

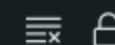


floodDetector.ino



```
1  /*
2  floodDetector.ino
3  Author - Hashir Sibtain
4  This program reads water level in a container, displays the reading on an OLED display and sends it to a ThingSpeak channel.
5  If the water level rises above a certain thershold a warning is displayed on the OLED screen in bold font and the buzzer is
6  sounded to attract attention.
7  */
8
9  #include <Arduino.h> // Arduino.h contains many commonly used definitions
10 #include "WiFi.h" // WiFi.h is included to connect to WiFi
11 #include <ThingSpeak.h> // ThingSpeak library is the official library to connect to a ThingSpeak channel
12
13 // Below libraries are required for interfacing SSD1306 OLED display with the microcontroller
14 #include <Wire.h> // The Wire library is used for I2C communication with the OLED display
15 #include <Adafruit_GFX.h> // Adafruit_GFX library is a low level library which is used as a base by the Adafruit_SSD1306 library for the OLED display
16 #include <Adafruit_SSD1306.h> //Contains all the functions and definitions to interface with the SSD1306 OLED display conveniently
17
18 // NewPing is a HC-SR04 Ultrasonic Sensor library which makes taking readings easy and convenient as it handles all the
19 // calculations and formulae.
20 #include <NewPing.h>
21
22 // We define the trigger and echo pins for the ultrasonic sensor and also define a maximum distance which it can measure
23 #define TRIG_PIN 5
24 #define ECHO_PIN 18
25 #define MAX_DIST 2000
26
27 // In the below two lines, we define the width and height of the OLED display in pixels
28 #define SCREEN_WIDTH 128
29 #define SCREEN_HEIGHT 64
30
31 // We define the WiFi network name, password and timeout in millisecond
32 #define WIFI_NETWORK "Your WiFi Network" // For privacy and security, this particular WiFi network is only a placeholder and should be replaced with the user's WiFi new
33 #define WIFI_PASSWORD "Your WiFi Password" // For privacy and security, this particular WiFi network password is only a placeholder and should be replaced with the user
34 #define WIFI_TIMEOUT 30000 // The ESP32 will try to connect to the WiFi network for 30 seconds before giving up
```

Output






floodDetector.ino



```
31 // We define the WiFi network name, password and timeout in millisecond
32 #define WIFI_NETWORK "Your WiFi Network" // For privacy and security, this particular WiFi network is only a placeholder and should be replaced with the user's WiFi network
33 #define WIFI_PASSWORD "Your WiFi Password" // For privacy and security, this particular WiFi network password is only a placeholder and should be replaced with the user's WiFi password
34 #define WIFI_TIMEOUT 20000 // The ESP 32 will try to connect to the WiFi network for 20 seconds before moving on
35
36 const char* writeAPIKey = "0123456789ABCDEF"; // The ThingSpeak Write API Key is required to write to a ThingSpeak Channel. For privacy and security, this is a dummy key
37 unsigned long channelNumber = 1234567; // The ThingSpeak Channel ID identifies a unique ThingSpeak Channel so the data reaches the correct channel. For privacy and security, this is a dummy ID
38
39 unsigned long lastTime = 0; // long variable to keep time
40 int upDelay = 5000; // Every 5 seconds, the ESP32 will try uploading to the ThingSpeak channel
41 NewPing levelSensor(TRIG_PIN, ECHO_PIN, MAX_DIST); // We create a NewPing variable which represents the HC-SR04 Ultrasonic Sensor
42
43 #define OLED_RESET -1 // Reset pin related to OLED Display
44 #define SCREEN_ADDRESS 0x3C // OLED Display's I2C Slave address
45 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); // We create a new display variable which represents the SSD1306 OLED Display
46
47 const int buzzer = 4; // This constant integer denotes the pin the buzzer is connected to
48
49 WiFiClient client; // We create a WiFi Client for use with the ThingSpeak library
50
51 void connectToWiFi()
52 {
53     /* This is a user defined function which connects the ESP32 to WiFi and initializes the ThingSpeak library.
54     | It returns nothing and takes no arguments
55     */
56     Serial.print("Connecting to WiFi"); // Print a line denoting the start of an attempt to connect to WiFi
57     WiFi.mode(WIFI_STA); // Set ESP32 WiFi mode to station mode to connect to a WiFi hotspot
58     ThingSpeak.begin(client); // Initialize the ThingSpeak library and network setting with the WiFi client
59     WiFi.begin(WIFI_NETWORK, WIFI_PASSWORD); // Connect the ESP32 to a WiFi network using the provided credentials
60
61     unsigned long startAttemptTime = millis(); // This variable keeps track of time while the ESP attempts connecting to the WiFi network
62     while(WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < WIFI_TIMEOUT) // This while loop prints a '.' while the ESP is attempting to connect to WiFi.
63     { // It ends when the connection is successful or when WIFI_TIMEOUT milliseconds (in milliseconds) have passed
```

