UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

DESARROLLO DE UN SISTEMA DE CONTROL DE CONFORT PARA CUNAS DE BEBÉS

AUTORA: Mónica Lerma Sánchez

TUTOR: Xavier Blasco Ferragud

Curso Académico: 2018-19

AGRADECIMIENTOS

A mi madre y Rosa, por apoyar mis decisiones y motivarme hacia nuevos objetivos.
A Sara y Toni, por ayudarme cada día de verano con todas las dudas sobre el *software* y dedicarme su tiempo y cariño.
A Carlos por compartir todo este tiempo de esfuerzo a mi lado, en los momentos buenos y malos.
A mi tutor por resolver mis dudas y creer en el desarrollo de este proyecto desde el principio.
A todas aquellas personas que he conocido a lo largo de mis estudios y me han ayudado de manera directa o indirecta a llevar a cabo este proyecto adelante.
Y, sobre todo, a mi padre, que con su entusiasmo, dedicación y paciencia ha aportado siempre todo lo que ha estado en su mano tanto en este proyecto, como en mi formación en general. Estés donde estés.

RESUMEN

Este proyecto contempla la implementación de un sistema de automatización instalado en una cuna para bebé con el objetivo de facilitar a los padres las atenciones sobre éste, ayudando a crear un clima de confort para el niño.

Dicho sistema estará dotado de varios sensores que proporcionarán información sobre las condiciones exteriores, como son el sensor de temperatura y humedad relativa, sensor de humedad absoluta y sensor de calidad de aire.

El control de este sistema se realizará mediante una Raspberry Pi 3 B, la cual procesará los datos emitidos por los diferentes sensores generando, a modo de respuesta, la activación de diferentes actuadores, como son la manta eléctrica y el humidificador, o bien generando un aviso informativo a los padres sobre la necesidad de cambiar de pañal al bebé.
También cuenta con otros actuadores instalados en la cuna como el movimiento balancín o un proyector de techo, que pueden ser activados a demanda del usuario mediante la pantalla táctil situada en la cuna, que actúa como interfaz gráfica, o mediante la aplicación móvil, con la que controlar el sistema a distancia.

En primer lugar, para la elaboración de este sistema, se ha analizado cada uno de los sensores utilizados, para entender sus principios de funcionamiento.
Para el control de todo el sistema se ha utilizado una Raspberry PI, donde se ejecuta la aplicación programada en lenguaje Java y desarrollada en el IDE IntelliJ.

Palabras Clave: Cuna, Raspberry PI, sensor de temperatura, sensor humedad relativa, sensor humedad absoluta, DHT11.

RESUM

Aquest projecte contempla l'implementació d'un sistema d'automatització instal·lat en un bressol amb l'objectiu de facilitar als pares les atencions sobre el xiquet, ajudant a crear un clima de confort per aquest.

Aquest sistema estarà dotat de diversos sensors que proporcionaran l'informació sobre les condicions exteriors, com són, el sensor de temperatura i humetat relativa, sensor d'humetat absoluta i sensor de qualitat d'aire.

El control d'aquest sistema es realitzarà mitjançant una Raspberry Pi 3 B, la qual processarà les dades emeses pels diferents sensors generant, com resposta, l'activació de diferents actuadors, com són, la manta elèctrica i el humidificador, o bé generant un avís informatiu als pares sobre la necessitat de canviar de bolquer al xiquet.
També compta amb altres actuadors instal·lats en el bressol com el moviment balancí o un projector, que poden ser activats a demanda de l'usuari mitjançant la pantalla tàctil situada al bressol, que actua com a interfaz gràfica o mitjançant l'aplicació mòbil, amb la qual controlar el sistema a distància.

En primer lloc, per a l'elaboració d'aquest sistema, s'ha analitzat cadascun dels sensors utilitzats, per entendre els seus principis de funcionament.
Per al control de tot el sistema s'ha utilitzat una Raspberry Pi, on s'executa l'aplicació programada en llenguatge Java i desenvolupada en l'IDE IntelliJ.

Paraules Clau: Bressol, Raspberry PI, sensor de temperatura, sensor humitat relativa, sensor humitat absoluta, DHT11.

ABSTRACT

The aim of this project is to implement an automation system installed in a baby crib to provide the parents with a system to monitor and care the state of the baby, helping to create a comfortable ambient to the baby.

This system will be equipped with several sensors that will provide information on the external conditions, such as the temperature and relative humidity sensor, absolute humidity sensor and air quality sensor.

The control of this system will be carried out by means of a Raspberry Pi 3 B, which will process the data emitted by the different sensors generating, as a response, the activation of different actuators, such as the electric blanket and the humidifier, or generating an informative notice to parents about the need to change the baby's diaper.
It also has other actuators installed in the crib such as the rocker movement or a ceiling projector, which can be activated at the user's request through the touch screen located in the crib, which acts as a graphic interface or through the mobile application, with which Remote control system.

First of all, for the elaboration of this system, each of the sensors used has been analyzed, to understand its operating principles.
A Raspberry PI has been used to control the entire system, where the application programmed in Java language and developed in the IntelliJ IDE is executed.

Keywords: Crib, Raspberry Pi, temperature sensor, relative humidity sensor, absolute humidity sensor, DHT11.

ÍNDICE DE LA MEMORIA

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

MEMORIA TRABAJO FIN DE MÁSTER

1. Introducción

Del cuidado de un bebé en su primer año de vida dependerán muchos aspectos importantes de cara al futuro, tanto de su personalidad como de sus cualidades físicas, así como sus defensas frente alguna enfermedad. Hay que estar muy pendientes de todo en los primeros doce decisivos meses.

El trastorno de rutina que supone traer al mundo a una nueva criatura afecta directamente al estado anímico de los padres, por ello la motivación de este proyecto es facilitar, a los padres o tutores, el cuidado del bebé siempre que éste se encuentre en su lugar de descanso. El sistema que se va a desarrollar debe conseguir unas condiciones idóneas para el bebé y permitir monitorizar y actuar de forma remota.

Todas las dificultades que se puedan encontrar entorno al cuidado del niño se facilitan mediante la automatización de una cuna, es decir, dotar de inteligencia a una cuna para que sea capaz de ayudar con el cuidado del bebé. El producto que se ha desarrollado posee el nombre de *IntelliCrib*, lo que en inglés se traduce como cuna inteligente.
Este producto pretende aunar en una sola estructura la detección de las condiciones en el entorno del bebé, así como la actuación sobre éstas y el control constante del estado del niño.

1.1 Objetivos

El objetivo de este proyecto es dotar a una cuna de bebé de un sistema de automatización que genere, en la habitación donde se encuentra, el clima de confort necesario para el niño, así como informar a los padres sobre el estado en que se encuentra su hijo.
Los objetivos que se pretenden con este trabajo son los siguientes:

- Crear una plataforma de ayuda a los padres para el cuidado del bebé durante todas las horas que éste pasa en la cuna, informando vía móvil sobre las necesidades del niño en cada momento.
- Creación de un ambiente de confort para el bebé de forma automática, mediante el control de los diferentes actuadores a partir de las señales proporcionadas por los sensores.
- Adaptar el medio en que se encuentra el niño a las condiciones que deseen los padres en cada momento, las cuales pueden modificar desde cualquier dispositivo o con la pantalla táctil situada en la cuna.
- Diseñar un prototipo de cuna automatizada que suponga el inicio del desarrollo un producto competente en el mercado y adaptable a todo tipo de hogares.

1.2 Estructura de la memoria

Esta memoria consta de seis capítulos que detallan el desarrollo del proyecto de principio a fin; comenzando por los requisitos que se pretenden conseguir, siguiendo con el detalle de cada pieza *hardware* y *software* que conforman la construcción del primer prototipo, así como un estudio de viabilidad del producto que pretende comercializarse.

El primer capítulo ofrece una presentación de objetivos del proyecto, así como la motivación que ha llevado a realizarlo y un análisis de la información que contiene cada capítulo de esta memoria.

El segundo capítulo comienza analizando el problema social que se pretende resolver con este producto y, con ello, marca unos requisitos técnicos que deben cumplirse para la correcta integración de este producto en la sociedad y para lograr el funcionamiento esperado de éste una vez se encuentre en el mercado, ofreciendo a los usuarios una respuesta técnica a su problemática.

Este capítulo sigue con un análisis de las posibles alternativas que se encuentran en el mercado para las diferentes áreas que conforman el prototipo desarrollado en el proyecto; de un lado se analizan los recursos de campo de los que se puede valer para llevar a cabo el funcionamiento de IntelliCrib y, para dotar de control a estos recursos, se analizan los recursos de control que ofrece el mercado.

En el capítulo tres se realiza una comparativa de los recursos *hardware* mostrados en el segundo capítulo y a partir de ésta, se escoge un recurso para cada necesidad del proyecto explicando porqué se ha escogido éste y no otro, haciendo distinción entre las tecnologías escogidas para la parte de elementos de campo, donde se añade una introducción de cómo se implementará cada uno de los sensores y actuadores en el proyecto, y para la parte de control.

El capítulo cuatro se centra en la exposición las tecnologías empleadas en el área *software* del proyecto, como son el sistema operativo, la herramienta de desarrollo, el lenguaje de programación y las librerías utilizadas en la placa de desarrollo.

El capítulo cinco expone el desarrollo de la solución; una vez expuestas todas las tecnologías *hardware* y *software* a emplear, se procede a exponer cómo se ha hecho uso de ellas. En la parte de *software* se expone la metodología empleada para el desarrollo del código de programación, así como la exposición de éste y culmina con los grafcets de flujo de trabajo para los diferentes actuadores.
En la parte *hardware* se expone el conexionado por áreas (sensores, actuadores, periféricos y control), es decir, de todos los elementos del proyecto, teniendo en cuenta las alimentaciones necesarias para cada uno de ellos.

El sexto capítulo expone la implementación del proyecto, es decir, los métodos y pasos que se han llevado a cabo, a partir del desarrollo, para conformar un prototipo que aúne todas las soluciones de cada área dando funcionalidad a IntelliCrib.

En el séptimo capítulo se realiza un estudio de viabilidad de mercado para valorar qué éxito tendría este producto en el mercado, si será un proyecto rentable, cuál sería su precio de lanzamiento y las líneas futuras para conformar un producto de mercado a partir del prototipo desarrollado en este proyecto.

El capítulo ocho ofrece las conclusiones del proyecto, donde también se expone la relación de los conocimientos desarrollados en el proyecto con los tomados durante el máster.

2. Análisis del problema

Al nacer un bebé las horas de descanso de los padres pasan a reducirse notablemente. Los bebés se despiertan a todas horas; desde que nacen hasta los 4-6 meses, el sueño de los bebés se da en dos fases, el hecho de alternar sólo dos fases hace que se despierten a menudo. A partir de estos 6 meses, cambian las fases de sueño a cinco, ya que el sueño de los niños es evolutivo, y cambia conforme lo hace su cerebro. Sin embargo, se estima que hasta los 2-3 años no empiezan a dejar atrás los despertares nocturnos [1].

Por otro lado, se estima un promedio de necesidad de cambio el pañal del bebé unas diez o doce veces al día; los recién nacidos llegan a miccionar unas veinte veces al día y los menores de un año, unas siete veces. La humedad provocada por la orina o las heces puede originar irritaciones, dermatitis, incluso un desequilibrio del pH de la piel debido a la bacteria “bacillus ammoniagenes”, presente en la orina, y formación de amoniaco por las enzimas fecales [2]. Por ello debe evitarse que el bebé permanezca demasiado tiempo húmedo, de lo que surge la preocupación de los padres sobre cuándo revisar el pañal del bebé.

En cuanto al ambiente de confort, la estancia donde se encuentra el bebé debe tener una temperatura y humedad adecuada consiguiendo que no pase ni frío ni calor, ya que si el bebé suda pueden causar problemas en la piel o resfriados. Según la Pediatra adjunta de Urgencias en el Hospital Sant Joan de Déu, “*La temperatura adecuada de la habitación de un recién nacido sano es de 22 - 24°C para el día. Durante la noche, puede descender hasta los 18 - 20°C, teniendo en cuenta el abrigo de la ropa de cama o del saco de dormir*” [3]. También señala que la temperatura es un factor fundamental para que el bebé consiga descansar bien.

Otro factor importante son las condiciones ambientales de la habitación, por lo que se recomienda un entorno suficientemente húmedo, ya que la falta de humedad dificulta la respiración del bebé al secarse la mucosidad que éste contiene en sus vías respiratorias. Según la psicóloga y asesora de lactancia, Tania Olivares, “*Los valores de humedad relativa considerados óptimos para nuestra salud y la de los bebés oscilan entre el 40% y el 60%”* [4].

Todas estas necesidades o comodidades para los bebés se convierten en preocupaciones para los padres, traducidas en falta de sueño, discusiones y ansiedad, sumado a que éstos normalmente deben seguir su rutina de trabajo independientemente de este bebé.
Por todo ello, se quiere ofrecer una ayuda a estos padres deseosos de un poco de sosiego con la cuna IntelliCrib, que facilitará todas las tareas relacionadas con el cuidado del bebé manteniendo

en aviso constante a los padres de las necesidades y entorno de su bebé mientras se encuentre en la cuna.

2.1 Especificación de requisitos

Partiendo de las necesidades de los bebés, mencionadas anteriormente, ya sean fisiológicas, afectivas, de conciliación de sueño o de entretenimiento, se han reunido las siguientes especificaciones que debe cumplir la cuna IntelliCrib para ofrecer el servicio esperado.

- Información continua del ambiente en el que se encuentra el bebé (temperatura y humedad relativa), mediante la aplicación.
- Información continua sobre el estado del pañal del bebé, mediante la aplicación.
- El usuario debe poder observar al bebé en directo siempre que lo desee y durante un tiempo ilimitado desde la aplicación.
- El usuario debe poder configurar los puntos de consigna de temperatura y humedad relativa de la habitación, desde la aplicación.
- Los actuadores y periféricos deben estar disponibles para su activación en todo momento ya sea desde la aplicación o la pantalla táctil de control.
- Los sensores deben estar en funcionamiento y la aplicación captando la información de éstos en todo momento.
- Alimentación del prototipo a 230V.

2.2 Alternativas posibles

En este apartado se van a barajar las alternativas disponibles en el mercado para algunos elementos *hardware* que dan funcionalidad a la cuna, como son la placa de control o los sensores de campo, basándonos en las necesidades y comodidades de los bebés, así como en la óptima comunicación entre los materiales de campo y el sistema de control.

2.2.1 Recursos de control

2.2.1.1 Placa de desarrollo

La placa de desarrollo supone el celebro de este proyecto, es la encargada de gestionar los datos que provienen de los sensores, procesarlos y, a partir de éstos, ofrecer una respuesta, ya sea informativa a través de la interfaz o con la activación de los diferentes actuadores.
Para el control de este proyecto se han encontrado en el mercado diferentes alternativas, que se van a exponer a continuación.

➢ Arduino

Arduino es una placa de desarrollo de *hardware* para construir dispositivos digitales e interactivos que puedan sensar y controlar objetos del entorno, facilitando el uso de la electrónica y programación de sistemas embebidos.

Figura 1. Placa de control Arduino. Obtenida de la página web Naylamps Mechatronics

Los productos Arduino son *hardware* y *software* libre, bajo la Licencia Pública General Reducida de GNU o la Licencia Pública General de GNU. [5]
El diseño de la placa de desarrollo Arduino está basada en un microprocesador, donde se vuelca el código de programación y una serie de pines, pudiendo agregar circuitería, sensores y módulos de comunicación externos a la placa original.
El *software* de Arduino consiste en dos elementos: un entorno de desarrollo, basado en el entorno de *processing* y en la estructura del lenguaje de programación *Wiring*, y en el cargador de arranque, *bootloader,* que es ejecutado de forma automática dentro del microcontrolador en cuanto este se enciende.

CARACTERÍSTICAS

- Energizada por un puerto USB o un puerto barrel Jack.
- Programadas a través del puerto Serial.
- Fáciles de reprogramar.
- Comunidad de usuarios amplia.
- Cabecera ICSP.
- Botón de reseteado.
- Posibilidad de salidas PWM.

ESPECIFICACIONES TÉCNICAS

- Microcontrolador: ATmega328.
- Pines digitales: 14 (de los cuales 6 son salida PWM).
- Pines analógicos: 6.
- Corriente por pin IO: 40 mA.
- Corriente por pin 3.3V: 50 mA.
- Flash Memory: 32 KB.
- SRAM: 2 KB.
- EEPROM: 1 KB.
- Frecuencia de reloj: 16 MHz.

PRECIO: 26,17 €

➢ Raspberry Pi

Raspberry PI es una minicomputadora que surge en el entorno educativo, en este caso con la finalidad de estimular la enseñanza de la informática.
Es una computadora completamente funcional con su propio procesador y memoria y puede ejecutar un sistema operativo. Emplea mayoritariamente sistemas operativos open source como GNU/Linux, pero puede instalar otros sistemas operativos que incluyan Android, Windows 10 o Firefox OS [6].

Figura 2. Placa de control RaspberryPi. Obtenida de la página web de Raspberry Pi

Su fuerte radica en que puede realizar múltiples tareas con varios programas con su SoC Broadcom BCM2837, así como, también se puede conectar a dispositivos Bluetooth e Internet de forma inmediata mediante Ethernet o conectándose a Wi-Fi.
Existen módulos extra para dotar a esta placa de desarrollo de mayor funcionalidad, mediante estos módulos es posible doblar el número de pines GPIO de la Raspberry, agregar una cámara y añadir la tecnología NFC mediante una placa de expansión.

La gran velocidad que ofrece Raspberry PI le da la capacidad de completar las tareas diarias que realizan las computadoras: reproducir videos, navegar por la web, escuchar música, etc.

CARACTERÍSTICAS

- Energizada por un puerto microUSB o GPIO Header.
- Puerto HDMI.
- Puerto para tarjetas SD.
- Puerto de audio por Jack de 3.5 mm.
- 4 puertos USB 2.0.
- Wi-Fi + Bluetooth: 2.4GHz y 5GHz.
- Puerto CSI de cámara.
- Puerto de pantalla DSI.
- Procesador gráfico (GPU) VideoCore IV.
- Conector Gigabit Ethernet.

ESPECIFICACIONES TÉCNICAS

- Microcontrolador: Broadcom BCM2837.
- Pines digitales: 28 terminales de entrada/salida.
- RAM: 1 GB.
- Frecuencia de reloj: 1.4 GHz.

PRECIO: 38 €

➢ Tessel

Tessel 2 es una placa de desarrollo con capacidades Wi-Fi integradas. Cada placa cuenta con herramientas completas de línea de comandos que facilitan la implementación de su código, el establecimiento de credenciales de WiFi y la fácil gestión de las necesidades de autenticación.
Con la placa de desarrollo Tessel 2 es posible interactuar con el mundo físico de detección, como son los sensores, así como la actuación sobre otros dispositivos, como son los actuadores.
La plataforma se puede programar en Javascript con la capacidad de soportar otros idiomas.

Figura 3. Placa de control Tessel. Obtenida de la página web de Core Electronics

Tessel 2 está diseñado para el camino más rápido posible hacia la producción. Los módulos Plug and Play y las API de alto nivel lo distinguen de la mayoría de las otras placas de desarrollo.

CARACTERÍSTICAS

- Energizada por medio de batería o micro USB.
- 2 puertos USB para periféricos de la cámara y de almacenamiento *flash.*
- Wifi 802.11bgn.
- Puerto Ethernet 10/100.

ESPECIFICACIONES TÉCNICAS

- Microcontrolador: ARM Cortex-M3 LPC1830.
- Dos puertos de módulo Tessel DE 10 pines de entrada y salida.
- RAM: 64MB.
- FLASH: 32MB Flash.
- Frecuencia de reloj: 180Mhz.

PRECIO: 32 €

2.2.2 Recursos de campo

Los sensores son dispositivos encargados de detectar las variaciones de distintas magnitudes, en este caso la temperatura, la humedad o la calidad del aire. Al variar estas magnitudes, en el ambiente donde se encuentra, también varía con cierta intensidad la propiedad del sensor, generando, a modo de respuesta, una señal recibida e interpretada por el sistema de control.

2.2.2.1 Sensor de calidad de aire

Las heces humanas están compuestas de fibra, celulosa de la comida ingestada, grasas, sal, células rojas muertas, agua y bacterias, entre las cuales encontramos las bacterias metanogénicas, que son capaces de producir metano a partir de sustratos orgánicos en el intestino.
Este gas metano es liberado al expulsar las heces del cuerpo, y es el que se va a utilizar para la detección de nuevas deposiciones en el pañal del bebé, mediante un detector de distintos gases llamado sensor de calidad de aire que detecta presencia de CH_4, entre otros gases contaminantes. Entre los sensores de calidad de aire que detecten la presencia de metano se han encontrado los siguientes en el mercado.

- MQ2

Los sensores de la serie MQ [7] son sensores electroquímicos, con lo que varían su resistencia cuando se exponen a determinados gases. Internamente poseen un calentador encargado de aumentar la temperatura interna y, así, el sensor pueda reaccionar con los gases provocando un cambio en el valor de la resistencia.

Figura 4. Sensor calidad de aire MQ2. Obtenida de la página web CDMX

Dependiendo de la clase de sensor MQ se tendrá más sensibilidad a ciertos gases, de esta serie los sensores que detectan gas metano son el MQ2, MQ4 y MQ9.
El material sensible del sensor de gas MQ2 [8] es SnO2, que tiene una menor conductividad en el aire limpio. Cuando existe el gas combustible objetivo, la conductividad del sensor es más alta junto al aumento de concentración de gas, por ello la resistencia del sensor cambia de acuerdo a la concentración del gas en el aire.

Los sensores MQ se encuentran en el mercado en módulos, los cuales disponen de una salida digital, ya que trabaja internamente con un comparador y, con la ayuda de un potenciómetro, es posible calibrar el umbral y así poder interpretar la salida digital como presencia o ausencia del gas.
La salida analógica del módulo proviene de un divisor de tensión que forma el sensor y una resistencia de carga, *Figura 5*.

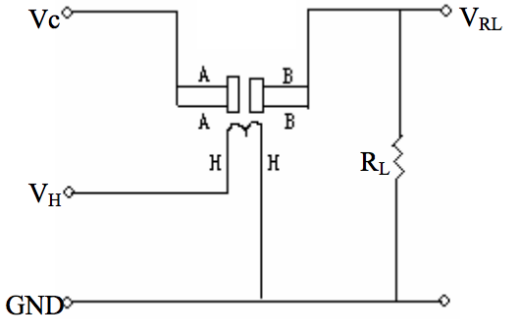


Figura 5. Circuito interno sensor calidad de aire. Obtenida del datasheet del MQ2

El MQ2 es sensible a GLP (Gas Licuado de Petróleo), i-butano, propano, metano, alcohol, hidrogeno y humo.

CARACTERÍSTICAS

- Buena sensibilidad al gas combustible en amplio rango.
- Larga vida y bajo costo.
- Circuito de accionamiento simple.
- Detector de gas portátil.
- Detector de fugas de gas doméstico.
- Detector de gas combustible industrial.

ESPECIFICACIONES TÉCNICAS

- Voltaje de entrada: 5 VDC.
- Rango de detección: 300 a 10000 ppm.
- Gas característico: 1000ppm, Isobutano.
- Tiempo de Respuesta: ≤ 10s.
- Tiempo de recuperación: ≤ 30s.
- Temperatura de trabajo: 20 °C ±2 °C.
- Humedad: 65% ±5% RH.
- Contenido de oxígeno ambiental: 21%.
- Consume menos de 150mA a 5V.

PRECIO: 2,89 €

- MQ4

Este sensor mantiene la estructura del MQ2; está compuesto por un tubo de cerámica micro AL2O3, una capa sensible al dióxido de estaño (SnO2), un electrodo de medición y un calentador, que se fijan en una corteza de plástico. El calentador proporciona las condiciones de trabajo necesarias para el rendimiento de componentes sensibles. El MQ4 [9] tiene 6 pines, 4 de ellos se usan para buscar señales, y otros 2 se usan para proporcionar corriente de calefacción, al igual que el MQ2.

Figura 6. Sensor calidad de aire MQ4. Obtenida de la página IndiaMart

El MQ4 se utilizan en equipos de detección de fugas de gas en hogares y en la industria ya que son adecuados para la detección de CH4, GNL, gas natural, humo de cocción y de cigarrillos.

CARACTERÍSTICAS

- Alta sensibilidad al CH4, gas natural.
- Pequeña sensibilidad al alcohol, al humo.
- Respuesta rápida.
- Estable y larga vida.
- Circuito de accionamiento simple.

ESPECIFICACIONES TÉCNICAS

- Voltaje de entrada: 5VDC.
- Detección de Metano: 200 a 10000 ppm.
- Detección de gas natural: 200 a 10000 ppm.
- Gas característico: 5000ppm, Metano.
- Temperatura de trabajo: 20 °C ±2 °C.
- Humedad: 65% ±5% RH.
- Contenido de oxigeno ambiental: 21%.

PRECIO: 3,59 €

- MQ9

Figura 7. Sensor calidad de aire MQ9. Obtenida de la página web CDMX

El sensor MQ9 [10] es el tercer sensor de la serie MQ capaz de detectar metano. Su funcionamiento es semejante e los sensores expuestos anteriormente; el material sensible del sensor de gas MQ9 es también el SnO2, que tiene una menor conductividad en el aire limpio. En cambio, este sensor detecta por el método del ciclo de temperatura alta y baja y detecta CO cuando baja la temperatura (calentado por 1.5V). La conductividad del sensor es más alta junto con la concentración de gas en aumento. Cuando la temperatura es alta (calentada a 5.0 V), detecta gas combustible de metano, propano, etc. y limpia los otros gases adsorbidos a baja temperatura.

El sensor de gas MQ-9 tiene una alta sensibilidad al monóxido de carbono, metano y GLP.

CARACTERÍSTICAS

- Alta sensibilidad al metano, propano y CO.
- Pequeña sensibilidad al alcohol, al humo.
- Estable y larga vida.
- Circuito de accionamiento simple.

ESPECIFICACIONES TÉCNICAS

- Voltaje de entrada: 5 VDC.
- Detección de CO: 100 a 10000 ppm.
- Detección de gas combustible: 100 a 10000 ppm.
- Temperatura de trabajo: 20 °C ±2 °C.
- Humedad: 65% ±5% RH.
- Contenido de oxígeno ambiental: 21%.

PRECIO: 7,29 €

2.2.2.2 *Sensor de humedad absoluta*

Dichos sensores también se conocen como higrómetros o sondas de humedad de suelo, debido a que su principal utilidad en el mercado consiste en la medida del contenido volumétrico de agua en un suelo determinado.
En este proyecto se va a hacer uso de él para comprobar que la orina del bebé no desborde del pañal, en caso contrario, mojaría las sábanas, donde está ubicado el sensor, lo que implica un aumento de la humedad absoluta, de lo que serían informados los padres a través de la interfaz.
En caso de mojarse las sábanas por otra razón, como el derrame de líquido del biberón, el sensor captaría, de la misma manera, un aumento de la humedad informando a los padres sobre la situación.

- FC-28

Este módulo FC-28 [11] consiste en dos placas separadas entre sí por una distancia determinada. Ambas placas están recubiertas de una capa de material conductor que, en caso de existir humedad entre ellas, se creará un puente entre las puntas, lo que será detectado por un circuito de control con un amplificador operacional, LM393, encargado de transformar la conductividad registrada a un valor analógico.

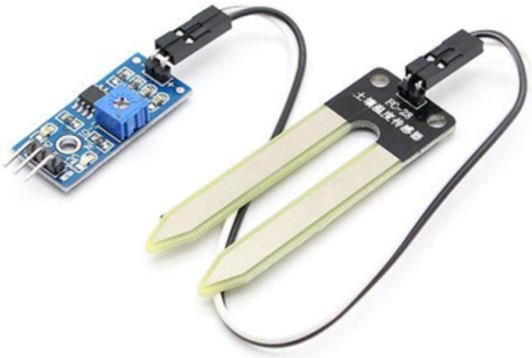


Figura 8. Sensor humedad absoluta FC-28. Obtenida de la página web Electro PC

En la salida analógica, el nivel de voltaje dependerá directamente de cuanta humedad haya en el entorno, es decir, de la conductividad existente entre las puntas del módulo.

El módulo también dispone de una salida digital, la cual entregará un pulso bajo cuando haya conductividad suficiente entre cada una de las puntas. Estos valores límite de conductividad que definirán la salida digital se regulan a través del potenciómetro que se encuentra en el circuito de control del módulo.

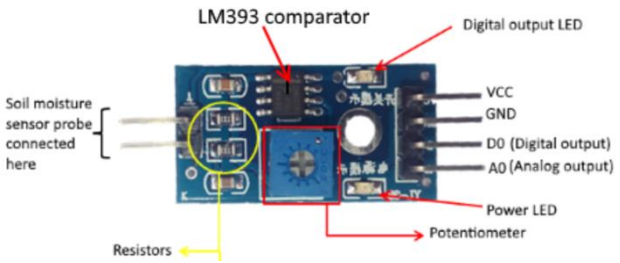


Figura 9. Módulo sensor humedad absoluta. Obtenida de la página web Electro PC

CARACTERÍSTICAS

- Potenciómetro de ajuste de umbral de salida digital.
- Indicadores LED de potencia y salida digital.
- Salidas analógicas y digitales.
- Orificio de montaje para una fácil instalación.

ESPECIFICACIONES TÉCNICAS

- Voltaje de entrada: 3,3 - 5 VCD.
- Voltaje de salida: 0 - 4,2 V.
- Corriente: 35 mA.
- A0: Salida analógica que entrega una tensión proporcional a la humedad.
- D0: Salida digital; este módulo permite ajustar cuándo el nivel lógico en esta salida pasa de bajo a alto mediante el potenciómetro.
- Dimensiones: 60 x 30 mm.

PRECIO: 3,79 €

- YL-69

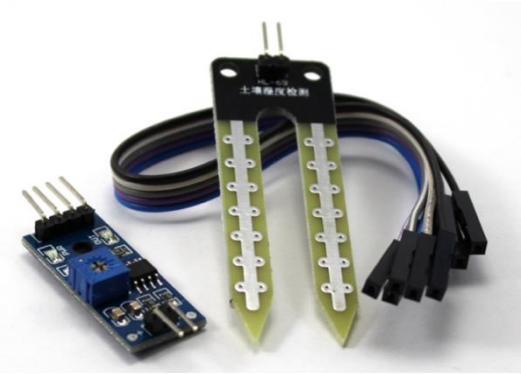


Figura 10. Sensor humedad absoluta YL-69. Obtenida de la página web LaRedElectrónica

El funcionamiento de este sensor se basa en la aplicación de una pequeña tensión entre los terminales del módulo YL-69 [12], que hace pasar una corriente que depende básicamente de la resistencia que se genera en el entorno, la cual depende mucho de la humedad. Por lo tanto, al aumentar la humedad la corriente crece y al bajar la corriente disminuye.

Del mismo modo que la sonda FC-28, el módulo consiste en una sonda YL-69 con dos terminales separados adecuadamente y un módulo YL-38 que contiene un circuito comparador LM393 SMD muy estable, un led de encendido y otro de activación de salida digital. Este circuito de control presenta 2 pines de conexión hacia el sensor, 2 pines para la alimentación y 2 pines de datos. VCC, GND, D0, A0, de lo que se deduce que existe una salida analógica y una digital que ofrecen respuesta del mismo modo que el módulo FC-28.

CARACTERÍSTICAS

- Potenciómetro de ajuste de umbral de salida digital.
- Indicadores LED de potencia y salida digital.
- Salidas analógicas y digitales.
- Utiliza Immersion Gold, que protege el níquel de la oxidación.

ESPECIFICACIONES TÉCNICAS

- Voltaje de entrada: 3,3 - 5 VCD.
- Voltaje de salida: 0 - 4,2 V.
- Corriente: 35 mA.
- A0: Salida analógica que entrega una tensión proporcional a la humedad.
- D0: Salida digital; este módulo permite ajustar cuándo el nivel lógico en esta salida pasa de bajo a alto mediante el potenciómetro.
- Dimensiones: 60 x 30 mm.

PRECIO: 1,95 €

- GS1

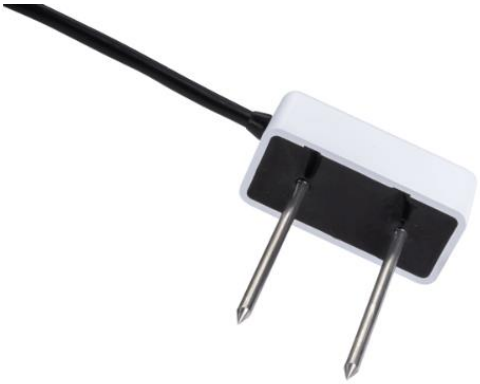


Figura 11. Sensor humedad absoluta GS1. Obtenida de la página web Lab Ferrer

La sonda GS1 [13] es un sensor capacitivo del tipo FDR (Reflectometría en el dominio de la frecuencia), los cuales miden la constante dieléctrica o permitividad del entorno para calcular su contenido de humedad. La fracción volumétrica del entorno ocupada por agua tiene una enorme influencia en la permitividad dieléctrica, ya que su valor dieléctrico ($\varepsilon = 80$) es mucho mayor que el de los otros constituyentes del entorno, como por ejemplo el aire ($\varepsilon = 1$).

Por este motivo, cuando la cantidad de agua varía, la sonda detecta y mide esta variación y la relaciona directamente con el cambio en el contenido de líquido.

CARACTERÍSTICAS

- Sonda NO sensible a la textura del entorno en que se encuentra.
- Circuito de control integrado.

ESPECIFICACIONES TÉCNICAS

- Voltaje de entrada: 3 - 15 VDC.
- Voltaje de salida: 1000mV a 2500 mV.
- Corriente: 15 mA.
- A0: Salida analógica que entrega una tensión proporcional a la humedad.
- Dimensiones: 9,3 x 2,4 x 6,5cm.

PRECIO: 9,69 €

2.2.2.3 Sensor de temperatura y humedad relativa

Los sensores de temperatura son dispositivos que transforman los cambios de temperatura en señales eléctricas que son procesados por equipo eléctrico o electrónico. Hay tres tipos de sensores de temperatura; los termistores, los RTD y los termopares.
Los sensores de humedad relativa se aplican para detectar el nivel de líquido en aire que se encuentra en una determinada área. Permiten medir la temperatura de punto de rocío, humedad absoluta y relación de mezcla mediante la detección del vapor de agua, midiendo la resistencia eléctrica entre dos electrodos.

En este proyecto el sensor de temperatura y de humedad relativa tiene la función de medir dichas magnitudes en el ambiente con el fin de informar y de regular los actuadores dependientes de éstas, como son la manta eléctrica y el humidificador.
Para ello se ha hecho uso de la familia DHTXX, que cuenta con un sensor capaz de obtener el valor de la temperatura y de la humedad relativa simultáneamente y, con un procesador, devuelve el valor obtenido.

El sensor de temperatura integrado en la familia DHTXX es de tipo termistor, éste está basado en que el comportamiento de la resistencia de los semiconductores es variable en función de la temperatura, aumentando los ohmnios de ésta con el descenso o aumento de la temperatura, según se trate de termistores tipo NTC o PTC, respectivamente.
El sensor de humedad integrado es de tipo capacitivo, el cual basa su funcionamiento en el cambio de la capacidad que sufre un condensador en presencia de humedad, al cambiar la permitividad del dieléctrico, por ejemplo, del aire con respecto a la humedad del ambiente.

➢ DHT21

El DHT21 [14] es un sensor digital de temperatura y humedad relativa de alta precisión. Integra un sensor capacitivo de humedad, un termistor para la medida de la temperatura y un microcontrolador encargado de realizar la conversión analógica a digital, como todos los sensores expuesto en este apartado de la familia DHTXX. A diferencia del resto, su empaque de plástico es más robusto comparado a los sensores DHT11 y DHT22, por lo que es ideal para aplicaciones en exteriores como control automático de temperatura, aire acondicionado y monitorización ambiental en agricultura.

Figura 12. Sensor Tª y humedad relativa DHT21. Obtenida de la página web Naylamps Mechatronics

La comunicación se realiza mediante el protocolo "Single bus".
El DHT21 posee mejores prestaciones respecto al sensor DHT11, como mejor resolución, mayor precisión y un empaque más robusto.

CARACTERÍSTICAS

- Extremadamente preciso.
- Estabilidad a largo plazo.
- Respuesta rápida.
- Fuerte capacidad anti-jamming.
- No compatible con el protocolo Dallas One-Wire.

ESPECIFICACIONES TÉCNICAS

- Voltaje de entrada: 3,3-5,5 VDC.
- Corriente: 1-1,5 mA.
- Rango de medición de temperatura: -40°C a 80 °C.
- Precisión de medición de temperatura: <±0.5 °C.
- Resolución Temperatura: 0.5°C.
- Rango de medición de humedad: De 0 a 100% RH.
- Precisión de medición de humedad: 3% RH.
- Resolución Humedad: 0.1%RH.

- Tiempo de sensado: 2s.
- Distancia: 25 m.

PRECIO: 3,75 €

➢ DHT11

El DHT11 [15] es un sensor con un microcontrolador de 8 bits y 2 sensores que se caracteriza por tener una señal de salida digital.
El protocolo de comunicación se realiza a través de un único hilo "Single-Bus", de forma que puede conectarse directamente a los pines de la Raspberry PI.

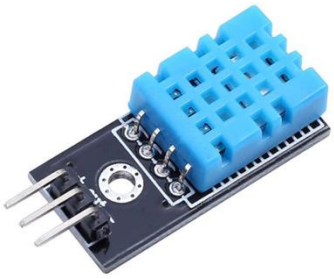


Figura 13. Sensor Tª y humedad relativa DHT11. Obtenida de la página web Naylamps Mechatronics

CARACTERÍSTICAS

- Rango de operación aceptable para este caso.
- Módulo con resistencia de protección.
- Condensador de filtrado.
- Respuesta rápida.

ESPECIFICACIONES TÉCNICAS

- Alimentación: 3 - 5 VDC.
- Corriente: 2,5 mA.
- Rango de medición de temperatura: 0°C a 50 °C.
- Precisión de medición de temperatura: ±2 °C.
- Resolución Temperatura: 0.1°C.
- Rango de medición de humedad: De 20 a 90% RH.
- Precisión de medición de humedad: 4% RH.
- Resolución Humedad: 1%RH.
- Tiempo de sensado: 1s.
- Distancia: 20 m.

PRECIO: 0,8 €

➢ DHT22

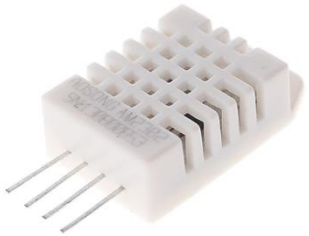


Figura 14. Sensor Tª y humedad relativa DHT22. Obtenida de la página web Naylamps Mechatronics

Este sensor [16] se caracteriza por tener la señal digital calibrada por lo que asegura una alta calidad y una fiabilidad a lo largo del tiempo, ya que contiene un microcontrolador de 8 bits integrado. Está constituido por dos sensores resistivos (NTC y humedad).

Cada sensor DHT22 está estrictamente calibrado en laboratorio, presentando una extrema precisión en la calibración.

Los coeficientes de calibración se almacenan como programas en la memoria OTP, que son empleados por el proceso de detección de señal interna del sensor.
El protocolo de comunicación es a través de un único hilo (protocolo 1-wire).

CARACTERÍSTICAS

- Alta calidad
- Estabilidad a largo plazo.
- Respuesta rápida.
- Fuerte capacidad anti-jamming.
- No compatible con el protocolo Dallas One-Wire.
- Sensor de humedad AM2301.

ESPECIFICACIONES TÉCNICAS

- Voltaje de entrada: 3,3-6VDC.
- Corriente: 2,5 mA.
- Rango de medición de temperatura: -40°C a 80 °C.
- Precisión de medición de temperatura: <±0.5 °C.
- Resolución Temperatura: 0.1°C.
- Rango de medición de humedad: De 0 a 100% RH.
- Precisión de medición de humedad: 2% RH.
- Resolución Humedad: 0.1%RH.
- Tiempo de sensado: 2s.
- Distancia: 20 m.

PRECIO: 3,40 €

3. Tecnologías hardware

Una vez estudiadas las posibilidades que se encuentran en el mercado para dar funcionalidad tanto a la parte de control como a los sensores de campo, en este apartado se expone la elección de cada componente que más se adapta a este proyecto, así como una explicación sobre la inclusión en el proyecto de cada elemento que lo compone.

3.1 Solución de control

La elección a debatir en este apartado radica en la placa de desarrollo, para esto se han visto tres opciones en el apartado anterior; Raspberry Pi, Arduino y Tessel 2.
De las tres opciones se ha escogido finalmente la placa de desarrollo Raspberry PI 3 Model B, la cual ofrece mayores prestaciones, aunque a costa de un precio más elevado, pero con esta elección aseguramos dar la correcta funcionalidad a todos los requisitos que se tienen previstos así como tener la opción de una futura mejora disponiendo de una placa de desarrollo que aúna el mayor número de posibilidades.

A continuación, se muestra una comparativa de la placa escogida con las otras dos opciones, donde podrá verse con mayor detalle las razones de elección de la Raspberry PI como unidad de control de este proyecto.

3.1.1 Raspberry Pi 3B vs Arduino

Partiendo de las especificaciones técnicas, se ha realizado una tabla comparativa para plasmar las diferencias entre ambas pacas de desarrollo.

	Raspberry PI	**Arduino**
Microcontrolador	64 bits	8 bits
RAM	1GB	2kB
Pines E/S	28	14
Precio (€)	38	26,17

Tabla 1. Rasperry PI vs Arduino

En cuanto a las entradas y salidas que ofrecen ambas placas; Arduino tiene un puerto USB-B, que puede ser usado por un ordenador para transferir nuevos programas, una entrada de alimentación y un conjunto de pines de E/S.
La Raspberry Pi tiene una salida de video, un puerto HDMI, un puerto para tarjetas SD, un conector de audio, un puerto para cámara CSI, un puerto de pantalla DSI, 4 puertos USB 2.0, un conector Gigabit Ethernet, LAN inalámbrica, Bluetooth 4.2 y pines de E/S (GPIO).

Respecto a la ejecución de aplicaciones diseñadas para rodar en estas placas; Arduino no tiene sistema operativo, solo puede ejecutar programas que fueron compilados para la plataforma Arduino, que en su mayoría significa programas escritos en C ++.
Raspberry Pi ejecuta un sistema operativo, que suele ser Linux y, por ello, el lenguaje de los programas volcados en ésta supone un abanico más amplio de posibilidades.

Teniendo en cuenta la necesidad de conexión con una pantalla táctil, que muestre constantemente la interfaz gráfica de la aplicación, así como el requisito de querer obtener imágenes en tiempo real con una cámara, siempre que el usuario lo desee, la balanza se inclina del lado de la Raspberry por las E/S que ofrece.

Así mismo, cabe destacar del lado de la programación, que también se cree conveniente el uso de Raspberry Pi por el hecho de que posibilita la realización de un programa de alto nivel que interactúe con una interfaz gráfica y con elementos de audio y vídeo.

3.1.2 Raspberry Pi 3B vs Tessel 2

Para seguir con la misma línea comparativa, se han plasmado en una tabla las principales características de ambas pacas de desarrollo.

	Raspberry PI	Tessel 2
Microcontrolador	64 bits	32 bits
RAM	1GB	32MB
Pines E/S	28	20
Precio (€)	38€	32€

Tabla 2. Raspberri Pi vs Tessel

Se puede observar en la *Tabla 2* una mejora en las prestaciones que ofrece la placa Tessel 2 respecto a la placa de Arduino, pero aún así no consigue superar las de Raspberry Pi.

En cuanto a las entradas y salidas que ofrecen ambas; El modelo B de Raspberry Pi 3 ofrece 28 pines GPIO más 12 pines de potencia y tierra mientras Tessel 2 ofrece 20 pines.

