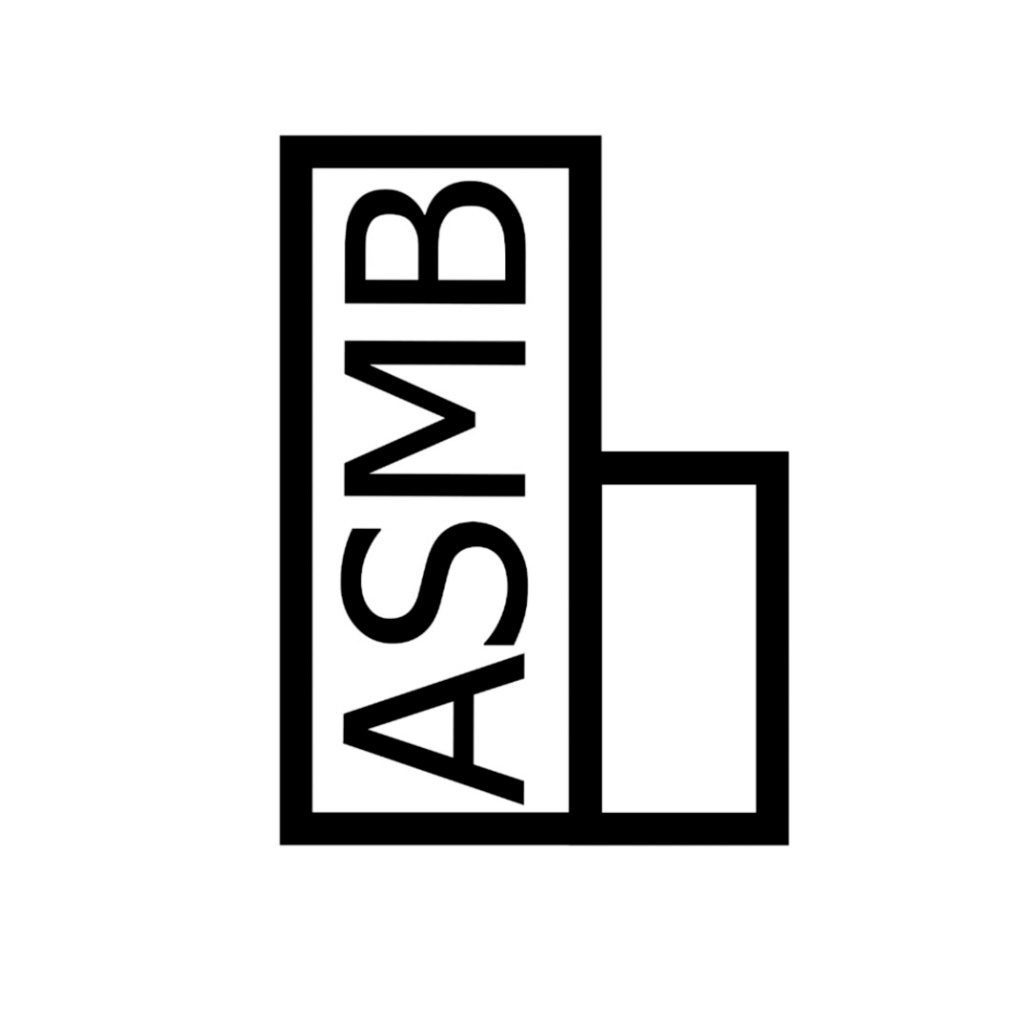
**Automatic Sanitary Modular Booth**



**Integrantes:**

**Casareski, Juan I.**

**Cuccaro, Marco F.**

**Murgia, Agustín N.**

**Torres, Iván B.**

**Índice:**

1. **Introducción**
   1. Objetivo del proyecto
   2. Descripción general
   3. Uso deseado
   4. Diagrama en bloques
   5. Diagrama de Gantt
2. **Componentes**
   1. Raspberry pi 4b
   2. Sensor de temperatura infrarrojo MLX90614
   3. Lector de tarjetas RFID RC522
   4. Sensor de proximidad Fc-51
   5. Sensores de reflexión E18 D80nk
   6. Cámara USB
   7. Shift Register 74HC595
   8. Bomba de agua Bt30
   9. Fuente genérica de computadora 500W
3. **Software**
   1. Clase general
   2. Sensor de temperatura
   3. Detección facial
      1. Detector de puntos faciales
      2. Detector de barbijos
   4. Lector de tarjetas RFID RC522
   5. Registro de empleados
   6. Registro de clientes
   7. Cartel de funcionamiento
   8. Dispensador de alcohol en gel
   9. Ingreso y egreso
   10. Panel de estado
   11. Funciones de día y hora
4. **Esquemáticos y PBC's**
   1. Motherboard
   2. Panel de estado
   3. Level shifters (Sensores de proximidad)
   4. Adaptador RC522
5. **Modelos 3D**
   1. Cabina completa
   2. Soporte de la cámara
   3. Carcasa del lector de tarjetas
   4. Carcasa del sensor de temperatura
   5. Dispensador de alcohol en gel
6. **Recursos**
7. **Introducción**
   1. **Objetivo del proyecto**

Este proyecto fue diseñado para la desinfección integral de un peatón/empleado. También cuenta con sensando de temperatura, reconocimiento facial, registro de asistencia y un indicador de funcionamiento para cada sensor. Este cuenta con un enfoque a la modularidad del sistema.

