UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO ELECTRÓNICO Y CONTROL DE UNA COCTELERA AUTOMÁTICA

Autor

Víctor de Nalda Tárrega

Director

Carlos Ricolfe Viala

Trabajo de Fin de Máster Ingeniería Mecatrónica

Universitat Politècnica de València

Escuela Técnica Superior de Ingeniería de Diseño

Septiembre 2019

DISEÑO ELECTRÓNICO Y CONTROL DE UNA COCTELERA AUTOMÁTICA

Autor

Víctor de Nalda Tárrega

Universitat Politècnica de València

Escuela Técnica Superior de Ingeniería de Diseño

Septiembre 2019

Resumen

En este Trabajo de Fin de Máster (TFM a partir de ahora) se va a realizar el diseño electrónico y el control de una coctelera automática. Además de esto, también se ha procedido a realizar la construcción de dicha coctelera junto a mi compañera Sonia Pérez de Villar Pascual. La idea del proyecto surge en un trabajo previo en el primer curso del máster donde se hizo un modelo simple basado en la plataforma de prototipado electrónico Arduino. La metodología que se seguirá es la misma que se utilizó en la asignatura previa, el desarrollo gradual desde el mínimo producto viable.

Las tareas controladas desde el microcontrolador Arduino son las de control de llenado del vaso y control de las distintas bombas peristálticas que regulan el llenado del vaso desde las distintas botellas disponibles. También se ha diseñado una interfaz para poder controlar la aplicación de la coctelera desde el móvil mediante Bluetooth.

En la memoria del TFM se detallan todos los pasos y programas que se han realizado para obtener el correcto funcionamiento del proyecto, así como los distintos procesos descartados a la hora de terminar el proyecto y el porqué de su modificación.

Agradecimientos

A Carlos Ricolfe, por los consejos y ayuda para organizar el trabajo de la manera más asequible y sencilla posible. A mi compañera de trabajo Sonia Pérez por colaborar cada vez que las modificaciones en el control de la máquina requerían nuevas modificaciones en el diseño mecánico. A mi familia y amigos por su apoyo, sin el cual no hubiera podido realizar este trabajo.

Índice

1. Introducción

1.1 Background del Proyecto

En la asignatura de Automatización Distribuida de este mismo máster se ideó por primera vez la realización de esta máquina. Se construyó un pequeño prototipo que, aunque cumplió las funciones de controlar de manera automática el llenado de un vaso desde dos distintas botellas, se consideró que aún se necesitaban ciertas mejoras para poder considerar a la máquina una coctelera automática y poder comercializarse.

En el prototipo anterior, el usuario controlaba el proceso con comunicación bluetooth y una aplicación móvil, se seleccionaba la bebida que quería tomar, pudiendo elegir entre tres opciones. La estructura de madera soportaba dos botellas boca abajo que, mediante un miniservo y un acoplamiento a una llave de paso, permitía el paso o no del líquido a través de unos conductos que llevaban al vaso. Para la medición mientras el llenado del vaso se utilizó un sensor ultrasonidos.

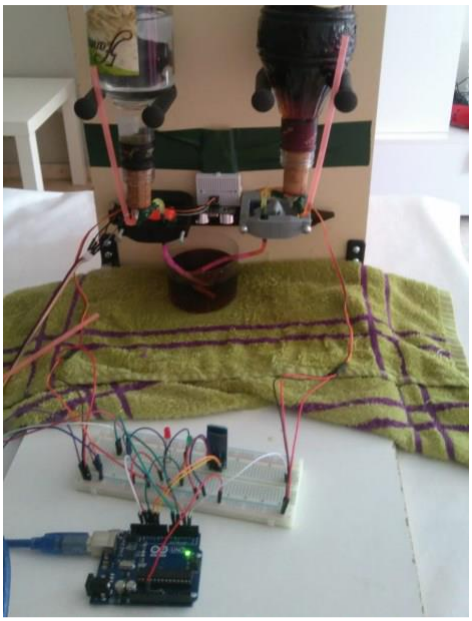

Ilustración 1 Primera versión de la coctelera automatizada.

De este trabajo previo se sacaron varias conclusiones que había que mejorar para la siguiente versión:

- **Inestabilidad en las lecturas con el sensor ultrasonido.** Al no caer el liquido de manera fluida, la condición del líquido dentro del vaso se convertía en régimen turbulento, creando oscilaciones e impidiendo una lectura estable al sensor ultrasonido.
- **Modificar el método de llenado.** Él método de llenado dejando caer el líquido por gravedad al abrirse las llaves de paso creaba muchos problemas al hacerse un vacio y siendo necesario la apertura de un segundo agujero que comunicara con el aire para que pudiera salir el aire de la botella por ahí.
- **Hacer el proyecto más escalable.** En las condiciones del programa anterior, la inclusión

de una nueva botella requería una modificación completa del programa.

El objetivo de este Proyecto es el desarrollo de una máquina real que cumpla la función de coctelera de forma automática. Partiendo del conocimiento obtenido en la experiencia previa, se han establecido unos requisitos mínimos y unas metas deseables.

Ilustración 2 Captura de pantalla de la interfaz de usuario de la primera versión.

1.2 Objetivos del Proyecto

El objetivo del trabajo es poder realizar un proyecto profesional centrado en aplicaciones mecatrónicas. Partiendo del proyecto inicial mencionado anteriormente y de los conocimientos adquiridos durante el Máster de Ingeniería Mecatrónica, se ha realizado esta coctelera automatizad. En este trabajo, se hablará sobre todo del control automático y diseño electrónico, ya que la solución desde un punto de vista más mecánico se realiza en otro TFM.

Hablando exclusivamente del contenido de este trabajo, los objetivos del proyecto eran utilizar los conocimientos adquiridos en asignaturas como Electrónica, Automatización Distribuida o Sistemas Embebidos para poder realizar aplicaciones reales. En este caso se controla el proceso de llenado, pesaje, elección de bebidas y diseño de interfaz de usuario con:

- **Arduino** Microcontrolador donde se controla las bombas peristálticas de llenado mediante la elección del usuario a través de AppInventor y el sensor de peso que determina la cantidad de líquido de cada bebida en el vaso.
- **AppInventor** Interfaz de usuario comunicada vía Blueetoth con la placa de Arduino.
- **Fritzing** Utilizado para realizar los esquemas de conexionado de la placa Arduino.

1.2.1 Requisitos mínimos

Para la realización de este proyecto, los requisitos mínimos que se han considerado, a raíz del análisis del trabajo anterior, que deberían cumplirse son los siguientes:

- **Interfaz de usuario sencilla.** Hablar de como es. Esta interacción puede ser mediante una botonera en la máquina o Wireless con módulos de bluetooth o Wifi. Es imprescindible que el usuario pueda seleccionar la bebida que desea de forma intuitiva y muy visual para asegurar un funcionamiento correcto. Finalmente, se ha decidido controlar la coctelera mediante una aplicación móvil con bluetooth que enlaza con el microcontrolador con una comunicación en serie. Desde la propia aplicación se envía cierta información que modifica algunas variables del programa. Se ha decidido realizar una interfaz sencilla en la que primero escoges el número de bebidas que se desea mezclar y la cantidad de cada una de ellas para, posteriormente, iniciar el llenado del vaso.
- **Control del contenido del vaso.** Debido a las perturbaciones provocadas en el vaso al llenarlo de líquido, un sensor de ultrasonidos no es el sensor óptimo para nuestra aplicación. Se requiere un control más preciso y rápido cuya fiabilidad sea mayor. Sea decido utilizar un sensor resistivo de fuerza FSR conectado mediante una entrada analógica a la placa de Arduino. Se ha comprobado que es mucho más estable, ya que es independiente del régimen en el que se encuentre el líquido del vaso.
- **Control de salida del líquido de la botella.** Se descartan también la salida del líquido de la botella mediante llaves de paso controladas por motores. En un primer momento se optó por no utilizar bombas peristálticas y sí un mecanismo que cogiera la botella y la volcara. Sin embargo, la dificultad de la solución mecánica hizo que se descartara también. Más adelante se profundizará en esta solución y porqué fue rechazada. La solución de las bombas peristálticas viene con una opción de limpieza, dejando pasar agua por los tubos y eliminando restos de azúcar que se puedan haber quedado debido al flujo constante de bebidas alcohólicas o refrescos.

1.2.2 Metas

Además de determinar unos requisitos mínimos provenientes de los fallos y observaciones realizados en el primero modelo, se ha querido determinar también unas metas a cumplir en esta nueva versión mejorada de la coctelera automatizada.

- **Optimización de espacio y recursos.** Se va a buscar la creación de un producto lo más compacto posible. Además, que uso no requiera constantemente de recambios y que su mantenimiento sea mínimo para utilización a nivel usuario. De esta manera, el enfoque de la coctelera es de vista comercial.
- **Escalabilidad.** A pesar de que de primeras se desarrolle un prototipo o sistema con un número limitado de botellas, la idea es crearlo de tal manera que funcione igual para 1 que para 10 botellas. Así, el usuario puede colocar en la máquina el número de botellas que quiera. Al incrementar el número de botellas simplemente habría que cambiar la placa de Arduino por una que admitiera más entradas/salidas digitales, la obtención de otro motor de bomba peristáltica. El objetivo es no tener que modificar el programa por

completo al añadir una nueva botella.

- **Mantenimiento mínimo.** Se pretende lograr que el mantenimiento y limpieza de la máquina se sencillo. No se pretende que el usuario final tenga un gran conocimiento ni de la programación ni del sistema mecánico incorporado en la coctelera.

1.3 Organización de la Memoria

El objetivo de esta memoria es explicar los procesos realizados durante el proceso de diseño de la parte electrónica y programación de la coctelera.

Primero, se ha hecho una mención a la primera versión de esta coctelera automática y como se han establecido las mejoras a realizar para este nuevo proyecto. Posteriormente, se divide la explicación del diseño electrónico y programa de la coctelera en tres partes:

- Control de volumen de líquido
- Control de los motores
- Interfaz de usuario y comunicación.

