CHILD TRACKER WITH EMERGENCY NOTIFIER

SUBMITTED BY

Naveen

M.Adarsh

# Abstract

This project presents a child safety system with an emergency tracker using GPS and GSM modules.This project aim is to protect the children when they are alone in the house or an area

The system can be interconnected with the notification and alert the parents.This detection and messaging system is composed of a GPS receiver, Microcontroller and a GSM Modem. GPS Receiver gets the location information from satellites in the form of latitude and longitude.

The Microcontroller processes this information and this processed information is sent to the user using GSM modem A GSM modem is interfaced to the MCU. The GSM modem sends an SMS to the predefined mobile number. When a child is in danger and in need of self-defense if the child crosses the GEOFENCE the entire system will be activated then immediately a SMS will be sent to concern person with location using GSM and GPS.

TECHNOLOGIES USED

* GSM TECHNOLOGY

DEFINITION OF GSM

Global system for mobile communication (GSM) is a globally accepted standard for digital cellular communication. GSM is the name of a standardization gro

up established in 1982 to create a common European mobile telephone standard that would formulate specifications for a pan-European mobile cellular radio system operating at 900 MHz. Figure3.1 GSM module 3.1.2 HISTORY OF GSM Global system for mobile communication is a globally accepted standard for digital cellular communication. GSM is the name of a standardization group established in 1982 to create a common European mobile telephone standard that would formulate specifications for a pan-European mobile cellular radio system operating at 900 MHz It is estimated that many countries outside of Europe will join the GSM partnership.GSM, the Global System for Mobile communications, is a digital cellular communications system, which has rapidly gained acceptance and market share worldwide, although it was initially developed in a European context. In addition to digital transmission, GSM incorporates many advanced services and features,

including ISDN compatibility and worldwide roaming in other GSM networks. The advanced services and architecture of GSM have made it a model for future third generation cellular systems, such as UMTS.

GPS RECEIVER: GPS, in full Global Positioning System, space-based radio-navigation system that broadcasts highly accurate navigation pulses to users on or near the Earth. In the United States’ Navstar GPS, 24 main satellites in 6 orbits circle the Earth every 12 hours. In addition, Russia maintains a constellation called GLONASS (Global Navigation Satellite)

**Create and Monitor Geofences**

Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest.

You can have multiple active geofences, with a limit of 100 per device user across all apps. For each geofence, you can ask Location Services to send you entrance and exit events, or you can specify a duration within the geofence area to wait, or *dwell*, before triggering an event. You can limit the duration of any geofence by specifying an expiration duration in milliseconds. After the geofence expires, Location Services automatically removes it.



**Hardware Required**

GPS:

The **Global Positioning System** (**GPS**), originally **Navstar GPS**, is a satellite-based [radionavigation](https://en.wikipedia.org/wiki/Radionavigation-satellite_service) system owned by the USA government and operated by the UNITED SATES AIR FORCE. It is a [global navigation satellite system](https://en.wikipedia.org/wiki/Satellite_navigation) that provides [geolocation](https://en.wikipedia.org/wiki/Geolocation) and time information to a [GPS receiver](https://en.wikipedia.org/wiki/GPS_receiver) anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. Obstacles such as mountains and buildings block the relatively weak [GPS signals](https://en.wikipedia.org/wiki/GPS_signals).

The GPS does not require the user to transmit any data, and it operates independently of any telephonic or internet reception, though these technologies can enhance the usefulness of the GPS positioning information. The GPS provides critical positioning capabilities to military, civil, and commercial users around the world. The United States government created the system, maintains it, and makes it freely accessible to anyone with a [GPS receiver](https://en.wikipedia.org/wiki/GPS_navigation_device).