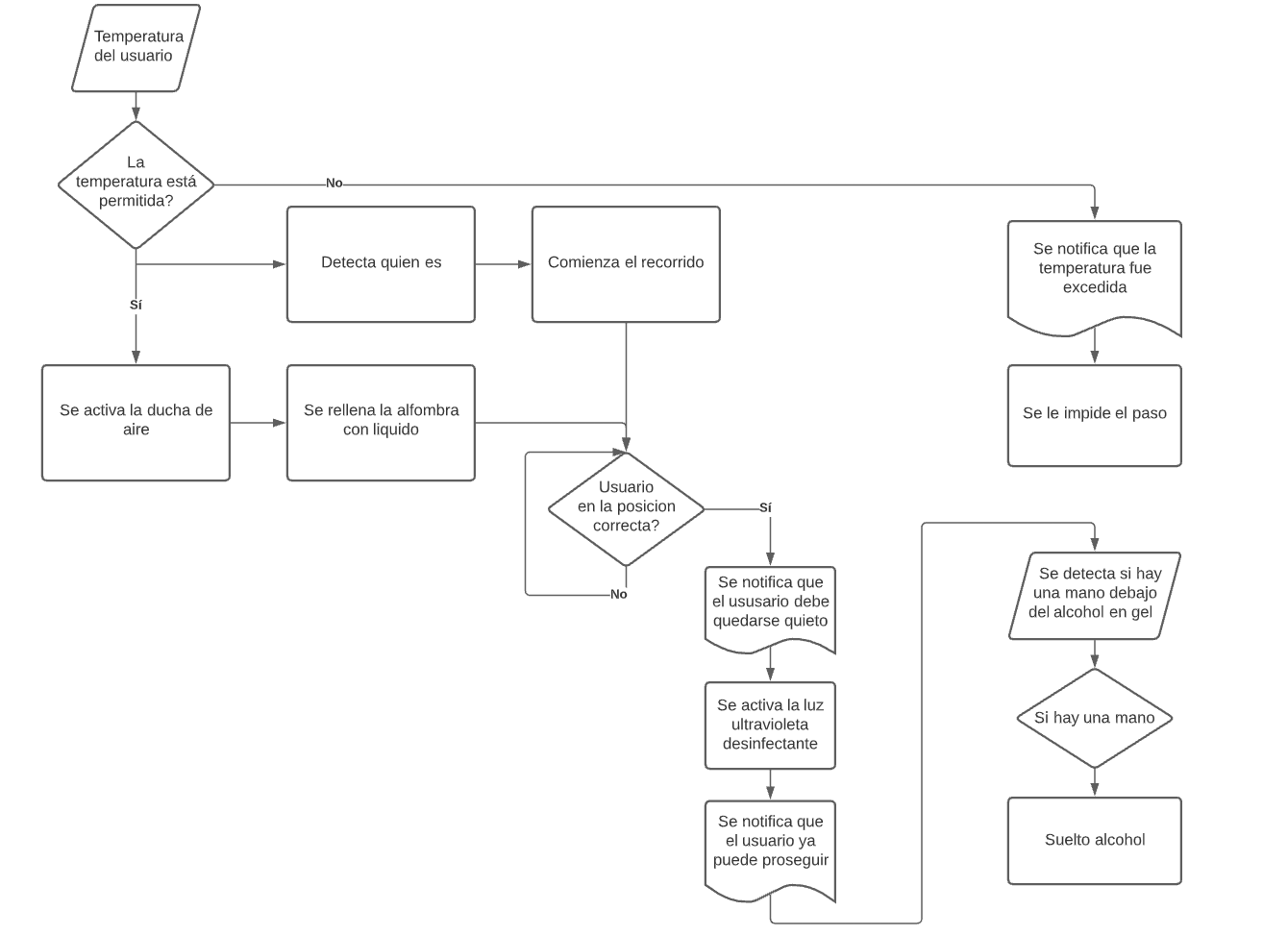
* 1. **Descripción general**

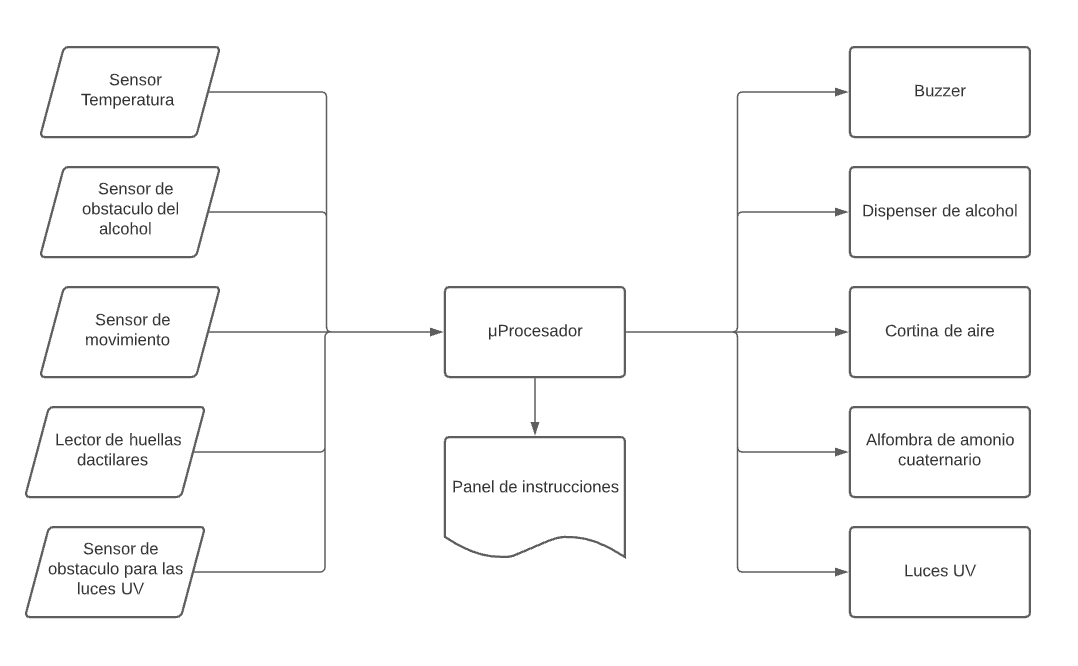
El proyecto ASMB fue construido desde la base por alumnos de la EEST N7 TRQ, cuya estructura hecha integralmente de materiales provisto en la escuela.

* 1. **Uso deseado**

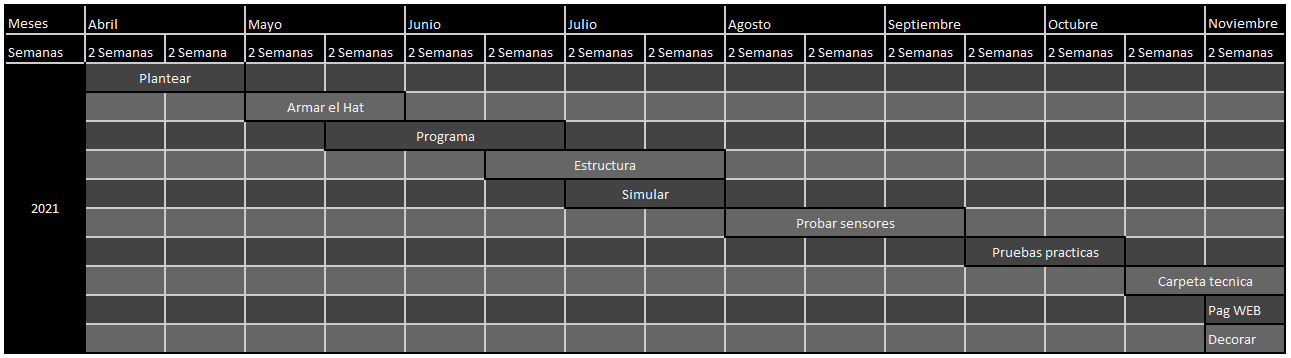
La cabina está diseñada para ser montada en un espacio abierto antes del acceso a un establecimiento, donde el usuario se tomará la temperatura antes de entrar a la cabina, en caso de que sea un empleado deberá también poner su tarjeta de identificación. Una vez dentro de la cabina se le detectará la cara al usuario para determinar si tiene los ojos y boca cerrados (u opcionalmente el uso de un barbijo), una vez confirmado esto se desinfectará al usuario y este mismo podrá salir de las misma, sin antes desinfectarse las manos con el dispensador de alcohol en gel. Este mismo ingreso será guardado en una base de datos local junto a el horario de entrada, temperatura y el supuesto horario de entrada.

* 1. **Diagrama en bloques**





* 1. **Diagrama de Gantt**



1. **Componentes**
   1. **Raspberry pi 4B**

Este microprocesador fue seleccionado ya que requeríamos de una potencia considerable para poder ejecutar los scripts de detección facial. Todas las conexiones que requieren de alguna lógica se conectan a este mediante la motherboard.

* 1. **Sensor de temperatura infrarrojo MLX90614**

El MLX90614 es un sensor de temperatura sin contacto. Este sensor tiene un error de aproximadamente 0.14 C y un rango de medición de –20 C a 120 C. Este comunica la temperatura leída al microprocesador mediante un protocolo serial I²C.

* 1. **Lector de tarjetas RFID RC522**

Este módulo lee las tarjetas RFID las cuales incluyen dos valores: id y texto.

* 1. **Sensor de proximidad Fc-51**

Este sensor detecta objetos en su rango de visión de 15 grados, a un máximo de 30cm. Es usado en el dispensador de alcohol sin contacto.

* 1. **Sensor de reflexión E18 D80nk**

Su funcionamiento es muy similar a el sensor de proximidad (De hecho, usan el mismo concepto para cumplir su función) con la diferencia de que son más sensibles y, por lo tanto, su rango de medición máximo aumenta de 30cm a 80cm.

* 1. **Cámara USB**

La cámara seleccionada fue la sinovisión sn-u4 principalmente a que fue probada con el microprocesador al cual se le pudo instalar los drivers de la misma. A esto se le suma que cuenta con los requerimientos necesarios para nuestra aplicación.

* 1. **Shift register 74HC595**

El 74hc595 es un shift register de 8 bits, esta cuadra con nuestro uso ya que no requerimos de una alta corriente en la salida.

* 1. **Bomba de agua Bt30**

La bomba tiene una potencia de 30W, funciona en 220W@50Hz. La mima tiene un caudal de 48L/h y una presión de 3bar, la presión de la cual requeríamos es de 2.5bar, pero tenemos un excedente principalmente debido a que el sistema puede tener perdidas.

