

Construyendo dispositivos IoT (Arduino + Sensor Ultrasónico + MySQL Server)



Por Francisco Riccio 



Introducción

En el siguiente artículo presentaré como podemos construir un dispositivo IoT utilizando una placa de Arduino junto con algunos dispositivos electrónicos manteniendo la persistencia de sus datos directamente en una base de datos MySQL.

El dispositivo IoT que construiremos detectará la distancia de cualquier objeto que se encuentre frente a él; en caso la distancia sea menos de 30 centímetros, se prenderá un led automáticamente y además se almacenará la información en una base de datos.

Para realizar la implementación se requiere los siguientes dispositivos:

Dispositivo	Imagen
(01) Placa de Arduino (modelo: UNO o MEGA) Información del Producto: https://www.arduino.cc/en/Reference/Board	
(01) Sensor Ultrasónico (Modelo: HC-SR04)	
(01) Led (Diodo emisor de luz) de 5v	

<p>(01) Ethernet Shield – Arduino</p> <p>Información del Producto:</p> <p>https://www.arduino.cc/en/Main/ArduinoEthernetShieldV1</p>	
<p>(01) Protoboard (Opcional)</p>	

También se requiere un servidor que tenga instalado MySQL Server 5.7 para el almacenamiento de los datos que la placa de Arduino vaya capturando y enviando.

Adicional necesitamos Arduino IDE, el cual instalará los drivers requeridos para poder programar con la placa además de entregarnos un IDE de desarrollo.

En el siguiente url se puede descargar el Arduino IDE:

<https://www.arduino.cc/en/main/software>

La implementación a desarrollar tendrá la siguiente arquitectura a alto nivel:



Como se puede apreciar, el dispositivo de Arduino será capaz de enviar los datos que él recolecte al servidor de Base de Datos MySQL mediante sentencias SQL. No requeriremos ningún servidor Web que exponga servicios RESTful para lograr este objetivo, el cual si sería recomendado en caso estemos buscando integrar múltiples dispositivos IoT o realizar acciones más complejas como: Enviar correos, utilizar APIs para realizar llamadas telefónicas, entre otros.

Nota 1: Recordar que la placa de Arduino utiliza un microcontrolador, por lo cual está concebido fundamentalmente para ser utilizados en aplicaciones puntuales, es decir, aplicaciones donde el microcontrolador debe realizar un pequeño número de tareas, al menor costo posible.

