Sistema de detección de robo

Grupo 17

Curso 2021 – 2022

Universidad Rey Juan Carlos

Claudia Recio Alcaide

Sergio Blazquez Teixeira

Daniel Alfonsel García

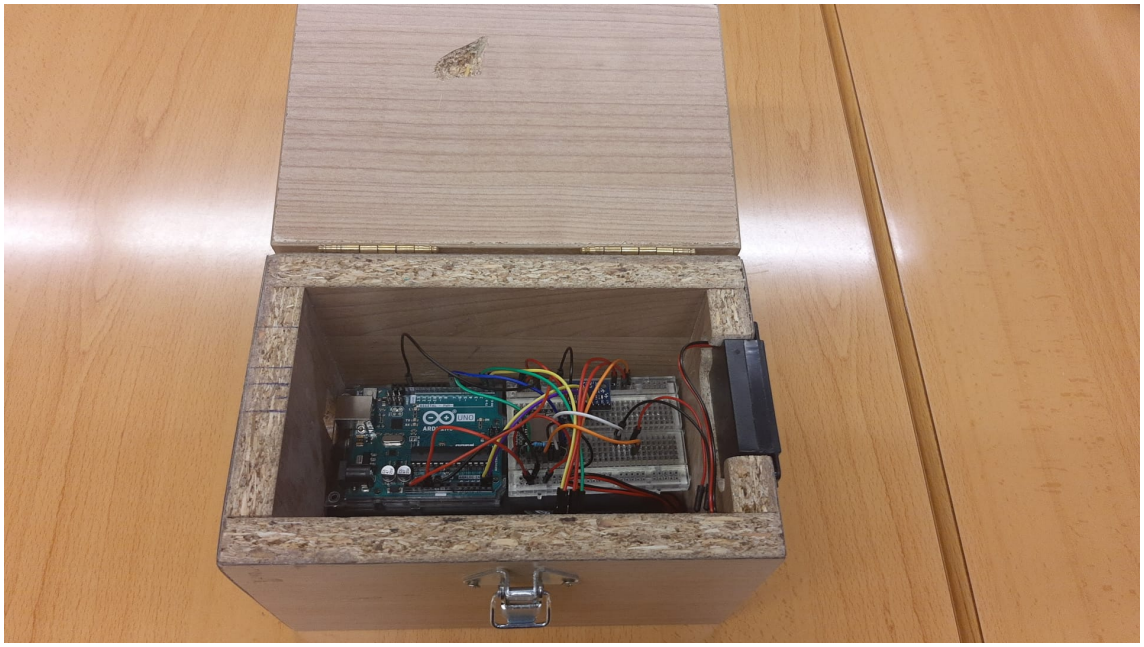
Índice

Propósito

La alarma

El proyecto consiste en una alarma que avisa al propietario de que le están robando una pertenencia. Esta alarma es una caja de madera cerrada (con posibilidad de ser abierta). En el interior de esta caja se encuentra el circuito y la placa Arduino, que se detallaran más adelante.

El funcionamiento es muy básico: conectar la placa Arduino a la powerbank que tiene debajo. Una vez conectados, se disponen de alrededor de 5 segundos (tiempo en el que se inicializan los módulos) para colocar la caja como se desee. Una vez pasados los 5 segundos, un movimiento en la caja hará sonar una alarma y enviará una notificación al móvil del propietario.



Hardware

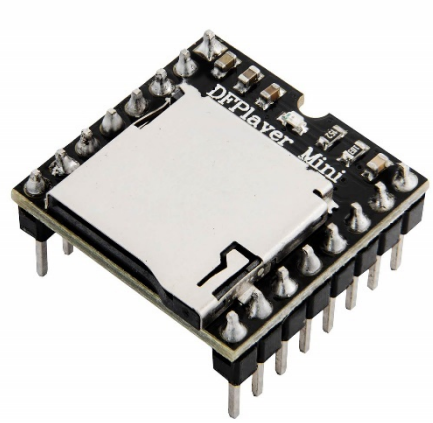
Aparte de la placa Arduino, se han utilizado un giroscopio, un módulo de sonido, una tarjeta SD, un altavoz y un módulo wifi. El desglose de precios es el siguiente:

Componente	Precio
Arduino Uno	14,9€
Giroscopio	2,83€
Módulo de sonido	8€
Altavoz	4,5€
Módulo WiFi	6,49€
Tarjeta SD	1,17€
TOTAL	37,8€

El giroscopio

El giroscopio se trata de un módulo MPU6050. En concreto el modelo GY-521 de AZDelivery. Este módulo es un sensor de orientación en el espacio y acelerómetro en los tres ejes cartesianos (X, Y, Z).

El módulo se comunica con el protocolo I2C/TWI, totalmente compatible con Arduino Uno mediante la librería *Wire.h* y los pines analógicos 4 y 5.



El módulo de sonido

El módulo de sonido se trata de un DFPlayer Mini de AZDelivery.

Este módulo facilita la reproducción de audio en el dispositivo. Se comunica con Arduino a través de comandos propios.

Contiene un lector de tarjetas miniSD donde almacenar los archivos de audio. Estos archivos pueden estar en los formatos .MP3, .WAV y .WMA.

El módulo Wifi

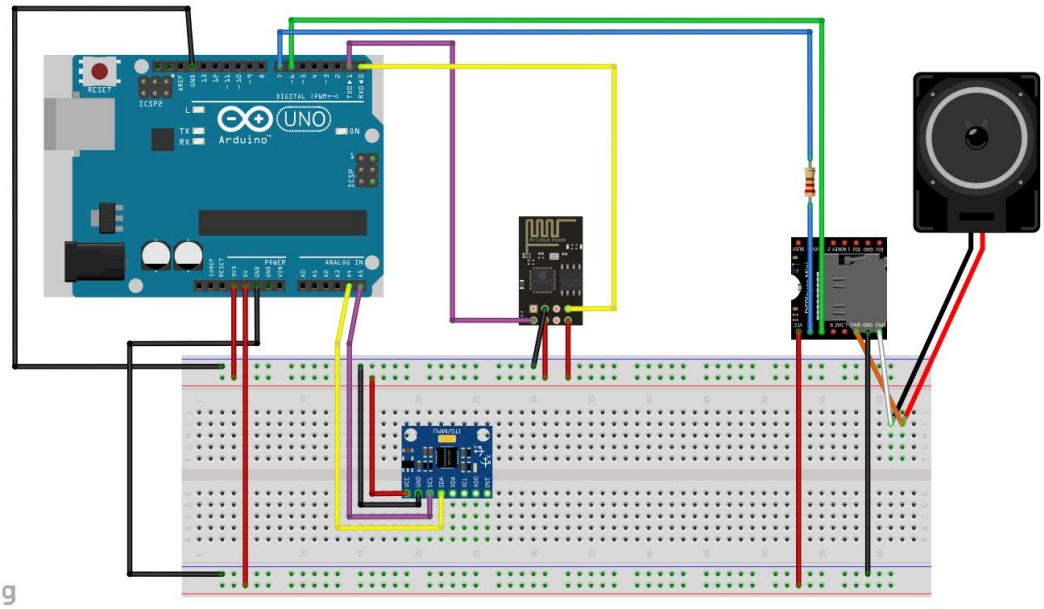
El módulo WiFi se trata de un ESP-8266. En concreto, es el modelo ESP8266-01S de AZDelivery.

El módulo tiene infinitas posibilidades, ya que al igual que Arduino, es reprogramable.

En este proyecto su función es establecer conexión con el servicio en línea *PushingBox* para que este mande una notificación al teléfono móvil al que se encuentra conectado.

Diagrama de conexiones

El diagrama de conexiones es el siguiente:



Software

Código WiFi

Como ya se ha mencionado, el módulo WiFi es reprogramable, por lo que tiene su propio código en memoria. El núcleo del código ha sido extraído de Geekstips.com.