The GPS project was launched by the [U.S.Department of Defense](https://en.wikipedia.org/wiki/U.S._Department_of_Defense) in 1973 for use by the United States military and became fully operational in 1995. It was allowed for civilian use in the 1980s. Advances in technology and new demands on the existing system have now led to efforts to modernize the GPS and implement the next generation of [GPS Block IIIA](https://en.wikipedia.org/wiki/GPS_Block_IIIA)satellites and Next Generation Operational Control System (OCX). Announcements from Vice President AL GORE and the [White House](https://en.wikipedia.org/wiki/Clinton_Administration) in 1998 initiated these changes. In 2000, the [U.S.Congress](https://en.wikipedia.org/wiki/U.S._Congress) authorized the modernization effort, [GPS III](https://en.wikipedia.org/wiki/GPS_Block_IIIA). During the 1990s, GPS quality was degraded by the United States government in a program called "Selective Availability"; this was discontinued in May 2000 by a law signed by President [Bill Clinton](https://en.wikipedia.org/wiki/Bill_Clinton).

The GPS service is provided by the United States government, which can selectively deny access to the system, as happened to the Indian military in 1999 during the [Kargil War](https://en.wikipedia.org/wiki/Kargil_War), or degrade the service at any time. As a result, several countries have developed or are in the process of setting up other global or regional satellite navigation systems. The Russian Global Navigation Satellite System ([GLONASS](https://en.wikipedia.org/wiki/GLONASS)) was developed contemporaneously with GPS, but suffered from incomplete coverage of the globe until the mid-2000s. GLONASS can be added to GPS devices, making more satellites available and enabling positions to be fixed more quickly and accurately, to within two meters (6.6 ft). China's [BeiDou Navigation Satellite System](https://en.wikipedia.org/wiki/BeiDou_Navigation_Satellite_System) is due to achieve global reach in 2020. There are also the European Union [Galileo positioning system](https://en.wikipedia.org/wiki/Galileo_(satellite_navigation)), and India's [NAVIC](https://en.wikipedia.org/wiki/Indian_Regional_Navigation_Satellite_System). Japan's [Quasi-Zenith SatelliteSystem](https://en.wikipedia.org/wiki/Quasi-Zenith_Satellite_System) (QZSS) is a GPS [satellite-based augmentation system](https://en.wikipedia.org/wiki/GNSS_augmentation#SatelliteBasedAugmentationSystem) to enhance GPS's accuracy.

When selective availability was lifted in 2000, GPS had about a five-meter (16 ft) accuracy. The latest stage of accuracy enhancement uses the L5 band and is now fully deployed. GPS receivers released in 2018 that use the L5 band can have much higher accuracy, pinpointing to within 30 centimetres or 11.8 inches.

The GPS project was launched in the United States in 1973 to overcome the limitations of previous navigation systems, integrating ideas from several predecessors, including classified engineering design studies from the 1960s. The [U.S. Department of Defense](https://en.wikipedia.org/wiki/U.S._Department_of_Defense) developed the system, which originally used 24 satellites. It was initially developed for use by the United States military and became fully operational in 1995. Civilian use was allowed from the 1980s. [Roger L. Easton](https://en.wikipedia.org/wiki/Roger_L._Easton) of the [Naval Research Laboratory](https://en.wikipedia.org/wiki/Naval_Research_Laboratory), [Ivan A. Getting](https://en.wikipedia.org/wiki/Ivan_A._Getting) of [The Aerospace Corporation](https://en.wikipedia.org/wiki/The_Aerospace_Corporation), and [Bradford Parkinson](https://en.wikipedia.org/wiki/Bradford_Parkinson) of the APPLUED PHYSICS are credited with inventing it. The work of [Gladys West](https://en.wikipedia.org/wiki/Gladys_West) is credited as instrumental in the development of computational techniques for detecting satellite positions with the precision needed for GPS.

The design of GPS is based partly on similar ground-based [radio-navigation](https://en.wikipedia.org/wiki/Radio-navigation) systems, such as [LORAN](https://en.wikipedia.org/wiki/LORAN) and the [Decca Navigator](https://en.wikipedia.org/wiki/Decca_Navigator_System), developed in the early 1940s.

[Friedwardt Winterberg](https://en.wikipedia.org/wiki/Friedwardt_Winterberg) proposed a test of general relativity – detecting time slowing in a strong gravitational field using accurate atomic clocks placed in orbit inside artificial satellites. Special and general relativity predict that the clocks on the GPS satellites would be seen by the Earth's observers to run 38 microseconds faster per day than the clocks on Earth.

