```
// Using an ESP8266 Nodemcu E12, and JSN sensor with WLAN connection and Over the Air (OTA)

// Please read the explanation below and delete it when using this example to copy in Arduino
IDE for your own project.

/* Note: this program is a very extended version with many routines, lable jumps etc., solved
encountered issues, while observing the results over a longer period and is used in order to
stabilize the data reads of the unit. If not and a simple version is used, any ultrawave
disturbances / echos / water level irregularities etc. and resulting in measurement deviations
will lead to peak jumps in the data line of the widget. Sometimes with dramatic peaks. Even if
the water level does not change. During filling and empty situation this version has included
a check for the distance and volume change with a signal "sending allowed or senden erlaubt.
The measurement cycle is executed in multiple cycles and an average value then is calculated.
Please note as well, that the cistern used in reality is a cylinder. Volume calculation is
therefore easy and the height changes proportional to volume changes. Other cistern may need
another algo rhythm or equation in determining the volume. As well the OT communication is
included as stated above, which needs extra commands and libraries */

// you may need to include / connect the libraries named for your PC (see that menu point in
the IDE software)

#include <ESP8266WiFi.h>
#include <ArduinoOTA.h> // needed for OTA
#include <ESP8266mDNS.h> // needed for OTA
#include <WiFiUdp.h> // needed OTA

const char* ssid = "add here your own SSID";
const char* password = "add here your own password for WLAN access";
const char* server = "api.thingspeak.com";
String apiKey = "your API"; // write API to Thingspeak
//String apiKey = "XXXXXXXX";

// NOTE: in case for tests without sending data to Thingspeak or without having yet a
Thingspeak account, remove the two //  at beginning of the line with the XXX and insert them
in front of the line with your API


//WiFiServer server(80);
WiFiClient client;

int senden_erlaubt = 0;   // variable to allow sending
int trigger = D7; // pin with trigger signal
int echo = D6; //  pin for echo signal
float dauer = 0.0; // dauer = duration
float abstand = 0.0; // abstand = distance

// Variablen für Umrechnung Abstand in Volumen und Dreiertausch
// Varibales for calculating the distance and volume

float Volumen = 0.0; // volume initial value
float Vol_m3;

float Vol_sum = 0;      // needed for averaging /  Mittelwertberechnung
float Vol_temp = 0;     // needed for averaging /  Mittelwertberechnung
float fuellhoehe_m3;
float fuellhoehe_sum = 0;      // needed for averaging /  Mittelwertberechnung
float fuellhoehe_temp = 0;     // needed for averaging /  Mittelwertberechnung
float fuellhoehe = 0.0; // fuellhoehe = filling height

// NEW OTA millis
long myTimer = 0;
long myTimeout = 120000;  // used 16000 at Testing // normal cycle = 120000 = 2min
// ENDE OTA

int zaehler = myTimeout;
int Counter = 0; // für 5 Durchgänge
int Count_val = 4; // hier die Anzahl der Schleifendurchgänge eintragen 5
int Count_Exit = 0; // Count_Exit = plus 1
int  Schleife_outof_range = 0;
int Sperre_Thingspeak = 0;

// Measuring showed we need to exlude wrong peak measures
// *** Variablen für Peakelminierung / Peakelimination
float Vol_neu = 0;
float Vol_delta = 0;
float Vol_alt = 0;
float Vol_buffer = 0;
int Peakzaehler = 0;
```

```
// SETUP ROUTINE
void setup() {
  Serial.begin(115200);
  delay(500);
  Serial.println("Zisternenmessung wird gestartet "); // indication text that programm is
starting

