FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
HUGO VEGA HUERTA
INTELIGENCIA ARTIFICIAL

Sistema de Simulación de un Timbre Inteligente Basado en Arduino y la Creación de una Aplicación en Android

Giuseppe Jefferson Cordova Silva, Ivan Paolo De La Cruz Guillen, Manuel Hernan Gonzales Rojas, Jose Luis Perez Grados, Hector Imanol Rafael Javier, Karen Antonia Rojas Hurtado, Alejandro Paul Torres Espinoza, Jamie Edinso Vilchez Giraldo

Universidad Nacional Mayor de San Marcos
Facultad de Ingeniería de Sistemas e Informatica

giuseppe.cordova@unmsm.edu.pe, ivan.delacruz1@unmsm.edu.pe, manuel.gonzales13@unmsm.edu.pe, jose.perez45@unmsm.edu.pe, hector.rafael@unmsm.edu.pe, karenantonia.rojas@unmsm.edu.pe, alejandro.torres4@unmsm.edu.pe, jamie.vilchez@unmsm.edu.pe

RESUMEN

En este proyecto, se desarrolló un Sistema Inteligente de Simulación de un timbre utilizando Arduino. El objetivo principal era crear un mecanismo de alerta automatizado y eficiente. El sistema se basa en un sensor de lluvia y un sensor Bluetooth que están conectados a un Arduino.Cuando el sensor de lluvia detecta presión, envía una señal al Arduino. Este, a través del sensor Bluetooth, se comunica con un dispositivo móvil como un celular. Cuando se activa el sensor de lluvia, el Arduino envía una señal al dispositivo móvil para realizar una llamada de emergencia. Esta llamada se realiza de forma automática y sin intervención humana, proporcionando una respuesta inmediata.Este Sistema Inteligente de Simulación de un timbre con Arduino garantiza una alerta temprana y ayuda a las personas a tomar medidas rápidas y adecuadas en caso de que estas se requieran, mejorando así la seguridad y la capacidad de respuesta en situaciones de emergencia.

Palabras clave: Sistema, GPS, Arduino, huerto, coordenadas, automatizar.

ABSTRACT

In this project, a Smart Bell Simulation System was developed using Arduino. The main goal was to create an automated and efficient alert mechanism. The system is based on a rain sensor and a Bluetooth sensor that are connected to an Arduino.When the rain sensor detects pressure, it sends a signal to the Arduino. This, through the Bluetooth sensor, communicates with a mobile device such as a cell phone. When the rain sensor is activated, the Arduino sends a signal to the mobile device to make an emergency call. This call is made automatically and without human intervention, providing an immediate response.This Smart Bell Simulation System with Arduino ensures early warning and helps people to take quick and appropriate actions in case these are required, thus improving safety and responsiveness in emergency situations.

Key words: System, GPS, Arduino, garden, coordinates, automate.

1. Introducción

En un mundo cada vez más conectado y automatizado, la integración de la tecnología en la vida cotidiana se ha vuelto esencial. En este contexto, el proyecto que presentamos se centra en el desarrollo de un Sistema Inteligente de Simulación de Timbre con Arduino. Este proyecto combina la versatilidad de Arduino como plataforma de hardware con una aplicación móvil desarrollada en Android Studio para crear un sistema de timbrado virtual. El sistema permite a los usuarios activar un timbre simulado a través de sus dispositivos móviles, lo que desencadena una serie de eventos, como la activación de un LED y la posibilidad de realizar una llamada telefónica. A lo largo de este artículo, exploraremos en detalle la implementación de este proyecto, desde el código en Arduino hasta la interfaz de usuario en Android Studio, destacando su funcionalidad y el potencial de aplicaciones en el ámbito de la automatización del hogar y la comodidad del usuario.

2. Fundamentación Teórica

2.1 Definición del Arduino Uno

El Arduino Uno es una placa de desarrollo de hardware de código abierto que se ha convertido en un pilar en el mundo de la electrónica y la informática física. Diseñado para ser asequible y fácil de usar, Arduino Uno ofrece una plataforma versátil para la creación de proyectos interactivos y la prototipación de dispositivos electrónicos. Desde su lanzamiento en 2005, Arduino ha desempeñado un papel fundamental en la promoción de la innovación y la educación tecnológica.

