Ohjeet kelluvan IoT-lämpötilamittarin rakentamiseksi

Tekijät: Mikko Auriola
Juuso Kakriainen
Eetu Laitinen
Mirko Mikkonen
Elias Tepsa

SISÄLLYS

1. Yleistä asiaa projektista

Tämä ohje on tarkoitettu Arduino-pohjaisen vesilämpömittarin rakentamiseen. Laite lähettää anturidataa GSM:n avulla ThingSpeak-pilvipalveluun. Kustannusarvio tälle rakennusprojektille on 150-200e välillä, riippuen siitä kuinka paljon komponenteista ja tarvikkeista täytyy ostaa ja kuinka paljon löytyy jo itseltä.

Kohderyhmä: Yli 15-vuotiaat, asiasta kiinnostuneet esim.

- Ammattikoulu
- Lukio
- Ammattikorkeakoulu
- Yhdistykset ja muu järjestäytynyt toiminta
- Harrastajat

2. Tarvikkeet

Arduino Uno	30€
Arduino GSM Shield 2 (M10 gsm piirillä)	60€
Adafruit Ultimate GPS Logger Shield	40€
DFRobot Waterproof DS18B20 Digital Temperature Senso	7€
DFRobot gravity Terminal Sensor Adapter	3€
Battery bank	10 - 50€

GPS:n, lämpötila sensorin (ds18b20) ja Sensor-adapterin voi tarvittaessa korvata muilla samankaltaisilla tuotteilla. Asiasta lisää ohjeen lopussa.

Viemäriputki muhvilla (110x1000, PVC)	15€
Viemärin tulppa (110)	7€
Viemärin tulppa (tiivisteuralla)	7,5€
Tiiviste	2,5€
Keulalenkki	5€

Lisäksi tarvitaan Sikaflex-massaa tiivistämiseen ja ruuveja tulpan kiinnittämiseen.

3. Käytettävät sovellukset ja Arduino-kirjastot

Tässä työssä käytetään:

- Arduino IDEä, jonka löytää https://www.arduino.cc/en/Main/Software
- Käytettävän koodikirjastot (libraryt)
    - TinyGPSplus
    - AltSoftSerial
    - OneWire
    - DallasTemperature
- Koodiluonnokset, joita käytetään tässä ohjeessa

Koodiluonnokset ja koodikirjastot löytyvät sivulta yhdessä paketissa:
https://drive.google.com/file/d/1K1QYyB_keqKsQCWcjTNivuXB5nGkOgdG/view?usp=sharing

(Jos paketin linkki ei toimi, niin koodi löytyy tämän ohjeen alaosasta tekstinä. Koodikirjastot tulee hakea netistä.)

4. Arduino-laitteiston testaaminen ja kokoaminen

Lämpötilasensori

Lämpötilasensorin toiminnan voi testata DFRobotin ohjeilla:
https://www.dfrobot.com/wiki/index.php/Waterproof_DS18B20_Digital_Temperature_Sensor_(SKU:DFR0198) .

GPS

Adafruit Ultimate GPS Shieldiin tulee juottaa pinnit, jotka tulevat GPS:n mukana. Adafruitin sivuilla on ohjeet tähän:

https://learn.adafruit.com/adafruit-ultimate-gps-logger-shield/shield-headers .

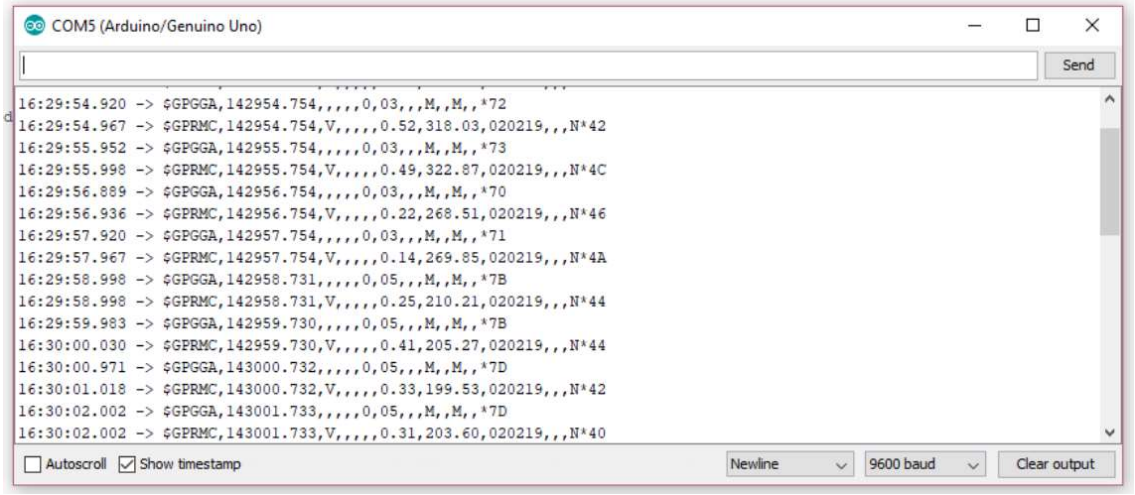
Ohjeet ovat englanniksi, mutta ohjeissa on hyvin havainnollistavia kuvia.

Juottamisen jälkeen GPS on valmis testattavaksi ja siihen toimii hyvin Adafruitin omat ohjeet, jotka löytyvät sivuilta:

https://learn.adafruit.com/adafruit-ultimate-gps-logger-shield/direct-connect .

Testi lyhykäisyydessään on, että tyhjä koodinpätkä ladataan Arduinoon ja GPS:stä valitaan Direct siihen tarkoitetulla kytkimellä. Serial monitorista valitaan 9600 baudia, jolloin GPS alkaa näyttämään koneelle GPS:n NMEA viestejä.

Jos kaikki on mennyt hyvin, tulee Arduino IDE:n Serial monitoriin seuraavanlaisia asioita:



Nämä ovat GPS:n lukemia paikannusviestejä.

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Kuvassa esimerkki tyhjästä sketchistä.

GSM

Koodissamme käytetään pin-kooditonta SIM-korttia, joten pin-koodin kysely tulee ottaa pois SIM-kortilta. SIMin pin-koodin kyselyn voi ottaa pois esim. puhelimella. Nettiohje.fi -sivustolla löytyy muutamia tapoja poistaa SIMiltä koodin kysely pois.

