiments and analysis, they demonstrate that this system guarantees aforementioned properties. Hayes et al. proposed an algorithm which tolerates non-congestion related packet loss [17]. They proved experimentally that the proposed algorithm improves the throughput by 150% under packet loss of 1% and improves the ability to share the capacity by more than 50%Akhshabi et al. proposed an experimental evaluation of rate adaption algorithms for streaming over HTTP . They experimentally evaluated three common video streaming applications under a range of bandwidth values. Results of this study showed that congestion control of TCP and its reliability requirement does not necessarily affect the performance of such streaming applications. Interaction of rateadaption logic and TCP congestion control is left as an open research problem. Chen et al. experimentally studied performance of multipath TCP over wireless networks [17]. They measured the latency resulting from different cellular data providers. Results of this study show that Multipath TCP offers a robust data transport under various network traffic conditions. Studying the energy costs and performance trade-offs should be considered as a possible extension of this study. Google is currently working on a new transport protocol for the Internet which is called QUIC(Quick UDP Internet Connections) [17]. QUIC uses UDP and solves problems of packet delay under different packet loss values in TCP connections. QUIC solves this problem by multiplexing and FEC. An experimental investigation on the Google Congestion Control (GCC) in the RTCWeb IETF WG was performed by Cicco et al. [17]. They implemented a controlled testbed for their experiment. Results of this experimental study show that the proposed algorithm works well but it does not utilize the bandwidth fairly when it is shared by two GCC flows or a GCC and a TCP flow. Cicco et al. have also experimentally investigated the High Definition (HD) video distribution of Akamai [17]. They explained details of Akamai's client-server protocol which implements the quality adaption algorithm. Their study shows that the proposed technique encodes any video at five different bit rates and stores all of them at the server. Server selects the bit rate that matches the bandwidth that is measured based on the signal receiving from the cilent. The bitrate level adaptively changes based on the available bandwidth. Authors of the paper also evaluated the dynamics of the algorithm in three scenarios. Winkler et al. ran a set of experiments to asses quality of experience on television and mobile applications [17]. Their proposed subjective experiment considers different bitrates, contents, codec, and network traffic conditions. Authors of the paper used Single Stimulus Continous Quality Evaluation (SSCQE) and Double Stimulus Impairment Scale (DSIS) on the same set of materials and compared these methods and analyzed results of experiments in view of codec performance. A mesh-pull-based P2P video streaming using Fountain codes is proposed by Oh et al. [17]. The proposed system offers fast and smooth streaming with low complexity. Experimental evaluations show that the proposed system has better performance than existing buffer-map-based video streaming systems under packet loss values. Considering jitter as another important factor and evaluation of behavior of proposed system considering jitter values can be a potential extension of this study. Application of Fountain Multiple Description Coding (MDC) in video streaming over a heterogeneous peer to peer networks is considered by Smith et al. [17]. They conclude that Fountain MDC codes are favorable in such cases, but there are some restrictions in real-world P2P streaming systems. Finally, Vukobratovic et al. proposed a novel multicast streaming system that is based on Expanding Window Fountain (EWF) codes for real-time multicast [17]. Using Raptor-like precoding has been addressed as a potential improvement in this area.

6.2 Experimental

Testbeds We design and implement our experimental testbed to study the performance of Apple Siri and GSR under loss and jitter. Clients transmit voice data through a network traffic shaper, in which is we change jitter and packet loss values in the communication network. We set a bandwidth to 2Mbps which is typical on 3G connections [17]. The server receives voice data, translates the voice into text, and sends the text and search results based on the converted text to the client. The client calculates the delay of the server response. To calculate the accuracy of transcription we use Levenshtein distance [17]. Accuracy is measured as the match percentage of the original string used to generate the voice and the resulting transcription. The client uses Wireshark Version 1.12.4 to timestamp the traffic of voice transmission to and from the server . We developed a Windows application using Visual C# to timestamp the voice playback. All experiments are performed on a Windows 7 platform for GSR, and on iOS 7.0 for Siri. The traffic shaper is a netem box which runs the Fedora Linux operating system. We ran our experiment 30 times for each value of loss and jitter and for each cloud speech recognizer.

6.2.1 Experimental

Testbed for GSR We use the GSR service available in Google Chrome. There is also another alternative for using Google voice recognition. Google offers a voice recognition Web service that can be used in Windows applications. Figure 1 shows the architecture of our experimental setup. Clients transmit voice packets to the Google server through the netem box that changes network traffic performance. We used a recorded voice with a length of 26.4 seconds for all experiments in order to have a consistent measurement. Google starts to recognize voice as soon as it receives the first voice packet, and sends converted text back to the client. The client records the time of each packet and also voice transmission time to calculate the transcription time of the experiment. The client also compares the resulting text to the original string; which was used to generate the voice command and calculates transmission accuracy using the Levenshtein distance .

6.2.2 Experimental Testbed for Siri

The experimental setup for Siri is similar to GSR. We use an iPhone as the client. A client is connected to the Internet through a WiFi router then to a netem box. Here we also used Wireshark to timestamp the transmission of voice packets and reception of results from the Siri server. Figure 2 depicts this setup

6.3 Overall Results

To investigate the effect of packet loss and jitter on delay and accuracy, we generate packet loss from 1% to 5% and jitter from 20 ms to 200 ms respectively on our testbeds and observe the resulting accuracy and delay. Siri and GSR both keep 100% accuracy under high values of packet loss and jitter, so we just consider delay values in the rest of our study. Overall results are shown in Figures 3 to 6, where the y axis displays delay(s), and the x axis displays packet loss (percentile) and jitter (ms), respectively. There are increasing trends as packet loss and jitter increases, for both Siri and GSR. For GSR, an increase of 1 packet loss unit (percentile), leads to delay increases in the range of 0-100 ms. An increase of 1 unit (20 ms) in packet loss leads to increases in delay from 0-100 ms. In addition, the variance of delay also increases as packet loss and jitter increase, indicating a trend of instability. For Siri, the increase in 1 unit (percentile) packet loss leads to increases in delay of 200 ms; which is worse than GSR. On the other hand, jitter has less impact on delay. In addition, the variance of delay is unchanged, compared to GSR.

6.4 Experiment Design

We evaluate our results and data using mathematical models and an ANOVA test. Our response variable is delay of transcription and our factors are loss and jitter

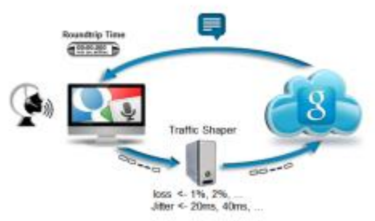


Figure 1: Experimental testbed for GSR.

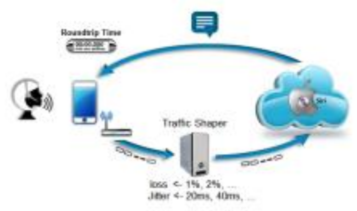


Figure 2: Experimental testbed for Siri.

Figure 45 Experiment Design for GSR and Siri

6.4.1 Model

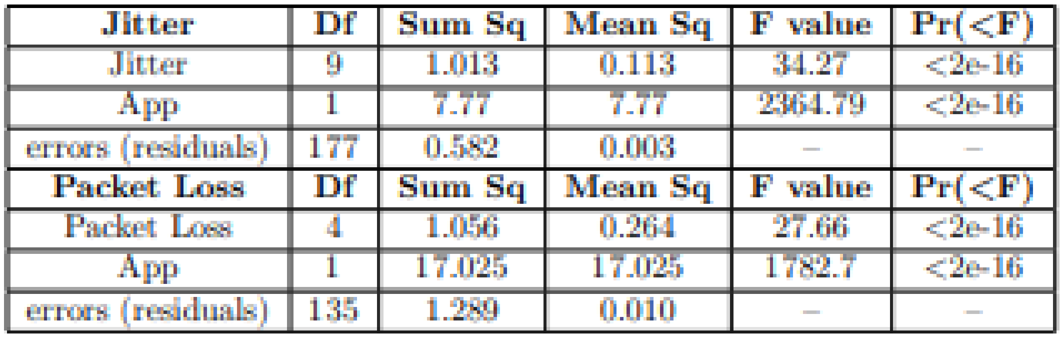
Since data is collected by varying jitter and packet loss respectively, we designed two models to assess the effect of jitter and packet loss on delay. Also, since data is collected from two applications (i.e., Siri and GSR), we treat the application as a blocking variable. Hence, we set up two models for jitter and packet loss respectively. Each model contains one factor and one blocking variable. For the first model, the response variable is delay, the independent variable is jitter and the blocking variable is application. Also, to guarantee that the assumptions still hold for the following ANOVA tests, we apply a logarithmic transformation on the response variable. Hence, the first model can be expressed as: $\log(y_{ij}) = \mu + \alpha_i + \beta_j + e_{ij}$ (1) where α is the jitter, and β represents the application. Similarly, the second model can be expressed as: $\log(y_{ij}) = \mu + \gamma_i + \beta_j + e_{ij}$ (2) where γ is the jitter, and β represents the application. For model 1, the factor (jitter) has 10 alternatives; which are the jitter duration values ranging from 20 to 200 ms. For model 2, the factor (packet loss) has 5 alternatives; which are the proportion of lost packets ranging from 1% to 5%. The blocking variable for both models has 2 alternatives; which are GSR and Siri, respectively.

