**UNIVERSIDAD TECNOLÓGICA DE TIJUANA**

[](http://www.uttijuana.edu.mx/)

**TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN**

**INTEGRADORA II**

**TRABAJO: PROPUESTA DE INNOVACIÓN**

**Profesor: M.C. Florencio López Cruz**

Presentan:

Yessenia Monrroy 10B

Alfredo Ruelas 10B

Eric Arellano 10C

Víctor Morales 10B

Tijuana, Baja California, 12 de octubre de 2016.

**ÍNDICE**

PÁG.

[**INTRODUCCIÓN** 3](#_Toc464125579)

[**OBJETIVO Y ALCANCE** 4](#_Toc464125580)

[**MEJORAS** 5](#_Toc464125581)

[**CRONOGRAMA** 7](#_Toc464125582)

# **INTRODUCCIÓN**

En el presente trabajo se hará mención de los diversos componentes que conforman la parte del hardware del proyecto E-guide, las cuales se indicaran distribuidas en cada capa del modelo de referencia IoT, nuestro proyecto está conformado por un módulo de conexión con el estándar de la IEEE 802.11b/g usando el protocolo UDP, 2 sensores ULTRASONICOS uno para lectura constante en el cual su uso se destina a situaciones de riesgo (sensor de segundo plano), el otro sensor es accionado a petición del usuario, un vibrador para alertar al usuario en situaciones de alerta, una bocina para emitir alertas sonoras, un módulo para trasmitir sonido atreves de un auricular, indicadores led para saber la situación de operación del artefacto, un módulo MP3 para la reproducción de alertas sonoras.

El proyecto se escalara a módulos que se apliquen mejor o representen mejora continua al mismo.

**Capa 1 Sensores y Controladores**

Se dispone solo de un sensor ultrasónico para que el envío y recepción de señal sea solo de frente.

Otro para alertas de peligro.

Se dispuso de un módulo GPRS para poder identificar la ubicación del usuario.

**Capa 2 Conectividad**

Se dispondrá de un servicio, el cual será contratado para que este tenga conectividad y poder realizar las consultas sobre su ubicación.

**Capa 3 Computación en la Frontera**

Se estarán tomando los datos del servicio contratado.

**Capa 4 Acumulación de Datos**

Una vez implementado el módulo GPRS y GPS se harán cambios importantes en la estructura de la base datos para soportar datos históricos.

**Capa 5 Abstracción de Datos**

Actualmente los datos se mandan de manera continua a través de parámetros NO ESTANDARIZADOS

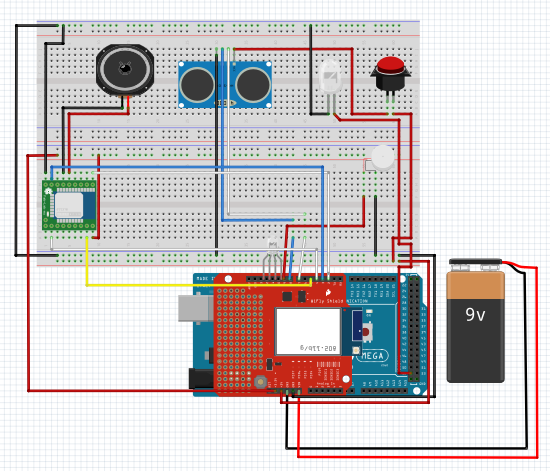
**Capa 6 Aplicación**

Se cambiaran algunos aspectos del dashboard, para que este tenga una mejor comprensión de los datos enviados por el dispositivo.

**Capa 7 Colaboración**

Comenzar con el funcionamiento del dispositivo logrando que enviara, reciba, procese los datos y estos puedan ser consultados en tiempo real y logre informar su localización precisa.