NODE MCU:

**NodeMCU** is an open source [IoT](https://en.wikipedia.org/wiki/Internet_of_Things) platform. It includes [firmware](https://en.wikipedia.org/wiki/Firmware)which runs on the [ESP8266](https://en.wikipedia.org/wiki/ESP8266) [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) [SoC](https://en.wikipedia.org/wiki/System_on_a_chip) from [Espressif Systems](https://en.wikipedia.org/w/index.php?title=Espressif_Systems&action=edit&redlink=1), 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](https://en.wikipedia.org/wiki/Lua_(programming_language)) 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](https://en.wikipedia.org/w/index.php?title=SPIFFS&action=edit&redlink=1).

NodeMCU was created shortly after the [ESP8266](https://en.wikipedia.org/wiki/ESP8266) came out. On December 30, 2013, [Espressif Systems](https://en.wikipedia.org/w/index.php?title=Espressif_Systems&action=edit&redlink=1)began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a [Tensilica](https://en.wikipedia.org/wiki/Tensilica) Xtensa LX106 core, widely used in IoT applications (see [related projects](https://en.wikipedia.org/wiki/NodeMCU#Related_projects)). NodeMCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the [gerber](https://en.wikipedia.org/wiki/Gerber_format) file of an ESP8266 board, named devkit v0.9.

Later that month, Tuan PM ported [MQTT](https://en.wikipedia.org/wiki/MQTT) client library from [Contiki](https://en.wikipedia.org/wiki/Contiki) to the ESP8266 SoC platform, and committed to NodeMCU project, then NodeMCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to NodeMCU project, enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays.

In summer 2015 the creators abandoned the firmware project and a group of independent contributors took over. By summer 2016 the NodeMCU included more than 40 different modules. Due to resource constraints users need to select the modules relevant for their project and build a firmware tailored to their needs.

As [Arduino.cc](https://en.wikipedia.org/wiki/Arduino) began developing new MCU boards based on non-[AVR](https://en.wikipedia.org/wiki/AVR_microcontrollers) processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the [Arduino IDE](https://en.wikipedia.org/wiki/Arduino_IDE) so that it would be relatively easy to change the IDE to support alternate toolchains to allow Arduino C/C++ to be compiled for these new processors. They did this with the introduction of the Board Manager and the SAM Core. A "core" is the collection of software components required by the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file for the target MCU's machine language. Some ESP8266 enthusiasts developed an Arduino core for the ESP8266 WiFi SoC, popularly called the "ESP8266 Core for the Arduino IDE".This has become a leading software development platform for the various ESP8266-based modules and development boards, including NodeMCUs.

The NodeMCU programming model is similar to that of [Node.js](https://en.wikipedia.org/wiki/Node.js), only in Lua. It is asynchronous and event-driven. Many functions, therefore, have parameters for callback functions. To give you an idea what a NodeMCU program looks like study the short snippets below. For more extensive examples have a look at the [/lua\_examples](https://github.com/nodemcu/nodemcu-firmware/tree/master/lua_examples) folder in the repository on GitHub.

**ESP8266:**The ESP8266 is a low-cost [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) chip with full TCP/IP stack and [microcontroller](https://en.wikipedia.org/wiki/Microcontroller) capability produced by Shanghai-based Chinese manufacturer. This small module allows microcontrollers to connect to a Wi-Fi network.

