Project description

**Introduction**

The following project uses a soil moisture sensor, DHT11 temperature sensor, and ESP8266 to collect temperature, humidity, and moisture readings and send them to ThingSpeak. This article will help you learn the basics of using ESP8266 and send a sensor data to a website via API like ThingSpeak.

**Sensors Brief**

A close-up of a blue and black electronic device

Description automatically generated with low confidence

DHT11 Sensor

﻿

**DHT11** humidity & temperature Sensor Module is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and outputs a digital signal on the data pin (no analog input pins needed).

A picture containing circuit component, electronic engineering, electronic component, passive circuit component

Description automatically generated

Soil Moisture Senor

﻿

**Soil moisture** sensor module is most sensitive to the ambient humidity is generally used to detect the moisture content of the soil; Module in the soil humidity less than a set threshold value when the DO port output high, when the when soil humidity exceeds the threshold value is set, the module D0 output low; Small plates digital outputs D0 can be directly connected with the microcontroller, microcontroller to detect high and low, and thus to detect soil moisture; Small plates digital output DO shop relay module can directly drive the buzzer module, and which can form a soil moisture alarm equipment; Small plates analog output AO and AD module connected through the AD converter, you can get more accurate values of soil moisture.

**Programming ESP8266**

In order to send the data over the internet we need to first connect the ESP8266 with a WI-FI network and then program it to collect values and send it to ThingSpeak API.

***The code for the same is explained as follows:***

First you need to download the libraries for ESP8266 and ThingSpeak, you can find these over here:

ThingSpeak library: [**$ https://github.com/mathworks/thingspeak-arduino $**](https://github.com/mathworks/thingspeak-arduino)

In order to download ESP8266 library follow these steps:

1. Copy this link: [**$ http://arduino.esp8266.com/stable/package\_esp8266com\_index.json $**](http://arduino.esp8266.com/stable/package_esp8266com_index.json)
2. Now open Arduino IDE and click on **File > Preferences** .
3. A window will open like this

A screenshot of a computer

Description automatically generated

window pop up

﻿

1. Paste the URL on the box and click “OK”.
2. Now go to **Tools > Board > Boards Manager.**

A screenshot of a computer

Description automatically generated with medium confidence

Board Manager

﻿

1. Type “ESP8266” on the box as shown and you will get installation option, select the **latest version** and click install.

Now, in order to use ESP8266 first configure the following circuit. Also, make sure to **remove the microcontroller from the Arduino UNO** before you proceed further,

A picture containing text, electronic engineering, circuit, circuit component

Description automatically generated

Flashing ESP setup

﻿

Now, we first need to set the ESP8266 in programming mode, we do this using the following steps:

1. Power ON the circuit.
2. Press and **hold the flash switch** .
3. While holding the flash switch, **press the reset switch** once.
4. Now, **let go of the flash switch** .

You may see a blue LED blink in the ESP8266 if you do this process right. Once, these steps are done we can move on to uploading the actual program

**Here is a brief understanding of the program:**

#include "ThingSpeak.h"

#include <ESP8266WiFi.h>

//------- WI-FI details ----------//

char ssid[] = " "; //SSID here

char pass[] = " "; // Passowrd here

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

//----------- Channel details ----------------//

unsigned long Channel\_ID = ; // Your Channel ID

const char \* myWriteAPIKey = " "; //Your write API key

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

const int Field\_Number\_1 = 1;

const int Field\_Number\_2 = 2;

const int Field\_Number\_3 = 3;

String value = "";

int value\_1 = 0, value\_2 = 0, value\_3 = 0;

int x, y, z;

WiFiClient client;

First we include the libraries and set the required variables. Under ssid and pass give your Wi-Fi credentials (name and password). Respectively, give the channel ID and write API key. If you wish to know how to setup thingspeak you can refer to this site: [**$ Thingspeak set up and basics - Robo India || Tutorials || Learn Arduino || Robotics $**](https://roboindia.com/tutorials/thingspeak-setup/). Since we are going to have three fields in our project (humidity, temperature, moisture) we use three variables for field number and three for the values.

void setup()

{

Serial.begin(115200);

WiFi.mode(WIFI\_STA);

ThingSpeak.begin(client);

internet();

}

Under setup we simply set the baud rate to 115200 and initialize the Wi-Fi and ThingSpeak client modes.

