Login with user postgres

CREATE DATABASE gps\_tracking\_db

ENCODING = 'UTF8'

TEMPLATE = template0

LC\_COLLATE = 'C'

LC\_CTYPE = 'C';

Type

\list

It will list all the databases in your PostgreSQL

Now connect to your newly created database for that type

\connect gps\_tracking\_db

A successful connection will be printed on the screen as

You are now connected to ….

Create a new table

CREATE SCHEMA main;  
COMMENT ON SCHEMA main IS 'Schema that stores all the GPS tracking core data.';

Create your first relation

CREATE TABLE main.gps\_data(  
gps\_data\_id serial,  
gps\_sensors\_code character varying,  
line\_no integer,  
utc\_date date,  
utc\_time time without time zone,  
lmt\_date date,  
lmt\_time time without time zone,  
ecef\_x integer,  
ecef\_y integer,  
ecef\_z integer,  
latitude double precision,  
longitude double precision,  
height double precision,  
dop double precision,  
nav character varying(2),  
validated character varying(3),  
sats\_used integer,  
main\_vol double precision,  
bu\_vol double precision,  
temp double precision,

humidity double precision,  
easting integer,  
northing integer,  
remarks character varying  
);

COMMENT ON TABLE main.gps\_data  
IS 'Table that stores raw data as they come from the sensors (plus the ID of the sensor).';

Alter the table to set PRIMARY KEY

ALTER TABLE main.gps\_data  
ADD CONSTRAINT gps\_data\_pkey  
PRIMARY KEY(gps\_data\_id);

To store TIME

ALTER TABLE main.gps\_data  
ADD COLUMN insert\_timestamp timestamp with time zone  
DEFAULT now();

To Prevent duplication

ALTER TABLE main.gps\_data  
ADD CONSTRAINT unique\_gps\_data\_record  
UNIQUE(gps\_sensors\_code, line\_no);

Check datestyle by typing

SHOW datestyle;

And set it by

SET SESSION datestyle = "ISO, DMY";

Show timezone

SHOW time zone;

Add timestamping by

ALTER TABLE main.gps\_data  
ADD COLUMN acquisition\_time timestamp with time zone;  
UPDATE main.gps\_data  
SET acquisition\_time = (utc\_date + utc\_time) AT TIME ZONE 'UTC';

Indexing your database table is a good idea to ease out searches in the future

CREATE INDEX acquisition\_time\_index  
ON main.gps\_data  
USING btree (acquisition\_time );

CREATE INDEX gps\_sensors\_code\_index  
ON main.gps\_data  
USING btree (gps\_sensors\_code);

Create timestamp and update automatically using the following functions

CREATE OR REPLACE FUNCTION main.timestamp\_last\_update()  
RETURNS trigger AS  
$BODY$BEGIN  
IF NEW IS DISTINCT FROM OLD THEN  
NEW.update\_timestamp = now();  
END IF;  
RETURN NEW;  
END;$BODY$  
LANGUAGE plpgsql VOLATILE  
COST 100;

COMMENT ON FUNCTION main.timestamp\_last\_update()  
IS 'When a record is updated, the update\_timestamp is set to the current  
time.';

We will also need a trigger to update the timestamp with each new record that is inserted

CREATE TRIGGER update\_timestamp  
BEFORE UPDATE  
ON main.gps\_data  
FOR EACH ROW  
EXECUTE PROCEDURE main.timestamp\_last\_update();

CREATE update\_timestamp COLUMN

ALTER TABLE main.gps\_data

ADD COLUMN update\_timestamp timestamp with time zone DEFAULT now();

Function to update the gps\_data table each time a new record is added

UPDATE main.gps\_data  
SET update\_timestamp = now();  
CREATE OR REPLACE FUNCTION main.acquisition\_time\_update()  
RETURNS trigger AS  
$BODY$BEGIN  
NEW.acquisition\_time = ((NEW.utc\_date + NEW.utc\_time) at time zone 'UTC');  
RETURN NEW;  
END;$BODY$  
LANGUAGE plpgsql VOLATILE  
COST 100;  
COMMENT ON FUNCTION main.acquisition\_time\_update()  
IS 'When a record is inserted, the acquisition\_time is composed from utc\_date and utc\_time.';

