SenseEgypt

User Guide

Table of Contents

Introduction

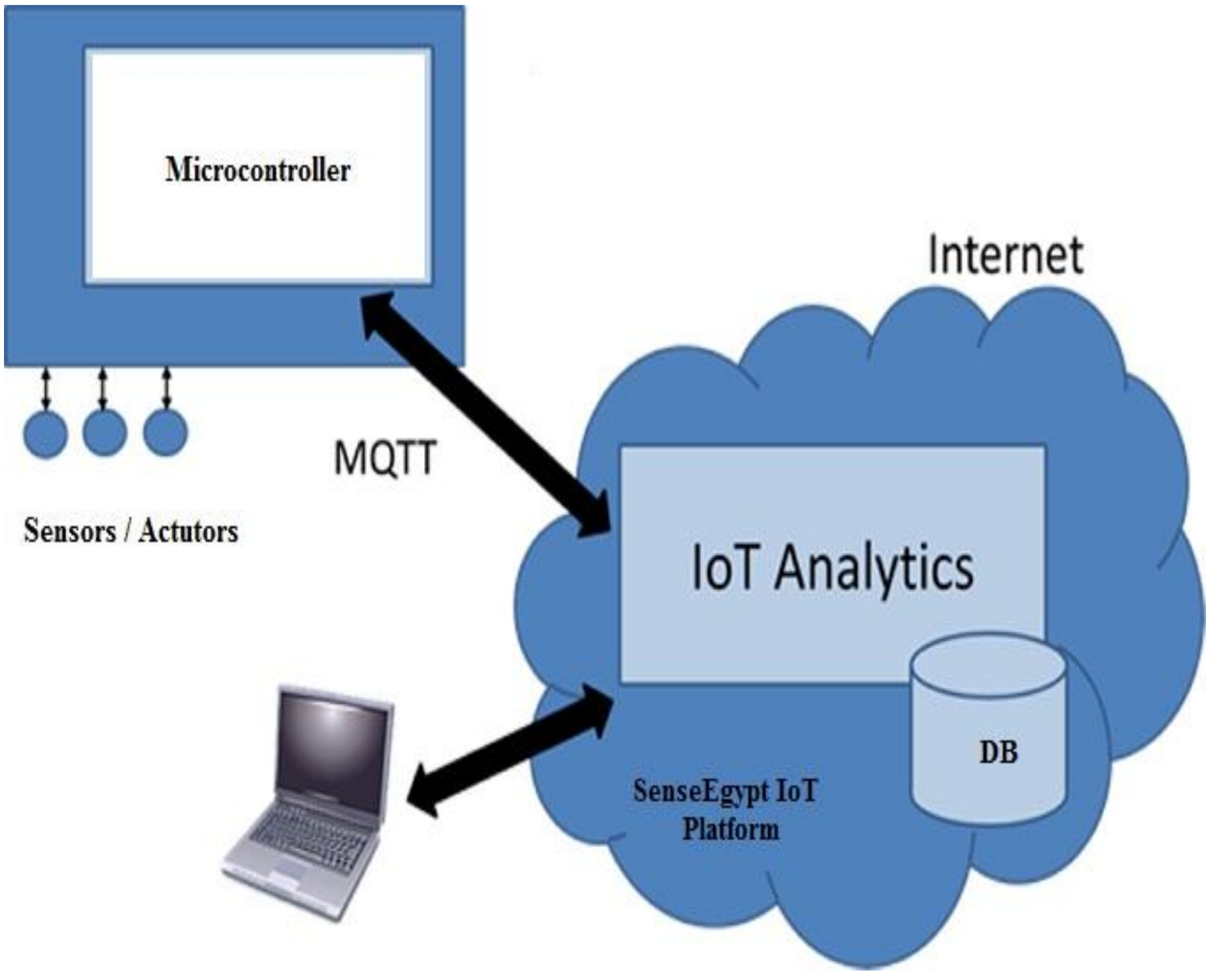
The Internet of Things (IoT) provides access to a broad range of embedded devices and web services. SenseEgypt is an IoT platform that enables you to collect, store, analyze, visualize, and act on data from sensors or actuators, such as Arduino, Raspberry P , Netduino , and other hardware. For example, with SenseEgypt you can create sensor-logging applications, location-tracking applications, and a social network of things with status updates, so that you could have your home thermostat control itself based on your current location.
SenseEgypt acts as the IoT platform for data collection and analytics that serves as a bridge connecting edge node devices such as temperature and pressure sensors to collect data and data exploratory analysis software to analyze data. SenseEgypt serves as the data collector which collects data from edge node devices and also enables the data to be pulled into a software environment for historical analysis of data.

