UNIVERSIDAD DEL PAÍS VASCO

INSTITUTO DE MÁQUINA HERRAMIENTA



MÁSTER DIGITAL MANUFACTURING:

SOLUCIONES INTEGRALES DE INGENIERÍA

**MONITORIZACIÓN DE PARÁMETROS EN EQUIPOS DE BOMBEO EN BENÍN**

Jon Duo

Behrooz Vakili

Jhonny Ricci Santos

Febrero de 2019

Profesor: Ivan Arakistain

# Solución Tecnológica

Los equipos de bombeo de agua permiten extraer agua del subsuelo y se pueden alimentar con energía renovable en zonas en las que la infraestructura existente es escasa y los recursos limitados, como es el caso de África.

Este proyecto propone el desarrollo de una solución tecnológica para facilitar la gestión remota de parámetros importantes de la maquinaria de bombeo para llevar un mantenimiento predictivo. El proyecto se centra en la extracción de agua de dos pozos ubicados en Benín, uno en [Tamarou](https://www.google.com/maps/place/9%C2%B044'22.4%22N+2%C2%B041'46.2%22E/@9.7398264,2.6947395,13.04z/data=!4m5!3m4!1s0x0:0x0!8m2!3d9.739565!4d2.696157) y otro en [Sansi Gando](https://www.google.com/maps/place/9%C2%B056'19.8%22N+3%C2%B012'54.6%22E/@9.9394802,3.213338,16.74z/data=!4m5!3m4!1s0x0:0x0!8m2!3d9.938838!4d3.215174).

Para la construcción de la estación de bombeo se seleccionó una bomba sumergible de tornillo para cada pozo. Los caudales promedio y profundidades de los pozos son de 2,0 con 60 metros de profundidad y 4,4 con 81 metros de profundidad respectivamente, por lo que se seleccionan bombas para agua limpia sumergibles de 21,6 , quedando muy por arriba de los parámetros de trabajo para asegurarnos el funcionamiento correcto cuando la bomba no esté al 100% por posibles insuficiencias energéticas. Para futuras instalaciones de estaciones de bombeo, el coste del equipo es de 200€, en la imagen 1 se muestra el equipo y redirecciona a la página web para su compra.



Imagen 1 Bomba sumergible Pedrollo

El desarrollo del proyecto se basará en los siguientes apartados tecnológicos:

## **Sensores Wireless IOT**

Se emplearán sensores, un [Arduino MEGA](https://es.rs-online.com/web/p/products/7154084/?grossPrice=Y&cm_mmc=ES-PLA-DS3A-_-google-_-PLA_ES_ES_CatchAll-_-Ad+Group+Catch+All-_-PRODUCT_GROUP&matchtype=&aud-343510635252:pla-293946777986&gclid=Cj0KCQiAzKnjBRDPARIsAKxfTRBjmvjuFdIG5Qn2xAW_n7oOIL1DzbujtfhCgGQK3YrBE3P9cedaVOYaAhhiEALw_wcB&gclsrc=aw.ds) (microcontrolador ATmega2560) y un nodo [TTGO LoRaWAN](https://tienda.bricogeek.com/arduino-compatibles/1122-ttgo-lora32-esp32-con-oled-900-mhz.html) para la transmisión de datos por la especificación para [redes](https://es.wikipedia.org/wiki/Red_de_telecomunicaci%C3%B3n) de baja potencia y área amplia. Lora​, es un módulo de comunicación por radiofrecuencia diseñada específicamente para dispositivos de bajo consumo de alimentación, que operan en redes de alcance local, regional, nacionales o globales.

El estándar de red LoRaWAN apunta a requerimientos característicos de [Internet de las Cosas](https://es.wikipedia.org/wiki/Internet_de_las_Cosas), tales como conexiones bidireccionales seguras, bajo consumo de energía, largo alcance de comunicación, bajas velocidades de datos, baja frecuencia de transmisión, movilidad y servicios de localización. Se selecciona esta especificación para redes de baja potencia porque permite la interconexión entre objetos inteligentes sin la necesidad de instalaciones locales complejas.

Por esta transmisión, la adquisición de datos se puede hacer cada 30 segundos o más, según se precise en cada caso, el microcontrolador transmitirá un valor determinado dentro del intervalo de tiempo. Para un seguimiento remoto y mantenimiento correctivo y predictivo de las estaciones de bombeo se seleccionaron los siguientes parámetros a sensorizar:

### **Aceleración / Vibración**: se instalará un [acelerómetro](https://www.digikey.es/product-detail/es/sparkfun-electronics/SEN-09269/1568-1044-ND/5140809?utm_adgroup=Evaluation+Boards+-+Sensors&mkwid=s&pcrid=278667046290&pkw=&pmt=&pdv=c&productid=5140809&slid=&gclid=CjwKCAiAqaTjBRAdEiwAOdx9xgD1_-WNdM6oFLS97cvXWYsmmII-Gu43uWQ5_y9PB5Ba6qbxlSdNYxoChhkQAvD_BwE) triaxial en el cuerpo del motor de la bomba. Cada 30 segundos o más, según se precise en cada caso, el microcontrolador transmitirá el valor máximo de amplitud en este intervalo de tiempo en cada una de las direcciones “X”, “Y” y “Z”. Esto permitirá hacer un seguimiento de las vibraciones de la bomba en todas las direcciones, gracias a esto, se puede conocer el estado general del equipo en todo momento. Además, da la opción de realizar un análisis frecuencial (FFT) pudiendo conocer al detalle el estado de la bomba y que parte está fallando o tiene tendencia a fallar.

### **Presencia de agua en el pozo**: se instalará un [sensor de humedad](https://www.digikey.es/product-detail/es/sparkfun-electronics/SEN-13322/1568-1360-ND/5764506?utm_adgroup=Evaluation+Boards+-+Sensors&mkwid=s&pcrid=278667046290&pkw=&pmt=&pdv=c&productid=5764506&slid=&gclid=CjwKCAiAqaTjBRAdEiwAOdx9xtyMUXopfDCTFGGYeiFzQuXvtqdahDuL_nvnZOO7HSn6fOTQPouLxBoCGyEQAvD_BwE) del suelo (por facilidad de adquisición) en el pozo del cual se extrae el agua para asegurarnos de que no esté seco y se averíe la bomba. El sensor emite valores de presencia y de no presencia de agua (1 y 0). Cada 30 segundos, el microcontrolador transmitirá un “0” si cualquier valor dentro del intervalo del tiempo es cero, si todos los valores dentro del intervalo de tiempo son “1”, transmitirá “1”.