En cada apartado se explica el porqué de haber escogido la solución que se ha terminado escogiendo y el porqué se ha rechazado otras posibles ideas que se han acabado descartando para lograr los objetivos establecidos anteriormente.

Durante las secciones se comentará también como se ha escrito el código en relación con la sección mencionada. Sin embargo, se incluye una sección final en la que el código está explicado de una manera más exhaustiva.

Finalmente se desglosa un presupuesto de la solución de ingeniería de la coctelera y de los componentes necesarios para la construcción del apartado electrónico y de control de la coctelera. Además, también hay una conclusión del trabajo y bibliografía de todas las páginas consultadas para recoger la información necesaria para poder realizar el proyecto.

1.4 Herramientas Utilizadas

Para lograr la solución final de este trabajo ha sido necesaria la utilización de los siguientes Softwares, cada uno para una aplicación concreta.

- **Arduino IDE.** Software de programación propio de Arduino. Junto con el software IDE es necesaria la placa del microcontrolador (hardware) para poder realizar todas las operaciones programadas y almacenar todas las variables que componen el programa de control de la coctelera.
- **Fritzing.** Dibujo de esquemas electrónicos en los que se muestra el conexionado de la placa con las distintas entradas y salidas.
- **AppInventor.** Aplicación para diseñar las propias aplicaciones para el móvil. En ella se diseña la interfaz de usuario.

2. Control de llenado de vaso

Lo primero que se ha de resuelto, durante el diseño electrónico de la máquina, es el cálculo de volumen de líquido, para saber qué cantidad de cada tipo de bebida se ha introducido en el cock-tail. La primera opción que se valoró fue utilizar un sensor de caudal conectado a una pequeña manguera proveniente de los distintos orificios de las botellas. Sin embargo, esto no cumplía con nuestro objetivo de hacer la máquina lo más escalable y óptima posible, debido a que la introducción de una nueva botella conllevaría un nuevo sensor de presión que tendría que ser instalado y configurado.

Por ello, se decidió finalmente trabajar con el peso del líquido, asumiendo que la densidad de las distintas bebidas alcohólicas son las mismas y considerando despreciable la posible diferencia de densidad entre dos bebidas alcohólicas de graduación similar (vodka y whiskey, por ejemplo). Siendo conocidas la densidad del etanol y del agua, y de las graduaciones alcohólicas de las distintas bebidas, calcular el volumen de líquido midiendo el peso es trivial.

2.1 Sensor de Peso

La primera opción que se valoró fue la de diseñar un puente Wheatstone con 4 sensores de peso, similar a los utilizadas en las básculas de peso domésticas.

2.1.1 Funcionamiento del Puente de Wheatstone

El puente de Wheatstone es un método para medir cambios de resistencia muy exacto, es por ello por lo que se suele configurar a la hora del uso de galgas extensiométricas.

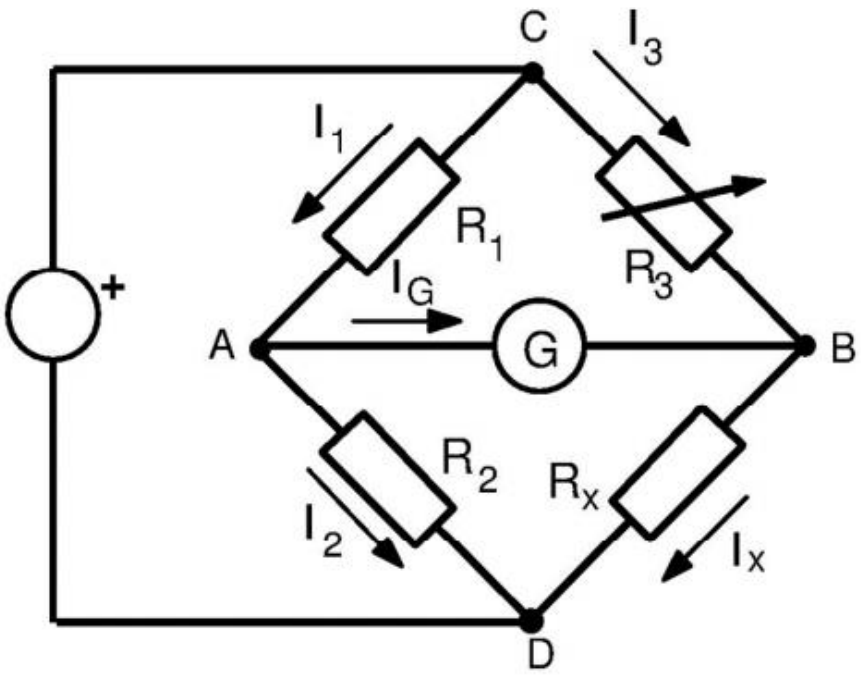

Ilustración 3 Configuración típica de un puente de Wheatstone

El funcionamiento más típico de un puente Wheatstone es el de alimentar dos puntos opuestos del puente a una tensión. Si todas las resistencias están equilibradas, no habrá existirá una diferencia de potencial entre los otros dos puntos opuestos que se están midiendo. Esto es porque, la tensión aplicada se divide entre las dos mitades del puente (R1, R2 y R3, R4 en el ejemplo de arriba) siendo una diferencia de tensión nula si las resistencias son iguales.

En el caso de la figura que se observa anteriormente, si la resistencia variable (asumimos que la de la galga extensiométrica para el caso de un sensor de peso) modifica su valor, las dos ramas del puente no tendrán ya la misma resistencia, por lo que surgirá una diferencia de potencial entre los lados opuestos que no están siendo alimentados. Es lo que se llama un puente desequilibrado.

Dependiendo de si la galga está siendo sometido a compresión o extensión, la resistencia de la rama de resistencia variable aumentará o decrecerá y, la diferencia de potencial entre ambas ramas será negativa o positiva.

2.1.2 Aplicaciones para nuestro proyecto

Debido a que nuestra máquina está controlada por un microcontrolador digital, la diferencia de potencial entre las dos ramas (valor analógico) ha de ser convertida a un valor digital y calibrada para poder trabajar con ello. Esta conversión se realizaba mediante el módulo HX711 al que iba conectado el puente y a la propia placa.

Sin embargo, debido a que los sensores comprados estaban pensados para una báscula de baño de uso doméstico, el rango de peso para el que estaba diseñado no coincidía con los pesos mucho menores de nuestra aplicación.

Por ello, el sensor de peso con galgas extensiométricas formando un puente de Wheatstone fue rechazado finalmente, y se decidió por un sensor de fuerza resistivo de unas especificaciones de peso adecuadas para nuestra aplicación.

2.2 Sensor de Presión

Tras desestimar realizar el control de llenado con sensores de peso formando un puente de Wheatstone, se ha decidido utilizar un sensor FSR (Force Sensitive Resistor).

2.2.1 Funcionamiento FSR

El sensor FSR se conecta por un lado a +5V proveniente de la placa de Arduino y por el otro a una resistencia *“pulldown”* que está conectada a masa. Se conecta creando un divisor de tensión. El punto entrela resistencia fija y el resistor variable (FSR) se conecta a la entrada analógica del microcontrolador.

Ilustración 4 Divisor de tensión formado por el FSR

Al incrementar la fuerza en el sensor y reducirse la resistencia del FSR, la resistencia total de la rama en la que están conectadas las dos resistencias (fija y variable) decrece y, por la tanto, aumenta la intensidad circulando por ambas resistencias. Es por ello que, como la resistencia fija no varía su valor, la tensión entre masa y el punto de conexión con la placa de Arduino aumenta.

Ilustración 5 Sensor FSR

La ecuación de la tensión entre la entrada analógica de la placa y masa es la siguiente:

$$V_o = V_{cc} * \frac{R}{(R + FSR)}$$

La tensión V_o es inversamente proporcional a la resistencia del sensor FSR.

2.2.2 Adaptación del FSR al programa de Arduino

Al introducir la tensión por una entrada analógica de la placa, el propio microcontrolador de Arduino realiza la conversión con una resolución de 10 bits (0-1023 niveles de tensión distintos).

Como el rango de funcionamiento del sensor era de 0-15N, se puede asumir que prácticamente tenia una resolución de 1 ml. Es por ello, que consideramos que este sensor se acoplaba mucho mejor a nuestra aplicación que el puente de Wheatstone mencionado previamente.

Sin embargo, como el tamaño del sensor era demasiado pequeño para poder cubrir la superficie del vaso con seguridad, se ha construido una pequeña superficie adaptadora con la superficie inferior acabada en tres pequeñas patas que se apoyan en el sensor y la parte superior con una superficie mayor en la que se apoya el vaso. De esta manera se consigue una lectura estable del peso del vaso (y por lo tanto cantidad de líquido introducida).

Ilustración 6 Adaptador vaso-sensor

Una de las mejoras con la versión anterior, es que no es necesario que los vasos pesen exactamente igual, ya que en el momento que se inicia un proceso de llenado de vaso, éste lo hace sin haber introducido el vaso en la superficie y, solo cuando ha detectado un incremento del peso mínimo (guardado como la variable *int peso_vaso*) se hace una tara automática y comienza el llenado de la primera botella.

2.2.3 Uso de la Tara

Durante el programa, antes de introducir una nueva bebida, la función de llenado realiza una tara, para que, si se ha echado más o menos bebida de lo que correspondía, ese error no afecte a la cantidad de la siguiente bebida. Es decir, para cada bebida se echa una cantidad específica, y no va acumulando con el peso de las cantidades anteriores.

Además, también se ha introducido un botón para que el usuario pueda realizar la tara en cualquier momento que quiera el usuario, independientemente del estado en el que se encuentre el programa.

```
// Tara Manual
if (digitalRead(11)==HIGH) {
  FSR_tare=fun_tara();
  digitalWrite(BUZZER,LOW);
}

// Restar la tara y evitar números negativos
FSR=analogRead(A0)-FSR_tare;
if  (FSR<0) FSR=0;
```

Ilustración 7 Muestra de cómo la entrada digital 11 realiza una tara del pesaje.

