

# Smart Toolbox

## Carpeta técnica

2022



**SMART**  
T O O L B O X

Web: <https://smart-toolbox.tk>

Trello: <https://trello.com/b/zU9TEU8G/kanban>

Github: [https://github.com/impatrq/722b\\_smart\\_toolbox](https://github.com/impatrq/722b_smart_toolbox)

Instagram: <https://www.instagram.com/smart.toolbox.2022>

SpaceApps: <https://2022.spaceappschallenge.org/challenges/2022-challenges/create-your-own-challenge/teams/smart-toolbox/project>

## Integrantes:

- ESQUIVEL Agustín Lautaro 7mo 2da Aviónica
- HERRERA Lucas Leandro 7mo 2da Aviónica
- MARIANI Ramiro Uriel 7mo 2da Aviónica
- MARTÍNEZ Joaquín Sebastián 7mo 2da Aviónica
- TORRES Federico Iván 7mo 2da Aviónica
- VANORE Miqueas Juan Bautista 7mo 2da Aviónica



## Docentes responsables

- MEDINA, Sergio
- BIANCO, Carlos
- MINUCCI, Mauro
- ALEGRE, Marcos

## Esfuerzo del proyecto:

- 28 semanas de trabajo
- 15 horas semanales repartidas entre 6 personas
- 420 horas de trabajo en total

# Índice

1. Objetivo
2. Descripción general
3. Alcance
4. Segmento destino
5. Captura del proyecto
6. Diagrama en bloques
7. Resultado conseguido
  - a. La caja de herramientas
  - b. Aplicación móvil
  - c. Interfaz gráfica
  - d. Base de datos
8. Software
  - a. Diagrama en bloques
  - b. Lenguajes de programación
  - c. Código
  - d. Interfaces visuales
  - e. Estructura de datos
9. Sistema embebido
  - a. Microcontrolador

- b. Software utilizados en el desarrollo
- c. Diagrama en bloques
- d. Lenguaje de programación
- e. Código
- f. Periféricos utilizados
- g. Estructuras de datos

## 10. Electrónica

- a. Diagrama en bloques de las partes
- b. Software utilizado para CPB y esquemáticos
- c. Esquemáticos
- d. PCB
- e. Modelo 3D de cada PCB
- f. Fuente de alimentación
- g. Especificaciones técnicas de cada componente

## 11. Estructura

- a. Diagrama general de la estructura
- b. Software de diseño utilizado
- c. Descripción de la estructura
- d. Imágenes de la estructura

## 12. Anexo

## 1. Objetivo

Diseñar y construir un sistema que pueda ser integrado a una caja de herramientas, que indique al usuario qué herramientas faltan, activando una alarma en caso que falte alguna de la dotación a fin de evitar que las mismas se olviden sitios que puedan generar un riesgo para el usuario.

## 2. Descripción general

Smart Toolbox es un sistema compuesto de 4 partes:

- Una aplicación móvil: es un software que le permite al usuario:
  - Registrarse con nombre y apellido.
  - Escanear el código QR de una caja de herramientas.
  - Ver las tareas a realizar.
  - Ver las herramientas faltantes vinculadas a una caja.
- Una interfaz gráfica: es una aplicación de escritorio que permite al encargado del taller observar información detallada de cada caja, subir tareas para cada usuario y decidir cuándo se deben guardar las cajas de herramientas, para que el sistema Smart Toolbox se active.
- Las cajas de herramientas: Éstas envían su estado y las herramientas faltantes a la base de datos. Posee un código QR en su exterior para que un usuario se pueda vincular a ésta.
- Una base de datos: Es la conexión entre todas las demás partes de Smart Toolbox.

## 3. Alcance

El alcance máximo de Smart Toolbox es poderse implementar en cualquier ámbito, desde la medicina hasta la educación, donde se utilicen herramientas, y de este modo reducir significativamente el riesgo de accidentes y costos de utensilios perdidos.

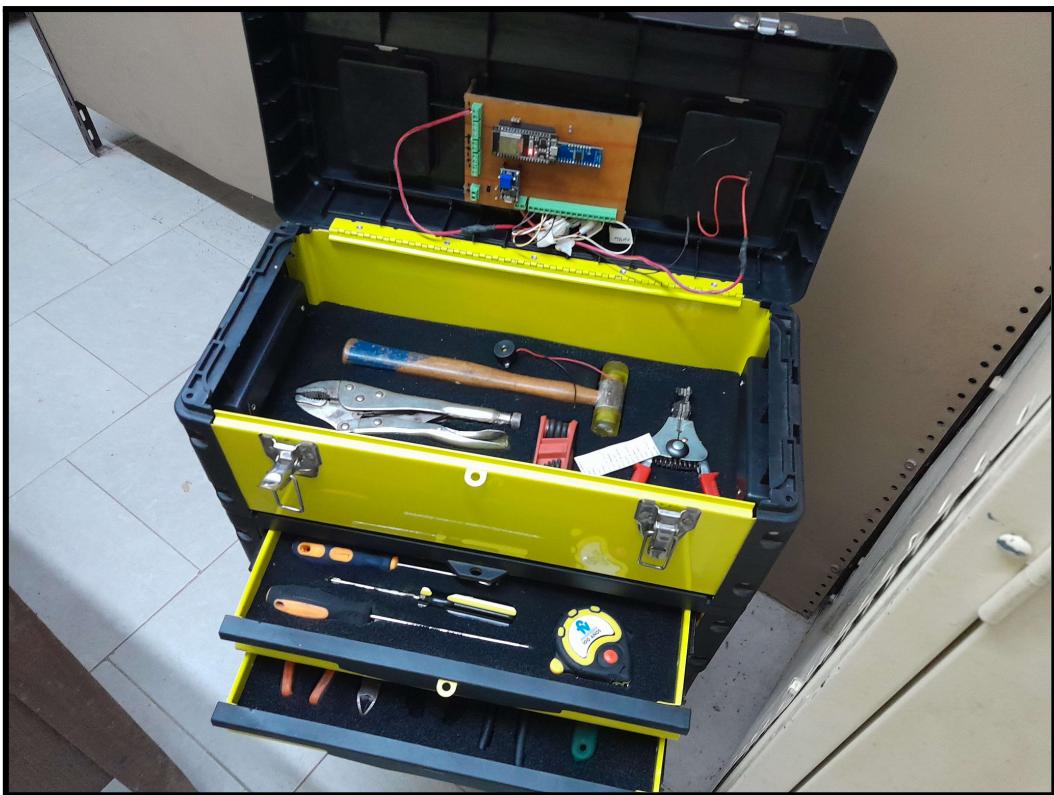
## 4. Segmento destino

El segmento de usuario principal que buscamos son talleres donde se haga uso de una caja de herramientas. Sin embargo, esto no excluye a aquellos