Este proyecto es para utilizar el módulo ESP8266 como fuente de notificaciones para un sistema Android a través de las plataformas *PushingBox* y *PushingBullet*.

El código resultante es el siguiente:

```
#include <ESP8266WiFi.h>

String deviceId = "v316966B1F2622E0";
const char* logServer = "api.pushingbox.com";

const char* ssid = "MotorolaOneVision";
const char* password = "HolaMundo3";

bool primerAcceso = true;

void sendNotification(String message){
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  WiFiClient client;

  if (client.connect(logServer, 80)) {

    String postStr = "devid=";
    postStr += String(deviceId);
    postStr += "&message_parameter=";
    postStr += String(message);
    postStr += "\r\n\r\n";

    client.print("POST /pushingbox HTTP/1.1\n");
    client.print("Host: api.pushingbox.com\n");
    client.print("Connection: close\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(postStr.length());
    client.print("\n\n");
    client.print(postStr);
  }
  client.stop();
}
void setup() {
  pinMode(3,FUNCTION_3);
  pinMode(1,FUNCTION_3);
}
void loop() {
  while(true){
    if(digitalRead(3) == HIGH){
      if(!primerAcceso){
        sendNotification("¡Te estan robando!");
      } else{
      primerAcceso = false;
      }
    delay(1500);
    }
 }
}
```

Básicamente, el código se compone de una función encargada de establecer la conexión, la configuración de los pines (para controlar el módulo a través de señales de Arduino), y el loop, donde se comprobará si hay que enviar una notificación o no.

Código Arduino

El código de la placa Arduino es mas extenso que el de módulo WiFi. Se compone de declaraciones de librerías, constantes y diversos objetos que utilizan los distintos módulos. Acto seguido se declaran y definen las variables que hacen funcionar todo el sistema. El código es autoexplicativo gracias a los comentarios. Las únicas funciones a destacar son:

- **hayMov():** El giroscopio realiza dos mediciones separadas por un segundo. Si hay una diferencia significativa, devolverá el booleano *true*, indicando que se ha detectado movimiento.
- **activarModuloSonido():** Activa la reproducción del archivo MP3 que contiene la alarma y activa un loop para que esta no pare de sonar.
- **activarModuloWifi():** Escribe una señal alta en el pin 0 del Arduino. Esto será interpretado por el módulo WiFi como que debe enviar una notificación. Tras un segundo y medio

```
// ------------------------------------------------- Librerías --------------------------
-------------------------------------------
#include "SoftwareSerial.h" // Comunicación con el módulo de sonido
#include "Wire.h"           // Comunicación con el giroscopio (protocolo I2C)

// ------------------------------------------------- Constantes del modulo de sonido -----
-------------------------------------------------------------------------
#define Start_Byte        0x7E
#define Version_Byte      0xFF
#define Command_Length    0x06
#define End_Byte          0xEF
#define Acknowledge       0x01

// ------------------------------------------------- Variables del modulo de sonido ------
-------------------------------------------------------------------------
SoftwareSerial mySerial(6, 7);  //  RX, TX, comunicación con el modulo de sonido
byte receive_buffer[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
byte volume = 0x00;

// ------------------------------------------------- Variables del giroscopio-acelerómetro
-------------------------------------------------------------------------------
const int MPU_ADDR = 0x68; // I2C address of the MPU-6050. If AD0 pin is set to HIGH, the I2C
address will be 0x69.
int16_t accelerometer_x, accelerometer_y, accelerometer_z; // variables for accelerometer raw
data
int16_t gyro_x, gyro_y, gyro_z; // variables for gyro raw data
int offset[6] = {-3431, 1919, 1765, -9, 34, 71}; // Offset de calibracion para los valores
del giroscopio-acelerometro
```

```
// ------------------------------------------------------ Funciones del modulo de sonido ------
-------------------------------------------------------------------------
void execute_CMD(byte CMD, byte Par1, byte Par2) {
  //  Sends the command to the module

  //  Calculate the checksum (2 bytes)
  word checksum = -(Version_Byte + Command_Length + CMD + Acknowledge + Par1 + Par2);

  //  Build the command line
  byte Command_line[10] = { Start_Byte, Version_Byte,
                            Command_Length, CMD, Acknowledge,
                            Par1, Par2, highByte(checksum),
                            lowByte(checksum), End_Byte};

  //  Send the command line to the module
  for(byte k = 0; k < 10; k++) {
    mySerial.write(Command_line[k]);
  }
}

void set_eq(uint8_t eq) {
  // Sets the state of EQ
  execute_CMD(0x07, 0, eq); // Sets the EQ
  delay(100);
}

void set_volume(uint8_t volume) {
  // Sets the volume level
  execute_CMD(0x06, 0, volume); //  Set volume level
  delay(100);
}

void module_init() {
  // Resets the module and sets the EQ state,
  // volume level, and plays first song on the
  // storage device
  execute_CMD(0x0C, 0, 0); // reset the module
  delay(1000);
  delay(100);
  delay(100);
  set_eq(0);
  set_volume(0x19);
}

void play_first() {
  // Plays first song on the storage device
  execute_CMD(0x03, 0, 1); // Play first song
  delay(100);
}

void loop_current() {
  // loops current playing song
  execute_CMD(0x19, 0, 0);
  delay(100);
}

void activarModuloSonido(){
  // Reproduce el sonido de alarma y lo pone en bucle
  play_first();
  loop_current();
}
```

```
// ------------------------------------------------- Funciones del giroscopio-acelerómetro ------------------------------------------------------------------------
int * activarSensores(){
  static int valores[6]; //Almacen de los valores del giroscopio y acelerómetro (Eje X, Eje Y, Eje Z de ambos sensores)
  //Activación de los sensores
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H) [MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2, p.40]
  Wire.endTransmission(false); // the parameter indicates that the Arduino will send a restart. As a result, the connection is kept active.
  Wire.requestFrom(MPU_ADDR, 7*2, true); // request a total of 7*2=14 registers

  //Valores del acelerometro
  valores[0] = Wire.read()<<8 | Wire.read(); // reading registers: 0x3B (ACCEL_XOUT_H) and 0x3C (ACCEL_XOUT_L)
  valores[1] = Wire.read()<<8 | Wire.read(); // reading registers: 0x3D (ACCEL_YOUT_H) and 0x3E (ACCEL_YOUT_L)
  valores[2] = Wire.read()<<8 | Wire.read(); // reading registers: 0x3F (ACCEL_ZOUT_H) and 0x40 (ACCEL_ZOUT_L)
  //Valores del giroscopio
  valores[3] = Wire.read()<<8 | Wire.read(); // reading registers: 0x43 (GYRO_XOUT_H) and 0x44 (GYRO_XOUT_L)
  valores[4] = Wire.read()<<8 | Wire.read(); // reading registers: 0x45 (GYRO_YOUT_H) and 0x46 (GYRO_YOUT_L)
  valores[5] = Wire.read()<<8 | Wire.read(); // reading registers: 0x47 (GYRO_ZOUT_H) and 0x48 (GYRO_ZOUT_L)

  for(int i=0; i<6; i++){
    valores[i] = valores[i] + offset[i];
  }

  //Retornamos el puntero
  return valores;
}

bool hayMov (){
  //hacer dos medidas y compararlas
  bool movimientoDetectado = false;
  int *p;
  int i;
  int medida1[6];
  int medida2[6];
  //primera medida
  p = activarSensores();
  for(i =0; i < 6; i++){ //guardamos los resultados en un array
    medida1[i] = *(p+i);
  }

  delay(1000); //tiempo de espera entre medidas en ms

  //segunda medida
  p = activarSensores();
  for(i =0; i < 6; i++){ //guardamos los resultados en un array
    medida2[i] = *(p+i);
  }

  //comparamos ambos arrays
  //teniendo en cuenta las respuestas de los sensores, 1500 parece un margen razonable para detectar movimiento

  for(i =0; i<6; i++){
    int diff = abs(medida1[i] - medida2[i]);
    if (diff > 1000){ //el valor absoluto de la resta debe permanecer por debajo de n
      movimientoDetectado = true;
    }
  }
  return movimientoDetectado;
}
```

```
// ------------------------------------------------- Funciones del modulo WiFi -----------
-----------------------------------------------------------------
void activarModuloWiFi(){
  // Se manda una señal al modulo WiFi para que mande notificaciones al movil. Se introduce
un delay para asegurar la sincronización con el mismo
  // al menos dos veces
  digitalWrite(0, HIGH);
  delay(3100);
  digitalWrite(0, LOW);

}

void setup() {
  // Inicialización del modulo WiFi
  pinMode(0,OUTPUT);
  digitalWrite(0,LOW);

  // Inicialización del módulo de sonido y de su canal de comunicación
  mySerial.begin(9600);
  delay(1000);
  module_init();

  // Inicialización del giroscopio-acelerómetro
  Wire.begin();
  Wire.beginTransmission(MPU_ADDR); // Begins a transmission to the I2C slave (GY-521 board)
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  delay(5000);
}

void loop() {
  // Si se detecta movimiento en el giroscopio-acelerómetro, se activan tanto el modulo de
sonido (para activar el altavoz)
  // como el módulo Wifi, y se bloquea el programa en un bucle infinito, ya que para
terminarlo, hay que apagar el arduino
  // por completo
  if(hayMov()){
    activarModuloSonido();
    activarModuloWiFi();
    while(true);
  }

}
```