void internet()

{

if (WiFi.status() != WL\_CONNECTED)

{

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

{

WiFi.begin(ssid, pass);

delay(5000);

}

}

}

void upload()

{

ThingSpeak.writeField(Channel\_ID, Field\_Number\_1, value\_1, myWriteAPIKey);

delay(15000);

ThingSpeak.writeField(Channel\_ID, Field\_Number\_2, value\_2, myWriteAPIKey);

delay(15000);

ThingSpeak.writeField(Channel\_ID, Field\_Number\_3, value\_3, myWriteAPIKey);

delay(15000);

value = "";

}

The internet and upload functions are defined to connect the ESP8266 to Wi-Fi and uploading the data received to Thingspeak.

Now, here is the central part:

void loop()

{

internet();

if (Serial.available() > 0)

{

delay(100);

while (Serial.available() > 0)

{

value = Serial.readString();

if (value[0] == '\*')

{

if (value[7] == '#')

{

value\_1 = ((value[1] - 0x30) \* 10 + (value[2] - 0x30));

value\_2 = ((value[3] - 0x30) \* 10 + (value[4] - 0x30));

value\_3 = ((value[5] - 0x30) \* 10 + (value[6] - 0x30));

}

}

}

}

upload();

}

In order to send the data we first align it in a desired stream and then add the start and stop symbols. This can be explained with an example:

lets say at a given time the value of temperature, humidity, and moisture is **30C, 71%, and 62%** respectively. Each of these values at max can be a 2 digit integer. If all these values are written together in the same order then that would be **30-71-62** . Or in the program we would see it as a string ' **307162** '. This way we send our sensor values to the serial port which is later separated by the program in ESP8266. However, we would have to define a start and stop symbol to differentiate between two sets of values. This is done using start symbol as '\*' (asterisk) and stop symbol as '#'(hash).

So, for this we declare the empty string type variable. You will see in the Arduino Program that when we wish to write the data to this variable, we first send the start symbol to the serial port, then send the data stream, and then send the stop symbol.

In the program these values are simply separated using the indices.

value\_1 holds temperature, value\_2 holds humidity and value\_3 holds moisture value.

Note, **YOU CANNOT SEND DECIMAL VALUES** . And this is understandable based on the method we use to send our data.

So, now if you have set the ESP8266 in programming mode as described you are ready to upload the code over the ESP8266. Again, **make sure all the connections are proper and DO NOT CONNECT any pin of ESP8266 with 5V else you will BURN the IC. Further, remember to REMOVE THE MICROCONTROLLER form the Arduino Board before uploading the code. Also make sure you select the board as** ' **GENERICESP8266module'.**

Once, you have done these steps you can press the upload button.

**Arduino Programming**

Now that you are done with programming the ESP8266. We can move on to uploading the Arduino program. Make sure you remove the ESP8266 connection and then move on. Also, **change the board as 'Arduino UNO'** .

The sensors used in this circuit requires one analog pin and one digital pin. The DHT11 is read via the digital pin which is set in the program as pin 5. And the soil moisture sensor pin is set at A0 analog pin.

Now, as mentioned earlier we cannot use decimal numbers. This becomes an issue while sending temperature and humidity values. So, we use the round() function form 'math.h' library to resolve this issue. This function simply approximates a decimal number to nearest integer.

The working of programming is explained in brief as follows:

**DHT11**

In order to use DHT11 we need to install its library: [**$ https://drive.google.com/open?id=1aWuF115pMaZbySQS2CIHBxJiPSM1ezhE $**](https://drive.google.com/open?id=1aWuF115pMaZbySQS2CIHBxJiPSM1ezhE)

Now in order to get the readings we simply use the function [**$ dht.humidity $**](http://dht.humidity/)and [**$ dht.temperature $**](http://dht.temperature/)this gives us the respective value in decimal. To send use this data we round them off by simply passing the value as round(DHT.humidity) and round(DHT.temperature).

mySerial.print('\*'); // Starting char

mySerial.print(round(DHT.temperature)); //2 digit data

mySerial.print(round(DHT.humidity)); //2 digit data

You can also refer the example program for DHT11 to understand the code better

here is the connections for DHT11:

A blue circuit board with wires

Description automatically generated with low confidence

DHT11 conneciton

﻿

Note, ***in this data pin is connected to pin 2. You can connect to your desired pin by changing the pin value in example program.***

**Soil Moisture**

The soil moisture sensor value is read on the analog pin. First you would need to know what is the maximum analog output of the soil moisture when it is used in your sample of MOISTURISED soil. Simply make the connections as shown here, and upload the following code: (observe the serial plotter)

void setup() {

// put your setup code here, to run once:

Serial.begin(9600); delay(2000);

}

void loop() {

// put your main code here, to run repeatedly:

Serial.print("Start");

Serial.println(analogRead(A0));

}

A picture containing text, screenshot, electronic engineering, circuit

Description automatically generated

Soil moisture connection

﻿

Now put the sensor in the soil and see the max readings reached. This value will represent the 100 percent moisture. In my case this was at 550. Now in order to get the respective integer percentage value we use the map() function. The analog reading is taken in op\_value variable, so all we do is:

op\_value = map(op\_value, 550, 0, 0, 100);

Writing it this way means 550 is 100 percent and 0 is 0 percent.

This is how the loop statement looks like now

ack = 0;

int chk = DHT.read11(DHTxxPIN);

switch (chk)

{

case DHTLIB\_ERROR\_CONNECT:

ack = 1;

break;

}

if (ack == 0)

{

Serial.print("Temperature(\*C) = ");

Serial.println(DHT.temperature);

Serial.print("Humidity(%) = ");

Serial.println(DHT.humidity);

op\_value = analogRead(A0);

op\_value = map(op\_value,550,0,0,100);

Serial.print("Moisture(%): ");

Serial.print(op\_value);

Serial.print("%");

Serial.println("\

------------------------- \

");

//------Sending Data to ESP8266--------//

mySerial.print('\*'); // Starting char

mySerial.print(round(DHT.temperature)); //2 digit data

mySerial.print(round(DHT.humidity)); //2 digit data

mySerial.print(op\_value); //2 digit data

mySerial.println('#'); // Ending char

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

delay(2000);

}

if (ack == 1)

{

Serial.print("NO DATA");

Serial.print("\

\

");

delay(2000);

}

We use the ack variable to simply check if DHT is connected properly or not. Note, **no need to use round() function for soil moisture value as it is already a 2 digit integer.**

Configure the circuit as the same as it was shown for the individual parts. **Make sure that GPIO0 and RST of ESP8266 are not connected to ground while you upload the Arduino code.**

**Circuit**

Here is what the connections should finally look like:

**DHT11**

**Datapin-->pin5**

**Vcc------->5V**

**GND------>GND**

**SoilmoistureSensor**

**A0-------->A0**

**Vcc-------->5V**

**GND------->GND**

**ESP8266**

**3.3V------->3.3V**

**EN--------->3.3V**

**GND------>GND**

**Rx--------->pin10**

**Tx--------->pin11**

**RST------->disconnected**

**GPIO0----->disconnected**

**GPIO2------>disconnected**

Finally the circuit may look like this:

A picture containing electrical wiring, cable, electronics, electronic engineering

Description automatically generated﻿

[PreviousNext](https://projecthub.arduino.cc/Ninety99/getting-started-with-agriculture-iot-thingspeak-matlab-299aac)

**ThingSpeak**

The first set of data that you will receive is all zero. This ensures that your Wi-Fi connection and ThingSpeak connections are all properly functioning. Now, remember that we have **added a delay of 15s before sending the readings of each field.** So, it will take 45 seconds in total to send all the data to their respective fields. The reading charts may look like this:

A picture containing text, screenshot, font, line

Description automatically generated﻿

[PreviousNext](https://projecthub.arduino.cc/Ninety99/getting-started-with-agriculture-iot-thingspeak-matlab-299aac)

You can further, compare the data with one another by a simple Matlab code. While using the thingSpeakRead() function we use the parameter NumMinutes to retrieve data within a specified time duration. So, writing 10 means retrieve data from 10 minutes ago.

Also, because of the 15 second delay their will always be two NaN values before a new reading in each field. So, we use ~isnan() to remove these values from the variable.