### **Temperatura del motor**: Se instalará un [sensor de temperatura y humedad DHT](https://www.digikey.es/product-detail/es/seeed-technology-co.,-ltd/101020011/1597-1124-ND/5482602?utm_adgroup=Evaluation+Boards+-+Expansion+Boards%2C+Daughter+Cards&mkwid=s&pcrid=278710408307&pkw=&pmt=&pdv=c&productid=5482602&slid=&gclid=CjwKCAiAqaTjBRAdEiwAOdx9xofjmxrLSN3LJcRnJjiXBDIPQ5fD6UBtBHXB-Aaz7ocfutBttXV0zxoCT4oQAvD_BwE)22 en el motor para predecir posibles fallas por calentamiento del equipo. De la misma manera se podrá analizar estos datos junto a los de vibración para obtener una mejor aproximación de la vida útil restante del motor. Este sensor será instalado en un contenedor hermético con grado de protección IP68. Cada 30 segundos o más, según se precise en cada caso, el microcontrolador transmitirá el valor máximo de temperatura en este intervalo de tiempo.

## **Recogida y gestión de datos**

La recogida de datos se realizará mediante un Gateway LoRa, obteniendo los datos del nodo TTGO LoRa, posteriormente, estos datos se almacenarán en una base de datos SQL.

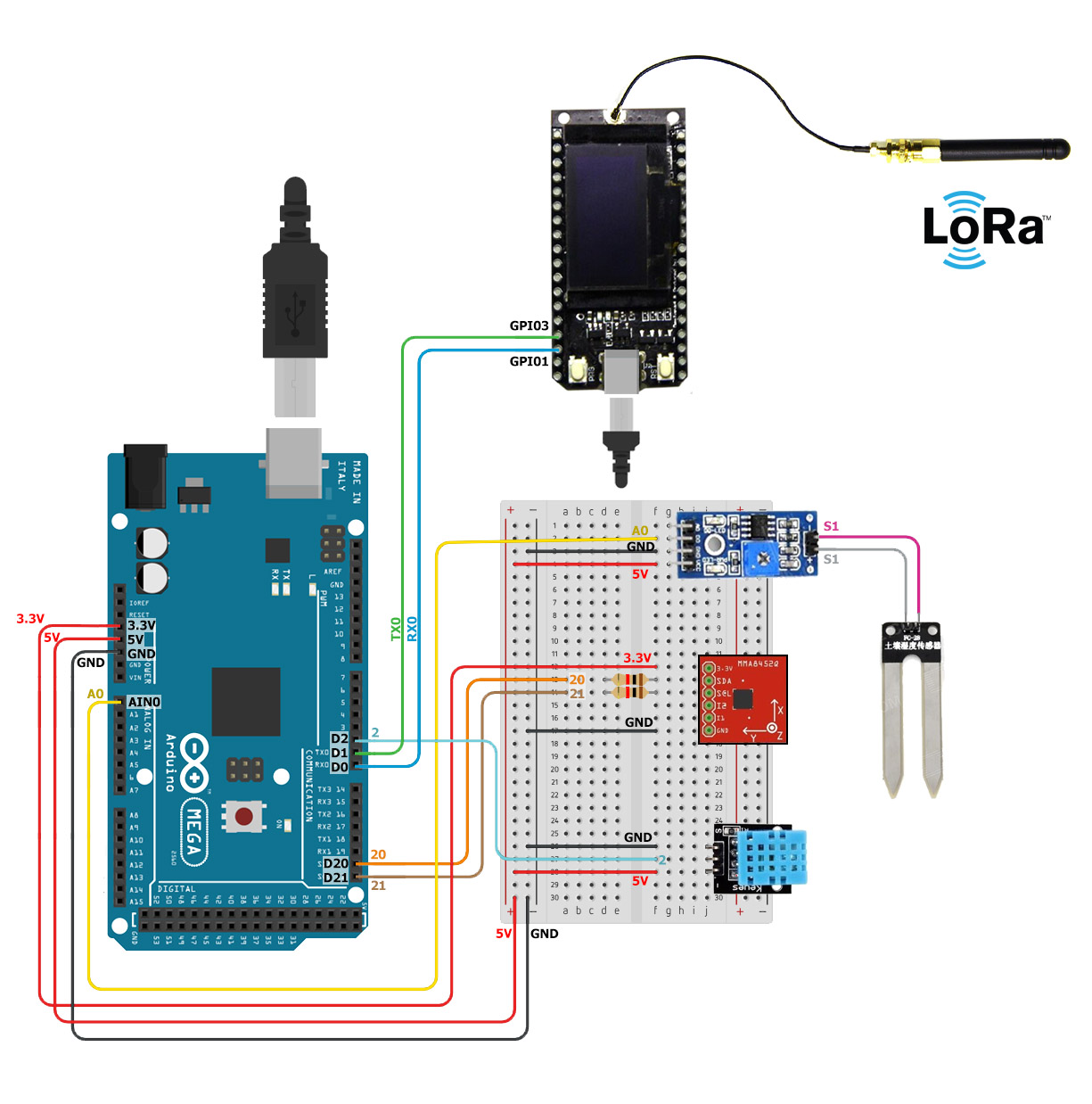


Imagen 2 Esquema de Montaje

A continuación, se muestra el código escrito en el módulo LoRa para que comunique con el Hub:

#include <Arduino.h>

#include "font.h"

#include <Wire.h>

#include "SSD1306.h"

#include <SPI.h>

#include <LoRa.h> //se incluyen las librerías necesarias

#define SX1278\_SCK  5

#define SX1278\_MISO 19

#define SX1278\_MOSI 27

#define SX1278\_CS   18

#define SX1278\_RST  14

#define SX1278\_IRQ  26

#define BAND 868500000.00

#define spreadingFactor 9

#define SignalBandwidth 31.25E3

#define preambleLength 8

#define codingRateDenominator 8 // se definen los valores de configuración de distintas librerias

const byte numChars = 80;

char receivedChars[numChars]; // an array to store the received data

boolean newData = false;

void setup() // programa principal

{

//inicialización del UART

Serial.begin(9600);

  while (!Serial);

// inicialización del LoRa

  SPI.begin(SX1278\_SCK, SX1278\_MISO, SX1278\_MOSI, SX1278\_CS);

  LoRa.setPins(SX1278\_CS, SX1278\_RST, SX1278\_IRQ);

  if(!LoRa.begin(BAND)) {

    Serial.println("Starting LoRa failed!");

    while(1);