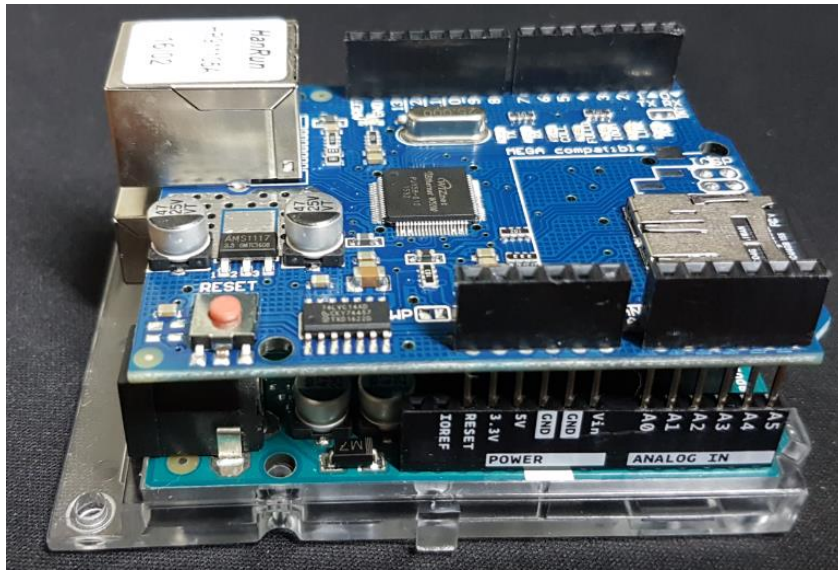
Implementación

I. Configuración de Dispositivos

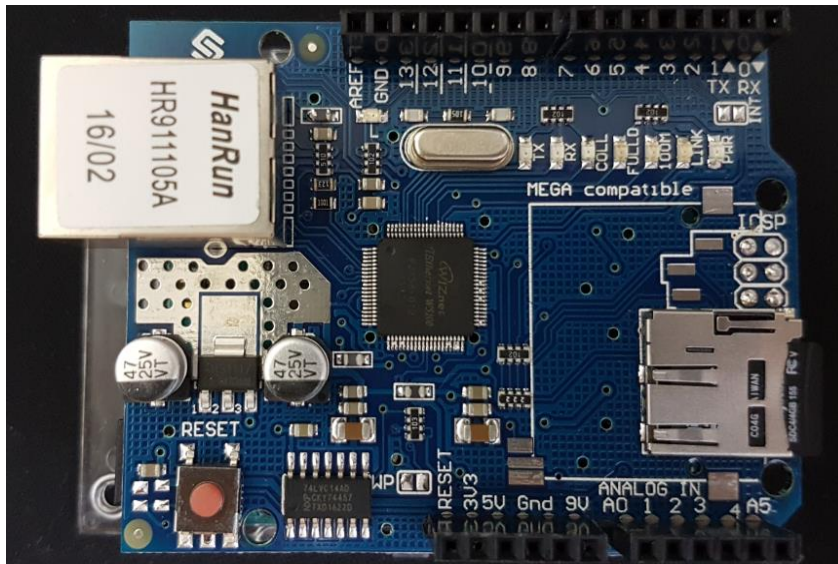
A continuación se presenta todos los pasos para realizar la implementación:

1. Ethernet Shield

Debemos colocar el Ethernet Shield encima de la placa de Arduino como a continuación se presenta:

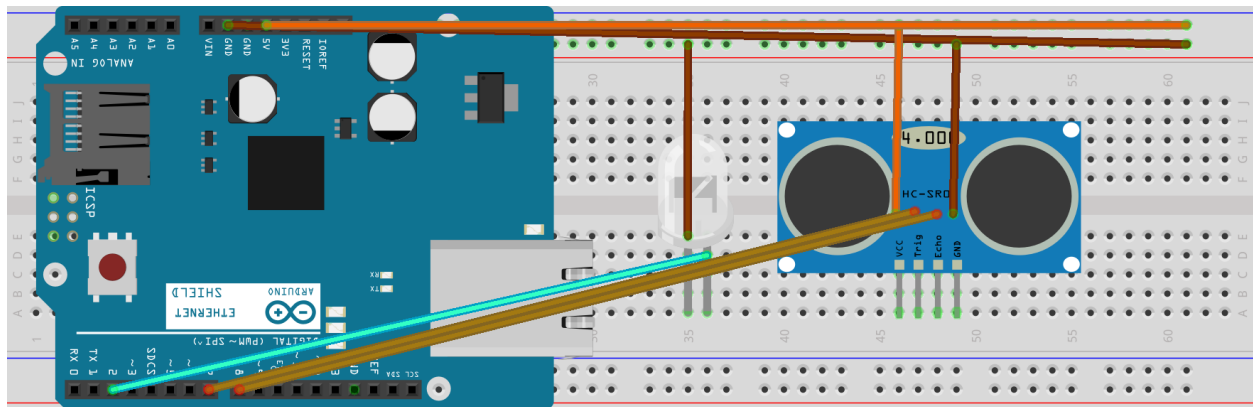


Una imagen desde la parte superior:



2. Sensor Ultrasónico (Modelo: HC-SR04) & Led

La configuración del Sensor Ultrasónico y el Led lo haremos apoyándonos de un Protoboard. Se presenta una maqueta de cómo debería ser configurado:



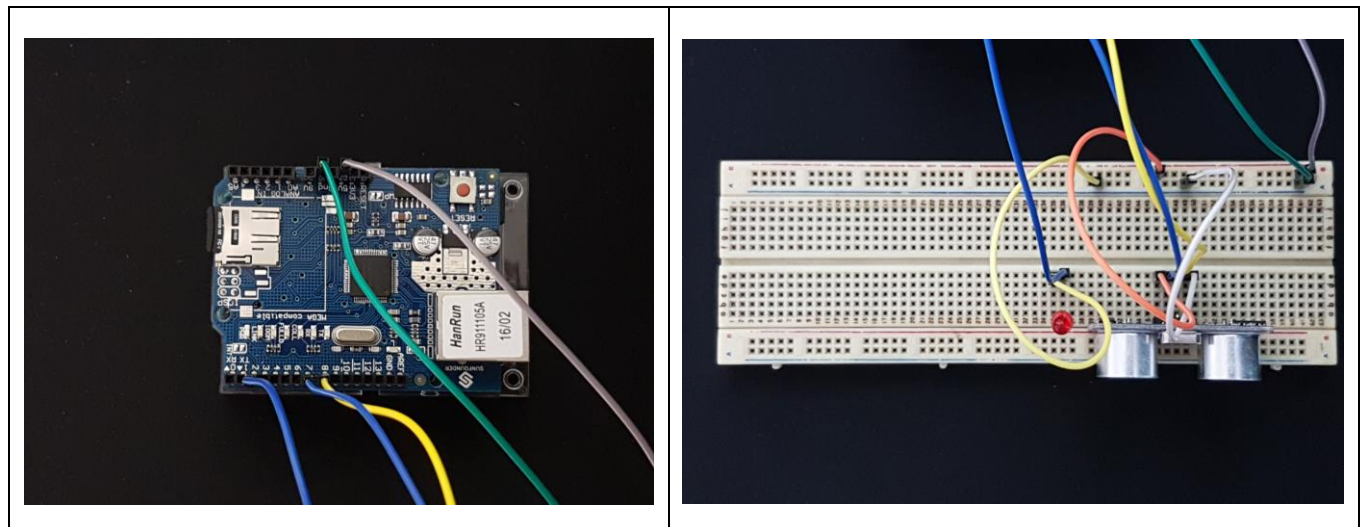
Led: Se instalará en las columnas 35 y 36 del Protoboard.