6.4.2 Assumption of Normality Check

Some assumptions should be checked before conducting the ANOVA tests. In this Section, the interaction of independent variables, the normality of errors and the constant variance of errors are tested for normality. We first test the interaction between factors. In Figures 7 and 8, the errors (residuals) give us confidence that they are constantly distributed as the fitted values change, indicating that the interactions between the blocking variable and factor are trivial for both jitter (Eq. 1) and packet loss models . Secondly, the error variance of each model also appears constant. Figures 9 through 12 show that errors (residuals) appear constant as the independent variables (jitter/packet loss and application) change, indicating that the error of models 1 and 2 are constant.

Finally, figures 13 and 14 show that the error distributions for model 1 and model 2 are normal, indicating that the assumption of normally distributed errors holds for both of the models. In summary, all the the assumptions for conducting an ANOVA test hold for both models (Eq. 1 and Eq. 2).

Table 8: Statistical Finding Of Jitter And Packet Loss



Jitter	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(<F)**
Jitter	9	1.013	0.113	34.27	<2e-16
App	1	7.77	7.77	2364.79	<2e-16
errors (residuals)	177	0.582	0.003	–	–
Packet Loss	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(<F)**
Packet Loss	4	1.056	0.264	27.66	<2e-16
App	1	17.025	17.025	1782.7	<2e-16
errors (residuals)	135	1.289	0.010	–	–

6.5 Results

Table 1 provides conclusive evidence that roundtrip delay of GSR and Siri are affected by both jitter (pvalue = 2e-16, f-value =34.27 on 9 df.) and packet loss (p-value = 2e-16 , f-value =27.66 on 4 df.). Jitter causes packets to arrive out of order and TCP needs to reorder packets before delivering them to the application layer. TCP also re-transmits lost packets. Both packet loss and jitter reduce the voice stream quality and this affects the performance of the speech recognition. The application, on the other hand, affects the delay much more seriously. Specifically, the f-values of application for jitter and packet loss are 1782.7 and 2364.79 on 1 df., respectively. To test the difference between Siri and GSR on delay, we also employ a Walch t-test to compare the samples obtained from Siri and GSR. The mean difference between Siri and GSR is 1.666s, with 95% confidence interval from -1.736 to - 1.600, (p-value < 2.2e-16). This suggests that there exists a statistically significant difference between GSR and Siri. By examining the response variable for each application separately, Siri causes much more delay than GSR. This is because the algorithm employed by Siri keeps the resulting text accurate by starting the speech recognition process just after receiving the voice date. That means Siri needs to receive the entire voice stream before starting to generate the text. As a result, this increases the delay in processing the whole text and accounts for the majority of the total delay. GSR, on the other hand, keeps the result accurate by adaptively adjusting the transport and application layers and so it offers less delay even under high values of packet loss and jitter compared to Siri.

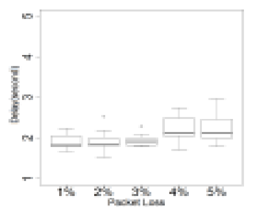


Figure 3: Impact of packet loss on delay of GSR

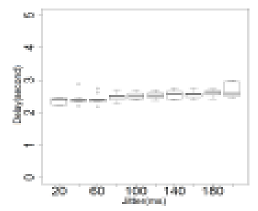


Figure 4: Impact of jitter on delay of GSR

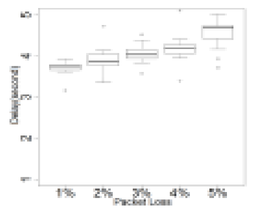


Figure 5: Impact of packet loss on delay of Siri

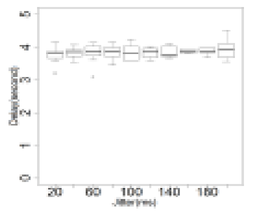


Figure 6: Impact of jitter on delay of Siri

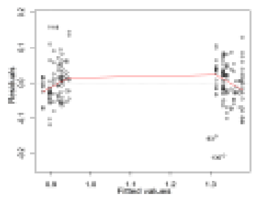


Figure 7: Jitter: Fitted Values vs. Errors (Residuals)

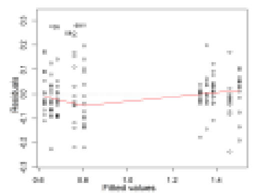


Figure 8: Packet Loss: Fitted Values vs. Errors (Residuals)

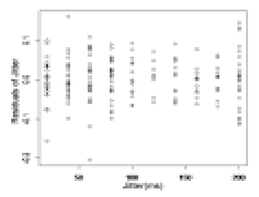


Figure 9: Jitter: Fitted Values vs. Errors (Residuals)

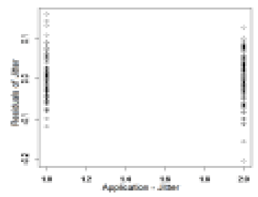


Figure 10: Application (Jitter): Fitted Values vs.Errors (Residuals)

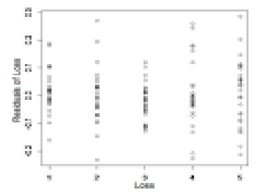


Figure 11: Packet Loss: Fitted Values vs.Errors (Residuals)

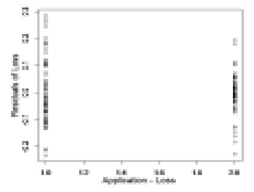


Figure 12: Application (Packet Loss): Fitted Values vs.Errors (Residuals)

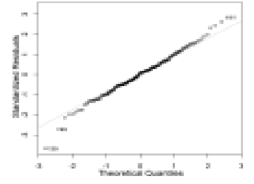


Figure 13: Normality of Jitter

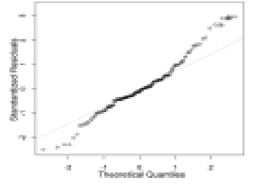


Figure 14: Normality of Packet Loss

Figure 46 Figures Of Results

6.6 Threats to Validity

6.6.1 Threats to Internal Validity

One of the possible threats to internal validity is the hardware limitations of the devices running GSR and Siri. More specifically, the processing speed of memory and CPU will affect the processing of data streams in a PC. Another possible threat is the status of the PC. For example, when the OS is busy, it does not have enough time to respond to the interruptions generated from GSR or Siri, hence generating and thus affecting delay.

6.6.2 Threats to External Validity

All of the experiments were conducted in our lab and through our campus network. It is likely that the configuration of our campus network is different from other networks, such as firewalls and TCP/UDP controls. Hence, the conclusion obtained from the experiment cannot be generalized to common network environments. In addition, the available bandwidth of different regions in United States is different. It is possible that this diversity affects the conclusion that it cannot be applied to the other regions in United States. Finally, the sample is small (the evaluation is run on one desktop in a laboratory setting). A larger scale experiment running on more desktops, as well as laptops and smart phones, will lessen external threats.

6.6.3 Threats to Construct Validity

Since the delay generated by the Internet (e.g., router, DNS, etc.) is complicated and unpredictable, it is hard to say the extent to which packet loss and jitter impact delay. Also, the transportation and routing layers employ self-adaptive mechanisms to adjust the performance of specific applications, e.g., GSR and Siri. In the end, both the jitter and the packet loss are generated by a specific program (i.e., simulated), rather than real network conditions. It is hard to know whether the simulated impact has the same effects of real jitter or packet loss.