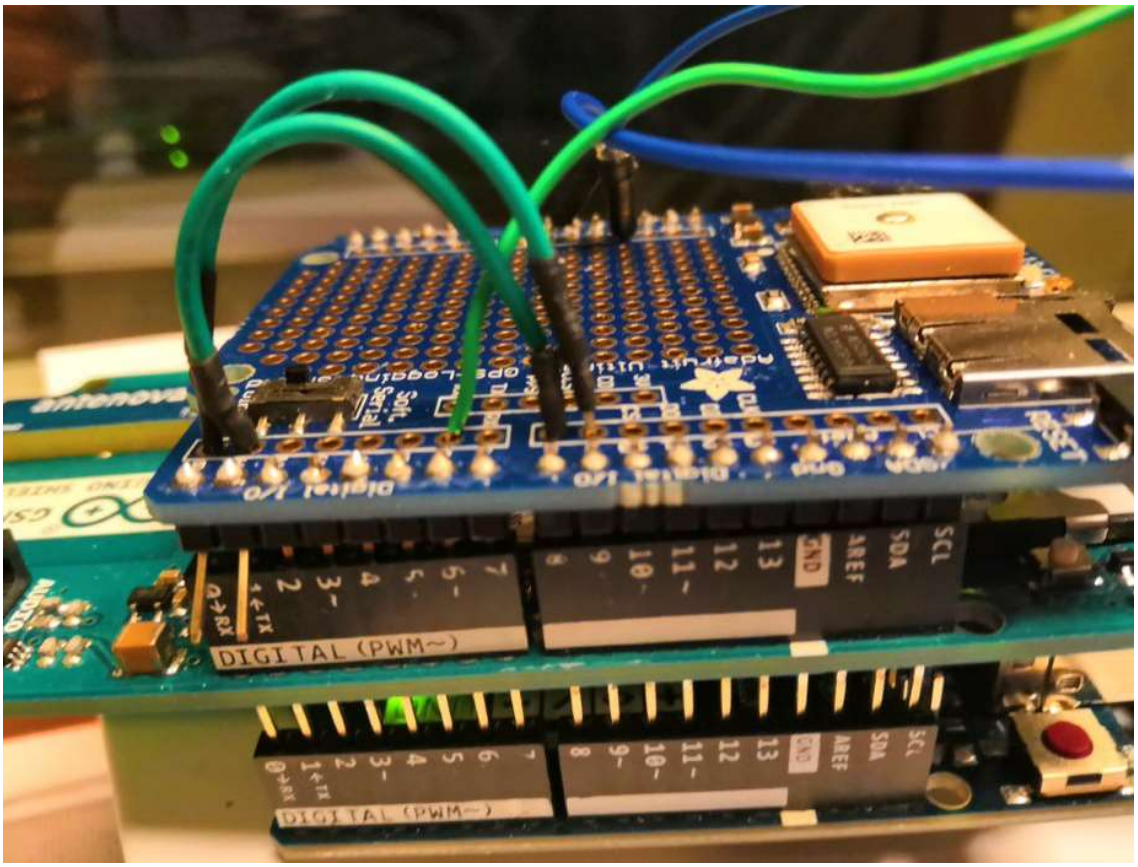
https://nettiohje.fi/ohjeartikkelit/pin-koodin-vaihtaminen/

GSM-laitteen pitäisi toimia, kunhan sinne laittaa sisään SIM-kortin, muita asioita ei tarvitse tehdä. Jos epäilette, ettei GSM toimi, niin testaamiseen voi käyttää Arduino.cc:n omia ohjeita

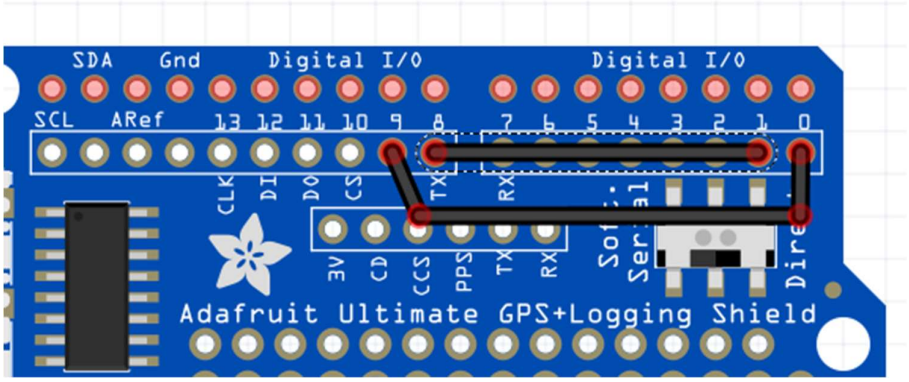
https://www.arduino.cc/en/Guide/ArduinoGSMShield .

Lopullinen laitteisto

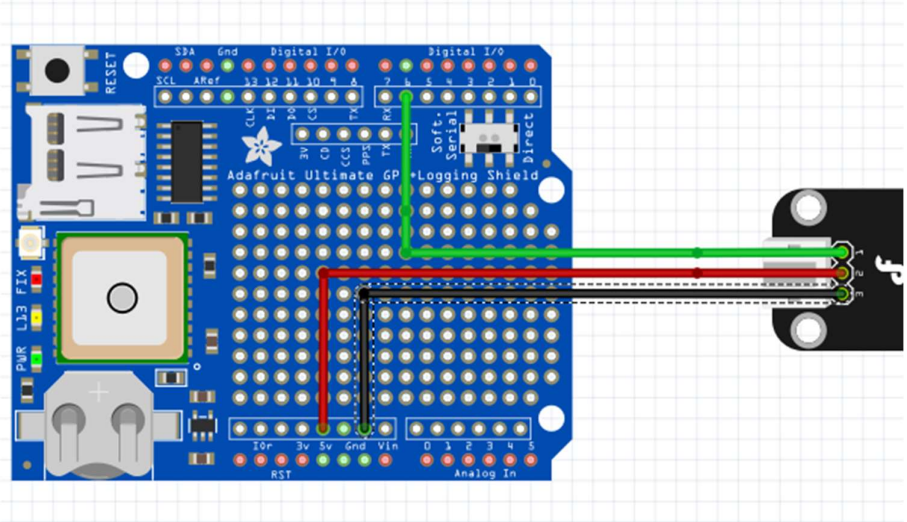
Laitteisto tulee kuvan mallin mukaan:



Alimmaisena on itse Arduino, toisena on M10 GSM-shield ja kolmantena päällä GPS-shield. Koska käytämme lopullisessa koodissa AltSoftSerial-koodikirjastoa, tulee GPS:n 0 ja 1 dataliittimet johdottaa uudestaan pinneille 9 ja 8. (Eli rx0 -> 9 ja tx1 -> 8). Kytkin tulee pitää direct-asennossa ja GPS:n pinnit 0 ja 1 tulee taivuttaa pois paikoiltaan.