  }

  LoRa.setSpreadingFactor(spreadingFactor);

  LoRa.setSignalBandwidth(SignalBandwidth);

  LoRa.setCodingRate4(codingRateDenominator);

  LoRa.setPreambleLength(preambleLength);

  delay(1000);

}

void loop() { //bucle de programa

 recvWithEndMarker();

 showNewData();

}

void recvWithEndMarker() { //integración del marcador final

 static byte ndx = 0;

 char endMarker = '\n';

 char rc;

   if (Serial.available() > 0) {

           while (Serial.available() > 0 && newData == false) {

 rc = Serial.read();

 if (rc != endMarker) {

 receivedChars[ndx] = rc;

 ndx++;

 if (ndx >= numChars) {

 ndx = numChars - 1;

 }

 }

 else {

 receivedChars[ndx] = '\0'; // terminar el string

 ndx = 0;

 newData = true;

 }

 }

}

void showNewData() { // si entran nuevos datos

 if (newData == true) {

  Serial.print(receivedChars);

  LoRa.beginPacket();

  LoRa.print(receivedChars);

  LoRa.endPacket();

 newData = false;

 }

}

Y aquí el código utilizado para extraer los datos de los sensores y enviarlos al módulo LoRa:

#include <DHT.h> // se incluyen las librerías necesarias

#include <Wire.h>

#define MMA8452\_ADDRESS 0x1D // se configuran los valores fijos

#define DHTPIN 2

#define DHTTYPE DHT11

#define OUT\_X\_MSB 0x01

#define XYZ\_DATA\_CFG  0x0E

#define WHO\_AM\_I   0x0D

#define CTRL\_REG1  0x2A

#define GSCALE 2

DHT dht(DHTPIN, DHTTYPE); // se configura el sensor de T y humedad

int sensor\_pin = A0;

int output\_value ;

int agua =0;

  float tablaH[30];

  float tablaT[30];

  float tablaF[30];

  float tablaX[30];

  float tablaY[30];

  float tablaZ[30];

  int n=0;

  float sumaH=0;

  float sumaT=0;

  float defH=0;

  float defT=0;

  float defX=0;

  float defY=0;

  float defZ=0;

void setup() { //función principal

  Serial.begin(9600); //inicialización del UART

  dht.begin(); // inicialización del DHT

  Wire.begin(); // inicialización del I2C

  initMMA8452(); // inicialización del acelerómetro

}

void loop() { //bucle de lecturas (se separa la lectura de cada sensor)

/////////TEMPERATURA Y HUMEDAD////////////

  int h = dht.readHumidity(); //guardar lecturas del sensor en variables

  float t = dht.readTemperature();

  float f = dht.readTemperature(true);

  if (isnan(h) || isnan(t) || isnan(f)) { //si las variables no reciben valor

    Serial.println("Error obteniendo los datos del sensor DHT11");

    return;

  }

  /\*

  Serial.print("Humedad: "); //código de prueba individual para test

  Serial.print(h);

  Serial.print(" % ");

  Serial.print(" \t");

  Serial.print("Temperatura: ");

  Serial.print(t);

  Serial.print(" \*C ");

  Serial.print(" \t");

\*/

  ///////////////////SENSOR AGUA/////////////////////////////////

   output\_value= analogRead(sensor\_pin); // lectura analogica del sensor

   output\_value = map(output\_value,550,0,0,100);

   if (output\_value==100) //si el valor es el máximo (100) es que hay agua

   {

   agua=1;

   }else{

   agua=0; //si es menor que 100, no hay agua

   }

   /\*

   Serial.print("Agua:  "); // código de prueba individual para test

   Serial.print(agua);

   Serial.print(" \t");

   \*/

//////////VIBRACION////////////

  /\*

  Serial.print(" Vibración (x, y, z):  "); // código de prueba individual para test

  \*/

  int accelCount[3];

  readAccelData(accelCount);  // lectura del acelerometro

  float accelG[3];

  for (int i = 0 ; i < 3 ; i++)

  {

    accelG[i] = (float) accelCount[i] / ((1<<12)/(2\*GSCALE));

  }

  for (int i = 0 ; i < 3 ; i++)

  {

    /\*

    Serial.print(accelG[i], 4);  // código de prueba individual para test

    Serial.print(" ");

    \*/

  }

  /\*

  Serial.println(); // código de prueba individual para test

  \*/

  delay(250); // <- el tiempo que queramos entre 30 por 1000

//////////GUARDAMOS EL DATO//////////////////////////

tablaH[n]=h;

tablaT[n]=t;

tablaX[n]=abs(accelG[0]); //valor absoluto

tablaY[n]=abs(accelG[1]);

tablaZ[n]=abs(accelG[2]);

n++;

//////////MEDIAS H/T Y MAXIMO ACELERACIONES///////////

if(n==29) // definimos cuantos valores lee antes de hacer la media

{

  n=0;

  sumaH=0;

  sumaT=0;

for(int i=0; i<=29 ; i++) // vamos guardando los valores de T y H en una tabla

{

  sumaH=sumaH+tablaH[i];

  sumaT=sumaT+tablaT[i];

}

defH=sumaH/29; // sacamos la media de humedad

defT=sumaT/29; // sacamos la media de temperatura

for(int i=0; i<=29 ; i++) // vamos guardando los valores de aceleración x, y, z en una tabla

{

  if(i==0)

  {

  defX=tablaX[i];

  defY=tablaY[i];

  defZ=tablaZ[i];

  }

  if(defX<tablaX[i+1]&&i!=0&&i!=29) //si el valor es mayor al anterior lo guardamos

  {

     defX=tablaX[i+1];

  }

    if(defY<tablaY[i+1]&&i!=0&&i!=29)

  {

     defY=tablaY[i+1];

  }

    if(defZ<tablaZ[i+1]&&i!=0&&i!=29)

  {

     defZ=tablaZ[i+1];

  }

}

/////////MOSTRAR DATOS DEFINITIVOS/////////////

  Serial.print("Humedad: ");

  Serial.print(defH);

  Serial.print(" % ");

  Serial.print(" \t");

  Serial.print("Temperatura: ");

  Serial.print(defT);

  Serial.print(" \*C ");

  Serial.print(" \t");

  Serial.print("Agua:  ");

  Serial.print(agua);

  Serial.print(" \t");

  Serial.print("X, Y, Z: ");

  Serial.print(defX);