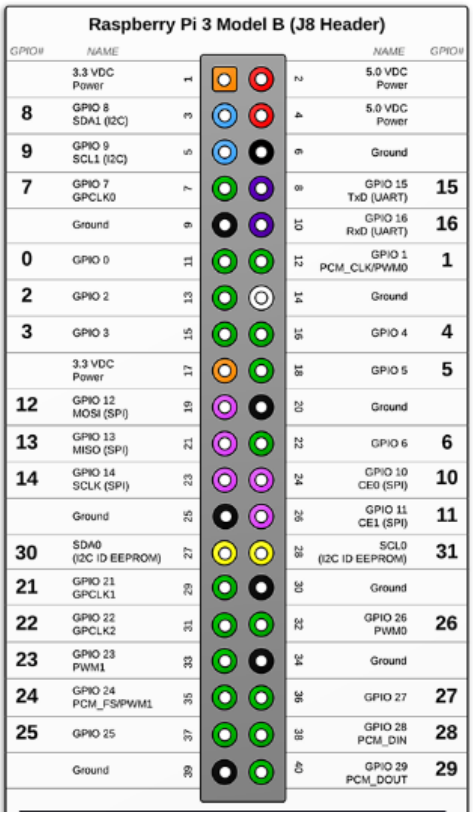


Figura 15. GPIOs Rasoberry Pi 3B. Obtenida de la página web de Raspberry Pi

Los protocolos de comunicación soportados por Raspberry son los siguientes: IIC (Circuito Inter-Integrado), SPI (Interfaz Periférica Serial) y UART (Receptor-Transmisor Asíncrono Universal), lo que amplía la funcionalidad de la placa ya que los buses IIC y SPI se pueden usar para conectar a las GPIOs múltiples conversores analógicos a digitales que se pueden usar para leer canales analógicos como sensores térmicos, sensores de humedad o sensores de CO2. Mientras tanto, la UART puede usarse para la comunicación entre múltiples Raspberry Pi.

Basándonos en la necesidad de conexión con una pantalla táctil, así como el requisito de emisión de audio y vídeo hace que sigamos a favor del uso de Raspberry Pi, ya que la placa de desarrollo Tessel 2 no dispone de conector HDMI ni control del táctil, mientras la Raspberry tiene capacidad

para transmitir videos Full HD de 60Hz, así como conectividad Bluetooth 4.1 y conector mini Jack para la conexión con altavoces.

Por último, cabe destacar como característica favorable que Raspberry Pi tiene un elevado número de seguidores, por lo que la cantidad de guías, tutoriales y *software* disponibles para la Raspberry Pi es mucho mayor que para Tessel.

3.2 Solución de campo

En cuanto a tecnología *hardware* de campo se entiende el conjunto de sensores, actuadores y periféricos que conforman el prototipo desarrollado. Se va a exponer la razón de cada elección, así como su adaptabilidad, en cuanto a conexionado y comunicación, de cada elemento en el proyecto.

3.2.1 Sensores

En el apartado anterior se han expuesto varios tipos de sensores de acuerdo a los parámetros que desean obtenerse del ambiente y que dotarán de información a la placa de desarrollo, con el fin de que ésta los procese y ofrezca una respuesta.

- *Calidad de aire*

En cuanto al sensor de calidad de aire, como se ha visto en el apartado anterior, para este proyecto, el gas del que se precisa la detección con mayor precisión es el gas metano, proveniente de las heces del bebé.
Con el fin de facilitar la elección se ha realizado una tabla comparativa con las características más relevantes para este caso de cada sensor de la familia MQ.

	MQ2	**MQ4**	**MQ9**
Rango detección (ppm)	300-10000	200-10000	100-10000
Gas característico	Isobutano	Metano	CO, GLP
Precio (€)	2,89	3,59	7,29

Tabla 3. Comparativa sensores calidad de aire

Se puede observar en la *Tabla 3* que el precio asciende en función del rango de detección de gases para cada sensor.
Debido a que el gas que se desea medir es el metano, aunque todos los sensores expuestos detecten este gas, se ha optado por el sensor **MQ4** debido a que es el gas al que mayor sensibilidad ofrece, siendo su gas característico. Además, se encuentra en un rango de precios aceptable.

Como se ha expuesto en el capítulo anterior, el sensor MQ4 está compuesto por un sensor electro-químico que varía su resistencia al estar en contacto con el gas metano.
Este sensor ofrece una salida digital y una analógica, teniendo en cuenta el uso de Raspberry Pi, la cual solo ofrece entradas digitales, será la salida digital la que se utilice, conectándola a una

entrada digital de la Raspberry Pi. Opcionalmente, se puede calibrar el umbral de disparo de la salida digital con el potenciómetro instalado en el módulo que acompaña a este sensor.

En cuanto al esquema eléctrico, *Figura 16*, el sensor requiere dos entradas de tensión; voltaje del calentador (VH) y voltaje del circuito (VC). VH se utiliza para suministrar temperatura de trabajo estándar al sensor y puede adoptar alimentación de CC o CA, mientras que VRL es el voltaje de la resistencia de carga RL que está en serie con el sensor. Vc suministra la tensión de detección a la resistencia de carga RL y debe adoptar alimentación de CC.

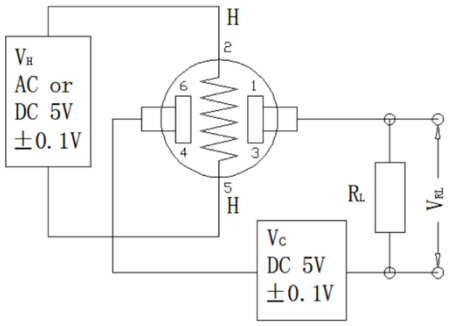


Figura 16. Circuito interno sensor MQ4. Obtenida del datasheet del sensor MQ4

Con lo cual, se alimentará el módulo MQ4 conectando los pines de alimentación a las salidas 5V y GND de la Raspberry Pi, 04 y 06 respectivamente, y la salida de datos a la entrada digital GPIO13 de la Raspberry, que corresponde con el pin 21.

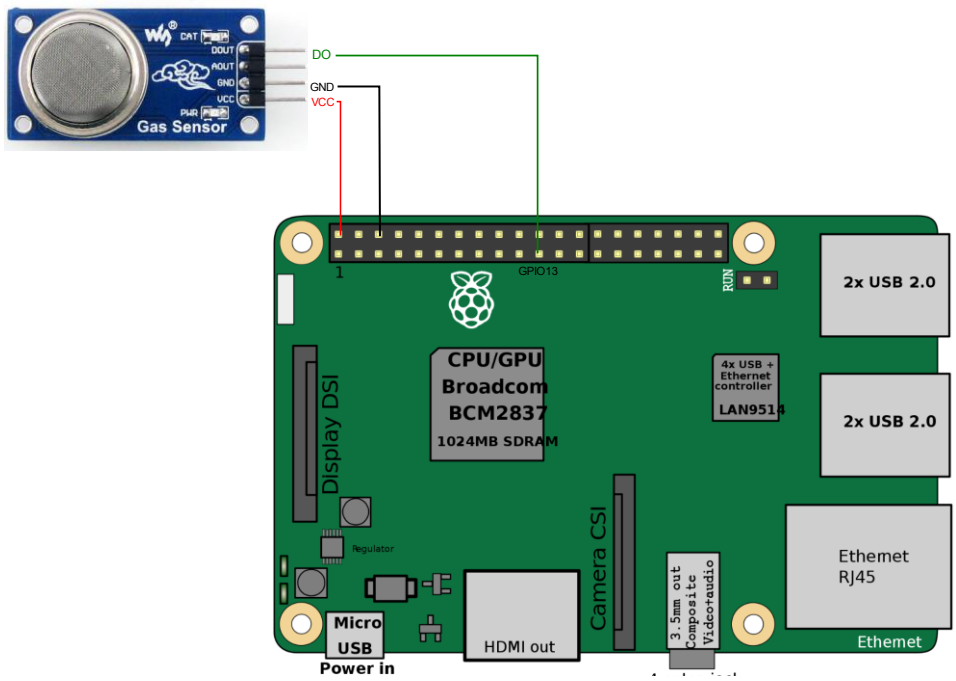


Figura 17. Conexión sensor calida de aire

- *Humedad absoluta*

En el capítulo anterior se presentaron tres tipos de higrómetros que detectaban la presencia de líquido en el medio, ya que en presencia de éste surgía el paso de corriente entre los pines de estos sensores.
Con el fin de facilitar la elección se ha realizado una tabla comparativa con las especificaciones y características de los estos tres sensores que más convendría valorar para la realización de este proyecto.

	FC-28	YL-29	GS1
Dimensiones (mm)	60x30	60x30	93x24x65
Salida digital	SÍ	SÍ	NO
Precio (€)	3,79	1,95	9,69

Tabla 4. Comparación sensores humedad absoluta

En primer lugar, se descarta la sonda GS1 debido a que no tiene salida digital, por lo que no tendría cavidad la entrada de datos en la Raspberry PI desde este dispositivo, aparte de por sus dimensiones, ya que los otros sensores son planos, por lo que ocupan menos espacio y finalmente, el precio de esta sonda supera al del resto.
En cuanto a los sensores FC-28 y YL-29 presentan características similares, por lo que se ha escogido el **YL-29** teniendo en cuenta el precio y la disponibilidad de mercado.

El sensor YL-69 tiene la capacidad de medir la humedad absoluta del medio en el que se encuentre. Su funcionamiento radica en la inyección de una pequeña tensión entre los terminales del módulo YL-69 que hace pasar una corriente que depende básicamente de la resistencia que se genera en el medio.
En cuanto a la conexión con la Raspberry PI, este sensor también requiere 5V de alimentación y la salida de datos se conectará a otra entrada digital de la placa de desarrollo, por lo que la conexión será la siguiente.

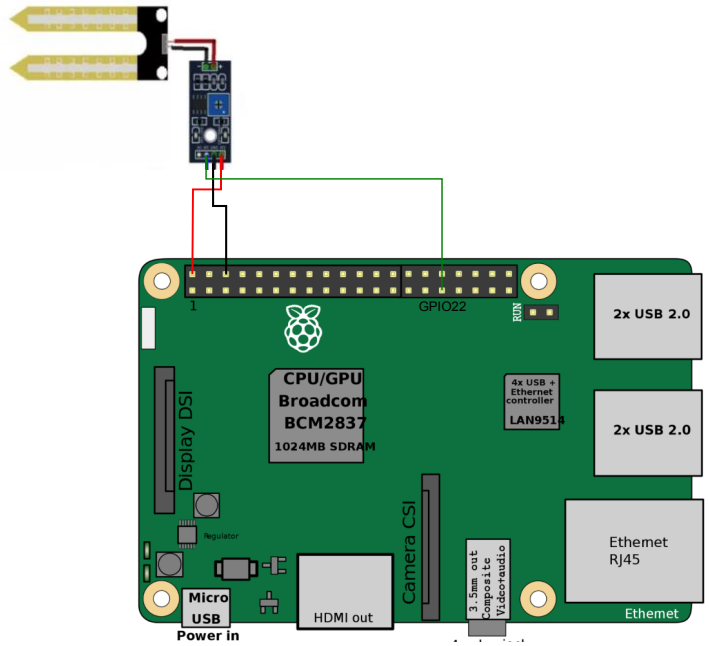


Figura 18. Conexión sensor humedad absoluta

La alimentación de 5V y GND será proporcionada por los pines 04 y 06 respectivamente, al igual que para el resto de sensores y la entrada de datos se realizará a partir de la GPIO 22, que corresponde con el pin 31.

- *Temperatura y humedad relativa*

La temperatura y la humedad relativa son valores comúnmente medidos en diversos sistemas de control, por este hecho en el mercado existen sensores que miden ambas magnitudes a la vez, este es el caso de la familia DHT que reúne un numeroso campo de sensores dedicados a la medición de la temperatura y humedad relativa, los cuales ofrecen diferentes rangos y precisiones en la medida de estas magnitudes. En este caso se ha realizado una tabla comparativa con dichos parámetros con el fin de escoger el que más se adapte a este proyecto teniendo en cuenta las condiciones del ambiente que pretende tomar.

	DHT11	DHT22	DHT21
Rango temperatura (ºC)	0 - 50	-40 - 80	-40 - 80
Precisión temperatura (ºC)	20 – 90	0 - 100	0 - 100
Rango hum. Relativa (% RH)	2	0,5	0,5
Precisión hum. Relativa (% RH)	4	0,5	0,5
Precio (€)	0,8	3,4	3,75

Tabla 5. Comparación sensores temperatura y humedad relativa

Puede observarse que el sensor DHT22 Y DHT21 tienen características iguales en cuanto a rango de medida y precisión, al igual que un precio similar.
El sensor DHT11 ofrece peores características de medida pero tiene un precio muy asequible. Por tanto, y teniendo en cuenta las condiciones en que se va a encontrar la cuna donde estará ubicada este sensor, se ha escogido el sensor **DHT11**, debido a que la cuna se situará normalmente en una habitación, dentro de la cual no se van a dar condiciones de temperatura y humedad relativa fuera de los rangos que ofrece este sensor.

La principal dificultad que presenta el sensor DHT11 es que ofrece salida analógica y ninguna digital, la cual no puede introducirse en la Raspberry Pi. Pero debido al gran uso que se le ha dado a este sensor por su funcionalidad y precio, se ofrece en el mercado un conjunto de sensor más módulo compactado en un mismo dispositivo que ofrece la conversión.

Entonces partimos de una señal analógica que luego es convertida a formato digital dentro del propio dispositivo.
La transmisión de los datos se realiza en formato digital, con una trama de 40 bits formando 4 grupos de 8 bits, Figura 19. Cada uno de estos grupos representan la información de temperatura ambiente y humedad relativa.

Figura 19. Conversión sensor DHT11. Obtenida de la página web ElectroniLab

Los bits de paridad nos permiten saber si la toma de la lectura es correcta, sumando los cuatro primeros grupos de 8 bits nos debe dar los bits de paridad del último grupo de 8 bits.

Se ha utilizado el modelo DHT11 que viene integrado en una PCB, el cual ya incluye la resistencia pull up de 5kΩ, conectada al pin de comunicación para elevar la tensión que sale del sensor mientras éste está en reposo y así evitar lecturas erróneas si este pin no está recibiendo una señal.

En cuanto a la conexión con la Raspberry PI, al igual que el resto, este sensor requiere 5V de alimentación y la salida de datos se conectará a otra entrada digital de la placa de desarrollo, por lo que la conexión será la siguiente.

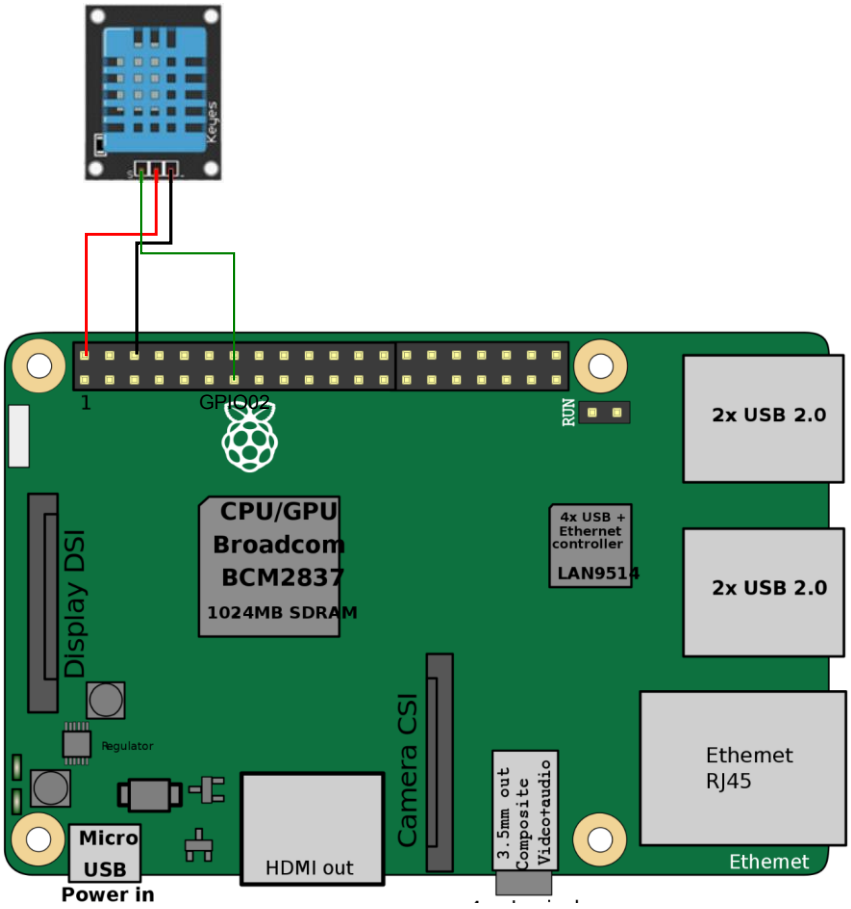


Figura 20. Conexión sensor de temperatura y humedad relativa

La alimentación de 5V y GND será proporcionada por los pines 04 y 06 respectivamente, al igual que para el resto de sensores, y la entrada de datos se realizará a partir de la GPIO 02, que corresponde con el pin 13.

3.2.2 Actuadores

En cuanto a los actuadores presentes en este proyecto, cabe destacar que se han escogido o modelado en función de las necesidades del prototipo, de la disponibilidad y de la economía.
Todos ellos basan su activación según el código de programación volcado en la Raspberry Pi y suponen salidas digitales de ésta, que serán activadas en función de valores de entrada, ya sea de sensores o de interactuación con la interfaz.

- *Placa de relés*

La Raspberry Pi ofrece una tensión máxima de salida de 5V en sus salidas digitales, sin embargo, los actuadores utilizados en este proyecto necesitan una mayor tensión para su activación. Por ello se va a emplear una placa que consta de un bloque de ocho relés, gobernada desde la Raspberry Pi, para manejar la alimentación de los actuadores que funcionan a 230VAC y 12VDC.
Los relés integrados en la placa soportan una intensidad de hasta 10A a 250V y ninguno de los actuadores que se van a utilizar superan esta intensidad.

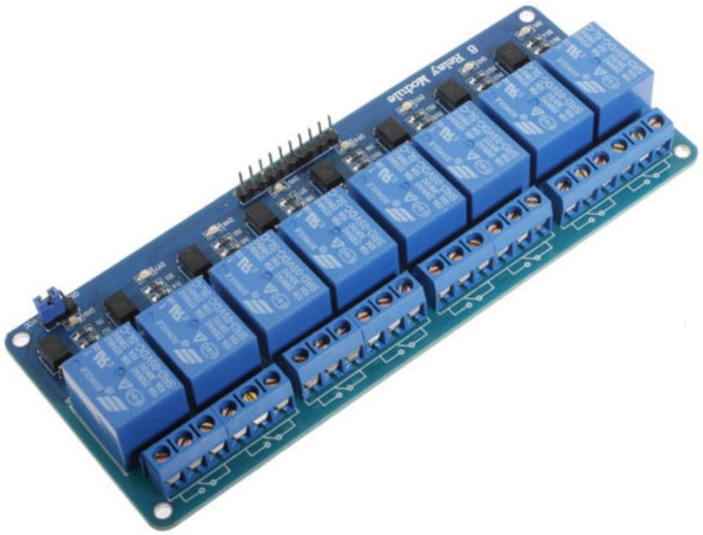


Figura 21. Placa de relés. Obtenida de la páina web PC components

La placa de relés por la que se ha optado es la que se muestra en la *Figura 21*, que trata de un módulo de ocho relés compatible con la Raspberry Pi, dado que funciona con alimentación de 5V. La función de este dispositivo es cerrar o abrir el circuito de alimentación a parir de una señal de 5V proveniente de la Raspberry Pi.

PRECIO: 10,99€

El uso de esta placa radica en el principio de funcionamiento de un relé, el cual está compuesto por una bobina que, al circular una pequeña corriente (3.3V, 5V o 12V) por ella genera un campo

magnético que produce el desplazamiento de la armadura metálica y, con ella, el juego entre los contactos NC-C o NA-C. Los contactos de los relés son tres; C (Común), NC (Normalmente Cerrado), y NA (Normalmente Abierto) respecto al común.
El contacto NA con C permanece abierto mientras no se actúe sobre el relé y cuando se actúa sobre él, se cierra y permite la circulación eléctrica a través de él. El contacto NC con C permanece cerrado la gran parte del tiempo, permaneciendo en un estado de reposo, hasta que se excite el relé y se ponga en contacto NA con C, dejando pasar la corriente de un contacto a otro y, por tanto, la tensión presente en el contacto C pasará a alimentar el actuador que esté conectado al contacto NA.

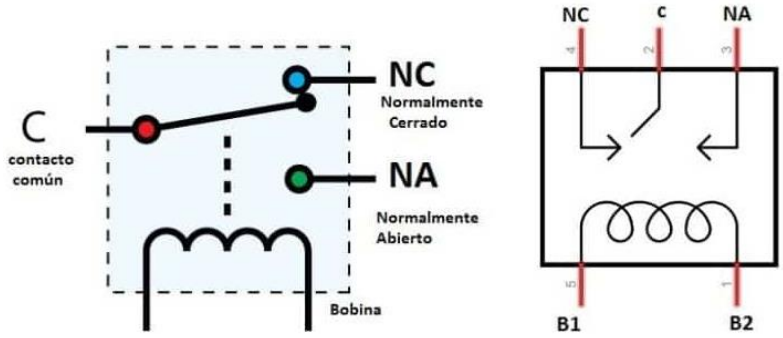


Figura 22. Principio funcionamiento relé. Obtenida de la página web MundoMotor

- *Humidificador*

Desde el ámbito médico, se recomienda que el ambiente del hogar y especialmente donde está el niño, se mantenga en un rango de humedad relativa entre el 40% y el 60%, en ciertas ciudades estos porcentajes son difíciles de obtener bajo condiciones ambientales normales, por lo que una forma de encontrarse en este rango es mediante un humidificador.
Con él se pretende mejorar la humedad de la estancia y favorecer que el aire sea menos agresivo para las vías respiratorias, lo que también se traduce en un menor riesgo de que los microorganismos causantes de infecciones respiratorias proliferen.
Además, conseguimos que el aire no sea tan seco, lo que evita que el aire produzca irritaciones en la garganta y la piel, muy sensibles en el caso de los bebés.
El modo de funcionamiento de los humidificadores de vapor caliente radica en la utilización de la evaporación por calor, los cuales calientan el agua del depósito hasta lograr una cierta temperatura que permite emitir el vapor.

El modelo de humidificador que se ha empleado para este prototipo es el i-o3 mini de la marca VIDA10, que es un generador ozono, agua y aire, que ofrece características similares a un humidificador, con salida de Ozono de 400 mg/h (HIGH) a 200mg/h (LOW).

Este humidificador tiene la posibilidad de regular la salida de ozono analógicamente, lo que no es útil para este proyecto debido a que esta regulación se desea realizar mediante la aplicación que controla toda la funcionalidad de la cuna. Por esto se han realizado unos ajustes para dotar a este humidificador únicamente de una funcionalidad digital todo/nada, debido a necesidad de salidas digitales de la Raspberry Pi. Estos ajustes radican en la desconexión de la parte de regulación del humidificador, conectando únicamente la alimentación al circuito de control de forma que queda limitado a la función on y off.

Figura 23. Humidificador i-o3 mini. Obtenida de la página web Amazon

En primer lugar, la conexión interna del humidificador era la que se muestra en la *Figura 24*, la cual integra en su configuración la placa de regulación.

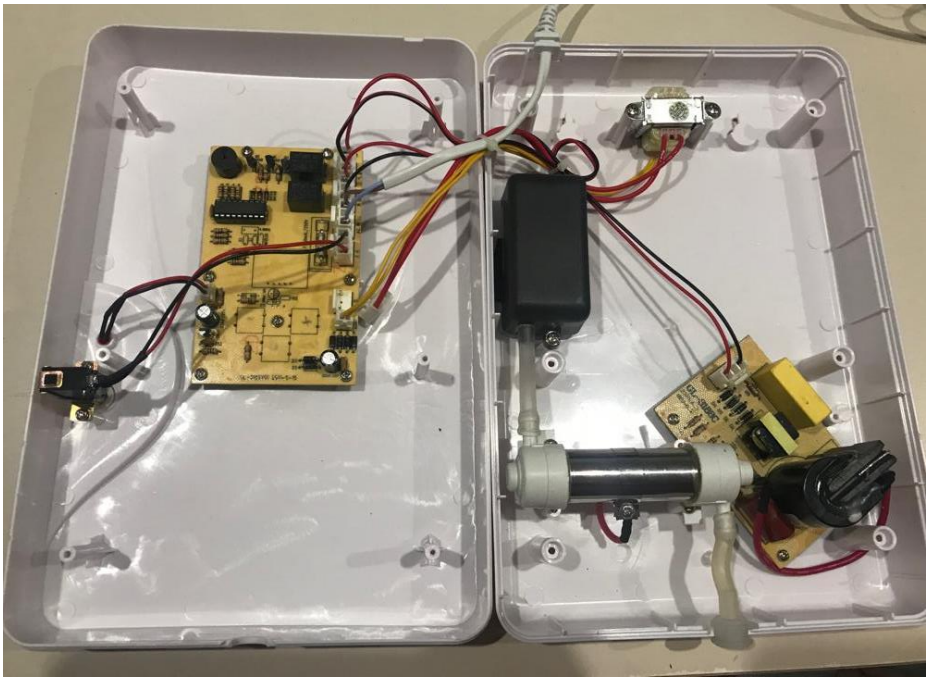


Figura 24. Configuraciuón interna humidificador con circuito de regulación

Como puede observarse, anteriormente, el cable de red (cable blanco) se conectaba a la placa de regulación, y ésta a la de alimentación.

Para limitar el humidificador a la función on-off, se han desconectado el cable de tensión de red de la placa de regulación y se ha conectado directamente a la parte de alimentación, quedando la placa de regulación inutilizada, sin ninguna conexión, y la placa de alimentación conectada directamente a la tensión de red mediante una borna de conexiones.

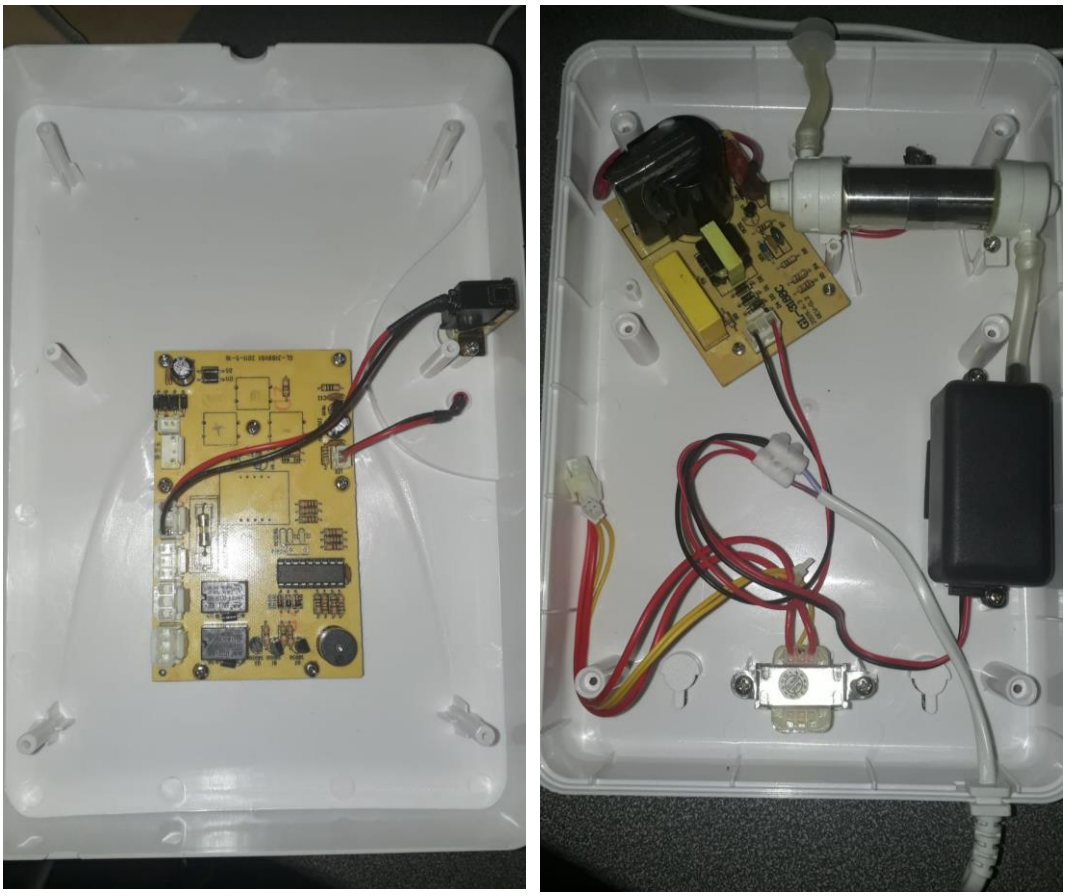


Figura 25. Configuración interna humidificador sin circuito de regulación

PRECIO: 59,00€

La activación del humidificador en este prototipo se puede realizar mediante el modo automático, lo que supone que, por debajo de un punto de consigna de humedad relativa, indicado por el usuario mediante la interfaz, se activará el humidificador y, por encima de éste, se apagará. También mediante la interfaz, puede activarse el humidificador de forma manual; cuando el usuario crea necesario puede activar directamente el humidificador desde la pantalla táctil, y del mismo modo, apagarlo. El flujo de trabajo que sigue la activación del humidificador puede verse de forma gráfica en el apartado 4.1.4.2, donde se muestra el grafcet de funcionamiento.

Este dispositivo se alimenta a 230 VAC, por lo que la entrada central al relé será el cable de fase de la toma de red, mientras el neutro de la toma de red y el negativo del dispositivo se unirán con el fin de igualar ambas tomas de tierra.

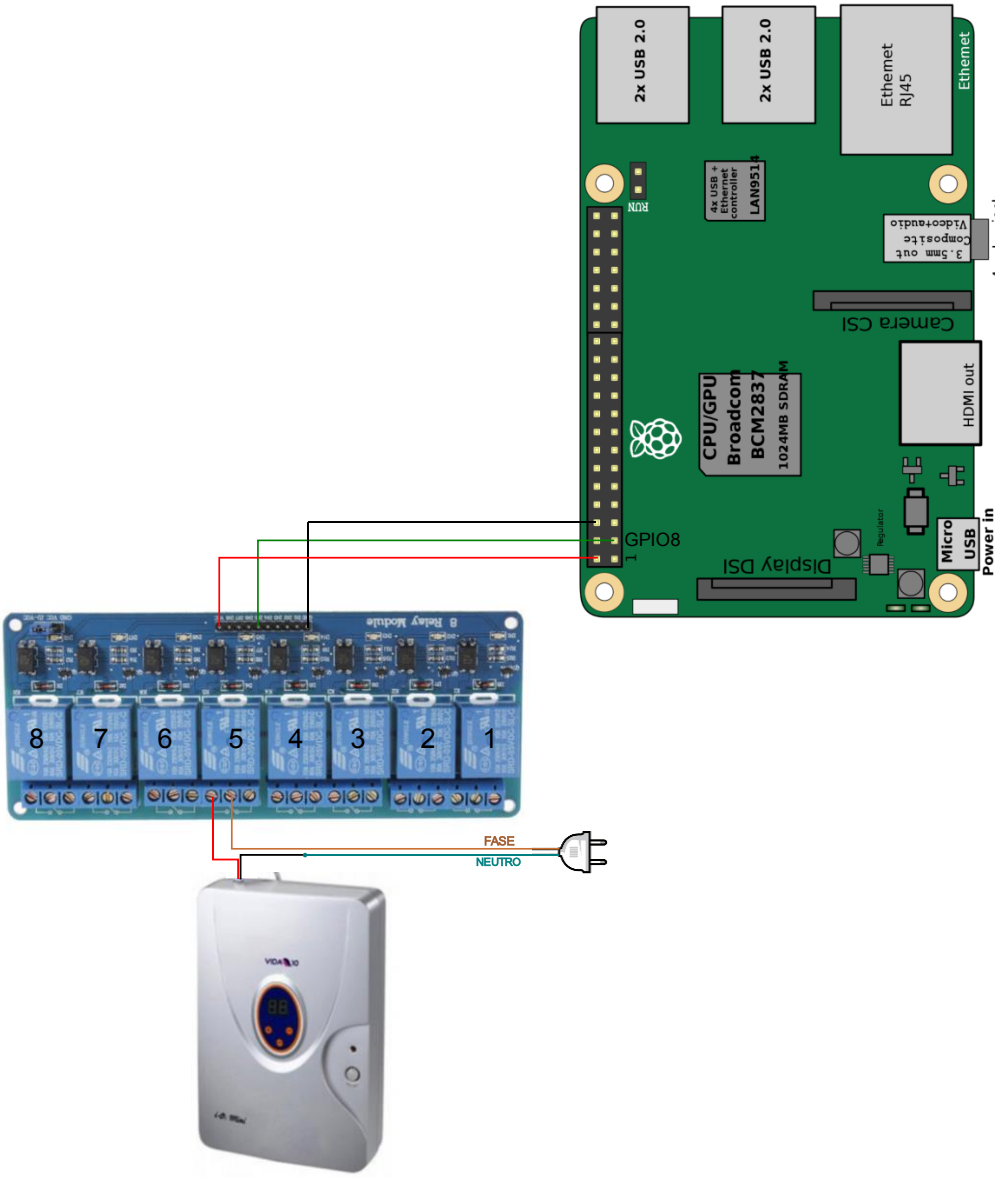


Figura 26. Conexión humidificador

Conforme muestra la *Figura 26*, el positivo del humidificador se conectará con el pin NA del relé, mientras el relé se mantendrá en estado normalmente cerrado (NC), sin alimentar el humidificador, conectando los 230VAC a un pin vacío. En el momento que lo ordene la Raspberry Pi, mediante un pulso de 5V que irá al relé 5, éste se excitará cambiando el contacto de estado, haciendo pasar la corriente desde el cable de alimentación de 230V (pin C) al positivo del humidificador (pin NA), resultando así su activación.

El relé que activa el humidificador es el número 5, que va conectado a la salida digital GPIO08 de la Raspberry Pi, que corresponde con el pin 03 de ésta.

- *Manta eléctrica*

Una de las condiciones que más debe cuidarse es la temperatura a la que se encuentran los bebés, por ello es una de las medidas que integra este prototipo, la temperatura alrededor de la cuna. Para actuar sobre ella pueden usarse diferentes dispositivos, aunque el que se ha integrado en este proyecto es una manta eléctrica, que se sitúa entre el colchón y la sábana, por seguridad del bebé.
Otra de las razones de la aplicación de calor mediante una manta eléctrica son los cólicos en los bebés, los cuales aparecen porque el sistema digestivo de los recién nacidos no está totalmente desarrollado y ello puede generar malestares en la tripa del niño, suelen ser más asiduos durante los primeros 3 meses de vida, y está demostrado que el calor puede ayudar a aliviar este malestar.

La manta eléctrica que se ha escogido es la almohadilla térmica eléctrica LESHP de 100W y 6 niveles de temperatura de 30 x 60cm de Amazon, la cual ofrece regulación de la temperatura. Al igual que el humidificador, se ha puenteado la regulación con el fin de limitar la manta eléctrica a la función on-off, de tal modo que la regulación forme parte de la aplicación.

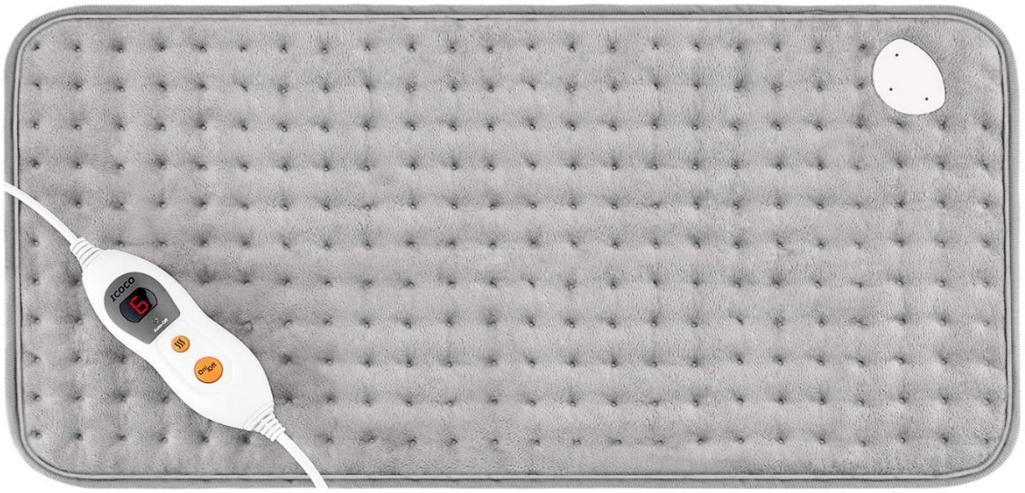


Figura 27. Manta eléctrica. Obtenida de la página web Amazon

Se ha pensado en la elección de una manta eléctrica con dimensiones similares a las de la cuna con el fin de calentar toda la base de ésta.

PRECIO: 23,99€

La activación de la manta eléctrica se controla mediante *software*, siguiendo el flujo de trabajo que se muestra en el apartado 4.1.4.2; se puede realizar en modo automático, lo que supone que, por debajo de un punto de consigna de temperatura indicado por el usuario mediante la interfaz, se activará la manta eléctrica y por encima de este punto la apagará. También mediante la interfaz, puede activarse la manta eléctrica de forma manual; cuando el usuario crea necesario puede activar la manta eléctrica desde la pantalla táctil y, del mismo modo, apagarla.

Este dispositivo se alimenta a 230 VAC, por lo que la entrada central al relé será el cable de fase de la toma de red, puenteado desde la borna de alimentación del relé del humidificador, mientras

el neutro de la toma de red y el negativo del dispositivo se unirán con el fin de igualar ambas masas.

El positivo de la manta eléctrica se conectará al pin NA del relé, de forma que el relé se mantendrá en estado normalmente cerrado (NC), conectando los 230VAC a un pin vacío, sin alimentar la manta, y ,en el momento en que lo ordene la Raspberry Pi, mediante un pulso de 5V que irá al relé número 7, éste se excitará cambiando el contacto de estado, haciendo pasar la corriente desde el cable de alimentación de 230V al positivo de la manta, resultando así su activación.

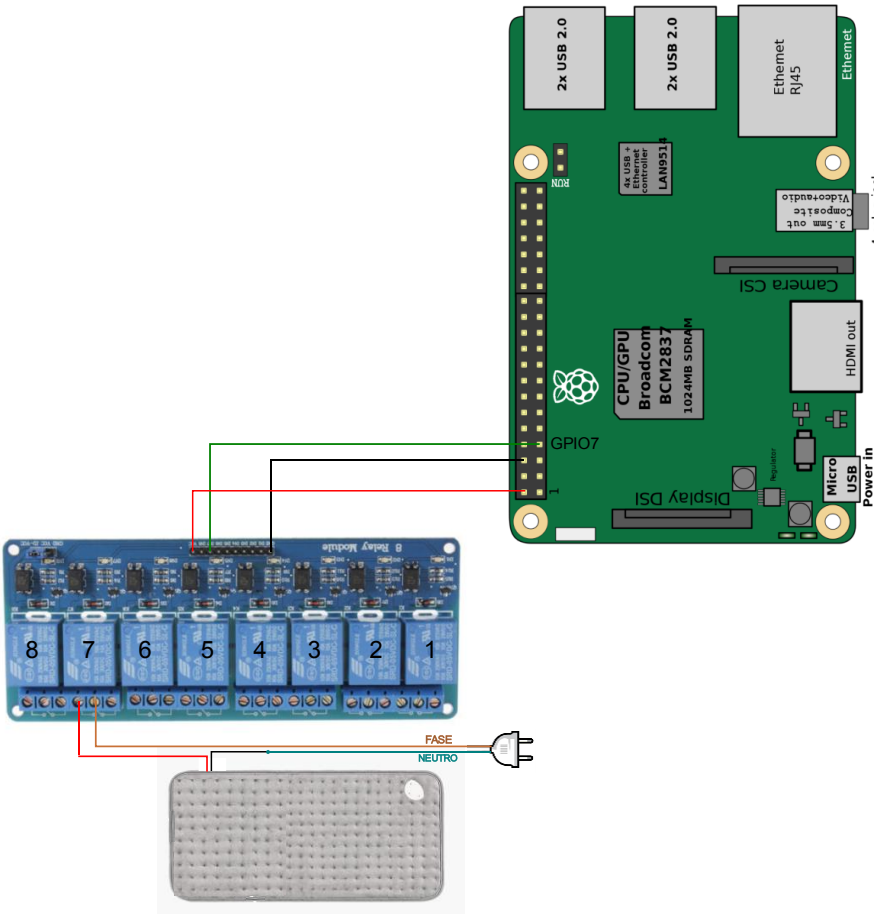


Figura 28. Conexión manta eléctrica

El relé que activa la manta eléctrica es el 7, que va conectado a la salida digital GPIO07 de la Raspberry Pi, que corresponde con el pin 07 de ésta.

- *Motores lineales*

La finalidad de los motores lineales en la cuna es el balanceo de ésta con el fin de relajar al bebé y dormirlo.
El balanceo se asemeja a la situación en el vientre materno; al estirar una pierna para dar un paso, el vientre materno baja unos centímetros solo para volver a subir de nuevo cuando los pies se cruzan, a este movimiento se le suma el de izquierda a derecha a medida que apoyamos un pie en el suelo y luego el otro durante la marcha. El bebé percibe todos estos movimientos de

balanceo cuando se encuentra en el útero por lo que al ser balanceados se sienten seguros y por tanto su sistema se relaja.
El balanceo se consigue posicionando un motor lineal a cada lado longitudinal del somier de la cuna, de forma que al extenderse los motores alternadamente, mueven el somier de lado a lado simulando el balanceo de la cuna.

Se ha escogido un accionador de extensión con una fuerza de empuje o tracción de hasta 150kg /140kg, el cual incluye un motor de alto rendimiento e interruptor de carrera.

Figura 29. Motores lineales. Obtenida de la página web Ebay

La longitud de carrera se ha escogido de 50 mm, la mínima disponible, debido a que para el balanceo de la cuna no es necesaria mayor longitud y la velocidad sin carga que proporciona el motor es de 6mm/s, ya que se necesita una velocidad reducida para que el movimiento de la cuna no sea brusco y se consiga un balanceo suave que relaje al bebé.

PRECIO: 28,8€

La activación de los motores lineales se realiza mediante la interfaz, como muestra el grafcet del apartado 4.1.4.4, cuando el usuario pulsa la opción de balanceo.

Para la acción de balanceo se utilizan 4 relés; dos para la acción impulsión y la de tracción del motor derecho y otros dos para la acción de impulsión y tracción del motor izquierdo. De forma que, cuando quiera activarse el balanceo, la Raspberry Pi, mediante la programación, manda pulsos de 5V alternados, cada cierto intervalo de tiempo, a los relés corresponientes. Estos pulsos son enviados través de sus salidas digitales a los relés, excitándolos y alimentando las acciones de impulso y tracción de ambos motores, consiguiendo así un balanceo constante durante el tiempo que el usuario desee.

La secuencia de pulsos será la siguiente:

1. Motor derecho e izquierdo arriba, asegurando el reposo para comenzar la secuencia.
2. Motor derecho arriba mientras el motor izquierdo desciende.

3. Motor izquierdo asciende mientras el motor derecho desciende.
4. Motor derecho asciende mientras motor derecho desciende.

Y así sucesivamente hasta que se mande la orden de paro; entonces el motor que se encuentre abajo o descendiendo, acabará su movimiento y ascenderá hasta su posición de reposo, mientras el que se encuentra arriba o subiendo acabará su movimiento y se quedará arriba, en su posición de reposo.

Con lo que, cuando se desactive desde la interfaz la opción de balanceo, la Raspberry deja de alimentar los relés y los motores lineales volverán a su posición de reposo antes de apagarse, quedando la base de la cuna recta en su punto más elevado y preparados para la próxima actuación.

Estos dispositivos se alimentan a 12 VDC, por lo que una de las entradas de los relés (NA) será el cable positivo de una batería de 12VDC, el cual se llevará al pin NA de los cuatro relés mediante puentes. Mientras el negativo de la batería se lleva a la entrada NC del relé y se puentea hasta todas las entradas NC de los demás relés, como se puede observar en la *Figura 30*.