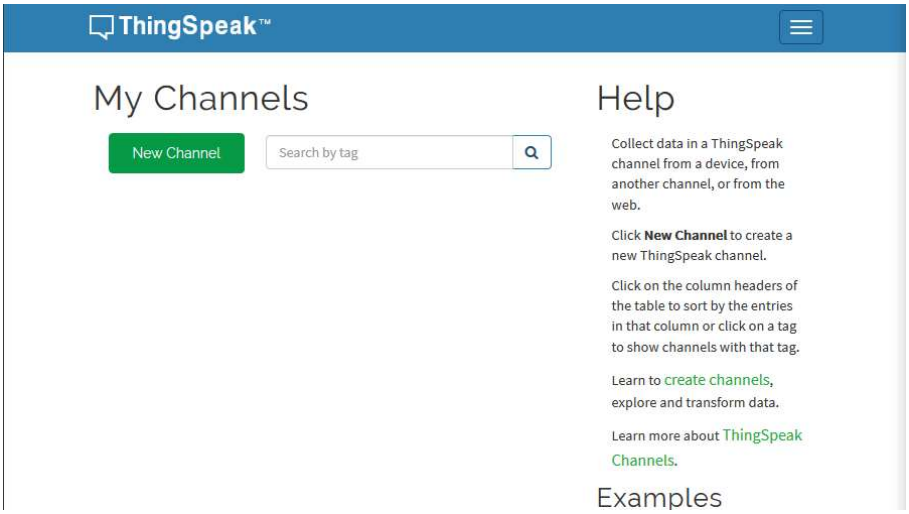


Käytettäessä DFRobotin Terminal sensor -adapteria, niin musta johdin tulee maadoitukseen (GND), punainen johdin 5V jännitteeseen ja vihreä johdin dataliittimeen 6.

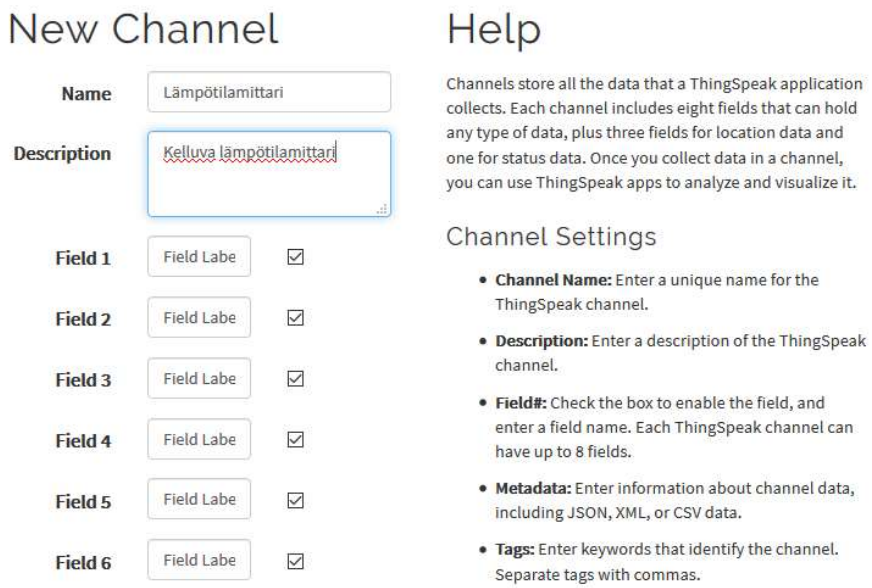


5. ThingSpeakin käytön aloittaminen

ThingSpeakiin tulee tehdä oma käyttäjä (jos on MATLAB-käyttäjä entuudestaan, niin sama käyttäjä käy myös ThingSpeakissa). Käyttäjätunnus luodaan sivulla https://thingspeak.com/users/sign_up . Käyttäjätunnuksen luomisen jälkeen tulisi tulla näkyviin sivusto, jossa voi tehdä oman kanavan projektille.



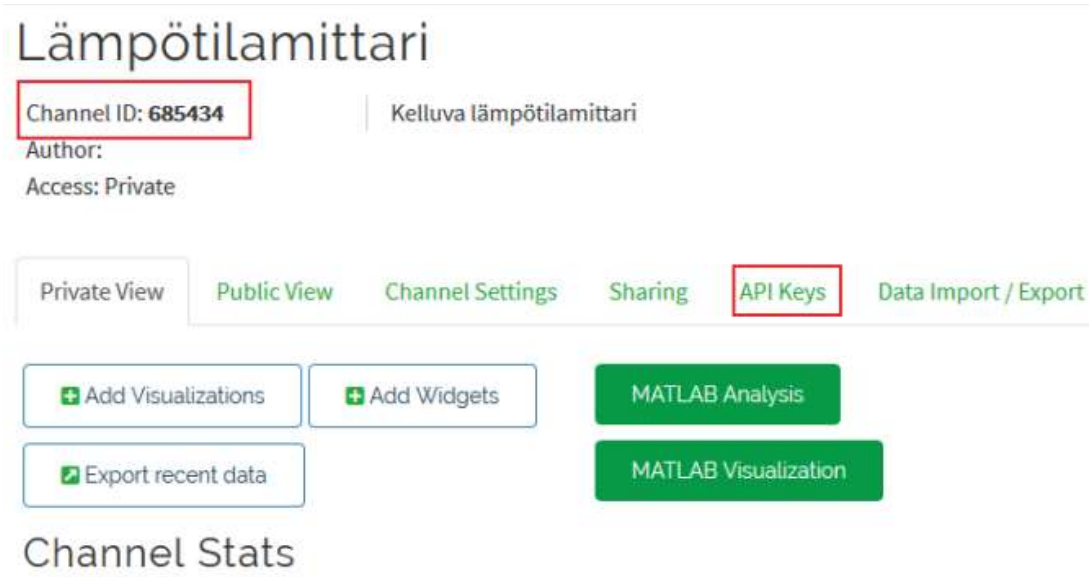
Tästä tulee valita “New Channel”, jolloin tulee seuraavanlainen näkymä:



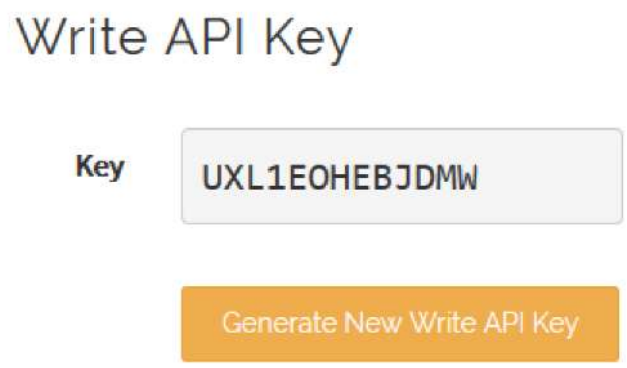
Tähän voi kirjoittaa projektin nimen ja kuvauksen. Fieldejä kannattaa klikata auki, sillä niihin tallentuu tiedot, mitä Arduino lähettää ThingSpeakiin.

Etenkin fieldit 4, 5 ja 6, koska näille fieldeille lähtetään tiedot. Valittuasi nämä fieldit, tulee painaa Save channel -painiketta sivun alaosassa.