7 Gesture recognition

The gesture is known as a form of non-verbal communication or non-vocal communication where utilize of the body's movement that can convey a particular message originating from parts of the human body, the hand or face are the most commonly adopt [19]. Gesture-based interaction introduced by Krueger as a new type of Human-Computer Interaction (HCI) in the middle 1970s has become a magnetic area of the research. In the Human-Computer-Interaction (HCI), building interfaces of applications with managing each part of the human body to communicate naturally are the great attention to do research, especially the hands as the most effective-alternative for the interaction tool, considering their ability [19]. Through Human-Computer-Interaction (HCI), recognizing hand gestures could help achieve the ease and naturalness desired [19]. When interacting with other people, hand movements have the meaning to convey something with its information. Ranging from simple hand movements to more complex ones. For example, we can use our hand to point something (object or people) or use different simple shapes of hand or hand movements expressed through manual articulations combined with their grammar and lexicon as wellknown as sign languages. Hence, using hand gestures as Advances in Engineering Research, volume 207 Proceedings of the 2nd International Seminar of Science and Applied Technology (ISSAT 2021) Copyright © 2021 The Authors. Published by Atlantis Press International B.V. This is an open access article distributed under the CC BY-NC 4.0 license - http://creativecommons.org/licenses/by-nc/4.0/. 101 a device then integration with computers can help people communicate more intuitively [19]. Currently, many frameworks or library machine learning for hand gesture recognition have been built to make it easier for anyone to build AI (Artificial Intelligence) based applications. One of them is MediaPipe. The MediaPipe framework is present by Google for solving the problem using machine learning such as Face Detection, Face Mesh, Iris, Hands, Pose, Holistic, Hair segmentation, Object detection, Box Tracking, Instant Motion Tracking, Objection, and KIFT. MediaPipe framework helps a developer focus on the algorithm and model development on the application, then support environment application through results reproducible across different devices and platforms which it is a few advantages of using features on the MediaPipe framework [19]. In this paper, we focus on developing a manual user guide application with improving architecture application by applying hand gesture recognition using the MediaPipe framework and camera of Kinect for capture hand pose to recognized. Using hand gesture recognition will improve our user guide application more interactive.

7.1 RELATED WORK

7.1.1 Hand Gesture Recognition

Gesture recognition is an essential topic in computer science and builds technology that aims to interpret human gestures where anyone can use simple gestures to interact with the device without touching them directly. The entire procedure of tracking gestures to their representation and converting them to some purposeful command is known as gesture recognition [19]. Identify from explicit hand gestures as input then process these gestures representation for devices through mapping as output is the aim in hand gestures recognition. Recognition of the hand gesture in kinds of literature based on extracted features is divided into three groups, as follows: ● High-Level Features-Based Approaches: Aim to figure out the position of the palm and joint angles such as the fingertips, joint location, or anchor points of the palm [18,19]. Whereas, effect collisions or occlusions on the image are difficult to detect after features are extracted [54], and sensitivity segmentation performance on 2D hand image are the problem that occurred frequently. The gestures are defined from the results with a set of rules and conditions from the vectors and joints of the hands [19]. ● Low-Level Feature-Based Approaches: Utilized these features for could be extracted quickly for robust to noise. Zhou [19] discovered recognition of the hand shape as a cluster-based signature using a novel distance metric called Finger Earth's Distance. Stanner [19] determines the bounding region of the hand elliptically for implement hand recognition based on principal axes. Yang [19] did research using the optical flow of the hand region as a lowlevel feature. Low-Level Feature-Based is not efficient when cluttered background [19]. ● 3D Reconstruction-Based Approaches: Use the 3D model of features for achieving the construe of hand completely. Research [19] showed that successfully segmenting the hand in skin color needs similarity and high contrast of the background related to the hand through structured light to bring in 3D of depth data. Another one [19] uses a stereo camera to track numerous interest points of the superficies of the hand which results in difficulty for handle robust 3D reconstruction, despite data contains 3D has valuable information that can help dispose of vagueness. See [19] for more 3D reconstruction-based approach. From kinds of literature, there are three Hand gesture recognition methods, as follow:

● Machine Learning Approaches: The resulting output came from the stochastic process and approach based on statistical modeling for dynamic gestures such as PCA, HMM, advanced particle filtering and condensation algorithm [19].

● Algorithm Approaches: Collection of encoded conditions and restraints manually for defining as gestures in dynamic gestures. Galveia [19] applied a 3rd-degree polynomial equation to determine the dynamic component of the hand gestures (create a 3rd-degree polynomial equation, recognition, reduced complexity of equations, and comparison handling in gestures library).

● Rule-based Approaches: Suitable for dynamic gestures either static gestures which are contained a set of pre-encoded rules and features inputs [19]. The features of input gestures are extracted and compared to the encoding rules that are the flow of Advances in Engineering Research, volume 207 102 the recognized gestures. Matching between gestures with rule and input which is outputted approved as known gestures [19].

7.1.2 MediaPipe Framework

Today, there are many frameworks or libraries of machine learning for hand gesture recognition. One of them is MediaPipe. The MediaPipe is a framework designed to implement production-ready machine learning that must build pipelines to perform inference over arbitrary sensory data, has published code accompanying research work, and build technology prototypes [19]. In MediaPipe, graph modular components come from a perception pipeline along with the function of inference model function, media processing model, and data transformations [19]. Graph of operations are used in others machine learning such as Tensor flow [19], MXNet [19], PyTorch[19],

CNTK,OpenCV 4.0[19]. Using MediaPipe for hand gesture recognition has been researched by Zhang [19] before, using a single RGB camera for AR/VR application in a real-time system that predicts a hand skeleton of the human. We can develop a combined MediaPipe using other devices. The MediaPipe implements pipeline in Figure 1. consists of two models for hand gesture recognition as follows [19] :

1. A palm detector model processes the captured image and turns the image with an oriented bounding box of the hand,

2. A hand landmark model processes on cropped bounding box image and returns 3D hand key points on hand.

3. A gesture recognizer that classifies 3D hand key points then configuration them into a discrete set of gestures.

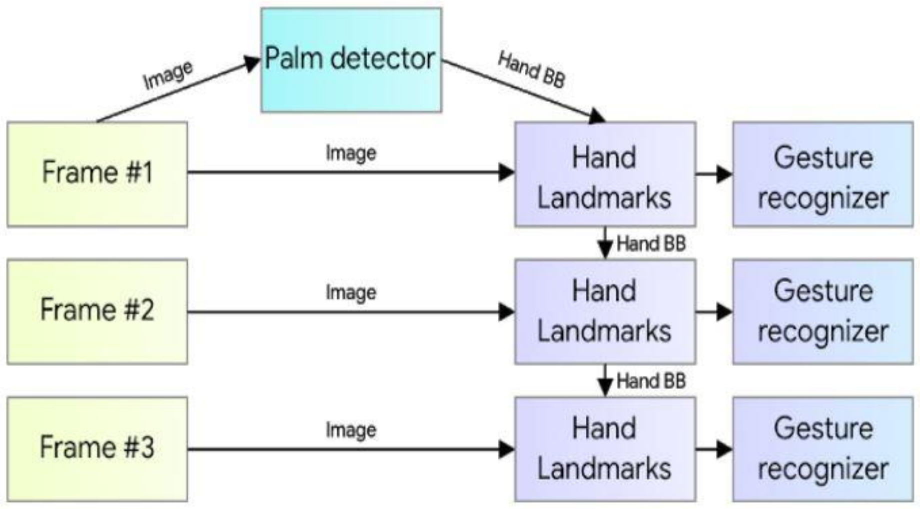


Figure 47 Hand Preception

7.1.2.1 Palm Detector Model

MediaPipe framework has built detect initial palm detector called BlazePalm. Detecting the hand is a complex task. Step one is to train the palm instead of the hand detector, then using the non-maximum suppression algorithm on the palm, where it is modeled using square bounding boxes to avoid other aspect ratios and reducing the number of anchors by a factor of 3-5. Next, encoder-decoder of feature extraction that is used for bigger scene context-awareness even small objects, lastly, minimize the focal loss during training with support a large number of anchors resulting from the high scale variance [19]. 2.2.2 Hand Landmark Achieves precise key point localization of 21 key points with a 3D hand-knuckle coordinate which is conducted inside the detected hand regions through regression which will produce the coordinate prediction directly which is a model of the hand landmark in MediaPipe [19]., see in Figure 2.