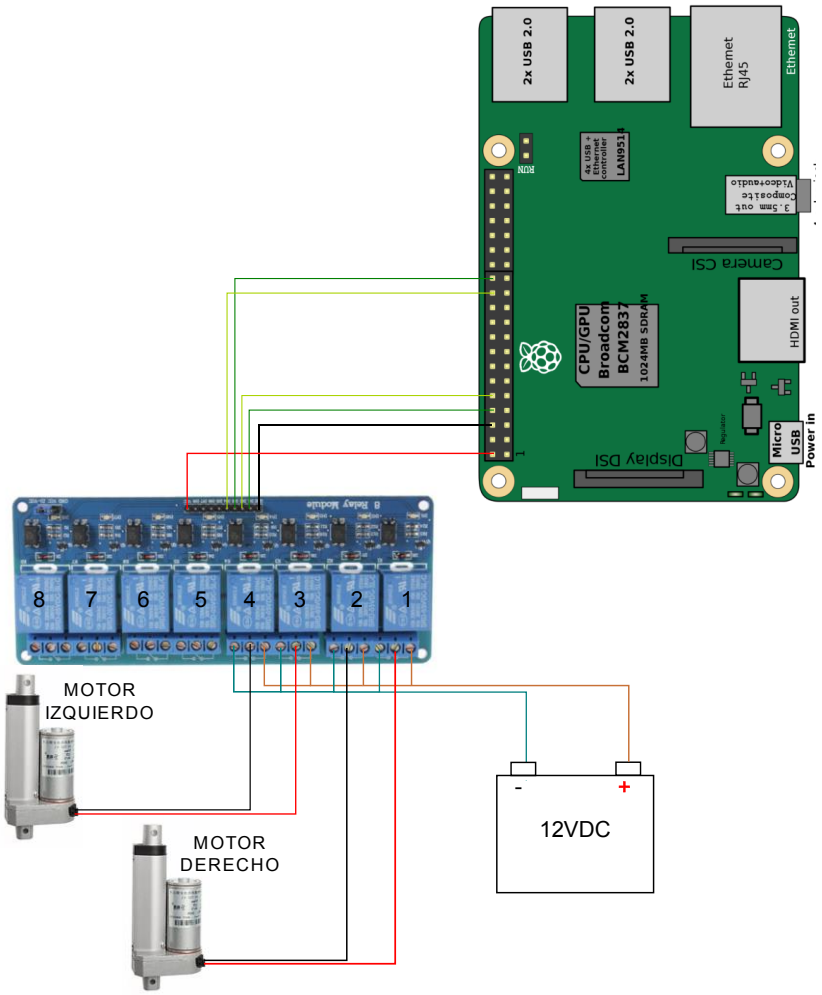


Figura 30. Conexión motores lineales

El positivo (motor derecho arriba) y el negativo (motor derecho abajo) del motor derecho se lleva al borne central de los relés 1 y 2, respectivamente y a las salidas digitales GPIO15 y GPIO16 de la Raspberry Pi.

El positivo (motor izquierdo arriba) y el negativo (motor izquierdo abajo) del motor izquierdo se lleva al borne central de los relés 3 y 4, respectivamente y a las salidas digitales GPIO11 y GPIO10 de la Raspberry Pi.

- *Proyector*

Tanto para la distracción como la relajación de los bebés se suelen utilizar lámparas quitamiedos que proyectan imágenes o luces en el techo, de forma que el bebé no se sienta totalmente a oscuras mientras se relaja con el movimiento de las luces proyectadas.

Figura 31. Proyector de luces. Obtenida de la página web Amazon

Por ello se ha querido integrar este dispositivo en el prototipo, de forma que pueda activarse también desde la interfaz que controla el usuario.

El proyector de luces que se ha escogido es una lámpara que emite luces de varios colores a través de tres leds, y con el movimiento giratorio de la cápsula que los envuelve se aprecia la proyección de las luces en movimiento.

Para poder gobernar la activación del proyector es necesario sacar dos cables de la bombilla que simule el cable positivo y negativo, para tratarlo como cualquier otro dispositivo con polaridad; para esto se ha utilizado un casquillo , como el que se muestra en la *Figura 32*, que se enrosca a la rosca metálica del proyector y que dispone de dos cables, aunque se pueden utilizar indistintamente como positivo y negativo.

Figura 32. Casquillo proyector. Obtenida de la página web Ebay

PRECIO: 3,99 €

La activación del proyector se realiza mediante la interfaz cuando el usuario pulsa la opción de proyección, conforme muestra el grafcet expuesto en el apartado 4.2.4.3.

Para la activación del proyector se utilizan un relé, de forma que, cuando quiera activarse el proyector, la Raspberry Pi, mediante la programación, manda un pulso de 5V al relé correspondiente y al excitarlo deja pasar la alimentación desde la toma de red.

Este dispositivo se alimenta a 230 VAC, por lo que una de las entradas al relé (C) será el cable de fase de la toma de red, el cual compartirán, mediante puentes, todos los relés correspondientes a actuadores que se alimenten a 230 VAC. Mientras el neutro de la toma de red y el negativo del proyector se unen con el fin de igualar ambas masas, conforme muestra la *Figura 33*.

El positivo del proyector se colocará en el pin NA del relé, mientras el relé se mantendrá en estado normalmente cerrado (NC) sin alimentar el proyector. En el momento en que lo ordene la Raspberry Pi, mediante un pulso de 5V que irá al relé pertinente, éste se excitará cambiando el contacto de estado, haciendo pasar la corriente desde el cable de alimentación de 230V al positivo del proyector, resultando así su activación.

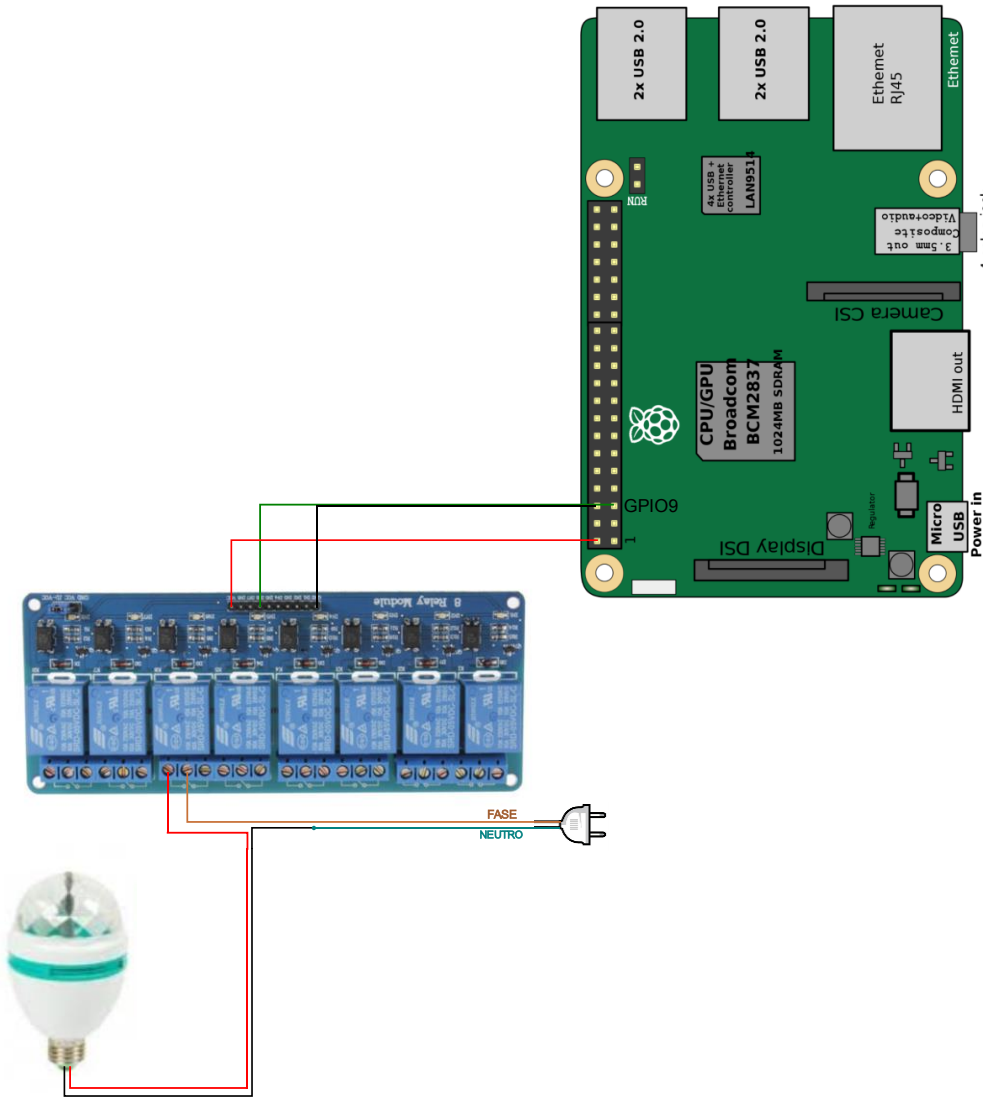


Figura 33. Conexión proyector

El relé que activa el proyector es el número 6, que va conectado a la salida digital GPIO09 de la Raspberry Pi, que corresponde con el pin 5 de ésta.

3.2.3 Periféricos

Los periféricos son dispositivos que sirven para la comunicación entre el procesador, en este caso la Raspberry PI, y el medio externo. Proporcionan la capacidad de introducir información en la placa de control desde el exterior, como es el caso de la cámara que recopila imágenes del exterior con el fin de reproducirlas. Así como también pueden proporcionar información al exterior desde

la Raspberry, como es el caso de los altavoces, en el que el control emite ondas sonoras mediante éste al exterior.

También existen periféricos que realizan ambas funciones, como es el caso de una pantalla táctil, la cual muestra información al usuario y éste puede introducir información al ordenador a través de la pantalla táctil.
Respecto a los periféricos presentes en este proyecto, cabe destacar que se han escogido en función de las necesidades del prototipo, de la disponibilidad y de la economía.

- *Display*

Una pantalla táctil es un periférico de entrada y de salida de datos. Permite al usuario la entrada de datos o el control del sistema mediante el contacto con la superficie de la pantalla ya sea con un lápiz especial o con los dedos y, a su vez, muestra los resultados introducidos desde la placa de control. Por lo tanto, la pantalla táctil permite al usuario interactuar directamente con lo que se muestra en lugar de utilizar un ratón, además se puede configurar un teclado virtual de manera que no sea necesario un teclado físico.

La pantalla elegida en este proyecto es la que se muestra en la *Figura 34* y se trata de una pantalla táctil gráfica LCD tipo FTF de 7 pulgadas que ofrece una resolución de 800x480 con una tensión de alimentación de 3-5 VCC de la marca SHCHV.

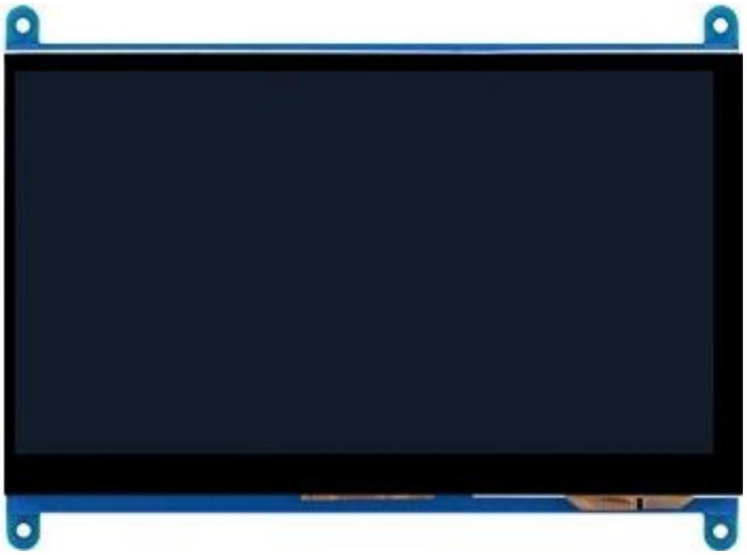


Figura 34. Pantalla táctil. Obtenida de la página web AliExpress

PRECIO: 44,70€

La función que desempeña el display es mostrar la interfaz, programada en la Raspberry Pi, y que el usuario pueda interactuar con los actuadores a través de ésta, así como obtener información sobre los parámetros recogidos por los sensores.
En resumen, mediante la pantalla táctil, el usuario puede observar tanto la temperatura como la humedad relativa, activar o desactivar cualquier actuador y periférico, configurar la temperatura

y humedad de consigna para las que se encenderán la manta eléctrica o el humidificador y vigilar al bebé en cualquier momento.

La misma interfaz se puede visualizar desde cualquier otro dispositivo mediante la herramienta VNC, como se explicará posteriormente.
Este display se ha dispuesto en el pie de cuna, para ofrecer la función de interacción e información desde la misma cuna.

En cuanto a la conexión del display con la Raspberry Pi, la pantalla dispone de dos conectores, un conector HDMI y un conector micro USB, el conector HDMI se utiliza para mostrar la imagen en la pantalla y se conecta con la entrada HDMI de la Raspberry Pi, mientras que el conector micro USB se utiliza para la función táctil y la transmisión de datos y éste se conecta a uno de los puertos USB de la Raspberry Pi, conforme miestra la *Figura 35*. Cabe destacar que para que funcione es necesario configurar los controladores correspondientes en la Raspberry Pi, dicha configuración se encuentra descrita en el apartado 6.1.3. Configuración pantalla táctil.

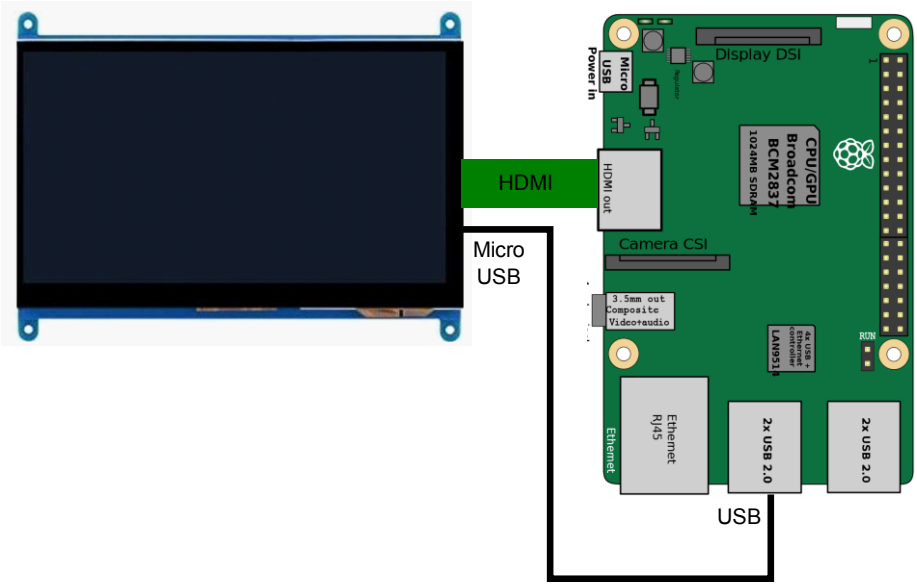


Figura 35. Conexión pantalla táctil

- *Cámara*

La cámara es un periférico de entrada, el cual recoge las imágenes con su objetivo, las almacena y envía a la placa de control, donde se reproducirán. En este prototipo permite al usuario observar al bebé cuando lo desee y por un tiempo ilimitado.

Se encuentra situada en el pie de la cuna, orientada a la base de ésta donde se encuentra el niño con el fin de ver si está llorando, si está durmiendo o distraído.

La cámara seleccionada en este proyecto es la mostrada en la *Figura 36* y se trata de una cámara de visión nocturna de la marca Waveshare, que viene junto con dos piezas de sensor IR luz LED que permiten visualizar la imagen emitida por la cámara en lugares oscuros.

Figura 36. Cámara de video. Obtenida de la página web Raspberry Pi

Esta cámara ofrece unas dimensiones de 25 mm x 24 mm x 6 mm con 2592x1944 píxeles, un campo de visión de 2,0x1,33 m a 2 m y una resolución de 1080p.
La tensión de alimentación que necesita es de 3,3 V, la cual es proporcionada por la Raspberry Pi, ya que este dispositivo es totalmente compatible con la placa de desarrollo y mediante un cable plano de 15 cm se realiza tanto la alimentación de la cámara como el intercambio de datos.

PRECIO: 12,20€

La función que desempeña la cámara es mostrar las imágenes captadas a través del display ubicado en la cuna o bien en cualquier dispositivo que desee conectarse mediante VNC.
La obtención de imágenes de la Raspberry desde la cámara, para su posterior reproducción, se ha programado en el código volcado en la placa de control, que se explicará posteriormente.

En cuanto a la conexión de la cámara con la Raspberry Pi, como se ha comentado anteriormente, la cámara que se ha utilizado está fabricada para su uso con Raspberry Pi, por lo que ésta dispone de un conector para esta función, como se observa en la *Figura 37*, se introduce en el conector de vídeo que está señalado en la placa con la palabra “CAMERA CSI”.

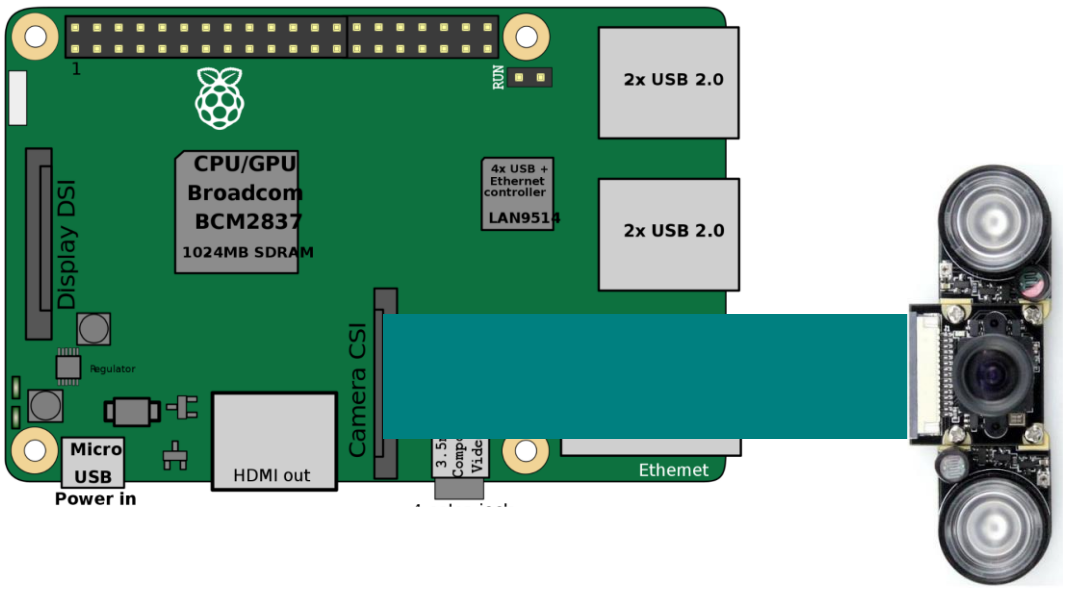


Figura 37. Conexión cámara

- *Altavoces*

Los altavoces son dispositivos que permiten la amplificación del sonido, son transductores eléctricos que convierten la corriente eléctrica en energía mecánica y ésta en ondas sonoras que se propagan por el aire, las cuales son percibidas por el cerebro como impulsos nerviosos y éste los interpreta.

En este proyecto, quien envía la información a los altavoces es la Raspberry Pi y éstos actúan como periférico de salida, ya que reproducen al exterior la información enviada por la palca de control.

Debido a que la Raspberry ofrece una salida de audio mediante conector Jack de 3,5 mm es posible la conexión de cualquier altavoz convencional. Los que se han incluido en el prototipo son los altavoces ZIPY 4WX4W RMS, Figura 38

Figura 38. Altavoces. Obteida de la página web Amazon

PRECIO: 11,95€

Se ha querido incluir la reproducción de audio en el prototipo mediante una lista de canciones infantiles en formato .wap entre las que se podrá escoger para su reproducción al exterior mediante los altavoces, que se encuentran anclados a la cuna.
La activación de la emisión de audio, así como la elección de la canción o el volumen de ésta, se realiza mediante la interfaz, durante el tiempo que se desee.

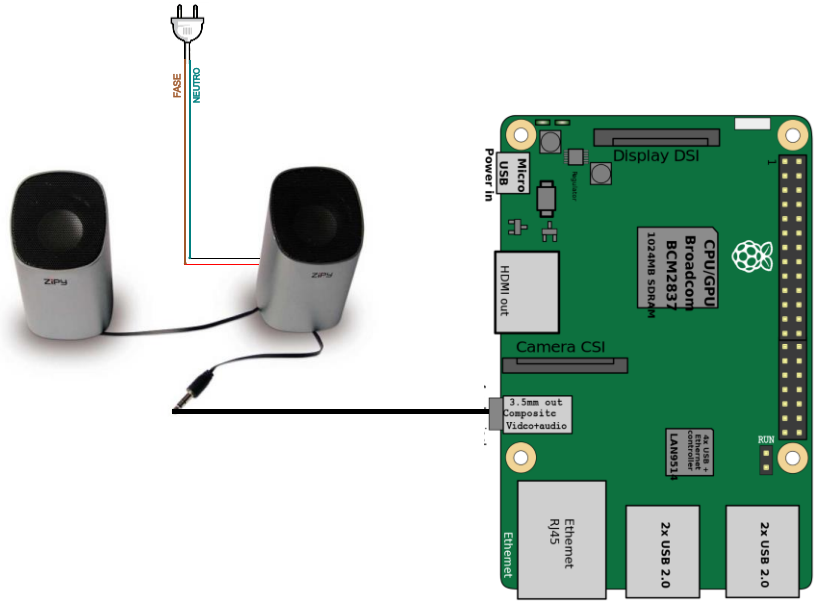


Figura 39. Conexión altavoces

El conector Jack macho de los altavoces se introduce en el conector hembra de la Raspberry, como muestra la *Figura 39*, mientras la alimentación de los altavoces se realiza conectando los cables fase y neutro de éstos con la fase y neutro del enchufe general de la cuna, respectivamente.

4. Tecnologías software

En este apartado se van a exponer los elementos *software* que se han utilizado para este proyecto, que tienen como placa de desarrollo la Raspberry Pi, como son el sistema operativo, el lenguaje de programación y el entorno de desarrollo utilizado. También se realiza una breve explicación de cómo se han implementado estas tecnologías en el proyecto.

4.1 Sistema operativo

El sistema operativo utilizado es Raspbian, que es una distribución del sistema operativo GNU/Linux basado en Debian; es el principal sistema operativo compatible con Raspberry Pi. Está optimizado para el *hardware* que presenta la Raspberry PI y contiene más de 35000 paquetes, es decir, *software* precompilado incluido en un formato en el que resulta fácil su instalación, que con el tiempo desarrolla nuevas versiones.

Figura 40. Sistema operativo Raspbian

4.2 Leguaje de programación

Java es un lenguaje de programación orientado a objetos creado en 1991 y publicado en 1995 por Sun Microsystem (adquirida por Oracle en 2010).

La principal razón por la que se ha escogido este lenguaje es por la necesidad de un lenguaje de alto nivel, que fuera capaz de crear una interfaz de usuario intuitiva y gobernar, mediante el código, todos los periféricos de los que se hacen uso en este proyecto.

Una vez tomada esta decisión, entre los lenguajes de programación de alto nivel se ha escogido Java por las siguientes características que ofrece:

Figura 41. Lenguaje de programación. Obtenida de la página web Java

- Está dentro de los lenguajes más usados en la actualidad y corre en casi todas las plataformas que hay en el mercado.
- Java es totalmente compatible con el sistema operativo utilizado.
- Existe gran soporte, documentación y comunidades de Java.
- Java también cuenta con una serie de librerías que amplían sus funcionalidades.
- Java no es un lenguaje complicado, ya que es un tipo de programación orientada a objetos, por lo que puede resultar intuitivo.
- Es seguro, ya que ofrece mucha seguridad frente a infiltraciones de terceros o virus.
- Java está diseñado para crear *software* robusto y fiable, para ello proporciona comprobaciones durante la compilación y en tiempo de ejecución.

4.3 Herramienta de desarrollo

El IDE en el que se ha desarrollado el código de programación es IntelliJ IDEA [17], que es un entorno de desarrollo integrado para el desarrollo de programas informáticos. Fue desarrollado por JetBrains, anteriormente conocido como IntelliJ.

IntelliJ IDEA ofrece un conjunto de herramientas como son el descompilador y el visor de bytecode. La funcionalidad de este IDE se amplía continuamente por usuarios y terceros a través de plugins. IntelliJ IDEA ofrece soporte para Java EE, Spring, Hibernate y otras tecnologías.

Este IDE analiza el código, buscando conexiones entre símbolos de entre todos los archivos del proyecto e idiomas. Utilizando esta información proporciona ayuda de codificación, navegación rápida y análisis inteligente de errores.

Figura 42. Herramienta de Desarrollo. Obtenida de la página web IntelliJ

La elección de este IDE radica en que la plataforma creada por Jetbrains reúne cantidad de IDEs con el mismo entorno para diferentes lenguajes de programación, de formal que, al conocer el funcionamiento de este IDE para Java, se conoce la metodología de funcionamiento de los demás IDEs, en caso de querer programar con otro lenguaje.

Una vez se ha presentado el IDE utilizado en este proyecto, cabe destacar la organización que se ha constituido para albergar toda la documentación y archivos de necesidad.

Una vez creado el proyecto de IntelliJ llamado IntelliCrib, el entorno nos ofrece tres prestañas; proyecto, estructura y favoritos.

En la pestaña de proyecto se encuentra todos los archivos necesarios para la compilación del código, así como el código de programación.

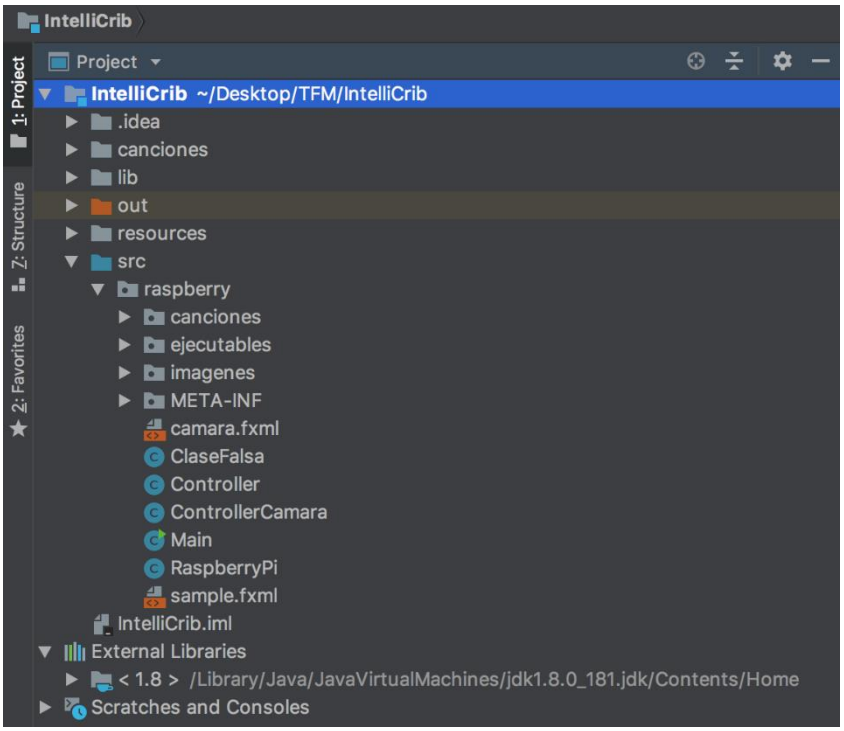


Figura 43. Jerarquía IDE

La carpeta raspberry, dentro de recursos (src), es la que contiene todo el peso del proyecto, por ello se ha dividido en varias subcarpetas.
En primer lugar, la carpeta de canciones alberga los tres archivos .wav que representan las tres nanas que se podrán escuchar en la cuna.
La carpeta de ejecutables guarda archivos en código Python, debido a que la lectura de algunos sensores necesita de éste código, llamado desde el código Java, para poder ser consultados y que ofrezcan la respuesta de estos sensores.
En la carpeta imágenes se encuentran las imágenes que son utilizadas para formar la interfaz de comunicación con el usuario.
Por último, la carpeta META-INF guarda la información que necesita el archivo .jar para construirse, este archivo es el que ejecuta la Raspberry. Dentro de META-INF se encuentran los ficheros .fxml, que son los utilizados por la vista para dar presencia a la interfaz gráfica, y los ficheros de código de programación divididos en:

- ClaseFalsa: Clase en la que se simulan de forma aleatoria datos falsos sobre los sensores para probar el funcionamiento de la programación sin tener que estar recibiendo datos reales del exterior.
- Controller y ControllerCamara: Código de programación que hace referencia al controlador, que se explica en el siguiente apartado, tanto para el control del proyecto en general como para el control de la cámara de vigilancia, respectivamente.
- Main: Código que se ejecuta en primer lugar
- RaspberryPi: Clase dedicada para que código que constituye el modelo, explicado en el apartado posterior.

4.4 Librerías

Se ha utilizado la librería Pi4J (Pi for Java) [18], que es un proyecto de código abierto que tiene como objetivo proporcionar una interfaz de programación de aplicaciones de entradas y salidas orientada a objetos y ofrece bibliotecas de implementación para los programadores de Java para acceder a todas las capacidades de E/S de la plataforma Raspberry Pi, con ella se permite interactuar con los pines GPIO (E/S de propósito general) de la Raspbery Pi 3B.

Para hacer uso de Pi4J se descarga el fichero comprimido que ofrece el proyecto Pi4J y se descomprime, el cual contiene las librerías que pueden verse en la *Figura 44*.

Entre estas, de las que se va a hacer uso en este proyecto son:

- pi4j-core-javadoc.jar: Ofrece la documentación mediante comentarios a lo largo del código, los cuales explican el funcionamiento de determinada librería.

- pi4j-core.jar: Código que se ejecuta con extensión .class, el cual representa un conjunto de instrucciones que entiende la máquina virtual de java. Este formato puede entenderse como una traducción del código .java a lenguaje de bajo nivel que entiende la máquina, pero no sería fácilmente comprensible por el desarrollador.

- pi4j-core-sources.jar: Mismo código que el que contiene la librería core pero con extensión .java, el cual es entendido por el desarrollador. Este código es sólo informativo, no se ejecuta.

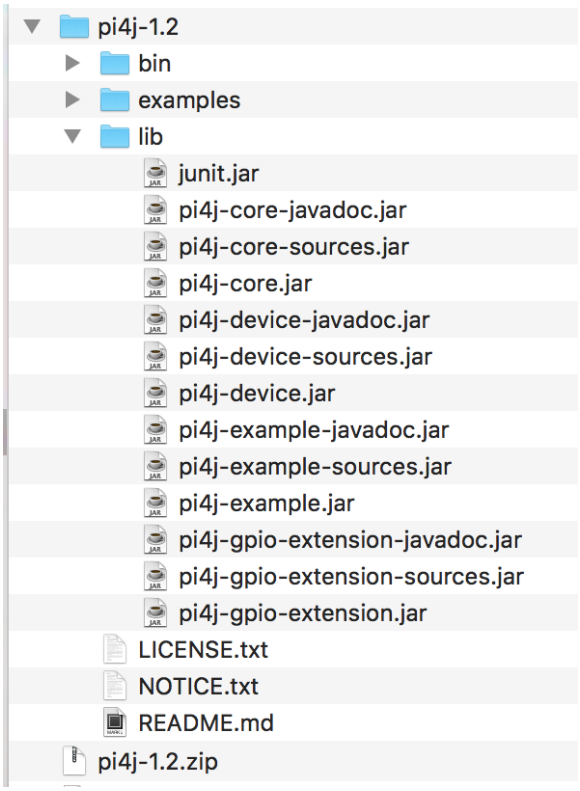


Figura 44. Librerías Pi4J

En primer lugar, se añaden estos tres ficheros .jar al directorio lib del proyecto en curso; arrastrándolos a la carpeta lib, *Figura 45*.

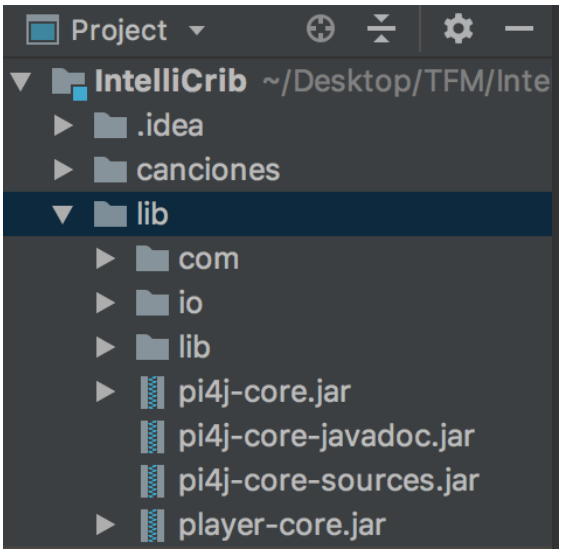


Figura 45. Inclusión de librerías en el proyecto

En segundo lugar, debe configurarse IntelliJ para que encuentre estas librerías que se han añadido y que haga uso de ellas. Para ello se accede a la página de ajustes del proyecto, *Figura 46.*

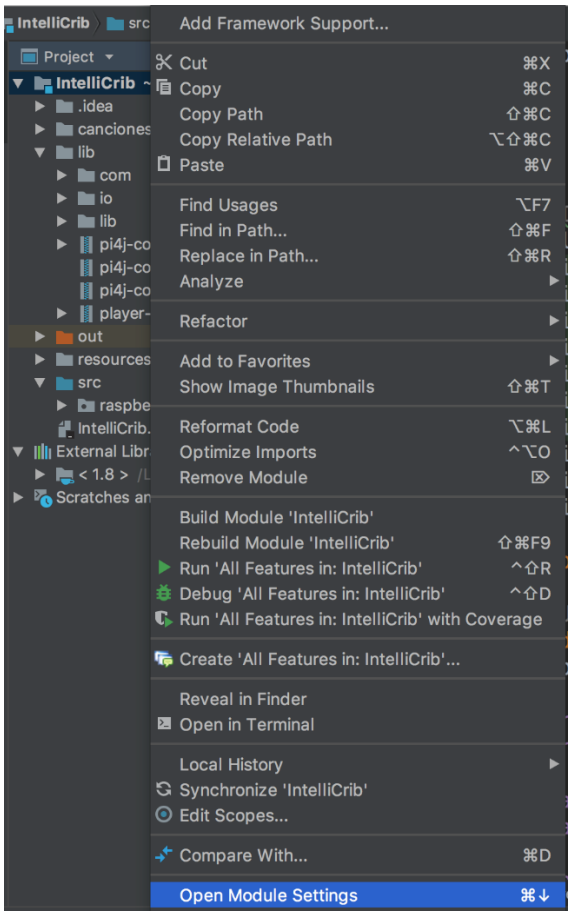


Figura 46. Acceso Ajustes IntelliJ

Y, a continuación, se añaden las tres librerías de Java vistas previamente, *Figura 47 y 48,* para que IntelliJ las reconozca.

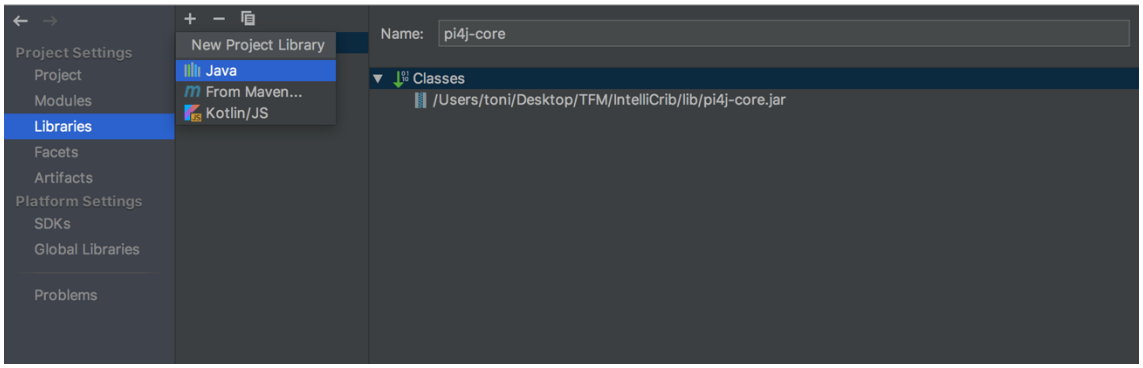


Figura 47. Inclusión librerías Java en el IDE

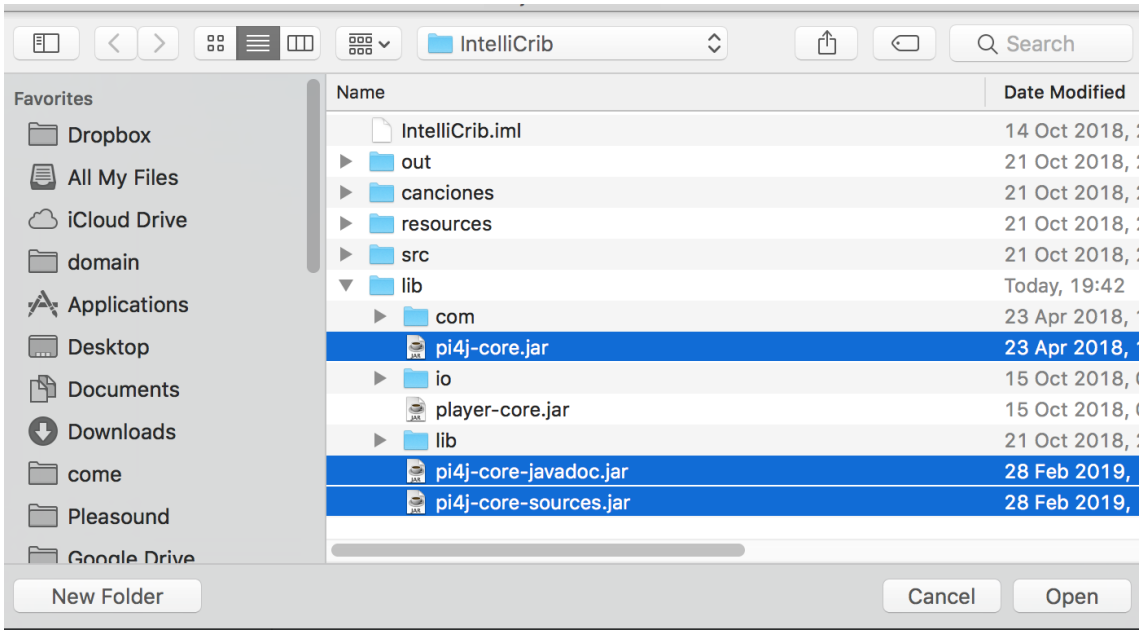


Figura 48. Búsqueda de librerías en el PC

5. Desarrollo de la solución

Conocidas las tecnologías de las que se va a hacer uso tanto para el bloque de *software* como para e de *hardware*, se procede a exponer cómo se han implementado en este proyecto. En primer lugar, se va a detallar el desarrollo *software* explicando la programación volcada en la placa de desarrollo, que ofrece el control de cada sensor y actuador, separado por apartados según su naturaleza. En segundo lugar, se expone desarrollo *hardware* con la presentación del conexionado de cada uno de los materiales de campo.

5.1 Desarrollo software

En este apartado se va a realizar una explicación sobre el patrón de arquitectura *software* utilizado y para cada componente de éste se detallará el código desarrollado para este proyecto.

El desarrollo del *software* se ha basado en el patrón de arquitectura Modelo-vista-controlador (MVC) [19], el cual tiene el propósito de partir el código en varias partes; una parte se encarga de los datos de la aplicación, otra parte se ocupa de la interfaz con la que interactúa el usuario y la otra parte de código controla cómo funciona la aplicación.

MVC es una forma de organizar las funciones principales del código para que quede más clara y más accesible, ya que permite separar los componentes de la aplicación dependiendo de la responsabilidad que tienen, lo que significa que al efectuar un cambio en alguna parte del código no afecte otra parte del mismo. Por ejemplo, si se modificara la base de datos, sólo deberíamos modificar el modelo que es quién se encarga de los datos y el resto de la aplicación debería permanecer intacta.

Un ejemplo de esta arquitectura para este proyecto sería el siguiente:

1. El usuario realiza una acción en la interfaz: Pulsa el botón de encender el proyector desde la interfaz gráfica.
2. El controlador trata el evento de entrada que previamente se ha registrado: Vincula la pulsación sobre el icono del proyector con la etiqueta declarada en el código "RadioButtonProyector".

```
@FXML
RadioButton RadioButtonProyector;
```

3. El controlador notifica al modelo (llamado raspberryPi) la acción del usuario, lo que puede implicar un cambio del estado del modelo: Si el botón del proyector ha sido pulsado desde la interfaz, el controlador envía la orden al modelo de activar el proyector.

```
@FXML
public void radioButtonProyectorPulsado() {
  if (RadioButtonProyector.isSelected()) {
    System.out.println("Activar Proyector");
    raspberryPi.activarProyector();
  } else {
    System.out.println("Desactivar Proyector");
    raspberryPi.desactivarProyector();
  }
}
```

4. El modelo ejecuta la orden del controlador: Lleva a cabo la ejecución de mandar 5V al pin de la raspberry pi donde está conectado el proyector.

```
 public void activarProyector() {
  pinProyector.low();
```

```
        System.out.println("Activamos proyector ");
    }
```

5. La interfaz de usuario (la vista) espera otra interacción del usuario, que comenzará otro nuevo ciclo.

5.1.1 Modelo

El modelo es el contacto más directo con el *hardware* de la Raspberry Pi, por lo tanto, gestiona todos los accesos de información de los pines provenientes de sensores y envía las órdenes a los pines que gobiernan los actuadores. Por tanto, también se encarga de enviar a la vista la información que en cada momento se le solicita para que sea mostrada. En este proyecto al modelo se le ha llamado "raspberryPi" y reúne todas las funciones para interactuar con los pines de la Raspberry.

Las peticiones de acceso o manipulación de información llegan al modelo a través del controlador.

En el código recogido en el modelo, en primer lugar, se hace referencia a la carpeta que recoge el código referido al modelo, llamado raspberry, después se importan las librerías que se han descargado anteriormente y las que ofrece java que vayan a ser necesarias a lo largo del proyecto.