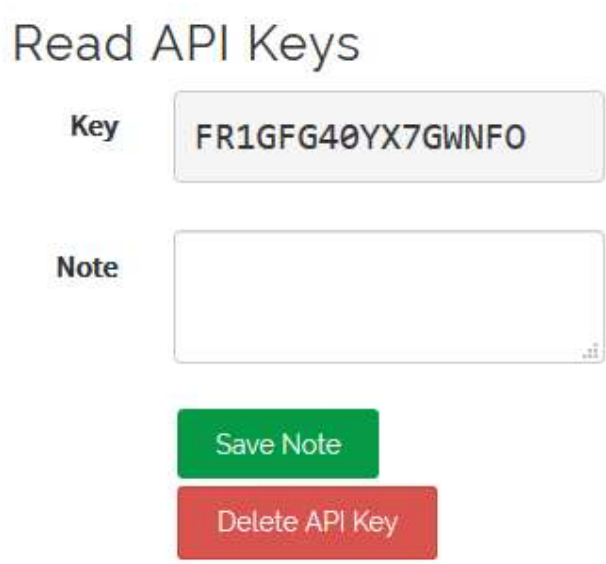
Tästä tulee sivu, josta löytyy anturien antamat mittaustiedot:



Täältä löytää Channel ID:n (kannattaa kirjoittaa muistiin) ja API Keys -välilehdeltä löytyy API-key, jota tarvitaan koodin lähettämiseen.



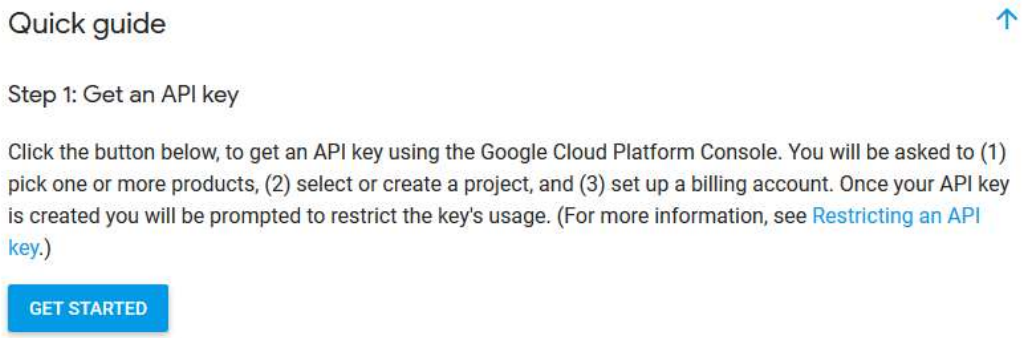
Täältä myös saa Read API -keyn, jolla voi jatkossa lukea tietoja kanavalta.



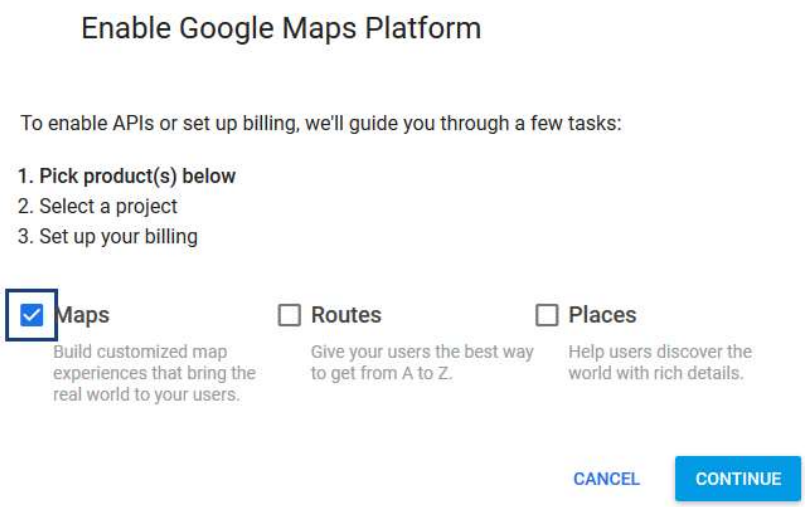
6. Google API -kartan käyttöönotto (tähän osioon tarvitsee Googlen maksutilin)

Jotta voi käyttää Googlen karttapalvelua laitteen paikannukseen, tulee sinulla olla Google tili (esim. Gmail). Sivulta https://developers.google.com/maps/documentation/javascript/get-api-key . Googlelta saa ei-kaupalliseen harrastekäyttöön12 kuukauden kokeilujakson ja 300 € saldoa käyttäjälle.

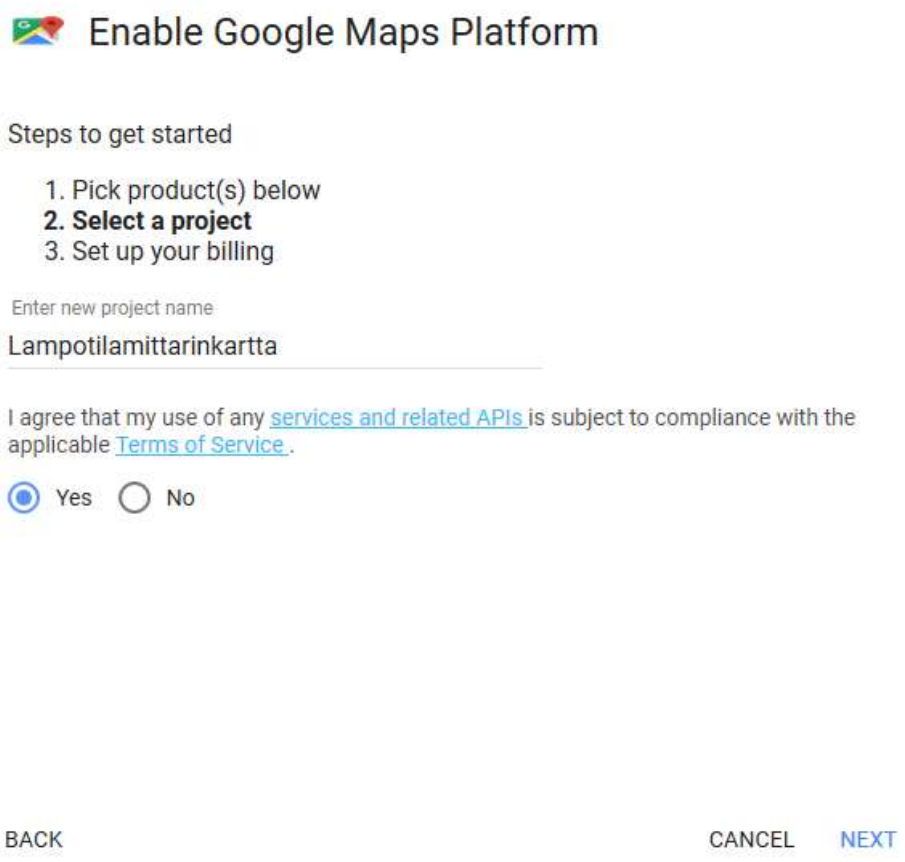
Sivuilta löytyy kohta:



Paina Get Started -nappia, josta avautuu seuraava sivu:



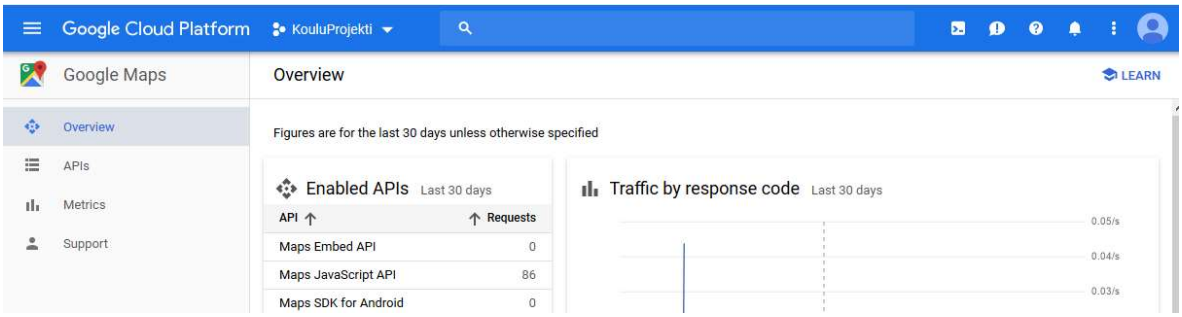
Valitse Maps ja paina Continue.



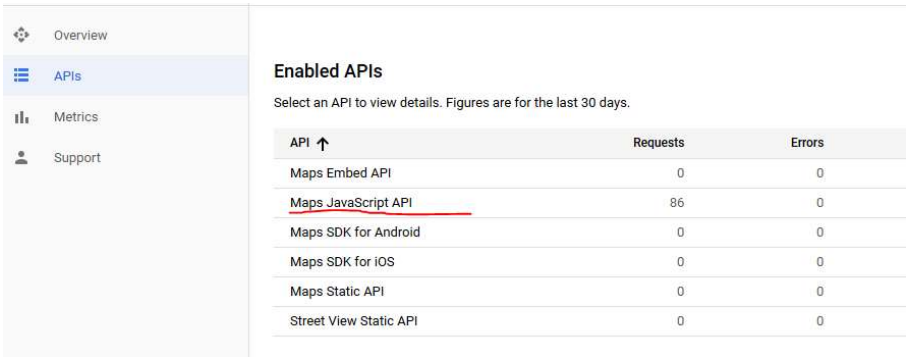
Anna sen jälkeen nimi projektille, hyväksy käyttöehdot ja paina Next.

Tämän jälkeen sivusto pyytää laskutusosoitetta. Täytä siihen omat tietosi.

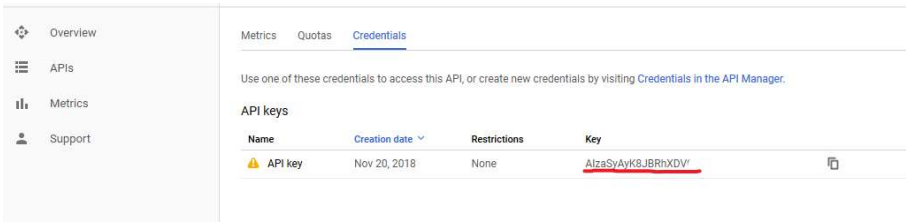
Nyt näkyvillä tulisi olla seuraavanlainen sivu:



Vasemmalta tulee valita APIs ja valikosta Maps JavaScript API.



Maps JavaScript APi –valikosta painetaan Credentials, josta löytyy Google API key.



Tätä API keytä tarvitaan kartan tekoon.

GitHubista ladattavassa paketissa on mukana HTML-koodi nimeltä “kartta”, jossa on 5 kohtaa, jotka tulee korvata seuraavalla tavalla:

(Korvauksen voi tehdä esimerkiksi notepadilla)

1. Laita oma Google API key
2. Laita oma ThingSpeak Read API key
3. Laita oma ThingSpeak Read API key
4. ThingSpeakin Channel ID
5. ThingSpeakin Channel ID

```
    padding: 0;
  }
</style>
<script src="https://maps.googleapis.com/maps/api/js?key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
  var map;
    var x;
    function loadmaps(){
        $.getJSON("https://api.thingspeak.com/channels/XXXXXXX/fields/4/last.json?api_key=XXXXXXXXXXXXXXXX", function(result){

        var m = result;
        x=Number(m.field4);
                    //alert(x);
    });
        $.getJSON("https://api.thingspeak.com/channels/XXXXXXX/fields/5/last.json?api_key=XXXXXXXXXXXXXXXX", function(result){

        var m = result;
        y=Number(m.field5);
```

Nyt kartta on valmis käytettäväksi.

7. Arduino IDE:n käyttö

Käytettävä tiedostopaketti tulee ladata omalle koneelle ja se on ladattavissa alla olevalla sivulta:

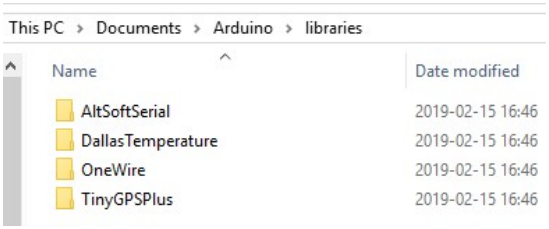
https://drive.google.com/file/d/1K1QYyB_keqKsQCWcjTNivuXB5nGkOgdG/view?usp=sharing .

Tämä pakattu kansio tulee purkaa Arduino-kansioon, joka löytyy:

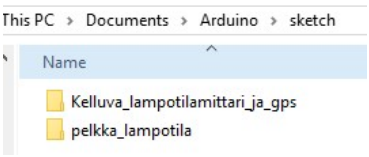
C:\Users**KÄYTTÄJÄ**\Documents\Arduino.

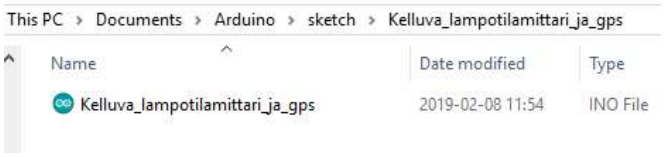
Kansiosta tulee kopioida koodikirjastot AltSoftSerial, DallaTemperature, OneWire ja TinyGPS. Nämä kansiot tulee olla sijoittaa libraries-kansioon:

C:\Users**KÄYTTÄJÄ**\Documents\Arduino\libraries .