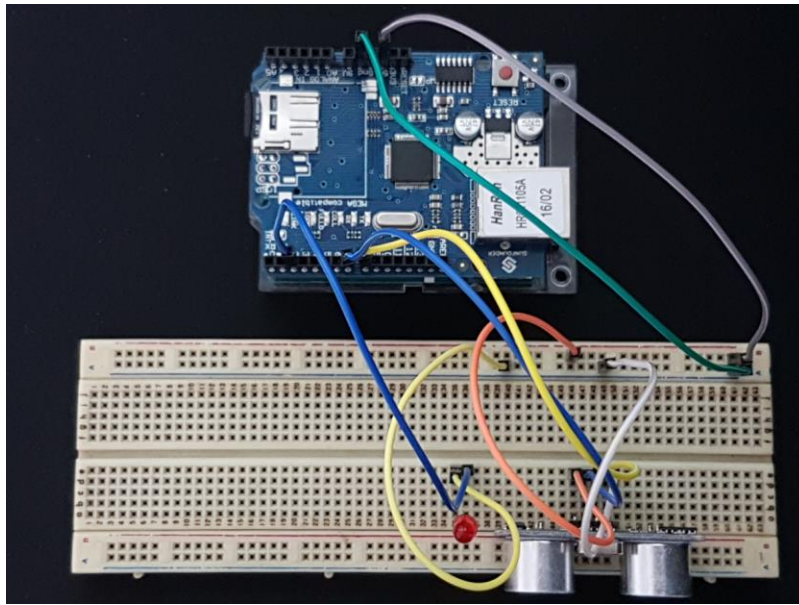
En la columna 35 se enlazará a Tierra y la columna 36 irá al puerto digital #2 del Ethernet Shield.

Sensor Ultrasónico: Se instalará en las columnas 46 al 49 del Protoboard, donde

- Columna 46, se enlazará a la fuente de alimentación de 5v.
- Columna 47, se enlazará con el puerto digital #7 del Protoboard. (Rol Trigger)
- Columna 48, se enlazará con el puerto digital #8 del Protoboard. (Rol Echo)
- Columna 49, se enlazará a Tierra.

A continuación se presenta imágenes de la configuración:





Antes de iniciar con el desarrollo del programa debemos colocar el cable Ethernet al puerto de red del Ethernet Shield.

II. Desarrollo del Programa

1. Configuración de la Base de Datos MySQL

Una vez instalado el MySQL Server en un servidor, debemos crear un usuario de base de datos que será utilizado por la aplicación de Arduino para conectarse. En la implementación utilizaremos el usuario root, el cual se le asignará permisos de conexión desde un terminal específico, en este caso desde el dispositivo Arduino (192.168.1.70).

```
C:\Users\Francisco>mysql -h localhost -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 527
Server version: 5.7.18-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE USER 'root'@'192.168.1.70' identified by 'oracle';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'192.168.1.70';
Query OK, 0 rows affected (0.00 sec)
```

Luego procederemos a crear una base de datos:

```
mysql> create database DATA_IOT;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> use DATA_IOT  
Database changed
```

Procedemos a crear las tablas que utilizaremos en la aplicación:

```
mysql> create table configuracion (param varchar(20) primary key, valor integer);  
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> insert into configuracion values ('LED',30);  
Query OK, 1 row affected (0.02 sec)
```

```
mysql> create table DISTANCIA (id integer not null auto_increment primary key, fecha date, valor integer);  
Query OK, 0 rows affected (0.03 sec)
```

Donde:

La tabla CONFIGURACIÓN almacenará la configuración de la aplicación. Para este ejemplo particular, se definirá cuál es el umbral de distancia que un objeto puede estar acercándose a la placa de Arduino antes de prender el Led y almacenar la información en la base de datos MySQL.