  Serial.print(" \t");

  Serial.print(defY);

  Serial.print(" \t");

  Serial.print(defZ);

  Serial.print(" \t");

  Serial.print(" \n");

}

  }

/////////////////// FUNCIONES INTERNAS DE LIBRERIAS /////////////////////////

void readAccelData(int \*destination)

{

  byte rawData[6];

  readRegisters(OUT\_X\_MSB, 6, rawData);

  for(int i = 0; i < 3 ; i++)

  {

    int gCount = (rawData[i\*2] << 8) | rawData[(i\*2)+1];

    gCount >>= 4;

    {

      gCount -= 0x1000;

    }

    destination[i] = gCount;

  }

}

void initMMA8452()

{

  byte c = readRegister(WHO\_AM\_I);  // Read WHO\_AM\_I register

  if (c == 0x2A) // WHO\_AM\_I should always be 0x2A

  {

    Serial.println("MMA8452Q is online...");

  }

  else

  {

    Serial.print("Could not connect to MMA8452Q: 0x");

    Serial.println(c, HEX);

    while(1) ;

  }

  MMA8452Standby();

  byte fsr = GSCALE;

  if(fsr > 8) fsr = 8;

  fsr >>= 2;

  writeRegister(XYZ\_DATA\_CFG, fsr);

  MMA8452Active();

}

void MMA8452Standby()

{

  byte c = readRegister(CTRL\_REG1);

  writeRegister(CTRL\_REG1, c & ~(0x01));

}

void MMA8452Active()

{

  byte c = readRegister(CTRL\_REG1);

  writeRegister(CTRL\_REG1, c | 0x01);

}

void readRegisters(byte addressToRead, int bytesToRead, byte \* dest)

{

  Wire.beginTransmission(MMA8452\_ADDRESS);

  Wire.write(addressToRead);

  Wire.endTransmission(false);

  Wire.requestFrom(MMA8452\_ADDRESS, bytesToRead);

  while(Wire.available() < bytesToRead);

  for(int x = 0 ; x < bytesToRead ; x++)

    dest[x] = Wire.read();

}

byte readRegister(byte addressToRead)

{

  Wire.beginTransmission(MMA8452\_ADDRESS);

  Wire.write(addressToRead);

  Wire.endTransmission(false);

  Wire.requestFrom(MMA8452\_ADDRESS, 1);

  while(!Wire.available()) ;

  return Wire.read();

}

void writeRegister(byte addressToWrite, byte dataToWrite)

{

  Wire.beginTransmission(MMA8452\_ADDRESS);

  Wire.write(addressToWrite);

  Wire.write(dataToWrite);

  Wire.endTransmission(); //Stop transmitting

}

## **Bases de datos**

SQL (lenguaje de consulta estructurado) es un lenguaje de programación diseñado para administrar datos en una base de datos relacional. Ha existido desde la década de 1970 y es el método más común para acceder a datos en bases de datos en la actualidad. SQL tiene una variedad de funciones que permiten a sus usuarios leer, manipular y cambiar datos. Si bien los ingenieros utilizan el SQL en el desarrollo de software, también es popular entre los analistas de datos por varias razones:

* Es semánticamente fácil de entender y aprender.
* Debido a que se puede usar para acceder a grandes cantidades de datos directamente donde se almacenan, los analistas no tienen que copiar datos en otras aplicaciones.
* En comparación con las herramientas de hoja de cálculo, el análisis de datos realizado en SQL es fácil de auditar y replicar.

Las bases de datos SQL son excelentes para realizar operaciones que normalmente podría hacer en una tabla dinámica de Excel (sumas, conteos, mínimos y máximos, etc.) pero en conjuntos de datos mucho más grandes y en varias tablas al mismo tiempo.

Al recibir los datos de los sensores de cada planta de bombeo, podemos observar el funcionamiento de nuestra bomba para detectar cualquier problema posible, por lo tanto, de acuerdo con la información recibida, tenemos la temperatura, la vibración en los ejes X, Y, Z, la humedad, la marca de tiempo y podemos verificar el punto crítico de cada variación con los siguientes códigos en "SQL":

Para encontrar la cantidad máxima de cada parámetro en un momento determinado (Imagen 3):

Select Max(Temperatura\_c),

Max(Humedad),

Max(Vibracion\_X),

Max(Vibracion\_Y),

Max(Vibracion\_Z)

from proyectoDATOS

where timestamp between '2018-12-18T05:50:35.708Z' and '2018-12-18T09:05:50.520Z'



Imagen 3 Búsqueda de máximos

Para encontrar el valor máximo de cada parámetro (Imagen 4):

Select Max(Temperatura\_c),

Max(Humedad),

Max(Vibracion\_X),

Max(Vibracion\_Y),

Max(Vibracion\_Z)