Google Drivestä ladatun kansion mukana tulee kansio sketch. Sieltä löytyy koodi, jolla saadaan Arduino lähettämään dataa ThingSpeakiin GSM:n avulla.





Kelluva_lampotilamittari_ja_gps avataan Arduino IDE:llä

Koodissa on kohta, missä tulee vaihtaa oma ThingSpeak Write API key XXXXX tilalle, jotta laite alkaa syöttämään informaatiota oikeaan osoitteeseen. (Tämän API keyn sai ohjeen aiemmasta osasta.)

```
if (step == 12) {
  GPRS.println("AT+QISEND");
  delay(delayT);
  // delay(2000); //More delay time could be require depend on network
}
if (step == 13) {
  String str="GET https://api.thingspeak.com/update?api_key=XXXXXXXXXXXXXXXXXX&field4=" + String(lati,6)+ "&field5=" + String(longi,6) + "&field6=" + String(temp);
  GPRS.println(str);
  delay(delayT);
  //delay(2000);//More delay time could be require depend on network
}
if (step == 14) {
  GPRS.println((char)26);// ASCII code of CTRL+Z
```

Koodin alkuosiossa säädetään erilaisia parametreja. Kohdassa unsigned long wait määritetään, kuinka monesti laite lähettää tiedon ThingSpeakiin. Testivaiheessa kannattaa käyttää lyhyitä aikoja, esimerkiksi 2min (120 000). Kaikki tähän laitettavat ajat tulee laittaa millisekunneissa. Joten 1 tunti on 1*60*60*1000 ms.

```
OneWire oneWire(6);
DallasTemperature DS18B20(&oneWire);
TinyGPSPlus gps;
AltSoftSerial ss;
SoftwareSerial GPRS(2, 3);

unsigned char buffer[64]; // buffer array for data recieve over serial port
int fixed = 0;
int count = 0;   // counter for buffer array
//int led = 13;
int setupStep = 0;
float longi;
float lati;


float temp;
unsigned long time_now = 0;
unsigned long wait = 7200000;   //Odotusväli milloin laite lähettää tieto millisekunneissa 7200000 on 2 tuntia
```

Arduinopiiri tulee liittää USB:lla tietokoneeseen ja Arduino IDE:stä valitaan Arduino piiri ohjelmoitavaksi reittiä:

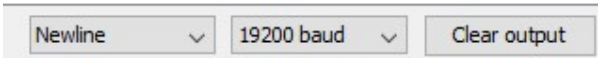
Tools-> Port -> COM (Missä Arduino näkyy)

Koodi ladataan Arduinoon painamalla upload-painiketta:

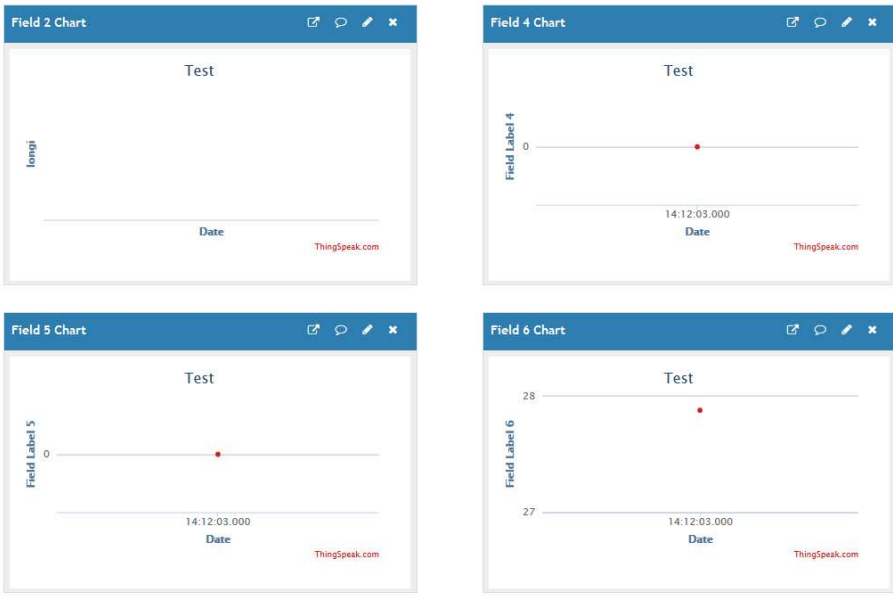
Heti kun koodin lataaminen Arduinolle on päättynyt, alkaa Arduino suorittamaan koodia. Koodin suoritusta voi seurata Arduino IDE:n Serial monitorista, joka löytyy:

Tools-> Serial Monitor

Serial monitorin alaosasta tulee valita 19200 baudia, jotta IDE ymmärtää, mitä tietoa Arduino piiri antaa monitorille.



Mittaustietojen lähettäminen ThingSpeakiin voi viedä muutaman minuutin. Kun lähetys on tapahtunut, niin ThingSpeakissä pitäisi näkyä datapisteet GPS:lle (Field 4 ja 5) ja lämpötilalle(Field 6).



Täältä saadut GPS-koordinaatit voi tarkistaa manuaalisesti esimerkiksi Google Mapsin kautta tai muusta karttapalvelusta. Jos olet tehnyt Google APIn kartan, sen avaamalla pitäisi tulla näkyviin kartta, jossa on pisteenä lämpötilamittarin sijainti.

8. Laitteen suojakuori

Arduinon osat ovat herkkiä, elektroniikkakomponentteja ja ne eivät kestä kosteutta tai iskuja. Tämä tarkoittaa sitä, että komponenteille on tarvittaessa rakennettava suojakuori. Vedenmittausaseman ollessa kyseessä, täytyy laitteen kestää sekä kosteutta ja iskuja. Suojakuoren materiaalina voidaan käyttää tavallisia kotitalouksista ja rautakaupasta löytyviä tarvikkeita.

1. Suunnittelu
2. Hankinta
3. Kasaaminen
4. Testaus

1. **Suunnittelu**

Ensimmäiseksi on selvitettävä, millaisia tarpeita ja vaatimuksia suojakuorelle on. Näitä ovat esimerkiksi: laitteiston fyysiset mitat, suojakuoren muoto, käyttökohde. Tässä ohjeessa tarkoituksena on tehdä itsestään kelluva suojakuori laitteelle. Suojakoteloon joudutaan työssä poraamaan reikiä, joten vuotojen ehkäisemiseksi kaikki reiät on massattava siihen soveltuvalla tiivistysmassalla.

2. **Hankinta**

Suojakuoren rungoksi soveltuu 110mm PVC-viemäriputki mainiosti. Rautakaupoissa on valmiita 110x1000mm muhvillisia viemäriputkia. Lisäksi tarvitaan tulpat molempiin päihin. Viemäriputken

muhvillisessa päässä on kumitiiviste valmiiksi asennettuna. Muhvipäähän laitetaan 110mm tulppa, jossa ei ole uria tiivisterenkaille. Muhvittomaan päähän laitetaan tulppa, jossa on urat 1-2 tiivisterenkaalle.