Problemas y soluciones

El giroscopio

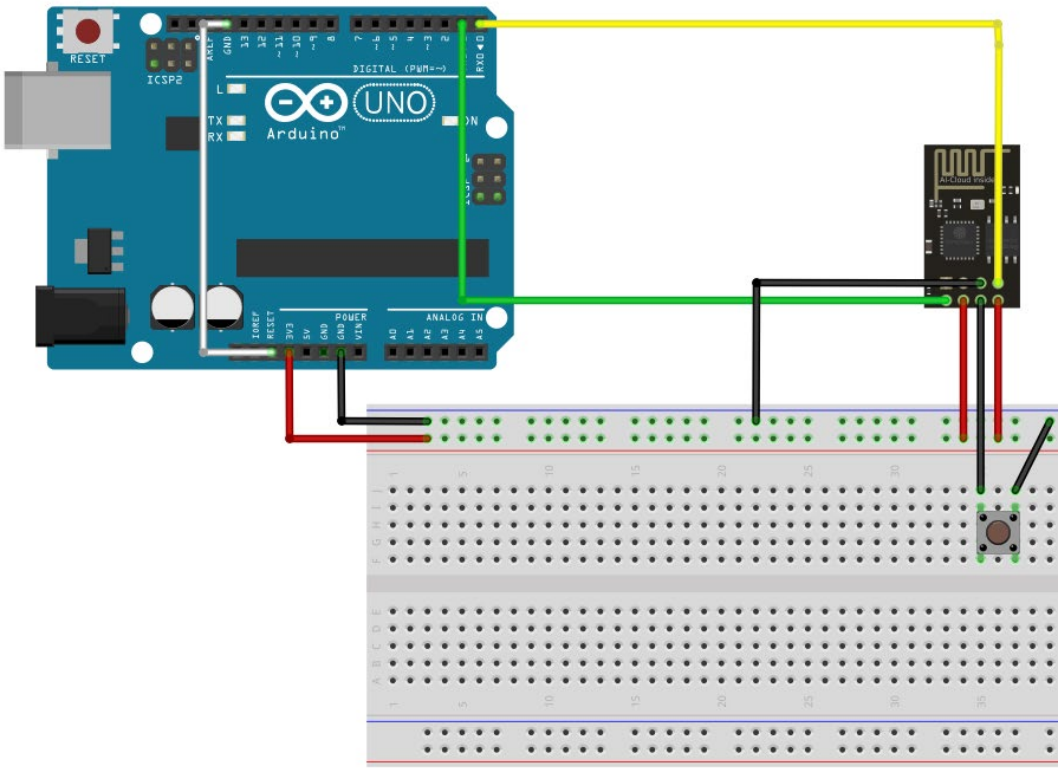
La conexión del giroscopio con el Arduino no se realizaba correctamente. Las pastillas incluidas con el módulo no hacían bien la conexión, al igual que los cables

