UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO DE SISTEMA DE DETECCIÓN DE ENFERMOS COVID

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Adrian Gervasini Navarro

TUTORIZADO POR

Roberto Capilla Lladró

CURSO ACADÉMICO: 2020/2021

Agradecimientos

Para empezar, me gustaría agradecer:

- A toda mi familia, principalmente, a mis padres por haberme brindado toda la educación recibida y a mi hermano por ser una motivación para mí.
- A mi novia por haberme apoyado durante todos estos meses y haberme motivado a seguir adelante.
- A Martí y a Víctor, por haberme acompañado en esta bonita etapa.
- A todos aquellos buenos profesores de la carrera que me han hecho disfrutarla y aprender.

Declaración de autoría

Aquí declaro, que soy el autor de este trabajo. Incluyendo el código fuente, así como, el montaje de la maqueta y la realización de esta memoria. He utilizado fuentes de información que citaré en el apartado final de bibliografía.

ÍNDICE

1. Introducción

En diciembre de 2019 surge en Wuhan (China) un brote de casos de neumonía aguda por parte de un grupo de gente, la mayoría trabajadores de un mercado mayorista de pescados. Estos fueron los primeros casos de SARS-CoV-2. Un virus de la familia de los coronavirus, que se ha extendido mundialmente y ha causado la pandemia de COVID-19 reconocida por la Organización Mundial de la Salud el 11 de marzo de 2020.

Durante el período de pandemia se han registrado alrededor de 135 millones de casos a nivel mundial y unos 3 millones de personas fallecidas. En este tiempo se ha descubierto que este virus afecta principalmente al sistema respiratorio, aunque los principales síntomas son: fiebre, tos seca y cansancio entre muchos otros.

Des de la OMS y con el objetivo de reducir los casos siempre han aconsejado que es muy importante limpiarse las manos con agua y con jabón, llevar mascarilla, tomarse la temperatura, tener distancia de seguridad interpersonal de 1,5 metros (mínimo) y un lago etcétera.

Y, a pesar de que en muchos países ya se puede salir sin mascarilla tranquilamente, en España, aún siguen llevándose a cabo muchas de las recomendaciones de la OMS. En todos los lugares públicos hay disponible a las entradas y, muchas veces, en las salidas un dispensador de gel hidroalcohólico y, además, en centros públicos como centros de salud, hospitales, ayuntamientos… es obligatorio tomarse la temperatura antes de entrar.

En conclusión, esta situación ha conducido a la sociedad a establecer objectos que sean rápidos y fáciles de usar para poder llevar a cabo todas las medidas de seguridad necesarias teniendo en cuenta la tecnología.

2. Justificación

La idea de este proyecto nace como consecuencia de la necesidad actual en la sociedad de encontrar la forma de reducir los contagios de covid-19.

El objetivo general de este proyecto, por tanto, es la prevención de contagios de COVID-19 ya que la alta tasa de infectados está provocando medidas de cierre o cese de negocios, produciendo unas grandes pérdidas económicas en empresarios y trabajadores.

La prevención se producirá controlando el acceso de las personas a locales o espacios cerrados tomando la temperatura de las personas y dispensando el gel de forma automática para que así se pueda acceder a estos siempre que la temperatura de la persona sea igual o menor a 37’1º.

Los objetivos específicos son cuatro:

1. Incorporar en el diseño del sistema diferentes sensores capaces de tomar la medida de temperatura de la persona y detectar la proximidad del usuario al sistema y la proximidad de las manos al dispensador de gel.
2. Añadir al diseño varios actuadores tales como: bomba peristáltica para la dispensación del gel de forma automática, un servomotor que se encargará de abrir o cerrar un pasador acoplable en cualquier tipo de puerta para controlar el acceso y un monitor y un altavoz con sonidos pregrabados para informar a las personas de los pasos a seguir.
3. Desarrollar el código para el microcontrolador donde se procesen las señales recibidas y se actúe en función de éstas.
4. Desarrollar una aplicación móvil para registrar la información del microcontrolador y controlarlo.

3. Metodología

La metodología de este proyecto queda dividida en cinco subapartados, donde se detallan con el fin de explicar el desarrollo del sistema.

3.1- Requerimientos del sistema

El centro del sistema será Arduino. Esta se puede definir como una plataforma de código abierto. Su software y hardware son de uso libre para todo el mundo y están pensados para ser accesibles y fáciles de usar. Arduino consta de una placa principal de componentes, ahí se encuentran los principales controladores del dispositivo, los cuales dirigen al resto de componentes. La placa de Arduino viene “preensamblada” con un acceso facilitado a los distintos elementos o conexiones que pueden ser necesarias, tanto alimentación, como pines de entrada y/o salida de datos, tanto analógicos como digitales. Dicho esto, se va a trabajar con una placa Arduino Uno.

El hardware que compone a la mayoría de las placas Arduino y entre ellas la que se va a utilizar en el proyecto, es un microcontrolador Atmel-AVR de 8 bits. Este se encargará de realizar los procesos lógicos y controlar los componentes conectados a la placa. Además, al procesador principal lo pueden acompañar microprocesadores complementarios con un peldaño menos de potencia como son el Atmega168, Atmega328, Atmega1280 y el Atmega8. Estos son los más usados por su gran versatilidad y su reducido precio. A parte, Arduino cuenta con entradas y salidas digitales y analógicas capaces de ser configuradas mediante conexión USB a un ordenador. Otros datos relevantes sobre el Arduino UNO son sus 32 kbytes de memoria Flash, 1 kbyte de memoria RAM, 13 pines para entradas/salidas digitales, 5 pines para entradas analógicas y 6 pines para salidas analógicas.

Por otra parte, el software que envuelve a Arduino es muy extenso. El lenguaje de programación en el que se basa es C/C++, aunque soporta varios lenguajes de programación derivados de este. En la propia web de Arduino se pone a disposición la herramienta de programación de software para ordenador totalmente libre y gratuita.

A Arduino se le añadirán sensores, actuadores y periféricos con el fin de conseguir el proceso de control de acceso deseado.

3.2- Control de temperatura personal

Respecto a la medida de temperatura personal es importante saber dónde se sitúa el valor de temperatura que se puede considerar como febrícula.

Termómetro	Temperatura normal	Fiebre
Oral	35’5 ºC – 37’5 ºC	37’6 ºC y superior
Rectal	36’6 ºC – 37 ºC	37’1 ºC y superior
Temporal (la frente)	35’8 ºC – 37 ºC	37’1 ºC y superior
Timpánica	35’8 ºC – 37 ºC	37’1 ºC y superior
Axilar	34’7 ºC – 37’3 ºC	37’4 ºC y superior

Tabla 1. Tabla de temperaturas normales y de fiebre en las distintas partes del cuerpo humano

Como se puede observar en la *Tabla 1*, en la frente se deberá considerar fiebre a partir de los 37’1º C. Este será nuestro primer valor límite que nos hará determinar si es posible el ingreso o no del usuario al local.

A parte, se incorporarán otros valores como límites superior e inferior, que determinarán si el usuario está bien colocado o no, o si hay algún objeto interfiriendo en la medida de la temperatura entre el sensor y la frente.

El límite superior se puede determinar en 40º C. Esta temperatura ya puede ser considerada excesiva para que sea la de un ser humano y por lo tanto es posible considerarlo que estamos sufriendo alguna interferencia.

El límite inferior de temperatura se sitúa en 34º C. Por debajo de esta temperatura corporal, se puede hablar de hipotermia en algunos casos y en nuestro sistema podemos estar teniendo interferencias por algún objeto como gafas de sol o gorra.

Dentro del rango de 34º C - 40º C, si nuestro usuario se encuentra entre el rango de [34-37’1]º C, se podrá proceder a la dispensación de gel hidroalcohólico. En cambio, si se sitúa entre [37’1-40]º C el propietario del local podrá determinar si permite la entrada o no del usuario en cuestión.

El sensor de temperatura a utilizar se trata del MLX90614, se trata de un sensor de temperatura que basa su funcionamiento en infrarrojos y trabaja a una tensión de funcionamiento de 5 V.

3.3- Dispensador de solución hidroalcohólica

Independientemente de haber tenido una temperatura inferior a 37’1º C y determinar que no hay fiebre, es posible tener o ser portador del virus o a ver tocado alguna superficie o persona que lo tuviese. Por lo que la OMS recomendó el uso de soluciones hidroalcohólicas para evitar la transmisión.

Por ello, el sistema estará dotado de un dispensador de gel hidroalcohólico automático, el cual se activará por proximidad de las manos y posteriormente a la medida de temperatura del usuario.

Este sistema estará formado por un recipiente que contenga el gel hidroalcohólico, una bomba peristáltica que dispensará sobre 1 ml de gel por uso y un detector de

proximidad para determinar que se han colocado las manos cerca del dispensador. La distancia que configurar para este sensor será de 10 – 15 cm.

3.4- Apertura de puerta

Tras haber comprobado una temperatura correcta en el usuario, y haber suministrado el gel hidroalcohólico, el usuario ya es apto para acceder al local, ya que tiene menos posibilidades de estar contagiado. Durante todo el proceso de desinfección de manos y lectura de temperatura la puerta se mantendrá cerrada por un pasador como el de la Ilustración 1.

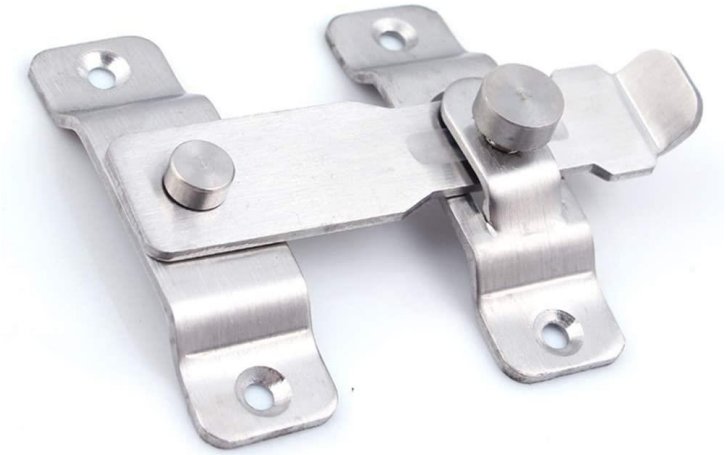


Ilustración 1. Pasador manual con eje de giro

Para automatizar la apertura de la puerta con este pasador, se le acoplara un servomotor en el eje de giro, pudiendo controlar mediante un giro de 90 grados la apertura y el cierre de la puerta.

3.5- Comunicación Bluethooth

Después del proceso de acceso al local, tanto si el usuario es apto para entrar o no, al propietario del local le quedarán registrados en una aplicación para su dispositivo Android una serie de datos como son: hora del acceso del usuario, temperatura del usuario, si se ha dispensado correctamente el gel de manos al usuario y, por último, la aplicación llevará un recuento de los usuarios que han accedido al local.

La comunicación de la placa Arduino y el dispositivo móvil del propietario se hará mediante conexión bluetooth. En nuestro caso con un módulo hc-05 que es la tecnología más eficiente actualmente.

4. Elección del hardware y software

4.1- Arduino

Aunque para realizar este proyecto se puede usar cualquier microcontrolador o DSC de propósito general, se ha escogido Arduino por una serie de ventajas y características que se pueden resumir en:

- **Precio**. Hay una gran cantidad de modelos de placas Arduino, además de haber una gran cantidad de fabricantes de estas placas, ya que son de código libre. Por lo que al haber mucha oferta los precios suelen ser muy económicos. Y este es uno de los requisitos de nuestro proyecto.
- **Hardware y código libre**. Todo el entorno de Arduino es Open Source, tanto el hardware como el software está disponible para ser modificado y/o mejorado por los usuarios.
- **Multiplataforma**. El software de Arduino puede ser instalado y ejecutado en los principales sistemas operativos como son Windows, Mac y/o Linux. Mientras que la mayoría de los programas de otros microcontroladores solo están disponibles para Windows.
- **Entorno de programación simple y claro**. El software de Arduino es lo suficientemente sencillo para que sea utilizado por usuarios principiantes y lo suficientemente versátil como para sacarle un gran provecho por usuarios más exigentes.

4.2- Placas de Arduino

La empresa Arduino ofrece una gran cantidad de productos de hardware: sensores de todo tipo, actuadores, módulos, kits de iniciación o para el desarrollo de algún proyecto concreto y, además, ofrece una gran cantidad de placas muy variadas con distintas características. Las placas pueden diferir en tamaño para su uso en proyectos de menor escala o en especificaciones si para el proyecto son necesarios por ejemplo un mayor rendimiento o una mayor cantidad de pines de conexión.

A continuación, se describen las especificaciones más destacables de algunas de las placas Arduino del mercado:

- **Arduino UNO.** Se trata de la mejor opción para iniciarse en Arduino. Esta placa tiene como microcontrolador un ATmega328P de 8 bits a 16 MHz alimentado a 7-12 V. Es más simple que por ejemplo una placa Arduino Mega. Además, la placa consta de 14 pines digitales de entrada/salida, donde 6 de estos pines pueden ser usados como salidas PWM, 6 entradas analógicas, etc.

Ilustración 2. Placa Arduino UNO

- **Arduino nano.** Esta pequeña placa está basada en el mismo microcontrolador que el anterior. Esta placa es bastante más pequeña que el Arduino UNO y su conexión de alimentación y comunicación es mini USB. Además, incluye 8 pines analógicos, 14 pines digitales de entrada/salida. Esta placa se usa para proyectos donde es necesario reducir el espacio de la placa por especificaciones del proyecto.

