Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Sistema electrónico para la manipulación precisa de segmentos líquidos a alta temperatura en lab on chips

Autor: David González Ruiz

Tutor: Francisco Antonio Perdigones Sánchez

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica (UMA-US)

Sistema electrónico para la manipulación precisa de segmentos líquidos a alta temperatura en lab on chips

Autor:
David González Ruiz

Tutor:
Francisco Antonio Perdigones Sánchez
Profesor Titular de Universidad

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Trabajo Fin de Grado: Sistema electrónico para la manipulación precisa de segmentos líquidos a alta temperatura en lab on chips

Autor: David González Ruiz

Tutor: Francisco Antonio Perdigones Sánchez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Sevilla,2023

Agradecimientos

Quisiera agradecer la labor que han tenido todos aquellos docentes que he tenido que se han esforzado siempre por hacer su labor lo mejor posible y transmitir tan bien tanto sus conocimientos como su pasión en los respectivos campos en los que trabajan. Ellos han sido una gran inspiración para mi persona.

Por supuesto, agradecer también el apoyo incansable de mi padre, mi madre y mi hermana, que siempre han sabido aconsejarme, animarme y celebrar conmigo a medida que los años han ido avanzando y, sobretodo, por se un apoyo incondicional. Además, agradecer.también a todos mis amigos tanto mis amigos de la infancia como los que he conocido en este periodo aquí en Sevilla por cumplir un papel esencial también en que yo haya llegado a este punto. A todos vosotros, muchísimas gracias.

Por último, agradecer a Francisco Perdigones Sánchez, por haberme ayudado y guiado de una forma excepcional para conseguir llevar a cabo este proyecto con éxito, proporcionarme sus conocimientos, su perspectiva y los medios necesarios, este proyecto no habría sido posible sin su colaboración.

David González Ruiz

Sevilla, 2023

Resumen

Se presenta el diseño de un sistema capaz de controlar el movimiento de segmentos microfluídicos inestables sometidos a altas temperaturas de forma precisa en lab on chips. Los requisitos exigidos son el uso de una alimentación de, como máximo, 5V, una corriente máxima admisible de 10A, el uso de componentes ópticos como fotodiodos y LEDs discretos (sólo encendidos o apagados, sin mayor regulación de intensidad) y que el coste sea lo más reducido posible. Para cumplir con ello, se hará uso de un sistema de control basado en los componentes ópticos ya mencionados y de un motor paso a paso de corriente continua que se encargará de transmitir el movimiento a través de un émbolo. Todo esto se aplicará concretamente a una muestra que será introducida en el lab on chip mencionado y, posteriormente, manipulado mediante la técnica de PCR (Polymerase Chain Reaction) para amplificar cadenas de ADN y así obtener el resultado correspondiente del análisis.

Abstract

This project shows the design of a system capable of moving unstable high-temperature microfluidic segments with great precision in a lab on chip. Requirements were: using, at maximum, a 5 V power supply, a maximum current of 10 A, use of optic devices as photodiodes and discrete LEDs (only ON/OFF mode) and keeping the cost as low as possible. To meet all these requirements a system controlled by the optics components just mentioned will be used, as well as a stepper motor that will extend its movement to the fluid. All of this will be applied to a sample inyected into the lab on chip being a complete PCR technique (Polymerase Chain Reaction) in order to amplify DNA chains and obtain the result correspondent to its analysis.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

PCB	Printed Circuit Board
LED	Light Emitting Diode
PCR	Polymerase Chain Reaction
LOC	Lab-on-chip
NTC	Negative Temperature Coefficient
BJT	Bipolar Junction Transistor
SEM	Scanning Electron Micrograph
cd	Candela
V	Voltio
ºC	Grado Celsius
A	Amperio
N	Newton

1 Introducción

La frase más emocionante para escuchar en la ciencia,
la que anuncia la mayor cantidad de descubrimientos,
no es: "¡Eureka!", sino: "Eso es gracioso…"

- Isaac Asimov -

En este trabajo se ha diseñado un sistema Lab-on-Chip para el control de segmentos microfluídicos con la idea de aplicarlo a la técnica PCR, que será descrita en este mismo capítulo ya que es esencial, para lo que se ha usado un sistema óptico para controlar el movimiento del fluido y un control de la temperatura para asegurar que el proceso se lleva a cabo correctamente. Además, mencionar que este trabajo surge de un proyecto de investigación de la Junta de Andalucía titulado "Sistema para la amplificación y detección de fragmentos de ADN empleando PCR en Lab-on-chip (PCR-on-a-Chip) (P18-RT-1745)". Este proyecto acabó en diciembre de 2022 y tuvo como solución una totalmente distinta a la que presenta este trabajo. El problema a resolver se presentará en un capítulo posterior, ahora se hará una descripción de los conceptos clave para abordar el presente trabajo.

1.1 Lab-on-a-Chip

1.1.1 Breve historia y descripción general

Los Lab-on-a-Chip, o también conocidos como LOC, son una tecnología que surgió poco antes de los años 80 (aunque sus primeros grandes avances no se dieron hasta finales de dicha década) y que está fuertemente ligada al campo de la microfluídica y de los MEMS (Micro-ElectroMechanical Systems), los cuáles surgieron alrededor de los años 60. Todo esto unido al uso de la litografía permitió desarrollar la tecnología Lab-on-a-Chip de forma que se han podido replicar experimentos o procedimientos que se llevarían a cabo normalmente en un laboratorio convencional, pero a una escala muy pequeña y que facilita enormemente la automatización de los mismos. La escala de la que hablamos puede variar en un rango desde centímetros hasta milímetros cuadrados, en el caso que se presenta en este trabajo el Lab-on-a-Chip usado tiene un área alrededor de los 46 cm^2.

1.1.2 Partes principales

Principalmente se componen por un conjunto de microcanales (de unos pocos micrómetros de radio), por los que se moverán los fluidos empleados en el procedimiento deseado, y también por un grupo de elementos que se encargarán de controlar el movimiento de dichos fluidos a través de dichos microcanales como pueden ser válvulas, émbolos, bombas, así como el uso de tanto el campo eléctrico como el magnético.

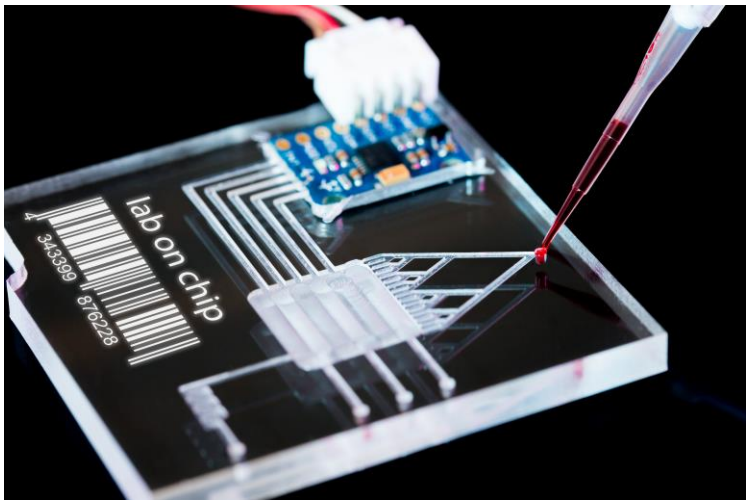

Figura 1-1 Dispositivo Lab-on-a-Chip.

1.1.3 Materiales

En cuanto a los materiales usados para la fabricación de los Lab-on-a-Chip los más comunes son los siguientes:

- PDMS (polidimetilsiloxano). Es un material flexible, transparente y bastante barato lo cual favorece que sea usado de cara al prototipado.
- Polímeros termoplásticos. También gozan de la cualidad de ser transparentes, pero a su vez interfieren menos químicamente hablando que el PDMS, aunque no tan barato.
- Cristal. también es otro gran candidato a la hora de la fabricación de un Lab-on-a-Chip al ser transparente y no interferir químicamente en el proceso; por otra parte.
- Silicio. fue el material que primero se usó para esta tecnología, pero su elevado coste en comparación con otros materiales y su opacidad hizo que fuera rápidamente reemplazado en este campo.

1.1.4 Aplicaciones

Algunos de los procedimientos o aplicaciones más comunes para los cuáles se hace uso de los Lab-on-a-Chip pueden ser, entre otros, los mencionados a continuación:

- Proteómica[1]. Esta tecnología ha permitido reducir significativamente el tiempo necesario para realizar análisis en este campo desde varias horas a unos pocos minutos. También ha facilitado la cristalización de proteínas ya que permite a los científicos controlar un gran número (o todos) de parámetros que intervienen en dicho proceso.
- Biología molecular. Uno de los campos que más favorecidos se han visto con el desarrollo de esta tecnología al ser usada para ciertos análisis relacionados con el ADN (también aplicados al ARN) como son su detección, secuenciación o amplificación. Siendo más concretos, una de las técnicas más reproducidas hoy en día es la PCR, caso de estudio del presente trabajo. Por otra parte, también se emplea para la secuenciación del genoma humano a una velocidad considerablemente mayor que usando los anteriormente eran medios tradicionales.
- Química. Muchas reacciones químicas requieren de ciclos de calentamiento o enfriamiento muy rápidos que son más eficientes a escala microscópica, por lo que la investigación en el campo de la química también se ha visto favorecida al usar LOCs como micro-reactores químicos.

[1] Estudia la función y estructura de las proteínas. También se puede encontrar en la literatura con el nombre de proteinómica.

1.2 Flujo continuo y flujo segmentado

A la hora de emplear LOCs para cualquier procedimiento una de las principales decisiones a tomar antes de comenzar con su diseño ha de ser qué tipo de flujo emplear, es decir, tener en cuenta los requerimientos del procedimiento a reproducir, así como los medios disponibles. Una vez se han tenido en cuenta estos factores es el momento de elegir si se diseñará de cara a flujo continuo o a flujo segmentado (discreto).

1.2.1 Flujo continuo

Esta selección, a parte de ser en muchos casos más sencilla a la hora de implementar, beneficia ciertos procedimientos como pueden ser la centrifugación de fluidos entre otros, sin embargo, también presenta varias desventajas como pueden ser la dispersión de Taylor[2] o la contaminación cruzada de los fluidos usados.

1.2.2 Flujo segmentado

Esta opción, mucho más reciente en cuanto a aplicación y uso que su contraparte continua, solventa muchas de las desventajas que tiene el flujo continuo al hacer que el flujo esté formado por pequeñas gotas (volúmenes de nanolitros a picolitros) dentro de un flujo continuo que es inmiscible. A pesar de sus ventajas, este sistema tiene una complejidad mucho mayor que el continuo de cara a la implementación. La ventaja del segmented flow líquido/gas que nosotros usamos, se basa en que los reactivos biológicos son caros, y con este tipo de segmented fow se minimiza la cantidad, aunque después haya que lidiar con las expansiones de los gases (que es lo que nos pasa)

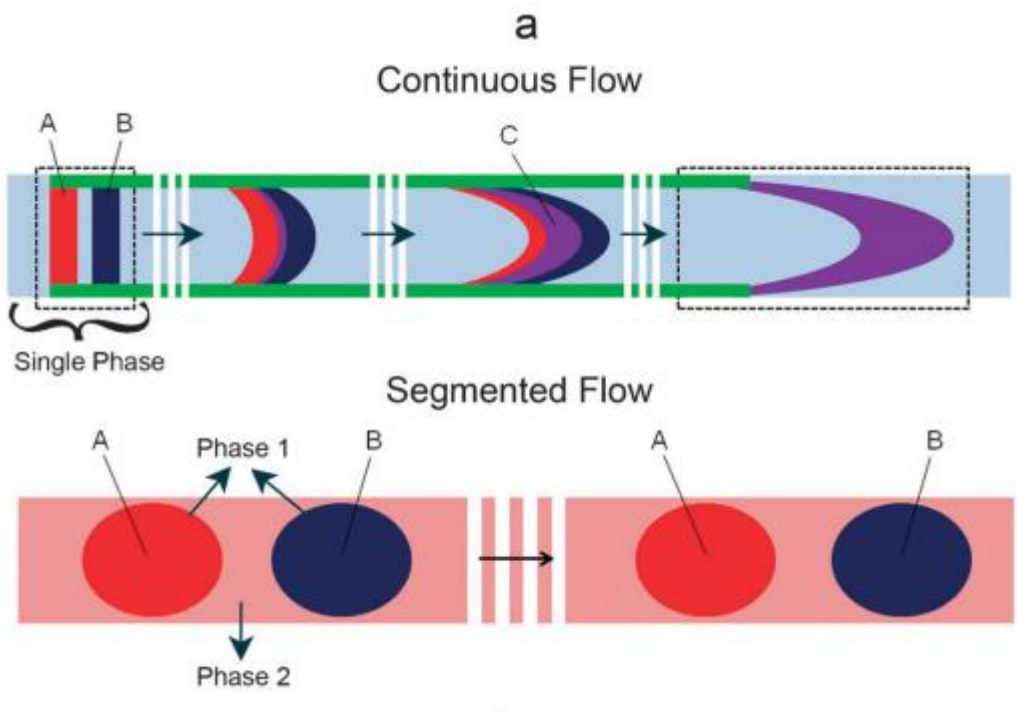

Figura 1-2 Flujo continuo (ilustración superior) y flujo segmentado (ilustración inferior).

[2] Efecto de la mecánica de fluidos por el cuál el flujo inducido por una fuerza (flujo cortante) aumenta la efectividad de la difusión turbulenta del fluido.

1.3 Procedimiento PCR

Una PCR (siglas de Polymerase Chain Reaction, en español Reacción en Cadena de la Polimerasa) es una técnica de laboratorio que fue desarrollada en los años 80. Esta técnica permite replicar material genético hasta obtener millones o incluso miles de millones de copias del mismo, haciendo que sea posible analizarlo con gran fiabilidad, generalmente para la detección de algún virus.

Esta técnica consiste en convertir el ARN (ácido ribonucleico) en ADN (ácido desoxirribonucleico) mediante la enzima ADN polimerasa (que funciona como catalizador del proceso) que permite amplificar el material genético, cosa que se consigue mediante varios ciclos con cambios periódicos de temperatura. Esta parte concreta es el objetivo del sistema diseñado en el presente trabajo, conseguir realizar esa reacción en cadena que provoca la polimerasa dentro del Lab-on-Chip.

Una vez esto se ha realizado, el siguiente paso es analizar todo el material genético amplificado en busca de los componentes objetivos para los que se ha realizado el procedimiento, como podría ser por ejemplo ciertos genes del SARS-Cov-2, y finalmente obtener el diagnóstico correspondiente. Esta última fase, es decir, el análisis, no entra dentro del objetivo para el cuál se ha diseñado el sistema que se presenta en este trabajo.

Volviendo a la parte de la reacción en cadena de la polimerasa, se van a especificar ciertos valores de temperatura que son necesarios para que el procedimiento de un ciclo se lleve a cabo correctamente. La muestra inicialmente ha de calentarse hasta una temperatura cercana a los 95ºC para conseguir la desnaturalización del ADN, posteriormente bajar hasta alrededor de unos 45ºC para la hibridación (o alineamiento), tras esto subir la temperatura de nuevo hasta unos 65ºC para la elongación (también llamada extensión) y, finalmente, repetir tantas veces como ciclos se quieran, teniendo en cuenta que la amplificación de la muestra tiene una relación exponencial respecto al número de ciclos (para 10 ciclos se amplifica por un factor de mil, para 20 por un factor de un millón y para 30 por un factor alrededor de un billón).

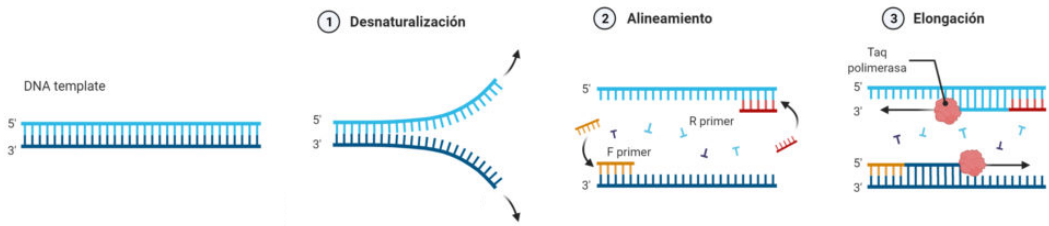

Figura 1-3. Representación de un ciclo de PCR. Fuente:

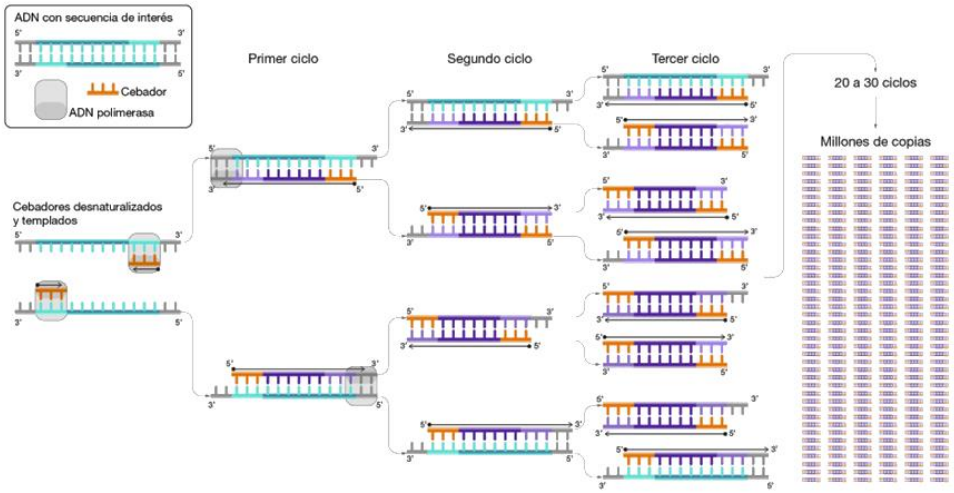

Figura 1-4. Ilustración del carácter exponencial respecto al número de ciclos del proceso PRC. Fuente:

2 OBJETIVOS

2.1 Problema a resolver

El presente trabajo trata de solucionar un problema que surge con el movimiento del fluido, volviéndose éste inestable, y que está intrínsecamente relacionado con la temperatura y la posición del fluido dentro del LOC. En la versión "previa" del sistema que se emplea en este trabajo, en la que ni el movimiento del fluido ni la temperatura eran controlados surgía un problema al sobrepasar el fluido un punto concreto de su recorrido. En ese momento, el aire entraba en contacto con la parte de alta temperatura (alrededor de 95ºC) y se expandía rápidamente, empujando y expulsando de forma violenta el fluido.

Además, durante el desarrollo de este trabajo se ha observado que este problema no sólo ocurre en la parte de alta temperatura de LOC, sino que también ocurre en la parte de baja temperatura. Esto era un fenómeno que no se había contemplado previamente, pero se ha concluído que el aire externo que se introduce en el LOC, para empujar y mover el fluido hacia delante, está a temperatura ambiente y cuando se inyecta entra en contacto con la parte que está a 65ºC súbitamente y esto también causa una expansión del mismo. Esta nueva perturbación, aunque de menor medida que la que ya conocíamos, nos da una explicación mucho mejor y más completa sobre la inestabilidad del fluido y su expulsión sea tan violenta ya que a medida que se va introduciendo más aire para mover el líquido hacia delante más aire se expande y más se empuja al líquido como resultado.

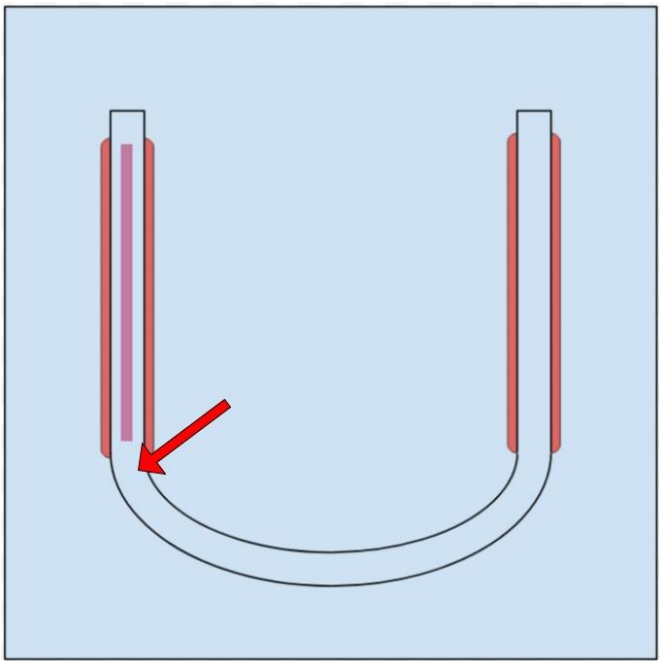

Figura 2-1. Esquema simple del LOC en el que se ve la posición del fluido (representado en color lila) en el punto límite en el que se produce el problema planteado y las zonas de los diferentes calentadores (sombreado rojo) que provocan la expansión del aire.