Make sure to fill channel ID and Read API key variables:

A screenshot of a graph

Description automatically generated with medium confidence

humidity vs moisture

﻿

readID = ;

readkey = ' ';

hum = thingSpeakRead(readID, 'Field', 2, 'NumPoints', 10, 'ReadKey', readkey);

moist= thingSpeakRead(readID, 'Field', 3, 'NumPoints', 10, 'ReadKey', readkey);

hum = hum(~isnan(hum));

moist = moist(~isnan(moist));

stem(moist(end-10:end),hum(end-10:end)')

xlabel('Moisture(%)')

ylabel('Humidity(%)')

A screen shot of a graph

Description automatically generated with medium confidence

temperature vs moisture

﻿

readID = ;

readkey = ' ';

temp = thingSpeakRead(readID, 'Field', 1, 'NumMinutes', 10, 'ReadKey', readkey);

moist= thingSpeakRead(readID, 'Field', 3, 'NumMinutes', 10, 'ReadKey', readkey);

temp = temp(~isnan(temp));

moist = moist(~isnan(moist));

stem(moist(end-10:end),temp(end-10:end)')

xlabel('Moisture (%)');

ylabel('Temperature (C)');

**Errors**

Here are the possible errors you may face:

1. ESP8266 error:

If you come across this error then it means that ESP8266 was not in programming mode when you uploaded the code. So, first make sure your connections are right. Especially **check the ground pin connections to GPIO0 and RST** . Also, make sure you use the same esp and thingspeak libraries as in this article to avoid any further errors.

A picture containing text, screenshot, software, website

Description automatically generated

﻿

1. ThingSpeak errors:

The first readings sent to ThingSpeak will all be zeroes. This shows that a successful connection has been made between the ESP8266 and ThingSpeak. Now, if you are only receiving zeroes, then you may need to check if the data from the Arduino is being passed on or not. We do this as follows,

Open your serial monitor and type any random values in the format **\*XXXXXX#,** this is similar to the way we send the sensor values. If you see the result in the chart then it means values are being passed. However, the sensor values are not in proper integer format that we need. See if you can find what type of integers are being sent in serial port, make sure the values passed in myserial.print() function are rounded off with round() function or by any other method based on your program. The only requirement is the values need to be a 2 digit integer.

Finally, at Thingspeak in Matlab visualization you will see these errors:

A screenshot of a computer

Description automatically generated with medium confidence

in channel

﻿

A screenshot of a computer

Description automatically generated with medium confidence

In visualiztion

﻿

This is because the variables don't have enough values to make the plot. As mentioned in the code, it needs **10 values to make the plot** . Also, it gets these 10 values from the **past 10 minutes of operation** . So, if you haven't used the system for that much duration then you won't have enough values. You can always change the time duration for plotting graphs as per your need. But remember the first time you run the system, you will have to wait for 10-15 minutes in order to have the plots. You can change the program for a shorter duration, but remember whatever time duration you pick, it must be sufficient enough to have 10 values ready to use in the field. The shortest duration you can go for is 5 minutes.

Code:

ESP8266 Program

arduino

1#include "ThingSpeak.h"

2#include <ESP8266WiFi.h>

3

4//------- WI-FI details ----------//

5char ssid[] = " "; //SSID here

6char pass[] = " "; // Passowrd here

7//--------------------------------//

8

9//----------- Channel details ----------------//

10unsigned long Channel\_ID = ; // Your Channel ID

11const char \* myWriteAPIKey = " "; //Your write API key

12//-------------------------------------------//

13

14const int Field\_Number\_1 = 1;

15const int Field\_Number\_2 = 2;

16const int Field\_Number\_3 = 3;

17String value = "";

18int value\_1 = 0, value\_2 = 0, value\_3 = 0;

19int x, y, z;

20WiFiClient client;

21

22void setup()

23{

24 Serial.begin(115200);

25 WiFi.mode(WIFI\_STA);

26 ThingSpeak.begin(client);

27 internet();

28}

29

30void loop()

31{

32 internet();

33 if (Serial.available() > 0)

34 {

35 delay(100);

36 while (Serial.available() > 0)

37 {

38 value = Serial.readString();

39 if (value[0] == '\*')

40 {

41 if (value[7] == '#')

42 {

43 value\_1 = ((value[1] - 0x30) \* 10 + (value[2] - 0x30));

44 value\_2 = ((value[3] - 0x30) \* 10 + (value[4] - 0x30));