* 1. **Fuente genérica de computadora 500W**

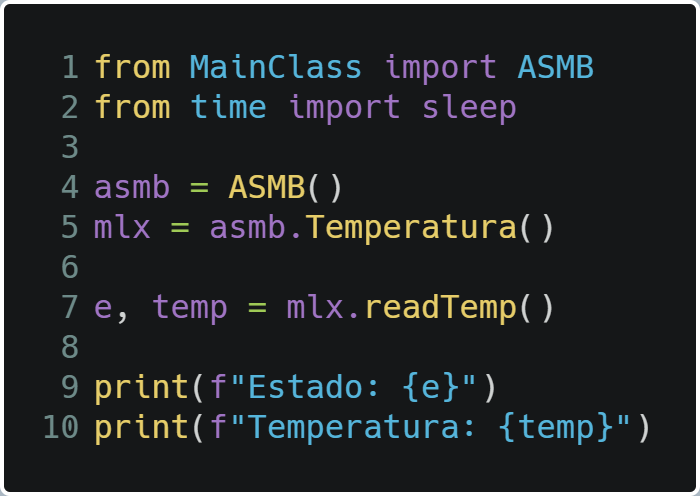
La fuente seleccionada fue una de computadora para mantener la simplicidad del sistema. Esta fuente se conecta a la motherboard (Ver sección 4.1.) mediante un conector molex de 24 pines.

1. **Software**
   1. **Clase general**

Esta fue ideada para la fácil reimplementación de los pines del sistema una situación en la que esto sea de utilidad seria, por ejemplo, El reemplazo de una RP3 por otro microprocesador por una RP4 (Situación la cual se dio en el mismo proyecto). Todos los archivos están disponibles en el repositorio (Ver sección 6).

* 1. **Sensor de temperatura**

Este sería el uso deseado de la clase <Temperatura>



El método <readTemp> devuelve dos valores, el primero hace referencia a si la persona está habilitada a entrar, siendo 'True' si se encuentra en el rango de temperatura permitido y 'False' si no se encuentra en dicho rango, el segundo valor es la temperatura leída.

* 1. **Detección Facial**

Esta parte cuenta con dos variaciones, la primera donde se detecta que los ojos y la boca estén cerrados y el segundo donde se detecta el uso de un barbijo.

* + 1. **Detector de puntos faciales**

Este se encarga de detectar el hecho de que el usuario tenga los ojos y boca cerrados por al menos 3 segundos para comenzar a desinfectar.



Cabe aclarar que al constructor <FaceDetection> se le pasa un argumento para seleccionar el modo de detección facial (En este caso 'eyes&Mouth')

* + 1. **Detector de barbijos**

Este detecta de detectar el uso de un barbijo para comenzar el proceso de desinfección



La única diferencia con el ejemplo anterior es la key distinta del constructor <FaceDetection>.

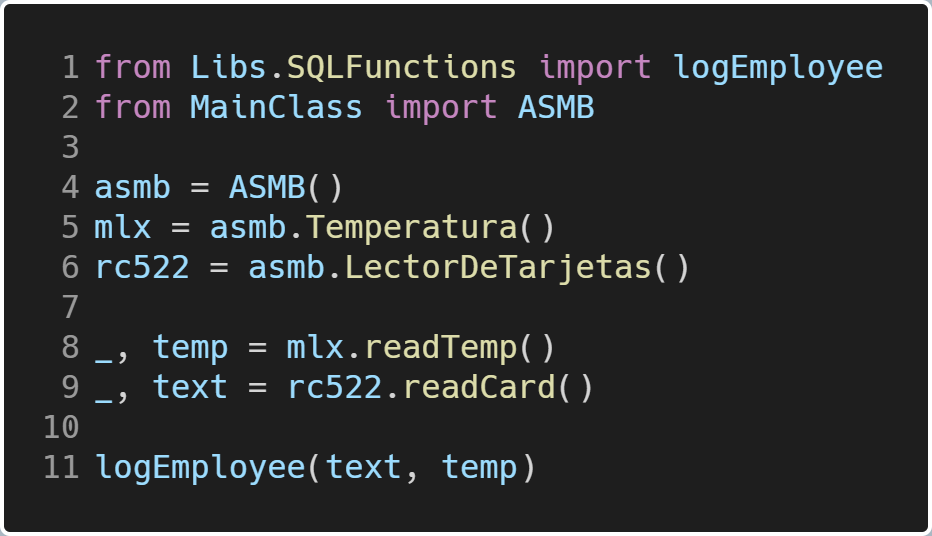
* 1. **Lector de tarjetas RFID RC522**

Este lector lee los dos datos dentro de la tarjeta, primero la id y segundo el text. El ejemplo siguiente lee estos dos anteriormente nombrados.



* 1. **Registro de empleados**

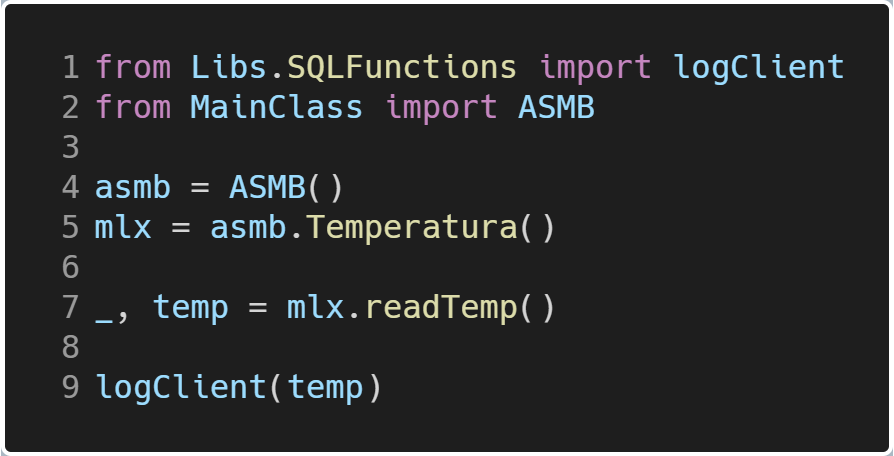
El registro de empleados se implementó mediante una función la cual pide con parámetros la id de este trabajador y su temperatura, esta misma función se encarga de conseguir la hora y día actual y de guardarlo todos los datos en una base de datos local.



En este ejemplo primero se le toma la temperatura al empleado, después su tarjeta de identificación y se guarda en la base de datos. Cabe aclarar que tanto el id de la tarjeta como el estado del sensor de temperatura no se utilizan en este ejemplo por lo que se les reemplazó por una '\_'.

* 1. **Registro de clientes**

El registro de empleados es una versión simplificada del registro de empleados, ya que un peatón normal no cuenta con una tarjeta de identificación por lo que no se ve sometido a un paso extra.



* 1. **Cartel de funcionamiento**

El fin de este cartel es de indicar cuando la cabina está fuera de funcionamiento o cuando está ocupada.



Este ejemplo alterna el estado del panel de funcionamiento cada .5 segundos. Cabe aclarar que en la línea 5 el constructor <Pin> requiere 3 argumentos: el primero es el número de pin, el segundo es su modo de funcionamiento y el tercero es el estado inicial de dicho pin.