  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  digitalWrite(trigger,LOW);
  Serial.println("Messung des Wasserstandes und Umrechung in Zisternenvolumen:");   //
indication of execution steps only

  // Connect to WiFi network
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println(WiFi.localIP());
  Serial.print("Bitte warten Sie für den ersten Zyklus: "); // please wait for first cycle

        long myTimeout_S = (myTimeout / 1000);
        Serial.print(myTimeout_S);
        Serial.println(" s");


 // make OTA communication available

  // Hostname defaults to esp8266-[ChipID]
  ArduinoOTA.setHostname("my water cistern");
  ArduinoOTA.onStart([]() {
  Serial.println("Start");
  });
  ArduinoOTA.onEnd([]() {
  Serial.println("\nEnd");
  });
  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
  Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  });
  ArduinoOTA.begin();
  //Serial.println("Ready");
  //Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
// ende

} // Ende Setup - End of setup


//***** Starting the LOOP

void loop()
{
      ArduinoOTA.handle(); // enables access via OTA
    // Non blocking: https://www.norwegiancreations.com/2017/09/arduino-tutorial-using-millis-
instead-of-delay/

        if (millis() > myTimeout + myTimer )
        {

        myTimer = millis(); //  weiter in der Schleife oben

        Serial.println(F("  "));
        Serial.print("Hallo - ich berechne nun alle: "); // Calculating now every seconds
        //Serial.print(period);
        long myTimeout_S = (myTimeout / 1000);
        Serial.print(myTimeout_S);
```

```
        Serial.println(" s");

        // hier der code, der mit "delay" laufen soll
        // put here the code that shall run with the new delay by timer, millis

        // Ende OTA


 // if measurements are not usable, then a jump to this section is done by the label:
nochmal_ohne_timer
// cycle will be executed without timer delay and calling again the subroutine ultrasonic!

// Lable
nochmal_ohne_timer: // Label Schnellschleife bei Peakwerten
   ultrasonic();                // Aufruf UP ultrasonic
   Counter = 0; // Rücksetzen Mittelwertschleifenzähler - reset avarage loop counter
   Serial.print("Rücksetzen Schleifen-Counter auf 0: ");Serial.println(Counter);
   Serial.println("Rücksetzen Vol_temp auf 0 ");
   Vol_temp = 0;               // Neu für Mittelwertberechnung - Rücksetzen, reset temporary
value for volume
   Serial.println("Rücksetzen fuellhoehe_temp auf 0 ");
   fuellhoehe_temp = 0;        // Neu für Mittelwertberechnung - Rücksetzen, reset filling
level



//   *** Neu Eliminierung der Wertesprünge in einem einem Messzyklus 7.0 -> 7.1 -> 7.0
//   *** New eliminating of jumping values in a measurement cycle, e.g., 7.0 -> 7.1 -> 7.0


   Vol_neu = Vol_m3;
   Vol_delta = (Vol_neu - Vol_alt);
   Vol_buffer = Vol_neu;
   Vol_alt = Vol_buffer;

   Serial.println(" ");
   Serial.print("Vol_delta: ");Serial.println(Vol_delta);
   Serial.println(" ");

   if (fabs(Vol_delta) > 0.04 && Peakzaehler <= 1){ // auch negative Sprünge
   Peakzaehler++;
   Serial.println(" ");
   Serial.print("Peakzähler im Eliminiermodus: ");Serial.println(Peakzaehler);
   Serial.print("direkt nochmal Werte ermitteln");
   Serial.println(" ");
   senden_erlaubt = 0;
   sendtoThingSpeak_F();
 goto nochmal_ohne_timer; // jump to label above, if sending is not allowed
    }


if (fabs(Vol_delta) > 0.18) {

// dann ist das ein zu hoher Sprung, der eigentlich bei der Befüllung nicht vorkommen kann
// fabs = absolute value is checked
// > 0.18 is not a realistic jump in volume, depending on your inlet pipe!!

