ELECTRONICS PROJECTS

RECENT ARTICLES

Bidirectional Visitor Counter using IR sensors and Arduino Uno

May 24, 2022

Electronic Dice using Arduino and 7 Segment Display

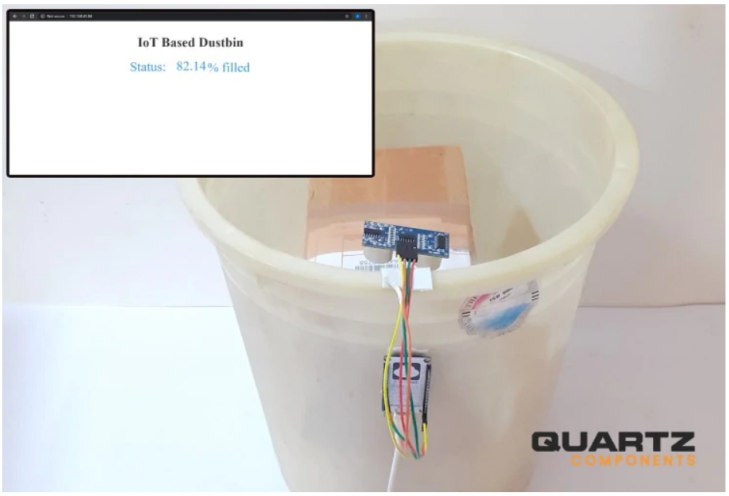
April 18, 2022

Raspberry Pi Pico: What you should know and How to get started?

March 14, 2022

IoT Based Smart Dustbin using NodeMCU

1 comment / Posted on January 20, 2020 by Quartz Components

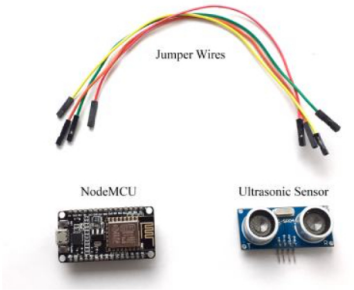


In this project tutorial, we are going to make an **IoT based Dustbin**. You can check whether the dustbin is full or empty through a webpage. This dustbin updates its status in percentage in every 5 seconds, and when the dustbin is filled more than 70%, it sends an email that your dustbin is almost full.

We used an ultrasonic sensor and NodeMCU to build this **IoT dustbin project**. The ultrasonic sensor calculates the occupancy by detecting the trash in the dustbin. The distance between ultrasonic and trash is converted to percentage so that instead of showing two or three levels, we can show the dustbin status in percentage. IFTTT Webhooks is used to get the data from NodeMCU and send an email whenever the trash level crosses the 70% criteria.

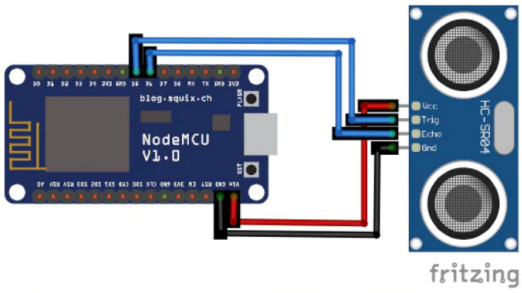
COMPONENT REQUIRED

- NodeMCU - Buy
- Ultrasonic Sensor - Buy
- Breadboard - Buy
- Jumper wires - Buy
- 12V Adapter - Buy



IOT SMART DUSTBIN CIRCUIT DIAGRAM

Circuit diagram for IoT based dustbin is given below.



In this project, we are **interfacing the ultrasonic sensor with NodeMCU**. HC-SR04 ultrasonic sensor works on 5V, so if you connect it to 3.3V, it won't work. V_{CC} pin of the ultrasonic sensor is connected to the V_{IN} pin of NodeMCU. Trig and Echo pins are connected to D5 and D6 pin of NodeMCU while the GND pin of the sensor is connected to the GND pin of NodeMCU. A 5V power supply powers NodeMCU.