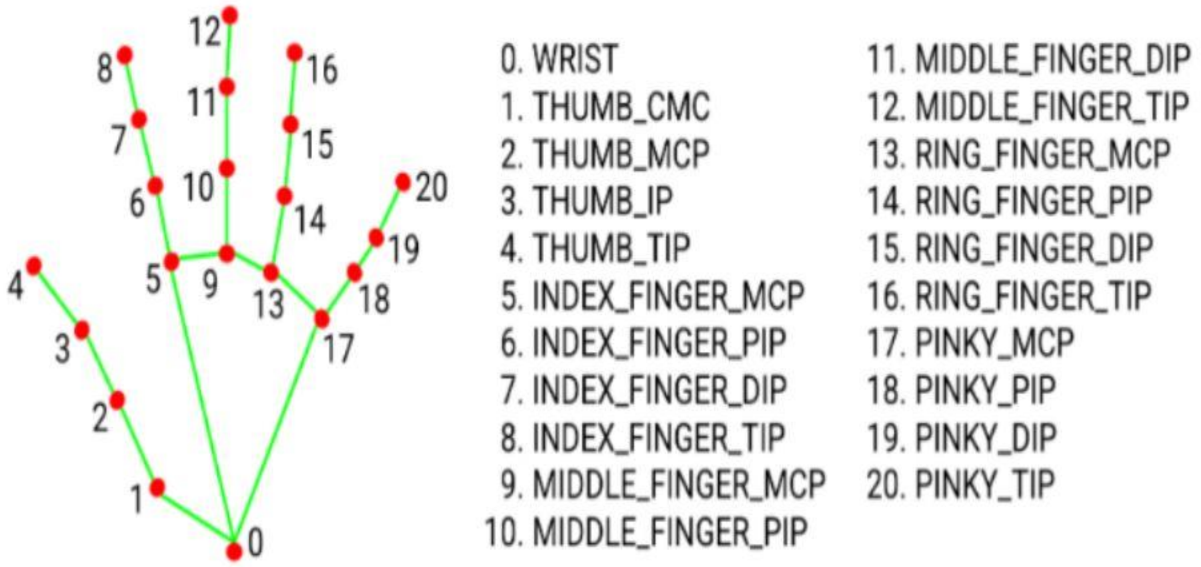


Figure 48 Hand Land Marks

Each hand-knuckle of the landmark has coordinate is composed of x, y, and z where x and y are normalized to [0.0, 1.0] by image width and height, while z representation the depth of landmark. The depth of landmark that can be found at the wrist being the ancestor. The closed the landmark to the camera, the value becomes smaller.

7.1.2.2 Hand Recognizer

For recognizing hand gestures, the implementation of a simple algorithm is to compute gestures with a determined accumulated angle of the joint state or conditions each finger such as bent finger or straight finger then do map the set of finger states that we got before to set the label of pre-defined gestures like numbers

7.2 RESEARCH METHODS

In this research, the user guide application is a guide for the user to display steps taken by the system by identifying hand gestures as a certain command. We develop a user guide application that implements hand gesture recognition using Kinect to capture hand pose and then recognize it for running the application. We are using the MediaPipe framework and Python programming language to develop an application user guide. For detail, can see Figure 5 below.

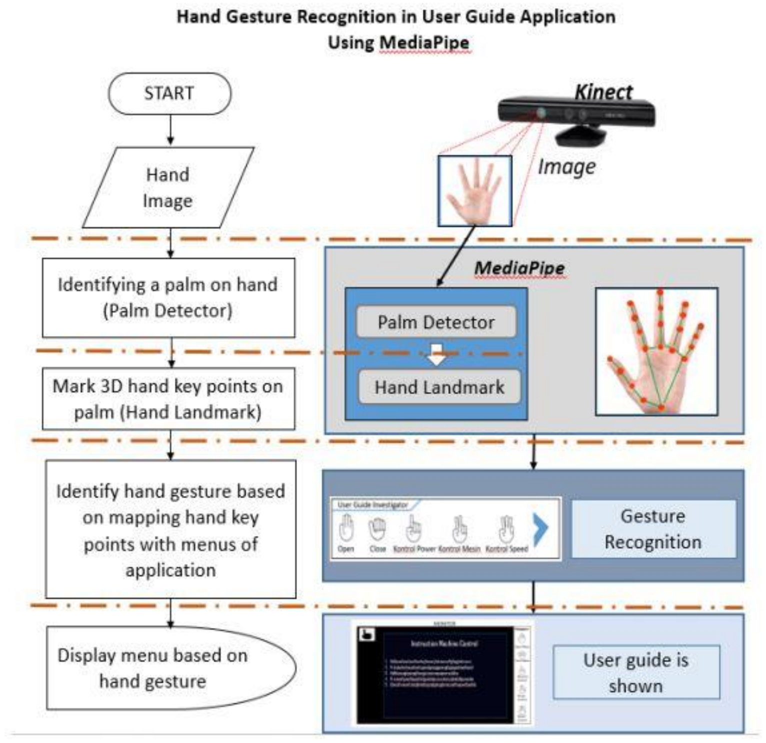

Hand Gesture Recognition in User Guide Application Using MediaPipe
START
Hand Image
Identifying a palm on hand (Palm Detector)
Mark 3D hand key points on palm (Hand Landmark)
Identify hand gesture based on mapping hand key points with menus of application
Display menu based on hand gesture
Kinect
Image
MediaPipe
Palm Detector
Hand Landmark
Gesture Recognition
User guide is shown

Figure 49 Work Flow Method Research

Numerous inspiring successes research in applying Kinect as a device for articulating human body tracking, pose even recognition systems. We use the Kinect x360 with RGB camera with a resolution of 640x240 pixel for capturing a real-time image for process using MediaPipe framework. MediaPipe will read the image that received from Kinect, then on an image will do palm detection and make hand landmark that made return 3D hand key points and joint it make up like skeleton. 3D key points in the palm that has been marked in the image will be computed and initialized as a tool for reading pose hand and recognition based that will be conveyed information based on hand pose had been initialized before.

7.2.1 Identification Hand Gesture in MediaPipe

To identify the poses of the hand differently could be calculated using 21 key points on hand landmarks explained in Section 2.2.2 above. This identifying could be done with, firstly, determine every finger of the hand condition is open or close. For more clearly, see Figure 6 below. Figure 6 shows a pseudocode or algorithm for identifying a condition finger of a hand-related to Figure 2. In Figure 2, we can see that coordinate [4,8,12,16,20] is a coordinate of tips of fingers and declared as fingertips. The Declaration of hand with coordinate 0 until 20 is obtained from 21 key points with a handknuckle coordinate in hand landmark. We will compare the coordinate of fingertip based on position x (horizontal) and y (vertical) with middle points [2,6,10,14,18]. If the compare coordinates a fingertip has a value higher than middle points, then set finger with value 1, mean finger in condition open and vice versa.

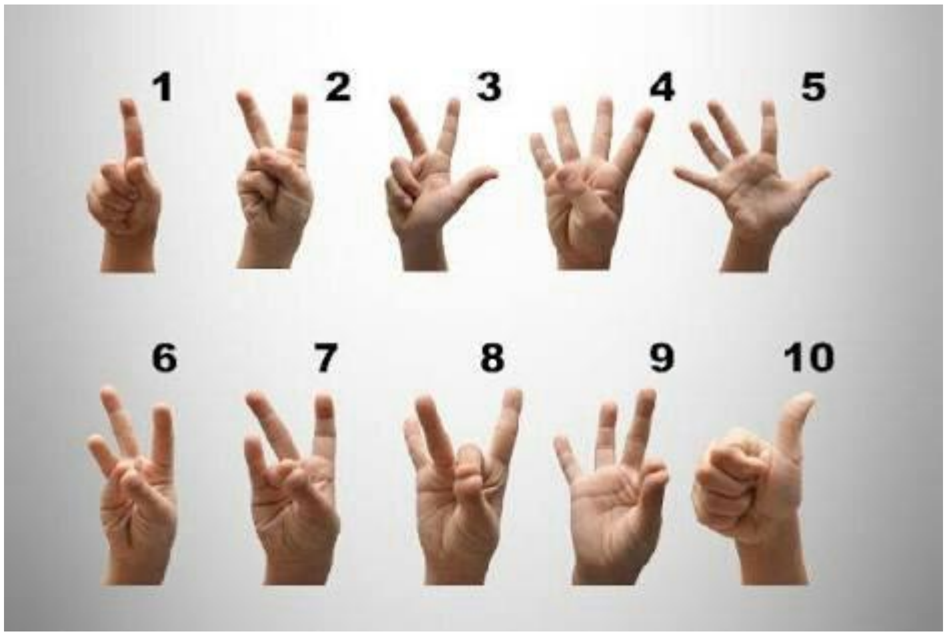
In the next step, after determined the condition each finger is open or close as mentioned before, we gathered all values of fingers and compared every condition of fingers of the hand (1 for the finger is open and 0 for the finger is close), see in Figure 7. If the condition has been fulfilled and resulted is true, then the program will execute according to the instruction that has been made.