The typical SenseEgypt workflow lets you:

1- Create a Channel and collect data
2- Analyze and Visualize the data
3- Act on the data using any of several Apps

This guide contains steps to connect and upload data from devices to senseEgypt Iot platform. IoT Analytics includes resources for the collection and analysis of sensor data.

This guide shows IoT developers how to access and take advantage of this valuable resource. Several elements are necessary in this multi-step process. Setting up an IoT Analytics account first and then proceeding with connectivity and sensors is the best pathway to success. Your device must be able to access the internet so that it can connect to the cloud. Networks with ports blocked, or firewalls may see difficulty in connectivity. Two pieces of software are involved on the device, one to collect data, the other to send data. This document will emphasize connectivity to the IoT Analytics site first (with device), then to collect data to send.



In order to use SenseEgypt , make sure that you have the appropriate hardware and browsers that satisfy the system requirements.

System Requirements

Devices

Sensor data can be sent to SenseEgypt™ from Arduino®, Raspberry Pi™, BeagleBone Black, and other hardware. For devices to communicate with SenseEgypt, they must support MQTT protocols. Your firewall must allow connection to the standard ports for these protocols.

Website

To access SenseEgypt through a web browser, please ensure your system meets the following requirements.

Supported Browsers

SenseEgypt is compatible with most modern web browsers running on Windows®, Mac, and Linux® operating systems as well as Chromebooks.

For browsers (like Chrome and Firefox) that update automatically, the current stable version is supported. For other browsers, supported versions are shown below. If you are using an older version, you should update to the version listed below.

For the best overall experience, we recommend using Google Chrome.

SenseEgypt is also compatible with other popular browsers.

Required Browser Settings

Cookies enabled

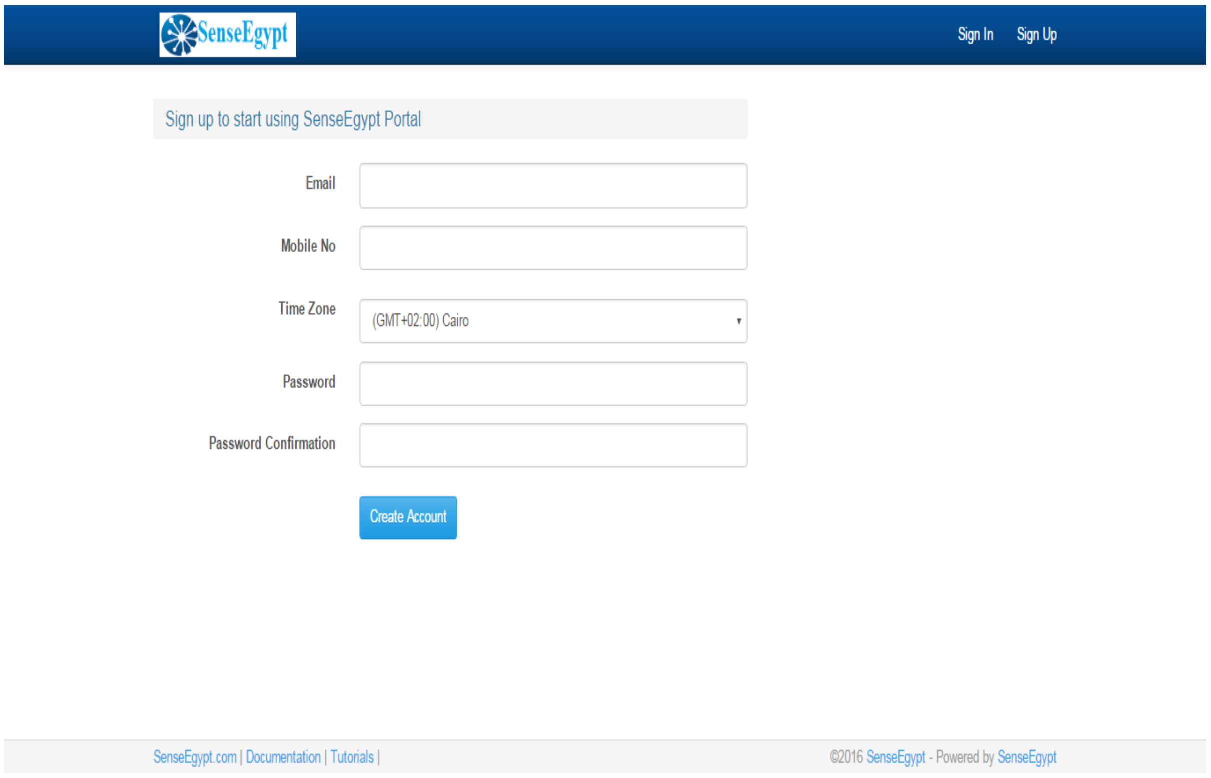
Pop-ups enabled

JavaScript enabled

Internet Connection

Broadband connection recommended for interactive use of the website

New User Registration


SenseEgypt
Sign In
Sign Up
Create Account
SenseEgypt.com | Documentation | Tutorials |
©2016 SenseEgypt - Powered by SenseEgypt