También se tiene en cuenta, que al haber realizado una tara con cualquier peso encima del sensor, y después retirarlo, la medida de la variable FSR podría ser negativa. Sin embargo, se ha tomado la precaución de que cuando esto ocurre, el valor de la variable sea 0.

Al guardar la variable *int FSR_tare* como la lectura de *analogRead(A0)*, la nueva variable *FSR* tendrá siempre en cuenta ese peso al hacer la resta y, por lo tanto, tomará ese peso instantáneo como 0.

2.2.4 Uso de la pantalla LCD

Además de la información que el usuario recibe desde la interfaz con la que se controla la máquina, y en la que elegir y monitorizar qué numero de bebidas y cantidades de líquido de cada bebida quiere usar en su *cocktail*, el usuario puede observar que cantidad de mililitros de cada bebida hay en cada momento en el vaso.

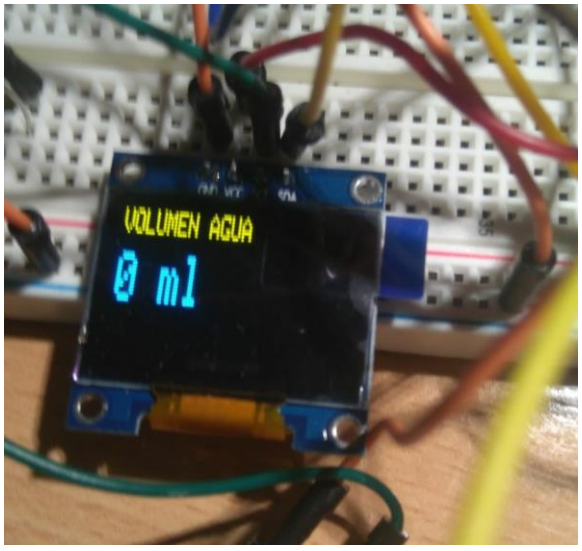

Ilustración 8 Pantalla LCD mostrando cantidad de líquido

La pantalla LCD de 1 pulgada reproduce periódicamente en cada ciclo (controlado para que sea cada 70 ms con la función *millis()*) el valor de la variable *int FSR_g*. Este valor es el que se obtiene tras realizar la función *map()* con los valores de la variable FSR.

La variable FSR muestra la resta de los valores en bits obtenidos de la medida de la entrada analógica A0 y la tara que tenga el programa en ese momento. Por lo tanto, estos valores discretos se traducen en una medida de "ml" con la variable *FSR_g* y que se utiliza para todos los cálculos, además de para reproducir en la pantalla e informar al usuario.

2.2. Interpolación de valores

Como se ha mencionado antes, es necesario traducir las medidas discretas de la variable *FSR* a unas medidas que se puedan considerar "ml" para que se pueda saber con certeza en qué momento la cantidad de cada líquido en el vaso es la deseada por el usuario y, por lo tanto, dejar de tener activado el motor de la bomba peristáltica.

Para conseguir que la interpolación lineal fuera lo más exacta posible, se ha realizado distintas medidas para distintas cantidades de líquido establecidas.

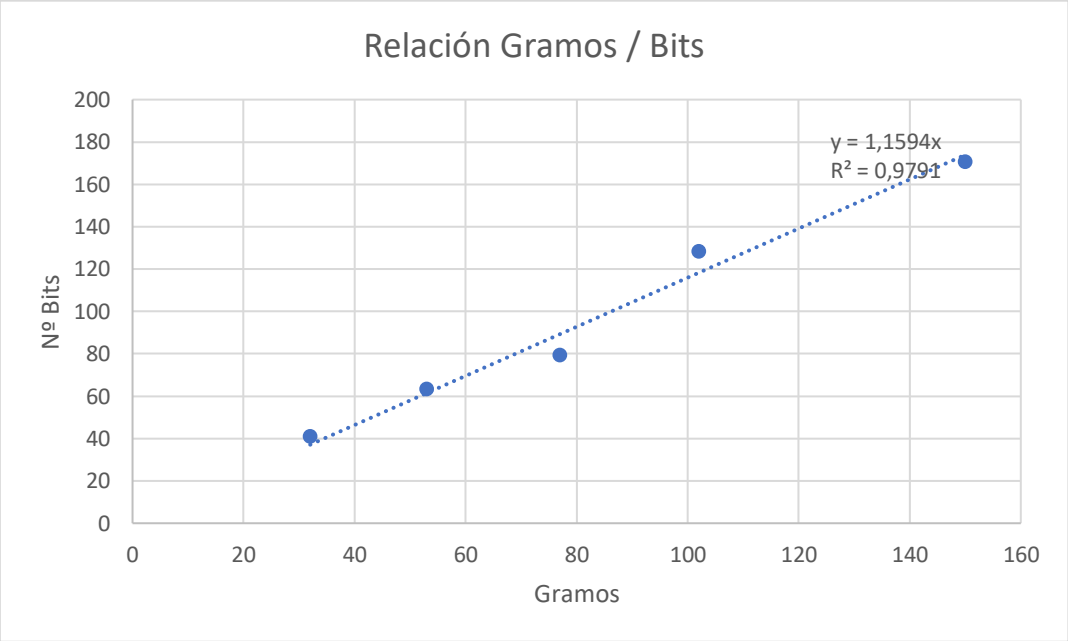

Ilustración 9 Relación entre gramos y nº bits leídos por el sensor FSR.

En este gráfico se puede ver la evolución lineal entre peso en gramos y número de bits que lee el microcontrolador desde la entrada analógica. Para realizar la prueba se pesaron distintas cantidades de líquido en una báscula de cocina comercial y se pesaron en el sensor FSR tres veces cada una, anotando el valor en bits para cada medida. La media de esas tres medidas es la que se ha computado en el gráfico, obteniendo una relación lineal muy aproximada como se puede observar.

```
FSR_g=map(FSR,0,1023,0,1186); // INTERPOLACIÓN EXCEL
```

Ilustración 10 Función con la que se ha interpolado los datos para obtener los ml.

Obteniendo pues la pendiente de la relación lineal mostrada en el gráfico, se utiliza la función *map()* para convertir el valor máximo de las medidas a 1186 ml y interpolar linealmente las demás medidas.

3. Control de los motores

Los motores son los encargados de llenar el vaso del líquido extraído de las botellas. Su funcionamiento y activación están controlados por el programa que, dependiendo distintos aspectos que el usuario final haya introducido y del peso instantáneo del vaso determinan si cas motor se ha de activar o no. La elección de qué tipo de motor y cómo programar dicho código que controlara los motores no ha sido trivial. Al principio del proyecto se comenzó con una idea que, finalmente se descartó debido a su complejidad mecánica. Finalmente se decidió por el uso de unas bombas peristálticas.

En ambos casos las salidas digitales controlan los relés que abren o no el circuito de potencia (los motores necesitan una tensión mayor que la que la propia placa de Arduino puede aportar). Sin embargo, en los motores que se habían barajado previamente (motor de apertura y cierre de pinza que agarraba la botella escogida y motor que giraba la propia pinza con la botella agarrada) el método de funcionamiento no era el mismo que con las bombas peristálticas finalmente escogidas. Los motores finalmente rechazados se explicarán brevemente y la memoria se centrará en los escogidos finalmente.

3.1 Motores de giro y apertura y cierre

La idea inicial del proyecto fue no tener que utilizar bombas peristálticas temiendo que su mantenimiento pudiera dar problemas. Por lo tanto, se optó por la opción de que las propias botellas, al inclinarse, dejarán pasar el líquido por gravedad al vaso.

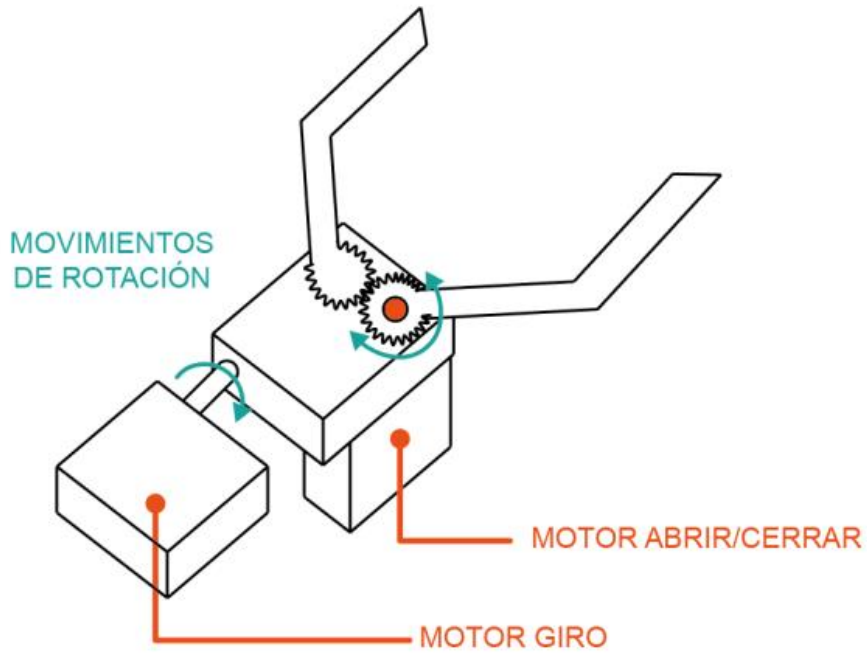

Ilustración 11 Idea inicial de los dos motores de la máquina.

4.1.1 Funcionamiento de los motores.

El motor de apertura y cierre y de la pinza iban regidos por un relé general que abría o cerraba la conexión del circuito de potencia y por un doble relé que gestionaba la dirección a la que giraba el motor. Dependiendo de si la pinza se debía de abrir o cerrar, la salida digital que gestionaba el control del relé de cambio de sentido estaba activa o no.

Las botellas irían equipadas con un tag RFID e irían pasando por delante del lector que, una vez identificada la botella deseada, pararía el carrusel y se iniciaría el proceso de cierre de pinza.

El motor de giro de la botella tendría un funcionamiento similar. Un relé doble es el que determinaría la dirección del giro de la botella. Si bien se ha de inclinar, o bien volver a poner recto en el carrusel otra vez.

