UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO Y DESARROLLO DE UN LABORATORIO REMOTO DE PRÁCTICAS DE ELECTRÓNICA DIGITAL

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Luis Alfonso Molina

TUTORIZADO Y COTUTORIZADO POR

Francisco Javier Ibáñez Civera

Javier Monreal Trigo

CURSO ACADÉMICO: 2019/2020

ESTRUCTURA DEL DOCUMENTO

I. MEMORIA
II. PLANOS
III. PLIEGO DE CONDICIONES
IV. PRESUPUESTO

I. MEMORIA DEL PROYECTO

Resumen

A raíz de la crisis sanitaria, las herramientas educativas que puedan utilizarse en entornos semipresenciales son de particular interés. En este trabajo se ha desarrollado una plataforma de laboratorio remoto fácilmente reproducible basado en software y hardware de uso libre en el que el estudiante, mediante conexión remota, puede programar un microcontrolador para controlar periféricos de medición de temperatura, una pantalla, LEDs, ver el comportamiento de los mismos mediante webcam, así como leer las entradas de potenciómetros y botones virtuales emulados físicamente en el microcontrolador controlados mediante un interfaz gráfica de usuario. La plataforma cuenta con un bus I2C accesible para futuras ampliaciones modulares.

Palabras clave: laboratorio remoto, electrónica digital, interfaz gráfica de usuario, enseñanza semipresencial.

Abstract

As a result of the health crisis, educational tools which can be used in semi-presential environments are of particular interest. In this project, an easily reproducible remote laboratory platform based on free-use software and hardware has been developed. In this platform, through remote connection, the student can program a microcontroller to control temperature measurement peripherals, a screen, LEDs, watch the behavior of all of them through webcam, as well as reading the inputs of potentiometers and virtual buttons physically emulated in the microcontroller controlled by a graphical user interface. The platform also has an accessible I2C bus for future modular extensions.

Keywords: remote laboratory, digital electronics, graphic user interface, semi-presential teaching.

Resum

Arrel de la crisi sanitària, les ferramentes educatives que poden utilitzar-se en entorns semipresencials són de particular interès. En aquest treball s'ha desenvolupat una plataforma de laboratori remot fàcilment reproduïble basada en software i hardware de lliure us en el que l'estudiant, mitjançant connexió remota, pot programar un microcontrolador per controlar perifèrics de mesura de temperatura, una pantalla, LEDs, vore el comportament dels mateixos mitjançant una webcam, així com llegir les entrades de potenciòmetres i botons virtuals emulats físicament en el microcontrolador controlats mitjançant una interfície gràfica d'usuari. La plataforma compta amb un bus I2C accessible per a futures ampliacions modulars.

Paraules clau: laboratori remot, electrònica digital, interfície gràfica d'usuari, ensenyament semipresencial.

Índice de contenido

Índice de figuras

Índice de fragmentos de código

1. Introducción

1.1 Contexto

Con el avance de las tecnologías de la información y comunicación ciertos obstáculos y dificultades infranqueables hasta hace poco se han diluido o superado, así como se han abierto múltiples posibilidades en la mayoría de ámbitos. Sin duda, la educación es uno de los ámbitos más beneficiados por estos avances, que han repercutido con una inmensa cantidad de herramientas: desde el apoyo con recursos multimedia en las presentaciones orales hasta los juegos pedagógicos (desde simples test con un entorno de puntuación en línea hasta verdaderos videojuegos que facilitan el aprendizaje), pasando por un sinfín de utilidades como los vídeos de apoyo, foros en línea de resolución de dudas o laboratorios virtuales que facilitan la comprensión en el aprendizaje y la eficiencia en la enseñanza.

Una de estas ventajas es la posibilidad de realizar tanto clases teóricas como prácticas sin la necesidad de presencialidad. Esta posibilidad de educación no presencial lleva varios años en auge, y ha posibilitado la creación de cursos y títulos completamente en línea, revolucionando la educación a distancia. También es relevante mencionar su combinación con la enseñanza presencial, como es el caso de la metodología de enseñanza inversa, consistente en la impartición de las sesiones de teoría básicas de forma asíncrona y telemática, reservando la presencialidad para las más avanzadas o la resolución de problemas más complejos y prácticas de laboratorio [1].

Debido a los hechos acaecidos desde marzo de 2020, con una crisis sanitaria debido a la pandemia provocada por el virus SARS-CoV-2, la educación presencial tuvo que adaptarse a un escenario completamente no presencial. Las prácticas de laboratorio fueron sustituidas, en el mejor de los casos, por simuladores, que, si bien son una herramienta de gran envergadura, también se hicieron patentes sus limitaciones: si la diferencia entre lo efectuado en un laboratorio presencial con la práctica profesional ya es considerable, esta se agranda cuando se elimina el contacto físico con el instrumental, dispositivos y materiales del laboratorio.

Los laboratorios remotos pretenden mantener las ventajas de los simuladores de accesibilidad y no presencialidad, con una experiencia similar a la clase presencial por estar controlando, de forma remota, el instrumental, dispositivos y materiales de un laboratorio real.

1.2 Laboratorio remoto

Como se ha demostrado en varios estudios de psicología cognitiva, la parte práctica en la que se realizan acciones y se reflexiona sobre estas es la que presenta mayor relevancia a la hora de adquirir un conocimiento [2]. En las ramas técnicas y científicas, este método experimental se vuelve imprescindible cuando lo enfocamos al entorno

educativo. Esto es debido a que la implicación de los alumnos en una práctica asienta los conocimientos teóricos de la asignatura en cuestión.

No obstante, la presencialidad en el laboratorio no siempre es posible. Por esta razón, están en auge otras alternativas para realizar esta experimentación de una forma no presencial con diversas ventajas respecto a la simulación computacional como son los laboratorios remotos.

Los laboratorios remotos se tratan de un sistema formado por software y hardware en el que se permita experimentar con las mismas funcionalidades que en un laboratorio de aula presencial. En estas mediante el software, que generalmente está compuesto por una interfaz, controla el material del laboratorio necesario para realizar prácticas, es decir, se produce una teleoperación con la planta real. El servidor a través del cual se realiza la teleoperación cuenta con sensores (cámaras, micrófonos), para dar información inmediata al estudiante de las operaciones que está realizando [3].

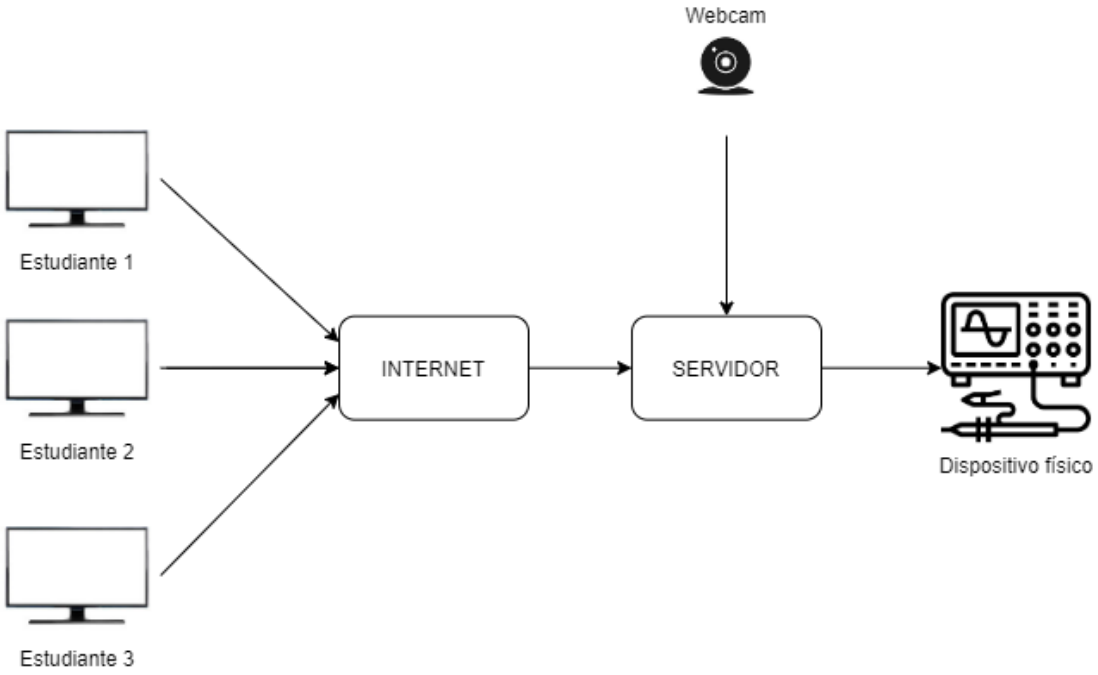


Figura 1. Esquema de la estructura de un laboratorio remoto. Fuente: Propia

Gracias al uso de los laboratorios remotos, en el caso particular de la electrónica digital, los estudiantes pueden controlar y experimentar en tiempo real con el material de laboratorio para comprobar telemáticamente si lo que han diseñado y programado coincide con el comportamiento esperado. Estos laboratorios para la enseñanza de electrónica son particularmente asequibles para la práctica con dispositivos reprogramables.

Este auge del uso de laboratorios remotos en lo referido a la enseñanza experimental se debe a las ventajas que presenta respecto a los laboratorios presenciales:

- Los estudiantes pueden acceder independientemente de los horarios.

- Reducción de coste en consecuencia a la posibilidad de compartir el material.
- Hace más accesible la colaboración con otras instituciones en lo referido a la educación e investigación.
- Más accesibilidad para las personas con discapacidad.
- Mayor autonomía en el aprendizaje del estudiante.
- Menor riesgo de dañar el material de laboratorio.

1.3 Aplicación de los laboratorios remotos

En cuanto a la aplicación de los laboratorios remotos una de los factores más importantes es el de asemejarse al contacto directo del laboratorio real. En un laboratorio presencial, se percibe la experimentación con prácticamente todos los sentidos. Debido a esto, la adquisición por parte del alumno de la máxima información posible de forma telemática es vital para el correcto desarrollo de las prácticas. Esta adquisición de datos generalmente estará recogida por una webcam para visualizar lo que se está produciendo, un micrófono para recoger el sonido, así como una interfaz que permite mandar y adquirir datos del hardware de forma sencilla e intuitiva. Esta permite accionar los elementos que podría manipular de forma presencial: botones, deslizadores, interruptores…

Gracias al uso de estas herramientas de adquisición de datos, los estudiantes pueden realizar las prácticas de las asignaturas correspondientes, aparte de estar más familiarizados con diversas herramientas importantes para su futuro profesional como pueden ser las TIC (Tecnologías de la Información y las Comunicaciones) o las herramientas de control remoto [4].

1.4 Objeto del proyecto

El objeto principal de este proyecto es el de diseñar y desarrollar un laboratorio remoto cuya funcionalidad sea la de que los estudiantes puedan llevar a cabo las prácticas de la asignatura Electrónica Digital del Grado en Ingeniería Electrónica Industrial y Automática de la Escuela Técnica Superior de Ingeniería del Diseño de la Universitat Politècnica de València, a cargo del Departamento de Ingeniería Electrónica sin necesidad de acudir presencialmente al laboratorio ubicado en la universidad.

Además, se han tenido en consideración otras asignaturas, de este y otros grados, para que este laboratorio remoto sea de utilidad también en estas, como son la asignatura de Tecnología Electrónica del mismo grado, las asignaturas de Electrónica y Automática, y Aplicaciones de los Microcontroladores en Ingeniería Mecánica del Grado en Ingeniería Mecánica de la misma escuela, como otras de los grados de Ingeniería de las Tecnologías y Servicios de la Telecomunicación, y en Tecnología Digital y Multimedia de la Escuela Técnica Superior de Ingeniería de la Telecomunicación de la misma universidad.

Este laboratorio remoto debe ser fácilmente reproducible a un coste de ejecución material asequible, con todo el software y hardware utilizado accesible, para que pueda utilizarse en las asignaturas mencionadas.

Queda fuera del ámbito del proyecto el desarrollo de la aplicación telemática para el control de accesos al servidor. Durante el desarrollo de este proyecto se utilizará AnyDesk como herramienta de control remoto del servidor.

El entorno de electrónica digital del laboratorio remoto como características principales debe contar con un microcontrolador programable por el estudiante que lo teleopere, un sistema de acceso remoto fácilmente utilizable, con una interfaz gráfica de usuario que permita establecer los estímulos necesarios.

El dispositivo físico debe contar con el microcontrolador programable con todos los periféricos a controlar necesarios para las asignaturas mencionadas, así como pudiendo ser visualizado por el estudiante a través de una cámara.

Otros aspectos reseñables es que la interfaz debe ser lo más similar posible al conjunto de botones, deslizadores e interruptores utilizados en las prácticas de laboratorio presenciales para que sea lo más cercano posible a las mismas. Contará con un módulo de expansión por si en el futuro se decidiera incluir más periféricos o estímulos al microcontrolador programable.

2. Métodos y materiales

Para llevar a cabo la realización de este proyecto se va a tener que utilizar un método y varias plataformas de desarrollo. Estas plataformas deberán adaptarse a las necesidades, así como cumplir las especificaciones del proyecto. En este apartado de la memoria, se va a explicar los programas de software empleados, así como el hardware necesario para el desarrollo del proyecto.

2.1 Método

El método empleado para la realización de este proyecto se basa en el marco de proyecto flexible ya que los requisitos se han ido detallando conforme se avanzaba en el proyecto. También, se ha dividido en varias etapas principales y se han implementado objetivos específicos para la realización de cada etapa.

2.2 Software

2.2.1 Software de diseño del esquemático y de la PCB

Para el diseño de la PCB se ha utilizado el software de "*Proteus Design Suite v8.9*". Este software se utiliza para diseñar y simular circuitos electrónicos, así como el diseño de PCBs a partir de esquemáticos. Una de las ventajas de este software es que es muy intuitivo y presenta todo lo necesario para la posterior fabricación de una PCB. Es el utilizado actualmente en las asignaturas de Electrónica Digital y Tecnología Electrónica del grado en Ingeniería Electrónica Industrial y Automática de la UPV.

Figura 2. Proteus, programa de diseño de circuitos electrónicos y PCBs. Fuente: Edasim

Proteus está compuesto por dos programas principales: ISIS y ARES, con un módulo integrados en cada uno de ellos llamados VSM y Electra.

- ISIS: Tiene como función el diseño del plano eléctrico del circuito (esquemático). Permite realizar circuitos con componentes muy variados y complejos, como por ejemplo un microcontrolador Arduino. Para el uso de estos componentes presenta una amplia selección de galerías con componentes predefinidos. Además, permite la creación de componentes con varios parámetros y funcionalidades.

El módulo VSM cuyas siglas significan “sistema virtual de modelado” es un módulo integrado en el ISIS que permite la simulación en tiempo real del plano eléctrico del circuito. Este permite incluso la programación y ejecución de programas en microcontroladores, de gran utilidad para simular el sistema antes de implementarlo físicamente.

- ARES: se define como el software de edición y rutado avanzado y es la herramienta de enrutado, ubicación y edición de los componentes que nos permite el diseño de la placa de circuito impreso. ARES asegura que todas las conexiones se realicen de acuerdo al diseño del esquemático y visualizarlas según la lista de redes del diseño. También permite editar las distintas capas del diseño.

  El módulo Electra se trata de un Auto Router, es decir, su función es establecer las conexiones entre los componentes de manera automática. Mediante una selección de requisitos y estrategias se consigue la optimización del diseño con esta extensión del software de ARES.

2.2.2 Software de diseño de la interfaz gráfica

La interfaz gráfica de usuario se ha realizado mediante *Processing*, una herramienta de uso libre ampliamente utilizada por desarrolladores de tecnologías interactivas, basada en Java.

Para la generación de estímulos para el microcontrolador programable se ha utilizado un microcontrolador Arduino, para el cual se ha desarrollado un firmware específico para la comunicación serie con la interfaz gráfica de usuario, así como un protocolo de comunicaciones a nivel de aplicación.

Figura 3. Firmware de Arduino y software de Processing, entorno de desarrollo de la interfaz gráfica. Fuente: DIYMakers

Arduino es una tarjeta de evaluación libre, con una aplicación de las mismas características multiplataforma en la que se permite la programación del microcontrolador en C++, con un conjunto de librerías que facilitan esta programación.

2.3 Hardware y materiales utilizados en la fabricación

Para desarrollar el proyecto se han llevado a cabo tres partes fundamentales: el diseño de la PCB (tanto el diseño del esquemático como el del layout), la programación de todo lo necesario para el funcionamiento de la interfaz gráfica y finalmente, la fabricación de la PCB.

El desarrollo de todo el software se ha implementado mediante el uso de un ordenador personal capaz de ejecutar todos los programas mencionados anteriormente.

En cuanto a la fabricación de la PCB cabe destacar la utilización de todos los componentes de la placa de circuito impreso (que se encuentran detallados en el apartado de Materiales del Presupuesto), así como todo el material de laboratorio necesario para llevar a cabo su fabricación: para la impresión de los negativos sobre el material fotosensible, los reactivos y utensilios para el revelado y atacado, las brocas para el mecanizado y el estaño de soldadura, flux, esponjas, etc. para la soldadura.

A continuación, se va a hacer un listado del material de laboratorio utilizado para la fabricación de la PCB, así como una breve explicación de este.

- **Máquina taladradora CNC:** Se trata de una máquina automatizada que se utiliza para perforar formas y posiciones predeterminadas. Su funcionamiento se basa en el control de los ciclos de su proceso mediante Control Numérico por Computador (CNC). La máquina taladrora utilizada en ester proyecto (Figura 4) es una CNC modelo Bungard Electronik CDC.

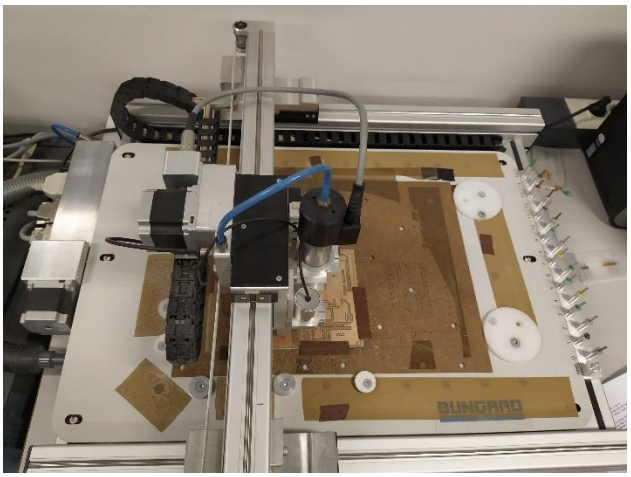


Figura 4. Máquina taladradora mediante CNC. Fuente: Propia

- **Insoladora:** El modelo utilizado es una Bungard Electronik Hellas (Figura 5), que irradia con luz ultravioleta lo que se deposita en su interior. Mediante la iluminación de un fotolito cubierto con un negativo de las capas a obtener (en ambas caras) es posible cargar eléctricamente el cobre que queremos atacar, que será menos susceptible a ser cubierto con una capa de recubrimiento cuando sumerjamos el fotolito sobre un agente básico.

Figura 5. Máquina insoladora. Fuente:Propia.

Las insoladoras también son conocidas como copiadoras por contacto: internamente se trata de una máquina con cristales planos con un sistema para crear el vacío poniendo a la placa, el cristal e imagen a copiar en contacto directo. La iluminación está incorporada en la máquina y se trata de tubos actínicos que emiten luz ultravioleta.

- **Estación de soldadura:** Es una máquina cuya función es la fijación de dos materiales mediante la fundición de un material que una vez fundido se coloca en la unión de los dos materiales que se pretenden fijar para que cuando el material fundido se enfrié se convierta en una unión fija. La utilizada, mostrada en la Figura 6, es modelo Weller WXR 3, incorporando una punta de soldadura, un soplete y una herramienta de absorbción de estaño, todas con temperatura y parámetros de funcionamiento configurables.

Figura 6. Estación de soldadura. Fuente: Propia.

Se suele utilizar como material de fundición el estaño que a una temperatura aproximadamente de 300 °C, esto depende de la composición material del estaño a utilizar, el facilitador de soldadura utilizado y el plano de cobre del punto a soldar.

3. Especificación de requisitos

Para lograr el objetivo de un proyecto se requiere que el conjunto de software y hardware presenten unas condiciones determinadas. Con estas condiciones llamadas requisitos se especifica el comportamiento del sistema que se va a desarrollar.

Los objetivos se dividen en funcionales o técnicos, y no funcionales o comerciales. Los primeros definen las características del sistema relacionadas con su comportamiento principal, mientras que los segundos se refieren a factores como accesibilidad y mantenimiento o a la forma en que el sistema realiza distintas acciones.

