Misr International University
جامعة مصر الدولية

Faculty of Engineering

Department of Electronics and Communication

"IoT Smart Kitchen Automation and Monitoring with ESP8266"

IoT Project

Submitted by:

Name	Student ID
Habiba Hassan Soliman Hassan	ID: 2018/11594
Hussein Ahmed Hussein Hassan	ID: 2018/15934
Yasmin Amr Ahmed Helmy	ID: 2018/03452

Supervised by:

Assoc. Prof. Mohamed Ali Mohamed ElZorkani

Eng. Ayman Shabana

Project Description

It has become remarkably common these days to hear that a house or a building was on fire due to gas leakage or a forgotten stove burner. This undeniable truth demands immediate attention. In such a world that hinges upon natural gas, a smart kitchen is much needed. Are you capable of monitoring your kitchen at all times every day? Surely not, but sensors can!

One no more needs to continuously check for leakage either it be water or gas. Why do so when water & gas sensors can? Besides, it is not necessary to take a leave from a good day out thinking you forgot your stove burners on when your fire sensors are there to do the job for you. How is this? Fire, water, and gas sensors can signal to actuators such as DC fans, servo, and stepper motors in order to take action:

- Stepper and servo motors respectively close water & gas valves to stop any further leakage.
- A DC fan is used to clear away excess and already leaked gas.
- Servo motors can also control stove knobs in order to shut the fire out.

The project idea is not over yet! If a person is present at home at such times, it is important that his/her attention is drawn to such risk, that is why a buzzer intervenes immediately causing a noisy signal to alert the resident.

Over and above that, the system shall keep track of the temperature of the fridge in case it drops down due to the electricity going off or for any similar reason. Upon the occurrence of such problems, the power source feeding the fridge is diverted so that the food does not rot. A light emitting diode (LED) is set to the fridge as an indicator whether it is powered on or off.

The project is implemented using the late concept of IOT (Internet of Things) as it is proven to be environmentally friendly due to its little energy consumption. The used software for display is “Blynk” and the microcontroller needed is the ESP8266 NodeMCU module.

Project Motivation: This is a matter of life & death, and the goal is always to save the valuable human life. According to the National Fire Protection Association in the USA, every year, there is an average of 358,300 home-based fires. This is in the US alone, and with the lack of media coverage elsewhere, kitchen fires can be a very serious worldwide problem.

"It can just take 30 seconds for a flame to turn into a massive blaze."

System Architecture

Smoke Sensor (D0)