Jos putki halutaan kiinnittää esimerkiksi ankkuriin, on siihen laitettava kiinnityslenkki. Kiinnityslenkiksi soveltuu ainakin veneen keulalenkki. Tiivistemassana voidaan käyttää SikaFlex-tiivistysmassaa.

3. **Kasaaminen**

Ensimmäiseksi täytyy tehdä lämpöanturille reikä sekä kiinnityslenkille reiät pohjatulppaan tai muuhun parhaaksi näkemäänsä paikkaan. Lämpöanturille porattava reikä tulee olla mahdollisimman lähelle samankokoinen kuin johdon halkaisija. Halkaisija voidaan mitata työntömitalla. Mittaustuloksen perusteella valitaan sopiva poranterä. Kun lämpötila-anturin johto on paikallaan, massataan läpivienti molemmin puolin.

Tämän jälkeen asennetaan kiinnityslenkki paikalleen tulpan pohjaan. On tärkeää käyttää aluslaattaa (“prikkaa”) kiinnityslenkkiä kiinnittäessä. Näin saadaan kuormitus jakautumaan laajemmalle alueelle, eikä muovi hajoa mahdollisista lenkkiin kohdistuvista voimista johtuen. Kun kaikki mutterit on kiristetty, saumataan kaikki saumat SikaFlex-massalla vedenpitävyyden varmistamiseksi.

Seuraavaksi asennetaan kiinnityslenkillinen pohjatulppa paikalleen viemäriputken muhvilliseen päähän. Vaikka muhvillinen liitos on tiivis ja tiukka, on tulppa kiinnitettävä ruuveilla runkoputkeen kiinni. Ennen kuin ruuvit voidaan ruuvata kiinni, täytyy niille porata apureiät, jotta muovi ei halkea ruuvien kiristysvaiheessa. Ruuveina voidaan käyttää itseporautuvia ruuveja. Ruuvien kantoihin ja saumakohtiin laitetaan tiivistemassaa. Kun tulppaa työnnetään paikalleen, voidaan väliin pursottaa saumausmassaa. Tulpan ollessa paikallaan, jää tulpan ja muhviosan väliin rako, joka täytyy myös massata. Lopuksi tasoitetaan ylimääräinen massa esimerkiksi sormella, jotta saadaan siistimmän näköinen lopputulos.

Putken muhvittomaan päähän, tässä tapauksessa putken yläpäähän tulee tiivisterenkaallinen tulppa. Laitteiston ollessa akkutoiminen, täytyy putken “sisuskaluihin” olla pääsy eli liimamainen saumausaine ei sovellu käytettäväksi. Silikonimassa sopii tähän käyttöön paremmin, kuin SikaFlex.

4. **Testaus**

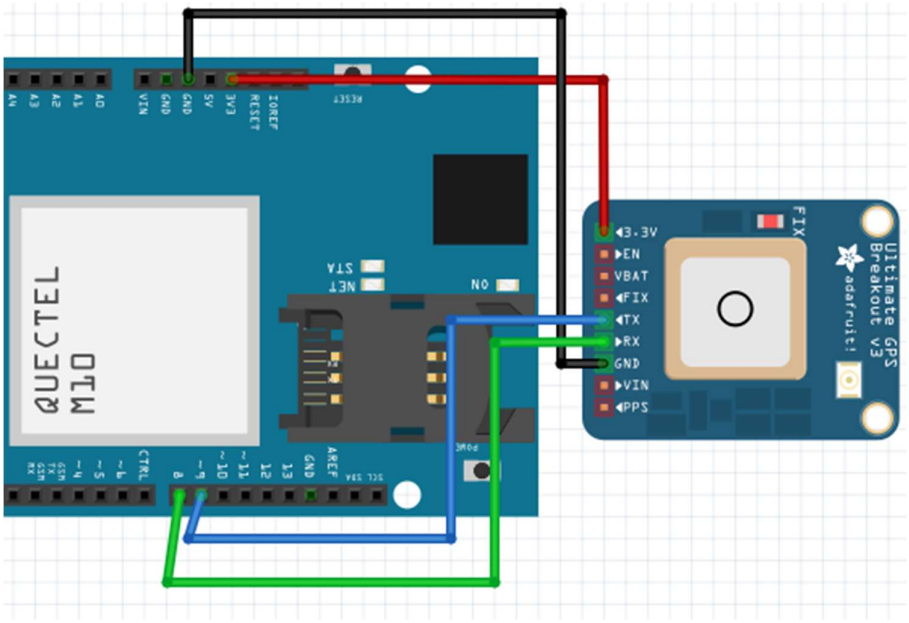
Kun kaikki mahdolliset vuotopaikat on massattu umpeen, voidaan aloittaa vesitiiveyden testaaminen hallituissa oloissa.

Jos halutaan, että putki kelluu vesistössä pystyasennossa, on putken alaosaa lisättävä niin paljon painoa, että putki pysyy aallokossakin pystyssä. Tämän voi toteuttaa sijoittamalla putken sisäpuolelle alaosaa esimerkiksi kiviä tai muuta raskasta materiaalia. Vaihtoehtoinen toteutustapa on pistää kiinnityslenkkiin jotain painavaa roikkumaan.

Putki voidaan jättää kellumaan veteen vedenpitävyyden varmistamiseksi

9. Muuta projektista

Tässä projektissa voi myös käyttää muita GPS-laitteita kuin Adafruit Ultimate GPS loggeria. Korvaavan GPS:n tulee vain antaa standardinmukaisia NMEA-viestejä, joita voi lukea TinyGPS-koodikirjastolla. Johdotukset tulee tehdä niin, että GPS syöttää tietoa tx9-pinniin ja kuuntelee rx8-pinniä. Tämä sen takia, koska käytämme GPS-tiedon lukemiseen AltSoftSerial-koodikirjastoa, joka toimii vain Arduinon pinneillä 8 ja 9.



Lämpötilasensorin myös voi korvata samantapaisella digitaalisella anturilla. DFRobotin Thermal Sensor Adapterin voi korvata pelkällä yhdellä 4700 ohmin vastuksella. Esimerkiksi näissä ohjeissa on tehty yksinkertainen lämpötilamittari:

https://randomnerdtutorials.com/guide-for-ds18b20-temperature-sensor-with-arduino/ .

GSM-shieldiä projektissa ei todennäköisesti voi korvata, sillä koodissa käytetään AT-viestejä, jotka voivat olla GSM-piirikohtaisia. AT-viestien selitykset löytyvät Quectel M10:n ohjeista:

https://www.quectel.com/UploadFile/Product/Quectel_M10_AT_Commands_Manual_V4.0.pdf .

