Introducción a la Programación de Microcontroladores

Proyecto final

Profesor: José Luis Di Biase

Alumnos: Jonathan De Rojas, Mauro Portillo, Pablo Pire

Índice

Descripción del proyecto

Este proyecto tiene como objetivo realizar una alarma doméstica, un sistema de detección de intrusos o gas que permita advertir el allanamiento en una vivienda o inmueble al usuario.

El usuario debe encender el sistema mediante un control remoto presionando el botón Power (esto se le indica en una pantalla LCD). También puede apagar el sistema mediante el mismo botón.
Una vez encendido, si el sistema detecta la presencia de gas o movimiento, comienza a sonar la alarma por el parlante, se muestra en el LCD que sensor lo detectó y se manda un mensaje al "servicio de monitoreo" (en este caso simulado por un servicio **Rest** deployado en Heroku) para que deje registro del movimiento detectado (id sensor y fecha) en una base de datos.
La alarma seguirá sonando hasta que el usuario la detenga mediante el botón Stop del control remoto.
También se creó una aplicación **Android** que permite ver los movimientos registrados en dicha base.

Descripción de conexiones

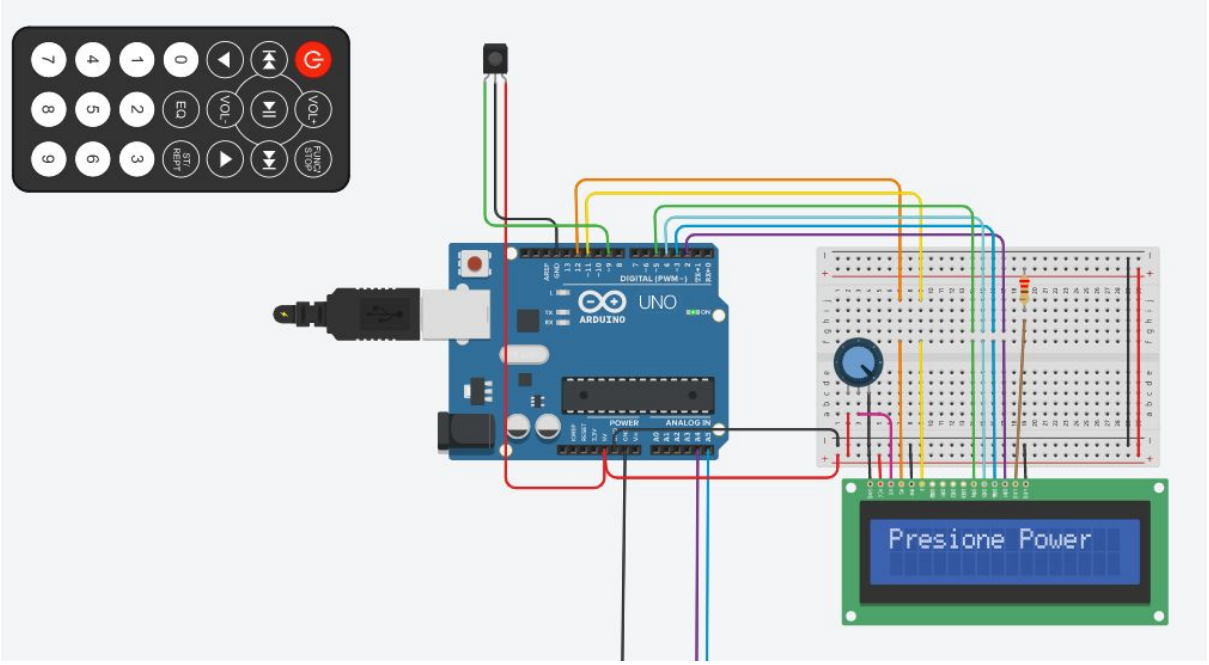
Para realizar este proyecto se utilizaron en Tinckercad dos Arduinos conectados entre sí (master y esclavo) utilizando la librería Wire.
El Arduino N°1 se utilizó para conectar la pantalla LCD y el sensor IR para el control remoto.
El Arduino N°2 se utilizó para conectar el sensor de distancia, el sensor de gas, el parlante de la alarma, leds que indican el encendido y el módulo WIFI.
El Módulo Wifi se configuró mediante los comandos AT (hayes).