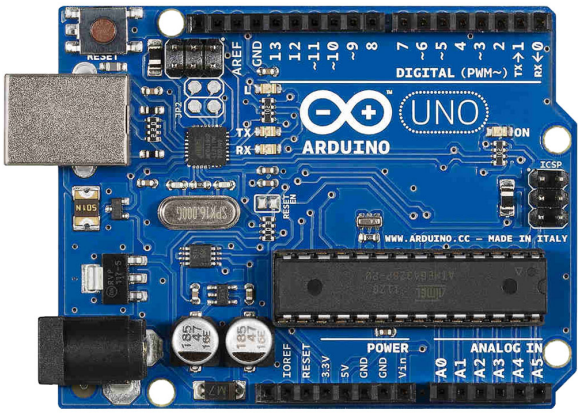


Figura N° 1. Arduino Uno

Arquitectura y Componentes Clave de Arduino Uno

Microcontrolador ATMega328P: El Arduino Uno se basa en el microcontrolador ATMega328P de 8 bits. Este microcontrolador es el cerebro del sistema y genera el código que especifica el comportamiento del programa. Tiene 32 KB de memoria flash, 2 KB de RAM y 1 KB de EEPROM.

Entrada/Salida de pines (E/S): El Arduino Uno proporciona un conjunto de pines de entrada/salida digitales y analógicas que permiten la conexión de sensores, actuadores y otros dispositivos electrónicos. Estas claves son importantes para comunicarse con el entorno y los dispositivos de control.

Entorno de desarrollo integrado (IDE): Arduino proporciona un entorno de código abierto fácil de usar que facilita la escritura, descarga y edición de código. El IDE se

basa en el lenguaje C/C++ y se puede utilizar en múltiples plataformas.

USB y Comunicación Serial: El Arduino Uno cuenta con un puerto USB que permite la programación y comunicación con una computadora. También admite comunicación en serie, lo que facilita la comunicación con otros dispositivos y microcontroladores.

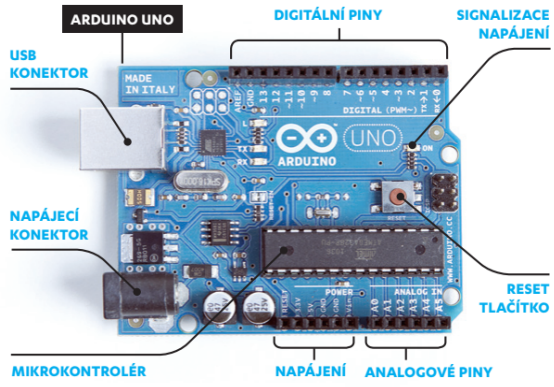


Figura N° 2. Componentes de Arduino Uno

2.2 Sensor de LLuvia

Un sensor de lluvia compatible con Arduino es un dispositivo que se utiliza para detectar la presencia o ausencia de lluvia en el ambiente. El sensor normalmente está diseñado para usarse con placas Arduino y otras plataformas de desarrollo y se basa en el principio de una resistencia de detección.

El rendimiento de este sensor depende de los cambios en la resistencia de su superficie cuando se expone a la lluvia. Cuanta más agua caiga sobre la batería, menor será su potencial. Normalmente, la resistencia de un sensor es mayor cuando está seco, pero menor cuando está mojado por la lluvia. La diferencia de resistencia eléctrica se utiliza para determinar la disponibilidad e intensidad de las precipitaciones.

Este sensor normalmente se conecta a una placa Arduino a través de su pin de entrada y utiliza el convertidor analógico a digital (ADC) de Arduino para medir la resistencia. Puede utilizar el código Arduino para calcular el valor de resistencia e interpretarlo para determinar si está lloviendo y cuánto.

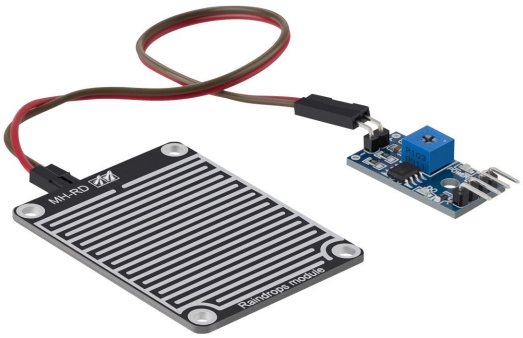


Figura N° 3. Sensor de Lluvia

Los sensores de lluvia compatibles con Arduino tienen una variedad de aplicaciones, incluidos sistemas de riego automático que pueden detener el riego cuando se detecta lluvia, sistemas de control del clima doméstico que registran fallas de lluvia y sistemas que cierran ventanas o toldos cuando se activan. lluvia. Estos sensores proporcionan una forma sencilla y eficaz de detectar las condiciones meteorológicas y controlar los equipos en respuesta a las precipitaciones.