A trigger to call the above function

CREATE TRIGGER update\_acquisition\_time  
BEFORE INSERT  
ON main.gps\_data  
FOR EACH ROW  
EXECUTE PROCEDURE main.acquisition\_time\_update();

Make your table spatial

CREATE EXTENSION postgis;  
CREATE EXTENSION postgis\_topology;

Here, you create a simple function to automatically find the UTM zone at defined coordinates:

CREATE OR REPLACE FUNCTION main.srid\_utm(longitude double precision,latitude double precision)  
RETURNS integer AS  
$BODY$  
DECLARE  
srid integer;  
lon float;  
lat float;  
BEGIN  
lat := latitude;  
lon := longitude;  
IF ((lon > 360 or lon < -360) or (lat > 90 or lat < -90)) THEN  
RAISE EXCEPTION 'Longitude and latitude is not in a valid format (-360 to 360; -90 to 90)';  
ELSEIF (longitude < -180)THEN lon := 360 + lon;  
ELSEIF (longitude > 180)THEN lon := 180 - lon;  
END IF;  
IF latitude >= 0 THEN  
srid := 32600 + floor((lon+186)/6);  
ELSE  
srid := 32700 + floor((lon+186)/6);  
END IF;  
RETURN srid;  
END;  
$BODY$  
LANGUAGE plpgsql VOLATILE STRICT  
COST 100;  
COMMENT ON FUNCTION main.srid\_utm(double precision, double precision)  
IS 'Function that returns the SRID code of the UTM zone where a point (in geographic coordinates) is located. For polygons or line, it can be used giving ST\_x(ST\_Centroid(the\_geom)) and ST\_y(ST\_Centroid(the\_geom)) as parameters. This function is typically used be used with ST\_Transform to project elements with no prior knowledge of their position.';

<?php

$lat=$\_POST['latitude'];

$lon=$\_POST['longitude'];

$temp=$\_POST['temperature'];

$hum=$\_POST['humidity'];

$conn = pg\_connect("host=localhost dbname=gps\_tracking\_db user=postgres password=Postgres123");

if (!$conn) {

echo "\_\_An error occurred.\n";

exit;

}

$result = pg\_query($conn, "INSERT INTO main.gps\_data (latitude, longitude,temp,humidity) VALUES ('$lat' ,'$lon', '$temp, '$hum')");

if (!$result) {

echo "An error occurred.\n";

exit;

}

while ($row = pg\_fetch\_row($result)) {

echo "ID: $row[0] Lat: $row[4] Long: $row[5]";

echo "<br />\n";