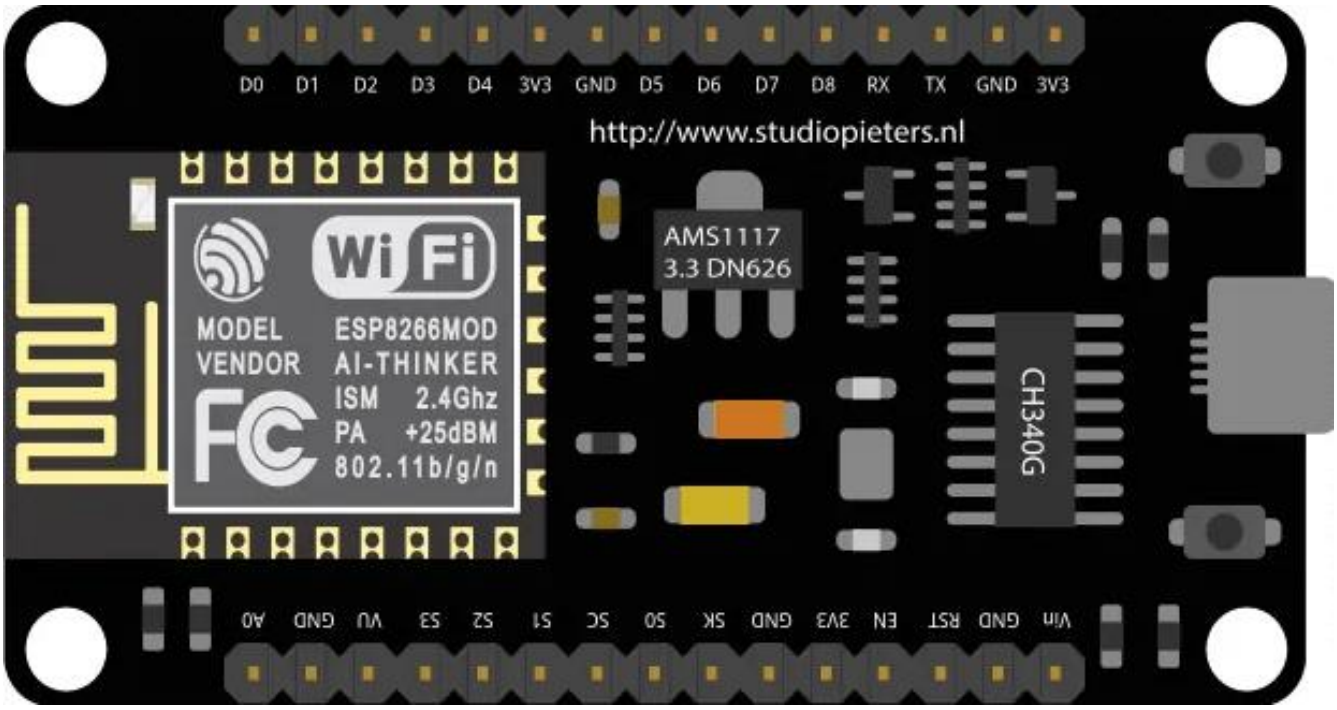
Flame Sensor (D1)

Temperature Sensor (D2)

Servo Motor (D3)

Stepper Motor (D4 TO D7)

Buzzer (D8)



Water Sensor (A0)

DC Fan (D8)

LED (TX)

Hardware Components

Sensors

1) Water Detection Sensor Module

2) Flame Sensor Module

3) Methane Gas Sensor MQ4

4) Temperature Humidity Sensor Module (DHT11)

Actuators

1) Servo Motor

2) Stepper Motor

3) DC Fan

4) Led

5) Buzzer

Roles

1. The DC fan functions as an exhaust fan, while the servo motor serves as a gas valve. When the gas level exceeds a set threshold, the gas valve is automatically closed, and the DC fan turns on to eliminate the gas until no further gas is detected. Furthermore, a buzzer acts a danger indicator to warn nearby people if gas is detected.
2. The person can manually turn the DC fan on and off through the GUI provided by the Blynk application. However, they cannot turn the fan off if there is gas present in the surrounding area. This is employed for the safety of the user.
3. Cooker status is displayed on the Blynk GUI for to monitor the stove. This is done to allow the user to check if they have accidentally left the stove on when leaving the house. It comes with a manual option to turn the gas valve on and off. However, they cannot turn the gas valve on if there is gas present in the surrounding area. To ensure user safety, this measure is likewise implemented.
4. If the temperature of the fridge exceeds a certain threshold, an external fridge source would be activated, with the LED functioning as an indicator of its activation.
5. The stepper motor acts as the water valve. If water is detected above a certain threshold, the water valve automatically closes to stop the water overflow. Additionally, the person can manually turn the water valve on

and off through the Blynk GUI. Despite that, if water is detected above the threshold, and the user tries to manually turn the water valve switch on, he won't be able to.