45 value\_3 = ((value[5] - 0x30) \* 10 + (value[6] - 0x30));

46 }

47 }

48 }

49 }

50 upload();

51}

52

53void internet()

54{

55 if (WiFi.status() != WL\_CONNECTED)

56 {

57 while (WiFi.status() != WL\_CONNECTED)

58 {

59 WiFi.begin(ssid, pass);

60 delay(5000);

61 }

62 }

63}

64

65void upload()

66{

67 ThingSpeak.writeField(Channel\_ID, Field\_Number\_1, value\_1, myWriteAPIKey);

68 delay(15000);

69 ThingSpeak.writeField(Channel\_ID, Field\_Number\_2, value\_2, myWriteAPIKey);

70 delay(15000);

71 ThingSpeak.writeField(Channel\_ID, Field\_Number\_3, value\_3, myWriteAPIKey);

72 delay(15000);

73 value = "";

74}

ThingSpeak-hum vs moist

matlab

1readID = ;

2readkey = ' ';

3hum = thingSpeakRead(readID, 'Field', 2, 'NumPoints', 60, 'ReadKey', readkey);

4moist= thingSpeakRead(readID, 'Field', 3, 'NumPoints', 60, 'ReadKey', readkey);

5hum = hum(~isnan(hum));

6moist = moist(~isnan(moist));

7stem(moist(end-10:end),hum(end-10:end)')

8xlabel('Moisture(%)')

9ylabel('Humidity(%)')

Arduino CODE

arduino

1#include <math.h>

2#include <dht.h>

3#include <SoftwareSerial.h>

4SoftwareSerial mySerial(10, 11); //(Tx,Rx)

5dht DHT;

6#define DHTxxPIN 5

7int op\_value;

8

9int ack;

10void setup()

11{

12 Serial.begin(9600);

13 mySerial.begin(115200);

14}

15void loop()

16{

17 ack = 0;

18 int chk = DHT.read11(DHTxxPIN);

19 switch (chk)

20 {

21 case DHTLIB\_ERROR\_CONNECT:

22 ack = 1;

23 break;

24 }

25 if (ack == 0)

26 {

27 Serial.print("Temperature(\*C) = ");

28 Serial.println(DHT.temperature);

29 Serial.print("Humidity(%) = ");

30 Serial.println(DHT.humidity);

31 op\_value = analogRead(A0);

32 op\_value = map(op\_value,550,0,0,100);

33 Serial.print("Moisture(%): ");

34 Serial.print(op\_value);

35 Serial.print("%");

36 Serial.println("\

37 ------------------------- \

38");

39 //------Sending Data to ESP8266--------//

40 mySerial.print('\*'); // Starting char

41 mySerial.print(round(DHT.temperature)); //2 digit data

42 mySerial.print(round(DHT.humidity)); //2 digit data

43 mySerial.print(op\_value); //2 digit data

44 mySerial.println('#'); // Ending char

45 //------------------------------------//

46 delay(2000);

47 }

48 if (ack == 1)

49 {

50 Serial.print("NO DATA");

51 Serial.print("\

52\

53");

54 delay(2000);

55 }

56}

Arduino CODE

arduino

1#include <math.h>

2#include <dht.h>

3#include <SoftwareSerial.h>

4SoftwareSerial mySerial(10, 11); //(Tx,Rx)

5dht DHT;

6#define DHTxxPIN 5

7int op\_value;

8

9int ack;

10void setup()

11{

12 Serial.begin(9600);

13 mySerial.begin(115200);

14}

15void loop()

16{

17 ack = 0;

18 int chk = DHT.read11(DHTxxPIN);

19 switch (chk)

20 {

21 case DHTLIB\_ERROR\_CONNECT:

22 ack = 1;

23 break;

24 }

25 if (ack == 0)

26 {

27 Serial.print("Temperature(\*C) = ");

28 Serial.println(DHT.temperature);

29 Serial.print("Humidity(%) = ");

30 Serial.println(DHT.humidity);

31 op\_value = analogRead(A0);

32 op\_value = map(op\_value,550,0,0,100);

33 Serial.print("Moisture(%): ");

34 Serial.print(op\_value);

35 Serial.print("%");

36 Serial.println("\

37 ------------------------- \

38");

