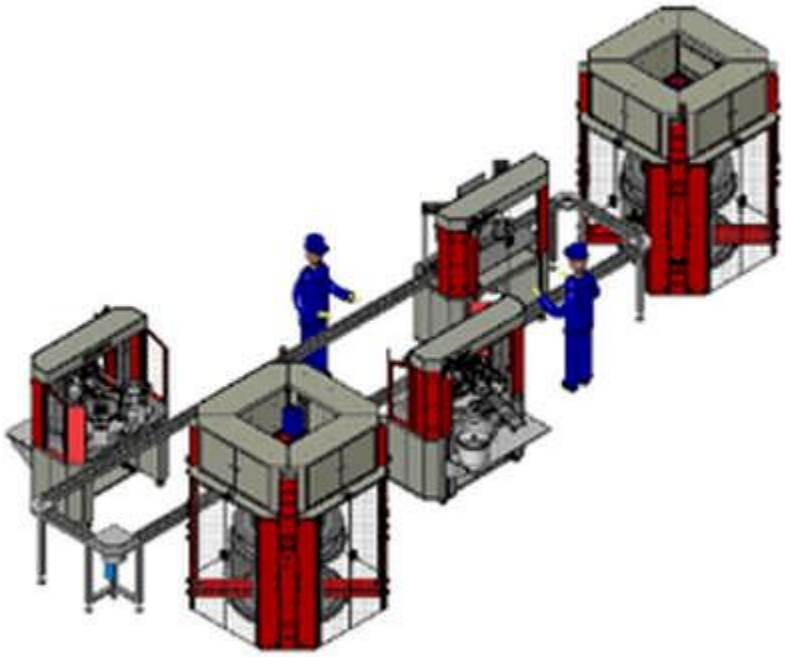
Manual Proyecto 1



Integrantes:

Javier Roberto Alfaro Vividor 201700644

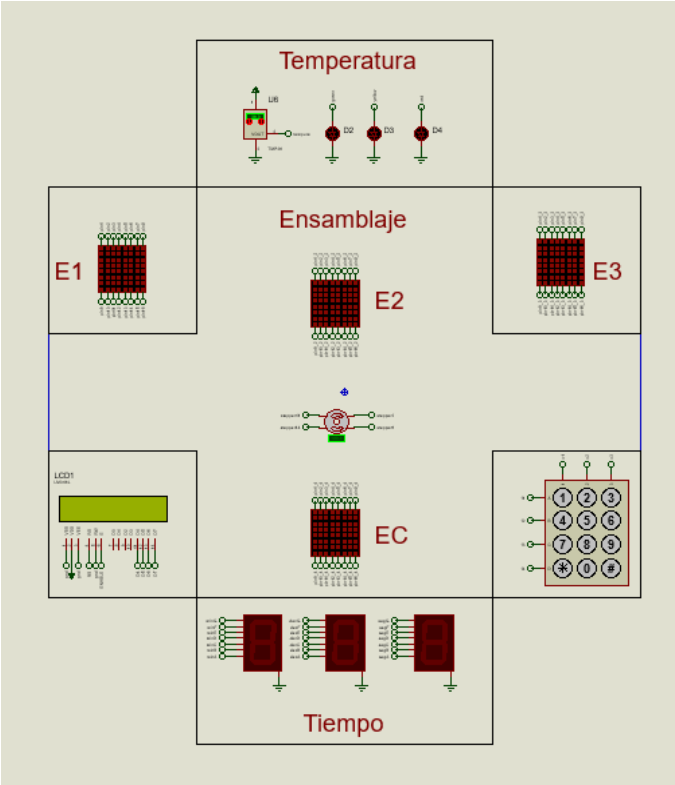
Joel Rodríguez Santos 201115018

Edin Emanuel Montenegro Vásquez 201709311

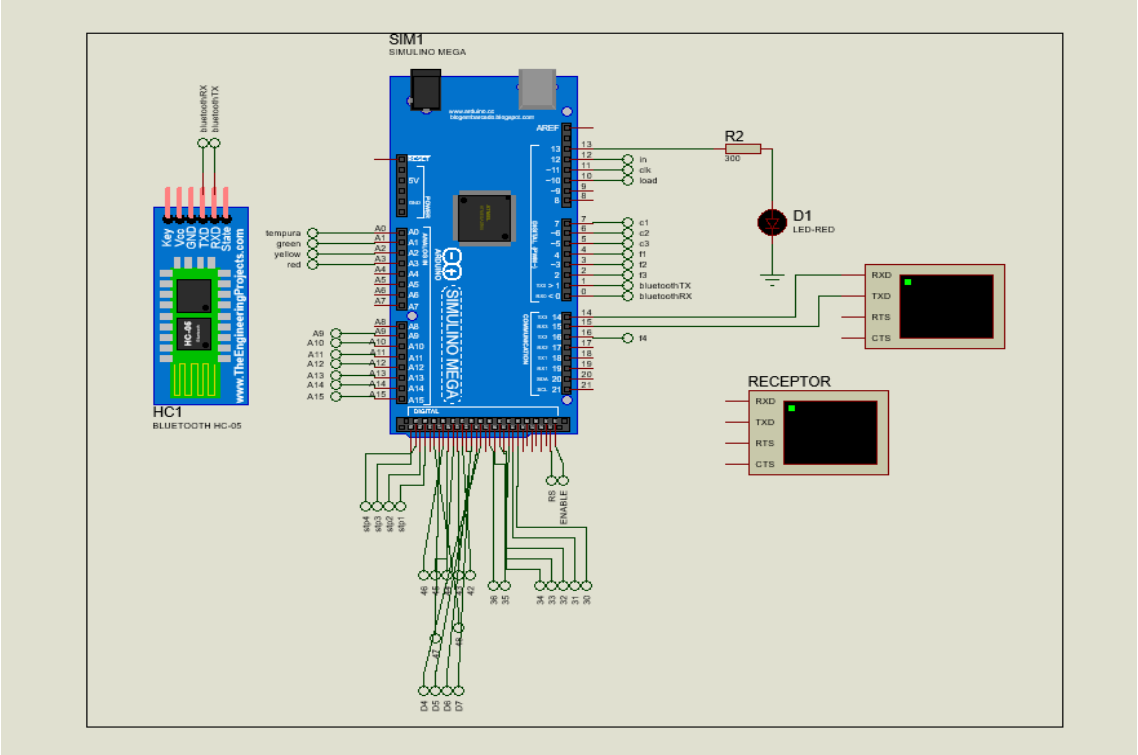
Fernando Augusto Armira Ramírez 201503961

Visualización de la aplicación:

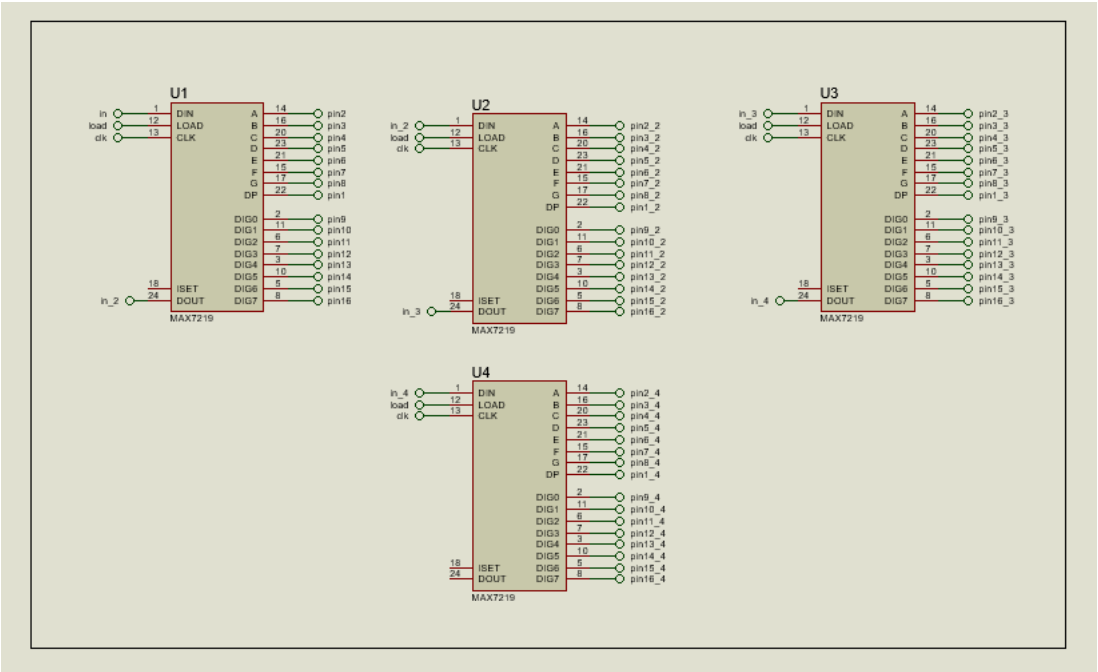
Circuito principal



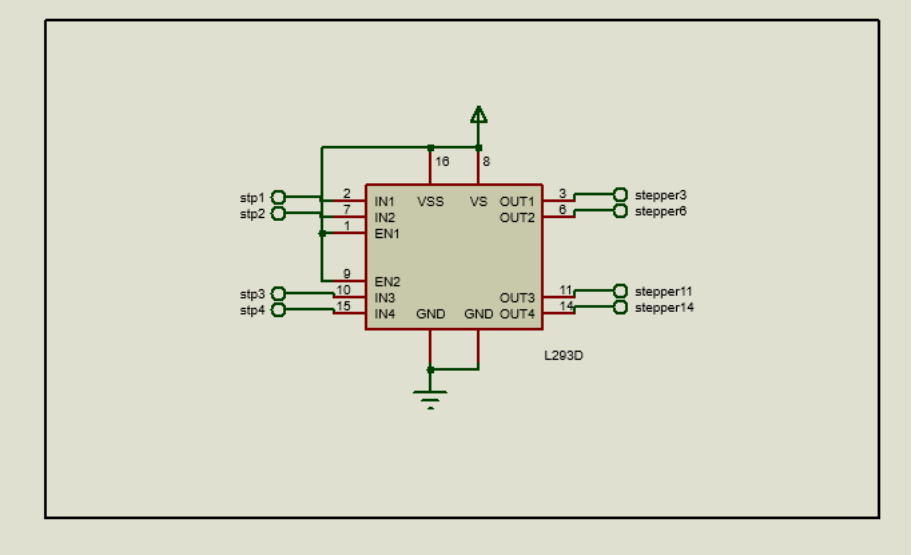
Arduino y HT05



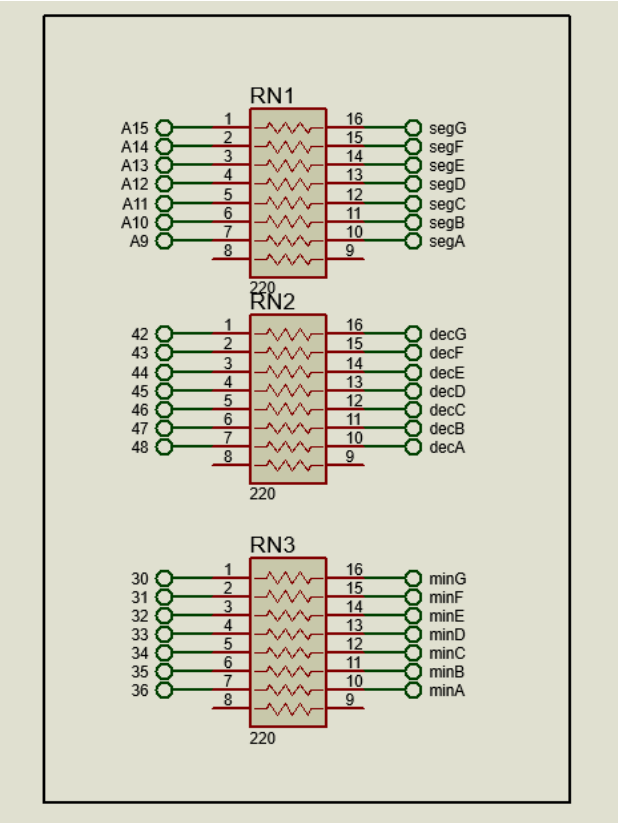
Max7219 que controlan las 4 estaciones



Circuito del motor Stepper



Circuito de los display 7 segmentos



Código en Arduino

Definición de los pines utilizados:

- Pines

PINES LCD

- RS (23)
- ENABLE(22)
- D4(38)
- D3(39)
- D2(40)
- D1(41)

PINES KEYPAD

- FILA 1 (4)
- FILA 2 (3)
- FILA 3 (2)
- FILA 4 (16)

- COLUMNA 1 (7)
- COLUMNA 2(6)
- COLUMNA 3(5)

PIN DE LED

PIN DE STEPPER

- LEDPIN (2)

Variables booleanas:

```
bool usuariocorrecto = false; //Bool para validación de usuario
static boolean mt1 = false; //Guarda el estado de la matriz 1
static boolean mt2 = false; //Guarda el estado de la matriz 2
static boolean mt3 = false; //Guarda el estado de la matriz 3
boolean datosRecibidos = false; //Determina si hay nuevos datos
recibidos
boolean matrizActiva = false; //Determina si alguna la matriz está
lista para imprimir
boolean stepperActivo = false; //Determina si el stepper fue activado
en la aplicación
```

Otras variables:

```
int vel_scroll = 0; //Variable para controlar la velocidad del mensaje
scroll
const String user = "123456"; //Contraseña
const byte rows = 4; //Variable que indica el # de filas en el teclado
const byte cols = 3; //Variable que indica el # de columnas en el
teclado
const int btnIn1 = 2; //boton de prueba pasar a pantalla 2 comentar
cuando ya no sea necesario
//int btnIn1State = LOW; // boton prueba
int contadorMatriz = 0; //contador para insertar en las diferentes
posiciones del array las filas de diseño
int contadorLugar = 1; //Contador para el lugar de impresión de cada
matriz
String instruccion = ""; // recibe instruccion
char opcion; // concatena entrada
int contador = 0; //contador para concatenar instrucciones
int producto[8]; //Matriz que guarda el producto a imprimir
int contadorTemp = 0;  //Contador temporal del número de matrices
long tiempoMillis = 0; //Variable para llevar el control del tiempo y
poder contar en el reloj
String mensajeEstacion = "ESTACION "; //Mensaje que mostrará la
estación actual
// variables temporales para el reloj
int dectemp = 0; //Variable que lleva el conteo de las decimas
transcurridas
int mintemp = 0; //Variable que lleva el conteo de los minutos
transcurridos
int sectemp = 0; //Variable que lleva el conteo de los segundos
transcurridos
```

```
int tempInicial = 0; //Inicialización del tiempo en 0
String valor; //Variable que guardará la conversión de la temperatura
para transferirla por medio del puerto serial
String conjunto = "";

byte UNO[8] {B00000000, B00011000, B00011000, B00111000, B00011000,
B00011000, B00011000, B01111110}; //Número 1 en formato byte[] para
mostrar en matriz

byte DOS[8] {B00000000, B00111100, B01100110, B00000110, B00001100,
B00110000, B01100000, B01111110}; //Número 2 en formato byte[] para
mostrar en matriz

byte TRES[8] {B00000000, B00111100, B01100110, B00000110, B00011100,
B00000110, B01100110, B00111100}; //Número 3 en formato byte para
mostrar en matriz
```

Librerías:

- *TimerOne* => Librearía agregada para utilizar excepciones de tiempo.
- *LedControl* => Librearía utilizada para controlar la pantalla LCD.
- *Key* => Librearía utilizada para controlar el teclado.
- *KeyPad* => Librearía utilizada para controlar el teclado.
- *LiquidCrystal* => Librería utilizada para controlar los max7219.
- *Stepper* => Librearía utilizada para controlar el motor Stepper.

Estructuras:

- *msj* => Struct utilizado para guardar los estados de las matrices de leds.