Entonces se da paso a la clase RaspberryPI, la cual alberga todo el código de programación referido al modelo.

```
Package raspberry;

import com.pi4j.io.gpio.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;

public class RaspberryPi {
```

Dentro de esta clase, en primer lugar, deben declararse los pines de la Raspberry, asociándolos a una variable que se utilizará durante el código.

```
//DECLARACIÓN DE LOS PINES DE LA RASPBERRY

GpioController gpio; //Se crea una variable llamada gpio de tipo GpioController (que es el grupo de gpios, de pines)
```

```
GpioPinDigitalOutput pinMantaElectrica; //Se declara una variable llamada pin mantaelectrica, que es de tipo gpio de salida

GpioPinDigitalOutput pinActivarMotorDerechoArriba;
GpioPinDigitalOutput pinActivarMotorDerechoBajo;
GpioPinDigitalOutput pinActivarMotorIzquierdoArriba;
GpioPinDigitalOutput pinActivarMotorIzquierdoBajo;
GpioPinDigitalOutput pinHumidificador;
GpioPinDigitalOutput pinProyector;
GpioPinDigitalInput pinTemperaturayHumedadRelativa;
GpioPinDigitalInput pinHumedadAbsoluta;
GpioPinDigitalInput pinCalidadAire;

private boolean esBalanceoActivado;

//MÉTODO QUE INICIALIZA TODOS LOS PINES, DEFINIENDO EL NÚMERO DE PIN Y SI ES DE ENTRADA O SALIDA

public void inicializar() {
  gpio = GpioFactory.getInstance(); //Se crea la instancia del controlador GPIO --> COGER TODAS LAS GPIOS DE LA RASPBERRY

  pinTemperaturayHumedadRelativa = gpio.provisionDigitalInputPin(RaspiPin.GPIO_02, "leer temperatura y hum relativa"); //Lee sensor dht11-entrada digital. Se indica que el pin 2 de las GPIOs de la Raspberry Pin va a ser una entrada digital.
  pinTemperaturayHumedadRelativa.setShutdownOptions(true, PinState.LOW); //Cuando se apague la app que el pin se quede en estado bajo

  pinHumedadAbsoluta = gpio.provisionDigitalInputPin(RaspiPin.GPIO_22, "leer hum absoluta"); //Lee sensor de humedad absoluta-entrada digital
  pinHumedadAbsoluta.setShutdownOptions(true, PinState.LOW);

  pinCalidadAire = gpio.provisionDigitalInputPin(RaspiPin.GPIO_13, "leer calidad de aire"); //Lee sensor calidad de aire-entrada digital
  pinCalidadAire.setShutdownOptions(true, PinState.LOW);

  pinMantaElectrica = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_07, "activar manta electrica", PinState.HIGH); //Se indica que el pin 7 va a ser una salida digital y que cuando se inicie debe hacerlo en estado ALTO. Se fija en estado alto porque la placa de relés trabaja de forma inversa; por lo que, al encenderse la Raspberry Pi, la salida del
```

```
pin 16 estará apagada; se manda un 0.
  pinMantaElectrica.setShutdownOptions(true,PinState.HIGH);
//Cuando se apague la app, los actuadores deben quedarse en estado alto, ya que los relés actúan de forma inversa

  pinActivarMotorDerechoArriba = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_15, "motor derecho arriba", PinState.HIGH);
  pinActivarMotorDerechoBajo = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_16, "motor derecho bajo", PinState.HIGH);
 pinActivarMotorDerechoArriba.setShutdownOptions(true, PinState.HIGH);//El pin 15 se desactivará si se cierra la app.
  pinActivarMotorDerechoBajo.setShutdownOptions(true, PinState.HIGH);//El pin 16 se desactivará si se cierra la app.

  pinActivarMotorIzquierdoArriba = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_11, "motor derecho arriba", PinState.HIGH);
  pinActivarMotorIzquierdoBajo = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_10, "motor derecho bajo", PinState.HIGH);
  pinActivarMotorIzquierdoArriba.setShutdownOptions(true, PinState.HIGH);
  pinActivarMotorIzquierdoBajo.setShutdownOptions(true, PinState.HIGH);

  pinHumidificador = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_08, "activar humidificador", PinState.HIGH);
  pinHumidificador.setShutdownOptions(true, PinState.HIGH);

  pinProyector = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_09, "activar proyector", PinState.HIGH);
  pinProyector.setShutdownOptions(true, PinState.HIGH);
}
```

Una vez identificados los pines de la Raspberry con sus respectivos sensores y actuadores, se presenta el código para la lectura de los diferentes sensores.

Para el caso de la lectura del sensor DHT11 (temperatura y humedad relativa) ha sido necesaria la utilización de unos ejecutables escritos en código Python, y llamados desde éste código Java, para poder obtener los valores sensados por el DHT11.

```
//MÉTODO PARA OBTENER EL VALOR DE TEMP Y H. RELATIVA DEL DHT11

  int[] obtenerTemperaturayHumedad() throws IOException { //Método que devuelve una cadena de dos enteros
```

```
    int[] humTemp = {0, 0}; //Variable que tiene un array de enteros, en este caso dos, el valor de la humedad y de la temperatura
    Runtime rt = Runtime.getRuntime(); //Te devuelve el sistema operativo sobre el que se ejecuta el programa, en este caso Raspbian

    File archivo = new File("ejecutables/dht11_example.py"); //Cogemos el fichero que se encuentra en esta ruta y se guarda en la variable “archivo”, que es de tipo file.

   Process procesoEjecutaArchivo = rt.exec("sudo python " + archivo.getAbsolutePath());
//Con Process ProcesoEjecutaArchivo ejecuto el comando sudo python + el fichero en el terminal; para ejecutar el archivo igual que hacemos con java .jar

  BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(procesoEjecutaArchivo.getInputStream())); //Lee la respuesta del ejecutable: (Humidity: x%, Temperature: xº) y lo guarda en un buffer llamado bufferedReader.

    // HUMEDAD
    String lineaHumedad = bufferedReader.readLine();
    int humedad;
    //El comando .readLine lee la primera línea del buffer, que es la información sobre la humedad e introduzco el valor en la variable lineaHumedad.
    if (lineaHumedad != null) {
      if (lineaHumedad.contains("Humidity")) {
//Si el mensaje no está vacío y contiene la parabra Temperature entra en el buble if; para asegurarnos de que estamos cogiendo la humedad en esta línea, ya que el ejecutable debe devolver el mensaje Humidity:50%, por ejemplo, en esta línea.

          lineaHumedad = lineaHumedad.replace("Humidity:", "").replace("%", "");
//Reemplazamos Humidity: y % por un vacío, con la finalidad de quedarnos sólo con el numero (50, por ejemplo).

        humedad = Integer.parseInt(lineaHumedad);
//Pasa el string que lee de la línea a un entero, ya que queda solo el número.

        humTemp[0] = humedad;
//Introduzco el int de la humedad en el primer espacio del array declarado en el inicio.

        System.out.println("Humedad es : " + humedad + " %RH");
      } else {
        System.out.println("Error en la lectura de la humedad");
```

```
//Si el ejecutable no ha devuelto un mensaje tipo Humidity: 50% en la línea del buffer
que corresponde, se deduce que ha habido un error en la lectura del sensado y así lo
comunica.
     }
   }

   // TEMPERATURA
   String lineaTemp = bufferedReader.readLine();
   int temperatura;

   if (lineaTemp != null) {
     if (lineaTemp.contains("Temperature")) {
//Si el mensaje no está vacío y contiene la parabra Temperature entra en el bucle if.

       // El ejecutable devuelve un mensaje tipo Temperature:20C en la línea del buffer
correspondiente.

       lineaTemp = lineaTemp.replace("Temperature:", "").replace("C", "");

       temperatura = Integer.parseInt(lineaTemp);

       humTemp[1] = temperatura; //Se introduce el int de la temperatura en el segundo
espacio del array

            System.out.println("Temperatura es : " + temperatura +    "C");
             } else {
              System.out.println("Error en la lectura de la temperatura");
             }
            }

            bufferedReader.close(); //Cierro el buffer de lectura; dejo de leer.

            try {
              procesoEjecutaArchivo.waitFor(); //Espera a que el proceso de ejecutar el
          archivo (sudo python) finalice.
            } catch (InterruptedException e) {
              // TODO Bloque catch generado automáticamente
              e.printStackTrace();
            }

            return humTemp; //Un vector de dos puntos devuelve los dos valores;
```

```
humedad relativa y temperatura.
 }
```

Como se ha explicado anteriormente, la detección de que el niño se haya meado y la orina haya desbordado del pañal o que la cuna se haya mojado con algún líquido se realiza mediante un sensor de humedad absoluta, que deja pasar la corriente entre los dos espadines del sensor, por conducción mediante el agua, de forma que pasa a través del sensor los 5V con los que se alimenta y éstos llegan al pin correspondiente de la Raspberry.

```
 public boolean seHaMeado() {
  //Método booleano que devuelve True si el pin 22 está recibiendo 5V (el
sensor está conduciendo; presencia liquido)

  return pinHumedadAbsoluta.isLow(); //Si el pin 22 (pinHumedadAbsoluta) está
en estado Low, es decir, está recibiendo 5V (YA QUE ACTUA DE FORMA
INVERSA), retornamos un true como respuesta del método seHaMeado.
Si no hubiera conductividad, llegarian 0V y el pin 22 estaría en HIGH, por lo que
pinHumedadAbsoluta.isLow() seria falso y se devolvería un FALSE como
respuesta del método
 }
```

Para el caso de la detección de heces, el sensor utilizado es el de calidad de aire, el cual deja pasar los 5V, con los que se alimenta en caso de detectar gas metano.

```
 public boolean seHaCagado() {
  //Método booleano que devuelve True si el pin 13 está recibiendo 5V (el sensor
está conduciendo, presencia liquido)

  return pinCalidadAire.isLow(); //Si el pin 13 (pinCalidadAire) es Low, es decir,
está recibiendo 5V (YA QUE ACTUA DE FORMA INVERSA), retornamos un true
como respuesta del método. Si no hubiera conductividad, llegarian 0V y el pin 13
estaría en HIGH, por lo que pinHumedadAbsoluta.isLow() sería falso y se
devolvería un FALSE como respuesta del método
 }
```

En el caso de los actuadores, la orden de activación de realiza desde el controlador, pero es el modelo quien ejecuta la acción.
Por ello en este apartado se exponen los métodos para activar cada uno de los actuadores, los cuales serán llamados desde el controlador cuando se precise.

La manta eléctrica se activa mediante la interfaz de usuario, la cual tiene la opción de encenderla o apagarla de forma directa o bien imponer un punto de consigna de temperatura a la que desee que esté la cuna, modo en el que entra en juego la respuesta del sensor de temperatura DHT11. En este caso, el modelo solo se encarga de ordenar al pin determinado la acción a realizar; activarse o desactivarse.

```
public void activarMantaElectrica() {

  pinMantaElectrica.low();
  System.out.println("Activamos manta electrica");
}

public void desactivarMantaElectrica() {
  pinMantaElectrica.high();
  System.out.println("Desactivamos manta electrica");
}
```

El balanceo de la cuna se consigue mediante dos motores lineales los cuales tienen un pin de la Raspberry asignado a cada movimiento; motor derecho arriba, motor derecho abajo, motor izquierdo arriba, motor izquierdo abajo.

Para el balanceo se han creado tres métodos; balanceo1 (motor izquierdo arriba y derecho abajo), balanceo 2 (motor derecho arriba e izquierdo abajo) y reposoBalanceo (ambos motores arriba).

```
public void activarBalanceo() {
  esBalanceoActivado = true; //Variable que se utiliza para decir que el botón de la interfaz de usuario está pulsado.

  Thread hiloBalanceo = new Thread(new Runnable() { // Se crea un hilo secundario para que cuando se meta en el bucle while no pare de balancear, mientras esté pulsado el botón, pudiendo responder al resto de acciones paralelamente.

    @Override //Quiere decir que el método Run ya existe y estás sobreescribiendo sobre este.
    public void run() { //2º plano; NO MODIFICAR NI ACTUAR SOBRE LA INTERFAZ EN EL SEGUNDO PLANO, en caso de querer hacerlo deberíamos volver al primer plano.
      reposoBalanceo(); //Se parte del reposo
```

```
        while (esBalanceoActivado) {
          System.out.println("Empieza balanceo");

          balanceo2(); //Derecho arriba e izquierdo abajo con espera de 8 segundos
(tiempo para que se efectúe el movimiento)

          balanceo1();  //Izquierdo arriba y derecho abajo con espera de 8 segundos
(tiempo para que se efectúe el movimiento)
        }
        System.out.println("Termina balanceo");
        reposoBalanceo();
      }
    });
    hiloBalanceo.start();
  }

  public void desactivarBalanceo() { //Se llama a este método aquí cuando el
usuario despulsa el botón de balanceo

    esBalanceoActivado = false; //En este momento parará el bucle while llamado
“activarBalanceo”
  }

//MÉTODO PARA DEJAR EN REPOSO AMBOS MOTORES LINEALES
  private void reposoBalanceo() {
    System.out.println("REPOSO BALANCEO - LOS DOS ABAJO");
    pinActivarMotorIzquierdoArriba.low();
    pinActivarMotorDerechoArriba.low();

    esperarMotor(true); //True porque en el caso del reposo queremos esperar el
tiempo para que ascienda el motor se encuentre abajo aunque el usuario
desactive el balanceo.

    System.out.println("REPOSO BALANCEO DESACTIVADO - LOS DOS
ARRIBA");
    pinActivarMotorIzquierdoArriba.high();
    pinActivarMotorDerechoArriba.high();
  }
```

```
//MÉTODO PARA EJECUTAR BALNCEO 1 (DER. ABAJO – IZQ. ARRIBA)
  private void balanceo1() {
    System.out.println("balanceo 1 - der bajo izq arriba");

    pinActivarMotorIzquierdoArriba.low(); //Se activa la salida 11, por la que saldrán 5V que hacen cambiar el relé desde NC a NA dejando pasar la alimentación de 12V al motor derecho (low es activar).

    pinActivarMotorDerechoBajo.low(); // Se activa la salida 16, por la que saldrán 5V que hacen cambiar el relé desde NC a NA dejando pasar la alimentación de 12V al motor derecho (low es activar)

    esperarMotor(false); //False porque queremos que pare en caso que el usuario desactive balanceo

//Se desactiva el balanceo 1 al pasar el tiempo de subida y bajada de los motores.
    System.out.println("balanceo 1 DESACTIVADO - der bajo izq arriba");

    pinActivarMotorIzquierdoArriba.high();
    pinActivarMotorDerechoBajo.high();
//Dejan de emitirse 5V hacia los pines que activan el motor izquierdo hacia arriba y el motor derecho hacia abajo

  }
//MÉTODO PARA EJECUTAR BALNCEO 1 (IZQ. ABAJO – DER. ARRIBA)
  private void balanceo2() {

    System.out.println("balanceo 2 - der arriba izq bajo");

    pinActivarMotorDerechoArriba.low();
    pinActivarMotorIzquierdoBajo.low();

    esperarMotor(false); //Para cortar la espera y, por tanto, el movimiento en caso de que el usuario lo desee.
Aunque en este método no haría falta, ya que la variable esperarMotor se mantiene en false desde el método balanceo1, el cual siempre se ejecutará antes que éste.

    System.out.println("balanceo 2 DESACTIVADO - der arriba izq bajo");
```

```
    pinActivarMotorDerechoArriba.high();
    pinActivarMotorIzquierdoBajo.high();
//Se desactiva la emisión de 5V por el pin correspondiente al motor derecho de subida y al motor izquierdo de bajada una vez pase el tiempo de movimiento.
  }
```

El tiempo de espera estimado es de 8 segundos. Dicho tiempo ha sido calculado en función de la velocidad que ofrecen estos motores lineales y la longitud final respecto de la inicial, es decir, la longitud de extensión.

```
//MÉTODO PARA EJECUTAR LA ESPERA DE ASCENSO O DESCENSO DE LOS MOTORES
  private void esperarMotor(boolean forzarEsperar) {
    //esBalanceoActivado será true cuando se esté pulsado el botón de balanceo. Pero hay sólo un caso en el que queremos que se meta en este bucle y espere, aunque el botón esté desactivado, que es REPOSO (queremos dejar los dos motores arriba en el momento que el usuario quiera parar el balanceo)
    for (int i = 0; i < 16 && (esBalanceoActivado==true || forzarEsperar==true); i++) {
      try {
        Thread.sleep(500); //500ms*16 = 8 SEGUNDOS, para que suba o baje el motor.
      } catch (InterruptedException e) { //Como el método sleep puede lanzar alguna excepción ponemos la función catch, para capturar la excepción y que nos muestre el error si se podrujera.
        e.printStackTrace();
      }
    }
  }
```

El humidificador se activa mediante la interfaz de usuario, la cual, al igual que la manta eléctrica, tiene la opción de encenderlo o apagarlo de forma directa o bien elegir un punto de consigna con la humedad a la que desee que esté la habitación, modo en el que entra en juego la respuesta del sensor de humedad relativa DHT11. En el modelo únicamente se recoge la orden de activación o desactivación desde el controlador y se traslada a los pines de la Raspberry Pi.

```
  public void activarHumidificador() {
    pinHumidificador.low(); //Se activa la salida 6 (escribimos low por ser relé inverso)
    System.out.println("Activamos humidificador "); //Para mostrar por pantalla que hemos activado el pin 6
  }
```

```
  public void desactivarHumidificador() {
    pinHumidificador.high();//desactivamos (relé inverso) la salida que nos han
pasado por argumento
    System.out.println("Desactivamos humidificador");
  }
```

El proyector también se activa mediante la interfaz de usuario, pero al igual que el balanceo únicamente se da la opción de encenderlo, pulsando el botón referido a la imagen del proyector, o apagarlo, despulsándolo.

```
  public void activarProyector() {
    pinProyector.low();
    System.out.println("Activamos proyector ");
  }

  public void desactivarProyector() {
    pinProyector.high(); //Se desactiva (relé inverso) la salida pinProyector
    System.out.println("Desactivamos proyector");
  }
}
```

5.1.2 Vista

La vista presenta la información del modelo en un formato adecuado para interactuar (interfaz de usuario), por tanto requiere de dicho 'modelo' la información que debe representar como salida, a través del controlador.
En este proyecto se ha hecho uso de la herramienta JavaFX Scene Builder para realizar la interfaz que se verá en la pantalla de la cuna y en la aplicación móvil, para que el usuario pueda interactuar con las funciones de la cuna. JavaFX Scene Builder [20] es una herramienta de diseño visual que permite diseñar interfaces de usuario de aplicaciones JavaFX donde se pueden arrastrar y soltar los componentes de la interfaz a un área de trabajo, modificar sus propiedades, aplicar hojas de estilo y el código FXML para el diseño que están creando se genera automáticamente en el fondo. El resultado es un archivo FXML, que se incluye en la carpeta de los ficheros de código, combinándolo con el proyecto Java, vinculando la interfaz de usuario a la lógica de la aplicación.
Como puede observase en la *Figura 49*, con la herramienta JavaFX Scene Builder se han generado dos ficheros; sample.fxml, que es la pantalla principal, y camara.fxml, que es la pantalla secundaria a la que se emerge en caso de querer ver las imágenes captadas por la cámara.

Deben declararse todos los elementos de la interfaz en la clase Controller; mediante @FXML se indica que es un elemento de la interfaz gráfica y, posteriormente, el tipo de elemento ya sea una

etiqueta (label) o una imagen, seguido del nombre de la variable que queramos darle para poder referirnos a ella a lo largo del código.

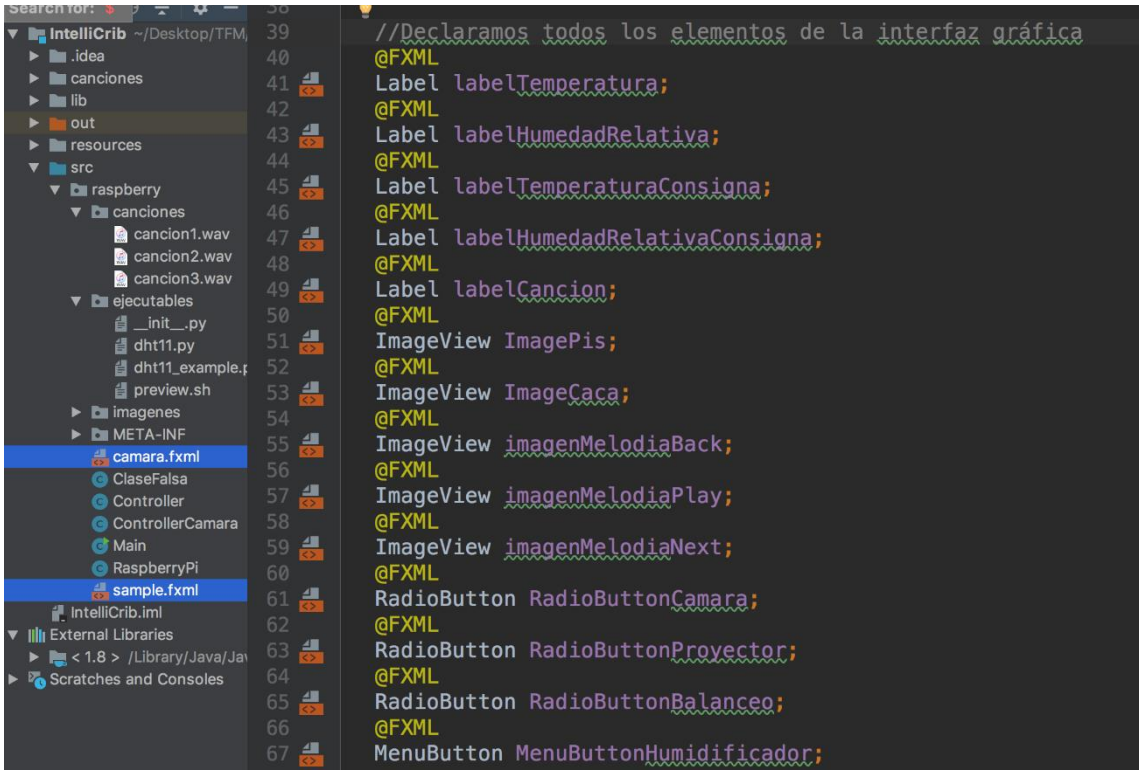


Figura 49. Declaración de elementos de la interfaz en el controlador

JavaFX Scene Builder ofrece un área de trabajo basada en un Panel, con unas dimensiones escogidas por el usuario en función de la pantalla donde será mostrada esta interfaz.
A la parte izquierda de esta herramienta se encuentra la librería y la documentación, *Figura 50*.

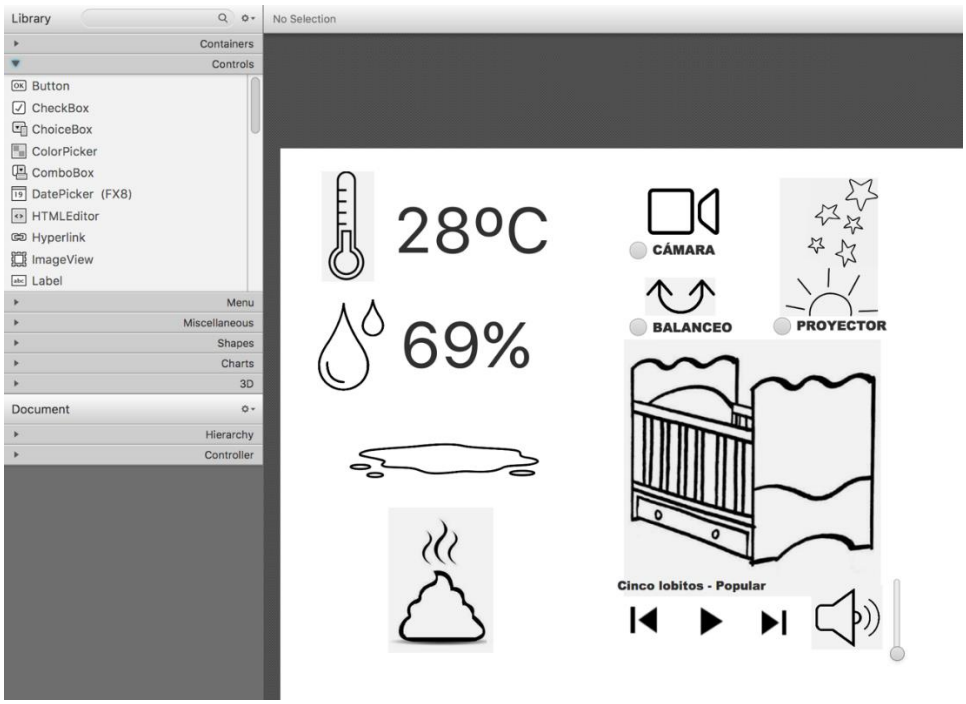


Figura 50. JavaFX Scene Builder - Librerías

En la librería aparecen todos los elementos que pueden utilizarse para realizar la interfaz; separados en controles, menús, formas, gráficos, incluso figuras en 3D. Desde esta pestaña se arrastran los elementos al panel o área de trabajo. En caso de querer ubicar un icono con una imagen guardada en el ordenador, debe arrastrarse el elemento ImageView al área de trabajo y posteriormente cargar encima la imagen que se desee.
Los elementos que se han utilizado para este proyecto han sido etiquetas, imágenes, *sliders* (controles deslizantes), menús de opciones y botones.

También a la izquierda de la herramienta se encuentra el apartado Documentación, que alberga los apartados Jerarquía y Controlador, *Figura 51*.

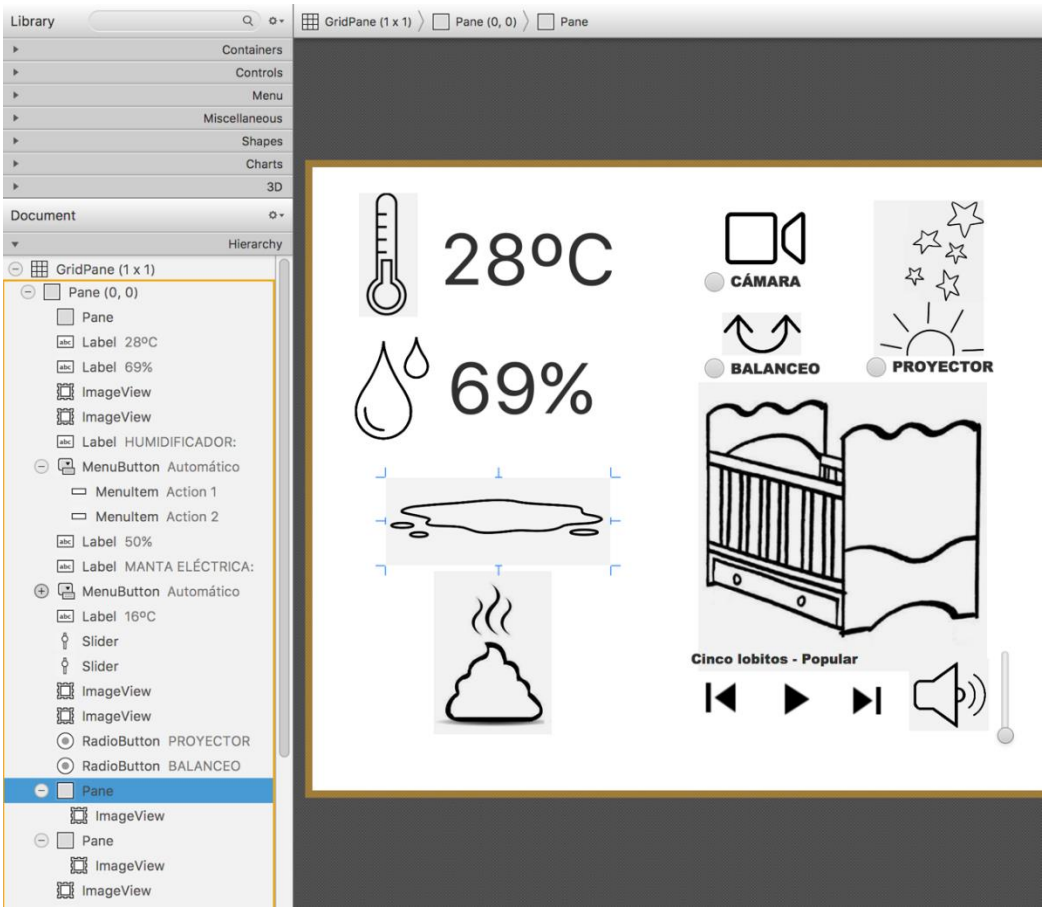


Figura 51. JavaFX Scene Builder - Documentación

La jerarquía muestra todos los elementos de los que se han hecho uso y a qué corresponden; en este caso, todos los elementos se encuentran dentro de Pane, que es el área de trabajo con dimensiones que posteriormente se verá en la pantalla. También puede observarse que el desplegable "MenuButton" alberga dos acciones, las cuales son las opciones del menú para seleccionar el modo manual o automático.

Todos estos elementos deben caracterizarse para poder vincularlos con el código de programación, esto se realiza con la pestaña situada a la derecha de la *Figura 52*, llamada "Inspector", que alberga tras apartados; propiedades, diseño y código.

En el apartado de propiedades se caracteriza al elemento, por ejemplo, el control deslizante puede caracterizarse con las siguientes propiedades:

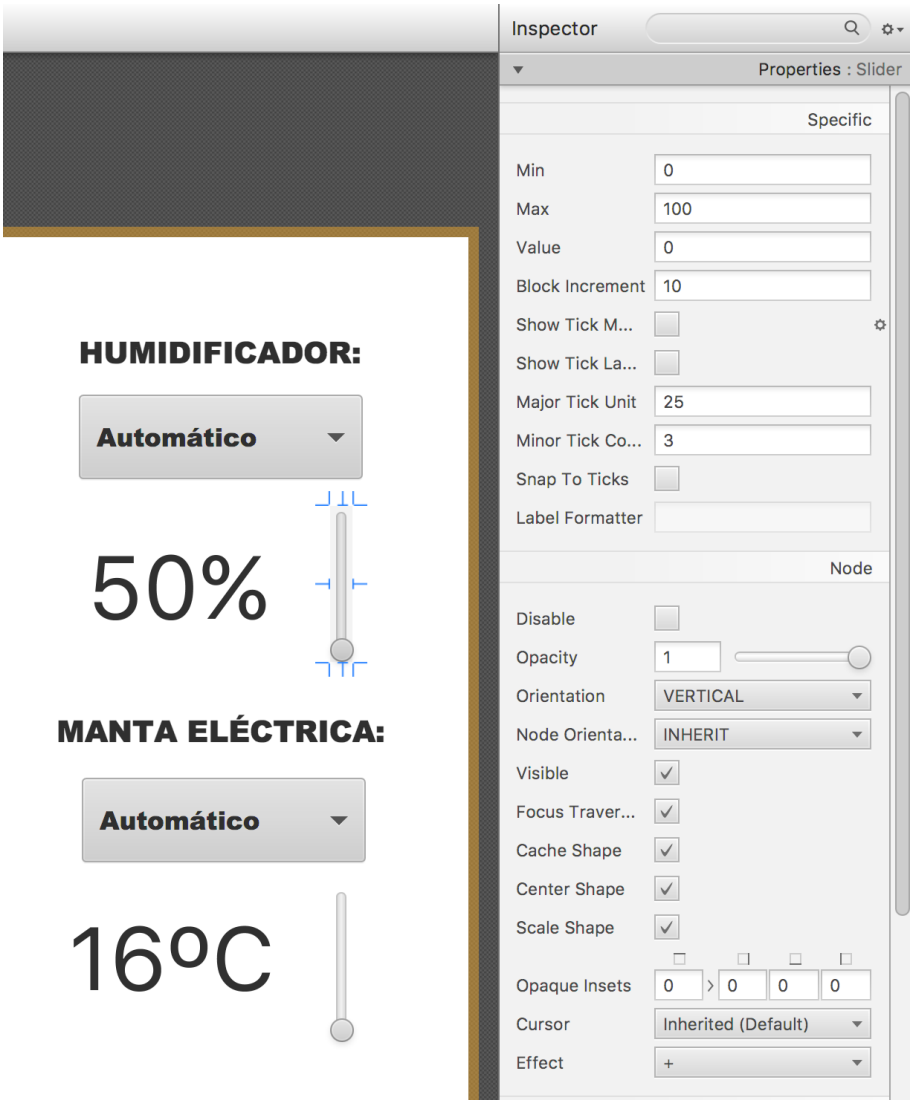


Figura 52. Caracterización de elementos de la interfaz

El valor mínimo y máximo; el *slider* devolverá valores entre 0 y 100; 0 cuando se encuentre el control en el punto inferior y 100 cuando se encuentre en el punto superior.
La unidad de distancia entre las principales marcas de graduación se indica en el apartado "Major Tick Unit". Por ejemplo, si el mínimo es 0 y el máximo es 100 y la "majorTickUnit" es 25, entonces habrá 5 marcas de graduación: una en la posición 0, una en la posición 25, una en la posición 50, una en la posición 75 y una final en la posición 100.
También en las propiedades se indica que se desea un *slider* vertical y con la máxima opacidad.

El siguiente apartado es el de diseño (*Layout*), donde puede indicarse las dimensiones y la ubicación de cada elemento, *Figura 53*.

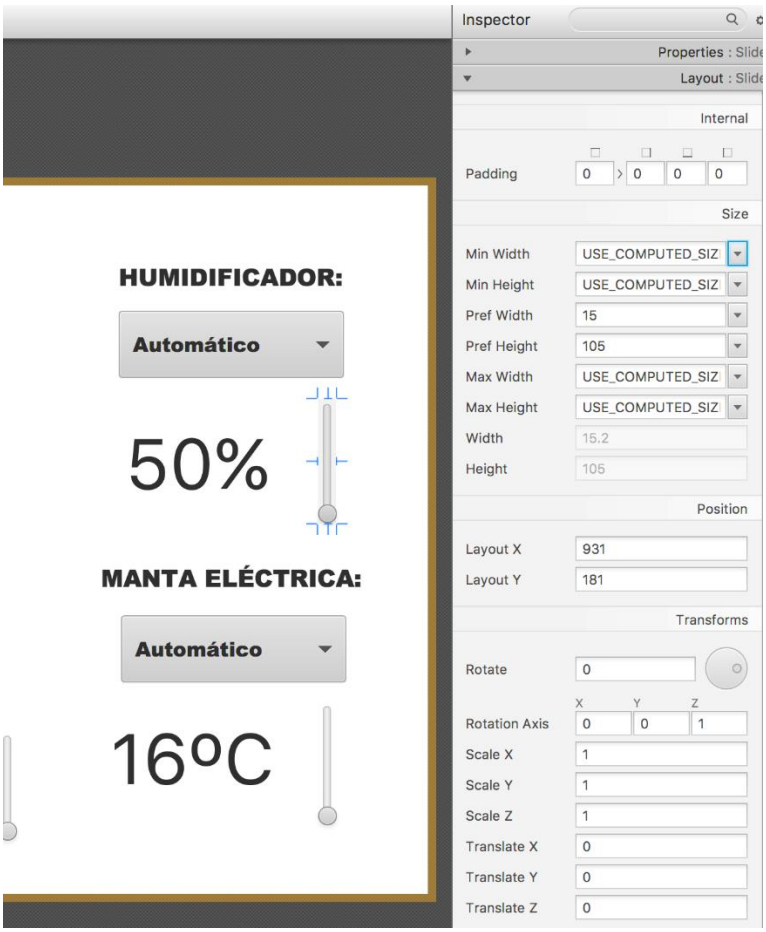


Figura 53. Diseño de elementros de la interfaz

Siguiendo con el ejemplo del *slider*, en este apartado definimos que sea de 105x15 y que se ubique en las coordenadas (931, 181) del panel. También se le podría dotar de cierta inclinación, con la característica de rotación.

Y, por último, la pestaña de código es donde se le asigna un nombre a cada elemento para poder utilizarlo como variable a lo largo del código, en el apartado *Identity*.

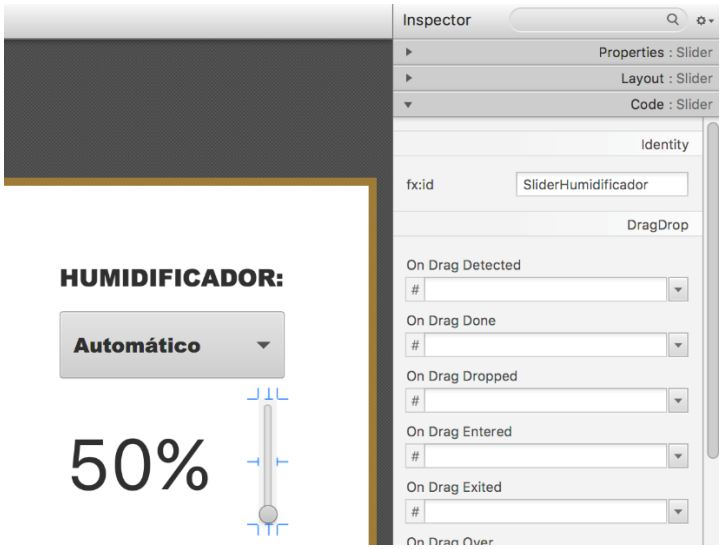


Figura 54. Identificación de los elemetos de la iterfaz

También es posible dar nombre a algunos eventos que puedan suceder sobre estos elementos. En el caso del *slider*, si el control asciende saltará el evento avisando que ha ascendido el control del *slider* (pondrá a TRUE la variable sliderHumidificadorUp) y, en caso de descender, cambiará a TRUE la variable sliderHumidificadorDown.

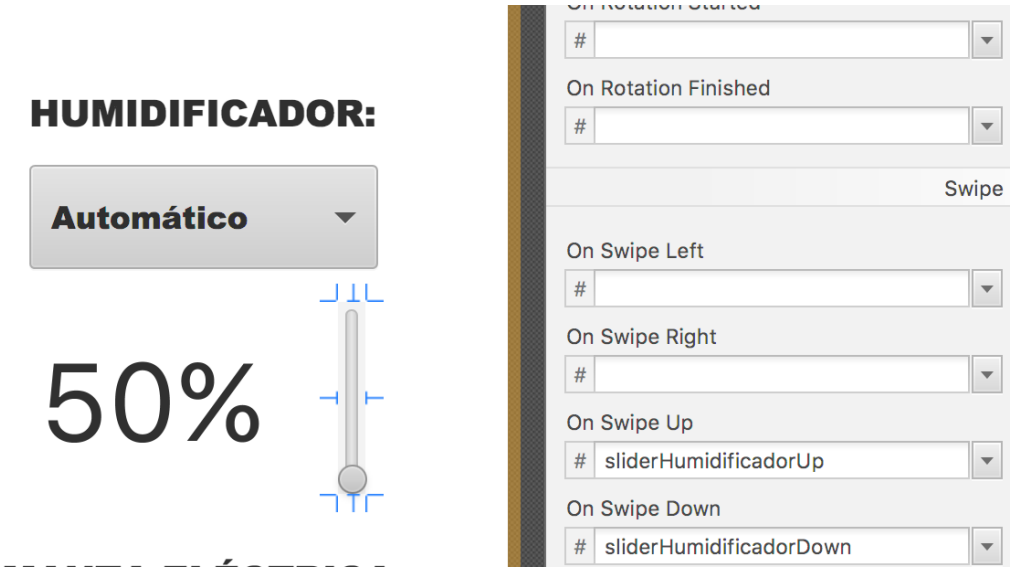


Figura 55. Identificaicón de eventos por cambios sobre la interfaz

Después de la construcción de ambos paneles (uno para cada pantalla), con sus respectivos identificadores, se puede hacer uso de éstos mediante variables a lo largo del código, declarando todos estos elementos previamente en la clase Controller.

Finalmente, la interfaz principal presenta este aspecto:

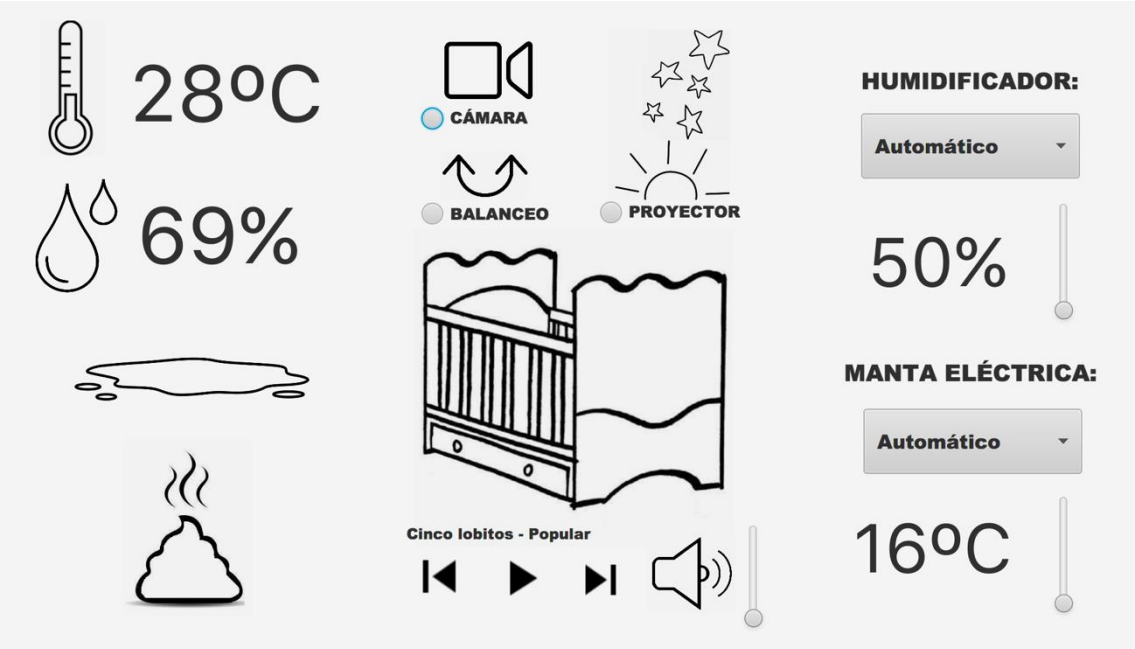


Figura 56. Interfaz principal

En la parte izquierda se encuentran los elementos informativos sobre temperatura, humedad relativa e información sobre si el bebé ha defecado o si la orina ha desbordado del pañal.

Al centro se sitúa la activación del balanceo de la cuna, del proyector y de la emisión de imágenes, así como la reproducción de nanas, la cual ofrece una etiqueta donde se puede ver el título y autor de la nana en curso, puede avanzarse o retroceder a la siguiente canción o pararla, así como subir y bajar el volumen de ésta.

En la parte derecha se encuentra la interactividad con el humidificador y la manta eléctrica, que consiste en un menú desplegable con tres acciones; encender, apagar y automático.
Las acciones de encender y apagar simplemente suponen el encendido o apagado directo de estos actuadores, mientras el modo automático habilita el *slider* con el que modificar el punto de consigna de temperatura y humedad relativa al que se quiere llegar en la habitación y los actuadores se activan o desactivan en función a las siguientes órdenes:

Tª ambiente < Tª consigna – Tª histéresis → Encender manta eléctrica
Tª ambiente >= Tª consigna + Tª histéresis → Apagar manta eléctrica

Humedad relativa ambiente < Humedad relativa consigna - H histéresis → Encender humidificador
Humedad relativa ambiente >= Humedad relativa consigna + H histéresis → Apagar humidificador

Existe una segunda pantalla, que es la que muestra las imágenes captadas desde la cámara de la Raspberry. Esta ventana emergerá en caso de pulsar el botón situado debajo de la imagen de la cámara, durante un tiempo ilimitado.

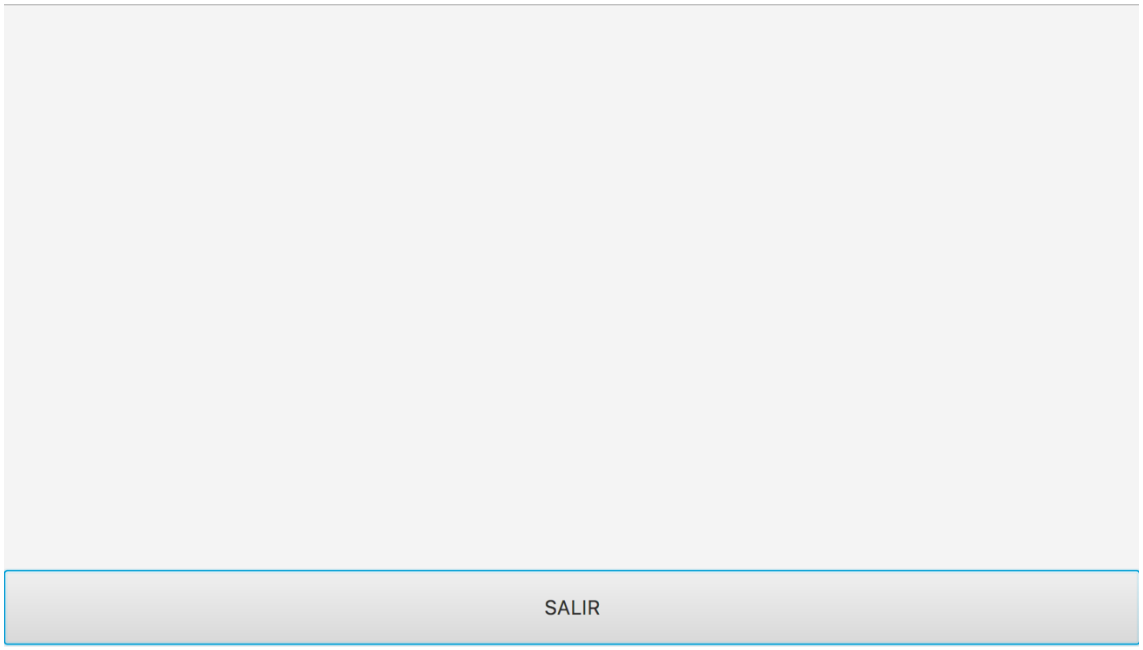


Figura 57. Interfaz secundaria

Únicamente consta de un botón para salir de la visualización y el resto está vacío para mostrar las imágenes en toda la dimensión de la pantalla.

5.1.3 Controlador

El controlador responde a eventos (acciones del usuario sobre la vista) e invoca peticiones al modelo, cuando se hace alguna solicitud sobre la información. Por tanto, se podría decir que el controlador hace de intermediario entre la vista y el modelo.