# **Diagrama**



# **MATERIALES**

Los materiales con los que se espera finalizar el proyecto son:

* Sensores HC-SR04
* Arduino mega
* Módulo MP3 MTV020M01
* Múltiples Leds
* Múltiples interruptores
* Batería de ácido de 6volts 4.5AH
* Circuito para auriculares E2130
* Módulo GPRS
* Modulo GPS
* Wifi shield
* Bocina de baja potencia 1.5 watts

# **CONFIGURACION.**

Pines digitales arduino los únicos implementados.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | - | 17 | - | 34 | - | 51 | WS |
| 1 | - | 18 | - | 35 | - | 52 | WS |
| 2 | IL | 19 | - | 36 | - | 53 | - |
| 3 | - | 20 | - | 37 | EUS | - |  |
| 4 | WS | 21 | - | 38 | TUS | - |  |
| 5 | - | 22 | RMP3 | 39 | - | - |  |
| 6 | - | 23 | CMP3 | 40 | - | - |  |
| 7 | - | 24 | DMP3 | 41 | - | - |  |
| 8 | - | 25 | BMP3 | 42 | - | - |  |
| 9 | - | 26 | - | 43 | - | - |  |
| 10 | - | 27 | - | 44 | - | - |  |
| 11 | - | 28 | - | 45 | - | - |  |
| 12 | IL | 29 | - | 46 | - | - |  |
| 13 | IL | 30 | - | 47 | - | - |  |
| 14 | - | 31 | - | 48 | - | - |  |
| 15 | - | 32 | - | 49 | - | - |  |
| 16 | - | 33 | - | 50 | WS | - |  |

Simbología.

IL =Indicador LED

WS= Wifi Shield

RMP3= Reset MP3

CMP3= Clock MP3

DMP3= Data MP3

BMP3= Bussy MP3

EUS= Echo Sensor Ultrasonico

TUS= Triger Sensor Ultrasonico

Librerías arduino:

#include <Wtv020sd16p.h>

#include <SPI.h>

#include <WiFi.h>

#include <WiFiUdp.h>

Librerias Java:

import hypermedia.net.\*;

import processing.net.\*;

# **SENSORES.**

Implementación: Se implementó el sensor HC-SR04



Visualización del sensor

Codificación:

#define Pecho 6

#define Ptrig 7

long duracion, distancia;

void setup() {

Serial.begin (9600); // inicializa el puerto seria a 9600 baudios

pinMode(Pecho, INPUT); // define el pin 6 como entrada (echo)

pinMode(Ptrig, OUTPUT); // define el pin 7 como salida (triger)

pinMode(13, 1); // Define el pin 13 como salida

}

void loop() {

digitalWrite(Ptrig, LOW);

delayMicroseconds(2);

digitalWrite(Ptrig, HIGH); // genera el pulso de triger por 10ms

delayMicroseconds(10);

digitalWrite(Ptrig, LOW);

duracion = pulseIn(Pecho, HIGH);

distancia = (duracion/2) / 29; // calcula la distancia en centimetros

if (distancia >= 500 || distancia <= 0){ // si la distancia es mayor a 500cm o menor a 0cm

Serial.println("---"); // no mide nada

}

else {

Serial.print(distancia); // envia el valor de la distancia por el puerto serial

Serial.println("cm"); // le coloca a la distancia los centimetros "cm"