Software Platforms and Frameworks

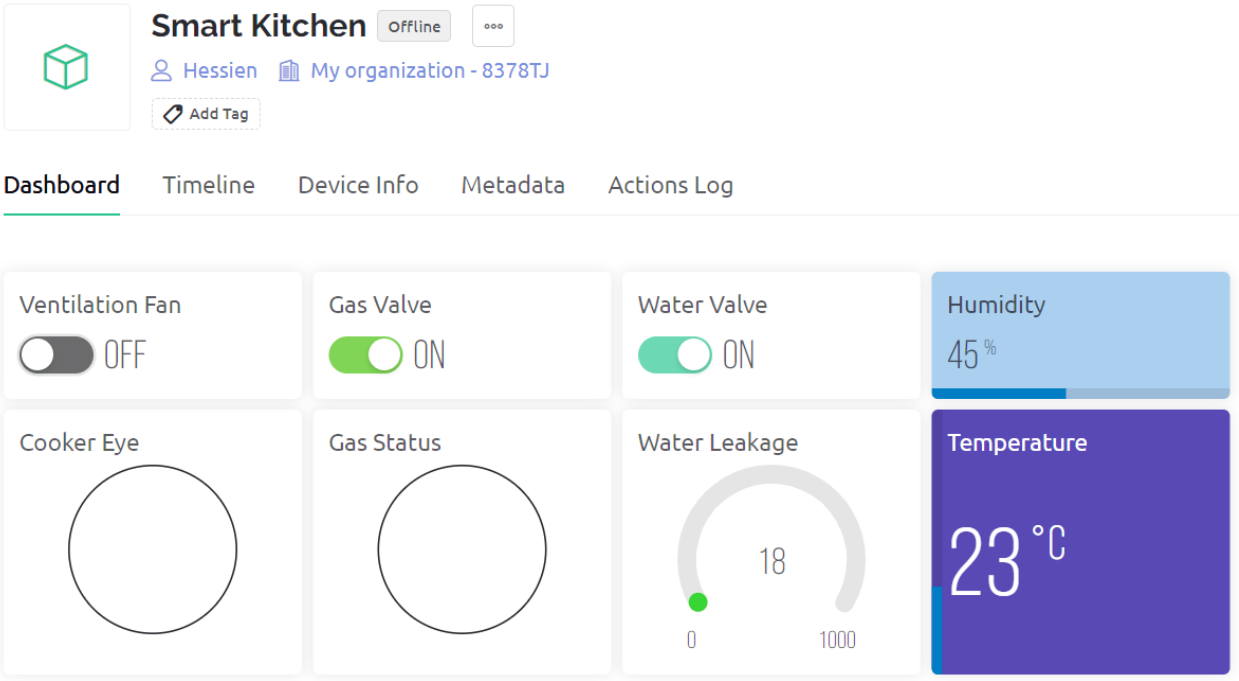
This project utilizes the NodeMCU ESP8266 microcontroller board programmed using the Arduino IDE. The software platform and framework for the IoT smart kitchen project are based on the Arduino ecosystem, which includes the Arduino IDE for code development and the libraries and APIs provided by the Arduino platform.

Communication & Networking Protocols

The Communication used in this project is WIFI while connecting the esp8266 to an access point and the Blynk application and GUI is connected to the same or a different access point, both connections should provide internet access. The Networking protocol used is a communication protocol used by the Blynk cloud servers which is Transport Layer Security protocol (TLS).