* 1. **Dispensador de alcohol en gel**

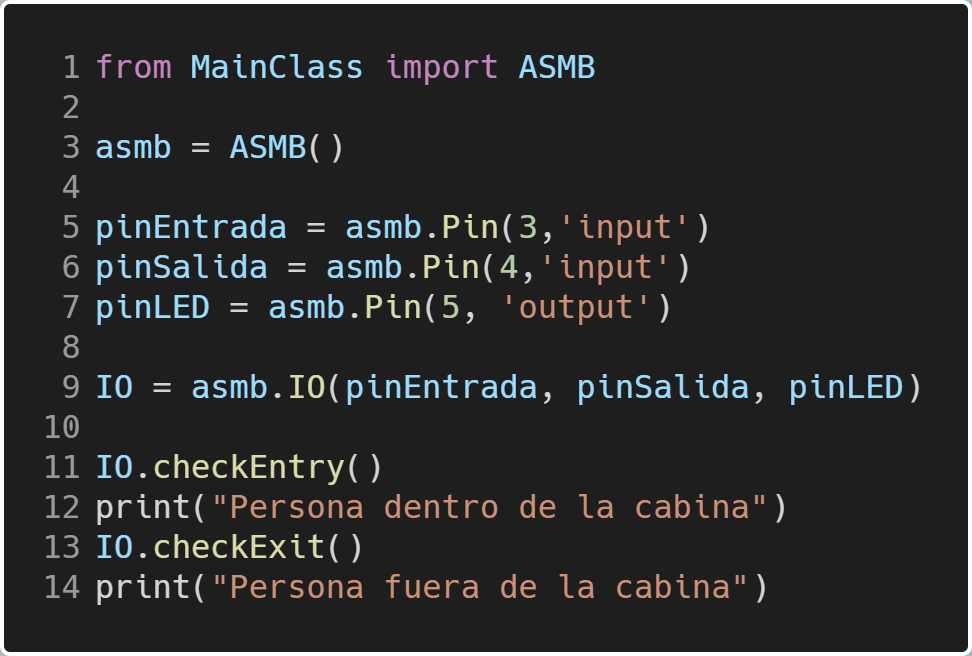
El dispensador de alcohol tiene la particularidad que tiene que funcionar de forma independiente al resto del sistema. Por esto utilizamos el módulo <threading>. Ignorando ese hecho el sistema lee un pin donde se conecta el sensor de proximidad para dispensar el alcohol.



Como se puede observar en el ejemplo primero se crea los objetos para los pines del sensor de proximidad y del motor, los cuales se pasan como argumento a el constructor <DispenserAlcohol>, constructor el cual toma 3 argumentos, siendo el tercero el tiempo total de activación del motor. Finalmente se crea un hilo el cual ejecuta el método principal del dispensador de alcohol.

* 1. **Ingreso y egreso**

La sección de ingreso y egreso se refiere a los sensores de proximidad que se encargan de lo anterior nombrado. La lógica detrás de este proceso es esperar hasta que algo corte el haz del sensor y el microprocesador lo lea.



En este ejemplo primero se definen los pines de los sensores de proximidad y de un led de debug (Que indica que una persona está en la cabina) y estos pines se dan como argumentos al constructor <IO>. Finalmente se verifica hasta que alguien pase por enfrente del sensor de proximidad de la entrada y posteriormente con el de la salida.

* 1. **Panel de estado**

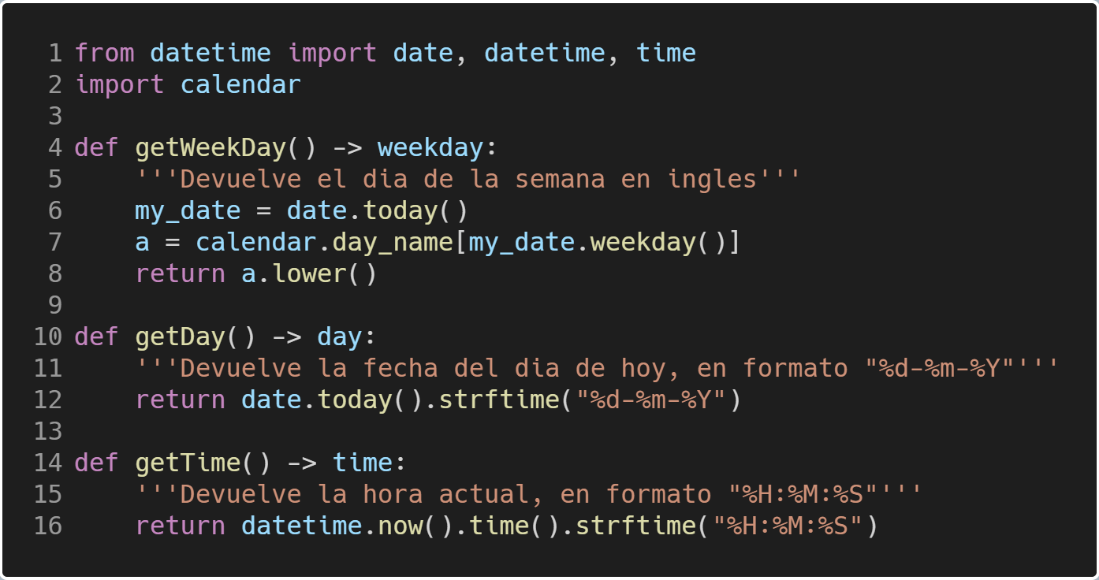
El estado de los sensores se manda mediante una comunicación serial sincrónica hacia un shift register (Ver sección 4.2.). El microprocesador comunica 1 byte de datos, en el cual cada bit indica si x sensor tiene una falla.



En el ejemplo anterior primero se crea el objeto del cartel de funcionamiento (Ver sección 3.7.), posteriormente se definen los pines del clock de los registros del shift register, el pin de datos y el pin de clock de la salida del shift register (Sh\_cp, ds y St\_cp, respectivamente). Después se definen los pines de entrada de los sensores de efecto hall. Finalmente crea el objeto del panel de estado con base el constructor <PanelEstado> y se lee el estado de todos los sensores, para posteriormente mostrarlos en el panel de estado.