Métodos:

- guardarObjeto(String cadena)
- imprimirMatriz(char numMatriz, int lugar)
- imprimirNumeroEstacion(int numeroMatriz, int lugar)
- ingreso()
- letrero()
- mensajeScroll()
- reiniciar()
- stepperMove()
- temperatura()
- tiempo()

Explicación métodos

guardarObjeto(String cadena):

Método encargado de guardar los estados de cada led del objeto dibujado en la aplicación para después imprimirlo en las estaciones del programa cuando sea solicitado. Estos estados son guardados en una matriz int llamada producto[8].

imprimirMatriz(char numMatriz, int lugar):

Método encargado de imprimir el producto guardado en la variable producto[8] en la matriz seleccionada y en el lugar seleccionado. Dependiendo del lugar seleccionado solo se imprimirá un porcentaje del producto en la matriz. Al finalizar de imprimir el producto en cada una de las 3 estaciones se imprimirá el producto final en la estación central. Las matrices están programadas para parpadear 4 veces mientras se imprime el producto. También se puede mandar a llamar el método stepperMove() si es que en la aplicación se activa esta opción. Antes de la impresión del producto siempre se manda a llamar al método imprimirNumeroEstacion() para imprimir el número de estación en donde se encuentre el producto.

imprimirNumeroEstacion(int numeroMatriz, int lugar):

Método encargado de imprimir en la cuarta estación el número de matriz en el que se encuentre el producto. Recibe como parámetro el número de la matriz y el lugar en el que se está utilizando, este puede ser 1,2 o 3. Cuando el lugar llega a 4 se le indica al método que se debe imprimir el producto en lugar del número de la matriz.

ingreso():

Método que valida que el usuario ingresado a través del keypad sea el correcto. De ser así el usuario tiene acceso a la aplicación móvil. De lo contrario se muestra mensaje de error en el LCD y el usuario no puede acceder a la aplicación.

letrero():

Método que se usa de intermediario para llamar al método mensajeScroll().

mensajeScroll():

Método encargado de imprimir el mensaje ACYE 1 G12 en las estaciones que no están siendo utilizadas. Utiliza una matriz de 2 dimensiones de tipo booleana guardada en la memoria flash para imprimir el mensaje con la librería LiquidCrystal.

reiniciar():

Método encargado de reiniciar todos los circuitos y todas las variables del programa cuando recibe la señal de la temperatura en estado rojo o del botón aceptar o emergencia desde la aplicación.

stepperMove():

Método que hace girar al motor stepper y pueda visualizarse en la aplicación, este método le asigna una velocidad de 8 y define 9 steps al motor stepper.

temperatura():

Método que verifica el estado del TMP 36 y que recibe dicho estado, luego con map convierto el valor obtenido para poder manejar la temperatura de una manera mas precisa, luego se compara esta variable en base a tres posibles estados(ok, warning, error), y si el estado es crítico (error) reinicia y para el sistema.

tiempo():

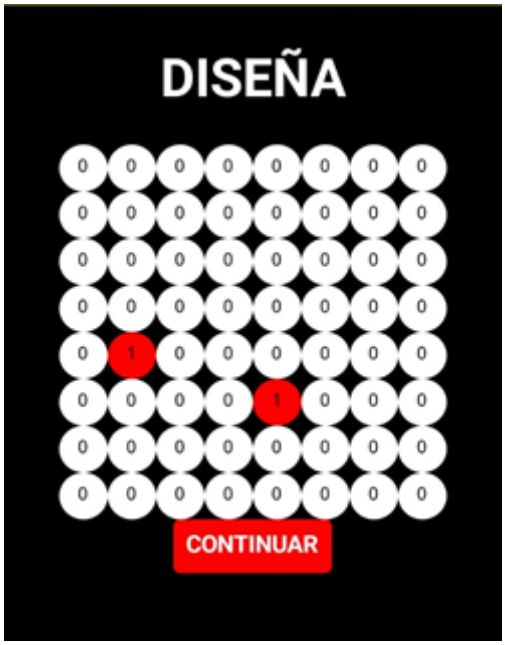
Método que cuenta el tiempo de ejecución del programa, se va sumando un segundo a cada segundo real este al llegar al decimo segundo entra en un modo de decimal de segundo el cual aumenta su valor, y cuando llega el quinto valor de decima hace lo mismo que decimas de segundo solo que para la variable de minuto, este al llegar al 9:59 se reinicia debido a que se llegó al tiempo máximo. Se hace uso de DDR para imprimir en los displays de 7 seg el valor correspondiente al segundo,decima,min.