Funcionales:

1. Debe constar de un microcontrolador programable por el usuario final.
2. El microcontrolador programable debe poder recibir estímulos en sus pines desde la interfaz gráfica de usuario.
3. Sistema fácilmente reproducible: materiales y elementos disponibles y económicos.
4. Accesibilidad total al software y al hardware.
5. Bus I2C para posibilitar la incorporación de ampliaciones tanto de estímulos controlados por la interfaz gráfica de usuario como de módulos controlables por el microcontrolador programable.
6. El microcontrolador programable debe incluir:
    - Driver para control de motor DC por PWM.
    - Capacidad de medida mediante ADC integrado del valor regulado de tensión para el motor.
    - Medición de temperatura.
    - Medición de dos valores analógicos mediante ADCs integrados, modificables desde la interfaz gráfica de usuario.
    - Medición de cinco botones por GPIO controlados desde la interfaz gráfica de usuario.
    - Tres LED conectadas a GPIO con resistencias de limitación de corriente adecuadas.
    - Pantalla controlada por I2C.
    - Posibilidad de RESET del microcontrolador programable a través de la interfaz gráfica de usuario.

No funcionales:

- Interfaz gráfica de usuario intuitiva, emulando los elementos controlables en el laboratorio físico que serán visibles a través de la cámara.
- Interfaz gráfica de usuario con información respecto a los datos enviados y control de la información enviada, notificando en el caso de existir error en las comunicaciones entre servidor y microcontrolador programable.
- Utilización de hardware y software de uso libre para facilitar su reproducción.
- Coste máximo de materiales de: 100€.

4. Análisis

En este apartado se va explicar los distintos planteamientos que han sido realizados en orden a llegar a la solución adoptada. Además, se valorarán las diversas opciones con relación a qué funcionalidad se adapta mejor al proyecto, así como la explicación de la solución adaptada.

4.1 Estudio de alternativas

Para llevar a cabo el desarrollo del presente proyecto se han utilizado varias herramientas software y hardware tanto para el diseño como para la implementación del presente proyecto. Estas se han escogido tras un estudio de alternativas que se muestra a continuación.

En primer lugar, se va a realizar una breve explicación de las alternativas, indicando en qué medida se adaptan al presente proyecto, a la hora de escoger los programas de software de desarrollo del sistema.

A continuación, se explicará las alternativas en de componentes de la PCB.

4.1.1 Alternativas para el diseño de la PCB

En primer lugar, se va a explicar las posibles opciones de software a la hora del diseño de la PCB.

El software de diseño de la PCB debe permitir la realización de un circuito esquemático con todos los compontes del diseño final, así como el correcto conexionado entre ellos. Asimismo, debe facilitar el diseño de la disposición de una placa de circuito impreso a partir de este circuito esquemático previamente comentado. Los entornos de desarrollo más utilizados para este propósito son los de Cadence, Altium, Eagle, KiCAD y Proteus.

A continuación, se va a analizar las ventajas que presentan cada uno estos programas y en qué medida se adaptan al proyecto:

- Altium: Es una herramienta profesional de altas prestaciones ampliamente utilizada en el entorno profesional. Presenta un elevado coste económico, una elevada complejidad de uso y no se utiliza en el ámbito educativo.
- Cadence: Es una herramienta profesional rival de la anterior, comparte las características de elevada complejidad de uso y elevado coste, tiene prestaciones inferiores en renderización 3D pero superiores en diseño microelectrónico y simulación. Sí es utilizada en el ámbito académico.
- Eagle: Propiedad de Autodesk, es una herramienta de diseño de esquemáticos y PCBs más sencilla y más intuitiva que las anteriores, que incluye capacidades de simulación limitadas y con una versión limitada educativa que está teniendo cada vez mayor alcance en el ámbito educativo.

- KiCAD: El origen de Eagle que ha tenido un desarrollo paralelo con prestaciones más variadas y mayor complejidad de uso, lo que ha limitado su alcance educativo. Por sus capacidades de renderización compite en el ámbito profesional con Altium.
- Proteus: Proteus es una herramienta de diseño de esquemáticos y PCBs privativa, pero con un coste muy inferior a Altium y Cadence. Aunque no es tan intuitiva como Eagle, tiene una capacidad de simulación muy superior, incluyendo la programación de microcontroladores y la ejecución y depuración de código en las mismas simulaciones. Es la alternativa utilizada actualmente en Electrónica Digital y otras asignaturas de grado en Ingeniería Electrónica Industrial y Automática.

4.1.2 Alternativas para la interfaz gráfica de usuario

En cuanto a la interfaz gráfica podría realizarse con diversas plataformas válidas como pueden ser las alternativas seleccionadas para este proyecto Matlab, LabVIEW, Python, Appium o Processing.

Matlab y LabVIEW son programas de gran utilidad para el prototipado rápido, pero presentan runtimes pesados, indispensables para la ejecución de los programas, y licencias de elevado coste.

Los programas de Appium y Python presentan un lenguaje y una accesibilidad con cierta complejidad, lo que puede ser de gran interés de cara a muchos proyectos. Sin embargo, en este caso no es la característica de mayor interés, debido a que se valora la más la sencillez e intuitividad del programa en cuestión. Esta característica sí que está presente en el programa de Matlab ya que presenta la herramienta llamada GUI (Interfaz Gráfica de Usuario) que permite la realización de interfaces de una manera rápida y sencilla, así como LabVIEW.

Processing es un programa de software libre enfocado al diseño de interfaces gráficas de usuario para profesionales gráficos y multimedia, para los cuales la programación no es su tarea principal. Permite por tanto crear interfaces gráficas de usuario adecuadas a las necesidades del proyecto y sin costes asociados a las licencias, que serán accesibles para su modificación o ampliación de forma directa y accesible.

4.1.3 Alternativas para el microcontrolador programable

En lo referido a la elección del microcontrolador que va a programar el estudiante se ha tenido en cuenta las siguientes opciones: STM32, PIC, Arduino Uno y C2000. Se va a realizar un breve análisis de estas alternativas en orden a encontrar la idónea para el presente proyecto.

- **STM32**: Se trata de un microcontrolador de 32 bits popularizado en el ámbito educativo a través de sus tarjetas de evaluación, que permiten desarrollar prototipos para aplicaciones integradas. Sus potentes prestaciones, amplia documentación y buen soporte lo han convertido en uno de los

microcontroladores más ampliamente utilizados tanto en el entorno profesional como en el entorno educativo, si bien es un microcontrolador complejo sin funcionalidades que permitan su programación a alto nivel para los primeros pasos de aprendizaje o el prototipado rápido.

- **PIC**: PIC es microcontrolador programable de la marca Microchip que contiene memoria de programa, base de tiempos y circuitos auxiliares. Presenta un entorno de desarrollo freeware llamado MPLAB que incluye un simulador de software y un ensamblador. Ha sido ampliamente utilizado en el ámbito educativo, pero ha sido sucedido por otros microcontroladores en la práctica profesional, lo que ha llevado consigo una presencia venida a menos en el ámbito educativo. Se sigue presentando en la introducción a microcontroladores de muchos estudios superiores por presentar una estructura relativamente sencilla que facilita ver en la práctica los conceptos teóricos impartidos.
- **Arduino Uno**: Se trata de una tarjeta de evaluación básica basada en el microcontrolador AtMEGA 328. Se caracteriza por ser de hardware y software libre, lo que proporciona total accesibilidad de uso. El software libre ofrecido es el llamado Arduino IDE (Entorno de Desarrollo Integrado) con las librerías de Arduino. Este es muy utilizado en el sector educativo debido a que, gracias a su intuitividad y su facilidad de uso, es ideal para introducirse a la programación de controladores, el prototipado rápido o un aprendizaje de alto nivel. Debido a sus características de software y hardware libre, puede profundizarse en el microcontrolador prescindiendo de las librerías (e incluso del IDE) de Arduino, programando a bajo nivel registro a registro, una funcionalidad especialmente útil en el ámbito educativo para poder realizar todo el proceso de aprendizaje de alto a bajo nivel con la misma plataforma. Esto, a su vez, hace que sea una de las plataformas más utilizadas en el ámbito profesional e investigación para el prototipado rápido.
- **C2000**: Propiedad de Texas Instruments, es un microcontrolador de alto rendimiento diseñado para el control de la electrónica de potencia, así como el procesamiento de señales digitales. El uso de este microcontrolador tiene un enfoque más industrial que los anteriores, su complejidad es mayor, y es utilizado en el ámbito educativo en asignaturas superiores.

4.1.4 Alternativas de buses de comunicación para la expansión modular

Para la ampliación modular del laboratorio remoto es importante contar con un protocolo de comunicaciones que facilite esta tarea.

El bus SPI es un bus con una arquitectura maestro–esclavo mediante es uso de 4 cables, pudiendo mandar valores a la misma vez valores del maestro esclavo y del esclavo al maestro. Presenta más funcionalidades posibles, pero con una estructura más compleja. Para el correcto funcionamiento del bus SPI son necesarias cuatro conexiones: CS (responsable de la dirección del esclavo), MOSI (envío de datos de master a esclavo), MISO (envío de datos de esclavo a master) y SCK (la señal de reloj que sincroniza todos los dispositivos del bus).

El bus I2C es un bus que también presenta una arquitectura maestro-esclavo. Únicamente se precisan de dos cables para su conexionado ya que omite los cables de conexión y no permite el envío simultáneo de datos por parte del esclavo y el maestro. Respecto al SPI, presenta una estructura más simple y con las características necesarias para el correcto funcionamiento del sistema. Además, este bus es compatible con muchos periféricos como memorias, convertidores, expansores de E/S, controladores de pantallas LCD y sensores.

4.2 Solución adoptada

En el presente apartado se va a determinar la solución adoptada del proyecto teniendo en cuenta las alternativas expuestas en el apartado anterior. Tras esta valoración de las alternativas se ha llegado a la siguiente solución:

En la elección del software del diseño de la PCB se va a elegir el programa de Proteus, de forma que el diseño de la PCB pueda emplearse también como simulador previo por parte del estudiante que utilice el laboratorio remoto. Es un software actualmente utilizado en el grado.

En cuanto al software destinado al diseño de la interfaz gráfica se va a seleccionar el Processing ya que se trata de software de uso libre destinado al desarrollo de interfaces sencillas.

Tanto como microcontrolador programable por el estudiante de forma remota, como para hacer las veces de generador de estímulos a este último comandado desde la interfaz gráfica de usuario se va a utilizar Arduino Uno, una tarjeta de evaluación basada en el microcontrolador AtMEGA 328 utilizada ampliamente en el ámbito educativo porque permite su programación a alto y bajo nivel, como se ha explicado en el apartado 4.1.3.

Finalmente, se escogerá la alternativa de bus I2C debido a que requiere menos cables y componentes para establecer su conexión. De esta forma también facilita la conexión de dispositivos futuros compatibles con este tipo de bus.

La explicación del software del diseño de la PCB y del diseño de la interfaz han sido previamente explicados en los apartados 2.2.1 y 2.2.2. A continuación, se va a hacer una breve explicación de la definición del bus I2C y de su funcionamiento debido a que es un elemento no explicado con detalle funcional en el apartado anterior y de relevancia para el diseño e implementación del laboratorio remoto.

4.2.1 Definición del bus I2C

El bus I2C se trata de uno de los principales sistemas de comunicación disponibles en Arduino. Su funcionamiento se basa en dos cables, uno encargado de transmitir la señal del reloj llamado SCL y otro encargado de transmitir datos llamado SDA.

El bus I2C se basa en una estructura maestro-esclavo en la que cada dispositivo presenta una dirección diferente. En esta estructura el maestro sólo puede iniciar la comunicación y puede mandar o recibir datos de los esclavos.

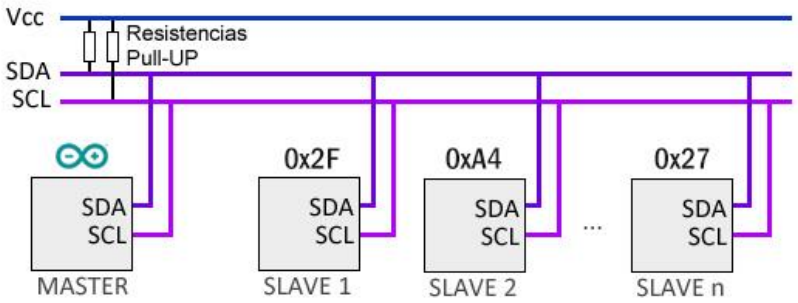


Figura 7. Estructura del bus I2C. Fuente: Aladuino electrónica

Se debe conectar ambos canales del bus a resistencias de pull-up con un valor comprendido entre 1k Ω y 4,7k Ω para acelerar la comunicación o aumentar la distancia de transmisión. Estas deben estar conectadas a Vcc como podemos observar en la Figura 7.

Además, la señal de reloj proporcionada por el maestro mantiene sincronizados a todos los esclavos.

4.2.2 Funcionamiento del bus I2C

En cuanto al funcionamiento del bus I2C, ya que el canal SCL únicamente sincroniza todos los dispositivos conectados al bus, la comunicación se realiza en el canal SDA.

Para establecer la comunicación se dispone de un bit de inicio. Tras este bit de inicio se transmite siete bits de dirección cuya función es indicar a qué dispositivo se pretende enviar los datos. El siguiente bit indica si se quiere enviar o recibir información (R/W). Finalmente, el último bit es un bit de confirmación (ACK)

A continuación, se disponen de ocho bits de envío o recepción de datos del esclavo con la memoria indicada. El bit final lo manda el maestro para ordenar el fin de comunicación (NACK).

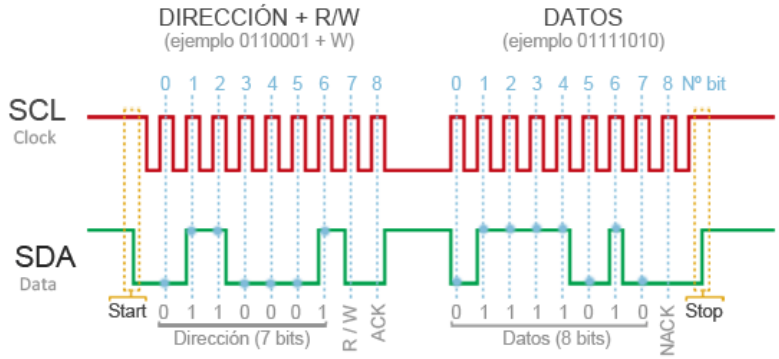


Figura 8. Funcionamiento del bus I2C. Fuente: Aprendiendo Arduino

5. Diseño

En este apartado de la memoria se va a explicar las diversas partes fundamentales llevadas a cabo para la realización del diseño de la PCB. En estas partes está comprendido el diseño del circuito esquemático de la PCB en Proteus. Además, se va explicar la conexión de ambos microcontroladores que controlan el circuito, el Arduino (de ahora en adelante, Arduino Virtual) que generará los estímulos al microcontrolador programable por el estudiante (de ahora en adelante Arduino Programable), y posteriormente se explicará el diseño de la PCB.

5.1 Esquemático

El diseño del esquemático de la PCB es el que determina los componentes y el conexionado entre estos. En este se especifica las componentes con sus respectivas alimentaciones y otro tipo de conexiones.

Sin embargo, no siempre todos los componentes deseados se encuentran incluidos por defecto en Proteus. Por ello, es conveniente comprobar si los componentes deseados se encuentran en las librerías de Proteus. En este caso, varios de los componentes deseados no se encontraban en las librerías, por lo tanto, se han debido de añadir manualmente con la herramienta de creación de componentes de Proteus.

Durante la selección de componentes se ha tenido en especial consideración la utilización de empaquetados (o packages, en inglés) que faciliten la fabricación de la PCB con los recursos disponibles en las instalaciones del departamento, en particular las del *Grupo en Dispositivos Electrónico y Sensores Impresos* de la UPV.

El orden de prioridad ha sido el siguiente:

1. Utilización de componentes estándar, con existencias en el laboratorio y en las librerías por defecto de Proteus.

2. Si no están en las librerías por defecto, pero se tienen existencias, se diseña el componente para su inclusión en el esquemático y PCB.

3. En su defecto, utilización de componentes económicos y en circulación presentes en las librerías por defecto.

4. Y en caso de no haber alternativa, diseñando también el componente.

Finalmente, una consideración relevante para la selección de componentes es el uso de empaquetados estándar que se encuentren en las librerías. Si no, es necesario diseñar el empaquetado.

Antes de la realización del esquemático se va especificar el BOM de los componentes de la PCB:

- Arduino Uno (Virtual)
- Arduino Uno (Programable).

- Conexión I2C:
    - Resistencias Pull-up: 4 (2.2K).
    - Conectores: 2.
- Potenciómetros digitales: 2.
- Motor con control PWM:
    - Motor DC 12V.
    - Transistor 2N2222.
    - Resistencia base transistor del motor (330).
    - Amplificador operacional LM2900N: Dos buffers.
    - Divisor de tensión: R16 (10K) y R17(3.3k).
- Pantalla LCD:
    - LCD MC21605G6WD 16x2.
    - Adaptador I2C para LCD (expansor de E/S).
    - Divisor de tensión VEE: R14 (4.7K) y R15 (820).
- Sensor de temperatura TMP36.
- Resistencias para los botones: 5 (220).
- LEDs:
    - LEDs: 3.
    - Resistencias: 3 (220).
- Conectores de alimentación:
    - VCC y GROUND.
    - 12V y GROUND.
    - Alimentación del motor.

Como se puede apreciar en el BOM se va a dividir en varias secciones el diseño para la correcta explicación de cada una de los circuitos que componen el esquemático.

5.1.1 Arduino Virtual

Los dos componentes principales son los dos Arduinos, el virtual (V) y el programable (P), que gestionarán la alimentación y recogerán los datos necesarios en sus entradas. También dispondrán de un bus I2C cada uno para el control de diversos dispositivos controlados mediante I2C del diseño. En primer lugar, se va a comentar las conexiones con el tipo de Arduino que presentan los dos Arduinos del diseño, así como una breve explicación de los tipos de entradas y salidas que presenta.

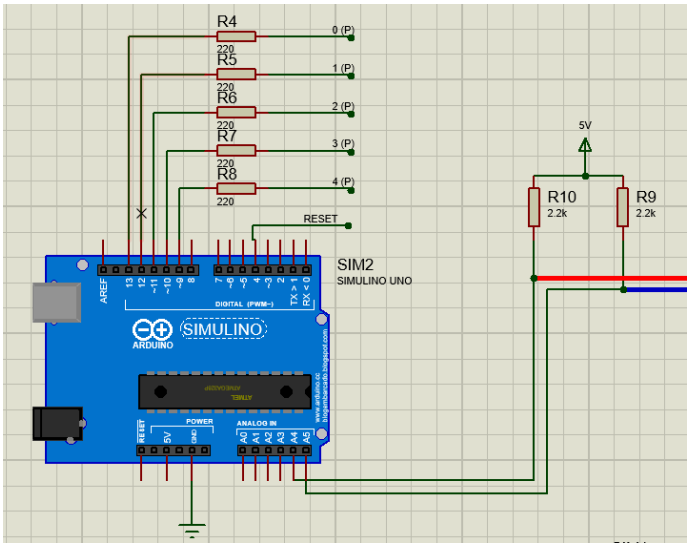


Figura 9. Conexiones del Arduino Virtual. Fuente: Propia.

Tanto el Arduino virtual como el programable son el tipo de microcontrolador llamado Arduino UNO. Este tipo es la gama básica que presenta varios tipos de pines con diversas funcionalidades. Presentan los pines de alimentación y Reset, las entradas analógicas (A1 a A5) cuyos pines A4 y A5 son los destinados al control del bus I2C, así como los digitales (0 a 13) cuyos pines 3, 5, 9, 10 y 11 presentan capacidades de modulación por ancho de pulso (PWM).

Como se puede observar en la figura, los pines digitales (9-13) se encuentran conectados a las entradas digitales del Arduino programable. El pin digital 4 está conectado al Reset del Arduino programable y los pines A4 y A5 están conectados ambos cables del bus I2C.

5.1.2 Arduino Programable

Este Arduino es el destinado al control de todo lo necesario para realizar las prácticas ya que es el que, en la aplicación de laboratorio remoto, los estudiantes deberán programar y comprobar su funcionamiento de estos circuitos específicos dependientes de este Arduino.

Como se ha mencionado previamente el tipo de Arduino es similar al Arduino cuyas funcionalidades de los pines se explican brevemente en el apartado anterior.

En cuanto a las conexiones de este componente cabe destacar que se conecta a la base del transistor del motor con una salida PWM para el control del motor. Los valores de los potenciómetros digitales y el valor medido del motor y el sensor de temperatura van a entradas analógicas.

Se realizará la conexión del bus I2C por ambos cables (A4 y A5) que se conectará al expansor de E/S utilizado para el funcionamiento de la pantalla LCD.