La tabla DISTANCIA básicamente almacena los centímetros de distancia entre la placa de Arduino y un objeto cuando este se encuentra a una distancia determinada.

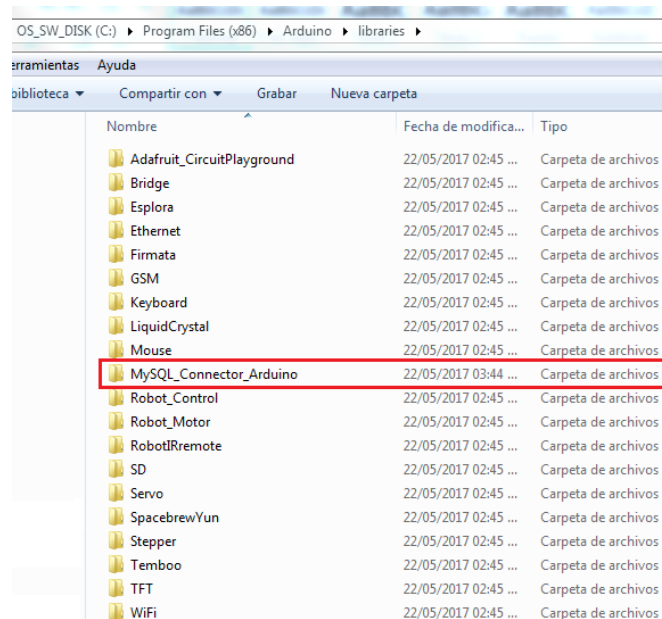
2. Librería de MySQL para Arduino

Actualmente existe la librería MySQL Connector/Arduino que nos permite conectarnos directamente a una base de datos MySQL desde nuestra placa de Arduino.

En el siguiente url se puede descargar la librería:

https://github.com/ChuckBell/MySQL_Connector_Arduino

Una vez descargado lo descomprimos y lo copiamos en el directorio libraries que es un sub-directorio de la instalación de Arduino IDE, ejemplo:



Asimismo podemos encontrar la documentación completa de la librería en el siguiente url:

https://github.com/ChuckBell/MySQL_Connector_Arduino/blob/master/extras/MySQL_Connector_Arduino_Reference_Manual.pdf

3. El código Fuente es el siguiente:

Se desarrollará utilizando el Arduino IDE.

A continuación se presenta diversas partes del código para realizar una explicación de cada una:

```
/* Manejo de Red */
#include <Ethernet.h>

/* Manejo de MySQL */
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>
```

1

```
/* Variables */
int trig = 7;
int echo = 8;
long tiempo = 0;
float distancia = 0;
int led = 2;
int umbral = 0;

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress servidorMySQL(192, 168, 1, 2);
IPAddress IPLocal(192, 168, 1, 70);
IPAddress DNS(192, 168, 1, 1);
IPAddress IPGateway(192, 168, 1, 1);
IPAddress IPSubnet(255, 255, 255, 0);
EthernetClient objRed;
```

2

```
MySQL_Connection cn((Client *)&objRed);
char userMySQL[] = "root";
char passwordMySQL[] = "oracle";
```

3

En la sección 1, nos encargamos de declarar las librerías a utilizarse en este proyecto, donde básicamente es el de Ethernet y MySQL.

En la sección 2, nos encargamos de declarar una serie de variables que serán utilizadas durante el desarrollo del programa. Cabe mencionar que las variables trig, echo y led deben ir acorde a los puertos digitales previamente indicados en este artículo.

Algo particular es que creamos una serie de objetos de tipo IPAddress que contendrán la configuración de Red de nuestro escenario, básicamente la IP que tendrá el dispositivo IoT (192.168.1.70), el gateway & dns (en mi caso es: 192.168.1.1), subnet entre otras configuraciones. También es importante notar que nosotros debemos definir una MAC con que se presentará el puerto de red y cuyos valores son inventados.

No olvidar que la IP que configuremos al dispositivo IoT debe tener accesos de conexión en la base de datos MySQL.

En la sección 3, procedemos a crear la variable de conexión hacia la base de datos de MySQL.