Sign up to start using SenseEgypt Portal

Email

Mobile No

Time Zone: (GMT+02:00) Cairo

Password

Password Confirmation

Users will be asked to enter or select the following information

E-Mail
Enter the user’s e-mail address. E-mail addresses are not case sensitive.

Time Zone
Select Time Zone.

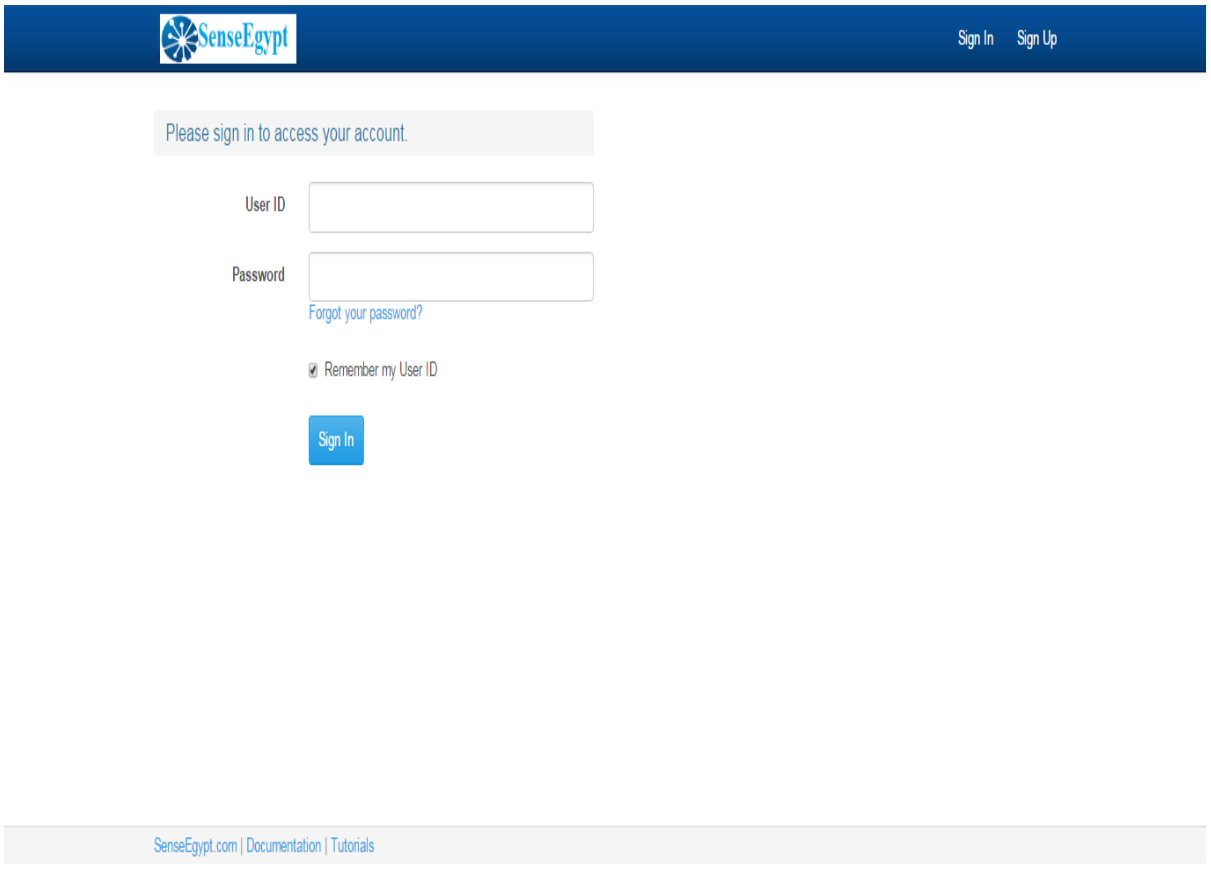
Mobile No
Enter the user’s Mobile number.

Password
Passwords must be at least eight characters long and contain at least one letter character and one numeric character. Passwords are case sensitive.

