MANUAL TÉCNICO

DISEÑO DESARROLLO E IMPLEMENTACION DE HARDWARE Y APLICATIVO MOVIL PARA LA CONSTRUCCIÓN DE UNA MASCARILLA INTELIGENTE

DAVID SEBASTIÁN BOWEN ALCÍVAR

MARCOS NEPTALÍ NUÑEZ CUJI

ANGELO SANCHEZ CONDO

ESCUELA SUPERIOR POLITÉCNICA
DEL LITORAL
APLICACIONES MÓVILES Y
SERVICIOS TELEMÁTICOS
2022

Contenido

Presentación

El siguiente manual guiará a los usuarios que harán soporte al sistema, el cual les dará a conocer los requerimientos y la estructura para la construcción del sistema, en el desarrollo de hardware y aplicativo móvil conectados mediante una base de datos en la nube, el cual muestra las herramientas necesarias para la construcción y la funcionalidad del sistema.

Objetivo

Informar y especificar al usuario la estructura y conformación del sistema con el fin de que puedan hacer soporte y modificaciones o actualizaciones al sistema en general.

Procesos

Procesos de entrada

- Hardware

Datos de distancia (Sensor ultrasónico).
Datos de tiempo (Módulo RTC)
Señal de apertura y cierre de mascarilla (Pulsador)

- Aplicativo móvil

Ingresar al aplicativo móvil (acceso).
Registrar datos para el registro de usuarios (clientes).

Procesos de salida

- Hardware

Sistema de Alarma (Buzzer y Leds)
Sistema de apertura y cierre (Motor Stepper)

- Software

Datos de cantidad de personas (app)
Datos de tiempo de cercanía (app)
Datos de estado de batería (app)

Herramientas utilizadas para el desarrollo

JAVA

El lenguaje de programación de Java es una herramienta de desarrollo orientada a objetos, fue diseñado para que no dependieran en muchas implementaciones, el cual permite a los desarrolladores ejecutar en cualquier dispositivo sin necesidad de recompilar el código, el cual se considera multiplataforma.

Firebase

Es una plataforma ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios y dando resultado también a la obtención de una mayor monetización.

ANDROID STUDIO

Es el entorno de desarrollo oficial de Android, reemplazando el lenguaje de programación de eclipse, el entorno de Android Studio está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux en especial para el desarrollo de Android.