En este proyecto se han dedicado dos clases a la parte del controlador; Controller y ControllerCamara, la primera alberga el control de la totalidad del proyecto y la segunda el control de la cámara de vídeo, debido a que debe hacerse un controlador por ventana.
Es decir, existe una primera ventana que es la pantalla principal de la interfaz de la aplicación y, si en ésta se elige la opción de activar la cámara, sale otra ventana emergente con las imágenes captadas desde la cámara.

En primer lugar, se va a exponer el código referido a la clase Controller:

```
package raspberry;
```

Se importan las librerías que se van a utilizar a lo largo del código, en esta clase.

Las necesarias para la reproducción de las nanas:

```
import io.neocdtv.player.core.mplayer.Amixer;
import io.neocdtv.player.core.mplayer.MPlayer;
```

También las librerías necesarias para interactuar con la interfaz de usuario:

```
import javafx.application.Platform;
import javafx.beans.InvalidationListener;
import javafx.beans.Observable;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.Label;
import javafx.scene.control.MenuItem;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.Pane;
```

```
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.stage.Stage;
```

Así como las librerías necesarias para realizar determinadas funciones del código:

```
import java.awt.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.LinkedList;

import static java.lang.Thread.sleep;

 //Comienza la clase controller
public class Controller {
```

Como se ha visto en el apartado anterior, la interfaz de usuario se basa en un cuadro de iconos representativos con los que el usuario podrá interactuar para lograr la activación de los distintos actuadores. Estos iconos tienen asignada una etiqueta representada por una variable de texto con el fin de poder jugar con la acción de estos iconos a lo largo del código.

```
 //Se declaran todos los elementos de la interfaz gráfica
 @FXML
 Label labelTemperatura;
 @FXML
 Label labelHumedadRelativa;
 @FXML
 Label labelTemperaturaConsigna;
 @FXML
 Label labelHumedadRelativaConsigna;
 @FXML
 Label labelCancion;
 @FXML
 ImageView ImagePis;
 @FXML
 ImageView ImageCaca;
 @FXML
 ImageView imagenMelodiaBack;
```

```
@FXML
ImageView imagenMelodiaPlay;
@FXML
ImageView imagenMelodiaNext;
@FXML
RadioButton RadioButtonCamara;
@FXML
RadioButton RadioButtonProyector;
@FXML
RadioButton RadioButtonBalanceo;
@FXML
MenuButton MenuButtonHumidificador;
@FXML
MenuButton MenuButtonMantaElectrica;
@FXML
Slider SliderHumidificador;
@FXML
Slider SliderMantaElectrica;
@FXML
Slider sliderVolumen;
@FXML
Pane panelImagenCaca;
@FXML
Pane panelImagenPis;
@FXML
GridPane gridPanePrincipal;
```

Cabe destacar que el tipo de variable *label* se refiere a un cuadro de texto donde se expone, en la interfaz gráfica, la información pertinente; en el que se muestra la información sobre la nana que se está emitiendo por el altavoz, o bien, de los valores referidos a la temperatura y a la humedad relativa, tanto del ambiente como de consigna.
El tipo de variable llamada *imagen* se refiere a un icono representativo sobre el que puede pulsarse para activar una orden; como puede ser el caso del icono de play del reproductor de música.
La variable *Radiobutton* se refiere a un botón, el cual se queda pulsado hasta que vuelva a pulsarse.
MenuButton se refiere a un menú desplegable del que se puede elegir la opción deseada de entre las que se presentan.
El tipo de variable *slider* representa un control deslizante que eleva o disminuye los puntos de consigna de la temperatura y de la humedad relativa, en función de si se sube o se baja el botón sobre la línea de acción.

La variable tipo *Pane* representa un panel en el que se introduce las imágenes referidas a la orina y las heces para poder crear determinados efectos sobre éstas, como es el parpadeo de la imagen. *Grid pane* es el tipo de variable referida a la cuadrícula que guarda toda la pantalla de interfaz de usuario.

```
//Se crea la clase RaspberryPi para ejecutar los métodos que tengan que ver con la Raspberry, como activar actuadores y leer sensores.

RaspberryPi raspberryPi; //Se inicializa la instancia de la clase raspberryPi (EN LA CLASE CONTROLLER)

//Se declaran las variables temp y hum fuera para poder utilizarlas tanto en el hilo primario como en el secundario
String temp;
String hum;
int HumedadRelativaConsignaInt;
int TemperaturaConsignaInt;
boolean parpadeoCaca;
boolean parpadeoPis;
boolean humidificadorEstaActivado = false;
boolean mantaElectricaEstaActivada = false;
boolean haPulsadoCaca;
boolean haPulsadoPis;
boolean modoAutomaticoMantaActivado = true;
boolean modoAutomaticoHumiActivado = true;
boolean estaPlay = true;

MPlayer mediaPlayer;
List listaCanciones = new List(); //Se crea una lista, que contendrá las canciones cuando se le añadan posteriormente
List listaTitulos = new List();

int indice = 1; //Por defecto, se empezará por la canción número 1.
boolean cancionesSeHanCargado=false;
```

En el siguiente método se inicializan los parámetros de la interfaz y se crea un segundo hilo para la lectura simultánea de los sensores.

```
@FXML
public void initialize() { //Cuando la interfaz de javafx ya está activa
```

```
    raspberryPi = new RaspberryPi();// Se crea una instancia de la clase raspberryPi
porque esto debe hacerse dentro de un método y no dentro de una clase

    mediaPlayer = new MPlayer(new Amixer("PCM"));  //Creamos el reproductor de
música y le decimos que reproduzca por el puerto Mini Jack de la raspberry
//Creamos un objeto llamado Amixer que controla el Amixer del reproductor; Amixer es
el programa que tiene linux (sistema operativo de la Raspberry) para controlar el
volumen a través de la entrada mini jack de la raspberry pi por el canal pcm.

    //LLAMO AL MÉTODO QUE DECLARA LOS PINES QUE FORMA PARTE DEL
MODELO
    raspberryPi.inicializar();
```

Como se ha expuesto anteriormente, la variable *MenuButton* se refiere a un menú desplegable que reúne tres opciones para elegir el modo de funcionamiento del control de la temperatura y de la humedad relativa, para el encendido de la mata eléctrica y del humidificador; los tres modos de funcionamiento a elegir son Automático (en función de la temperatura de consigna indicada), Encender y Apagar.

```
    //MENÚ HUMIDIF Y MANTA ELÉCTRICA

//Se indica que ambos menús van a estar formados por tres opciones 0, 1 y 2, llamadas
Items
    MenuButtonHumidificador.getItems().remove(0,2);
    MenuButtonMantaElectrica.getItems().remove(0,2);

//Se cargan los ítems de los menús declarándolos mediante la variable MenuItem
MenuItem automaticoHumi = new MenuItem("Automático");
MenuItem encenderHumi = new MenuItem("Encender");
MenuItem apagarHumi = new MenuItem("Apagar");

MenuItem automaticoManta = new MenuItem("Automático");
MenuItem encenderManta = new MenuItem("Encender");
MenuItem apagarManta = new MenuItem("Apagar");

//HUMIDIFICADOR
//Cuando se seleccione la opción automática del humidificador, desde la interfaz, saltará
un evento y vendrá a esta parte del código.
automaticoHumi.setOnAction(new EventHandler<ActionEvent>()
{
```

```
    @Override
    public void handle(ActionEvent event)
        {
        System.out.println("Automático HUMI");

        MenuButtonHumidificador.setText("Automático"); //Para mostrar en la cabecera
        del menú que el modo seleccionado es automático

        labelHumedadRelativaConsigna.setDisable(false); //NO deshabilitar el label de
        consigna de humedad cuando estemos en modo AUTOMATICO (disable → false)

        SliderHumidificador.setDisable(false); //NO deshabilitar el slider de consigna de
        humedad cuando estemos en modo AUTOMATICO (disable → false)

        modoAutomaticoHumiActivado = true; //La variable booleana de modo
        automático del humidificador se pone a true
        }
});

encenderHumi.setOnAction(new EventHandler<ActionEvent>() { //Cuando se seleccione
la opción manual de ENCENDIDO del humidificador, desde la interfaz, saltará un evento
y vendrá a esta parte del código.
    @Override
    public void handle(ActionEvent event)
        {
        System.out.println("encender HUMI");

        MenuButtonHumidificador.setText("Encender");

        raspberryPi.activarHumidificador();
        //Redirige la acción de activar humidificador a la clase raspberryPi, AL MODELO,
        encargado de cominunicarse con el hardware (pines de la raspberry) conectado
        con el humidificador.

        labelHumedadRelativaConsigna.setDisable(true);//SI deshabilitar el label de
        consigna cuando estemos en modo manual (disable → true)

        SliderHumidificador.setDisable(true); //SI deshabilitar el slider de consigna
        cuando estemos en modo manual (disable → true)
```

```
        modoAutomaticoHumiActivado = false;
        }
});

apagarHumi.setOnAction(new EventHandler<ActionEvent>() { //Cuando se seleccione la opción manual de APAGADO del humidificador, desde la interfaz, saltará un evento y vendrá a esta parte del código.
    @Override
    public void handle(ActionEvent event)
        {
        System.out.println("apagar HUMI");

        MenuButtonHumidificador.setText("Apagar");

        raspberryPi.desactivarHumidificador();//Redirige la acción de desactivar el humidificador a la clase raspberryPi, AL MODELO, encargado de cominunicarse con el hardware (pines de la raspberry) conectado con el humidificador.

        labelHumedadRelativaConsigna.setDisable(true); //SI deshabilitar el label de consigna cuando estemos en modo manual (disable → true)

        SliderHumidificador.setDisable(true); //SI deshabilitar el label de consigna cuando estemos en modo manual (disable → true)

        modoAutomaticoHumiActivado = false;
        }
});
```

Se configura el menú sobre el control de la temperatura y, por tanto, activación de la manta eléctrica, al igual que para el caso de la humedad relativa con el humidificador.

```
   //MANTA ELÉCTRICA
automaticoManta.setOnAction(new EventHandler<ActionEvent>()
{
    @Override
    public void handle(ActionEvent event)
        {
    System.out.println("Automático manta");

    MenuButtonMantaElectrica.setText("Automático");
```

```
    labelTemperaturaConsigna.setDisable(false);
    SliderMantaElectrica.setDisable(false);
    modoAutomaticoMantaActivado = true;
    }
 });

encenderManta.setOnAction(new EventHandler<ActionEvent>()
{
    @Override
    public void handle(ActionEvent event)
        {
    System.out.println("encender manta");

    MenuButtonMantaElectrica.setText("Encender");

        raspberryPi.activarMantaElectrica();//Redirige la acción de activar la
        manta eléctrica a la clase raspberryPi, al modelo.

    labelTemperaturaConsigna.setDisable(true);
        //Deshabilitar el label de consigna cuando estemos en modo manual

        SliderMantaElectrica.setDisable(true);

        modoAutomaticoMantaActivado = false;
        }
 });


apagarManta.setOnAction(new EventHandler<ActionEvent>()
{
    @Override
    public void handle(ActionEvent event)
        {
        System.out.println("apagar manta");

        MenuButtonMantaElectrica.setText("Apagar");

        raspberryPi.desactivarMantaElectrica();//Redirige la acción de activar la manta
        eléctrica a la clase raspberryPi, AL MODELO.
```

```
            labelTemperaturaConsigna.setDisable(true);
            //deshabilitar el label de consigna cuando estemos en modo manual.

            SliderMantaElectrica.setDisable(true);

            modoAutomaticoMantaActivado = false;
        }
    });
```

Una vez creados los objetos con la función new y descrito lo que se debe hacer en cada caso, se añaden estos objetos a los menús correspondientes.

```
MenuButtonHumidificador.getItems().addAll(automaticoHumi, encenderHumi,
apagarHumi);

MenuButtonMantaElectrica.getItems().addAll(automaticoManta, encenderManta,
apagarManta);
```

A continuación, se expone el código referido a la inicialización del reproductor de nanas. Son tres las canciones que pueden reproducirse por los altavoces de la cuna, las cuales han sido almacenadas en la rapsberry en formato .wap.
La reproducción se controla con la interfaz gráfica, la cual dispone de una etiqueta (label) donde se muestra la canción que se está reproduciendo, cuatro botones; play, stop, next (siguiente canción), back (canción anterior) y un *slider* (control deslizante) para regular el volumen de los altavoces.

```
    labelCancion.setText("Ninna Nanna Brahms"); //Para que, al abrir la app, en el label
que muestra las canciones, salga ese título ya que por defecto es la primera nana que
se va a escuchar (índice 1)

    sliderVolumen.setValue(50.0); //Para que el slider empiece a la mitad (volumen
medio), ya que se ha diseñado el slider con un valor máximo de 100 y mínimo de 0.

    sliderVolumen.valueProperty().addListener(new InvalidationListener() { //Con
valueProperty() coges las propiedades del slider y le ponemos un listener al slider para
que en el momento se toque se meta en este método.
```

Cabe destacar que *Listener* describe la función que redirige a una línea determinada del código, en caso de que se produzca un cambio de valor ejecutado desde la interfaz.
El método *invalidated* es uno de los más utilizados para redirigir a una línea del código determinada en caso de detectarse un cambio en el valor del *slider*, un cambio en el valor del

volumen en este caso. Este método invalida el valor anterior de volumen y actúa en función del nuevo valor del *slider*.

```
@Override
public void invalidated(Observable observable)
{ //Método que ejecuta el Listener cuando se pulse encima del slider

        double volumenActual = sliderVolumen.getValue(); //Se mete el valor del slider
        en la variable volumenActual, de tipo double.
                System.out.println(sliderVolumen.getValue()); //Imprime por pantalla el
        valor del volumen

        mediaPlayer.setVolume(volumenActual / 100); //Se modifica el volumen de
        emisión según el valor cogido del slider, se divide entre 100 porque el rango de
        valores del slider es entre 0 y 100 y el rango que domina el reproductor es entre
        0 y 1.
                System.out.println(mediaPlayer.getVolumeInMillibels()); //Se imprime
        por pantalla el valor del volumen que se está emitiendo
}
}

);
```

A continuación, se lleva a cabo la lectura de los sensores. Para ello se ha creado un hilo en segundo plano de forma que se esté escuchando constantemente a los sensores, con un bucle *while*, de forma simultánea a la ejecución de la aplicación.

```
   Thread hiloSecundario = new Thread(new Runnable()
      { //Creanción de un hilo secundario

        @Override //Quiere decir que el método Run ya existe y
        estás sobreescribiendo sobre este.

     public void run()
     { //Segundo plano

       while (true)
                { //Bucle While para que no pare de preguntar a los   sensores

                int[] humTemp = {0, 0}; //La declaramos fuera porque la vamos a utilizar
                también en el if de comparación con la consigna
```

```
try { //SE LLAMA AL MÉTODO QUE ME DEVUELVE LA TEMP Y H. RELATIVA DEL DHT11 (este método puede lanzar una excepción, por eso lo pongo dentro de un try catch)

humTemp = raspberryPi.obtenerTemperaturayHumedad();

temp = String.valueOf(humTemp[1]); //Se pasa el int con la temp. que me devuelve el metodo a string para poder mostrarlo por pantalla

hum = String.valueOf(humTemp[0]); // Se pasa el int con la hum. relativa a string para poder mostrarlo por pantalla
   } catch (Exception e) {
         e.printStackTrace();
        }

try {
//SE LLAMA AL MÉTODO QUE ME DICE SI SE HA MEADO (LEE SENSOR HUM. ABSOLUTA), dentro de otro try catch por si el método lanza una excepción.

         boolean seHaMeado = raspberryPi.seHaMeado(); //Meto   el resultado del método (true o false) en la variable SeHaMeado de tipo booleana.

         if (seHaMeado == true)
         {

    //Mostrar imagen pipi parpadeando
  parpadeoPis=true;

         }
   } catch (Exception e) {
         e.printStackTrace();
   }

try {
//LLAMO AL MÉTODO QUE ME DICE SI SE HA CAGADO (LEE SENSOR CALIDAD AIRE) , dentro de otro try catch por si el método lanza una excepción.
boolean seHaCagado = raspberryPi.seHaCagado(); //Meto el resultado
```

```
            del método (true o false) en la variable SeHaCagado

            if (seHaCagado == true)
            {

      //Mostrar imagen caca parpadeando
      parpadeoCaca = true;
    }

  } catch (Exception e) {
    e.printStackTrace();
  }
```

Platform.*runLater*(new Runnable() { //Para volver al primer plano porque queremos modificar algo de la interfaz gráfica.

```
        @Override public void run()
        {
        //MODIFICAR LA INTERFAZ DE USUARIO AQUÍ
        labelTemperatura.setText(temp + "ºC"); //Muestra en la interfaz la temperatura
        medida con el DHT11. (temperatura del ambiente)

        labelHumedadRelativa.setText(hum + "%"); //Muestra en la interfaz la humedad
        relativa medida con el DHT11. (humedad relativa del ambiente)

        //Sliders; Aprovechamos la excursión a primer plano para coger la información
        del slider de temperatura y humedad de consigna de la interfaz y modificar con
        esto las etiquetas de consigna:

        HumedadRelativaConsignaInt = (int)

        SliderHumidificador.getValue(); //Se coge el valor del slider de hum relativa de
        consigna, lo paso a int y lo meto en la variable HumedadRelativaConsignaInt.

        labelHumedadRelativaConsigna.setText(Integer.toString(HumedadRelativaConsi
        gnaInt) + "%"); //Modifico el texto del label de la humedad de consigna y el texto
        que pongo es el valor de la variable HumedadRelativaConsignaInt, pasada a
        string (para poder mostrar; settext necesita un string)

        TemperaturaConsignaInt = (int)
```

```
SliderMantaElectrica.getValue(); //Cojo el valor del slider de temperatura de
consigna, lo paso a int y lo meto en la variable TemperaturaConsignaInt.

labelTemperaturaConsigna.setText(Integer.toString(TemperaturaConsignaInt) +
"ºC"); //Coger el valor marcado en el slider del humidificador y pasarlo a string
para mostralo en el label de consigna de la humedad relativa.
```

En el segundo plano, mediante la lectura tanto del sensor de humedad absoluta como el de calidad de aire, se pone a TRUE la variable parpadeoCaca o parpadeoPis en caso de que la respuesta del sensor, consultada al modelo "RaspberryPi" sea afirmativa.
Es en el primer plano del Controller donde se manda a la interfaz la acción de hacer parpaear el icono de la orina o de las heces, siempre y cuando las variables parpadeoCaca o parpadeoPis estén a TRUE y que no se haya presionado sobre estos iconos, desde la interfaz para que dejen de parpadear, ya que cuando los padres se hayan dado cuenta de que el bebé ha defecado u orinado, por el parpadeo de los iconos, y ya hayan tomado acciones sobre ésto, pueden hacer que deje de parpadear el icono presionando sobre éste.
El parpadeo se realiza de forma sencilla; si la imagen está visible se procede a hacerla invisible y viceversa.

```
if(parpadeoCaca && !haPulsadoCaca )
{

                if(panelImagenCaca.isVisible())
                { //Si la imagen está visible → ocultarla
                panelImagenCaca.setVisible(false);
                }
                else
                { //Si ya estuviera invisible → visibilizar
                panelImagenCaca.setVisible(true);
                }
}
else
{
panelImagenCaca.setVisible(true); //Si no ha defecado dejar
la imagen visible en pantalla normal sin parpadear
}

if(parpadeoPis && !haPulsadoPis )
        {
            if(panelImagenPis.isVisible())
```

```
                { //Si la imagen está visible → ocultarla
                panelImagenPis.setVisible(false);
            else
                { //Si ya estuviera invisible → visibilizar
            panelImagenPis.setVisible(true);
                }
    }
    else
    {
    panelImagenPis.setVisible(true); //Si no ha orinado dejar
   la imagen visible en pantalla normal sin parpadear
     }
     }
     });
```

En el caso del humidificador y de la manta eléctrica, se necesita la información sobre la temperatura y la humedad del ambiente, proveniente del modelo "raspberryPi". Dichos valores ya se han consultado anteriormente en el hilo secundario, dedicado a lectura de sensores, y se han guardado en el array humTemp.
En esta clase, Controller, disponemos de los datos especificados por el usuario mediante la interfaz como la temperatura y humedad de consigna, por lo que para ello no tenemos que llamar al modelo "raspberryPi". Aunque si debemos invocarlo a la hora de ofrecer la respuesta; si se quiere encender o apagar el humidificador o la manta eléctrica, ya que el modelo es el que se comunica con el *hardware*; con los pines de la Raspberry.

A diferencia del caso anterior, para tratar el aviso de orina y heces, únicamente se necesita del modelo "raspberryPi" la información de los sensores, pero la respuesta era puramente representativa (hacer parpadear imágenes de la interfaz) por lo que no ha sido necesario invocar de nuevo al modelo.

Anteriormente, en esta misma clase Controller, cuando se producía un evento sobre el menú de la manta o del humidificador para encenderlos o apagarlos se ha enviado al modelo "raspberryPi" la orden de encender o apagar el humidificador o la manta eléctrica.
En el caso del modo automático, únicamente se ha puesto la variable modoAutomaticoHumiActivado o modoAutomaticoMantaActivado a TRUE para tratarla a continuación, donde después de comparar las condiciones reales con las de consigna, se dará la orden al modelo de encender o apagar dichos dispositivos

```
//LLAMADA AL MÉTODO PARA ACTIVAR HUMIDIFICADOR
if(modoAutomaticoHumiActivado == true)
{
    if (HumedadRelativaConsignaInt + Hh > humTemp[0] && !humidificadorEstaActivado)
```

```
        { //Si la humedad de consigna + Hh es mayor a la del ambiente y no está
        encendido aún el humidificador → Se enciende
                humidificadorEstaActivado = true;
                raspberryPi.activarHumidificador();
        }
        else if (HumedadRelativaConsignaInt – Hh <= humTemp[0] &&
        humidificadorEstaActivado)
        { //Si la humedad de consigna - Hh es menor o igual a la del ambiente (la
        habitación ha conseguido la humedad de consigna deseada) y está encendido el
        humidificador → Se apaga
                humidificadorEstaActivado = false;
                raspberryPi.desactivarHumidificador();
        }
 }

//LLAMADA AL MÉTODO PARA ACTIVAR MANTA ELÉCTRICA
if(modoAutomaticoMantaActivado == true)
{
        if (TemperaturaConsignaInt +Th > humTemp[1] && !mantaElectricaEstaActivada)
        {
                        mantaElectricaEstaActivada = true;
                        raspberryPi.activarMantaElectrica();
        }
        else if (TemperaturaConsignaInt - Th <= humTemp[1] &&
        mantaElectricaEstaActivada)
        {
                mantaElectricaEstaActivada = false;
                raspberryPi.desactivarMantaElectrica();
         }
  }
         try {
           sleep(500);
         } catch (InterruptedException e) {
           e.printStackTrace();
         }
       }
     }

   });
   hiloSecundario.start(); //Es al final del código del segundo plano cuándo se le
 invoca.
```

```
} //Fin del método initialize
```

Una vez inicializados los parámetros de la interfaz y leídos los sensores, realizado en un hilo secundario, se va a exponer el código referido a la interacción con la vista. Ya se ha visto anteriormente alguna interacción con la interfaz, ya que la acción era dependiente de la lectura o acción de sensores o actuadores, pero a continuación se tratan los demás.

Como se ha expuesto anteriormente, si se detecta que el bebé ha defecado u orinado comenzará a parpadear el icono correspondiente, entonces los padres deberán pulsar sobre este icono para detener el parpadeo, como señal de que se han enterado y proceden a tomar acciones sobre ello. El caso es que en el lapso de tiempo en que pulsan el icono y realizan la acción de cambiar al bebé, los sensores de humedad absoluta y calidad de aire seguirán dando respuesta afirmativa; por ello se deja un tiempo de 15 minutos en que los iconos no parpadean aunque la respuesta de los sensores sea afirmativa, ésto también se realiza en un hilo secundario, llamado esperarParaVolveraParpadearCaca, para que mientras pasan los 15 minutos se siga ejecutando el resto de código y, por tanto, la aplicación.

```
@FXML
public void panelCacaPulsado(){ //Si se pulsa sobre el icono
  System.out.println("panelCacaPulsado");
  parpadeoCaca = false; //Se para el parpadeo
  haPulsadoCaca = true; //Se pone a true la variablehaPulsadoCaca para que no
entre en el if visto anteriormente y no se ponga a parpadear
  Thread esperarParaVolveraParpadearCaca = new Thread(new Runnable() {
    @Override
    public void run() {
      try {
        Thread.sleep(60000); //60000s = 15m; tiempo para que
        los padres cambien al niño
      } catch (InterruptedException e) {
        e.printStackTrace(); //Puesto que el temporizador
        puede lanzar una excepción, la espera se realiza
        dentro de un try catch
      }
      haPulsadoCaca = false; //Pasados los 15 minutos, se pone
      a false la variable haPulsadoCaca y volverá a parpadear
      en caso ser afirmativa la respuesta del sensor de
      calidad de aire
    }
  });
  esperarParaVolveraParpadearCaca.start();
```

```
}
```

Del mismo modo que con el icono de las heces, se esperan 15 minutos para que pueda volver a parpadear el icono de la orina, en caso de que la respuesta del sensor de la humedad absoluta sea afirmativa.

```
@FXML
public void panelPisPulsado(){
  System.out.println("panelPisPulsado");
  parpadeoPis = false;
  haPulsadoPis = true;
  Thread esperarParaVolveraParpadearPis = new Thread(new Runnable() {
    @Override
    public void run() {
      try {
        Thread.sleep(60000);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
      haPulsadoPis = false;
    }
  });
  esperarParaVolveraParpadearPis.start();
}
```

El reproductor que se muestra en la interfaz para emitir las nanas consta de un botón con una imagen de *play* que, al pulsar sobre él, se convierte en una imagen de *pause,* dos flechas que dan a opción de pasar a la anterior o a la siguiente canción, un *slider* para el control del volumen y una etiqueta en la que se muestra la nana y el autor de ésta que se está emitiendo.

```
//CÓDIGO MELODÍA
@FXML
public void imagenBackPulsada()
      {
  if(!estaPlay){ //Si no está reproduciendo nana, comienza a hacerlo
    estaPlay=true;
  }
       else
       {
    estaPlay=false;
  }
```

```
            imagenMelodiaNext.setOpacity(1); //Porque en el momento me voy para atrás ya no estoy en el límite máximo (canción 3) y tengo que volver a habilitar y hacer invisible el botón de ir para adelante

        imagenMelodiaNext.setDisable(false); //Habilitar

   if (indice > 0) { //Si no está en la primera canción de la lista, porque en ese caso ya no se podría ir más hacia atrás.
     mediaPlayer.stop(); //Para parar la melodía que se está reproduciendo anteriormente
     indice = indice -1; //Se va a la canción anterior
     imagenPlayPulsada(); //Como si le volvieras a dar a play para que la canción anterior se reproduzca. Le damos a play internamente (con la programación)
   }
       if(indice==0)
       {
                imagenMelodiaBack.setDisable(true); //Si ya estoy en la primera canción inhabilitar el botón de back porque no se puede ir más para atrás.
                 imagenMelodiaBack.setOpacity(0.4);
        }
 }
  public void cargarCanciones() { //Se ha llamado a este método en el inicialize en cuanto se pulsa el botón de play para meter las canciones en la lista de canciones
    cancionesSeHanCargado=true; //Para que se del ok de que las canciones se han cargado
    listaCanciones.add("cancion1",0); //Añadir las canciones a la lista asignándoles un orden con los índices
    listaTitulos.add("Cinco lobitos - Popular",0); //Ponerle un título a cada canción según su índice
    listaCanciones.add("cancion2",1);
    listaTitulos.add("Ninna Nanna Brahms",1);
    listaCanciones.add("cancion3",2);
    listaTitulos.add("Música Relax Dormir",2);

  }
  @FXML
    public void imagenPlayPulsada() { //Cuando se pulsa el botón play/pause se viene a este método
    //Inicialmente está mostrada la imagen de play, y no la de pause (estaPlay = true)
    if(estaPlay) { //Si la imagen mostrada es la del play
```

```
    Image imagenMelodiaPause = new Image("raspberry/imagenes/pause.png"); //Se crea un objeto de imagen con la url donde se encuentra la imagen pause.png
    imagenMelodiaPlay.setImage(imagenMelodiaPause); //Se cambia el icono play por el de pause
    estaPlay=false; //Se pone a false para que si se le vuelve a pinchar sobre este icono (ahora mostrando pause) se vaya al else de esta declaración if y vuelva a ponerse el play
    if(!cancionesSeHanCargado)
       { //Si las canciones NO se han cargado
      cargarCanciones(); //Para asegurarme de que se han cargado ya las canciones ya que cada vez que se abra la app tienen que cargarse de 0.
    }

    String cancion = "canciones/"+listaCanciones.getItem(indice)+".wav"; //Se busca la canción en la carpeta canciones según el índice, que ha podido variarse con next o back, y se mete en un string llamado canción (cancion1, cancion2 o cancion3)
    labelCancion.setText(listaTitulos.getItem(indice)); //Para mostrar título de canción en el label cogiéndolo de la lista de títulos mediante el índice
    mediaPlayer.setVolume(sliderVolumen.getValue() / 100); //Coge el valor del slider de volumen y lo envía al reproductor para que ya desde el principio emita el volumen que demanda el slider.
    mediaPlayer.play(cancion); //Reproducir
  }
      else
      {
    estaPlay=true; //Si vuelve a pulsarse sobre el icono que está mostrando la imagen de pause, porque el play está en ejecución, vuelva a mostrarse el icono play

    Image imagenMelodiaPause = new Image("raspberry/imagenes/play.png"); //Coger la imagen de play
    imagenMelodiaPlay.setImage(imagenMelodiaPause); //Cambiar la imagen de pause que hay por la de play
    mediaPlayer.pause(); //Parar de reproducir
  }
 }

 @FXML
 public void imagenNextPulsada() {
  if(!estaPlay){
   estaPlay=true;
  }else{
```

```
      estaPlay=false;
    }
    imagenMelodiaBack.setOpacity(1); //Porque en el momento me voy para adelante
ya no estoy en el límite mínimo (canción 0) y tengo que volver a habilitar y hacer
invisible el botón de ir para atrás
    imagenMelodiaBack.setDisable(false);
    if (indice < 3) { //Si no está ya en la última canción
      mediaPlayer.stop(); //Para parar la melodía que se está reproduciendo
anteriormente
      indice = indice + 1; //Se va a la siguiente canción
      imagenPlayPulsada(); //Se da a play internamente
    if(indice==2){
      imagenMelodiaNext.setDisable(true); //Si ya estoy en la última canción inhabilitar el
botón de next porque no se puede ir más para delante
      imagenMelodiaNext.setOpacity(0.4);
    }

    }
  }
  @FXML
  public void sliderMantaElectricaDown() {
  }

  @FXML
  public void sliderMantaElectricaUp() {
  }

  @FXML
  public void sliderHumidificadorDown() {
  }

  @FXML
  public void sliderHumidificadorUp() {
  }
  @FXML
  public void sliderVolumenMoved() {
  }

  @FXML
  public void radioButtonProyectorPulsado() {
    if (RadioButtonProyector.isSelected()) {
```

```
      System.out.println("Activar Proyector");
      raspberryPi.activarProyector();
    } else {
      System.out.println("Desactivar Proyector");
      raspberryPi.desactivarProyector();
    }
  }

  @FXML
  //LLAMADA AL MÉTODO PARA ACTIVAR BALANCEO
  public void radioButtonBalanceoPulsado() {
    if (RadioButtonBalanceo.isSelected()) {
      System.out.println("Activar Balanceo");
      raspberryPi.activarBalanceo();
    } else {
      System.out.println("Desactivar Balanceo");
      raspberryPi.desactivarBalanceo();
    }
  }

  @FXML
  //LLAMADA AL MÉTODO PARA ACTIVAR CAMARA
  public void radioButtonCamaraPulsado() throws IOException { //Cuando se le pulse al
radio button camara
    Parent root = FXMLLoader.load(getClass().getResource("camara.fxml"));
        //Representar el fichero camara.fxml en código para pasársela al stage
    Stage camaraStage = new Stage(); //Creamos el Stage(Jframe) de la interfaz de la
cámara, la secundaria
    camaraStage.setTitle(""); //No le ponemos título,  lo vamos a tapar
    camaraStage.setScene(new Scene(root, 1024, 600)); //Meter las dimensiones del
JFrame
    camaraStage.show(); //Hago que se vea la pantalla de la cámara en vez de la de la
app, se pone encima
    ControllerCamara.stageCamara = camaraStage; //Scene builder crea un controlador
para cada ventana, por lo que ha creado uno para la cámara, por esto no podemos
crear un Controller cámara ya que crearía uno diferente al que crea el scene builder.
Se crea una variable tipo stage para poder controlar el Jframe de la cámara desde el
controlador

    RadioButtonCamara.setSelected(false); //Como ya se ha realizado la orden de sacar
las imágenes de la cámara por pantalla, se deselecciona el botón de mostrar
```

```
 imágenes.
  }
}
```

Existe otro controlador, ya que debe de haber uno por pantalla de la interfaz, por lo que se ha realizado un controlador para la cámara, que alberga el siguiente código:

```
package raspberry;

//Se importan las librerías necesarias para poder realizar este proceso
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;

public class ControllerCamara { //Como no tengo el new del controller camara, porque me crearía uno diferente al que me crea la interfaz secundaria

  public static Stage stageCamara; //Es un stage estático para poder cargar un stage (el de la camara) desde cualquier lugar, esto lo hacemos para poder acceder al ControladorCamara que crea JavaFx; modificamos el Stage desde fuera(desde la clase Stage), sin acceder al objeto

  @FXML
  public void initialize(){ //Para que empiece a mostrarme imágenes cuando se active la segunda pantalla(pantalla de la cámara)
    Runtime rt = Runtime.getRuntime();
    File archivo = new File("ejecutables/preview.sh"); //Cogemos el archivo preview indicándole la ruta relativa (dentro del IntelliJ) --> dentro de éste se ejecuta el proceso raspistill que es el que muestra las imagenes de la cámara
    try {
      process = rt.exec("bash " + archivo.getAbsolutePath()); //Abrimos el terminal (cmd) del pc y con el comando bash ejecutamos el archivo, siguiendo la ruta absoluta (desde cualquier parte del pc)
    } catch (IOException e) {
      e.printStackTrace();
```

```
    }
  }

//Se declaran los elementos de la pantalla dedicada a la cámara
  @FXML
  Button buttonSalirCamara;
  @FXML
  Pane paneCamara;
  Process process;

  @FXML
  public void buttonSalirCamaraPulsado() {
    invisible(); //Llamo el método invisible
  }

  public void invisible() {
    stageCamara.hide(); //Hacer desaparecer todo el Stage de la pantalla secundaria(el
cuadro entero; con botones de cerrar, minimizar..)
    terminarCamara();//Llamo a terminar camara
  }

  public void terminarCamara() { //Quitar las imagenes de la cámara, dejar de ejecutar el
preview deshabilitando el raspistill
    try {
      Runtime rt = Runtime.getRuntime();
      Process psAux = rt.exec("pidof raspistill"); //Abrimos el terminal y le pedimos el id del
proceso raspistill (que se ejecuta dentro del preview)
      BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(psAux.getInputStream())); //Leer la lÍnea que me devuelve el
proceso ejecutado arriba
      String pid = bufferedReader.readLine(); //Guardo el número del proceso raspistill en
la variable string llamada pid
      rt.exec("sudo kill " + pid); //Ejecutamos el comando kill + el numero del proceso para
dejar de ejecutar el proceso raspistill --> para dejar de ver por la cámara
    } catch (IOException e) {
      e.printStackTrace();
    }

  }
}
```

5.1.4 Grafcets

En este apartado van a exponerse los grafcets sobre la activación de los actuadores que constituyen el proyecto para dar una idea sobre la estructura del código de programación más visual, con el fin de dar a entender jerárquicamente con qué pautas se ha constituido el código de programación y sea más fácil la comprensión de éste.

La simbología que se ha utilizado es la común en los conocidos grafcets, donde las líneas horizontales representan una condición que, de cumplirse, llevará a un estado; rectángulos que definen la ejecución de una acción (activación o desactivación del actuador) y los rectángulos con otro rectángulo dentro representan el inicio del flujo de trabajo.

5.1.4.1 Manta eléctrica

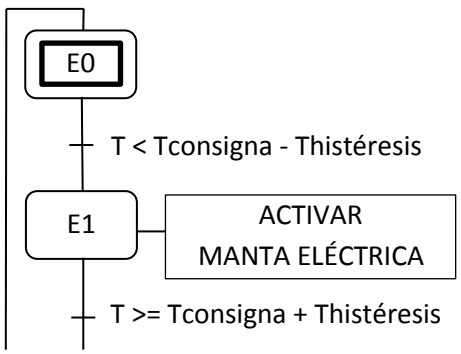
En primer lugar, se va a desarrollar el diagrama de flujo para la activación de la manta eléctrica.

Tanto la manta eléctrica como el humidificador tienen dos modos de funcionamiento a elegir desde la interfaz gráfica de la cuna.

- **Manual**: El primer modo es el manual, el cual reúne dos de las tres opciones que muestra el menú desplegable de estos dos actuadores.
  Estas dos opciones son “Encendido” y “Apagado”, las cuales fuerzan directamente, el encendido o apagado, respectivamente de la manta eléctrica o del humidificador.

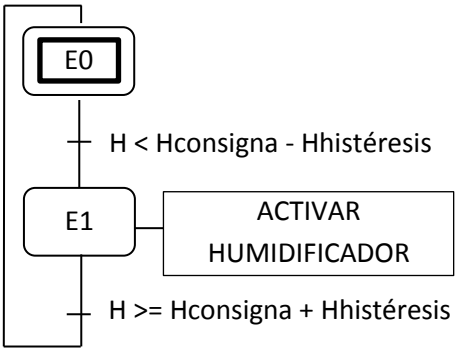
- **Automático:** Este modo basa su funcionamiento en dos variables; temperatura sensada y temperatura de consigna.
  La temperatura sensada es la que se lee del sensor DHT11 constantemente y la cual es comparada con la temperatura de consigna, constantemente también, para dar paso a la activación o desactivación de la manta eléctrica, que se basa en un control todo/nada.
  La temperatura de consigna es a la que el usuario desea que se encuentre la habitación donde se ubica la cuna y lo indica mediante interfaz gráfica.

  El grafcet que se muestra es el flujo de trabajo que se realiza cuando se selecciona el modo automático, ya que es el que presenta mayor complejidad por el hecho de estar basado en la lectura de temperatura del sensor del DHT11 así como en la temperatura de consigna marcada por el usuario.



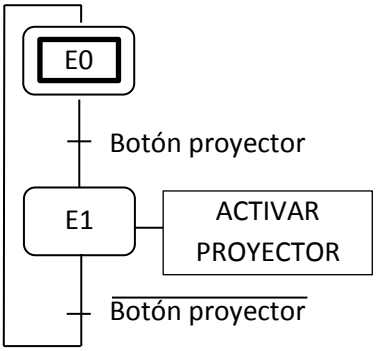
5.1.4.2 Humidificador

El flujo de trabajo del humidificador es similar al de la manta eléctrica pero basado en la lectura de la humedad relativa del sensor del DHT11, así como la humedad relativa de consigna marcada por el usuario desde la aplicación de la cuna.
Del mismo modo que en el caso anterior, el diagrama que se muestra es el flujo de trabajo que se realiza cuando se selecciona el modo automático, por depender de dos variables.



5.1.4.3 Proyector

El flujo de trabajo del proyector es muy sencillo debido a que su activación o desactivación depende únicamente de si es pulsado el botón de la interfaz gráfica referido a este actuador o si, por lo contrario, es despulsado. Es decir, se basa en un control todo/nada.

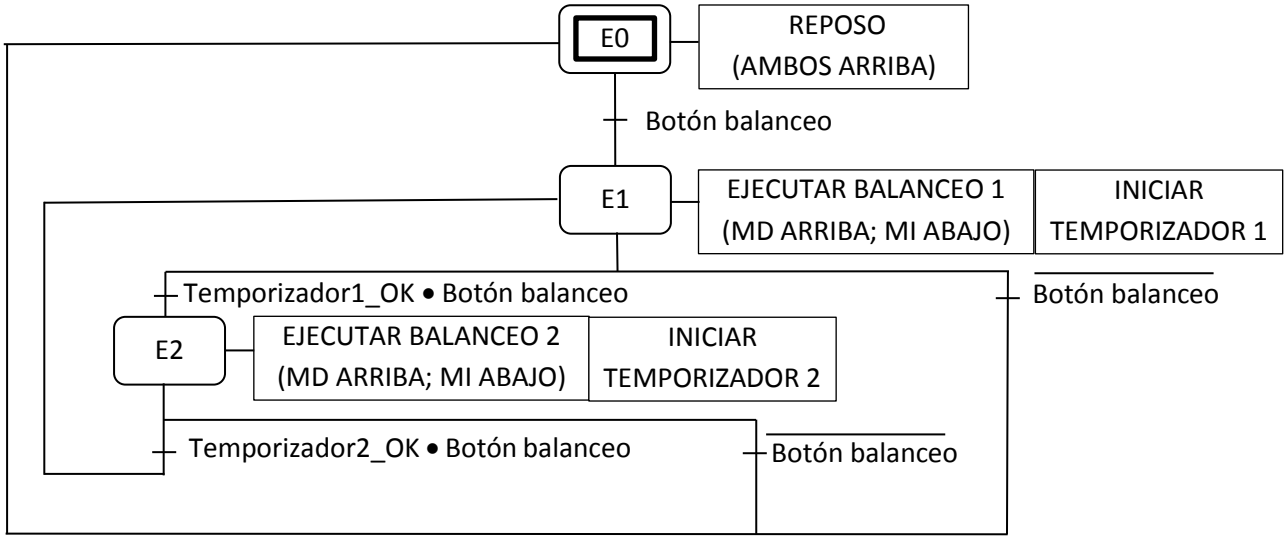


5.1.4.4 Motores lineales

Los motores lineales basan su activación o desactivación dependiendo de si es pulsado el botón de la interfaz gráfica referido al balanceo o, por lo contrario, si es despulsado.
Para programar su funcionamiento se han creado tres métodos basados en los distintos movimientos del motor situado a la derecha (motor derecho) y el situado a la izquierda (motor izquierdo):

- **Balanceo1**: Consiste en activar el motor derecho para que ejecute la subida hasta su tope mientras se ejecuta la bajada del motor izquierdo hasta su tope.
- **Balanceo2**: Consiste en activar el motor izquierdo para que ejecute la subida hasta su tope mientras se ejecuta la bajada del motor derecho hasta su tope.
- **Reposo**: Consiste en efectuar la subida de ambos motores, bien para dejar ambos motores en el límite superior al despulsar el botón del balanceo o para asegurarse que ambos motores están arriba al inicio del balanceo.

Cabe destacar que la espera para que se efectúe el movimiento de los motores hasta el tope inferior y superior se realiza mediante un temporizador, al que se le ha asignado un tiempo de espera de ocho segundos, tiempo calculado a partir de la distancia recorrida en vertical por los motores y la velocidad que ofrecen éstos.



5.2 Desarrollo hardware

Conociendo la tecnología *hardware* que se ha empleado, detallada en el apartado tres, en éste se va a mostrar el conexionado a la Raspberry PI, tanto de sensores como de actuadores y periféricos, con el fin de proporcionar una idea genérica del proyecto.

En primer lugar, se han tenido en cuenta las alimentaciones necesarias para cada uno de los elementos del proyecto y de dónde proceden éstas, *Tabla 6*, dividiendo en varios grupos; sensores, actuadores, periféricos y control, que se refiere a la tarjeta de relés y a la Raspberry Pi.