La dificultad de trabajar con estos motores proviene del par que debería de soportar el motor que gira las botellas, y los tirantes que la propia estructura necesitaría para poder realizar el giro. El diseño y montaje de tal estructura desde el punto de vista mecánico era muy difícil.

4.1.2 Control de los motores.

Cómo se ha explicado anteriormente, los motores solo se activarían para el cierre y apertura; una vez cerrada la pinza, el motor no ejercería fuerza, sino que sería que el propio husillo no permitiría la apertura de la misma pinza.

El método de control del motor de giro sería análogo. Se activaría en el momento de giro y, el sentido vendría marcado por el nivel al que estaría la salida digital que controla el relé doble de dicho motor. La fuerza sometida por la propia botella al estar inclinada o girada sería soportada mecánicamente por los tirantes y el propio husillo que no podría avanzar si el motor no está activo.

En este caso, al enviar la información necesaria por el usuario (cantidad y tipos de bebida a mezclar), el carrusel empezaría a girar hasta detectar la primera botella por el lector RFID y comenzaría el proceso de llenado gestionado por los motores comentada anteriormente.

3.2 Motor bombas peristálticas.

Una vez rechazada la idea de coger la botella e inclinarla, se decidió volver a los motores que controlan bombas peristálticas para realizar el proceso de llenado del vaso. Mecánicamente, la solución es mucho más sencilla que anteriormente y, además, se sigue respetando la intención inicial de hacer el proyecto escalable. La introducción de nuevas botellas solo necesitaría la adquisición de otra bomba peristáltica. El programa se ha diseñado de manera robusta para que, al incrementar el número de botellas, solo haría falta incrementar el valor de una variable.

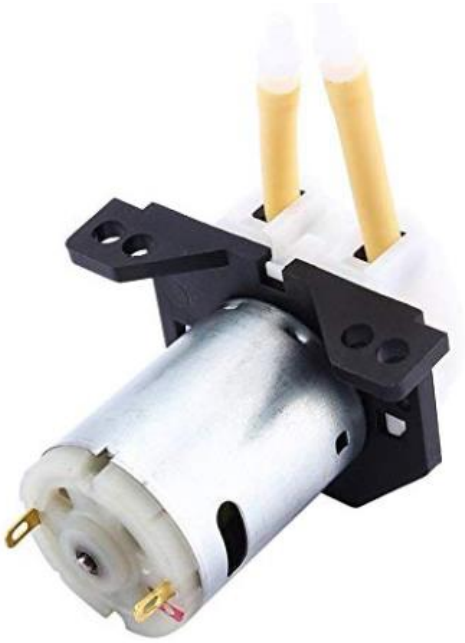

Ilustración 12 Bomba peristáltica de 12V

3.2.1 Tipo de motor

Las bombas peristálticas son un tipo de bomba hidráulica. El fluido es contenido y transportado dentro de un tubo flexible dentro de una cubierta circular de la propia bomba. Un rotor es el encargado de comprimir el tubo flexible y, mientras que el rotor da vueltas en un sentido u otro, la parte del tubo bajo compresión se cierra forzando al fluido a moverse a través del tubo.

De esta manera conseguimos el flujo del líquido hacia arriba de una manera sencilla y controlada. Para evitar posibles bloqueos por la densidad de los líquidos con alto contenido en azúcar, se ha tenido en cuenta la posibilidad de facilitar la limpieza de todas las bombas haciendo fluir agua por lo tubos.

Otra ventaja es la reducción de tamaño de la máquina con estos motores. Simplemente es necesario conectar los tubos a las botellas próximas a la máquina y existe mucho menos componentes mecánicos que en la idea anterior.

3.2.2 Control de nuestra aplicación.

La manera en la que se ha planteado controlar la apertura y cierre de los motores de las bombas peristálticas es mediante relés que abren y cierren el circuito de potencia para tensiones mayores. Dichos relés son controlados por las salidas digitales de la placa Arduino. Las salidas digitales, al principio del programa están asociadas a las distintas bebidas en las que estará conectadas las bombas peristálticas de cada salida. De esa manera es más intuitivo a la hora de depurar saber la bomba peristáltica de qué bebida es la que estás modificando.

```
#define RON 3
#define COLA 4
#define VODKA 5
#define LIMON 6
#define GIN 7
#define TONICA 8
#define BUZZER 12
```

Ilustración 13 Asignaciones de salidas digitales a distintas bebidas en el programa

Dependiendo del número de bebidas y de qué bebidas son las escogidas, la salida digital que activa el relé correspondiente es distinto cada vez.

Por ejemplo, si el usuario ha pedido un *cocktail* de tres bebidas; la primera de ron, la segunda de cola y la tercera de limón; la primera salida digital que se activará en el momento que se detecte el peso del vaso sería la denominada *RON* (en este caso la salida 3). Con la salida digital activada la bomba peristáltica entraría en funcionamiento, haciendo que el ron fluyera de la botella al vaso, hasta que el sensor FSR detectara que la cantidad determinada ya ha sido alcanzada. Tras pasar la salida digital 3 a *LOW*, el programa taraería el peso del sensor, y activaría la salida *COLA* (la 4 en este caso) y funcionaría de manera análoga. Con la tercera bebida pasaría igual, indicando el final del proceso con una señal sonora y desactivando el bit que indica que el proceso de llenado está en marcha.

La manera exacta de cómo funciona el programa al completo se detallará más adelante.

4. Interfaz de usuario y comunicación con microcontrolador.

Uno de los retos que suponía utilizar una interfaz de usuario desde el móvil era la comunicación con el microcontrolador. La comunicación se realiza a través de un módulo Bluetooth HC05 como el que se muestra en la imagen.

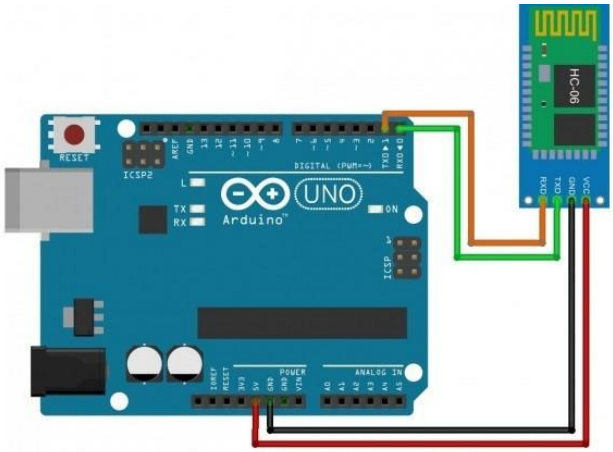

Ilustración 14 Conexionado módulo bluetooth HC05 con la placa Arduino.

Los pines de RX y TX del módulo están conectado a los pines de TX y RX de la placa respectivamente. Al principio, se pretendió no utilizar esos pines de la placa para la recepción y envío de datos al módulo bluetooth y tenerlos para la comunicación serial con el ordenador. Para ello se utilizaba la biblioteca *SoftwareSerial.h*, y podemos definir dos pines de salidas digital como los RX y TX del módulo. Sin embargo, eran dos pines que perdíamos para poder añadir salidas digitales y, una vez probado el programa de la máquina, la comunicación será exclusivamente vía Bluetooth con la interfaz del móvil.

Una vez establecido el método de comunicación con la placa Arduino, corresponde analizar qué tipo de caracteres se enviaban a la placa para controlar el proceso de llenado.

4.1 Interfaz de usuario

Como se ha mencionado antes, para realizar la interfaz del usuario en el móvil, se ha utilizado la plataforma AppInventor desarrollada por el MIT. La aplicación se ha diseñado pensando en la comodidad para el usuario. Es decir, la activación de distintos botones, aparte de enviar tramas de información a la placa, hacen que otras opciones se hagan visibles o invisibles.

Lo primero que se ve al acceder a la aplicación es la siguiente pantalla:

Ilustración 15 Menú principal de la interfaz.

El botón de nueva bebida permite al usuario acceder a todas las opciones de llenado del programa. El botón de parar anula cualquier orden de llenado y para el proceso si es que se está realizando en ese momento. Si el bit de llenado no está activo, el botón no tiene ningún efecto. El botón de conectar abre un menú de todos los dispositivos bluetooth vinculados al móvil.

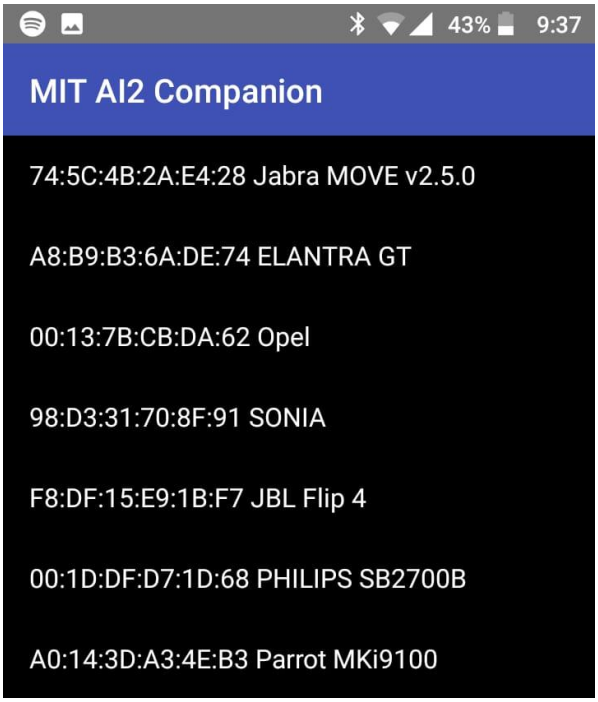

Ilustración 16 Menú desplegable dispositivos bluetooth vinculados al móvil.

Si la conexión ha sido exitosa, el botón de conectar aparece en verde y el texto de *Conexión realizada* aparece visible.