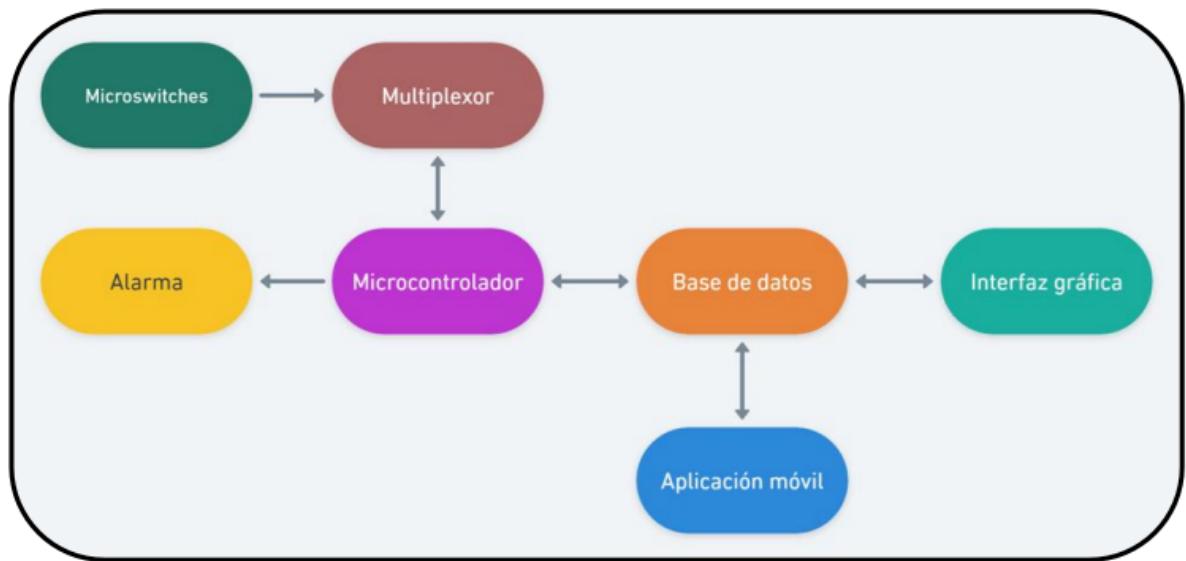
ámbitos en los que se utilicen instrumentos, y la pérdida de éstas puedan provocar un accidente o costos adicionales.

## 5. Captura del proyecto





## 6. Diagrama en bloques



## 7. Resultado

El resultado fue poder crear un prototipo que cumple la siguiente funcionalidad:

- a. La caja de herramientas

Después de terminar la jornada de trabajo, el usuario deberá guardar todas las herramientas en la caja y, de este modo, evitar el extravío de éstas. En el caso contrario, al cerrar todos los cajones, el sistema detectará que la caja se terminó de usar y hay instrumentos faltantes, avisándole de esto mediante un buzzer de 24V regulado con una fuente step-up.

La Smart Toolbox espera cada una cierta cantidad de segundos, mediante el método “get” de urequests, una señal de que la caja debe guardarse. Cuando esta señal es *True*, empieza la detección automática de herramientas: El código del microcontrolador posee una clase *Herramientas* que permite construir objetos con los siguientes atributos: un nombre y una combinación.

El microcontrolador empezará a iterar una lista de objetos (las herramientas) y por cada iteración cambiará el selector del multiplexor; para posteriormente leer el pin que indicará si el microswitch asociado a este selector está pulsado o no. En caso que sí lo estuviera, remueve la herramienta de la lista global “herramientas faltantes”. Si se detecta que no está pulsado, agrega la herramienta a la lista global previamente mencionada y activará la alarma cuando los cajones están cerrados.

Luego de iterar todas las salidas del multiplexor, enviará la lista a la base de datos.

Este ciclo continuará hasta que el sistema se apague o no se reciba la señal de guardado.

## b. Aplicación móvil

En primer lugar, el usuario debe registrarse con nombre y apellido pulsando el botón en la esquina superior derecha. Mediante el hook de React “useContext”, otros componentes de la aplicación pueden acceder a este usuario.

La aplicación móvil, desarrollada en Ionic con React, tiene 3 pestañas:

- 1) Tareas: En esta pestaña se observan las tareas a realizar en el día.

- 2) Herramientas: En esta pestaña se observan las herramientas faltantes vinculadas a una caja.
- 3) Escáner QR: Esta pestaña permite escanear una caja para vincularla a un operario.

### c. Interfaz gráfica

La interfaz gráfica es una aplicación de escritorio desarrollada en React y ElectronJS.

Posee un “switch” que permite cambiar el estado de las herramientas a modo “guardar”. Esto lo hace cambiando el estado de una de las llaves principales de la base de datos.

Posee dos pestañas muy importantes:

- 1) Tareas: Se puede crear o seleccionar un usuario y asignarle tareas. Las tareas actuales se muestran en pantalla, y es posible agregar más o eliminarlas para cada usuario.
- 2) Cajas: Esta pestaña contiene una tabla que muestra cada caja de herramientas, sus operarios vinculados, su estado y las herramientas faltantes en ésta, todo esto en una tabla dividida por columnas.

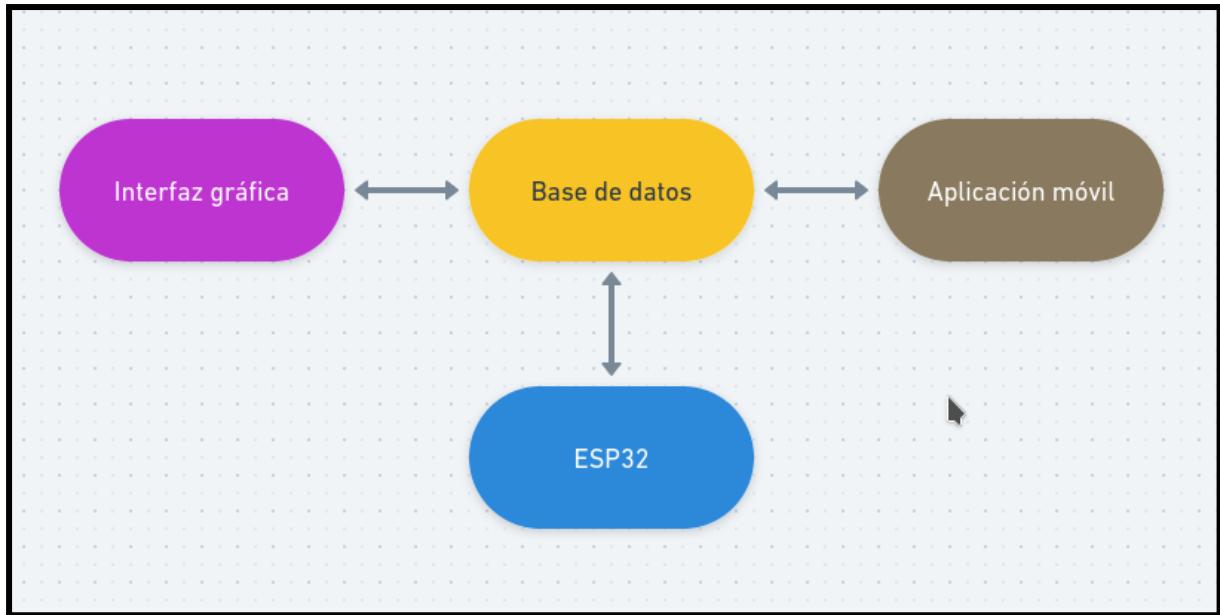
Adicionalmente, cada pestaña posee un botón en la esquina superior izquierda para volver a la pantalla de inicio.