Aplicación android

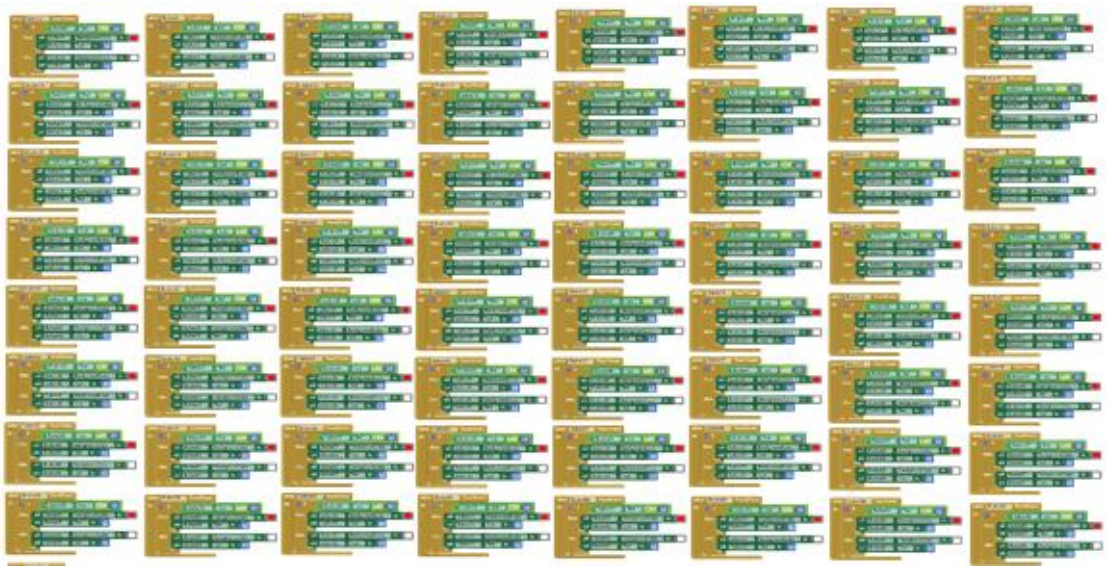
Esta aplicación se comunica con Arduino mega por el módulo bluetooth del teléfono hacia el de arduino(se especifica en este documento el módulo bluetooth utilizado).

- **PANTALLA DISEÑO**

Esta pantalla ayuda a dibujar el diseño que se desea fabricar a través de las tres estaciones, la pantalla tiene el siguiente aspecto:

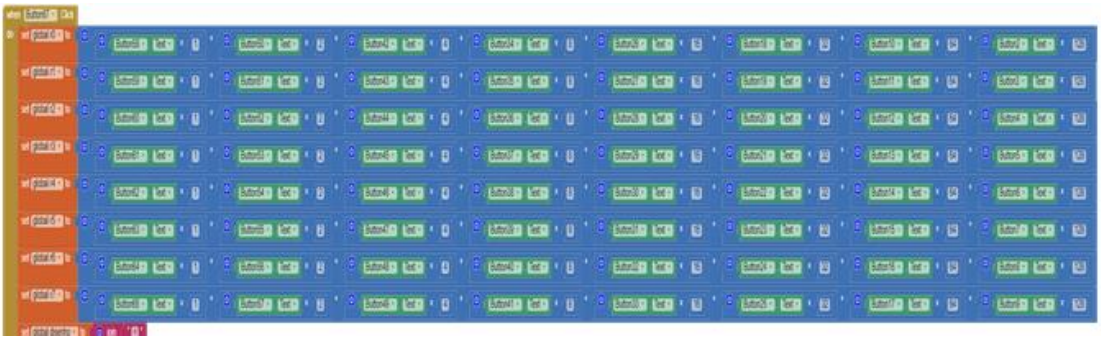


los círculos de color rojo nos indicaran los segmentos que se mostraran en la matriz, para esta funcionalidad se utilizaron 64 botones los cuales al mostrarse en rojo guardan un numero “1” y de lo contario guardan un “0” como se puede apreciar en este diagrama de bloques:



Cada bloque representa un botón(círculos de la pantalla de diseño).