	ELEMENTO	ALIMENTACIÓN	FUENTE
SENSORES	DHT11	5VDC	RASPBERRY
	CALIDAD AIRE	5VDC	RASPBERRY
	HUM ABSOLUTA	5VDC	RASPBERRY
ACTUADORES	PROYECTOR	230VAC	RED-RELÉ
	MANTA ELÉCTRICA	230VAC	RED-RELÉ

	HUMIDIFICADOR	230VAC	RED-RELÉ
	MOTOR DERECHO	12VDC	BATERÍA
	MOTOR IZQUIERDO	12VDC	BATERÍA
PERIFÉRICOS	ALTAVOCES	230VAC	RED
	CÁMARA	3,3VDC	RASPBERRY
	PANTALLA	3-5VDC	RASPBERRY
CONTROL	RASPBERRY PI	230VAC	RED
	TARJETA RELÉS	5VDC	RASPBERRY

Tabla 6. Alimentaciones

5.2.1 Sensores

Como puede observarse en la tabla, todos los sensores precisan una alimentación de 5V corriente continua, esta alimentación la ofrece la Raspberry Pi, la cual puede proporcionar tensiones de 3,3V y 5V. En este caso, es necesaria la alimentación de 5V, que se consiguen del pin 2 de la Raspberry, al igual que la masa, que se consigue del pin 6 de la Raspberry Pi, y se llevan hasta una regleta de conexión formada por cinco bornas, dos de las cuales suponen la entrada de alimentación (Vcc y GND) desde la Raspberry Pi y la salida de éstas ofrecen alimentación a los tres sensores y a la placa de relés, conforme muestra la *Figura 58*.

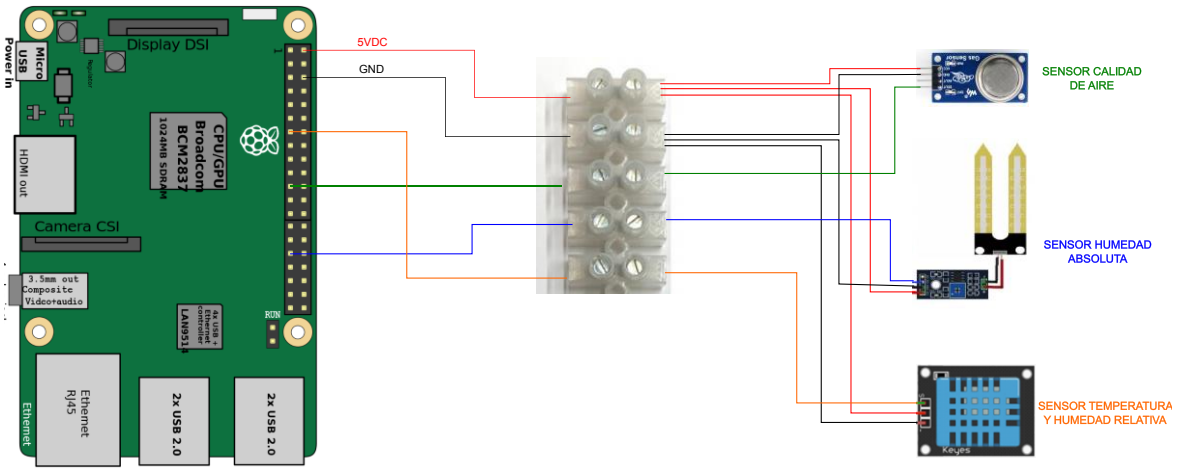


Figura 58. Conexión Raspberry - sensores

Las tres bornas restantes de la regleta son utilizadas para la transmisión de datos desde los sensores a la Raspberry. Como se ha visto en capítulos anteriores, la salida de datos de los sensores llegan a GPIOS de la Raspberry para que ésta pueda leer los datos sensados mediante el código de programación. Esta conexión se lleva a cabo mediante estas bornas para hacer más robusto el montaje.

5.2.2 Periféricos

Como se ha visto anteriormente, este proyecto presenta tres periféricos; cámara, pantalla y altavoces. Tanto la cámara como la pantalla tienen conexión directa con la Raspberry, con ésta intercambian datos y les ofrece alimentación.

Los altavoces, como puede verse en la tabla, precisan una alimentación de la toma de red de 230Vac, por lo que se alimenta directamente a través del enchufe general de la cuna, por ello se mantendrán encendidos siempre que la cuna lo esté, pero sólo emitirán sonido cuando el usuario lo ordene a la Raspberry. Los altavoces están conectados a ésta mediante un puerto minijack para que emitan las nanas que la Raspberry esté reproduciendo.

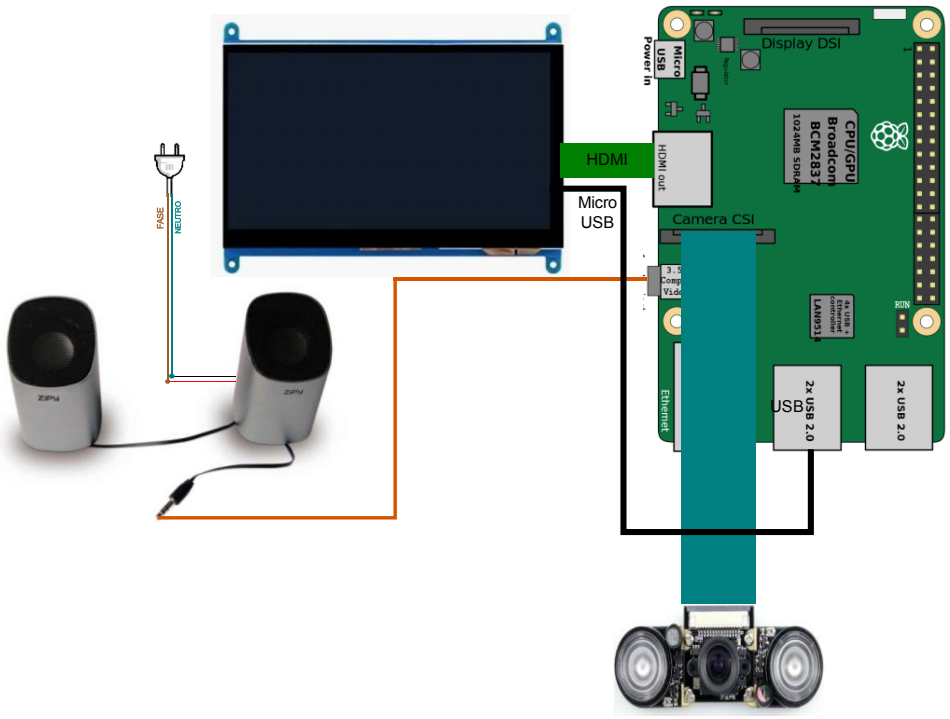


Figura 59. Conexión Raspberry – Periféricos

5.2.3 Actuadores

La gran parte de los actuadores se alimentan a 230VAC, excepto los dos motores lineales que se alimentan a 12VDC, pero todas estas alimentaciones se realizan mediante los relés, los cuales dejan pasar la tensión necesaria o no a los actuadores, según lo ordene la Raspberry.

La cuna tiene un enchufe general que, al conectarse con la toma de red, proporcionará los 230VAC que precisan estos actuadores. Los cables fase y neutro de este enchufe llegan a dos bornas de otra regleta de conexión, conforme muestra la *Figura 60*, y desde la borna de fase se cablea hasta la entrada Vcc del primero de los relés y desde éste se cortocircuita la alimentación hasta la entrada Vcc de los otros dos relés.
El neutro del enchufe se lleva a otra borna, a la cual acuden todos los neutros de los diferentes actuadores.

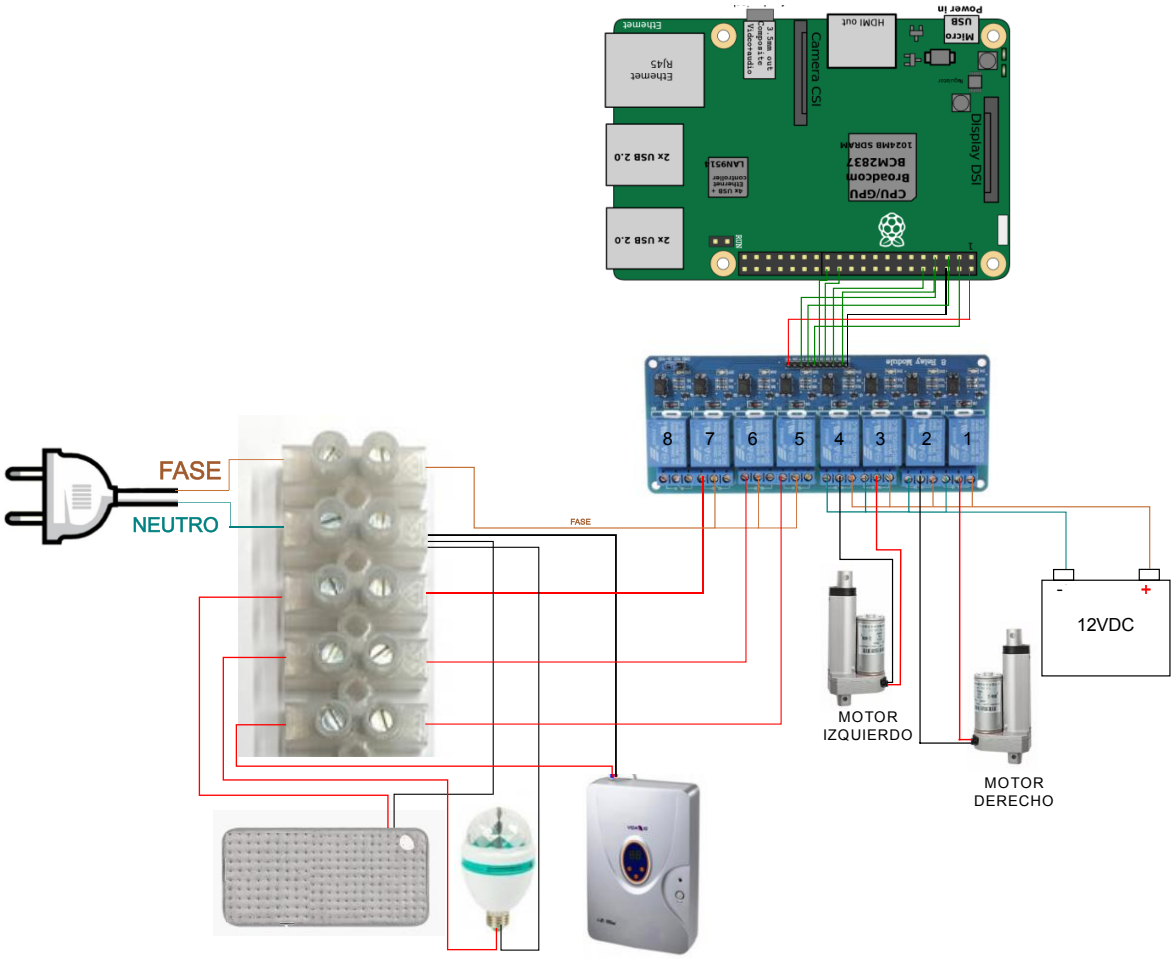


Figura 60. Conexión Raspberry – Actuadores

También mediante los relés se alimentan los motores lineales, pero en este caso con una batería de 12VDC. El positivo de la batería se conecta con la entrada Vcc de uno de los relés dedicados a motores y desde éste se cablea a las entradas Vcc de los tres relés restantes, para dotar de alimentación a los motores lineales siempre que lo demande la Raspberry mediante el cambio de estado del relé.
En función de si se alimenta el positivo o el negativo del motor, éste ascenderá o descenderá, respectivamente, conforme se indica en el apartado 3.2.2.

5.2.4 Controles

Definimos controles a la tarjeta de relés y al cerebro de este proyecto, la Raspberry Pi.
Como se ha mencionado en el apartado de sensores, la tarjeta de relés se alimenta a 5VDC, por lo que se aprovecha que la Raspberry Pi emite esta tensión y se hace pasar un cable desde las bornas de 5Vcc y GND de la Raspberry hasta los pines Vcc y GND de la tarjeta de relés.
La Raspberry Pi también necesita conectar sus GPIOs, destinadas a la activación de actuadores, con la tarjeta de relés para darle a éstos la orden de paso de alimentación para la activación del actuador que se desee.

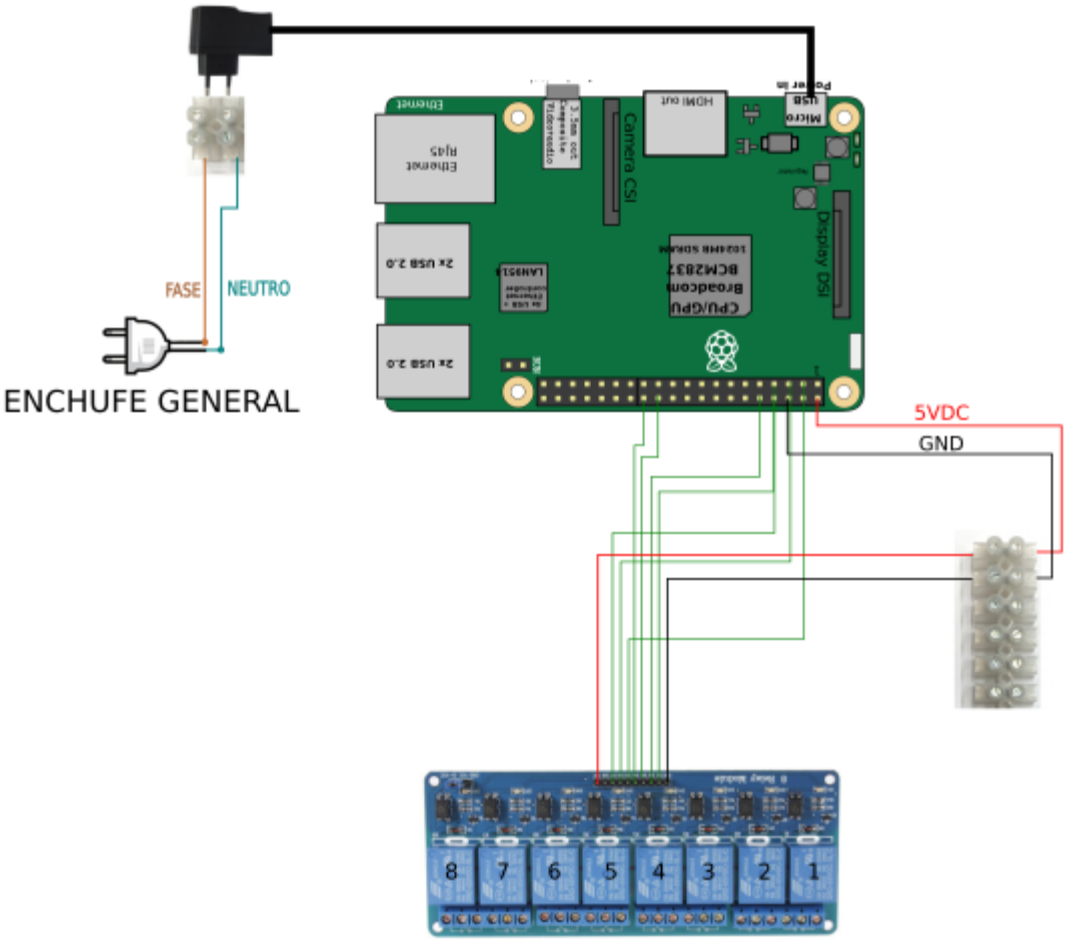


Figura 61. Conexión Raspberry - Tarjeta de relés

En cuanto a la Raspberry Pi, se alimenta a 5VDC mediante un transformador de 230VAC a 5VDC y 2,5A (Cargador de la Raspberry). Para que la Raspberry esté alimentada siempre que la cuna lo esté y pueda ofrecer el servicio deseado, se han extraído dos cables de la borna donde se encuentran los cables de fase y neutro del enchufe general y se han llevado a una regleta de dos bornas de elevado diámetro, *Figura 61*, en las cuales puede introducirse el enchufe del transformador, que estará a su otro extremo constantemente conectado al puerto microUSB de la Raspberry, cargándola, de este modo, siempre que esté conectada la cuna a la toma de red.

6. Implementación

6.1 Implementación Software

6.1.1 Instalación SO en Raspberry PI

En primer lugar, para poder hacer uso de la Raspberry es necesario instalarle el sistema operativo, que como se ha mencionado en apartados anteriores es Raspbian.
Todas las versiones de Raspberry Pi instalan el sistema operativo en una tarjeta micro-SD y por lo tanto, ese es el medio desde el que se inicia cualquier Rasberry, aunque éstas son relativamente lentas tanto en operaciones de lectura como de escritura de datos. Por esta razón, para aumentar el rendimiento de la Raspberry Pi, se puede instalar la partición del sistema operativo en una memoria USB o disco duro externo.

Aunque cabe destacar que, aunque se instale el sistema en un dispositivo USB, seguiremos necesitando la tarjeta SD para que contenga al menos la partición de arranque (boot).

Por esta razón y por el hecho de que para la aplicación de este proyecto no es necesaria esta velocidad extra, se toma la opción de instalar el sistema operativo en la tarjeta SD.
Para ello es necesario una tarjeta SD de al menos 8 GB (preferiblemente 16 GB).

En la página web principal de Raspberry Pi [21] se encuentra el apartado “Descargas”

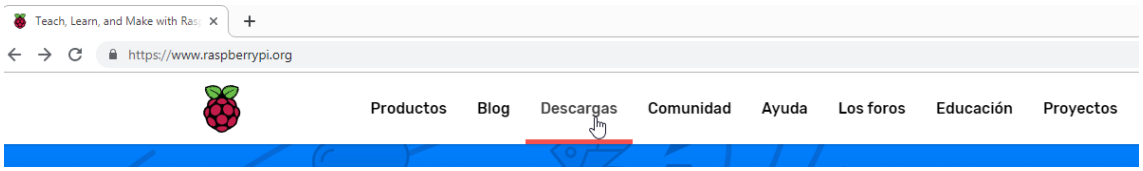


Figura 62. Web Raspberry Pi

Clicando sobre éste, se pueden descargar diferentes sistemas operativos, por ejemplo, Ubuntu mate o incluso windows 10.

Figura 63. Descargas disponibles para Raspberry

Al clicar sobre la imagen de Raspbian nos redirige a otra ventana, donde debe descargarse el archivo .zip que se muestra a continuación.

Figura 64. Descarga de Raspbian

También en esta ventana nos ofrecen una guía de instalación donde advierten sobre la necesidad de usar una herramienta de escritura de imágenes para instalar la imagen que ha descargado en la tarjeta SD; Etcher es una herramienta gráfica de escritura de tarjetas SD que funciona en Mac OS, Linux y Windows que también admite la escritura de imágenes directamente desde el archivo zip, sin necesidad de descomprimir. Para escribir la imagen con Etcher se debe seguir os siguientes pasos:

- Descargar la herramienta Etcher e instalarlo.
- Conectar al pc un lector de tarjetas SD con la tarjeta SD de la Raspberry en el interior.
- Abrir Etcher y seleccionar la imagen que desea escribirse en la tarjeta SD.
- Seleccionar la tarjeta SD en la que desea escribir su imagen.
- Hacer clic en 'Flash' para comenzar a escribir datos en la tarjeta SD.

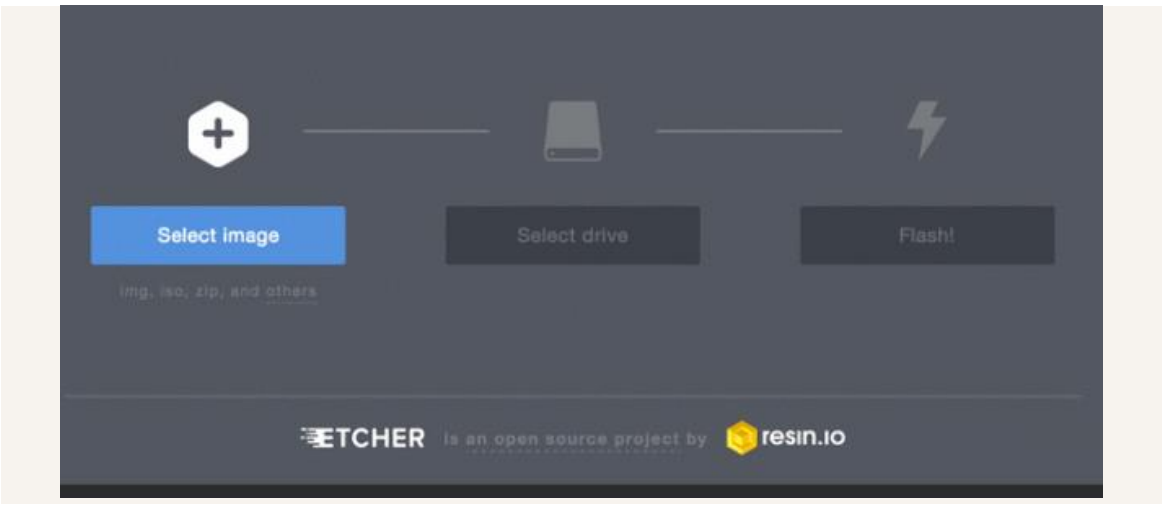


Figura 65. Escribir imagen en tarjeta SD

Cuando finalice la escritura de la imagen de Raspbian en la SD se extrae de lector de tarjetas del ordenador y se introduce en la Raspberry Pi.

6.1.2 Configuración Raspberry PI

Una vez se tenga instalado el sistema operativo Raspbian en la tarjeta SD de la Raspberry Pi, se procede a la configuración de la Raspberry Pi para que pueda adaptarse a los requisitos de este proyecto.
Los requisitos que demanda el proyecto que están relacionados con la configuración de la Raspberry Pi son los siguientes:

- Posibilidad de capturar imágenes con la cámara de Raspberry, siempre que se desee y por un tiempo ilimitado.
- Posibilidad de conexión a la Raspberry mediante VNC desde cualquier dispositivo.

Para informar a la Raspberry de estas posibles actuaciones es necesario activar ambos campos siguiendo los siguientes pasos:
En primer lugar, se pulsa sobre "*Preferences*" y posteriormente sobre "*Raspberry PI confiuration*".

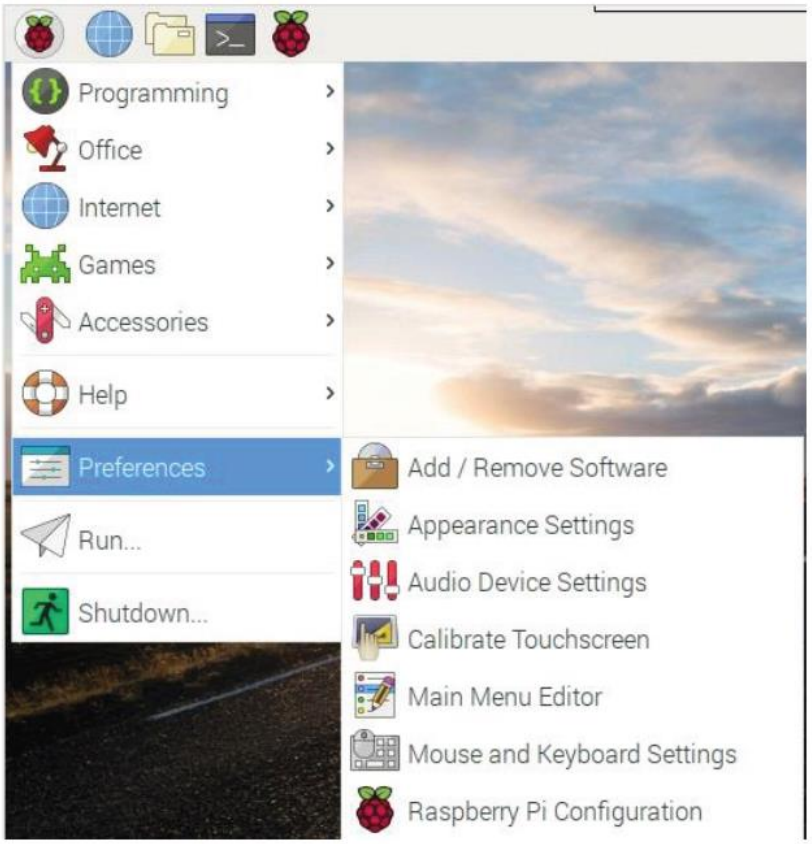


Figura 66. Ajustes Raspberry Pi

Lo que nos lleva al menú de configuración de la Raspberry, que se utilizará en más ocasiones.
En este caso, se pulsa sobre la pestaña "*Interfaces*" y se habilita la cámara y el control remoto mediante VNC, *Figura 67*.

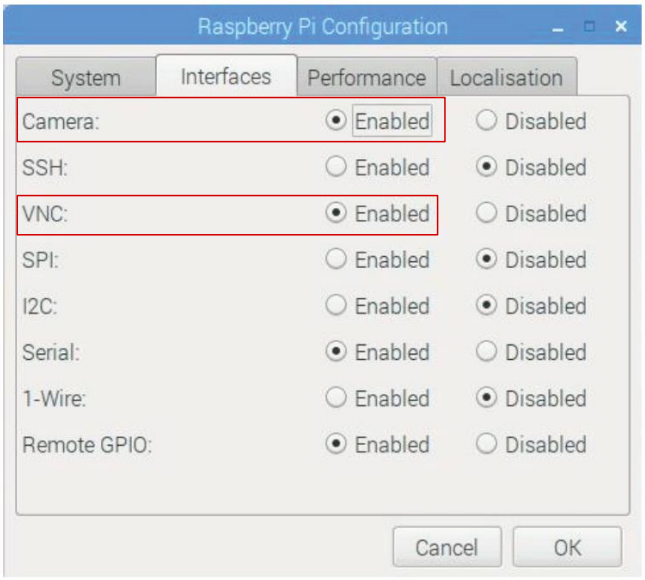


Figura 67. Habilitación Cámata y VNC

En caso de no estar habilitado también debería habilitarse el campo de los puertos GPIO.

6.1.3 Configuración pantalla táctil

Como se ha comentado a lo largo de la memoria, la interfaz programada y volcada en la Raspberry Pi estará constantemente mostrada en una pantalla ubicada en la cuna, siempre que ésta esté encendida.

Mediante el conector HDMI se logra mostrar las imágenes en la pantalla y mediante un puerto USB de la Raspberry se consigue controlar el táctil de la pantalla.

En primer lugar, debe configurarse la resolución de pantalla, para ello se vuelve al menú de Configuración de Raspberry Pi, *Figura 67*, y, en la pestaña de *System*, se pincha sobre el botón de Modificar Resolución y se seleccionar la resolución de la pantalla que se desee.

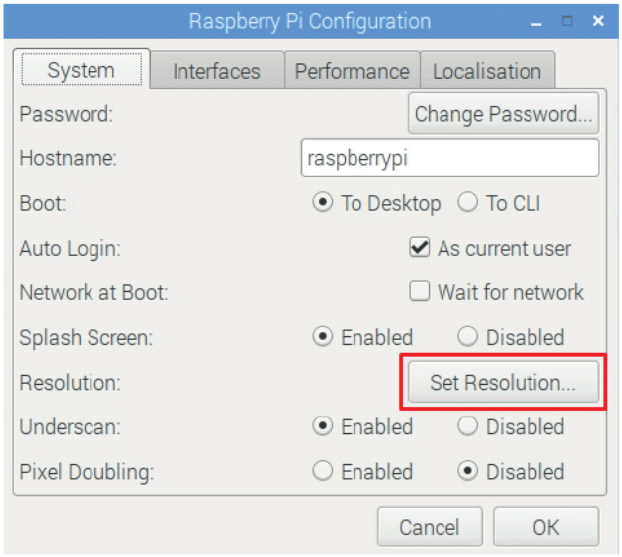


Figura 68. Resolución pantalla

6.1.4 Creación del archivo ejecutable jar

El código de programación, una vez realizado, debe volcarse en la Raspberry Pi para que ésta lo ejecute; el archivo que debe enviarse a la Raspberry es un archivo con extensión .jar, el cual genera la herramienta de desarrollo IntelliJ, siguiendo los pasos siguientes.

En primer lugar, para crear por primera vez un archivo .jar se deben seguir los siguientes pasos; En la sección de ajustes del proyecto, en el apartado de dispositivos (*Artifacts*), se pulsa sobre el botón derecho y aparece la opción de crear un archivo .jar, inicialmente vacío.

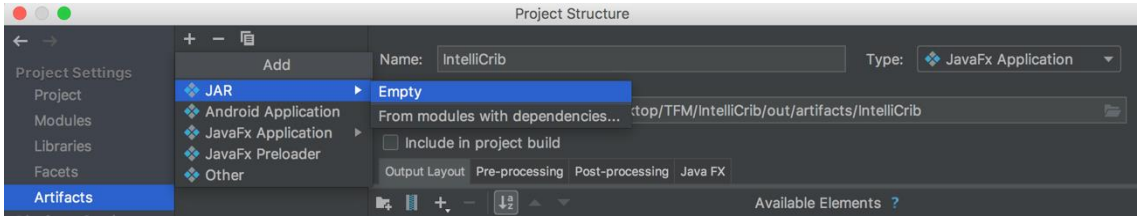


Figura 69. Creación archivo jar

Una vez creado, se le asigna un módulo de salida.

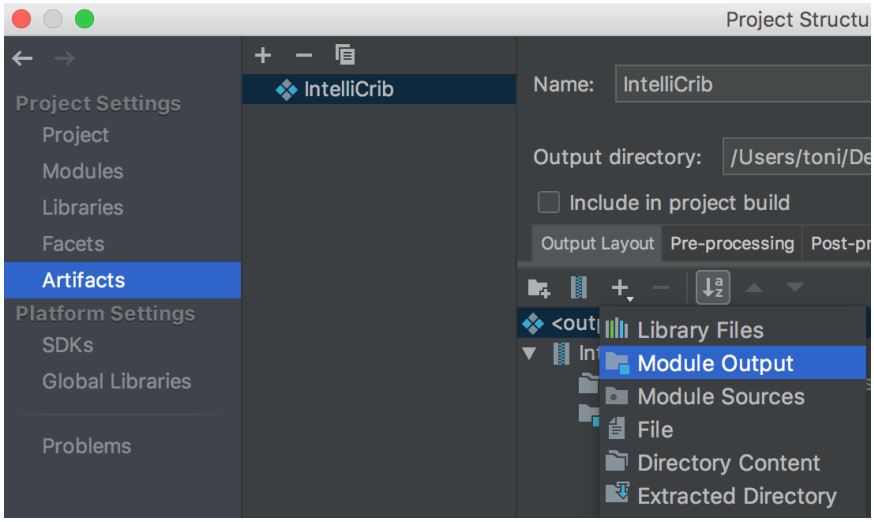


Figura 70. Módulo de salida archivo jar

Se selecciona entonces el nombre del proyecto, llamado IntelliCrib, el cual alberga el código de programación redactado para este proyecto.

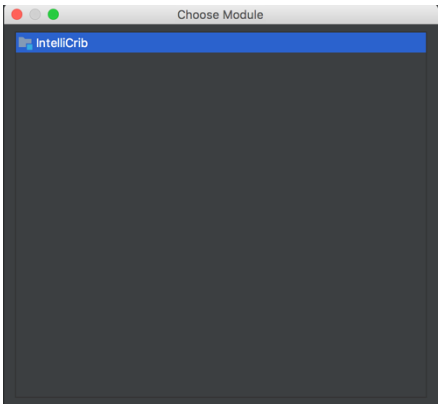


Figura 71. Asignación de proyecto

Entonces se añade el contenido del directorio del proyecto IntelliCrib, que está constituido por las librerías que se han añadido en el apartado de librerías anteriormente.

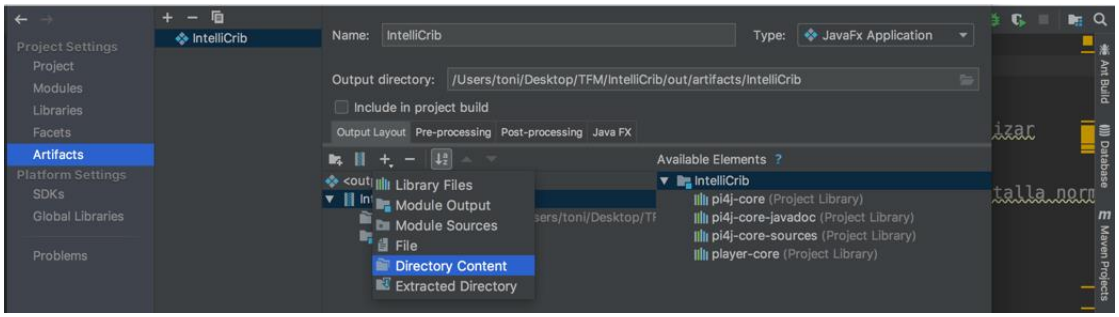


Figura 72. Contenido del directorio

En segundo lugar, debe crearse el archivo MANIFEST, necesario para la construcción del archivo jar, siguiendo los siguientes pasos:
Dentro de la carpeta de recursos del proyecto, llamada src, se crea un paquete, *Figura 73*.

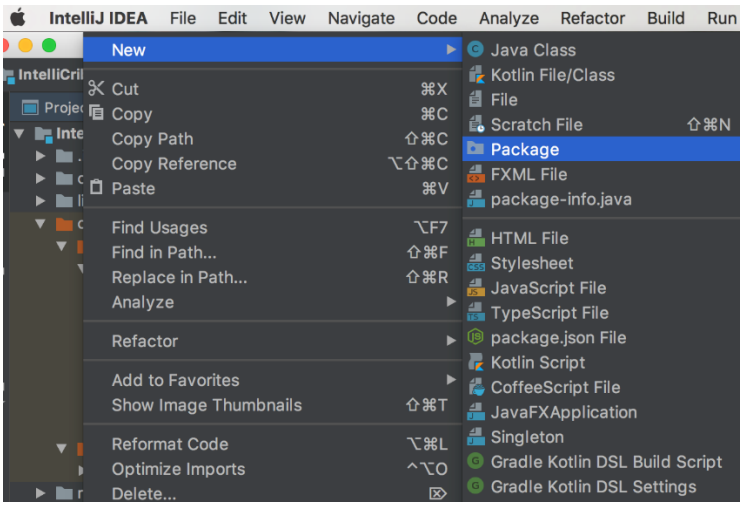


Figura 73. Nuevo paquete

Dentro del cual, se crea un archivo de texto.

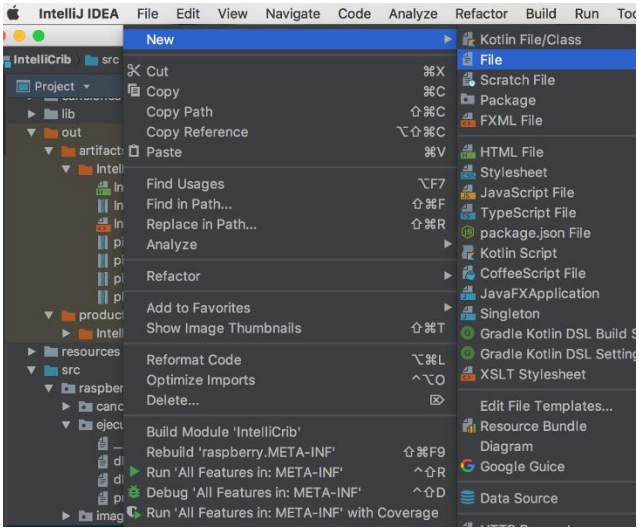


Figura 74. Nuevo documento de texto

Al que se le nombra “MANIFEST.MF”

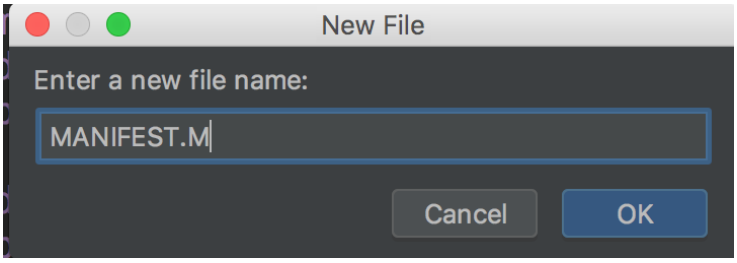


Figura 75. Archivo MANIFEST

El contenido del archivo Manifest en un archivo JAR creado con la versión 1.0 del Java Development Kit, *Figura 76*.

Figura 76. Contenido del archivo MANIFEST

Los archivos de un programa de ordenador deben tener un documento de manifiesto que describa el nombre, número de versión, licencia y los archivos que forman parte del programa.

En la plataforma Java, un archivo *manifest* es un archivo específico, contenido en un archivo JAR, que se usa para definir datos relativos a la extensión y al paquete. Si se pretende usar el archivo JAR como ejecutable, el archivo de *manifest* debe especificar la clase principal de la aplicación.

En este caso, el documento *manifest* especifica la versión 1.0 de Java con la que ha sido creado, también define la ubicación en la que se encuentran las librerías vistan anteriormente y, finalmente, dependiendo de la función prevista del archivo JAR, es posible que el *manifiest* predeterminado deba modificarse.
Si el archivo JAR se crea solo para archivar, el archivo MANIFEST.MF no precisa modificaciones. La mayoría de los usos de los archivos JAR van más allá del simple archivado y requieren información especial en el archivo de manifiesto. Dependiendo de las características que requiera el archivo JAR el encabezado del archivo de manifiesto debe albergar diferente información:

- Si una aplicación está empaquetada en un archivo JAR, se debe informar a la Máquina Virtual de Java cuál es el punto de entrada a la aplicación. Un punto de entrada es cualquier clase con un método public void main. En este caso, se define como entrada el método raspberry.Main, como puede observarse en el apartado Main-Class de la *Figura 76,* esto define por qué método debe comenzar a ejecutar el código.
- Si una aplicación contiene extensiones de descarga. Las extensiones de descarga son archivos JAR a los que hacen referencia los archivos de manifiesto de otros archivos JAR.

En este proyecto tenemos un único archivo jar, por lo que no es necesario definir este apartado en el archivo *manifest*.

El archivo *manifest* debe guardarse en una carpeta llamada META-INF, que es un meta directorio de Java, que contiene datos de configuración de la aplicación.

Se realiza desde la pestaña de ajustes del proyecto, *Figura 77*.

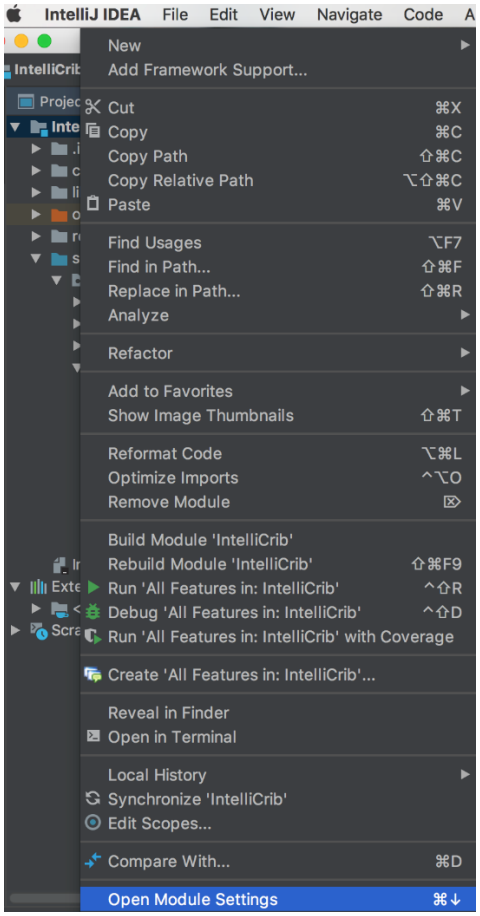


Figura 77. Ajustes IntelliJ

Creando un directorio llamado META-INF, *Figura 78 y 79*.

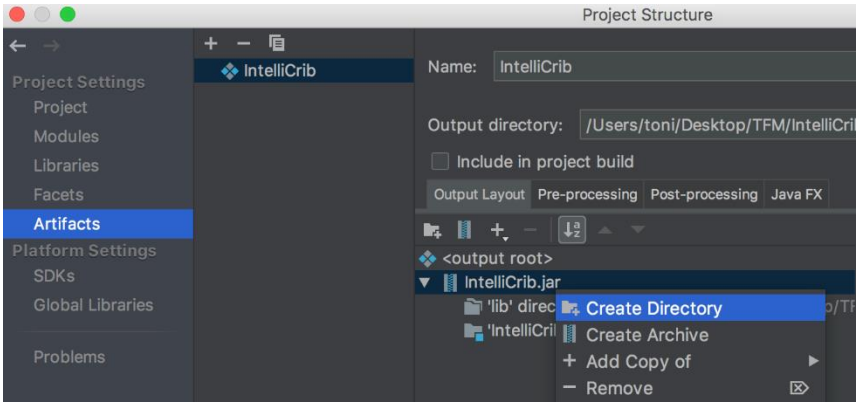


Figura 78. Creación de directoro

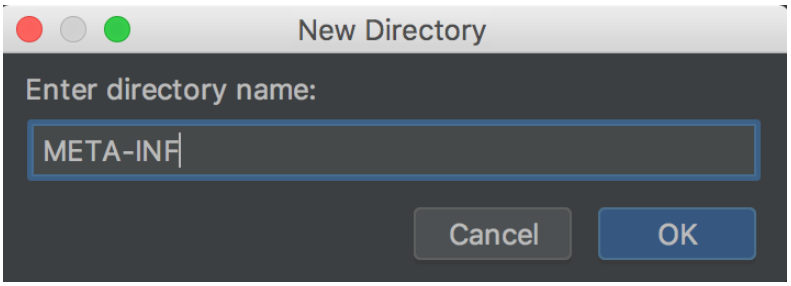


Figura 79. Caracterizaión del directorio

Finalmente, señalando la carpeta META-INF creada, se añade el fichero manifest.

Una vez se haya creado por primera vez el archivo JAR, siempre que se modifique el código, deberá reconstruirse, *Figura 80*.

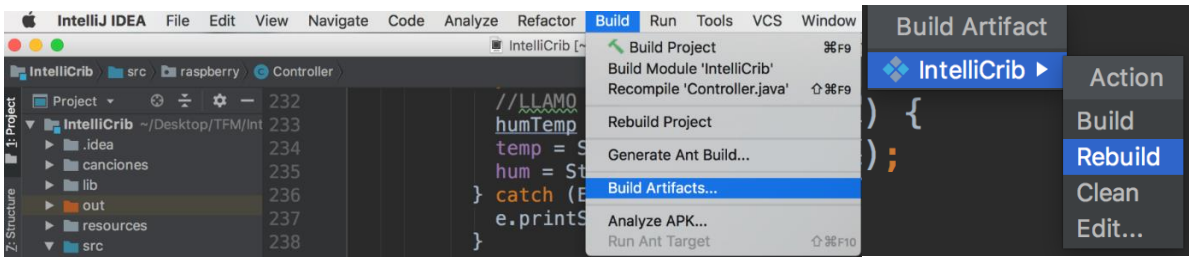


Figura 80. Reconstruir JAR

El archivo JAR se encuentra en el directorio out/artifacts/IntelliCrib (nombre del proyecto), *Figura 81*.

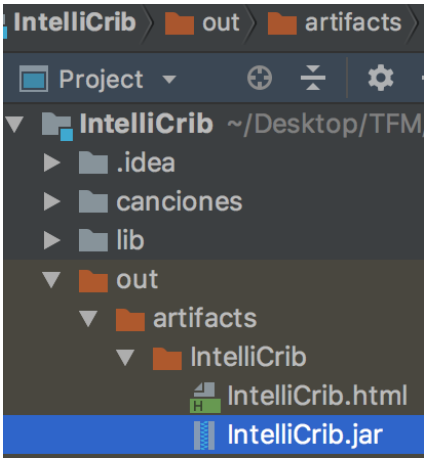


Figura 81. Directorio archivo JAR

6.1.5 Conexión PC con Raspberry PI

Una vez creado el archivo ejecutable JAR, se puede proceder a volcarlo en la Raspberry Pi para que lo ejecute.

En primer lugar, debe conocerse la IP de la Raspberry Pi. La dirección IP es un número que identifica a un dispositivo en red, ya sea ordenador, tableta, portátil o teléfono inteligente, que utilice el protocolo IP (*Internet Protocol*), que corresponde al nivel de red del modelo TCP/IP.

La dirección IP cambia muy a menudo debido a cambios en la red o porque el dispositivo encargado dentro de la red de asignar las direcciones IP, decida asignar otra IP, mediante el protocolo DHCP. A esta forma de asignación de dirección IP se le denomina también dirección IP dinámica; por esta razón cada vez que se quiera conectar con la Raspberry se debe mirar la IP que se le haya asignado, esto se realiza entrando en la configuración de *router*.