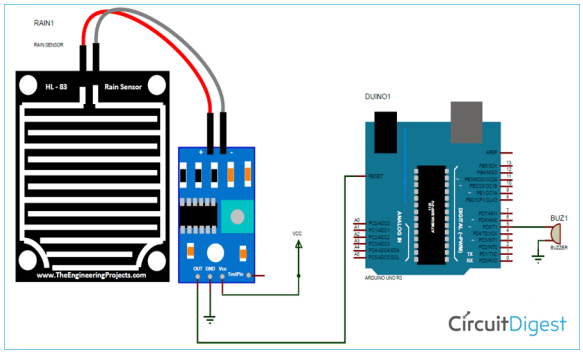


Figura N° 4. Conexión del Sensor de lluvia

2.3 Definición de Módulo Bluetooth hc-06

El módulo Bluetooth HC-06 es un pequeño dispositivo electrónico que se utiliza para habilitar la comunicación inalámbrica Bluetooth en una variedad de aplicaciones y proyectos. Este módulo se destaca por su simplicidad y facilidad de uso, lo que lo convierte en una opción popular para aquellos que desean agregar capacidades de conectividad Bluetooth a sus creaciones electrónicas sin requerir un profundo conocimiento técnico.

El HC-06 se basa en la versión 2.0 del estándar Bluetooth y opera en la banda de frecuencia de 2.4 GHz. Su principal función es actuar como un "esclavo" (slave) en una conexión Bluetooth. Esto significa que puede emparejarse con un dispositivo "maestro" (master), como un teléfono inteligente o una computadora, para establecer una comunicación inalámbrica bidireccional.

Una característica clave del HC-06 es su capacidad de comunicación a través de una interfaz UART (Universal Asynchronous Receiver/Transmitter). Esto permite la transmisión y recepción de datos mediante pines específicos para la transmisión (TX) y la recepción (RX). Esta comunicación UART simplifica la integración del módulo Bluetooth en proyectos, ya que se asemeja a la comunicación serie convencional.

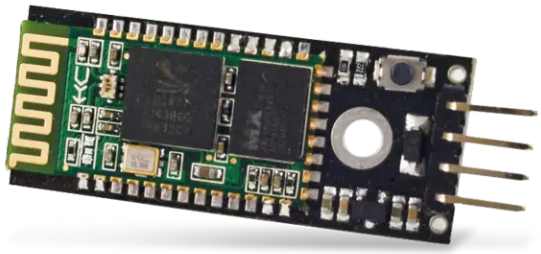


Figura N° 5. Modulo Bluetooth HC-06

Además, el HC-06 generalmente se suministra con una contraseña y un nombre de dispositivo Bluetooth preconfigurado, lo que facilita su uso inicial. Sin embargo, estos valores se pueden modificar para personalizar la identificación y la seguridad del módulo según las necesidades específicas de cada proyecto.

3. Metodología

3.1. Codificación en Arduino IDE

```
#include <SoftwareSerial.h>
#include <pt.h>
SoftwareSerial BTSerial(2, 3);
int pinPul =4;
 int intentos = 0;
 int cuenta =0;
 int cuenta2 =0;
int pinLluvia =A0;
int pinPul2 =5;
int pinLed= 6;
struct pt hilo1;
struct pt hilo2;


void setup() {
```

```
  Serial.begin(9600);
  BTSerial.begin(9600);
}

void loop() {
  tarea1(&hilo1);
  tarea2(&hilo2);
  int valor=analogRead(pinLluvia);

  if (valor<950){
    intentos++;
  }
  Serial.println(intentos);
  Serial.println(cuenta);
   Serial.println(cuenta2);
  Serial.print("Valor = ");
  Serial.println(valor);
  if(intentos == 3){
    if(cuenta < 2){
      BTSerial.write("X");
      cuenta =0;
      intentos = 0;
      BTSerial.println();
    }
  }

  if (BTSerial.available()) {
    Serial.write(BTSerial.read());
  }

  if (Serial.available()) {
    BTSerial.write(Serial.read());
  }
  delay(2500);
}

void tarea2(struct pt *pt) {
    PT_BEGIN(pt);
    //Control de puerta
    pinMode(pinPul,INPUT);

    static long t = 0;
    if(digitalRead(pinPul) ==  HIGH){
      cuenta++;
      t = millis();
    PT_WAIT_WHILE(pt, (millis()-t)<1000);
     }
     PT_END(pt);
  }

void tarea1(struct pt *pt) {
    PT_BEGIN(pt);
    //Control de puerta
    pinMode(pinPul2,INPUT);
    pinMode(pinLed,OUTPUT);
    static long t = 0;
    if(digitalRead(pinPul2) ==  HIGH){
      cuenta2++;
      t = millis();
    PT_WAIT_WHILE(pt, (millis()-t)<1000);
      if(cuenta2 == 3){
        digitalWrite(6,HIGH);
        t = millis();
    PT_WAIT_WHILE(pt, (millis()-t)<2000);
        digitalWrite(6,LOW);
        cuenta2 =0;
      }
    }
    PT_END(pt);
}
```