Kiitos Ali Shwaiheenille, hän on alkuperäisen koodin tekijä, jonka pohjalta lähdimme työstämään parannettua versiota laiteprojektiimme.

Alkuperäiset ohjeet

https://www.instructables.com/id/A-Journey-With-Arduino-GSM-Shield-SMS-Calls-TCP-We/

Koodit tekstinä

Kelluvan lämpötilamittarin koodi tekstinä:

```
/* GSM Shield Version 2 GPS Real time Tracking
*  CAn display GPS location On Google Maps using the JavaScrpts
By: Ali Shwaiheen April 2018*

/*
* Muokannut Savonian Ympräsitötekniikan opiskelijat
* Projekti3: Mikko Auriola
*
*/


/*****************************************************************
*  AltSoftSerial always uses these pins for Serial communication:  *
*                                                                  *
* Board          Transmit  Receive   PWM Unusable                 *
* -----          --------  -------   ------------                 *
* Teensy 3.0 & 3.1  21       20        22                         *
* Teensy 2.0        9        10        (none)                     *
* Teensy++ 2.0      25       4         26, 27                     *
* Arduino Uno       9        8         10                         *
* Arduino Leonardo  5        13        (none)                     *
```

```
* Arduino Mega     46       48      44, 45                *
* Wiring-S         5        6       4                     *
* Sanguino         13       14      12                    *
*************************************************************************/

#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include <AltSoftSerial.h>
#include <OneWire.h>
#include <DallasTemperature.h>

OneWire oneWire(6);
DallasTemperature DS18B20(&oneWire);
TinyGPSPlus gps;
AltSoftSerial ss;
SoftwareSerial GPRS(2, 3);

unsigned char buffer[64]; // buffer array for data recieve over serial port
int fixed = 0;
int count = 0;   // counter for buffer array
//int led = 13;
int setupStep = 0;
float longi;
float lati;


float temp;
unsigned long time_now = 0;
unsigned long wait = 7200000;   //Odotusväli milloin laite lähettää tieto millisekunneissa 7200000 on 2 tuntia
```

```
void setup()
{

 ss.begin(9600); //Start GPS
 GPRS.begin(9600);            // the GPRS baud rate
 DS18B20.begin();
 Serial.begin(19200);          // the Serial port of Arduino baud rate.
 Serial.println("Setup OK, Ootellaan gps");
 delay(250000);              //delay gps
}

void loop() {
time_now = millis();
  PowerGSM();
   Serial.println("GPRS päälle");
  delay(30000);                 //Delay että gprs ehtii kunnolla avata itsensä

  Serial.println("Lati ja longi jos asiat ok");
  GPSCord();

  longi = gps.location.lng();
  lati = gps.location.lat();
  DS18B20.requestTemperatures();
  temp = DS18B20.getTempCByIndex(0);
  Serial.println("temp:" + String(temp));
  Send_pubnub();
  Serial.println("Send pubnub ready");
  delay(100);
```

```
  setupStep = 0;
  Serial.println("Loop Ready");
  Serial.println("Delay Started");
  PowerGSM();
  Serial.println("GPRS pois päältä");
  while(millis() < time_now + wait){
  //wait approx. [period] ms
 }
}

void PowerGSM()
{
 Serial.println("Power button press");
 pinMode(7,OUTPUT);
 digitalWrite(7,LOW);
 delay(1000);
 digitalWrite(7,HIGH);
 delay(2000);
 digitalWrite(7,LOW);
 delay(3000);
}


void clearBufferArray()          // function to clear buffer array
{
 for (int i = 0; i < count; i++)
 {
   buffer[i] = NULL; // clear all index of array with command NULL
 }
}
```

```
void GPSCord(){
 while (ss.available() > 0)
 gps.encode(ss.read());

 if (gps.location.isUpdated())
 {
  fixed = 1;
 Serial.print("LAT="); Serial.print(gps.location.lat(), 6);
 Serial.print(" LNG="); Serial.println(gps.location.lng(), 6);
 }


}

void Send_pubnub() {

 if (GPRS.available())            // if date is comming from softwareserial port ==> data is comming from gprs shield
 {
  while (GPRS.available())       // reading data into char array
  {
   buffer[count++] = GPRS.read();   // writing data into array
   if (count == 64)break;
  }
  Serial.write(buffer, count);         // if no data transmission ends, write buffer to hardware serial port
  clearBufferArray();            // call clearBufferArray function to clear the storaged data from the array
  count = 0;                     // set counter of while loop to zero
 }
```

```
 if (setupStep < 15) {
   ConnectToInternet(setupStep);
   delay(1000);

   return Send_pubnub();
 }
}


void ConnectToInternet(int step) {

 int delayT = 1000;

 if (step == 0) {
   GPRS.println("AT");
   delay(delayT);
 }

 if (step == 1) {
   GPRS.println("AT+CPIN?");
   delay(delayT);
 }

 if (step == 2) {
   GPRS.println("AT+CREG?");
   delay(250);
 }

 if (step == 3) {
   GPRS.println("AT+CGATT?");
```

```
  delay(delayT);
}

if (step == 4) {
  GPRS.println("AT+QIDEACT");
  delay(delayT);
}

if (step == 5) {
  GPRS.println("AT+QISTAT=?");
  delay(250);
}

if (step == 6) {
  GPRS.println("AT+QIMUX=0");
  delay(250);
}

if (step == 7) {
  GPRS.println("AT+QIREGAPP=\"zain\",\"\",\"\"");
  delay(delayT);
}

if (step == 8) {
  GPRS.println("AT+QIACT");
  delay(delayT);
}

if (step == 9) {
  GPRS.println("AT+QILOCIP");//get local IP adress
```

```
  delay(250);
 }
 if (step == 10) {
  GPRS.println("AT+QIPROMPT=0"); //No prompt ">" and show "SEND OK" when sending successes
  delay(delayT);
  delay(250);
 }

 if (step == 11) {
  GPRS.println("AT+QIOPEN=\"TCP\",\"184.106.153.149\",\"80\"");
  delay(delayT);
  delay(2000);
 }

 if (step == 12) {
  GPRS.println("AT+QISEND");
  delay(delayT);
  // delay(2000); //More delay time could be require depend on network
 }
 if (step == 13) {
   String str="GET https://api.thingspeak.com/update?api_key=X66AIY02M2TC7L8K&field4=" +
String(lati,6)+ "&field5=" + String(longi,6) + "&field6=" + String(temp);  // after api_key= put your
own apikey
   GPRS.println(str);
   delay(delayT);
   //delay(2000);//More delay time could be require depend on network
 }
 if (step == 14) {
   GPRS.println((char)26);// ASCII code of CTRL+Z
   GPRS.println();
   delay(delayT);
```

```
  //delay(2000);//More delay time could be require depend on network
 }
 setupStep++;
}
```