from proyectoDATOS

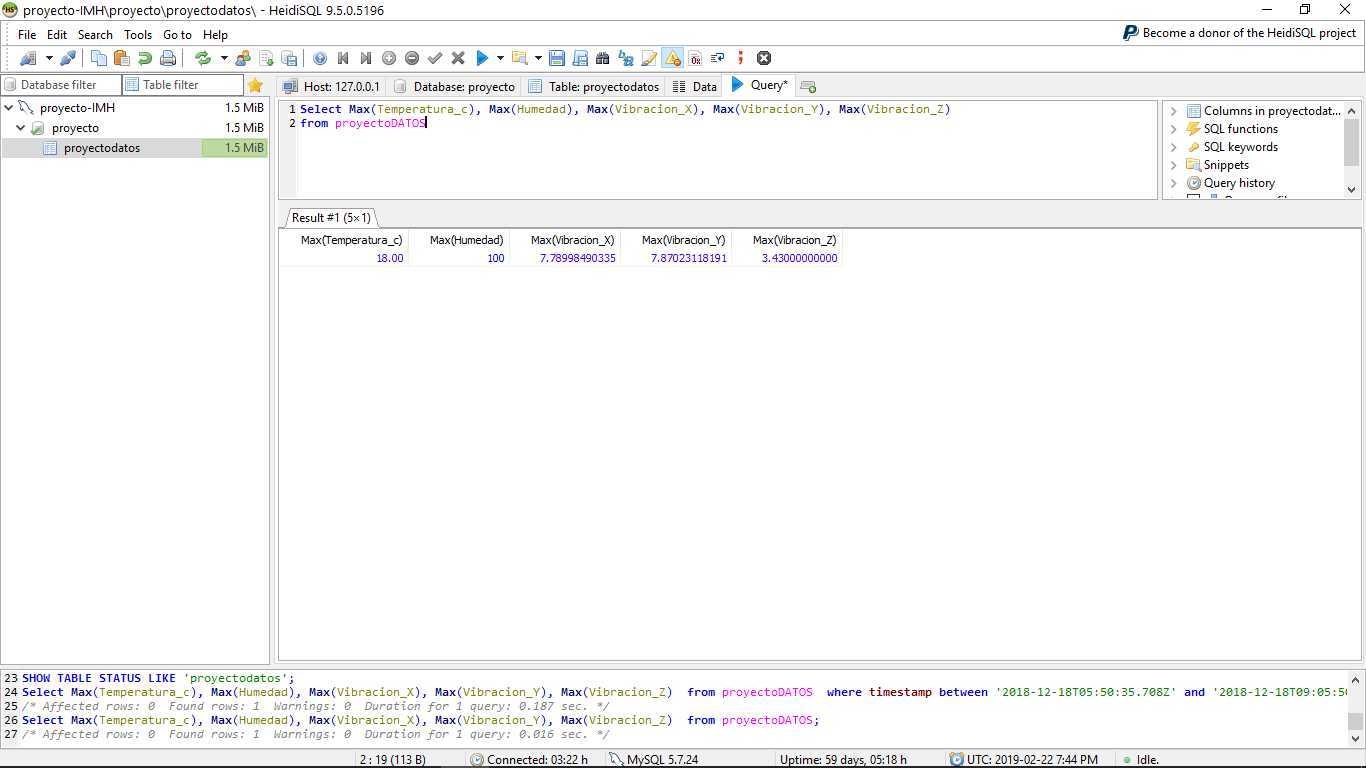


Imagen 4 Valor máximo de cada parámetro

Para encontrar valores máximos de cada parámetro cuando un parámetro obtenga un rango de valores determinados (Imagen 5):

Select Max(Temperatura\_c),

Max(Humedad),

Max(Vibracion\_X),

Max(Vibracion\_Y),

Max(Vibracion\_Z)

from proyectoDATOS

Where Vibracion\_X between 3.5 and 4

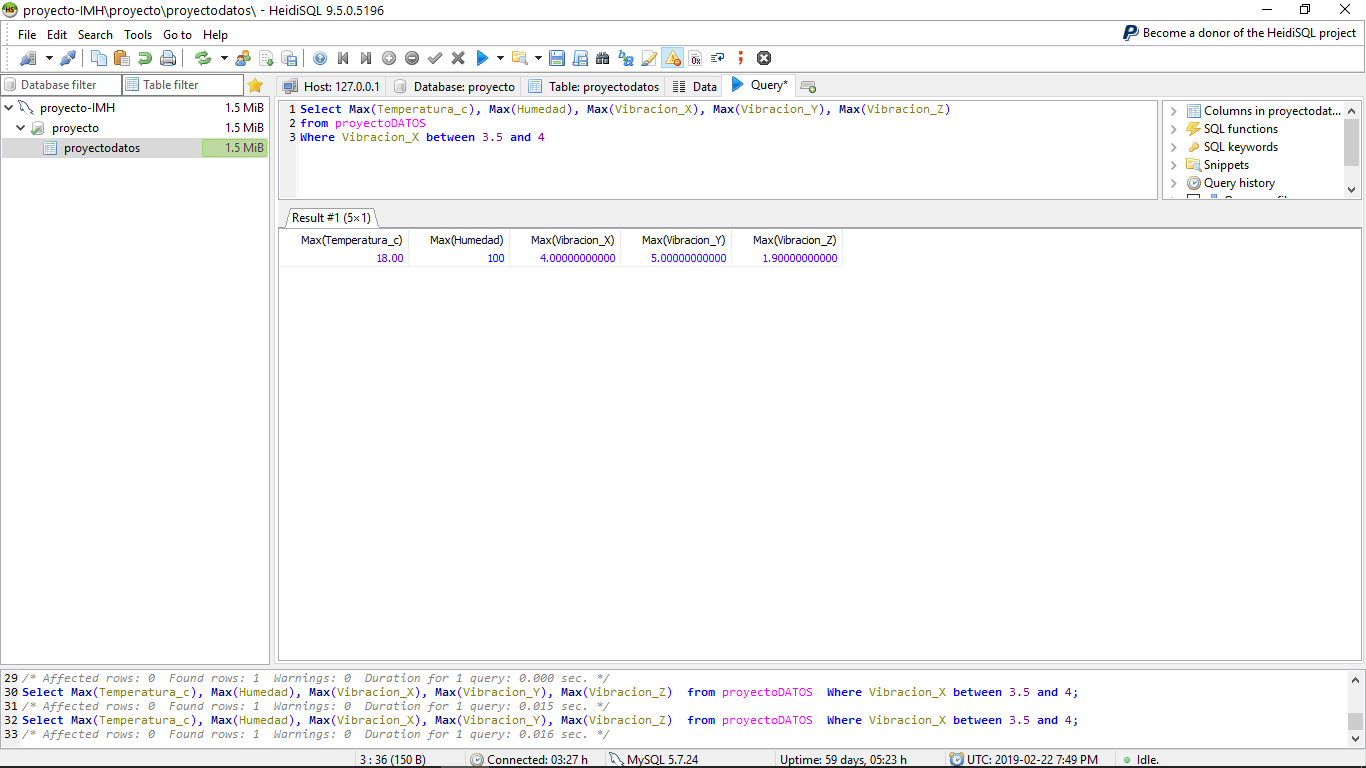


Imagen 5 Valores máximos de cada parámetro cuando un parámetro obtenga un rango de valores determinados

Así que ahora podemos analizar el comportamiento de nuestra bomba para protegerlo de posibles daños para el futuro.

## **Visualización web de datos**

Se realizará un dashboard de visualización web con el software Tableau 2018.3.3 en el que se representarán los datos recibidos. Como página web previa a la visualización de datos, se creó un “pen” en [www.codepen.io](http://www.codepen.io), donde se muestra el mapa de Benin, específicamente donde se encuentran los dos pozos estudiados. Si se quiere agregar más estaciones de bombeo, se pueden agregar fácilmente editando una zona específica del código. En la imagen 6 se muestra la visualización del Codepen, con botones para cada bomba que redireccionan hacia el dashboard de Tableau con la visualización de los datos.