3.2. Codificación del Layout de la app en Android Studio

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.Constrain
tLayout
xmlns:android="http://schemas.android.com
/apk/res/android"
xmlns:app="http://schemas.android.com/apk
/res-auto"
xmlns:tools="http://schemas.android.com/to
ols" android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<LinearLayout
android:layout_width="0dp"
android:layout_height="0dp"
android:orientation="vertical"
```

```
app:layout_constraintBottom_toBottomOf="
parent"
app:layout_constraintEnd_toEndOf="parent
"
app:layout_constraintStart_toStartOf="paren
t"
app:layout_constraintTop_toTopOf="parent"
>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal">

<Button
android:id="@+id/idBtnOnBT"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="Activar" />

<Space
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1" />
<Button android:id="@+id/idBtnOffBT"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1" android:text="
Desactivar" />
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal">

<Button
android:id="@+id/idBtnDispBT"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Dispositivos BT" />

<Spinner
android:id="@+id/idSpinDisp"
android:layout_width="match_parent"
android:layout_height="50dp"
android:contentDescription="dispositivos"
/>

</LinearLayout>

<Button
android:id="@+id/idBtnConect"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Conectar" />

<TextView
android:id="@+id/textView"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="TEXTO A ENVIAR" />

<EditText
android:id="@+id/idTextOut"
android:layout_width="match_parent"
android:layout_height="50dp"
android:ems="10" android:hint="Texto a
Enviar"
android:inputType="textPersonName" />

<Button
android:id="@+id/idBtnEnviar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Enviar Texto" />

<TextView
android:id="@+id/txtrecibido"
android:layout_width="match_parent"
android:layout_height="116dp"
android:text="Mensajes recibidos:" />