### d. Base de datos

La base de datos de Smart Toolbox es una Realtime Database de Firebase Firestore. Esta almacena los datos del sistema en un árbol de objetos, para ser recolectados o modificados por los demás componentes.

## 8. Software

### a. Diagrama en bloques



## b. Lenguajes de programación

Tanto para la interfaz gráfica como para la aplicación móvil, utilizamos React (tecnología web), que es una librería de Javascript que me permite trabajar con componentes. En el caso de la aplicación móvil utilizamos Ionic y para la interfaz gráfica utilizamos ElectronJS.

## c. Código

Éste es el código que se utiliza para tomar las tareas de la base de datos y se utiliza en casi todas las partes de la aplicación con variaciones.

```
useEffect(() => {
  const personasRef = ref(db, "sector1/personas");
  onValue(personasRef, (snapshot) => {
    const raw_data = snapshot.val();
    setOperarios(Object.keys(raw_data));
    setTareas(raw_data);
  });
}, []);
```

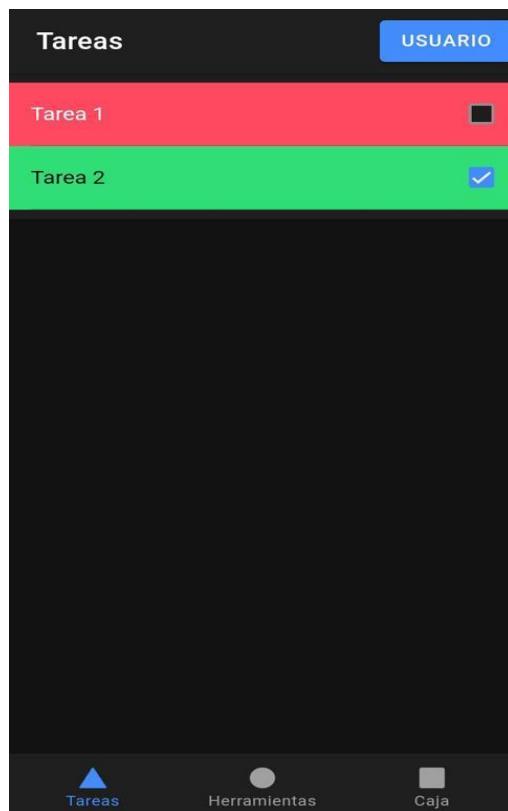
En éste código se agregar un usuario a la base de datos, se utiliza en todos los sistemas con distintas variaciones:

```
const handleAddUser = () => {
  const key = `/sector1/cajas/${newBox}`;
```

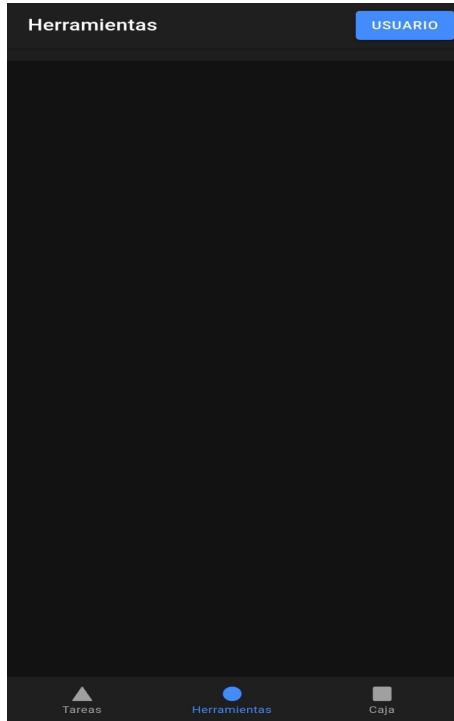
```
update(ref(db), { [key]: { missing_tools: "", state: false } });
handleClose();
};

};
```

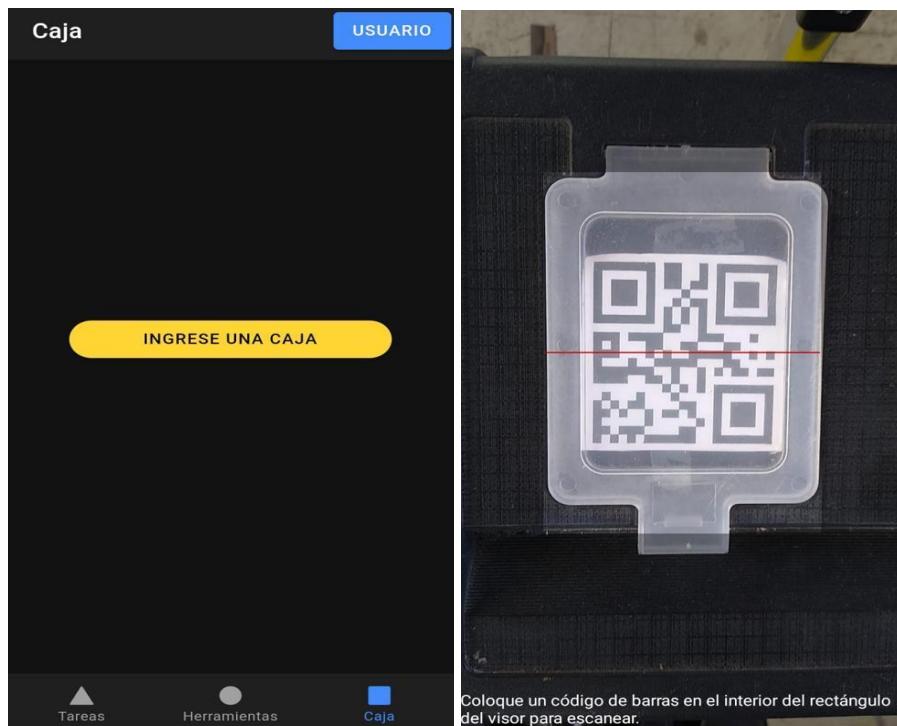
#### d. Interfaces visuales



El componente principal de esta pestaña accede a la variable que contiene el nombre de usuario, y mediante la API de Firebase para React (utilizando la función `onSnapshot`), obtiene las tareas para ese usuario y las muestra en pantalla. Éstas se pueden tildar para indicar que están completadas.