Imagen 6 Codepen web previa a visualización de datos

En la pantalla también es posible hacer click encima de cada bomba y se podrá visualizar los datos de la bomba y del pozo. Para visualizar los datos, se debe hacer click en cada botón correspondiente a cada bomba.

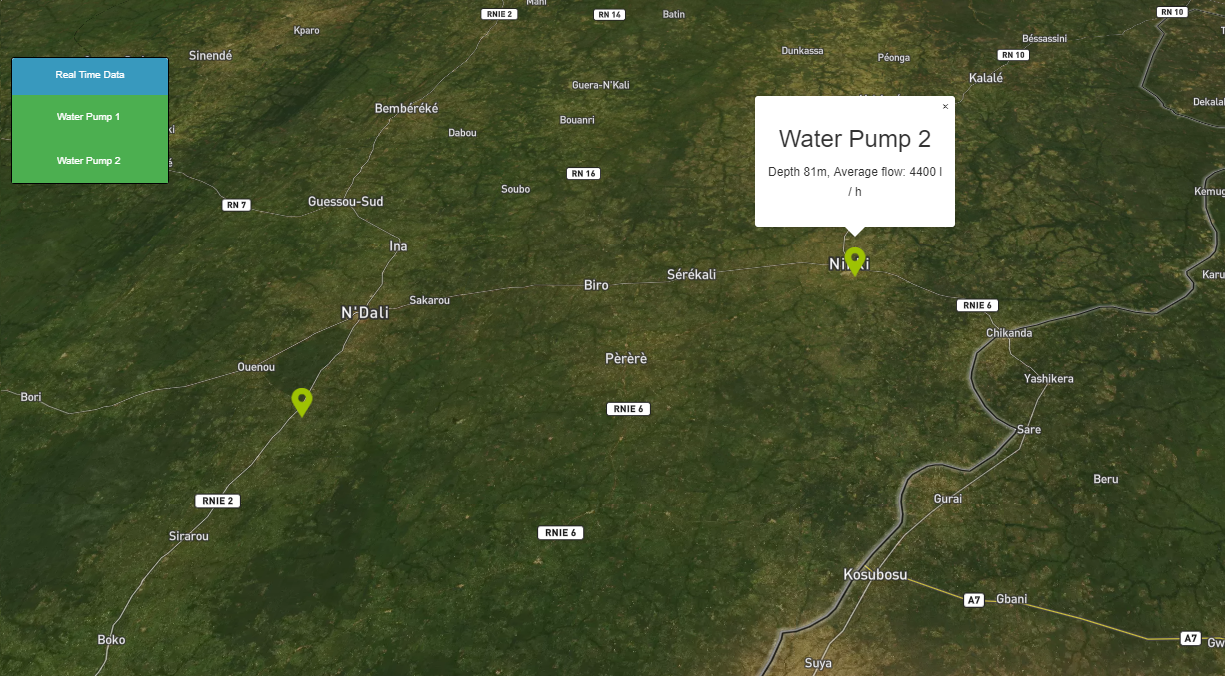


Imagen 7 Codepen información del pozo

Para realizar esta vista previa, se describe el código hecho en lenguaje HTML en [CodePen](https://codepen.io/jhonnyricci/pen/mvaLEW?editors=1010), si se quieren modificar datos o agregar más puntos de nuevas estaciones de bombeo, se muestran los siguientes snippet:

<--Para asignar una dirección web a cada botón del lado derecho de la pantalla, se modifica la dirección en esta sección del código-->

<form>

<input class="MyButton" onclick="window.location.assign('https://public.tableau.com/views/Bomba1/WaterPump1?:embed=y&:display\_count=yes&publish=yes?:showVizHome=no#3')" type="button" value=" Water Pump 1 " />

<form>

<input class="MyButton" onclick="window.location.assign('https://public.tableau.com/views/Bomba1/WaterPump1?:embed=y&:display\_count=yes&publish=yes?:showVizHome=no#3')" type="button" value=" Water Pump 2 " />

<--Para definir el centro del mapa y definir el zoom correspondiente, en el cual se va a mostrar de manera predeterminada, se modifican las siguientes coordenadas-->

var apiKey = "pk.eyJ1Ijoiam9yZGFuYnJhdW4iLCJhIjoiY2piNDBnZHphMGRqMTMzcXl4dDB4ZWt

oYyJ9.GUxwxcO9h7PJCWWuv0HbtA";

var mapCenter = [3.2, 9.8] <--map center-->

mapboxgl.accessToken = apiKey;

var map = new mapboxgl.Map(

{

style: 'mapbox://styles/mapbox/satellite-streets-v9',

center: mapCenter,

zoom: 9.5,

pitch: 45, <--zoom, pitch of the map-->

bearing: -360,

hash: true,

container: 'map'

});

<--En esta parte del código se pueden modificar las coordenadas de cada punto del mapa, correspondiente a cada estación de bombeo, al igual que la información que aparecerá cuando se haga click sobre la imagen-->

var geojsonall = {

type: 'FeatureCollection',

features: [{

type: 'Feature',

filter: 'water',

geometry: {

type: 'Point',

coordinates:

[2.696157, 9.899565]

},

properties: { <--info in each point of the map-->

title: 'Water Pump 1',

description: 'Depth 60m, Average flow: 2000 l / h'

}

},

{

type: 'Feature',

filter: 'outdoor',

geometry: {

type: 'Point',

coordinates: [3.215174, 9.938838]

},

properties: {

title: 'Water Pump 2',

description: 'Depth 81m, Average flow: 4400 l / h'

}

}]

};

Para realizar el dashboard de Tableau se utilizan los datos de la base de datos SQL compuesto de los datos obtenidos mediante el Gateway LoRa. Con ellos se general gráficos de amplitud de vibración máxima en cada intervalo de tiempo, la temperatura del motor y la presencia de agua en el pozo, en la imagen 5 se puede observar la página principal del dashboard.