**Block Diagram:**

|  |
| --- |
| 5V power supply |

|  |
| --- |
| Arduino Board |

|  |
| --- |
| GPS Module |

|  |
| --- |
| Cloud account |

|  |
| --- |
| Notification |

**CODE:**

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

TinyGPSPlus gps; // The TinyGPS++ object

SoftwareSerial ss(4, 5); // The serial connection to the GPS device

const char\* ssid = "GOOD BOY";

const char\* password = "naveen421";

float latitude , longitude;

int year , month , date, hour , minute , second;

String date\_str , time\_str , lat\_str , lng\_str;

int pm;

#define ORG "hyorfg"

#define DEVICE\_TYPE "NAVEEN"

#define DEVICE\_ID "FaSaK007"

#define TOKEN "Naveen9848115682"

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);

//WiFiServer server1(80);

void setup()

{

Serial.begin(115200);

ss.begin(9600);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

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

{

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

// server.begin();

// Serial.println("Server started");

// Print the IP address

Serial.println(WiFi.localIP());

}

void loop()

{

while (ss.available() > 0)

if (gps.encode(ss.read()))

{

if (gps.location.isValid())

{

latitude = gps.location.lat();

lat\_str = String(latitude , 6);

Serial.println(lat\_str);

longitude = gps.location.lng();

lng\_str = String(longitude , 6);

Serial.println(lng\_str);

}

if (gps.date.isValid())

{

date\_str = "";

date = gps.date.day();

month = gps.date.month();

year = gps.date.year();

if (date < 10)

date\_str = '0';

date\_str += String(date);

date\_str += " / ";

if (month < 10)

date\_str += '0';

date\_str += String(month);

date\_str += " / ";

if (year < 10)

date\_str += '0';

date\_str += String(year);

}

if (gps.time.isValid())

{

time\_str = "";

hour = gps.time.hour();

minute = gps.time.minute();

second = gps.time.second();

minute = (minute + 30);

if (minute > 59)

{

minute = minute - 60;

hour = hour + 1;

}

hour = (hour + 5) ;

if (hour > 23)

hour = hour - 24;

if (hour >= 12)

pm = 1;

else

pm = 0;

hour = hour % 12;

if (hour < 10)

time\_str = '0';

time\_str += String(hour);

time\_str += " : ";

if (minute < 10)

time\_str += '0';

time\_str += String(minute);

time\_str += " : ";

if (second < 10)

time\_str += '0';

time\_str += String(second);

if (pm == 1)

time\_str += " PM ";

else

time\_str += " AM ";

}

}

// Check if a client has connected

//WiFiClient client = server.available();

PublishData(lat\_str,lng\_str);

if (!client.loop())

{

mqttConnect();

return;

}

// Prepare the response

String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n <!DOCTYPE html><html><head><title>GPS Interfacing with NodeMCU</title><style>";

s += "a:link {background-color: YELLOW;text-decoration: none;}";

s += "table, th, td {border: 1px solid black;} </style></head><body><h1 style=";

s += "font-size:300%;";

s += " ALIGN=CENTER> GPS Interfacing with NodeMCU</h1>";

s += "<p ALIGN=CENTER style=""font-size:150%;""";

s += "><b>Location Details</b></p><table ALIGN=CENTER style=";

s += "width:50%";

s += "><tr><th>Latitude</th>";

s += "<td ALIGN=CENTER >";

s += lat\_str;

s += "</td></tr><tr><th>Longitude</th><td ALIGN=CENTER >";

s += lng\_str;

s += "</td></tr><tr><th>Date</th><td ALIGN=CENTER >";

s += date\_str;

s += "</td></tr><tr><th>Time</th><td ALIGN=CENTER >";

s += time\_str;

s += "</td></tr></table> ";

if (gps.location.isValid())

{

s += "<p align=center><a style=""color:RED;font-size:125%;"" href=""http://maps.google.com/maps?&z=15&mrt=yp&t=k&q=";

s += lat\_str;

s += "+";

s += lng\_str;

s += """ target=""\_top"">Click here!</a> To check the location in Google maps.</p>";

}

s += "</body></html> \n";

// client.print(s);

delay(100);

}

void mqttConnect() {

if (!client.connected()) {

Serial.print("Reconnecting MQTT client to "); Serial.println(server);

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

Serial.print(".");

delay(500);

}

initManagedDevice();

Serial.println();

}

}

void initManagedDevice() {

if (client.subscribe(topic)) {

Serial.println("subscribe to cmd OK");

} else {

Serial.println("subscribe to cmd FAILED");