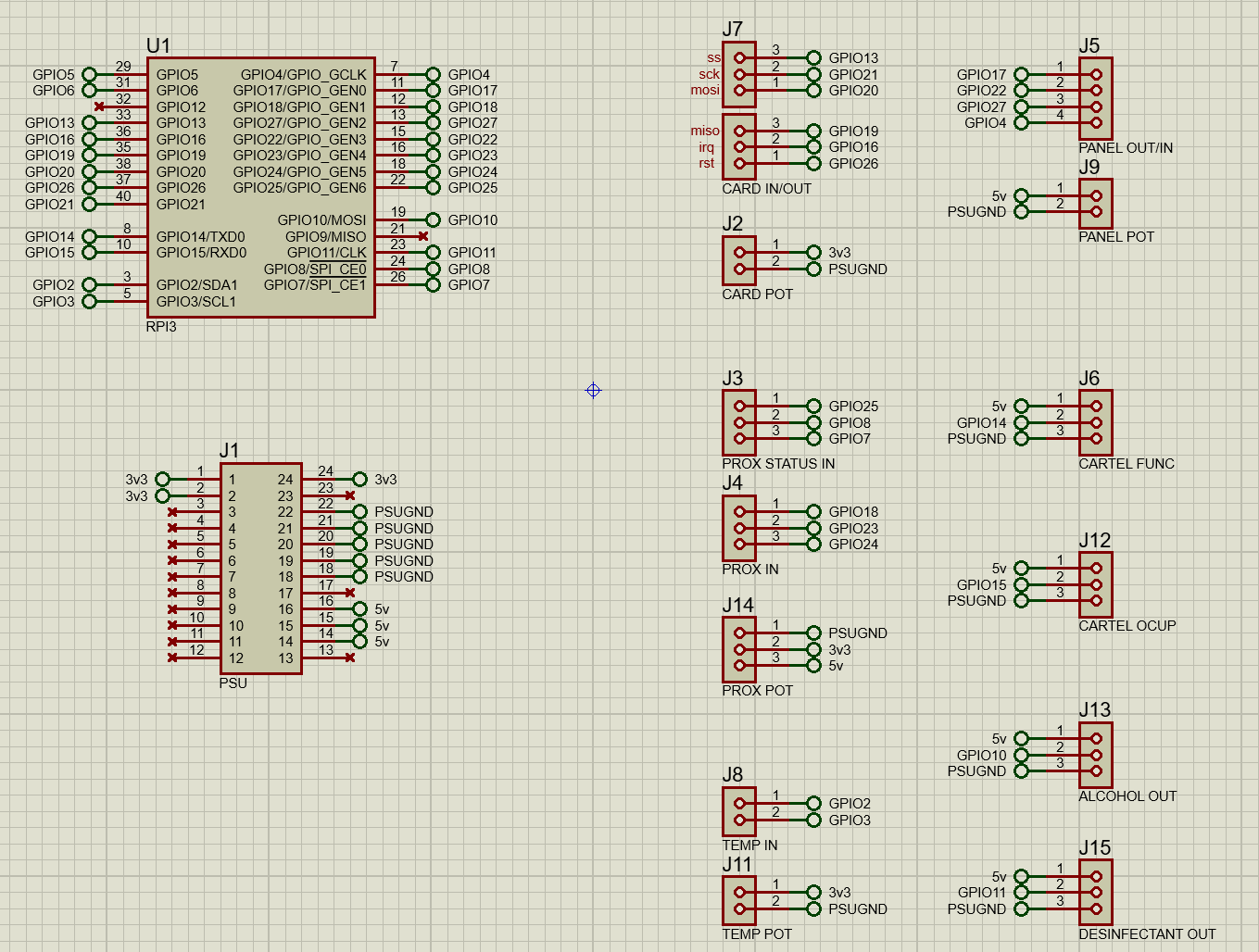
* 1. **Funciones de día y hora**

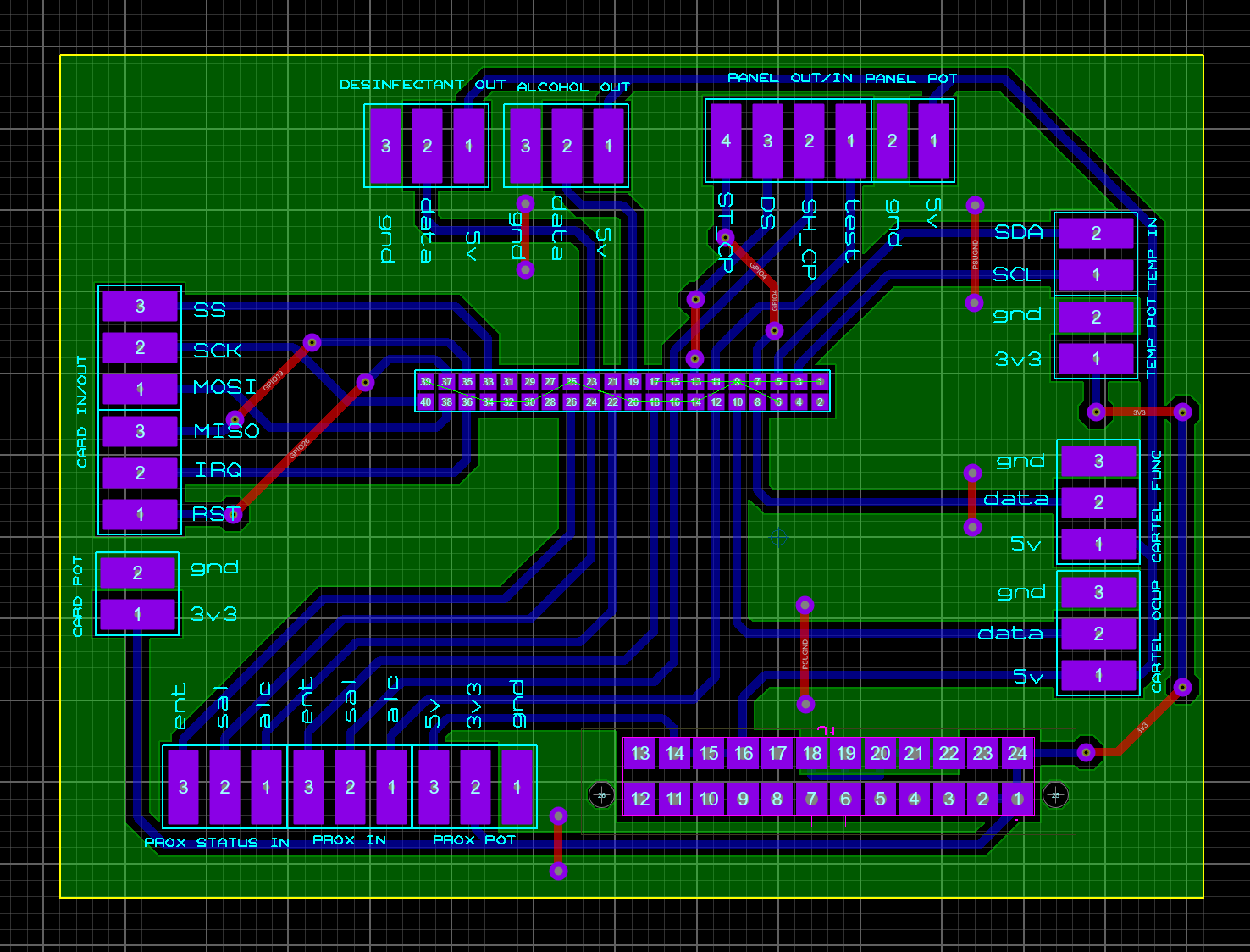
Esta sección hace referencia a un set de funciones escritas para estandarizar la obtención de hora/día/día de la semana dentro del proyecto. Estas utilizan el módulo <datetime> para su funcionalidad.



1. **Esquemáticos y PBC's** 
   1. **Motherboard**

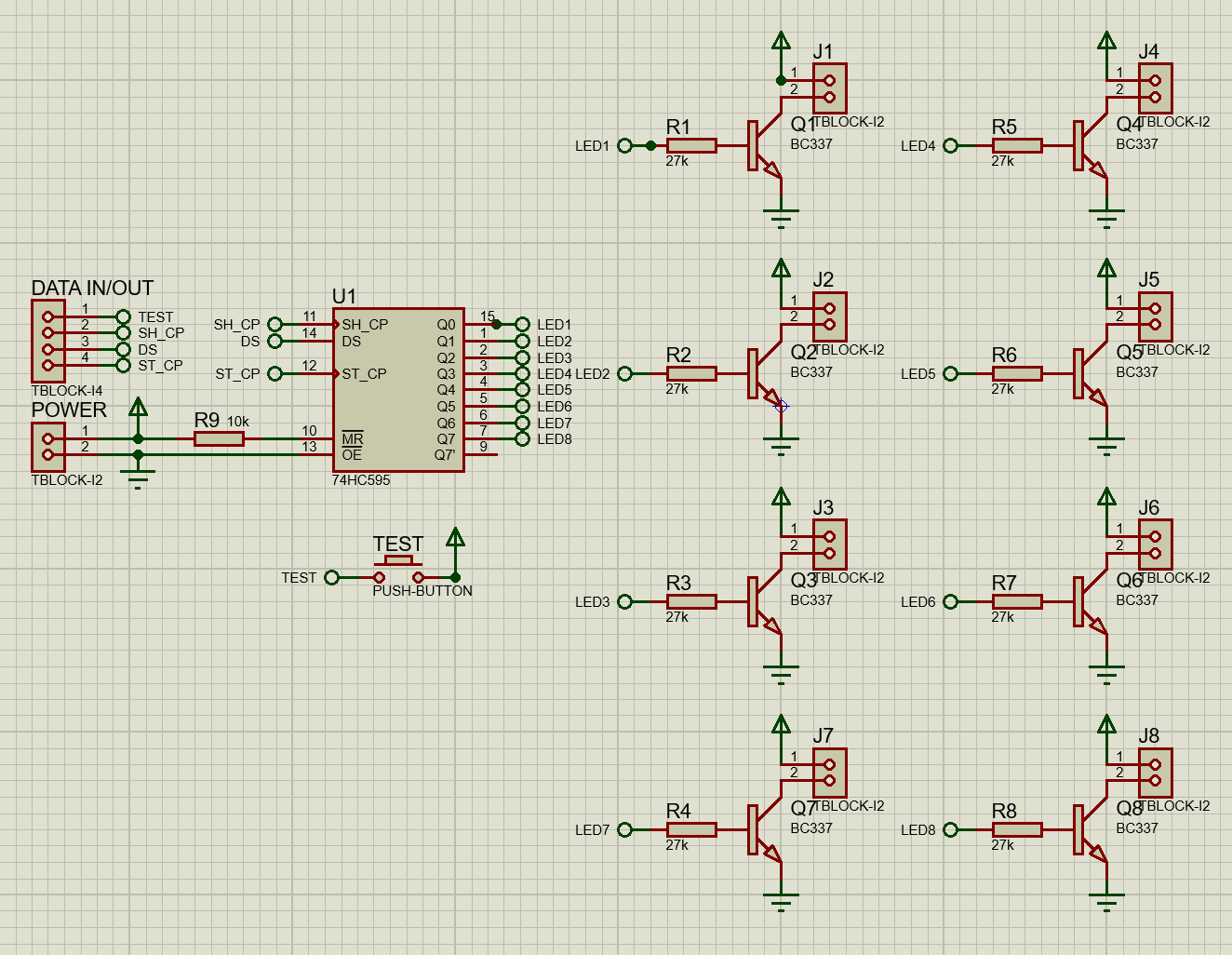
El fin este circuito es de conectar todos los dispositivos periféricos al microprocesador.