digitalWrite(13, 0); // en bajo el pin 13

}

if (distancia <= 10 && distancia >= 1){

digitalWrite(13, 1); // en alto el pin 13 si la distancia es menor a 10cm

Serial.println("Alarma......."); // envia la palabra Alarma por el puerto serial

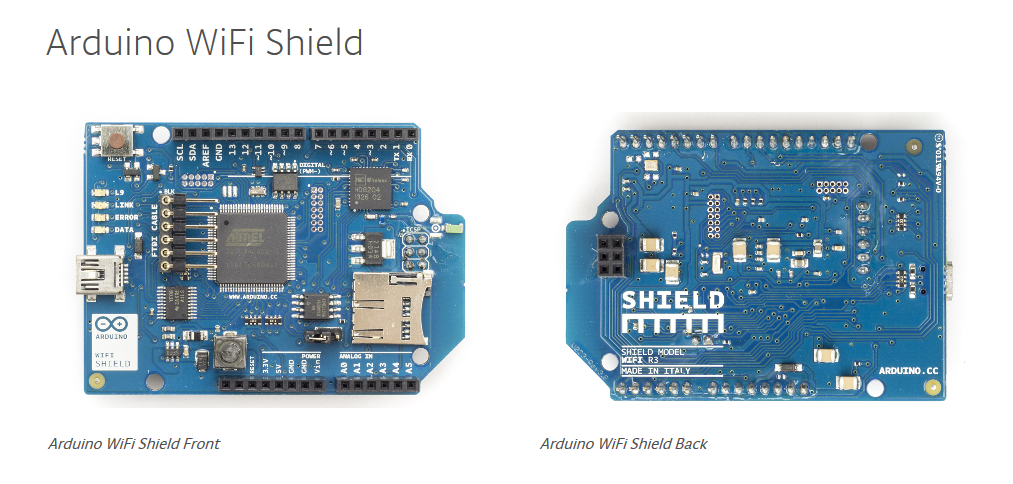
}

delay(400); // espera 400ms para que se logre ver la distancia en la consola

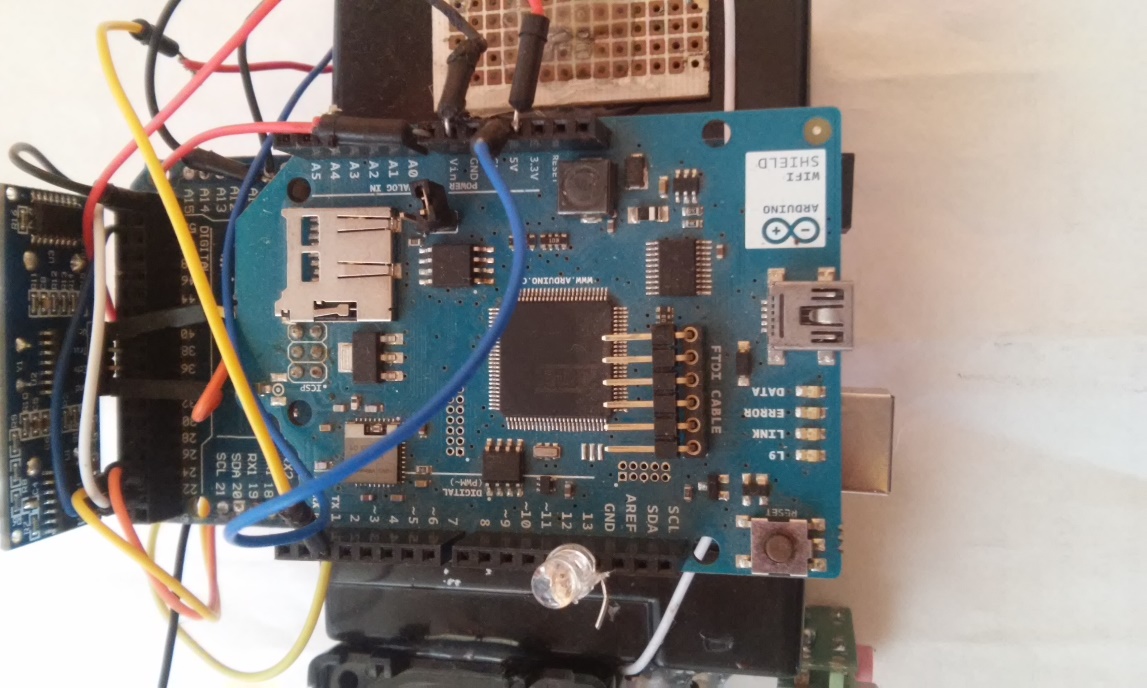
}

# **MODULOS DE COMUNICACION.**

Implementación



Visualización del Shield



Shield implementado

Codificación

Serial.println("Connected to wifi");

printWifiStatus();

Serial.println("\nStarting connection to server...");