En la ***Figura 2-1*** queda representado dicho problema así como el punto (aproximado) a partir del cuál el aire entraría en contacto con la parte en la que está el calentador de 95ºC y eso junto con la expansión que viene del aire "nuevo" que entra en la parte de 65ºC provoca esa rápida y violenta expulsión del fluido.

Esto ocurría debido a que no existía ningún tipo de control en el movimiento del fluido, simplemente el motor que empuja o atrae el émbolo hacía un recorrido fijo y siempre desplazaba el mismo volumen, tanto hacia delante como hacia atrás. Uniéndose esto al hecho de que el proceso para realizar una PCR requiere número elevado de ciclos, las imperfecciones en el movimiento del fluido, es decir, la posición en la que queda tras un ciclo completo puede variar algún milímetro; cosa que a simple vista puede parecer una diferencia insignificante, pero al trabajar en una escala tan pequeña en los microcanales, esto supone una diferencia significativa. Un último factor que también contribuye a este problema es que las paredes de los microcanales no son completamente "impermeables" y una pequeñísima parte del fluido se va quedando pegado a estas a medida que se mueve en vez de seguir con el grueso principal (otro error ínfimo, pero significativo a las escalas en las que se trabaja).

2.2 Objetivo

Una vez localizado el problema, el objetivo se vio claro, era necesario poder controlar la temperatura y el movimiento del fluido para asegurar que el proceso se lleva a cabo correctamente y de forma que la perturbación introducida por la expansión del aire no hiciera que entrase en una posición en la que no debería estar. Para implementar esta solución se propusieron ciertos requisitos: el sistema debe tener en cuenta la temperatura real del LOC para asegurar que el proceso PCR se lleva a cabo correctamente, que el control del movimiento del fluido ha de hacerse mediante el empleo de un sistema de detección óptico y que el sistema ha de ser controlado por un microcontrolador.

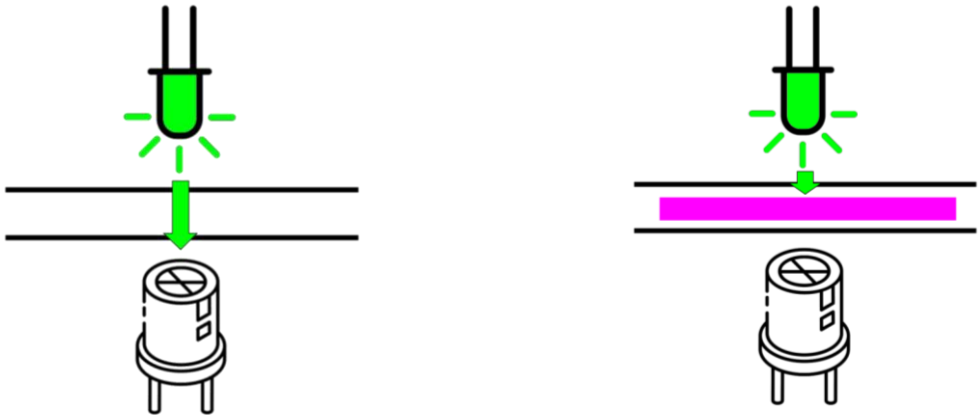

Figura 2-2. Esquemas de la posición relativa del sistema de control óptico y el LOC. En la imagen de la izquierda (a) se representa el sistema cuando el fluido no obstruye al fotodiodo y en la imagen de la derecha (b) se aprecia que el fluido bloquea la luz que emite el LED cuando está posicionado entre ambos componentes ópticos.

3 ESTADO DEL ARTE

Se mostrará en esta sección el punto de desarrollo actual y reciente en el que se encuentran tanto la tecnología referente a Lab-on-Chips aplicados a la realización de PCRs como las diferentes variantes y/u opciones que hay a la hora de realizar este procedimiento con sus ventajas y desventajas correspondientes. Pero antes que nada, se presenta una breve descripción de lo que es el procedimiento de una PCR.

3.1 Lab-on-Chips para PCR

Una vez descrito cómo es el procedimiento a seguir para una PCR, hay muchas formas diferentes de adaptarlo para que pueda reproducirse en un Lab-on-Chip. La idea principal es miniaturizar el procedimiento que se lleva a cabo en un laboratorio a gran escala a una escala mucho más pequeña en la que trabajan los Lab-on-Chips. Esto conlleva una serie de ventajas que se expondrán a continuación:

- Necesidad de un volumen de muestras considerablemente bajo.
- Tiempo del proceso rápido.
- Consumo de energía muy bajo.
- Protocolos de temperatura fácilmente replicables.
- Procesamiento de grandes series de muestras bajo las mismas condiciones.
- Requerimiento de espacio para los dispositivos muy bajos.

Por todo esto, los dispositivos Lab-on-Chip tienen un abanico muy interesante para la aplicación de la PCR tanto en diagnósticos en laboratorios como en point-of-care (es decir, diagnósticos en el momento, al lado del paciente); entre otras cosas.

Una de las principales ventajas (ya mencionada arriba) es el tiempo que requiere el proceso, para poner un poco más en perspectiva, se procede a dar unos valores aproximados de tiempo de duración tanto para el procedimiento en laboratorio a gran escala como para el procedimiento en un Lab-on-Chip de cara a familiarizar un poco más al lector. El rango de tiempo para el procedimiento a gran escala puede oscilar entre 90 minutos y 3 horas, mientras que en la contraparte de un Lab-on-Chip este rango oscila entre unos pocos minutos y media hora.

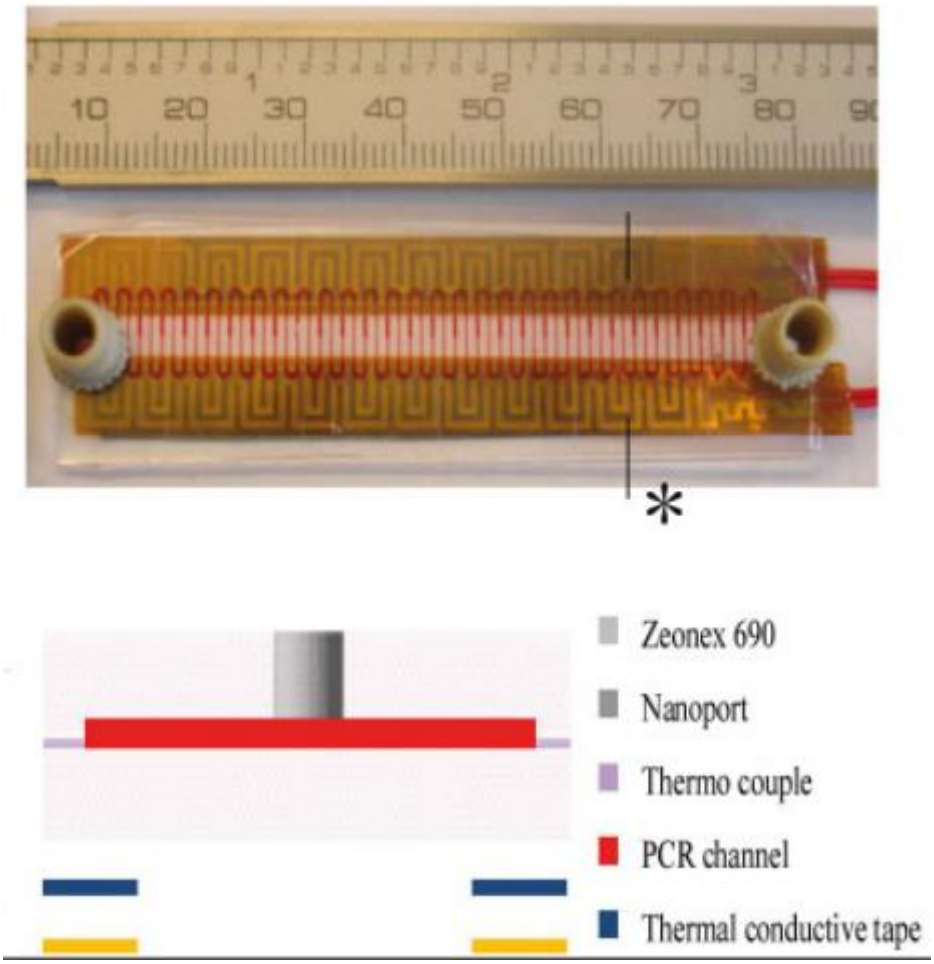

Figura 3-1. Ejemplo de un dispositivo para PCR de flujo continuo junto con una vista de su sección en el punto marcado por el *.

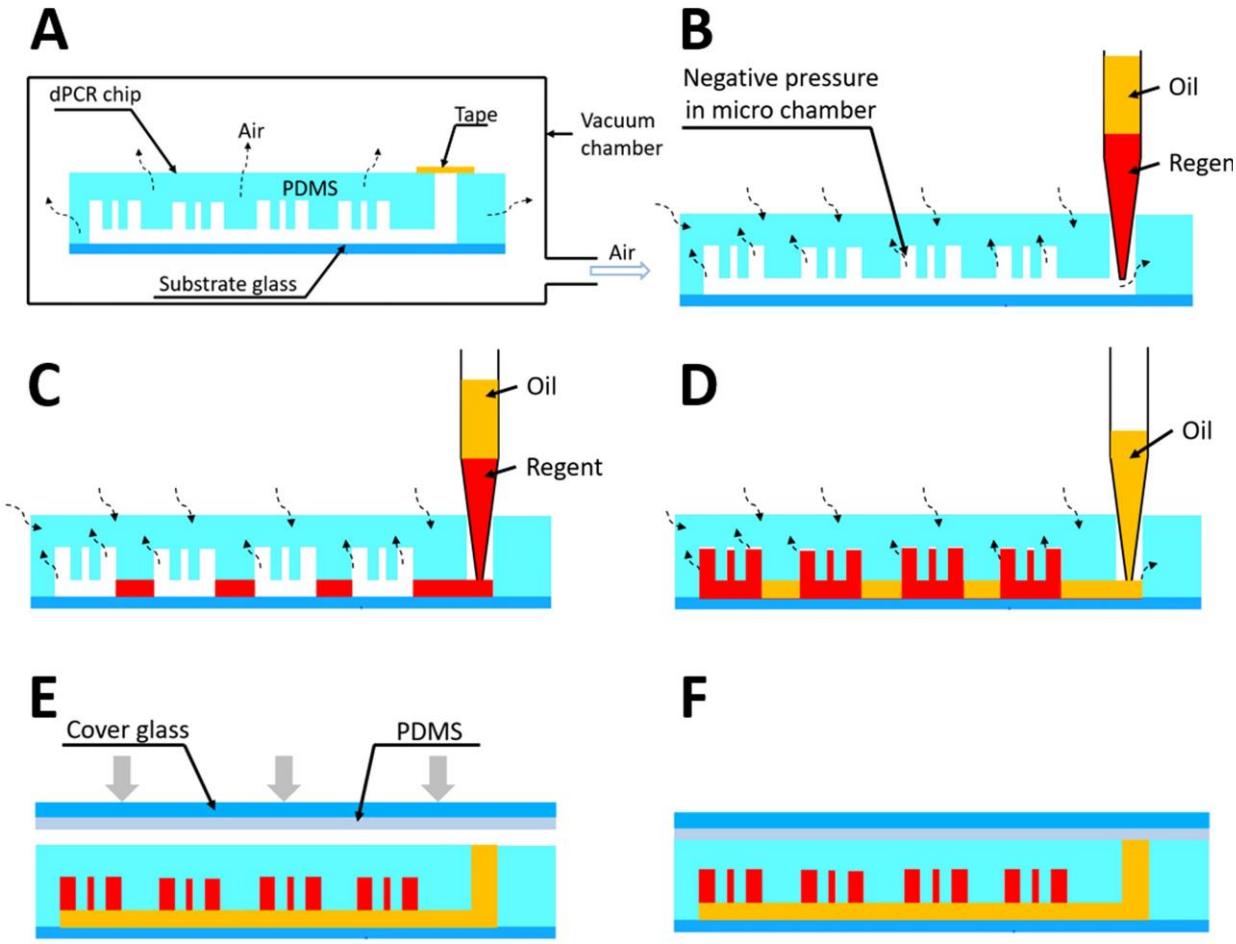

Figura 3-2. Ejemplo de un PCR con flujo segmentado. Vista de la sección en la que se observa el flujo de un LOC para dPCR (digital PCR).

3.2 PCR de dos o tres etapas

En cuanto a la realización del proceso hay dos opciones que se manejan actualmente: realizar el proceso en dos etapas, es decir, usar dos niveles de temperatura o usar tres niveles de temperatura. Siendo la PCR de tres etapas la que sigue el proceso estándar (explicado anteriormente en el apartado 1.3).

La idea tras la PCR de dos etapas se basa en unir la etapa de hibridación (o alineamiento) con la de extensión (o elongación) dado que ambas etapas presentan un rango relativamente amplio y gracias a esto se puede conseguir agilizar el proceso y hacerlo más eficiente. Es por ello que la opción elegida para realizar este trabajo es la PCR de dos etapas.

Por último y aunque todo parecen ventajas ya que en general varios estudios muestran que la PCR en dos etapas es más eficiente, hay que tener en cuenta que esta se ve afectada (obteniendo peores resultados) si la muestra de ADN no tiene un cierto mínimo de calidad, es decir, si la muestra de ADN está degradada y tiene muy mala calidad, la PCR de tres etapas sale ventajosa en la comparación.

3.3 Técnicas de temperaturas fijas (isotérmicas)

Ciertas técnicas o protocolos para realizar una PCR se basan en mantener las diferentes temperaturas (en caso de que exista más de una temperatura clave) que se necesitan para el proceso fijas en diferentes zonas del LOC y simplemente mover la muestra a dichas zonas según sea el momento correspondiente. Algunas de estas técnicas son las llamadas LAMP (Loop-Mediated Isothermal Amplification) o RPA (Recombinase Polymerase

Amplification). Cabe decir que esta última, la RPA, se basa en un protocolo alternativo a la PCR que utiliza otra enzima (la recombinasa) para amplificar el ADN. A continuación se profundiza brevemente en ambos procesos para dar al lector una pincelada sobre estas técnicas.

La técnica RPA es, como se acaba de mencionar, una alternativa a la PCR, ya que usa otro procedimiento para amplificar el ADN que, en vez de usar la reacción en cadena de la polimerasa, se basa en usar la enzima recombinasa. Este procedimiento es mucho más rápido que el de la PCR, dando resultados en un intervalo de 3 a 10 minutos y, además, requiere una única temperatura para su realización (la cual puede variar en un rango de los 37 a los 42ºC). Esta alternativa, por tanto, ofrece una mayor rapidez y es más adecuada quizás para aplicaciones de point-of-care o con pocos recursos disponibles ya que su coste es más reducido. Añadiendo otras de las ventajas de esta alternativa son:

- Velocidad. El tiempo necesario para completar el proceso es mucho menor que para la PCR.
- Sensibilidad. La muestra usada no requiere de una preparación previa para ser sometida al proceso.
- Especificidad. Esta técnica es capaz de detectar una única molécula de ADN aún estando en presencia de muchas otras dentro de la muestra.
- Gran abanico de aplicaciones.
- Escalado. La simplicidad de esta técnica. permite que en un solo dispositivo se realicen simultáneamente diferentes análisis.

Por otro lado, la técnica LAMP, al contrario que la PCR, se lleva a cabo a una temperatura constante que ronda los 60-65 ºC. Esta técnica permite una amplificación rápida del ADN y, además, se ha visto que es más robusta a la hora de usar muestras complejas (como pueden ser la sangre o el tejido vegetal). Añadir también que, como en el caso de la RPA, esta técnica tiene un claro enfoque de aplicación en el campo point-of-care. Algunas ventajas de esta técnica podrían ser las siguientes:

- Rapidez. Tiempo estimado entre 15 y 60 minutos.
- Simplicidad. Proceso isotérmico y de un único paso.
- Coste. Bastante reducido (en comparación) al no requerir un equipo muy sofisticado.

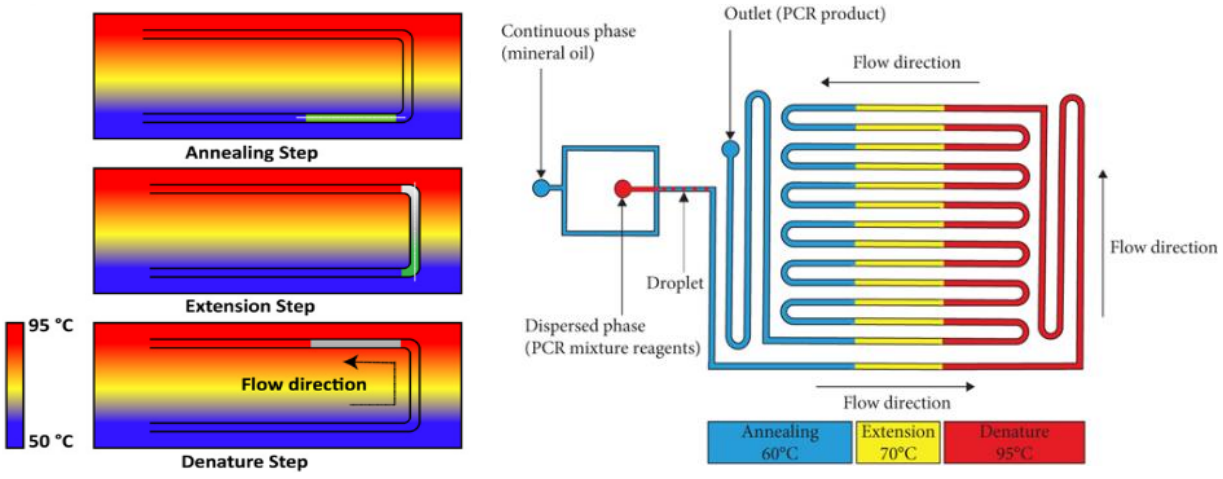

Figura 3-3. Ejemplos de PCRs de tres tres temperaturas fijas, en la imagen de la izquierda se presenta un LOC con forma de U (misma disposición que el usado en el presente trabajo) y a la derecha un LOC con forma de serpentín.

3.4 Técnicas de temperaturas variables

Estas técnicas se corresponden con las PCR tradicionales y son aquellas en las que la muestra se mantiene en el mismo lugar durante todo el proceso y, para poder completar la PCR con las diferentes temperaturas que requiere, se calienta y enfría el sistema en cada ciclo. Por regla general, esto se lleva a cabo en unos aparatos de laboratorio denominados termocicladores (thermal cyclers, en inglés) que logran modificar la temperatura de las muestras introducidas mediante una resistencia eléctrica distribuida por una placa que logra homogeneizar la temperatura. El uso de un termociclador permite procesar un gran número de muestras al mismo tiempo (pueden variar dependiendo del modelo), pero por otra parte, tiene la desventaja del tiempo que tarda en realizar dicho proceso, ya que variar la temperatura de las muestras de forma artificial (es decir, calentando o enfriando) es considerablemente más complejo que transportar dichas muestras a un entorno cuya temperatura sea diferente (como en el caso presentado en el punto anterior). A esto hay que añadir que estos aparatos no son apropiados para aplicaciones point-of-care y presentan un coste bastante elevado.

Figura 3-4. Termociclador automático de ThermoFischer Scientific.

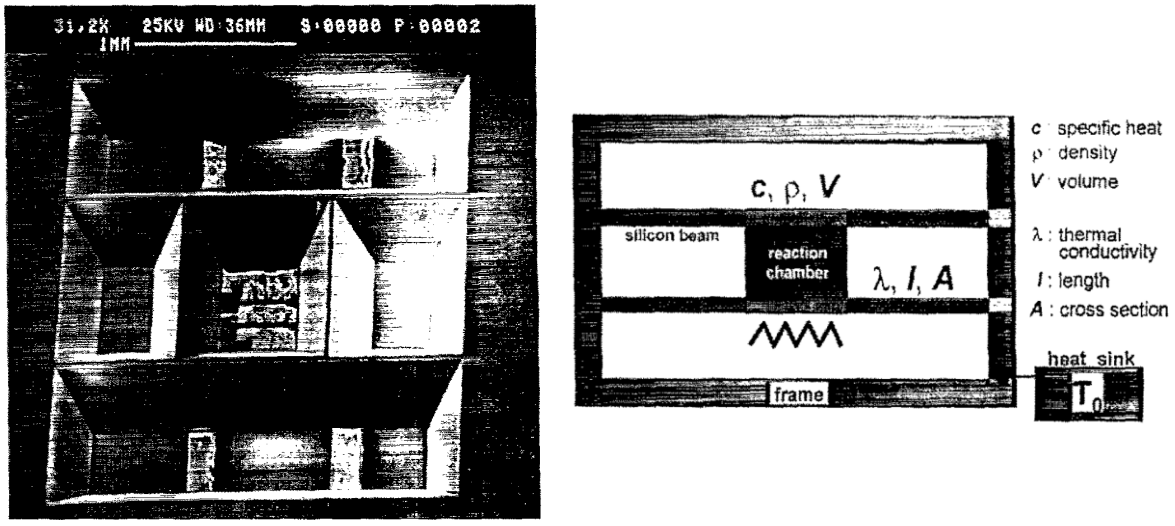

Figura 3-5. Ejemplo de una PCR de temperatura variable, a la izquierda tenemos la SEM del dispositivo y a la derecha una vista esquemática del mismo.

4 DESCRIPCIÓN DEL SISTEMA

En este capítulo se describirá de forma general el sistema usado para implementar la solución al problema planteado en el Capítulo 2. Se hará uso de un diagrama de bloques que representará el sistema de una forma mucho más esquemática y visual.

4.1 Diagrama general

A continuación se incluye el diagrama de bloques general del sistema que otorga una vista modular del sistema usado para una primera aproximación.

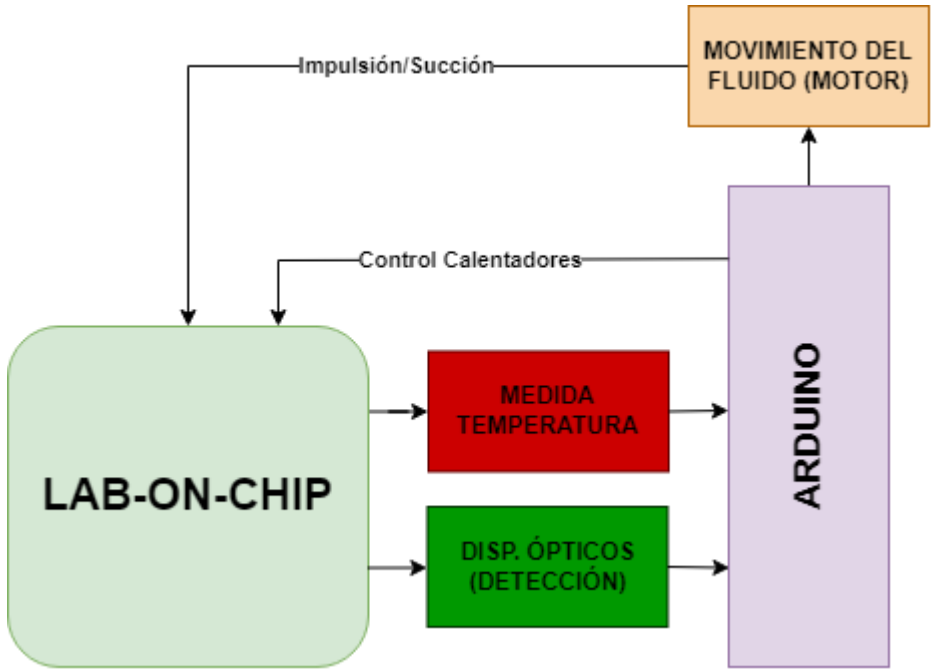

Figura 4-1. Diagrama de bloques general del sistema.

Como se puede observar en el diagrama representado, el bloque principal del sistema es el LOC, alrededor del cual gira todo el sistema. Otro bloque esencial es el que representa al microcontrolador (Arduino) que es la herramienta que nos permitirá controlar el sistema correctamente. Por último se distinguen los tres bloques que se han diseñado para el sistema: el bloque de medida y control de temperatura (en color rojo), el bloque de los dispositivos ópticos que nos permiten detectar la posición del fluido (representado en verde) y el bloque

correspondiente al motor, que se encarga del movimiento del fluido (en tono naranja). Todos estos bloque se comunican, a su vez, tanto con el bloque del Arduino como con el LOC. A continuación daremos una descripción breve sobre los tres bloque diseñados.

4.2 Bloque 1: movimiento del fluido (motor)

Este bloque es el encargado del movimiento del fluido y se encarga de que siempre esté en el lugar adecuado. Está compuesto por un motor paso a paso cuyo émbolo está conectado a una jeringuilla y ésta conectada al LOC de forma que cuando el motor avanza se introduce aire en el LOC y el fluido se desplaza hacia delante (es decir, desde la parte de 65ºC a la de 95ºC) y cuando el motor retrocede en pasos el fluido hace lo propio. Este sistema además está conectado al microcontrolador Arduino, que es el encargado de dar las órdenes, mediante una PCB que se ha diseñado para unir todo el conexionado necesario para el funcionamiento del motor (tanto alimentación como el acondicionamiento).

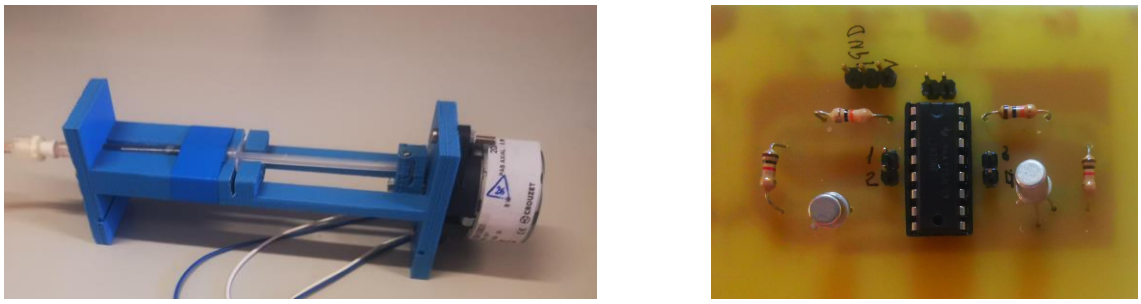

Figura 4-2. Componentes y conjunto que conforma el Bloque 1.

4.3 Bloque 2: medida y control de temperatura

Este bloque se encarga de controlar la temperatura en cada parte del LOC y mantenerla al nivel correspondiente para que el proceso de la PCR se pueda llevar a cabo correctamente. Este sistema se compone por un par de resistencias NTC que están conectados a una PCB que engloba todo el conexionado y acondicionamiento de los sensores y de ahí conectan con el Arduino donde la señal recibida se filtra y procesa con un algoritmo para obtener la temperatura real que hay en cada zona del LOC.

Una vez hemos obtenido la medida de cada resistencia NTC éstas se introducen cada una a un PID que se encarga de generar la salida correspondientes hacia los calentadores para que la temperatura siga siendo la correcta. Estas señales generadas por los PIDs salen del Arduino y van a la PCB de sensores y permiten que los transistores mosfet regulen la corrienet que pasa por cada calentador para así incrementar o decrementar la temperatura.

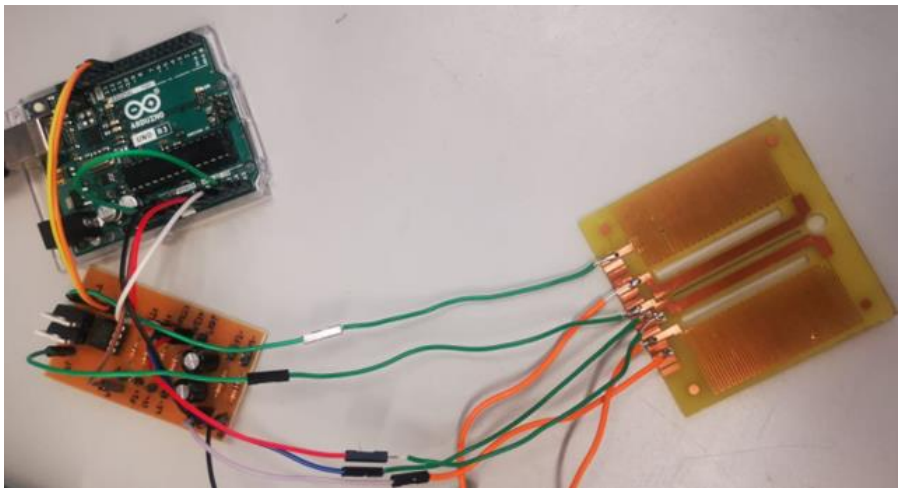

Figura 4-3. Conjunto del bloque 2.

4.4 Bloque 3: sistema de detección óptico

El último bloque que compone el sistema completo es el sistema óptico que se encargará de comunicar al microcontrolador la posición en la que se encuentra el líquido. Esto se consigue gracias a que los fotodiodos nos permiten saber si el fluido se encuentra en su posición ya que éste último obstruirá la luz emitida por los LEDs (tal y cómo se mostró previamente en la ***Figura 2-2***) y así disminuirá la señal que mandan los fotodiodos. En cuanto al conexionado, tanto los fotodiodos como los LEDs están conectados a la PCB de los sensores y de ahí, como el resto de componentes, se comunican con el Arduino y ahí se usará dicha información para tener un historial de los puntos por los que ha pasado el fluido.

Figura 4-4 Conjunto de componentes del sistema de detección óptico, a la izquierda los fotodiodos y a la derecha los LEDs.

5 BLOQUE 1: MOVIMIENTO DEL FLUIDO

.

En este capítulo se describirá de forma más exhaustiva el diseño del bloque 1 del sistema conjunto, referido al movimiento del fluido, y los resultados obtenidos. Añadir que estos resultados son en base a las pruebas aisladas de este bloque, los resultados de la integración completa se verán más adelante en el Capítulo 8.

5.1 Movimiento del motor y conexionado a la PCB

El movimiento del motor se realiza de forma muy sencilla, gracias a la librería Stepper.h de Arduino que nos permite definir una clase para nuestro motor paso a paso y lo único que tenemos que indicarle es el número de pasos de nuestro motor y los pines que comunican el Arduino con el motor.

En cuanto al diseño de la PCB, se ha seguido uno de los diseños típicos para control de motores paso a paso basados en el número de bobinas del motor, en este caso es para dos bobinas. Se mostrará a continuación varias imágenes, tanto de la PCB como del circuito esquemático.

Se puede observar en la Figura 5-1 que el circuito presenta los componentes típicos: el integrado L293D que es un driver de motores, las resistencias pertinentes para el buen conexionado y un par de transistores BJT.

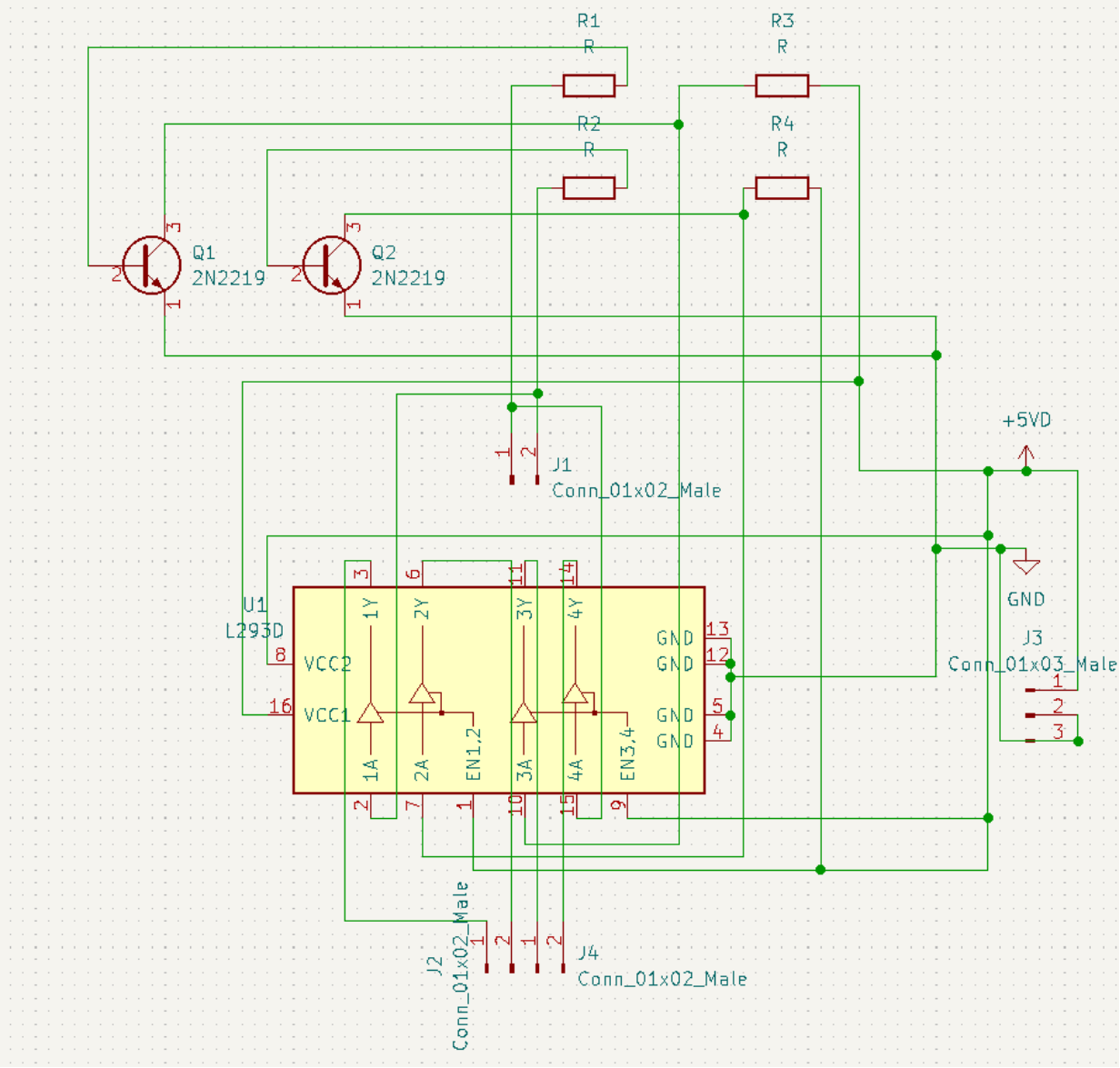

Figura 5-1. Imagen del circuito esquemático de la PCB del motor

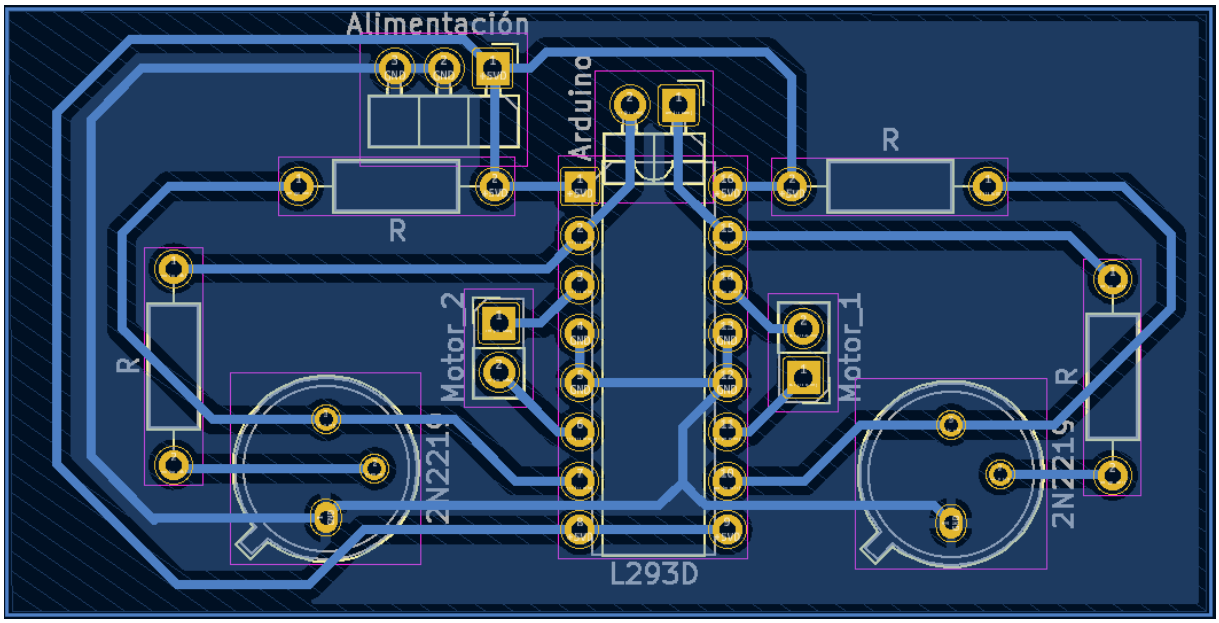

Figura 5-2. Diseño de la PCB de conexionado del motor..

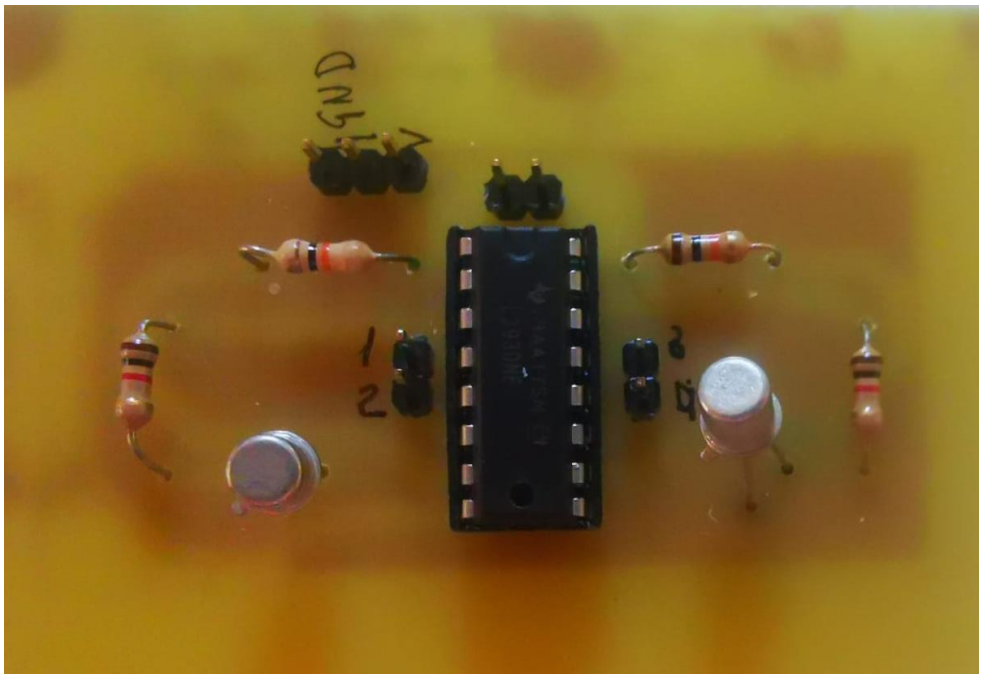

Figura 5-3. Imagen de la PCB del conexionado del motor.

5.2 Resultados

Los resultados fueron los esperados, el motor se movía según le llegaba la señal y sin ningún tipo de problema apreciable. El experimento que se planteó fue tratar de desplazar el fluido en un LOC de prueba compuesto simplemente por un canal recto, como se ve en la ***Figura 5-3***. El número de pasos final elegido para cada acción del motor fue uno, tanto para avanzar como para retroceder ya que el volumen de fluido que se ha de mover es muy pequeño.

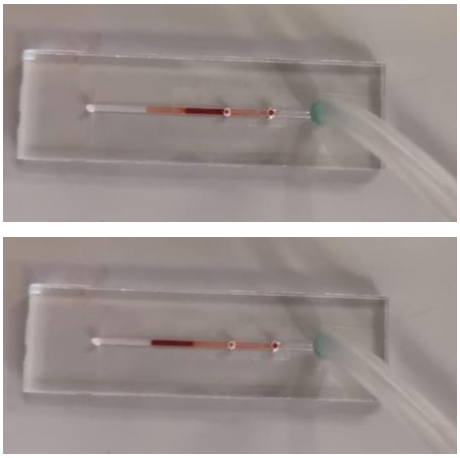

Figura 5-4. Resultados obtenidos. La imagen superior muestra el fluido en el punto inicial y la imagen inferior muestra el fluido ya desplazado debido a la acción del motor.

6 BLOQUE 2: MEDIDA Y CONTROL DE LA TEMPERATURA REAL DEL LOC

Pasamos a ver a continuación el bloque 2 del sistema que, como ya se ha comentado anteriormente varias veces, es el encargado de controlar la temperatura que hay en ambas partes del LOC de forma que la temperatura real sea la idónea.

6.1 Obtención de la temperatura real

La señal que envía la resistencia NTC pasa a través de la PCB de los sensores y llega a una entrada analógica de Arduino, dicho valor se lee y es procesado a través de un algoritmo y de un filtro paso bajo para evitar grandes variaciones en el valor debido a perturbaciones que puedan surgir. Dicho algoritmo es el siguiente:

Figura 6-1. Diagrama de flujo del algoritmo del cálculo y filtrado de la temperatura.

6.2 Control de la temperatura a través de los calentadores

Una vez tenemos el valor real de la temperatura que hay en cada parte del LOC, ese valor se pasa a un controlador de tipo PID (implementado gracias a la librería PID_v1.h de Arduino) que tiene como salida el nivel de tensión que enviar a la puerta de los transistores mosfet que son los encargados de regular la cantidad de corriente que pasa por cada calentador de forma que la temperatura aumente o descienda según sea necesario.

6.3 Resultados

Este bloque permite controlar la temperatura a unos tiempos del orden de los necesarios para que se lleve a cabo el proceso y mantenerla dentro de unos márgenes que no afectan al proceso. A continuación se muestran varias imágenes referentes a este bloque. El experimento que se realizó fue marcar una temperatura como referencia para cada PID de cada calentador y, con un termopar, comprobar que dicha premisa se cumplía con cierto márgen.

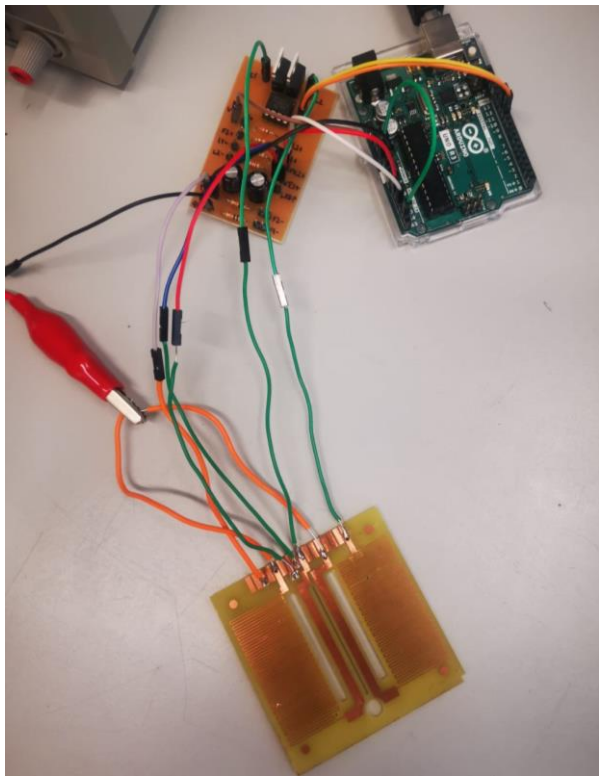

Figura 6-2. Imagen del conexionado del bloque 2.

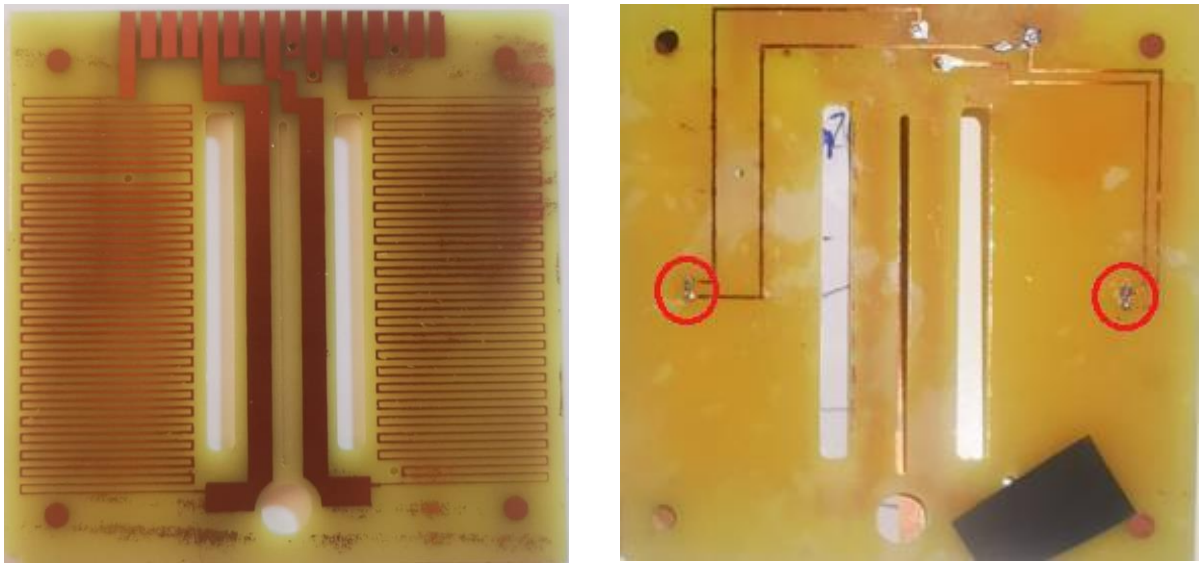

Figura 6-3. A la izquierda: imagen de la capa inferior del PCB que muestra un detalle de los calentadores en forma de serpentín. A la derecha: imagen de la capa superior del PCB que muestra un detalle de las resistencias NTC.

7 BLOQUE 3: SISTEMA DE DETECCIÓN ÓPTICO

Finalmente se presenta el diseño y los resultados obtenidos para el bloque del sistema de detección óptico que, como se ha mencionado ya varias veces, está compuesto por unos LEDs y unos fotodiodios que hacen la función de ojos, nunca mejor dicho, para el sistema y permiten conocer la posición en la que se encuentra el fluido, principalmente cerca de los puntos críticos del recorrido.

7.1 Diseño del bloque y resultados

Este bloque se compone por dos fotodiodos Hamamatsu S13948-01SB y dos LEDs de color verde básicos y todos se conectan a través de una resistencia de 330 Ohmios. El experimente que se llevo a cabo para comprobar que los fotodiodos servían para la aplicación que buscábamos y eran capaces de discernir si el fluido estaba bloqueando la luz emitida por el LED o no se hizo con el mismo LOC de prueba que se usó para el bloque 1 y se vieron los valores que el fotodiodo transmitía según estuviese el fluido o no.

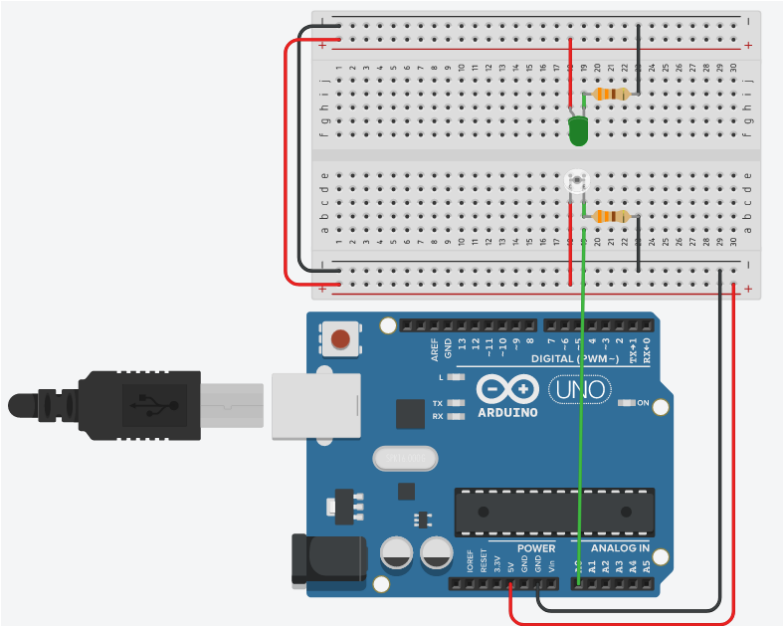

Figura 7-1. Imagen del conexionado tanto del fotodiodo como del LED.

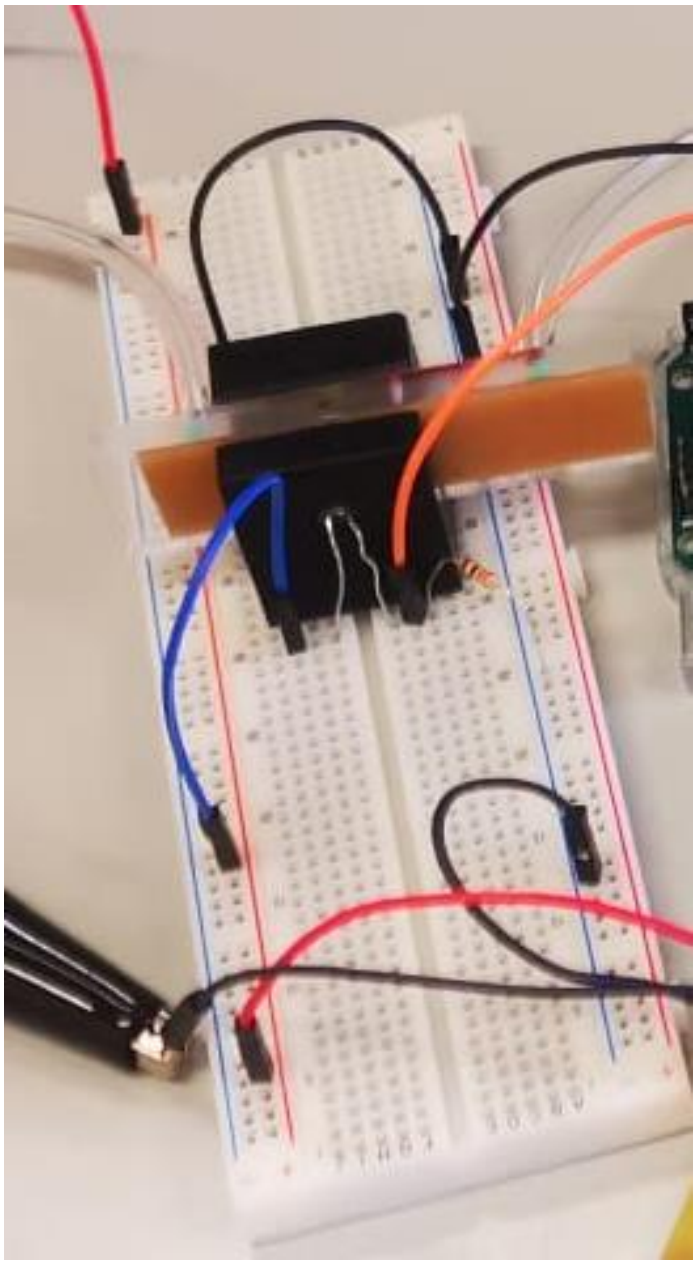

Figura 7-2. Experimento realizado para comprobar el funcionamiento del bloque 3.

Como se puede apreciar en la ***Figura 7-2***, para realizar este experimento se diseño e imprimió en 3D una pieza que permitió mantener el LOC en posición vertical y facilitar el alineamiento del fotodiodo y el LED a través de dos ranuras situadas en los lados opuestos del pasillo central en el que encajaba el LOC.

8 SISTEMA COMPLETO Y RESULTADOS

Pasamos ahora, una vez comprobado que cada bloque funciona de forma independiente, a analizar y desglosar el tramo final que consiste en la integración de todos los bloques de cara a obtener el sistema final objeto de este trabajo. Para ello, primero se hicieron pruebas de funcionamiento y depuración del funcionamiento de los bloques de dos en dos, por ejemplo, el funcionamiento del bloque 1 (movimiento del fluido) con el bloque 3 (sistema de detección óptico). Una vez comprobado el funcionamiento de cada par de bloques, se procedió, finalmente a la integración y depuración del sistema completo.

8.1 Integración motor – sistema óptico

Para comprobar el correcto funcionamiento conjunto de estos dos bloques se diseñaron un par de experimentos, cada uno de ellos enfocado a un punto clave de la interacción de estos dos bloques: rechazo de perturbaciones y completar el ciclo de movimiento necesario para la PCR.

8.1.1 Experimento 1 y resultados

En este primer experimento se quería comprobar que el motor controlaba el movimiento del fluido según la información que transmitía el sistema óptico introduciendo además una perturbación con una jeringa repleta de aire conectada a la entrada del LOC para mover el fluido de forma externa al motor. El objetivo era simular el comportamiento de la expulsión violenta del fluido debido a la expansión del aire, es decir, el problema a resolver por este trabajo.

En caso de funcionar correctamente, el motor, al llegar el fluido al punto del segundo fotodiodo (el que está colocado en la parte de 95ºC) debería contrarrestar el movimiento introducido por la perturbación manual haciendo que el fluido no sobrepasara dicha posición.

En cuanto a los resultados, el experimento se realizó y se comprobó el correcto funcionamiento coincidiendo con lo esperado. El motor era capaz de contrarrestar perfectamente y sin ningún problema la perturbación introducida. Añadir que hubo ciertos problemas con la señal enviada por los fotodiodos al funcionar simultáneamente con el motor, apareciendo ciertos picos indeseados en los momentos en los que el motor estaba en funcionamiento. Para solucionar esto se decidió introducir un condensador en paralelo con cada uno de los fotodiodos (están presentes en la PCB de los sensores) para filtrar la señal y solucionar estos problemas con la variación de la señal.

8.1.2 Experimento 2 y resultados

El segundo experimento se diseñó para comprobar que con dos fotodiodos podíamos controlar el movimiento del fluido con suficiente precisión como para asegurar que se podía replicar el ciclo necesario para realizar una PCR, siendo este ciclo el siguiente:

1. Motor en espera a que se introduzca la muestra. (Inicio)

2. Se mueve el fluido a la zona de 95ºC (Avance 95ºC).
3. Motor en espera hasta cumplir el tiempo necesario a 95ºC (Parte 95ºC).
4. Retroceso del fluido hasta la zona de 65ºC (Vuelta 65ºC + Avance 65ºC).
5. Motor en espera hasta cumplir el tiempo necesario a 65ºC (Parte 65ºC).
6. Repetir (volver al paso 2) hasta completar los ciclos que sean necesarios.
7. Fin del proceso

El correcto seguimiento de este procedimiento se implementó mediante una máquina de estados en el microcontrolador que, en base al estado de cada fotodiodo (libre o cubierto por el fluido) en cada momento y en el estado anterior, permitía determinar la posición del fluido y completar sin ningún problema el ciclo de forma correcta.

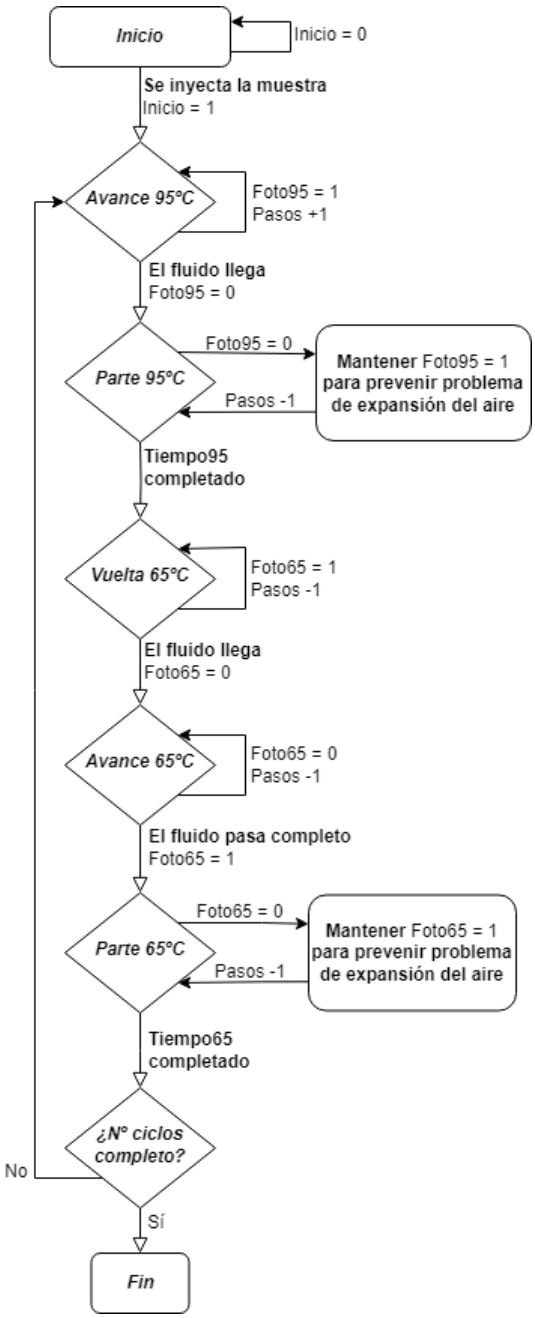

Figura 8-1. Diagrama de flujo de la máquina de estados implementada.

En referencia a la ***Figura 8-1***, se va a indicar el significado de ciertos términos importantes:

- Inicio: pin analógico de Arduino que se usó para señalizar el inicio (valor igual a uno) del proceso.
- Pasos: se ha usado para indicar la dirección en la que el motor funciona, siendo +1 avanzar y -1 retroceder.
- Foto95 y Foto65: variables usadas para indicar si dicho fotodiodo está tapado por el fluido (valor igual a cero) o está libre (valor igual a 1).

8.2 Integración motor – control y medida de temperatura

Para la integración de estos bloque se planteó englobar todo el control de temperatura (medida, procesado mediante el algoritmo y ejecución del PID) en una función a la que llamar mediante una interrupción en el programa de Arduino a cada segundo para que pudiera hacerse de forma simultánea al movimiento del motor, es decir, mientras el motor está (por ejemplo) empujando el fluido durante tres segundos, que cada segundo saltase una interrupción para ejecutar el control de temperatura.

El resultado obtenido fue negativo, dado que la instrucción de movimiento del motor bloquea por completo el programa de Arduino, es decir, mientras el motor está dando X pasos, hasta que no acabe esos pasos no ocurre nada más, no se podía interrumpir. Luego, se llegó a la conclusión de que esto realmente no era necesario, ya que la temperatura varía de una forma muy lenta y no se ha de ser tan exhaustivos con su control. Esto, sumado a que decidimos finalmente hacer que cada instancia del motor fuese de un único paso, hizo que el programa fuese mucho más fluido y no pasase tanto tiempo bloqueado en la acción del motor, por lo que se concluyó que no era determinante el uso de las interrupciones para el control de la temperatura.

Además, se decidió añadir un condensador de 110µF de filtrado para la alimentación del motor ya que al salir de la misma fuente de tensión (aunque de otra salida) que la alimentación de los calentadores, debido a que éstos demandan en ciertos puntos gran cantidad de corriente haciendo que haya picos en el consumo y algunos se veían reflejados en la alimentación del motor.

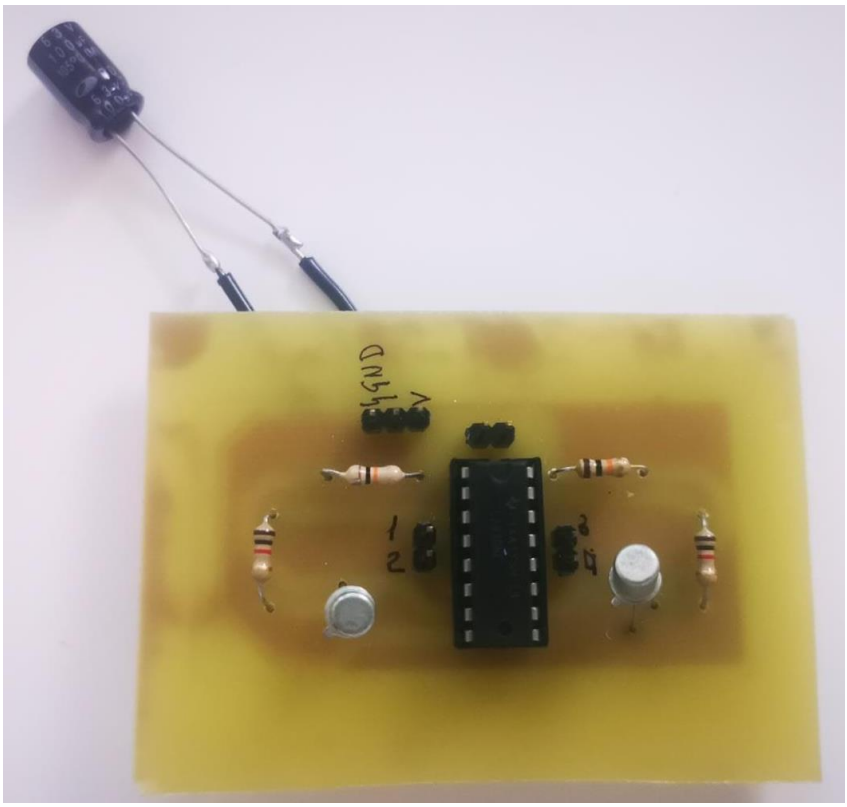

Figura 8-2. Imagen de la PCB del motor con el condensador de filtrado añadido.

8.3 Integración control y medida de la temperatura – sistema óptico

Este paso de la integración fue bastante sencillo, simplemente se conectaron ambos sistemas a la vez y se comprobó que no había ningún tipo de interferencia entre ellos. Esto se hizo moviendo manualmente el fluido mientras el sistema estaba controlando la temperatura y leyendo el valor de los fotodiodos. El experimento fue fructífero y no hubo ningún problema. Por último, mencionar que este experimento fue el que dio lugar a la idea del diseño de una PCB que incluyese todos los sensores (ya ha sido mencionada varias veces previamente) para facilitar el montaje y a continuación se presenta dicho diseño.

8.4 Diseño de la PCB de los sensores

Tal y como se acaba de comentar, esta PCB cumple con el objetivo de facilitar el montaje y el cableado necesario para todos los sensores del sistema. Se muestra a continuación una imagen del diseño, el circuito esquemático y de la PCB final.

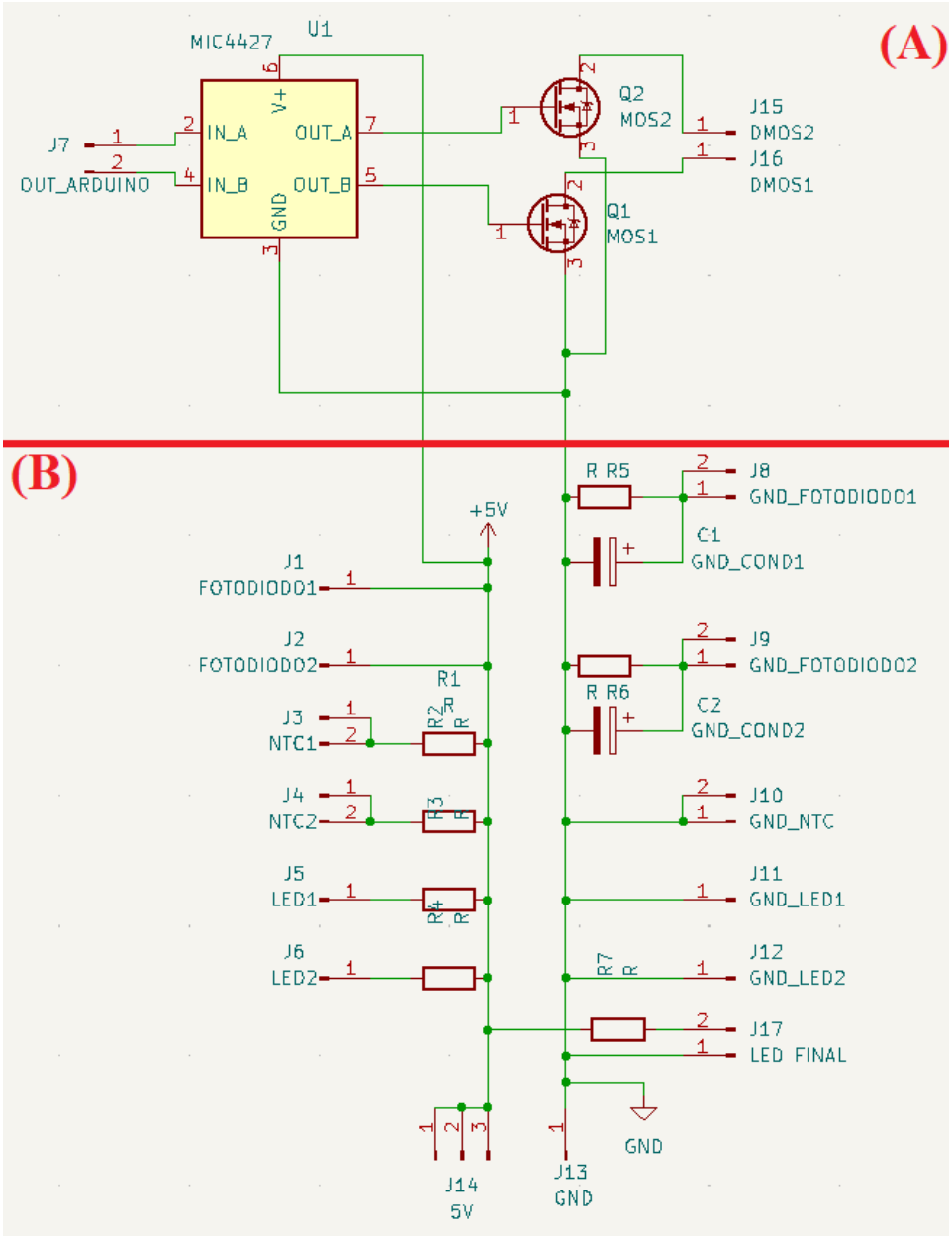

Figura 8-3.Imagen del circuito esquemático de la PCB de los sensores.

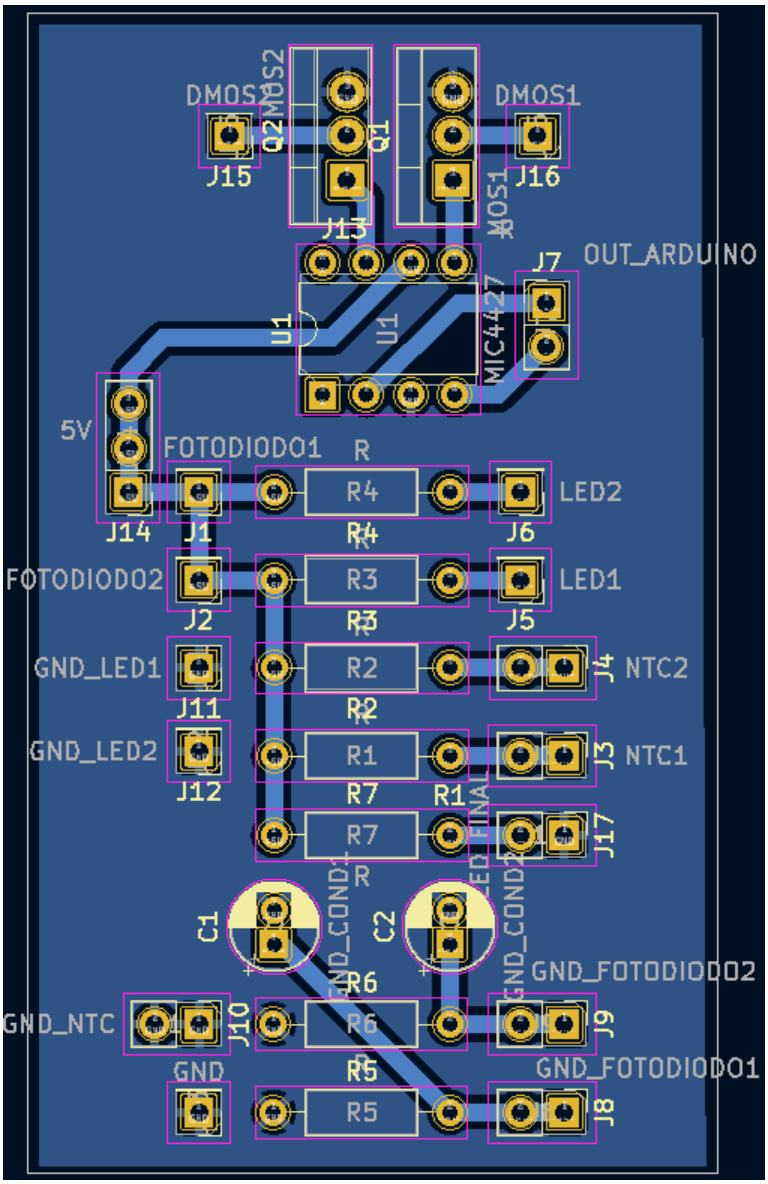

Figura 8-4. Diseño de la PCB de los sensores.

Se puede observar que el circuito se ha dividido en dos partes para separar así las dos funciones que consigue esta PCB. Por una parte, la sección A se corresponde con el circuito que nos permite regular la corriente que pasa por los calentadores de forma independiente ya que cada uno acaba conectado a una entrada diferente de Arduino, siendo todo esto regulado por los transistores MOS que se observan.

Por otro lado, la sección B pertenece a todo el conexionado de la parte del sistema óptico de detección (a excepción también de los pines colocados para la respectiva alimentación y tierra que se ven en la parte inferior de la imagen). Esta sección presenta las resistencias colocadas para los LEDs y los fotodiodos, los condensadores de filtrado para los fotodiodos y, además, incluye pines para un LED extra que se pensó usar para marcar que el proceso completo había finalizado, aunque finalmente no se ha llegado a usar.

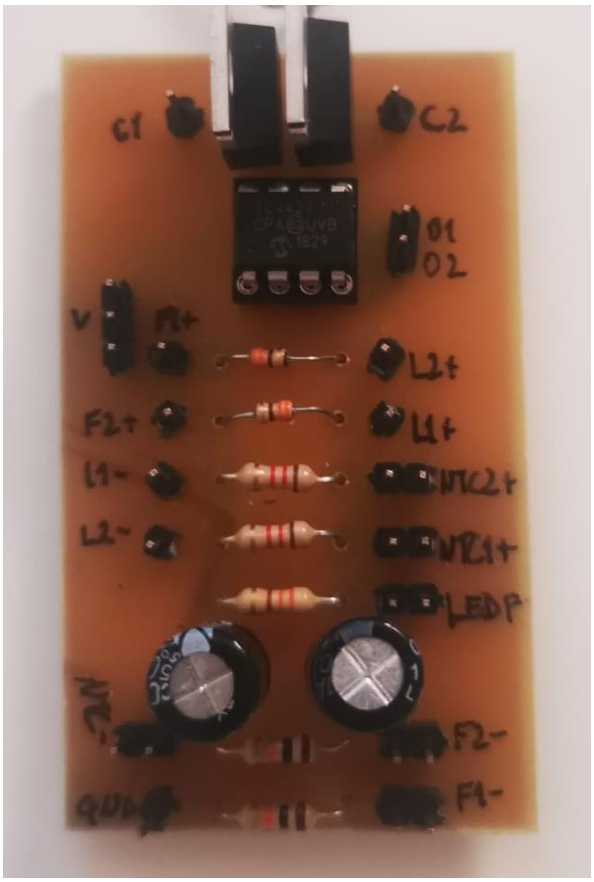

Figura 8-5. Imagen de la PCB de los sensores.

Posteriormente, tras su diseño y fabricación, se comprobó el funcionamiento de cada sensor conectado a través de esta PCB y se vio que todo funcionaba perfectamente.

8.5 Integración final de los tres bloques

Tras comprobar que inicialmente no hay ningún tipo de interferencia entre los bloques se pasa al último escalón del diseño que es integrar todos los bloques y depurar el sistema para que todo funcione simultánea y correctamente.

La primera modificación que hubo que hacer fue el incluir en la máquina de estados que gobierna el programa de Arduino el control de temperatura en cada estado de forma que se asegure que la temperatura siempre va a estar en torno a las referencias que elegimos. Tras esto se incluyó en cada estado una nueva función encargada de imprimir por pantalla los valores de temperatura obtenidos de cada resistencia NTC (y sus referencias), los valores obtenidos por cada fotodiodo y un valor que indicase si el motor estaba en movimiento hacia delante, detrás o parado, para poder hacer un seguimiento completo del funcionamiento del sistema final.

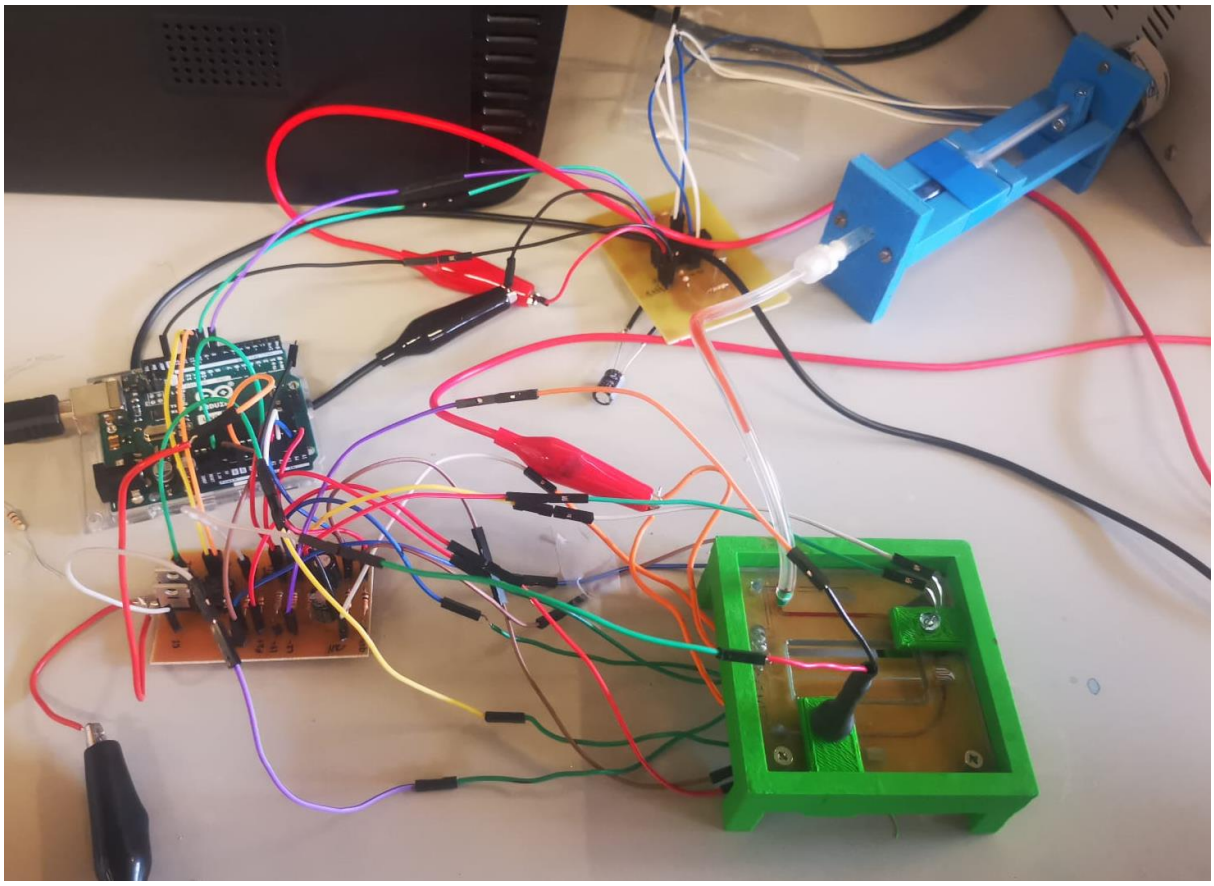

Figura 8-6. Imagen del sistema completo.

Tras diferentes problemas irrelevantes, mayoritariamente solucionados tras depurar un poco el código del programa y alguna que otro cable desconectado al manipular el sistema, se consiguió el resultado deseado, el sistema era capaz de completar varios ciclos completos mateniendo la temperatura deseada en cada parte del LOC y teniendo un control más que estable de la posición y el movimiento del fluido, eliminando por completo el problema a resolver previamente planteado. Se muestran a continuación varias imágenes que permiten ver el resultado final:

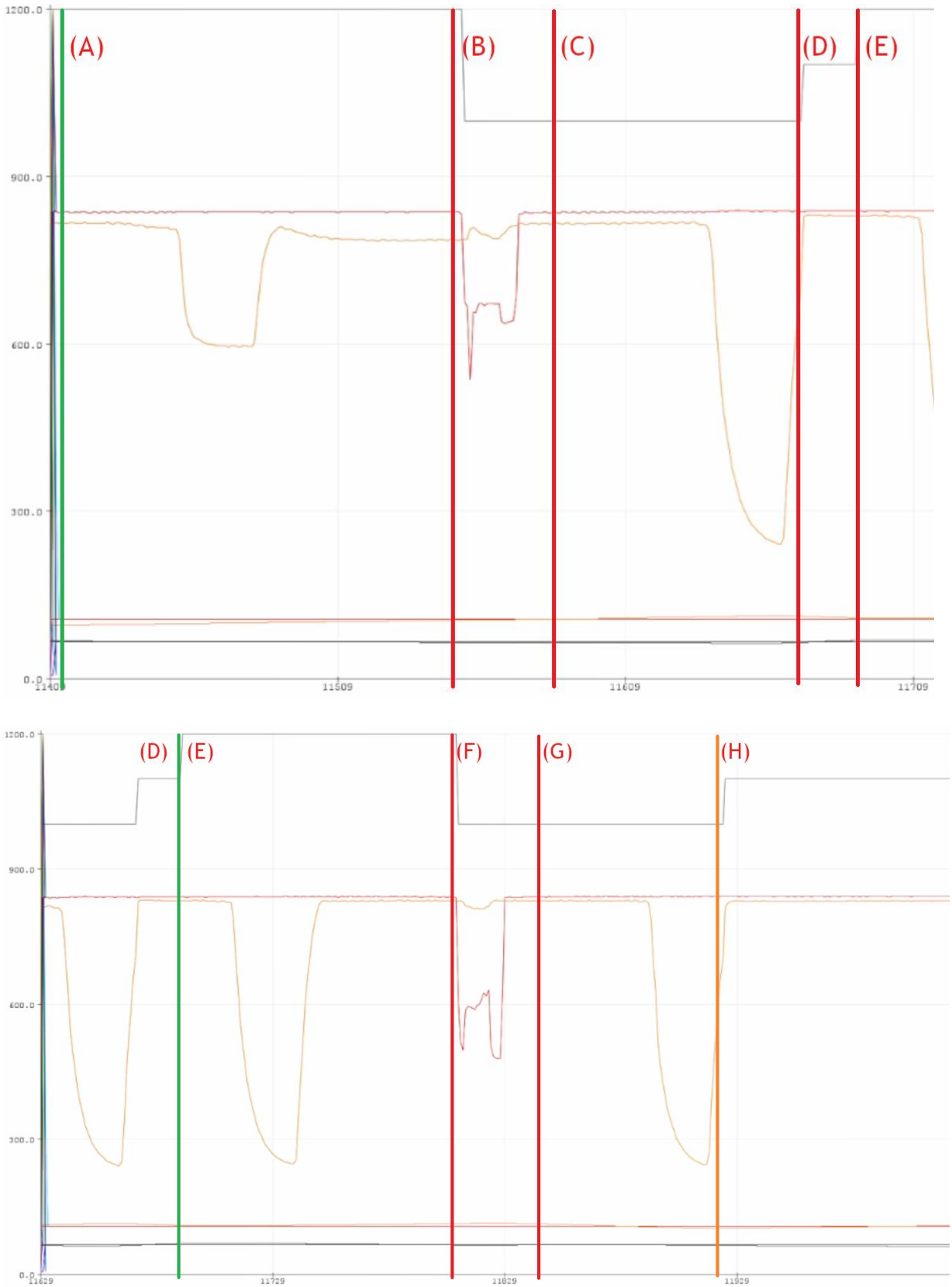

Figura 8-7. Resultado del proceso final de la impresión por pantalla de los valores de los fotodiodos, temperaturas medidas por cada resistencia NTC y la dirección de los pasos del motor. La imagen superior se corresponde con el ciclo inicial y la imagen inferior con el ciclo final.

Las imágenes mostradas en la ***Figura 8-6*** están separadas por líneas verticales que dividen el proceso completo en las diferentes etapas que lo conforman siendo diferenciados entre sí por una letra, a continuación se identifica cada tramo:

- Tramo A: inicio del proceso, el motor está a un valor de 1200 (valor que se escogió para representar que el motor empuja el fluido. En este tramo se aprecia cómo una de las señales tiene un valle, esta señal se corresponde con la del fotodiodo de la parte de 65ºC, que ve cómo el fluido pasa en dirección a la parte de 95ºC, la etapa inicial del proceso. Es tramo finaliza al llegar el fluido al fotodiodo de la parte de 95ºC.
- Tramo B: en esta parte el motor está constantemente compensando el efecto de la expansión del aire (valor del motor en 1000 que indica que está retrayendo el émbolo) y, tras cumplirse el tiempo necesario que ha de estar el fluido en la parte de 95ºC, comienza a mover el fluido hacia atrás, camino de la parte de 65ºC.
- Tramo C: en este tramo el fluido está retrocediendo (el motor sigue teniendo un valor de 1000) y vemos que finalmente está en la posición de 65ºC cuando el fotodiodo de dicha parte ha vuelto a pasar por un valle (indicando la entrada y salida del fluido).
- Tramo D: este tramo se corresponde con el tiempo de espera del fluido en la parte de 65ºC (el motor tiene un valor del 1100 que se escogió para representar que está parado) hasta cumplirse el tiempo necesario. En este instante se evalúa si se han cumplido el número de ciclos totales, como no es el caso, se pasa al tramo E.
- Tramo E: análogo al tramo A, marca el inicio del segundo ciclo (y último) y se desplaza el fluido hasta la parte de 95ºC.
- Tramo F: análogo al tramo B, correspondiente a la parte de espera a 95ºC.
- Tramo G: análogo al tramo C, el fluido se lleva de vuelta a la parte de 65ºC.
- Tramo H: análogo al tramo D, la única diferencia es que en este caso sí se ha cumplido el número total de ciclos establecido, por lo que se pasa al estado final en el que el sistema ha completado el proceso (el motor queda en un valor de 1100).

Por último, se añaden varias imágenes del LOC que representan un ciclo completo, se han identificado de la misma forma que se han identificado los tramos en la ***Figura 8-6***, correspondiéndose la imagen (A) con el tramo A y así sucesivamente. También se ha remarcado con un cuadro rojo la posición del fluido para que sea más fácil localizarlo. Mencionar que, al ser imágenes extraídas de un vídeo, la calidad no es la mejor posible (de ahí que se haya decidido resaltar la posición del fluido) y tampoco se puede apreciar la dirección en la que va el fluido (por ello se han definido las imágenes con los tramos correspondientes):

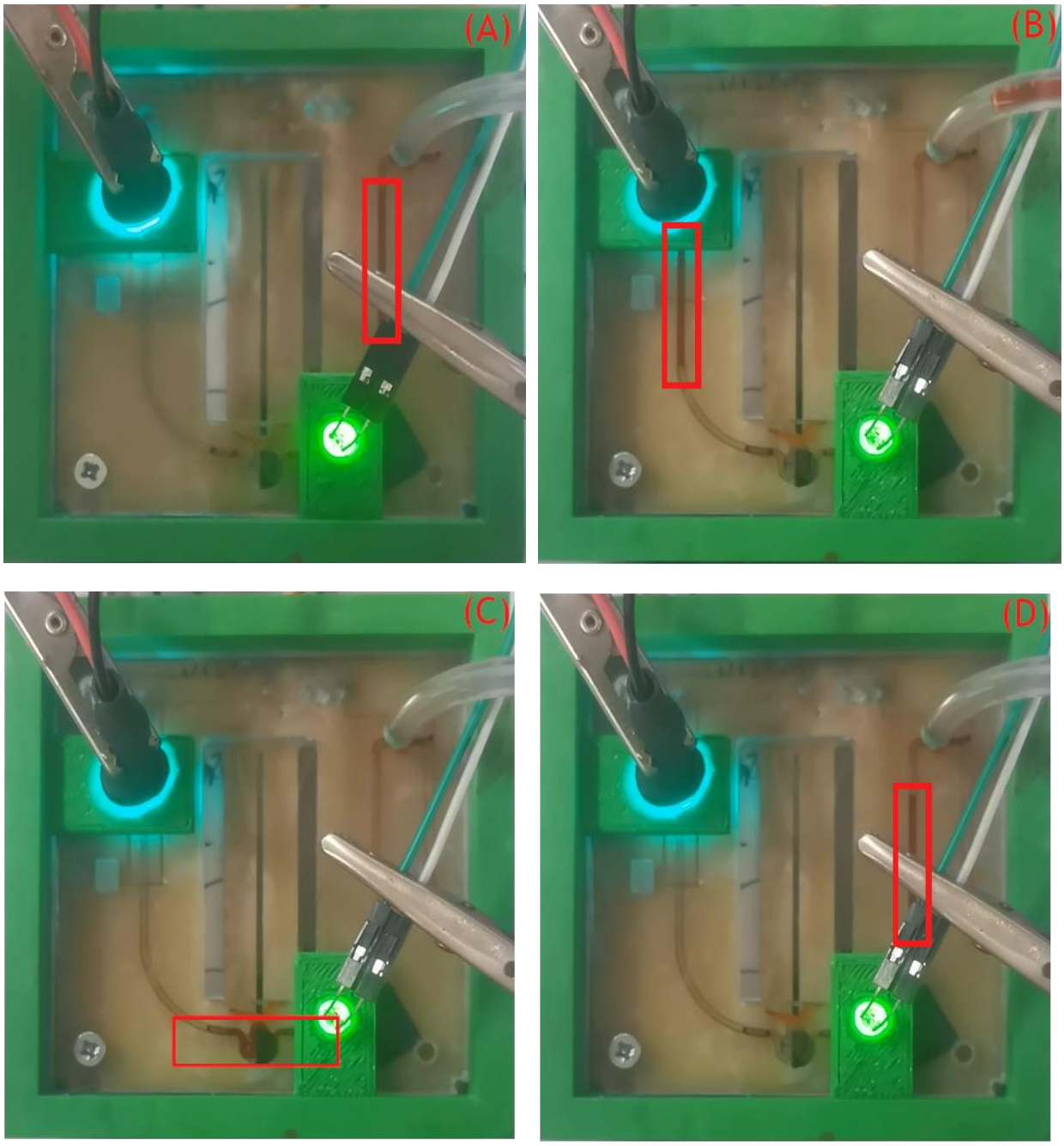

Figura 8-8. Imágenes del LOC durante el proceso final en diferentes puntos de un ciclo.

9 CONCLUSIONES

.

Presentamos aquí las conclusiones obtenidas tras haber completado el trabajo propuesto y habiendo obtenido grandes resultados en base a los requerimientos planteados para el mismo. Se escribirán unas conclusiones generales al respecto y se adjuntará también una hoja de características y limitaciones.

9.1 Conclusiones generales

Este trabajo ha obtenido grandes resultados, cumpliendo perfectamente los objetivos y ajustándose completamente a los requerimientos propuestos. Se ha demostrado que se podría replicar la técnica PCR en un lab on chip, formato idóneos para aplicaciones de campo donde se requiere usar esta técnica en el mismo momento y no tener la necesidad de hacer uso de un laboratorio completo. Además, se ha completado usando componentes de bajo coste lo que hace muy sencilla su reproducibilidad a la hora de su fabricación "en masa". Mencionar también que la puesta en marcha del sistema es realmente sencilla, ya que sólo requeriría introducir la muestra y una vez situada en su posición inicial bastaría con pulsar un botón que inicie la máquina de estados programada en Arduino y el proceso se llevará a cabo automáticamente desde ese punto hasta su finalización. Otro punto a favor con el que cuenta este sistema es que es fácilmente reprogramable, es decir, todo gira en base al programa codificado en Arduino el cuál está parametrizado de forma que, si se quiere realizar el proceso con 30 ciclos o con 25 simplemente basta con cambiar el valor de dicho parámetro; lo mismo ocurre con los tiempos de espera en cada parte del lab on chip, etc.

También es interesante resaltar que el sistema de control propuesto es fácilmente extendible a otro tipo de técnicas, es decir, no es válido únicamente para PCR. Cualquier aplicación que trabaje con fluidos a altas temperaturas usando LOCs es fácilmente adaptable a este sistema.

Como conclusión final se puede obtener que el campo de técnicas como la PCR (y similares) tienen gran interés como aplicaciones para los lab on chips, permitiendo minimizar costes, agilizar el proceso completo y proporcionar cierta inmediatez, siendo esto un gran avance el cuál se hace más vicisble, por ejemplo en países en vías que desarrollo que hoy día no disponen de tantísima infraestructura o instalaciones para este tipo de procedimientos. Concluir finalmente que, sin duda alguna, estas técnicas de biología molecular van a ver una gran adaptación al medio lab on chip.

9.2 Hoja de características y limitaciones

	Precio (€/por unidad)	Características	Limitaciones
Actuador lineal eléctrico Crouzet	84.23	-Fuerza: 43N -Velocidad: 1.67mm/s	La distancia mínimo de paso.
Resistencia NTC Murata NCP15XH103F03RC	0.024€	-Resistencia: 10kΩ -Montaje superficial	Rango de temperaturas.
Fotodiodo Hamamatsu S13948-01SB	3.02€	-Longitud de onda de máxima sensibilidad: 560nm -Voltaje (en inversa) máximo: [-0.5V,12V] -Corriente máxima: 5mA	Rango de la respuesta espectral.
LED	0.55€	-Longitud de onda dominante: 570nm -Intensidad luminosa: 50mcd	-

10 TRABAJOS FUTUROS

En algún lugar, algo increíble está esperando ser descubierto.

- Carl Sagan -

Se proponen en este último capítulo del trabajo algunas ideas o mejoras que se pueden aplicar al sistema que se ha diseñado y presentado como líneas en las que seguir el desarrollo y mejora del sistema, pero que debido a diferentes aspectos, no se han podido aplicar en este trabajo al quedar fuera del alcance del mismo.

Primeramente se propone hacer más ahínco en la integración de los diferentes bloques que conforman el sistema, es decir, podría agregarse el contenido de la PCB del motor junto con la PCB de los sensores, para así tener, finalmente todo integrado en un mismo lugar y facilitar aún más todo el conexionado y el montaje.

Además, como continuación de lo que se acaba de proponer surge la idea de hacer que dicha PCB final, que ya englobaría todo el conexionado, tenga un diseño y una forma tal que pueda ser conectada directamente al microcontrolador Arduino. De esta forma se conseguiría reducir aún más el espacio que el sistema requiere y facilitar aún más tanto el conexionado como el montaje.

Todo esto se podría incluir en un dispositivo que ya tuviese integrados los fotodiodos, LEDs, resistencias NTC y los calentadores para que así lo único que fuese necesario sería introducir el lab on chip con la muestra que se quiere procesar e iniciar el sistema.

REFERENCIAS

[1] Elveflow, "Introduction to lab-on-a-chip 2020: review, history and future" [En línea] Disponible en: https://www.elveflow.com/microfluidic-reviews/general-microfluidics/introduction-to-lab-on-a-chip-review-history-and-future/.

[2] A. Kapoor, S. Balasubramanian, V. Vaishampayan and R. Ghosh, "Lab-on-a-Chip: A Potential Tool for Enhancing Teaching-Learning in Developing Countries Using Paper Microfluidics," 2017 International Conference on Transforming Engineering Education (ICTEE), Pune, India, 2017, pp. 1-7

[3] Elveflow, "Materials for microfluidic devive fabrication: a review" [En línea] Disponible en: https://www.elveflow.com/microfluidic-reviews/general-microfluidics/materials-for-microfluidic-chips-fabrication-a-review-2017/

[4] Bioanalysis Zone, "What is a lab-on-a-chip technology?" [En línea] Disponible en: https://www.bioanalysis-zone.com/lab-chip-technology_loc/

[5] AZO Life Sciences, "What is a Lab-on-a-Chip?" [En línea] Disponible en: https://www.azolifesciences.com/article/What-is-Lab-on-a-Chip.aspx

[6] Lignos, I.G., Wootton, R.C.R., DeMello, A.J., Stone, B.M. (2013). Segmented Flow Microfluidics. In: Roberts, G.C.K. (eds) Encyclopedia of Biophysics. Springer, Berlin, Heidelberg.

[7] Wikipedia, The Free Encyclopedia, "Taylor dispersion" [En línea] Disponible en: https://en.wikipedia.org/wiki/Taylor_dispersion

[8] Casadevall i Solvas, Xavier & deMello, Andrew. (2010). Droplet microfluidics: Recent developments and future applications. Chemical communications (Cambridge, England). 47. 1936-42. 10.1039/c0cc02474k.

[9] TwistDx, "Recombinase Polymerase Amplification, or RPA, is the breakthrough, isothermal replacement to PCR" [En línea] Disponible en: https://www.twistdx.co.uk/rpa/

[10] Jin, Hong & Seo, Seung & Lee, Hye & Cho, Sohee & Ge, Jianye & King, Jonathan & Budowle, Bruce & Lee, Soong. (2013). Differences of PCR efficiency between two-step PCR and standard three-step PCR protocols in short tandem repeat amplification. Australian Journal of Forensic Sciences. 46. 1-11. 10.1080/00450618.2013.788681.

[11] Köhler, M. (2008). PCR Lab-on-Chip Devices. In: Li, D. (eds) Encyclopedia of Microfluidics and Nanofluidics. Springer, Boston, MA.

[12] CSIC, "Así funciona la prueba PCR, la foto diagnóstica del coronavirus" [En línea] Disponible en: https://www.csic.es/es/actualidad-del-csic/asi-funciona-la-prueba-pcr-la-instantanea-diagnostica-del-coronavirus

[13] SÁNCHEZ, EDWIN, NINA, MARITZA, AGUIRRE, PAMELA, ARCE, MAYRA, TORO, NEYDA, & VILELA, RODRIGO. (2014). Amplificación isotérmica mediada por LOOP (LAMP) de ácidos nuclecios en el diagnostico clínico. Revista CON-CIENCIA, 2(1), 127-140. Recuperado en 06 de julio de 2023, de http://www.scielo.org.bo/scielo.php?script=sci_arttext&pid=S2310-

02652014000100014&lng=es&tlng=es.

[14] Foo, P.C., Nurul Najian, A.B., Muhamad, N.A. et al. Loop-mediated isothermal amplification (LAMP) reaction as viable PCR substitute for diagnostic applications: a comparative analysis study of LAMP, conventional PCR, nested PCR (nPCR) and real-time PCR (qPCR) based on Entamoeba histolytica DNA derived from faecal sample. BMC Biotechnol 20, 34 (2020).

[15] Jena Bioscience, "Isothermal DNA Amplification & LAMP" [En línea] Disponible en: https://www.jenabioscience.com/molecular-biology/isothermal-amplification-lamp

[16] Wikipedia, la enciclopedia libre, "Termociclador" [En línea] Disponible en: https://es.wikipedia.org/wiki/Termociclador

[17] Zhang, C., Xu, J., Ma, W., & Zheng, W. (2006). PCR microfluidic devices for DNA amplification. Biotechnology advances, 24(3), 243–284.

[18] Madhusudan B Kulkarni and Sanket Goel 2020 Eng. Res. Express 2 042001

Anexos

En este anexo se adjuntarán los diferentes códigos programados en Arduino que se han usado en el desarrollo del presente trabajo.

valorDiodo_ADC.ino

Código empleado para obtener y mostrar por pantalla el valor que transmite cada fotodiodo.

```
const int diodo1 = A4;
const int diodo2 = A5;
int valorDiodo1;
int valorDiodo2;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

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

  valorDiodo1 = analogRead(diodo1);
  valorDiodo2 = analogRead(diodo2);
  Serial.println(valorDiodo1);
  Serial.println(valorDiodo2);
  delay(200);

}
```

Prueba_motor.ino

Código empleado simplemente para comprobar el correcto funcionamiento y conexionado del motor.

```
#include <Stepper.h>

float V1 = 20; //Volumen impulsión (µL)
float radio = 2.25; //radio del émbolo
float tiempo = 4; // tiempo en segundos
float area = 3.141592654*radio*radio;
float tiempospeed = 60/tiempo;
float L1 = V1/area;
int pasos = round(3/0.033);
```

```
  Stepper motorcillo(pasos, 8, 9); //Stepper nombre motor (número de
pasos por vuelta, pins de control)

  void setup()
  {
    // Velocidad del motor en RPM
    motorcillo.setSpeed(tiempospeed);
  }

  void loop()
  {
    //Girar una vuelta entera en un sentido
    Serial.begin(9600);
    Serial.println("patras");
    motorcillo.step(-1);
    //delay(5000); //Pequeña pausa

    //Serial.println("palante");
    //motorcillo.step(pasos);
    delay(100);
    //Girar una vuelta entera en sentido contrario
    //stepper.step();
    //delay(500); //Pequeña pausa
  }
```

controlTempDoble.ino

Código empleado para comprobar que los dos controladores PID funcionan correctamente y consiguen que los calentadores tengan la temperatura indicada.

```
  #include <PID_v1.h>

  double Ref, Input, Output, Output1, Output2;
  double Ref95, Input95, Output95;
  double Ref65, Input65, Output65;
  double Kp=0.001,Ki=30,Kd=50;
  float alfa = 0.05;
  float error = 0;
  float error95 = 0;
  float error65 = 0;
  float Temp95 = 50;
  float Temp65 = 50;
  int R95 = 1200;
  int R65 = R95;
  float beta95 = 3445.0; //Mismos valores que para beta95, ya que son
muy próximos y no existe gran diferencia entre beta80 y beta95
  float beta65 = beta95;
  float To = 298.15;
  int Vcc = 5;
  int Ro = 10000;
  int ciclo = 0;

  PID myPID95(&Input95, &Output95, &Ref95, Kp, Ki, 200, DIRECT);
  PID myPID65(&Input65, &Output65, &Ref65, Kp, Ki, Kd, DIRECT);
```

```
  unsigned long InicioTiempo = 0;
  unsigned long TomaMedidaTiempo = 0;
  unsigned long DiferenciaTiempo;
  unsigned long INTERVALO_CALENTADO = 600000; //Si 1 min son 60000
milisegundos, 10 min serán 600000 milisegundos. Ponemos el tiempo en
milisegundos para trabajar con la función "millis()"
  int comienzo = 0;
  const int inicio = A4;
  double medida95;
  double medida95f = 0;
  double medida65;
  double medida65f = 0;

  void setup() {
    comienzo = 0;
    myPID95.SetMode(AUTOMATIC);
    Ref95 = Temp95;

    myPID65.SetMode(AUTOMATIC);
    Ref65 = Temp65;

    pinMode(11, OUTPUT);  //Salida 95°C
    pinMode(10, OUTPUT);  //Salida 65°C
    Serial.begin(9600);

    Output1 = 0;
    Output2 = 0;
  }

  void loop() {

    if (analogRead(inicio) >= 512){
      if (comienzo == 0){
        comienzo = 1;
        InicioTiempo = millis();
      }
      else{
        //ciclo = ciclo+1;
        //Serial.println(ciclo);

        TomaMedidaTiempo = millis();
        DiferenciaTiempo = TomaMedidaTiempo - InicioTiempo;


        float ADC95 = analogRead(A0); //Leemos NTC del lado de 95°C
        //Calculos para 95°C
        float Vout95=ADC95*5/1023;
        float VNTC95=Vout95;
        float RNTC95 = (VNTC95*R95)/(Vcc-VNTC95);
        float alfa95 = (1000/To)+((1000/beta95)*(log(RNTC95/Ro)));
        float T95 = 1000/alfa95 - 273.15;


        float ADC65 = analogRead(A2); //Leemos NTC del lado de 65°C
        //Calculos para 65°C
```

```
        float Vout65=ADC65*5/1023;
        float VNTC65=Vout65;
        float RNTC65 = (VNTC65*R65)/(Vcc-VNTC65);
        float alfa65 = (1000/To)+((1000/beta65)*(log(RNTC65/Ro)));
        float T65 = 1000/alfa65 - 273.15;


        medida95 = T95; //Medida va a guardar el valor de temperatura
calculado previamente en unidades de temperatura y no de ADC
        medida95f = medida95*alfa + (1-alfa)*medida95f;
        Input95 = medida95f;
        error95 = Ref95 - medida95f;

        medida65 = T65; //Medida va a guardar el valor de temperatura
calculado previamente en unidades de temperatura y no de ADC
        medida65f = medida65*alfa + (1-alfa)*medida65f;
        Input65 = medida65f;
        error65 = Ref65 - medida65f;

        myPID95.Compute();
        //delay(10);
        myPID65.Compute();
        //delay(10);

        Serial.print(medida95f);
        Serial.print(",");
        Serial.print(Ref95);
        Serial.print(",");
        Serial.print(medida65f);
        Serial.print(",");
        Serial.println(Ref65);
 //       Serial.print(",");
 //       Serial.print(Output95);
 //       Serial.print(",");
 //       Serial.println(Output65);

        analogWrite(11, Output95);
        analogWrite(10, Output65);

        if(DiferenciaTiempo >= INTERVALO_CALENTADO){
          //digitalWrite(11, LOW); //Después de 10 minutos de calentado,
apagamos el microcalentador.
          myPID95.SetMode(MANUAL);
          Output95 = 0;
          if (T95 > 35 && T95 <= Temp95+2)
          {
            //Serial.println("Caliente");
          }
         else if (T95 <= 35)
         {
            //Serial.println("Temp. Ambiente");
         }
        }

     }
 }
```

```
}
```

motorTemp.ino

Código empleado para comprobar la integración del bloque del motor junto con el de control de la temperatura, correspondiente a lo expuesto en el apartado 8.2 del presente trabajo.

```
#include <PID_v1.h>
#include <Stepper.h>

//----------------Stepper
float V1 = 20;           //Volumen impulsión (µL)
float radio = 2.25;      //radio del émbolo
float tiempo = 4;        //tiempo en segundos
float area = 3.141592654*radio*radio;
float tiempospeed = 60/tiempo;
float L1 = V1/area;
//int pasos = round(L1/0.033);
int pasos = round(3/0.033);

int t65 = 15000;          //tiempo total (en ms) que requiere la parte
de 65°C
int t95 = 15000;          //tiempo total (en ms) que requiere la parte
de 95°C
int t65i = 0;            //marca de tiempo de cuándo comienza la parte
de 65°C en el ciclo actual
int t95i = 0;            //marca de timepo de cuándo comienza la parte
de 95°C en el ciclo actual
int estado = 1;           //variable que marca el estado de la rutina
del proceso
int ciclos = 0;           //contador del número de ciclos que se han
producido
int t95f = 0;
int t65f = 0;

Stepper motorcillo(24, 8, 9); //Stepper nombre motor (número de pasos
por vuelta, pins de control)

//---------------Control temperatua

double Ref, Input, Output, Output1, Output2;
double Ref95, Input95, Output95;
double Ref65, Input65, Output65;
double Kp=0.001,Ki=30,Kd=50;
float alfa = 0.05;
float error = 0;
float error95 = 0;
float error65 = 0;
float Temp95 = 50;
float Temp65 = 50;
```

```
  int R95 = 1200;
  int R65 = R95;
  float beta95 = 3445.0; //Mismos valores que para beta95, ya que son
muy próximos y no existe gran diferencia entre beta80 y beta95
  float beta65 = beta95;
  float To = 298.15;
  int Vcc = 5;
  int Ro = 10000;
  int ciclo = 0;

  PID myPID95(&Input95, &Output95, &Ref95, Kp, Ki, 200, DIRECT);
  PID myPID65(&Input65, &Output65, &Ref65, Kp, Ki, Kd, DIRECT);

  unsigned long InicioTiempo = 0;
  unsigned long TomaMedidaTiempo = 0;
  unsigned long DiferenciaTiempo;
  unsigned long LlamadaControlTemp = 0;
  unsigned long SalidaControlTemp = 0;
  unsigned long TiempoControlTemp = 0;
  unsigned long MovimientoMotor = 0;
  unsigned long TiempoEjecucion = 0;
  unsigned long InicioIf = 0;
  unsigned long FinalIf = 0;
  unsigned long TiempoIf = 0;
  unsigned long INTERVALO_CALENTADO = 600000; //Si 1 min son 60000
milisegundos, 10 min serán 600000 milisegundos. Ponemos el tiempo en
milisegundos para trabajar con la función "millis()"
  int comienzo = 0;
  const int inicio = A4;
  double medida95;
  double medida95f = 0;
  double medida65;
  double medida65f = 0;

  void ControlTemp(){     //Función que mide la temperatura en cada
calentador y se encarga de ejecutar el control correspondiente
      float ADC95 = analogRead(A0); //Leemos NTC del lado de 95°C
      //Calculos para 95°C
      float Vout95=ADC95*5/1023;
      float VNTC95=Vout95;
      float RNTC95 = (VNTC95*R95)/(Vcc-VNTC95);
      float alfa95 = (1000/To)+((1000/beta95)*(log(RNTC95/Ro)));
      float T95 = 1000/alfa95 - 273.15;

      medida95 = T95; //Medida va a guardar el valor de temperatura
calculado previamente en unidades de temperatura y no de ADC
      medida95f = medida95*alfa + (1-alfa)*medida95f;
      Input95 = medida95f;
      error95 = Ref95 - medida95f;

      myPID95.Compute();
      //delay(10);
      analogWrite(11, Output95);

      float ADC65 = analogRead(A2); //Leemos NTC del lado de 65°C
      //Calculos para 65°C
      float Vout65=ADC65*5/1023;
```

```
      float VNTC65=Vout65;
      float RNTC65 = (VNTC65*R65)/(Vcc-VNTC65);
      float alfa65 = (1000/To)+((1000/beta65)*(log(RNTC65/Ro)));
      float T65 = 1000/alfa65 - 273.15;

      medida65 = T65; //Medida va a guardar el valor de temperatura
calculado previamente en unidades de temperatura y no de ADC
      medida65f = medida65*alfa + (1-alfa)*medida65f;
      Input65 = medida65f;
      error65 = Ref65 - medida65f;

      myPID65.Compute();
      //delay(10);
      analogWrite(10, Output65);

        Serial.print(medida95f);
        Serial.print(",");
        Serial.print(Ref95);
        Serial.print(",");
        Serial.print(medida65f);
        Serial.print(",");
        Serial.println(Ref65);
  }

  void setup() {

    // Velocidad del motor en RPM
    motorcillo.setSpeed(tiempospeed);
    Serial.begin(9600);
    ciclos = 0;

    comienzo = 0;
    myPID95.SetMode(AUTOMATIC);
    Ref95 = Temp95;

    myPID65.SetMode(AUTOMATIC);
    Ref65 = Temp65;

    pinMode(11, OUTPUT);  //Salida 95°C
    pinMode(10, OUTPUT);  //Salida 65°C
    Serial.begin(9600);

    Output1 = 0;
    Output2 = 0;
  }

  void loop() {

    if (analogRead(inicio) >= 512){
      if (comienzo == 0){
        comienzo = 1;
        InicioTiempo = millis();
      }
      else{
        //ciclo = ciclo+1;
        //Serial.println(ciclo);
```

```
        TomaMedidaTiempo = millis();
        DiferenciaTiempo = TomaMedidaTiempo - InicioTiempo;

        LlamadaControlTemp = millis();
        ControlTemp();
        SalidaControlTemp = millis();
        TiempoControlTemp = SalidaControlTemp - LlamadaControlTemp;
        Serial.print("Tiempo temperatura: ");
        Serial.println(TiempoControlTemp);

        InicioIf = millis();
  //                                                                    if(D
iferenciaTiempo >= 10000){
  //                                                                    moto
rcillo.step(-1);
  //    MovimientoMotor = millis()  ;
  //
  //                                                                    }
   FinalIf = millis();
   TiempoIf = FinalIf - InicioIf;
   TiempoEjecucion = FinalIf - LlamadaControlTemp;
   Serial.print("Temperatura + motor: ");
   Serial.println(TiempoEjecucion);
   Serial.println(TiempoIf);
  //                                                                    else
if(DiferenciaTiempo <= 60000 && DiferenciaTiempo >= 41000){
  //
                                                                        moto
rcillo.step(1);
  //    motorcillo.step(-2);
  //                                                                    }


        if(DiferenciaTiempo >= INTERVALO_CALENTADO){
          //digitalWrite(11, LOW); //Después de 10 minutos de calentado,
apagamos el microcalentador.
          myPID95.SetMode(MANUAL);
          myPID65.SetMode(MANUAL);
          Output65 = 0;
          Output95 = 0;

        }

  }

  }
  }
```

motorTemp_Pert.ino

Código empleado para replicar la perturbación introducida por la expansión del aire, es decir, para replicar el

problema a resolver.

```
#include <PID_v1.h>
#include <Stepper.h>

//----------------Stepper
float V1 = 20;           //Volumen impulsión (µL)
float radio = 2.25;      //radio del émbolo
float tiempo = 4;        //tiempo en segundos
float area = 3.141592654*radio*radio;
float tiempospeed = 60/tiempo;
float L1 = V1/area;
//int pasos = round(L1/0.033);
int pasos = round(3/0.033);

int t65 = 15000;          //tiempo total (en ms) que requiere la parte
de 65°C
int t95 = 15000;          //tiempo total (en ms) que requiere la parte
de 95°C
int t65i = 0;            //marca de tiempo de cuándo comienza la parte
de 65°C en el ciclo actual
int t95i = 0;            //marca de timepo de cuándo comienza la parte
de 95°C en el ciclo actual
int estado = 1;           //variable que marca el estado de la rutina
del proceso
int ciclos = 0;           //contador del número de ciclos que se han
producido
int t95f = 0;
int t65f = 0;

Stepper motorcillo(24, 8, 9); //Stepper nombre motor (número de pasos
por vuelta, pins de control)

//---------------Control temperatua

double Ref, Input, Output, Output1, Output2;
double Ref95, Input95, Output95;
double Ref65, Input65, Output65;
double Kp=0.01,Ki=30,Kd=50;
float alfa = 0.05;
float error = 0;
float error95 = 0;
float error65 = 0;
float Temp95 = 106;
float Temp65 = 65;
int R95 = 1200;
int R65 = R95;
float beta95 = 3445.0; //Mismos valores que para beta95, ya que son
muy próximos y no existe gran diferencia entre beta80 y beta95
float beta65 = beta95;
float To = 298.15;
int Vcc = 5;
int Ro = 10000;
int ciclo = 0;
```

```
  PID myPID95(&Input95, &Output95, &Ref95, Kp, Ki, 200, DIRECT);
  PID myPID65(&Input65, &Output65, &Ref65, Kp, Ki, 200, DIRECT);

  unsigned long InicioTiempo = 0;
  unsigned long TomaMedidaTiempo = 0;
  unsigned long DiferenciaTiempo;
  //unsigned long LlamadaControlTemp = 0;
  //unsigned long SalidaControlTemp = 0;
  //unsigned long TiempoControlTemp = 0;
  //unsigned long MovimientoMotor = 0;
  //unsigned long TiempoEjecucion = 0;
  //unsigned long InicioIf = 0;
  //unsigned long FinalIf = 0;
  //unsigned long TiempoIf = 0;
  unsigned long INTERVALO_CALENTADO = 600000; //Si 1 min son 60000 milisegundos, 10 min serán 600000 milisegundos. Ponemos el tiempo en milisegundos para trabajar con la función "millis()"
  int comienzo = 0;
  const int inicio = A0;
  double medida95;
  double medida95f = 0;
  double medida65;
  double medida65f = 0;

  void ControlTemp(){     //Función que mide la temperatura en cada calentador y se encarga de ejecutar el control correspondiente
      float ADC95 = analogRead(A1); //Leemos NTC del lado de 95°C
      //Calculos para 95°C
      float Vout95=ADC95*5/1023;
      float VNTC95=Vout95;
      float RNTC95 = (VNTC95*R95)/(Vcc-VNTC95);
      float alfa95 = (1000/To)+((1000/beta95)*(log(RNTC95/Ro)));
      float T95 = 1000/alfa95 - 273.15;

      medida95 = T95; //Medida va a guardar el valor de temperatura calculado previamente en unidades de temperatura y no de ADC
      medida95f = medida95*alfa + (1-alfa)*medida95f;
      Input95 = medida95f;
      error95 = Ref95 - medida95f;

      myPID95.Compute();
      //delay(10);
      analogWrite(10, Output95);

      float ADC65 = analogRead(A2); //Leemos NTC del lado de 65°C
      //Calculos para 65°C
      float Vout65=ADC65*5/1023;
      float VNTC65=Vout65;
      float RNTC65 = (VNTC65*R65)/(Vcc-VNTC65);
      float alfa65 = (1000/To)+((1000/beta65)*(log(RNTC65/Ro)));
      float T65 = 1000/alfa65 - 273.15;

      medida65 = T65; //Medida va a guardar el valor de temperatura calculado previamente en unidades de temperatura y no de ADC
      medida65f = medida65*alfa + (1-alfa)*medida65f;
      Input65 = medida65f;
```

```
    error65 = Ref65 - medida65f;

    myPID65.Compute();
    //delay(10);
    analogWrite(11, Output65);

      Serial.print("NTC_95:");
      Serial.print(medida95f);
      Serial.print(",");
      Serial.print("Referencia_95:");
      Serial.print(Ref95);
      Serial.print(",");
      Serial.print("NTC_65:");
      Serial.print(medida65f);
      Serial.print(",");
      Serial.print("Referencia_65:");
      Serial.println(Ref65);
}

void setup() {

  // Velocidad del motor en RPM
  motorcillo.setSpeed(tiempospeed);
  Serial.begin(9600);
  ciclos = 0;

  comienzo = 0;
  myPID95.SetMode(AUTOMATIC);
  Ref95 = Temp95;

  myPID65.SetMode(AUTOMATIC);
  Ref65 = Temp65;

  pinMode(11, OUTPUT);  //Salida 95°C
  pinMode(10, OUTPUT);  //Salida 65°C
  Serial.begin(9600);

  Output1 = 0;
  Output2 = 0;
}

void loop() {

  if (analogRead(inicio) <= 512){
    if (comienzo == 0){
      comienzo = 1;
      InicioTiempo = millis();
    }
//    else{
//      if(analogRead(inicio) >= 512){
//        Ref65 = 65;
//      }

      ControlTemp();
```

```
  //      if(DiferenciaTiempo >= INTERVALO_CALENTADO){
  //       //digitalWrite(11, LOW); //Después de 10 minutos de calentado,
apagamos el microcalentador.
  //         myPID95.SetMode(MANUAL);
  //         myPID65.SetMode(MANUAL);
  //         Output65 = 0;
  //         Output95 = 0;
  //
  //      }

  }

  }
  //}
```

motorFotodiodo.ino

Código empleado para comprobar emplear el control del motor en base a un único fotodiodo.

```
  #include <Stepper.h>

  float V1 = 20; //Volumen impulsión (µL)
  float radio = 2.25; //radio del émbolo
  float tiempo = 4; // tiempo en segundos
  float area = 3.141592654*radio*radio;
  float tiempospeed = 60/tiempo;
  float L1 = V1/area;
  int pasos = round(L1/0.033);

  const int entrada = A1; //entrada donde se conecta el fotodiodo
  float umbral = 1; //el valor medido real (aprox) es 0.71 cuando el
líquido cubre el fotodiodo
                    //100 umbral de ser tapado ~ 300 umbral de aire //
150 ~ 600 en el lab

  Stepper motorcillo(pasos, 8, 9); //Stepper nombre motor (número de
pasos por vuelta, pins de control)

  void setup()
  {
    // Velocidad del motor en RPM
    motorcillo.setSpeed(tiempospeed);
    Serial.begin(9600);
  }

  void loop()
  {

    //motorcillo.step(-pasos); //aseguramos que el motor esté atrás del
todo
```

```
    Serial.println(analogRead(entrada));

    while(analogRead(entrada) <= 170){ //mientras el diodo esté tapado
movemos hacia detrás
          motorcillo.step(-1);
          delay(100);//ajustar el numero de pasos por bucle para la
precisión deseada
          Serial.println("volvemos patrás");
    }

    Serial.println("no hay líquido"); //el líquido está donde el
fotodiodo
    //delay(5000); //Pequeña pausa

    //motorcillo.step(-pasos);

  }
```

Motor_Fotodiodo_Doble.ino

Código empleado para el control del motor, ya usando los dos fotodiodos con el fin de realizar el ciclo de movimiento del fluido completo, es decir, implementando la máquina de estados.

```
  #include <Stepper.h>

  float V1 = 20;           //Volumen impulsión (µL)
  float radio = 2.25;      //radio del émbolo
  float tiempo = 4;        //tiempo en segundos
  float area = 3.141592654*radio*radio;
  float tiempospeed = 60/tiempo;
  float L1 = V1/area;
  //int pasos = round(L1/0.033);
  int pasos = round(3/0.033);

  const int entrada1 = A1; //entrada donde se conecta el fotodiodo
  const int entrada2 = A2; //entrada donde se conecta el fotodiodo
  int foto1;               //variable para la entrada del fotodiodo1
  int foto2;               //variable para la entrada del fotodiodo2
  const int lleno = 4;     //variable que indique que se ha inyectado el
líquido en el LoC
  //float umbral1 = 1;     //el valor medido real (aprox) es 0.71 cuando
el líquido cubre el fotodiodo
  //float umbral2 = 1;      //100 umbral de ser tapado ~ 300 umbral de
aire // 150 ~ 600 en el lab
  int t65 = 15000;         //tiempo total (en ms) que requiere la parte
de 65°C
  int t95 = 15000;         //tiempo total (en ms) que requiere la parte
de 95°C
  int t65i = 0;           //marca de tiempo de cuándo comienza la parte
de 65°C en el ciclo actual
  int t95i = 0;           //marca de timepo de cuándo comienza la parte
de 95°C en el ciclo actual
```

```
  int estado = 1;           //variable que marca el estado de la rutina
del proceso
  int ciclos = 0;           //contador del número de ciclos que se han
producido
  int t95f = 0;
  int t65f = 0;


  Stepper motorcillo(24, 8, 9); //Stepper nombre motor (número de pasos
por vuelta, pins de control)

  void setup()
  {
    // Velocidad del motor en RPM
    motorcillo.setSpeed(tiempospeed);
    Serial.begin(9600);
    ciclos = -1;
    estado = 1;
  }

  void loop()
  {
    switch (estado) {
      case 1:      //Esperamos que se inyecte la muestra
        Serial.println("inicio");
        if (digitalRead(lleno) == HIGH){
          estado = 2; //Una vez dentro, comienza el proceso, avance95
        }
        else{
          estado = 1; //volvemos al inicio si no se ha inyectado la
muestra
        }
        break;

      case 2:    //Movemos el líquido a la zona de 95°C
        Serial.println("avance95");
        foto1 = analogRead(entrada1);
        Serial.println(foto1);
          while (analogRead(entrada1) >= 180){
            foto1 = analogRead(entrada1);
            Serial.println(foto1);
            Serial.println("motor avanza");
            motorcillo.step(1);
            delay(200); //PARA DEPURAR Y ENCONTRAR ERRORES
          }
          Serial.println("motor retrocede 1");
          motorcillo.step(-1);
          t95f = millis();    //Una vez llegado al fotodiodo que hace de
"fin de carrera", iniciamos
          t95i = t95f;
          estado = 3;        //el "timer" para el tiempo a 95°C (parte95)
          break;

      case 3:
        Serial.println("parte95");
        foto1 = analogRead(entrada1);
        Serial.println(foto1);
```

```
        t95f = millis();
          if ((t95f - t95i) <= t95){        //Mientras no se cumpla el
tiempo nos aseguramos de que el
            delay(200);
            Serial.println((t95f-t95i));
            foto1 = analogRead(entrada1);
            Serial.println(foto1);
            if(analogRead(entrada1) <= 180){  //líquido no sobrepasa la
línea del fotodiodo
              motorcillo.step(-1);
              Serial.println("motor retrocede 2");
            }
          }
          else {
          estado = 4; //vuelta65 ---- REVISAR
          break;
          }
          break;

      case 4:    //Comenzamos el movimiento a la parte de 65°C
        Serial.println("vuelta65");
        foto2 = analogRead(entrada2);
        Serial.println(foto2);
          while(analogRead(entrada2) >= 180){ //Movemos el líquido hasta
sobrepasar el fotodiodo
            delay(200);
            foto2 = analogRead(entrada2);
            Serial.println(foto2);
            Serial.println("motor retrocede 3");
            motorcillo.step(-1);
          }
          motorcillo.step(-1);
          estado = 5; //avance65
          break;

      case 5:  //Terminamos de avanzar el líquido hasta que sobrepase
del todo el fotodiodo
        Serial.println("avance65");
          while(analogRead(entrada2) <= 180){
            delay(200);
            foto2 = analogRead(entrada2);
            Serial.println(foto2);
            motorcillo.step(-1);
            Serial.println("motor retrocede 4");
          }
          motorcillo.step(-1);
          Serial.println("motor retrocede 5");
          ciclos = ciclos+1;
          Serial.print("N° CICLOS: ");
          Serial.println(ciclos);
          if(ciclos == 3){  //Si se ha cumplido el n° de ciclos, pasamos
al estado final
            estado = 7; //Fin
          }
          else{             //En caso de no haber completado los ciclos,
iniciamos el "timer" de 65°C
```

```
            t65f = millis();
            t65i = t65f;
            estado = 6; //parte65
          }
          break;

      case 6:
        Serial.println("parte65");
        t65f = millis();
          if ((t65f-t65i) <= t65){      //Esperamos a que se cumpla el
tiempo mientras vigilamos
            delay(200);
            Serial.println((t65f-t65i));
            foto2 = analogRead(entrada2);
            Serial.println(foto2);
            if(analogRead(entrada2)  <=  180){  //que  el  líquido  no
sobrepase el límite del fotodiodo
              foto2 = analogRead(entrada2);
              Serial.println(foto2);
              Serial.println("motor retrocede 6");
              motorcillo.step(-1);
            }
          }
          else{
            estado = 2;  //Repetimos el proceso (avance95)
            break;
          }
          break;

      case 7:             //Estado final, ya ha acabado el proceso y se
puede retirar la muestra
        Serial.println("Fin del proceso");
        estado = 7; //volvemos a fin
        break;

      default:
        Serial.println("Esperando");
        break;
    }

  }
```