}

}

void PublishData(String lat\_str, String lng\_str){

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\":{\"latitude\":";

payload += lat\_str;

payload+="," "\"longitude\":";

payload += lng\_str;

payload += "}}";

Serial.print("Sending payload: ");

Serial.println(payload);

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

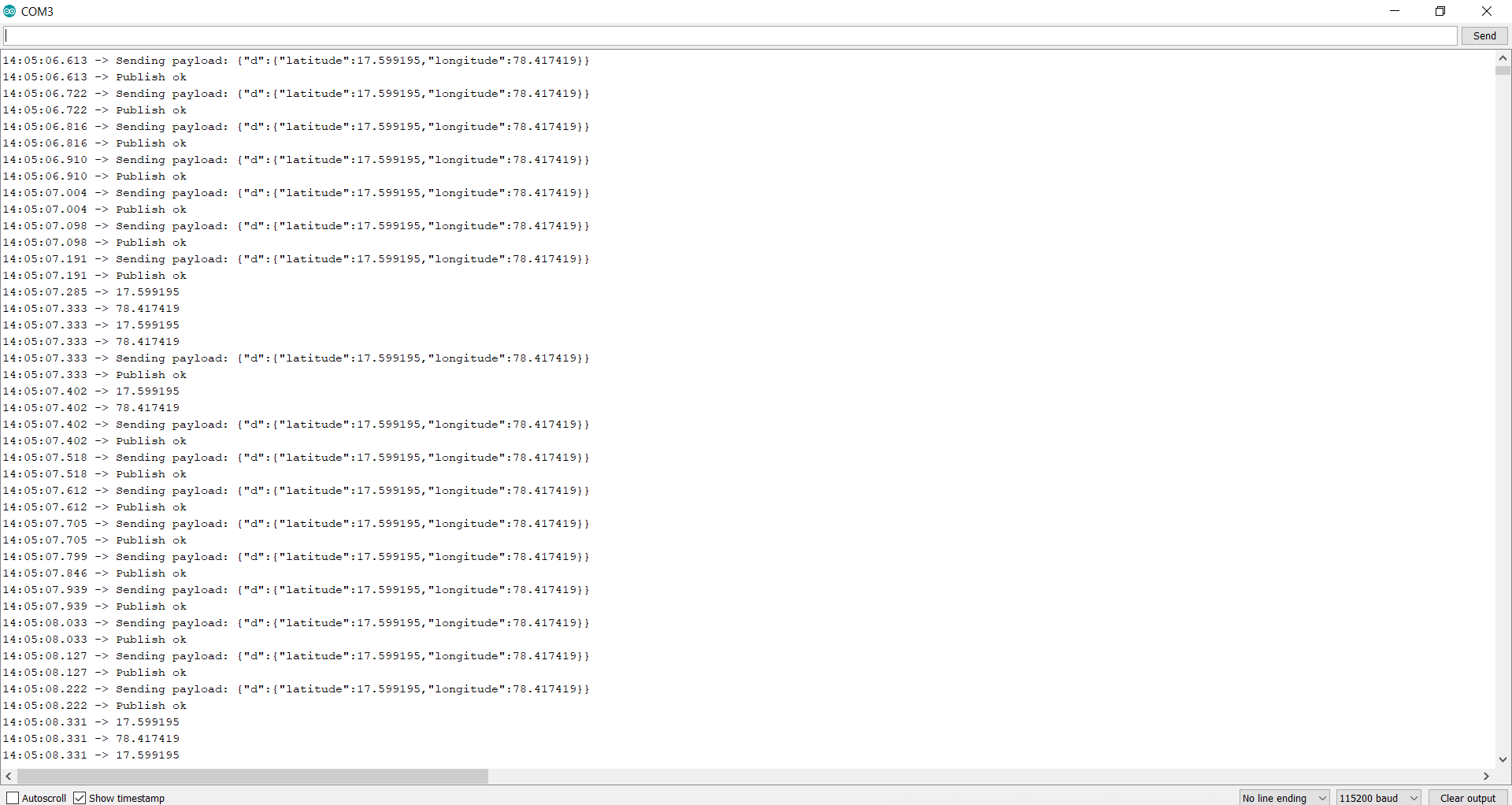
Serial.println("Publish ok");

} else {

Serial.println("Publish failed");

}

}

**OUTPUT:**

**NODE-RED:**

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.Node-RED provides a browser-based flow editor that makes it easy to wire together flows using the wide range of nodes in the palette. Flows can be then deployed to the runtime in a single-click.

JavaScript functions can be created within the editor using a rich text editor.

A built-in library allows you to save useful functions, templates or flows for re-use.

The light-weight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

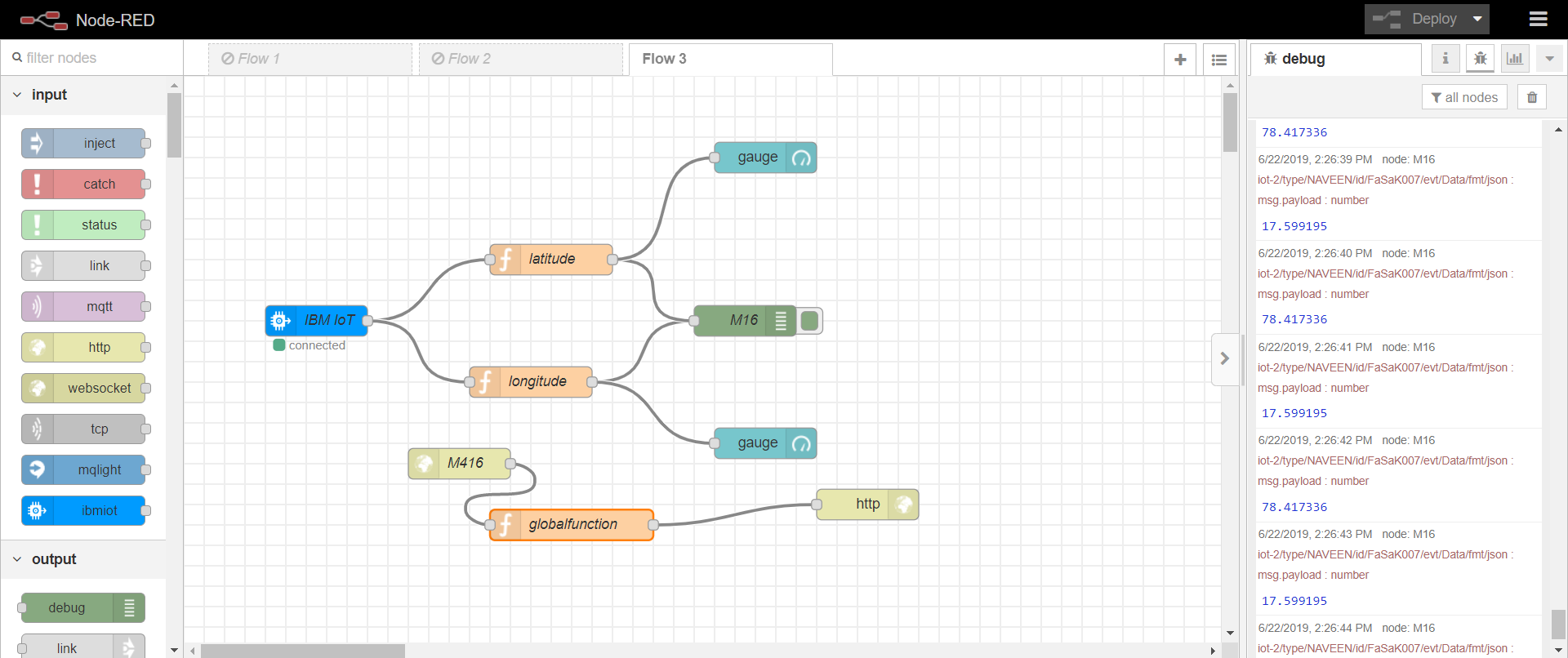
With over 225,000 modules in Node's package repository, it is easy to extend the range of palette nodes to add new capabilities.Node-RED is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud. It provides a browser-based editor that makes it easy to wire together flows using the wide range of node.Within the browser you create your application by dragging nodes from your palette into a workspace and start to wire them together. With a single click, the application is deployed back to the runtime where it is run.