7.2.2 User Gesture Control Application

To develop a User Gesture Control application, we prepared a code to identify user gesture and translate it into commands to control appliances within the smart home. The system will capture an image of a hand pose, identifying it then do command to control lighting of the house. If the user makes a gesture and the application recognizes it from the dataset then a command will be sent to the server via a http request

7.2.3 Description of The Dataset

We use 10 hand gestures representing the numbers in sign language shown in this figure



From those gestures we deduced the set of rules that when applied to the coordinates (x,y,z) of hand joints provided by Mediapipe hand landmark can determine the number that gesture express then send a command according to that number. in our code lmlist contains hand joints coordinates and is accessed by putting the id of the joint(shown in figure 2) followed by the id of the required coordinate then the output is the value of the coordinate

7.2.4 example:

lmlist[8][2]

output: index finger tip y coordinate

lmlist[12][1]

output: middle finger tip x coordinate

Gesture of Number 1:

If the index finger is up and the rest are down

if lmlist[8][2] < lmlist[6][2] and lmlist[20][2] > lmlist[18][2]

Gesture of Number 2:

If index and middle finger are up and the rest are down

if lmlist[8][2] < lmlist[6][2] and lmlist[12][2] < lmlist[10][2] and not lmlist[4][1] > lmlist[8][1]:

Gesture of Number 3:

If index and middle finger and thumb are up and the rest are down

lmlist[4][1] > lmlist[12][1] and lmlist[8][2] < lmlist[6][2] and lmlist[12][2] < lmlist[10][2] and \
lmlist[16][2] > lmlist[14][2]

Gesture of Number 4:

If all fingers are up except the thumb

```
if lmlist[8][2] < lmlist[6][2] and lmlist[12][2] < lmlist[10][2] and lmlist[16][2] < lmlist[14][2] and \
    lmlist[20][2] < lmlist[18][2]
```

Gesture of Number 5:

If all fingers are up

```
if lmlist[4][1] > lmlist[12][1] and lmlist[8][2] < lmlist[6][2] and lmlist[12][2] < lmlist[10][2] and \
    lmlist[16][2] < lmlist[14][2] and lmlist[20][2] < lmlist[18][2]
```

Gesture of Number 6:

If all fingers are up and the tip of thumb and little finger tip are touching

```
lmlist[8][2] < lmlist[6][2] and lmlist[12][2] < lmlist[10][2] and lmlist[16][2] < lmlist[14][2] and \
    lmlist[4][1] < lmlist[12][1] and lmlist[20][2] > lmlist[18][2]
```

Gesture of Number 7:

If all fingers are up and the tip of thumb and ring finger tip are touching

```
lmlist[8][2] < lmlist[6][2] and lmlist[12][2] < lmlist[10][2] and lmlist[16][2] > lmlist[14][2] and \
    lmlist[4][1] > lmlist[20][1] and lmlist[20][2] < lmlist[18][2]
```

Gesture of Number 8:

If all fingers are up and the tip of thumb and middle finger tip are touching

```
lmlist[8][2] < lmlist[6][2] and lmlist[12][2] > lmlist[10][2] and lmlist[16][2] < lmlist[14][2] and \
lmlist[4][1] > lmlist[20][1] and lmlist[20][2] < lmlist[18][2]
```

Gesture of Number 9:

If all fingers are up and the tip of thumb and index finger tip are touching

lmlist[8][2] > lmlist[6][2] and lmlist[12][2] < lmlist[10][2] and lmlist[16][2] < lmlist[14][2] and \
lmlist[4][1] > lmlist[20][1] and lmlist[20][2] < lmlist[18][2]

8 Face recognition

One of the most prominent problems of computer vision system is the face recognition. Many developments which took place in recent years by many of the tech giants is one of most supporting statements. In 2017 September Apple.inc which is one the most break through companies in the world, during the WWDC event announced the Face ID technology.

These libraries can be used as per our requirements in order to solve our required problem. Development of many libraries and many other APIs using different technologies and in many supported programming languages is done.

Many technologies also find one or more face in the given image as per the given model. Development of so many pathways for a single problem may cause confusion while choosing the solution for the given problem. This Article will mainly look into the libraries such as face_recogniton and dlib in order to solve our problem of an access control system. The advantages and the limitations of the libraries will be discussed in the following paper. The reason and the feasibility of the best required solution will also be discussed. Before we discuss the above-mentioned points, Firstly, we discuss the basics of face recognition, we'll discuss the hurdles we face while solving the problem using these technologies, and their solutions. The research objective is to create a biometric access control system. The system would be designed such as the algorithm analyze the faces in the video stream then open the door only if there was a known face in front of the door. To come to such an outcome, we should firstly look into the following tasks:

- consider the methods and basic principles of face recognition
- Analyse the methods followed by the libraries to solve the problem These points are to be considered while constructing an algorithm, this is done using the libraries and the HOG method in python.

8.1 RELATED WORK

In these modern times where AI and machine learning have gained so much importance and usage in our day-to-day life, technologies of computer vision are developing actively, with these developments we are able to solve our problems more effectively and precisely. The result of active development has actually led to creation and updates of large number of libraries and solve the problems associated with computer vision. Particularly, in this article we discuss the task of implementing an access control System by recognizing faces by using the libraries such as face_recognition and dlib.

The best way to get acquainted with such libraries is to read the documentation that will in turn be useful in solving the task and the issues we face. The article [18] has actually helped with the documentation of the libraries as mentioned above. Articles [18] have focused on theoretical aspects of the issues and building of the face recognition system. The researchers [18] describe the actual methods and technologies for all stages of the development of the recognition system, since in the field of recognition, a huge number of unique solutions have been developed.

The researchers introduce the method of face recognition using the method of Support Vector Machines (SVM), which significantly improve the speed and efficiency of the process of comparison of faces. Scholars have introduced the facial landmark estimation algorithms and techniques which are used to position the faces in the frame. It in turn helps the model in identifying the faces more efficiently and increase the overall quality of the system we are trying to achieve. Based on the study of documentation and analysis we have come to a conclusion that there no single pathway to conduct a facial recognition rather there are many methods and ways one could achieve this goal. But, pertaining to this article we find it easier to use face_recognition library. We discuss the advantages and hurdles we face during the development of this system.

8.2 MATERIALS AND METHODS

The following problems should be addressed before devising an algorithm for the access control system.

- Find Faces – the faces should be recognized no matter what the input is a video from a camcorder or a video from a cc tv footage or any video
- Position of faces – In real world test cases we mainly find that the face is often rotated or not in right position i.e., towards the camera. The main objective of this point will be turning the photo such that it was directly taken facing the camera.
- Identifying the unique facial features – this step can be named as the main step in the facial recognition where the unique facial features of the face are acquired and stored in digital valued forms.
- Identifying the person – the received data from the input video is later compared to the data available with us, if both of the data is similar, we will draw a bounding box green in color, around the face of the person along with the names. If the person is not within the known values, we bound it with a red box.

The first step to develop such a kind of access control system the main step will be identifying a face from the given frame of video. If any face remains undetected or any other object is treated as a face the system, we are developing remains no good and the results which will be produced will be unsatisfactory. So, to overcome this problem we use one of the most popular algorithms for finding faces in an image which was invented in 2005 by Navneet Dalal and Bill Triggs, Histogram oriented gradient [18].

The algorithm works on the following steps. To find faces in an image, we'll start by making our image black and white because we don't need color data to find faces:

Our goal will be considering each and every pixel and consider their surrounding pixel and the draw arrows towards the darker pixel. The same processes are continued to all the pixel. At last, we arrive to a situation where all the pixels are replaced by arrows. These arrows are called gradients and they show the flow light in the frame.

We'll later break the image into squares of 16 x16 pixels and each block is replaced by the arrow direction which is the strongest in the given block. The end result will be a very simple representation of the face which is captured in the frame. Later the image is compared to the HOG pattern which was extracted from a bunch of training faces, the most common pattern with the known pattern and marked. This would result us the marking the faces in the obtained frame from the video we received as the input.

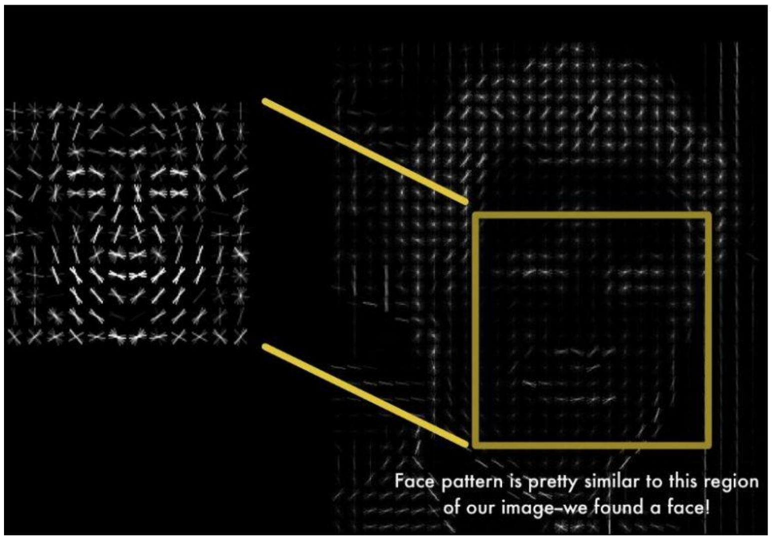

Face pattern is pretty similar to this region
of our image–we found a face!