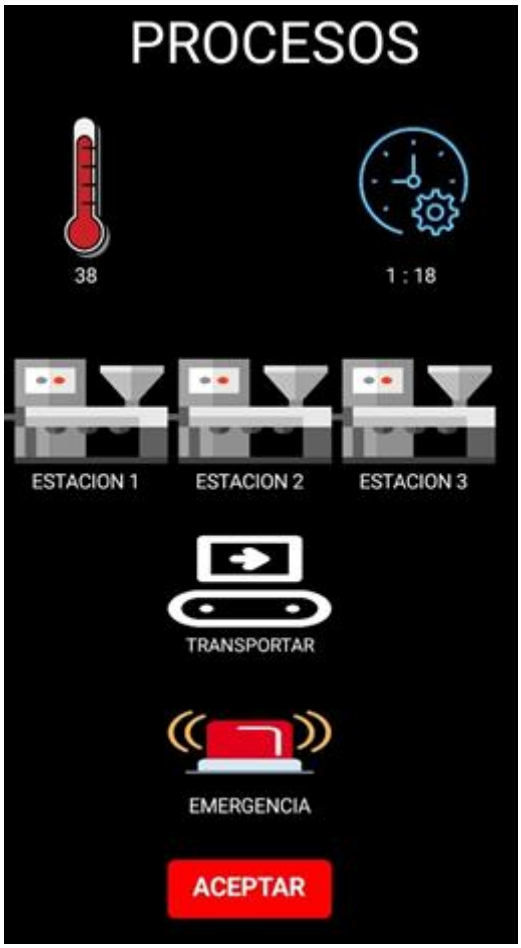
Al presionar el botón continuar de esta pantalla se manda el diseño a través de la conexión serial del bluetooth, y el modo en que se envía es haciendo una validación matemática en la cual se envía la columna de ponderación respectiva, empezando de la columna menos ponderada (1) hasta las más ponderada (128), en el siguiente diagrama de bloques se especifica el uso de esta validación:



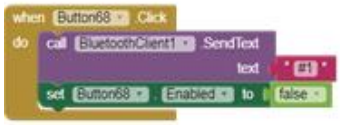
Toda esta información se guarda por cada columna en variables luego se concatena la información en una sola variable de la siguiente forma:
"<col0,col1,col2,col3,col4,col5,col6,col7", en el siguiente bloque se muestra como se concatena y se envía la información:

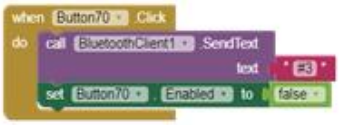
- PANTALLA PROCESOS

Esta pantalla es la encargada de escoger el orden en el cual las estaciones trabajaran y se pasaran, para esto se cuenta con tres botones indicando la estación y su número y un botón para la cinta transportadora, la cual se encarga de transportar el diseño de una a otra estación. La pantalla tiene el siguiente aspecto:



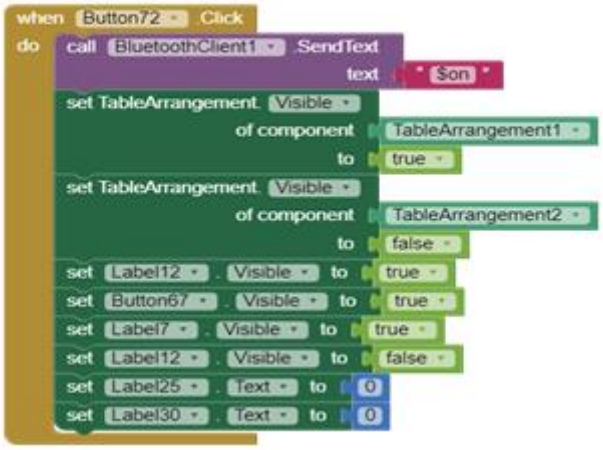
La manera en que se envía la información de las estaciones es por la conexión bluetooth enviando #numero_de_estacion como se puede apreciar en el siguiente diagrama de bloques:





La cinta transportadora se envía a través del siguiente bloque:

El botón de emergencia pausa en cualquier momento el proceso de las estaciones volviendo a la pantalla de diseño en la app, como se puede apreciar en los siguientes bloques:



El botón aceptar redirige a la pantalla de diseño cuando ya se ha pasado por las tres estaciones, los bloques utilizados para esta funcionalidad se muestran a continuación: