**IMPLEMENTATION:**

There is no standard implemented algorithm. The task of arduino uno is very simple. It connects all the sensors and stores the data in its memory.

**Arduino code snippets:**

1. #include<DHT.h>

The library used for reading temperature and humidity via DHT11 sensor

2. #define MQ2\_PIN A0

3. #define MQ3\_PIN A1

4. #define MQ4\_PIN A2

5. #define MQ135\_PIN A3

6. #define LDR A4

7. #define DHT11\_PIN 8

8. #define DHTTYPE DHT11

Line-2 to Line-7 defines the constants says which sensor maps to arduino pins

Line-8 defines the type of DHT sensor

9. float MQ2\_read=0, MQ3\_read=0, MQ4\_read=0, MQ135\_read=0;

10. float temperature=0, humidity=0;

11. float LDRReading = 0;

12. DHT dht(DHT11\_PIN, DHTTYPE);

Line-9 to Line-11 declares the variables which are used to read the sensor data. All variables are initialized to zero so that it will display zero if no data is collected.

Line-12 initializes the dht object and this object is used to read the temperature and humidity values.

13. void setup() {

14. Serial.begin(9600);

15. dht.begin();

16. pinMode(MQ2\_PIN, INPUT);

17. pinMode(MQ3\_PIN, INPUT);

18. pinMode(MQ4\_PIN, INPUT);

19. pinMode(MQ135\_PIN, INPUT);

20. }

The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup() function will only run once, after each powerup or reset of the Arduino board.

Line-14 passes the value 9600 to the speed parameter. This tells the Arduino to get ready to exchange messages with the Serial Monitor at a data rate of 9600 bits per second.

Line-15 enables or begins the DHT11 sensor

Line-16 to Line-19 configures the specified pin to behave as input.

21. void loop() {

22. MQ2\_read = analogRead(MQ2\_PIN);

23. MQ3\_read = analogRead(MQ3\_PIN);

24. MQ4\_read = analogRead(MQ4\_PIN);

25. MQ135\_read = analogRead(MQ135\_PIN);

26. LDRReading = analogRead(LDR);

27. temperature = dht.readTemperature();

28. humidity = dht.readHumidity();

The loop() function (Line-21 to Line-42) does precisely what its name suggests, and loops consecutively, allowing your program to change and respond.

Line-22 to Line-26 reads the analog values from the corresponding sensors and stores accordingly.

Line-27 to Line-28 reads the temperature and humidity values and the dht object as predefined functions to do the task.

29. Serial.print(LDRReading);

30. Serial.print(" ");

31. Serial.print(MQ2\_read);

32. Serial.print(" ");

33. Serial.print(MQ3\_read);

34. Serial.print(" ");

35. Serial.print(MQ4\_read);

36. Serial.print(" ");

37. Serial.print(MQ135\_read);

38. Serial.print(" ");

39. Serial.print(temperature);

40. Serial.print(" ");

41. Serial.println(humidity);

42. delay(40000);

Line-29 to Line-41 displays the recorded values on the serial monitor in a single line.

Line-42 pauses the execution for 40 seconds.

**Nodemcu code snippets:**

The stored sensor data is send to nodemcu module via serial communication. The nodemcu is like the heart of the project. Nodemcu’s task is to connect to Wifi, send data to cloud that is thingspeak and sending emails.

1. #include<SoftwareSerial.h>

2. #include <ESP8266WiFi.h>

3. #include <ThingSpeak.h>

The SoftwareSerial library (Line-1) has been developed to allow serial communication on other digital pins of the Arduino.

Line-2 library is used for nodemcu module. It is various features which makes our task easy.

Line-3 library is used for thingspeak connection and data uploading to cloud.

4. SoftwareSerial mySerial(D1, D2);

Line-4 initializes softwareSerial object and D1, D2 denotes the RX and TX pins of the nodemcu module.

5. char\* ssid = "\*\*\*\*\*\*\*";

6. char\* password = "\*\*\*\*\*\*\*";

Line-5 denotes the WiFi SSID and Line-6 denotes the wifi password.

7. const char\* host = "maker.ifttt.com";

Line-7 defines the website name which is used to create an applet for send emails.

8. WiFiClient client;

9. const int httpPort = 80;

Line-8 creates a client that can connect to a specified internet IP address and port

Line-9 denotes the standard http port number

10. unsigned long myChannelNumber = 1400396;

11. const char\* myWriteAPIKey = "08E4IMSBJX21ZBPP";

Line-10 and Line-11 denotes the channel number and write api key which was created (in thingspeak.com )

12. String message, LDR, MQ2, MQ3, MQ4, MQ135, temperature, humidity;

13. int count\_spoiled=0,count\_air\_quality=0, count=0;

Line-12 and Line-13 are the variables which are used to read the sensor data and counters to control the flow of emails

12. void setup() {

13. Serial.begin(115200);

14. mySerial.begin(9600);

15. ThingSpeak.begin(client);

Line-13 enables the serial monitor for nodemcu module.