Password Confirmation
As a security check, users will be asked to retype their password.

Create Account
Press "Create Account" to submit the registration information and continue to the next screen

Login Page



Please sign in to access your account.

User ID

Password

Forgot your password?

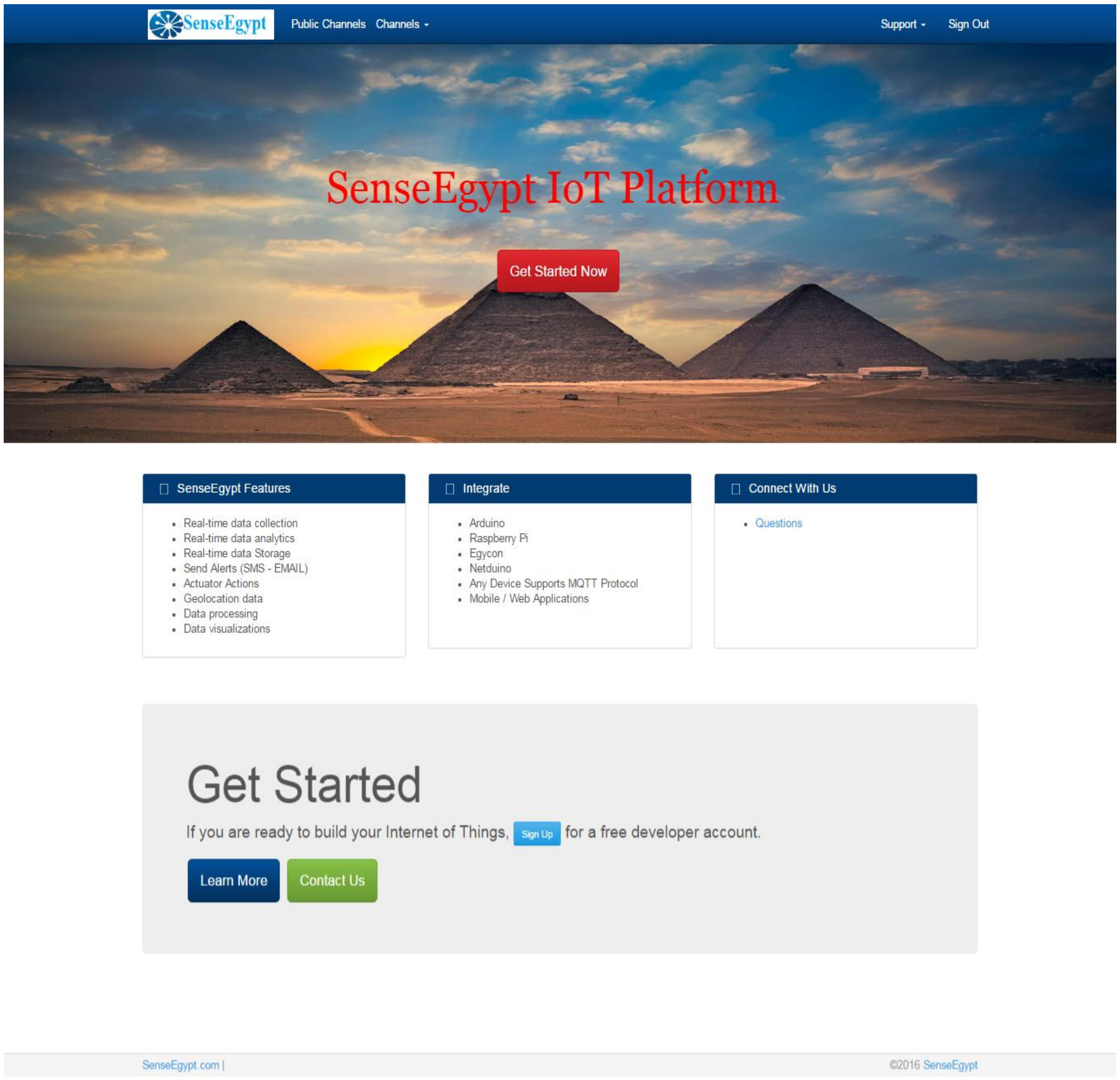
Remember my User ID

Users who have previously registered in the platform as shown in the previous page must login by:

Entering their User ID.
Entering their Password.
Press on Sign In button to advance to the next screen and begin using the application.

Site Registration
Users who have not previously registered in the platform must select “Sign Up" from the top panel ” to access the “New User Registration” page.