Figure 50 Face Pattern

After finding the face in the image, the next problem we face is the positioning of the face, in most of the images the face not center positioned as it should be required by the algorithm otherwise it'll worsen the performance and accuracy of the algorithm. To solve this situation, we use the face landmark estimation which was invented by Vahid Kazemi and Josephine Sullivan.

The basic idea is we will come up with 68 specific points (called landmarks) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face.

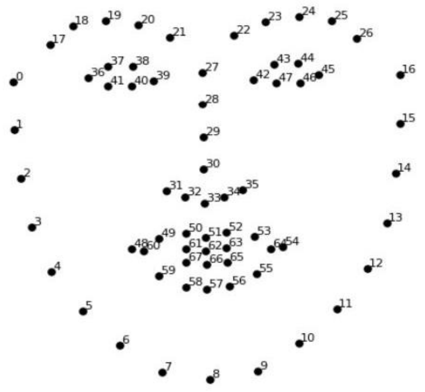


Figure 51 68 Specific Points On Any Face

Now that we have an idea where the mouth and eyes are, we'll rotate, scale, shear the image such that the eyes and mouth are centered as much as possible. By doing this step i.e., centering the face it'll help us a lot in the further step by making it more accurate and efficient.

The next step is to focus on is the way to tell different faces apart. We can do this by extracting basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements. For example, we might measure the size of each ear, or the space between our eyes etc. So, which measurements shall we take into account? Another problem we might face is while we have a huge database of faces the comparison process will take a lot of time by which it'll be quite inefficient process. At first glance we may feel that the main characteristics of face such as distance between the eyes or the distance between the ears or the length of the nose might be good for the main measurements of the face but later while we take deeper dive, we see that the computer doesn't perceive the face as a whole but it considers pixels of the image. To solve this problem, it was suggested that we can use a DCNN, which will be trained to identify 128 unique numerical facial features. But instead of training the network to recognize pictures objects like we did last time, we generate the 128 measurements: The training process works by following the below steps: • Load the training face image of a known person • Load another picture of the same person • Load a picture of a different person

The Deep convolutional network will adjust the results obtained from the images 1 and 2 such that the obtained 128 characteristics are as similar as possible and image which loaded in 3rd step is as different as possible. The next step includes the training the deep convolutional neural networks is to generate unique numerical values of the 128 characteristics from a huge no. of faces in the databases and it takes a lot of computational power to train the Deep convolutional neural network. Even though we use NVIDIA Tesla graphic card which has very powerful GPU cores from the company NVIDIA along with the support for CUDA the learning process takes about 24hrs of time. However, when the neural network will be trained it can take the input of the face never seen before and generate the unique characteristics instantly. This makes the step the most important among the other steps we have discussed, incomplete training may result in unsatisfactory results which may lead to inefficiency. If you do not want to train a model you can always use a pre-trained model and use it in your algorithm so that it can generate the features. In this article we follow the same technique by using a pre-trained model so that our work and economic costs are reduced by great amount.

The last step of the algorithm will be comparing the faces to available faces i.e., the available 128 features which were obtained in the previous step will be compared to the data we have, if we data will match, we enter the time in the file along with the name of the person which will help us create the access control System. The task we face is how do we compare the faces for this algorithm we choose to use the Support Vector Machine or SVM.

Using the available features, we train the classifier, the more homogenous the features are the better we can determine the face. The reason for this is that for the classifier we send the value in the form of data set:

$(\vec{x1}, y1), \ldots., (\vec{x}n, yn)$

where y is 1 or -1, and each one indicates the class to which the point ￼belongs. During the training, support vector method, our challenge is to find the maximum separation hyperplane (see Fig.3), simply put, the dataset is divided into classes so that they are difficult to confuse during comparison. The classifier's working time is several milliseconds, ideal for face-to-face identification. The below points can help in better precision and improved learning of the model [18]: • Between the data there should be clear intervals by which the model can recognize them into clear groups. • Huge data may not help in the betterment of model rather the training images should be with less noise and any data of any class should not overlap with another.

8.3 IMPLEMENTATION

Now let us build the attendance system as mentioned above. The above steps are followed while implementing the above-mentioned algorithm, a video stream inputs the frames into the access control system then the faces get analyzed and matched with the known faces If a match exists then the access control system will send a message informing the server that a known person is trying to enter the home. Then the server gives command to open the door.

9 CONCLUSION AND RECOMMENDATIONS

9.1 Conclusion

The home automation system is a framework based on Arduino or any other microcontroller, it helps people to rely on technology to turn their houses from traditional houses to smart houses, to help them reduce the usage of water or electricity, and enhance the way they live, by connecting the house appliances to that system for controlling to turn them on or off automatically or by the user via http requests or voice or a mobile application or hand signs, also to open the doors. along with these features.

In this project we designed, implemented, and tested our Home Automation System. The home automation system consists of an embedded system microcontroller, DHT11 sensor, gas sensor, and relays. Our Home Automation System allows us to control house appliances remotely by the mobile application. We implemented a http server that remotely controls home appliances such as Fan, Light, and TV. Also, We can know the state of each electrical device from the mobile app, which displays the state of appliances when it turned on or off.

Home Automation makes not only an economical but also an efficient use of electricity and reduces much of the wastage. Home automation makes housing activities easier, more accessible, secure, and efficient.

9.2 Recommendations

We hope to increase the control of other functions of home appliances and not just control the operation, and improve software from hackers' attack to be secure.