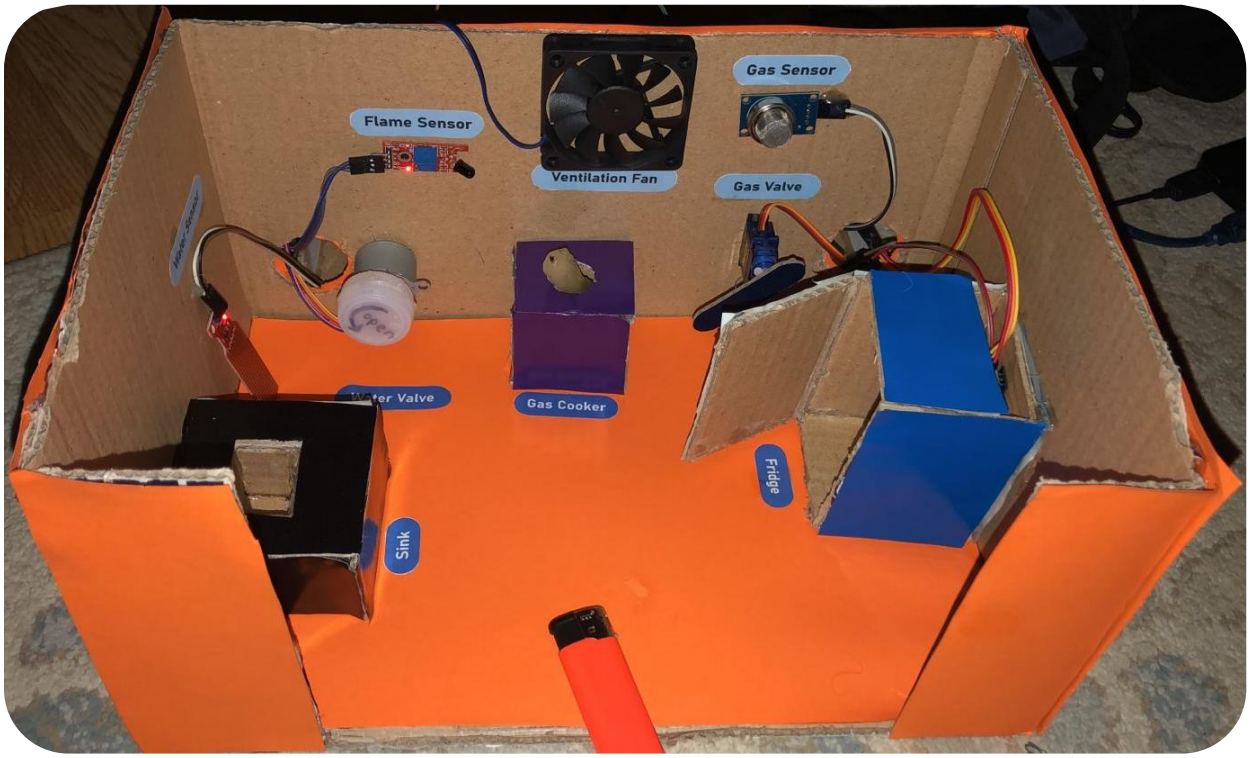
User Interface and Control

The proposed graphical user interface in Blynk application is achieving our purpose for controlling actuators and visualizing sensors' readings. Each widget is having a virtual pin to be assigned, these virtual pins will be used in the Arduino code to have full access to the widget either reading or writing from it. The following figure will show the graphical user interface, which contains three switches for gas valve, water valve, and fan, moreover, five indicators for gas detection, fire presence, water leakage, temperature, and humidity.



Prototype

The following image will show our prototype for the Smart Kitchen Project:



Code

```
#include <Servo.h>
#define BLYNK_PRINT Serial
#include <SPI.h>
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define water A0
#define gas D0
#define fire D1
#define temp D2
#define servopin D3
#define stepper1 D4
#define stepper2 D5
#define stepper3 D6
#define stepper4 D7
#define fan D8
#define led 1
Servo servo;

#define BLYNK_TEMPLATE_ID "TMPL2ryEZuPjZ"
#define BLYNK_TEMPLATE_NAME "new"
#define BLYNK_AUTH_TOKEN "_JgpkeEA5JBeZNQo4DGoqh3pIAvqCT1z"

char ssid[] = "Yasmin's Phone";
char pass[] = "hoda1234";
char gas_valve_check;  // Checks if gas valve is open or closed: 1 -> open  0 ->
closed
char openWaterValve=0;
char fan_check;  //used to read fan switch value from blynk write function
int sensorthres = 500;    //Water Threshold
int step_number = 0;    //counts the 4 coil steps to reach 1 full step
int counter =0;  // counts number of full steps taken , until 512 steps. Max
speed: 512 steps/sec, full revolution needs 2048 steps -> 4 seconds.
bool direc=true; // true clockwise (open) false anticlockwise (close)
char water_valve_check=1;  // Checks if water valve is open or closed: 1 ->
open  0 -> closed
bool WaterValveCommand=false;  //used to read water switch value from blynk write
function

BlynkTimer timer;
#define DHTTYPE DHT11
#define DHTPIN D2
```

```
DHT dht(DHTPIN, DHTTYPE);

// The following function is detecting water presence
void sendwater(){
  int data_water = analogRead(water);     //Read water value
  Blynk.virtualWrite(V4,data_water);    //Send water value to blynk
  Serial.print("pin A0 water: ");
  Serial.println(data_water);
  if(data_water>=sensorthres && water_valve_check==1)      //If water value>
threshold & water valve is open
  {
    FullStep(true); //Close Water Valve
    water_valve_check=0; //Water valve is closed
    Blynk.virtualWrite(V7, 0); //Turn of water value switch in blynk GUI
    WaterValveCommand=false; //Switch is off
  }
  }

// The following function is measuring gas
void checkGasSensor() {
  int gasvalue = digitalRead(gas);
  Serial.println(gasvalue);
  Blynk.virtualWrite(V9, !gasvalue);   //sending !gasvalue as 0 digital voltage-
>gas detected
  if (gasvalue == 0) {   //gas detected
    servo.write(90); //closed
    gas_valve_check = 0; // gas valve is closed
    digitalWrite(fan,HIGH); //fan & buzzer
    Blynk.virtualWrite(V5, 1);  //Turn fan switch off in GUI
  }
  else if(gasvalue == 1 && fan_check==1){    //if no gas is detected, and fan
switch is on
    digitalWrite(fan,HIGH); //fan & buzzer
    gas_valve_check = 1; // gas valve is ready to open
  }
  else{
    digitalWrite(fan,LOW);
    gas_valve_check = 1; // gas valve is ready to open
  }
}
// The following function is detecting fire
void Flame_Detection()
{
  int flame_sensor_read = digitalRead(fire); //reading the sensor data on D0 pin
  Serial.print("Flame Status:");
```

```
  Serial.println(flame_sensor_read);
  Blynk.virtualWrite(V8,flame_sensor_read);
}
// The following function is measuring temperature and humidity
void sendTemp()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Blynk.virtualWrite(V1, h);  //V5 is for Humidity
  Blynk.virtualWrite(V0, t);  //V6 is for Temperature
  Serial.print("Temperature: ");
  Serial.println(t);
  Serial.print("Humidity: ");
  Serial.println(h);
  if(t>30)
  {
    digitalWrite(led,HIGH);     //Refrigerator is not cooling
  }
  else
  {
     digitalWrite(led,LOW);   //Refrigerator is working as usual
  }
}

void FullStep(bool dir){
    counter=0;
while(counter<512){   //512 steps
    if(dir){   //clockwise
switch(step_number){
  case 0:                                 //determines which coil of the stepper
motor should be energized to achieve the desired step.
  digitalWrite(stepper1, HIGH);
  digitalWrite(stepper2, LOW);
  digitalWrite(stepper3, LOW);
  digitalWrite(stepper4, LOW);
  break;
  case 1:
  digitalWrite(stepper1, LOW);
  digitalWrite(stepper2, HIGH);
  digitalWrite(stepper3, LOW);