IFTTT Setup

Here we are using IFTTT to send Email notifications when the temperature goes past to critical value. IFTTT (If This Then That) is a web-based service by which

we can create chains of conditional statements, called applets. Using these applets, we can send Emails, Twitter, Facebook notifications.

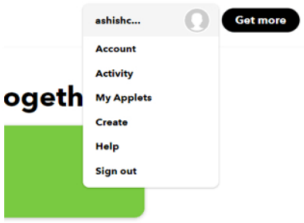
To use the IFTTT sign in to your IFTTT account if you already have one or create an account.

Now search for 'Webhooks' and click on the Webhooks in Services section.

Now, in the Webhooks window, click on 'Documentation' in the upper right corner to get the private key.

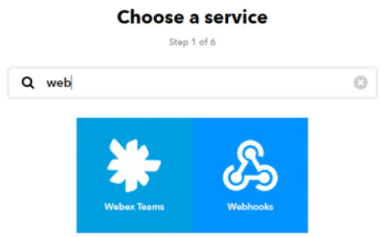
Copy this key. It will be used in the program.

After getting the private key now, we will create an applet using Webhooks and Email services. To create an applet, click on your profile and then click on 'Create.'

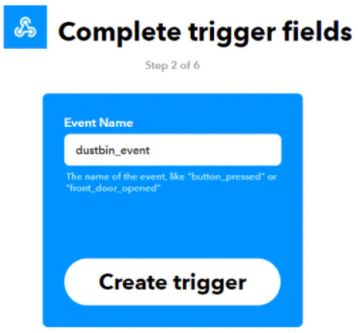


Now in the next window, click on the 'This' icon.

Now search for Webhooks in the search section and click on 'Webhooks.'



Now choose 'Receive a Web Request' trigger and in the next window, enter the event name as dustbin_event and then click on create trigger.



After this, click on 'Then That' and then click on Email.

Now in email, click on 'send me an email' and enter the email subject and body and then click on create action.

In the last step, click on 'Finish' to complete the Applet setup.

IOT SMART DUSTBIN CODE EXPLANATION

We are using **Arduino IDE** to program **NodeMCU**. So, make sure you have downloaded NodeMCU board files. If you don't know how to install the NodeMCU board files in Arduino IDE, then follow this article.

Programming NodeMCU Using Arduino IDE

Complete code is given at the end of the document. Here we are explaining the code step by step.

So start your code by including all the required library files. The ultrasonic sensor doesn't require a library file, so we only need ESP8266WiFi.h library file.

```
#include <ESP8266WiFi.h>
```

After that, define the pins where you connected the Trig and Echo pins and also define two variables for calculating distance and duration.

```
const int trigPin = D5;
const int echoPin = D6;
long duration;
int distance;
```

After that, make instances for Wi-Fi name, Wi-Fi password, IFTTT host name and private key.

```
const char* ssid = "Wi-Fi Name";
const char* password = "Password";
const char *host = "maker.ifttt.com";
const char *privateKey = "Private key";
```

Now to access the WiFiServer, we declared an object WifiServer library. 80 is the default port for HTTP.

```
WiFiServer server(80);
```

Now inside the void loop function, calculate the time between triggered and received signal. This time will be used to calculate the distance.

```
duration = pulseIn(echoPin, HIGH);
distance = duration * 0.0340 / 2;
```

After that, we distance converted the distance into percentage to show the dustbin occupancy.

```
level =((28-distance)/28.0)*100;
```

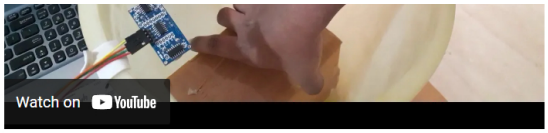
Then we compared the dustbin occupancy, and if the occupancy level is 70 or more than 70, then it will trigger an IFTTT event to send warning Email.

```
if ( level >= 70) {
      send_event("dustbin_event");
      }
```

TESTING THE IOT BASED DUSTBIN PROJECT

Once your hardware and code are ready, upload the code and put the ultrasonic sensor at the top of the dustbin. Now check the webpage using the IP address that is printed on Serial Monitor. It should show the dustbin occupancy level. And if the dustbin is filled more than 70%, then it will send you a warning email. In this project, I defined the warning level according to the length of my dustbin. You can change it according to yours.