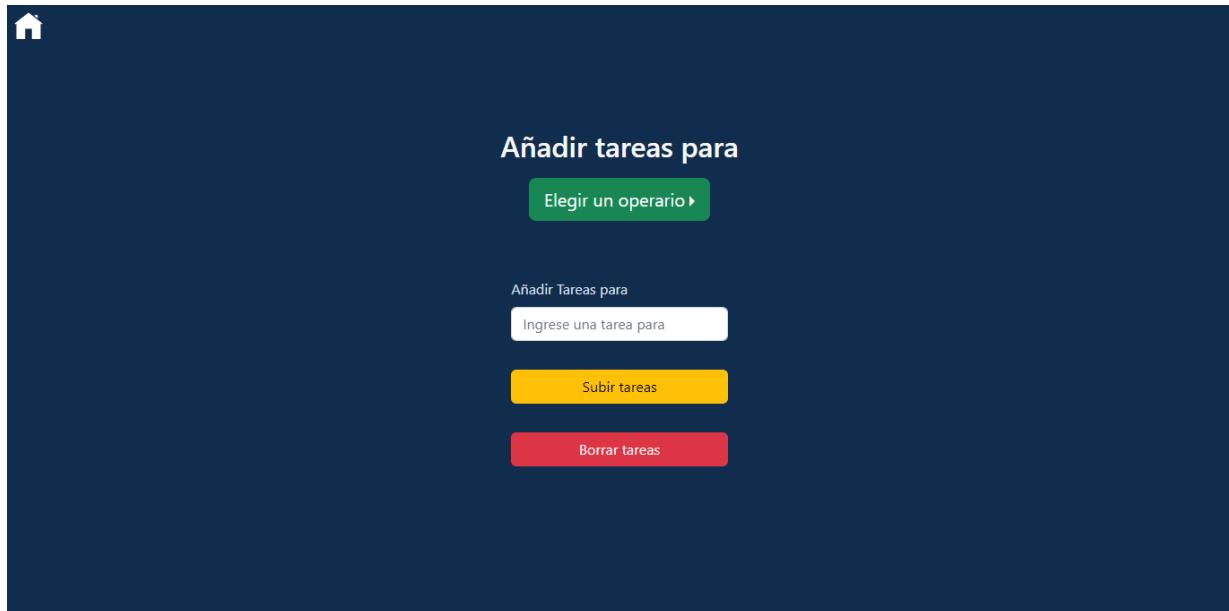


Nuevamente, mediante la API de Firebase para React, se accede a las herramientas faltantes vinculadas a la caja de herramientas escaneada utilizando el escáner QR en tiempo real.



El código QR es un número de 8 cifras que se encuentra en la parte exterior de la caja. El escáner es un plugin de Ionic que permite el uso sencillo de la

cámara para capturar el código. Después de ser escaneado, mediante el hook “useContext” de React, se permite acceder al número de caja en cualquier componente de la aplicación.



Se puede crear o seleccionar un usuario y asignarle tareas. Las tareas actuales se muestran en pantalla, y es posible agregar más o eliminarlas para cada usuario.

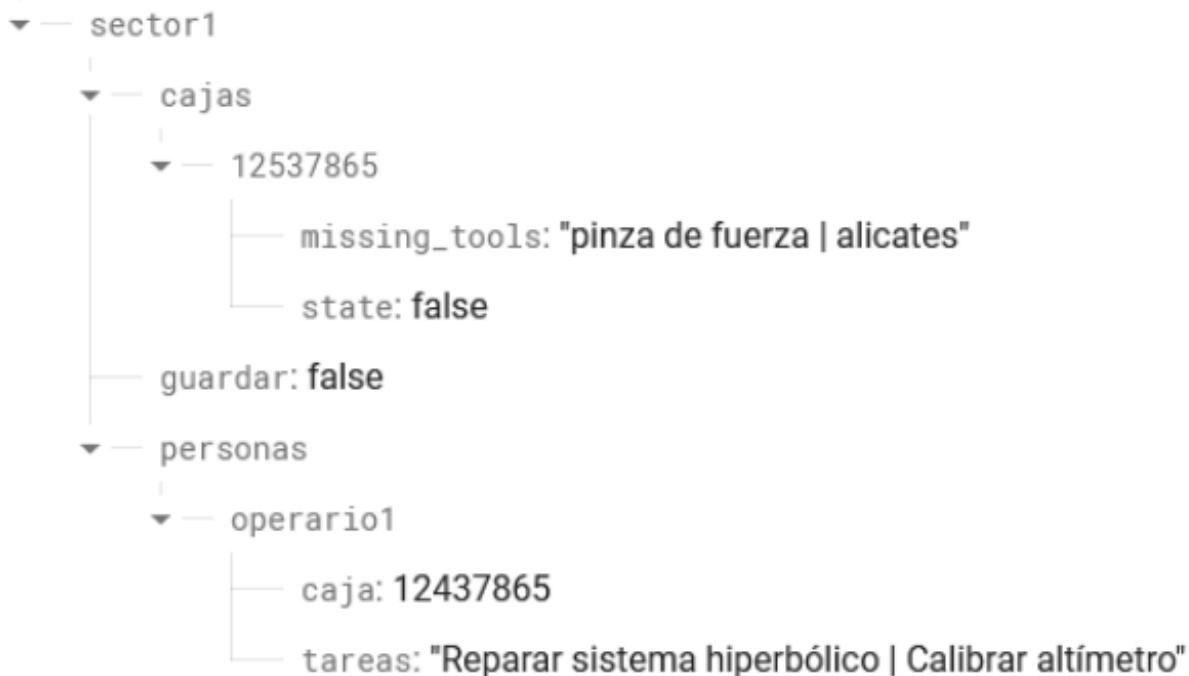
A screenshot of a mobile application interface showing a table of toolboxes. The table has columns for Caja (Box), Estado (Status), Operario (Operator), and Herramientas faltantes (Missing tools).

Caja	Estado	Operario	Herramientas faltantes
12537865	Encendida		Martillo, Cinta metrica
14537934	Encendida		
32327864	Apagada		
33434898	Apagada		
78718890	Apagada	jose	Phillips 1, Cinta metrica, Llave crique

Esta pestaña contiene una tabla que muestra cada caja de herramientas, sus operarios vinculados, su estado y las herramientas faltantes en ésta, todo

esto en una tabla dividida por columnas. Se hace un get a la base de datos por cada dato que se necesita. Además, se pueden crear nuevas cajas, ingresando el código QR de ésta.