}

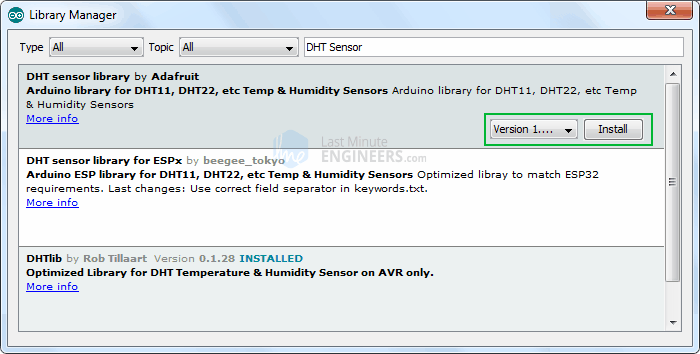
?>

A picture containing diagram

Description automatically generated

Add Library

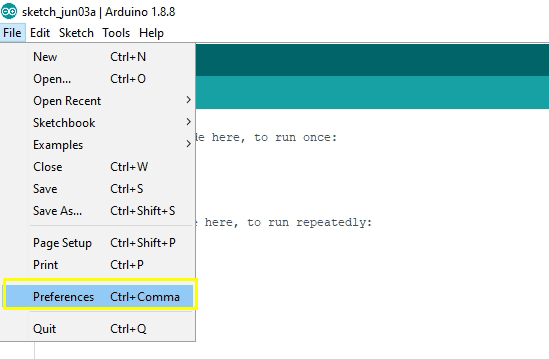
DHT Sensor



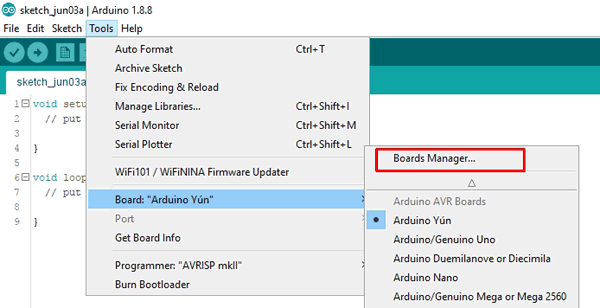
Add Board in the preferences

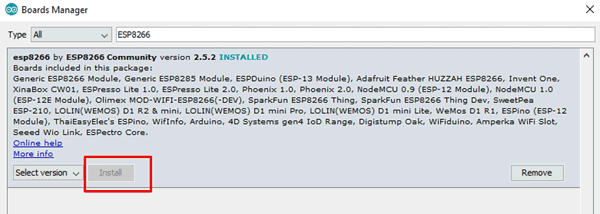
You will need to add additional boards for your computer to pick NodeMCU.

<http://arduino.esp8266.com/stable/package_esp8266com_index.json>

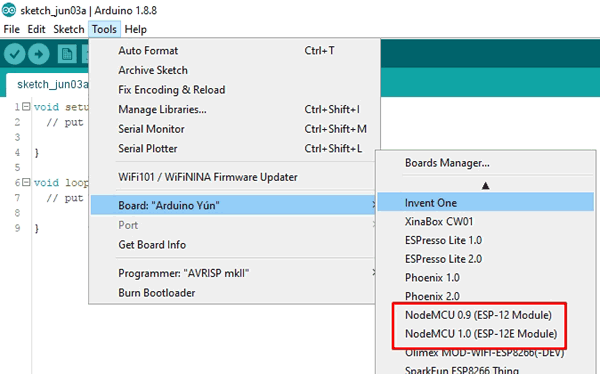


Select board





Select the installed board



Ardunio Code

//Libraries

#include <DHT.h>;

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include <WiFiClient.h>

#include <Wire.h>

#include <Adafruit\_Sensor.h>

//Constants

#define DHTPIN D4 // what pin we're connected to

#define DHTTYPE DHT22 // DHT 22 (AM2302)

DHT dht(DHTPIN, DHTTYPE); //// Initialize DHT sensor for normal 16mhz Arduino

static const int RXPin = 4, TXPin = 5;

static const uint32\_t GPSBaud = 9600;

HTTPClient http;

// The TinyGPS++ object

TinyGPSPlus gps;

// The serial connection to the GPS device

SoftwareSerial ss(RXPin, TXPin);

// REPLACE with your Domain name and URL path or IP address with path

const char\* serverName = "http://192.168.0.168/sensor\_connect.php";

void setup()

{

Serial.begin(9600);

dht.begin();

ss.begin(GPSBaud);

}

void loop()

{

while (ss.available() > 0) {

gps.encode(ss.read());

}

// Your Domain name with URL path or IP address with path

http.begin(serverName);

// Specify content-type header

http.addHeader("Content-Type", "application/x-www-form-urlencoded");

// Prepare your HTTP POST request data

String httpRequestData = "value1=" + String(gps.location.lat(), 6)

+ "&value2=" + String(gps.location.lng(), 6) +

"&value3=" + String(dht.readTemperature()) +

"&value4=" + String(dht.readHumidity()) + "";

Serial.print("httpRequestData: ");

Serial.println(httpRequestData);

// Send HTTP POST request

int httpResponseCode = http.POST(httpRequestData);

if (httpResponseCode > 0) {

Serial.print("HTTP Response code: ");

Serial.println(httpResponseCode);

}

else {

Serial.print("Error code: ");

Serial.println(httpResponseCode);

}

// Free resources

http.end();

//Send an HTTP POST request every 0.5 seconds

delay(499);

}

-Code END-

IF you don’t have geom column in the gps\_tracking\_db or it shows NULL,

Delete the index, column and create again with following steps.

Add geometry column to the table for QGIS to understand the data location

ALTER TABLE main.gps\_data  
ADD COLUMN geom geometry(Point,4326);

Add the geometry column to index

CREATE INDEX gps\_data\_geom\_gist  
ON main.gps\_data  
USING gist (geom );

UPDATE  
main.gps\_data

SET  
geom = ST\_SetSRID(ST\_MakePoint(longitude, latitude),4326)  
WHERE  
latitude IS NOT NULL AND longitude IS NOT NULL;