SenseEgypt Home Page



SenseEgypt Features

- Real-time data collection
- Real-time data analytics
- Real-time data Storage
- Send Alerts (SMS - EMAIL)
- Actuator Actions
- Geolocation data
- Data processing
- Data visualizations

Integrate

- Arduino
- Raspberry Pi
- Egycon
- Netduino
- Any Device Supports MQTT Protocol
- Mobile / Web Applications

Connect With Us

- Questions

Get Started

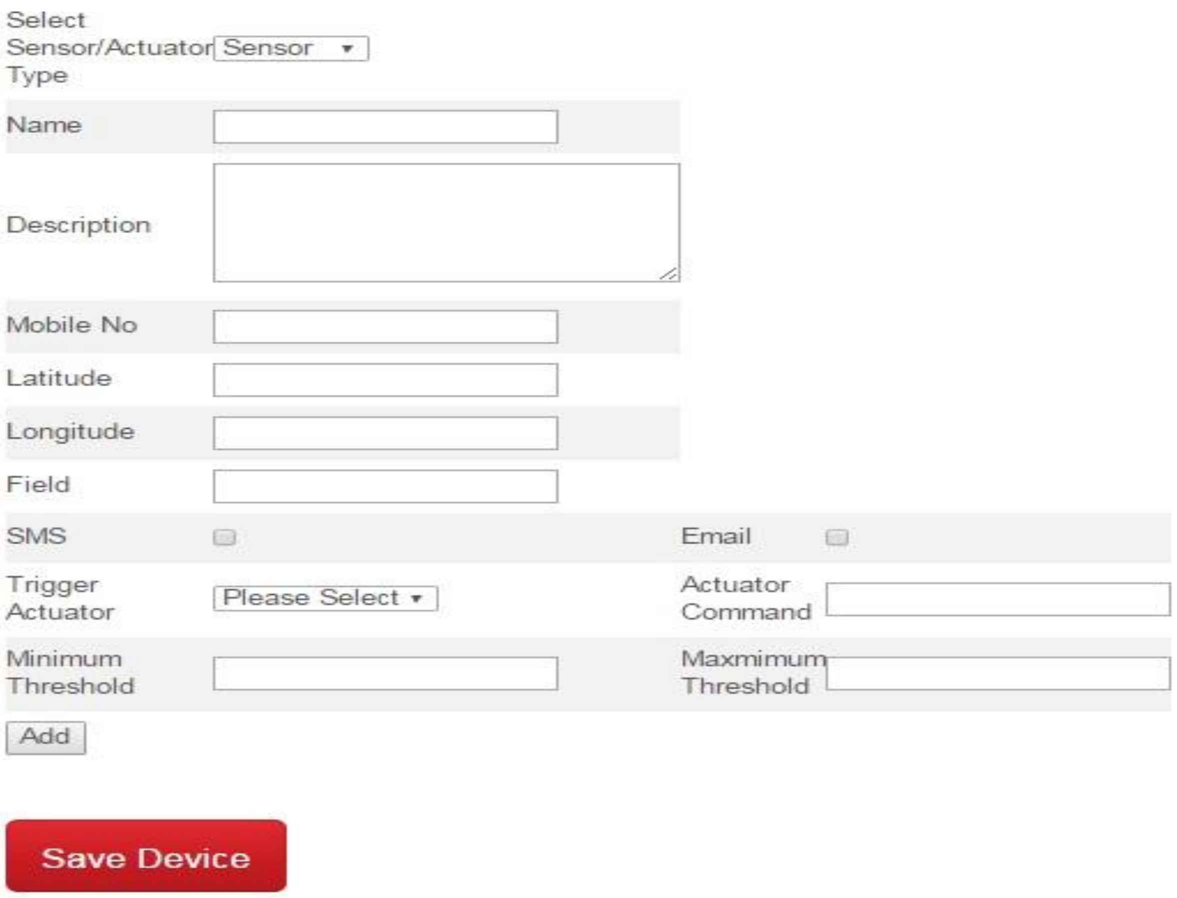
If you are ready to build your Internet of Things, Sign Up for a free developer account.

SenseEgypt.com | ©2016 SenseEgypt

After logging into senseEgypt Web Application, the user will advance to the HomePage. From the top panel in the header user can select the following :

1- **Public Channels**, when the user press on this button then the channels page will be displayed and all channels that are created to be public will be displayed here for public use.
2- **My Channels,** when the user select this item then the channels page will be displayed and he can add new channels to his account .
3- **Support > Documentation,** when user select this item then the user guide manual will be previewed in a new tab.