Lista de componentes

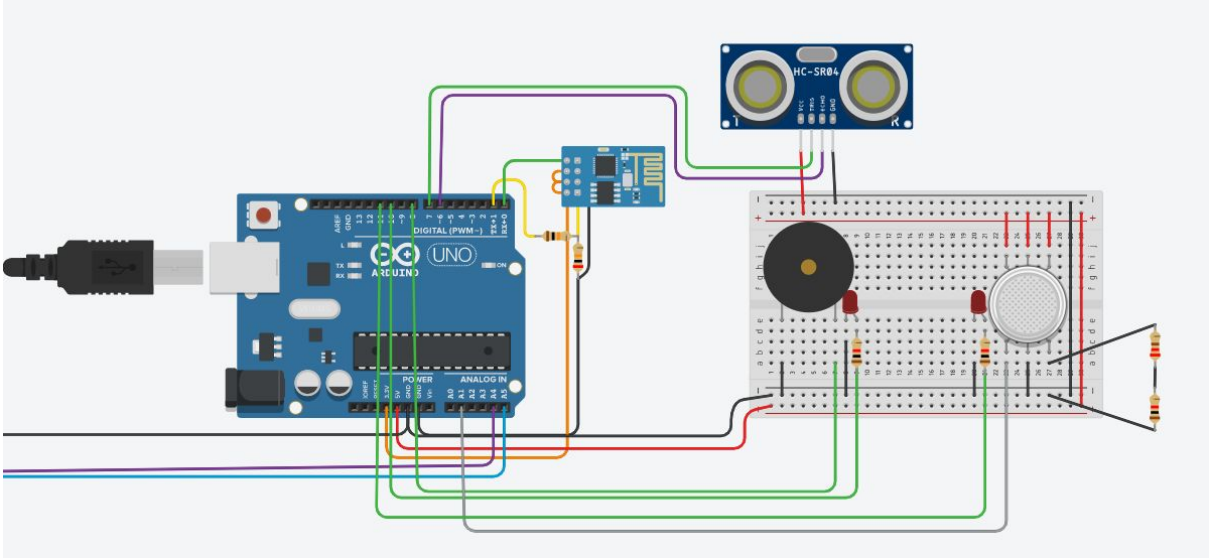
- 2 Arduino uno.
- Módulo ESP8266(WIFI).
- Sensor de distancia Ultrasonico.
- Sensor de gas.
- Sensor IR.
- Control Remoto por IR.
- 2 leds.
- 6 resistencias.
- 2 placas de pruebas.
- LCD 16 X 2.
- Potenciómetro.
- Piezo(Parlante)