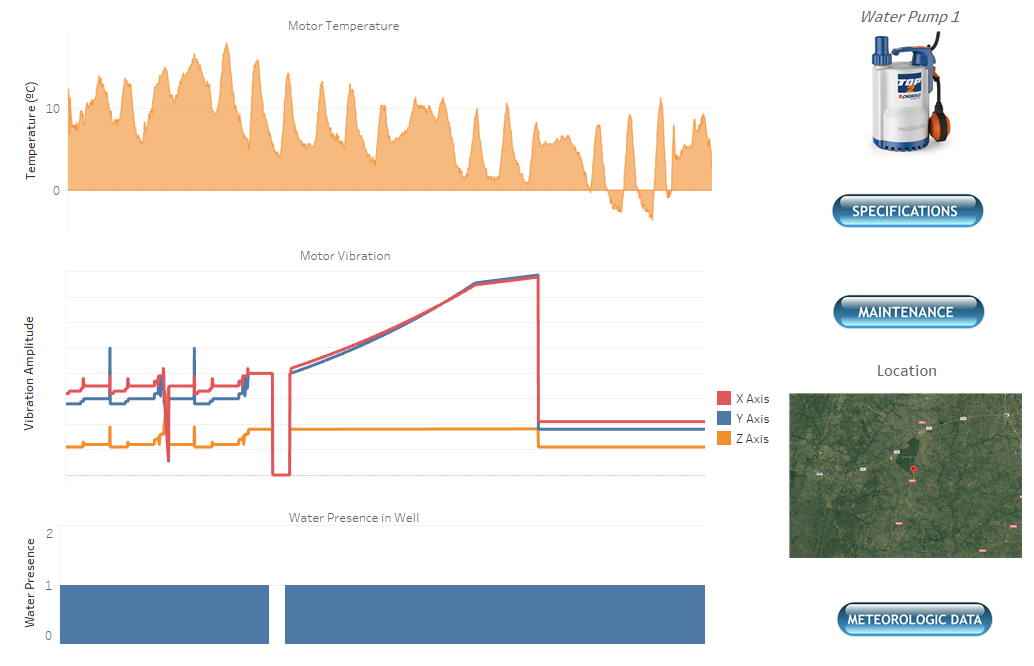


Imagen 8 Dashboard principal en Tableau

Del lado derecho del dashboard hay un botón que redirecciona a las especificaciones de la bomba, otro que indica los mantenimientos realizados a la central de bombeo, una imagen de la ubicación del pozo, en el que se puede hacer clic para redireccionar a los mapas con las coordenadas de la ubicación real de la estación, y por último otro botón que indica en otro dashboard el historial de los datos meteorológicos del lugar donde se encuentra la central (ver imagen 6).

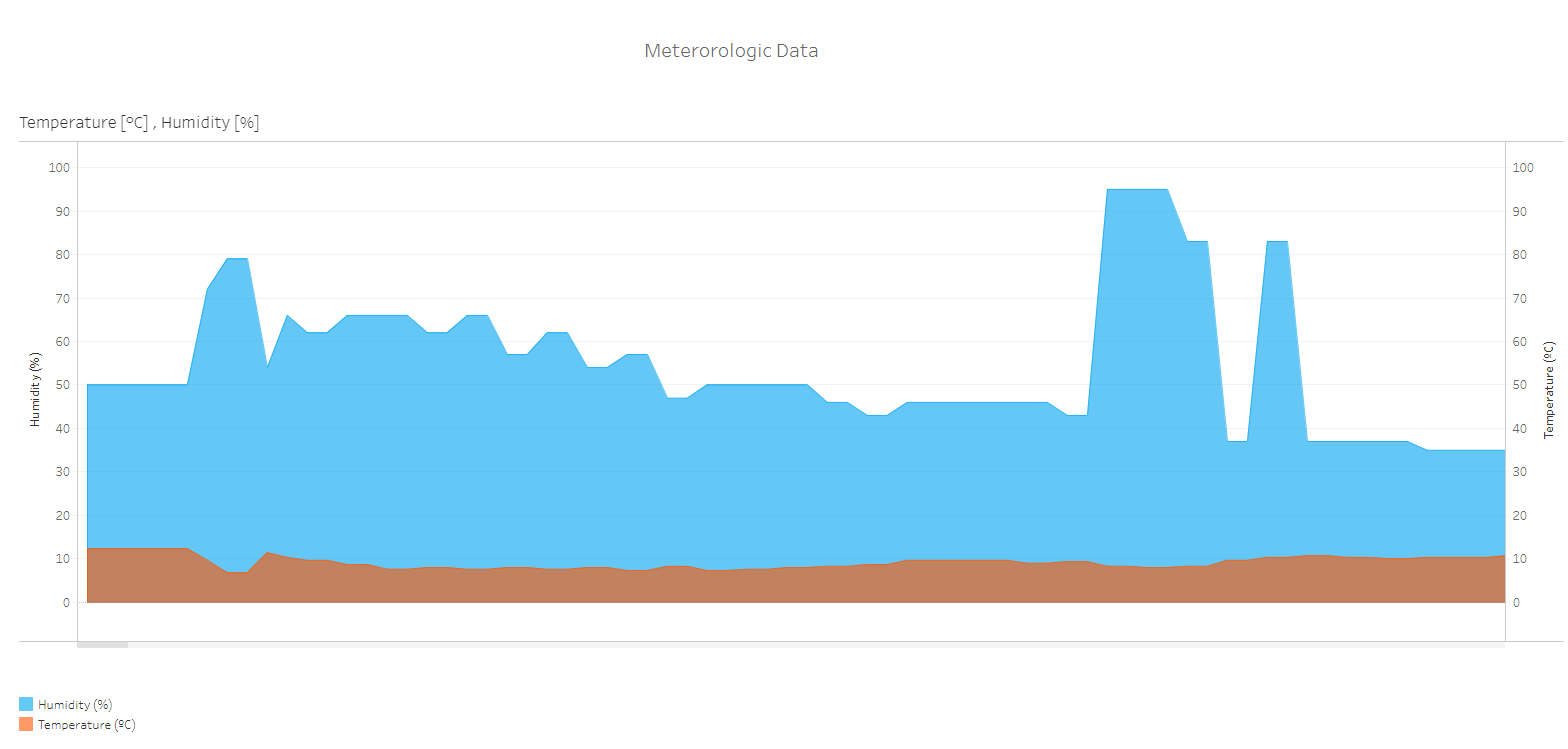


Imagen 9 Dashboard Datos Meteorológicos

# Metodología seguida para su implementación

Para la realización del prototipo ha sido necesario seguir un orden:

1. Estudio del problema a alto nivel
2. Desarrollo del diagrama temporal y repartir las distintas cargas de trabajo
3. Análisis de alternativas de los distintos sensores y módulos de comunicación
4. Estudio del problema a bajo nivel
5. Prototipo del circuito y del código
6. Juntar las distintas partes del trabajo
7. Prototipo y pruebas finales
8. Documentar el trabajo realizado

Para la posterior implementación real serían necesarios seguir los siguientes pasos:

1. Desarrollo del diagrama temporal y repartir las distintas cargas de trabajo
2. Análisis de alternativas de los distintos sensores y módulos de comunicación reales.
3. Estudio a bajo nivel de los nuevos componentes seleccionados.
4. Prototipo del circuito y código.
5. Pruebas de calidad, consumos, certificados, etc.
6. Despliegue de prueba.
7. Correcciones de los problemas encontrados
8. Despliegue a gran escala.
9. Mantenimiento y constante feedback para la mejora del producto

# Dificultades encontradas en la implementación

En el proceso de implementar un prototipo de nuestro sistema, hemos tenido que lidiar con varios problemas:

* Serigrafía LoRa incorrecta: Hemos identificado que la serigrafía del módulo LoRa es incorrecta. Se ha solucionado siguiendo literalmente las pistas de los componentes de la PCB y buscando sus datasheets. Este problema ha sido el que más atraso ha supuesto al proyecto ya que no se esperaba este tipo de fallo, pero al fin de al cabo es una de las posibles problemáticas que ocurren en proyectos reales.
* Comunicación entre Arduino y LoRa: La comunicación serial Tx Rx entre los dos módulos ha sido una tarea que ha consumido un tiempo notable. Tras buscar documentación en internet y solucionar el tema de las serigrafías, hemos sido capaces de comunicar correctamente entre los dos dispositivos.
* Lectura de sensores: Cada uno de los sensores tiene su propio modo de trabajar y sus peculiaridades. Este proceso es necesario en todos los proyectos, más que un problema es una dificultad por la cual es necesaria pasar para la realización correcta de un proyecto. Hemos tenido que leer documentación específica de cada uno de los sensores y logramos leer correctamente con cada uno de los sensores.
* Comunicación LoRa con el Hub: Al igual que en el punto anterior, es necesario pasar por este punto para realizar el proyecto. Nos consumió un gran tiempo conseguir comunicar con el Hub investigando entre distintos tipos formatos y baudios de comunicación.

# Mejoras necesarias para llevar a un entorno real

Idealmente se deberán utilizar carcasas sumergibles IP/68 para la protección de todos los componentes electrónicos. Los paneles solares deben estar conectados al sistema también de manera estanca.

Para escalar la solución y llevarla al campo de trabajo, se podrán optar por componentes electrónicos más confiables, Arduino nace como una herramienta de prototipado rápido**,** la producción en escala o para soluciones más complicadas el microcontrolador se queda corto.

Su facilidad obvia cosas necesarias, además, por ser una herramienta de prototipado, no es fiable, una mejor solución podría ser la de las placas de Texas Instruments que se asemeja al de Arduino, de hecho, se puede programar como un Arduino convencional y el propio programa se encarga de transformar lo programado al lenguaje C admitido por las placas de Texas.

Los acelerómetros, y sensores utilizados también están pensados para el prototipado, sin embargo, existe hardware adecuado para su utilización en el campo, los cuales ya se encuentran certificados con IP/68.

# **Plan técnico-económico para escalar la solución**

Para la venta de la estación de bombeo, el plan para escalar el producto a 50 máquinas de bombeo es necesario subcontratar una empresa que realice el montaje y testeo de cada una unidad. También es necesario contratar a una empresa de mantenimiento que quede cerca de las instalaciones para evitar largos trayectos de desplazamiento y por lo tanto de gastos. Se estima que el coste para un cliente será de unos 5k € por bomba sin incluir los gastos de mantenimiento.

Si aun fuese necesario escalar más el proyecto (cerrar contratos de, por ejemplo, de diez mil equipos) cambiaría el modelo de negocio de vender máquinas de bombeo a vender litros bombeados. De este modo el cliente podrá predecir los gastos con mucha facilidad y los gastos de mantenimiento los tendría cubiertos. Por la parte de fabricación, se dejaría de subcontratar a personal de montaje/testeo, y se pasaría a hacerlo todo en una planta de producción, aumentando los márgenes de gestión, mejora y beneficio. Se estima que los gatos para el cliente serían de 1€/100L con los gastos de mantenimiento incluidos.

Si siguiéramos expandiendo nuestro producto, habría que ampliar el equipo de trabajo para poder generar mayor valor añadido en el producto implementando más funciones, mejorando las ya implementadas y reduciendo los costes de los componentes.

Para la venta de la solución de monitoreo de estaciones de bombeo, el prototipo presentado es válido para una prueba de concepto, modelo de negocio, previsión de costes. Para llevarlo a un producto real con requisitos industriales, es necesario adquirir equipos más confiables, con microcontroladores industriales, directamente con un módulo LoRa, con un precio que va desde los 250 hasta los 500€. Deberemos también adquirir una carcasa con protección IP68 para el resguardo del hardware, que cuesta alrededor de 220€. Además de ello, se deben adquirir sensores de temperatura (95€), [acelerómetros industriales](http://sine.ni.com/nips/cds/view/p/lang/es/nid/213177#productlisting) (167€), sensores de humedad o de detección de agua y cables de alimentación y los correspondientes para cada sensor. En lo que es posible gastar de 600 a 900€ en todo el hardware necesario, dependiendo de la calidad y precio del equipo seleccionado.

Al tener un gran número de equipos de bombeo, debemos optar por una base de datos masiva que, por otro lado, debemos tener en cuenta la escalabilidad en el proyecto, tanto de equipos de bombeo, como para variables a obtener, por lo que una base de datos NO-SQL será la seleccionada para tal fin.

Es muy importante tener en cuenta el factor de escala, puesto que, para más estaciones de bombeo, los costes unitarios se reducirán, pero la base de datos crecerá, los equipos de captación de señales deberán ser mejores, y la rapidez de acceso a los datos se verá afectada, por lo que la base de datos que utilicemos deberá ser escalable. El costo de desarrollo, mantenimiento y funcionamiento será menor para más equipos, sin embargo, el software y el hardware utilizado deberá ser escalable.

Además de esto, al aplicar la solución a una máquina, los costes de equipos serán mucho mayores que al aplicarlo a un gran numero de ellas, los costos tanto del hardware como del software disminuyen a medida que la compra sea de un mayor número de unidades. Así, comprando 10 en lugar de uno, la disminución del precio puede ser inclusive un 10%, y si en lugar de 10, se compran mil unidades, puede suponer un ahorro de alrededor de 30%.

Se puede concluir que, a un mayor número de unidades vendidas, el precio final disminuirá, sin embargo tendremos que expandir el negocio contratando mas personal, expandiendo nuestras bases de datos y capacidad computacional.