Finalmente, el resto de pines destinados a los botones y LEDs serán de tipo digital.

5.1.3 Bus y conexiones I2C

Cada Arduino tiene un bus I2C propio para distintas funcionalidades. El bus I2C del Arduino Virtual tendrá la funcionalidad de comunicarse con los potenciómetros digitales, mientras que el del Arduino Programable se encargará de controlar el expansor de pines de entradas y salidas encargado de controlar la pantalla LCD. Asimismo, ambos buses nombrados anteriormente contarán con un conector disponible para conectar más módulos en el futuro.

El bus del I2C de ambos Arduinos se va a controlar mediante los pines de A4 y A5 (SDA y SCL respectivamente). Además, para el correcto funcionamiento del bus se debe implantar resistencias pull-up en todos los canales como se ha explicado en el apartado 4.2.1.

5.1.4 Potenciómetros digitales

Mediante el bus I2C, el Arduino Virtual controlará el valor de ambos potenciómetros digitales independientemente en función de la dirección de cada uno de estos. Posteriormente los valores de los potenciómetros serán recogidos mediante dos salidas analógicas del Arduino Programable.

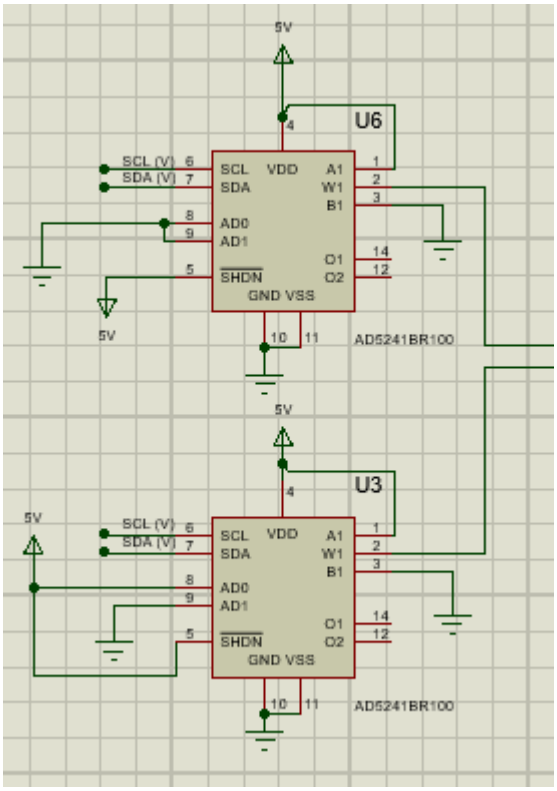


Figura 10. Conexionado de los potenciómetros digitales. Fuente: Propia.

Los potenciómetros digitales AD5241 como se puede ver en su Datasheet y en la figura 14 se deben conectar terminales A1 y B1 a alimentación y tierra respectivamente, Vss y GND a tierra, y !SHDN y Vdd a alimentación. Se conecta SDA y SCL a los cables correspondientes del Arduino Virtual y se establece la dirección según el valor de los

ADO y AD1. Finalmente, se conecta la salida del potenciómetro (W1) a un de las salidas analógicas del Arduino programable.

5.1.5 Control PWM de la velocidad del motor DC

Para el correcto funcionamiento del motor, ya que el microcontrolador de Arduino no permite suministrar la corriente necesaria para generar movimiento en un motor DC se ha realizado un control de velocidad con transistores. Este se ha realizado mediante el transistor 2N2222.

Con la intención de poder medir el voltaje del motor se debe realizar un divisor de tensión con la ayuda de dos amplificadores operacionales seguidores de tensión (búfer).

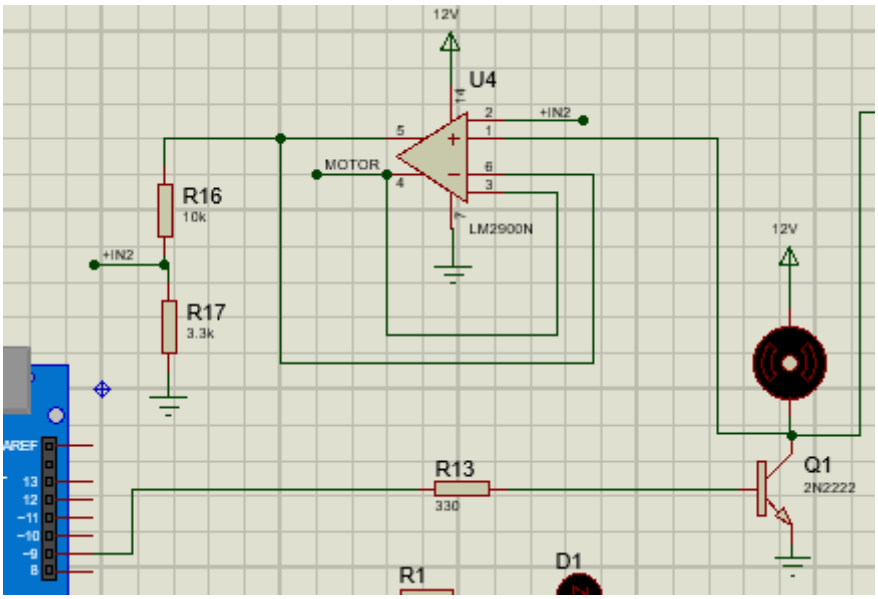


Figura 11. Control PWM de la velocidad del motor DC. Fuente: Propia

5.1.6 Pantalla LCD

La pantalla LCD se va a controlar mediante un expansor de entradas y salidas que se encargará de adaptar los datos enviados mediante el bus I2C del Arduino Programable. De esta forma, el expansor de pines (PCF8574) mandará los 8 pines necesarios para el correcto funcionamiento de la pantalla teniendo en cuenta la información recogida en el Datasheet de esta.

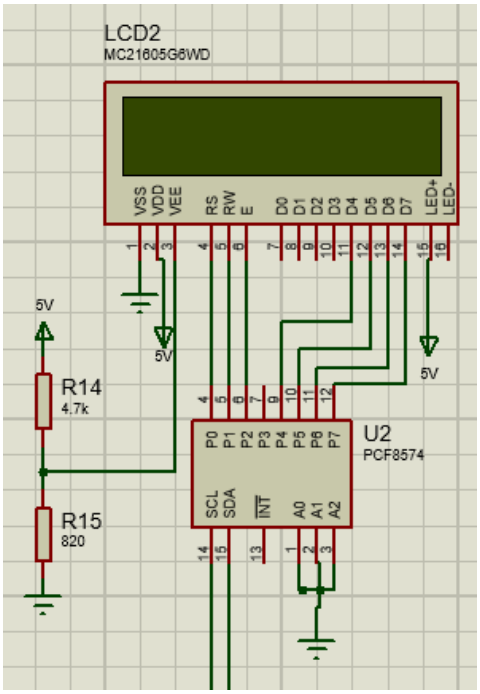


Figura 12. Conexionado de la pantalla LCD y el expansor de E/S. Fuente: Propia

El expansor en función de los datos recibidos por I2C establecerá un valor u otro en los pines que llegarán a la pantalla. La pantalla LCD recogerá estos valores y mostrará el contenido especificado en la pantalla. La pantalla debe alimentarse, como se puede apreciar en la figura x, mediante el valor de Vcc (5 voltios) y el valor resultante del divisor de tensión formado por R14 y R15.

5.1.7 Sensor de temperatura

El componente TMP36 tendrá la función de medir la temperatura del ambiente. Se alimenta con Vcc y se conecta a Ground. Finalmente, la salida (2) esta puesta en contacto con una de las entradas analógicas del Arduino programable.

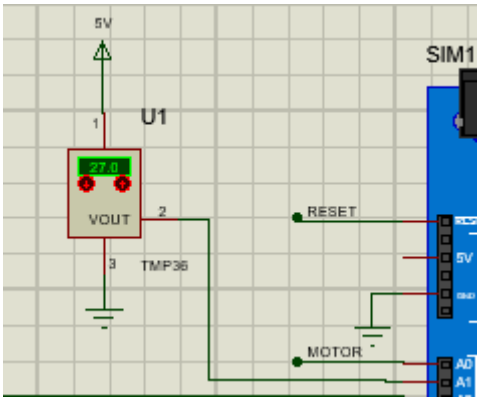


Figura 13.Alimentación y salida del sensor de temperatura. Fuente: Propia.

5.1.8 Resistencias asociadas a los botones

La funcionalidad de estas 5 resistencias es la de evitar el posible cortocircuito entre ambos Arduinos en caso de configurarse los GPIO del Arduino Programable como salidas. Están entre 5 salidas digitales del Arduino Virtual a 5 entradas digitales del Arduino Programable que posteriormente se activarán siguiendo los mandos de la interfaz gráfica.

5.1.9 LEDs

Este circuito consiste en 3 LEDs alimentados por 3 salidas digitales del Arduino Programable y conectados a una resistencia cuya funcionalidad es limitar la corriente que pase por estos.

5.1.10 Conectores

En el esquemático se encuentran varios conectores con distintas funcionalidades.

El primero de estos se encarga de la alimentación de 12V y puesta a tierra (Ground).

El segundo alimenta a 5V (Vcc) y está conectado a tierra.

Finalmente, el tercero tiene el propósito de alimentar al motor ya que este es externo a la PCB. Por lo tanto, mediante la conexión del motor DC a los dos pines correspondientes del conector se podrá activar el motor DC por la salida digital con control PWM para su posterior medición por una entrada analógica del Arduino programable.

5.2 Diseño de la PCB: layout y enrutado.

Durante el presente apartado se va a explicar cómo se ha realizado el diseño de la distribución de la placa de circuito impreso. Se ha diseñado de forma estructurada añadiendo cada parte del esquemático una a una. Todas las referencias incluidas en este apartado están referidas al Anexo 1.

En la PCB podemos encontrar distintos tipos de componentes, que se listan en el primer apartado.

5.2.1 Componentes de la PCB

- Placa: Compuesta por FR4, material aislante y resistente al calor, que dota de rigidez mecánica al circuito ensamblado en él. En el caso del presente proyecto tiene en los dos lados las pistas y planos de cobre y no presenta capas internas.

- Pistas: Se tratan de las superficies lineares de cobre que se encargan de establecer conexiones eléctricas entre dos o varios puntos del circuito.

- Planos: Es una superficie de cobre que cubre toda la superficie de placa ininterrumpidamente. Además, existen planos que únicamente cubren una superficie de la placa que tienen como función de distribuir la potencia entre varios puntos de la placa, así como el apantallamiento de señales contra interferencias.

- Vías: Son perforaciones con la superficie interior metalizada de manera que permiten la conexión entre ambas capas de la placa, Bot (inferior) y Top (superior).

- Pads y AntiPads: los pads son las superficies de cobre destinadas a la colocación de los componentes a la capa externa de Top en la placa mediante una soldadura o la conexión entre una pista y una vía. Son de un tamaño bastante reducido y dependen del package de cada componente en cuanto a la distancia entre los pines y el tamaño. Sin embargo, un antiPad el cobre que se debe eliminar alrededor del pad para que el pad no entre en contacto con planos y pistas a los que no está destinado provocando un error o un cortocircuito.

5.2.2 Restricciones y jerarquía de colocación de componentes

Ya que la fabricación de la placa de circuito impreso del presente proyecto se va a realizar en un laboratorio de manera fundamentalmente manual se debe acatar una serie de requisitos:

- Debe presentar unas dimensiones máximas de 220 mm por 150 mm.
- Pistas de tamaño mínimo de 36 mil.
- Keep-out de 36 mil (separación entre pistas, entre pista y plano de masa).
- Los cables de comunicaciones I2C deben tener al otro lado de la placa plano de masa.
- Ya que las vías se van a metalizar a mano, se va optimizar su número para metalizar las menos posibles.

Además, los componentes que vayan through-hole con dificultades en la soldadura en la capa de Top se deben implementar de manera que de un lado este el dispositivo, y de otro la pista, así cuando se suelda se puede asegurar que haya contacto eléctrico pista-pin-dispositivo. Estos componentes que presentan estas dificultades son ambos Arduinos, la pantalla LCD y todos los conectores, tanto los de alimentación como los de I2C.

Se debe separar los buses de comunicación de las medidas analógicas ya que existe el riesgo de que produzcan ruido en estas. Para ello se debe implementar una "zona analógica" en la PCB por donde no pasen PWM ni I2C.

Con el objetivo de facilitar la implementación y el diseño de la PCB se va a establecer una jerarquía en el posicionamiento de los componentes. También, ayudará a la producción de un diseño más coherente y estético. Esta jerarquía se basa en la colocación de los componentes en capa de Top y en la priorización de la soldadura y las pistas por la capa de Bot. Asimismo, los componentes se van a agrupar de acuerdo a las secciones previamente definidas en el diseño del esquemático.

En primer lugar, para la colocación de los Arduinos se debe tener en cuenta una serie de requisitos: deben estar en un extremo de la placa, deben estar situados en Bot Copper ya que finalmente se conectará boca abajo en Top Copper, deben facilitar las conexiones entre ambos y las pistas deben de llegar todas por Bot Copper.

Los conectores también deben estar colocados a los extremos de la placa, concretamente orientando todos apuntando al extremo más cercano y todas las pistas deben llegar por Bot Copper. Los requisitos de la pantalla LCD son similares a los definidos en el presente párrafo. Sin embargo, se prioriza la colocación de esta en una esquina de la placa.

Los potenciómetros digitales y el amplificador operacional presentan la restricción contraria que la mencionada anteriormente ya que se trata de un package no through-hole, es decir, los pines del componente únicamente pueden ser soldados y conectados por una de las capas externas. Por lo tanto, van a soldarse en la capa de Top y todas las pistas deben llegar por dicha capa. Además, el package necesita unas pistas más reducidas ya que las distancias entre pines son extremadamente pequeñas y con las pistas utilizadas en el resto del diseño no se puede evitar no hacer cortocircuito.

El resto de componentes presentan las restricciones generales previamente comentadas. Debido a esto, se van a situar en la capa de Top y se va a soldar todas los pads de Bot para proporcionar rigidez y los pads conectados a pistas en la capa Top.

5.2.3 Realización del diseño

Una vez se ha definido todas las restricciones y se ha establecido una jerarquía en la colocación de componentes se va explicar el procedimiento empleado para llevar a cabo la realización del diseño. Ya que la jerarquía explicada anteriormente seguía las secciones del esquemático, el procedimiento del diseño estará dividido mayoritariamente en estas secciones.

Para la realización del proceso de enrutado se va a realizar el siguiente procedimiento: Se colocarán las agrupaciones de componentes que se van a especificar a continuación y una vez definidas las especificaciones necesarias en el módulo “auto-router” del programa de Proteus se ejecutará este módulo. Tras esta ejecución se añadirán las pistas y las vías necesarias restantes manualmente.

En cuanto al diseño del layout de la PCB se ha seguido el siguiente procedimiento:

1. En primer lugar, se colocaron ambos Arduinos en los extremos de la placa concretamente en las esquinas opuestas de la placa, esquina inferior izquierda el Arduino Virtual y esquina superior derecha el programables. Se prioriza la colocación de los Arduinos ya que son componentes con un tamaño sustancialmente superior al resto y, por ello, los que más restringen el diseño. Se orientaron de forma que las conexiones del cable para enviar la programación se encuentren en un en el extremo de la placa, así como que las pistas que fueran entre uno y otro no se entrelazarán (Anexo 1). Ya que estas pistas mencionadas anteriormente iban conectadas a resistencias (R4 a R8), se evitaban también entrelazar las pistas que llegaban a estas. Además, se coloca el conector (J2: 5V / GND) ya que ayuda a facilitar el conexionado de pistas de la alimentación de la placa.

2. A continuación, se implementan los conectores necesarios para la expansión de posibles módulos futuros (U7: I2C (Arduino V) y U8: I2C (Arduino P)), así como las resistencias pull-up (R9 a R12) para el correcto funcionamiento del bus. Se han implementado los conectores de acuerdo a las restricciones definidas anteriormente y se han dispuesto las resistencias de con la orientación que más facilitará el conexionado de estas al bus I2C.

3. El siguiente paso llevado a cabo ha sido el posicionamiento y la orientación de los potenciómetros digitales (U3 Y U6). Se han ubicado en la zona superior derecha ya que se sitúan cerca de las conexiones con el bus I2C del Arduino Virtual necesarias y alejados de la zona analógica para evitar posible ruido. Similarmente a la orientación de los Arduinos se han orientado de forma que las conexiones entre sí se encuentren lo menos entrelazadas posible.

4. El siguiente paso es el de la colocación de todos los componentes necesarios para el correcto funcionamiento del control PWM del motor DC que posteriormente se conectará de forma externa a la placa mediante el conector (J3: Motor) destinado a la alimentación del este motor. Para ello, se colocan el conector que alimenta toda la sección del control (J1: 12V / GND), así como el conector mencionado previamente (J3), ambos siguiendo las restricciones definidas en el apartado anterior. Se sigue con la colocación de las resistencias del divisor de tensión (R16 y R17) y el amplificador operacional con dos seguidores de tensión implementados (U4). Finalmente, se conecta la resistencia de la base del transistor (R13) y el transistor (Q1).

5. El quinto paso que se ha realizado es la colocación y orientación de la pantalla LCD (LCD2), así como las resistencias necesarias para el divisor de tensión destinado a la alimentación de dicha pantalla (R14 y R15) y el expansor de E/S (U2) conectado al bus I2C del Arduino Programable. Ya que el tamaño de esta agrupación de componentes es considerablemente más grande, a lo largo de los pasos anteriores se ha reservado un espacio en la esquina inferior de la placa. En esta agrupación de componentes se ha intentado optimizar el número de vías ya que la pantalla LCD presenta una gran cantidad de pistas conectadas únicamente por la capa de Bot.

6. En penúltimo paseo es colocar el sensor de temperatura (U1) y los LEDs (D1 a D3) con sus respectivas resistencias (R1 a R3). Como en la conexión de la mayoría de componentes anteriores se ha priorizado evitar el entrelazado de las pistas.

7. Por último, se implementarán los planos de masa en ambas capas de la placa, tanto Top como Bot, respetando las mismas distancias de pista a pista, a vía con ambos planos.

6. Implementación

En el presente apartado se va a explicar la programación de la interfaz gráfica con el software de Processing y la programación del firmware de Arduino, así como el sistema de comunicación entre ellos explicado en el *Diagrama de flujo 1*.

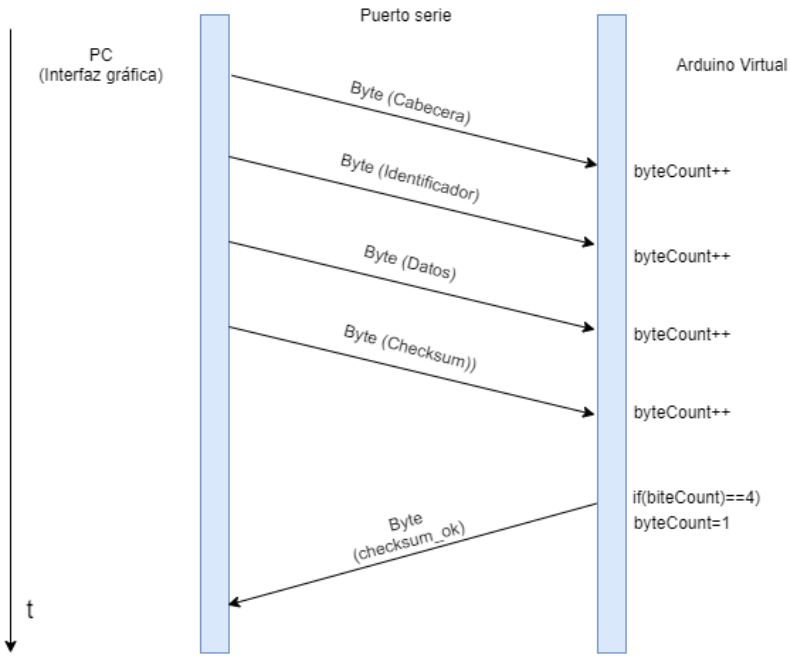


Diagrama de flujo 1. Sistema de comunicación entre la interfaz gráfica y el Arduino Virtual. Fuente: Propia.

6.1 Processing

El programa de Processing tendrá el control indirecto del Arduino Virtual, por lo tanto, desde la interfaz gráfica se deben controlar tanto las salidas como la entrada de información que devuelve el Arduino.

Las salidas del Arduino Virtual están compuestas cinco salidas digitales, dos potenciómetros digitales y una salida digital para el Reset del Arduino Programable. Para el control de todas las salidas mencionadas anteriormente se van a implementar seis botones con el fin de proporcionar 5V (Vcc) cuando se encuentren pulsados. También se va a implementar dos deslizadores con un valor que posteriormente debe ser escalado como datos del bus I2C.