Esquema de conexión

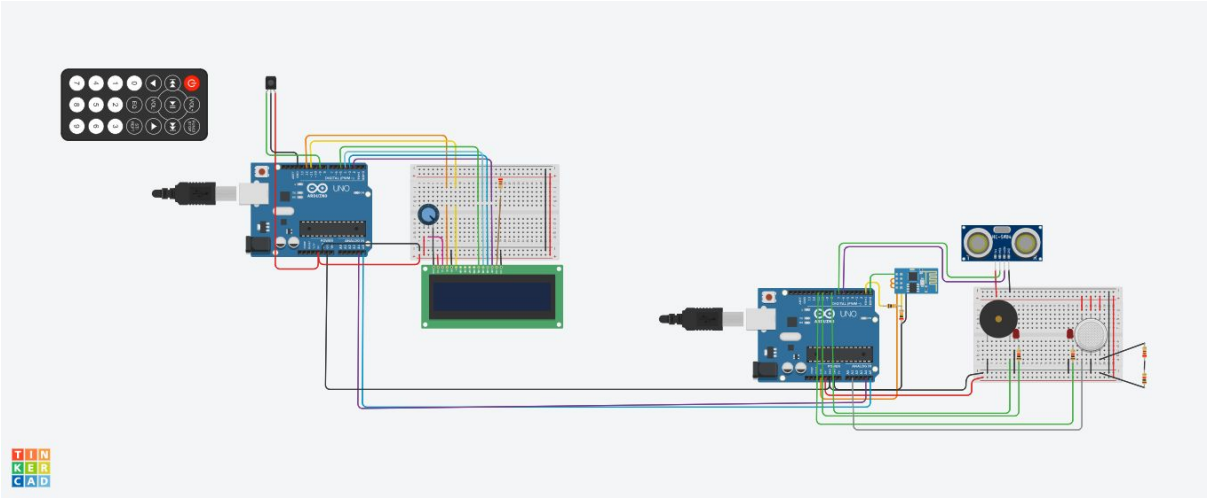
Arduino N°1(Master)



Arduino N°2 (Esclavo)

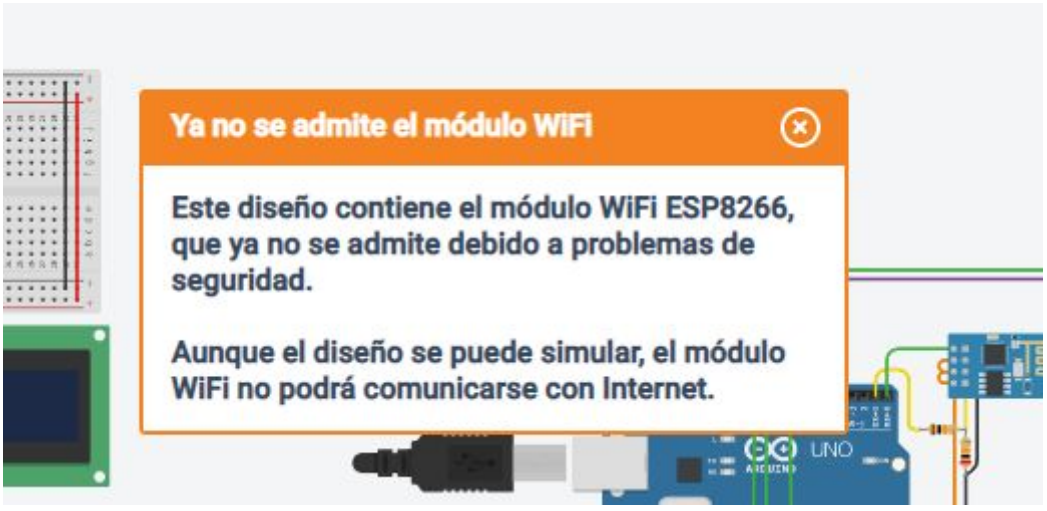


Esquema Terminado

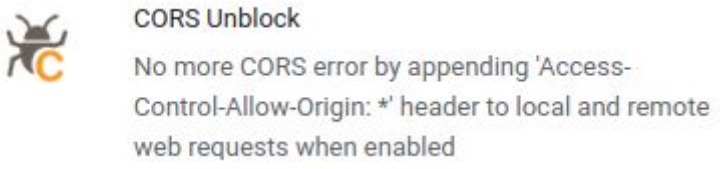


Problemas

- En el desarrollo del proyecto, al momento de integraciones de código, notamos que el Módulo ESP8266(módulo de WIFI) dejó de estar disponible en Tinkercad para nuevos proyectos, pero si se podía mantener los de los proyectos que ya estaban armados. Por lo tanto, se utilizó el que tenía el ESP8266 como base para integrar los distintos desarrollos que se hicieron.