Channels Page


Select Sensor/Actuator Type	Sensor ▾		
Name			
Description			
Mobile No			
Latitude			
Longitude			
Field			
SMS	☐	Email	☐
Trigger Actuator	Please Select ▾	Actuator Command	
Minimum Threshold		Maxmimum Threshold	

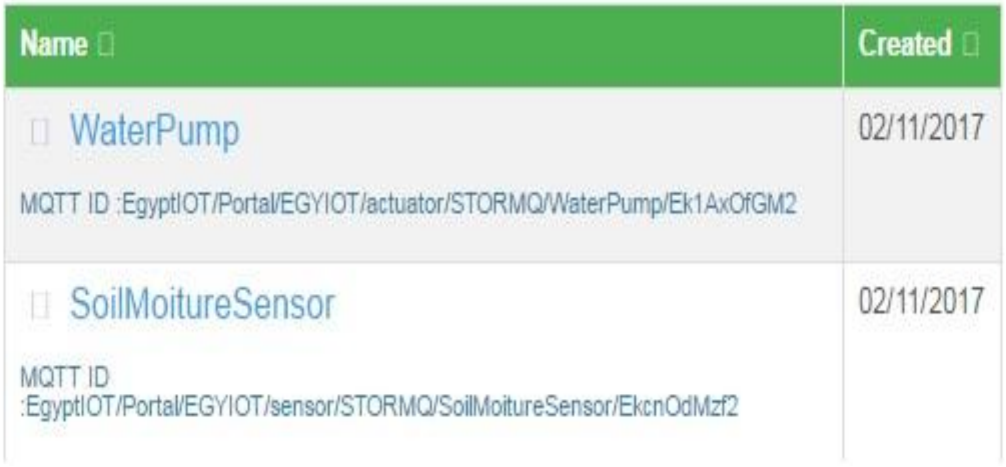
SenseEgypt is an IoT platform that uses channels to store data sent from devices. With the settings described in Channel Configurations, you create a channel, and then send and retrieve data to and from the channel. You can make your channels public to share data with every one or private for your use only. Users will be asked to enter or select the following information

Field	Value
Select Sensor/Actuator Type	Select Device Type Sensor or Actuator
Name	Enter Device Name
Description	Enter Device Description
Mobile No	Enter Mobile No that used in sending SMS alerts
Latitude	Enter Latitude value where the device is locating
Longitude	Enter Longitude value where the device is locating
SMS & Email	Check the SMS & Email if you need alerts if there is an abnormal events have been detected to sensor values
Trigger Actuator	Select The actuator if you need to send commands to it when an events have been detected
Actuator Command	The Command value to be sent to selected actuators
Minimum Threshold	Enter the Minimum threshold for the rules to be validated against sensor measurements
Maximum Threshold	Enter the Maximum threshold for the rules to be validated against sensor measurements

While registering a new sensor, the user has to enter the minimum and maximum thresholds as they are used in the analytics layer so we can detect if an abnormal events have been occurred according to sensor measurements as follow:
If (Sensor Measurement < Minimum Threshold || Sensor Measurement > Maximum Threshold) then there is an abnormal event has been detected and there are actions that should be taken as follows :

1- Send command to an actuator with the command value set during senor registration.
2- Send SMS / Email Alerts if they are selected during sensor registration.

After user pressing on the Save Device button then this request is sent to the platform and a MQTT ID is created for that device as will shown in table in the left side of the page as shown in the below screenshot



Name	Created
WaterPump MQTT ID :EgyptIOT/Portal/EGYIOT/actuator/STORMQ/WaterPump/Ek1AxOfGM2	02/11/2017
SoilMoitureSensor MQTT ID :EgyptIOT/Portal/EGYIOT/sensor/STORMQ/SoilMoitureSensor/EkcnOdMzf2	02/11/2017

SenseEgypt, an IoT platform, enables clients to update a channel feed by publishing messages to the MQTT broker. MQTT is a publish/subscribe model that runs over TCP/IP sockets or WebSockets. MQTT over WebSockets can be secured with SSL. A client device connects to the MQTT broker and can publish to a channel. SenseEgypt supports publishing or subscribing to/from channels using MQTT.
The MQTT ID if it is created for a sensor then it will be used to publish sensor measurements to the MQTT broker or if it is created for an actuator then it will be used to subscribe for commands sent from the MQTT broker .
Code for interacting with SenseEgypt
https://github.com/mqtt/mqtt.github.io/wiki/libraries