### e. Estructura de datos



## 9. Sistema embebido

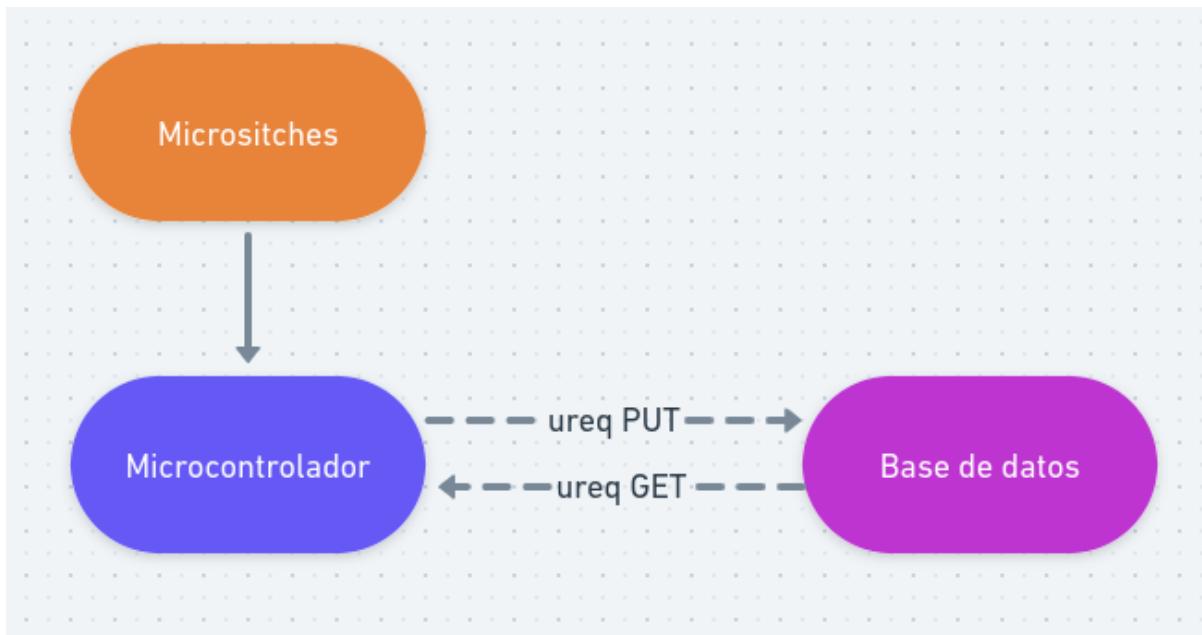
### a. Microcontrolador

En nuestro caso usamos el ESP-32 WROOM. Lo elegimos porque tiene tecnología WI-FI, además de que es fiable y económico. Este modelo consta de 36 pines. Mediante el módulo urequests, este puede recibir la señal de guardar cada 10 segundos aproximadamente, para reducir el consumo de energía.

### b. Software utilizados en el desarrollo

Hemos utilizado Thonny, es un entorno de desarrollo integrado para uPython diseñado para principiantes, que nos permite fácilmente subir código al ESP-32.

### c. Diagrama en bloques



### d. Lenguaje de programación

Utilizamos MicroPython, que es una implementación Python optimizado para poder ejecutarse en un microcontrolador. Esto nos fue de mucha utilidad, ya que permite aplicar nuestros conocimientos previos de python para programar el microcontrolador. Además posee módulos descargables muy útiles. Lo más importante son los métodos GET y PATCH:

```
def getReq(param):
    return ureq.get(f"{dbURL}/{area_trabajo}/{param}.json").json()
def patchReq(param,js):
    return ureq.get(f"{dbURL}/{area_trabajo}/{param}.json", json=js).json()
```

### e. Código

```

while True:

    while not getReq("guardar"):
        patchReq(f"cajas/{toolbox}", {"missing_tools": "", "state": False})
        time.sleep(10) # Waits until the store signal arrives

    for tool in tools:
        if tool.sel[3] == 1:
            s0.on()
        if tool.sel[2] == 1:
            s1.on()
        if tool.sel[1] == 1:
            s2.on()
        if tool.sel[0] == 1:
            s3.on()

    # ! This delay ensures an effective multiplexer switching

    if not sig.value():
        # * The tool is not in its place
        if not checkDuplicates(tool.nombre):
            missing_tools.append(tool.nombre)
    else:
        # * The tool is in its place
        if checkDuplicates(tool.nombre):
            missing_tools.remove(tool.nombre)

    s0.off()
    s1.off()
    s2.off()
    s3.off()

    if missing_tools != [] and checkContacts():
        alarm.on()
    else:
        alarm.off()

```

En este pedazo de código ocurre lo más importante del programa: se itera cada salida del multiplexor cambiando el selector y se registra el estado del común y se activa la alarma si alguna entrada es verdadera.

## f. Periféricos utilizados

### Multiplexor

Un multiplexor es un circuito combinacional con varias entradas y una única salida de datos. Están dotados de entradas de control capaces de seleccionar solo una de las entradas de datos para permitir su transmisión

desde la entrada seleccionada hacia dicha salida. Nosotros lo hemos utilizado para que seleccione cada una de las herramientas, que serían las entradas (16 por cada multiplexor); y la salida SIG, le envía la información del estado de las herramientas al microcontrolador.

#### **Buzzer**

Un buzzer es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente de un mismo tono. Sirve como mecanismo de señalización, como una alarma.

#### **Step up**

El regulador step-up MTt3608 es un dispositivo electrónico que permite la regulación o conversión de voltaje de corriente continua a otro voltaje de corriente continua más alto.