- Se tuvo que instalar un plugin en el navegador para que pueda ejecutar request desde el simulador hacia la api REST. Se puede instalar cualquiera, pero este puede ser un ejemplo:



- A la hora de desarrollar la API REST para las notificaciones de los eventos que las alarmas emitían, surgió un problema, que cuando se consumía ésta desde otro sistema, en este caso una aplicación android, la api lanzaba un error `Access-Control-Allow-Origin` is a CORS (Cross-Origin Resource Sharing) header. Por lo tanto, se tuvo que hacer una configuración especial en la Api, Desarrollada en java.

- Se había pensado en un principio, agregar la funcionalidad que permite desactivar la alarma en forma remota por medio de una llamada a un server que estaría escuchando peticiones, configurado con el módulo ESP8266. El mismo se configuró, pero no se encontró la forma de acceder desde el exterior por medio del simulador.

Desarrollos extras

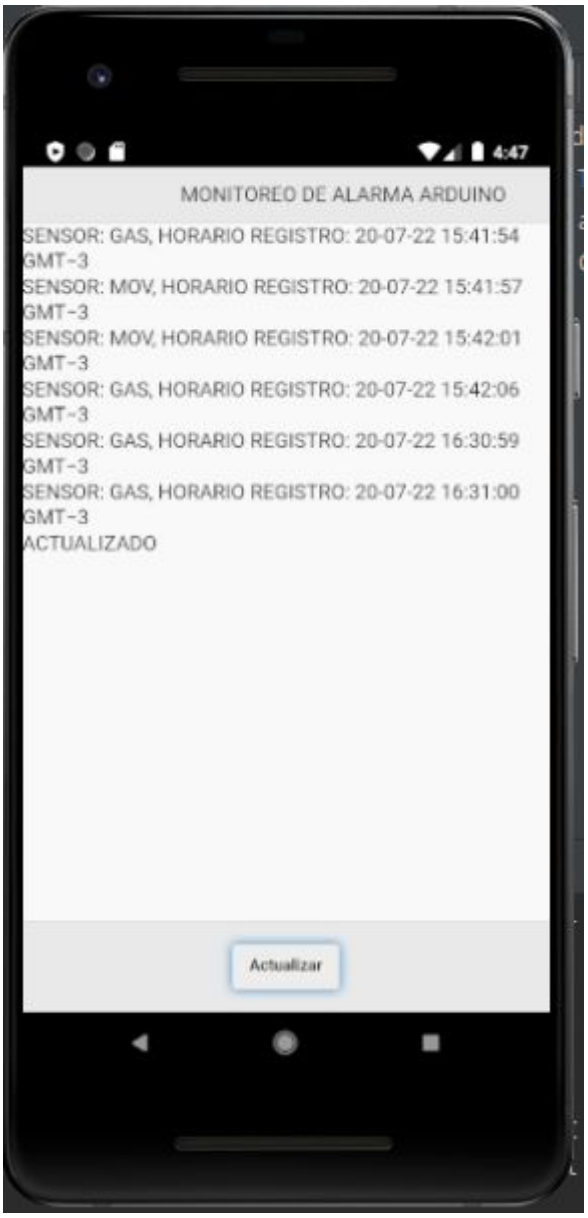
API Rest:
Desarrollamos una API Rest propia para manejar la información que se envía desde el módulo Wifi en Arduino. Este microservicio fue desarrollado en Springboot y utilizando una base de datos en memoria H2. La API dispone de dos métodos, el primero se encarga de guardar el id del sensor y la fecha en la que registró el movimiento en la base, el segundo método es de consulta y devuelve la lista de los movimientos guardados en la base.
El código del desarrollo se encuentra en GitHub y fue deployado en Heroku.

App Android:
Desarrollamos una aplicación en Apache Cordova para la plataforma Android. Se desarrolló pensando en realizar consultas desde el celular para saber si hubo algún evento en el lugar donde está configurada la alarma con los sensores. Esta aplicación consume la Api rest deployada en heroku y mencionada anteriormente. Los fuentes están en github en un repositorio público.