Line-14 enables and begins connection to arduino uno with the speed parameter as 9600 bitsPerSecond.

Line-15 enables and begins the thingspeak connection.

16. WiFi.mode(WIFI\_STA);

17. WiFi.begin(ssid,password);

18. while(WiFi.status()!=WL\_CONNECTED) {

19. Serial.print(".");

20. delay(500);

21. }

22. Serial.println("");

23. Serial.print("IP Address: ");

24. Serial.println(WiFi.localIP());

25. }

Line-16 denotes the mode of Wifi which is station mode

Line-17 starts the connection with respect to the given wifi ssid and wifi password.

Line-18 to Line-21 loop terminates until the wifi is connected to nodemcu.

Line-22 to Line-25 prints the IP address of the WiFi connected.

26. void loop() {

27. message = mySerial.readStringUntil('\n');

28. LDR = getValue(message,' ',0);

29. MQ2 = getValue(message,' ',1);

30. MQ3 = getValue(message,' ',2);

31. MQ4 = getValue(message,' ',3);

32. MQ135 = getValue(message,' ',4);

33. temperature = getValue(message,' ',5);

34. humidity = getValue(message,' ',6);

35. Serial.println(message);

36. thingspeak(temperature, humidity, MQ2, MQ3, MQ4, MQ135, LDR);

37. sending\_email();

38. delay(40000); }

Line-27 reads the sensor data send from arduino in an single line.

Line-28 to Line-34 getValue function splits the message and the corresponding sensor data is being stored into variables accordingly.

Line-36 calls the thingspeak function which sends the data to cloud.

Line-37 calls the sending\_email function which sends the mails to the user.

39. String getValue(String data, char sep, int index) {

40. int found = 0;

41. int strIndex[] = {0, -1};

42. int maxIndex = data.length()-1;

43. for(int i=0; i<=maxIndex && found<=index; i++){

44. if(data.charAt(i)==sep || i==maxIndex){

45. found++;

46. strIndex[0] = strIndex[1]+1;

47. strIndex[1] = (i == maxIndex) ? i+1 : i;

48. }

49. }

50. return found>index ? data.substring(strIndex[0], strIndex[1]) : "";

51. }

Line-39 to Line-51 takes data, sep and index as parameters and returns the string which has corresponding index word. So, by this function we can split the data by seperator which is provided.

52. void thingspeak(String temperature, String humidity, String MQ2, String MQ3, String MQ4, String MQ135, String LDR) {

53. ThingSpeak.setField(1, temperature);

54. ThingSpeak.setField(2, humidity);

55. ThingSpeak.setField(3, MQ2);

56. ThingSpeak.setField(4, MQ3);

57. ThingSpeak.setField(5, MQ4);

58. ThingSpeak.setField(6, MQ135);

59. ThingSpeak.setField(7, LDR);

60. int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

61. if(x == 200){

62. Serial.println("Channel update successful."); }

63. else {

64. Serial.println("Problem updating channel. HTTP error code " + String(x)); } }

Line-53 to Line-59 sets which field maps to sensor variable.

Line-60 writes the sensor data to corresponding fields and returns value

Line-61 to Line-64 checks whether the data is send to cloud or not. If all is properly done, Line-60 returns 200 otherwise returns -210

65. void sending\_email() {

66. if(!client.connect(host, httpPort)) {

67. Serial.println("email connection failed");

68. return; }

69. if(count == 10) {

70. count\_spoiled = 0;

71. count\_air\_quality = 0;

72. count=0; }

73. if(count\_spoiled!=3 && MQ2.toInt() >= 470 && MQ3.toInt() >= 345 && MQ4.toInt() >= 280) {

74. String url1 = "/trigger/food\_spoiled/with/key/oOAEkILKPPinkD1S3IJ5I7FCYNdDNmWFzlgsNFoqBD";

75. Serial.print("Requesting url: ");

76. Serial.println(url1);

77. client.print(String("GET ") + url1 + " HTTP/1.1\r\n" +

"Host: " + host + "\r\n" +

"Connection: close\r\n\r\n");

78. count\_spoiled += 1;

79. count+=1; }

80. else {

81. Serial.println("email not sent because food is not spoiled yet."); }

82. delay(5000);

83. if(count\_air\_quality!=3 && LDR.toInt() > 300 && MQ135.toInt() > 270) {

84. String url = "/trigger/air\_quality\_and\_light\_around\_the\_food\_is\_not\_good-Move\_your\_food\_to\_a\_darker\_place/with/key/oOAEkILKPPinkD1S3IJ5I7FCYNdDNmWFzlgsNFoqBD";

85. Serial.print("Requesting url: ");

86. Serial.println(url);

87. client.print(String("GET ") + url + " HTTP/1.1\r\n" +

"Host: " + host + "\r\n" +

"Connection: close\r\n\r\n");

88. count\_air\_quality += 1;

89. count+=1; }}

90. else {

91. Serial.println("email not sent because air quality is good."); }

92. count+=1; }