39 //------Sending Data to ESP8266--------//

40 mySerial.print('\*'); // Starting char

41 mySerial.print(round(DHT.temperature)); //2 digit data

42 mySerial.print(round(DHT.humidity)); //2 digit data

43 mySerial.print(op\_value); //2 digit data

44 mySerial.println('#'); // Ending char

45 //------------------------------------//

46 delay(2000);

47 }

48 if (ack == 1)

49 {

50 Serial.print("NO DATA");

51 Serial.print("\

52\

53");

54 delay(2000);

55 }

56}

ESP8266 Program

Arduino

1#include "ThingSpeak.h"

2#include <ESP8266WiFi.h>

3

4//------- WI-FI details ----------//

5char ssid[] = " "; //SSID here

6char pass[] = " "; // Passowrd here

7//--------------------------------//

8

9//----------- Channel details ----------------//

10unsigned long Channel\_ID = ; // Your Channel ID

11const char \* myWriteAPIKey = " "; //Your write API key

12//-------------------------------------------//

13

14const int Field\_Number\_1 = 1;

15const int Field\_Number\_2 = 2;

16const int Field\_Number\_3 = 3;

17String value = "";

18int value\_1 = 0, value\_2 = 0, value\_3 = 0;

19int x, y, z;

20WiFiClient client;

21

22void setup()

23{

24 Serial.begin(115200);

25 WiFi.mode(WIFI\_STA);

26 ThingSpeak.begin(client);

27 internet();

28}

29

30void loop()

31{

32 internet();

33 if (Serial.available() > 0)

34 {

35 delay(100);

36 while (Serial.available() > 0)

37 {

38 value = Serial.readString();

39 if (value[0] == '\*')

40 {

41 if (value[7] == '#')

42 {

43 value\_1 = ((value[1] - 0x30) \* 10 + (value[2] - 0x30));

44 value\_2 = ((value[3] - 0x30) \* 10 + (value[4] - 0x30));

45 value\_3 = ((value[5] - 0x30) \* 10 + (value[6] - 0x30));

46 }

47 }

48 }

49 }

50 upload();

51}

52

53void internet()

54{

55 if (WiFi.status() != WL\_CONNECTED)

56 {

57 while (WiFi.status() != WL\_CONNECTED)

58 {

59 WiFi.begin(ssid, pass);

60 delay(5000);

61 }

62 }

63}

64

65void upload()

66{

67 ThingSpeak.writeField(Channel\_ID, Field\_Number\_1, value\_1, myWriteAPIKey);

68 delay(15000);

69 ThingSpeak.writeField(Channel\_ID, Field\_Number\_2, value\_2, myWriteAPIKey);

70 delay(15000);

71 ThingSpeak.writeField(Channel\_ID, Field\_Number\_3, value\_3, myWriteAPIKey);

72 delay(15000);

73 value = "";

74}

75

ThingSpeak-hum vs moist

matlab

1readID = ;

2readkey = ' ';

3hum = thingSpeakRead(readID, 'Field', 2, 'NumPoints', 60, 'ReadKey', readkey);

4moist= thingSpeakRead(readID, 'Field', 3, 'NumPoints', 60, 'ReadKey', readkey);

5hum = hum(~isnan(hum));

6moist = moist(~isnan(moist));

7stem(moist(end-10:end),hum(end-10:end)')

8xlabel('Moisture(%)')

9ylabel('Humidity(%)')

ThingSpeak-temp vs moist

matlab

1readID = ;

2readkey = ' ';

3temp = thingSpeakRead(readID, 'Field', 1, 'NumMinutes', 60, 'ReadKey', readkey);

4moist= thingSpeakRead(readID, 'Field', 3, 'NumMinutes', 60, 'ReadKey', readkey);

5temp = temp(~isnan(temp));

6moist = moist(~isnan(moist));

7stem(moist(end-10:end),temp(end-10:end)')

8xlabel('Moisture (%)');

9ylabel('Temperature (C)');

Downloadable files

project\_bb\_CkKjcAO14a.png

project\_bb\_CkKjcAO14a.png

A picture containing text, diagram, screenshot, plan

Description automatically generated

project\_bb\_CkKjcAO14a.png

project\_bb\_CkKjcAO14a.png

A picture containing text, diagram, screenshot, plan

Description automatically generated