A continuación se detalla las funciones creadas que serán utilizadas posteriormente:

```
void iniciarTarjetaRed() {  
    Serial.println("Iniciando configuracion de Red...");  
    Ethernet.begin(mac, IPLocal, DNS, IPGateway, IPSubnet);  
}
```

1

```
int getUmbralDistanciaLed() {  
    if (cn.connect(servidorMySQL, 3306, userMySQL, passwordMySQL)) {  
        row_values *fila = NULL;  
        long resultado = 0;  
        MySQL_Cursor *cur_mem = new MySQL_Cursor(&cn);  
        char query[] = "select valor from data_iot.configuracion where param='LED'";  
        cur_mem->execute(query);  
        column_names *columns = cur_mem->get_columns();  
        do {  
            fila = cur_mem->get_next_row();  
            if (fila != NULL) {  
                resultado = atol(fila->values[0]);  
            }  
        } while (fila != NULL);  
        delete cur_mem;  
        cn.close();  
        return resultado;  
    } else {  
        Serial.println("Conexion a la BD MySQL ha fallado.");  
    }  
}
```

2

```
void grabarDatoBD(int pdato) {  
    if (cn.connect(servidorMySQL, 3306, userMySQL, passwordMySQL)) {  
        char SQL[] = "INSERT INTO data_iot.distancia (fecha,valor) VALUES (NOW(),%d)";  
        char query[128];  
        sprintf(query, SQL, pdato);  
        MySQL_Cursor *cur_mem = new MySQL_Cursor(&cn);  
        cur_mem->execute(query);  
        delete cur_mem;  
        cn.close();  
        Serial.println("Registro Grabado ");  
    } else {  
        Serial.println("Conexion a la BD MySQL ha fallado.");  
    }  
}
```

3

En la sección 1, nos encargamos de iniciar la configuración de la tarjeta de red con los atributos previamente especificados.

En la sección 2, creamos una función que devolverá el umbral de distancia permitido antes de prender el Led. Cabe mencionar que el código únicamente está preparado para leer 1 columna de 1 fila específica.

Nota: La librería permite obtener la información de una sentencia SQL que devuelve varias filas con n columnas.

En la sección 3, el procedimiento recibe la distancia entre un objeto y placa de Arduino a través del parámetro pdato y lo almacena en la tabla DISTANCIA. Para ello, utilizamos la función sprintf que nos permitirá construir la sentencia SQL.

A continuación se presenta la sección setup() de la aplicación:

```
void setup() {  
  // put your setup code here, to run once:  
  pinMode(led, OUTPUT); /* INPUT para entrada */  
  pinMode(trig, OUTPUT);  
  pinMode(echo, INPUT);  
  Serial.begin(9600);  
  iniciarTarjetaRed();  
  umbral = getUmbralDistanciaLed();  
}
```

El sensor ultrasónico permite detectar proximidad a través de impulsos de frecuencias bastantes altas. El dispositivo envía una frecuencia alta y recibe el eco, calculando el tiempo que ha viajado este impulso desde que partió hasta que retornó. Dicho impulso viaja a la velocidad del sonido (340 metros/segundo). Por ende, obteniendo el tiempo de la trayectoria y la velocidad del impulso, podemos calcular la distancia del objeto que ha cortado la trayectoria utilizando la siguiente fórmula:

Velocidad = Distancia / Tiempo

Despejando valores:

$$\frac{340m}{s} \times \frac{1s}{1000000us} \times \frac{100cm}{1m} = \frac{2d}{t}$$
$$d(cm) = \frac{t(us)}{59}$$

Por lo tanto, obtenemos la distancia del objeto en centímetros en función del tiempo transcurrido.

Revisando el código es importante indicar que la función pinMode nos permitirá definir el comportamiento del puerto si será de entrada o de salida.

La variable Led y Trig lo trabajamos como OUTPUT, porque será un dato que será enviado desde la placa de Arduino al dispositivo externo (sensor y el led).

La variable Echo la trabajamos como INPUT, porque nos devolverá el tiempo que ha demorado el impulso en un viaje completo, el cual será devuelto por el sensor a la placa de Arduino.

Por último, nos conectamos a la base de datos y leemos el umbral permitido entre la distancia de la placa de Arduino y un objeto. Para nuestro caso, el valor es de 30 centímetros.

Nota: El procedimiento setup() solo se ejecutará una vez y será cuando prendamos la placa de Arduino.

Finalmente, se detalla la sección loop() de la aplicación:

```
void loop() {  
  // put your main code here, to run repeatedly|||||||:  
  digitalWrite(trig, HIGH);  
  delayMicroseconds(10);  
  digitalWrite(trig, LOW);  
  tiempo = pulseIn(echo, HIGH);  
  distancia = tiempo / 59;  
  Serial.println("DISTANCIA = " + String(distancia) + "cm");  
  if (distancia <= umbral) {  
    digitalWrite(led, HIGH);  
    grabarDatoBD(distancia);  
  }  
  else {  
    digitalWrite(led, LOW);  
  }  
  delay(500);  
}
```