      senden_erlaubt = 2; // 2 is only for checking if this condition is executed
      sendtoThingSpeak_F();
      Peakzaehler = 0;
      goto nochmal_ohne_timer;   // again restart subroutine by "goto" the label: → first new
real different value will be ignored
      }

 // *** Ende  Eliminierung der Wertesprünge in einem einem Messzyklus 7.0 -> 7.1 -> 7.0
// *** End of peak elimination in a measurement cycle e.g., 7.0 -> 7.1 -> 7.0



   if (senden_erlaubt == 1 && Sperre_Thingspeak == 0 && fabs(Vol_delta) < 0.3){
// Neu wegen Peaks - only if all three conditions are given, then the new value is send to
Thingspeak
// sende_erlaubt set to one in the ultrasonic sub-routine

   sendtoThingSpeak();          // sendtoThingSpeak
   Sperre_Thingspeak = 0;
   Peakzaehler = 0; // rücksetzen nach erfolgtem senden
   Serial.println(" ");
```

```
   Serial.print("Peakzähler nach Sendemodus: ");Serial.println(Peakzaehler);
   Serial.println(" ");
   }
}
}  // *** Ende Loop



// ************************************
// ab hier die Unterprogramme / sub-routines
// ************************************


//**** Unterprogramm Sensormesswert und Umrechung in Volumen
void ultrasonic()
{

Schleife: // lable
// Variante JSN
  digitalWrite(trigger, HIGH);
  delayMicroseconds(20);    // 20 Bei diesem Sensortyp 20 mü-sek warten nicht 10 wie HC-SR04
  digitalWrite(trigger, LOW);

  /* Variante HC-SR04
  digitalWrite(trigger, LOW);   // ausschalten Triggersignal - stop trigger signal
  delay(5);
  digitalWrite(trigger, HIGH); // einschalten Triggersignal - start trigger signal
  delay(10);
  digitalWrite(trigger, LOW);     // stop triffer signal
*/
  float dauer = 0.0;
  dauer = pulseIn(echo, HIGH); // Ermitteln der Gesamtlaufzeit und Zuweisung dauer

  abstand = (dauer/2) * 0.03432;

// calculation of distance by half duration and multiplication with speed of sound. See link
for values related to tempertures!

// Berechnung Entfernung in cm. Man teilt zunächst die Zeit durch zwei, damit nur einfache
Strecke berechnet wird.
//Den Wert multipliziert man mit der Schallgeschwindigkeit in der Einheit
Zentimeter/Mikrosekunde und erhält dann den Wert in Zentimetern.
 // http://www.sengpielaudio.com/TemperaturSchall.htm Bei 15 Grad Celsius ist der Wert eher
auf 340,51 m/s zu wählen. Bei 20 Celsius = 343,42 m/s

  abstand = abstand + 0; // alt: im Trockenlauf ergab sich Messungenauikgeit von 3 cm
  abstand = round(abstand);  // Wert glätten weil JSN nicht immer gleiche Werte zurückgibt

// **** Umrechnung Abstandsmessung Wasseroberfläche zu Volumen ****
// **** Enmalig vor Ort ermitteln (manuell messen!): Abstand von Boden zu Sensorunterkante =
277 (aktuell) Fixer Wert in cm ****

fuellhoehe = 278 - abstand;    // 278 = distance of sensor to bottom of cistern (in this case
a cylindric one!)