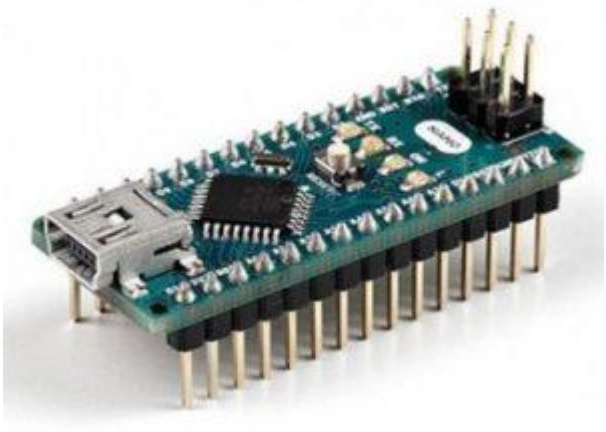


Ilustración 3. Placa Arduino nano

- **Arduino micro.** El Arduino micro tiene como microcontrolador el ATmega32U4 de 8 bits. Tiene un tamaño similar al nano (unos milímetros más largos), pero se diferencia en sus 12 pines analógicos, 20 pines digitales de entrada/salida donde 7 de ellos son PWM, oscilador de cristal de 16 MHz, etc.

Ilustración 4. Placa Arduino micro

- **Arduino Due.** Esta placa de Arduino funciona con un ARM Cortex-M3 de 32 bits, es decir un mayor rendimiento frente a los otros modelos de placas que hemos visto. A parte, también tiene más pines que las demás placas, 54 pines digitales de entrada/salida, donde 12 de estos son salidas PWM. También tiene 12 entradas analógicas y se alimenta a 3’3 V mientras que el resto de las placas funcionan entre 5-12 V.

Ilustración 5. Placa Arduino Due

- **Arduino Mega.** El Arduino mega es bastante parecido al UNO, pero con dimensiones superiores y una mayor cantidad de pines, en total 54 pines de entrada/salida digital, donde 15 son PWM. Como entradas analógicas tenemos 16 pines.

Ilustración 6. Placa Arduino Mega

- **Arduino Leonardo.** La placa es bastante similar al Arduino Uno. Como diferencias principales se pueden destacar su tamaño más reducido, el número de pines se amplía a 20 en los pines de entrada/salida digital, además los pines son perforaciones en la placa, no se cuenta con tiras de pines para conectarse.

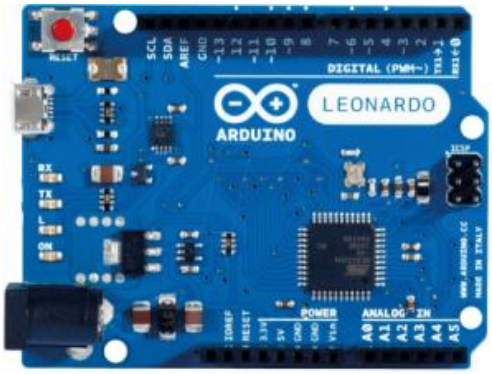


Ilustración 7. Placa Arduino Leonardo

4.3- Sensores

4.3-1.Sensor de temperatura – MLX906

Al hablar de sensores o transductores de temperatura tenemos muchos tipos.

1. **Bimetales**. Basan su funcionamiento en 2 metales distintos con diferente coeficiente de dilatación unidos a una misma estructura. Los cambios de temperatura hacen que el metal que más cambie “estire” del otro produciendo un movimiento o deformación proporcional a la variación de temperatura.

2. **Resistencias termométricas.** Su funcionamiento viene determinado por el aumento de resistencia eléctrica que experimenta cualquier metal con la temperatura. Habitualmente el metal utilizado es el platino por su gran linealidad, su buen rango de temperaturas y su gran precisión.

3. **Termopares**. El principio de funcionamiento de estos transductores es el efecto seebeck, donde en un circuito con 2 metales conductores homogéneos, distintos, unidos en sus extremos, si las 2 uniones de estos metales están a distinta temperatura, aparece una corriente eléctrica en función de la variación de temperatura.

4. **Termistores.** Estos transductores, como las resistencias termométricas, basan su funcionamiento en la variación de la resistencia con la temperatura, pero en este caso están compuestos por semiconductores.

5. **Sensores en circuitos integrados**. Basados en la sensibilidad frente a la temperatura de las uniones semiconductoras.

   Todos estos sensores son muy útiles si las medidas que se desean realizar son del ambiente o de un cuerpo en contacto con el transductor, pero en el proyecto se intenta evitar el contacto para prevenir los contagios. Por lo que se ha escogido un transductor que no necesita el contacto para tomar una medida de la temperatura.El sensor de temperatura de nuestro sistema es el MLX90614. Es un sensor de temperatura infrarrojo. Estos sensores basan su funcionamiento en la radiación a diferentes longitudes de onda que emiten los cuerpos en función de su temperatura. El diseño más sencillo consiste en una lente que enfoca los rayos infrarrojos que emite el cuerpo a un pirómetro. Esta lectura que recibe el pirómetro se convierte en una señal analógica a procesar por nuestra placa.

Ilustración 8. Sensor de temperatura MLX90614

Las características principales del sensor MLX90614 son:

- Sensor de tamaño reducido (8’2 × 8’2 × 17 mm)
- Tiene fácil integración
- El rango de temperatura oscila entre -70ºC y 380 ºC
- Resolución de medida de 0’02 ºC
- Alimentación de 3V a 5V

4.3-2.Detector de proximidad

En cuanto al diseño del dispensador de gel, es necesario saber cuándo el usuario coloca las manos bajo el dispensador para que así, se le proporcione el gel hidro-alcohólico. Para ello se necesita un detector de proximidad/posición.

Los distintos tipos de sensores de posición pueden basar su funcionamiento en triangulación, donde el emisor suele ser un LED o un diodo láser, y el receptor puede ser un PSD o un sensor CCD.

También pueden basarse en tiempo de vuelo, donde el emisor es un diodo láser y el receptor un fotodiodo.

En este proyecto se va a utilizar un sensor basado en triangulación. En concreto el GP2Y0A21YK0F. Éste está compuesto por un PSD (position sensitive detector) y por un IRED (infrared emiting diode) y un circuito de procesado de señal.

Ilustración 9. Sensor de posición - GP2Y0A21YK0F

Las características de este sensor son:

- Rango de medida óptimo que fluctúa entre los 10 y los 80 cm
- Salida analógica
- La alimentación bascula entre 4’5 – 5’5 V
- El tamaño es de 29’5×13×13’5 mm

4.4- Actuadores

4.4-1.Bomba peristáltica

A fin de que el dispensador de gel hidroalcohólico suministre el gel es necesaria una bomba. Y, para eso, se ha escogido una bomba peristáltica.

Las bombas peristálticas tienen en su interior unos rodillos que comprimen la manguera al girar, de esta forma se crea un vacío dentro de la manguera que absorbe el fluido de un lado y lo suministra por el otro.

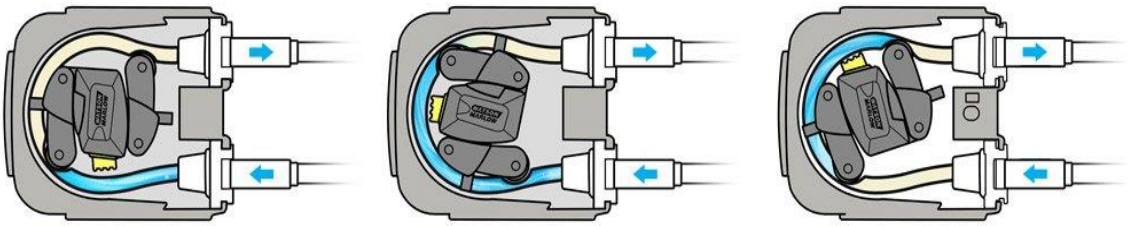


Ilustración 10. Interior bomba persitáltica

Una ventaja que hay que tener en cuenta es que en las bombas peristálticas ningún componente a parte de la manguera está en contacto con el fluido. Otra ventaja es la precisión de la bomba para dosificar el volumen de fluido deseado.

Para el proyecto, se ha elegido una bomba peristáltica de 12 V con el objetivo de que el flujo sea el suficiente para una dosificación en poco tiempo. Por lo que también se ha añadido un relé al proyecto que tiene como finalidad controlar la bomba peristáltica ya que los pines de salida del Arduino no son capaces de suministrar 12 V de tensión necesarios para hacer funcionar la bomba.

4.4-2.Relé

No es posible alimentar y controlar la bomba peristáltica de 12 V con ningún pin de Arduino. La alimentación es de 12 V pero los pines digitales tienen una salida máxima de 5 V, por lo que es necesario utilizar un relé.

Un relé es un interruptor, que se controla mediante un circuito eléctrico, así pues, se podrá controlar que el motor se encienda o no mediante un pin de salida digital, aunque esté conectado a la alimentación de 12 V de entrada del Arduino.

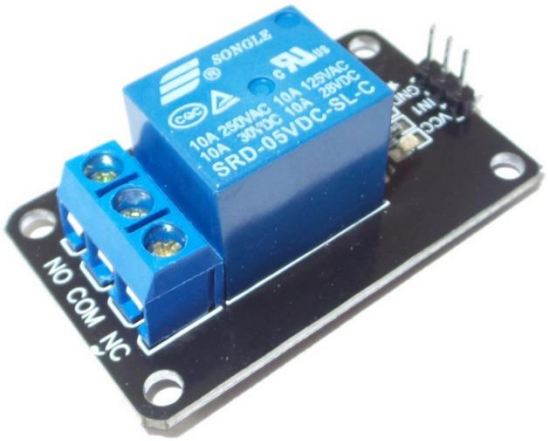


Ilustración 11. Relé

4.4-3.Servomotor

Ya que es necesario un método de apertura de la puerta del local/establecimiento se va a diseñar un mecanismo de un pasador con eje de rotación y un servomotor para controlar la apertura y cierre de dicho pasador.

La diferencia entre los servomotores y los motores de corriente continua es que los servomotores pueden posicionarse y mantenerse en una posición concreta dentro de un rango. El servomotor puede ser cualquiera con el suficiente par para hacer rotar un pasador de unas pocas decenas de gramos.

Por otro lado, la alimentación del servomotor será suficiente con los 5 V que ofrece el Arduino para la mayoría de los servomotores que se utilicen. En el caso de que el servomotor fuese de 9 V o 12 V, se podrían obtener de la placa ya que esta estará alimentada a 12 V

4.4-4.Display LCD de 16x2 segmentos

Durante todo el proceso de control de acceso se incorporará un display de 16 x 2 segmentos que irá mostrando toda la información necesaria y los pasos a seguir por el usuario.

Este display funciona con una tensión de 5V para su alimentación y necesita de 6 pines digitales para la conexión con la placa.

En el display se mostrará la temperatura del usuario. Posteriormente saldrá un mensaje diciendo que se coloquen las manos bajo del dispensador de gel. Finalmente, el display mostrará un mensaje diciendo que la puerta está abierta y se puede acceder al local.

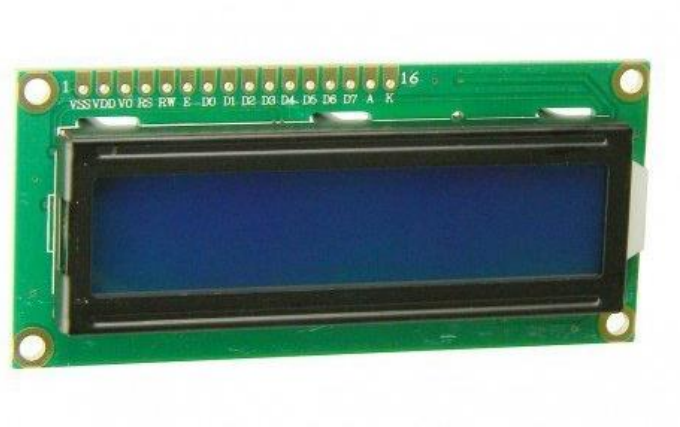


Ilustración 12. Display LCD de 16 x 2 segmentos

4.4-5.Altavoz

Como se quiere diseñar un sistema fácil de entender por el usuario, se añadirá un pequeño altavoz de 8 Ω de resistencia interna el cual producirá sonidos a distintas frecuencias con la intención de hacer más sencillos los pasos a seguir por los distintos clientes.

Ilustración 13. Altavoz

4.5- Elección de Arduino

Entre todas las opciones de placas que se han descrito, finalmente en el proyecto se procede a usar la placa Arduino UNO como motor principal de nuestro sistema.

Esta elección ha sido principalmente porque era la placa que tengo en mi posesión. Cabe añadir que, aunque no la hubiese tenido, la habría escogido por diversos motivos: su precio, accesibilidad y oferta en el mercado y porque cumple todos los requisitos necesarios para el proyecto ya que los pines necesarios son suficientes con esta placa.

El sensor de temperatura requiere 2 pines de entrada analógica. El detector de proximidad también necesita de 1 pin de entrada analógica. El relé necesita 1 pin de salida digital como el servomotor. El módulo de comunicación bluetooth, que se comentará posteriormente, únicamente necesita los dos pines TX y RX que se sitúan en los pines de entrada/salida digital. Por último, el display LCD utiliza 6 pines digitales de salida

4.6- Comunicación

4.6-1.Módulo Bluetooth

Se pretende establecer una comunicación entre el sistema y un dispositivo Android del propietario. Frente eso, se ha decidido utilizar un módulo bluetooth para una conexión maestro – esclavo donde el sistema enviará toda la información recogida de los usuarios al dispositivo.