En el *Diagrama de flujo* 2 se explica el funcionamiento general del programa expuesto a lo largo de este apartado. Más tarde, para la correcta y estructurada explicación del programa se va dividir en segmentos de código en relación a su funcionalidad.

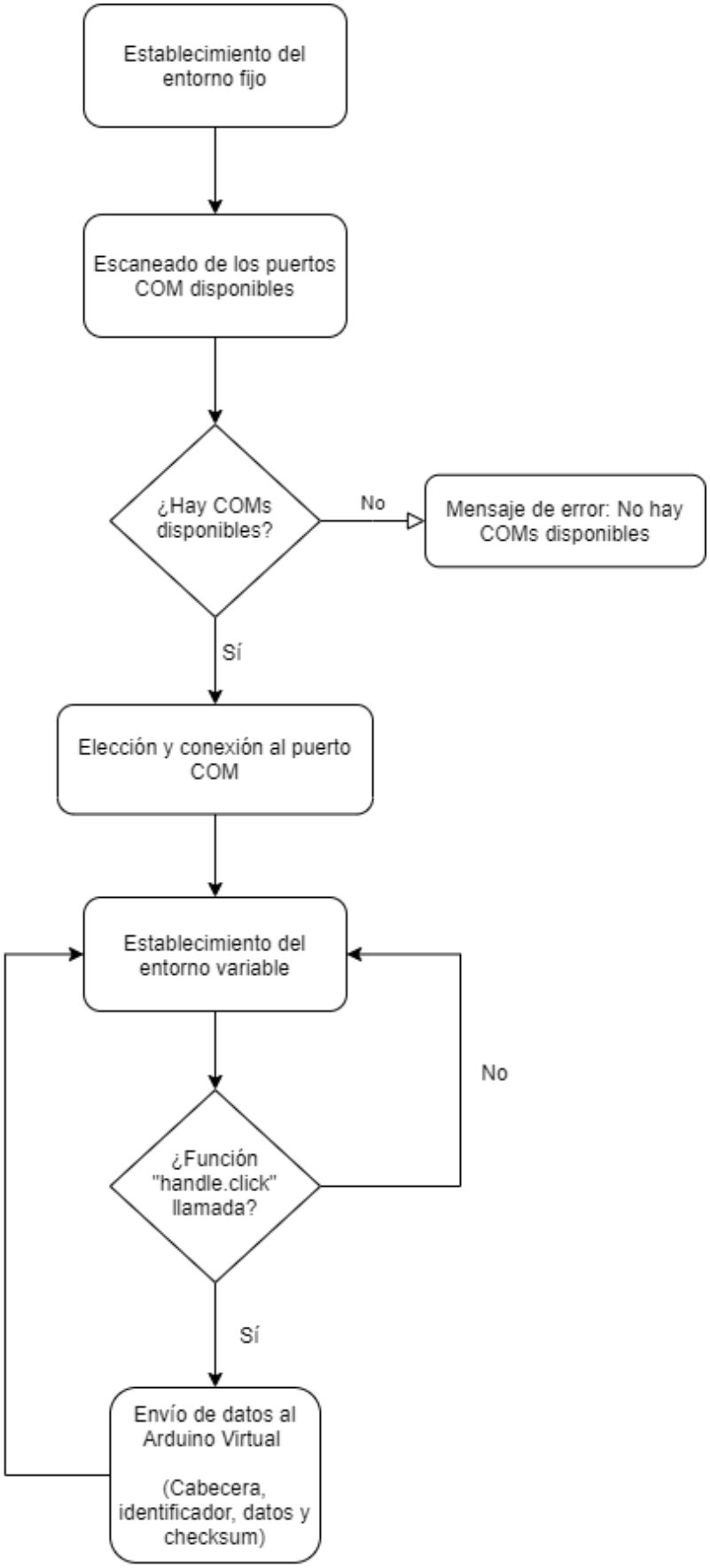


Diagrama de flujo 2. Funcionamiento a rasgos generales del programa de Processing. Fuente: Propia

6.1.1 Librerías importadas

Se va a empezar por una breve explicación de las librerías y la creación de las clases importadas para el correcto funcionamiento del programa.

interfaz

```
1 //Importamos la librerías
2 import controlP5.*;
3 import processing.serial.*;
4 import javax.swing.JOptionPane;
5 import javax.swing.JDialog;
6
```

Fragmento de código 1. Librerías usadas para la interfaz. Fuente: Propia

Como se puede apreciar en la *Fragmento de código 1* se importa cuatro librerías. La primera de ellas llamada `controlP5` está destinada a facilitar y homogeneizar la creación de los componentes de la interfaz modificables por el usuario, como son los botones y deslizadores. Se trata de una clase compuesta por varias clases secundarias como pueden ser los botones.

La librería `Serial` tiene como función la implementación de un puerto serie que en este caso se va a utilizar para establecer conexión con el Arduino Virtual.

Por otro lado, las librerías de `javax.swing.JOptionPane` y `javax.swing.JDialog` se utilizan para facilitar la selección manual a uno de los puertos serie COM disponibles mediante una ventana emergente.

6.1.2 Declaración e inicialización de variables

En este apartado se van a declarar, e inicializar si es conveniente, las variables necesarias para el funcionamiento de la interfaz gráfica.

```
10 //Definiciones
11 Serial arduinoV; //Nombre del puerto serie
12 ControlP5 AV; //Nombre de la clase del control P5
13
14     //Definición de fuentes de texto
15 PFont fontinv;
16 PFont fontvis;
17 PFont fontsecond;
18    //Definición de imágenes
19 PImage imagen;
20 PImage imagen2;
21     //Definición e inicialización de variables
22 int x=75;
23 int y=90;
24 int size_buttonx = 72;
25 int size_buttony = 60;
26 float Potenciometro1,Potenciometro2;
27 final boolean debug = true;
28 String portName = " ";
```

Fragmento de código 2. Declaración de variables. Fuente: Propia

En primer lugar, se establece un nombre al puerto serie y a la clase de `controlP5`.

A continuación, se define el nombre de las distintas fuentes del programa, así como las imágenes a mostrar. Tras esto, se definen las variables inicializándose cuando sea oportuno, es decir, variables que no vayan a ser modificadas por el programa o variables que requieren tener un estado inicial.

6.1.3 Funciones utilizadas en el *setup*

La función `setup` se trata de una función única (solo puede haber una por programa) en la que se establecen las propiedades iniciales del entorno. Se ejecuta solamente una vez al iniciarse el programa.

```
33 void setup() {
34
35   size(1080,720); //Definimos el tamaño de la ventana
36
37   AV = new ControlP5(this);  //Se llama al constructor para la creación de la clase
38
39   declare_fonts(); //Función que define las fuentes de texto
40
41   add_buttons(); //Función para agregar botones
42
43   add_sliders(); //Función para agregar los deslizadores
44
45   connect_to_arduinoV(); //Función para conectar al serial del Arduino
46
47
48 }
```

Fragmento de código 3.Función setup de la interfaz. Fuente: Propia

Lo primero a definir es el tamaño de la ventana, que en este caso se va a definir 1080x720 ya que es una medida muy estandarizada.

La siguiente instrucción se refiere a la declaración de las fuentes de texto donde se declara la fuente primaria y secundaria del programa como se puede observar en la *Fragmento de código 4*.

```
62 void declare_fonts(){
63
64   fontvis = createFont("calibri bold", 20);
65
66   fontsecond = createFont("calibri", 15);
67
68
```

Fragmento de código 4. Declaración de fuentes. Fuente: Propia

Tras la declaración de las fuentes, se añaden los botones mediante la clase `controlP5`. Para ello, se implementan las funciones de `add_button` y `add_button_reset` en la función destinada a agregar todos los botones (*Fragmento de código 5*).

```
175 void add_buttons() {
176   add_button(1,1,"B1");
177   add_button(3,1,"B2");
178   add_button(2,2,"B3");
179   add_button(1,3,"B4");
180   add_button(3,3,"B5");
181   add_button_reset();
182
183 }
```

Fragmento de código 5. Declaración de botones. Fuente: Propia

La función `add_button` como se puede observar en la *Fragmento de código 6*, recibe dos valores de coma flotante (`float`) y un `string`. Los dos valores `float` designan la posición del botón y el `string` el nombre.

```
152 void add_button(float a, float b, String c) {
153 AV.addButton(c)
154   .setPosition(145+a*x,135+b*y)
155   .setSize(size_buttonx,size_buttony)
156   .setFont(fontvis)
157   ;
158 }
```

Fragmento de código 6. Creación de los botones. Fuente: Propia

En cuanto a la función `add_button_reset`, presenta una función muy similar a la anterior, pero con una posición y un nombre definidos, así como un tamaño y colores distintos (*Fragmento de código 7*).

```
160 void add_button_reset( ) {
161
162   AV.addButton("RESET")
163   .setPosition(130 + 4.5*x,135 + 4.5*y)
164   .setSize(140,140)
165   .setFont(fontvis)
166   .setColorActive(#00FF00)
167   .setColorBackground(#FF6464)
168   .setColorForeground(#FF0000)
169   ;
170 }
```

Fragmento de código 7. Creación del botón Reset. Fuente: Propia

La función siguiente es bastante similar a la anterior ya que se utiliza también la clase `controlP5` para agregar dos deslizadores con unas características definidas de posición, rango, tamaño y valor inicial respectivamente como se puede apreciar en la *Fragmento de código 8*.

```
72 void add_sliders(){
73   AV.addSlider("Potenciometro1").setPosition(635,220).setRange(0.0,5.0).setSize((3*x),(x)).setValue(0.00);
74   AV.addSlider("Potenciometro2").setPosition(635,350).setRange(0.0,5.0).setSize((3*x),(x)).setValue(0.00);
75 }
76
```

Fragmento de código 8. Creación de los Sliders. Fuente: Propia

Finalmente se establece la conexión por un puerto serie (Serial COM) con el firmware de Arduino mediante la función `connect_to_arduino`.

```
110 void connect_to_arduinoV() {
111
112     String COMx, COMlist = "";
113
114   try {
115     if(debug) printArray(Serial.list());
116     int i = Serial.list().length;
117     if (i != 0) {
118       if (i >= 2) {
119         for (int j = 0; j < i;) {
120           COMlist += char(j+'a') + " = " + Serial.list()[j];
121           if (++j < i) COMlist += ",  ";
122         }
123         JDialog dialog = new JDialog();
124         dialog.setAlwaysOnTop(true);
125         COMx = JOptionPane.showInputDialog(dialog, "Seleccione el puerto COM al que desea conectarse:\n"+COMlist);
126         if (COMx == null)
127         exit();
128         if (COMx.isEmpty())
129         exit();
130         i = int(COMx.toLowerCase().charAt(0) - 'a') + 1;
131       }
132       set_portName(Serial.list()[i-1]);
133       if(debug) println(portName);
134       arduinoV = new Serial(this, portName, 9600);
135       arduinoV.bufferUntil('\n');
136     }
137     else {
138       JOptionPane.showInputDialog(frame,"El dispositivo no se encuentra conectado al PC");
139       exit();
140     }
141   }
142   catch (Exception e)
143   { //Print the type of error
144     JOptionPane.showInputDialog(frame,"El puerto COM port no esta disponible (puede estar utilizandose por otro programa)");
145     println("Error:", e);
146     exit();
147   }
148 }
```

Fragmento de código 9. Conexión al puerto serie de arduino. Fuente: Propia

En el *Fragmento de código 9* se define una lista con todos los puertos serie COM disponibles (instrucciones 112-122) y se crea una ventana emergente en la que mostrará la lista de puertos disponibles (instrucciones 123-125) y preguntará al usuario el puerto al que desea conectarse como se puede observar en la Figura 6.

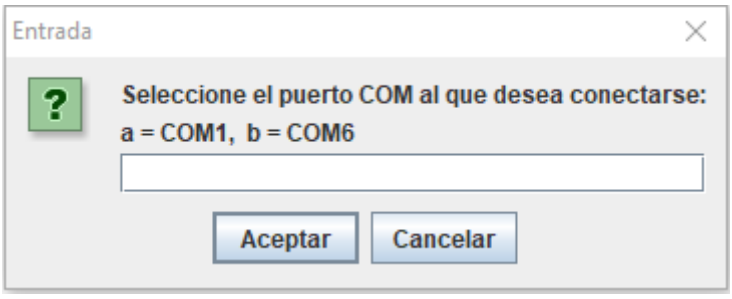


Figura 14. Selección del puerto Serie. Fuente: Propia

Una vez se ha recibido el valor introducido por el usuario, se considerarán los posibles casos de error de valores nulos y se asignará la variable indistintamente si se ha introducido mayúscula o minúscula (instrucciones 126-131).

A continuación, se abrirá el puerto elegido por el usuario siempre que sea posible. Se producen dos tipos de excepciones: una está referida a que el dispositivo, en este caso el Arduino no esté conectado al PC del servidor y la otra salta en el caso de que el puerto serie ya este ocupado por otro programa como puede ser el caso de un monitor Serial (instrucciones 132-148).

6.1.4 Funciones utilizadas en el loop *draw*

La función `draw` se trata de una función única de tipo loop ya que se ejecuta indefinidamente una y otra vez mientras este el programa en funcionamiento.

En esta función se van a utilizar las funciones que requieren estar constantemente en funcionamiento ya que pueden ser modificadas por el programa.

En primer lugar, mediante la función `background` se establece el color de la parte principal del programa según la escala de grises.

```
50 void draw() {
51
52   background(200);
53
54   add_shapes(); //Función para mostrar formas
55   add_texts(); //Función para mostrar textos
56   add_images(); //Función para mostrar imágenes
57   add_time(); //Función para mostrar la fecha y la hora
58
59
60 }
```

Fragmento de código 10. Función loop (draw). Fuente: Propia

La siguiente instrucción se trata de la función `add_shapes` en la cual se definirán las distintas formas del programa *Fragmento de código 11*. Estas formas son los tres rectángulos destinados a la agrupación de los botones, deslizadores e información respecto al canal serie y un círculo representando el valor de una variable.

```
77 void add_shapes() {
78   fill(230);
79   rect(155,145,350,350,15);
80   rect(575,145,350,350,15);
81   rect(20,20,140,80,10);
82 
83   inp_chk=byte(arduinoV.read());
84   chk=byte_to_int(inp_chk);
85 
86   if(chk != 0){
87   fill(0,255,0);
88 }
89   else{
90   fill(255,0,0);
91  }
92   ellipse(130,75,25,25);
93 }
```

Fragmento de código 11. Función que agrega las figuras del programa. Fuente: Propia

La primera instrucción indica la escala de grises que van a presentar los rectángulos mencionados anteriormente. Las instrucciones siguientes `rect` son las referidas a la creación al muestreo de los rectángulos indicando su posición, tamaño y curvatura de las esquinas respectivamente.

Por último, se lee un valor por el canal serie que se trata de un valor entero en el que indique si el valor del Checksum coincide. Este valor, como se explicará posteriormente, será enviado por el Arduino para validar que los datos se hayan recibido correctamente. Una vez conocido el valor de esta variable se modifica el color del círculo de manera que, si el Checksum es correcto, se establezca de color verde y si no lo es de color rojo (Figuras 7 y 8).

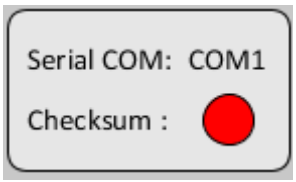


Figura 15. Checksum = false. Fuente: Propia

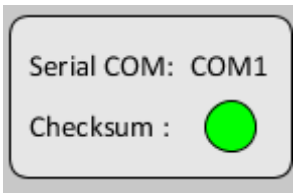


Figura 16. Cheksum = True. Fuente: Propia

La función de `add_texts` es la destinada a abrir todos los textos visibles en la interfaz excepto los mostrados previamente por el control P5.

```
179 void add_texts() {
180 
181   textFont(fontvis,30);
182   fill(0);
183   text("BOTONES",270,190);
184   text("DESLIZADORES",665,190);
185   text("Prácticas de electrónica digital",360,40);
186 
187 
188   float value1 = AV.getController("Potenciometro1").getValue();
189   textFont(fontvis,22);
190   fill(0);
191   text(value1,635,320);
192 
193   float value2 = AV.getController("Potenciometro2").getValue();
194   textFont(fontvis,22);
195   fill(0);
196   text(value2,635,450);
197   textFont(fontsecond);
198   text("Serial COM: ",30,50);
199   text(get_portName(), 110,50);
200   text("Checksum : ", 30, 80);
201   text(hex(controller),30,120);
202   text(hex(id),55,120);
203   text(hex(data),80,120);
204   text(hex(checksum),110,120);
205   text("Realizado por: Luis Alfonso Molina",840,700);
206 
207 }
```

Fragmento de código 12. Función que agrega los textos visibles de la interfaz. Fuente: Propia.

Se muestran los textos de título de la interfaz (“Prácticas de electrónica digital”), el de los distintos apartados de programa como botones o deslizadores. Además, se muestra el valor de los potenciómetros, así como el nombre del puerto serie actualmente utilizado, los valores en hexadecimal de la última instrucción dada al Arduino y el nombre del desarrollador de esta interfaz gráfica.

La instrucción siguiente es la función que muestra las imágenes constantemente. Se muestran la imagen del logo de la Universitat Politècnica de València y la Escuela Superior de Ingeniería del Diseño respectivamente (*Fragmento de código 13*). Se puede apreciar en la Figura 9.

```
 96 void add_images(){
 97
 98    imagen = loadImage("marca_UPV_secundaria_negro300.png");
 99    imagen2 = loadImage("etsid.png");
100
101    image(imagen,20,550,130,151);
102    image(imagen2,830,610,220,80);
103 }
```

Fragmento de código 13. Función que agrega los logos de la UPV y de la ETSID

La última función tiene como objetivo el muestreo de la fecha y la hora actual y se implementa mediante el código mostrado en la *Fragmento de código 14*. Se puede apreciar su aspecto en la Figura 9

```
335 void add_time(){
336
337   int sec = second();
338   String s,m;
339   if (sec<10) s="0"+str(sec);
340   else s=str(sec);
341
342   int min = minute();
343   if (min<10) m="0"+str(min);
344   else m=str(min);
345
346   int h = hour();
347
348   int d = day();
349   int mo = month();
350   int y = year();
351
352   String fecha;
353
354   fecha = str(d) + "/" + str(mo) + "/" + str(y) + "\n" + str(h) +":"+ m +":"+s;
355
356   textFont(fontsecond);
357   text(fecha,1000,25);
358
359 }
```

Fragmento de código 14. Función que agrega la fecha y hora local. Fuente: Propia

A continuación, se va a mostrar en la Figura 9 cómo queda la interfaz gráfica en su totalidad tras la implementación del código explicado previamente.