Introduciendo en el navegador la IP del *router*, que siempre es 192.168.1.1:

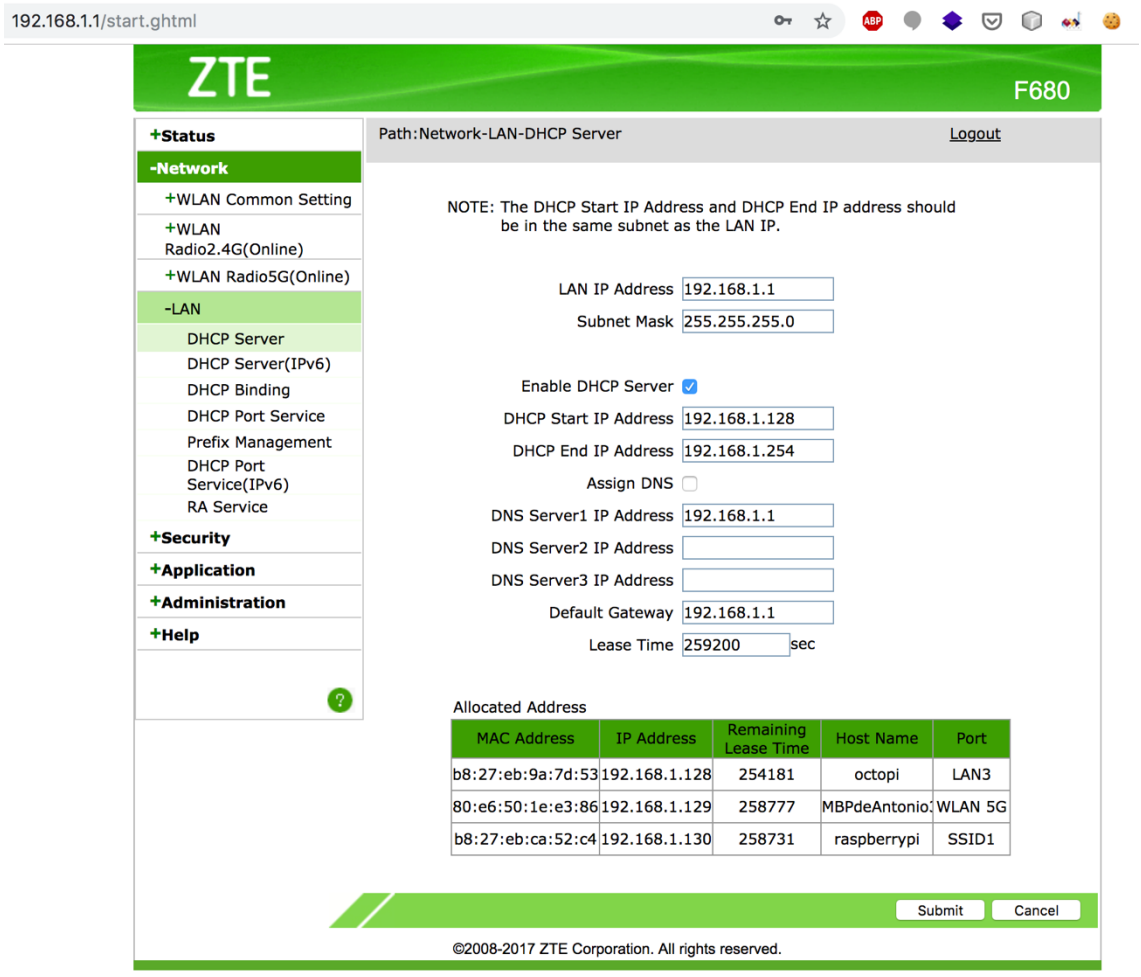


Figura 82. IP dinámica Raspberry Pi

Aquí aparece la IP de todos los dispositivos conectados a este *router*, entre ellos la Raspberry Pi, que, como puede verse en la *Figura 82,* tiene asignada la dirección IP: 192.168.1.130

Sabiendo esto se procede a la conexión del PC con la Raspberry Pi, según los pasos siguientes:

En primer lugar, se abre un terminal en el que se escribe la dirección IP de la Raspberry Pi.

- ssh pi@192.168.1.130

En este terminal ya estamos dentro de la Raspberry.
Una vez se ha establecido conexión con la Raspberry PI, se abre otro terminal en el que se escribe la dirección para acceder a la carpeta del PC donde está guardado el archivo JAR que se quiere desplazar a la Raspberry, llamado IntelliCrib.

- cd /Users/Moni/Desktop/TFM/IntelliCrib/out/artifacts/IntelliCrib

Con el comando cd nos ubicamos donde se encuentra el archivo InterlliCrib.jar en el PC, entonces se procede a copiarlo y mandarlo a la Raspberry PI mediante el comando scp.

- scp IntelliCrib.jar pi@192.168.1.130:/home/pi/Downloads

En la Raspberry se guarda el archivo en la dirección home/pi/Downloads.
Entonces retornamos al primer terminal, el que está dentro de la Raspberry, y se procede a ejecutar el archivo JAR.
Entramos en la carpeta Downloads de la Raspberry mediante el comando cd.

- pi@raspberrypi:~ $ cd Downloads/

Y mediante el comando ls vemos los archivos que contiene esta carpeta, asegurándonos que se encuentra el archivo IntelliCrib.jar.

- pi@raspberrypi:~/Downloads $ ls

Entonces se procede a ejecutarlo mediante el siguiente comando:

- pi@raspberrypi:~/Downloads $ java -jar IntelliCrib.jar

6.1.6 Conexión a escritorio remoto

El programa que se utiliza para poder observar la interfaz gráfica que se muestra en la pantalla de la cuna en cualquier otro dispositivo, como por ejemplo el móvil, es VNC (*Virtual Network Computing*); es un programa de *software* libre basado en una estructura cliente-servidor que permite controlar remotamente otro ordenador. Transmite los eventos ejecutados desde un dispositivo a otro, transmitiendo las actualizaciones de la pantalla gráfica en la otra dirección (pe.: Cambiar el punto de consigna de temperatura), a través de una red. De esa forma se consigue que, desde cualquier dispositivo, ya sea tableta, móvil u ordenador e independientemente del

sistema operativo que dispongan dichos dispositivos, se pueda acceder remotamente a la pantalla que se ubica en la cuna.

Cabe destacar las dos aplicaciones que se necesitan para la comunicación mediante VNC; de un lado está la aplicación *VNC Server* para el equipo que se desea controlar remotamente, que en este caso es la Raspberry Pi y, del otro lado, una aplicación *VNC Viewer* para el equipo o dispositivo móvil desde el cual desea ejercer el control remoto, en este caso el teléfono móvil o tablet de los padres.

VNC Server captura el escritorio de la Raspberry Pi en tiempo real, en este caso lo que se está mostrando por la pantalla táctil situada en la cuna, y lo envía a *VNC Viewer* para su visualización. *VNC Viewer* recopila su entrada (ratón, teclado o tacto), en este caso el tacto de la pantalla del móvil o Tablet, y la envía para que *VNC Server* la inserte y *consiga* de forma efectiva el control remoto.

Para disponer de esta herramienta en el proyecto, en primer lugar, se ha creado una cuenta en la página web RealVNC [22]. Una vez registrada, se descarga la aplicación de *VNC Server* en la Raspberry Pi y se procede a realizar la identificación con los datos de la cuenta creada en la página web RealVNC.

En el último paso de la instalación debe seleccionarse "*Direct and cloud conectivity*" para poder establecer una conexión en la nube desde cualquier dispositivo que ejecuta *VNC Viewer*, aunque que no se ubique en la misma red de área local.

Para la versión gratuita de VNC el número máximo de dispositivos que pueden conectarse a un mismo servidor son cinco, aunque con la misma cuenta es posible instalar *VNC Server* en distintos equipos que quieran gobernarse remotamente y, al descargar *VNC Viwer*, se seleccionará a cuál de ellos se desea conectarse remotamente. A cada servidor de VNC se le puede adjudicar una contraseña distinta, la cual se demandará cuando se quiera acceder a alguno de ellos desde un dispositivo con *VNC Viwer*.

Del lado cliente (Tablet, móvil o PC), se debe descargar *VNC Viewer* y registrarse con los datos de la cuenta creada. Una vez realizado el registro, aparecen los equipos remotos que se han registrado mediante *VNC Server* con la misma cuenta, entre los que se encuentra la Raspberry Pi en la que corre este proyecto, a la que se le ha llamado IntelliCrib, *Figura 83*.

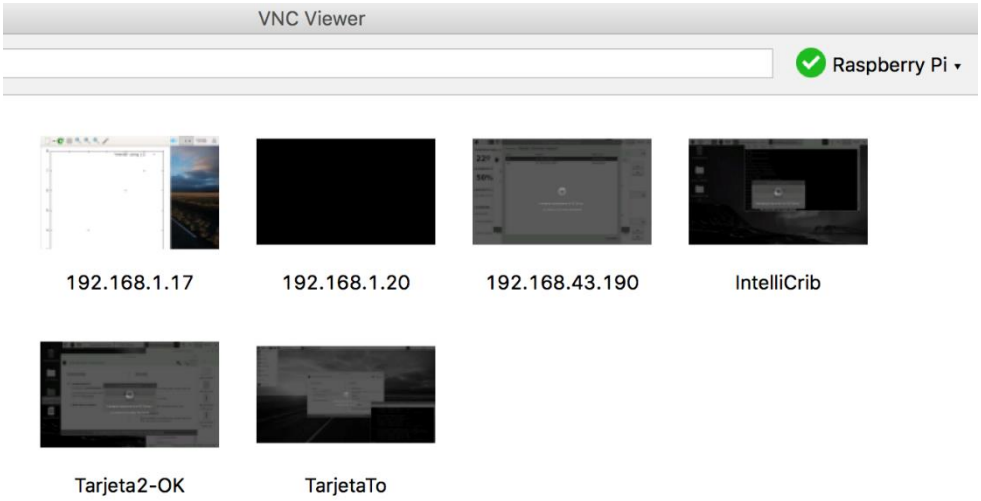


Figura 83. Servidores VNC bajo la misma cuenta

Entonces se selecciona el escritorio IntelliCrib y aparece una ventana como la de la *Figura 84,* en la que se ha de realizar la autenticación a la Raspberry Pi, mediante los credenciales que se han asignado a este servidor en concreto.

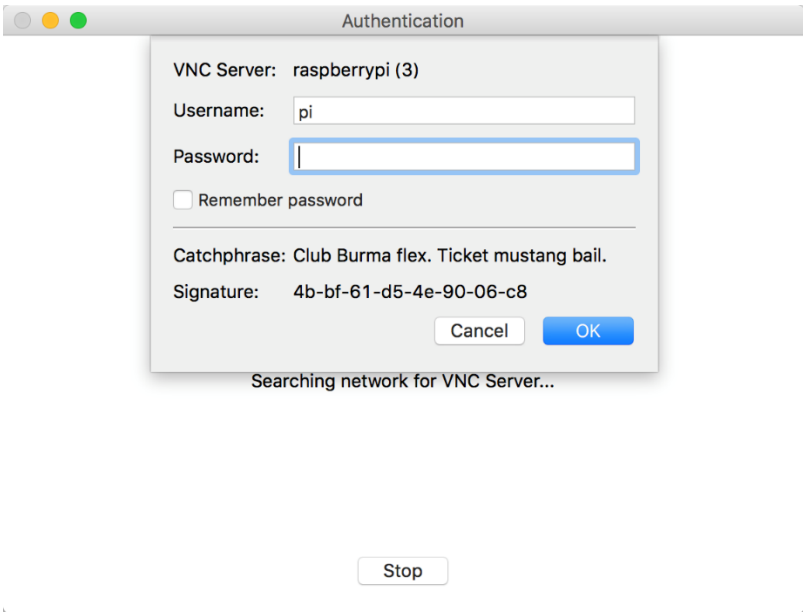


Figura 84. Conxión a servidor IntelliCrib

Una vez establecida la conexión con el servidor se puede observar el escritorio de la Raspberry Pi, como se muestra en la *Figura 85*.

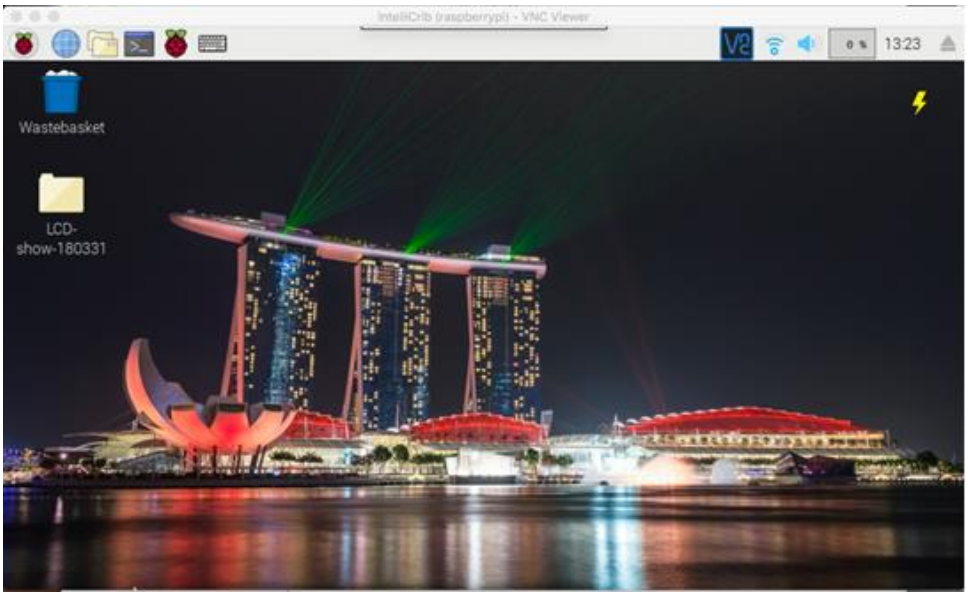


Figura 85. Escritorio Raspberry Pi

Teniendo acceso al escritorio de la Raspberry Pi, cabe la opción de ejecutar el archivo JAR desde el terminal de este escritorio; accediendo a la carpeta Downloads, mediante el comando cd, donde se encuentra el archivo ejecutable IntelliCrib.jar.

- pi@raspberrypi:~ $ cd Downloads/
- pi@raspberrypi:~/Downloads $ java -jar IntelliCrib.jar

Y ejecutándolo mediante el comando java –jar, de la misma forma que se ha realizado en el apartado anterior desde el terminal adicional del PC.

Una vez se ejecuta el programa, aparece la interfaz gráfica en la pantalla de la cuna y, por tanto, en la pantalla del dispositivo VNCViwer, ya sea un pc o el móvil del usuario que quiere interactuar con la cuna.

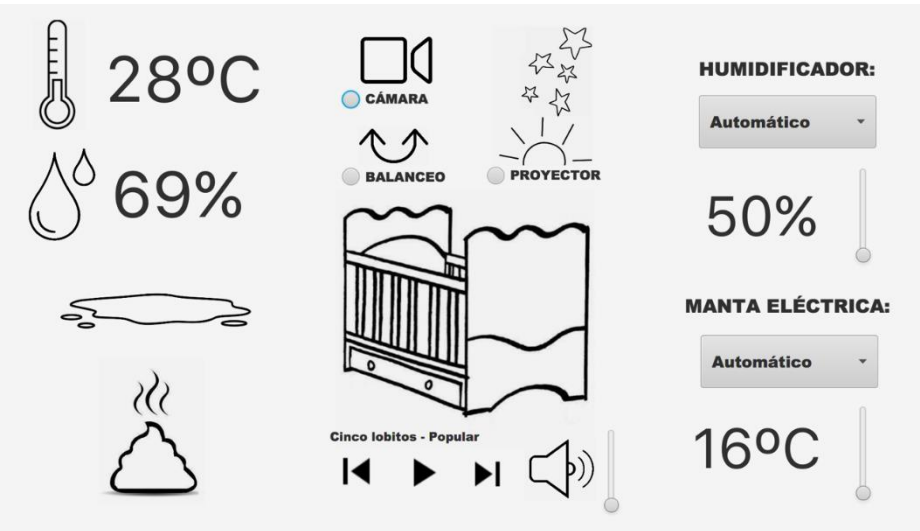


Figura 86. Interfaz gráfica mostrada en pantalla táctil

6.1.7 Ejecución aplicación al arranque

La idea es que, tanto la pantalla táctil como el dispositivo que se conecte mediante VNC a la Raspberry Pi, muestre la interfaz gráfica desde el primer momento, siempre que esté encendida la cuna, para que el usuario pueda interactuar con ella siempre que lo desee. Para ello debe configurarse la Raspberry Pi para que el usuario no tenga que ejecutar ningún comando para iniciar la aplicación y, en caso de que se produzca una interrupción de energía eléctrica en la casa, cuando vuelva la luz, la aplicación se ejecute automáticamente.
Para ello, se ha decidido configurar los ficheros de inicio de Raspbian para que ejecute la aplicación al iniciarse la Raspberry Pi.

En primer lugar, se crea un fichero ejecutable, el cual se ha llamado "ejecutable.sh, escribiendo el comando sudo nano ejecutable.sh en el terminal de la Raspberry Pi.

En este fichero se introduce el comando cd /home/pi/Download/IntelliCrib/, que busca el fichero dentro del dominio de la Raspberry y, posteriormente, con el comando sudo java -classpath '.:classes:*classes:/opt/pi4j/lib/*' -jar app.jar raspberry se permiten la ejecución de la aplicación.

Para que el fichero sea ejecutable se le debe dar permiso de ejecución, introduciendo en el terminal el comando sudo chmod +x ejecutable.sh.

Posteriormente se añade al final del fichero ".bashrc" mediante sudo nano .bashrc y para que se ejecute el ejecutable donde está la línea de ejecución de la aplicación, se escribe bash ejecutable.sh.

Por último, se modifica el fichero "autostart" que se ubica en la ruta /.config/lxsession/LXDEpi/autostart, mediante sudo nano/etc/xdg/lxsession/LXDE-pi/autostart, introduciendo la siguiente línea @lxterminal en el fichero.

De esta forma se logra que, cuando arranque el sistema operativo en la Raspberry Pi, se ejecute el terminal que se ha escrito, el cual ejecuta el fichero "ejecutable.sh" y dicho fichero ejecuta la aplicación IntelliCrib, dejando en la pantalla la interfaz gráfica para que el usuario interactúe con ella.

6.2 Implementación Hardware

6.2.2 Montaje hardware

En este apartado se expone el conexionado general, a partir del conexionado específico de cada grupo, expuesto en el capítulo anterior, y se describe cómo se va a llevar a cabo el montaje de éste en la estructura de la cuna.

6.2.2.1 Bloque baja tensión

El montaje se ha realizado en dos grandes bloques; el primer bloque, que se ubica en una caja visible y cercana a la pantalla táctil, se compone por el conjunto de elementos de baja tensión.
Este bloque alberga la Raspberry con su conexión a la cámara y a la pantalla, así como la alimentación e intercambio de datos con los sensores, los cuales están ubicados en el exterior de la caja de baja tensión, alrededor de la cuna.
El bloque de baja tensión se conecta con el de alta para alimentar la tarjeta de relés, así como para pasarle a ésta las órdenes de activación de sus relés y precisa del bloque de alta tensión la alimentación de la Raspberry Pi.
El bloque de baja tensión presenta mayormente una tensión de 5VDC proporcionados por la Raspberry.

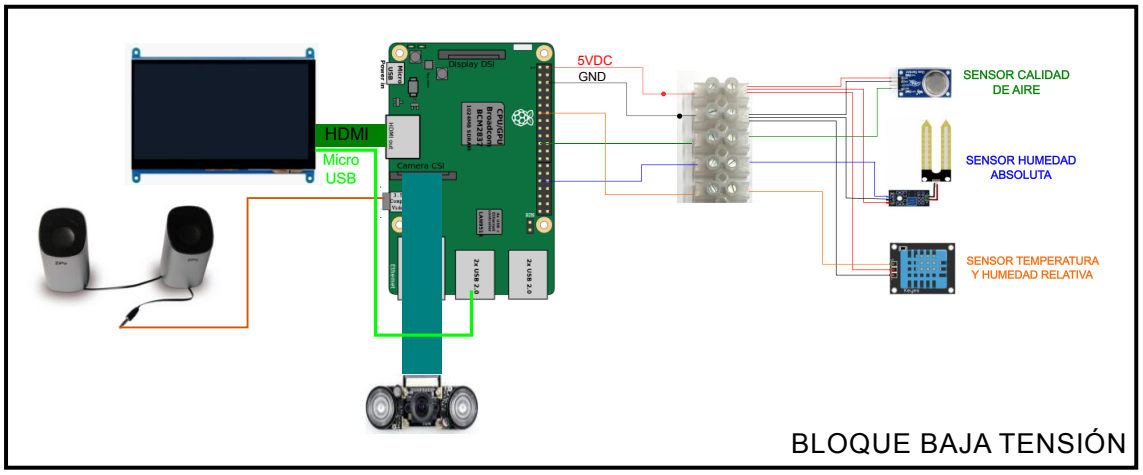
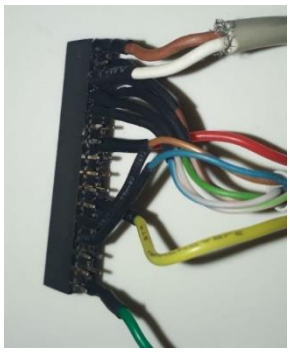


Figura 87. Esquema bloque baja tensión

Este bloque alberga la Raspberry, la cual va conectada directamente a los periféricos, como son los altavoces, la cámara y la pantalla. Éste último, marca la posición de la caja de baja tensión, debido a que la pantalla debe estar en un sitio visible y accesible por el usuario. Esto obliga a ubicar la caja de baja tensión en un lugar accesible que permita la conexión directa de la pantalla con la Raspberry Pi.
En este mismo bloque se encuentra la alimentación de los sensores, extraída de la misma Raspberry, así como la transacción de datos de éstos que son llevados desde el pin de datos de los sensores a diferentes GPIOs de la Raspberry.
También de las GPIOs de la Raspberry, salen las órdenes de activación de los relés para alimentar los actuadores pertinentes.



Para realizar todas estas conexiones de salida de la Raspberry Pi se ha optado por un conector que presenta por un lado huecos para encajar con los pines de la Raspberry y por el otro orquillas en las que se sueldan los cables que van a los sensores, a los relés y ofrecen alimentación, protegiendo con cale termoretráctil las soldaduras, Figura 88.

Figura 88. Conector Raspberry Pi

Por otra parte, se ha destinado un cable de tres hilos para cada uno de los sensores, a través de los cuales se alimentan los sensores (con Vcc y GND) y se intercambian datos, estos cables se ha soldado a un conector de tres pines para poder conectarlo a los pines de los sensores con facilidad*, Figura 89*.

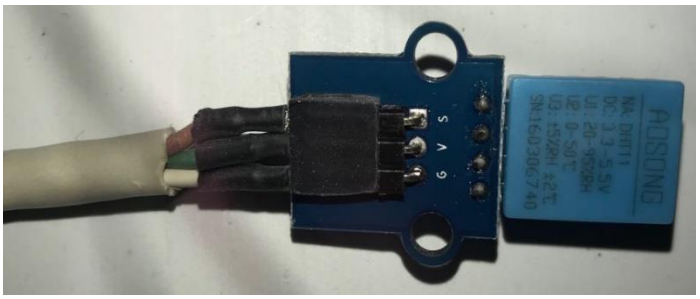


Figura 89. Conector sensores

Finalmente, en la caja de baja tensión se han realizado agujeros donde atornillar la Raspberry Pi, la borna de conexión y las torretas que sujetan la pantalla táctil.

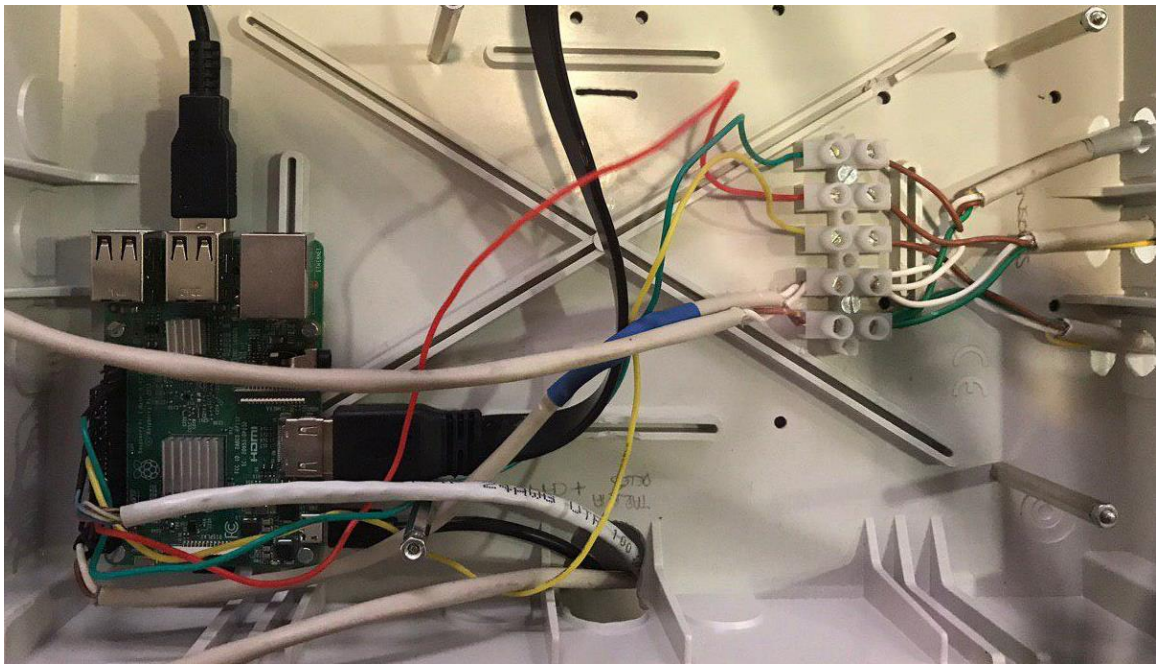


Figura 90. Caja baja tensión sin pantalla

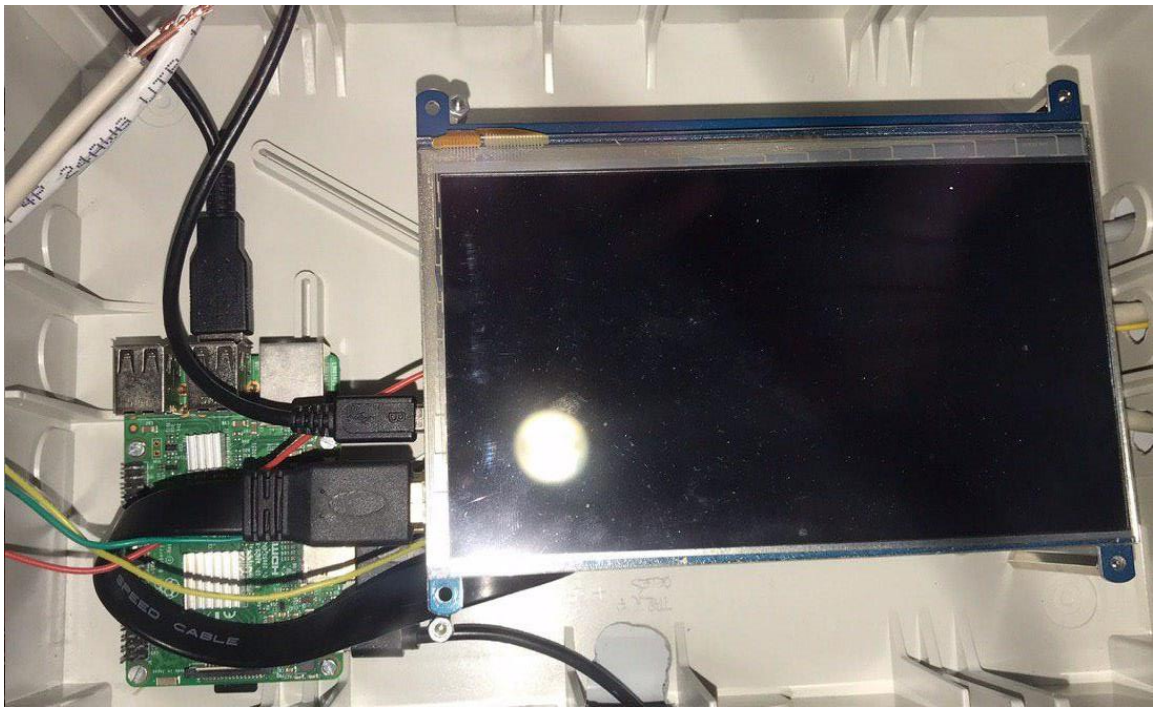


Figura 91. Caja baja tensión con pantalla

6.2.2.1 Bloque alta tensión

En cuanto a la caja de alta tensión, se ha denominado así por el hecho de alimentar a los actuadores a 230VAC, mediante tres relés, ya que a esta caja llegan los 230VAC de la toma de red. Los cuatro relés restantes son los que van a los dos motores lineales alimentados mediante una batería de 12VDC.

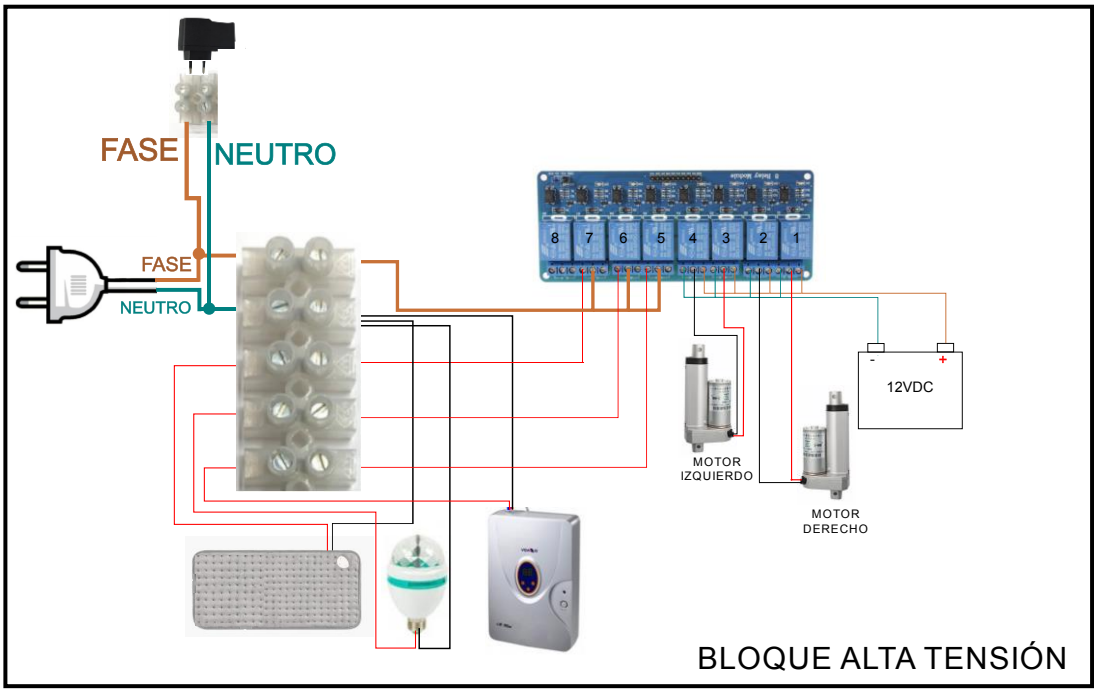


Figura 92. Esquema bloque alta tensión

La tensión de red que llega a la caja de alta tensión se conecta a una regleta de conexiones, como se ha expuesto en el apartado anterior, y desde ésta la fase se lleva al pin de alimentación de los relés conectados a los actuadores de 230V, y éstos se puentean hasta los pines de Vcc del resto de relés conectados al resto de actuadores a 230V.
El neutro de la toma de red se conecta con el neutro de todos los actuadores a 230V mediante otra borna de la regleta de conexiones.
Existen dos motores lineales alimentados a 12V, uno se situará a la derecha del eje transversal de la cuna y el otro a la izquierda.

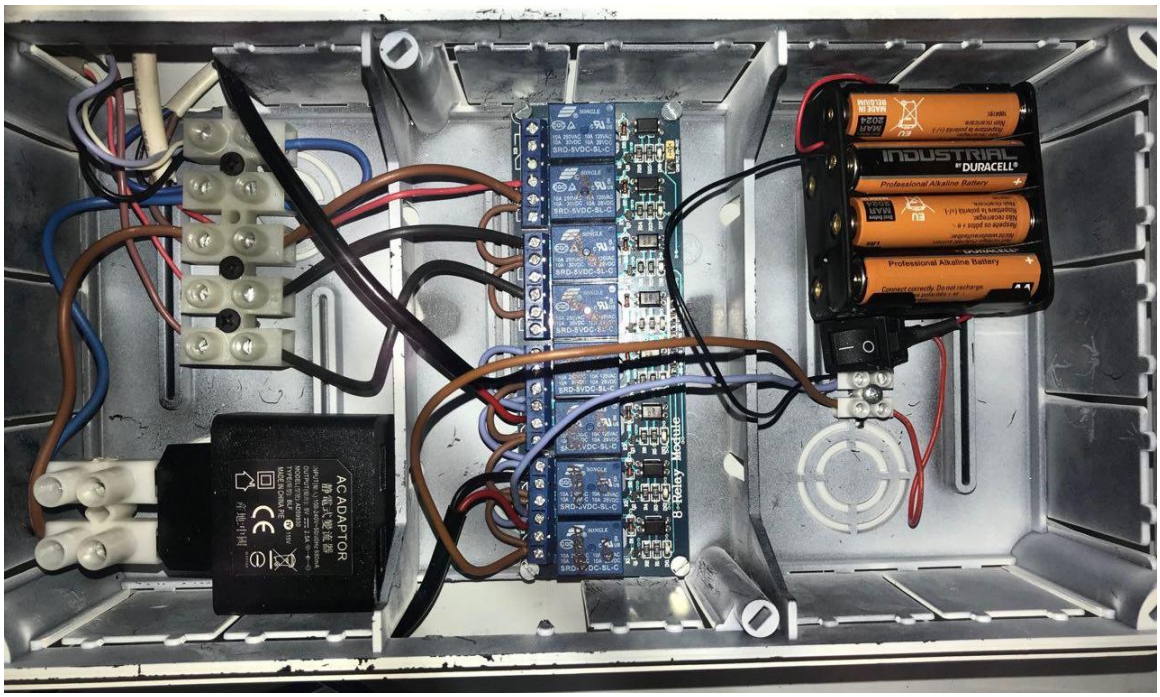


Figura 93. Caja alta tensión

De la Raspberry Pi sale hacia la tarjeta de relés un cable de 7 hilos. Cada uno de ellos es una salida digital para los actuadores. Estos hilos se conectarán tanto a los pines de la Raspberry Pi como a los pines de la tarjeta de relés mediante un conector, al que se han soldado ordenadamente los hilos y se ha cubierto la soldadura con cable termoretractil.

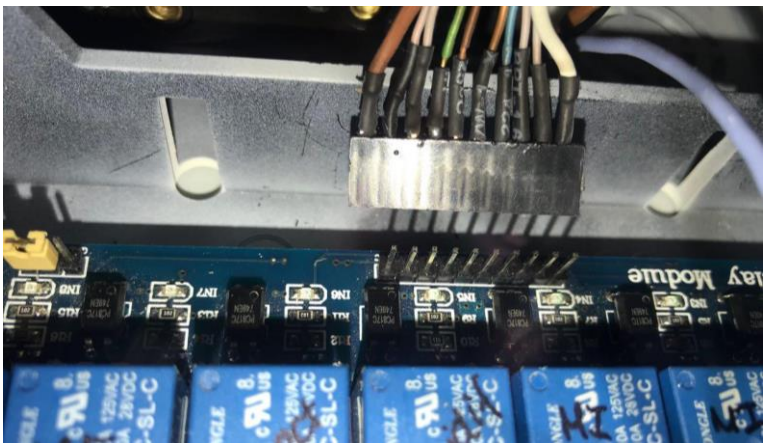


Figura 94. Conector tarjeta de relés

En definitiva, el conjunto de conexiones del proyecto, aunando el bloque de alta y de baja tensión queda como sigue, *Figura 95*.

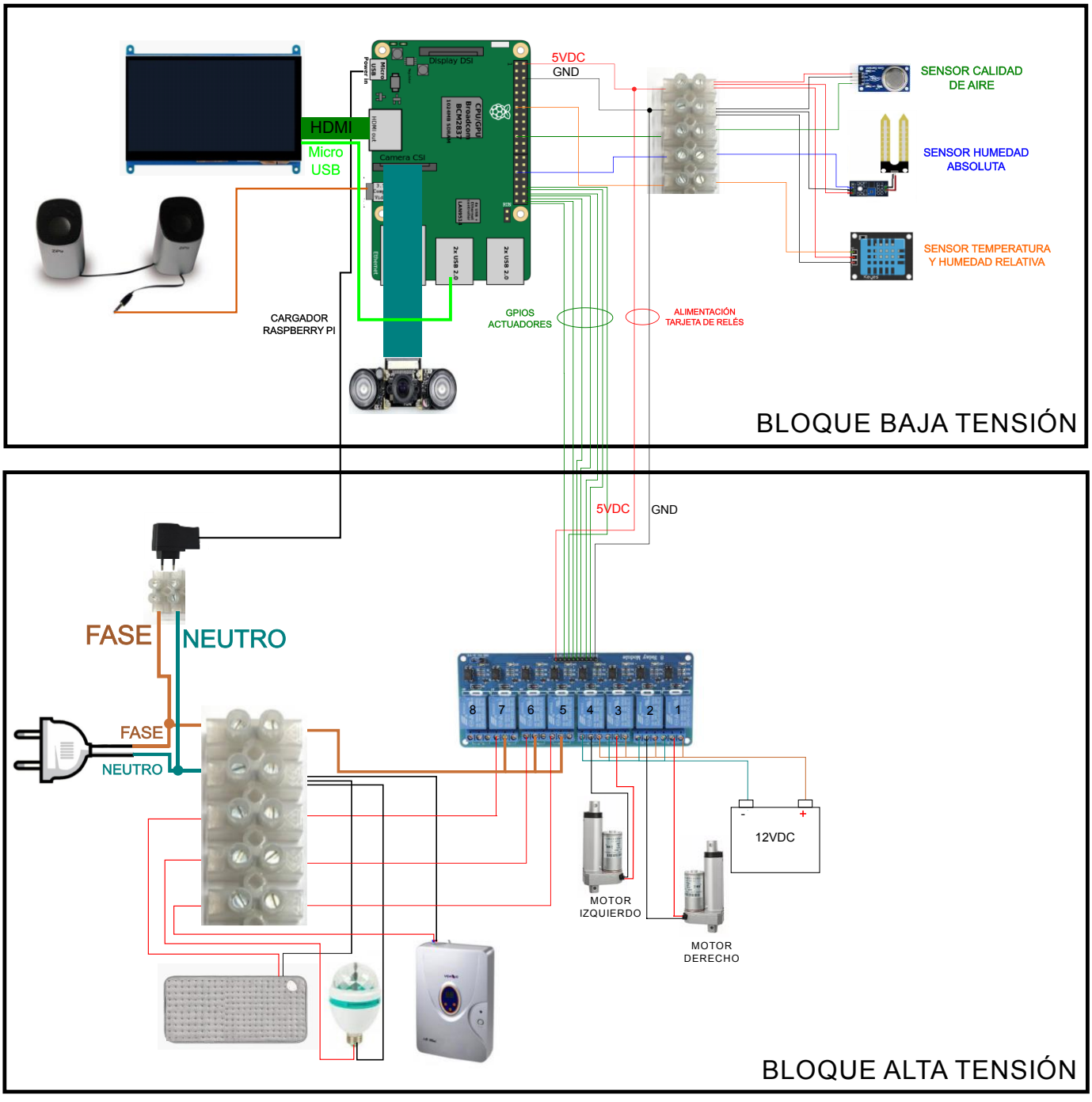


Figura 95. Conexión entre bloques

El bloque de alta tensión alimenta a la Raspberry PI, mediante un transformador, mientras que el bloque de baja tensión ofrece la alimentación de 5VDC desde la Raspberry a la tarjeta de relés, así como las salidas digitales correspondientes a los actuadores, gobernados mediante la tarjeta de relés.

6.2.3 Montaje mecánico

En este apartado se va a exponer el montaje de todas las partes que componen IntelliCrib sobre la estructura de la cuna.
Como se ha explicado anteriormente, la caja de baja tensión alberga la Raspberry Pi y a ésta se le conectan los sensores, la salida de audio, la pantalla, la cámara y la alimentación y señales para la placa de relés.

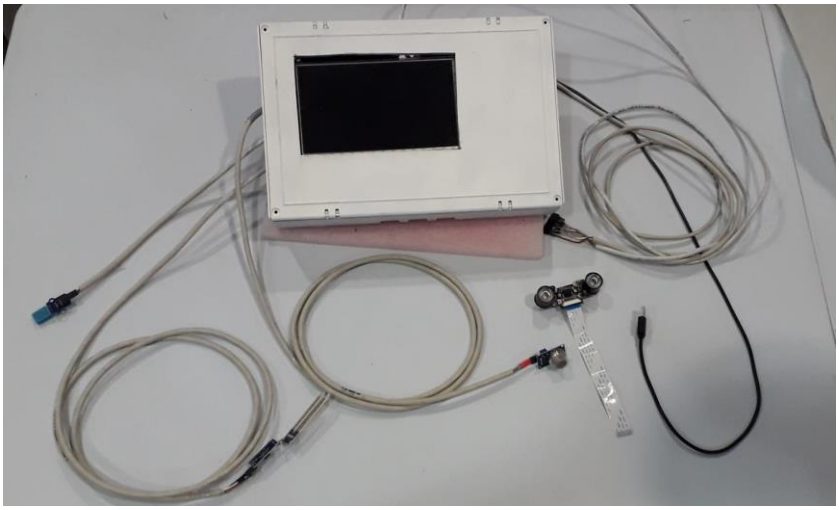


Figura 96. Conjunto baja tensión

A esta caja se le han realizado diferentes agujeros para la salida de cables hacia sensores o hacia la tarjeta de relés y un rectángulo hueco para poder interactuar con la pantalla que se encuentra dentro de la caja. Finalmente ha quedado conforme se muestra en la *Figura 96*.
La caja de alta tensión alberga la tarjeta de relés, la batería y el transformador. Un enchufe general conectado a la toma de red alimenta a todos los actuadores, excepto a los motores lineales, alimentados mediante la batería, conforme muestra la Figura 97

Figura 97. Conjunto alta tensión

El montaje de estas cajas sobre la estructura de la cuna se ha realizado de la forma más estética posible, por lo que, tanto el humidificador como la caja de alta tensión se han ubicado bajo del somier de la cuna, sobre una tabla de madera anclada mediante escuadras y tornillos a los laterales de la cuna. Los motores lineales también se han anclado a esta tabla de madera.