Udp.begin(localPort);}

void loop() {

// if there's data available, read a packet

int packetSize = Udp.parsePacket();

if (packetSize) {

Serial.print("Received packet of size ");

Serial.println(packetSize);

Serial.print("From ");

IPAddress remoteIp = Udp.remoteIP();

Serial.print(remoteIp);

Serial.print(", port ");

Serial.println(Udp.remotePort());

// read the packet into packetBufffer

int len = Udp.read(packetBuffer, 255);

if (len > 0) {

packetBuffer[len] = 0;

}

Serial.println("Contents:");

Serial.println(packetBuffer);

Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());

Udp.write(ReplyBuffer);

Udp.endPacket();

}}

void printWifiStatus() {

// print the SSID of the network you're attached to:

Serial.print("SSID: ");

Serial.println(WiFi.SSID());

// print your WiFi shield's IP address:

IPAddress ip = WiFi.localIP();

Serial.print("IP Address: ");

Serial.println(ip);

// print the received signal strength:

long rssi = WiFi.RSSI();

Serial.print("signal strength (RSSI):");

Serial.print(rssi);

Serial.println(" dBm");

}

#include <SPI.h>

#include <WiFi.h>

#include <WiFiUdp.h>

int status = WL\_IDLE\_STATUS;

char ssid[] = "yourNetwork"; // your network SSID (name)

char pass[] = "secretPassword"; // your network password (use for WPA, or use as key for WEP)

int keyIndex = 0; // your network key Index number (needed only for WEP)

unsigned int localPort = 2390; // local port to listen on

char packetBuffer[255]; //buffer to hold incoming packet

char ReplyBuffer[] = "acknowledged"; // a string to send back

WiFiUDP Udp;

void setup() {

//Initialize serial and wait for port to open:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

// check for the presence of the shield:

if (WiFi.status() == WL\_NO\_SHIELD) {

Serial.println("WiFi shield not present");

// don't continue:

while (true);

}

String fv = WiFi.firmwareVersion();

if (fv != "1.1.0") {

Serial.println("Please upgrade the firmware");

}

// attempt to connect to Wifi network:

while (status != WL\_CONNECTED) {

Serial.print("Attempting to connect to SSID: ");

Serial.println(ssid);

// Connect to WPA/WPA2 network. Change this line if using open or WEP network:

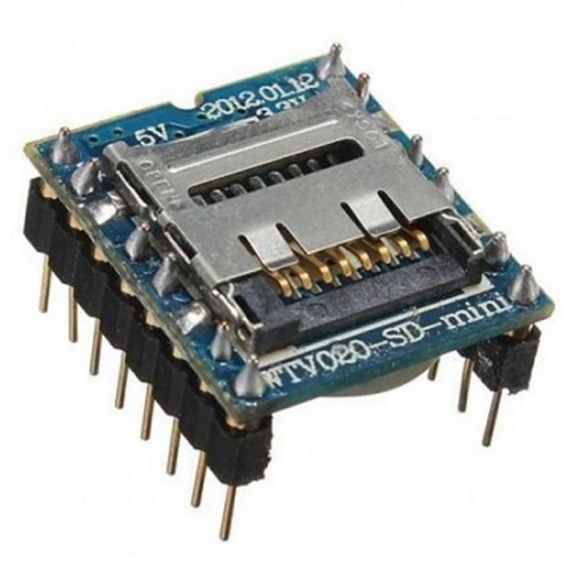
status = WiFi.begin(ssid);

// wait 10 seconds for connection:

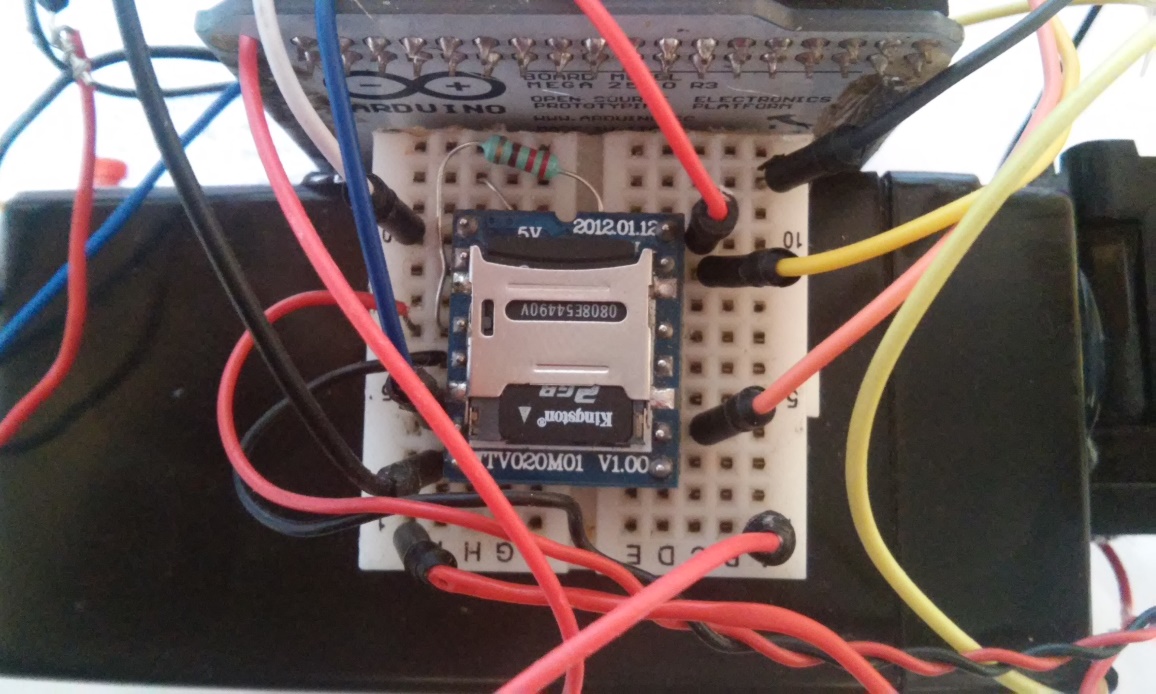
delay(10000);

}

Sensores con módulo de comunicación



Visualización del módulo MP3.



Modulo implementado.

Codificación

#include <Wtv020sd16p.h>

int resetPin = 2; // The pin number of the reset pin.

int clockPin = 3; // The pin number of the clock pin.

int dataPin = 4; // The pin number of the data pin.

int busyPin = 5; // The pin number of the busy pin.

Wtv020sd16p wtv020sd16p(resetPin,clockPin,dataPin,busyPin);

void setup() {

wtv020sd16p.reset();

}

void loop() {

wtv020sd16p.playVoice(0);

wtv020sd16p.asyncPlayVoice(1);

delay(5000);

wtv020sd16p.pauseVoice();

delay(5000);

wtv020sd16p.pauseVoice();

delay(5000);

wtv020sd16p.stopVoice();

wtv020sd16p.asyncPlayVoice(2);

delay(2000);

wtv020sd16p.mute();

delay(2000);

wtv020sd16p.unmute();

delay(2000);

wtv020sd16p.stopVoice();