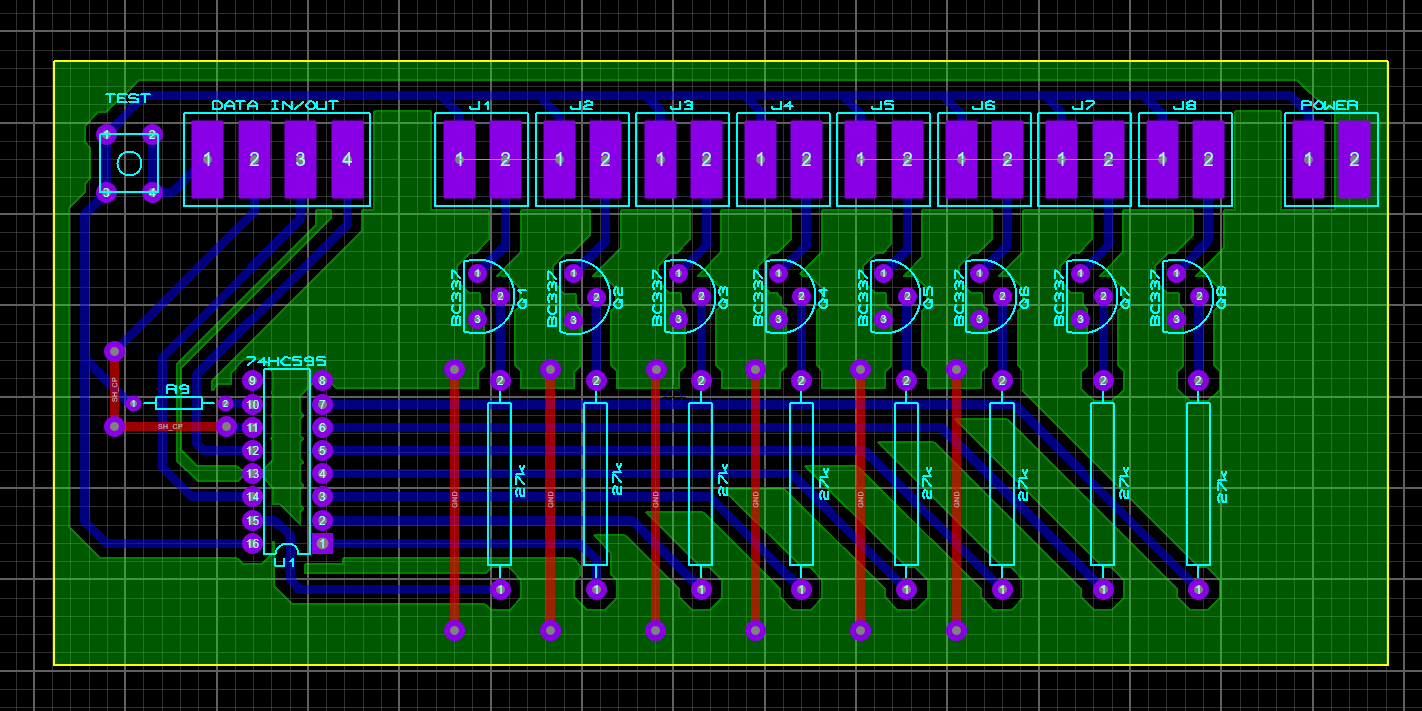




* 1. **Panel de estado**

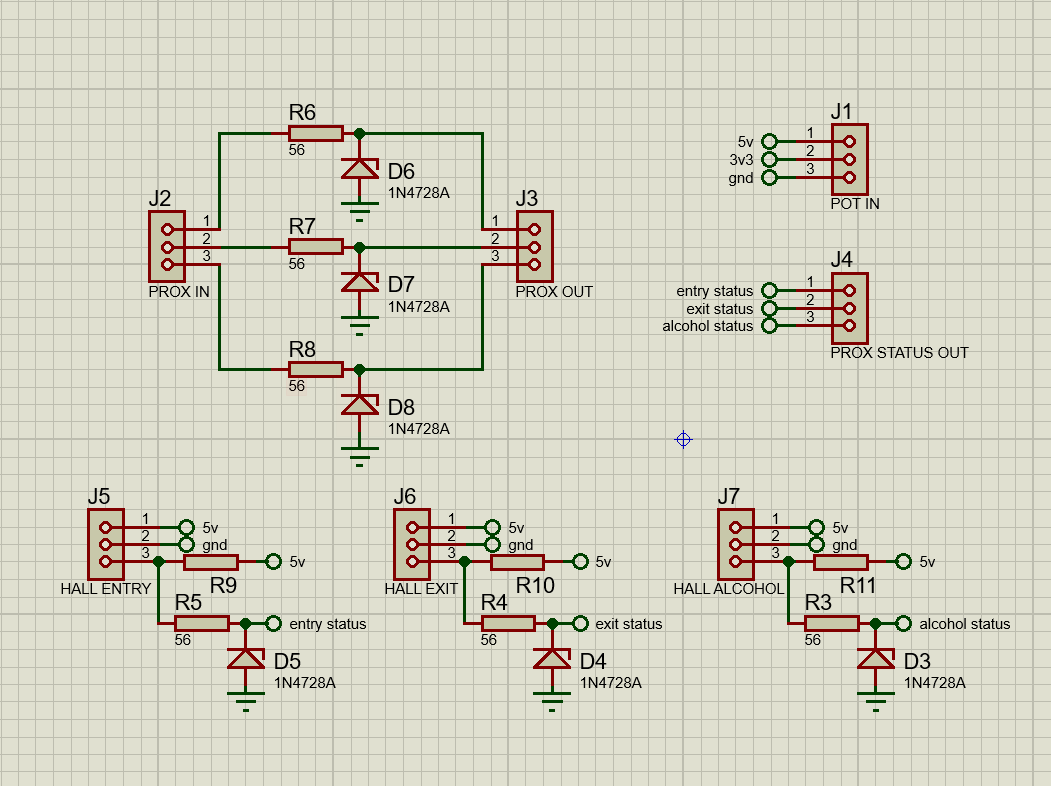
Este panel recibe el estado de todos los sensores desde el microprocesador y lo indica en 8 LED's. Esta señal se recibe como un paquete de 1 byte, el cual indica este en 8 LED's distintos mediante un 74HC595, a la salida de cada registro del anterior nombrad hay un BC337 ya que la corriente máxima soportada por el shift register es facilmente excedida por un par de LED's prendidos.