```

```
  digitalWrite(stepper4, LOW);
  break;
  case 2:
  digitalWrite(stepper1, LOW);
  digitalWrite(stepper2, LOW);
  digitalWrite(stepper3, HIGH);
  digitalWrite(stepper4, LOW);
  break;
  case 3:
  digitalWrite(stepper1, LOW);
  digitalWrite(stepper2, LOW);
  digitalWrite(stepper3, LOW);
  digitalWrite(stepper4, HIGH);
  break;
}
  }else{    //anticlockwise
    switch(step_number){
  case 0:
  digitalWrite(stepper1, LOW);
  digitalWrite(stepper2, LOW);
  digitalWrite(stepper3, LOW);
  digitalWrite(stepper4, HIGH);
  break;
  case 1:
  digitalWrite(stepper1, LOW);
  digitalWrite(stepper2, LOW);
  digitalWrite(stepper3, HIGH);
  digitalWrite(stepper4, LOW);
  break;
  case 2:
  digitalWrite(stepper1, LOW);
  digitalWrite(stepper2, HIGH);
  digitalWrite(stepper3, LOW);
  digitalWrite(stepper4, LOW);
  break;
  case 3:
  digitalWrite(stepper1, HIGH);
  digitalWrite(stepper2, LOW);
  digitalWrite(stepper3, LOW);
  digitalWrite(stepper4, LOW);
  break;
}
  }
step_number++;   //increment step number after breaking case
  if(step_number > 3){     //full step has been completed
```

```
    counter++;   //increment steps
    step_number = 0;  //repeat step
  }
  delay(2);   //minimum delay=2ms
}
}

BLYNK_WRITE(V5) {
  int value = param.asInt();  //Read Fan Switch
  if(value==1){
    digitalWrite(fan, HIGH);
    fan_check =1; //Switch is on
  }
  else{
    digitalWrite(fan, LOW);
    fan_check=0;   //Switch is off
  }
}
BLYNK_WRITE(V6) {
  int value = param.asInt(); //Read Gas Valve Switch
  if(value ==1){
    servo.write(180); //open
  }
  else{
    servo.write(90); //closed
  }
}
BLYNK_WRITE(V7) {
  int value = param.asInt(); //Checks Water Valve Switch
  if(value ==1){
    WaterValveCommand=true;  //switch open
  }
  else{
    WaterValveCommand=false; //switch closed
  }
}
// This function is called every time the device is connected to the Blynk.Cloud
BLYNK_CONNECTED()
{
  Blynk.setProperty(V3, "offImageUrl", "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.png");
  Blynk.setProperty(V3, "onImageUrl",  "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations_pressed.png");
  Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-started/what-do-
i-need-to-blynk/how-quickstart-device-was-made");
```

```
}

// This function sends Arduino's uptime every second to Virtual Pin 2.
void myTimerEvent()
{
  Blynk.virtualWrite(V2, millis() / 1000);
}

void setup()
{
  // Debug console

  Serial.begin(9600);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);    //Connect to Blynk
  servo.attach(servopin);   //Attach the Servo variable to a pin.
  dht.begin();
  pinMode(gas, INPUT);
  pinMode(fire, INPUT);
  pinMode(fan,OUTPUT);
  pinMode(led,OUTPUT);
  pinMode(stepper1,OUTPUT);
  pinMode(stepper2,OUTPUT);
  pinMode(stepper3,OUTPUT);
  pinMode(stepper4,OUTPUT);
  // You can also specify server:
  timer.setInterval(1000L, myTimerEvent);
}

void loop()
{
  Blynk.run();
  timer.run();
  //checking senor values
  sendwater();
  sendTemp();
  checkGasSensor();
  Flame_Detection();
  if(gas_valve_check == 0)
  {
    Blynk.virtualWrite(V6, 0);    //Turn gas valve switch off in GUI if gas valve
is off
  }
```

```
  //water valve check makes sure that no water is overflowing (im not giving a
command to the stepper motor to close while giving it a command to open at the
same time)
  //if there is water overflowing, watervalvecommand will always be false, so the
water valve will not be opened.

  if(WaterValveCommand==true && water_valve_check==0) //if water valve switch is
on & water valve is closed -> Open Water Valve
  {
    direc=false;
    water_valve_check=1;  //water valve is open
    FullStep(direc);
  }

  else if(WaterValveCommand==false && water_valve_check==1) //If water valve
switch is off & water valve is open ->Close Water Valve
  {
    direc=true;
    water_valve_check=0;   //water valve is closed
    FullStep(direc);
  }

}
```