DISEÑO

Diseño de Hardware

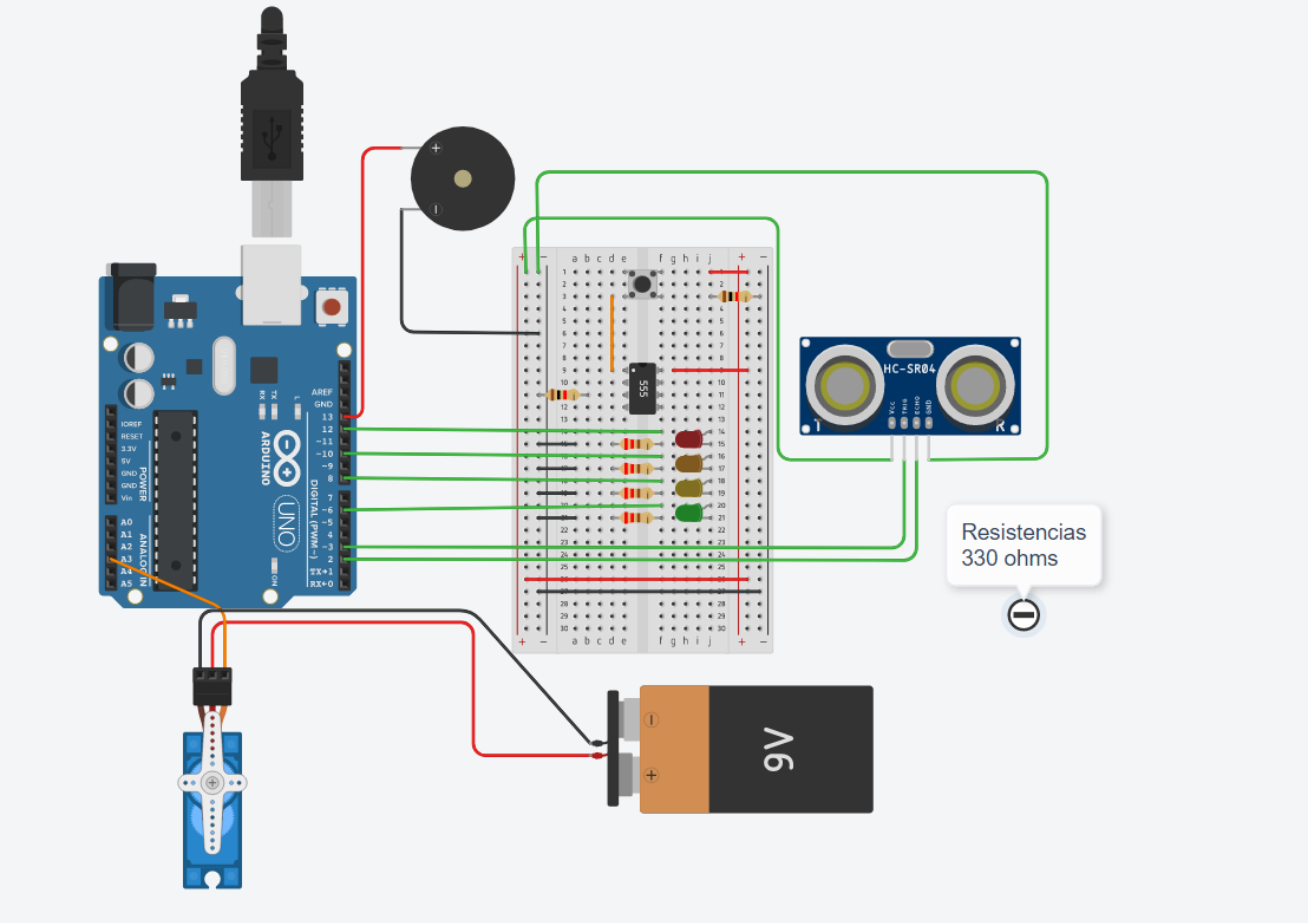

Ilustración 1. Diseño de circuito

Construcción y conexiones

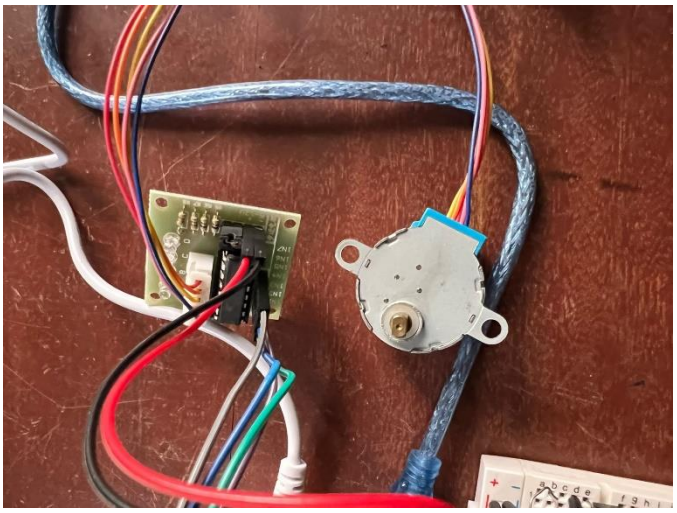

Ilustración 2. Motor Stepper

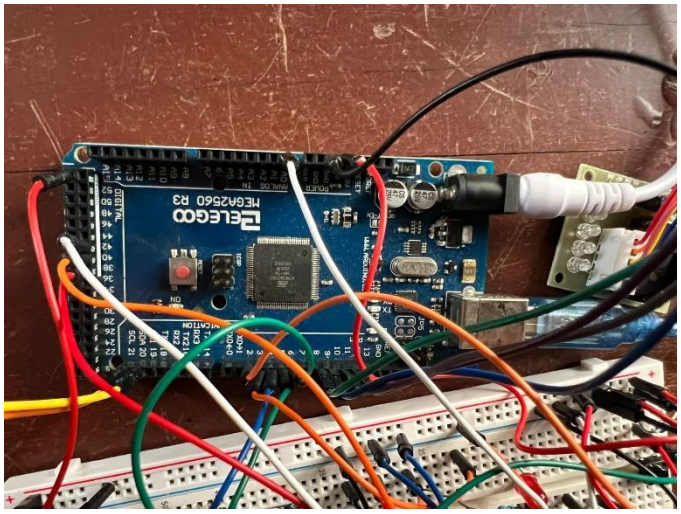

Ilustración 3. Arduino

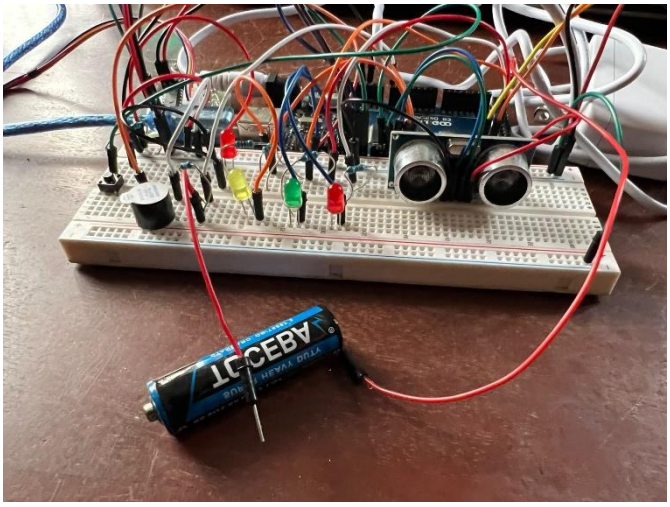