Output

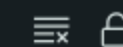


floodDetector.ino



```
61 unsigned long startAttemptTime = millis(); // This variable keeps track of time while the ESP attempts connecting to the WiFi network
62 while(WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < WIFI_TIMEOUT) // This while loop prints a '.' while the ESP is attempting to connect to WiFi.
63 { // It ends when the connection is successful or when WIFI_TIMEOUT milliseconds (in
64     Serial.print("."); // prints the '.';
65     delay(100); // delay for 100 ms
66 }
67
68 if(WiFi.status() != WL_CONNECTED) // Print 'Failed if WiFi not connected
69 {
70     Serial.println("Failed.");
71 }
72 else // else print connected successfully and print local IP
73 {
74     Serial.println("Connected successfully");
75     Serial.println(WiFi.localIP());
76 }
77 }
78
79 void warn(int waterLevel)
80 {
81     /* This function is user-defined and deals with the display of the high water-level warning on the
82     SSD1306 OLED display. It does various things like set cursor to a correct position, set an appropriate
83     font size and an appropriate colour and finally it displays the warning which consists of the word 'WARNING'
84     written in large font with white background and black text on it and the water level on the next line
85     in bold font */
86     display.setCursor(11, 16); // Set the cursor position to 11 pixels from the right (x co-ordinate) and 16 pixels from the top (y co-ordinate)
87     display.clearDisplay(); // Clear the display to get a fresh display and space to write on
88     display.setTextSize(2); // Set text size to 2
89     display.setTextColor(BLACK, WHITE); // Set text colour to black and background colour to white where 'BLACK' and 'WHITE' are macros defined in Adafruit_SSD1306.h file
90     display.println(" WARNING "); // Display the text 'WARNING' as specified above
91     display.setTextColor(WHITE); // Set text colour to white and display colour to black
92     display.setCursor(50, 40); // Set the cursor position to 50 pixels from the right (x co-ordinate) and 40 pixels from the top (y co-ordinate)
93     display.print(waterLevel); // Display the water level as specified by the above two lines
```

Output



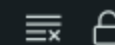


floodDetector.ino



```
81  // ... ..
82  SSD1306 OLED display. It does various things like set cursor to a correct position, set an appropriate
83  font size and an appropriate colour and finally it displays the warning which consists of the word 'WARNING'
84  written in large font with white background and black text on it and the water level on the next line
85  in bold font */
86  display.setCursor(11, 16); // Set the cursor position to 11 pixels from the right (x co-ordinate) and 16 pixels from the top (y co-ordinate)
87  display.clearDisplay(); // Clear the display to get a fresh display and space to write on
88  display.setTextSize(2); // Set text size to 2
89  display.setTextColor(BLACK, WHITE); // Set text colour to black and background colour to white where 'BLACK' and 'WHITE' are macros defined in Adafruit_SSD1306.h file
90  display.println(" WARNING "); // Display the text 'WARNING' as specified above
91  display.setTextColor(WHITE); // Set text colour to white and display colour to black
92  display.setCursor(50, 40); // Set the cursor position to 50 pixels from the right (x co-ordinate) and 40 pixels from the top (y co-ordinate)
93  display.print(waterLevel); // Display the water level as specified by the above two lines
94  display.print("cm"); // Display the unit 'cm' with the water level
95  display.display(); // Actually draw on the OLED display
96
97  }
98
99  void buzz()
100  {
101    /* This function is concerned with sounding the buzzer and returns nothing. It sounds the alarm
102    according to a particular pattern */
103    tone(buzzer, 1000); // the tone() function plays a particular takes two parameters, the buzzer pin and the frequency. It plays a tone of that frequency on the buzzer.
104    delay(500); // The program waits for 0.5 seconds. So, for these 0.5 seconds the buzzer keeps playing the 1 KHz tone
105    noTone(buzzer); // the noTone() function takes only one argument, the buzzer pin and stops any tone that is playing on that buzzer. So, the 1KHz tone stops
106  }
107  void setup() {
108    // In this section we setup Serial communication, WiFi, and OLED display
109    Serial.begin(115200); // we begin serial communication at 115200 bps
110    connectToWiFi(); // we call the connectToWiFi() function
111
112    if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) // we try initializing the SSD1306 OLED display module. if an error occurs we print 'SSD1306 allocation failed'
113    {
114      Serial.println(F("SSD1306 allocation failed"));
```