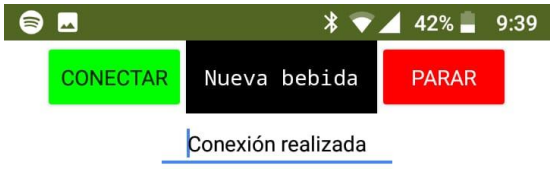

Ilustración 17 Conexión realizada.

Con la conexión establecida, si accedemos a la opción de nueva bebida se despliega la opción de escoger cuantas bebidas desea el usuario.

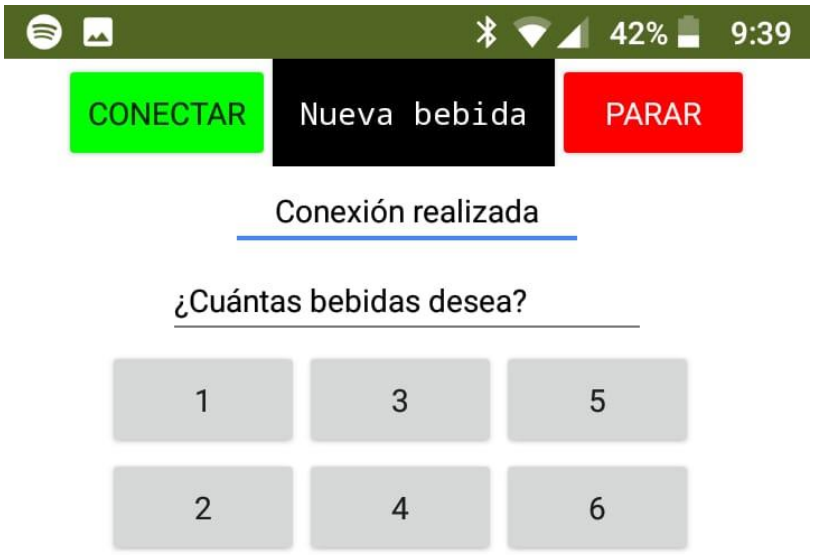

Ilustración 18 Elección número de bebidas.

Para seguir con el ejemplo, en este caso se ha optado por escoger 3 bebidas. Por lo tanto en la aplicación se hacen visibles las opciones de qué tipo de bebida se desean para esas tres bebidas.

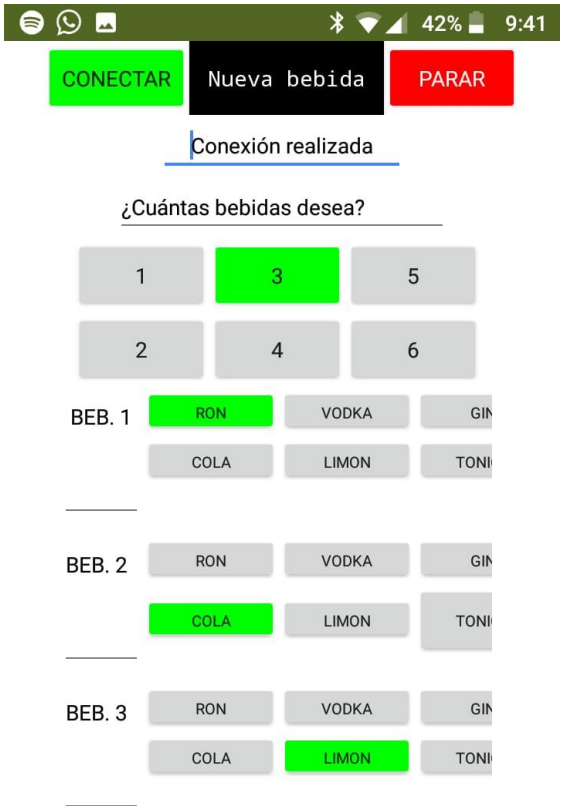

Ilustración 19 Elección tipo de bebidas.

Una vez se han escogido los tres tipos de bebidas, se procede a escoger las cantidades para cada bebida. Otra vez, solo estará disponible la elección de cantidades para tres bebidas, ya que son las que se han escogido previamente.

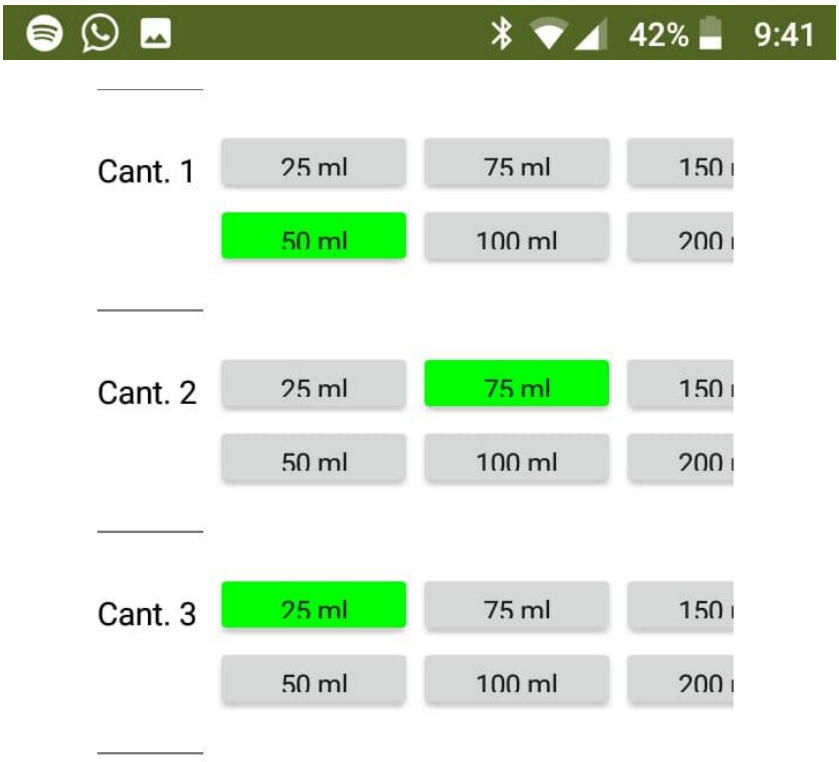

Ilustración 20 Elección de cantidades.

Y finalmente, una vez todas las opciones se han configurado correctamente, el usuario ya puede presionar al botón *EMPEZAR*, que activa el bit de llenado y empieza el proceso para obtener nuestra bebida mezclada. Si el usuario quisiera una nueva bebida, debería esperar a que acabara la que se está llenando y retirar el vaso, o bien presionar el botón de *PARAR* y comenzar con el proceso de elección de bebida otra vez.

Al presionar el botón *EMPEZAR*, las opciones de configuración mostradas previamente vuelven a ser invisibles para el usuario.

Ilustración 21 Llenado en proceso.

4.2 Programación pantalla.

Descrita la interfaz del usuario, se procede a explicar como se ha realizado la programación de la pantalla. La plataforma AppInventor permite una programación por bloques predefinidos. A simple vista puede parecer muy simple y de poca elaboración, pero permite hacer un sinfín de variaciones a nuestras aplicaciones como se muestra a continuación.

Al iniciar la aplicación, se configura que botones o sectores de la pantalla están visible y cuáles no.

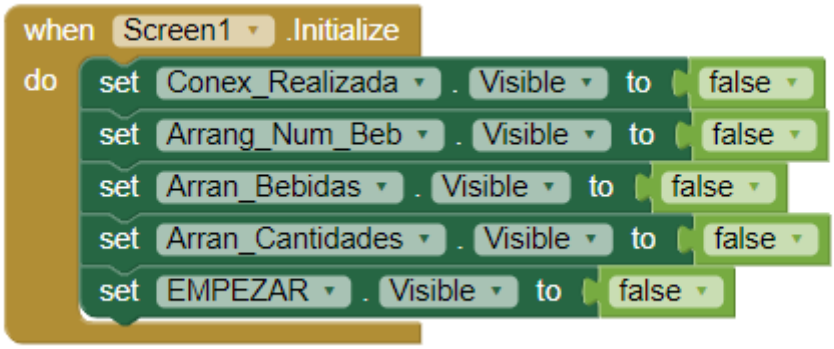

Ilustración 22 Configuración de visibilidad menú principal.

Estos distintos sectores de la pantalla se van mostrando visibles o no dependiendo de la configuración que haya realizado el usuario.

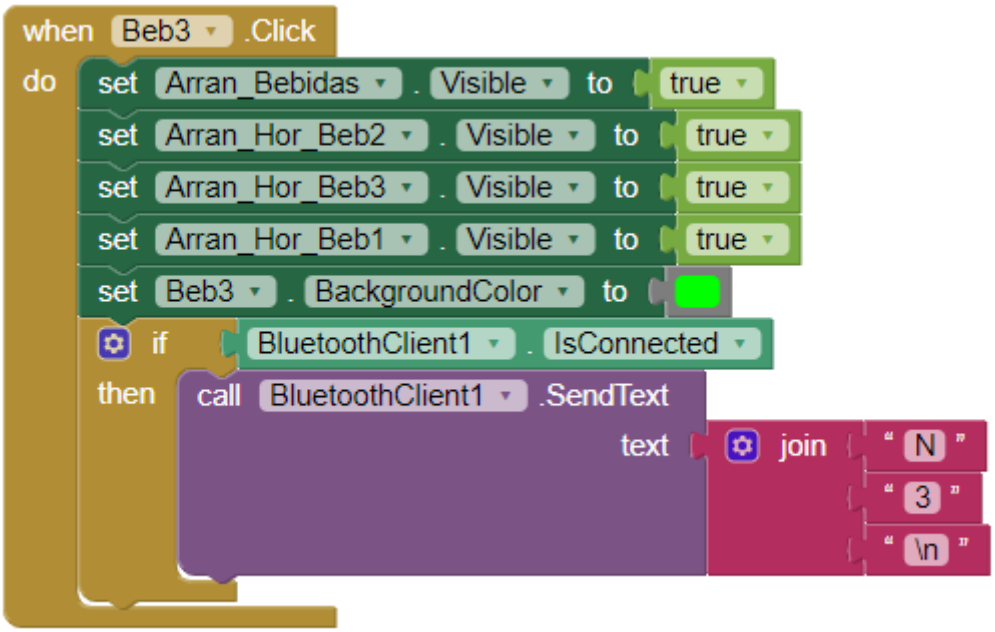

Ilustración 22 Acciones realizada al escoger tres bebidas.