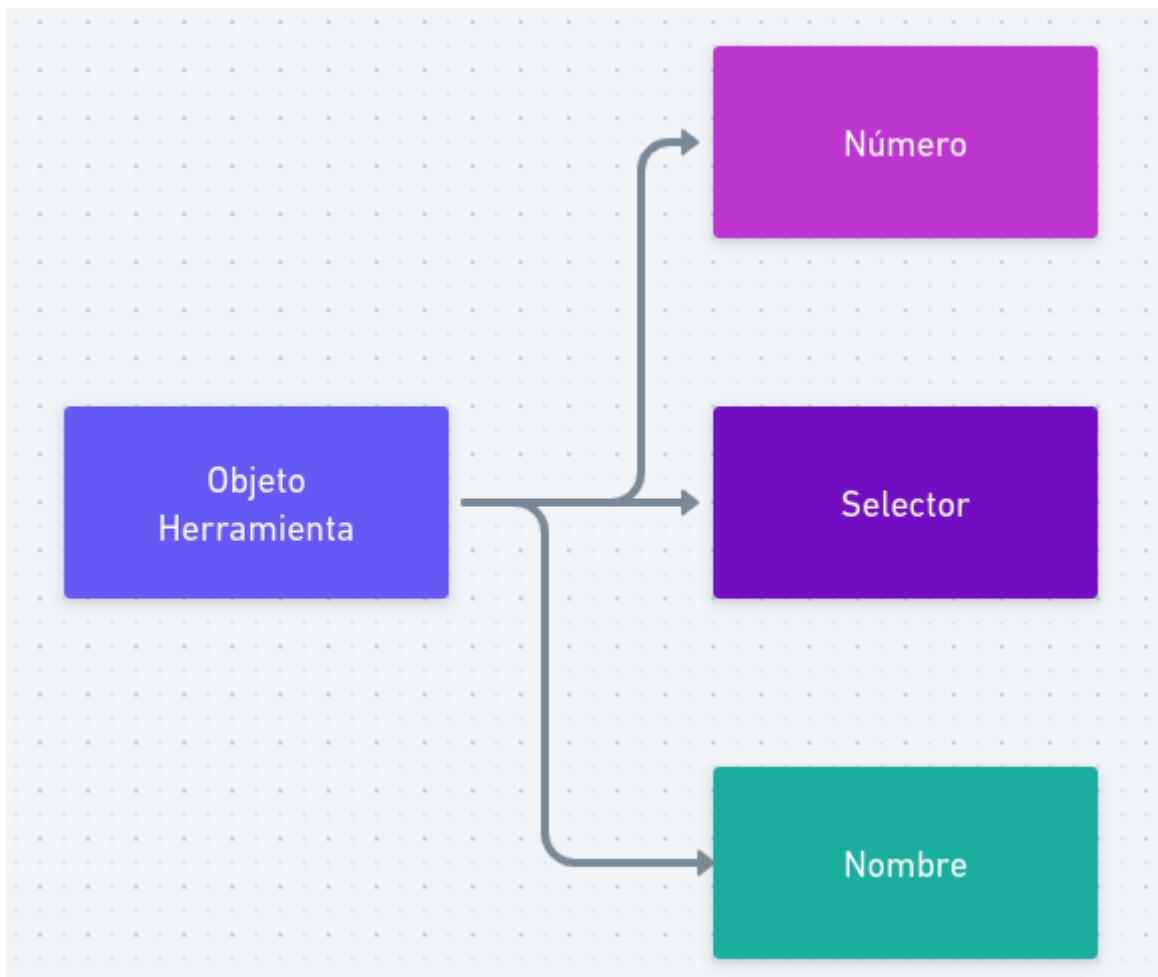
#### **Microswitch**

Un microswitch, es un dispositivo electrónico usado para detectar si la herramienta está o no. Si está abierto, dicha herramienta estaría faltante. Están ubicados en cada cuna de herramienta.

#### **Contacto magnético**

Son un simple interruptor que se abre y cierra mediante la aproximación de un imán.

### **g. Estructuras de datos**



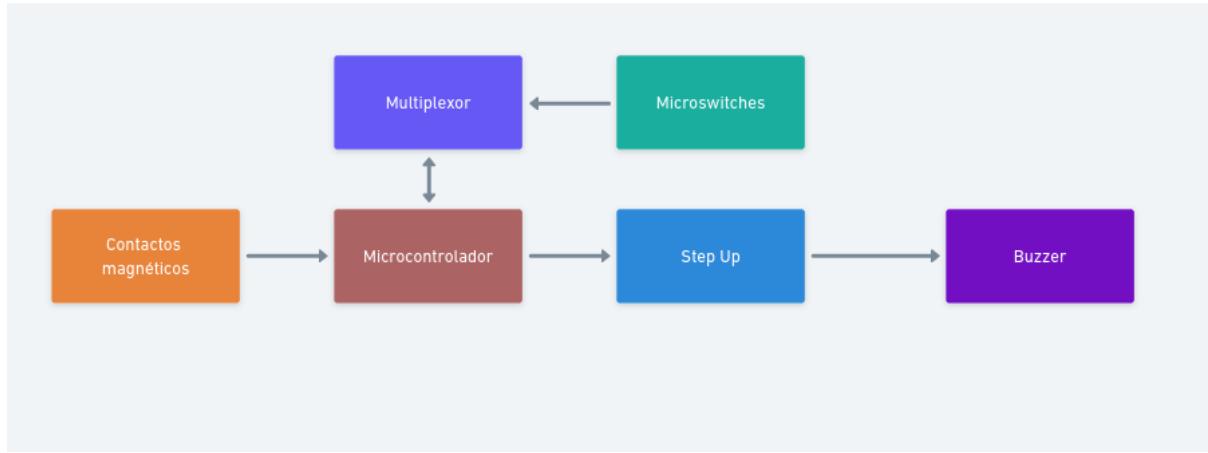
En el cuadro de arriba se muestra un objeto herramienta que tiene los atributos número (para ordenarlas), selector (para saber qué código binario va en el multiplexor) y nombre (es el nombre que se sube a la base de datos).

Para subirse a la base de datos, se concatenan todas las herramientas en una cadena separadas por un carácter especial.

**“pinza de fuerza | martillo”**

## 10. Electrónica

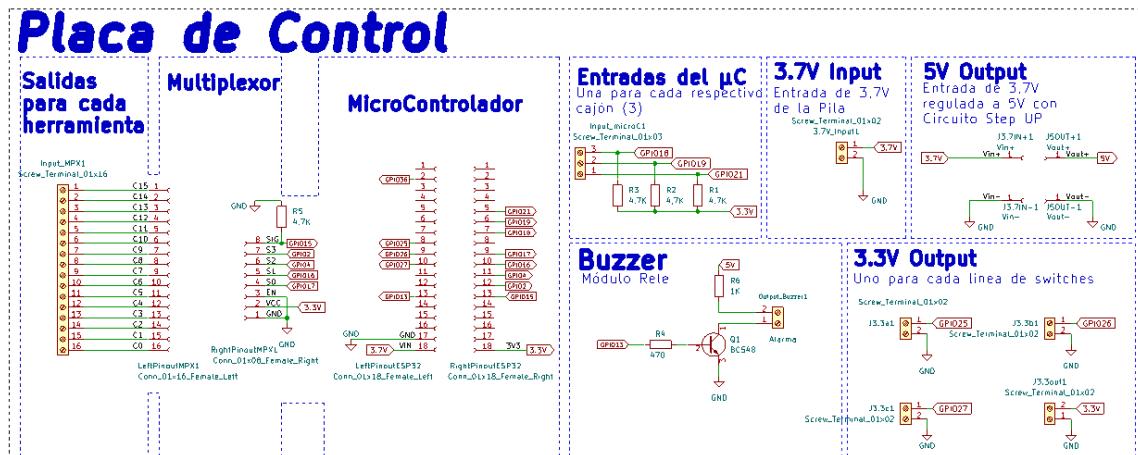
### a. Diagrama en bloques de las partes