Al principio no sabíamos que estaba sucediendo, pero más tarde nos dimos cuenta de que si movíamos los cables o la pastilla, el giroscopio comenzaba a funcionar correctamente, por lo que optamos por soldar la pastilla al módulo. Esta solución fue la adecuada, y el giroscopio comenzó a funcionar como se esperaba.

El módulo Wifi

Por su parte, el módulo Wifi nos dio varios problemas. Uno de ellos fue que, en el datasheet del mismo, no se explicaba como conectarle en modo programación utilizando la placa Arduino como puente. En el datasheet se explicaba cómo utilizar componentes adicionales para programarlo.

Al final, investigando encontramos una conexión, pero esta funcionaba de forma aleatoria, es decir, algunas veces si conseguíamos actualizar el código del módulo, y otras veces no. El botón se debía pulsar antes de subir el código al módulo, para resetearlo manualmente. Además, la señal recetó de la placa debía estar conectada a tierra, ya que si no el IDE de Arduino era incapaz de conectar con el ESP8266.



Con una investigación más exhaustiva, descubrimos que para utilizar la placa Arduino Uno como puente, esta tenía que tener en su memoria un programa en blanco (es decir, setup y loop vacíos).

Al aplicar estas dos soluciones en conjunto (las conexiones y el programa en blanco en la memoria del Arduino) el módulo se podía programar.

Luego hubo problemas con la activación del mismo. Este se activaba cuando le llegaba la corriente, pero este no era el comportamiento deseado: este debía actuar cuando el giroscopio detectase algún cambio. Se intentó utilizar una puerta lógica AND con un divisor de tensión a su salida, pero esta idea no funcionaba.

Al final se optó por intentar controlarlo a través de señales por el Arduino, pero ninguno de los pines del módulo eran buenos candidatos a ser usados para esta función (excepto algunos pines GPIO, pero para acceder a ellos se requería soldar piezas muy pequeñas). Al final se optó por utilizar el pin de recepción (RX) de la placa Arduino Uno y el módulo como vía de comunicación. El problema estaba en que al incializarlo, se enviaba una notificación fantasma al teléfono móvil. Este problema fue solucionado a través de código y un delay de 1.5 segundos entre observaciones de las señales (como se observa en el loop del código Wifi).

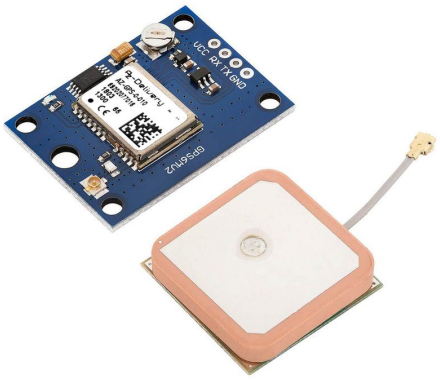
Mejoras futuras

Diseño exterior y disposición

Una mejora clara del proyecto es su diseño. Podría ser mucho mas compacto, y además se podría hacer camuflado para el entorno donde se vaya a utilizar o buscar alguna manera de anclarlo para que sea más complejo de extraer.

Módulo GPS

Otra adición al proyecto podría ser un módulo GPS, que te indique en todo momento en que posición se encuentra. Además, esto abre la posibilidad de distintos modos, como por ejemplo de sonido apagado, pero GPS activo, para que la persona que ha robado no sepa que está siendo geolocalizada en todo momento.



Bibliografía

Documentación del fabricante (Datasheet) – Información general de uso

geektips.com – Uso de ESP32866 como fuente de notificaciones

Stackoverflow.com – Ayuda generalizada