Por este motivo, se ha escogido la tecnología bluetooth para la conexión inalámbrica, en concreto el módulo DSD TECH HC-05. Este módulo trabaja a una tensión de operación de entre 3’6 V y 6 V, por lo que podemos alimentarlo con la tensión de 5 V que ofrece el Arduino. Además, la distancia de conexión es de hasta 10 metros, que es una distancia suficiente para realizar la conexión.

Ilustración 14. Módulo DSD TECH HC-05

4.7- Software

4.7-1.Arduino IDE

Arduino IDE es el programa que enviará a la placa Arduino todo el código que se desarrolle durante el proyecto. Las siglas IDE, hacen referencia a "Integrated Development Environment" y es un entorno muy simple para trabajar.

Inicialmente se tienen las funciones de "setup" y "loop" donde se configura y se ejecuta el código, respectivamente. Pero también es posible crear funciones propias para todo lo que sea necesario.

Este programa instalado en nuestro ordenador se enviará el código a la placa Arduino mediante una conexión por USB con la placa. El ordenador automáticamente debe reconocer la tarjeta.

Posteriormente, en el entorno IDE se debe seleccionar en la pestaña "tarjeta" del menú "herramientas" el tipo de tarjeta con la que se va a trabajar. En este caso será la tarjeta "Arduino UNO". Luego seleccionaremos en la pestaña "Puerto Serial" el bus donde está conectada nuestra tarjeta.

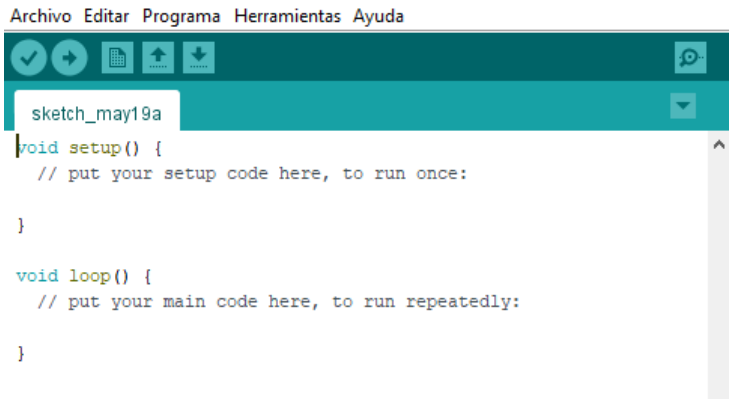


Ilustración 15. Entorno de Arduino IDE

4.7-2.MIT App Inventor

Se trata de una aplicación online muy sencilla e intuitiva, pero con todos los elementos necesarios para desarrollar una aplicación lo suficientemente completa y así poder recibir todos los datos del sistema.

En esta aplicación web se desarrollará el programa donde se reflejarán los datos obtenidos por nuestro Arduino.

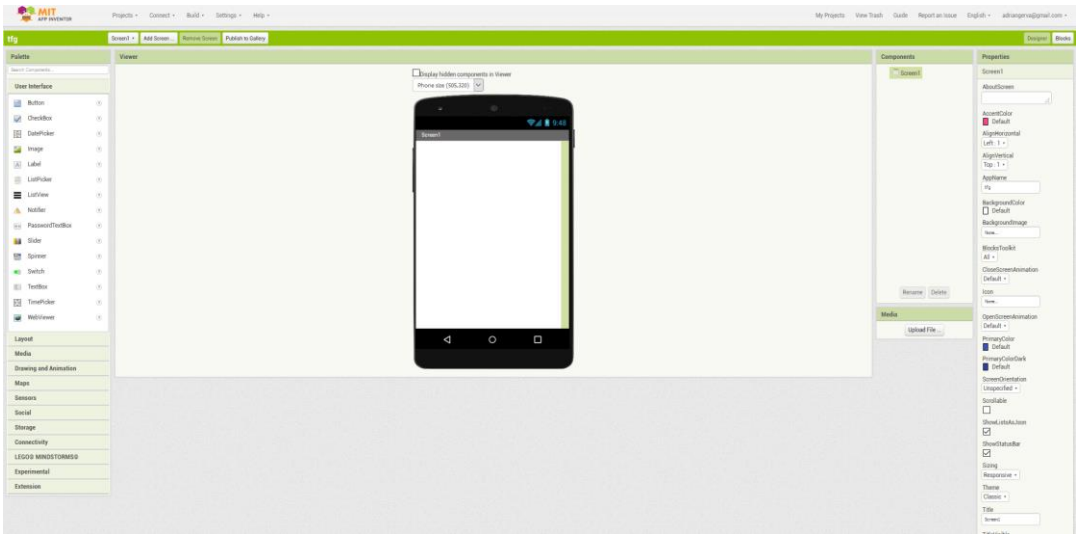


Ilustración 16. Entorno MIT App Inventor

4.7-3.Proteus 8 professional

Proteus es un software utilizado para la creación de proyectos de equipos y circuitos electrónicos en sus diferentes etapas.

Desde diseño del esquema electrónico, programación software, diseño de la placa de circuito impreso, ect.

Se ha utilizado el software para la simulación de las conexiones entre la placa arduino y los distintos elementos que componen el sistema ya que de esta forma las conexiones son mucho más claras visualmente.

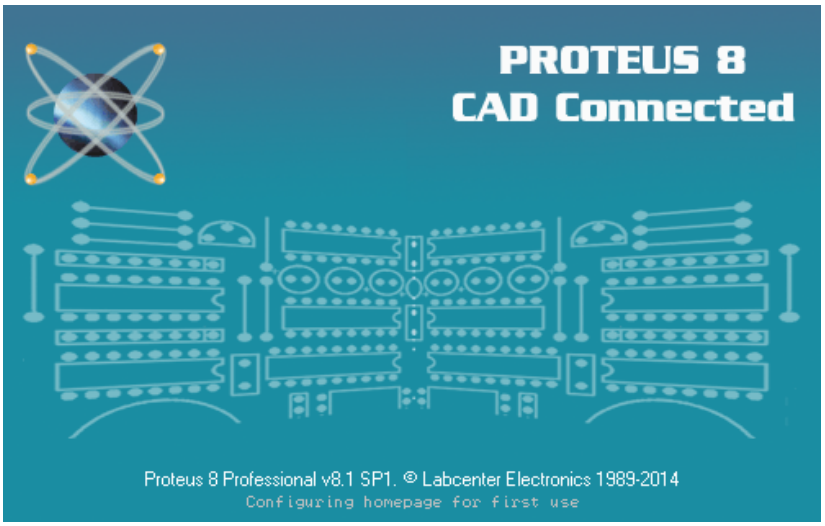


Ilustración 17. ISIS PROTEUS 8

5. Diseño

5.1- Diseño del proyecto

En cuanto al diseño del sistema, podemos observar en la *Figura 1* los módulos de los que se compone: detector de proximidad y temperatura, módulo bluetooth, Display LCD…

La realización del proyecto se ha diseñado dentro de una caja para el fácil transporte de éste y tener todas las conexiones en la misma ubicación, aunque es posible distribuir los elementos de la forma que se desee.

5.2- Sensor de temperatura

Como se ha comentado anteriormente, el sensor de temperatura es el mlx90614, este sensor dispone de 4 pines de conexión.

El primer pin es el de alimentación a 5 V que obtenemos de la alimentación de 12 V de la placa Arduino y el regulador de tensión a 5 V que tiene esta. Luego se encuentra el pin común que se conectará con uno de los GND de la placa. Por último, encontramos los pines SDA y SCL que son los pines que se utilizan para la comunicación I2C entre el sensor y la placa. Estos pines se conectan con los pines A4 y A5 respectivamente de nuestro Arduino como se puede observar en la *Ilustración 18*.

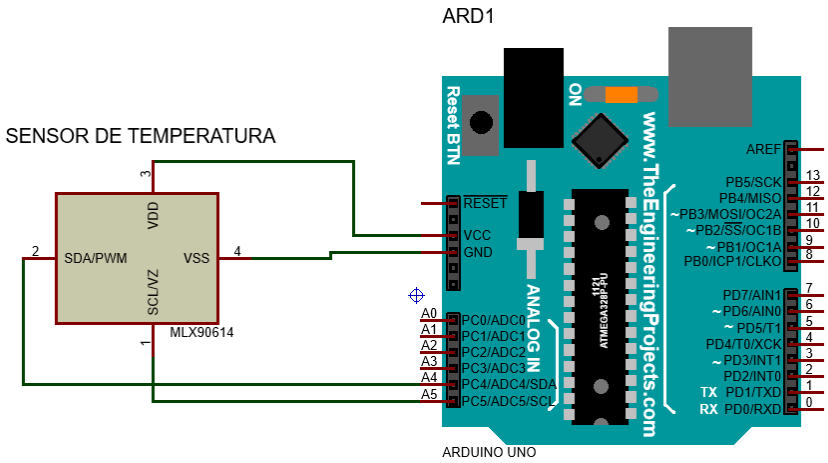


Ilustración 18. Conexión sensor de temperatura - arduino

Respecto a su utilización, hay que usar una librería específica para poder trabajar con el sensor. Ésta, es la "Adafruit_MLX90614.h" donde la principal función de lectura de datos del sensor es *readObjectTempC()* que lee la temperatura del objeto frente al sensor a unos pocos centímetros de distancia.

Para una cómoda lectura de la temperatura de la frente, el sensor se ha colocado a una cierta altura por encima del resto del sistema en la caja mediante una varilla metálica.

5.3- Detector de proximidad

El Sharp GP2Y0A21YK0F es el detector de proximidad a utilizar en nuestro proyecto con 3 pines de conexión.

Es un sensor analógico con una tensión de alimentación de 5 V, por lo que el sensor irá conectado a los pines de Vcc, Gnd y a una de las entradas analógicas, en este caso a la A0 como se muestra en la *Ilustración 19.*

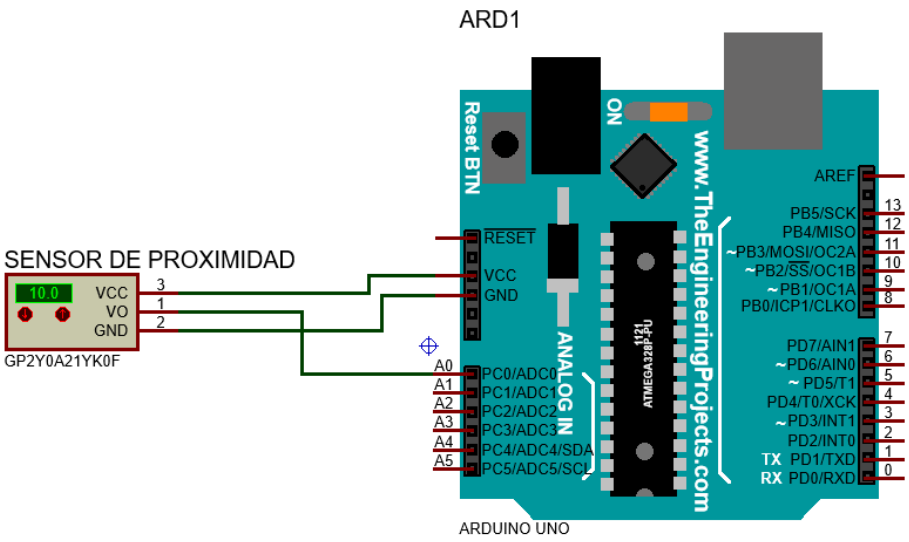


Ilustración 19. Conexión detector de proximidad - Arduino

Sobre la ubicación en el sistema, el sensor se ha colocado de forma perpendicular a la pared, para poder programarse para que detecte varias distancias. Una primera de usuario, que detectará si el sujeto está cerca del sistema y desea entrar, y una distancia de manos, la cual informará de que están lo suficientemente próximas como para dispensar gel hidroalcohólico.

Se ha estudiado la respuesta del sensor con las variaciones de la lectura de la distancia de un objeto, y el resultado que dio es el siguiente:

Distancia del objeto (cm)	Tensión de salida (V)	Lectura del puerto analógico
10	2.34	480
20	1.46	300
30	0.95	195
40	0.71	146
50	0.59	120
60	0.50	102
70	0.44	90
80	0.38	77

Tabla 2. Medida de valores del detector de proximidad

Las medidas se han realizado en un rango de entre 10 cm y 80 cm porque son las distancias de uso que recomienda el fabricante.

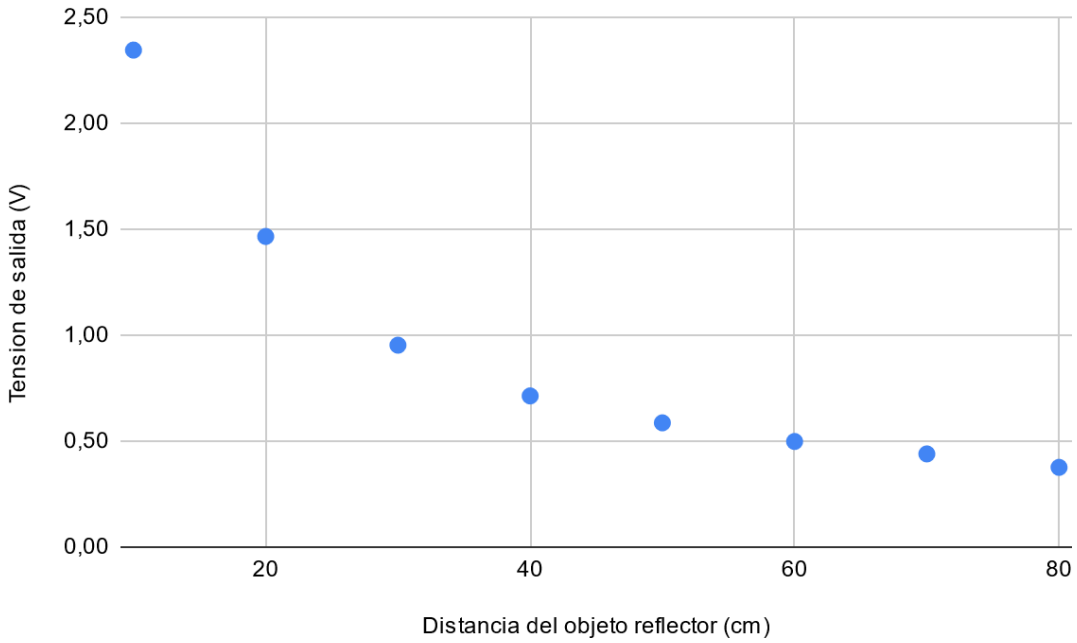


Gráfico 1. Respuesta de tensión frente a distancia - reales

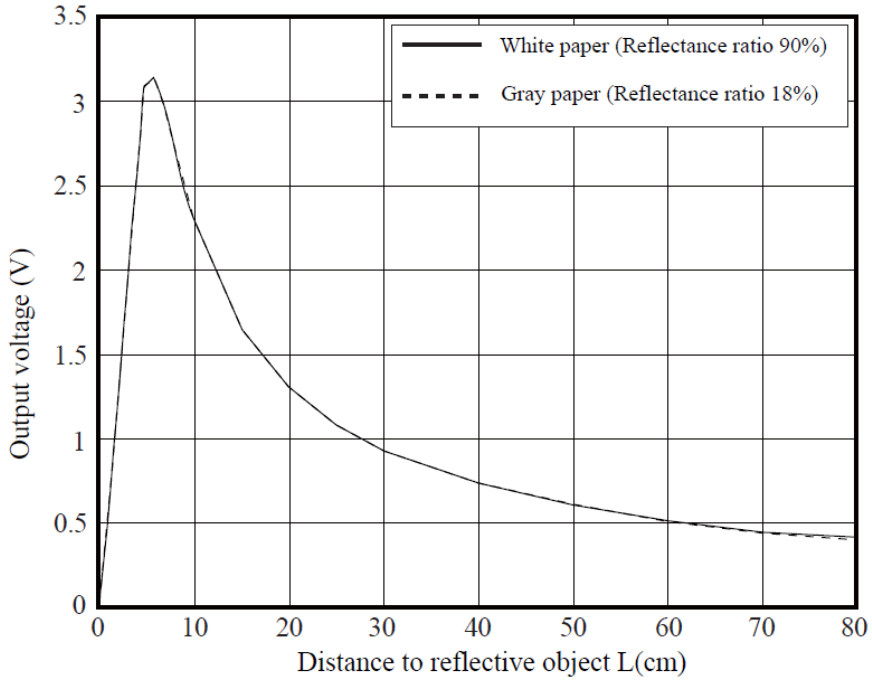


Gráfico 2.Respuesta de tensión frente a distancia - teóricas

Como se puede observar, los valores obtenidos que forman el *Gráfico 1* son muy similares a los que se deberían obtener, como muestra el *Gráfico 2*, que es la respuesta del sensor que nos ofrece el fabricante.

5.4- Dispensador de gel hidroalcohólico

En el sistema del proyecto, el dispensador de gel está formado por una botella de gel hidroalcohólico que se sitúa dentro de la caja, a la cual se le ha añadido el tubo de entrada de líquido de la bomba y el tubo de salida de líquido sale a través de un agujero en la pared de la caja, 3 cm por debajo de donde está situado el detector de proximidad.

Además, la bomba peristáltica se encuentra conectada como se observa en la *Ilustración 20*, a un relé y a la placa.

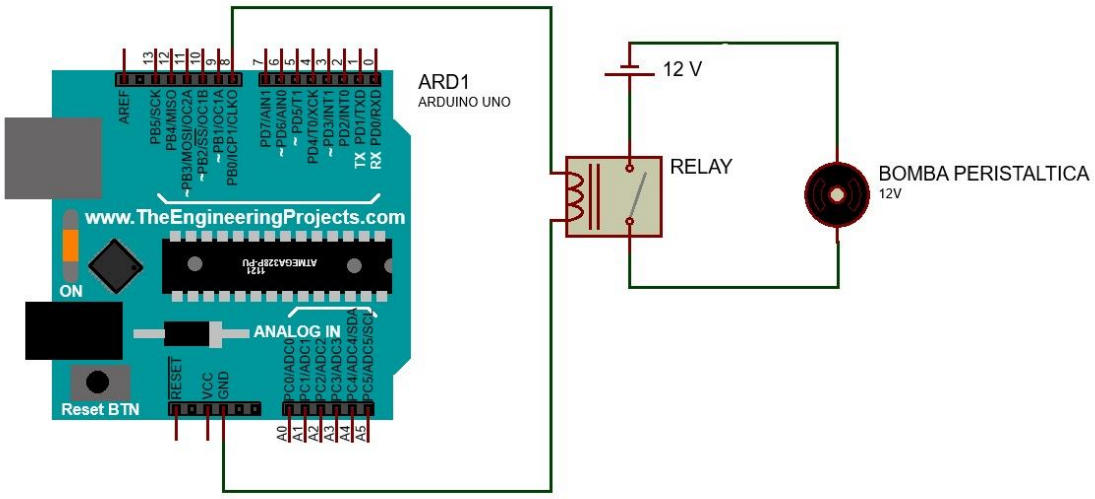


Ilustración 20. Conexión bomba - relé - arduino

La entrada del relé se encuentra conectado al pin digital 8, que será el que activará la salida del relé cuando deseemos que la bomba se ponga en funcionamiento y dispense el gel hidroalcohólico ya que cerrará el circuito con la tensión de alimentación de 12 V obtenida del pin Vin, que es el pin de la placa que suministra la misma tensión con la que se está alimentando la placa.

5.5- Comunicación con el usuario

Con el objetivo de que el usuario sepa que pasos ha de seguir, se han añadido dos dispositivos. Uno visual, como es un display LCD de 16x2 que mostrará la información necesaria para el usuario y un pequeño altavoz que reproducirá sonidos y melodías que complementará al display. La conexión de estos dispositivos con la placa se muestra en la *Ilustración 21*.

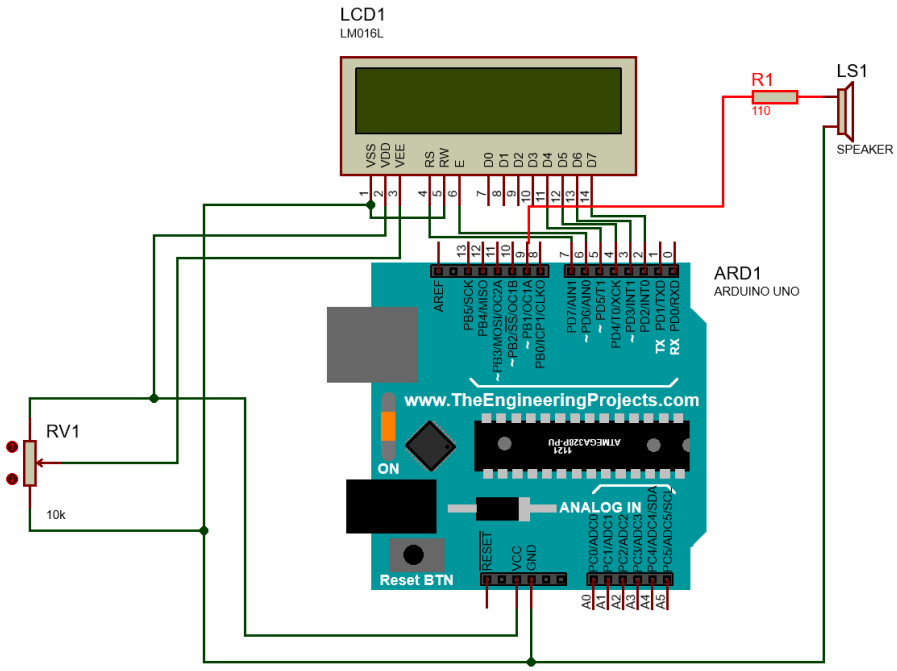


Ilustración 21. Conexión Display - altavoz - arduino

Los pines D4, D5, D6, D7, E y RS son aquellos que de forma digital recibirán la información que se desea escribir, posteriormente se explicará cómo se han programado. También se utiliza un potenciómetro conectado a VEE para regular el contraste de las letras y los demás pines son de alimentación a 5 V o comunes a GND.

Por otra parte, el altavoz ha sido conectado al pin digital 10 y se le ha añadido una resistencia de 110 Ω para limitar la corriente, ya que la resistencia interna del altavoz es de unos pocos ohm.

5.6- Servomotor

El servomotor se ha utilizado para la apertura de una puerta tradicional, este servomotor puede engancharse a un pasador o pestillo con eje de giro para mantener la puerta cerrada o abrirla.

El servomotor tiene 3 pines de conexión como podemos observar en la *Ilustración 22*, de ellos uno es el de alimentación, en nuestro caso es a 5 V porque no era necesario un par motor más grande, pero podría alimentarse hasta 12 V si se necesitase, de la misma forma que se ha conectado la bomba peristáltica.

El otro pin de conexión es el común (Gnd) y por último el pin de conexión de salida digital, el cual se ha conectado al pin 9 de la placa Arduino.

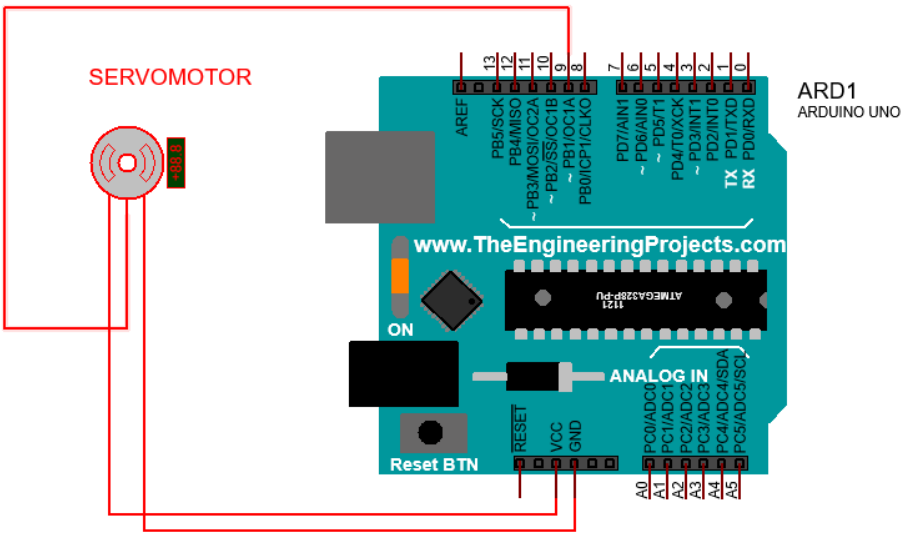


Ilustración 22. Conexión Servomotor - arduino

5.7- Modulo bluetooth

Como se ha comentado previamente, el módulo bluetooth se va a utilizar para comunicarse con el dispositivo Android del propietario del local. Las conexiones serán las que se muestran en la *Ilustración 23.*

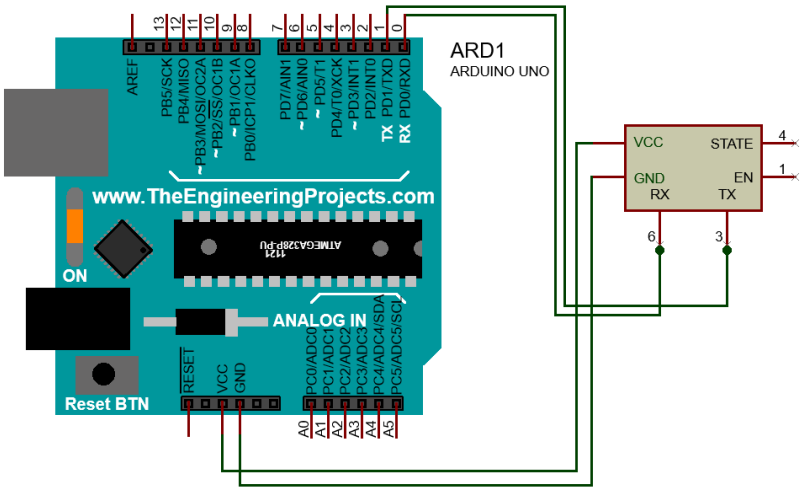


Ilustración 23. Conexión modulo bluetooth - arduino

5.8- Montaje final

El montaje final del proyecto, por tanto, ha quedado de la siguiente forma:

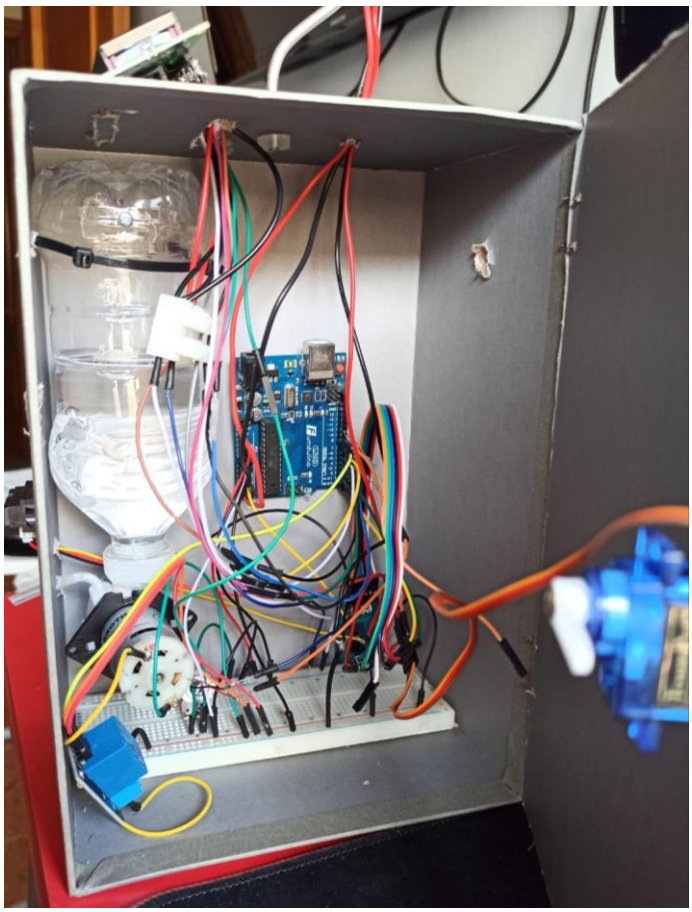


Ilustración 24. Interior del sistema

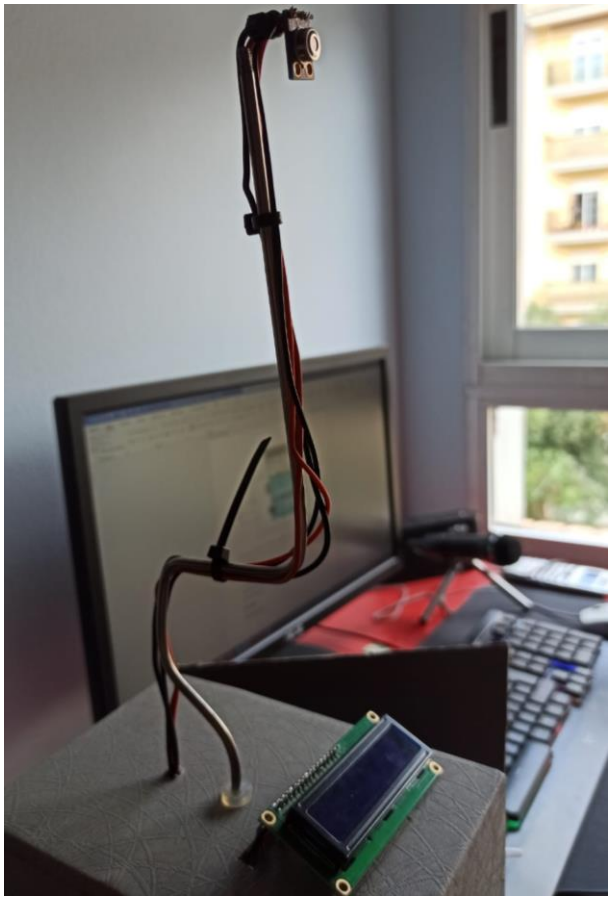


Ilustración 25. Parte superior del sistema

Ilustración 26. Frontal del proyecto

6. Programación

Respecto a cómo se va a abordar el desarrollo del código del proyecto, se ha decidido implementar una máquina de estados finita (MEF) formada por 5 estados distintos: *ESPERANDO*, *COLOCACIÓN*, *LECTURA*, *DISPENSACIÓN* y *PUERTA*. A continuación, se puede observar el diagrama de bloques en el cual se basa el código del proyecto.

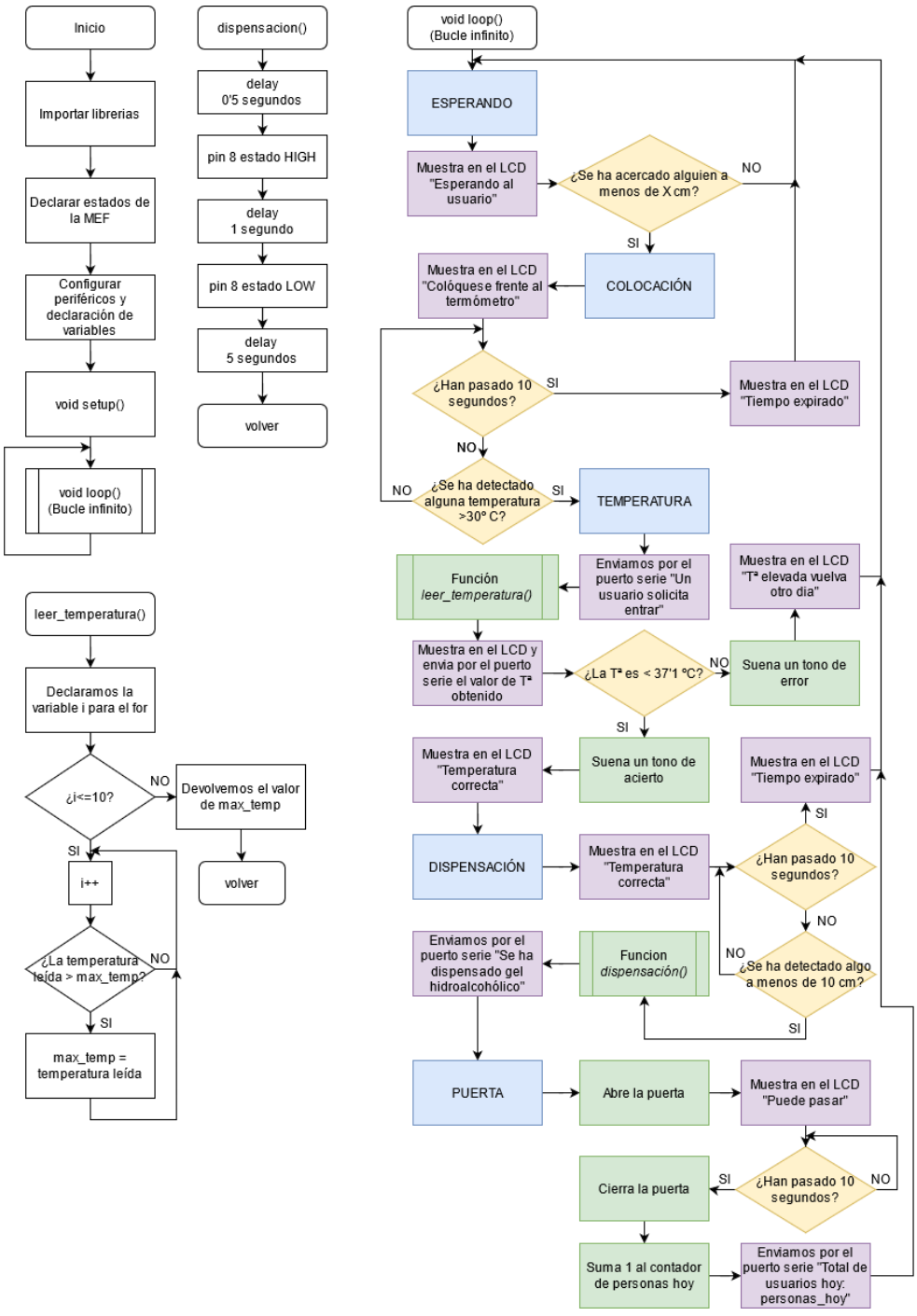


Ilustración 27. Flujograma del proyecto completo

6.1- Arduino IDE

6.1-1.Librerias y declaración de variables

Las librerías son necesarias si queremos un correcto funcionamiento de los diferentes sensores y actuadores. En este caso, se han utilizado las siguientes:

- ***LiquidCrystal.h.*** Esta librería es la que permite utilizar el display LCD con las funciones de configuración como es la función *LiquidCrystal lcd(7,6,5,4,3,2)* la cual permite configurar los pines entre paréntesis como los pines de transmisión de datos. Otras funciones que se usan posteriormente son por ejemplo *lcd.clear()*, *lcd.setCursor()* y *lcd.print()*.

- ***Wire.h***. Librería utilizada para la comunicación I2C entre el Arduino y el sensor de temperatura *mlx90614*

- ***Adafruit_MLX90614.h***. Esta librería está desarrollada por la compañía Adafruit, es la desarrolladora del sensor de temperatura utilizado en el proyecto. En esta librería se configura el dispositivo y nos proporciona las funciones a utilizar para el proyecto, entre ellas estarán *mlx.readAmbientTempC()*, *mlx.readAmbientTempF()*, *mlx.readObjectTempC()* y *mlx.readObjectTempF()*

- ***Servo.h.*** Librería incluida para controlar el servomotor. Las funciones a utilizar serán *servomotor.attach()* y *servomotor.write()*.

```
#include <LiquidCrystal.h>      // importa libreria para el display LCD
#include <Wire.h>               //para la comunicación I2C
#include <Adafruit_MLX90614.h>  //Libreria del sensor de temperatura MLX90614
#include <Servo.h>              //Libreria para controlar el servomotor

//declaramos los estados de la maquina de estados finita
#define ESPERANDO 1
#define COLOCACION 2
#define TEMPERATURA 3
#define DISPENSACION 4
#define PUERTA 5


Adafruit_MLX90614 mlx = Adafruit_MLX90614();
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);        // pines RS, E, D4, D5, D6, D7 de modulo 1602A
Servo myservo;                              // crea un objeto "servo" para controlar un servo
static unsigned char estado = ESPERANDO;    // estado inicial de la MEF

int buzzerPin = 10;             // pin para el altavoz
int sensor_proximidad = 0;      // pin analógico para conectar el sensor de proximidad sharp
int proximidad = 0;             // valor analógico obtenido del sensor de proximidad
unsigned long tiempo2 = 0;      // variable utilizada para refrescar el valor de millis()
float max_temp=0;               // variable que almacena la temperatura máxima
int personas_hoy = 0;           // variable que almacena las personas registradas en un dia
```

Ilustración 28. Fragmento de código con librerias y variables

Posteriormente, en la *Ilustración 28* se puede observar declarados los distintos estados que se utilizarán en el switch que conformará la máquina de estados.

Después se encuentran las funciones del sensor de temperatura, del LCD display y del servomotor mencionadas anteriormente para configurar estos. Y finalmente, encontramos las distintas variables declaradas para utilizar en el código principal. La variable estado = ESPERANDO se declara para iniciar el programa en el estado ESPERANDO, en el cual el sistema se encuentra esperando a que un usuario se acerque al sensor. La variable *buzzerPin* =10 se declara con la intención de configurar el pin 10 del Arduino posteriormente como salida conectada al altavoz. Después, se encuentra la variable *sensor_proximidad* = 0 que posteriormente indicará que esta recibe la información del pin analógico 0. La de proximidad = 0 registra el valor obtenido de la lectura analógica del detector de proximidad. La siguiente variable, tiempo2 = 0, actualizará la lectura de la función *millis()* cuando se necesite. Posteriormente la variable *max_temp* = 0 almacenará el valor de temperatura máximo que se usará posteriormente. Y finalmente, la variable *personas_hoy* = 0 llevará una cuenta de las personas que han entrado en el local en un día.

6.1-2.Configuración o void setup()

Este apartado de configuración se ejecutará una vez al inicio donde principalmente se definirá el funcionamiento de los pines de la placa Arduino, porque algunos se pueden definir como entradas y otros como salida. Además, aquí se debe definir la velocidad de transmisión de la comunicación serie, que en este caso será a 9600 baudios.

```
void setup() {
  Serial.begin(9600);          // inicia la comunicación serie a 9600 baudios
  lcd.begin(16, 2);            // inicializa a display de 16 columnas y 2 lineas
  lcd.clear();                 // Borra la pantalla LCD y coloca el cursor en la esquina superior izquierda.
  mlx.begin();                 // inicia el sensor
  myservo.attach(9);           // configura el servo en el pin 9
  pinMode(8, OUTPUT);          // configura el pin 8 como salida digital
  pinMode(buzzerPin, OUTPUT); // inicializo el pin del altavoz para que sea salida
  analogWrite(buzzerPin, 0);  // lo mismo con el pin del altavoz
}
```

Ilustración 29. Código del void setup

6.1-3.Funciones

Las funciones son herramientas de programación en fragmentos del programa que calculan un valor o ejecutan unas acciones de forma independiente al resto del código

principal. En el código que sea desarrollado hay diferentes funciones que ayudarán a compactar y simplificar el código.

6.1-3.1. Función de dispensación

En esta función con el nombre de *dispensación()* se encuentra una activación del pin digital 8 durante 1 segundo, el cual activará el relé para alimentar la bomba peristáltica durante 1 segundo, así dispensar una cantidad de entre 3 y 4 ml.

```
void dispensacion (void){
  delay(500);
  digitalWrite(8,HIGH);
  delay(1000);
  digitalWrite(8,LOW);
  delay(3000);
}
```

Ilustración 30. Función de dispensación

La función es muy sencilla. Cuando es llamada en el bucle principal, la acción que hará esta función es, primeramente, esperar medio segundo con la sentencia *delay(500)*, para asegurar que el usuario haya acabado de colocar las manos bajo el dispensador, posteriormente activará el pin digital 8 mediante la función *digitalWrite(8, HIGH)*, esta, establece el valor del pin digital 8 de salida a un valor alto, de 5 V, durante un segundo (*delay(1000)*) y después, establece el valor del pin digital 8 a un valor bajo de 0 V, cerrando así la dispensación. Y, por último, espera 3 segundos para hacer el siguiente paso.

6.1-3.2. Función de leer temperatura

```
float leer_temperatura (void){
  int i;

  for (i=0;i<=10;i++){
    lcd.setCursor(i, 0);
    lcd.print(".");

    if(mlx.readObjectTempC()> max_temp)
      max_temp=mlx.readObjectTempC();
      delay(300);
  }
  lcd.setCursor(3,1);
  return max_temp;
}
```

Ilustración 31. Funció de leer temperatura

En este proyecto una de las principales y más importantes características es la lectura de temperatura. La lectura de temperatura debe controlar que el usuario no supere los 37'1 ºC, y para evitar que haya errores, se ha desarrollado la función *leer_temperatura()*, la cual al llamarla realiza un bucle *for* donde se realizan 10 medidas consecutivas de la temperatura del usuario, con un intervalo de tiempo de 0'3 segundos y almacena el valor máximo leído. Se ha decidido establecer el valor máximo leído como el valor de temperatura final porque después de distintas pruebas, el sensor no detecta ninguna interferencia de ninguna fuente de calor superior a la que se mide habitualmente pero sí que suele detectar temperaturas por debajo de las que debería si el usuario no está correctamente situado frente al sensor, por lo que se toman 10 medidas durante 3 segundos para que se pueda hacer una buena selección de la temperatura.

Por último, en esta función se devuelve el valor de *max_temp*, es decir, cuando se llame a esta función en el código principal, devolverá el valor de la temperatura máxima registrada.

6.1-3.3. Funciones de sonido

```
void tono_acierto (void){
  tone(buzzerPin, 524, 100);
  delay(100);
  tone(buzzerPin, 660, 100);
  delay(100);
  tone(buzzerPin, 784, 100);
  delay(100);
  tone(buzzerPin, 1048, 300);
  delay(100);
}
void tono_error (void){
  tone(buzzerPin, 1048, 100);
  delay(100);
  tone(buzzerPin, 784, 100);
  delay(100);
  tone(buzzerPin, 660, 100);
  delay(100);
  tone(buzzerPin, 524, 300);
  delay(100);
}
```

Ilustración 32. Funciones de tonos de error y acierto

Se han desarrollado dos funciones de sonidos para acompañar a los mensajes del display cuando las temperaturas sean superiores o inferiores al valor límite establecido.

La melodía de la función *tono_acierto()* es un arpegio mayor ascendente de la tonalidad de DO mayor, que se relaciona con un sonido "alegre" por lo tanto, sonaría cuando la temperatura es inferior a 37'1 ºC, entonces el usuario podría pasar al local. Esta melodía, se ha diseñado conociendo las longitudes de onda de cada nota y añadiéndolas a la función *tone()*. También se le añade posteriormente la duración de la nota y después un delay con el tiempo de espera entre notas.

La melodía de la función *tono_error()* se usa en el momento en el que un usuario supera los 37'1 ºC y la melodía es la misma que la de *tono_acierto()* pero invertida. Es decir, un arpegio descendente, que se puede relacionar con algo más "triste" y que es erróneo.

6.1-4.Bucle infinito o void loop()

En esta parte el código principal se estará ejecutando continuamente mientras el Arduino esté en funcionamiento. Este código se puede dividir en distintos apartados para poder explicarlo más claramente.

6.1-4.1. Switch y estado ESPERANDO

```
void loop() {
  // se guarda en la variable estado los valores del sensor de proximidad
  proximidad = analogRead(sensor_proximidad);
  // se crea un switch con los distintos estados del sistema
  switch(estado){
  // Estado donde el sistema se encuentra esperando a que un usuario se acerque a la puerta
      case ESPERANDO:
  // Volvemos a poner la temperatura máxima registrada a 0 después haberla usado en la lectura de temperatura
            max_temp=0;
            lcd.setCursor(0, 0);                // Se muestra el mensaje deseado en el display
            lcd.print(" Esperando al ");
            lcd.setCursor(0, 1);
            lcd.print(" usuario ");


            if(proximidad>=120){                // Detecta un usuario a 50 cm del sistema
                tone(buzzerPin, 1048, 1000);    // El altavoz suena si alguien se acerca al sistema
                lcd.clear();
                tiempo2= millis();              // Actualizamos la variable que almacena el valor de millis()
                estado = COLOCACION;

            }
      break;
```

Ilustración 33. Codigo del switch y estado ESPERANDO

Como se puede observar en la *Ilustracion 33*, dentro del bucle infinito o código principal, se encuentra la variable proximidad almacenando constantemente el valor

de la lectura analógica del sensor, posteriormente este valor constantemente leído se usará para comprobar si alguien se ha acercado al sistema.

Luego, se inicia la declaración switch se utiliza como mecanismo de selección de estados, donde se irá saltando entre los distintos estados si se cumplen las distintas condiciones que se le impongan.

El primer estado que se encuentra y el que se ha determinado como inicial es el estado ESPERANDO. Cuando el switch entra en este caso, primeramente, establece la temperatura máxima a un valor de 0 al principio del programa, que será útil posteriormente. Después escribe en el display LCD el mensaje "Esperando al usuario". Durante todo este tiempo, se está comprobando si algún usuario se acerca a menos de 50 cm del sistema, si esto no sucede, el mensaje continua indefinidamente hasta que alguien se acerca. Cuando alguien se acerca, el altavoz produce una nota de 1 segundo de duración y una frecuencia de 1048 Hz, la cual es un Do6.

Posteriormente, se actualiza el valor de *tiempo2* y se iguala al valor de la función *millis*(). Esta función devuelve el valor en milisegundos que ha pasado desde que se ha iniciado el Arduino y posteriormente será usado. Por último, se actualiza el estado y se pasa al estado de COLOCACIÓN, donde el usuario deberá colocarse frente al sensor de temperatura.

6.1-4.2. Estado COLOCACIÓN

```
// El usuario se coloca para la lectura de su temperatura
case COLOCACION:
      lcd.setCursor(0, 0);
      lcd.print("Coloquese frente ");
      lcd.setCursor(0, 1);
      lcd.print("al termometro");
// Si durante 10 segundos no se cumple lo siguiente volveremos al inicio
      if(millis()-tiempo2<=10000){
// Si el sensor de temperatura detecta algún valor superior a 30° en estos 10 segundos se hace una lectura de temperatura
          if(mlx.readObjectTempC()>=30.00){
              lcd.clear();
              tiempo2=millis();            // Actualizamos la variable que almacena el valor de millis()
              estado = TEMPERATURA;
        }
      }
      else {
        lcd.clear();                       // Volvemos al estado inicial por si se ha detectado algun usuario por error
        lcd.setCursor(0, 0);
        lcd.print("Tiempo expirado");
        delay(2000);
        lcd.clear();
        estado = ESPERANDO;
      }
  break;
```

Ilustración 34. Código de estado de COLOCACIÓN

Continuando dentro del bucle principal y del switch, pasamos al estado de COLOCACIÓN, dentro de este estado, al principio, podemos encontrar que el display muestra un mensaje con el texto “Colóquese frente al termómetro”. Después se crea una condición, la cual, después de a ver detectado a alguien acercarse, pone un contador de 10 segundos para el usuario, esto se hace restando el valor de la función *millis()*, que tiene el contador de milisegundos funcionando constantemente, con el valor que se ha almacenado anteriormente de *tiempo2* . Si en ese tiempo el sensor detecta una lectura de temperatura de objeto superior a 30ºC, se pasará al estado de lectura de temperatura (TEMPERATURA), no sin antes actualizar el valor de la variable *tiempo2* = *millis*() para su posterior uso. Pero si pasan los 10 segundos sin ninguna lectura superior a estos 30ºC, significará que alguien se ha acercado al detector de proximidad, pero no quería acceder al local, y se volverá al estado inicial de ESPERANDO. Además, se mostrará un mensaje en el display con el texto “Tiempo expirado”.

6.1-4.3. Estado TEMPERATURA

```
case TEMPERATURA:
        Serial.println("Un usuario solicita entrar");
        lcd.print(leer_temperatura());lcd.print("C");
        Serial.print("  Su temperatura es de: ");
        Serial.print(leer_temperatura());
        Serial.println(" °C");

        if(leer_temperatura()<37.10){    // Si la función de lectura de temperatura da un valor inferior a 37'1°C el valor es correcto
            lcd.clear();
            tono_acierto();              // El altavoz hace un sonido de un arpegio ascendente de Do Mayor, como un sonido de acierto
            lcd.setCursor(0, 0);
            lcd.print("Temperatura ");
            lcd.setCursor(0, 1);
            lcd.print("correcta ");
            delay(1000);
            lcd.clear();
            tiempo2=millis();
            estado = DISPENSACION;       // Se pasa al estado de dispensar gel hidroalcoholico
          }
        else{                            // Si la función de lectura de temperatura da un valor superior a 37'1°C el valor es demasiado elevado
            lcd.clear();
            tono_error();                // El altavoz hace un sonido de un arpegio descendente de Do Mayor, como un sonido de error
            lcd.setCursor(0, 0);
            lcd.print("T Elevada");
            lcd.setCursor(0, 1);
            lcd.print("Vuelva otro dia");
            delay(5000);
            estado = ESPERANDO;          // Se vuelve al estado inicial para que pase otro usuario
        }

    break;
```

Ilustración 35. Código del estado DISPENSACIÓN

En este estado del switch, se encuentra al principio de todo, el primer envió de datos por el puerto serie que se había configurado a 9600 baudios. Este envió de datos será una frase para la aplicación, la cual mostrará un mensaje diciendo que “Un usuario solicita entrar”. Posteriormente, se muestra en el display LCD el valor obtenido de la

lectura de temperatura mediante el uso de la función *leer_temperatura()* explicada anteriormente.

Luego, se hará otro envío de datos por el puerto serie con la información del valor de la temperatura del usuario y se mostrará ese valor en la aplicación del dispositivo Android.

Cuando la temperatura haya sido comparada con el valor límite establecido de 37'1ºC, se darán dos casos. El primero, es el caso en el que la lectura de temperatura es inferior a 37'1ºC y, por lo tanto, un valor correcto para acceder. Entonces, sonaría el altavoz con la función de *tono_acierto()*, en el display LCD se mostrará un mensaje de temperatura correcta. Por último, en esta condición, se actualiza el valor de *valor2* como se ha hecho constantemente y, finalmente, se actualiza el estado a DISPENSACIÓN.

Si, por el contrario, la temperatura es igual o superior a 37'1 ºC, sonará por el altavoz la función *tono_error()*, se mostrará en el display un mensaje de "Tª elevada, vuelva otro día" y el estado volverá a ser ESPERANDO, a la espera de un nuevo usuario.

6.1-4.4. Estado DISPENSACIÓN

```
case DISPENSACION:
        lcd.setCursor(0, 0);
        lcd.print("Se va a");
        lcd.setCursor(0, 1);
        lcd.print("dispensar el gel");

        if(millis()-tiempo2<=10000){
          if(proximidad>=500){       // Cuando se detectan las manos a menos de 10 cm del sensor pasa a dispensar
            dispensacion();
            lcd.clear();
            tiempo2=millis();
            Serial.println("    Se ha dispensado gel hidroalcohólico");
            estado = PUERTA;                 // Pasamos al estado de apertura de la puerta
  }
        }
        else{
            lcd.clear();
            lcd.setCursor(0, 0);
            tono_error();
            lcd.print("Tiempo expirado");
            delay(2000);
            lcd.clear();
           estado = ESPERANDO;     // Si no se dispensa el gel hidroalcohólico no se puede pasar y volvemos al inicio
}
  break;
```

Ilustración 36. Código del estado DISPENSACIÓN

El estado actual es el de dispensación de gel hidroalcohólico. Aquí se pone una cuenta atrás de 10 segundos como se ha hecho anteriormente restando el valor de la

función *millis()* menos el valor de *tiempo2*. Durante el tiempo de la cuenta atrás, si se detecta algún movimiento a menos de 10 cm de proximidad, suponemos que serán unas manos, por lo que, llamaremos a la función *dispensacion()*, comentada anteriormente. Después de esto se envía un mensaje a la aplicación móvil con el mensaje "Se ha dispensado gel hidroalcohólico" y se actualiza el estado a PUERTA.

Si, por el contrario, pasan los 10 segundos sin que se dispense el gel hidroalcohólico, expirará el tiempo de dispensación de gel y se volverá al estado principal.

6.1-4.5. Estado PUERTA

```
   case PUERTA:
          if((millis()-tiempo2)<=10000){   // Se abre la puerta durante el tiempo que queramos establecer
            myservo.write(180);
            lcd.setCursor(0, 0);
            lcd.print("Puede pasar");

          }
          else{
            myservo.write(90);
            personas_hoy = personas_hoy +1;    // Añade un nuevo usuario al contador de usuarios
            Serial.print("      Total de usuarios hoy: ");Serial.println(personas_hoy);
            estado=ESPERANDO;}
     break;

  }

}
```

Ilustración 37. Código del estado PUERTA

Este se trata del último estado, donde se abrirá la puerta con la función *myservo.write()* poniendo el servomotor en el ángulo deseado durante el tiempo deseado, en este caso son 10 segundos pero pueden ser los necesarios dependiendo de la puerta del local.

Después de haber superado el tiempo establecido, el servomotor vuelve a la posición original, se añade un usuario al contador de usuarios y se envía por comunicación serie a la aplicación del dispositivo el número de usuarios que han pasado en el día al interior del local. Finalmente, se vuelve al estado inicial.

6.2- MIT App Inventor

Tal y como se ha comentado anteriormente, MIT APP Inventor se trata del software que se va a utilizar para desarrollar la aplicación del dispositivo Android. Este software online consta de 2 partes, la parte diseño visual, y la parte de programación por bloques.

6.2-1.Diseño visual

El diseño de la aplicación se realiza añadiendo a los componentes de la pantalla los distintos elementos deseados en una lista de selección. El primer elemento que añadir es una disposición horizontal, donde dentro de esta, se añadirá un selector de lista para seleccionar el dispositivo bluetooth a conectar y un botón para la desconexión del dispositivo. A continuación, se añadirá una disposición vertical en la cual, se añadirá una etiqueta, la que escribirá la información que se irá recibiendo del Arduino. Por último, se añadirán 4 elementos fuera de disposiciones, un botón para borrar el texto recibido de la placa, un cliente bluetooth que se trata del dispositivo al que se está conectado, un timer y un notificador. Todos ellos se explicarán en el diagrama de bloques.

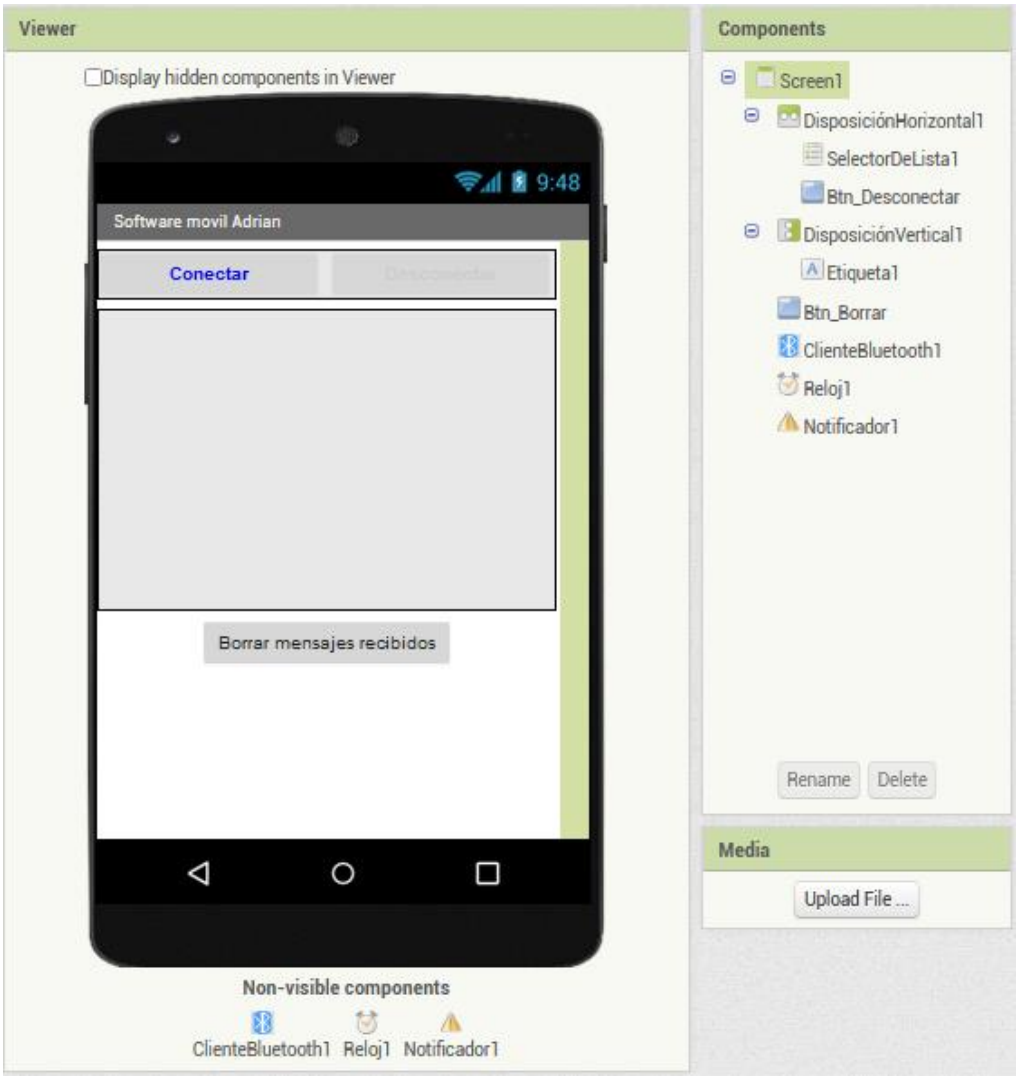


Ilustración 38. Diseño de la interfaz

6.2-2.Bloques

Ilustración 39. Bloques de selector de lista antes de elegir

En el primer bloque, cuando se entra al selector de lista, se muestran los elementos bluetooth y sus direcciones detectados por el dispositivo

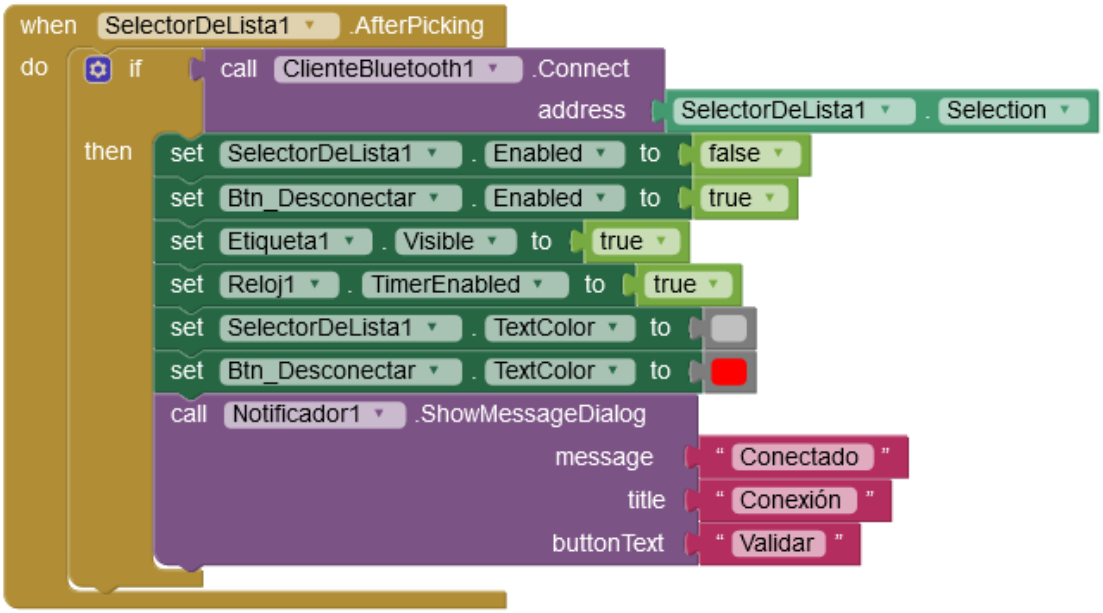


Ilustración 40. Bloques de selector de lista después de elegir

Después de seleccionar el dispositivo al que queremos conectarnos, en nuestro caso, al módulo bluetooth HC-05 que se mostrará en la lista de selección, se desactivará el selector de lista, para que no se pueda conectar desde esta aplicación a otro dispositivo, y activará el botón de desconexión, además, también se hará visible la etiqueta donde se escribirá el código y se iniciará el timer.

El selector de lista y el botón de desconexión también cambiarán de color para diferenciar el cambio de estado. Y, por último, se mostrará una notificación informando de que se ha conectado el dispositivo con el módulo bluetooth.

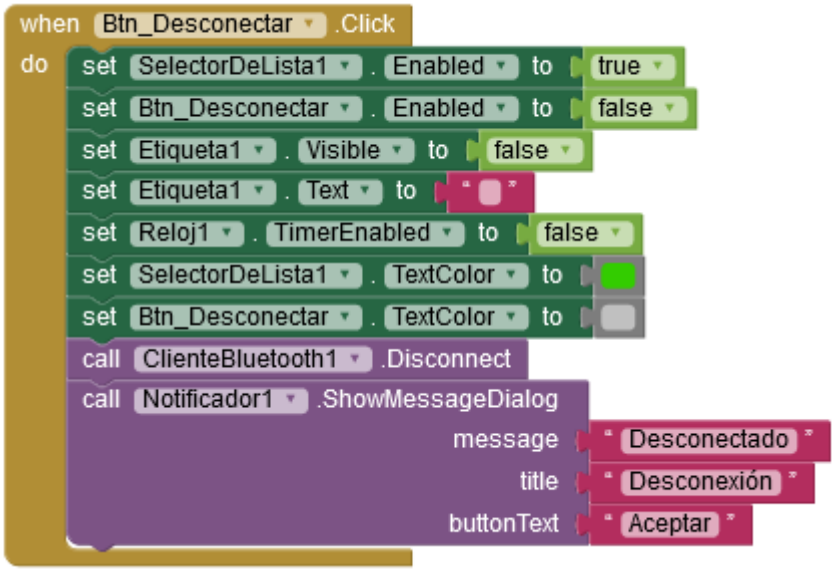


Ilustración 41. Bloques del botón de desconectar

El siguiente bloque de configuración que se encuentra es el del botón de desconectar, este es más sencillo que el anterior. Cuando se pulsa este botón, se vuelven a invertir los estados y los colores de selector de lista y botón de desconexión. Además, se apaga el reloj y se borra todo lo que se muestre en la etiqueta de datos. Finalmente, en este módulo, se desconecta del cliente bluetooth y se muestra un notificador con el mensaje de “Desconectado”.

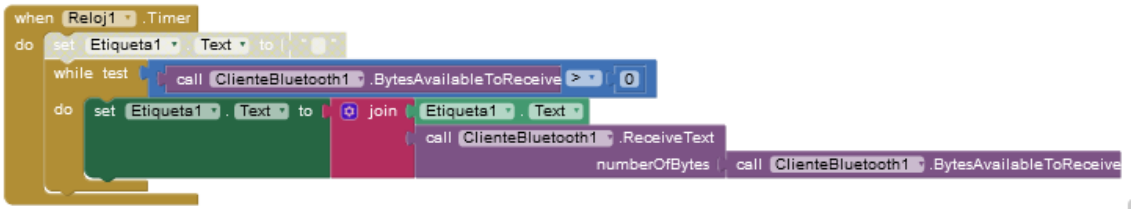


Ilustración 42. Bloques del timer

Dentro de esta función de timer, mientras se encuentre activa, se realiza constantemente una lectura del cliente bluetooth, si hay bytes disponibles para la recepción de datos, se mostrará en la etiqueta toda la información transmitida por el módulo bluetooth de Arduino. En este caso, los mensajes enviados en el código de Arduino con la función *Serial.print()*.

Ilustración 43. Bloques del botón de borrar

Finalmente, en los bloques, encontramos el botón de borrar, que limpia todo lo almacenado en la etiqueta.

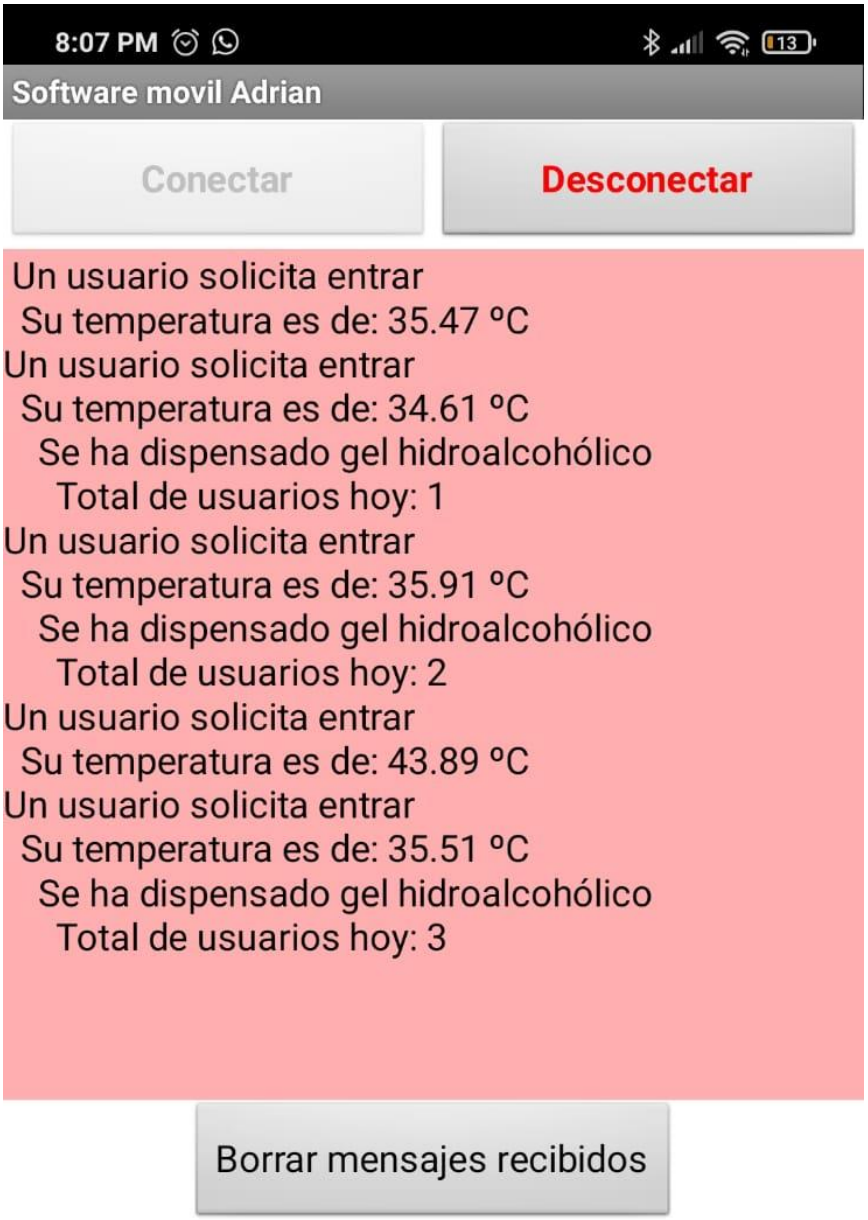


Ilustración 44. Captura de pantalla de la aplicación funcionando

En la *Ilustración 44* se puede ver un ejemplo de los mensajes mostrados en la aplicación después del control a varios usuarios. Aquellos que tenían una temperatura correcta y se han dispensado el gel, han sumado 1 usuario al contador, pero si alguno ha superado la temperatura o no se ha dispensado el gel, no ha podido pasar y no ha entrado en el recuento de usuarios.

7. Problemas y mejoras

Durante el diseño del proyecto han surgido varios problemas fácilmente solucionables con más tiempo o recursos. El principal problema ha sido con el sensor de temperatura, ya que el objeto a medir la temperatura, en este caso la frente del usuario debe colocarse muy cerca de éste, ya que, tiene un campo de visión o FOV de 90º, y es un ángulo tan amplio que, si no se está lo suficientemente cerca, toma la temperatura de otros puntos fuera de la frente. Esto, sería fácil de solucionar con un sensor con un menor FOV, pero estos son más difíciles de conseguir.

El otro problema que se ha encontrado es la corta longitud del tubo de silicona incluido con la bomba peristáltica, lo que ha hecho que la botella de solución hidroalcohólica se haya tenido que colocar de una forma diferente a la planeada. Si se hubiera encontrado un tubo de silicona del tamaño deseado esto no habría sido ningún problema.

8. Planos

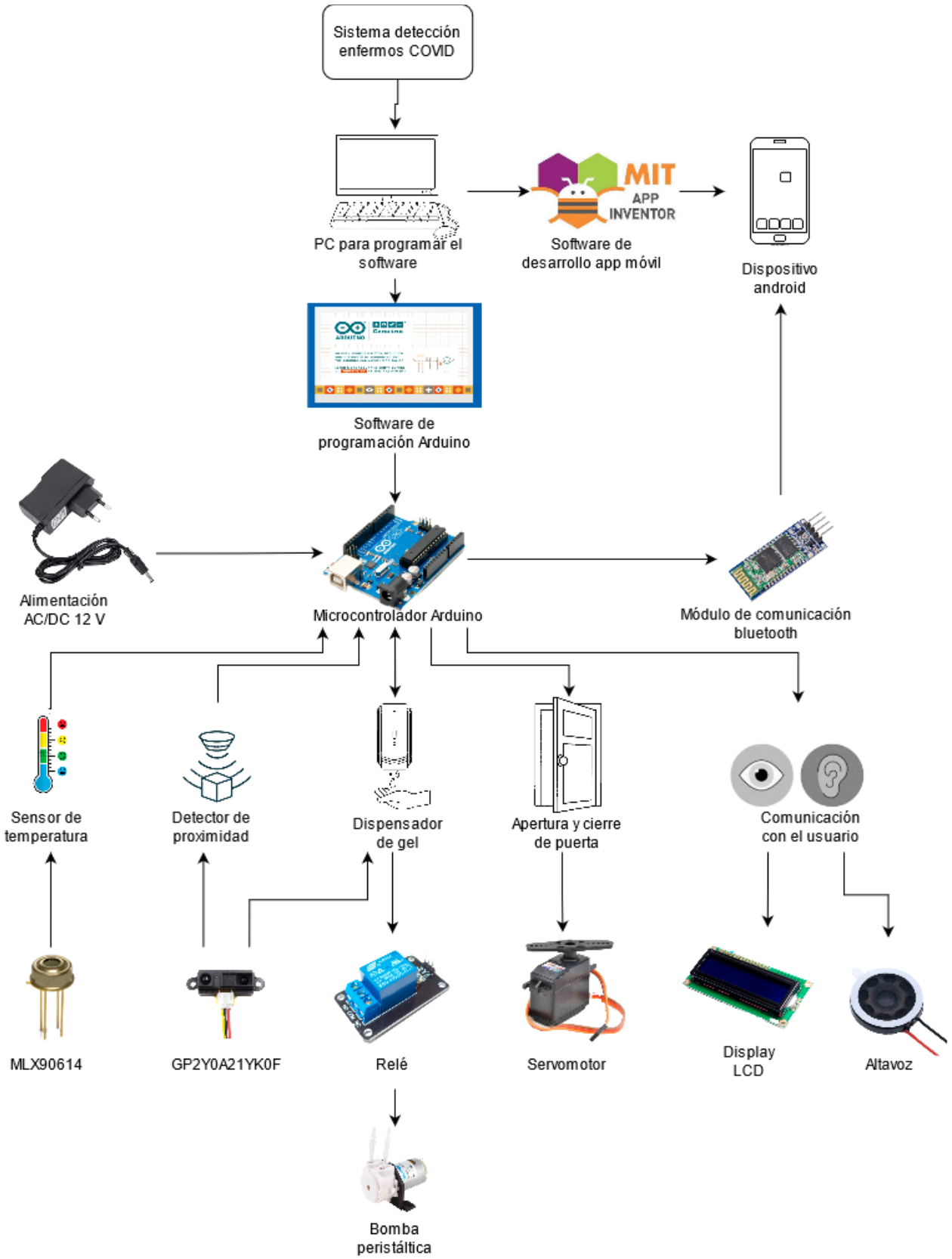
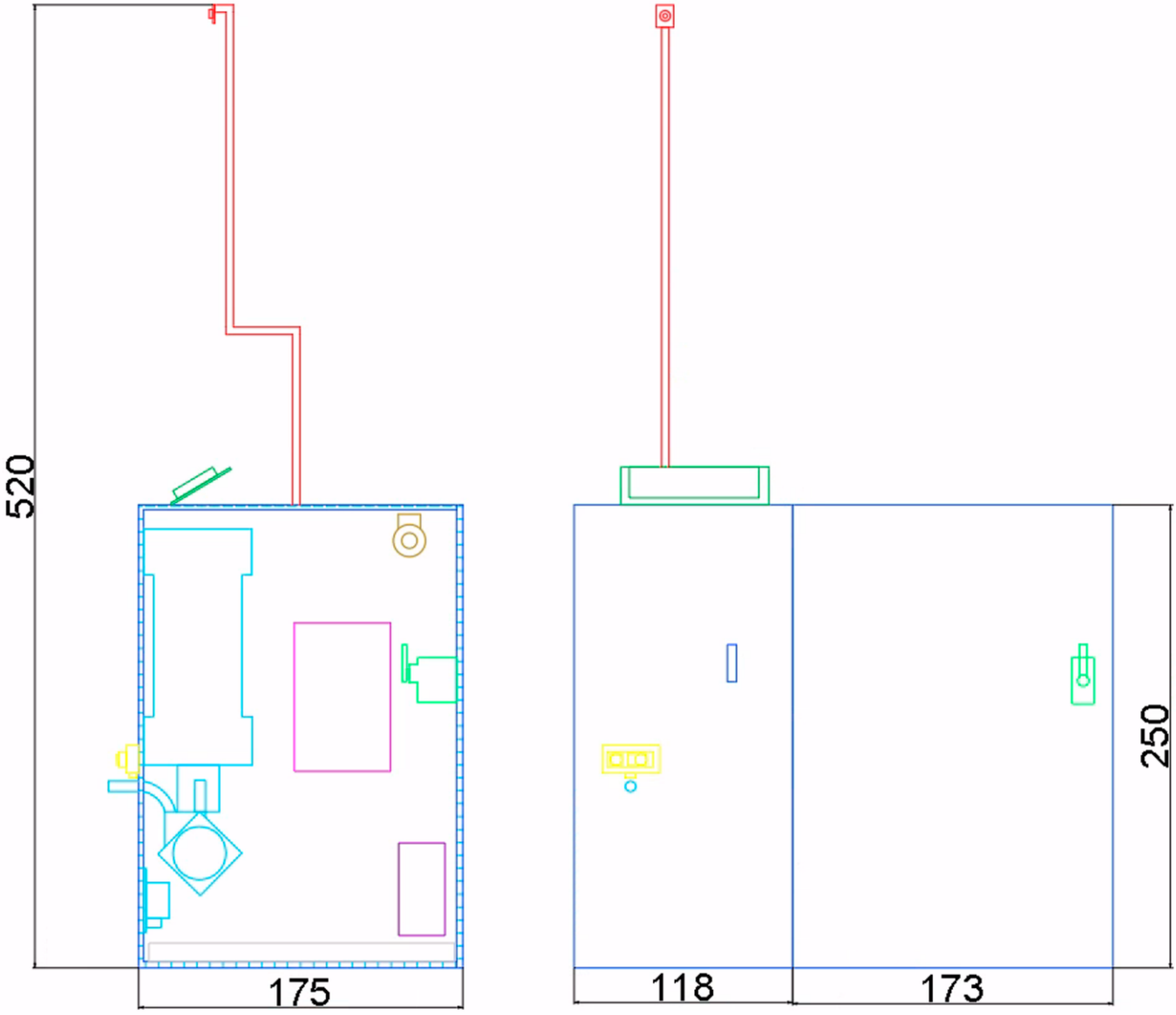
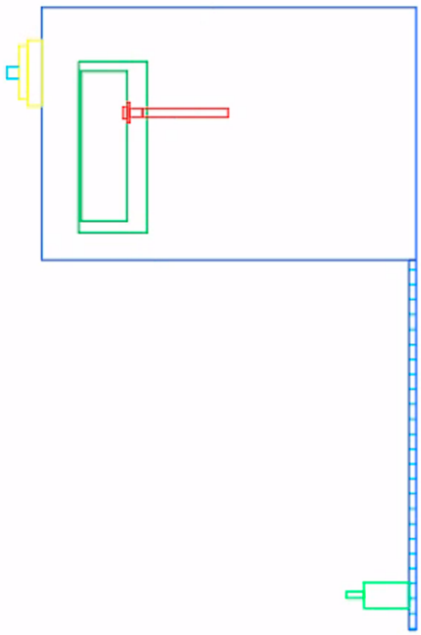


Ilustración 45. Organigrama del montaje





Proyecto: Sistema de detección enfermos COVID		
Alumno: Adrian Gervasini Navarro		
Plano: Vistas del proyecto con componentes		
01/07/2021	Nº: 001	1:3,5

9. Presupuesto

Respecto a la elaboración del presupuesto, se van a tener en cuenta el tiempo que se ha necesitado para realizar el diseño, los recursos materiales utilizados y la amortización de los equipos. A continuación, se muestra la tabla de precios de los materiales utilizados.

Item	**Unidades**	**Precio/ud.**	**Precio (€)**
Arduino UNO	1	4.52	4.52
Alimentación 12 V	1	3.03	3.03
Mlx90614	1	10.71	10.71
GP2Y0A21YK0F	1	4.92	4.92
Servomotor	1	2.51	2.51
Bomba peristáltica	1	7.76	7.76
Display LCD	1	2.25	2.25
Altavoz	1	1.12	1.12
Módulo bluetooth	1	8.26	8.26
Relé	1	0.60	0.60
Resistencia	1	1.03	1.04
Potenciómetro	1	1.55	1.55
Protoboard	1	1.10	1.10
Cables de conexión	2 m	1.32	2.36
Caja	1	2.48	2.48
Total: 53.80 €			

Tabla 3. Presupuesto de recursos materiales

Posteriormente, se va a realizar una tabla con el presupuesto de los costes del diseño del proyecto. Para calcularlo se va a tener en cuenta que el proyecto ha sido diseñado y construido únicamente por una persona, durante los últimos meses y se han trabajado unas 120 horas en este.

Diseño	Unidades (h)	Precio por unidad (€)	Total (€)
Personal	120	25	3000

Tabla 4. Presupuesto de personal

Además, se va a incluir el presupuesto de la amortización de los equipos utilizados. Estos equipos han sido adquiridos exclusivamente para la realización del proyecto.

Equipo	Vida útil aprox. (h)	Precio (€)	Tiempo de uso (h)	Precio/hora (€/h)	Gasto total (€)
Multímetro	300	16.53	1	0.06	0.06
Soldador	500	24.80	3	0.05	0.15
Pistola de silicona	3000	12.50	0.5	0.01	0.005
Destornillador	200	4.10	0.5	0.02	0.01
Total: 0.23 €					

Tabla 5. Presupuesto de la amortización de equipos

Por último, obtendremos el valor total del presupuesto del proyecto añadiendo el IVA.

Proyecto	Coste (€)	IVA 21 % (€)	Total (€)
Materiales	53.80	11.30	65.10
Diseño	3000	630	3630
Equipos	0.23	0.05	0.28
Total: 3695.38 €			

Tabla 6. Presupuesto total del proyecto con IVA

Anexos

Listado de ilustraciones

Listado de tablas

Listado de gráficos

Bibliografía

https://news.google.com/covid19/map?hl=es

https://arduinodhtics.weebly.com/iquestqueacute-es.html

https://www.elprocus.com/different-types-of-arduino-boards/

https://es.omega.com/prodinfo/termometros-infrarrojos.html

Tormos, A (2021). Sensores e instrumentación Virtual. *Sensores de temperatura.* UPV, Valencia, España.

Tormos, A (2021). Sensores e instrumentación Virtual. *Sensores ópticos.* UPV, Valencia, España.