En este caso se puede observar cómo al escoger 3 bebidas, se harán visible solamente las 3 secciones que corresponden a la elección de bebidas para esas 3 bebidas escogidas.

4.3 Envío de caracteres

A continuación, se va a mostrar que acciones realiza la pantalla y qué información envía a la placa de Arduino para asegurarnos que el código en la placa Arduino reconoce las acciones que el usuario desea realizar.

Lo primero es asegurarnos que la conexión Bluetooth con el módulo integrado en la placa. Como se ha mencionado anteriormente, si la conexión es correcta saldrá un mensaje en pantalla.

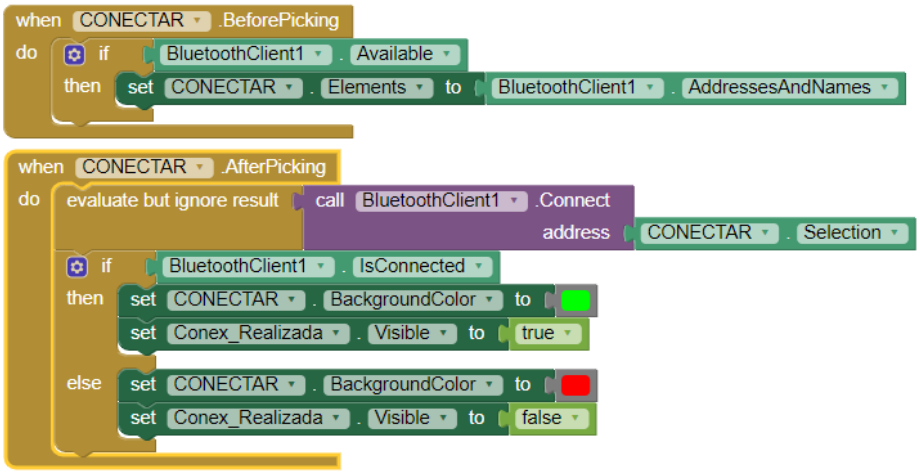

Ilustración 23 Muestra de cómo realiza la pantalla la conexión con el microcontrolador.

Se puede observar que el botón de conectar es de tipo *List*, por lo que al presionarlo abre la lista de *AdressesAndNames* de dispositivos Bluetooth vinculados al móvil. Una vez escogido un dispositivo, comprueba que la conexión se ha hecho de manera correcta y despliega la información indicándolo al usuario.

Para enviar la información correspondiente al número de botellas que se desean mezclar en la bebida se envía el carácter *N* seguido de un número del 1-6 indicando el número de botellas. La razón exacta de esto se explica en el análisis del código que viene a continuación.

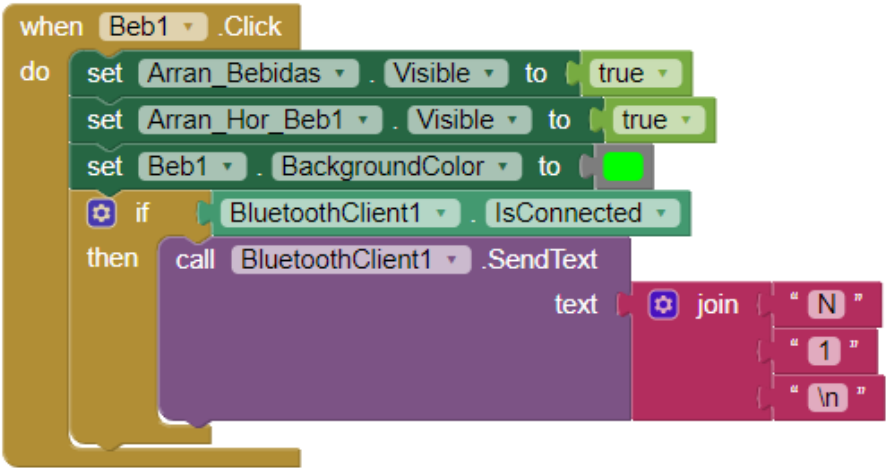

Ilustración 24 Envío de caracteres al escoger número de bebidas.

De la misma manera, para escoger que tipo de bebida queremos mezclar se envía el carácter que indica que número de bebida vamos a escoger (de la U a la Z) seguidos del número del 1 al 6 asociada al tipo de bebida en el código del Arduino.

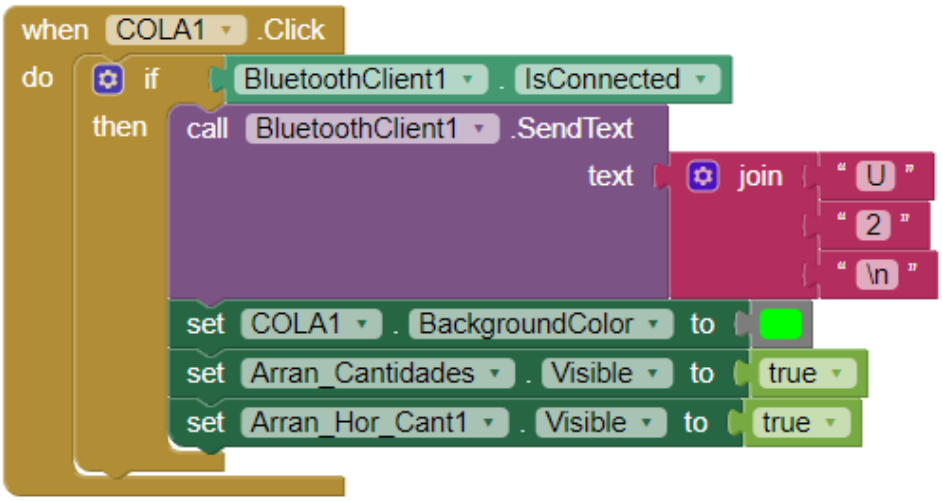

Ilustración 25 Envío de caracteres al escoger tipo de bebida.

Para esocger la cantidad a mezclar también se hace de manera similar. Se envía el carácter de la A a la F seguidos de un número del 1-6 que significa a qué número de bebida a mezclar se refiere el usuario (la letra) y que cantidad es la que se va a mezclar (el número está asociado con una cantidad de líquido).

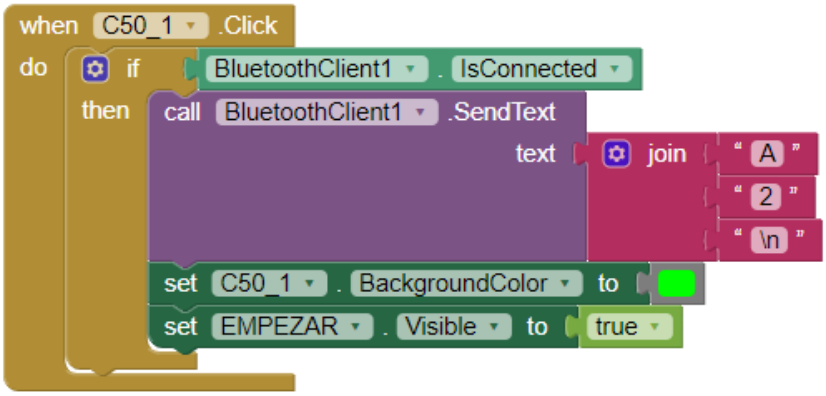

Ilustración 26 Envío de caracteres al escoger cantidad a mezclar.

Finalmente para empezar un nuevo proceso de llenado o parar el existente, se envía el carácter *H* o *L* respectivamente. Además tambien se realizan acciones de ocultar botones o de poner en verde el botón de empezar para que el usuario sepa si el proceso está funcionando correctamente o no.

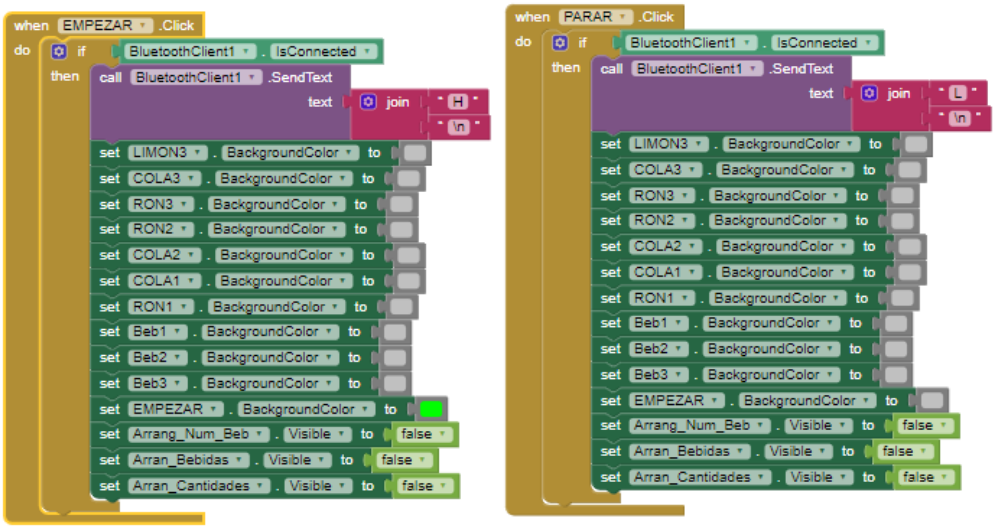

Ilustración 27 Acciones realizadas para EMPEZAR o PARAR.

5. Análisis del código.

Una vez explicado los componentes que forman el control de la máquina y de como se ha diseñado y comunica la interfaz con el microcontrolador, falta por detallar como se ha estructurado el código del programa de Arduino.

La intención a la hora de realizar el código es de que éste se realizara en cada ciclo de arriba abajo. Es decir, como hay muchas interacciones que dependen del ser humano (poner vaso) o de la velocidad de flujo de las bombas peristálticas, no interesaba que el código se quedara '*enganchado*' en un *for* o *while* a la espera de que las condiciones se cumplieran. Por ello se ha decidido tratar el código como si de un autómata fuera, utilizando *if else* y usando *flags* para activar o desactivar dependiendo de si no se desea entrar en una sección del programa o sí.