Output



floodDetector.ino

```
111
112 if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) // we try initializing the SSD1306 OLED display module. if an error occurs we print 'SSD1306 allocation failed'
113 {
114     Serial.println(F("SSD1306 allocation failed"));
115     while(true);
116 }
117 // The following four lines are related to OLED display module
118 display.clearDisplay(); // clear the display
119 display.setTextSize(1); // set font size to 1
120 display.setTextColor(WHITE); // set font colour to White
121 display.setCursor(0,28); // set cursor position to co ordinate (0, 28)
122
123 }
124
125 void loop() {
126     // This is the main loop of the program where the code repeatedly runs. It takes readings, sends them to ThingSpeak, display them on OLED screen and sound the buzzer
127     delay(50); // small delay for calibration
128
129     int lev = 22 - levelSensor.ping_cm(); // lev keeps track of the water level. The levelSensor.ping_cm() function is from the NewPing library and returns reading
130     // from the HC-SR04 sensor in cm. Since, the initial distance of the sensor from the bottom of the container is 22cm, we
131     // subtract the reading from 22 to get the height of the water.
132
133     if((millis() - lastTime) > upDelay) // If, upDelay amount of time or more has passed, then send the data to the ThingSpeak channel.
134     {
135         lastTime = millis(); // update the time in tracking variable
136         Serial.println("Uploading to ThingSpeak"); // print "Uploading to ThingSpeak" on serial monitor
137         int x = ThingSpeak.writeField(channelNumber, 1, lev, writeAPIKey); // Actually attempt uploading the data to the ThingSpeak channel using given credentials. It returns
138         if (x == 200) // If x is 200, upload was successful so print a success message on serial monitor
139         |     Serial.println("Successfully uploaded to ThingSpeak!");
140         |     else Serial.println(x); // else print out the error code
141     }
142     if(lev > 10) // If lev is greater than 10 cm then display the warning on OLED display by calling the warn() function and sound the buzzer by calling the buzz() function
143     {
```

Output



floodDetector.ino

```
130 // From the HC-SR04 sensor in cm. Since, the initial distance of the sensor from the bottom of the container is 22cm, we
131 // subtract the reading from 22 to get the height of the water.
132
133 if((millis() - lastTime) > upDelay) // If, upDelay amount of time or more has passed, then send the data to the ThingSpeak channel.
134 {
135     lastTime = millis(); // update the time in tracking variable
136     Serial.println("Uploading to ThingSpeak"); // print "Uploading to ThingSpeak" on serial monitor
137     int x = ThingSpeak.writeField(channelNumber, 1, lev, writeAPIKey); // Actually attempt uploading the data to the ThingSpeak channel using given credentials. It returns
138     if (x == 200) // If x is 200, upload was successful so print a success message on serial monitor
139     |     Serial.println("Successfully uploaded to ThingSpeak!");
140     else Serial.println(x); // else print out the error code
141 }
142 if(lev > 10) // If lev is greater than 10 cm then display the warning on OLED display by calling the warn() function and sound the buzzer by calling the buzz() function
143 {
144     warn(lev);
145     buzz();
146 }
147 else // Else, print the water level normally on OLED display
148 {
149     display.clearDisplay(); // clear display
150     display.setTextSize(1); // set font size to 1
151     display.setTextColor(WHITE); // set font colour to White
152     display.setCursor(5,28); // set cursor position to co-ordinate (5,28)
153     display.clearDisplay(); // clear display
154     display.print("Water Level: "); // print "Water Level: " at the configured co-ordinate and font size and style on the OLED display
155     display.print(lev); // print the actual water level on the OLED display
156     display.print("cm"); // print "cm" on the OLED display
157     display.display(); // finally, display the text on the OLED display so that everything shows up
158 }
159
160
161 }
162
```

Output