}

# **Resultados**

# **Fotografías del proyecto**

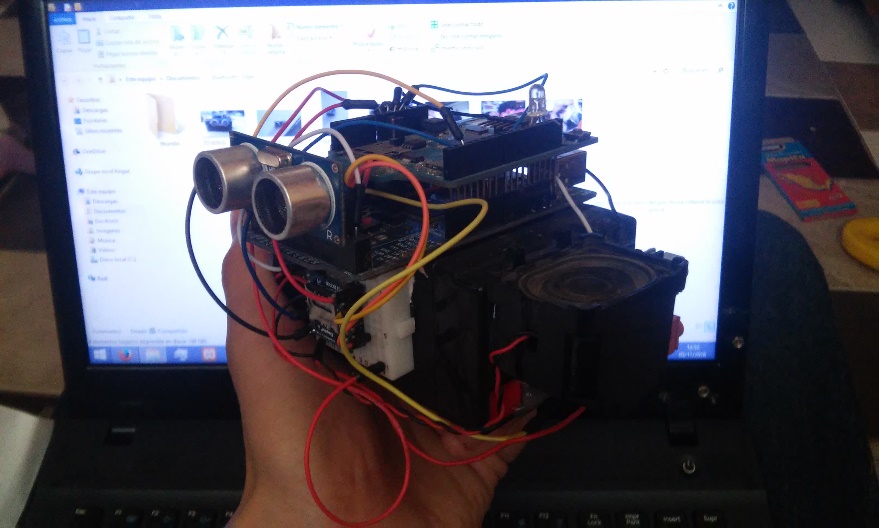
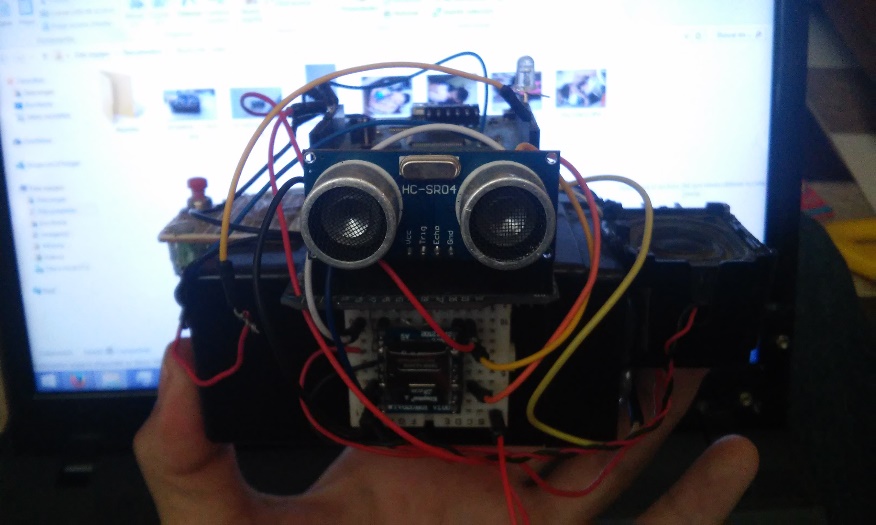
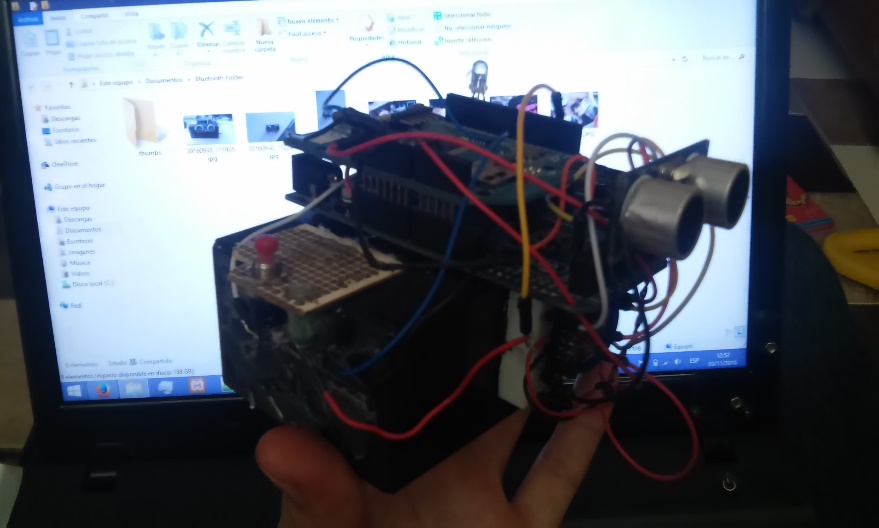


Imagen del proyecto (condición actual).

# **APORTACION A INSTRUCTABLES.COM**