foto_temp.ino

Código empleado para comprobar que no había ningún problema de funcionamiento al emplear los fotodiodos y los calentadores al mismo tiempo, se corresponde con el apartado 8.3, integración de ambos bloques.

```
#include <PID_v1.h>
#include <Stepper.h>

//----------------Stepper
float V1 = 20;           //Volumen impulsión (µL)
float radio = 2.25;      //radio del émbolo
float tiempo = 4;        //tiempo en segundos
float area = 3.141592654*radio*radio;
float tiempospeed = 60/tiempo;
float L1 = V1/area;
//int pasos = round(L1/0.033);
int pasos = round(3/0.033);

int t65 = 15000;          //tiempo total (en ms) que requiere la parte
de 65°C
int t95 = 15000;          //tiempo total (en ms) que requiere la parte
de 95°C
int t65i = 0;            //marca de tiempo de cuándo comienza la parte
de 65°C en el ciclo actual
int t95i = 0;            //marca de timepo de cuándo comienza la parte
de 95°C en el ciclo actual
int estado = 1;           //variable que marca el estado de la rutina
del proceso
int ciclos = 0;           //contador del número de ciclos que se han
producido
int t95f = 0;
int t65f = 0;

Stepper motorcillo(24, 8, 9); //Stepper nombre motor (número de pasos
por vuelta, pins de control)

const int diodo1 = A4;
const int diodo2 = A5;
int valorDiodo1;
int valorDiodo2;

//---------------Control temperatua

double Ref, Input, Output, Output1, Output2;
double Ref95, Input95, Output95;
double Ref65, Input65, Output65;
double Kp=0.01,Ki=30,Kd=50;
float alfa = 0.05;
float error = 0;
float error95 = 0;
float error65 = 0;
float Temp95 = 106;
float Temp65 = 65;
```

```
  int R95 = 1200;
  int R65 = R95;
  float beta95 = 3445.0; //Mismos valores que para beta95, ya que son
muy próximos y no existe gran diferencia entre beta80 y beta95
  float beta65 = beta95;
  float To = 298.15;
  int Vcc = 5;
  int Ro = 10000;
  int ciclo = 0;

  PID myPID95(&Input95, &Output95, &Ref95, Kp, Ki, 200, DIRECT);
  PID myPID65(&Input65, &Output65, &Ref65, Kp, Ki, 200, DIRECT);

  unsigned long InicioTiempo = 0;
  unsigned long TomaMedidaTiempo = 0;
  unsigned long DiferenciaTiempo;
  //unsigned long LlamadaControlTemp = 0;
  //unsigned long SalidaControlTemp = 0;
  //unsigned long TiempoControlTemp = 0;
  //unsigned long MovimientoMotor = 0;
  //unsigned long TiempoEjecucion = 0;
  //unsigned long InicioIf = 0;
  //unsigned long FinalIf = 0;
  //unsigned long TiempoIf = 0;
  unsigned long INTERVALO_CALENTADO = 600000; //Si 1 min son 60000
milisegundos, 10 min serán 600000 milisegundos. Ponemos el tiempo en
milisegundos para trabajar con la función "millis()"
  int comienzo = 0;
  const int inicio = A0;
  double medida95;
  double medida95f = 0;
  double medida65;
  double medida65f = 0;

  void ControlTemp(){     //Función que mide la temperatura en cada
calentador y se encarga de ejecutar el control correspondiente
      float ADC95 = analogRead(A1); //Leemos NTC del lado de 95°C
      //Calculos para 95°C
      float Vout95=ADC95*5/1023;
      float VNTC95=Vout95;
      float RNTC95 = (VNTC95*R95)/(Vcc-VNTC95);
      float alfa95 = (1000/To)+((1000/beta95)*(log(RNTC95/Ro)));
      float T95 = 1000/alfa95 - 273.15;

      medida95 = T95; //Medida va a guardar el valor de temperatura
calculado previamente en unidades de temperatura y no de ADC
      medida95f = medida95*alfa + (1-alfa)*medida95f;
      Input95 = medida95f;
      error95 = Ref95 - medida95f;

      myPID95.Compute();
      //delay(10);
      analogWrite(10, Output95);

      float ADC65 = analogRead(A2); //Leemos NTC del lado de 65°C
      //Calculos para 65°C
      float Vout65=ADC65*5/1023;
```

```
      float VNTC65=Vout65;
      float RNTC65 = (VNTC65*R65)/(Vcc-VNTC65);
      float alfa65 = (1000/To)+((1000/beta65)*(log(RNTC65/Ro)));
      float T65 = 1000/alfa65 - 273.15;

      medida65 = T65; //Medida va a guardar el valor de temperatura
calculado previamente en unidades de temperatura y no de ADC
      medida65f = medida65*alfa + (1-alfa)*medida65f;
      Input65 = medida65f;
      error65 = Ref65 - medida65f;

      myPID65.Compute();
      //delay(10);
      analogWrite(11, Output65);

        Serial.print("NTC_95:");
        Serial.print(medida95f*10);
        Serial.print(",");
        Serial.print("Referencia_95:");
        Serial.print(Ref95*10);
        Serial.print(",");
        Serial.print("NTC_65:");
        Serial.print(medida65f*10);
        Serial.print(",");
        Serial.print("Referencia_65:");
        Serial.print(Ref65*10);
        Serial.print(",");
  }

  void setup() {

    // Velocidad del motor en RPM
    motorcillo.setSpeed(tiempospeed);
    Serial.begin(9600);
    ciclos = 0;

    comienzo = 0;
    myPID95.SetMode(AUTOMATIC);
    Ref95 = Temp95;

    myPID65.SetMode(AUTOMATIC);
    Ref65 = Temp65;

    pinMode(11, OUTPUT);  //Salida 95°C
    pinMode(10, OUTPUT);  //Salida 65°C
    Serial.begin(9600);

    Output1 = 0;
    Output2 = 0;
  }

  void loop() {

    if (analogRead(inicio) <= 512){
      if (comienzo == 0){
        comienzo = 1;
```

```
      InicioTiempo = millis();
    }
//    else{
//      if(analogRead(inicio) >= 512){
//        Ref65 = 65;
//      }

      ControlTemp();
      valorDiodo1 = analogRead(diodo1);

  valorDiodo2 = analogRead(diodo2);
  Serial.print("Foto_95:");
  Serial.print(valorDiodo1);
  Serial.print(",");
  Serial.print("Foto_65:");
  Serial.println(valorDiodo2);




//      if(DiferenciaTiempo >= INTERVALO_CALENTADO){
//       //digitalWrite(11, LOW); //Después de 10 minutos de calentado,
apagamos el microcalentador.
//        myPID95.SetMode(MANUAL);
//        myPID65.SetMode(MANUAL);
//        Output65 = 0;
//        Output95 = 0;
//
//      }

}

}
//}
```

Motor_Temp_Fotodiodo_FINAL.ino

Código final en el que se han integrado los tres bloques, completando así el sistema, y con el que se llegó a la finalización del presente trabajo.

```
#include <Stepper.h>
#include <PID_v1.h>

//--------------- DEFINICIONES MOTOR ----------------
float V1 = 20;           //Volumen impulsión (µL)
float radio = 2.25;      //radio del émbolo
float tiempo = 4;        //tiempo en segundos
float area = 3.141592654*radio*radio;
float tiempospeed = 60/tiempo;
float L1 = V1/area;
//int pasos = round(L1/0.033);
int pasos = round(3/0.033);
```

```
 const int entrada1 = A4; //entrada donde se conecta el fotodiodo de la
parte de 95°C
 const int entrada2 = A5; //entrada donde se conecta el fotodiodo de la
parte de 65°C
 int foto1;               //variable para la entrada del fotodiodo1
 int foto2;               //variable para la entrada del fotodiodo2
 const int lleno = A0;      //variable que indique que se ha inyectado
el líquido en el LoC
 //float umbral1 = 1;     //el valor medido real (aprox) es 0.71 cuando
el líquido cubre el fotodiodo
 //float umbral2 = 1;      //100 umbral de ser tapado ~ 300 umbral de
aire // 150 ~ 600 en el lab
 int t65 = 2000;          //tiempo total (en ms) que requiere la parte
de 65°C
 int t95 = 2000;          //tiempo total (en ms) que requiere la parte
de 95°C
 int t65i = 0;            //marca de tiempo de cuándo comienza la parte
de 65°C en el ciclo actual
 int t95i = 0;            //marca de timepo de cuándo comienza la parte
de 95°C en el ciclo actual
 int estado = 1;          //variable que marca el estado de la rutina
del proceso
 int ciclos = 0;           //contador del número de ciclos que se han
producido
 int t95f = 0;
 int t65f = 0;

 Stepper motorcillo(24, 8, 9); //Stepper nombre motor (número de pasos
por vuelta, pins de control)

 //--------------- DEFINICIONES CONTROL TEMPERATURA ----------------
 double Ref, Input, Output, Output1, Output2;
 double Ref95, Input95, Output95;
 double Ref65, Input65, Output65;
 double Kp=0.001,Ki=30,Kd=50;
 float alfa = 0.05;
 float error = 0;
 float error95 = 0;
 float error65 = 0;
 float Temp95 = 106;
 float Temp65 = 65;
 int R95 = 1200;
 int R65 = R95;
 float beta95 = 3445.0;
 float beta65 = beta95;
 float To = 298.15;
 int Vcc = 5;
 int Ro = 10000;
 int ciclo = 0;

 PID myPID95(&Input95, &Output95, &Ref95, Kp, Ki, 200, DIRECT);
 PID myPID65(&Input65, &Output65, &Ref65, Kp, Ki, 200, DIRECT);

 unsigned long InicioTiempo = 0;
 unsigned long TomaMedidaTiempo = 0;
```

```
  unsigned long DiferenciaTiempo;
  unsigned long INTERVALO_CALENTADO = 600000; //10min en ms para usar
"millis()"
  int comienzo = 0;
  //const int inicio = A0;
  double medida95;
  double medida95f = 0;
  double medida65;
  double medida65f = 0;

  void plotFotodiodos(int adelante){
    int n = 0;
      float ADC95 = analogRead(A1); //Leemos NTC del lado de 95°C
      //Calculos para 95°C
      float Vout95=ADC95*5/1023;
      float VNTC95=Vout95;
      float RNTC95 = (VNTC95*R95)/(Vcc-VNTC95);
      float alfa95 = (1000/To)+((1000/beta95)*(log(RNTC95/Ro)));
      float T95 = 1000/alfa95 - 273.15;

      medida95 = T95; //Medida va a guardar el valor de temperatura
calculado previamente en unidades de temperatura y no de ADC
      medida95f = medida95*alfa + (1-alfa)*medida95f;
      Input95 = medida95f;
      Serial.print("Ref_95: ");
      Serial.print(106);
      Serial.print(",");
      Serial.print("Temp_95: ");
      Serial.print(Input95);
      Serial.print(",");

      float ADC65 = analogRead(A2); //Leemos NTC del lado de 65°C
      //Calculos para 65°C
      float Vout65=ADC65*5/1023;
      float VNTC65=Vout65;
      float RNTC65 = (VNTC65*R65)/(Vcc-VNTC65);
      float alfa65 = (1000/To)+((1000/beta65)*(log(RNTC65/Ro)));
      float T65 = 1000/alfa65 - 273.15;

      medida65 = T65; //Medida va a guardar el valor de temperatura
calculado previamente en unidades de temperatura y no de ADC
      medida65f = medida65*alfa + (1-alfa)*medida65f;
      Input65 = medida65f;
      Serial.print("Ref_95: ");
      Serial.print(65);
      Serial.print(",");
      Serial.print("Temp_65: ");
      Serial.print(Input65);
      Serial.print(",");

    float entrada1 = analogRead(A4);
    Serial.print("Fotodiodo_95: ");
    Serial.print(entrada1);
    Serial.print(",");

    float entrada2 = analogRead(A5);
    analogRead(entrada2);
```

```
    Serial.print("Fotodiodo_65: ");
    Serial.print(entrada2);
    Serial.print(",");
    Serial.print("Motor: ");
    if(adelante == 1){
      n = 0;
    }
    else if(adelante == -1){
      n = -200;
    }
    else{
      n = -100;

  }
    Serial.println(1200+n);
  }
  //Definimos función que toma medida de la temperatura en ambos
calentadores y calcula los PIDs correspondientes para cada salida
  void ControlTemp(){     //Función que mide la temperatura en cada
calentador y se encarga de ejecutar el control correspondiente
      float ADC95 = analogRead(A1); //Leemos NTC del lado de 95°C
      //Calculos para 95°C
      float Vout95=ADC95*5/1023;
      float VNTC95=Vout95;
      float RNTC95 = (VNTC95*R95)/(Vcc-VNTC95);
      float alfa95 = (1000/To)+((1000/beta95)*(log(RNTC95/Ro)));
      float T95 = 1000/alfa95 - 273.15;

      medida95 = T95; //Medida va a guardar el valor de temperatura
calculado previamente en unidades de temperatura y no de ADC
      medida95f = medida95*alfa + (1-alfa)*medida95f;
      Input95 = medida95f;
      error95 = Ref95 - medida95f;

      myPID95.Compute();
      //delay(10);
      analogWrite(10, Output95);

      float ADC65 = analogRead(A2); //Leemos NTC del lado de 65°C
      //Calculos para 65°C
      float Vout65=ADC65*5/1023;
      float VNTC65=Vout65;
      float RNTC65 = (VNTC65*R65)/(Vcc-VNTC65);
      float alfa65 = (1000/To)+((1000/beta65)*(log(RNTC65/Ro)));
      float T65 = 1000/alfa65 - 273.15;

      medida65 = T65; //Medida va a guardar el valor de temperatura
calculado previamente en unidades de temperatura y no de ADC
      medida65f = medida65*alfa + (1-alfa)*medida65f;
      Input65 = medida65f;
      error65 = Ref65 - medida65f;

      myPID65.Compute();
      //delay(10);
      analogWrite(11, Output65);
```

```
//      Serial.print("NTC_95:");
//      Serial.print(medida95f);
//      Serial.print(",");
//      Serial.print("Referencia_95:");
//      Serial.print(Ref95);
//      Serial.print(",");
//      Serial.print("NTC_65:");
//      Serial.print(medida65f);
//      Serial.print(",");
//      Serial.print("Referencia_65:");
//      Serial.println(Ref65);
}

void setup()
{
  // Velocidad del motor en RPM
  motorcillo.setSpeed(tiempospeed);
  Serial.begin(9600);
  ciclos = -1; //Número de ciclos del proceso completo
  estado = 1;

  comienzo = 0; //Iniciamos los PIDs correspondientes a los controles
de temp.
  myPID95.SetMode(AUTOMATIC);
  Ref95 = Temp95;
  myPID65.SetMode(AUTOMATIC);
  Ref65 = Temp65;

  pinMode(11, OUTPUT);  //Salida 65°C
  pinMode(10, OUTPUT);  //Salida 95°C

  Output1 = 0;
  Output2 = 0;
}

void loop()
{


  switch (estado) {
    case 1:      //Esperamos que se inyecte la muestra
      //Serial.println("inicio");
      ControlTemp();
      plotFotodiodos(1);
      //motorcillo.step(-1);
      if (analogRead(lleno) >= 100){
        estado = 2; //Una vez dentro, comienza el proceso -> avance95
        InicioTiempo = millis();
      }
      else{
        estado = 1; //volvemos al inicio si no se ha inyectado la
muestra
      }
      break;

    case 2:    //Movemos el líquido a la zona de 95°C
      //Serial.println("avance95");
```

```
        //foto1 = analogRead(entrada1); //Esto es simplemente para debug
        //Serial.println(foto1);
        ControlTemp(); //debatible que sea necesario
          while (analogRead(entrada1) >= 700){
            //foto1 = analogRead(entrada1); //Esto es simplemente para
debug
            //Serial.println(foto1);         //Esto es simplemente para
debug
            //Serial.println("motor avanza"); //Esto es simplemente para
debug
            plotFotodiodos(1);
            motorcillo.step(1);

            delay(10); //PARA DEPURAR Y ENCONTRAR ERRORES
          }
          //Serial.println("motor retrocede 1");
          plotFotodiodos(-1);
          motorcillo.step(-1);
          t95f = millis();    //Una vez llegado al fotodiodo que hace de
"fin de carrera", iniciamos
          t95i = t95f;
          estado = 3;           //el "timer" para el tiempo a 95°C ->
parte95
          break;

      case 3:   //PARTE 95°c
        //Serial.println("parte95");
        //foto1 = analogRead(entrada1);
        //Serial.println(foto1);
        t95f = millis();
        ControlTemp();
          if ((t95f - t95i) <= t95){        //Hasta que no se cumpla el
tiempo nos aseguramos de que el
            //Serial.println((t95f-t95i));
            //foto1 = analogRead(entrada1);
            //Serial.println(foto1);
            if(analogRead(entrada1) <= 700){  //líquido no sobrepasa la
línea del fotodiodo
              plotFotodiodos(-1);
              motorcillo.step(-1);
              //Serial.println("motor retrocede 2");
            }
          }
          else {
          estado = 4; //vuelta65 ---- REVISAR
          break;
          }
          delay(10); //PARA DEPURAR Y ENCONTRAR ERRORES
          break;

      case 4:    //Comenzamos el movimiento a la parte de 65°C
        //Serial.println("vuelta65");
        //foto2 = analogRead(entrada2);
        //Serial.println(foto2);
        ControlTemp();  //discutible que sea necesario
```

```
          while(analogRead(entrada2) >= 700){ //Movemos el líquido hasta
sobrepasar el fotodiodo
            //foto2 = analogRead(entrada2);
            //Serial.println(foto2);
            //Serial.println("motor retrocede 3");
            plotFotodiodos(-1);
            motorcillo.step(-1);
            delay(10);
          }
          plotFotodiodos(-1);
          motorcillo.step(-2);
          estado = 5; //avance65
          break;

      case 5:  //Terminamos de avanzar el líquido hasta que sobrepase
del todo el fotodiodo
        //Serial.println("avance65");
        ControlTemp();  //debatible que sea necesario
          while(analogRead(entrada2) <= 700){
            //foto2 = analogRead(entrada2);
            //Serial.println(foto2);
            plotFotodiodos(-1);
            motorcillo.step(-1);
            delay(10);
            //Serial.println("motor retrocede 4");
          }
          plotFotodiodos(-1);
          motorcillo.step(-4);
          //Serial.println("motor retrocede 5");
          ciclos = ciclos+1;
          //Serial.print("N° CICLOS: ");
          //Serial.println(ciclos);
          if(ciclos == 1){  //Si se ha cumplido el n° de ciclos, pasamos
al estado final
            estado = 7; //Fin
          }
          else{             //En caso de no haber completado los ciclos,
iniciamos el "timer" de 65°C
            t65f = millis();
            t65i = t65f;
            estado = 6; //parte65
          }
          break;

      case 6:
        //Serial.println("parte65");
        t65f = millis();
        ControlTemp();
          if ((t65f-t65i) <= t65){      //Esperamos a que se cumpla el
tiempo mientras vigilamos
            //Serial.println((t65f-t65i));
            //foto2 = analogRead(entrada2);
            //Serial.println(foto2);
            plotFotodiodos(0);
            if(analogRead(entrada2)  <=  700){  //que  el  líquido  no
sobrepase el límite del fotodiodo
              plotFotodiodos(-1);
```

```
              motorcillo.step(-1);
              //Serial.println("motor retrocede 6");
            }
          }
          else{
            estado = 2;  //Repetimos el proceso (avance95)
            break;
          }
          delay(10);
          break;

      case 7:              //Estado final, ya ha acabado el proceso y se
puede retirar la muestra
        //Serial.println("Fin del proceso");
        estado = 7; //volvemos a fin
        myPID95.SetMode(MANUAL);
        myPID65.SetMode(MANUAL);
        Output1 = 0;
        Output2 = 0;
        plotFotodiodos(0);
        break;

      default:
        //Serial.println("Esperando");
        break;
    }

  }
```