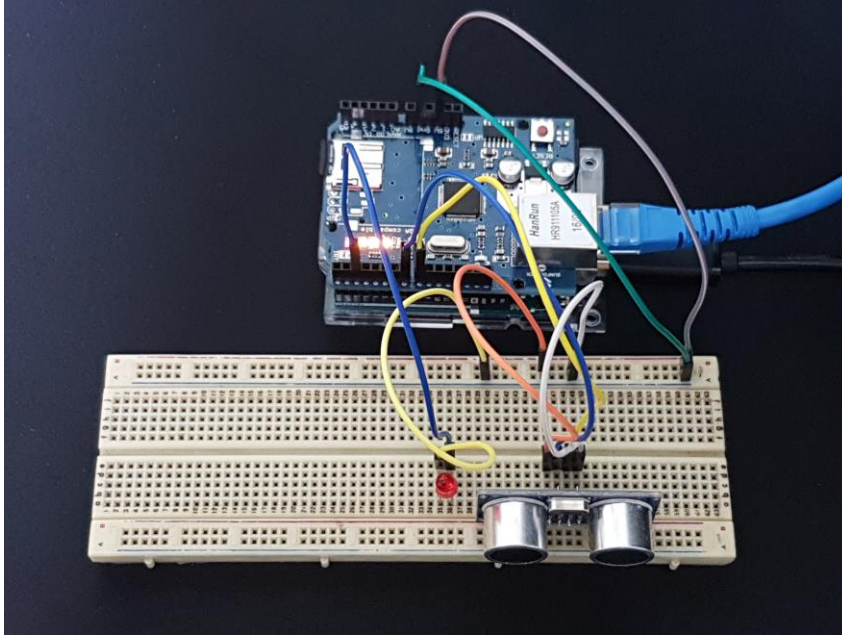
A continuación se detalla las principales líneas del código:

Código	Explicación
digitalWrite(trig, HIGH); delayMicroseconds(10);	Enviamos un pulso por 10 microsegundos.
digitalWrite(trig, LOW);	Detenemos el envío del pulso desde el sensor.
tiempo = pulseIn(echo, HIGH);	Leemos el tiempo transcurrido del pulso que ha retornado.
distancia = tiempo / 59;	Calculamos la distancia del objeto basado en la fórmula previamente explicada.
digitalWrite(led, HIGH);	Prendemos el Led.
digitalWrite(led, LOW);	Apagamos el Led.

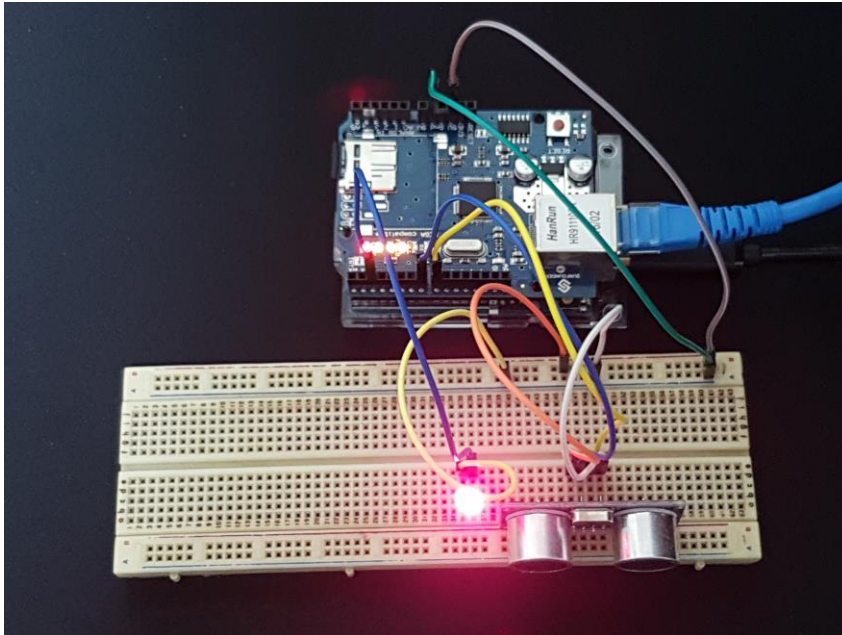
Nota: El procedimiento loop() se ejecutará de manera indefinida después de ejecutarse el procedimiento void() hasta que apaguemos la placa de Arduino.

4. Probamos la aplicación:

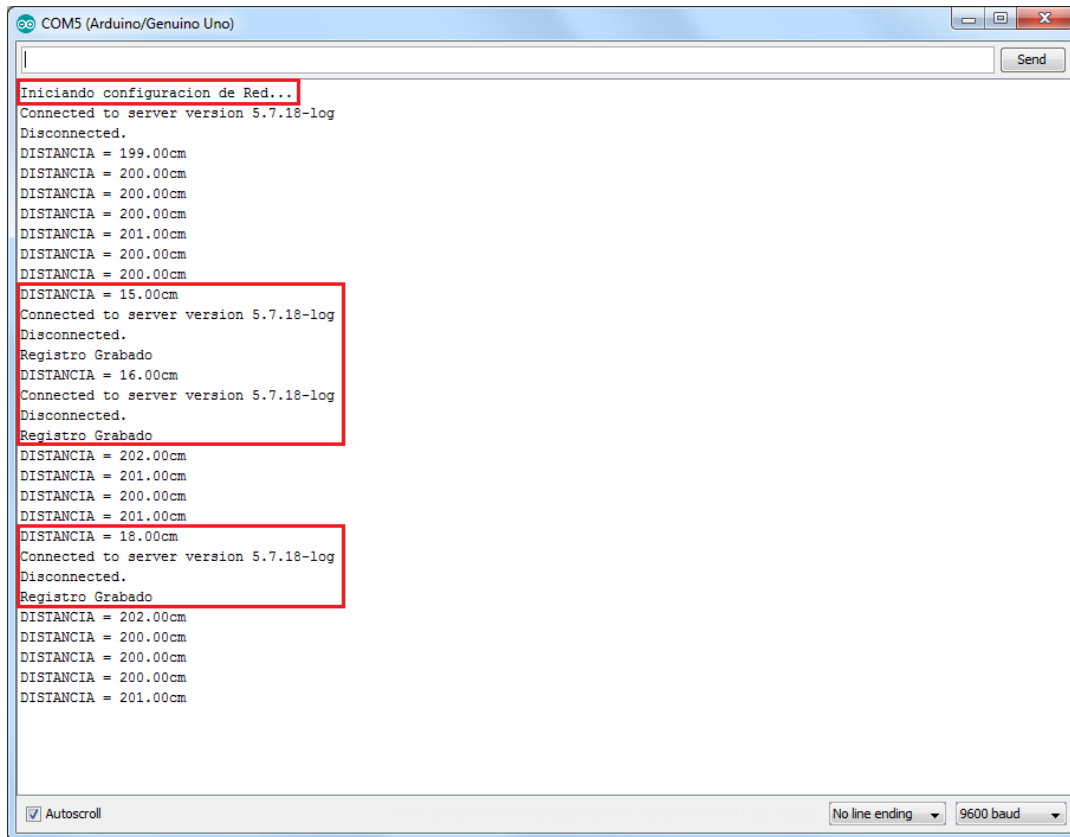
En un inicio el dispositivo no prenderá el Led debido a que ningún objeto está a una distancia menor de 30 centímetros.



Una vez que acercamos un objeto a una distancia menor a 30 centímetros el Led se prende e inicia su registro directamente en la base de datos MySQL.



Asimismo en el spool del terminal veremos la siguiente salida:



```
COM5 (Arduino/Genuino Uno)
Iniciando configuracion de Red...
Connected to server version 5.7.18-log
Disconnected.
DISTANCIA = 199.00cm
DISTANCIA = 200.00cm
DISTANCIA = 200.00cm
DISTANCIA = 200.00cm
DISTANCIA = 201.00cm
DISTANCIA = 200.00cm
DISTANCIA = 200.00cm
DISTANCIA = 15.00cm
Connected to server version 5.7.18-log
Disconnected.
Registro Grabado
DISTANCIA = 16.00cm
Connected to server version 5.7.18-log
Disconnected.
Registro Grabado
DISTANCIA = 202.00cm
DISTANCIA = 201.00cm
DISTANCIA = 200.00cm
DISTANCIA = 201.00cm
DISTANCIA = 18.00cm
Connected to server version 5.7.18-log
Disconnected.
Registro Grabado
DISTANCIA = 202.00cm
DISTANCIA = 200.00cm
DISTANCIA = 200.00cm
DISTANCIA = 200.00cm
DISTANCIA = 201.00cm
```

En la base de datos MySQL podemos comprobar los datos almacenados:

```
C:\Users\Francisco>mysql -h localhost -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 564
Server version: 5.7.18-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use DATA_IOT;
Database changed
mysql> select * from DISTANCIA;
+----+-----+-----+
| id | fecha   | valor |
+----+-----+-----+
| 1  | 2017-05-23 | 28    |
| 2  | 2017-05-23 | 6      |
| 3  | 2017-05-23 | 18     |
| 4  | 2017-05-23 | 17     |
| 5  | 2017-05-23 | 6      |
| 6  | 2017-05-23 | 15     |
| 7  | 2017-05-23 | 16     |
| 8  | 2017-05-23 | 16     |
| 9  | 2017-05-23 | 15     |
| 10 | 2017-05-23 | 16     |
| 11 | 2017-05-23 | 18     |
+----+-----+-----+
11 rows in set (0.00 sec)
```

Nota: La conexión de red de la placa de Arduino se logró gracias al dispositivo Ethernet Shield que nos permitió conectarnos vía Red, pero también es posible hacerlo vía Wireless utilizando el Arduino Wireless SD Shield.

Se adjunta todo el código fuente:

```
/* Manejo de Red */
#include <Ethernet.h>

/* Manejo de MySQL */
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>

/* Variables */
int trig = 7;
int echo = 8;
long tiempo = 0;
float distancia = 0;
int led = 2;
int umbral = 0;

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress servidorMySQL(192, 168, 1, 2);
IPAddress IPLocal(192, 168, 1, 70);
IPAddress DNS(192, 168, 1, 1);
IPAddress IPGateway(192, 168, 1, 1);
IPAddress IPSubnet(255, 255, 255, 0);
EthernetClient objRed;

MySQL_Connection cn((Client *)&objRed);
char userMySQL[] = "root";
char passwordMySQL[] = "oracle";

void iniciarTarjetaRed() {
  Serial.println("Iniciando configuracion de Red...");
  Ethernet.begin(mac, IPLocal, DNS, IPGateway, IPSubnet);
}

int getUmbralDistanciaLed() {
  if (cn.connect(servidorMySQL, 3306, userMySQL, passwordMySQL)) {
    row_values *fila = NULL;
    long resultado = 0;
    MySQL_Cursor *cur_mem = new MySQL_Cursor(&cn);
    char query[] = "select valor from data_iot.configuracion where param='LED'";
    cur_mem->execute(query);
    column_names *columns = cur_mem->get_columns();
    do {
      fila = cur_mem->get_next_row();
      if (fila != NULL) {
        resultado = atol(fila->values[0]);
      }
    } while (fila != NULL);
    delete cur_mem;
    cn.close();
    return resultado;
  } else {
```



```

    Serial.println("Conexion a la BD MySQL ha fallado.");
}
}

void grabarDatoBD(int pdato) {
    if (cn.connect(servidorMySQL, 3306, userMySQL, passwordMySQL)) {
        char SQL[] = "INSERT INTO data_iot.distancia (fecha,valor) VALUES (NOW(),%d)";
        char query[128];
        sprintf(query, SQL, pdato);
        MySQL_Cursor *cur_mem = new MySQL_Cursor(&cn);
        cur_mem->execute(query);
        delete cur_mem;
        cn.close();
        Serial.println("Registro Grabado ");
    } else {
        Serial.println("Conexion a la BD MySQL ha fallado.");
    }
}

void setup() {
    // put your setup code here, to run once:
    pinMode(led, OUTPUT); /* INPUT para entrada */
    pinMode(trig, OUTPUT);
    pinMode(echo, INPUT);
    Serial.begin(9600);
    iniciarTarjetaRed();
    umbral = getUmbralDistanciaLed();
}

void loop() {
    // put your main code here, to run repeatedly| ||||| |||||:
    digitalWrite(trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(trig, LOW);
    tiempo = pulseIn(echo, HIGH);
    distancia = tiempo / 59;
    Serial.println("DISTANCIA = " + String(distancia) + "cm");
    if (distancia <= umbral) {
        digitalWrite(led, HIGH);
        grabarDatoBD(distancia);
    }
    else {
        digitalWrite(led, LOW);
    }
    delay(500);
}

```

Conclusión

Hemos revisado una serie de consideraciones para implementar una solución IoT que nos ha permitido detectar objetos cuya distancia ha sido menor a 30 centímetros y poderlo registrar vía red en un servidor de base de datos MySQL.

Esta implementación podemos expandirla a otros ejemplos como: sonar una alarma en vez de un led (utilizando altavoces y un relé), medición de temperatura, sensor de llamas, captura de fotos, sensor de humedad, entre otras funcionalidades; lo importante es que podemos recoger los datos y almacenarlos en MySQL donde luego podemos explotar la información a través de herramientas de análisis de datos para conseguir tendencias y estimaciones.

Publicado por Ing. Francisco Riccio. Es un Cloud Architect en IBM Perú e instructor de cursos oficiales de certificación Oracle. Está reconocido por Oracle como un Oracle ACE y certificado en productos de Oracle Application & Base de Datos.

e-mail: franciscoriccio@gmail.com

web: www.friccio.com