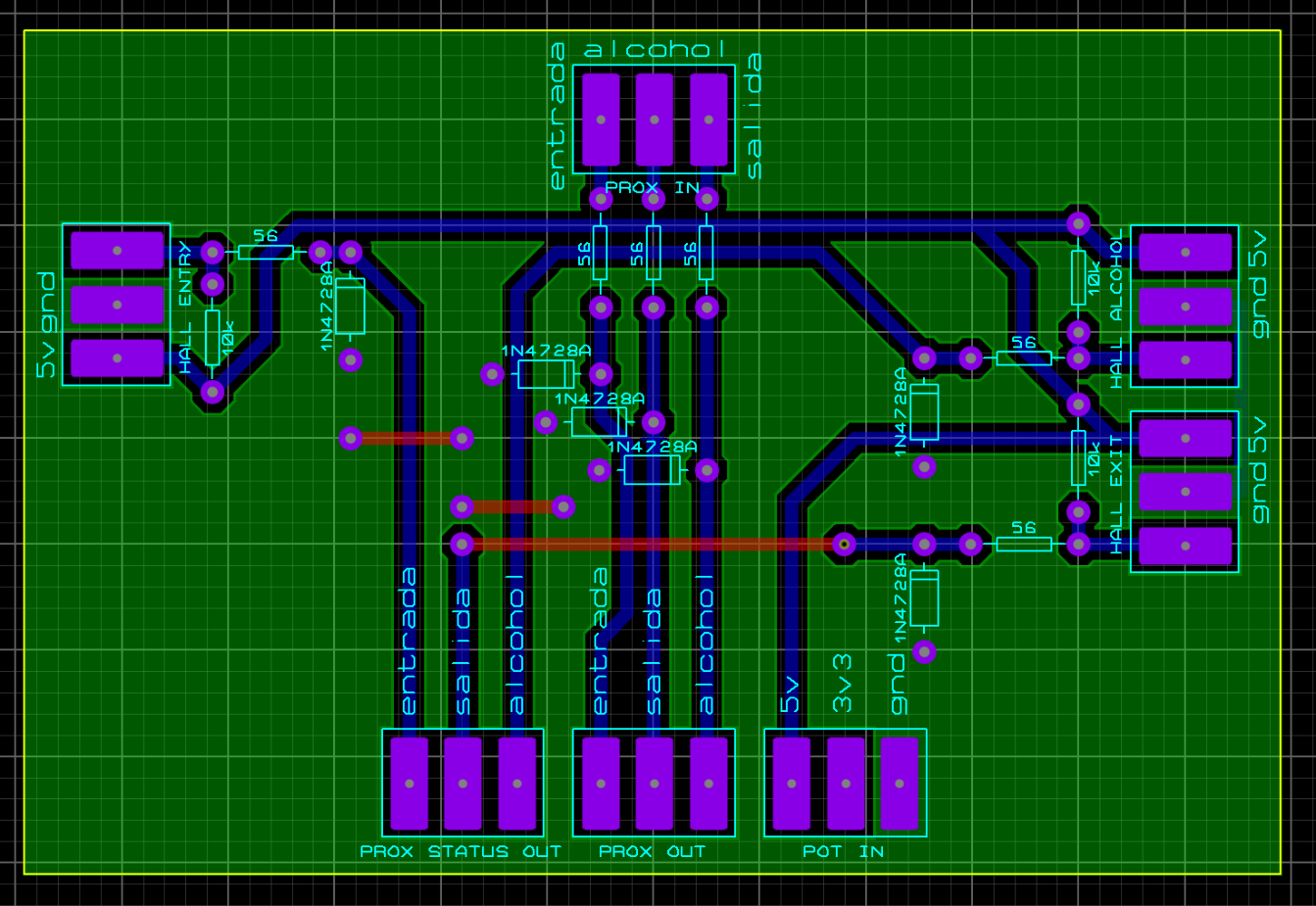




* 1. **Level shifters**

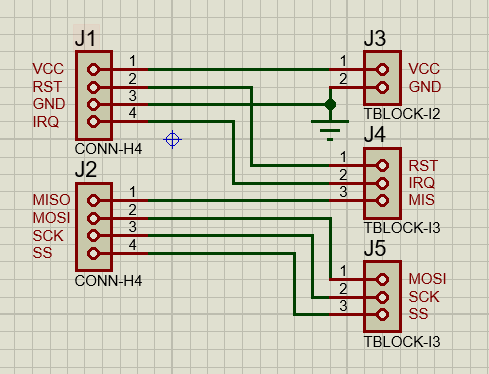
Este circuito es el encargado de adaptar los niveles lógicos de los sensores de efecto hall y de los sensores de proximidad y reflexión. Los anteriores nombrados usan 5v para representar un 1 lógico, este circuito es el encargado de convertir estos 5v a 3v3, mediante un diodo Zener.

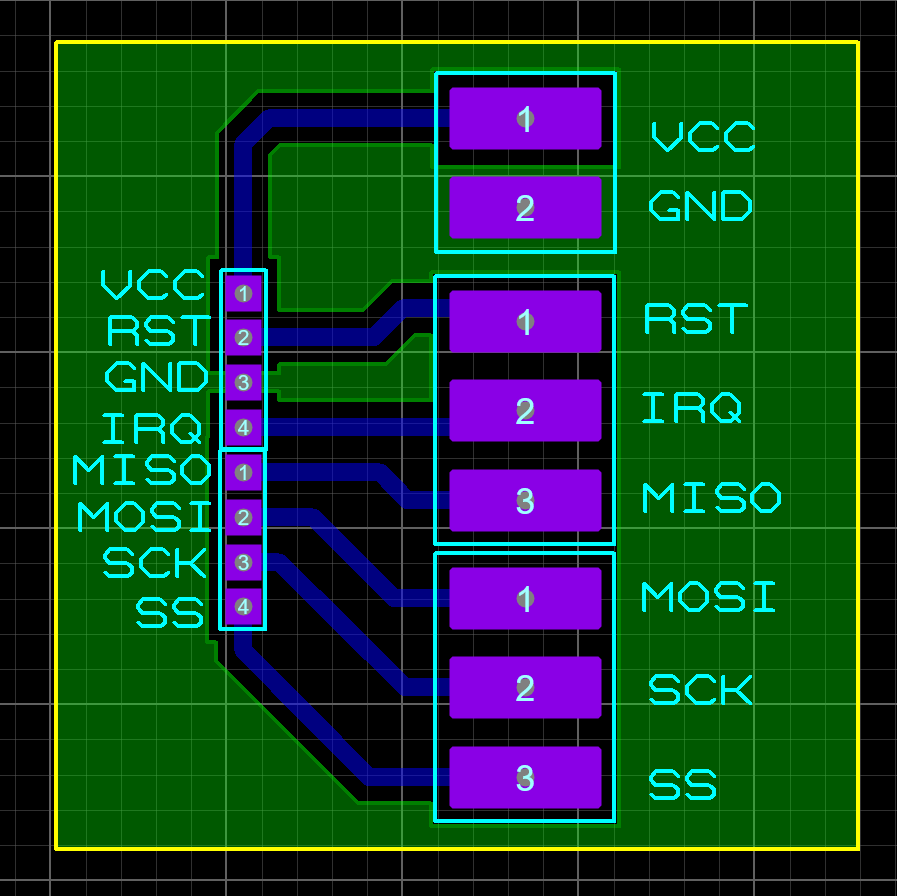




* 1. **Adaptador RC522**

Este circuito tiene la única y sola función de adaptar la tira de pines del RC522 a unas borneras para su fácil conexión.