Código fuente

API Rest: https://github.com/pdpire/tparduino

App Android:
https://github.com/meportillo/SemArduino-App-Android-Alarma



Arduino N°1:

```
#include <IRremote.h>
#include <Wire.h>
#include <LiquidCrystal.h>


int RECV_PIN = 9; // Pin del receptor infrarrojo
IRsend irsend;
IRrecv irrecv(RECV_PIN);
decode_results results;

int led = 11;
int pot1 = A0;
int valor1 = 0;
```

```
int val = 0;
int lectura1 = 0;

int valAnt = 100;

int estado = 0;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);
  lcd.print("Presione Power");

  Wire.begin(); // Conexión al Bus I2C
  Serial.begin(9600); // Velocidad de conexión
  pinMode(led, OUTPUT);
  irrecv.enableIRIn();

}

void loop() {

  if (irrecv.decode(&results)) {
    Serial.println(results.value);
    switch (results.value) {

      //boton stop:
      case 16597183:
        lcd.clear();
        lcd.print("Alarma detenida");

        Wire.beginTransmission(2);
        Wire.write(99);
        Wire.endTransmission();

        break;

      //boton power
      case 16580863:
            estado = estado + 1;
        if(estado == 1){
          lcd.clear();
          lcd.print("Encendido");
        }
        else{
          lcd.clear();
          lcd.print("Apagado");
          estado = 0;
```

```
      }

      Wire.beginTransmission(2);
      Wire.write(50);
      Wire.endTransmission();

      break;
    }

    irrecv.enableIRIn(); //Reset
  }

  Wire.requestFrom(2, 1); // Le pide 10 bytes al Esclavo 2

  while (Wire.available()) // slave may send less than requested
  {

    val = Wire.read(); // Recibe byte a byte
    if (val != 0) {

      switch (val) {
      case 10:
        lcd.clear();
        lcd.print("Sensor MOV");
        break;
      case 11:
        lcd.clear();
        lcd.print("Sensor GAS");
        break;
      }

      valAnt = val;

    } else {
      //lcd.clear();

    }
  }

  Serial.println();
  Serial.println(val); // Cámbia de línea en el Serial Monitor.
  delay(500);

}
```