1- Use Arduino client to publish measurements to senseEgypt

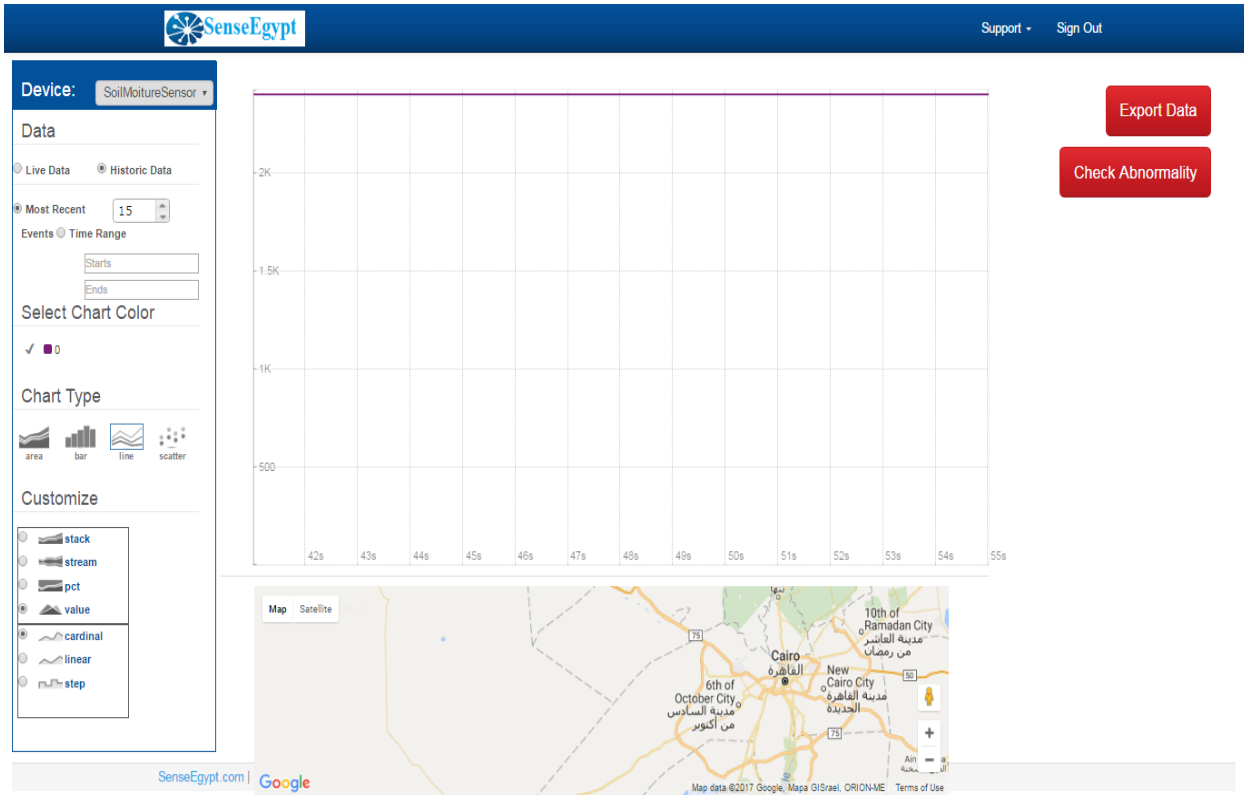
```
#include <SPI.h>
#include <WiFi101.h>
#include <PubSubClient.h>
#include "DHT.h"
// DHT Sensor connected to digital pin 2
#define DHTPIN 2
// Type of DHT sensor
#define DHTTYPE DHT11
// Analog light sensor connected to analog pin A0
#define LIGHTPIN A0
// Initialize DHT sensor
DHT dht(DHTPIN, DHTTYPE);
//  your network SSID (name)
char ssid[] = "YOUR-NETWORK-SSID";
// your network password
char pass[] = "YOUR-NETWORK-PWD";
// Initialize the Wifi client library
WiFiClient client;
// Initialize the PuBSubClient library
PubSubClient mqttClient(client);
// Define the MQTT broker
const char* server = " broker.mqtt-dashboard.com";
// track the last connection time
unsigned long lastConnectionTime = 0;
// post data every 20 seconds
const unsigned long postingInterval = 20L * 1000L;
void setup() {
  // Begin serial transfer
  Serial.begin(9600);
  // Set a temporary WiFi status
  int status = WL_IDLE_STATUS;
  // Attempt to connect to WiFi network
  while (status != WL_CONNECTED)
  {
    // Connect to WPA/WPA2 Wi-Fi network
    status = WiFi.begin(ssid, pass);
    // Wait 10 seconds for connection
    delay(10000);
  }
  Serial.println("Connected to wifi");
  // Set the MQTT broker details
  mqttClient.setServer(server, 1883);
}
void loop() {
 // Check if MQTT client has connected else reconnect
  if (!mqttClient.connected())
  {
    reconnect();
  }
  // Call the loop continuously to establish connection to the server
  mqttClient.loop();
  // If interval time has passed since the last connection, Publish data to SenseEgypt
  if (millis() - lastConnectionTime > postingInterval)
  {
    mqttpublish();
  }
}
void reconnect()
{
  // Loop until we're reconnected
  while (!mqttClient.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // Connect to the MQTT broker
    if (mqttClient.connect("ArduinoWiFi101Client"))
    {
      Serial.println("connected");
    } else
    {
      Serial.print("failed, rc=");
      // Print to know why the connection failed
// See http://pubsubclient.knolleary.net/api.html#state for the failure code and its reason
      Serial.print(mqttClient.state());
```

```
        Serial.println(" try again in 5 seconds");
        // Wait 5 seconds before retrying to connect again
        delay(5000);
       }
     }
   }
   void mqttpublish() {
     // Read temperature from DHT sensor
     float t = dht.readTemperature(true);
     String data = String(t, DEC);
     // Get the data string length
     int length = data.length();
     char msgBuffer[length];
     // Convert data string to character buffer
     data.toCharArray(msgBuffer,length+1);
     Serial.println(msgBuffer);
     // Publish data to SenseEgypt use the MQTT ID for the sensor from devices table
     mqttClient.publish("MQTT ID from the devices table",msgBuffer);
     // note the last connection time
     lastConnectionTime = millis();
   }
```