VIDEO



CODE

```
#include <ESP8266WiFi.h>
const int trigPin = D5;
const int echoPin = D6;
long duration;
int distance;

void send_event(const char *event);
float level;

const char* ssid = "Wi-Fi Name";
const char* password = "Passwoed";

const char *host = "maker.ifttt.com";
const char *privateKey = "Private Key"; Enter the Key that you copied from Webhooks.

WiFiServer server(80);

void setup() {
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
Serial.begin(9600);
Serial.print("Connecting to Wifi Network");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("Successfully connected to WiFi.");
Serial.println("IP address is : ");
Serial.println(WiFi.localIP());
server.begin();
Serial.println("Server started");


}

void loop() {

digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = duration * 0.0340 / 2;
Serial.println("Distance");
Serial.println(distance);
level =((28 -distance)/28.0)*100;
Serial.println("level");
Serial.println(level);

delay(1000);


WiFiClient client = server.available();

if (client)
{
Serial.println("Web Client connected ");
String request = client.readStringUntil('\r');
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("Connection: close"); // the connection will be closed after completion of the response
client.println("Refresh: 10"); // update the page after 10 sec
client.println();
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<style>html { font-family: Cairo; display: block; margin: 0px auto; text-align: center;color: #333333; background-color: ##f3ffee;}");
client.println("body{margin-top: 50px;}");
client.println("h1 {margin: 50px auto 30px; font-size: 50px; text-align: center;}");
client.println(".side_adjust{display: inline-block;vertical-align: middle;position: relative;}");
client.println(".text1{font-weight: 180; padding-left: 5px; font-size: 50px; width: 170px; text-align: left; color: #3498db;}");
client.println(".data1{font-weight: 180; padding-left: 1px; font-size: 50px;color: #3498db;}");
client.println(".data{padding: 1px;}");
client.println("</style>");
client.println("</head>");
client.println("<body>");
client.println("<div id=\"webpage\">");
client.println("<h1>IoT Based Dustbin</h1>");
client.println("<div class=\"data\">");
client.println("<div class=\"side_adjust text1\">Status:</div>");
client.println("<div class=\"side_adjust data1\">");
client.print(level);
client.println("<div class=\"side_adjust text1\">% filled</div>");
client.println("</div>");
client.println("</div>");
client.println("</body>");
client.println("</html>");
//client.println("<h1>Level Indicator</h1>");

if ( level >= 70) {
send_event("dustbin_event");
}

}
}

void send_event(const char *event)
```

```
{
Serial.print("Connecting to ");
Serial.println(host);

// Use WiFiClient class to create TCP connections
WiFiClient client;
const int httpPort = 80;
if (!client.connect(host, httpPort)) {
Serial.println("Connection failed");
return;
}

// We now create a URI for the request
String url = "/trigger/";
url += event;
url += "/with/key/";
url += privateKey;

Serial.print("Requesting URL: ");
Serial.println(url);

// This will send the request to the server
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
"Host: " + host + "\r\n" +
"Connection: close\r\n\r\n");
while(client.connected())
{
if(client.available())
{
String line = client.readStringUntil('\r');
Serial.print(line);
} else {
// No data yet, wait a bit
delay(50);
};
}

Serial.println();
Serial.println("closing connection");
client.stop();
}
```

TAGS : IoT Projects, NodeMCU

← Older Post

Newer Post →

1 COMMENT

Posted on Jan 20, 2020 by Harshitha

I am getting the email but this happens only once .
Again if the bin fills above 70% I could not receive email I.e for the next time,what should I do for that. What might be d reason ?

LEAVE A COMMENT

Your Name

Your Email

Your Comment

All blog comments are checked prior to publishing

POST COMMENT