10 APPENDICES

Index 8051 software code

Main.c

```
#include<reg51.h>
#include "UART.h"
#include "DHT11.h"
#include "delay.h"
#include "ESP_interface.h"
#include "stepper_motor.h"

sbit RELAY1 =P2^0;
sbit RELAY2 =P2^1;
sbit fan =P2^3;
sbit doorstate =P2^4;
sbit RELAY3 =P2^5;
sbit RELAY4 =P2^6;
unsigned char door = 0;
unsigned char I_RH=0,D_RH,I_Temp=0,D_Temp,CheckSum;
short int counter=0;
short int counter_DHT=0;
extern unsigned char get_flag;
extern unsigned char OK_flag;
extern unsigned char ESP_Responce[36];
unsigned char int_to_string[3];
unsigned char int_to_string2[2];
unsigned char x=0,y=0;
unsigned int counter_out=0;
void main()
{

            UART_init();
            ESP_eInit();
            RELAY1=0;
            RELAY2=0;
            RELAY3=0;
            RELAY4=0;
            fan=0;
           doorstate=0;
            while(1){
                        if(counter_DHT > 3500){
                                    Request();       /* send start pulse */
                                    Response();     /* receive response */
                                    I_RH=Receive_data(); /* store first eight bit in I_RH */
                                    D_RH=Receive_data();            /* store next eight bit in D_RH */
                                    I_Temp=Receive_data();         /* store next eight bit in I_Temp */
                                    D_Temp=Receive_data();       /* store next eight bit in D_Temp */
                                    CheckSum=Receive_data();/* store next eight bit in CheckSum */
                                    if ((I_RH + D_RH + I_Temp + D_Temp) != CheckSum)
                                    {
                                                I_RH=0;
                                                I_Temp=0;
                                    }
```

```
                                        counter_DHT=0;

        }

                if(counter == 5000){

                        start_send();



                                        x=I_Temp/100;

                                        int_to_string[0]=x+'0';

                                        x=I_Temp-(x*100);

                                        y=x/10;

                                        int_to_string[1]=y+'0';

                                        int_to_string[2]=(x-(10*y))+'0';

                                        UART_sendString("AT+CIPSEND=23\r\n");

for(counter_out=0;OK_flag==0&&counter_out<1000;counter_out++){}
                                                        OK_flag=0;

                                                        ESP_Responce[0 ]='G';ESP_Responce[1 ]='E';
ESP_Responce[2 ]='T'; ESP_Responce[3 ]=' '; ESP_Responce[4 ]='/';

                                                ESP_Responce[5 ]='t';ESP_Responce[6 ]='e'; ESP_Responce[7
]='m'; ESP_Responce[8 ]='p'; ESP_Responce[9 ]=',';ESP_Responce[13 ]=' ';

                                          ESP_Responce[14 ]='H';ESP_Responce[15 ]='T'; ESP_Responce[16
]='T'; ESP_Responce[17 ]='P'; ESP_Responce[18 ]='/';

                                                        ESP_Responce[19 ]='1';ESP_Responce[20 ]='.';
ESP_Responce[21 ]='1'; ESP_Responce[22 ]='\r'; ESP_Responce[23 ]='\n';

                                                        ESP_Responce[10 ]=int_to_string[0];

                                                        ESP_Responce[11]=int_to_string[1];

                                                        ESP_Responce[12]=int_to_string[2];

                                                //-----------------------------------------------------

                                                        UART_sendString(ESP_Responce);
                                        Delay_ms(2);
                                        UART_sendString("AT+CIPCLOSE\r\n");
                             }else{*/
                                   y=I_Temp/10;
                                        int_to_string2[0]=y+'0';
                                        int_to_string2[1]=(I_Temp-(10*y))+'0';
                                       client_send2();

for(counter_out=0;OK_flag==0&&counter_out<50;counter_out++){
                                                  Delay_ms(100);
                                        }
                                        OK_flag=0;
                                        ESP_Responce[0 ]='G';ESP_Responce[1 ]='E';
ESP_Responce[2 ]='T'; ESP_Responce[3 ]=' '; ESP_Responce[4 ]='/';
                                   ESP_Responce[5 ]='t';ESP_Responce[6 ]='e'; ESP_Responce[7
]='m'; ESP_Responce[8 ]='p'; ESP_Responce[9 ]=',';ESP_Responce[12 ]=' ';
                               ESP_Responce[13 ]='H';ESP_Responce[14 ]='T'; ESP_Responce[15
]='T'; ESP_Responce[16 ]='P'; ESP_Responce[17 ]='/';
                                        ESP_Responce[18 ]='1';ESP_Responce[19 ]='.';
ESP_Responce[20 ]='1'; ESP_Responce[21 ]='\r'; ESP_Responce[22 ]='\n';ESP_Responce[23 ]='\0';
                                        ESP_Responce[10 ]=int_to_string2[0];
                                        ESP_Responce[11]=int_to_string2[1];
                               UART_sendString(ESP_Responce);
                               Delay_ms(2);
                                        client_close();
                    }
                    if(counter == 10000){
                         start_send();
                                   x=I_RH/100;
```

```
                                                            int_to_string[0]=x+'0';
                                                                x=I_RH-(x*100);
                                                                y=x/10;
                                                                int_to_string[1]=y+'0';
                                                                int_to_string[2]=(x-(10*y))+'0';
                                                            client_send1();

for(counter_out=0;OK_flag==0&&counter_out<50000;counter_out++){}
                                                        OK_flag=0;
                                                            ESP_Responce[0 ]='G';ESP_Responce[1 ]='E';
ESP_Responce[2 ]='T'; ESP_Responce[3 ]=' '; ESP_Responce[4 ]='/';
                                                            ESP_Responce[5 ]='h';ESP_Responce[6 ]='u';
ESP_Responce[7 ]='m'; ESP_Responce[8 ]='i'; ESP_Responce[9 ]=',';ESP_Responce[13 ]=' ';
                                                        ESP_Responce[14 ]='H';ESP_Responce[15 ]='T'; ESP_Responce[16
]='T'; ESP_Responce[17 ]='P'; ESP_Responce[18 ]='/';
                                                                ESP_Responce[19 ]='1';ESP_Responce[20 ]='.';
ESP_Responce[21 ]='1'; ESP_Responce[22 ]='\r'; ESP_Responce[23 ]='\n';ESP_Responce[24 ]='\0';
                                                                ESP_Responce[10 ]=int_to_string[0];
                                                                ESP_Responce[11]=int_to_string[1];
                                                                ESP_Responce[12]=int_to_string[2];
                                                                UART_sendString(ESP_Responce);
                                                                Delay_ms(2);


//-----------------------------------------------------------------------
                                                client_close();
                                                //UART_sendString("AT+CIPCLOSE=4\r\n");
                                                //client_request();
                                                counter=0;
                                }
```

```
                Delay_ms(2);

                counter++;

                counter_DHT++;

                if(get_flag == 0 ){

                                continue;

        }

                if ((ESP_Responce[16]=='l') && (ESP_Responce[17]=='e') &&
(ESP_Responce[18]=='d') && (ESP_Responce[19]=='1') && (ESP_Responce[20]=='o')&&
(ESP_Responce[21]=='n')){

                         RELAY1=1;

                }else if((ESP_Responce[16]=='l') && (ESP_Responce[17]=='e') &&
(ESP_Responce[18]=='d') && (ESP_Responce[19]=='1') && (ESP_Responce[20]=='o')&&
(ESP_Responce[21]=='f')){

                  RELAY1=0;

                }

                if ((ESP_Responce[16]=='l') && (ESP_Responce[17]=='e') &&
(ESP_Responce[18]=='d') && (ESP_Responce[19]=='2') && (ESP_Responce[20]=='o')&&
(ESP_Responce[21]=='n')){

                         RELAY2=1;

                }else if((ESP_Responce[16]=='l') && (ESP_Responce[17]=='e') &&
(ESP_Responce[18]=='d') && (ESP_Responce[19]=='2') && (ESP_Responce[20]=='o')&&
(ESP_Responce[21]=='f')){

                  RELAY2=0;

                }


                if ((ESP_Responce[16]=='l') && (ESP_Responce[17]=='e') &&
(ESP_Responce[18]=='d') && (ESP_Responce[19]=='4') && (ESP_Responce[20]=='o')&&
(ESP_Responce[21]=='n')){

                         RELAY3=1;

                }
```

```
                    else if((ESP_Responce[16]=='l') && (ESP_Responce[17]=='e') &&
(ESP_Responce[18]=='d') && (ESP_Responce[19]=='4') && (ESP_Responce[20]=='o')&&
(ESP_Responce[21]=='f')){

                       RELAY3=0;

                    }


                    if ((ESP_Responce[16]=='l') && (ESP_Responce[17]=='e') &&
(ESP_Responce[18]=='d') && (ESP_Responce[19]=='5') && (ESP_Responce[20]=='o')&&
(ESP_Responce[21]=='n')){

                               RELAY4=1;

                    }else if((ESP_Responce[16]=='l') && (ESP_Responce[17]=='e') &&
(ESP_Responce[18]=='d') && (ESP_Responce[19]=='5') && (ESP_Responce[20]=='o')&&
(ESP_Responce[21]=='f')){

                       RELAY4=0;

                    }

                    //door-------------------------------------

                    if ((ESP_Responce[16]=='l') && (ESP_Responce[17]=='e')  &&
(ESP_Responce[18]=='d') && (ESP_Responce[19]=='3')&& (ESP_Responce[20]=='o')&&
(ESP_Responce[21]=='n')){


                              if(door==0){

                                                doorstate=1;

                                                Left();

                                                door=1;

                                        doorstate=0;

                                        }

                    }else if((ESP_Responce[16]=='l') && (ESP_Responce[17]=='e')  &&
(ESP_Responce[18]=='d') && (ESP_Responce[19]=='3')&& (ESP_Responce[20]=='o')&&
(ESP_Responce[21]=='f')){

                                        if(door==1){

                                                doorstate=1;
                                        Right();

                                        door=0;

                                        doorstate=0;

                                }

                        }

                        //------------------------------------------

                        if ((ESP_Responce[16]=='f') && (ESP_Responce[17]=='a') &&
(ESP_Responce[18]=='n') && (ESP_Responce[19]=='o') && (ESP_Responce[20]=='n')){

                                fan=1;

                        }else if((ESP_Responce[16]=='f') && (ESP_Responce[17]=='a') &&
(ESP_Responce[18]=='n') && (ESP_Responce[19]=='o') && (ESP_Responce[20]=='f')){

                                fan=0;

                        }

                        get_flag=0;

                        //client_request();

                }


}
```

ESP_prog.c

```
// Include libraries

#include<reg51.h>

#include "STD_TYPES.h"

#include "BIT_MATH.h"


// Include files


#include "ESP_interface.h"
```

```
//#include "ESP_config.h"

//#include "ESP_private.h"

#include "delay.h"

 void setup_server(void){

	UART_sendString("AT+CIPMUX=1\r\n");

	while(OK_flag==0);

	OK_flag=0;

	UART_sendString("AT+CIPSERVER=1,80\r\n");

	while(OK_flag==0);

	OK_flag=0;

 }


void ESP_eInit(void)

 {


	Delay_sec(4);

	UART_sendString("AT+RST\r\n");

	Delay_sec(4);

	OK_flag=0;

	UART_sendString("ATE0\r\n");

	for(counter_out=0;OK_flag==0&&counter_out<50000;counter_out++){

	}

	OK_flag=0;

	UART_sendString("AT+CWJAP_CUR=\"Vodafone-7669\",\"30243660\"\r\n");

	while(OK_flag==0);

	OK_flag=0;

UART_sendString("AT+CIPSTA_DEF=\"192.168.2.14\",\"192.168.2.2\",\"255.255.255.0\"\r\n");
```

```
        for(counter_out=0;OK_flag==0&&counter_out<50000;counter_out++){
                Delay_ms(20);
        }
        setup_server();
 }
void start_send(void){
            setup_server();
            UART_sendString("AT+CIPSTART=4,\"TCP\",\"192.168.2.4\",8080\r\n");
                for(counter_out=0;OK_flag==0&&counter_out<50000;counter_out++){}
                        OK_flag=0;


}


void client_request(void){
            UART_sendString("AT+CIPCLOSE=0\r\n");
}
void client_send1(void){


UART_sendString("AT+CIPSEND=4,24\r\n");
}
void client_send2(void){


UART_sendString("AT+CIPSEND=4,23\r\n");
}
void client_close(void){
UART_sendString("AT+CIPCLOSE=4\r\n");
}
/*************** END OF FUNCTIONS ***********************************************/
```