Ilustración 4. Sistema de alarma con Buzzer y Leds

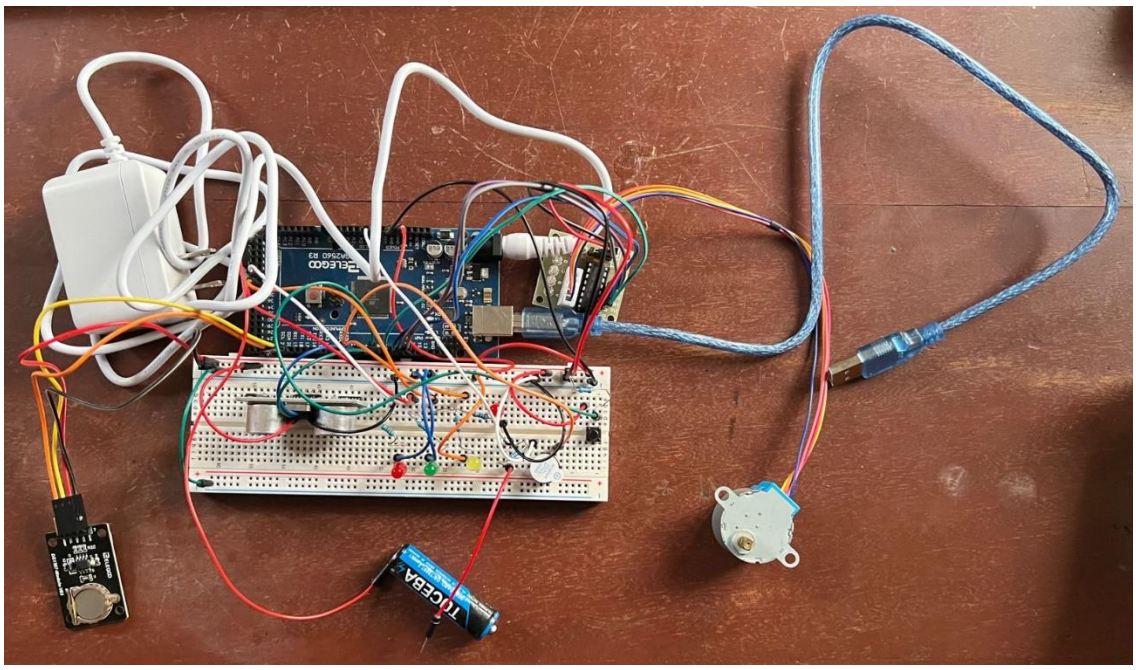

Ilustración 5. Conexión completa

Diagrama de Proyecto

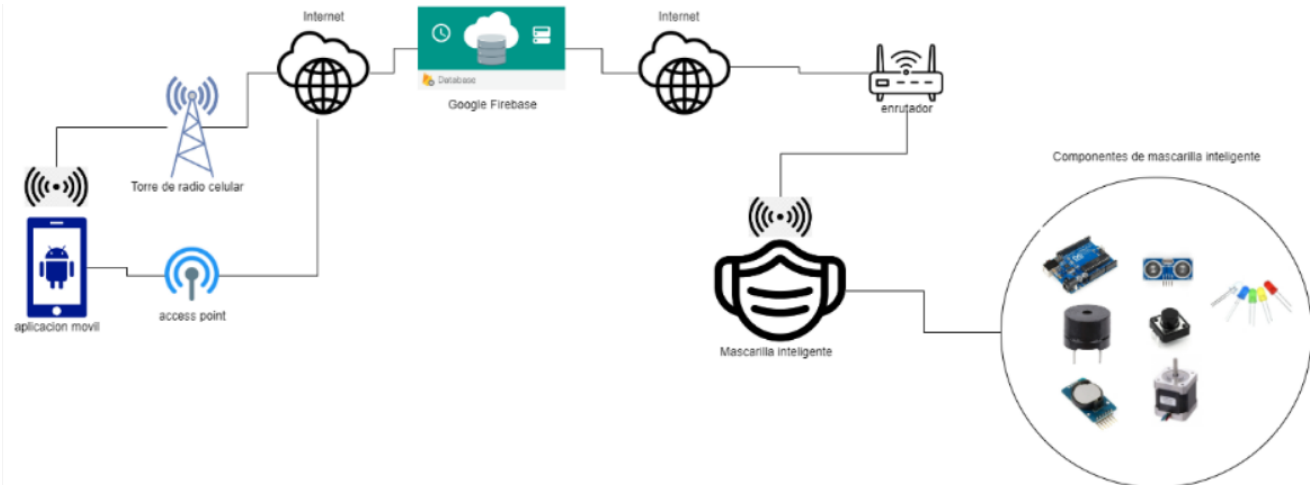

Ilustración 6. Diagrama de Proyecto

Diagrama de clases

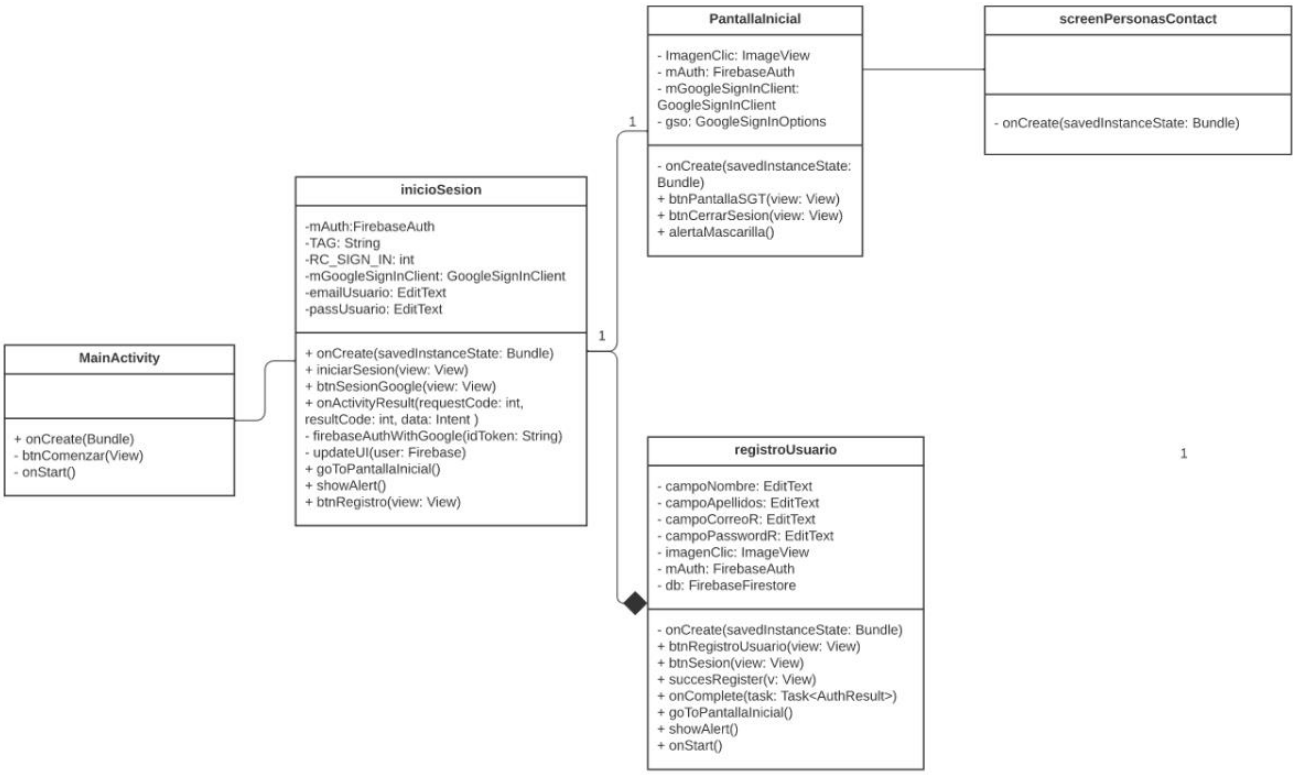

Ilustración 7. Diagrama de Clases

Diagrama de casos de uso