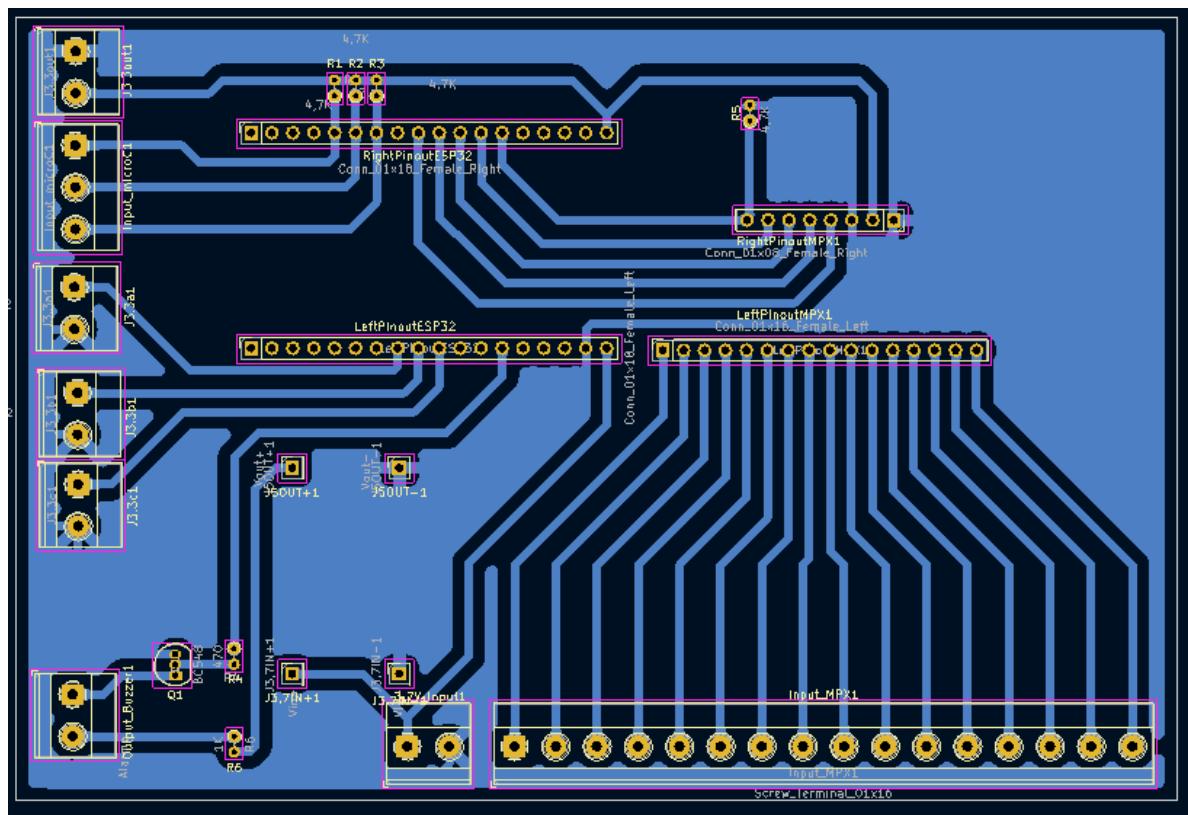
## b. Software utilizado para PCB y esquématicos

Utilizamos KiCad, que es un software para la automatización del diseño electrónico. Se usó para el diseño de esquemáticos para circuitos electrónicos y su conversión a placa de circuito impreso. Muchas pistas tienen una distancia mayor entre éstas para reducir la probabilidad de un cortocircuito con GND u otra pista. Además, la placa de control posee teardrops que reducen la probabilidad de que el cobre se despegue.