Stepper.c

```
#include "stepper_motor.h"
#include "delay.h"
void Right(void){
                    unsigned char i=0;
       for(i=0; i<80; i++)
             {
                    Stepper_Port = 0x81;
                    Delay_ms(50);
                    Stepper_Port = 0x82;
                    Delay_ms(50);
             }
}
void Left(void){
                    unsigned char i=0;
                    for(i=0; i<80; i++)
             {
                    Stepper_Port = 0x88;
                    Delay_ms(50);
                    Stepper_Port = 0x84;
                    Delay_ms(50);
             }
}
```

Uart.c

```
#include<reg51.h>
```

```
#include "UART.h"

#include "ESP_interface.h"

void UART_init(void)

{

        SCON=0x50;                          //8-bit UART  &  Enable Reception.

  TMOD=0x21;                         //8-bit auto-reload timer1.

  TH1=TL1=0xfd;                      //Baud Rate 9600.

  TR1=1;                                        //Start Timer1.

}

void UART_sendByte(unsigned char byte)

{

        IE=0x00;

        SBUF=byte;

        while(TI==0);          //Waiting for the transmit interrupt flag.

        TI=0;

        IE=0x90;

}

unsigned char ESP_Responce[36]; // global var

unsigned char get_flag=0;

unsigned char OK_flag =0;

void Interrupt_TerimaChar() interrupt 4

{


     static unsigned char i =0;

            static unsigned char client =0;

            static unsigned char O_K =0;

     unsigned char ch;
```

```
TI=0;

if(RI){

ch = SBUF;

RI =0;

                        if(ch == '+'){

                                    client=1;

                                    i=0;

                         }

                        if(ch != 'K'&& O_K == 1 ){

                                    O_K=0;

                         }

                        if(ch == 'O'){


                                    O_K=1;

                         }

                                   if(ch == 'K'&& O_K == 1 ){

                                    OK_flag=1;

                                    O_K=0;

                         }


 if(client==1){

                                    ESP_Responce[i] = ch;

                          i++;

                                    if(ch == '\n'){

                                                      get_flag=1;

                                                      i=0;

                                                      client=0;
                                    client_request();
                                   }
   }
   ES =1;
  }
       }

void UART_sendString(unsigned char *string)
{
       unsigned char i = 0;
       IE=0x00;
       while(string[i] != '\0'){
              UART_sendByte(string[i]);
              i++;
 }
       IE=0x90;
}
```

11 Refrences

[1] What is the IoT value chain and why is it important? (analysysmason.com) 13/5/2022

[2] Stolojescu-Crisan, C., Crisan, C., & Butunoi, B. P. (2021). An IoT-based smart home automation system. *Sensors*, *21*(11), 3784.

[3] Upton, E.; Halfacree, G. Raspberry Pi User Guide, 4th ed.; John Wiley and Sons Ltd.: Hoboken, NJ, USA, 2016.

[4] Hypertext Transfer Protocol - Wikipedia 16/5/2022

[5] RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1 (ietf.org) 16/5/2022

[6] Universal asynchronous receiver-transmitter - Wikipedia 17/5/2022

[7] UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices 17/5/2022

[8] Raspberry Pi - Wikipedia 20/5/2022

[9] ESP32 - Wikipedia 24/5/2022

[10] Raspberry Pi - Introduction (tutorialspoint.com)

[11] DHT 11 Humidity & Temperature Sensor - Geeetech Wiki 28/5/2022

[12] Gas Sensor - an overview | ScienceDirect Topics 11/6/2022

[13] What is a Gas Sensor? Construction, Types & Working of Gas Sensors (components101.com) 11/6/2022

[14] Stepper motor - Wikipedia 13/6/2022

[15] Pin diagram of 8051 Microcontroller - GeeksforGeeks 13/6/2022

[16] Patton, E. W., Tissenbaum, M., & Harunani, F. (2019). MIT app inventor: Objectives, design, and development. In *Computational thinking education* (pp. 31-49). Springer, Singapore.

[17] Assefi, M., Liu, G., Wittie, M. P., & Izurieta, C. (2015). An experimental evaluation of apple siri and google speech recognition. *Proccedings of the 2015 ISCA SEDE*, *118*.

[18]] Tambi, P., Jain, S., & Mishra, D. K. (2019). Person-dependent face recognition using histogram of oriented gradients (HOG) and convolution neural network (CNN). In *International Conference on Advanced Computing Networking and Informatics* (pp. 35-40). Springer, Singapore.

[19] Harris, M., & Agoes, A. S. (2021, November). Applying Hand Gesture Recognition for User Guide Application Using MediaPipe. In *2nd International Seminar of Science and Applied Technology (ISSAT 2021)* (pp. 101-108). Atlantis Press.

الملخص العربي

إنترنت الأشياء (IoT) هي تقنية ناشئة تجعل عالمنـا أكثـر ذكـاءً. لا يمكـن تخيـل فكـرة العـالم المتصل بدون إنترنت الأشياء. إنترنت الأشياء هو نظام يستخدم أجهزة الكمبيـوتر أو الأجهـزة المحمولة للتحكم في الوظائف والميزات المنزلية الأساسية تلقائيًا من خـلال الإنترنـت مـن أي مكـان حـول العـالم ، ويسـمى المنـزل الآلـي أحيانًـا المنـزل الـذكي. يهـدف إلـى تـوفير الطاقـة الكهربائية وطاقة الإنسان وسلامته. يختلف نظام أتمتة المنزل عن الأنظمة الأخرى من خـلال السماح للمستخدم بتشغيل النظام من أي مكان حول العالم من خلال الاتصال بالإنترنت. يهدف هذا المشروع إلى تطوير منزل ذكي يتم التحكم فيه بالصوت والإيماءات باستخدام WI-FI و IOT، والذي يتم التحكم فيه عن بُعد بواسطة أي هاتف ذكي يعمل بنظام تشغيل Android أو ميكروفونات منزلية أو كاميرات. تصبح أتمتة المنزل أمرًا حيويًـا ، حيـث تمـنح المسـتخدم الراحة ونظامًا خاليًا من المتاعب لاستخدام الأجهزة المنزلية ، مما يجعل الحياة أفضل وأسـهل وأكثر أمانًا.

نظام أتمتة المنزل هو إطار عمل يعتمد على Arduino أو أي متحكم آخر ، فهو يساعد الناس على الاعتماد على التكنولوجيا لتحويل منازلهم من منازل تقليدية إلى منازل ذكية ، لمساعدتهم على تقليل استخدام المياه أو الكهرباء ، وتحسين طريقة حياتهم ، من خلال توصيل الأجهزة المنزلية بهذا النظام للتحكم في تشغيلها أو إيقاف تشغيلها تلقائيًا أو بواسطة المستخدم عبر طلبات http أو الصوت أو تطبيق الهاتف المحمول أو الإشارات اليدوية ، وكذلك لفتح الأبواب. إلى جانب هذه الميزات.

في هذا المشروع ، قمنا بتصميم وتنفيذ واختبار نظام أتمتة المنزل الخاص بنا. يتكون نظام التشغيل الآلي للمنزل من متحكم نظام مضمن ، ومستشعرDHT11 ، ومستشعر الغاز ، والمرحلات. يسمح لنا نظام أتمتة المنزل لدينا بالتحكم في الأجهزة المنزلية عن بُعد من خلال تطبيق الهاتف المحمول. قمنا بتطبيق خادم http الذي يتحكم عن بعد في الأجهزة المنزلية مثل المروحة والضوء والتلفزيون. أيضًا ، يمكننا معرفة حالة كل جهاز كهربائي من تطبيق الهاتف المحمول ، والذي يعرض حالة الأجهزة عند تشغيلها أو إيقاف تشغيلها.

لا تجعل أتمتة المنزل استخدامًا اقتصاديًا فحسب ، بل توفر أيضًا استخدامًا فعالًا للكهرباء وتقلل الكثير من الفاقد. تجعل أتمتة المنزل أنشطة الإسكان أسهل وأكثر سهولة وأمانًا وفعالية.