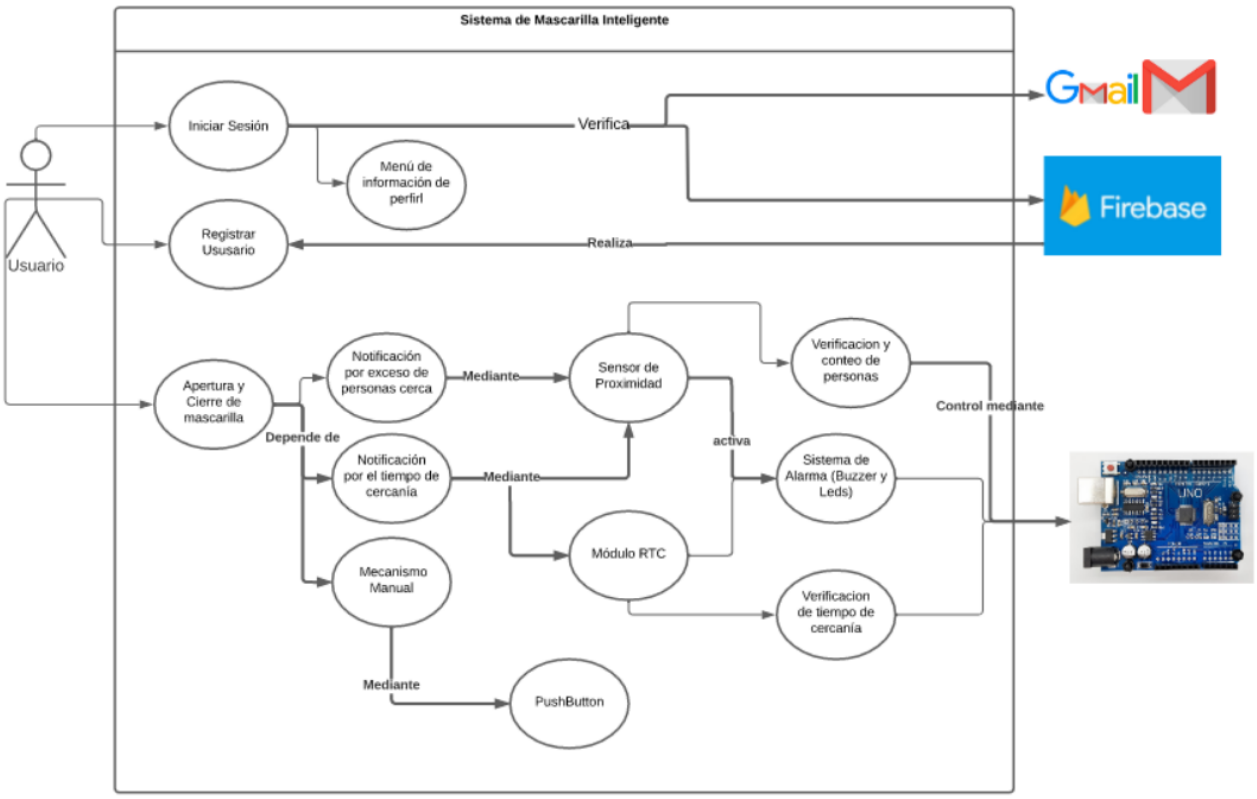

Ilustración 8. DIagrama de caso de usos

Diagrama de árbol

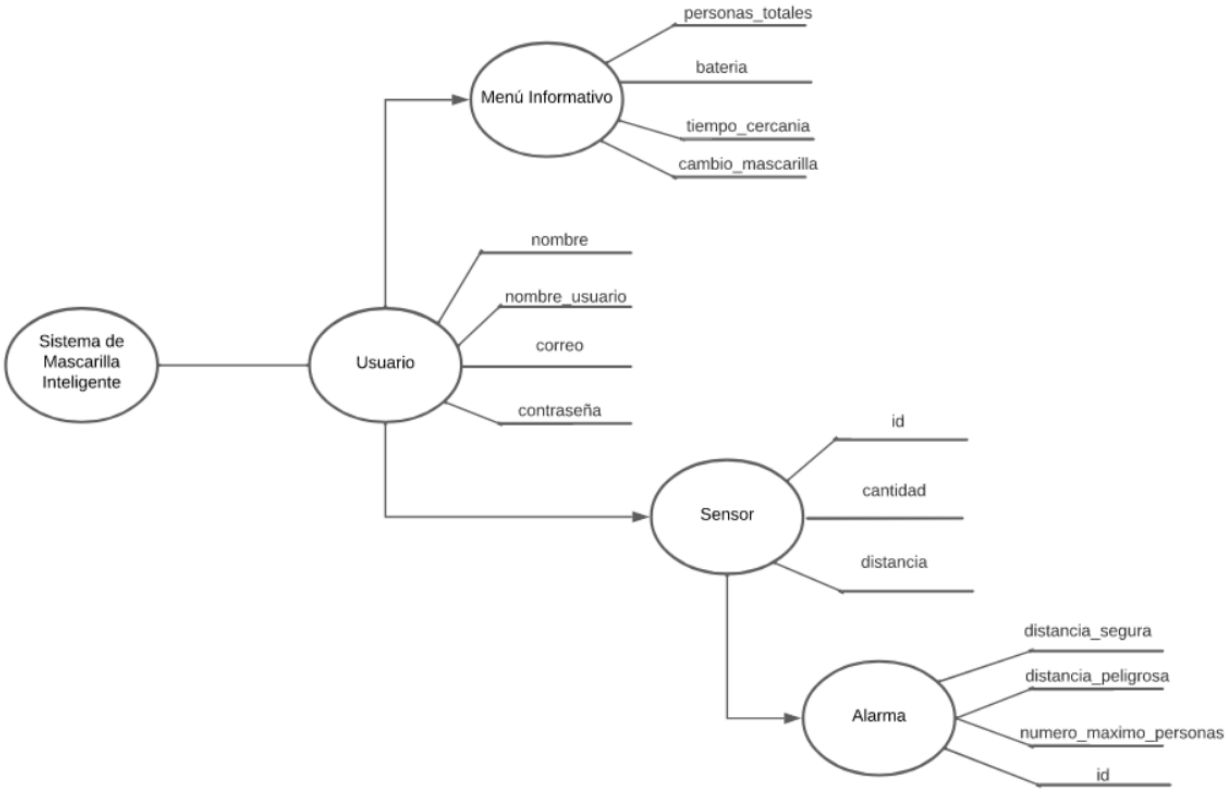

Ilustración 9. Diagrama de arbol

Resultados

Video del funcionamiento del sistema: https://espolec-my.sharepoint.com/:v:/g/personal/mnnunez_espol_edu_ec/EVk9TeCGyrlDgMIfG97kP-sBsHqENmoAARm5UwkGVKi6mg?e=FBsjbh

Conclusiones

- Se realizó la construcción del hardware del sistema de funcionamiento de una mascarilla inteligente capaz de medir distancias de peligro y registrar datos de tiempo y cercanía para alertar al usuario, además de contar con un sistema de apertura y cierre manual controlado mediante una botonera.
- Se realizó la implementación una aplicación móvil que permite al usuario observar la información requerida como lo es la cantidad de personas con las que se tuvo interacción, tiempo de la misma, identificación y estado de la mascarilla.
- Se trabajó la recolección de datos mediante el sensor ultrasónico y módulo RTC hacia una base de datos en tiempo real como lo es Firebase, de manera que fue posible llevar los datos tomados hacia la aplicación desarrollada

Tabla de precios

Componente	Cantidad	Precio
Arduino	1	$11.99
Sensor ultrasónico	1	$3.40
Módulo RTC	1	$5.85
Leds	4	$0.50
Buzzer	1	$1.20
Push Button	1	$0.20
Motor Stepper	1	$5.00
TOTAL		$28.14