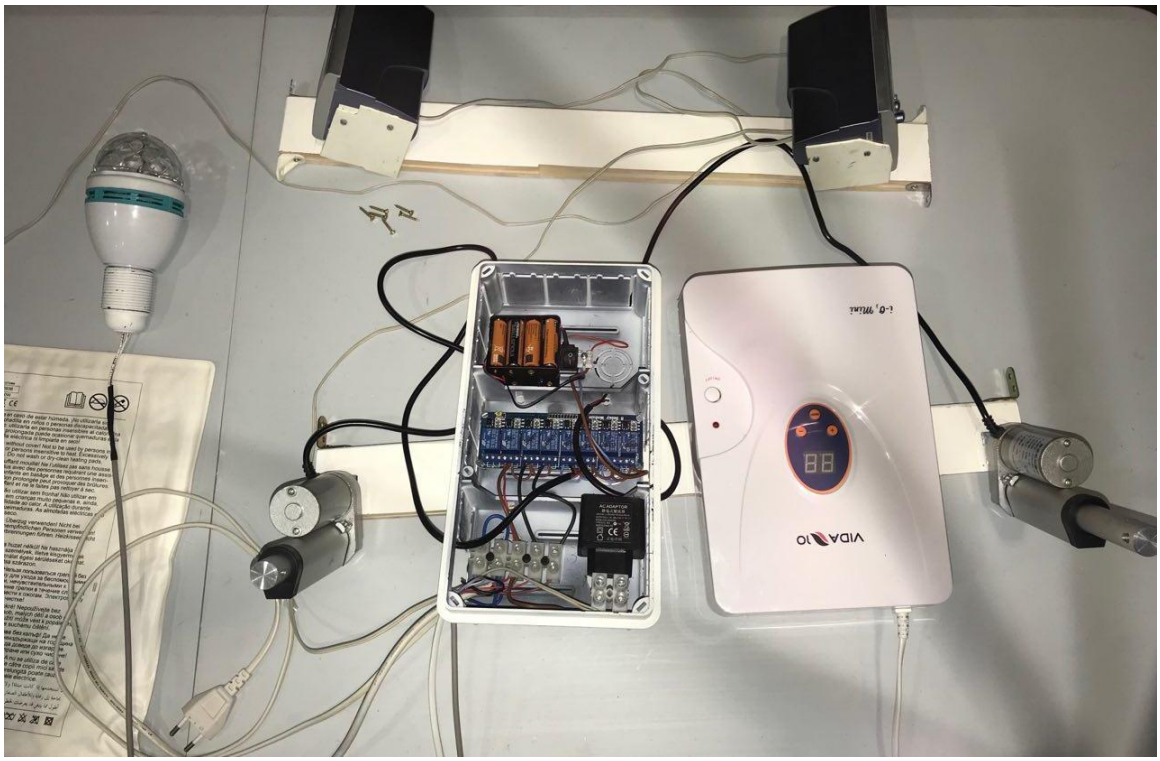


Figura 98. Montaje cuna

La máxima altura de estos motores indica la ubicación del somier, el cual se fijará únicamente por el centro mediante un eje longitudinal, de forma que el eje transversal tenga movilidad según la subida o bajada de los motores.

La caja de baja tension se ubica a los pies de la cuna, un sitio accesible para que e usuario puede interactuar con la pantalla.

7. Estudio de viabilidad de mercado

7.1 Planificación

Se va a examinar la planificación del desarrollo del prototipo de cuna inteligente, con el fin de crear el producto IntelliCrib.
Cabe destacar que este primer prototipo está sujeto a posteriores modificaciones o mejoras, las cuales no se han considerado en la planificación, ya que ésta únicamente considera el desarrollo del prototipo inicial.

7.1.1 Plan de recursos humanos

Para la realización de la planificación se ha optado por simular un equipo de trabajo que consta de tres perfiles diferentes, con el fin de plasmar una situación de realización de un proyecto real en el que coexisten diversos profesionales, cada uno con su experiencia en distintos ambitos del proyecto, que realizan determinadas tareas en paralelo.

El primer perfil propuesto es el Director del proyecto, el cual organiza y dirige el proyecto, llevando un seguimiento del trabajo y tomando decisiones en los diferentes puntos de inflexión que se puedan presentar. También se encargará de especificar los requisitos que debe cumplir el producto y las características técnicas de éste, así como considerar su cumplimiento durante la realización y una vez acabado el producto.
Esta última tarea normalmente la lleva a cabo un puesto inferior al Director del proyecto, el analista, pero en esta primera fase de prototipado, con escasos recursos económicos, va a desempeñarla el director del proyecto. Cabe la posibilidad de, que con el paso del tiempo y el crecimiento de la empresa, el director delegue la tarea de especificación de requisitos sobre un analista, pero siempre bajo la supervisión del director del proyecto.

El segundo perfil es el de un ingeniero industrial especializado en electrónica, el cual se hace cargo del desarrollo de la parte *hardware* del proyecto y tiene como objetivo cubrir los requisitos del producto final cumpliendo las características técnicas marcadas por el Director del proyecto. Este perfil también se encargará de realizar el montaje mecánico de IntelliCib, ya que es el que planteará la ubicación de cada elemento del proyecto en el prototipo final. Pero, del mismo modo que sucede con el perfil del analista, se tiene en cuenta a largo plazo, cuando la situación económica mejore y el planteamiento esté claro, la implantación de un perfil de operario, que se encargue de realizar el montaje mecánico de las cunas e incluso del conexionado *hardware*, a partir de un esquema eléctrico facilitado por el ingeniero de *hardware*.

El tercer perfil a destacar es el de un ingniero informático, que lleve a cabo el desarrollo de la parte *software* del proyecto, cubriendo los requisitos demandados. Este perfil debe trabajar en paralelo y en constante comunicación con el ingeniero de *hardware*, ya que las modificaciones en el desarrollo *hardware* afectan directamente a desarrollo *software* y a la inversa.

7.1.2 Condiciones de ejecución

Este apartado describe las condiciones de trabajo en las que se van a llevar a cabo el proyecto, como son la ubicación, los horarios y las tecnologías necesarias.

Dadas las necesidades de desarrollo del proyecto, éste se llevará a cabo en una oficina que dispone de acceso a internet y diferentes herramientas mecánicas para la ejecución del desarrollo *hardware*.

Las tecnologías de las que se disponen son un soldador, componentes y cableado necesario, una taladradora dremel y herramientas básicas como destornilladores y pelacables.
En cuanto las tecnologías *software* se necesitará una licencia de Windows y Microsoft Office. En el ordenador mencionado deberá descargarse el entorno de desarrollo de programación, en este proyecto, IntelliJ, así como la herramienta en la que implementar la interfaz gráfica, JavaFX Builder, y los programas necesarios para realizar la documentación del proyecto, como por ejemplo Inkscape, ya que el paquete Office ya incluye las herramientas Excel y Word.

El horario de trabajo que se ha estipulado contará con 4 horas diarias, durante 5 días a la semana, debido a que el proyecto, en un inicio, se encuentra en proceso de desarrollo y no aporta beneficios económicos, por lo que todos los integrantes del equipo trabajan de forma paralela a la realización de este proyecto.

7.1.3 Interlocución

Para que exista una comunicación fluida entre los integrantes del equipo y que cada uno de éstos esté al tanto de los avances del resto, se concierta una reunión de seguimiento por semana para detectar problemas que puedan surgir y plantear diferentes alternativas para su resolución.

El director del proyecto es el responsable de los acuerdos, confirmaciones o aprobaciones de cada fase o decisión del proyecto, basándose en los requisitos concretados al inicio y el plazo de ejecución estipulado para cumplirlos.

7.1.4 Plan de trabajo

En primer lugar, se van a exponer todas las tareas a desarrollar durante la realización del desarrollo del prototipo de cuna inteligente.
Respecto a cada tarea se va a señalar las tareas que le preceden, así como el perfil, de los tres expuestos, que va a llevarla a cabo y el tiempo que se estima para su ejecución con las correspondientes fechas de inicio y fin de cada tarea.

- **T1. Especificación de requisitos y organización**
  - T1.1. Recolección de requisitos
  - T1.2. Determinación de características técnicas
  - T1.3. Elaboración de cronograma

- **T2. Análisis de soluciones**
  - T2.1. Búsqueda de alternativas *software*
  - T2.2. Búsqueda de alternativas *hardware*
  - T2.3. Elección soluciones *software*
  - T2.4. Elección soluciones *hardware*
  - T2.5. Compatibilidad de soluciones

- **T3. Diseño del prototipo**
  - T3.1. Desarrollo solución *software*
  - T3.2. Desarrollo solución *hardware*
  - T3.3. Pruebas de compatibilidad
  - T3.4. Soluciones compatibilidad y diseño completo

- **T4. Implementación del producto**
  - T4.1. Implementación en Raspberry PI
  - T4.2. Implementación *hardware*
  - T4.3. Montaje mecánico
  - T4.4. Pruebas funcionales

Tarea	Previa	Perfil	Días	Horas	Comienzo	Fin
T1.1	-	Director proyecto	4	16	07/01/2019	10/01/2019
T1.2	T1.1	Director proyecto & Ingeniero *hardware*	3	24	11/01/2019	15/01/2019
T1.3	T1.2	Director proyecto	2	8	16/01/2019	17/01/2019
T2.1	T1.3	Ingeniero *software*	3	12	18/01/2019	23/01/2019
T2.2	T1.3	Ingeniero *hardware*	4	16	18/01/2019	24/01/2019
T2.3	T2.1	Ingeniero *software* & Director proyecto	2	16	23/01/2019	24/01/2019
T2.4	T2.2	Ingeniero *hardware* & Director proyecto	2	16	24/01/2019	28/01/2019
T2.5	T2.3, T2.4	Director proyecto & Ingeniero *hardware* & Ingeniero *software*	2	24	29/01/2019	30/01/2019
T3.1	T2.5	Ingeniero *software*	30	120	31/01/2019	13/03/2019
T3.2	T2.5	Ingeniero *hardware*	25	100	31/01/2019	06/03/2019
T3.3	T3.1, T3.2	Ingeniero *software* & Ingeniero hardware	5	40	14/03/2019	20/03/2019
T3.4	T3.3	Director proyecto & Ingeniero *hardware* & Ingeniero *software*	5	60	21/03/2019	27/03/2019
T4.1	T3.3	Ingeniero *software*	7	28	21/03/2019	29/03/2019
T4.2	T3.3	Ingeniero *hardware*	6	24	21/03/2019	28/03/2019
T4.3	T4.1, T4.2	Ingeniero *hardware*	9	36	01/04/2019	11/04/2019
T4.4	T4.3	Director proyecto & Ingeniero *hardware* & Ingeniero *software*	8	96	12/04/2019	24/04/2019

Tabla 7. Planificación de tareas

Cabe destacar que, para la elaboración de plazos de cada tarea, se ha tenido en cuenta el calendario laboral de 2019 de la comunidad valenciana y se han otorgado como festivos los fines de semana.

A continuación, se muestra el diagrama de Gantt, con las tareas previstas, y un diagrama de recursos, teniendo en cuenta los tres perfiles de los que se dispone.

El diagrama de Gantt muestra el cronograma de realización de todas las tareas, prediciendo que se finalizará el prototipo en la semana 19 si el comienzo se da en la semana 2. Este diagrama está sujeto a posibles modificaciones en el transcurso del desarrollo.

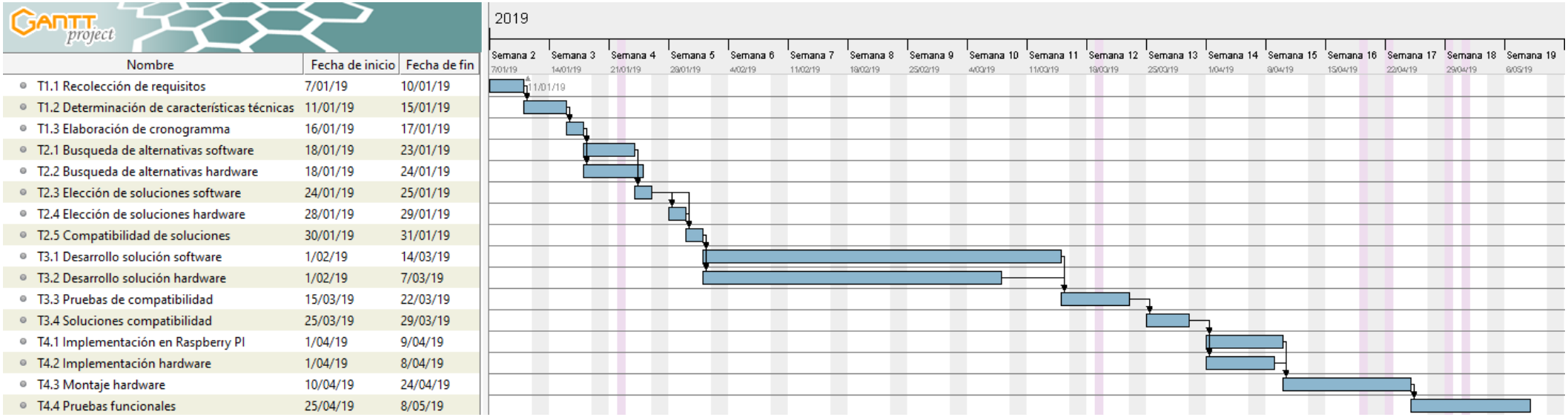


Figura 99. Diagrama Gantt

En diagrama de recursos muestra la ocupación de los diferentes perfiles que trabajan en el prototipo. Es importante cerciorarse de que no existen solapes de un mismo recurso para tareas diferentes.

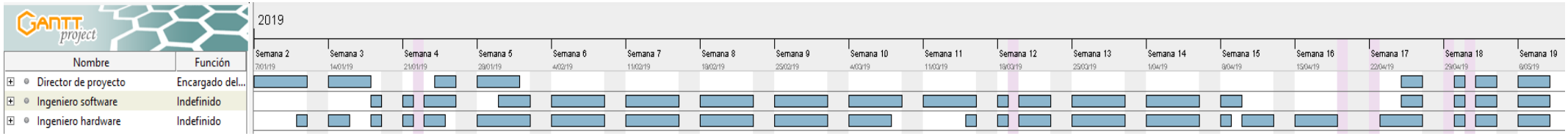


Figura 100. Diagrama de recursos

Como puede observarse, el perfil del director de proyecto realiza su máxima de trabajo al inicio y al fin del proyecto, mientras los ingenieros centran más su trabajo en el desarrollo del prototipo. Con ello no se descarta que la figura del director esté siempre presente para asegurar el cumplimiento de requisitos.

7.2 Estudio de la competencia

En este apartado se va a llevar a cabo una valoración de la competencia existente en el mercado de cunas de bebés inteligentes, realizando una exposición las competencias encontradas y una comparativa entre éstas e IntelliCrib.
Entre las cunas inteligentes encontradas en el mercado, se ha destacado la cuna BBSITTER y la cuna SNOO, puesto que se ha valorado que son las que más competencia pueden suponer.

7.2.1 Cuna BBSITTER

La cuna BBSITTER es una cuna inteligente con aprendizaje integrado, ya que integra un sensor de llanto que hará que la cuna aprenda qué movimientos calman al bebé para recrear sus secuencias preferidas siempre que sea necesario.
Entre ellos, incorpora un control para escoger entre los 6 movimientos disponibles.

BBSITTER dispone también de luz de vigilancia o compañía nocturna, medición de la temperatura y el nivel de humedad de la estancia.

Figura 101. Cuna BBSITTER. Obtenida de la página web micuna

Todo ello se puede controlar a través de su APP para móvil, donde se tiene la opción de seleccionar entre seis movimientos disponibles o bien, se podrá activar el modo inteligente para que sea la misma cuna la que seleccione el movimiento más adecuado que ha aprendido del bebé. Además, se podrá activar la luz de compañía para poder ver al bebé sin despertarlo o controlar la humedad y la temperatura de la habitación, para conseguir el entorno más confortable para el descanso del bebé.

El precio de mercado por el que se puede encontrar esta cuna inteligente es de **1.148,40 €.**

- COMPARACIÓN

En cuanto a las semejanzas que guarda BBSITTER con IntelliCrib, cabe destacar que, para el control de ambas, existe una aplicación móvil. Aunque BBSITTER solo ofrece el control remoto del balanceo y de la luz de vigilancia, así como la información sobre la humedad relativa y temperatura, mientras IntelliCrib permite el control remoto de un humidificador, una manta eléctrica, un proyector de luces, altavoces y cámara de video integrados, además de la información sobre la humedad relativa y temperatura de la habitación y sobre el estado de las sábanas y el pañal del bebé.
Ambas cunas ofrecen movimientos que tienen como objetivo tranquilizar al bebé, pero BBSITTER está más desarrollado en este aspecto ya que aprende sobre los gustos del bebé a partir de su llanto.

Como conclusión, BBSITTER ofrece un mayor abanico de posibilidades en cuanto al balanceo, pero en cuanto a los demás factores a tener en cuenta en el cuidado de un bebé, IntelliCrib presenta muchas más prestaciones, puesto que cubre los factores que ofrece BBSITTER pero ofreciendo muchísimos más que se consideran fundamentales para crear un ambiente de confort alrededor del niño.

7.2.2 Cuna SNOO

Snoo es una cuna robot que incorpora los últimos avances tecnológicos. En primer lugar, dispone de wi-fi, que envía datos a una app para permitir a los padres monitorizar en su teléfono los patrones de sueño de sus hijos. En segundo lugar, incorpora una serie de sensores que captan los movimientos y los sonidos del bebé: ante el más mínimo llanto, activan un sistema de suave mecido que trata de imitar al que los movimientos del útero, al tiempo que sus altavoces emiten sonidos similares a los que el bebé escucha cuando se encuentra en la tripa de su madre. En función de la intensidad del llanto del niño, la cuna variará sus sonidos y movimientos para tratar de calmarlo.

Figura 102. Cuna SNOO. Obtenida de la página web Bebés y más

Su marco de madera oculta sensores, micrófonos, altavoces y motores que responden a los movimientos del bebé y se apoya sobre patas de aluminio en forma de horquilla.
La altura es la estándar de las camas de adultos para permitir que los padres puedan ver a sus hijos sin tener que levantarse.
Las placas circulares bajo el colchón que giran hacia adelante y hacia atrás para simular el balanceo, varían de intensidad como haría un padre en función del estado de su bebé: los ingenieros del *MIT Media Lab* han desarrollado algoritmos que pueden diferenciar el llanto del ruido externo y que puede determinar el nivel de angustia dentro de un lamento.

El precio de mercado por el que se puede encontrar esta cuna inteligente es de **1.200 €**

- COMPARACIÓN

En esta cuna inteligente se encuentran prácticamente las mismas prestaciones que con la cuna BBSITTER, ya que ofrece un movimiento oscilante para calmar al bebé en función de su llanto, aunque también incorpora unos altavoces que emiten sonidos similares a los del vientre materno. SNOO también ofrece una app desde la cual los padres pueden ser informados sobre los patrones de sueño de sus hijos, aunque esta app no permite a los padres interactuar con los movimientos de la cuna como es el caso e BBSITTER o IntelliCrib.

Como ventajas a favor de SNOO puede destacarse la información, a través de la app, sobre los patrones de sueño del bebé y, al igual que BBSITTER, la baraja de movimientos que ofrece en función del llanto del bebé.
Como desventajas en comparación con IntelliCrib, SNOO no ofrece muchos de los parámetros informativos e interactivos que ofrece IntelliCrib, como son el control remoto del balanceo, del proyector de luces o sobre la manta eléctrica y el humidificador.
IntelliCrib también ofrece, a través de la app, información sobre el sensado de la humedad relativa y temperatura, así como del estado del pañal o de las sábanas donde se encuentra el bebé, aparte de proporcionar un sistema de vigilancia mediante la cámara de vídeo.

Como conclusión, SNOO, al igual que BBSITTER, ofrece un mayor abanico de posibilidades en cuanto al balanceo ligado al llanto del niño, pero en cuanto a los demás factores IntelliCrib continúa siendo mucho más completa.

7.3 Análisis financiero

En este apartado se va a llevar a cabo un análisis de costes de los diferentes factores que han contribuido a la formación del prototipo; un presupuesto de los componentes que integran el proyecto, un estudio de costes de la mano de obra y un análisis de la amortización de los equipos que han sido necesarios para llevar a cabo el proyecto.
El capítulo concluye con un análisis de la explotación de este producto en el mercado del que se obtiene la viabilidad de IntelliCrib según una línea temporal.

7.3.1 Presupuesto

En primer lugar, se va a analizar los costes de todos los componentes del equipo, como son sensores, actuadores, controles, elementos de montaje, etc.

Nombre	Descripción	Cantidad	Precio Unitario (€)	Precio total (€)
Sensor de calidad de aire	MQ4	1	3,59	3,59
Sensor de humedad absoluta	YL-29	1	1,95	1,95
Sensor de Tª y humedad relativa	DHT11	1	0,80	0,80
Humidificador	i-O3 VIDA10	1	59,00	59,00
Manta eléctrica	LESHP 100W	1	23,99	23,99
Proyector		1	3,99	3,99
Motores lineales	140kg Lift 1500N	2	28,80	57,6
Cámara	5MP 1080p	1	12,20	12,20
Altavoces	ZIPY 4WX4W RMS	1	11,95	11,95
Pantalla táctil	LCD Tipo TFT 7’	1	44,70	44,70
Cable HDMI	2.2 Alta velocidad	1	4,99	4,99
Cable MicroUSB	Qualtek	1	2,77	2,77
Placa de relés	ELEGOO 8 channel	1	10,99	10,99
Placa de control	Raspberry PI 3B	1	38	38
Cargador Raspberry	5,1V/3A	1	9,99	9,99
Batería 12V	8 pcs AA pilas	1	1,99	1,99
Protoboard	ELEGOO 3 pcs	1	7,99	7,99
Cajas plástico electrónica	ABS para PCBs	2	8,26	16,52
Cuna SUNDVIK	IKEA-blanca 60x120 cm	1	59,90	59,90
Otros materiales	Cables, conectores, estaño, tornillos, torretas			12,65
TOTAL MATERIAL				**385,56€**

Tabla 8. Coste material

Cabe destacar que el coste de la tecnología *software* es cero debido a que por ser estudiante de la Universidad Politécnica de Valencia se tiene licencia gratuita para todas las herramientas *software* utilizadas en este proyecto.

7.3.2 Importe horario

En este apartado se realiza un análisis del coste de la mano de obra teniendo en cuenta los tres perfiles expuestos en el primer apartado y las tareas a realizar. Suponiendo que los proyectistas que realizan el prototipo son ingenieros titulados y que, como tales, a los cálculos del coste del prototipo se le ha de añadir el precio de las horas dedicadas a las diferentes tareas realizadas.

Tarea	Nº de horas	Precio unitario	Coste total (€)
Especificación de requisitos y organización	48	15 €/h	720
Análisis de soluciones	64	15 €/h	960
Diseño del prototipo	320	15 €/h	4.800
Implementación del producto	184	15 €/h	2.760
TOTAL HORAS	**616**	**TOTAL MANO DE OBRA**	**9.240€**

Tabla 9. Coste mano de obra

Este coste total de mano de obra supone gran parte de la inversión inicial del proyecto, que debe recuperarse posteriormente con las ventas. Cabe destacar que dicho desembolso únicamente se realiza para el desarrollo del primer prototipo, en adelante se necesitará de estos perfiles únicamente para futuras mejoras, lo que supone un descenso notable en los costes de mano de obra. Aunque cabe añadir la necesidad futura de un perfil de operario que realice el montaje del equipo diseñado.

7.3.3 Amortización de equipos

Finalmente, para calcular la amortización, se debe considerar todos los equipos que se supone que han tenido que comprarse para la elaboración del proyecto y asumir en cuántos años se desea recuperar dicha amortización. Como equipos utilizados, se han considerado los siguientes:

Nombre	Descripción	Coste total (€)
Ordenador	Portátil	560
Fuente de alimentación	PROMAX FAC-662B	150
Multímetro	FLUKE 17B+	99,99
Soldador	30W	7,99
Herramientas	Alicates de corte, pela cabes, taladro dremel, destornillador	75,85
TOTAL EQUIPOS		**893,83€**

Tabla 10. Coste amortización de equipos

Suponiendo que se desea amortizar los equipos en los primeros dos años:

$$Coste\ mensual = \frac{893{,}83\ €}{24\ meses} = 37{,}24\ €/mes$$

$$Meses\ de\ desarrollo = 5\ meses$$

Entonces, el coste de amortización que debe tenerse en cuenta en el precio del prototipo es:

$$Coste\ amortización = 37{,}24 \frac{€}{mes} * 5\ meses = 186{,}20\ €$$

7.3.4 Coste global prototipo

Finalmente se suman todos los costes de los apartados previos para calcular el coste final del prototipo:

Gasto	Importe (€)
Coste material	385,56
Coste mano de obra	9.240
Coste amortización de equipos	186,20
TOTAL PROTOTIPO	9.811,76

Tabla 11. Coste global prototipo

Según el análisis de costes realizado, la realización del primer prototipo se valora en **9.811,76€**

7.3.5 Producción seriada

En este capítulo se realiza un estudio de los costes para la producción de IntelliCrib en serie. Una vez realizado el primer prototipo, se estudia el importe que supondrá la fabricación de una serie de cunas en cadena.

Dado que muchos de los costes que se han expuesto anteriormente se deben al desarrollo de la fase inicial y que, por tanto, solo deben considerarse una vez (como la búsqueda de alternativas o el diseño *hardware* y *software*, con sus correspondientes pruebas de compatibilidad y funcionamiento), el precio de fabricación de las siguientes cunas IntelliCrib será mucho menor.

El cálculo se realizará para una producción de 100 cunas inteligentes, ya que los diferentes distribuidores de componentes electrónicos hacen descuentos importantes si se compra en grandes cantidades, proveedores como Mouser o Farnell suelen hacer aproximadamente un 40 % de descuento en tales cantidades.
Suponiendo este descuento del 40% para todos los componentes y materiales que componen IntelliCrib, por demandarlo al por mayor, se tiene un coste de material por producto de:

$$Coste\ material = 385{,}56€ * 0{,}6 = \mathbf{231{,}40}\ €$$

También se considera que, al estar ya diseñado el prototipo, no es necesario que se tenga que contratar a un ingeniero profesional para realizar un montaje repetitivo, sino que se contratarán operarios, los cuales también se encargarán de realizar las pruebas funcionales dirigidas por un ingeniero. Estos operarios se estima que cobrarán 8€/h, por lo que el coste de mano de obra por cuna asciende a 71€.

Tarea	Nº horas	Precio hora	Coste total
Montaje *Hardware*	5h	8€/h	40€
Pruebas funcionales	2h	8€/h	16€
Estudios ingeniería	1h	15€/h	15€
TOTAL MANO OBRA	8h		**71€**

Se ha establecido un apartado de estudios de ingeniería, que supone el tiempo empleado en supervisión de pruebas funcionales y estudios de mejora. Este tiempo se ha estipulado en una hora por cuna.

Cabe destacar que la amortización de equipos equivale a **8,94€** por cuna, teniendo en cuenta que el coste total de quipos son 893,83€ y que la producción se considera de 100 unidades.

Con todo esto, se calcula el precio al que debería venderse IntelliCrib en el mercado:

COSTE UNIDAD = COSTE DE MATERIAL + COSTE MANO DE OBRA + AMORTIZACION EQUIPOS

SUBTOTAL= 231,40 € + 71 € + 8,94 €= 311,34 €

Finalmente se añadirá un beneficio que se desea obtener de cada producto fabricado; se establece que será un **30 %** del coste del producto neto, es decir que el precio de mercado de intelliCrib será:

TOTAL P.V.P DE PRODUCCION EN CADENA: 404,75 €/unidad

7.4 Viabilidad

En este apartado se va a llevar a cabo un estudio de viabilidad para los primeros años de inmersión de IntelliCrib en el mercado, teniendo en cuenta las inversiones de capital, los gastos y los beneficios que se preveen para cada año, con el fin de realizar una estimación del tiempo en que se recuperará la inversión y se comenzarán a obtener beneficios.

- AÑO 1

Del análisis financiero, se deduce que, para sacar al mercado este producto partiendo de cero, sería necesario desembolsar un gasto inicial de 9.811,76€ en concepto al desarrollo del primer prototipo.

Este prototipo se expondría a la sociedad mediante una campaña de marketing, que incluye la presencia de IntelliCrib en ferias de tecnología, anuncios, promociones en redes sociales, etc.
Se calcula que la inversión en cuanto a la campaña de marketing se refiere ascendería a un total de 10.000 €.

Esta inversión inicial en marketing y gastos operativos se costea con una inversión de 3.000€ de cada socio de la empresa, que son los tres perfiles expuestos anteriormente, con lo que se tendría un capital inicial de 9.000€ y los 11.811,76€ restantes se tendrán que costear mediante un préstamo, el cual se va a demandar de 20.000€, por gastos imprevistos y por disponer un margen financiero.
Cabe destacar que el banco nos ofrece el préstamo con Un interés del 5% TAE.

De forma que, el balance del primer año queda como sigue en la *Tabla 12*.

	INVERSIONES (€)	GASTOS OPERATIVOS (€)	BENEFICIOS VENTAS (€)	PRÉSTAMOS (€)	CAPITAL INICIAL (€)	DINERO EN CAJA (€)
AÑO 1	10.000	9.811,76	0	20.000	9.000	9.188,24

Tabla 12. Balance AÑO 1

A final de este primer año se tienen 9.188,24€ en caja y una deuda con el banco de 21.000€

- AÑO 2

Una vez realizado con éxito el desarrollo el producto deseado y la campaña de marketing, se podría producir en cadena el producto. Para este segundo año se quiere realizar una producción en serie de 100 cunas, para lo cual se ha estimado una inversión en maquinaria de 13.000€.
A la que debe sumarse la campaña de marketing, que debe seguir promocionando el producto y para la que se estima un coste de 10.000€ al año.
También sería necesario sumar a estos gastos el coste de materiales, mano de obra y coste de amortización de equipos con los que se desarrolló IntelliCrib para la primera producción en serie de 100 cunas, que sumaría un total de gastos operativos de 31.134€.
Para hacer frente a estos gastos se pide un segundo préstamo de 40.000€, el cual debe sumarse al del primer año, generando una deuda de 63.000 € con el banco.

De este modo el precio de venta al público de IntelliCrib sería de 404,75 €/unidad, obteniendo con cada cuna un beneficio neto de 93,40 €.

Suponiendo que se venden los 100 cunas producidas, el balance del año 2 queda como sigue.

	INVERSIONES (€)	GASTOS OPERATIVOS (€)	VENTAS (€)	PRÉSTAMOS (€)	CAPITAL INICIAL (€)	DINERO EN CAJA (€)
AÑO 2	23.000	31.134	40.475	40.000	9.188,24	35.529,24

Tabla 13. Balance AÑO 2

Al importe que se tenía en caja del año anterior, 9.188,24€, se le suman los 40.000€ del préstamo y los 40.475€ obtenidos en las ventas y se le restan las inversiones necesarias de 23.000€ y los gastos operativos de 31.134€, quedando un total de 35.529,24€ en caja, los cuales se destinarán a la producción del siguiente año. Cabe destacar que se sigue teniendo una deuda bancaria de 63.000€; 60.000€ de préstamo y 3.000€ en concepto de los intereses impuestos por el banco.

- AÑO 3

Este tercer año se pretende realizar una producción de 300 cunas, para lo que debe tenerse en cuenta los gastos de material y de mano de obra por unidad, que en total ascienden a 90.720€ ((311,34 - 8,94) · 300); 302,20€/cuna.
En esta suma no se ha tenido en cuenta la amortización de equipos que se utilizaron inicialmente puesto que ya se ha cubierto su coste durante los dos primeros años, con la venta de las cien primeras cunas.
Aunque el coste de mercado por cuna siga siendo de 404,75€, lo que supone que el beneficio a partir de este año asciende del 30% al 34%, es decir, el beneficio neto por cuna pasa a ser de 93,40 a 102,34€.

Para este gasto operativo se solicita un préstamo de 100.000€, por lo que la deuda asciende a 168.000€.

Este año se va a realizar una menor inversión en maquinaria, debido a que el pasado año se hizo una fuerte inversión y para éste simplemente serían necesarias algunas máquinas más, para las que se estima un total de 3.000€.
También se va a realizar una menor inversión en marketing debido a que el producto ya es conocido y no es necesario tanto importe para la campaña. Se van a destinar 3.000€ en este año para la campaña de marketing. Lo que asciende a un total de 6.000€ en inversiones.

	INVERSIONES (€)	GASTOS OPERATIVOS (€)	VENTAS (€)	PRÉSTAMOS (€)	CAPITAL INICIAL (€)	DINERO EN CAJA (€)
AÑO 3	6.000	90.720	121.425	100.000	35.529,24	160.234,24

Tabla 14. Balance AÑO 3

En la actualidad se tiene en caja 160.234,24€ y una deuda bancaria de 168.000€

- AÑO 4

En el cuarto año se pretende realizar una producción de 500 cunas, puesto que se entiende que se expande la comercialización del producto y, con ello, aumenta la demanda para lo que debe tenerse en cuenta los gastos de material y de mano de obra, que en total ascienden a 151.200€.
Este gasto operativo se hace frente con el dinero en caja por lo que no es necesario solicitar préstamo para este año.

En cuanto a inversiones de marketing se destinan 1.000€ y otros 1.000€ en concepto de mantenimiento de equipos.

	INVERSIONES (€)	GASTOS OPERATIVOS (€)	VENTAS (€)	PRÉSTAMOS (€)	CAPITAL INICIAL (€)	DINERO EN CAJA (€)
AÑO 4	2.000	151.200	202.375	0	160.234,24	209.409€

Tabla 15. Balance AÑO 3

En la actualidad se tiene en caja 209.409€ y una deuda bancaria de 168.000€

- AÑO 5

En el quinto año se mantiene la producción de 500 cunas, para lo que debe tenerse en cuenta los gastos de material y de mano de obra, que en total ascienden a 151.200€.
Este gasto operativo se hace frente con el dinero que se tiene en caja, aparte de pagar al banco 50.000€ de la deuda del préstamo.

En cuanto a inversiones de marketing se destinan 1.000€ y otros 1.000€ en concepto de mantenimiento de equipos.

	INVERSIONES (€)	GASTOS OPERATIVOS (€)	VENTAS (€)	PRÉSTAMOS (€)	CAPITAL INICIAL (€)	DINERO EN CAJA (€)
AÑO 5	2.000	151.200	202.375	-50.000	209.409€	208.584

Tabla 16. Balance AÑO 5

El quinto año se tiene en caja 208.584€ y una deuda bancaria de 118.000€

	INVERSIONES (€)	GASTOS OPERATIVOS (€)	VENTAS (€)	PRÉSTAMOS (€)	CAPITAL INICIAL (€)	DINERO EN CAJA (€)
AÑO 6	2.000	151.200	202.375	-50.000	208.584	207.759

Tabla 17. Balance AÑO 6

El sexto año se tiene en caja 207.759€ y una deuda bancaria de 68.000€

	INVERSIONES (€)	GASTOS OPERATIVOS (€)	VENTAS (€)	PRÉSTAMOS (€)	CAPITAL INICIAL (€)	DINERO EN CAJA (€)
AÑO 7	2.000	151.200	202.375	-68.000	207.759	188.934

Tabla 18. Balance AÑO 7

El séptimo año se tiene en caja 188.934 € y se ha solventado la deuda bancaria.

	INVERSIONES (€)	GASTOS OPERATIVOS (€)	VENTAS (€)	PRÉSTAMOS (€)	CAPITAL INICIAL (€)	DINERO EN CAJA (€)
AÑO 8	2.000	151.200	202.375	0	188.934	238.109

Tabla 19. Balance AÑO 8

El octavo año se tiene en caja 238.109€ de beneficio, pero teniendo en cuenta la inversión reservada para el siguiente año, se estima de beneficio neto de 84.909€

	INVERSIONES (€)	GASTOS OPERATIVOS (€)	VENTAS (€)	PRÉSTAMOS (€)	CAPITAL INICIAL (€)	DINERO EN CAJA (€)
AÑO 9	2.000	151.200	202.375	0	238.109	287.284

Tabla 20. Balance AÑO 9

El noveno año se tiene en caja 287.284 € de beneficio, pero teniendo en cuenta la inversión reservada para el siguiente año, se estima de beneficio neto de 134.084€

7.5 Conclusiones estudio viabilidad

Manteniendo la producción en 500 cunas anuales y las inversiones en marketing y maquinaria, se calcula que, a partir del **séptimo año,** comenzarían a obtenerse grandes beneficios del producto IntelliCrib. Por ello se obtiene como conclusión que es un producto altamente viable a largo plazo.

Respecto a la competencia económica con las otras dos cunas inteligentes del mercado, cabe destacar que IntelliCrib es un 67% más asequible que sus competidoras, ofreciendo muchas más prestaciones, como se ha expuesto en el apartado de estudio de competencia.

En cuanto al perfil de clientes que les podría interesar este tipo de producto claramente serían futuros padres, aunque no gocen de un nivel adquisitivo excesivo.
También se ha pensado en la posibilidad de ofrecer este servicio en hospitales de niños, los cuales estaría perfectamente monitorizados y cabría la posibilidad de crear una aplicación en la que observar las condiciones de todas las cunas de una sala en una sola pantalla, incluso de monitorizar patrones de interés demandados por el hospital.

8. Conclusiones

En primer lugar, cabe destacar el cumplimiento de los objetivos generales definidos al inicio de la memoria, ya que se ha desarrollado un producto que ofrece un entorno de confort al bebé que se encuentra en la cuna InteliCrib con el plus de que los padres están continuamente informados, incluso de forma remota.

En cuanto a los requisitos específicos, se ha conseguido desarrollar una aplicación con la que interactuar directamente con la cuna, la cual ofrece información visual de las condiciones ambientales de la habitación del niño, así como del estado del pañal y las sábanas que lo arropan y la también ofrece la posibilidad de observar al bebé en la cuna mediante vídeo siempre que se desee.
Todos los sensores y actuadores que componen el producto están activos y ofreciendo funcionalidad constantemente, siempre que la cuna está conectada a la toma de red eléctrica de los hogares mediante un sólo enchufe que ofrece alimentación a todo el conjunto.

Respecto a la viabilidad de este proyecto como producto, se ha observado que es un producto altamente viable a largo plazo y que supone una gran competencia para las cunas inteligentes que se encuentran actualmente en el mercado, con lo que se puede prever su éxito.

8.1 Relación conceptos MII

Cabe destacar la estrecha relación de muchos de los conceptos tratados a lo largo de este proyecto con los conocimientos adquiridos en las diferentes asignaturas del Máster en Ingeniería Industrial.

En primer lugar, la asignatura que guarda mayor relación de conceptos con este proyecto es la de Instrumentación y Control Industrial, donde se tomaron diferentes conocimientos sobre electrónica y sobre el control de diferentes sensores y actuadores que se pueden encontrar en cualquier proyecto de automatización.

En segundo lugar, el capítulo siete se basa en los conocimientos adquiridos en las asignaturas de Dirección de Empresas y Dirección de Proyectos, donde se vieron las formas de planificación de un equipo en un proyecto, así como el estudio de viabilidad de un producto que pretende hacerse hueco en el mercado.

En cuanto al montaje de todos los elemementos en la estructura de la cuna, se podría destacar la asignatura de estructuras, por el hecho de combinar el movimiento de los motores con una estructura fija, como sería el somier de la cuna.

Y, por último, si se tiene en cuenta, como línea futura, la producción de IntelliCrib en serie, cabría destacar los conocimientos adquiridos en la asignatura de Diseño y Aplicación de Equipos Industriales y de Dirección de Operaciones, por los aspectos destacados en ambas asignaturas sobre la producción en cadena y las pautas a seguir de cómo hacerla más efectiva.

Bibliografía

1. Lola Rovati. Las fases del sueño del bebé [en línea]. España: Bebés y más; Diciembre 2008 [fecha de consulta: Enero 2019]. Disponible en < https://www.bebesymas.com/desarrollo/las-fases-de-sueno-del-bebe>
2. Eva Paris. ¿Cuándo cambiar el pañal al bebé? [en línea]. España: Bebés y más; Noviembre 2013 [fecha de consulta: Enero 2019]. Disponible en < https://www.bebesymas.com/consejos/cuando-cambiar-el-panal-del-bebe >
3. Cristina Parra Cotanda. Temperatura en el hogar [en línea]. España: Miniland; Febrero 2017 [Fecha de consulta: Enero 2019]. Disponible en: < https://blog.minilandbaby.com/temperatura-hogar-cual-idonea-bebe>
4. Tania Olivares. Temperatura en el hogar [en línea]. España: Miniland; Febrero 2017 [Fecha de consulta: Enero 2019]. Disponible en: < https://blog.minilandbaby.com/temperatura-hogar-cual-idonea-bebe>
5. Colaboradores de Wikipedia. Arduino [en línea]. Wikipedia: La enciclopedia libre; Actualizado en Agosto 2019 [Fecha de consulta: Enero 2019]. Disponible en < https://es.wikipedia.org/wiki/Arduino>
6. Colaboradores de Wikipedia. Raspberry PI [en línea]. Wikipedia: La enciclopedia libre; Actualizado en Agosto 2019 [Fecha de consulta: Enero 2019]. Disponible en < https://es.wikipedia.org/wiki/Raspberry_Pi>
7. Naylamp. Tutorial sensores de gas MQ2, MQ3, MQ7 y MQ135 [en línea]. Perú: naylampmechatronics; Julio 2016 [Fecha de consulta: Enero 2019]. Disponible en <https://naylampmechatronics.com/blog/42_Tutorial-sensores-de-gas-MQ2-MQ3-MQ7-y-MQ13.html>
8. Hanwei Electronics. MQ2 gas sensor Data Sheet. USA: Robotshop; 2001 [Fecha de consulta: Enero 2019]. Disponible en <https://www.robotshop.com/media/files/pdf/MQ-2-datasheet.pdf>
9. Hanwei Electronics. MQ4 gas sensor Data Sheet. USA: Robotshop; 2001 [Fecha de consulta: Enero 2019]. Disponible en <https://www.robotshop.com/media/files/PDF/datasheet-sen0129.pdf>
10. Henan Hanwei Electronics. MQ9 semiconductor sensor Data Sheet. USA: Robotshop; 2001 [Fecha de consulta: Enero 2019]. Disponible en <https://www.robotshop.com/media/files/pdf/datasheet-sen0134.pdf>
11. Eodos. Sensor de humedad FC-28 [en línea]. España: Eodos blog; Mayo 2015 [Fecha de consulta: Enero 2019]. Disponible en: <https://eodos.net/proyectos/sensor-de-humedad#.XXOA_y4zaCg>
12. Rafael lozano. Sensor de humedad del suelo yl38 y yl69 [en línea]. España: Talos electronics SA; Junio 2018 [Fecha de consulta: Enero 2019]. Disponible en: <https://www.taloselectronics.com/blogs/tutoriales/sensor-de-humedad-del-suelo-yl38-y-yl69 >
13. LabFerrer. Sonda GS1[en línea]. España: LabFerrer; 2012[Fecha de consulta: Enero 2019]. Disponible en: <https://www.lab-ferrer.com/sensores/instrumentacion-y-sensores/humedad-del-suelo/gs1.html>

14. Naylamp. Sensor de temperatura y humedad relativa DHT21 (AM2301) [en línea]. Perú: naylampmechatronics; Julio 2016 [Fecha de consulta: Enero 2019]. Disponible en < https://naylampmechatronics.com/sensores-temperatura-y-humedad/354-sensor-de-temperatura-y-humedad-relativa-dht21-am2301.html >
15. Francesc. Sensor DHT11 de temperatura y humedad [en línea]. España: fpez.com; Agosto 2016 [Fecha de consulta: Enero 2019]. Disponible en <https://naylampmechatronics.com/sensores-temperatura-y-humedad/57-sensor-de-temperatura-y-humedad-relativa-dht11.html >
16. Naylamp. Sensor de temperatura y humedad relativa DHT22 (AM2302) [en línea]. Perú: naylampmechatronics; Julio 2016 [Fecha de consulta: Enero 2019]. Disponible en <https://naylampmechatronics.com/sensores-temperatura-y-humedad/58-sensor-de-temperatura-y-humedad-relativa-dht22-am2302.html>
17. Colaboradores de Wikipedia. IntelliJ IDEA [en línea]. Wikipedia: La enciclopedia libre; Actualizado en Junio 2019 [Fecha de consulta: Enero 2019]. Disponible en <https://es.wikipedia.org/wiki/IntelliJ_IDEA >
18. Colaboradores RasperryPi. The Pi4Java Project [en línea]. USA: Raspberrypi.com; Actualizado en Mayo 2019 [Fecha de consulta: Enero 2019]. Disponible en <https://pi4j.com/1.2/index.html>
19. Victor Robles. MCV (Modelo Vista Controlador) en PHP nativo [en línea]. España: VictorRoblesweb; Febrero 2014 [Fecha de consulta: Febrero 2019]. Disponible en <https://victorroblesweb.es/2013/11/18/tutorial-mvc-en-php-nativo/>
20. Consultor de escena. Java fx [en línea]. España: Riptutorial; Febrero 2018 [Fecha de consulta: Marzo 2019]. Disponible en <https://victorroblesweb.es/2013/11/18/tutorial-mvc-en-php-nativo/>
21. Colaboradores Raspberry Pi. Downloads [en línea]. Reino Unido: Raspberry Pi; Mayo 2009 [Fecha de consulta: Mayo 2019]. Disponible en < https://www.raspberrypi.org >
22. Colaboradores de RealVNC. VNC [en línea]. Reino Unido: RealVNC; 2002 [Fecha de consulta: Marzo 2019]. Disponible en < https://www.realvnc.com/es/>