// 277 Differenz aus lichte Weite Boden zu Sensorunterkante minus des ermittelten Abstandes
Wasseroberfläche
//Volumen =  125 * 125 * 3.14159 * fuellhoehe; // Zylindervolumen mit Durchmesser Mall 250 cm
(r = 125 cm)


// averaging the Volume by adding up multiple cycles and dividing by the number of cycles
// Dreiertausch um Werte aufzuaddieren und nach der Schleife vor dem Senden zu "mitteln" - Neu
EXTRA
// Vol_m3 = Volumen / 1000000; // Umrechung cm3 auf m3

Serial.print(F("Errechneter Abstand in der Schleife, Teilergebnis Fuellhoehe:   "));
Serial.print(fuellhoehe);
Serial.println("  cm");

//für Mittelwertberechnung
Serial.print(F("fuellhoehe_Temp vor Tausch: "));
Serial.print(fuellhoehe_temp);
fuellhoehe_sum = fuellhoehe_temp + fuellhoehe;
fuellhoehe_temp = fuellhoehe_sum;

Serial.print("   Fuellhoehe nach Tausch: ");
Serial.print(fuellhoehe_temp);
Serial.print("   Fuellhöhe_sum aufaddiert nach Tausch: ");
```

```
Serial.println(fuellhoehe_sum);
delay(2000); // Verzögerung 2 sec
Counter++;
//}


// prinicple check if the values are within range of the given cistern to exclude negative
values and extreme distance values
// **** Prüfung ob Abstandswert innerhalb der Grenzwerte von 0=Boden oder der maximalen
Zisternenhöhe Oberkante Deckel laut Mall ist ****+

if (abstand >= 300 || abstand <= 0)  // 3 Meter gemessen von Boden zu Unterkante Verengung
Auslass Schachtdeckel
  {
    fuellhoehe_temp = 0; // in dieser Bedingung den Wert aus der Schleife hart auf "0" setzen
    Serial.print("Messung außerhalb Zisternengrenzen, daher: Kein Messwert ");
    Serial.println(fuellhoehe_temp);
    Serial.print("Fehler in Volumenberechnung: ");Serial.println(Vol_m3);
    Serial.println(" ");
    senden_erlaubt = 0;
    Serial.print("senden erlaubt = ");Serial.println(senden_erlaubt);

    Schleife_outof_range ++;
    Serial.print("out of range Schleife ");Serial.println(Schleife_outof_range);
    Sperre_Thingspeak = 1;
    Serial.print("Sperre Thingspeak normaler Modus aktiv (1):
");Serial.println(Sperre_Thingspeak);

    if ( Schleife_outof_range == 10) { // 10 Versuche danach reset der Berechnung
    //Counter = 0; // Rücksetzen Mittelwertschleifenzähler
    //Serial.print("rücksetzen Counter auf 0 ");Serial.println(Counter);
    Vol_temp = 0;               // Neu für Grenzwerte - Rücksetzen
    Vol_m3 = 0;               // Neu für Grenzwerte - Rücksetzen
    fuellhoehe = 0;           // Neu für Grenzwerte - Rücksetzen
    fuellhoehe_temp = 0;        // Neu für Grenzwerte - Rücksetzen
    fuellhoehe_sum = 0;        // Neu für Grenzwerte - Rücksetzen
    Schleife_outof_range = 0;   // Rücksetzen out of range Schleife Messbereich
    sendtoThingSpeak_F(); // Aufruf UP Fehlerstatus senden_erlaubt = 0
    }

    goto Schleife;
  }
  else
  {
    senden_erlaubt = 1;
    Serial.println("  ");
    Serial.print(F("Counter = ")); // Prüfung of Szahler = gesetzt und rückgesetzt wird für
Mittelwertberechnung
    Serial.println(Counter);
    Serial.print("Abstand zur Wasseroberfläche: ");
    Serial.println(abstand);
    fuellhoehe_m3 = fuellhoehe_temp;              // damit letzter Vol_temp Wert genutzt wird
für Mittelwertberechnung
    fuellhoehe_m3 = fuellhoehe_m3 / Counter;     //für Mittelwertberechnung
    fuellhoehe_m3 = round(fuellhoehe_m3*100)/100.0; // Klammer*10 / 10 = eine Dezi
Klammer*100/100 = 2 Dezi
    Serial.print("gemittelter Wert = ");
    Serial.println(fuellhoehe_m3);
    Volumen =  125 * 125 * 3.14159 * fuellhoehe_m3;
    Vol_m3 = Volumen / 1000000; // Umrechung cm3 auf m3

    // *** Runden des Wertes, von float NKStellen auf "eine" --> Die Kurven im Thingspeak
werden glatter dargestellt
    Vol_m3 = round(Vol_m3*10)/10.0; // runden auf eine Dezimalstelle. Bei zweien dann 100
nutzen / oder round einfach weglassen ACHTUNG: nimmt aus der Zahl die ersten beiden NKStellen
und rundet diese. ROUND OF VALUE ONE DECIMAL
    //Float ist zwei NK-stellig, weil es nur 4 Byte. --> xx.yy


    // **** Nur für Arduino-SW: Ausgabe Monitor oder Plotter zu Prüfzwecken ****
// this section is only to print the results in the monitor after the calculation

    Serial.print(fuellhoehe_m3); // ohne Zeilenvorschub mit print
    Serial.print(" cm Füllhöhe ergibt:     "); // Hinter dem Wert der Entfernung soll auch am
Serial Monitor die Einheit "cm" angegeben werden.