A continuación, se ofrece prácticamente todo el código totalmente comentado. Las partes más importantes se detallarán más detenidamente.

```
// Sketch TFM 2 con FSR versión 2. Sensor de peso controla salidas digitales.
// LCD de Sonia 4 pines. Separado en funciones. 6 posibles relés. Se puede mezclar todas las bebidas que quieras.
// Interfaz de control y monitorización (APPINVENTOR BLUETOOTH)
// A0 entrada sensor analógica
// Salidas digitales para controlar relés bomba peristática 1 sentido.

// ==========================================
//            CROCKTELERA 2.0
// ==========================================

#define tiem_ciclo 70 // Tiempo de cada ciclo de scan del micro controlador 70 ms

// Librerías
#include <Adafruit_GFX.h> // Pantalla
#include <Adafruit_SSD1306.h> // Pantalla
#include <SoftwareSerial.h> //Librería que permite establecer comunicación serie en otros pins aparte del 0 y 1
// Pantalla variables
#define OLED_RESET A4 // Display Pantalla
Adafruit_SSD1306 display(OLED_RESET);
```

Al primcipio del ccódigo se hace un breve resumen y se declaran las librerias y los *#define* que sustituyen el nombre de las bebidas por el de las salidas a las que están conectadas las bombas peristálticas.

```
#define RON 3  // Definiciones de las salidas digitales. Asocia las salidas a la bebida a la que estan conectadas mediante relé y bomba peristáltica.
#define COLA 4
#define VODKA 5
#define LIMON 6
#define GIN 7
#define TONICA 8
#define BUZZER 12

//SoftwareSerial BT1(10,11);

//BEBIDAS // Variables que determinan que tipo de bebida será la primera, segunda.. en llenarse. Un barman jamás echaría priemro la tónica y luego la ginebra.
int bebida1;
int bebida2;
int bebida3;
int bebida4;
int bebida5;
int bebida6;

//FSR variable
int FSR, FSR_tare, FSR_g;
```

Se declaran todas las variables.

```
// Variables cantidades // Cantidad que el usuario desea para cada bebida.
int cantidad1=0;
int cantidad2=0;
int cantidad3=0;
int cantidad4=0;
int cantidad5=0;
int cantidad6=0;
int peso_vaso=100;

int num_beb_or; // Número de bebidas que el usuario desea
int num_beb; // Número de bebidas que la máquina ya ha terminado de llenar
char mode; //
int nbeb;
int candis;

//Struct para función calc_peso. //
struct CROKI {
  int tara; // valor que repondrá la variable tara del programa
  bool modo; // HL
  bool rflag; // flag para saber si hacer tara o no cuando se entre a la función
  bool modo2; // GL
};


// Motor variables
bool HL=false;
bool GL=false;
bool taraflag=true;

//Zumbador
bool mute=false;
long tiempo_ejecucion, loop_timer;

CROKI output_funcion = {FSR_tare,HL,taraflag};
```

Las variables del motor son los distintos *flags* que se activan o desactivan dependiendo de donde esté el programa en el proceso de llenado para ir entrar a una zona del código o no.

```
//////////////////////////////////////////////////////////// SETUP ////////////////////////////////////

void setup() {
  // put your setup code here, to run once:
// Inicializaciones
Serial.begin(9600); //Velocidad del puerto serie
//BT1.begin(9600);  //Velocidad del puerto del módulo Bluetooth

// I/O pines
pinMode(A0,INPUT); // Entrada del sensor FSR
pinMode(RON,OUTPUT); // Relé 1
pinMode(COLA,OUTPUT); // Relé n2
pinMode(VODKA,OUTPUT); // Relé 3
pinMode(LIMON,OUTPUT); // Relé n4
pinMode(GIN,OUTPUT); // Relé 5
pinMode(TONICA,OUTPUT); // Relé n6
pinMode(11,INPUT); // Botón de tara manual. Conectada a +5V. Valor predeterminado LOW.
pinMode(2,OUTPUT); // LED indica programa está en HL;
pinMode(BUZZER,OUTPUT); // ZUmbador indica final del llenado de vaso. Se puede tener desactivado

// Setup FSR
FSR_tare=analogRead(A0); // Inicializa el sistema con peso de 0


digitalWrite(RON,LOW);
digitalWrite(COLA,LOW);
digitalWrite(VODKA,LOW);
digitalWrite(LIMON,LOW);
digitalWrite(GIN,LOW);
digitalWrite(TONICA,LOW);
digitalWrite(2,LOW);
digitalWrite(BUZZER,LOW);

// Serial.println("Introducir H para modo High Level y L para Low Level Mode.");
// Serial.println("Introducir 1 o 2 para escoger botella.");


// Setup Display
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3C (for the 128x32)//
// init done
// Clear the buffer.
display.clearDisplay();


loop_timer = millis(); //Empieza a contar el tiempo de ciclo para la primera vez
}


//////////////////////////////////////////////// LOOP ////////////////////////////////////////////////

void loop() {
  // put your main code here, to run repeatedly:
  // Eleccion de modos

/// ACTIVAR/DESACTIVAR

if (Serial.available()>0) { // Cuando se detecta bytes enviados desde la aplicación movil evalua la
  //información y relaiza distintas acciones
  mode=Serial.read(); //
  if (mode=='H') { // Si se envia una H es el sistema de llenado empieza a funcionar.
    //Antes de empezar se debe de haber determinado previamente toda la información necesaria de como se ha de realizar.
    HL=true;
    num_beb=0;
    Serial.print("Modo funcionamiento: ");
    Serial.println(mode);
    Serial.print("Cantidad de agua: ");
    Serial.println(FSR_g);
  }
```

En el momento que se detecta que bytes esperando a ser leidos en el buffer de comunicación

serial el primer carácter que hay almacenado se guarda como la variable *mode* y, dependiendo de el carácter que sea, el programa realizará una acción u otra.

Si el carácter es una *H*, se activa el flag que activa el proceso de llenado y se reseta la variable que contabiliza las bebidas mezcladas en cada proceso de llenado a 0. Si por el contrario, recibe una *L*, el proceso se para porque el flag de de activación de llenado se desactiva y las salidas digitales que activan las bombas se ponen a nivel bajo.

```
else if (mode=='L') { // EL proceso de llenado se para si le llega una L en cualquier momento del llenado.
  HL=false;
  GL=false;
  Serial.print("Modo funcionamiento: ");
  Serial.println(mode);
  Serial.print("Cantidad de agua: ");
  Serial.println(FSR_g);
}

// número botellas
else if (mode == 'N' && not(HL)) { // Cuando le llega una N y el proceso de llenado no está activo,
  //el programa esperará al siguiente carácter (1-6) para saber cuantas bebidas va a tener que mezclar
  nbeb=Serial.read();
   if (nbeb>48 && nbeb<55) { // Utiliza el valor de tabla ASCII y lo evalúa como el número que
    //representa tras hacer la resta
    num_beb_or=nbeb-48;
    Serial.print("Num.bebidad: ");
    Serial.println(num_beb_or);
   }
}
```

El envío del caracter N y el tener el flag *HL* a nive bajo (que no esté el proceso de llenado activo) hace que el sistema compruebe el siguiente carácter para determinar el número de bebidas que se van a mezclar cuando reciba el programa una *H.*

```
// cantidades de cada botella
else if (mode == 'A' && not(HL)) { // Una vez determinado el número de distintas bebidas que se van mezclar.
  //La A y número a contincuación indica que cantidad de la primera bebida se desea
  nbeb=Serial.read();
  if (nbeb>48 && nbeb<55) {
    candis=nbeb-48;
    switch (candis) {
      case 1: cantidadl=25;
      break;
      case 2: cantidadl=50;
      break;
      case 3: cantidadl=75;
      break;
      case 4: cantidadl=100;
      break;
      case 5: cantidadl=150;
      break;
      case 6: cantidadl=200;
      break;
    }
    Serial.print("Cantidad bebidal: ");
    Serial.println(cantidadl);
  }
}
```

Los caracteres A-F determinan que cantidad se va a mezclar para cada bebida. EL carácter A modifica la cantidad de la primera bebida a llenar, la B la segunda, y así sucesivamente. Si se han escogido tres bebidas a mezclar, la propio interfaz no permitirá enviar caracteres de la D-F.

```
// elección de bebidas
else if (mode == 'U' && not(HL)) { // Con los carácteres de la U a la Z el la aplicación
//determina que tipos de bebidas quiere el usuario por orden.
// La letra U precedida de un número del 1 al 6 significa que bebida se desea llenar primero.
  nbeb=Serial.read();
  if (nbeb>48 && nbeb<55) {
    candis=nbeb-48;
    switch (candis) {
      case 1: bebida1=RON;
      break;
      case 2: bebida1=COLA;
      break;
      case 3: bebida1=VODKA;
      break;
      case 4: bebida1=LIMON;
      break;
      case 5: bebida1=GIN;
      break;
      case 6: bebida1=TONICA;
      break;
    }
    Serial.print("Tipo bebida1: ");
    Serial.println(bebida1);
  }
}
```

Los caracteres de la U-Z funcionan de manera similiar. Sin embargo, estos caracteres modifican el tipo de bebida que se va a servir primero, segundo… Si el número de bebidas escogido es 3, tampoco se podrían recibir las letras X-Z.