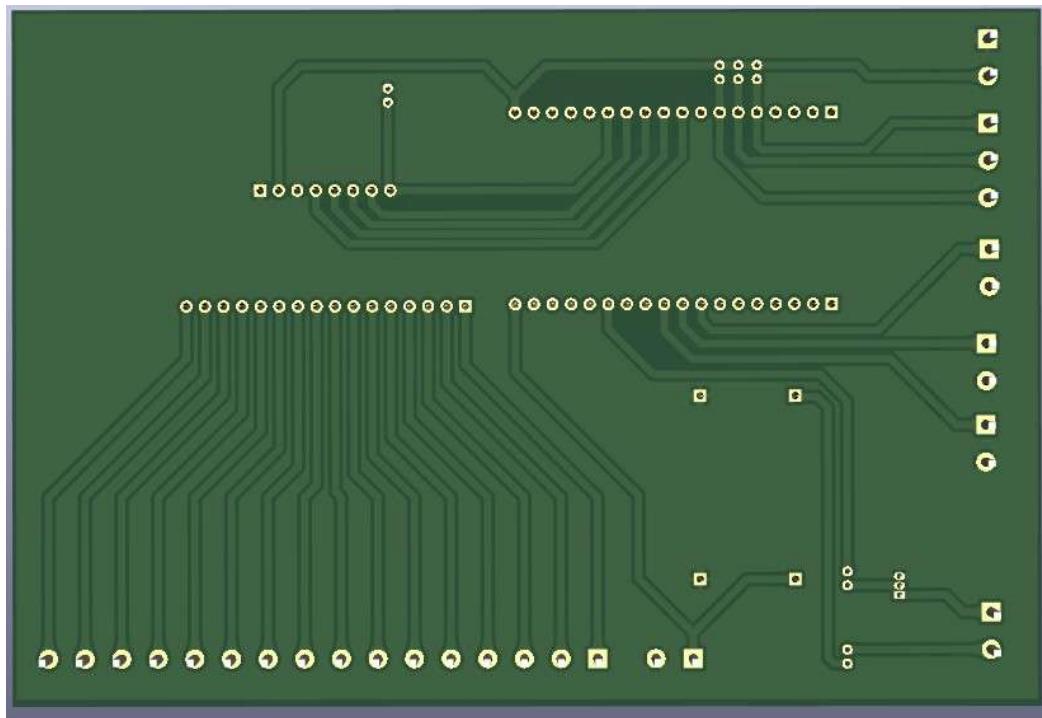
## c. Esquématicos

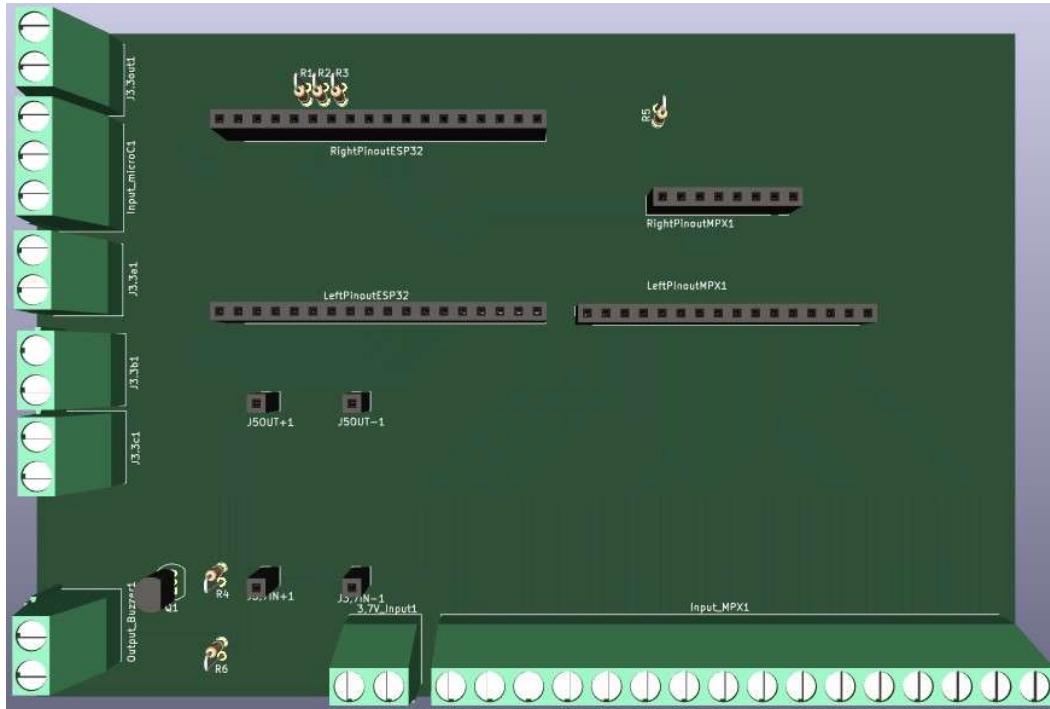


## d. PCB



e. Modelo 3D de cada PCB





## f. Fuente de alimentación

Utilizamos una batería 18650 de 3,7V que permite mantener encendido el sistema por más de 30 horas (con consumo máximo de corriente) y un cargador especial para esta batería.



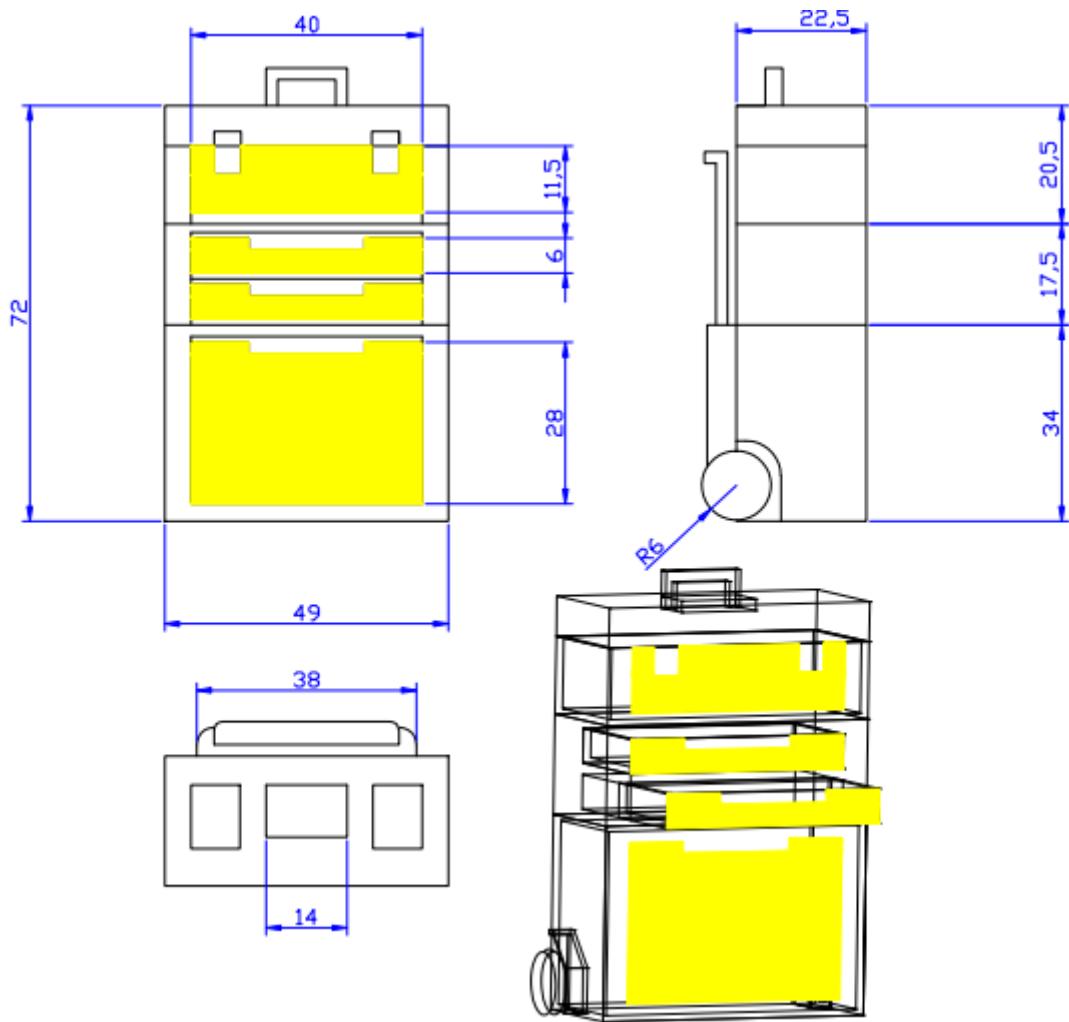
## g. Especificaciones técnicas de cada componente

- Multiplexor
  - Alimentacion: 2V a 6V
  - Resistencia en "On": 70 Ohms

- Buzzer
  - Rated Voltage DC 12V
  - Operating Voltage DC 3-24V
  - Rated Current 30mA
  - Decibel 90DB
  - Resonant Frequency 3000+/-500
  - Working Temperature -20 to +60
  - Overall Size 30 x 15mm/ 1.2" x 0.6"(D\*H)
  - Mounting Hole Dia 3mm/ 0.12"
  - Mounting Hole Distance 40mm/ 1.6" (separación entre agujeros)
  - Wire Length 10cm / 4"
  - External Material ABS
  - Weight 15g
- Step up
  - Tensión de entrada: 2 a 24V DC
  - Salida de tensión: 5 a 28V DC ajustable
  - Corriente Máxima de salida: 2A
- Microswitch
  - 5A 250V máx
  - Tamaño: 20 x 6 x 15mm

## 11. Estructura

### a. Diagrama general de la estructura



### b. Software de diseño utilizado

Para el diseño hemos utilizado AutoCAD, que es un software de diseño asistido por computadora utilizado para dibujo 2D y modelado 3D.

### c. Descripción de la estructura

La estructura que utilizamos es una caja marca CROWNMAN con 3 cajones que se abren horizontalmente y ruedas para transportarla. La idea de Smart Toolbox es que se pueda aplicar a cualquier caja similar a la que utilizamos.

### d. Imágenes de la estructura



## 12. Anexo

- <https://www.typescriptlang.org/docs/>
- <https://firebase.google.com/docs>
- <https://ionicframework.com/docs>
- <https://tailwindcss.com/>
- <https://getbootstrap.com/>
- <https://nextjs.org/docs>
- <https://www.electron.build/>
- <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html>
- <https://docs.micropython.org/en/latest/>
- <https://docs.kicad.org/>
- <https://reactjs.org/docs/getting-started.html>
- <https://maker.pro/pcb/tutorial/how-to-make-a-printed-circuit-board-pcb>
- <https://vercel.com/>