Arduino N°2:

```
#include <Wire.h>
#define NOTE_G3 196
#define NOTE_GS3 208

String ssid = "Simulator Wifi";
String password = "";
String host = "https://tp-arduino.herokuapp.com"; //"arduinoget.herokuapp.com"; // Open Weather Map API
const int httpPort = 80;
String uri = "/notification?sensorId=";
int isSetup = false;

int PIN_ECHO_1 = 6;
int PIN_TRIG_1 = 7;
int LED_SENMOV_1 = 10;

int LED_GAS = 11;

int flagCtrl_1 = 0;

long duration_1, duration_2, duration_3, cm_1, cm_2, cm_3;

int flagCtrl = 0;

int flagLed = 4;

int sensorGas = A1;
int sensorGasValue = 0;

int encendido = 0;

int setupESP8266(void) {
  Serial.begin(115200);
  Serial.println("AT"); // Serial connection on Tx / Rx port to ESP8266
  delay(10); // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 1;

  // Connect to Simulator Wifi
  Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");
  delay(10); // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 2;

  //Verify Ip Address
  Serial.println("AT+CIFSR");
  delay(100);
  String ret = Serial.readString();
```

```
  Serial.print(ret);

  // Open TCP connection to the host:
  Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPort);
  delay(50); // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 3;
  isSetup = true;
  return 0;
}

void sendRequest(String pinLed) {

  String httpPacket = "GET " + uri + pinLed + " HTTP/1.1\r\nHost: " + host  + "\r\n\r\n";
  int length = httpPacket.length();

  // Send our message length
  Serial.print("AT+CIPSEND=");
  Serial.println(length);
  delay(10); // Wait a little for the ESP to respond if (!Serial.find(">")) return -1;

  // Send our http request
  Serial.print(httpPacket);
  delay(10); // Wait a little for the ESP to respond
  if (!Serial.find("SEND OK\r\n")) return;

}

int melody[] = {NOTE_G3,NOTE_GS3};

int noteDurations[] = {
  4,
  4
};

void setup() {
  setupESP8266();

  pinMode(PIN_TRIG_1, OUTPUT);
  pinMode(PIN_ECHO_1, INPUT);

  pinMode(LED_SENMOV_1, OUTPUT);

  pinMode(LED_GAS, OUTPUT);

  Wire.begin(2); // Este Esclavo es el número 2
  Wire.onRequest(requestEvent); // Cuando el Maestro le hace una petición,
  Wire.onReceive(recibidoEvento); // realiza el requestEvent
```

```
}

void loop() {

  if(encendido != 0){
    updateSensores(LED_SENMOV_1, sensorGas);

    // Retraso entre mediciones para el correcto funcionamiento del sensor
    delay(250);
  }
}

void updateSensores(int pinLed, int sensorGas) {
  updateSensorMov(pinLed);
  updateSensorGas(sensorGas);
}

void updateSensorGas(int pinLed) {

  sensorGasValue = analogRead(pinLed);
  if (sensorGasValue > 125) {
    flagCtrl = 1;
    digitalWrite(LED_GAS, HIGH);
    sonar();

    if (isSetup) {
      sendRequest("GAS");
    }
    flagLed = 11;
  } else {
    if (flagCtrl == 1) {
      digitalWrite(LED_GAS, LOW);
      sonar();
    }

  }

}

void updateSensorMov(int pinLed) {
  // Primero, generar un pulso corto de 2-5 microsegundos.
  digitalWrite(PIN_TRIG_1, LOW);
  delayMicroseconds(5);
  digitalWrite(PIN_TRIG_1, HIGH);
  // Después de ajustar un nivel de señal alto, esperamos unos 10 microsegundos. En este punto
el sensor enviará señales con una frecuencia de 40 kHz.
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG_1, LOW);
```

```
  // Tiempo de retardo de la señal acústica en el sonar.
  duration_1 = pulseIn(PIN_ECHO_1, HIGH);
  // Ahora es el momento de convertir el tiempo a distancia
  cm_1 = (long)(duration_1 / 2) / 29.1;

  //delayMicroseconds(5);
  if (cm_1 < 330) {
    flagCtrl = 1;
    digitalWrite(pinLed, HIGH);
    sonar();

    if (isSetup) {
      sendRequest("MOV");
    }
    flagLed = pinLed;
  } else {

    if (flagCtrl == 1) {
      sonar();
    }
    digitalWrite(pinLed, LOW);
    flagLed = 0;
  }

}

void sonar() {
  for (int thisNote = 0; thisNote < 2; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);

    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
  }
}

// Esto es lo que envía cuando le hace la petición.
void requestEvent() {
  if (flagLed != 0) {
    Wire.write(flagLed); // Envía 10 bytes
  }
}

void recibidoEvento(int recepcion) {
  unsigned int pedido;
  while (Wire.available()) {
    pedido = Wire.read();
    if (pedido == 99) {
```

```
      flagCtrl = pedido;
    }

    if (pedido == 50) {
      if(encendido == 0){
        encendido = pedido;
      }
      else{
        encendido = 0;
      }
    }

    Serial.println(pedido);
  }
}
```

Referencias

https://aprendiendoarduino.wordpress.com/2017/09/13/uso-esp8266-con-arduino-puerto-serie/

http://slaxtrack.github.io/HC-SR04lib/a00004.html

https://javiergarciaescobedo.es/arduino/100-robot-smart-car/427-programacion-de-sensor-ultrasonidos-hc-sr04-con-arduino

https://proyectosconarduino.com/sensores/sensor-de-distancia-hc-sr04

https://www.arduino.cc/en/reference/wire

https://i2.wp.com/randomnerdtutorials.com/wp-content/uploads/2015/01/commands.png