2- Use Netduino client to publish measurements to senseEgypt

```
using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware;
using SecretLabs.NETMF.Hardware.NetduinoPlus;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
namespace Netduino.Messaging.MQTT
{
    public class Program
    {
        public static OutputPort pump;
        public static void Main()
        {
    MqttClient client = new MqttClient("broker.mqtt-dashboard.com", 1883, false, null);

            SecretLabs.NETMF.Hardware.AnalogInput A0 = new
SecretLabs.NETMF.Hardware.AnalogInput(Pins.GPIO_PIN_A0);
            pump = new OutputPort(Pins.GPIO_PIN_D0, false);
            client.MqttMsgPublishReceived += client_MqttMsgPublishReceived;
            client.MqttMsgSubscribed += client_MqttMsgSubscribed;
            client.MqttMsgUnsubscribed += client_MqttMsgUnsubscribed;
            client.MqttMsgPublished += client_MqttMsgPublished;
//Use the MQTT ID for the device created from the devices table
            client.Subscribe(new string[] {
"EgyptIOT/Portal/EGYIOT/actuator/STORMQ/WaterPump/Ek1AxOfGM2" }, new byte[] {
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE });
            var state = client.Connect("netduino_Cl1515151ient5545469");

            float temp = 80;
            while (true)
            {
                temp = temp + 1;
                int analogReading = (int)A0.Read();
client.Publish("EgyptIOT/Portal/EGYIOT/sensor/STORMQ/SoilMoitureSensor/EkcnOdMzf2",
Encoding.UTF8.GetBytes(analogReading.ToString()));
                Thread.Sleep(2000);

            }
        }

 public static void client_MqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs
e)
        {
            // access data bytes throug e.Message
            Debug.Print("Message: " + e.Message);

            var message = UTF8Encoding.UTF8.GetChars(e.Message);

            string command = new string(message);



            if (command == "open")
            {
                pump.Write(true);

            }
            else
            {
                pump.Write(false);

            }

        }

        public static void client_MqttMsgUnsubscribed(object sender,
MqttMsgUnsubscribedEventArgs e)
        {
            // write your code
            Debug.Print("Unsubscribed ");
        }

        public static void client_MqttMsgSubscribed(object sender,
MqttMsgSubscribedEventArgs e)
        {
            // write your code
            Debug.Print("Subscribed ");
        }

        public static void client_MqttMsgPublished(object sender,
MqttMsgPublishedEventArgs e)
        {
            // write your code
            Debug.Print("Published ");
        }


        static void delayLoop(int interval)
        {
            long now = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;
            int offset = (int)(now % interval);
            int delay = interval - offset;
            Thread.Sleep(delay);
        }

    }
}
```

<u>Dashboard Page</u>



The Dashboard page enables us to perform the following :
1- View the sensor measurements in real time and also we can view the historic measurements of the sensor sent before to senseEgypt platform.
2- View the real-time commands sent to actuators and also we can view the historic commands sent to actuators from senseEgypt platform.
3-We can export sensors/ Actuators data.
4-Check Abnormality of sensors measurements in real-time.
5-View Sensor / Actuator location on google Map.