    Serial.print(Vol_m3);
    Serial.println(" m3 Wasservolumen"); // Hinter dem Wert der Entfernung soll auch am Serial
Monitor die Einheit "m3" angegeben werden.
```

```
  }


if (Counter <= Count_val)
{
 Sperre_Thingspeak = 0; // Rücksetzen der Sperre
 goto Schleife;
}

Count_Exit = (Count_val +1); // eine Schleife mehr wegen 0
if (Counter == (Count_Exit)) {
 }
} // Ende UP Ultrasonic


//**** Unterprogramm Daten an die Cloud senden
// subroutine sending to cloud

void sendtoThingSpeak()

{
        Serial.println("");
        Serial.print("Sende an Thingspeak gemittelten Wert Füllstand = ");
        fuellhoehe_m3 = round(fuellhoehe_m3*10)/10.0; // Klammer*10 / 10 = eine Dezi
Klammer*100/100 = 2 Dezi
        Serial.println(fuellhoehe_m3);
        Serial.print("Zisternenvolumen = ");Serial.println(Vol_m3);
        Serial.print("Meldung an Thingspeak - senden erlaubt = ");
Serial.println(senden_erlaubt);

    if (client.connect(server,80))  // "184.106.153.149" or api.thingspeak.com

    {
        String postStr = apiKey;
        postStr += "&field1=";
        postStr += String(Vol_m3); // the calculated Volume

        postStr += "&field2=";
        postStr += String(senden_erlaubt); // the status of sending to cloud

        postStr += "&field3=";
        postStr += String(Vol_delta); // the value of volume delta

        client.print("POST /update HTTP/1.1\n");
        client.print("Host: api.thingspeak.com\n");
        client.print("Connection: close\n");
        client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
        client.print("Content-Type: application/x-www-form-urlencoded\n");
        client.print("Content-Length: ");
        client.print(String(postStr.length()));
        client.print("\n\n");
        client.println(postStr);
        delay(1000);

        client.flush(); // Wartet bis alle Daten gesendet wurden
        client.stop();

    }


       // tu nix - do nothing
       // Serial.println("mache nix im loop"); // Not used, otherwise it will print always
text like do nothing in the loop
       delay(100); // hiermit Watchdog aktivieren sonst Überlauf!
https://forum.arduino.cc/index.php?topic=466092.0

} // *** Ende UP Thingspeak


// *** UP nur wenn außerhalb Grenzwerte der Messung
// only if measurements are outside of range F = failure

void sendtoThingSpeak_F()

{
        Serial.println("");
        Serial.print("Fehlermeldung an Thingspeak - senden erlaubt (0 - nein, 1 - ja) = ");
Serial.println(senden_erlaubt);
```

```
    if (client.connect(server,80))

    {
        String postStr = apiKey;
        postStr += "&field2=";
        postStr += String(senden_erlaubt);

        postStr += "&field3=";
        postStr += String(Vol_delta);

        client.print("POST /update HTTP/1.1\n");
        client.print("Host: api.thingspeak.com\n");
        client.print("Connection: close\n");
        client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
        client.print("Content-Type: application/x-www-form-urlencoded\n");
        client.print("Content-Length: ");
        client.print(String(postStr.length()));
        client.print("\n\n");
        client.println(postStr);
        delay(1000);

        client.flush(); // Wartet bis alle Daten gesendet wurden
        client.stop();

    }

// *** Ende UP Fehlermeldung



       // tu nix
       // Serial.println("mache nix im loop");
       delay(100);
// delay needed as watchdog - hiermit Watchdog aktivieren sonst Überlauf!
https://forum.arduino.cc/index.php?topic=466092.0

} // *** Ende UP Thingspeak_F
```