Para modificar las variables de *bebida1-6* y *cantidad1-6*, el *flag* HL no ha de estar activo.

```
// Restar la tara y evitar números negativos
FSR=analogRead(A0)-FSR_tare;
if  (FSR<0) FSR=0;

FSR_g=map(FSR,0,1023,0,1186); // INTERPOLACIÓN EXCEL

// Func. de motor.
if (HL) { // Modo botella detectada
  digitalWrite(2,HIGH); // LED indica proceso de llenado en funcionamiento
  // Esperar a detectar vaso
  if (FSR_g> peso_vaso) GL=true; // inicio abrir motor tras reset. Se inicia a detectar peso de
  //vaso para activar GL y empezar realmente proceso llenado.
```

Una vez se activa el bit *HL,* hasta que no se detecte el peso del vaso no se activara el segundo *flag GL* donde ya se activan las salidas digitales y empieza a fluir el líquido.

```
if (GL) {
  if (num_beb_or==num_beb) { // Modo llenado se desactiva cuando contabiliza que se
    //han llenado todas las bebidas. Se activa buzzer
    HL = LOW;
    GL = LOW;
    if (not(mute)) digitalWrite(BUZZER,HIGH); //Quitar si molesta con mute;
  }
  if (num_beb_or-num_beb==num_beb_or) { // Proceso de llenado primera bebida.
    //Al acabar de llenarse la variable num_beb ++
    output_funcion=calc_peso(cantidad1,taraflag,bebida1);
    FSR_tare=output_funcion.tara;
    HL=output_funcion.modo;
    taraflag=output_funcion.rflag;
    GL=output_funcion.modo2;
  }
```

En el momennto en el que el número de bebidas contabilizadas en el proceso de llenado iguala el número de bebidas que el usuario quiere mezclar el proceso de llenado se acaba desactivando los flags HL y GL. Si no se ha igualado ese número, compara en que número de bebida está para introducir como argumentos en la función de *calc_peso()* el tipo de bebida y la cantidad a llenar correspondiente.

```
  else if (num_beb_or-num_beb==num_beb_or-2) { // Llenado 3
    output_funcion=calc_peso(cantidad3,taraflag,bebida3);
    FSR_tare=output_funcion.tara;
    HL=output_funcion.modo;
    taraflag=output_funcion.rflag;
    GL=output_funcion.modo2;
  }
  else if (num_beb_or-num_beb==num_beb_or-3) { // Llenado 4
    output_funcion=calc_peso(cantidad4,taraflag,bebida4);
    FSR_tare=output_funcion.tara;
    HL=output_funcion.modo;
    taraflag=output_funcion.rflag;
    GL=output_funcion.modo2;
  }
  else if (num_beb_or-num_beb==num_beb_or-4) { // 5
    output_funcion=calc_peso(cantidad5,taraflag,bebida5);
    FSR_tare=output_funcion.tara;
    HL=output_funcion.modo;
    taraflag=output_funcion.rflag;
    GL=output_funcion.modo2;
  }
```

```
    else if (num_beb_or-num_beb==num_beb_or-5) { // 6
      output_funcion=calc_peso(cantidad6,taraflag,bebida6);
      FSR_tare=output_funcion.tara;
      HL=output_funcion.modo;
      taraflag=output_funcion.rflag;
      GL=output_funcion.modo2;
    }
  }

}
else { // Si proceso de llenado no está activo las salidas están siempre en LOW
  digitalWrite(bebida1,LOW);
  digitalWrite(bebida2,LOW);
  digitalWrite(bebida3,LOW);
  digitalWrite(bebida4,LOW);
  digitalWrite(bebida5,LOW);
  digitalWrite(bebida6,LOW);
  digitalWrite(2,LOW);
}
```

Si el flag de *HL* no está activado las salidas digitales conectadas a los relés se desactivan automáticamente.

```
// PANTALLA //

// Primera linea
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.print("VOLUMEN AGUA");

//Segunda linea
display.setTextSize(2);
display.setTextColor(WHITE); // 'inverted' text
display.setCursor(0,12);
display.print(FSR_g);
display.print(" ml");

//Mostrar en pantalla el FSR_g actual del sistema.
display.display();
display.clearDisplay();


  while (millis() - loop_timer < tiem_ciclo);  // Control tiempo de muestreo
  loop_timer = millis();

}
```

```
//////////////////////////////////////////// FUNCIONES ////////////////////////////////////////////////

struct CROKI calc_peso (int cantidad, bool resetflag, int rele) {
  if (resetflag==HIGH) { // Cada vez que se llama a la función se hace un reset para empezar
    //proceso de llenado de cada bebida desde 0
    delay(1000);
    FSR_tare=fun_tara();
    resetflag=LOW; // Con este flag a LOW ya no entra a esta parte de la función
    GL=HIGH;
    HL=HIGH;
  }
  else if (FSR_g<cantidad){
    digitalWrite(rele,HIGH); // La salida digital correspondiente esta activa y por lo
    //tanto se lleba el vaso hasta llegar a la cantidad determinada.
    HL=HIGH;
    }
  else {
      digitalWrite(rele,LOW); // Desactiva salida digital
     resetflag=HIGH; // la proxima que se llame a esta función irá a la parte de la tara porque
     //el flag esta HIGH
     num_beb++; // Aumneta el contador de bebidas llenadas para este proceso de llenado.
    }
  CROKI output_funcion = {FSR_tare,HL,resetflag,GL};  // Devuelve la variable tipo struct
  //declarada al principio del script.
  return output_funcion;
}


int fun_tara() { // Función tara
  FSR_tare=analogRead(A0);
  return FSR_tare;
}
```

Aquí se muestran las definiciones de las dos funciones usadas en el programa. La de *calc_peso()* utiliza *resetflag* para hacer una tara la primera vez que se llama a esta función en un mismo proceso de llenado. Cuando se ha conseguido la cantidad deseada la salida digital cambia a nivel *LOW*, se vuelve a poner *resetflag* como *HIGH* para que se haga tara la próxima vez que se llame a la función y el contador de bebidas mezcladas sube en una unidad.

6. *Presupuesto del desarrollo del proyecto*

Los costes estimados del desarrollo de la solución de ingeniería y montaje del control de la coctelera automática se desglosan a continuación.

6.1 Coste en Componentes Electrónicos

COMPONENTE ELECTRÓNICO	**PRECIO TOTAL (€)**
Placa Arduino UNO	19.95
Bomba peristáltica (x6)	66.60
Módulo Relé Arduino (x6)	9.18
Display LCD	2.95
Sensor FSR	12.50
Módulo Bluetooth	6.95
Fuente alimentación dual	28
TOTAL	**146.13€**
TOTAL (IVA DEDUCIDO) (/1.21)	**120.77**

Tabla 1: Desglose coste componentes electrónicos.

6.2 Coste desarrollo solución

Por otro lado, existe el coste fijo del desarrollo de la solución. Cogiendo el suelo medio de un ingeniero en España y estimando un tiempo de desarrollo de 100 horas hasta lograr una solución final y depurada:

$$Sueldo\ medio\ ingeniería\ en\ España = \frac{26€}{hora}$$

$$100\ \times 26 = 2600\ €$$

6.3 Precio de Venta

Suponiendo que un primer encargado se encargaron los circuitos y placas para unas 200 cocteleras automáticas el coste variable en la compra de componentes sin IVA y coste de desarrollo fijo sumaría:

$$(120.77 * 200) + 2600 = 26754\ €$$

Por lo tanto, el coste del circuito de control de cada coctelera sería:

$$\frac{26754€}{200} = 133.77\ €$$

Añadiendo un 30% de beneficio por la venta del circuito y programa de cada coctelera y sumando el IVA el precio de venta sería:

$$133.77 * 30\% * 21\% = 210\ €$$

Este precio se ha calculado con los costes unitarios de cada componente electrónico. Si se pidieran en lotes de mayor número, probablemente se podría abaratar el precio de variable de los componentes para cada coctelera. De la misma manera, al aumentar el número de circuitos de control vendidos, el coste fijo se reduciría en comparación con el variable y se podría ajustar algo más el precio venta

7. Conclusión

En este último apartado se resumen todas las conclusiones que he podido sacar tras realizar este trabajo desarrollando la solución a la automatización de la coctelera automática. En ese tiempo, la idea inicial fue variando hasta prácticamente no asemejarse en nada a la del resultado final. Las conclusiones son las siguientes:

- He puesto en práctica varios conocimientos adquiridos en el máster sobre sistemas embebidos o automatización de procesos. El hecho de trabajar durante tanto tiempo en un mismo proyecto, modificar constantemente las ideas iniciales debido a problemas que un principio no se habían valorado, me han hecho darle más importancia a la correcta planificación de un proyecto para encontrar la solución más viable en un futuro.
- El concepto de *Maker* cada vez está más de moda y cualquiera puede encontrar mucha información en Internet sobre como automatizar y desarrollar tus propios procesos más o menos caseros. La ventaja de utilizar Arduino aparte del precio es la cantidad y calidad de muchos tutoriales que hay colgados en la red para ayudar al usuario a hacer sus propios proyectos.
- He aprendido a desarrollar mis pequeñas aplicaciones para el teléfono móvil con la plataforma AppInventor.
- He ampliado mis conocimientos en desarrollar programas de automatización. He mejorado mi método de programación para conseguir un programa robusto, que no se quede colgado cuando ha de interactuar con sistemas que van mucho más lento que el procesado del programa por el microcontrolador.

8. Bibliografía

- **Información sobre FSR** *https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr*
- **Información sobre bombas peristálticas** *https://es.wikipedia.org/wiki/Bomba_perist%C3%A1ltica*
- **Información general sobre Arduino** *http://www.arduino.cc*
- **Información sobre módulo Bluetooth HC05** *https://i1.wp.com/www.electrogeekshop.com/wp-content/uploads/2018/02/1a8e142733a46f4a140162dfc93c3f18.jpeg?fit=840%2C420&ssl=1*
- **Compra Arduino Uno en tienda online de electrónica Bricogeek:** *https://tienda.bricogeek.com*
- **Fabricante de bombas peristálticas Kaomer:** *http://kamoer.com/index.php?lang=en*
- **Compra módulos relé distribuidor Iberobotics:** *https://www.iberobotics.com/*
- **Compra display LCD en distribuidor createc3d:** *www.createc3d.com*
- **Compra sensor FSR en tienda online Bricogeek:** *https://tienda.bricogeek.com*
- **Compra de módulo bluetooth en distribuidor createc3d:** *www.createc3d.com*
- **Proveedor de fuente de alimentación gzkaihui**: *https://es.gzkaihui.com*

Anexo I. Conexionado del Circuito.