

IoT Based river water quality monitoring system

Using IBM Watson



May 24, 2019

Team microes

Bhasskar & Roshan

9502867595

Abstract

Degradation of water resources has become a common problem. The conventional methods of water quality monitoring involve the manual collection of water sample from different locations. These water samples were tested in the laboratory using rigorous skills. Such approaches are time consuming and no longer considered to be efficient. The older method of water quality detection was time consuming, low precision and costly. By focusing on the above issues, a low-cost water quality monitoring system is developed and designed that can monitor water quality in real time using IOT. In the proposed system water quality parameters are measured by different sensors communicating data onto a platform via microcontroller system. So, in order to meet all these requirements, other technologies can be used such as MQTT (Message Queuing Telemetry Transport) which allows publishing and subscribing of data between the sensor and end device. And with the help of MQTT algorithm there will be simultaneous flow of data between the sensors and the servers

Introduction

River water which is used as drinking water is a very precious commodity for all human beings, as drinking water utilities face a lot of new challenges in real-time operation. In this project we intend to present the design and development of a low-cost system for real monitoring of water quality in an IoT environment. The system consists of several sensors which are used for measuring physical and chemical parameters of water.

Using this system, a person can detect pollutants from a water body from anywhere in the world. We can monitor the water quality parameters by using some sensors and these sensor parameters are sent to the IBM Watson cloud and stored in the data base. These parameters are visualized in the User Interface which is created using Node Red of IBM Watson platform. Whenever the water quality is not good, we can send notifications to authorities through the third-party services.

Problem Statement

# In this project we intend to present the design and development of a low cost system for real monitoring of water quality in an IoT environment. The system consists of several sensors which are used for measuring physical and chemical parameters of water.

# Highlights

* Working with IBM Watson cloud services
* Accessing Sensor data from anywhere in the world.
* Notifications and SMS are triggered based on sensor values automatically.
* Reduces Manual effort involved in monitoring river water quality.
* Constant Visualization of sensor data by the cloud.
* It can be used even in remote areas where manual presence is impossible.

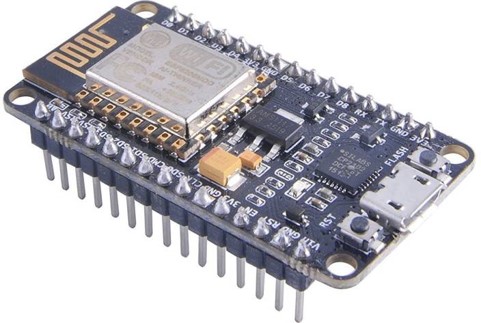
Requirements

1. Hardware
   1. NodeMCU
   2. Turbidity Sensor
   3. Temperature Sensor
   4. GSM Module
2. Software
   1. Arduino IDE

Component description

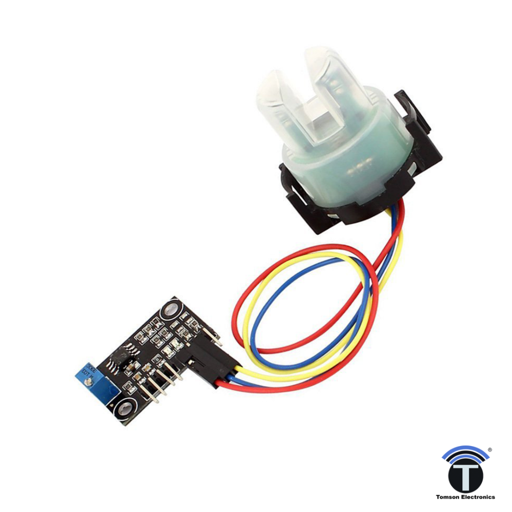
1. NodeMCU

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS.



1. Turbidity Sensor

When viewed from front, the turbidity sensor appears like an Android bot. Two horn like structure, a top to bottom mono material body. A black colored cap is placed at the bottom of the sensor. Thick alloyed contact legs provide means for various connectors to hold to the sensor. A white plastic slab protects the legs from damage and act as a fixture for good clamping of the sensor. The plastic used to make outer structure can survive high temperature variations as well as mechanical abrasions.



1. Temperature Sensor

A temperature sensor is exactly what it sounds like – a sensor used to measure ambient temperature. It is a DHT sensor which is integrated with NodeMCU or Arduino Uno R3, and use the Arduino IDE's serial monitor to display the temperature.



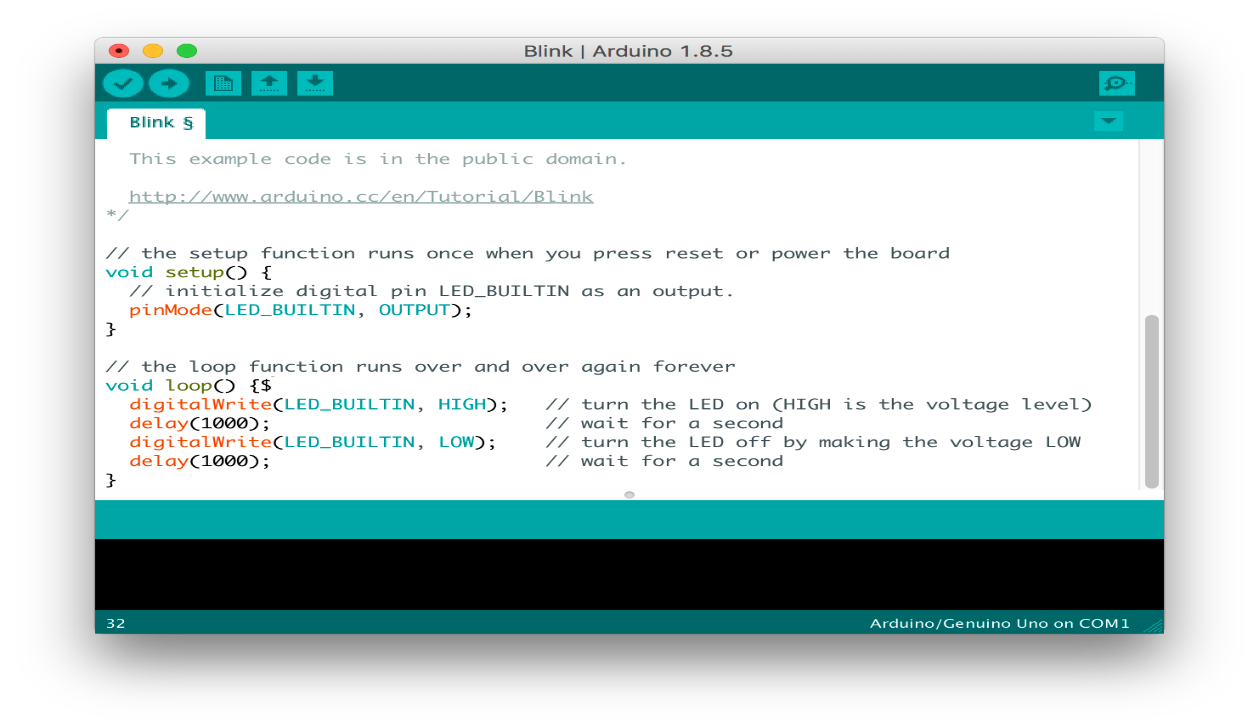
1. GSM module

A GSM module is a chip or circuit that will be used to establish communication between a mobile device or a computing machine and a GSM or GPRS system. The modem (modulator-demodulator) is a critical part here. These modules consist of a GSM module or GPRS modem powered by a power supply circuit and communication interfaces (like RS-232, USB 2.0, and others) for computer. A GSM modem can be a dedicated modem device with a serial, USB or Bluetooth connection, or it can be a mobile phone that provides GSM modem capabilities.



1. Arduino IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.



Connections

The connections are made accordingly from NodeMCU to temperature sensor, turbidity sensor, GSM module, basic board.

* + NodeMCU to basic shield – Connect VCC to 3.3V, ground to ground and DHT 11 pin to D2
  + NodeMCU to turbidity board – Connect VCC to V(in), ground-ground, A0-A0
  + GSM module to NodeMCU - Connect ground-ground, Tx- D7, Rx-D8.
  + Turbidity sensor to its board- The three pins are arranged in a way as of the thermistor output is connected to ground, the photodiode output is connected to TIN and the last of all the phototransistor input is connected to VCC.

Code:

--------------------------------------------------------------------------------

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include<SoftwareSerial.h>

SoftwareSerial mySerial(D7, D8);

//-------- Customise these values -----------

const char\* ssid = "thelonepair";

const char\* password = "159374rs";

#include "DHT.h"

#define DHTPIN D2    // what pin we're connected to

#define DHTTYPE DHT11   // define type of sensor DHT 11

DHT dht (DHTPIN, DHTTYPE);

#define ORG "fbf9fj"

#define DEVICE\_TYPE "BB-08"

#define DEVICE\_ID "11222808"

#define TOKEN "08223000"

//-------- Customise the above values --------

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char topic[] = "iot-2/evt/Data/fmt/json";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

WiFiClient wifiClient;

PubSubClient client(server, 1883,wifiClient);

void setup() {

  mySerial.begin(9600);

Serial.begin(9600);

delay(100);

Serial.println();

dht.begin();

Serial.print("Connecting to ");

Serial.print(ssid);

WiFi.begin(ssid, password);

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

delay(500);

Serial.print(".");

}

 Serial.println("");

 Serial.print("WiFi connected, IP address: ");

Serial.println(WiFi.localIP());

}

void loop() {

float t = dht.readTemperature();

int sensorValue = analogRead(A0);// read the input on analog pin 0:

  float voltage = sensorValue \* (5.0 / 1024.0); // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):

  //Serial.println(sensorValue);// print out the value you read:

  delay(500);

  int turb=sensorValue;

if (isnan(t))

{

Serial.println("Failed to read from DHT sensor!");

delay(1000);

return;

}

PublishData(t,turb);

delay(1000);

if(turb<600)

{

  SendMessage(turb);

}

}

void PublishData(float temp, int turb){

if (!!!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(500);

}

Serial.println();

}

  String payload = "{\"d\":{\"temperature\":";

  payload += temp;

  payload+="," "\"Turbidity\":";

  payload += turb;

  payload += "}}";

Serial.print("Sending payload: ");

Serial.println(payload);

 if (client.publish(topic, (char\*) payload.c\_str())) {

Serial.println("Publish ok");

} else {

Serial.println("Publish failed");

}

}

void SendMessage(int turb)

{

  Serial.println(".............................");

  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode

  delay(1000);  // Delay of 1000 milli seconds or 1 second

  mySerial.println("AT+CMGS=\"+919502867595\"\r"); // Replace x with mobile number

  delay(1000);

  mySerial.println("Water at unsafe levels : ");// The SMS text you want to send

  int unsafe=600-turb;

  mySerial.println(unsafe);

  mySerial.print("Below limits");

  delay(100);

  mySerial.println((char)26); // ASCII code of CTRL+Z

  delay(1000);

}

--------------------------------------------------------------------------------

The code is used for the microcontroller board which is the NodeMCU. The simple task of the microcontroller is to read the sensor values and publish them to the IBM Watson cloud. If the pollution level of the water falls below a threshold limit the GSM module is triggered which send notifications via SMS to the registered users. The cloud side is coded using node-red UI which forwards and stores the payload into the Cloudant Database.