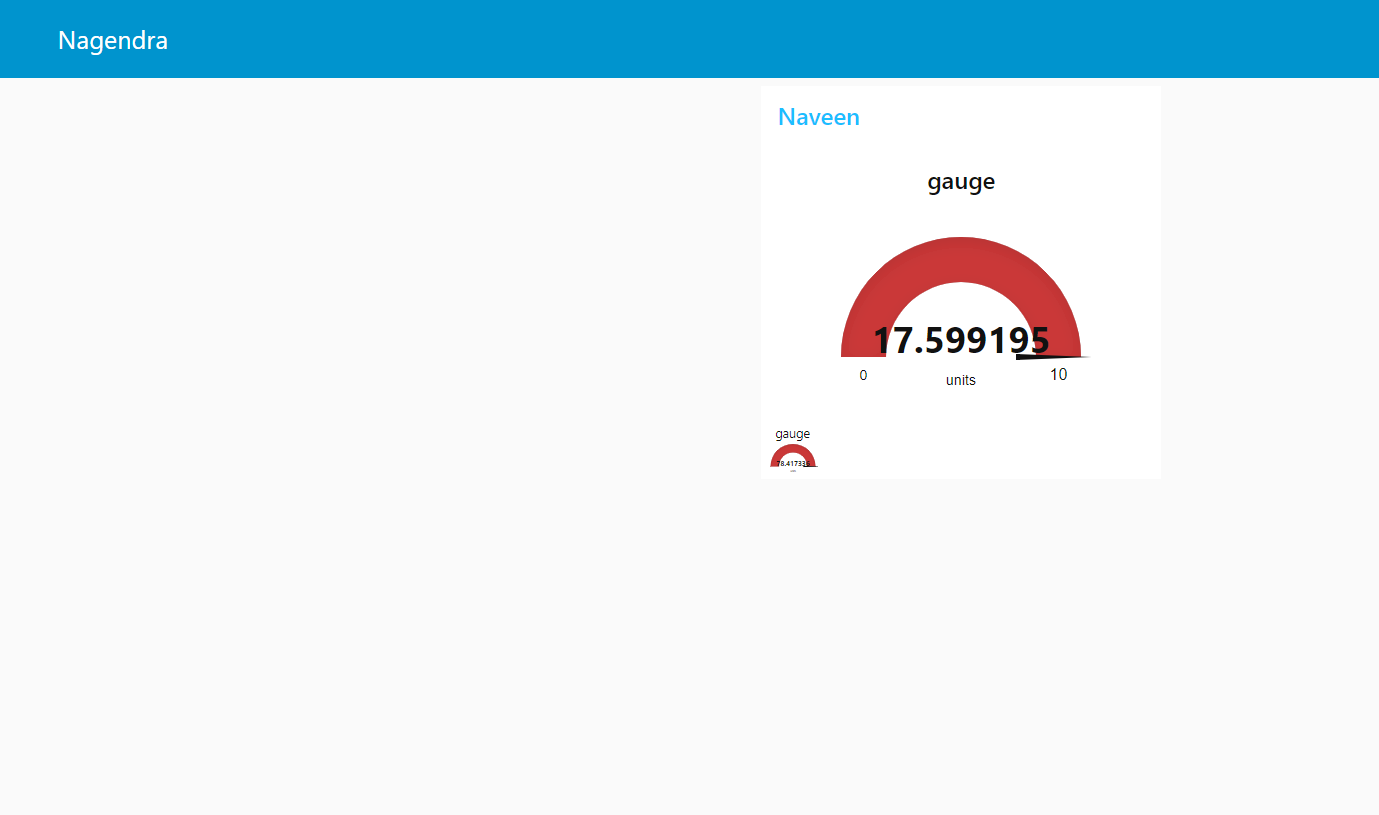
The palette of nodes can be easily extended by installing new nodes created by the community and the flows you create can be easily shared as JSON files

Node-RED started life in early 2013 as a side-project by Nick O’Leary and Dave Conway-Jones of IBM’s Emerging Technology Services group.