Tabla 1. Tabla de componentes y precios

Apéndice

Código Hardware

```
//Instalación de librerías para el módulo RTC y control del STEPPER
#include <Stepper.h>
#include <Wire.h>
#include "RTClib.h"

//Configuración de los pasos en el motor STEPPER
Stepper motor1(2048, 8, 10, 9, 11);
//Declaración de los pines en el arduino MEGA
RTC_DS1307 rtc;
int TRIG=5;
int ECO=4;
int LED=3;
int PULSADOR=6;
int DURACION;
int DISTANCIA;
int BUZZER=2;
int LED_ROJO=42;
int LED_VERDE=38;
int LED_AMARILLO=40;
int ANALOG_BATERIA=0;
String ESTADO_MASK="CERRADA";
int analogValor=0;
float voltaje=0;

// Umbrales para la medición del voltaje en la batería
float maximo=1.5;
float medio=1;
float minimo=0;


void setup() {
  //Sincronización del módulo RTC con el tiempo actual de la computadora a utilizar
  rtc.begin();
  rtc.adjust(DateTime(F(__DATE__),F(__TIME__)));
  //Configuración de los pines como entradas o salidas
  pinMode(TRIG, OUTPUT);
  pinMode(ECO, INPUT);
  pinMode(LED, OUTPUT);
  pinMode(PULSADOR,INPUT);
  pinMode(BUZZER, OUTPUT);
  motor1.setSpeed(2); //Configuración de la velocidad de giro del motor
  Serial.begin(9600);
}

void loop() {
  //Lectura del voltaje de la batería
  analogValor=analogRead(ANALOG_BATERIA);
  //Conversión del voltaje según el datasheet del motor
```

```
  voltaje=0.0048*analogValor;
  //Serial.println(analogValor);
  //Serial.print("Voltaje: ");
  //Serial.println(voltaje);

  // Dependiendo del voltaje mostramos un LED u otro
  if(voltaje>=(0.3*maximo)){
    digitalWrite(LED_VERDE,HIGH);
    delay(2000);
    digitalWrite(LED_VERDE,LOW);
    }
  if(voltaje<(0.3*maximo)){
    digitalWrite(LED_AMARILLO, HIGH);
    delay(2000);
    digitalWrite(LED_AMARILLO, LOW);
    }
  if(voltaje==minimo){
    digitalWrite(LED_ROJO, HIGH);
    delay(2000);
    digitalWrite(LED_ROJO, LOW);
    }
  //Sensor ultrasónico y sus lecturas de onda para la obtención de la distancia
  digitalWrite(TRIG, HIGH);
  delay(1);
  digitalWrite(TRIG, LOW);
  DURACION=pulseIn(ECO, HIGH);
  DISTANCIA=DURACION/58.2; //Fórmula obtenida del datasheet del sensor ultrasónico para la conversión
a distancia
  Serial.println(DISTANCIA);
  delay(200);

  //Casos de uso para el encendido de la alarma por proximidad insegura
  if(DISTANCIA <= 30 && DISTANCIA >=0){
    digitalWrite(LED, HIGH);
    digitalWrite(BUZZER, HIGH);
    delay(DISTANCIA*10);
    digitalWrite(LED, LOW);
    digitalWrite(BUZZER, LOW);
  }
  //Serial.println(digitalRead(6));

  //Movimiento manual del motor por el push button
  if(digitalRead(PULSADOR) == HIGH){
    motor1.step(512);
    delay(500);
    if(ESTADO_MASK=="CERRADA"){
      ESTADO_MASK="ABIERTA";
      }else{
      ESTADO_MASK="CERRADA";
      }
  }
  Serial.println(ESTADO_MASK);
```

```
  mostrarFecha();
  mostrarHora();
}

//Función para mostrar la fecha en el monitor serial
void mostrarFecha(){
  DateTime ahora= rtc.now();
  char _bufferFecha[12];
  formatoFecha(_bufferFecha, ahora.day(), ahora.month(), ahora.year());
  Serial.println(_bufferFecha);
  }

//Función para mostrar la hora en el monitor serial
void mostrarHora(){
  DateTime ahora= rtc.now();
  char _buefferHora[10];
  formatoHora(_buefferHora, ahora.hour(), ahora.minute(), ahora.second());
  Serial.println(_buefferHora);
  }
//Función para establecer un formato adecuado para la fecha
void formatoFecha(char bufferFecha[12], int numDia, int numMes, int numA){
  sprintf(bufferFecha, "%02d/%02d/%04d", numDia, numMes, numA);
  }
//Función para establecer un formato adecuado para la hora
void formatoHora(char bufferHora[10], int hora, int minu, int seg){
  sprintf(bufferHora, "%02d:%02d:%02d", hora, minu, seg );
  }
```

Conexión Serial

```
#Importación de la librería SERIAL para la comunicación por serial con el arduino y de la librería FIREBASE para
#la conexión con la base de datos
import serial, time
from firebase import firebase
#Configuración del puerto serial COM6 con arduino IDE
arduino=serial.Serial('COM6',9600)
time.sleep(2)
#Lectura de la distancia, estado de mascarilla, fecha y hora que son enviados por el puerto serial desde ARDUINO hacia
#este IDE de PYCHARM y su posterior conversión en datos de 20 bits
lecturaDistancia = arduino.readline()
arduino.write(b'20')
lecturaEstado = arduino.readline()
arduino.write(b'20')
lecturaFecha = arduino.readline()
arduino.write(b'20')
lecturaHora= arduino.readline()
arduino.write(b'20')
#Decodificación de las lecturas enviadas por SERIAL en formato STRING para una correcta lectura de los datos
```

```
distancia=lecturaDistancia.decode()
estado=lecturaEstado.decode()
fecha=lecturaFecha.decode()
hora=lecturaHora.decode()
#Conexión con la base de datos en FIREBASE
firebase = firebase.FirebaseApplication('https://emaskg5-default-rtdb.firebaseio.com', None)
#Estructura del formato para el envío de los datos hacia FIREBASE
datos={
   'Distancia': distancia,
   'Estado': estado,
   'Fecha:':fecha,
   'Hora:': hora
}
#Método POST para subir los datos hacia FIREBASE siguiendo la estructura anterior
resultado=firebase.post('RegistroDatos/SmartGuard1',datos)
arduino.close()
```