</LinearLayout>
</androidx.constraintlayout.widget.Constrai
ntLayout>
```

4. Resultados

Se logró desarrollar timbre inteligente con Arduino que utiliza una aplicación móvil desarrollada en Android Studio para establecer una conexión Bluetooth con el módulo Arduino. El sistema permite activar un timbre virtual a través de la aplicación móvil, que envía una señal al Arduino, que interpreta dicha señal como un evento de timbrado y activa un LED en respuesta. Además, el Arduino monitorea un sensor de lluvia y cuenta los intentos de activación del timbre, respondiendo de manera adecuada cuando se alcanza un cierto número de intentos. El sistema está diseñado para simular una situación de timbrado y activar una llamada telefónica cuando se cumple una condición específica.

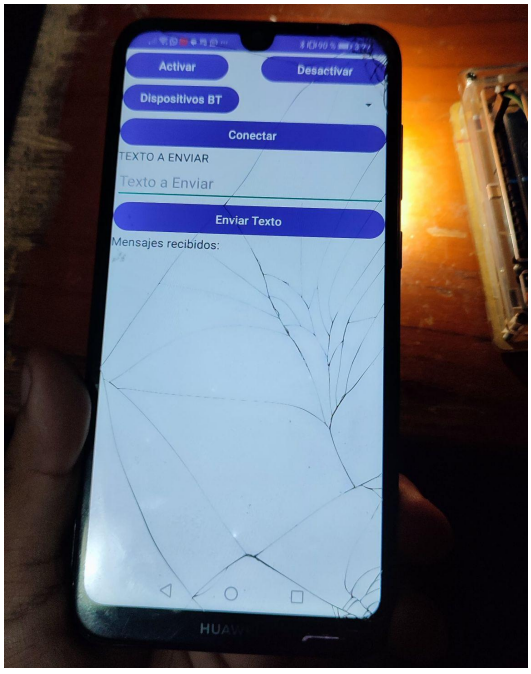
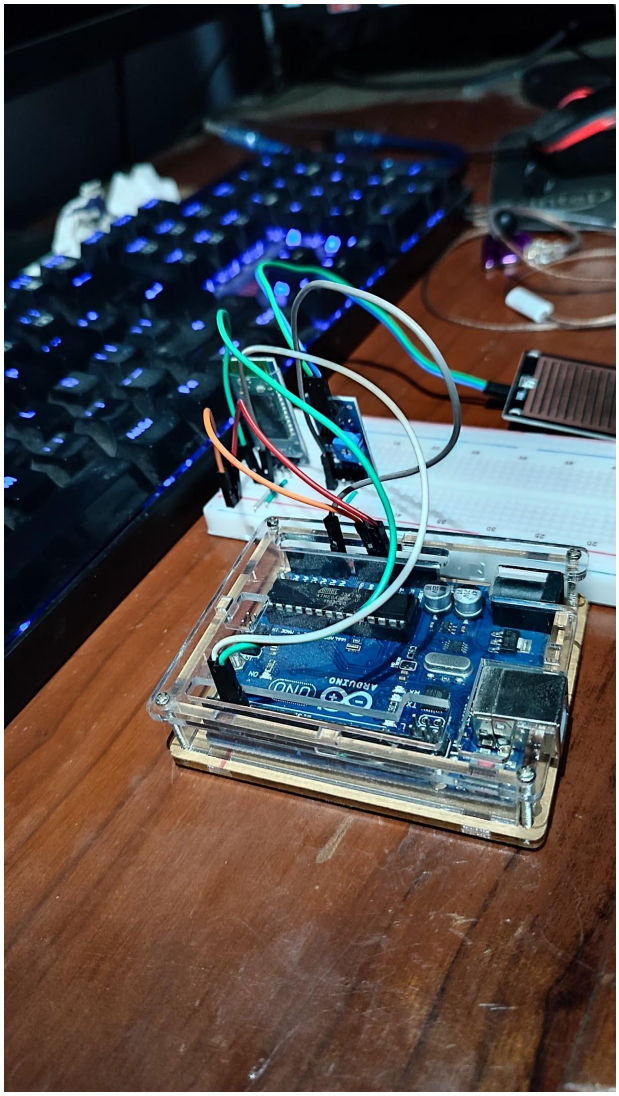
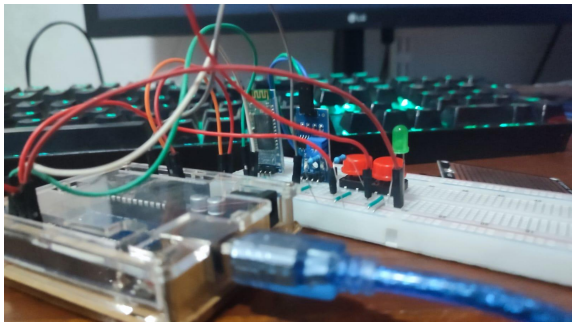


Figura N° 6. Interfaz de la App Móvil





5. Conclusiones

El desarrollo de este Sistema Inteligente de Simulació

n de timbre con Arduino y la interfaz móvil en Android Studio demuestra

la capacidad de integrar tecnología de hardware y software para crear soluciones prácticas. El proyecto ofrece una forma novedosa de simular un timbre real y controlar un dispositivo desde una aplicación móvil. Además, demuestra la utilidad de la comunicación Bluetooth para la interacción entre dispositivos. Este sistema tiene aplicaciones potenciales en la automatización del hogar, la seguridad y la comodidad del usuario. El trabajo futuro podría incluir mejoras en la interfaz de usuario, la expansión de las capacidades del Arduino y la implementación de más características de control remoto.

1. Referencias

Arduino, S. A. (2015). Arduino. *Arduino LLC*, *372*.

Uno, A., & Uno, G. (1999). Arduino. *línea]. Available: http://arduino. cc/en/pmwiki. php*.

Mucientes San José, D. (2021). Implementación de un entorno de comunicación Bluetooth basado en el módulo HC-06.