What began as a proof-of-concept for visualising and manipulating mappings between MQTT topics, quickly became a much more general tool that could be easily extended in any direction.

It was open-sourced in September 2013 and has been developed in the open ever since, culminating in it being one of the founding projects of the JS Foundation in October 2016.





**MIT APP :**

**App Inventor for Android** is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) web application originally provided by [Google](https://en.wikipedia.org/wiki/Google), and now maintained by the [Massachusetts Institute of Technology](https://en.wikipedia.org/wiki/Massachusetts_Institute_of_Technology) (MIT), which allows newcomers to [computer programming](https://en.wikipedia.org/wiki/Computer_programming) to create [software applications](https://en.wikipedia.org/wiki/Application_software) for the [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) operating system (OS).

It uses a graphical interface very similar to [Scratch](https://en.wikipedia.org/wiki/Scratch_(programming_language)) and the [StarLogo TNG](https://en.wikipedia.org/wiki/StarLogo_TNG) [user interface](https://en.wikipedia.org/wiki/User_interface), which allows users to [drag-and-drop](https://en.wikipedia.org/wiki/Drag-and-drop)visual objects to create an application that can run on Android devices. In creating App Inventor, Google drew upon significant prior research in educational computing, as well as work done within Google on online development environments.

App Inventor and the projects on which it is based are informed by [constructionist learning](https://en.wikipedia.org/wiki/Constructionist_learning) theories, which emphasizes that programming can be a vehicle for engaging powerful ideas through active learning. As such, it is part of an ongoing movement in computers and education that began with the work of [Seymour Papert](https://en.wikipedia.org/wiki/Seymour_Papert) and the MIT Logo Group in the 1960s and has also manifested itself with [Mitchel Resnick](https://en.wikipedia.org/wiki/Mitchel_Resnick)'s work on [Lego Mindstorms](https://en.wikipedia.org/wiki/Lego_Mindstorms)and [StarLogo](https://en.wikipedia.org/wiki/StarLogo).

App Inventor also supports the use of cloud data via an experimental [FirebaseDB](https://en.wikipedia.org/wiki/Firebase#Realtime_Database) component.

The application was made available through request on July 12, 2010, and released publicly on December 15, 2010. The App Inventor team was led by Hal Abelson[1] and Mark Friedman.[5] In the second half of 2011, Google released the source code, terminated its server, and provided funding for the creation of The MIT Center for Mobile Learning, led by App Inventor creator Hal Abelson and fellow MIT professors Eric Klopfer and Mitchel Resnick.[6] The MIT version was launched in March 2012.[7]

On December 6, 2013 (the start of the Hour of Code),[7] MIT released App Inventor 2, renaming the original version "App Inventor Classic"[8] Major differences are:

The blocks editor in the original version ran in a separate Java process, using the Open Blocks Java library for creating visual blocks programming languages and programming

App Inventor Classic Blocks Editor

Open Blocks is distributed by the Massachusetts Institute of Technology's Scheller Teacher Education Program (STEP) and is derived from master's thesis research by Ricarose Roque. Professor Eric Klopfer and Daniel Wendel of the Scheller Program supported the distribution of Open Blocks under an MIT License.[2] Open Blocks visual programming is closely related to StarLogo TNG, a project of STEP, and Scratch, a project of the MIT Media Lab's Lifelong Kindergarten Group. App Inventor 2[8] replaced Open Blocks with Blockly, a blocks editor that runs within the browser.

The MIT AI2 Companion app enables real-time debugging on connected devices via Wi-Fi, not just USB.

As of May 2014, there were 87,000 weekly active users of the service and 1.9 million registered users in 195 countries for a total of 4.7 million apps built.[citation needed]

As December 2015, there were 140,000 weekly active users and 4 million registered users in 195 countries, run total of 12 million built application