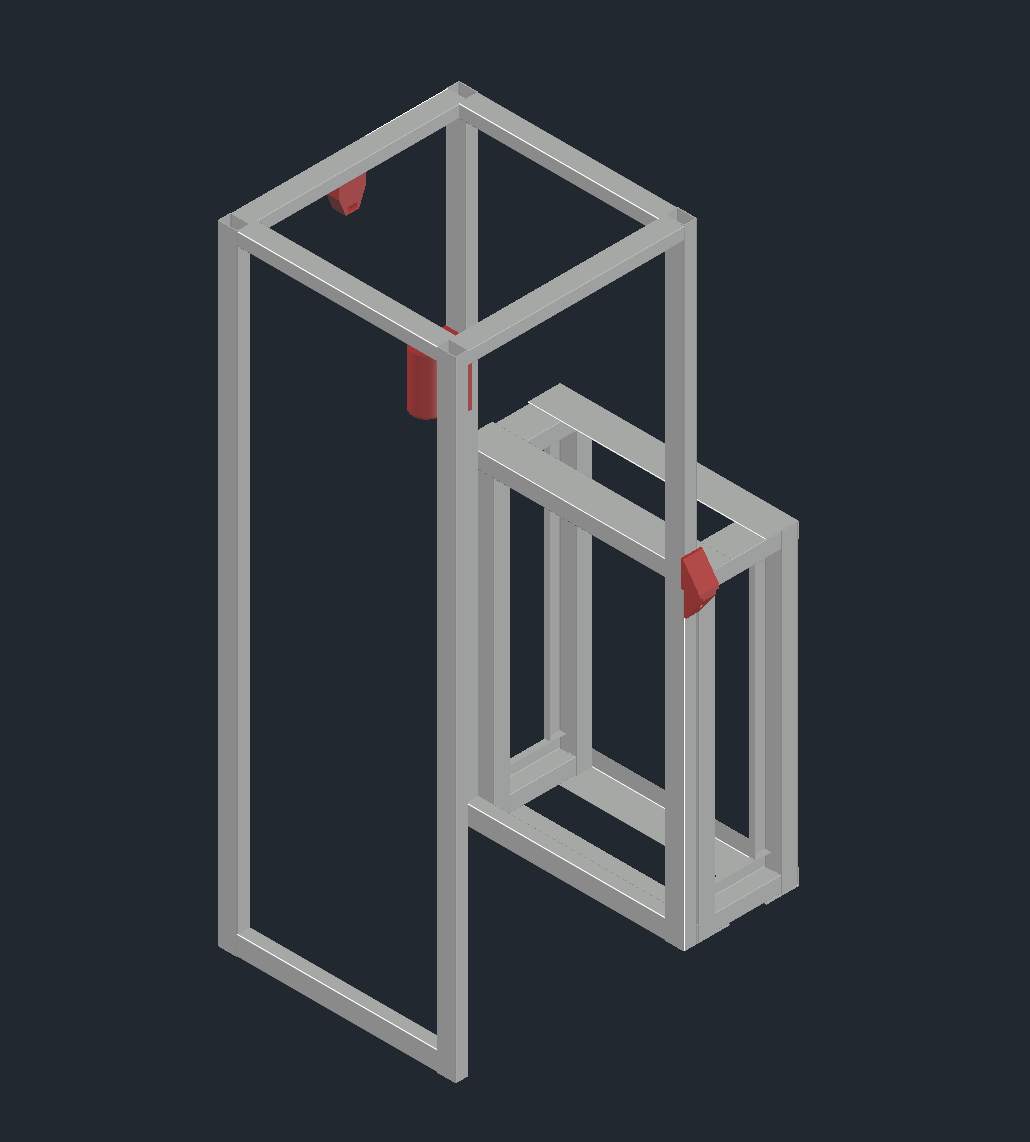




1. **Modelos 3D**

Todos los modelos están disponibles en el repositorio del proyecto (Ver sección 6), en el directorio src>Recursos>Modelos

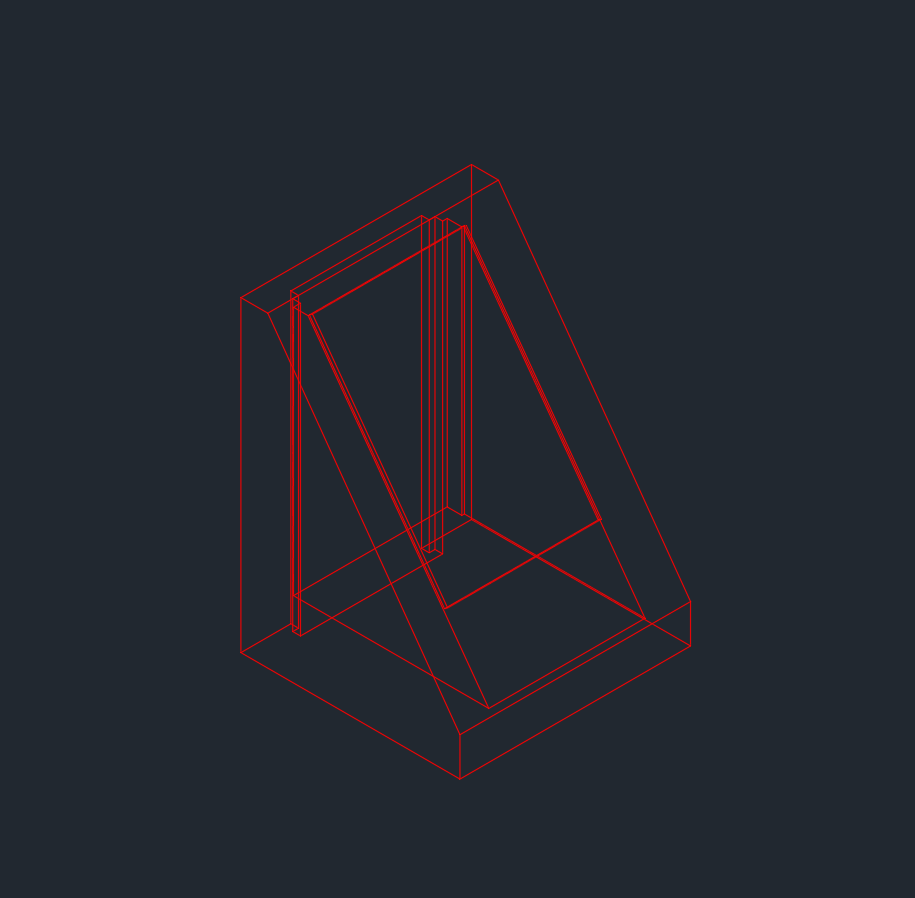
* 1. **Cabina completa**



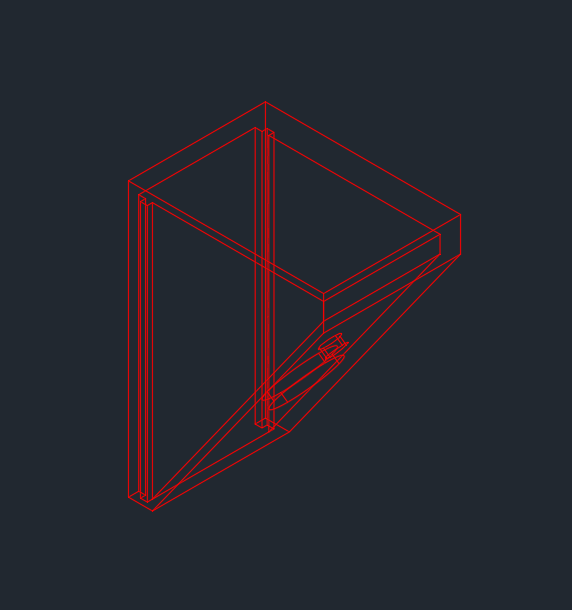
* 1. **Soporte de la cámara**



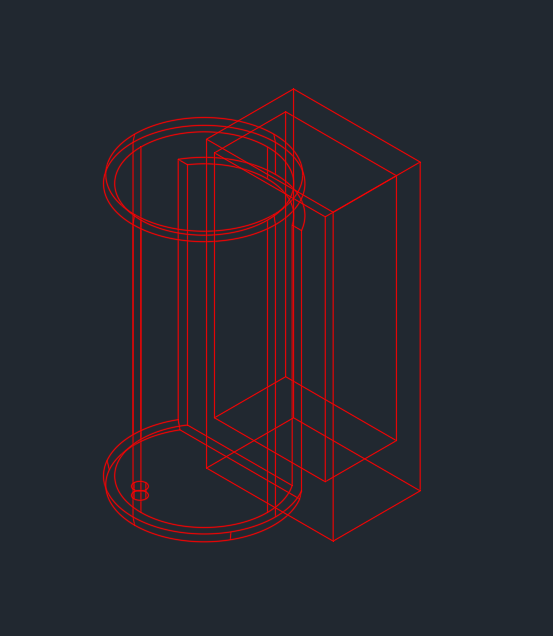
* 1. **Carcasa del lector de tarjetas**



* 1. **Carcasa del sensor de temperatura**



* 1. **Dispensador de alcohol en gel**



1. **Recursos**

[**Modelos 3D**](https://github.com/JuanICasareski/Automatic-Sanitary-Modular-Booth/tree/master/src/Recursos/Modelos)

[**Circuitos y PCB's**](https://github.com/JuanICasareski/Automatic-Sanitary-Modular-Booth/tree/master/src/Recursos/Circuitos)

[**Script**](https://github.com/JuanICasareski/Automatic-Sanitary-Modular-Booth/tree/master/src/Librerias)