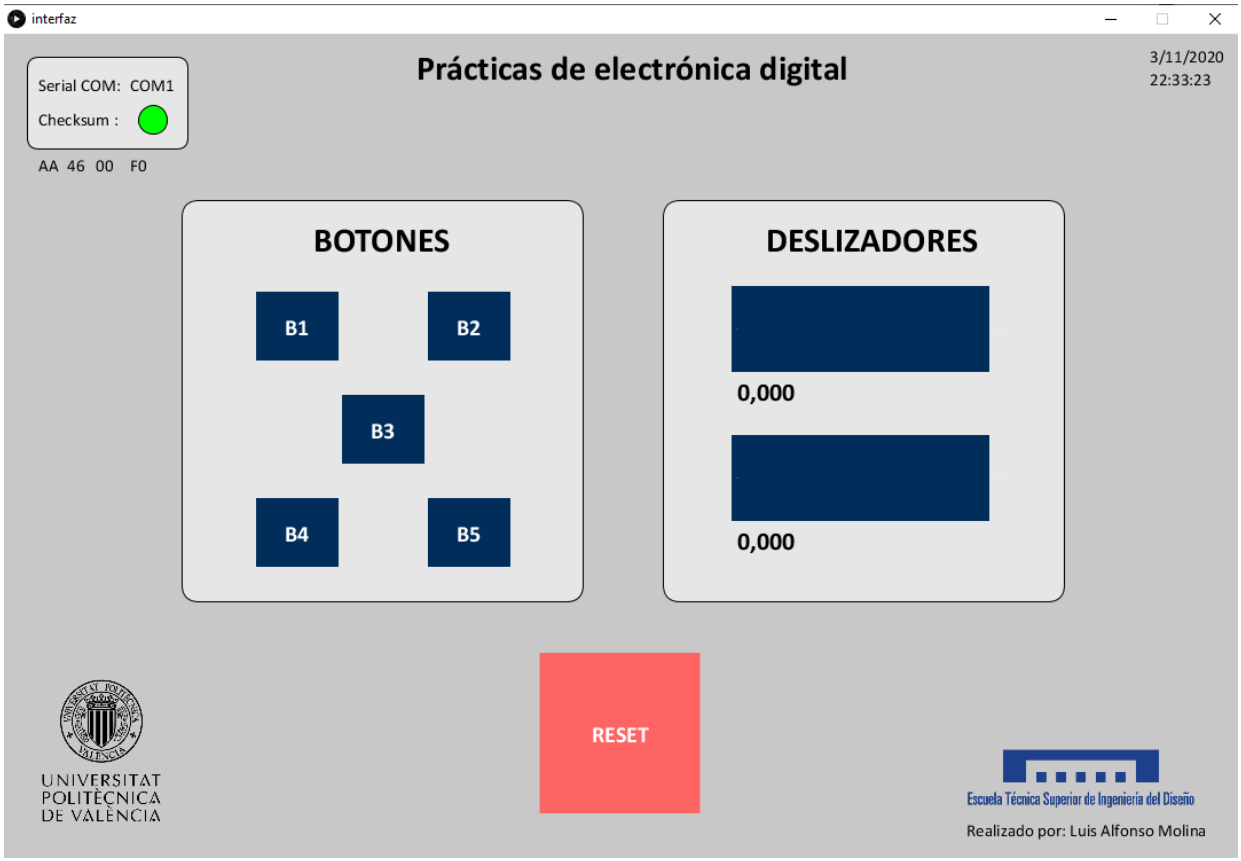


Figura 17. Interfaz gráfica completa. Fuente: Propia

6.1.5 Funciones destinadas al protocolo de comunicaciones

Las funciones secundarias que se van a explicar en el presente apartado reciben este nombre ya que no se utilizan directamente por el programa principal. Dos de estas funciones se tratan de un getter y un setter para leer y modificar el valor del nombre de la variable `portname` destinada a almacenar el puerto COM disponible elegido por el usuario cuyo código se puede observar en la *Fragmento de código 15*.

```
209 String get_portName(){
210   return portName;
211 }
212
213 void set_portName(String name){
214   portName = name;
215 }
```

Fragmento de código 15. Funciones "get" y "set" del portname. Fuente: Propia.

El siguiente tipo de funciones son las destinadas a mandar los datos al canal serie que posteriormente serán recibidos por el Arduino Virtual. Para ello, se va a implementar una función para cada uno de los componentes de la interfaz fija inicial por el usuario: seis para los botones y dos para los deslizadores.

Para la explicación de estas funciones, es necesario conocer los datos que se van a transmitir al Arduino. Estos están situados debajo del pequeño rectángulo de in formación del puerto serie y están compuestos de una cabecera llamada "`controller`", un identificador llamado "`id`", el valor del dato llamado "`data`" y una variable llamada "`checksum`" que depende de estas tres anteriores con el objetivo de la validación de la transmisión.

- Cabecera: Se trata de un valor constante donde se le asignará al dispositivo donde va recibida la transmisión.
- Identificador: Su objetivo es el de determinar el componente desde el que se manda la transmisión.
- Dato: Esta variable se encarga de almacenar el valor que se pretende mandar al Arduino Virtual. En el caso de los botones es 0 o 1 en función si está apagado o encendido y en el caso de los deslizadores es el valor escalado del rango de 0 a 255 que se mandará posteriormente mediante el bus I2C.
- Checksum: La variable "`checksum`" que se calcula en ambos programas, tanto el Processing como el firmware de Arduino, a partir de las tres variables anteriores con la intención de verificar que todos los datos se han mandado y recibido correctamente. Se calcula mediante la función llamada `Calculate_checksum` cuyo código se muestra en el *Fragmento de código 16*.

```
311 byte Calculate_checksum(byte c, byte i, byte d){
312
313    intc = byte_to_int(c);
314    inti = byte_to_int(i);
315    intd = byte_to_int(d);
316
317
318
319
320   return byte((intc + inti + intd)%256);
321
322 }
```

Fragmento de código 16. Función que devuelve el checksum. Fuente: Propia

En esta función mediante el casting realizado por la función `byte_to_int(byte)`, el código de la cual se muestra en la *Fragmento de código 17*, se consigue operar correctamente con valores enteros y posteriormente realizar el casting opuesto `byte(int)`. La operación tiene como resultado el módulo de la suma de las tres variables.

```
324 int byte_to_int (byte b){
325
326   int n = int(b);
327   n = n + 0xFF +1;
328   return n%256;
329
330 }
```

Fragmento de código 17. Función que transforma una variable de tipo byte a una de tipo int. Fuente: Propia

Una vez explicados los datos de la transmisión, se puede explicar las funciones que se encargan de esta transmisión. Estas funciones se denominan `handle.click` ya que se activan únicamente mediante el click izquierdo del ratón en los componentes deseados. Todas ellas mandan las variables previamente explicadas y modificadas correctamente según el componente seleccionado como se puede apreciar en el código del Anexo 4.

Cabe destacar que tanto los valores de la variable del identificador como el del dato son bastante intuitivos. Además, el valor de los potenciómetros se escala como se ha comentado previamente del rango de valores que puede seleccionar el usuario al del potenciómetro digital. También, que las cuatro variables transmitidas cada vez que entra a una de estas funciones son de tipo byte para facilitar el proceso de lectura del Arduino Virtual ya que lee byte a byte.

6.2 Implementación del firmware (microcontrolador-interfaz)

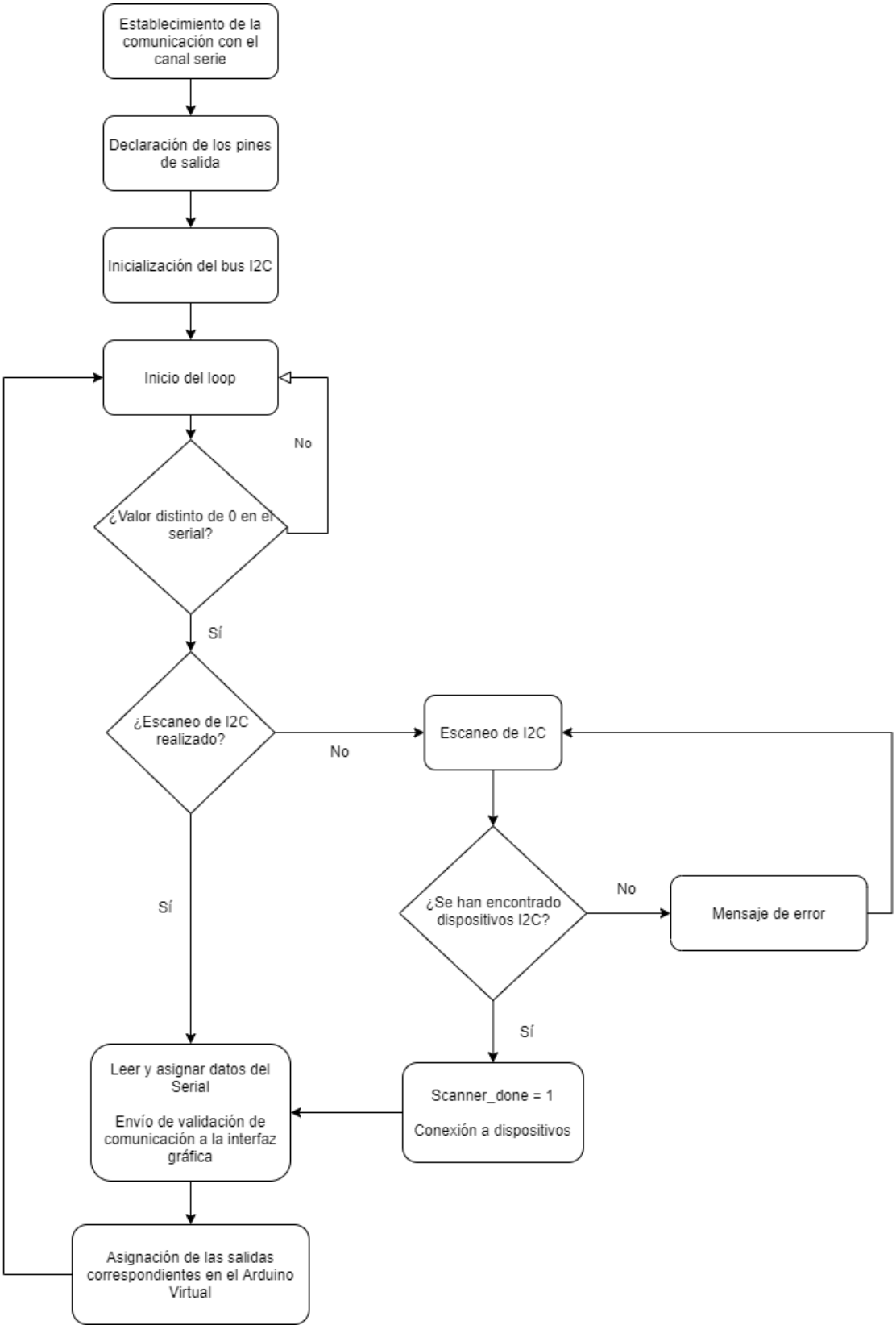


Diagrama de flujo 3. Funcionamiento general del firmware del Arduino Virtual. Fuente: Propia

En la página anterior, en el *Diagrama de flujo 3* se puede ver el funcionamiento a grandes rasgos del programa explicado durante el presente apartado.

Mediante el firmware de Arduino se logra el control del Arduino Virtual. Por lo tanto, en el caso del presente proyecto, este programa va a tener la función de transmitir los datos recibidos por la interfaz a las acciones del microcontrolador Arduino correspondientes. Además, confirma que los datos recibidos por el puerto Serial se hayan recibido correctamente.

Para la correcta explicación del código se va a dividir el presente apartado en varios subapartados destinados a explicar la parte del programa en cuestión.

6.2.1 Librerías y declaración de variables

En este apartado se va a realizar una breve explicación de las librerías incluidas para el funcionamiento, así como la definición de las variables definidas.

```
#include <Wire.h> //incluir librería Wire.h

//Declaración de variables
 //Declaración de las constantes
const int button1_pin = 13;
const int button2_pin = 12;
const int button3_pin = 14;
const int button4_pin = 10;
const int button5_pin = 9;
const int RESET_pin = 4;
 //Declaración de las variables modificables
int byteCount = 1;
uint16_t C,I,D,CHK,checksum;;
bool checksum_ok;
int val_pot1=0, val_pot2=0;
int address_pot1, address_pot2;
int Scanner_done;
uint8_t cid_c;
```

Fragmento de código 18. Libreria Wire.h y declaración de variables. Fuente: Propia

Como se puede observar en la *Fragmento de código 18*, en el programa que se está tratando únicamente se incluye una librería llamada `Wire.h` cuya función es permitir el correcto funcionamiento del bus I2C.

Tras esta instrucción, se ubica la declaración de variables, en las cuales se puede distinguir unas variables constantes que contienen el valor correcto de los pines de los botones. Y también unas variables modificables cuyo fin es la comunicación con el canal Serial y el bus I2C.

6.2.2 Funciones utilizadas en el setup

La función `setup` del firmware de Arduino comparte las mismas características que la explicada previamente de Processing. En esta se pueden distinguir las funciones que la componen (*Fragmento de código 19*).

```
void setup() {

  Serial.begin(9600); //Inicia la comunicación Serial a 9600 bits
por segundo

  Declare_outputs(); //Declaración de los pines de output de los
botones

  Wire.begin(); //Inicialización del canal wire

  while (!Serial); //Espera hasta que se produzca comunicación con
el puerto serie

  Serial.println("\nScanner I2C"); // Muestra que ya se ha
producido comunicación y empieza el proceso de scaneado
```

Fragmento de código 19. Función Setup Arduino. Fuente: Propia

En primer lugar, mediante la instrucción `Serial.begin(9600)` el Arduino Virtual inicia la comunicación con el puerto Serial con una velocidad de transmisión de 9600 bits por segundo.

La siguiente instrucción se trata una función que tiene la funcionalidad de declarar los pines que van a ser salidas del Arduino Virtual. Los números de estos pines son los mismos que las salidas del Arduino Virtual en el esquemático. El código para la implementación de esta función se puede apreciar en la *Fragmento de código 20*.

```
void Declare_outputs(){

  pinMode(button1_pin, OUTPUT);
  pinMode(button2_pin, OUTPUT);
  pinMode(button3_pin, OUTPUT);
  pinMode(button4_pin, OUTPUT);
  pinMode(button5_pin, OUTPUT);

  pinMode(RESET_pin, OUTPUT);

  }
```

Fragmento de código 20. Declaración de las salidas del Arduino. Fuente: Propia.

La función `Wire.begin` llamada en el `setup` del programa (*Fragmento de código 19*) inicia el canal Wire, es decir, permite el Scanner y la comunicación con dispositivos I2C.

A continuación, se sitúa la instrucción de un bucle mientras no se produzca una comunicación con el canal Serial. Por ello el programa no ejecutará el resto de instrucciones hasta no establecer dicha comunicación.

La última línea de código muestra por el canal Serial que está todo preparado para ejecutar el Scanner I2C.

6.2.3 Funciones utilizadas en el *loop*

La función `loop` del software Arduino es similar a la función `draw` del Processing ya que es el bucle principal del programa. En este bucle, tras la comprobación de que el canal Serial tenga cualquier valor (`if(Serial.available())`), se implementan una serie de funciones con el objetivo de recibir datos del interfaz e indicar el valor y la salida especificada en cada posible caso. Se puede ver la implementación del código del bucle principal en la *Fragmento de código 21*.

```
void loop() {


  if(Serial.available()){


   while(Scanner_done==0){Scanner_done = Scanner_I2C();}//Espera a
que se haya realizado el scanner

   cid_c = Serial.read(); //Lee byte a byte los datos llegados del
canal Serial y los asigna a la variable cid_c

   Read_interface_data(); //Asigna las variables llegadas de la
interfaz correctamente

   Set_outputs_value(); // Asigna el valor correspondiente de las
salidas del Arduino en cada caso

 }

}
```

Fragmento de código 21. Función loop de Arduino. Fuente: Propia

La primera de estas funciones se denomina `Scanner_I2C` y es la encargada del escaneado de todos los dispositivos I2C conectados al Arduino Virtual y la correcta asignación de las direcciones de esto. La implementación de esta función se puede observar en el Anexo 5.

La función devuelve un entero definido como `Scanner_Done` que se inicializa en 0 y se establece como 1 una vez se haya realizado el scanner y se hayan asignado las direcciones correctamente. También se declaran otras variables importantes para el

funcionamiento del scanner como son la dirección del bus que se está consultando en cada iteración, la variable error devuelta por los dispositivos I2C, así como el número de dispositivos registrados por el scanner.

Tras la declaración de estas variables, se implementa un bucle `for` en el que se recorren todas las direcciones posibles del bus I2C una a una. Se establece conexión con la dirección de la iteración actual y comprueba si se reconoce un dispositivo.

En el caso de que la función `Wire.endTransmission` devuelva un valor de 0, significa que un dispositivo con la dirección actual ha reconocido la comunicación del canal I2C. Tras este reconocimiento se incrementará el valor de la variable `nDevices` y se asignará la dirección de memoria al dispositivo correspondiente (`address_pot (1 o 2) = address`). Después de realizar dicha asignación se imprimirá por el canal Serial el valor de la dirección actual.

Por el contrario, si la función `Wire.endTransmission` devuelve un valor de 4 significará que se ha producido un error des conocido en la dirección actual.

Una vez realizadas todas las iteraciones del bucle, se comprobará el número de dispositivos registrados y se imprimirá por pantalla si el scanner ha sido satisfactorio o no. En el caso de ser satisfactorio se asignará el valor de 1 a `Scanner_done` que, ya que la función `Scanner_I2C` está implementada de forma que hasta que no devuelva un valor distinto de 0 se mantenga en un bucle (Fragmento de código 21), permitirá salir de dicho bucle.

La posterior instrucción se encarga en la lectura del valor del byte actual recibido por el canal Serial y su asignación a la variable `cid_c`. Esta variable irá tomando el valor de las distintas variables llegadas de la interfaz siempre que el usuario las modifique mediante la interfaz gráfica. Este procedimiento se explicará correctamente y con más detalle en la siguiente función.

La función siguiente tiene como objetivo asignar las 4 distintas variables llegadas cada vez que el usuario realiza una modificación mediante la interfaz. La implementación se hace mediante el código que puede visualizarse en el *Fragmento de código 22*.

```
void Read_interface_data(){
   switch(byteCount){
   case 1:
     C = cid_c;
     byteCount++;
     if (C!=0xAA)
     byteCount=1;
     break;
   case 2:
     I = cid_c;
     byteCount++;
     break;
   case 3:
     D = cid_c;
```

```
      byteCount++;
      break;
    case 4:
      CHK = cid_c;
      checksum=(C+I+D)%256;

      if(CHK==checksum) checksum_ok=true;
      else checksum_ok=false;

      byteCount=1;
      break;


  }Serial.print(uint8_t(checksum_ok));
}
```

Fragmento de código 22. Función de la lectura de los datos de la interfaz Processing y asignación a las variables correspondientes. Fuente: Propia

Este código está compuesto por un `switch` que depende de la variable contador `byteCount` que depende de cada iteración realizada por el bucle principal que determina si se encuentra en un caso o en otro. Primero, se asigna el valor de la cabecera comprobándose que sea el mismo dispositivo asignado en el Processing para poder continuar e incrementa el valor del contador. En segundo lugar, asigna el valor identificador e incrementa el contador. En el tercer caso, se repite la estructura del caso anterior, pero asignando el valor a la variable de datos. En último lugar, se asigna el valor del Checksum (`CHK`) y se compara con el Checksum calculado por las variables asignadas a lo largo de las 3 iteraciones anteriores del `switch` y asignará el valor de la variable `checksum_ok` en función de si coinciden ambos valores o no. Tras esto, se le enviará el valor de dicha variable al Processing para la validación de la transmisión.

La última función utilizada en el bucle `loop` es la destinada a establecer el valor de las salidas necesario en cada caso para el correcto funcionamiento del Arduino Virtual. Se llama `Set_outputs_value` y se va implementar como se aprecia en el *Fragmento de código 23*.

```
void Set_outputs_value(){
  if (byteCount==4){
    digitalWrite(button1_pin,LOW);
    digitalWrite(button2_pin,LOW);
    digitalWrite(button3_pin,LOW);
    digitalWrite(button4_pin,LOW);
    digitalWrite(button5_pin,LOW);
  //Se establece el valor de los pines destinados a los botones a 0
(LOW)


    Send_to_outputs_ArduinoV(I,D); //Cada vez que se envié una
instrucción relacionada con cualquiera de los seis botones
se activará el pin especificado
```

```
    if((I==60)||(I==70)){
    if(I==60)val_pot1= D;
    else val_pot2= D;
    I2C_connection();// Se establece conexión con el dispositivo
I2C que se pretende modificar, se transmite el dato y se cierra la
conexión
    }
  }
}
```

Fragmento de código 23. Implementación de las salidas del Arduino virtual. Fuente: Propia

Se llamará a esta función una vez se hayan realizado todos los casos del `switch` de la función `Read_interfaz_data`. La primera etapa corresponde al establecimiento de todos los pines de los botones excepto el del Reset al estado bajo. Una vez realizado este establecimiento se usará la función denominada `Send_to_outputs_ArduinoV` a la cual se le introducirá los valores del identificador y del dato estableciendo las salidas objetivo al valor correspondiente en cada caso mediante el código mostrado en el *Fragmento de código 24*.

```
void Send_to_outputs_ArduinoV(uint16_t  id, uint16_t dat){
  switch(id){
  case 10: if (dat==1)
          digitalWrite(button1_pin,HIGH);
          break;

  case 20: if (dat==1)
          digitalWrite(button2_pin,HIGH);
          break;

  case 30: if (dat==1)
          digitalWrite(button3_pin,HIGH);
          break;

  case 40: if (dat==1)
          digitalWrite(button4_pin,HIGH);
          break;

  case 50: if (dat==1)
          digitalWrite(button5_pin,HIGH);
          break;

  case 100: if (dat==1)
          digitalWrite(RESET_pin,HIGH);
          delay(10);
          digitalWrite(RESET_pin,LOW);
          break;
   }
}
```

Fragmento de código 24. Función envío de salidas al Arduino virtual. Fuente: Propia

Esta función depende del identificador de la lectura actual establece el valor de los pines correspondientes y en el caso del pin al cual se asigna el Reset se activará un breve momento de tiempo suficiente para la activación del Reset del Arduino Programable y después se volverá a poner a estado bajo.

Finalmente, se comprueba si el identificador actual se trata de uno de los dos potenciómetros digitales (deslizadores) y le asigna el dato actual al identificador del deslizador potenciómetro en cuestión. Una vez realizada esta asignación se llamará a la función `I2C_connection` encargada de establecer conexión con ambos potenciómetros digitales y actualizar del potenciómetro con el identificador actual. Esta última función presenta el código siguiente (*Fragmento de código 25*).

```
void  I2C_connection(){

  Wire.beginTransmission(address_pot1); // transmitir a la
dirección del potenciómetro digital 1

  Wire.write(val_pot1);              // enviar valor al
potenciómetro digital 1

  Wire.endTransmission();     // cerrar transmisión


  Wire.beginTransmission(address_pot2); // transmitir a la
dirección del potenciómetro digital 2

  Wire.write(val_pot2);              // enviar valor al
potenciómetro digital 2

  Wire.endTransmission();     // cerrar transmisión

  }
```

Fragmento de código 25. Conexión I2C de los potenciómetros digitales

En esta función se distingue claramente la implementación de su funcionalidad: establecer la conexión con el dispositivo I2C correspondiente, enviar el valor establecido previamente por el usuario y finalmente, el cierre de dicha transmisión. Esta operación se repite con ambos potenciómetros.

7. Fabricación

La fabricación de una placa de circuito impreso conlleva realizar una serie de pasos para llegar al producto final. En este apartado se van a explicar los diversos procedimientos realizados y el orden en el que han sido realizados en la fabricación de la PCB de dos capas.

La placa en la que se va a implementar el diseño de la PCB realizado en el presente proyecto se trata de una placa con una superficie de material fotosensible compuesta por un barniza, seguidamente las dos placas de cobre que posteriormente se convertirán en las capas de Top y Bot y finalmente, un material no conductor que separe estas dos capas.

7.1 Fototransmisión del diseño del layout

En primer lugar, se imprime el diseño del layout en unas láminas transparentes mediante una impresora de alta precisión. Se imprimen ambas capas del diseño, tanto Top como Bot.

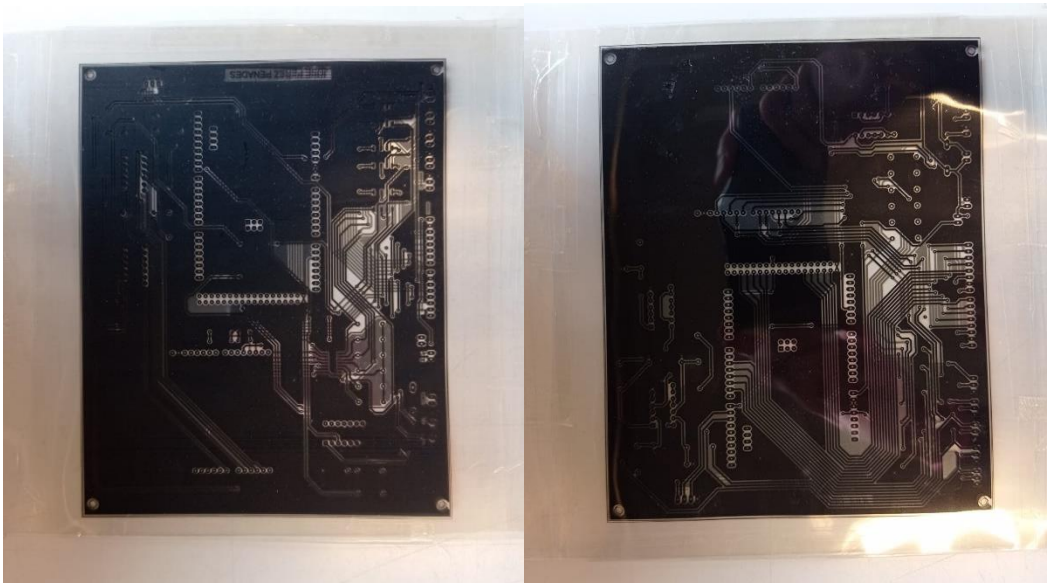


Figura 18. Impresión del diseño de la PCB en las láminas transparentes. Fuente: Propia.

Estas láminas se deben alinear de manera que coincidan todas las pistas y vías y, posteriormente, se fijan a la placa con el material fotosensible.

El siguiente paso es transferir el diseño del layout a la placa por insolación. Esto se realiza mediante la exposición de la placa a una potente luz ultravioleta, mediante la cual se carga eléctricamente la capa fotosensible solamente en las zonas donde las láminas con

el diseño impreso son transparentes. Este proceso es realizado por la máquina insoladora cuyo funcionamiento se explica en el apartado 2.3.

Una vez realizado este tratamiento es conveniente que la placa no quede expuesta a ninguna fuente de luz hasta el siguiente proceso.

7.2 Tratamiento químico

En esta etapa de fabricación se van a realizar varios procesos químicos con el fin de eliminar el cobre en las zonas no conductoras del diseño.

Para ello se utiliza el equipo de protección individual adecuado: guantes de latex, bata, gafas y mascarilla, realizándose todo el proceso dentro de una campana extractora con cristal de protección para evitar la inhalación de gases y minimizar el riesgo por salpicaduras.

Se empieza por el proceso de revelado en el que se sumerge la placa en una disolución a base de agua y revelador (sosa caustica) en un recipiente de plástico. Se debe zarandear el recipiente de manera que el revelador este repartido a lo largo de la superficie de la placa durante treinta a sesenta segundos hasta apreciar claramente el diseño. Se voltea la placa y se repite el mismo proceso. Con este proceso se consigue la generación de un cubrimiento protector en la zona no insolada. Este recubrimiento dejará más expuestas las áreas insoladas, que en el siguiente pasado sufrirán un ataque ácido.

Al final de este proceso, hay que tener especial atención en el lavado de la placa con abundante agua para asegurarse que no queden restos de sustancia básica.

A continuación, se realiza el proceso conocido como atacado en el cual la placa se sumerge en una disolución de agua con ácido clorhídrico (HCl) y con dióxido de hidrógeno (H_2O_2), conocido como agua oxigenada, pero en este caso con una concentración considerablemente mayor. De una manera similar al proceso anterior, se debe zarandear el recipiente que contiene la disolución y la placa, tomando todas las medidas de seguridad necesarias. Este proceso dura escasos segundos y requiere de supervisión visual y reacción inmediata. Una vez se haya desprendido el cobre no cubierto por el barniz por ambas capas de la placa se debe retirar para evitar que ataque al resto de cobre y lavar con agua abundante.

Finalmente, se le aplica acetona ($CH_3(CO)CH_3$) en ambas superficies de la placa para eliminar el recubrimiento básico de los planos, pistas y vías, y se limpia con abundante agua para terminar el proceso de desprendimiento del cobre.

7.3 Taladradora CNC

La siguiente etapa de fabricación tiene como fin el agujereado de la placa en las zonas deseadas, que son las vías y las posteriores conexiones through-hole. Para ello, se pasa

el diseño del layout con la ubicación y el diámetro de cada perforación a un programa encargado del funcionamiento de una taladradora por CNC. El funcionamiento de esta máquina se explica en el apartado 2.3.

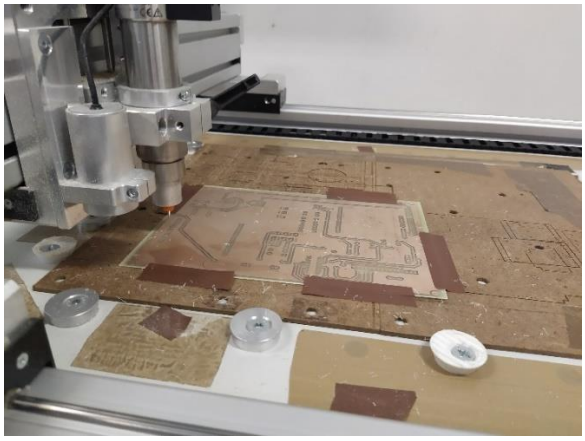


Figura 19. Taladrado de la placa por CNC. Fuente: Propia

El siguiente paso es introducir puntos de referencia a la taladradora mediante la cámara que esta presenta. Estos puntos son el punto inicial y la esquina opuesta. Una vez introducidos los puntos de referencia la taladradora empieza el proceso de taladrado. La taladradora coge una de las brocas con el diámetro especificado en el diseño y cuando haya perforado todos los agujeros de ese diámetro deja la broca. Este proceso se repite con todas las brocas necesarias para conseguir el diseño.

7.4 Metalizado de las vías

Una vez realizadas todas las perforaciones necesarias, se procede a metalizar las vías. Para ello, se debe disponer de un cable conductor, estaño y una soldadora. Se atraviesa la vía con el cable y se suelda en ambas capas de la placa cortando el cable excedente. Este proceso se realiza en todas las vías del diseño.

7.5 Soldado de los componentes

Previamente a realizar cualquier soldadura de los componentes que se van a mencionar a lo largo de este apartado hay que revisar exhaustivamente de que se esté colocando en la misma orientación de la especificada en el diseño, si hiciera falta comprobarla correcta orientación en el datasheet del componente en cuestión. Además, para la realización de las diferentes soldaduras se debe seguir una serie de especificaciones

redactadas en el apartado 5.4.2. El funcionamiento de la soldadora se encuentra especificado en el apartado 2.3

7.5.1 Soldado de los componentes SMD

Los componentes SMD son los denominados Surface Mounted Device ya que se sueldan directamente a la superficie. En primer lugar, se colocan en la orientación adecuada y se suelda un pin para fijarlo en la placa. Por otro lado, se sueldan el resto de los pines con especial atención en no provocar algún cortocircuito en los pines ya que se encuentran separados por una distancia bastante reducida.

Los componentes SMD del diseño son los potenciómetros digitales B2541BR que presentan 8 pines y se sitúan en la parte superior izquierda de la placa, en la capa Top, como se puede observar en la Figura 20.

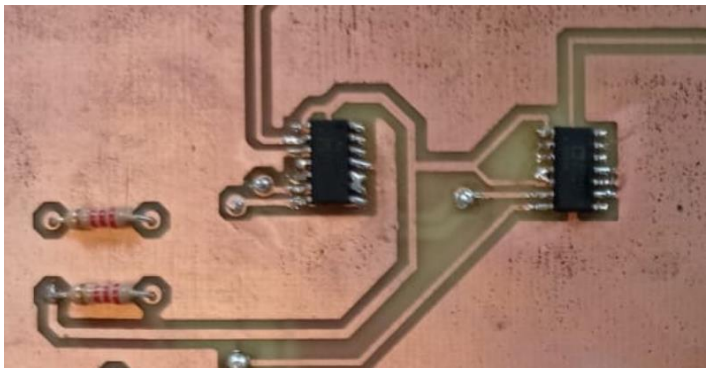


Figura 20. Potenciómetros digitales. Fuente: Propia

Cabe destacar que en el diseño se especificaba que el amplificador operacional era un componente con un package SMD que finalmente se ha implementado con un package Through-Hole con el mismo procedimiento de soldadura que los potenciómetros digitales. Además, se ha realizado una lijadura para evitar la conductividad de la placa en la capa Bot en las posiciones de los pines.

7.5.2 Soldado de los componentes Through-Hole

Los componentes Through-Hole son los referidos a la tecnología de los agujeros pasantes. Los pines de estos se insertan en las perforaciones desde la capa de Top a la capa Bot. Se sueldan en los pines que reciban pistas ya sea por una capa u otra debido a que, a diferencia de los componentes SMD, los componentes Through-Hole pueden recibir pistas tanto por Top como por Bot.

Los componentes del diseño de este proyecto son mayoritariamente de este tipo. Estos son los Arduinos, los conectores, el expansor de entradas y salidas para la pantalla LCD, la propia pantalla LCD, el transistor del control de velocidad del motor, el amplificador operacional, el sensor de temperatura, las resistencias y los LEDs.

Cabe destacar que las conexiones a los Arduinos se han realizado con la ayuda de pines “macho-macho” como se puede apreciar en la figura x, y en el caso de la pantalla LCD, con la ayuda de unos pines “macho-hembra”.

8.Resultados

A lo largo de este apartado, se van analizar los resultados tras la realización del proyecto. Para ello, se va a comparar este resultado con las especificaciones definidas en el apartado 3 punto por punto.

1. **Debe constar de un microcontrolador programable por el usuario final.**
   Este se trata del microcontrolador referido en el proyecto como Arduino Programable ubicado en la parte superior de derecha de la PCB (Anexo 2).

2. **El microcontrolador programable debe poder recibir estímulos en sus pines desde la interfaz gráfica de usuario.**
   Este microcontrolador está conectado directamente o indirectamente a los pines del Arduino Virtual (Anexo 3), cuyos dichos pines están controlados por la GUI.

3. **Sistema fácilmente reproducible: materiales y elementos disponibles y económicos.**
   Todos los componentes del sistema se pueden comprar con facilidad en las principales webs de componentes electrónicos. Estos ascienden a una cantidad ligeramente superior a 80 euros que va decrementando en función de las unidades que se reproduzcan.

4. **Accesibilidad total al software y al hardware.**
   El laboratorio remoto implementado en este proyecto está diseñado con una interfaz gráfica de usuario muy intuitiva con un estructura simple y visual. Además, se presenta un sistema en el que no relevante el sistema auditivo.

   Gracias a que el Arduino Uno, se trata de un hardware libre programado por un software libre se somete a un proceso de revisión pública que mejora ampliamente el dinamismo del proceso de corrección de errores.

5. **Bus I2C para posibilitar la incorporación de ampliaciones tanto de estímulos controlados por la interfaz gráfica de usuario como de módulos controlables por el microcontrolador programable.**
   Esto se consigue mediante la implementación de dos conectores del bus I2C, uno para para cada bus de los Arduinos. El conector ubicado en la parte izquierda se conecta al bus del Arduino Virtual (Figura 21), mientras que el de la derecha se conecta al bus del Arduino Programable (Figura 22).

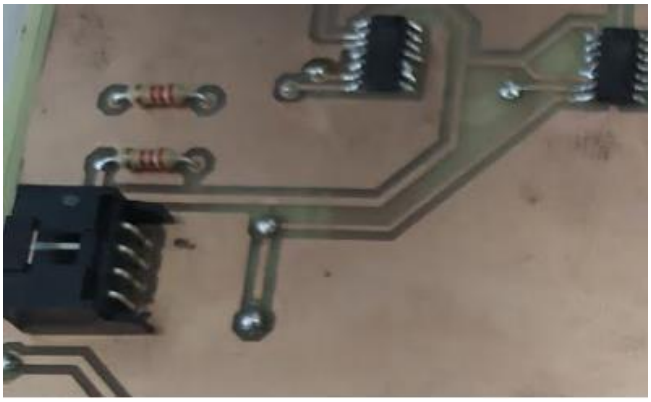


Figura 21. Conector del bus I2C del Arduino Virtual. Fuente: Propia.

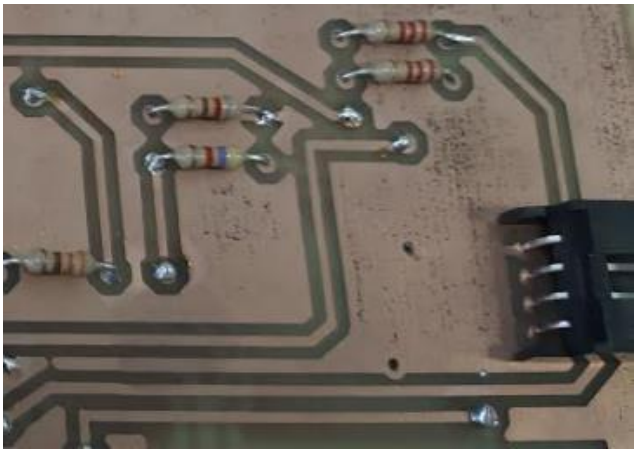


Figura 22. Conector del bus I2C del Arduino Programable. Fuente: Propia.

6. El microcontrolador programable debe incluir:

- Driver para control de motor DC por PWM: Como se ha especificado en el apartado 5.1.1 y 5.1.2 el Arduino Uno presenta varias salidas digitales con un control PWM en una de las cuales se conecta el transistor encargado de aportar la corriente necesaria al motor. Mediante estas salidas y la programación del Arduino IDE con las librerías de Arduino en las que se facilita el establecimiento de este control.

- Capacidad de medida mediante ADC integrado del valor regulado de tensión para el motor: Se consigue mediante el uso de un amplificador operacional con dos seguidores de tensión y, finalmente, un divisor de tensión encargado de escalar la tensión de la alimentación del motor (12V) a la máxima medible por la entrada analógica del Arduino (5V), que presenta un convertidor ADC integrado.

- Medición de temperatura: El dispositivo TMP36 ubicado próximo al Arduino Programable (Figura 23) tiene como funcionalidad la medición de la temperatura ambiente convirtiendo este en un voltaje proporcional medido por una entrada analógica del Arduino Programable.

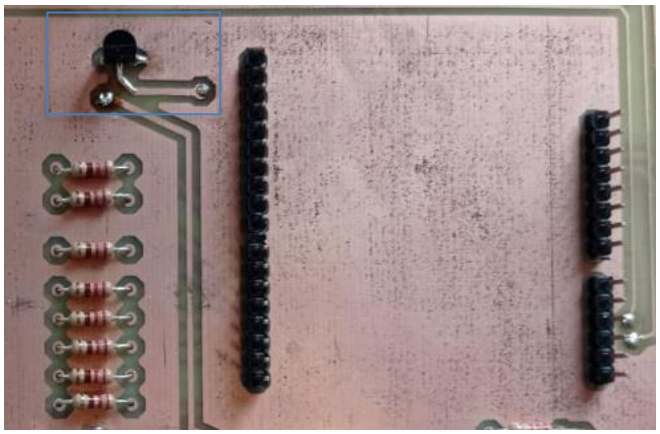


Figura 23. Sensor de temperatura TMP36. Fuente: Propia

- Medición de dos valores analógicos mediante ADCs integrados, modificables desde la interfaz gráfica de usuario: En la interfaz gráfica de usuario se encuentran dos deslizadores cuyo valor se envía a través del bus I2C del Arduino Virtual a dos potenciómetros digitales que varían su resistencia en función del valor de estos deslizadores (Figura 20). Las salidas de los potenciómetros van conectadas a entradas analógicas del Arduino Programable, con el conversor ADC incorporado.

- Medición de cinco botones por GPIO controlados desde la interfaz gráfica de usuario: En la GUI también se presentan 5 botones que en el caso de ser pulsados establecerán a nivel alto de la salida digital del Arduino Virtual correspondiente a ese botón. Estas cinco salidas digitales se conectan en serie con una resistencia cada una y con cinco salidas digitales del Arduino Programable.

- Tres LED conectadas a GPIO con resistencias de limitación de corriente adecuadas: Tres salidas digitales GPIO del Arduino van conectadas en serie con una resistencia y un LED cada una y, finalmente, a tierra (Figura 24). Las resistencias son de un valor de 220 Ω para conseguir una corriente adecuada para los LEDs (23mA).

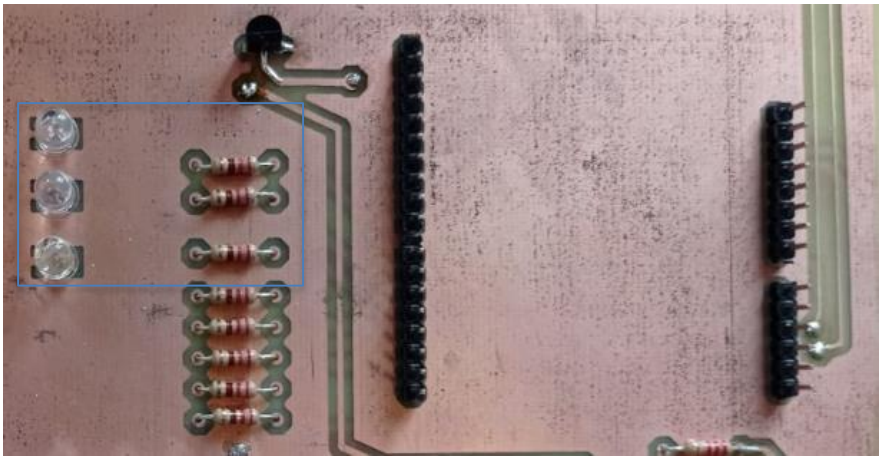


Figura 24. LEDs con sus respectivas resistencias. Fuente: Propia

- Pantalla controlada por I2C: Como se explica con más detalle en el apartado 5.1.6, la pantalla LCD se controla por un expansor de entradas y salidas regulado por el bus I2C del Arduino Programable. La pantalla LCD va conectada a los pines hembra visibles en la Figura 25.

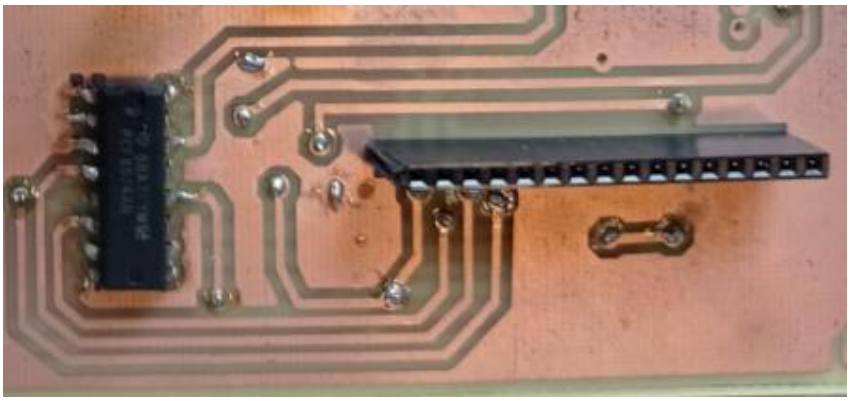


Figura 25.Expansor de E/S y pantalla LCD. Fuente: Propia

- Posibilidad de RESET del microcontrolador programable a través de la interfaz gráfica de usuario: La interfaz gráfica de usuario presenta un botón llamado RESET con la funcionalidad de cambiar el estado de una de las salidas digitales del Arduino Virtual de bajo a alto. Por último, esta salida digital va directamente conectada al pin RESET del Arduino Programable.

9.Conclusiones

En este proyecto se ha desarrollado un laboratorio remoto para la asignatura de Electrónica Digital, de acuerdo con las especificaciones indicadas para que sea adecuado en otras asignaturas del mismo y otros grados relacionados. En particular, representa una adaptación a las nuevas necesidades de semipresencialidad producidas por la crisis sanitaria ocasionada por el virus SARS-CoV-2, que ha repercutido en todos los ámbitos de la sociedad. En el ámbito de la educación, las asignaturas de un carácter mayormente práctico, entre ellas las científico-técnicas que implican la manipulación de instrumental de laboratorio, han sido las que han requerido de una adaptación mayor.

Los simuladores han sido un potente aliado para una adaptación rápida de estas prácticas de laboratorio, pero a su vez han mostrado ciertas limitaciones que afectan a la calidad del aprendizaje, en particular que estos se encuentran más distanciados de la práctica profesional física (uso de materiales, instrumental y dispositivos). La alternativa propuesta es el uso de laboratorios remotos ya que mediante el uso de estos laboratorios se consigue la teleoperación del instrumental y materiales del laboratorio físico. Es decir, el estudiante, aun telemáticamente, tiene las mismas posibilidades de actuación y estímulos visuales y auditivos que si se encontrara presencialmente en el laboratorio, realizando la práctica de laboratorio de forma tradicional.

El laboratorio remoto desarrollado en este proyecto está compuesto por una placa de circuito impreso controlada remotamente por una interfaz gráfica de usuario. Esta placa lleva incorporados dos microcontroladores Arduino Uno, uno encargado de recibir los datos correspondientes de la interfaz y el otro, conectado a todos los periféricos necesarios para el desarrollo de las prácticas, programable por el estudiante que teleoperará la placa.

La interfaz gráfica presenta diversos elementos (botones y deslizadores) para enviar los estímulos necesarios al microcontrolador programable: cinco estímulos digitales, dos estímulos analógicos y la posibilidad de resetearlo. Para ello, la interfaz gráfica de usuario del ordenador, mediante comunicación serie, comanda un microcontrolador que traduce los comandos en estímulos. Con este objetivo se ha realizado un protocolo de comunicaciones a nivel de aplicación y un firmware para el microcontrolador que hace de puente entre la interfaz y el microcontrolador programable.

El microcontrolador programable por el estudiante presenta conectado varios periféricos para la realización de las prácticas:

- Un sensor de temperatura.
- Un motor DC controlado por PWM.
- Tres LEDs.
- Una pantalla LCD controlada por un expansor de E/S basado en I2C.
- Un conecto I2C para permitir la inclusión de más periféricos en un futuro.

Con todo esto y con la visualización de la placa mediante una cámara se consigue un laboratorio remoto con una experiencia muy cercana al laboratorio presencial. Dada la situación presente, la implementación de este laboratorio remoto es un aporte

sustancial en las asignaturas de este ámbito debido a que permite realizar las prácticas de laboratorio de una forma interactiva y física manteniendo las ventajas del aprendizaje telemático.

El desarrollo de los laboratorios remotos permite disponer de un entorno de aprendizaje equivalente a las prácticas de laboratorio presenciales en la programación de microcontroladores, con mayor accesibilidad y disponibilidad. Más allá del cubrimiento de necesidades inmediatas, este proyecto constituye un paso adelante en el desarrollo de nuevas y mejores formas de enseñanza y aprendizaje en la educación superior científico-técnica.

10. Referencias

1. Collins, A., & Halverson, R. (2018). *Rethinking education in the age of technology: The digital revolution and schooling in America*. Teachers College Press.

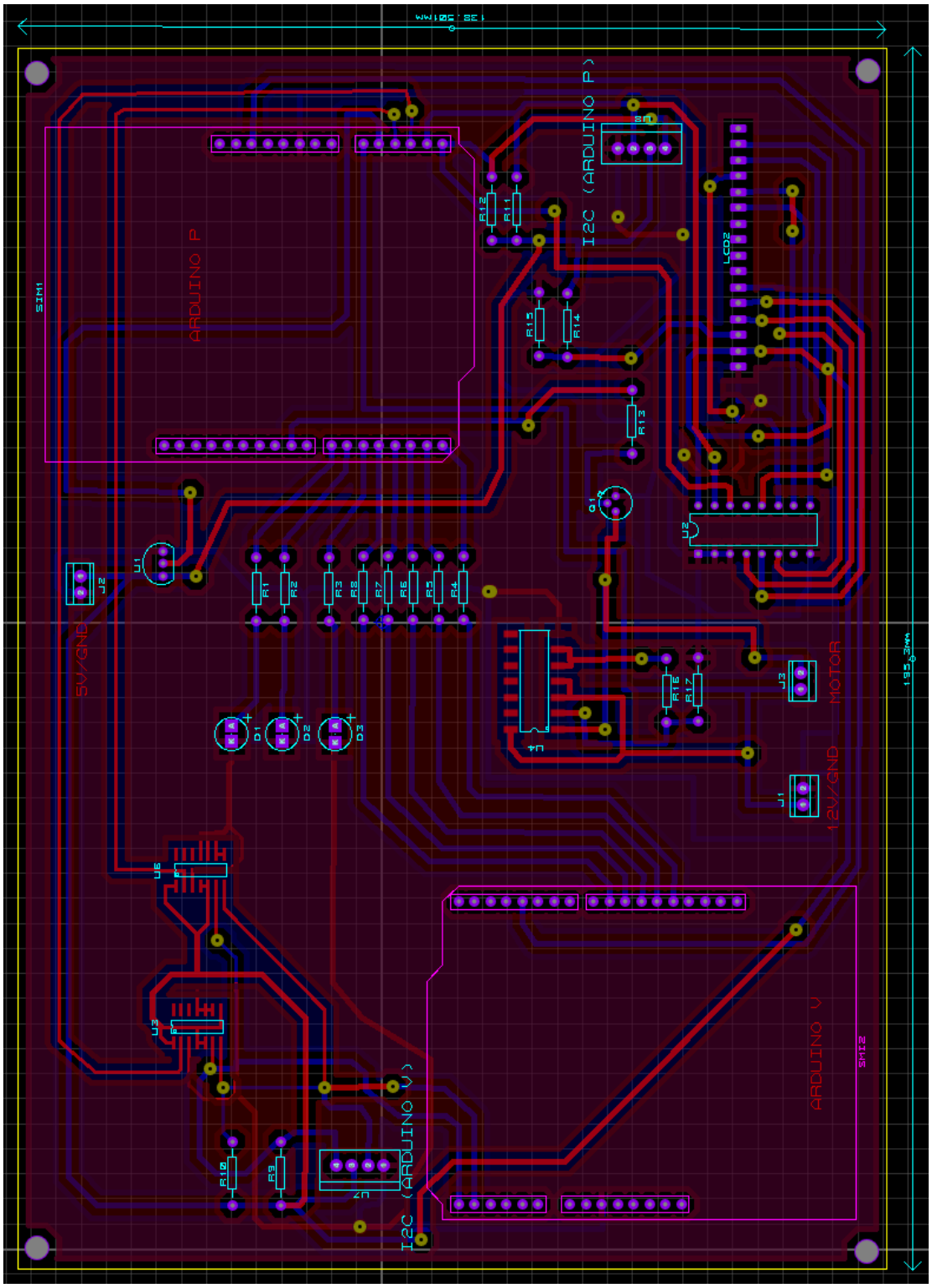
2. Colomer, J., Serra, T., Cañabate, D., & Bubnys, R. (2020). Reflective Learning in Higher Education: Active Methodologies for Transformative Practices.

3. Calvo, I., Zulueta, E., Gangoiti, U., López, J. M., Cartwright, H., & Valentine, K. (2009). *Laboratorios remotos y virtuales en enseñanzas técnicas y científicas* (Vol. 3, No. 3, pp. 1-21). Ikastorratza.

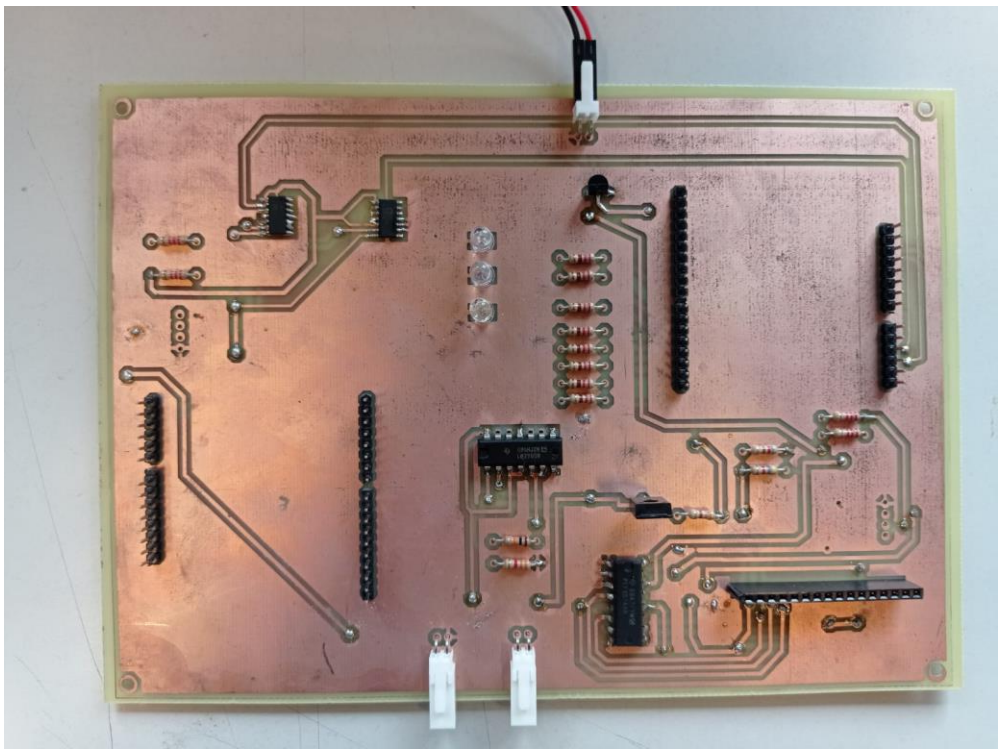
4. Musa, R. Z. (2012). Laboratorios remotos: actualidad y tendencias futuras. *Scientia et technica*, *2*(51), 113-118.

11. Anexos

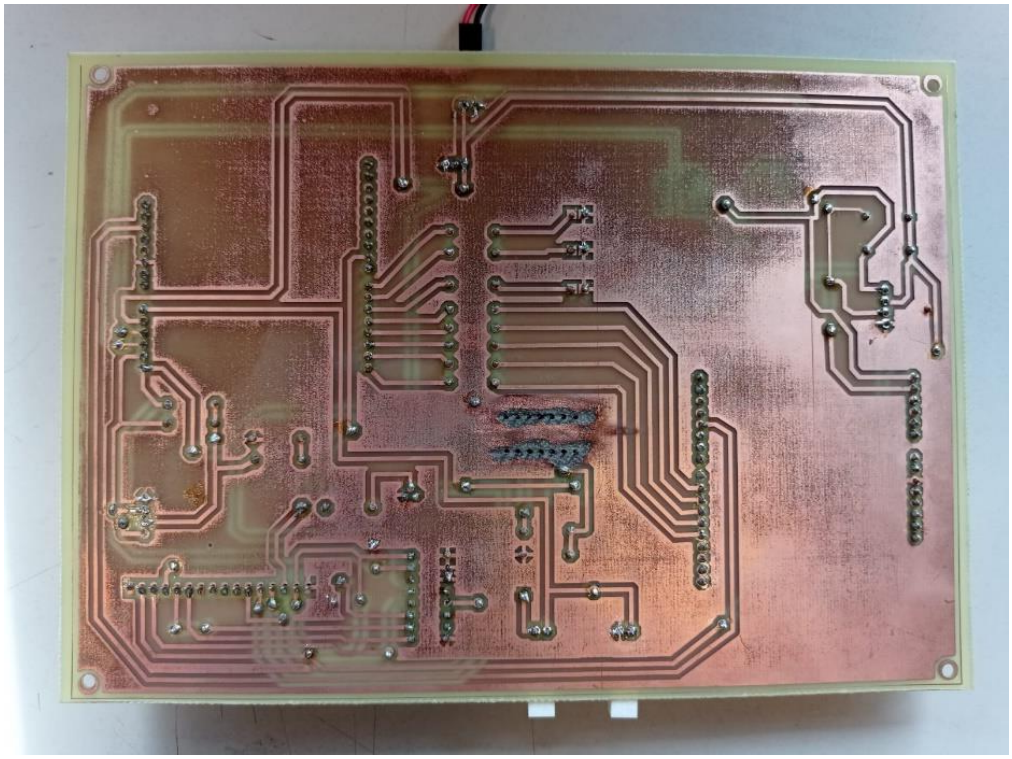
Anexo 1: Layout y enrutado de la PCB



Anexo 2: Top copper de la PCB



Anexo 3: Bot copper de la PCB



Anexo 4: Código de las funciones handle.click

4.1 Primer fragmento

```
void B1() {

 data = 1;
 id =10;
 checksum=Calculate_checksum(controller,id,data);
 arduinoV.write(controller);
 arduinoV.write(id);
 arduinoV.write(data);
 arduinoV.write(checksum);

}

void B2() {

  data = 1;
  id = 20;
  checksum=Calculate_checksum(controller,id,data);
  arduinoV.write(controller);
  arduinoV.write(id);
  arduinoV.write(data);
  arduinoV.write(checksum);

}

void B3() {

  data = 1;
  id = 30;
  checksum=Calculate_checksum(controller,id,data);
  arduinoV.write(controller);
  arduinoV.write(id);
  arduinoV.write(data);
  arduinoV.write(checksum);
}

void B4() {

  data = 1;
  id = 40;
  checksum=Calculate_checksum(controller,id,data);
  arduinoV.write(controller);
  arduinoV.write(id);
  arduinoV.write(data);
  arduinoV.write(checksum);
}
```

4.2 Segundo fragmento

```
263 void B5() {
264 
265   data = 1;
266   id = 50;
267   checksum=Calculate_checksum(controller,id,data);
268   arduinoV.write(controller);
269   arduinoV.write(id);
270   arduinoV.write(data);
271   arduinoV.write(checksum);
272 }
273 
274 void RESET() {
275   data = 1;
276   id = 100;
277   checksum=Calculate_checksum(controller,id,data);
278   arduinoV.write(controller);
279   arduinoV.write(id);
280   arduinoV.write(data);
281   arduinoV.write(checksum);
282 }
283 
284 void Potenciometro1( float Pot1) {
285 
286   Pot1 *= 100;
287   float valor = map(int(Pot1),0,500,0,255);
288   data = byte(valor);
289   id = 60;
290   checksum=Calculate_checksum(controller,id,data);
291   arduinoV.write(controller);
292   arduinoV.write(id);
293   arduinoV.write(data);
294   arduinoV.write(checksum);
295 
296 }
297 
298 void Potenciometro2( float Pot2){
299   Pot2 *= 100;
300   float valor = map(int(Pot2),0,500,0,255);
301   data = byte(valor);
302   id = 70;
303   checksum=Calculate_checksum(controller,id,data);
304   arduinoV.write(controller);
305   arduinoV.write(id);
306   arduinoV.write(data);
307   arduinoV.write(checksum);
308 }
```

Anexo 5: Código del scanner I2C

```
int Scanner_I2C(){
  Scanner_done=0;//Se inicializa la variable que devuelve si ya se
ha realizado, o no, el scanner
  byte error, address; //Se declaran variables para la dirección y
el error
  int nDevices=0; //Se inicializado el número de dispositivos
encontrados por el scanner
  Serial.println("Escanenado...");


  for (address = 1; address < 127; address++ )
  {
   Wire.beginTransmission(address);
    error = Wire.endTransmission(); //Devuelve un valor u otro en
función de si se ha reconocido un dispositivo en la dirección
actual

    if (error == 0)
    {
      Serial.print("dispositivo I2C encontrado en la dirección
0x");
      nDevices++;
      if(nDevices==1) address_pot1= address; //Asignación de la
dirección del potenciómetro digital 1
      else if(nDevices==2) address_pot2= address; //Asignación de
la dirección del potenciómetro digital 2
      else{} //Asignación de la dirección de posibles módulos
futuros

      if (address < 16)
        Serial.print("0");
      Serial.print(address, HEX);
      Serial.println("  !");

    }
    else if (error == 4)
    {
      Serial.print("Error desconocido en la direccion 0x");
      if (address < 16)
        Serial.print("0");
      Serial.println(address, HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No se encuentran dispositivos I2C\n");
  else{
    Serial.println("Finalizado\n");
     if (nDevices>1)Scanner_done=1;
  }
 delay(5000); // Cada 5s se realizara un scan
  //58(hex) dirección del potenciómetro digital
  return Scanner_done;
}
```

II. PLANOS DEL PROYECTO

Durante el presente punto se va a mostrar los planos del diseño realizado en este proyecto. Estos son los planos que componen el diseño de la placa de circuito impreso, así como el esquemático de esta. Este diseño ha sido implementado mediante el software de Proteus.

1.Circuito esquemático de la PCB

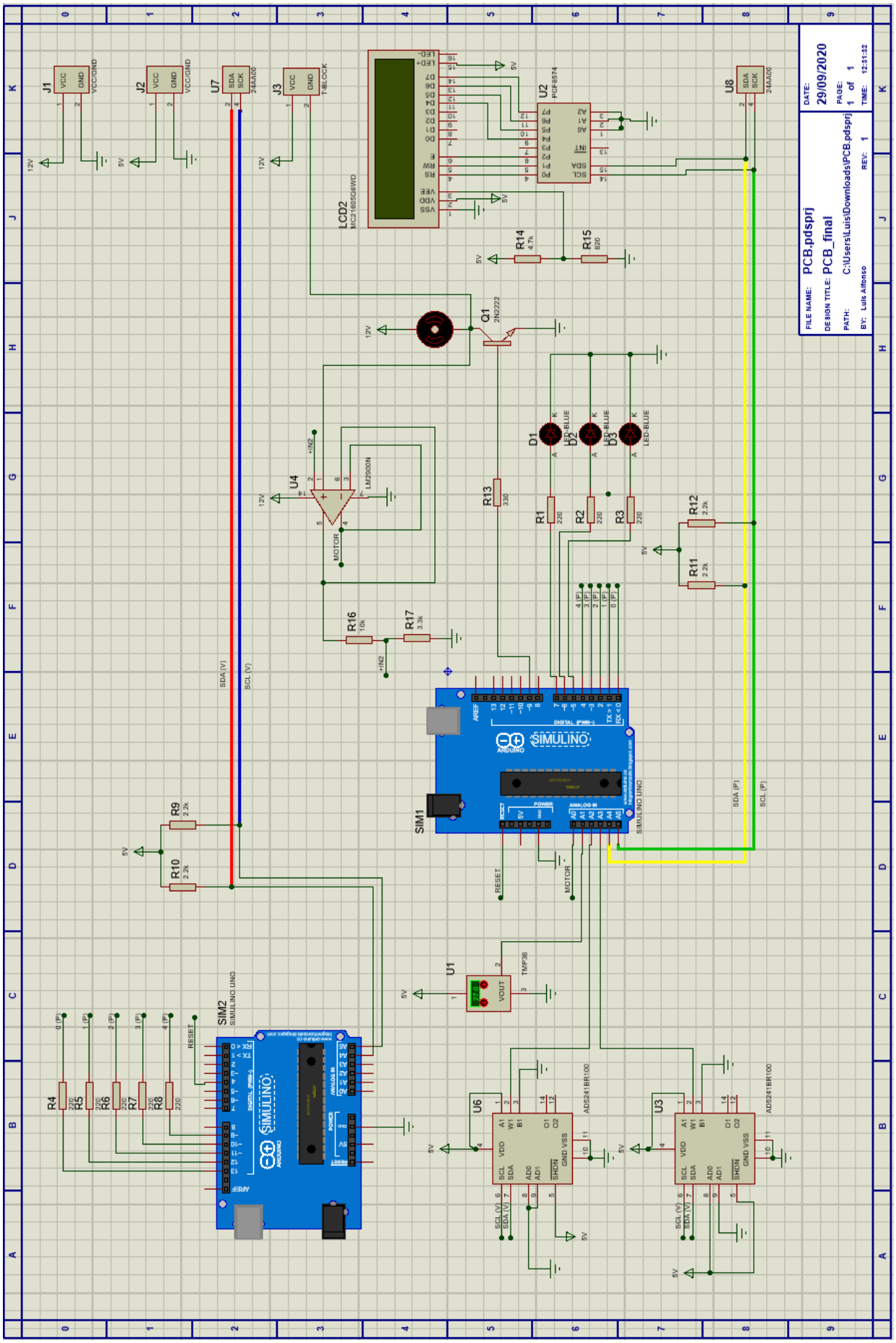
En primer lugar, se muestra el diseño del circuito esquemático de la PCB en el cual se definen los componentes y las conexiones entre estos. Se puede ver con claridad ambos Arduinos encargados del control de la placa, así como que sus buses I2C. El bus I2C del Arduino Virtual está resaltado con los colores rojo y azul, mientras que el del Arduino Programable se encuentra resaltado de amarillo y verde

2. Diseño del layout y el enrutado de la PCB

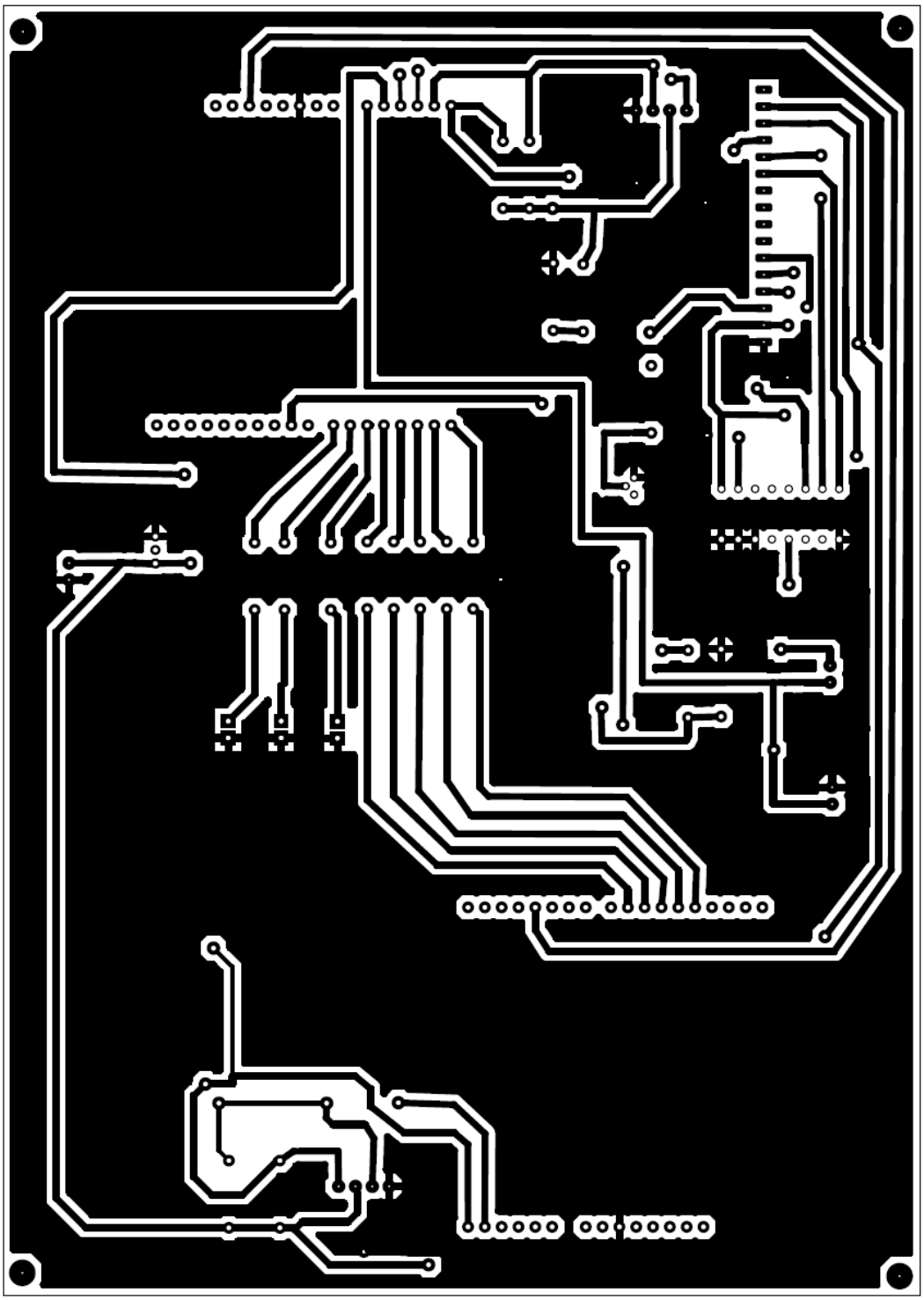
Seguidamente, se pueden ver el diseño del layout y el enrutado de la PCB en ambas superficies de Bot Copper y Top Copper respectivamente. Están mostrados en blanco y negro para distinguir con mayor claridad todos los componentes presentes en los planos. Además, en estas se puede apreciar la posición de los componentes, las pistas, las vías y los planos de masa final.

Como puede apreciarse en la posición de las vías, ambas placas están orientadas de la misma forma. Presentan unas dimensiones aproximadas de 200x140mm y se muestran en escala 1:1, es decir, a tamaño real.

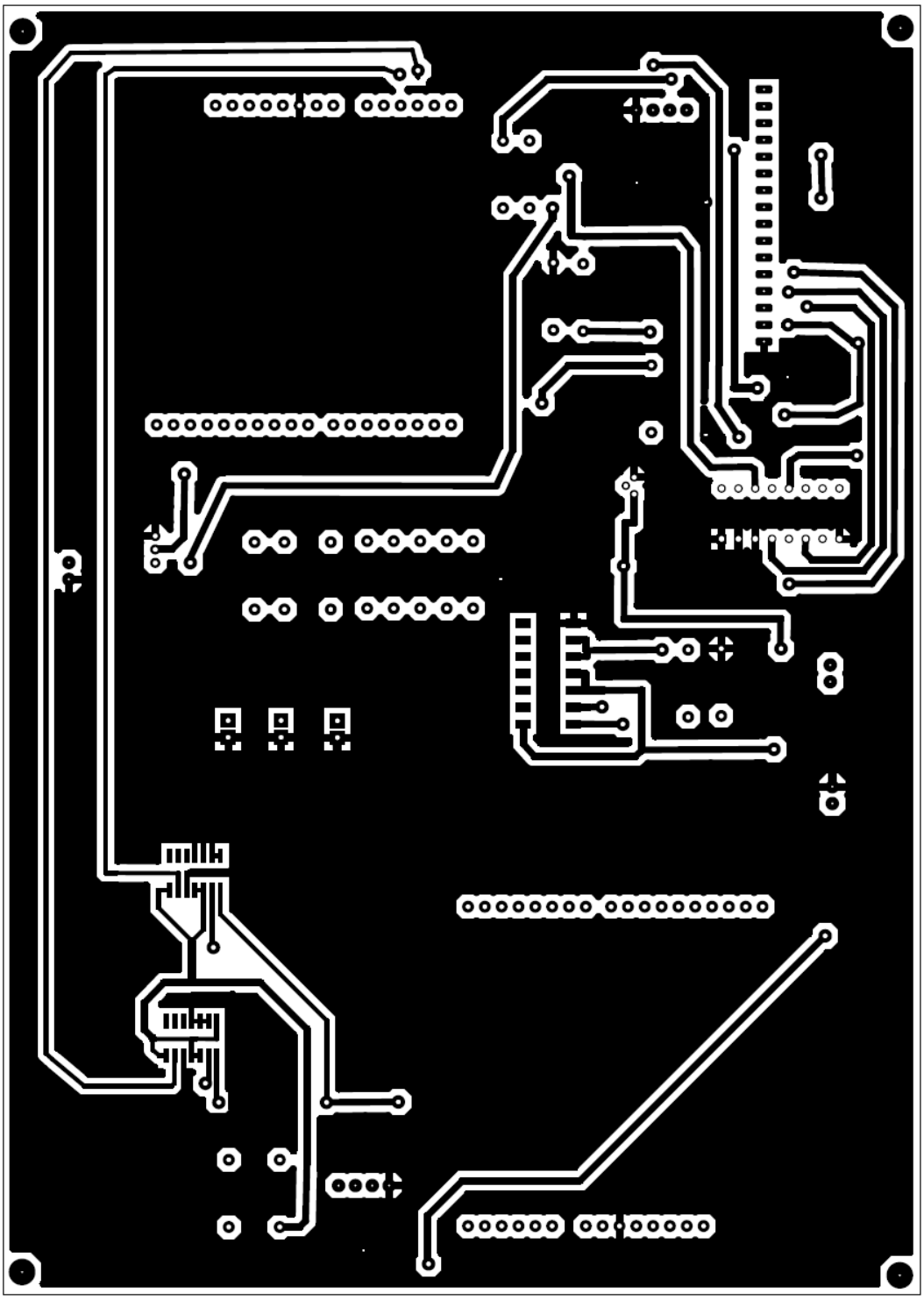
Luis Alfonso Molina



2.1 Capa Bot de la PCB



2.2 Capa Top de la PCB



III. PLIEGO DE CONDICIONES

El presente documento tiene la finalidad de presentar los niveles técnicos y de calidad exigibles, realizando las intervenciones necesarias de acuerdo con el contracto y la legislación aplicable a la parte técnica y a la parte contratante. Asimismo, se van a redactar las relaciones entre estas dos partes y las obligaciones correspondientes para el cumplimiento del contrato de desarrollo del presente proyecto.

Se considera **parte contratante**: al Departamento de Ingeniería Electrónica de la Escuela Técnica Superior de Ingeniería del Diseño de la Universidad Politécnica de Valencia, bajo la responsabilidad de En FRANCISCO JAVIER IBÁÑEZ CIVERA, como viene dado en el contrato de empresa, con la obligación verificar los documentos antes de comenzar de comenzar el desarrollo del proyecto y justo después de recibir los documentos. Además, dada cualquier tipo de discrepancia, contradicción u omisión deberá transmitirla a la parte técnica, así como facilitar o financiar todo el material necesario para el desarrollo del proyecto.

Se considera **parte técnica**: al En LUIS ALFONSO MOLINA, como viene especificado en el contrato de empresa, con las responsabilidades facultativas a la vista del proyecto y el contrato. Asimismo, presenta las responsabilidades de planificar y establecer las condiciones técnicas, realizar la selección del material correspondiente, diseñar, implementar y comprobar el funcionamiento del sistema, así como redactar el conjunto de documentos necesarios.

En dicho contrato, se integran los documentos ordenados por el valor de sus especificaciones en los casos previamente comentados de discrepancia u omisión:

1. Las condiciones fijadas en el propio documento del contrato de empresa.
2. Este pliego de condiciones
3. El resto de la documentación del proyecto. Como son la memoria, los planos y el presupuesto económico.

Ante cualquier voluntad de modificación del proyecto por parte de la parte contratante, se debe notificar la propuesta con antelación suficiente para que la parte técnica pueda realizarla. Esta modificación se realizará siempre y cuando sea una propuesta de mutuo acuerdo.

La fecha límite del proyecto se establece el día: 30 de noviembre de 2020. En esta fecha la documentación actualizada del proyecto debe ser entregada a la parte contratante, junto con el prototipo y el software desarrollado. Por parte de la parte contratante, en dicha fecha debe haber entregado la cuantía económica pactada en el contrato legal.

1. Condiciones técnicas

Las condiciones técnicas son el conjunto de puntos que el sistema que se desarrolla en el presente proyecto debe cumplir. Este sistema, compuesto por una placa de circuito impreso y una interfaz gráfica, presenta las siguientes características:

- Precio unitario inferior a 120 euros.
- Compactibilidad de la placa de circuito impreso con todos los componentes no externos correctamente soldados a la placa.
- Interfaz gráfica intuitiva y sencilla, facilitando el manejo del usuario final.
- El conjunto del sistema permite la correcta realización de las prácticas para las que el sistema está diseñado e implementado.
- Hardware con diversos módulos que faciliten su ampliación.
- Software y hardware totalmente accesibles.
- Componentes económicos y accesibles.
- Los datos modificados mediante el usuario en la interfaz deben ser convenientemente interpretados y enviados a los correspondientes pines de la placa.
- El sistema de la interfaz gráfica debe proporcionar mensajes de error que sean informativos y orientados a usuario final.
- El diseño de la placa de circuito impreso debe cumplir las restricciones necesarias para su posterior fabricación el laboratorio.

2. Materiales utilizados en el desarrollo y normativas

La normativa a cumplir en el presente proyecto es bastante básica ya que no tiene fines comerciales. Se comparte para la reproducción autónoma. La normativa a cumplir es la siguiente:

- Los elementos electrónicos deben cumplir con la normativa de emisiones electromagnéticas de *class 1 ANSI/IPC-D-275*.
- Los plásticos empleados deben cumplir la normativa de *ISO10993* y *USP class VI*

A continuación, se va a indicar el material necesario para el desarrollo del proyecto:

- Una placa de material fotosensible de 220 mm x 150 mm.
- Dos Arduino UNO.
- Dos conectores I2C de cuatro pines.
- Dos potenciómetros digitales B2541.
- Un motor DC de 12V.
- Un transistor 2N2222.
- Un amplificador operacional LM2900N.
- Una pantalla LCD: MC21605G6WD.
- Un expansor de E/S: PC8754N.
- Un sensor de temperatura TMP36.

- Tres conectores de dos pines.
- Tres LEDs.
- Resistencias.

3. Test de verificación a realizar

Test de soldadura de los componentes: Este test consiste en la comprobación de que todos los componentes se encuentren correctamente soldados en orden a lograr un dispositivo compacto. Además, se debe comprobar que todos los componentes se encuentren en la posición y orientación correspondientes.

Test de comprobación del enrutado: Para el correcto funcionamiento de la placa de circuito impreso es de vital importancia asegurarse que las pistas de esta cumplan su función de conectar entre sí dos puntos, esto se realiza mediante la comprobación de continuidad entre puntos conectados mediante un multímetro de mano, haciendo énfasis en aquellas de potencia y comunicación o con soldadura más estrecho. De la misma forma se comprobará el aislamiento entre pistas cercanas y a masa.

Test de comunicación con la interfaz gráfica: Este test consiste en, mediante la interfaz gráfica diseñada en el presente proyecto, establecer la conexión de esta a la placa comprobándose la correcta respuesta del sistema dadas las instrucciones posibles desde la interfaz por el usuario.

Test de estímulos: Desde la interfaz gráfica se enviarán las órdenes de ejecución de estímulo relacionadas con el reset del Arduino Programable, los cinco botones y los dos deslizadores, que se comprobarán desde este último.

Test de los buses I2C: Se realizará una comprobación de los buses I2C, Virtual y Programable, conectando en su extremo un módulo sencillo que deberá ser identificado por el Arduino pertinente.

Test de periféricos desde Arduino Programable: Se comprobará la temperatura, encendido / apagado de LEDs, control de motor DC y pantalla LCD programando el Arduino Programable.

IV. PRESUPUESTO

En este punto se va a describir el estudio económico del proyecto, en referencia al diseño, desarrollo y coste de los materiales para la producción.

1. Precio del desarrollo del prototipo

En lo referente al desarrollo del prototipo es conveniente dividir los gastos en dos apartados:

1.1 Honorarios del desarrollador del proyecto

Se considera las horas invertidas en el proyecto y el precio por hora correspondiente:

HONORARIOS DEL PROYECTISTA			
CONCEPTO	**€/HORA**	**HORAS**	**TOTAL**
Estudio del proyecto	20	10	200,00 €
Diseño	15	35	525,00 €
Programación	15	30	450,00 €
Montaje	10	25	250,00 €
		TOTAL	1.425,00 €

2. Amortización de las licencias de los entornos de desarrollo del proyecto y amortización del material empleado en el laboratorio.

Para el desarrollo del presente proyecto se han utilizado diversas herramientas cuya amortización se va a mostrar a continuación:

AMORTIZACIÓN	
CONCEPTO	**IMPORTE**
Software Processing	0,00 €
Firmware Arduino	0,00 €
Software Proteus	200,00 €
Insoladora	50,00 €
Taladradora CNC	50,00 €
Soldadora	50,00 €
TOTAL	350,00 €

La licencia de Proteus que se va a usar es la más básica ya que para el desarrollo de este proyecto es suficiente.

3. Material

Para calcular el precio del material de los componentes empleados en el montaje de la PCB, primero vamos a calcular el precio de lo que valdría fabricar una sola PCB. Posteriormente calcularemos el precio de los componentes si fabricáramos 200 unidades. Los componentes han sido obtenidos del catálogo oficial *RS Components*.

Precio de la PCB si solo fabricáramos una unidad:

MATERIAL PROTOTIPO			
CONCEPTO	**UNIDADES**	**€/UNIDAD**	**IMPORTE**
Arduinos	2	18,87 €	37,74 €
Resistencias Pull-Up (2.2K)	4	0,07 €	0,27 €
Potenciometros digitales	2	2,34 €	4,68 €
Motor DC 12 V	1	14,10 €	14,10 €
Transistor 2N2222	1	0,16 €	0,16 €
Amplificador LM2900N	1	0,34 €	0,34 €
Resistencia 3.3K	1	0,14 €	0,14 €
Resistencia 10K	1	0,13 €	0,13 €
Pantalla LCD MC21605G6WD 16x2	1	10,54 €	10,54 €
Adaptador I2C PCF8574	1	1,21 €	1,21 €
Resistencia 820	1	0,04 €	0,04 €
Resistencia 4.7K	1	0,13 €	0,13 €
Sensor temperatura TMP36	1	1,28 €	1,28 €
Resistencias botones (220)	5	0,12 €	0,62 €
LEDs	3	0,42 €	1,25 €
Resistencias LEDs (220)	3	0,12 €	0,37 €
Placa de cobre PCB	1	9,87 €	9,87 €
		TOTAL	82,86 €

Precio de una PCB si fabricáramos 200 unidades:

MATERIAL PROTOTIPO			
CONCEPTO	**UNIDADES**	**€/UNIDAD**	**IMPORTE**
Arduinos	2	17,94	35,88 €
Resistencias Pull-Up (2.2K)	4	0,023	0,09 €
Potenciometros digitales	2	1,594	3,19 €
Motor DC 12 V	1	13,24	13,24 €
Transistor 2N2222	1	0,159	0,16 €
Amplificador LM2900N	1	0,316	0,32 €
Resistencia 3.3K	1	0,138	0,14 €
Resistencia 10K	1	0,126	0,13 €
Pantalla LCD MC21605G6WD 16x2	1	6,55	6,55 €
Adaptador I2C PCF8574	1	0,84	0,84 €
Resistencia 820	1	0,035	0,04 €
Resistencia 4.7K	1	0,126	0,13 €
Sensor temperatura TMP36	1	1,026	1,03 €
Resistencias botones (220)	5	0,072	0,36 €
LEDs	3	0,418	1,25 €
Resistencias LEDs (220)	3	0,072	0,22 €
Placa de cobre PCB	1	7,46 €	7,46 €
		TOTAL	71,01 €

4. Precio final

Para calcular el precio de venta por unidad dividimos los honorarios del proyectista, la amortización entre el número de unidades (200) y el precio de los materiales de la PCB. Posteriormente agregamos el beneficio industrial y el IVA, en el caso de que este servicio de producción se externalizase o se fabricase para terceros.

PRECIO DE VENTA	
CONCEPTO	**IMPORTE**
Honorarios del proyectista	7,13 €
Amortización	1,75 €
Material	71,01 €
Beneficio Industrial (17%)	12,07 €
IVA (21%)	19,31 €
TOTAL	111,27 €

Una vez incorporado el beneficio industrial y los impuestos correspondientes el precio final por unidad asciende a una cantidad de 111,27 euros.