UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FíSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO E IMPLEMENTACIÓN DE UNA PUERTA DE SEGURIDAD INTELIGENTE PARA INFANTES COMO UN PRODUCTO PARA EL COMERCIO MASIVO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

JAVIER IGNACIO WITTO ARAUJO

PROFESOR GUÍA:
MATÍAS SILVA CARES

MIEMBROS DE LA COMISIÓN:
ANDRÉS CABA RUTTE
FRANCISCO RIVERA SERRANO

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: JAVIER IGNACIO WITTO ARAUJO
FECHA: 2021
PROF. GUÍA: MATÍAS SILVA CARES

DISEÑO E IMPLEMENTACIÓN DE UNA PUERTA DE SEGURIDAD INTELIGENTE PARA INFANTES COMO UN PRODUCTO PARA EL COMERCIO MASIVO

En el presente trabajo de título se presenta el proceso de diseño e implementación de una puerta de seguridad inteligente para infantes, para ser comercializada de manera masiva. La inteligencia de la puerta está dada por un sistema embebido, capaz de detectar dos escenarios de peligro distintos: cuando la puerta no está bien cerrada y cuando el infante trata de sobrepasarla, escalando la puerta. Ante la detección de estos escenarios, se activa una alarma sonora que da aviso al adulto responsable para que venga a observar la situación.

La etapa de diseño contempla la elección de los componentes del sistema embebido y el diseño de la arquitectura del *firmware* de control del sistema embebido. El diseño se basa en criterios de mínimo costo, mínimo consumo energético y máxima eficacia en la detección de los escenarios.

La etapa de implementación implica la construcción del sistema embebido a partir del diagrama circuital, la integración en placa PCB, el diseño e impresión de una carcasa de contención mediante tecnologías de impresión de 3D y la integración del sistema embebido en la puerta, para la detección de los escenarios en el ambiente real. El requisito primordial de la implementación, además del correcto funcionamiento del sistema de detección de los escenarios, es tener un consumo energético tal que el sistema pueda durar encendido, por 30 días continuos cómo mínimo.

Posterior a la implementación, se procede a hacer una evaluación del costo unitario total para la implementación de la puerta inteligente, para su lanzamiento al mercado.

Como resultado, se logra obtener un producto que funciona con alta eficacia en la detección de cada escenario de peligro, con una duración que puede alcanzar hasta 53 días continuos y por un costo de implementación unitario total de USD30.01, lo que permite insertarlo de manera competitiva en el mercado nacional.

A mis padres, mis hermanas, la Pepa y el Milo

Agradecimientos

Agradezco a mis padres, Reinaldo y María del Pilar, por todo lo que me han dado y el cariño que me han brindado a lo largo de mi vida. Agradezco a mis hermanas, Carolina y Constanza, por el apoyo, las risas y cada momento vivido. Gracias familia por estar siempre conmigo. Gracias también a la Pepa y el Milo, nuestros peludos integrantes, por siempre sacarme una sonrisa.

Agradezco a mis eternos amigos del colegio, a YFS, al Tomás, al Álvaro, al Arturo, al Benja, al Chino, al David, al Diego, al Javier, al Coca, al Nacho, al Negro, al Pipe, al Rodri, al Seba, al Milky, al Vicho, al Toño y al Felipe. Amigos eternos.

Agradezco a Matías y a Daniel. No puedo concebir la U sin ustedes, gracias por cada momento que vivimos y disfrutamos durante los últimos 7 años. Gracias a Cristóbal, a Mike y a Bastián, por sufrir y reír conmigo en eléctrica.

Finalmente, agradezco a Aníbal, Matías y Thomas, mis compañeros de SoyMomo, por cada risa y cada rabia que nos saca este trabajo, y que tanto disfrutamos. Que sean muchos más.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

Desde que es capaz de circular de manera independiente, caídas, golpes y accidentes son una constante en la vida de un niño. Parte importante de su desarrollo y crecimiento se da a través de la exploración e interacción con el entorno que le rodea por lo que desde el nacimiento, un padre debe resignarse a buscar un balance entre la seguridad de su hijo y su desarrollo y crecimiento personal a través de la interacción con su entorno, lo cual puede inevitablemente provocar accidentes.

Dado que durante los primeros años de vida, un niño pasa la mayor parte del tiempo en su hogar, es aquí donde ocurre una parte significativa de los accidentes y lesiones de su infancia. Es por esto que un padre está siempre en la búsqueda de hacer de su casa un lugar a prueba de accidentes, tal que cualquier caída o golpe que sufran sus hijos durante su desarrollo no sean más que una mera anécdota, sin consecuencias en su salud y bienestar general.

Ahora bien, el peligro de accidentes de mayor gravedad, dentro del hogar, siempre está latente. De manera general, la Organización Mundial de La Salud (OMS) clasifica los peligros del hogar en 11 categorías distintas, de las cuales el peligro de ahogamiento, quemaduras, envenenamiento, heridas corto punzantes y traumas por caídas y golpes afectan principalmente a infantes entre los 1 y 5 años. Los accidentes por estas causas que terminan con resultado de muerte del infante no es menor. La OMS indica que, en promedio, 2000 infantes mueren en el mundo a causa de lesiones provocadas por accidentes [4]. En Estados Unidos, en particular, el *National Safety Council*, organización sin fines de lucro que aboga por la seguridad de las personas, reporta que solo durante el 2018, murieron 1900 niños, entre los 0 y los 4 años, debido a accidentes en la casa donde vivían [5].

Para reducir estas cifras, la medida primordial es la prevención y la vigilancia permanente de los infantes que circulan por el hogar. Una medida de prevención importante, es evitar la circulación de los menores por lugares con gran potencial de peligro. Por ejemplo, la cocina concentra varias de las categorías de peligro expuestas en el párrafo anterior, como quemaduras, heridas corto punzantes y trauma por golpe. En el baño existe el peligro de ahogamiento

y envenenamiento (artículos de limpieza, gabinetes medicinales, etc.). Por último, de haber una, las escaleras representan un gran peligro relacionado con los traumas por caídas.

Una solución existente para lo anterior son las puertas de seguridad. Éstas se colocan en el marco de las puertas o en el umbral de las escaleras y permiten bloquear el paso de los menores hacia zonas de peligro utilizando cerraduras diseñadas para evitar que éstos puedan abrirlas. En muchos casos, también incorporan sistemas mecánicos para que se cierren solas, una vez el adulto haya pasado por ellas, con el objetivo de obstaculizar lo menos posible la libre circulación de los adultos y asegurando que la puerta esté siempre cerrada, como corresponde.

Desafortunadamente, este mismo mecanismo de cierre seguro y automático provoca que, en muchas ocasiones, la puerta quede mal cerrada, ya sea junta o con el seguro mal puesto. Por otro lado, es común que los menores escalen y pasen por encima de la puerta, las que no suelen ser de más de 75 centímetros a un metro de alto. Ambas situaciones provocan que el infante ingrese a una zona de peligro, al mismo tiempo que el adulto responsable está menos atento, dado que confía en el buen funcionamiento de la puerta.

Si bien este producto existe en el mercado desde hace varios años, no han habido innovaciones significativas en el producto que resuelvan los problemas anteriormente expuestos, siendo que el producto está inserto en un mercado que ha presentado un crecimiento sostenido durante la última década. Según un reporte de *Grand View Research*, consultora de mercado estadounidense, el mercado de la seguridad para bebés fue tasado en 94 millones de dólares el 2018 y se proyecta que crezca a una tasa anual del 5 % de aquí al 2025 [6].

En el presente trabajo de título, se busca diseñar e implementar una puerta de seguridad, de manera masiva, que incorpore un sistema electrónico de seguridad redundante que resuelva los problemas anteriormente expuestos, para ser probada en el mercado de productos de seguridad para bebés como un producto innovador que posee un claro valor agregado frente a la competencia existente actualmente. El producto se desarrolla para y en conjunto a la empresa SoyMomo S.A, que está especializada en el desarrollo y comercialización de tecnología enfocada en bebés y niños.

1.2. Descripción del problema

El problema a resolver consiste en diseñar e implementar una puerta de seguridad para infantes, con un sistema electrónico embebido en ella que sea capaz de detectar dos escenarios concretos: cuando la puerta no queda bien cerrada por un período de tiempo mayor a lo normal, siendo normal el tiempo que necesite un adulto para abrir la puerta, pasar y cerrarla correctamente (escenario 1); y cuando un infante trate de escalar la puerta para sobrepasarla (escenario 2). Ante la detección de estos escenarios, el sistema embebido activará una alarma sonora, con el fin de que un adulto responsable venga a apagarla presencialmente.

El diseño de la puerta inteligente contempla 3 partes principales, las que deben estar en sintonía con el objetivo de implementarla de manera masiva, en serie, para su comerciali-

zación. Las 3 partes son: la puerta base, que será adquirida en masa desde un proveedor internacional, el sistema electrónico embebido, cuyos componentes deberán ser de bajo costo y fácil montaje, en relación al objetivo recién planteado y la carcasa de contención del sistema electrónico, que tiene el objetivo de integrar al sistema y sus periféricos con la puerta base. El desarrollo del trabajo de título se centra principalmente en la implementación del sistema electrónico y su integración con la puerta base, a través de la carcasa de contención.

El sistema embebido contará con: Un sensor que detecte el escenario 1, un sensor que detecte el escenario 2, un *buzzer* que actuará como la alarma sonora, una batería LiPo recargable, un circuito para recargar la batería con entrada USB, un sistema de detección de batería baja, un interruptor de encendido y apagado y un botón para apagar la alarma manualmente. Todo el sistema será controlado a través de un microcontrolador. Lo anterior se resume en la figura 1.1. Para cada uno de los componentes y el sistema electrónico en general, se considerarán criterios de bajo costo, que será medido en base al cálculo iterativo del costo unitario de la puerta completa, y de bajo consumo energético, que será medido en base al consumo de corriente del sistema completo, tal que pueda durar más de un mes continuamente con una batería de 2000 mAh.

La metodología a seguir puede dividirse en las etapas de diseño e implementación. La etapa de diseño tiene como objetivo principal la elección de los componentes finales del sistema embebido, a ser utilizados en la producción masiva, bajo los criterios antes mencionados de costo unitario, consumo energético y, en el caso de los sensores, robustez de la medición. La etapa de implementación tiene como objetivo principal el desarrollo completo del producto final, lo que comprende un proceso iterativo de construcción del sistema embebido, considerando su distribución espacial en una placa PCB y el consumo energético total del sistema embebido, más el diseño y construcción de una carcasa que lo contenga y que permita su integración exitosa en la puerta, siendo exitosa cuando el sistema logra detectar de manera robusta, ya integrado en la puerta, los escenarios planteados anteriormente y activa la alarma, en consecuencia. Todo esto sin interrumpir el funcionamiento normal de la puerta base.

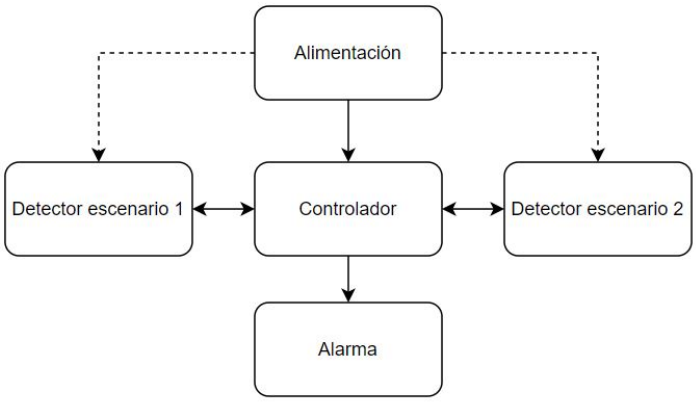

Figura 1.1: Diagrama de bloques general del sistema embebido. Las flechas discontinuas representan un flujo opcional.

1.3. Objetivos

1.3.1. Objetivos generales

Diseñar y construir una puerta de seguridad inteligente como producto para el mercado masivo, capaz de detectar, por medio de un sistema electrónico de bajo consumo energético integrado en la puerta, cuando ésta queda mal cerrada o cuando un infante trata de escalarla, activando, en consecuencia, una alarma sonora.

1.3.2. Objetivos específicos

1. Adquirir una puerta de seguridad base para la implementación del sistema electrónico, desde un proveedor que permita la producción masiva.
2. Diseñar un sistema electrónico embebido que detecte dos escenarios diferentes de vulneración de la puerta y active, en consecuencia, una alarma sonora.
3. Diseñar e implementar el sistema electrónico embebido para que tenga una duración de, por lo menos, 30 días continuos.
4. Implementar el sistema electrónico embebido en una placa PCB, previamente diseñada.
5. Diseñar e implementar una carcasa que contenga al sistema electrónico embebido y permita acoplarlo a la puerta base, procurando su correcto funcionamiento.
6. Estimar el costo unitario de fabricación de la puerta para producción masiva.

1.4. Estructura de la memoria

El presente informe de trabajo de título está dividido en 5 capítulos tal como se muestra a continuación.

En el capítulo 1 se desarrolla la motivación para realizar este trabajo. Se describe el problema a resolver, se denotan las razones por las que es importante resolver este problema y lo que se propone para ello. Junto con esto se entregan los objetivos generales y específicos que se deben cumplir y los resultados esperados.

En el capítulo 2 se desarrolla el marco teórico que sustenta el trabajo realizado, además de presentar las principales alternativas comerciales que actualmente existen, en Chile y el mundo, para resolver el problema presentado.

En el capítulo 3 se describe la metodología. Esta comprende la etapa de diseño del sistema embebido, la elección de los componentes finales y la implementación del sistema, desde su esquema circuital, el diseño de la placa PCB y la programación del microcontrolador, junto con el diseño y construcción de la carcasa de contención para el acople del sistema embebido a la puerta de seguridad base.

En el capítulo 4 se presentan y analizan los resultados obtenidos en cada etapa de la metodología.

En el capítulo 5 se presentan las conclusiones obtenidas sobre el trabajo realizado.

Capítulo 2

Marco Teórico y Estado del Arte

En el presente capítulo se hace una revisión del estado del arte, correspondientes a las soluciones comerciales que hoy existen para enfrentar el problema a resolver en este trabajo de título, tanto en Chile como en el resto del mundo. Posteriormente, se describen los antecedentes teóricos que sustentan el trabajo realizado, los que guardan relación, principalmente, con el desarrollo del sistema embebido que va acoplado a la puerta de seguridad. Se describen sus componentes principales y su funcionamiento en general. Luego de esto, se hace un repaso por las características y el funcionamiento del microcontrolador Arduino Pro Mini, núcleo del sistema embebido desarrollado. Se revisan sus componentes, especificaciones técnicas, periféricos y modos de programación. Se revisan, también, las máquinas de estado, de Mealy y Moore, que sirvieron como base para la programación del sistema embebido. Finalmente, se hace una revisión de los sensores utilizados durante la etapa de diseño.

2.1. Puertas de seguridad disponibles en el comercio

En la presente sección se hace una revisión de las puertas de seguridad que hay disponibles en el comercio tanto nacional como internacional. Se describen las características de los modelos disponibles, en general, y se listan los precios de cada uno.

2.1.1. Comercio nacional

Para el comercio nacional se hizo una revisión de las principales tiendas de retail del país, correspondientes a Falabella, Paris y Ripley, más las dos principales tiendas especializadas en productos para bebés e infantes, BabyTuto e Infanti. En estas tiendas, es posible encontrar puertas de seguridad de las siguientes marcas:

- Dreambaby
- Kidscool
- Bebeglo

- Glowup
- Kidco
- Safety 1st

En la tabla 2.1 se resumen los modelos disponibles en el mercado nacional, en las tiendas previamente mencionadas.

Tabla 2.1: modelos de puertas de seguridad disponibles en el mercado nacional junto con sus precios al 26/12/2020.

Marca	**Modelo**	**Precio [CLP]**
Dreambaby	Liberty	48.990
	Retráctil	90.990
	Xtra Hallway	57.990
	Xtra Tall	64.990
Kidco	Retráctil	129.990
	Cierre automático	129.990
	Escalera	79.990
Glowup	Escalera	52.990
Safety 1st	Crystal Clear	54.990
Bebeglo	RS-80150	49.990
Kidscool	YBD001	49.990
Infanti	Metálica	57.990

La revisión de los modelos permite identificar dos tipos de puertas de seguridad: de apertura convencional y de apertura deslizante, retráctil. Las primeras se abren como una puerta común, presentan barrotes verticales y tienen estructura metálica, con seguros y soportes plásticos. Se instalan a presión, con topes de goma que se apoyan contra el muro o marco de puerta donde se instalan. Son las más comunes y sus precios van desde CLP48.990 a CLP129.990. Las segundas están hechas de tela resistente que se desenrrollan desde un eje vertical para asegurarlas en el otro extremo, que va fijo sobre el muro donde está instalado. Solo hay dos modelos, de las marcas Dreambaby y Kidco, y sus precios van desde CLP90.990 a CLP129.990.

Todas las puertas cuentan con un doble seguro, que busca dificultar la apertura por parte de un infante. Los modelos más avanzados cuentan con cierre automático y son compatibles con extensiones para aumentar el ancho de cada una. Ningún modelo cuenta con seguridad redundante, dada por un sistema electrónico u otro.

2.1.2. Comercio internacional

Se hizo una revisión de la oferta internacional a través del sitio Amazon, en sus versiones de Estados Unidos y Europa (España, específicamente). En la tabla 2.2 se resumen las 10 puertas de seguridad más vendidas en Estados Unidos.

Tabla 2.2: Las 10 puertas de seguridad para infantes más vendidas en Amazon EE.UU. al 28/12/2020.

Ranking	**Marca**	**Modelo**	**Precio [USD]**
1	Regalo	Easy Step	35.99
2	Regalo	Easy Step Xtra-Tall	39.99
3	EasyBaby	Retractable Baby Gate	39.99
4	Regalo	Easy Open	36.99
5	Perma	Retractable Baby Gate	42.50
6	Regalo	Super Wide Gate	89.99
7	Safety 1st	Easy Install	68.99
8	Summer Infant	Multi-Use Deco	70.70
9	Toddleroo	Easy Swing & Lock	46.49
10	Regalo	Extra WideSpan	58.99

Por otro lado, en la tabla 2.3 se resumen las 10 puertas de seguridad más vendidas en España.

Tabla 2.3: Las 10 puertas de seguridad para infantes más vendidas en Amazon España al 28/12/2020.

Ranking	**Marca**	**Modelo**	**Precio [EUR]**
1	Safety 1st	Easy Close	35.00
2	Hauck	Auto Close & Stop	44.99
3	Pawhut	Barrera de seguridad extensible	56.99
4	Reer	Puerta de seguridad para niños	41.71
5	Safety 1st	Wall Fix Extending	42.13
6	OttoLives	Barrera retráctil	59.99
7	Safety 1st	Rejilla para escaleras	35.00
8	Munchkin	Maxi-Secure	35.99
9	Safety 1st	Easy Close Wood	46.09
10	Momcozy	Barrera de seguridad	59.99

Para el comercio internacional se identifican, en líneas generales, los mismos dos tipos de puertas que en el mercado nacional. No se identifican, tampoco, innovaciones en el diseño de ninguno de los modelos vistos, aunque sí se aprecia una disminución importante del valor de venta final, haciendo la conversión a moneda nacional para cada caso (Precio de referencia para el dólar: CLP750. Precio de referencia para el euro: CLP900). Por último, se aprecia

una variabilidad de precios mucho menor que en el caso del mercado nacional, con una concentración alta de precios en el rango 35.00-45.00 tanto en dólares como en euros.

Al igual que para los modelos presentes en el mercado nacional, cada modelo cuenta con doble seguro, para dificultar la apertura por parte de un infante, además de ofrecer extensiones (para el caso de las puertas convencionales, ya que las retráctiles tienen ancho variable). Se aprecian también modelos hechos en madera, en vez del tradicional marco metálico coloreado.

2.2. Sistemas embebidos

El sistema electrónico de alarmas a desarrollar en el presente trabajo de título entra en la categoría de sistema embebido. Los sistemas embebidos corresponden a sistemas electrónicos diseñados para realizar tareas específicas, en ambientes específicos. Dado lo anterior, son, en general, sub sistemas dentro de un sistema electrónico o mecánico de mayor envergadura, compuestos por una unidad de procesamiento, una unidad de memoria y dispositivos periféricos de entrada/salida [7]. Dentro del universo de fabricación de microprocesadores, aquellos destinados al control de sistemas embebidos corresponden al 98 % del total fabricado cada año, mientras que el 2 % restante forman parte del *hardware* operativo de computadores de escritorio [8].

Dada la caracterización de partes dentro de un sistema más grande, muchas veces los sistemas embebidos deben cumplir con exigencias temporales dentro de funcionamiento, lo que implica no solo cumplir con sus rutinas, sino que cumplirlas antes de cierto límite de tiempo. Aquellos sistemas embebidos que caen dentro de esta categoría son llamados sistemas en tiempo real y su programación exige el uso de sistemas operativos en tiempo real (o RTOS, por sus siglas en inglés), los que garantizan matemáticamente el cumplimiento de cada subrutina dentro del plazo establecido.

Existen un sinfín de aplicaciones para los sistemas embebidos, que van desde la electrónica de consumo, el control de electrodomésticos, tecnología médica, telecomunicaciones, la industria comercial y militar. De la aplicación deriva el nivel de exigencia temporal demandada al sistema. Aplicaciones médicas y/o militares demandan un nivel de robustez mucho mayor que aplicaciones comerciales de consumo masivo, lo que se traduce en un cumplimiento mucho más estricto de las *deadlines* de cada tarea [9].

2.2.1. Arquitectura de hardware y componentes

Cómo fue mencionado en el párrafo anterior, la composición de los sistemas embebidos, en general, está dada por una unidad de procesamiento, una unidad de memoria y periféricos que interactúan con el entorno en donde está inserto el sistema, dados por unidades de entrada, o sensores, que miden información del exterior, y unidades de salida, o actuadores, que traducen la información recibida en una respuesta para el exterior, dentro del contexto en el que está inserto el sistema [2]. En adición a lo anterior, los sistemas embebidos cuentan con unidades de hardware adicionales que permiten la realización de tareas dentro del sistema, tales como

relojes, conversores ADC y DAC y unidades específicas a la aplicación que realiza el sistema. En la figura 2.1 se presenta un esquema de la arquitectura general de un sistema embebido.

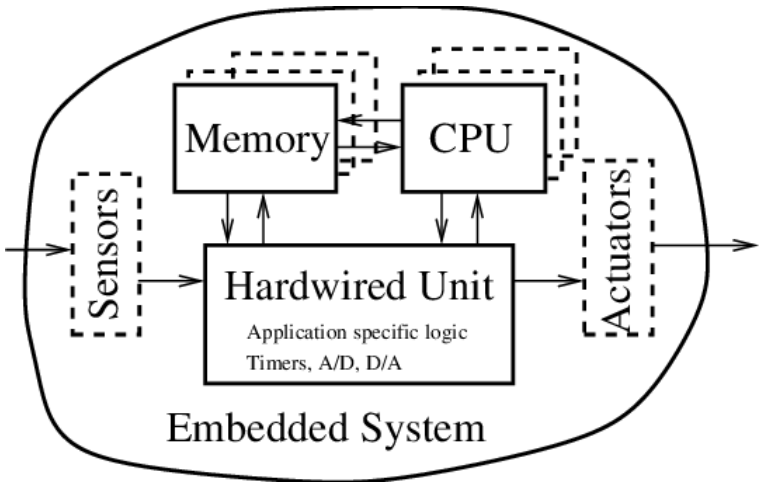

Figura 2.1: Esquema de la composición general de un sistema embebido [2]

La CPU, o *Central Processing Unit*, corresponde al núcleo de control de un sistema embebido. En este contexto, se pueden clasificar en dos categorías: Los microprocesadores y los microcontroladores. Los microprocesadores requieren de unidades de memoria y periféricos que no están integrados al microprocesador en sí, mientras que los microcontroladores evolucionan desde los microprocesadores, como unidades que integran periféricos y unidades de memoria en el mismo circuito integrado. Bajo esta clasificación, es posible considerar al microprocesador como un componente del microcontrolador, sin perjuicio de la distinción entre ambos en el contexto de la construcción de un sistema embebido.

Así como existen variadas aplicaciones para un sistema embebido, también existen variadas alternativas para el diseño de su CPU. Pueden tener arquitectura Harvard o Von Neumann, procesadores RISC o CISC y registros de 8, 26, 32 o 64 bit. No obstante, cualquier CPU contará con los siguientes componentes:

- **CU o *Control Unit***. Corresponde a la unidad de control y se encarga de manejar la data.
- **ALU o *Arithmetic Logic Unit***. Realiza todos los cálculos y operaciones entre unidades de datos.
- **Registros**. Unidades de memoria dentro de la CPU que contienen instrucciones específicas para ser ejecutadas.
- **Buses**. Contienen las direcciones de la memoria (Bus de direcciones) y los datos que van a almacenarse en la memoria (bus de datos).

Una representación simple de la arquitectura de una CPU, se aprecia en la figura 2.2.

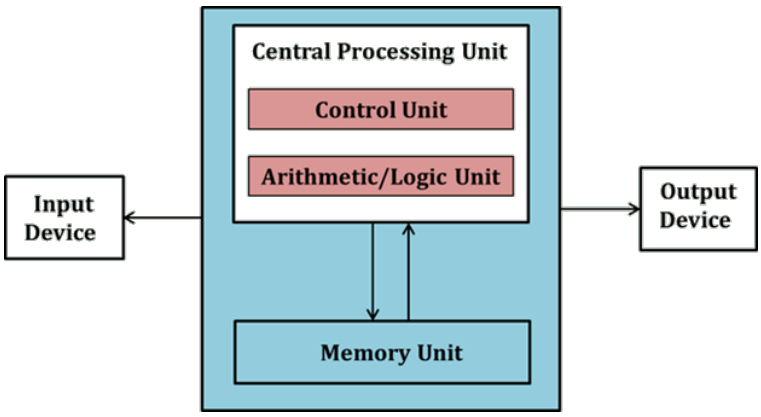

Figura 2.2: Representación de la arquitectura de una CPU.

Respecto a la memoria, ésta almacena datos y puede clasificarse en dos categorías principales: memoria volátil y memoria no volátil. La memoria volátil corresponde a aquellas unidades de memoria que necesitan mantenerse energizadas para conservar los datos. La memoria RAM, o *Random Access Memory*, es un ejemplo de memoria volátil, la que almacena las instrucciones a ejecutar por la CPU. La memoria no volátil, es aquella que se conserva aún cuando el sistema que la contiene no está energizado. La memoria ROM, o *Read-Only Memory*, entra en esta categoría y, como su nombre indica, es una memoria de sólo lectura, por lo que usualmente se utiliza para almacenar el programa de ejecución del sistema embebido, o *firmware* [10].

Los periféricos median las interacciones con el entorno del sistema embebido. Ejemplos de periféricos son las interfaces de comunicación, como I2C o SPI, conectores USB, módulos de conexión a internet, como Ethernet o tarjetas WiFi, conectores Entrada/Salida multipropósito (o GPIO, por sus siglas en inglés), entradas adicionales de memoria como tarjetas SD, etc. A los puertos GPIO de un sistema embebido, es posible conectar sensores, los que miden variables externas para ser procesadas por la CPU del sistema, o actuadores, los que envían una señal al exterior, de ser así requerido por el sistema.

2.2.2. Arquitectura de software de un sistema embebido

Como fue mencionado previamente, los sistemas embebidos están diseñados para ejecutar una tarea, o el mismo conjunto de tareas, durante su funcionamiento, de manera repetitiva. Debido a esto, la estructura del *firmware* de un sistema embebido caerá dentro de una de las siguientes categorías [11].

- ***Simple control loop***. Esta estructura cuenta con una rutina de *setup* que inicializa variables, puertos de comunicación, define el uso de los puertos GPIO, entre otros, para luego entrar en un ciclo continuo de ejecución de subrutinas que componen la, o las, tareas a cargo del sistema embebido.
- **Sistema basado en rutinas de interrupción.** Esta arquitectura mantiene vacío al

loop principal, o bien con tareas de tipo *preemptive*[1], pero cuenta con rutinas que son ejecutadas por el sistema si es que un evento, ya sea temporal periódico o dado por el cambio de estado de un periférico, las gatilla. Una vez ejecutadas, el sistema resume el *loop* principal.

- ***Preemptive multi-tasking***. Como fue mencionado antes, existen sistemas embebidos que son en tiempo real, lo que significa que deben ejecutar cada una de sus tareas dentro de una ventana de tiempo. Para garantizar que esto suceda, se utilizan RTOS, los que cuentan con algoritmos de *scheduling* que organizan las rutinas que el sistema debe ejecutar, acorde a la periodicidad, el tiempo de ejecución, la prioridad y el *deadline* de cada una.

2.2.3. Máquinas de estado

Sin desmedro de la sección anterior, el diseño del flujo secuencial de tareas a ejecutar por un sistema embebido puede ser modelado como una máquina de estado finito, cuya implementación puede ser realizada bajo las arquitecturas de la sección 2.2.2. Existen, principlamente, dos tipos de máquinas de estado, las máquinas de Mealy y las máquinas de Moore.

Una máquina de estados (o FSM, por sus siglas en inglés) es un modelo de computación basado en uno o más estados de funcionamiento, en donde sólo uno puede estar activo cada vez. El estado activo cambia (o no) en base a entradas y cada estado está asociado a una salida. El siguiente estado activo está determinado por la entrada y el estado actual del sistema [12]. Las máquinas de Mealy y las de Moore se diferencian entre sí por cómo determinan la salida del sistema en cada estado.

2.2.3.1. Máquina de Moore

Las máquinas de Moore corresponden a máquinas de estado cuyas salidas dependen únicamente del estado activo. Matemáticamente, se caracterizan por un vector de 6 componentes [13]:

- Un conjunto finito de estados Q
- Un estado inicial único q_0
- Un conjunto finito de entradas Σ
- Un conjunto finito de salidas O
- Una función de transición δ que define las transiciones entre estados, según el estado activo y la entrada actual.
- Una función de salida G que define las salidas para cada estado activo.

En la figura 2.3 se denota gráficamente la caracterización de Moore.

[1]Una tarea *preemptive* es aquella que puede ser interrumpida durante su ejecución, si es que la CPU necesita ejecutar otra de mayor prioridad.

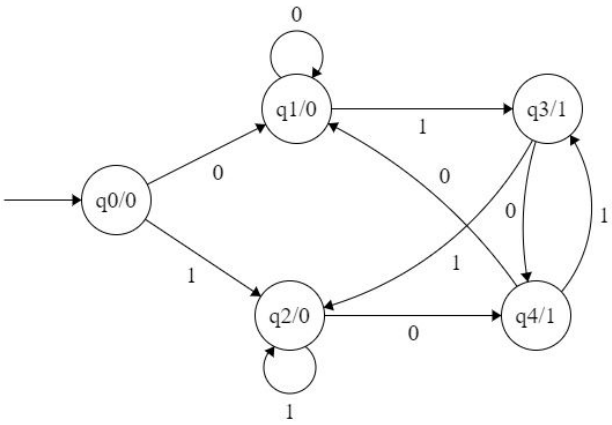

Figura 2.3: Caracterización de Moore para una máquina de estados.

2.2.3.2. Máquina de Mealy

En contraparte, las máquinas de Mealy corresponden a máquinas de estado en las que la salida dependen del estado activo y las entradas, por lo que pueden haber más de una salida para cada estado. Formalmente, se caracterizan a través de un vector de 6 componentes [14]:

- Un conjunto finito de estados Q
- un estado inicial único q_0
- Un conjunto finito de entradas Σ
- Un conjunto finito de salidas O
- Una función de transición δ que define las transiciones entre estados, según el estado activo y la entrada actual.
- Una función de salida G que define las salidas según las tuplas $(Q_\mathrm{i}, \sigma_\mathrm{i})$, donde σ_i corresponde a una entrada al estado Q_i.

En la figura 2.4 se denota gráficamente la caracterización de Mealy.

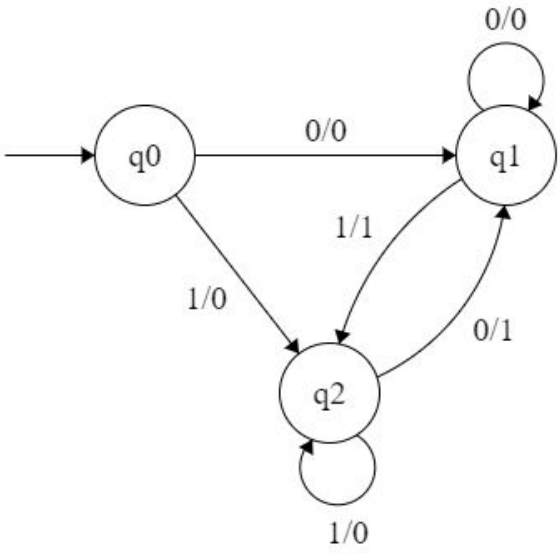

Figura 2.4: Caracterización de Mealy para una máquina de estados.

2.3. Microcontroladores

En la presente sección se revisan los microcontroladores relevantes al trabajo de título. Como fue mencionado, estos cumplen el rol de la CPU del sistema embebido y, a diferencia de los microprocesadores, integran memoria y periféricos dentro del mismo circuito integrado. Además, los microcontroladores actuales cuentan con memorias EEPROM reprogramables, por lo que pueden reutilizarse para distintas funciones a lo largo del tiempo [15].

2.3.1. NodeMCU

Actualmente, se comercializan diversos modelos de microcontroladores con miras al usuario masivo, otorgando entornos de desarrollo de fácil uso, tanto a nivel de *hardware* como de *software*. Uno de estos modelos es el denominado *NodeMCU*, el que cuenta con un núcleo procesador ESP8266, de la compañía *Espressif Systems*, con conexión WiFi integrada [1]. En la figura 2.5 se denota la estructura de este microcontrolador.

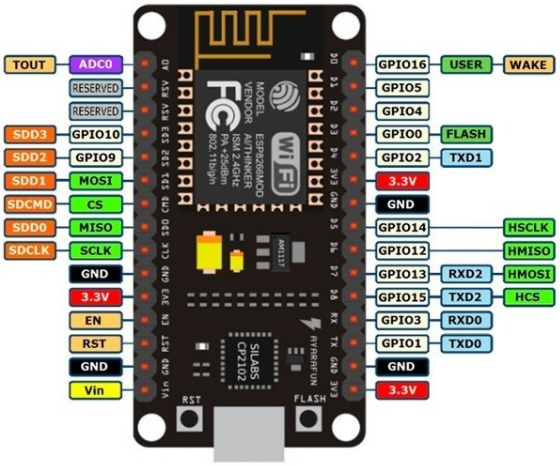

Figura 2.5: Diagrama del diseño del microcontrolador NodeMCU.

Este microcontrolador cuenta con:

- 16 pines GPIO
- 1 pin analógico
- 1 pin SPI
- 1 pin I2C
- Puerto serial CH340
- Regulador de voltaje AMS1117
- Memoria Flash de 4 Mb
- Memoria RAM de 64 Kb
- Frecuencia 80 MHz

Una característica importante de este microcontrolador, es que cuenta con 3 modos de ahorro energético, con distintos grados de limitación de sus funciones. En la tabla 2.1 se resume el consumo energético para cada modo de ahorro.

Modo de ahorro	**Limitación de funciones**	**Consumo de corriente**
Modem Sleep	CPU funcionando Limitaciones a la conexión WiFi	15 mA
Light Sleep	Funcionamiento limitado de la CPU	0.9 mA
Deep Sleep	Solo funciona el reloj	20 μA

Tabla 2.4: Consumo energético del NodeMCU en sus distintos modos de ahorro energético. Tabla extraída de [1]

2.3.2. Arduino Pro Mini

Otro microcontrolador enfocado en el desarrollo masivo, con licencias de desarrollo, tanto de hardware como de software, libres, es el Arduino Pro Mini, de la compañía Arduino. Este cuenta con un procesador ATmega328P, de la compañía *Atmel*, con arquitectura RISC (*Reduced Instruction Set Computer*) para la ejecución de instrucciones de tamaño fijo, almacenadas en su memoria [16]. En la figura 2.6 se denota la estructura de este microcontrolador.

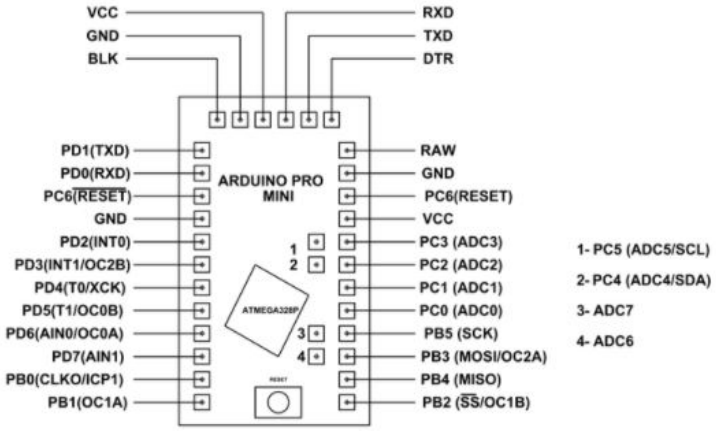

Figura 2.6: Diagrama del diseño del microcontrolador Arduino Pro Mini.

Este microcontrolador cuenta con:

- 23 pines I/O en total.
- 7 pines I/O analógicos.
- 6 pines que soportan salidas PWM de 8 bits.
- 2 pines para interrupciones externas.
- Interfaz UART.
- Interfaz SPI.
- Regulador de voltaje MIC5205

Las características técnicas del Arduino Pro Mini se aprecian en la tabla 2.5.

Tabla 2.5: Características técnicas del Arduino Pro Mini.

Característica	**Valor**
Voltaje de operación	3.3 V
Voltaje de entrada	[5-12] V
Corriente máxima en pin I/O	40 mA
Memoria Flash	32 Kb
Memoria EEPROM	1 Kb
Memoria RAM	2 Kb
Frecuencia de reloj	8 MHz

El procesador Atmel ATmega328P del Arduino Pro Mini, cuenta con una serie de registros de 8 bits que pueden ser utilizados para la ejecución de instrucciones específicas al procesador, según la hoja de datos provista por Atmel [16]. A continuación, se revisan aquellos relevantes al trabajo de título.

- **CLKPR o *Clock Prescale Register***. Este registro permite modificar la frecuencia de oscilación del reloj de ejecución, a partir de la división del reloj interno del sistema por un factor de 2, 4, 8, 16, 32, 64, 128 o 256. Disminuir la frecuencia del reloj aporta en la reducción del consumo energético del procesador.
- **WDTCSR o *Watchdog timer Control Register***. El *watchdog timer* es un temporizador que, durante el funcionamiento normal del sistema, se reestablece cada cierto tiempo. Si no se reestablece, es un indicador de una falla en el sistema. Dada esta característica, se puede utilizar para establecer una rutina de interrupción del Arduino dada por un período de activación en vez de una interrupción externa, cuando el microcontrolador está en modo de bajo consumo.
- **ADCSRA o *ADC Control and Status Register A***. Este registro permite el control del ADC[2] del ATmega328P. En caso de no ser usado puede desactivarse, incurriendo en un ahorro energético.
- **MCUCR o *MCU Control Register***. Este registro permite, entre otras cosas, activar y controlar as interrupciones externas conectadas a los pines INT0 e INT1 del ATmega328P (dados por los pines 2 y 3 del Arduino Pro Mini), además de desactivar el *Brown-Out Detection*[3] mediante los pines 5 y 6 (BODS y BODSE) del registro MCUCR, para la activación del microcontrolador cuando está en modo de bajo consumo, a través de interrupciones externas.
- **SMCR o *Sleep Mode Control Register***. Mediante la manipulación de los bits de este registro, es posible acceder a uno de los modos de ahorro energético del procesador ATmega328P. De los 8 bits del registro, se utilizan 3 (SM1, SM2 y SM3) para acceder a cualquiera de los 8 modos de ahorro energético disponibles, mientras que el bit restante (SE) se utiliza para activar efectivamente el modo *sleep*, una vez ejecutada la instrucción.

2.4. Sensores

En la presente sección se describe el funcionamiento teórico de los sensores utilizados durante la realización del presente trabajo de título.

[2]ADC o *Analog to Digital Converter* corresponde a una unidad de hardware que muestrea una señal análoga para convertirla en una señal digital. En el caso del Arduino Pro Mini, la señal digital es de 10 bits.

[3]*Brown-Out Detection* es un mecanismo que tienen los sistemas computacionales para reestablecerse en caso de detectar bajas de voltaje en el sistema de alimentación y así evitar fallas mayores.

2.4.1. Sensor infrarrojo

El sensor infrarrojo corresponde a un par emisor-receptor. El emisor es un LED (*Light Emitting Diode*) que emite ondas electromagnéticas de 940 nm de longitud. Tienen un voltaje de operación de 1.7 V y utilizan una corriente máxima de 100mA [17]. Por otro lado, el receptor es un fotodiodo. Los fotodiodos son dispositivos semiconductores que permiten la conducción de corriente a través de ellos en el caso de que ondas electromagnéticas de 940 nm de longitud incidan sobre él [18]. En la figura 2.7 se aprecia una implementación típica para este sensor.

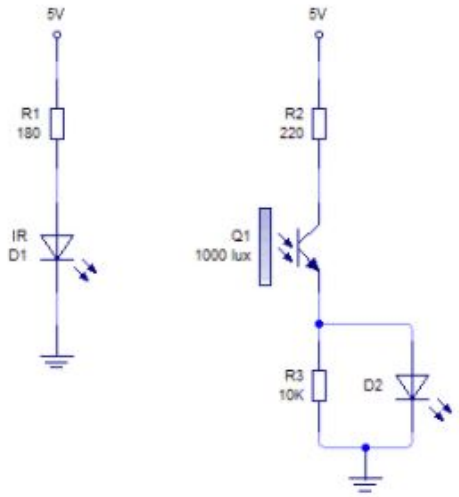

Figura 2.7: Implementación para el sensor infrarrojo de par emisor-receptor. A la izquierda, etiquetado como IR, está el par emisor. A la derecha, etiquetado como Q1, está el par receptor.

2.4.2. Sensor de ángulo

El sensor de ángulo es un potenciómetro implementado como divisor del voltaje de operación del sensor, correspondiente a 5 V (ver figura 2.8). Al rotar el eje del potenciómetro, varía el voltaje medido a la salida del sensor, lo que puede transformarse en una medición angular utilizando una función lineal [19].

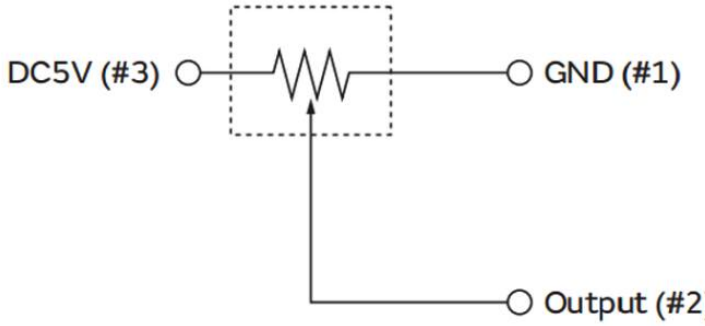

Figura 2.8: Diagrama conectivo interno del sensor de ángulo CJMCU-103.

2.4.3. Sensor magnético

El sensor magnético se compone de 2 imanes, uno de los cuales posee 2 pines conectivos. Cuando los imanes están cerca, la fuerza magnética entre ellos provoca que se cierre el circuito dentro del imán que posee los pines, generando continuidad entre ellos. De esta manera, es posible detectar un cambio en el estado lógico del sensor, cada vez que se acercan o alejan los imanes que lo componen [20]. En la figura 2.9 se aprecia el esquema de funcionamiento de este sensor.

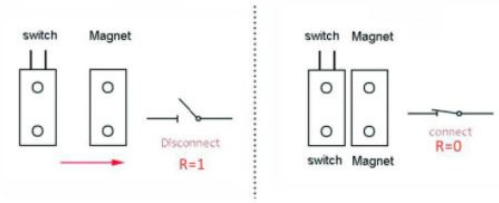

Figura 2.9: Esquema de funcionamiento del sensor magnético MC-38.

2.4.4. Sensor ultrasónico

El sensor ultrasónico es un sensor digital que consta de dos componentes integrados en la misma placa de operación: Un emisor (*Trigger*) y un receptor (*Echo*). El emisor envía un pulso ultrasónico al ambiente, la cual se refleja en cualquier obstáculo que haya en su camino y al retornar, es detectada por el receptor. A partir de la velocidad del sonido y el tiempo entre la emisión y la recepción del pulso por parte del sensor, es posible computar la distancia entre el sensor y el obstáculo, acorde a la ecuación 2.1 [21].

$$\mathrm{d}_{obstáculo} = \frac{v_{sonido} \cdot tiempo}{2} \tag{2.1}$$

Para la ecuación 2.1 se utiliza 340 m/s como estimación a la velocidad del sonido, mientras que el tiempo es medido por el sensor ultrasónico.

2.4.5. Sensor de presión

El sensor de presión es, estrictamente hablando, una resistencia variable. El rango de valores que toma va desde los 100 *Ohm* a los 100 *kOhm* y dependen directamente de la fuerza aplicada sobre el sensor, tal como se ve graficado en la figura 2.10.

La variabilidad de su resistividad se da porque cuenta con una capa de sustrato de elementos piezoeléctricos[4] que al ser presionados aumentan su conductividad y, por ende, la resistividad entre los pines del sensor disminuye [3].

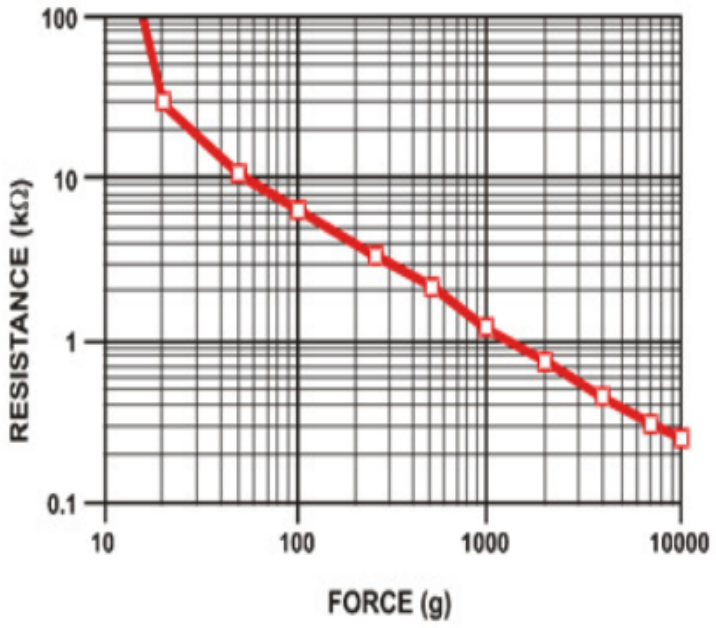

Figura 2.10: Gráfico de fuerza v/s resistividad de un sensor de presión FSR 402. Curva obtenida de [3].

[4]Los materiales piezoeléctricos son aquellos que generan un diferencial de voltaje al ser deformados mecánicamente.

Capítulo 3

Diseño del sistema embebido

En la figura 3.1 se aprecia una visión general de la metodología desarrollada durante el presente trabajo de título. Esta contempla, a grandes rasgos, el diseño y la implementación de el sistema embebido y la integración de éste a la puerta de seguridad base, para dar paso a una estimación del costo de fabricación de cada unidad a escala masiva.

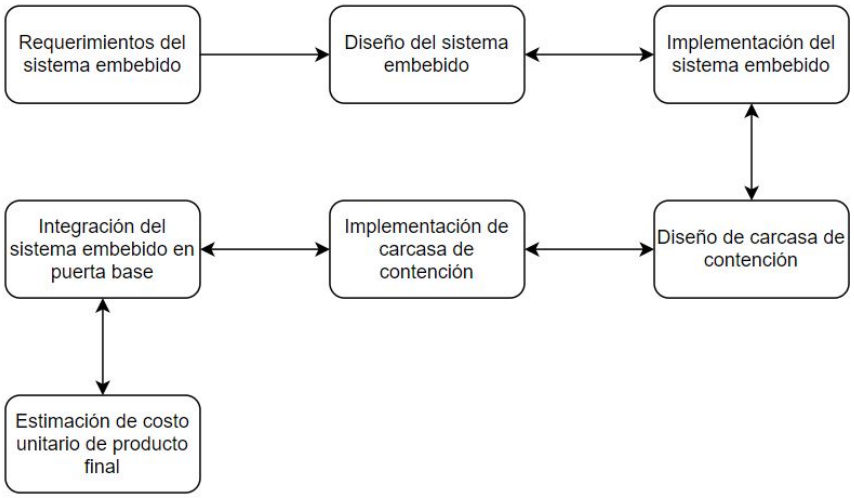

Figura 3.1: Metodología para el diseño e implementación de la puerta de seguridad inteligente.

Se entiende la metodología como un proceso iterativo, compuesto de las siguientes etapas:

1. **Requerimientos del sistema embebido:** Se establecen los requerimientos a cumplir por el sistema embebido, una vez esté integrado en la puerta base, en materia de funcionamiento, costo y consumo energético.
2. **Diseño del sistema embebido:** En base a los requerimientos, se hace un desglose de los componentes y funciones que debe realizar el sistema embebido, para la elección de sus sensores, actuadores, alimentación, CPU y arquitectura de software.
3. **Implementación del sistema embebido:** Esta etapa contempla la integración de todos los componentes del sistema embebido, a través del diseño e implementación de

una placa PCB, más pruebas de funcionamiento independiente y de consumo energético.

4. **Diseño de carcasa de contención:** Esta etapa contempla el diseño de la carcasa de contención del sistema embebido, que debe considerar la sujeción a la puerta base, la distribución de los periféricos del sistema embebido y el funcionamiento normal de la puerta base por sí sola.
5. **Implementación de la carcasa de contención:** Esta etapa contempla el desarrollo de la carcasa de contención mediante impresión 3D, considerando el material y la técnica de impresión, además de la correcta distribución de los periféricos del sistema a lo largo de la carcasa.
6. **Integración del sistema embebido a la puerta base:** Esta etapa contempla las iteraciones necesarias para que el sistema embebido funcione correctamente en la puerta base, sin molestar el funcionamiento de ésta y con la firmeza suficiente para que el sistema sea duradero.
7. **Estimación del costo unitario del producto final:** Esta etapa contempla el desglose de la estructura de costos que implica el desarrollo del producto completo, a escala masiva, para su posterior evaluación económica y comercialización.

3.1. Requerimientos del sistema embebido

Como fue mencionado en el capítulo 1, el producto consiste en una puerta de seguridad para infantes con un sistema electrónico embebido en ella, que hace de sistema de seguridad redundante. Esto se traduce en que es capaz de detectar dos escenarios, mediante los sensores que tiene incorporados: Cuando la puerta no queda cerrada correctamente (escenario 1 o **E1**) y cuando un infante trata de escalarla (escenario 2 o **E2**). Ante la detección de estos escenarios, el sistema activa una alarma sonora que se espera sea desactivada por el adulto responsable a cargo.

El sistema embebido deberá cumplir con los siguientes requerimientos:

1. El sistema embebido no puede entorpecer el funcionamiento normal de la puerta de seguridad.
2. El sistema embebido no puede incurrir en falsas detecciones, tanto positivas como negativas, de los escenarios 1 y 2.
3. La alarma generada por el sistema debe ser audible en un radio de 10 metros.
4. El sistema embebido debe estar energizada de manera independiente al sistema eléctrico del hogar.
5. Se debe garantizar el funcionamiento del sistema embebido por un mínimo de 30 días continuos.
6. El diseño del sistema embebido debe contemplar el menor costo posible que cumpla con los requerimientos anteriores.

3.2. Diseño del sistema embebido

Para el diseño del sistema embebido, se considera la estructura de bloques que se observa en la figura 3.2.

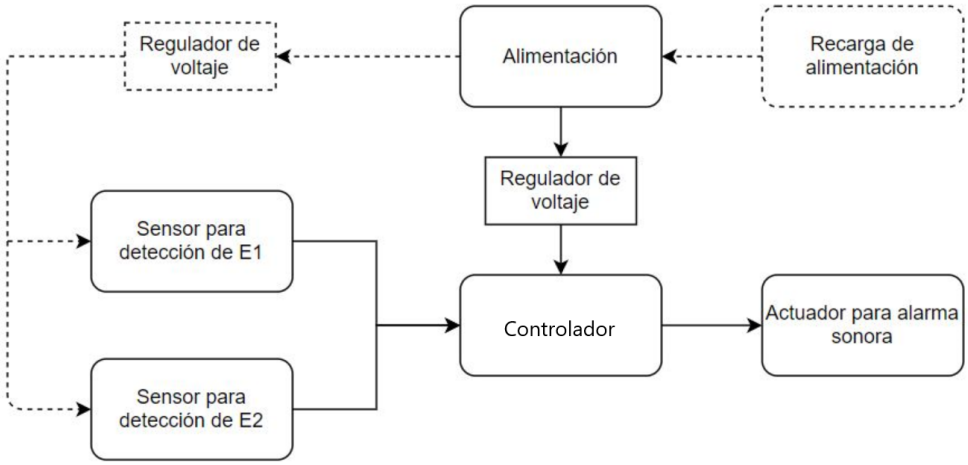

Figura 3.2: Estructura general de bloques del sistema embebido. Se utilizan líneas discontinuas para representar elementos que no son estrictamente necesarios, a priori, y que podrían no estar presentes en el resultado final.

3.2.1. Alimentación del sistema

El sistema embebido debe ser alimentado de manera externa, con una fuente independiente a la red eléctrica. El sistema de alimentación se compondrá de la fuente de poder, un sistema de recarga de la fuente (de ser necesario), un conjunto de reguladores de voltaje para cada componente alimentado directamente desde la fuente de poder, un *switch* de encendido y un sistema medidor del estado de la batería.

La elección de la fuente de poder se determina bajo dos características técnicas: el voltaje nominal de la fuente y su capacidad. La elección del voltaje nominal de la fuente está sujeto a los voltajes de operación de cada componente que pueda ser alimentado directamente a través de la fuente, dados por los sensores del sistema y la CPU.

Se consideran dos voltajes de operación posibles para cada componente: 3.3 V y 5 V. El voltaje nominal de la fuente debe ser, por lo menos, 2 V mayor al del componente con mayor voltaje nominal.

En base al voltaje nominal de la fuente de poder y el voltaje de operación de los componentes, se procede a elegir el regulador de voltaje que media cada interacción entre fuente y componente. Para la elección del regulador de voltaje, también se considera el consumo pasivo de corriente del regulador, el cual debe ser menor a 1 mA.

Respecto a la capacidad de la fuente, ésta debe determinarse en función del consumo total

del sistema embebido, en cada ciclo de trabajo, y la duración mínima de 30 días continuos en la que debe estar el sistema funcionando. La capacidad de la fuente se determina a partir de la ecuación 3.1.

$$Capacidad[mAh] > Consumo_{D.C.}[mA] \cdot 720[h] \tag{3.1}$$

En la ecuación 3.1, $Consumo_{D.C.}$ se refiere al consumo medio de corriente, de un ciclo de trabajo del sistema embebido. El valor 720, representa la cantidad de horas en un mes (30 días).

El sistema de alimentación contempla la implementación de un medidor de batería baja. Este medidor está compuesto por un divisor de voltaje, para calcular el voltaje de la fuente y un LED cuyo fin es dar la alerta en caso de haber batería baja. En la figura 3.3 se aprecia la implementación general del divisor de voltaje.

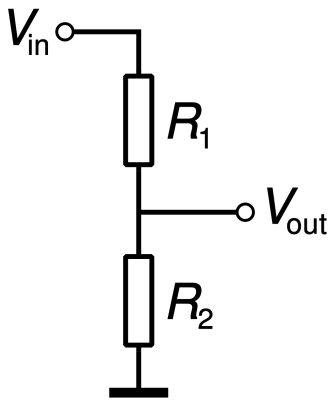

Figura 3.3: Diagrama circuital de un divisor de voltaje.

En la figura 3.3, V_{in} representa al voltaje de la batería, mientras que V_{out} corresponde al valor medido, finalmente, por la CPU del sistema. La necesidad de implementar el divisor de voltaje radica en que el voltaje de la batería puede ser superior al voltaje de operación de la CPU, por lo que se eligen R_1 y R_2 según la ecuación 3.2 para que V_{out} esté dentro del rango de operación de la CPU.

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2} \tag{3.2}$$

Para determinar el estado de batería baja y, por ende, encender el LED de alerta, se utiliza como umbral el 15 % de batería restante. Este valor se calcula como $1{,}15 \cdot V_{cut-off}$, donde $V_{cut-off}$ corresponde al voltaje de corte de la fuente de poder, determinado por su hoja de datos.

Finalmente, de requerirse un sistema de carga para la fuente, este comprenderá dos partes: un circuito cargador de protección y el cargador. La estructura del sistema de carga está dada por la figura 3.4.

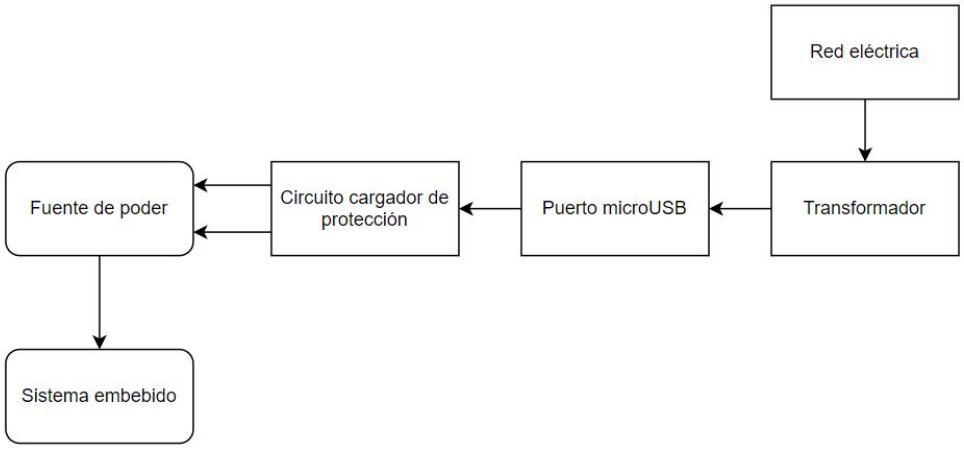

Figura 3.4: Diagrama de bloques del sistema de carga para la fuente de alimentación del sistema embebido.

3.2.2. Sensores

Dentro de los componentes del sistema embebido, los sensores se utilizan para la detección de los escenarios de activación de la alarma, E1 y E2. Se utilizará un sensor para cada escenario. Cada sensor se evalúa bajo los siguientes criterios, los que están ordenados de mayor a menor relevancia: robustez de la medición, consumo energético y costo. La evaluación bajo estos criterios se realiza en base a *ranking* entre los sensores utilizados para cada escenario.

Para cada sensor a analizar, se procede con la siguiente metodología:

1. Se establece como medir cada escenario.
2. Se define el rango de mediciones que puede generar el sensor en cuestión.
3. En base al rango de mediciones del punto anterior, se establece un umbral de activación para el escenario relevante al sensor en cuestión.

Para evaluar el consumo energético, se procede como sigue:

1. Se monta un circuito de pruebas para el sensor.
2. Se mide el consumo basal del circuito, sin conectar aún el sensor.
3. Se mide la variación de consumo energético del circuito, una vez conectado el sensor, para establecer su consumo en pasivo.
4. Se fuerza la activación del sensor para medir su consumo energético en activo.
5. Se establece el ciclo de trabajo del sensor para la medición de su escenario relevante, dado por el tiempo en activo y pasivo del sensor en cada período de muestreo.
6. En base a los puntos 4 y 5, se define el consumo medio del sensor, por período de muestreo.

Respecto al punto 6, la medición de consumo energético se hace midiendo la corriente utilizada por el sensor en cada ciclo de trabajo. Esto se realiza utilizando un multímetro, en las implementaciones de las figuras 3.5 y 3.6.

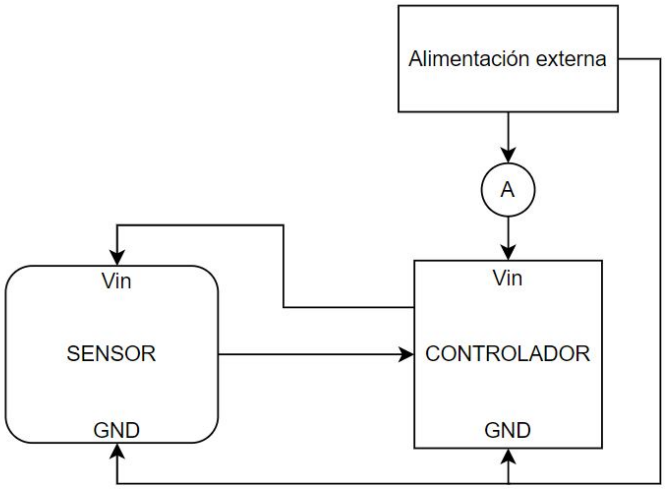

Figura 3.5: Esquema circuital para la medición del consumo energético de cada sensor desde la variación de consumo del controlador. El elemento nominado *'A'* corresponde al multímetro para la medición de corriente. Vin y GND corresponden a los pines de alimentación y tierra de cada componente, respectivamente.

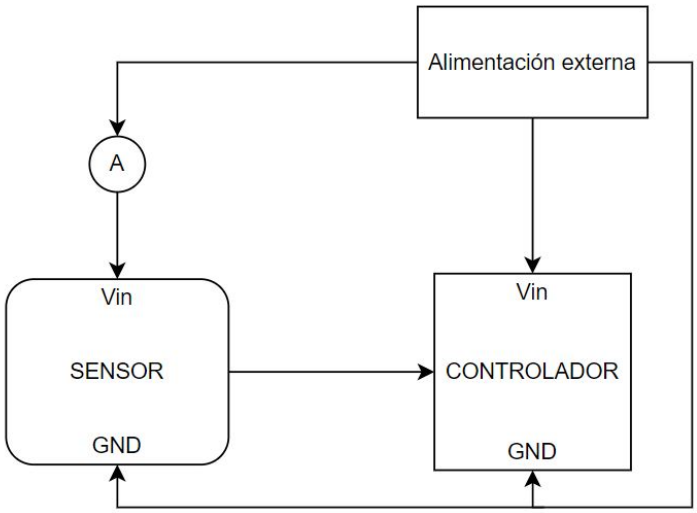

Figura 3.6: Esquema circuital para la medición del consumo energético directamente sobre el sensor. El elemento nominado *'A'* corresponde al multímetro para la medición de corriente. Vin y GND corresponden a los pines de alimentación y tierra de cada componente, respectivamente.

Mientras que el consumo medio se mide acorde a la ecuación 3.3.

$$CM[mA] = \frac{CA[mA] \cdot t + CP[mA] \cdot (1 - t)}{T_s},\ t \in [0, T_s] \tag{3.3}$$

en donde CM es el consumo medio de corriente, CA es el consumo activo (consumo de corriente del sensor cuando está midiendo), CP es el consumo pasivo (consumo de corriente del sensor cuando está inactivo), t es el tiempo que tarda el sensor en hacer la medición y T_s es el período de muestreo.

3.2.2.1. Detección de E1

Para la detección del escenario 1, en primer lugar, se hace un estudio de la puerta base para determinar que implica en su estructura, que no esté bien asegurada. Esto arroja una lista de 3 sensores candidatos para la medición del escenario 1:

- Sensor infrarrojo
- Sensor de ángulo
- Sensor magnético

3.2.2.2. Detección de E2

Análogamente al escenario 1, se hace estudio de la puerta base para determinar como reacciona en su estructura ante la aplicación de cargas sobre ella, replicando el peso de un infante al tratar de sobre pasarla. Dado lo anterior, se consideran 2 sensores para la medición de este escenario:

- Sensor ultrasónico
- Sensor de presión

3.2.3. Alarma sonora

Para la alarma sonora, se considera la utilización de un *buzzer*. La elección del modelo de buzzer se da en base a dos características: Su voltaje de operación, el que debe estar dentro del rango de operación de la CPU del sistema embebido, y su volumen máximo, el cuál debe estar por sobre o igual a 80 dB.

3.2.4. Controlador

El diseño del controlador se hará en base a un microcontrolador. Lo anterior se debe a que existen varios modelos de bajo costo que pueden realizar las tareas requeridas por el

controlador, además de que son fáciles de programar, por lo que se puede escalar sin mucha complejidad en el futuro. En línea con los criterios anteriores, la elección del modelo final estará dada por criterios de menor costo y menor consumo energético posible. En adición a estos parámetros, se considera, en menor medida, el método que requiera cada controlador para recargarle el *firmware* de control. Dado que se busca en el desarrollo del producto una futura producción masiva, en serie, se busca que la recarga del *firmware* requiera la menor intervención posible a nivel de implementación circuital. En base a lo anterior, existen dos posibilidades, ordenadas según su idoneidad:

1. Puerto serial universal integrado en el microcontrolador.
2. Uso de un convertidor USB-TTL, a través de los pines Tx y Rx del microcontrolador.

Para la medición de consumo energético, se procede como sigue.

1. Se hace recarga de un *firmware Bare Minimum*, es decir, que cumple la mínima funcionalidad del microcontrolador, sin ejecutar ninguna tarea.
2. Se conecta un sistema de alimentación externo al microcontrolador y se mide su consumo energético basal.
3. Se desglosa el consumo energético del microcontrolador en sus componentes, dados, principalmente, por el procesador, el regulador de voltaje integrado y el puerto serial (de haber).

La medición de consumo se hace análogamente a la figura 3.5, tal como muestra la figura 3.7.

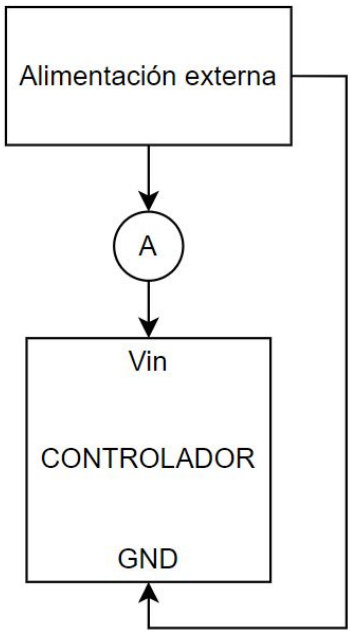

Figura 3.7: Esquema circuital para la medición del consumo del microcontrolador. *'A'* representa al multímetro medidor de corriente.

3.2.5. Diseño del *firmware* de control del sistema embebido

El diseño del *firmware* de control del sistema embebido pasa, en primer lugar, por la elección de la arquitectura de control, dadas por las opciones revisadas en el capítulo 2. Posterior a esto, se diseña la funcionalidad del sistema, en base a una máquina de estados de Moore.

El diseño del *firmware* debe considerar los siguientes atributos:

1. La elección de los pines de entrada para los sensores.

2. La elección de los pines de salida para la alarma y el indicador de batería baja.

3. Los tiempos de muestreo de los sensores.

4. Las condiciones de activación y desactivación de la alarma.

Además, se consideran los siguientes elementos, en relación a la reducción del consumo energético del sistema:

1. Desactivación de periféricos inactivos o inutilizados.

2. Desactivación del ADC en períodos inactivos.

3. Reducción de la velocidad de reloj mediante *Prescaler*.

4. Desactivación de sistema de *Brown Out Detection*.

5. Incorporación de rutinas de ultra bajo consumo, dadas por el microcontrolador en cuestión, para períodos de inactividad.

En la figura 3.8 se muestra un diagrama general del *firmware* del sistema embebido, desde la etapa de *setup* hasta el *loop* de control.

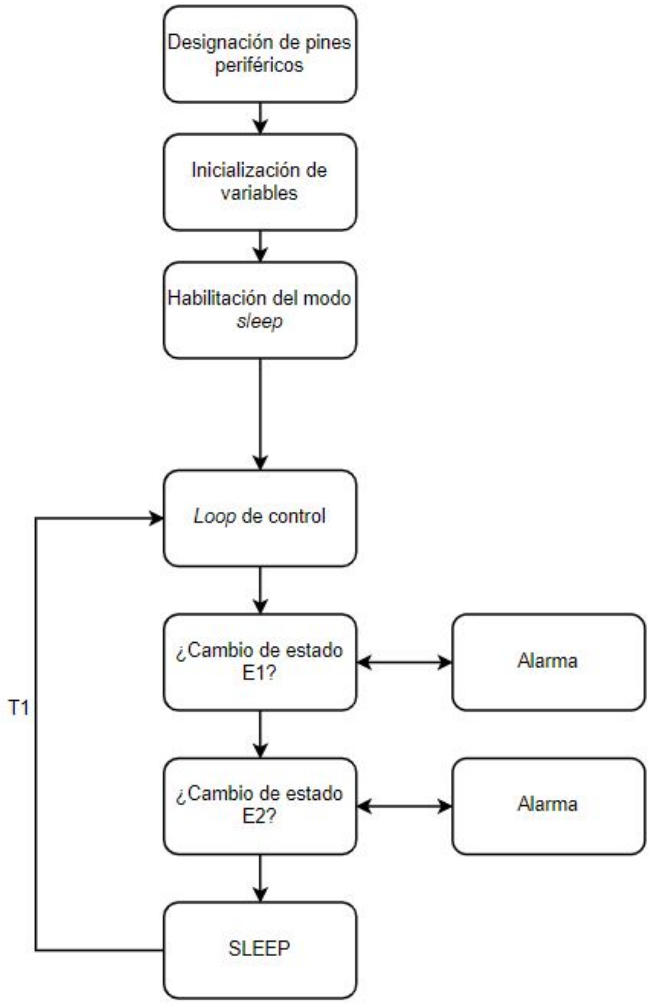

Figura 3.8: Diagrama de flujo del funcionamiento general del *firmware* del sistema embebido. Las flechas bidireccionales indican codependencia de eventos. T1 indica el período de activación, luego del modo sleep, para consultar los estados del sistema.

Capítulo 4

Implementación del sistema embebido

Posterior a la elección de los componentes, o los candidatos a componentes, del sistema embebido, se procede con el montaje del sistema. En esta etapa, se procede con la integración de los componentes, para evaluar el funcionamiento y el consumo energético de manera global. Esta evaluación puede arrojar resultados que induzcan a un rediseño del sistema si es que no cumple con los requerimientos. En la figura 4.1 se resume esquemáticamente esta etapa.

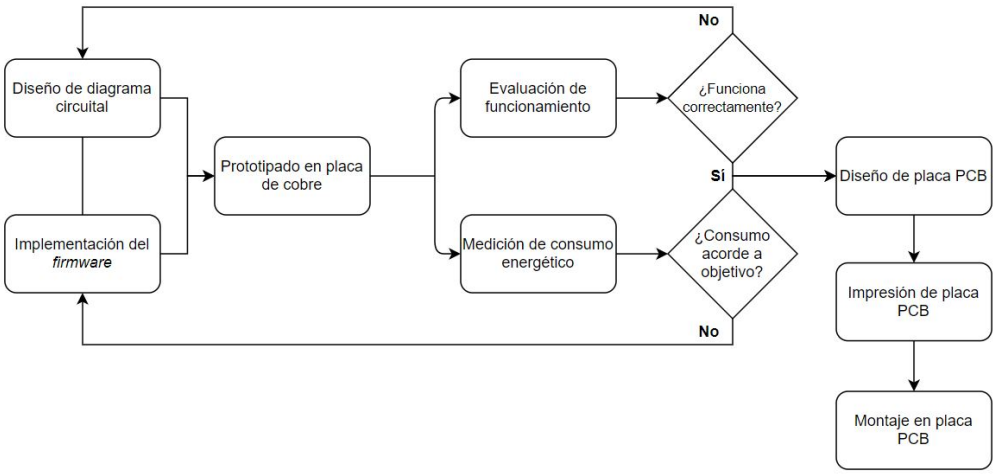

Figura 4.1: Diagrama de flujo para el proceso de implementación del sistema embebido.

4.1. Diagrama circuital e implementación del *firmware* de control

Una vez elegido un sistema de alimentación, los sensores, el microcontrolador y la bocina, se procede al diseño de un diagrama circuital integrado. Este debe cumplir con las siguientes consideraciones:

1. La distribución espacial de los componentes, en particular la de los sensores.

2. Las conexiones de los periféricos al microcontrolador.
3. Las conexiones a tierra de cada componente.
4. La alimentación de los periféricos desde el microcontrolador o directamente desde la fuente de poder.
5. La conexión de elementos auxiliares como transistores, condensadores y resistencias.

Respecto al *firmware* de control del sistema embebido, en esta etapa se completa su desarrollo, incorporando las conexiones de los periféricos con el microcontrolador, además de establecer las funciones de los pines de propósito general utilizados.

4.2. Prototipado en placa de cobre

Con el diagrama circuital diseñado, se procede a hacer un montaje experimental en placa de cobre. A través de este montaje se hace una evaluación del funcionamiento en general del sistema y una medición del consumo energético total del sistema. Esta medición de consumo energético va a estar dada por la suma de las corrientes de salida del sistema de alimentación, en caso de ser más de una, tal como muestra la figura 4.2.

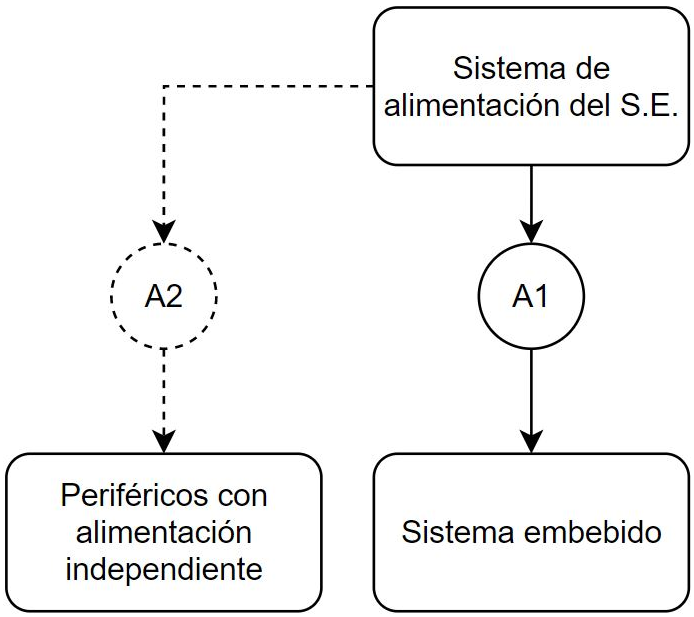

Figura 4.2: Diagrama para la medición del consumo energético total del sistema embebido. $A1$ representa al multímetro para la medición de corriente para el sistema embebido, mientras que $A2$ representa al multímetro para la medición de corriente de los periféricos alimentados de manera independiente. Se denota con línea punteada los elementos que no son obligatorios para la implementación final.

Para la implementación del sistema de producción, el prototipo debe considerar lo siguiente:

1. Una ubicación estimada del núcleo del sistema embebido en la puerta base.
2. La ubicación exacta de los sensores para la medición de los escenarios, en la puerta base.
3. En base a los puntos 1 y 2, el largo de los conectores y su recorrido desde la posición final de los sensores y la posición del núcleo del sistema embebido.

Como fue mencionado, se hace una evaluación de consumo y funcionamiento del sistema a través del prototipo. En caso de haber evaluación negativa, se procede a iterar nuevamente desde el diseño del diagrama circuital y el *firmware* de control. Solo en caso de tener un funcionamiento correcto del sistema, más un consumo energético que cumple con el objetivo de duración continua, se procede con el diseño del sistema para montaje en placa PCB.

4.3. Impresión y montaje en placa PCB

La evaluación exitosa del funcionamiento del sistema embebido más el cumplimiento del objetivo de consumo energético generan la aprobación del diseño circuital, por lo que se procede a trasladar el diseño a una placa PCB. Para ello, se utiliza el software de diseño de circuitos y PCB *EasyEDA*.

Luego del diseño, y para evaluar la distribución de los componentes en la placa acorde al diseño, y así corroborar que los componentes utilizados en el software son equivalentes a los utilizados en la realidad, se procede con un prototipado rápido de la PCB, utilizando una placa recubierta de cobre conductor. Para el prototipado, se procede como sigue:

1. Se toma el diseño de la PCB y se imprime a escala en papel térmico, con los elementos reflejados especularmente.
2. Se pega la impresión sobre la placa recubierta de cobre, con el lado de la tinta hacia el cobre.
3. Se procede a planchar sobre la placa y el papel, utilizando una plancha convencional para ropa, por, aproximadamente, 10 minutos.
4. Se remueve el papel de la placa de cobre, comprobando que el diseño que estaba impreso se traspasó a la placa de cobre por conducción térmica.
5. Se sumerge la placa en ácido férrico ($FeCl_3$) por, aproximadamente, 30 minutos.
6. Se remueve la placa del ácido, procurando no entrar en contacto directo con éste, y se remueve el exceso con agua.
7. Se remueve la tinta de la placa con acetona.
8. Se perfora en los puntos donde se colocan los componentes, utilizando un *Dremel*.

De resultar correcto el proceso anterior, el ácido férrico habrá removido todo el cobre de la placa, excepto el que estaba cubierto por la tinta, dejando las pistas conductoras para el montaje del sistema embebido.

Posteriormente, se realiza el montaje del sistema embebido en esta placa de prototipado de PCB, y se evalúa su correcto funcionamiento. Si funciona correctamente, se procede con

una impresión de placa PCB de manera industrial.

4.4. Carcasa de contención del sistema embebido

Una vez que se tiene el sistema embebido en su versión final, montado en placa PCB, se procede con el diseño de la carcasa de contención del sistema embebido. Ésta debe tener las siguientes consideraciones:

1. La posición final del sistema embebido en la puerta base.
2. La posición final de los sensores en la puerta base.
3. La distribución de los periféricos del sistema embebido, considerando, además, la salida de conectores en una instalación distribuida.
4. Una pieza auxiliar que contenga el sensor receptor para la detección de E1, de ser necesario.
5. Una pieza auxiliar para la medición de E2.
6. La protección de los componentes ante el movimiento de la puerta.
7. La necesidad de recargar la fuente de poder del sistema embebido.
8. La protección del *switch* de encendido, y el botón de desactivación de la alarma, ante la intervención del (los) infante(s) presentes durante el uso de la puerta.

La implementación se realizará mediante impresión 3D. El diseño en 3D se realiza utilizando el software *Rhinoceros 3D*. Para ello, se debe considerar:

1. El material para producción (PLA, ABS o Resina).
2. La técnica de impresión (SLA o FDM)
3. La orientación de la carcasa para impresión, la cual influye en el acabado final.
4. La división de la estructura total en partes, para insertar el sistema embebido y, posteriormente, sellar la carcasa.
5. El color de la carcasa, acorde al diseño de producción de la puerta base.

Tomando estas consideraciones, se procede con la implementación de un prototipo para la carcasa, el que tiene como objetivo la contención correcta del sistema embebido, en términos de sujetar correctamente todos sus componentes, dar el espacio suficiente y bien colocado de sus periféricos, permitir el correcto funcionamiento de los sensores y no bloquear la salida de la alarma sonora. Esto puede demandar modificaciones menores a la distribución de los componentes en la placa PCB, lo que induce a iteraciones de la versión final de ésta.

4.5. Integración con la puerta base

A partir de la integración exitosa del sistema electrónico embebido con su carcasa de contención, se procede a iterar sobre el diseño exterior de la carcasa de contención para

integrarla en la puerta base. Esto debe considerar las sujeciones del sistema principal a la puerta, además de la colocación de piezas auxiliares para los sensores. La integración exitosa del sistema embebido en la puerta base debe permitir:

1. La detección robusta de cada escenario acorde a los casos de uso objetivo.
2. El funcionamiento normal de la puerta base, evaluado según como se utiliza sin la presencia del sistema embebido en ella.
3. La sujeción firme del sistema embebido, considerando movimientos bruscos de la puerta base durante su uso y/o la manipulación de infantes.

La integración exitosa del sistema embebido en la puerta base da cuenta del diseño definitivo para producción de la carcasa de contención. Sumado lo anterior a la implementación definitiva del sistema electrónico, se tiene como resultado la implementación del producto final para producción masiva.

4.6. Estimación del costo unitario del producto final

Para la estimación del costo unitario del producto final, se considera una cantidad inicial de 100 unidades. El desglose, para el cálculo del costo total unitario, se muestra a continuación.

1. Costo total de componentes electrónicos del sistema embebido.
2. Costo de la fuente de poder.
3. Costo de producción de placas PCB.
4. Costo de carcasa de contención.
5. Costo de puerta base, incluyendo *packaging*.
6. Costo de HHs para montaje de una unidad.

Para el cálculo del costo de la electrónica, se utiliza el portal de venta de componentes electrónicos *Mouser*. Las placas PCB se estiman mediante cotización en el sitio web de la fábrica *JLCPCB*. El costo de la carcasa se obtendrá mediante cotización en fábricas de impresión 3D en China, mientras que la puerta también será cotizada para producción en China.

Para la estimación del costo de montaje, se realiza una implementación completa de una unidad y se mide el tiempo que tarda esta. En base a este tiempo, se calcula la cantidad de unidades que se pueden producir en un mes, en base a una jornada diaria de 8 horas y 22 días hábiles. Se establece un sueldo mensual de CLP450.000 para, finalmente, establecer el costo unitario de montaje por unidad.

El costo total de la puerta está dado por la suma de todos los elementos de la lista detallada arriba.

4.7. Implementación de primera versión

4.7.1. CPU

El diseño, inicialmente, se basa en el microcontrolador NodeMCU, de la compañía *Espressif*, el que tiene un costo de USD1.80 unitario, para 100 unidades.

El microcontrolador cuenta con un puerto serial CP2102 integrado en su placa, que permite la recarga de software a través de una conexión con cable microUSB a un computador de escritorio. La programación puede realizarse a través del lenguaje C++ en el entorno de desarrollo de *Arduino* (Arduino IDE).

Además, cuenta con un regulador de voltaje AMS1117 integrado en placa, que permite alimentar el microcontrolador con una fuente entre los 5 V y los 12 V. El voltaje de operación nominal del NodeMCU es de 3.3 V, aunque funciona correctamente dentro del rango [3.0, 3.3] V.

La medición de consumo basal, acorde a la figura 3.7 y el procedimiento de la sección 3.2.4, arroja un consumo de corriente en el rango [83.5, 84.3] mA. Este consumo requeriría de 60.120 mAh de capacidad en la fuente de alimentación para lograr el objetivo de consumo, lo que es absolutamente inviable, pero considerando que este microcontrolador cuenta con capacidad de conexión Bluetooth y WiFi, que en comunicación pueden alcanzar *peaks* de corriente de hasta 200 mA, se agrega un limitador de funcionamiento de la CPU WiFi del NodeMCU en el *firmware Bare Minimum*, logrando un consumo de corriente en el rango [15.8, 16.4] mA. Sumado lo anterior a los modos *sleep* del microcontrolador dados por la tabla 2.4, se considera factible su utilización.

4.7.2. Alimentación

Para la alimentación del sistema, se opta por la utilización de 4 pilas AA. Estas se disponen en serie, como muestra la figura 4.3. La disposición en serie otorga un voltaje total de 6 V, dado por la suma de los voltajes individuales de cada batería. El voltaje entregado está dentro del rango requerido por el microcontrolador NodeMCU. En contraparte, la capacidad de las 4 pilas en conjunto es la misma que la de 1 pila. La capacidad de una pila AA oscila en el rango [1100, 2500] mAh. Para efectos del cálculo de la autonomía, se considera el peor caso (1100 mAh).

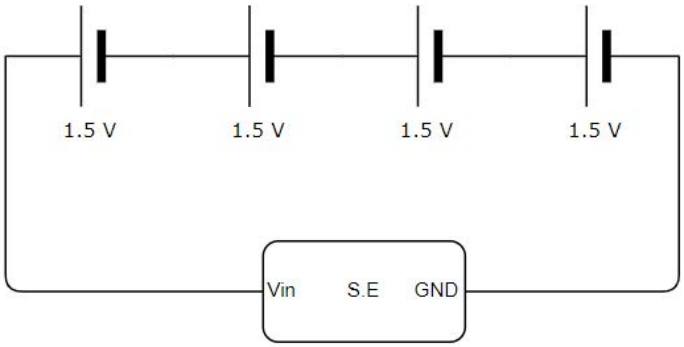

Figura 4.3: Conexión utilizada para la alimentación del sistema embebido basado en 4 pilas AA.

Como enseña esquemáticamente la figura 4.3, la fuente se conecta al pin Vin del NodeMCU, el cuál está conectado al regulador de voltaje AMS1117 integrado del NodeMCU, por lo que no se considera un regulador externo. La alimentación en base a pilas no contempla la adición de un circuito cargador para ellas.

En base a la capacidad especificada y la ecuación 3.1, el consumo medio del sistema ($Consumo_{D.C.}$) debe ser menor o igual a 1.5 mA. Para mediar el encendido y apagado del sistema, se utiliza un interruptor tipo *Rocker* como el que se aprecia en la figura 4.4.

Figura 4.4: *Rocker Switch* utilizado para el encendido y apagado del sistema embebido.

Para el divisor de voltaje se utilizó $R1 = R2 = 680k\Omega$. Con esto, según la ecuación 3.2, $V_{out} = V_{in}/2 = 3V$. La conexión de V_{out} se realiza en un pin análogo del NodeMCU, el que está conectado a el ADC integrado de 10 bits. Esto quiere decir que el rango de valores medidos es [0, 1024] para mediciones de voltaje entre [0, 3.3] V, respectivamente. Se utiliza un LED amarillo para la alerta de batería baja, conectado directamente al pin digital D2 del NodeMCU, sin resistencia mediante.

4.7.3. Sensores

4.7.3.1. Detección de E1

El estudio de la puerta base dio cuenta de la siguiente situación, que propicia a E1, reflejado en la figura 4.5.

Figura 4.5: Resultado común del cierre automático de la puerta base.

La puerta base cuenta con un mecanismo de cierre automático, dado por un resorte que deja que la puerta se bata sola luego de abierta. El uso reiterado de este mecanismo dio cuenta de lo que se observa en la figura 4.5, en donde el seguro de la puerta evita el cierre efectivo de ésta, dado que no se bate con suficiente fuerza, lo que genera una desalineación del marco de la puerta con esta misma. Se utiliza esta desalineación para detectar E1, mediante los sensores infrarrojo, de ángulo y magnético descritos en el capítulo 2.

- **Sensor infrarrojo**

El esquema de utilización del sensor infrarrojo para la detección de E1 se observa en la figura 4.6.

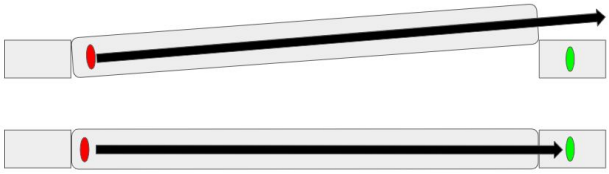

Figura 4.6: Esquema de utilización del sensor infrarrojo para la detección del escenario 1. El esquema corresponde a una vista superior de la puerta base. En rojo, el emisor del sensor. En verde, el receptor

EN base a la situación de la figura 4.5, se plantea detectar el escenario cuando el receptor no reciba la señal del emisor, acorde a la vista superior de la figura 4.6. Finalmente, se descartó el uso del sensor porque el haz del emisor tiene una apertura tal que el receptor igual capta la señal, aunque la puerta no esté bien cerrada, lo que derivaba en falsos negativos.

Por otro lado, este sensor tiene un costo de USD0.025 por unidad, pero un consumo energético alto, dado por la emisión de la señal del par emisor.

- **Sensor de ángulo**

El esquema de utilización del sensor de ángulo para la detección de E1 se observa en la figura 4.7.

Figura 4.7: Esquema de utilización del sensor de ángulo para la detección del escenario 1. El esquema corresponde a una vista superior de la puerta base.

La detección de E1 a través de este sensor, se realiza mediante la detección de la rotación del eje de la puerta. El sensor se instala en la parte superior de la puerta, justo sobre su eje de rotación y se establece el valor 0 para cuando la puerta está cerrada. Cualquier valor medido mayor a 0 implica que la puerta está abierta. La ventaja de este sensor es que podría utilizarse para determinar cualquier grado de apertura de la puerta, no sólo el estado binario de puerta correcta o incorrectamente cerrada. De todas maneras, se descartó su uso porque la implementación es de alta complejidad comparado con las otras dos alternativas, ya que requiere mantener la base del sensor estática e incorporar un eje que calce con su centro y que se integre al eje de la puerta, para medir la rotación. Además, las mediciones de ángulo resultaron muy ruidosas, dado que no se logró medir el 0 al cerrar la puerta en mediciones consecutivas.

En relación al costo y consumo energético, este sensor tiene un valor de USD0.8 por unidad y un consumo energético despreciable.

- **Sensor magnético**

El esquema de utilización del sensor magnético para la detección de E1 se observa en la figura 4.8.

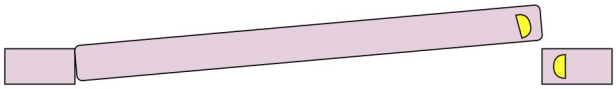

Figura 4.8: Esquema de utilización del sensor magnético para la detección del escenario 1. El esquema corresponde a una vista superior de la puerta base. En amarillo, ambas partes del sensor magnético.

La detección de E1 a través de este sensor se realiza colocando el extremo del sensor que se conecta al sistema embebido en el marco de la puerta, mientras que el otro extremo se coloca en la puerta como tal, frente a frente. Mientras la puerta está cerrada, ambas partes se enfrentan y cierran el circuito y cuando la puerta se abre, las partes se separan y el circuito se abre. Las pruebas en la puerta con este sensor tuvieron 100 % de eficacia, logrando medir la apertura de la puerta en cada iteración. Sumado a lo anterior, este sensor solo genera una adición de consumo energético de 0.1 mA cuando se cierra el circuito, y presenta un costo

unitario de USD0.28, por lo que es elegido como el sensor a utilizar para la medición del escenario 1.

4.7.3.2. Detección de E2

El estudio de la puerta base para determinar como detectar el escenario 2, dio cuenta de una deformación en el seguro inferior de la puerta base al someterla a cargas (ver figura 4.9), simulando un infante que trata de sobrepasarla. Esta deformación genera una presión sobre la parte inferior del marco que se utiliza para la detección mediante el uso de un sensor de presión.

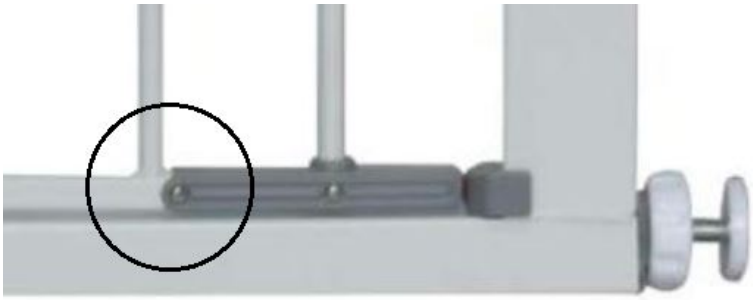

Figura 4.9: Acercamiento a la zona de la puerta que se ve deformada al someterla a presión, generando contacto entre la puerta y el marco.

Aplicando una carga de 5 kilogramos se logra la deformación necesaria para que el sensor detecte el contacto. Esto deriva en una eficacia de casi el 100 % en la detección, dado que, según la evolución estadística de la masa corporal de un infante, dada por las figuras A.1 y A.2, un 97 % de los infantes de 1 año en adelante pesan más de 7 kilos (damas) y 8 kilos (varones).

Por otro lado, dado el rango de valores resistivos que toma el sensor (ver figura 2.10), se tiene un consumo energético despreciable cuando el sensor no está presionado. Sin embargo, este sensor tiene un costo muy alto, igual a USD4.88

Como alternativa a este sensor, se utiliza un sensor ultrasónico, cuya utilización se resume en la figura 4.10.

Figura 4.10: Esquema de implementación del sensor ultrasónico (en rojo) para la medición del escenario 2.

Para la detección del escenario 2, se plantea que la distancia a medir por el sensor ultrasónico será de 76 centímetros, aproximadamente, correspondiente al ancho de la puerta, cuando ésta esté libre de carga. Si un infante intenta sobrepasar la puerta, interrumpirá el haz ultrasónico del sensor y la distancia medida será menor, incurriendo en la detección del escenario. Las mediciones de distancia de este sensor tienen una alta precisión, con un margen de error menor a 1 centímetro. Sin embargo, cuando la interrupción se genera muy cerca del sensor, la distancia medida se vuelve errática, arrojando valores mayores al real y mayores, también, a 76 centímetros, por lo que el escenario no era detectado con un alto porcentaje de efectividad.

Por otro lado, el sensor ultrasónico presenta un consumo pasivo constante de 2 mA y 15 mA de consumo cuando emite el haz ultrasónico. El único criterio a su favor respecto al sensor de presión es el costo, que es de USD0.57, casi 9 veces menor que el costo del sensor de presión.

En base a lo planteado anteriormente, se elige, finalmente, al sensor de presión como el sensor a utilizar para la medición del escenario 2, dada la robustez de sus mediciones y su despreciable consumo energético.

4.7.4. Alarma sonora

Para la alarma sonora se utiliza un *buzzer* activo, el que se puede apreciar en la figura 4.11. Este tiene un voltaje de operación de 3V, lo que está dentro del rango de operación del NodeMCU, y consume 5 mA, aproximadamente, cuando está sonando, lo que significa un sonido de volumen de 80 dB (valor nominal) a una frecuencia de 2300 Hz, aproximadamente.

Figura 4.11: *Buzzer* utilizado como alarma sonora del sistema embebido.

4.7.5. Firmware de control del sistema embebido

El *firmware* de control implementado en el NodeMCU tiene la arquitectura de un *Simple Control Loop* y fue diseñado como una máquina de estados de Moore. En la figura 4.12 se aprecia el diagrama de estados del sistema embebido.

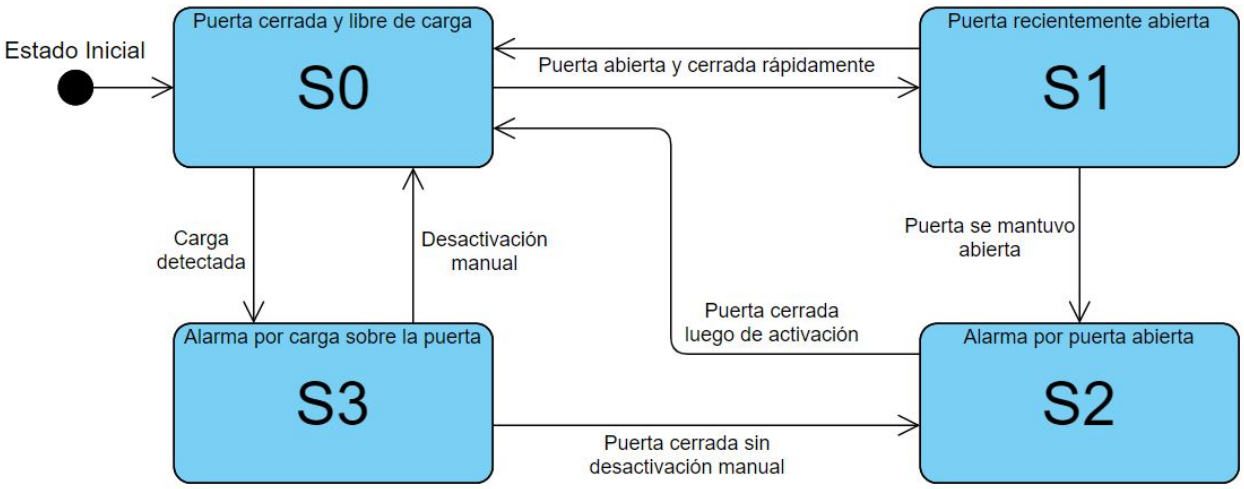

Figura 4.12: Diagrama de estados que representa el *firmware* de control del sistema embebido, basado en el microcontrolador NodeMCU.

El diseño contempla los siguientes elementos:

- Q = {S0, S1, S2, y S3} son los estados.
- E = {E_{puerta}, $E_{presión}$, T, M} son las entradas. Cada una toma un valor binario y representan, respectivamente, Puerta abierta o cerrada, Puerta presionada o no presionada, *deadline* vencido o no vencido, Alarma desactivada o no desactivada manualmente).
- S = {A, I} son las salidas. A=Alarma activa, I=Alarma Inactiva.

Se considera un estado inicial que deriva en S0, representando la puerta libre de carga

y cerrada correctamente. La salida en este estado es I. Si $E_{puerta} = 1$, Se pasa a un estado intermedio S1 de puerta abierta, pero en el cual la salida aún es I. La máquina de estados se mantiene en S1 si $E_{puerta} = 1$ y $T = 0$. El *deadline* corresponde a 5 segundos y su fin es que se pueda utilizar la puerta normalmente sin activar innecesariamente la alarma.

Si transcurren los 5 segundos y $E_{puerta} = 1$ aún, se pasa a un estado de alarma por puerta abierta S2, en donde la salida es A, es decir, se activa la alarma. La alarma sólo se detendrá si cierro correctamente la puerta, es decir, $E_{puerta} = 0$ y paso a S0 nuevamente,.

Por otro lado, si la puerta está cerrada y un infante trata de sobrepasarla, activando el sensor de presión, $E_{presión} = 1$ y se pasa desde S0 a S3, en donde la salida es A. Sólo se puede volver a S0 mediante $M = 1$, es decir, presionando el botón de desactivación manual de la alarma.

4.8. Ensamble del primer prototipo

En la figura 4.13 se denota el diagrama circuital para la implementación del sistema embebido a partir del NodeMCU.

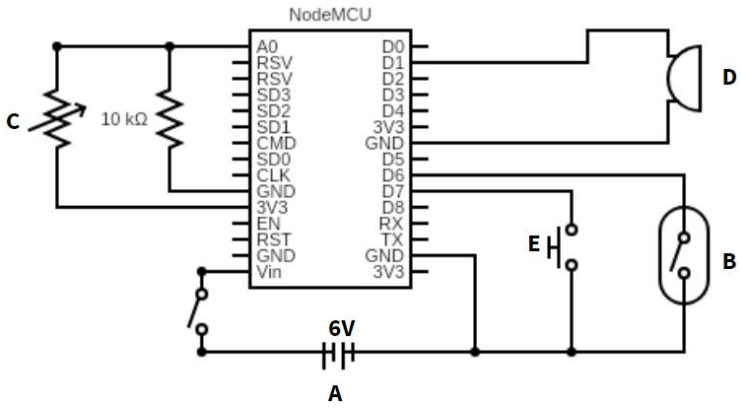

Figura 4.13: Diagrama circuital para la implementación del sistema embebido utilizando el microcontrolador NodeMCU.

En la sección A.2 del Anexo se aprecia el código de control para la implementación de la figura 4.13. La implementación se desglosa como se denota a continuación:

1. **Alimentación del sistema embebido (A).** Se conecta al pin V_{in} del NodeMCU, el cual es el que tiene conexión con el regulador de voltaje AMS1117, y tierra (GND). Para el indicador de batería baja, el ADC arroja 930, en el rango [0, 1024] como valor para 100 % de batería. En base al voltaje de corte de una pila AA (1.3 V) se define 800 como el valor a medir por el ADC cuando se agoten las pilas. En base a este rango [800, 930], la activación del indicador de batería baja se realizará cuando la medición sea menor a 820.

2. **Sensor magnético (B).** El sensor magnético se conecta al pin digital D6, que se configura como *INPUT PULLUP*, y tierra (GND). Esto quiere decir que se utiliza una resistencia interna del NodeMCU para que el pin esté por defecto a 3.3 V. De esta manera, cuando la puerta está cerrada, el circuito está cerrado y la conexión a GND hace que el pin D6 esté a 0 V (0 lógico). Al abrirse la puerta, se abre el circuito del sensor magnético y el pin D6 pasa a su estado por defecto, 3.3 V (1 lógico) detectando la apertura de la puerta.
3. **Sensor de presión (C).** El sensor de presión al ser, en esencia, una resistencia, se conecta como divisor de voltaje junto a una resistencia de 10 $k\Omega$, valor determinado de manera empírica. Siguiendo la figura 3.3, El sensor de presión es R1, mientras que la resistencia de 10 $k\Omega$ es R2. Si el valor dado por el ADC es mayor a 900, en el rango [0, 1024], se interpreta que el sensor está presionado, dando lugar a la detección de E2. El valor 900 también fue definido de manera empírica.
4. **Alarma (D).** La alarma se conecta al pin digital D1 y a GND. Se activa al asignar 3.3 V al pin D1.
5. **Botón de desactivación manual (E).** Análogo al sensor magnético, este botón, dado por un pulsador táctil, se conecta a un pin digital (D7) definido como *INPUT PULLUP* y GND, de tal manera de pasar de 1 a 0 lógico (3.3 V a 0 V) al ser pulsado, procediendo a desactivar la alarma.

4.8.1. Montaje en placa de cobre

En la figura 4.14 se observa la implementación del sistema embebido en placa de cobre.

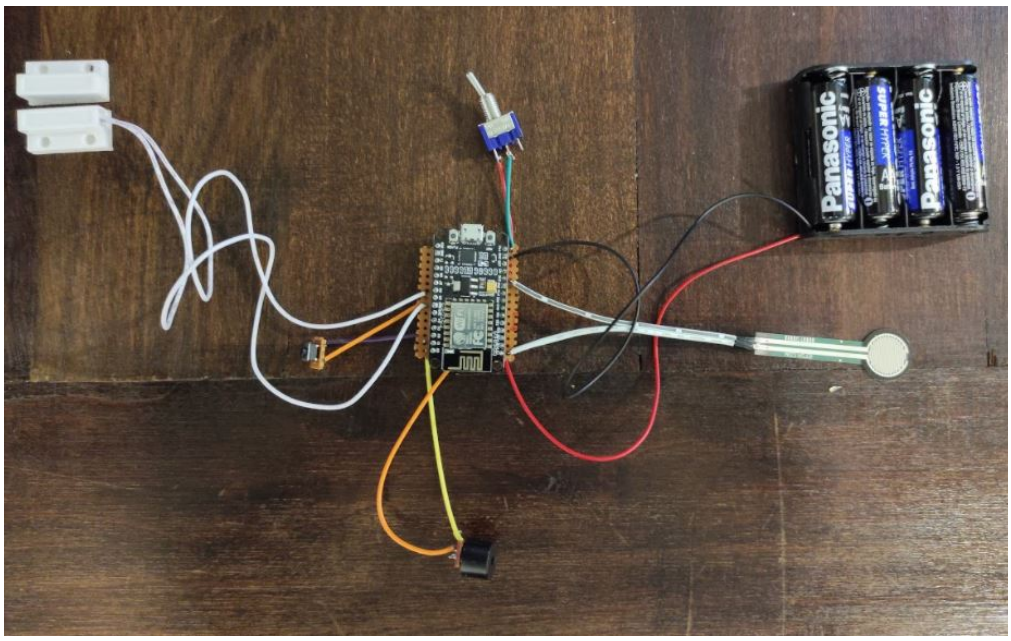

Figura 4.14: Implementación del sistema embebido, basado en el microcontrolador NodeMCU, en placa de cobre.

Esta implementación presenta un funcionamiento correcto del sistema a nivel de laboratorio, esto es, activando manualmente los escenarios 1 y 2, fuera del ambiente real de producción, dado por la instalación en la puerta base.

4.8.2. Reducción y evaluación del consumo energético del sistema embebido

Dado el correcto funcionamiento de la primera implementación del prototipo en placa de cobre, se procede a evaluar el consumo energético del sistema embebido. Para ello, y en primer lugar, se implementa la función *deep sleep* del NodeMCU. Esta función permite reducir el consumo energético del procesador ESP8266 a alrededor de 20 μA.

La implementación conlleva las siguientes modificaciones al diagrama circuital y el *firmware* de control:

1. Conectar el pin digital D0 con el pin *Reset* (RST) del NodeMCU.
2. Trasladar el código de ejecución del *Single Control Loop* del NodeMCU a la función *setup()* y agregar la función de *deep sleep* al final de ésta, designando un período de activación de 1 segundo.

De esta manera, cada un segundo, el NodeMCU activa el pin RST, mediante un 0 lógico aplicado desde el pin D0 (el pin RST está por defecto a 3.3 V). Esto deriva en un *full reset* del NodeMCU, por lo que el código de control vuelve a ejecutarse desde el inicio del *setup()*, ejecutando una iteración del código de control dado por la máquina de estados, hasta que al final se vuelve a ejecutar la función de *deep sleep* y el NodeMCU reduce su consumo por 1 segundo, dando pie a una nueva iteración.

Con la implementación de la rutina de reducción de bajo consumo, se procede a evaluar el consumo total del sistema embebido, a través del montaje presentado en la figura 4.2. La implementación realizada sólo contempla la alimentación de los periféricos a través del NodeMCU, por lo que se omite el uso del multímetro A2 de la figura. Los resultados se observan en la tabla 4.1.

Tabla 4.1: Consumo de corriente y tiempo de ejecución de cada modo de funcionamiento del sistema embebido basado en el microcontrolador NodeMCU.

Estado del sistema	**Consumo [mA]**	**Tiempo de ejecución**
Modo *deep sleep*	9.2	1 s
Modo normal	84	<1 ms
Alarma activada	94	-

Los resultados denotados en la tabla 4.1 dan cuenta de un consumo absolutamente inviable con respecto al objetivo de mantener al sistema funcionando de manera continua por, por lo menos, 30 días. Sólo considerando el consumo en modo *deep sleep*, se tiene que, utilizando como alimentación pilas AA en serie, la duración máxima posible es de 11.3 días en el mejor

caso (pilas AA con capacidad de 2500 mAh).

Ante lo anterior, se estudia una modificación al contenedor de las pilas, con el fin de utilizar pares en paralelo, lo que duplicaría la capacidad total. Sin embargo, el voltaje de operación se vería reducido a la mitad, equivalente a 3 V, justo en el límite inferior del voltaje de operación del NodeMCU. Además, duplicar la capacidad solo permitiría llegar a un mejor caso de 22.6 días, aún por debajo de los 30 días requeridos.

4.9. Rediseño del sistema embebido

En primer lugar, se hizo un análisis de porqué el consumo del sistema embebido llega a 9.2 mA si la función *deep sleep* implica un consumo del procesador de, aproximadamente 20 μA. Se hizo un desglose del consumo del microcontrolador por partes, y se midió un consumo pasivo cercano a los 4 mA en el puerto serial CP2102 y también en el regulador de voltaje AMS1117. Esta medición concuerda con lo mostrado en las hojas de datos de estos componentes.

Por ende, se procede a rediseñar el controlador del sistema embebido, procurando utilizar un microcontrolador que cumpla con el requerimiento de consumo desde la medición basal, previamente implementadas todas las rutinas de bajo consumo que tenga integradas.

4.9.1. Arduino Pro Mini

Se analiza el uso del microcontrolador Arduino Pro Mini. Este no cuenta con puerto serial de comunicación para la conexión directa al puerto USB del computador, por lo que requiere de un convertidor USB-TTL para recargarle el *firmware* de control. Con lo anterior se ahorra el consumo energético que tiene este componente cuando está integrado en la placa del microcontrolador.

El Arduino Pro Mini también cuenta con un rango de operación de [3.0, 3.3] V, pero el regulador de voltaje que tiene integrado en placa es el MC5205 de la compañía *Microchip*, el que tiene un consumo pasivo de corriente menor a 5 μA, acorde a su hoja de datos.

Se recarga el *firmware Bare Minimum* para la medición del consumo energético basal de este microcontrolador, la que arroja como resultado un consumo de 5.5 mA. Este microcontrolador cuenta con varios modos de bajo consumo, al igual que el NodeMCU, por lo que se procede con la implementación.

Respecto al *firmware* de control para el Arduino Pro Mini, y considerando los modos de bajo consumo que tiene este, se opta por una nueva arquitectura, basada en rutinas de interrupción (ISR, por sus siglas en inglés) en vez de utilizar un *Single Control Loop*, diseñado en base a una máquina de estados. La implementación de esta arquitectura permite mantener al sistema en un modo de bajo consumo constante, hasta que la detección de un escenario lo lleva al funcionamiento normal con el fin de activar la alarma. De esta manera, se evita

activar de manera periódica al sistema embebido, incurriendo en un gasto de energía, quizás, innecesario, tal como ocurría con la implementación basada en el NodeMCU.

El Arduino Pro Mini cuenta con 2 pines GPIO con manejo de ISR. Estos 2 pines son digitales, por lo que la implementación del sensor de presión, para la detección del escenario 2, como un divisor de voltaje carece ya de sentido. Sin embargo, dado que el sensor cuenta con un rango muy amplio para su resistividad, al conectarlo al pin digital para activar la ISR del escenario 2, el sensor se comportaba como un botón *switch* táctil. Dado lo anterior, y agregando el muy alto costo que tiene el sensor, en relación al resto de los componentes, se reemplaza directamente por un pulsador táctil para detectar el escenario 2, mediante la pulsación de este al cargar la puerta base.

4.9.2. Alimentación del sistema basado en el Arduino Pro Mini

De todas maneras, se opta por rediseñar el sistema de alimentación. Se procede a utilizar una batería LiPo de 2000 mAh de capacidad, la que opera en el rango [3.0, 4.2] V, correspondientes al voltaje de corte y el voltaje de carga máxima, respectivamente. Esta batería es de carácter recargable.

Para recargar la batería, se incorpora al diseño al circuito cargador TP4056, junto con un puerto microUSB e indicadores de carga y carga completa, dados por luces LED integrados en el circuito. El circuito cargador se aprecia en la figura 4.15.

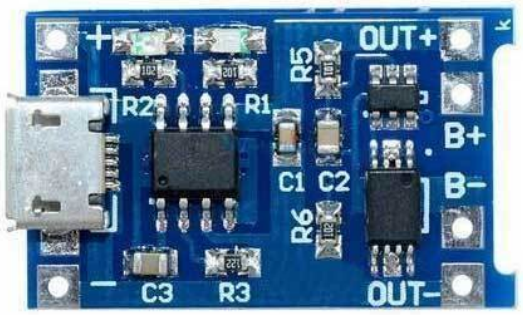

Figura 4.15: Circuito Integrado TP4056 para la recarga de baterías utilizando un puerto microUSB.

4.10. Implementación basada en el Arduino Pro Mini

En la figura 4.16, se denota el diagrama circuital para el sistema embebido, en base al rediseño realizado utilizando el microcontrolador Arduino Pro Mini.

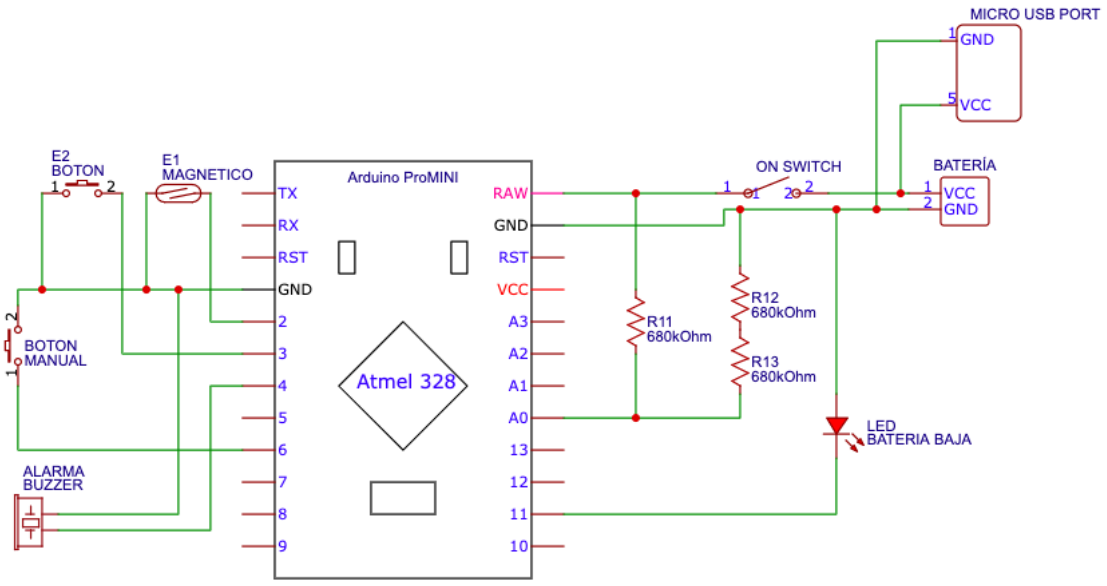

Figura 4.16: Diagrama circuital del sistema embebido en base al microcontrolador Arduino Pro Mini. Se incorporan la batería recargable, junto con su circuito de carga con conexión microUSB, y se reemplaza el sensor de presión por un pulsador táctil.

El sensor magnético, para la detección de E1, y el pulsador táctil, para la detección de E2, se conectan a los pines digitales GPIO 2 y 3, respectivamente, los que soportan la implementación de ISR a partir de sus cambios de estado. Se mantiene el botón de desactivación manual, con la misma implementación que con el NodeMCU, conectado al pin digital GPIO 6, mientras que el *buzzer* de la alarma sonora se conecta al pin digital GPIO número 4.

A la derecha de la figura 4.16 se resumen todos los componentes del sistema de alimentación. En primer lugar, se tiene la batería, conectada al *Rocker Switch* para el encendido y apagado del sistema, y al circuito cargador TP4056 para recargarla. Además, se implementa un divisor de voltaje basado en tres resistencias de $680k\Omega$ ($R1 = 680k\Omega$ y $R2 = 2 \cdot 680k\Omega$, con $R1$ y $R2$ acorde a la figura 3.3), que permite la lectura de el voltaje de la batería (V_{in}), a través del pin análogo A0, como $V_{out} = 2/3 \cdot V_{in}$.

En la tabla 4.2 se denotan los valores relevantes a la medición del estado de la batería por parte del pin A0, determinados de manera empírica a través del Arduino Pro Mini. El ADC del Arduino Pro Mini es de 10 bits, al igual que el NodeMCU, por lo que arroja mediciones en el rango [0, 1024], para valores de voltaje en el rango [0, 3.3] V, respectivamente. El LED indicador de batería baja parpadeará periódicamente cuando el nivel de la batería esté bajo el 15 %.

Tabla 4.2: Mediciones de voltaje relevantes al indicador de batería baja del sistema embebido.

V_{out} [V]	**Valor medido por el ADC**	**Nivel de batería** [%]
2.61	843	100
2.03	650	15
1.92	615	0

4.10.1. Implementación del *firmware* de control

Como fue mencionado anteriormente, la arquitectura del *firmware* de control pasa de un *Single Control Loop*, con un diseño basado en máquina de estados, a una basada en ISR. El *firmware* basado en el Arduino Pro Mini, el que incluye las rutinas de bajo consumo propias del procesador AtMega328P, se describe a continuación. El detalle puede encontrarse en la sección A.3 del Anexo.

1. Se implementa la función *setup()* para la inicialización de pines y variables. En esta función se modifican los registros WDTCSR para activar y utilizar el *Watchdog Timer*, ADCSRA para desactivar el ADC y SMCR para activar y utilizar los modos de bajo consumo. Además se implementan las ISR en los pines 2 y 3, para el sensor magnético y el pulsador táctil, respectivamente.
2. En la función *loop()* de control se implementan las rutinas para la activación y desactivación de la alarma en cada escenario. Se conservan los tiempos y métodos de activación y desactivación de la implementación anterior con el NodeMCU. Estas rutinas cuentan con un *flag* booleano que se inicia en *False*. Solo cuando la *flag* cambia a *True*, se ejecuta la rutina para activar la alarma correspondiente. Posteriormente, se activa el modo *sleep* de menor consumo energético (*Extended Standby*), utilizando el registro MCUCR para desactivar el *Brown Out Detection* y ejecutando el comando en assembler *asm volatile ("sleep")*.

Las ISR se implementan fuera del *loop* de control. Estas corresponden a 3, y cada una tiene la capacidad de desactivar el modo *sleep* y ejecutar su rutina. Se describen a continuación:

1. **ISR por detección de E1:** Para esta ISR, se configura el pin 2 como *INPUT PULLUP*. El sensor magnético se conecta también a tierra (GND). Al abrir la puerta, se abre el circuito y el pin pasa de 0 a 1 lógico, lo que activa la ISR (activación por señal *RISING*). La rutina de esta interrupción cambia el *flag* de E1, de tal manera que en el *loop* principal, se ejecute la rutina de activación y desactivación de la alarma para el escenario. Luego, vuelve a cambiar el *flag* para E1 a *False*.
2. **ISR por detección de E2:** Análogo al caso anterior, se conecta el pulsador táctil al pin, el que está configurado como *INPUT PULLUP*. Dado que el pulsador está conectado en su otro extremo a tierra (GND), al pulsarse éste por presión sobre la puerta, el pin pasa de 1 a 0 lógico, lo que activa la ISR (activación por señal *FALLING*). Esta interrupción cambia el *flag* de E2, tal que se ejecute la rutina de activación de la alarma en el *loop* principal. Luego, vuelve a cambiar el *flag* para E2 a *False*.
3. **ISR para medición de nivel de batería:** Finalmente, se implementa una ISR de

activación temporal, que se ejecuta cada 8 segundos mediante el *Watch dog Timer*, y mide el nivel de batería. Si está bajo el 15 %, hace parpadear el LED indicador y termina su ejecución.

Cada vez que se ejecuta una ISR, el *firmware* reanuda la ejecución en el *loop* principal, en el punto en donde se ejecutó la ISR correspondiente. Al final del *loop*, se ejecuta la instrucción para entrar en el modo *sleep Extended Standby*.

4.10.2. Consumo energético del sistema embebido

En la tabla 4.3 se resume el consumo energético del sistema embebido, basado en el microcontrolador Arduino Pro Mini, en cada uno de sus estados de funcionamiento.

Tabla 4.3: Consumo energético del sistema embebido para cada modo de funcionamiento. No se incluyen tiempos de ejecución, ya que son dependientes de la activación de las ISR, ante la detección de E1 y E2.

Estado del sistema	**Consumo energético [mA]**
Modo *sleep (Extended Standby)*	1.54
Modo normal	5.5
Alarma activada	10.1

En este caso, la evaluación de mejor caso (Estado *sleep* permanente) arroja una duración continua de más de 54 días de duración continua, dada la batería LiPo de 2000 mAh. La duración real dependerá de la frecuencia y duración de cada activación de la alarma, en cada escenario.

4.11. Implementación final en placa PCB

Dado lo anterior se procede a diseñar la placa PCB para la integración del sistema embebido. En la figura 4.17 se denota el primer diseño.

Figura 4.17: Diseño de primera placa PCB para la integración del sistema embebido.

Con este diseño, se realizó a continuación el prototipo en placa de cobre, tal como se describe en la sección 3.3.3. El resultado se observa en la figura 4.18.

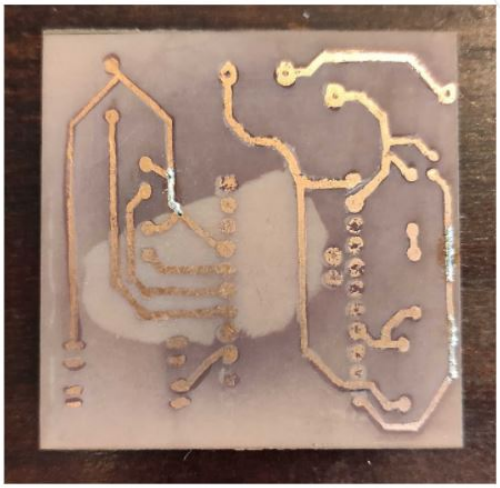

Figura 4.18: Implementación de PCB en placa laminada de cobre utilizando método con ácido férrico.

La implementación del sistema embebido realizada sobre la placa de la figura 4.18 resultó exitosa, por lo que se procede con la implementación final en placa de producción. Se realizan leves modificaciones a la distribución de los componentes en el diseño, además de aumentar el tamaño de ésta para contener sobre ella al extremo conectado del sensor magnético. Esto se aprecia en la figura 4.19.

Figura 4.19: Implementación final del sistema embebido basado en microcontrolador Arduino en placa PCB.

4.12. Implementación de carcasa de contención

El diseño y prueba de la carcasa de contención se realiza desde la primera implementación con el NodeMCU para evaluar la colocación del sistema embebido en la puerta base y el funcionamiento correcto de los sensores en el ambiente real de producción.

4.12.1. Primera implementación

En la figura 4.20 se observa la primera implementación, basada en el microcontrolador NodeMCU, alimentación con pilas AA y el sensor de presión.

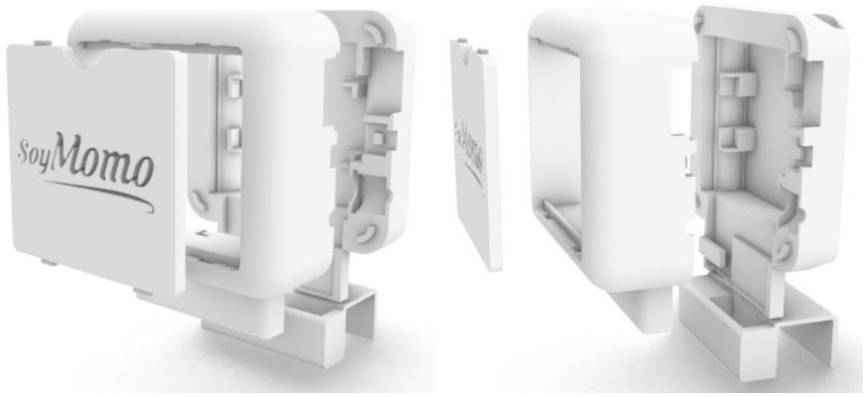

Figura 4.20: Renderización del diseño de la carcasa para la implementación basada en el NodeMCU, con pilas AA.

Este diseño contemplaba la contención de las pilas en la parte delantera y una pieza extra para colocar el otro extremo del sensor magnético en la parte inferior del marco de la puerta base.

4.12.2. Implementación final

Dado el rediseño del sistema en base al Arduino Pro Mini, la batería recargable y el reemplazo del sensor de presión por el pulsador táctil, se obtiene la implementación de la figura 4.21.

Figura 4.21: Renderización del diseño de la carcasa para la implementación basada en el Arduino Pro Mini, con batería recargable.

Esta carcasa está basada en la implementación del sistema embebido que se aprecia en la figura 4.19. En conjunto, se integran como se observa en la figura 4.22. La impresión se realiza utilizando filamento PLA, con impresión tipo FDM (*Fused Deposition Modeling*).

Figura 4.22: Integración del sistema embebido con la carcasa de contención final.

4.13. Integración del sistema embebido con la puerta base

En la figura 4.23 se observa el resultado final de integrar el sistema embebido con la puerta base.

Figura 4.23: Integración del sistema embebido, a través de su carcasa de contención, con la puerta base. A la izquierda, el detalle de la integración, realizada en la parte inferior de la puerta base, con el sistema embebido instalado en la parte móvil. A la derecha, la visión completa del producto final.

La integración con la puerta, tal como se observa en la figura 4.23, procura mantener los periféricos tales como el *Rocker Switch* de encendido y el botón de desactivación manual de la alarma en el lado del adulto e inaccesible al infante que está al otro lado.

El sistema embebido se instala en la parte móvil de la puerta, con ambos sensores en la parte inferior, con el fin de no exponer cables a lo largo de la puerta ni realizar una integración distribuida en distintos lugares de la puerta de seguridad base. Bajo el sistema, colocado sobre el marco, va la pieza auxiliar que completa la implementación del sensor magnético y, además, destina una superficie especial para la presión del pulsador táctil para el escenario 2. Esta pieza se observa mejor en la parte inferior de la figura 4.22.

Para sujetar el sistema embebido en la puerta, se añaden al exterior de la carcasa 2 anillos que permiten acoplarla en los barrotes contiguos. La mitad de estas anillas son parte de la carcasa principal, mientras que las otras dos mitades que las completan son piezas extra que se colocan por el frente para cerrarlas. Finalmente, se incluyen 3 espacios para cerrar el frente de la carcasa con tornillos.

4.14. Estimación de costo unitario del producto final

En la tabla 4.4 se resume el desglose de costos del producto final.

Tabla 4.4: Desglose del costo unitario para la implementación del producto final. Los costos de componentes se calculan en base a la adquisición de 100 unidades.

Item	**Costo [USD]**
Puerta base	16.8
Componentes electrónicos	3.7
Batería	1.3
PCB	0.2
Carcasa	7
HHs para montaje	1.01
Total	**30.01**

El costo de la puerta base se puede desglosar, a su vez, en 3 partes:

- La puerta como tal, que tiene un costo de 13.9 USD.
- La customización de la puerta, que implica cambios de colores y *branding* de la marca SoyMomo, tiene un costo de 1 USD.
- El uso de un *packaging* y manual de usuario a color, especialmente diseñado para este modelo, tiene un costo de 1.9 USD.

Por otro lado, el costo de los componentes electrónicos comprende el microcontrolador Arduino Pro Mini, los sensores, el botón de desactivación manual, el LED indicador de batería baja, el *Rocker Switch* de encendido, el *buzzer* para la alarma y el circuito cargador

TP4056. La batería se considera como un costo separado, ya que se obtiene desde un proveedor distinto. Finalmente, el costo de la PCB para la integración incluye las tres resistencias del divisor de voltaje para la medición del nivel de baterías, las que vienen ya montadas en la placa.

El costo de la carcasa incluye todos los componentes necesarios para su integración final en la puerta. El valor contempla la impresión de cada unidad utilizando ABS tipo resina, de color blanco, con impresión hecha con SLA (estereolitografía) de alta definición.

Para finalizar, se considera el costo de las HH necesarias para montar el producto final. La estimación de USD1.01 se realizó mediante una prueba de montaje. Ésta arrojó un tiempo de 15 minutos para realizarlo de principio a fin, lo que incluye, integrar todos los componentes electrónicos en la PCB, colocar el sistema embebido en la carcasa y montarlo en la puerta base. Se consideran 20 minutos por unidad para dar holgura. A partir de una jornada de 8 horas diarias, por 22 días hábiles al mes, se tiene una producción mensual de 528 puertas. A partir de un sueldo de CLP400.000 mensual destinado para este fin, y definido por la empresa mandante SoyMomo, se tiene el valor unitario de las HH para montaje, considerando una tasa de cambio de CLP750 ppor cada dólar.

Capítulo 5

Conclusiones

En el presente trabajo de título se presenta el proceso de desarrollo de un sistema embebido, cuyo fin es actuar como sistema de seguridad redundante para una puerta de seguridad para infantes, detectando dos posibles escenarios de peligro y dando aviso al adulto responsable a través de una alarma sonora. El desarrollo contempla el diseño del sistema embebido, dado por la elección e integración de los componentes electrónicos del sistema, y su implementación, a través de una placa PCB y una carcasa de contención que permite acoplarlo a la puerta de seguridad. El desarrollo en cuestión tiene una perspectiva comercial, buscando un diseño que no dificulte el uso normal de la puerta de seguridad, que sea robusto y que minimice los costos de producción.

Los resultados presentados en el capítulo anterior dan cuenta del cumplimiento absoluto del objetivo general presentado en el capítulo 1. Se logra un diseño e implementación que deriva en un producto de bajo costo, capaz de detectar de manera robusta ambos escenarios de peligro planteados. Todo lo anterior bajo un consumo energético mínimo, que permite la duración del sistema embebido por más de un mes en funcionamiento permanente.

En específico, el desarrollo permitió establecer relación con un proveedor no sólo de la puerta base, sino que de todos los componentes necesarios para la completa implementación del producto. A partir de esto, se puede asegurar una implementación continua en el tiempo, capaz de adaptarse a la demanda que pueda tener este producto.

Por otro lado, el diseño del sistema embebido permite generar grandes ahorros energéticos. Si bien la implementación realizada en base a ISR no permite determinar ciclos de trabajo fijos para el cálculo exacto del consumo medio, los consumos absolutos medidos en cada estado del sistema, permitirían alcanzar el objetivo de duración continua aún en casos extremos, que se consideran altamente improbables. Por ejemplo, dados los resultados de la tabla 4.3, la alarma sonora podría sonar por 4 días y medio seguidos y aún así el sistema es capaz de mantenerse encendido por 30 días, considerando que el resto del tiempo está en el modo *sleep Extended Standby*.

En relación al costo de desarrollo del producto completo, la incorporación de este criterio como uno de alta prioridad durante la etapa de diseño, permitió alcanzar un costo competi-

tivo. Sucesivas iteraciones en la electrónica permitieron reducir el costo unitario de este ítem de USD8.83 a USD3.70 (reducción del 58 %). El diseño también consideró el tiempo requerido para el montaje del producto final, con el fin de reducir el costo por HH. Llevando lo anterior a cifras, el costo total logrado, más un piso establecido del 50 % de margen bruto, significa tener que vender este producto a por lo menos, USD60, lo que en el mercado nacional, y a una tasa de cambio de CLP750 por cada dólar, significa un precio de venta de CLP45.000. Precio menor a cualquiera de los presentados en la tabla 2.1 y aún teniendo en cuenta el valor agregado del producto dado por el sistema embebido. Lo anterior es considerado por el mandante como un indicador de factibilidad económica, dejando en manos del área comercial de la empresa la determinación del precio final de venta.

Para finalizar, como trabajo futuro se proponen modificaciones en la etapa de implementación que apuntan a reducir aún más el costo de desarrollo del producto completo. En específico, se plantea invertir en la construcción de un molde industrial, que tiene un costo de inversión alto, pero que permitiría reducir el costo unitario de la carcasa de contención al rango [2, 3] USD. También, es posible derivar todo el desarrollo del sistema embebido al proveedor de placas PCB, con el fin de ahorrar HH en la construcción de cada puerta inteligente.

Por otro lado, se propone modificar el diseño para incorporar un nuevo nivel de alertas, dada por notificaciones a dispositivos móviles a través de conectar la puerta a una red WiFi. Esto requiere utilizar un procesador que soporte conexión a internet, como el ESP8266, el uso de servidores y el rediseño del sistema de alimentación, dado el alto consumo que necesita la conexión a Internet. También, se propone incorporar un nuevo escenario de peligro: que el infante se presione los dedos debido al sistema de cierre automático de la puerta. Se pretende que, al detectar este escenario, se active un actuador que bloquee el cierre total de la puerta y de aviso mediante la alarma sonora para que venga el adulto responsable a cerrarla de manera correcta.

Bibliografía

[1] Espressif Systems, *ESP8266EX Datasheet*, v6.5 ed., 2020.

[2] S. Cotofana, S. Wong, and S. Vassiliadis, "Embedded processors: Characteristics and trends," 06 2003.

[3] I. E. S. Technologies, "Fsr 402 series round force sensing resistor technical datasheet."

[4] M. Peden, K. Oyegbite, J. Ozanne-Smith, A. Hyder, C. Branche, and R. Akmf, "World report on child injury prevention," *Genève: WHO, UNICEF*, 01 2008.

[5] N. S. C. U. States, "Deaths in the home by age group," 2018. Online; Available at `https://injuryfacts.nsc.org/home-and-community/deaths-in-the-home/deaths-in-the-home-by-age-group/`.

[6] G. V. Research, "Baby safety products market size, share & trends analysis report by product type (car seats, monitors), distribution channel (offline, online), by region, and segment forecasts, 2019 - 2025," *Homecare & Decor - Market Analysis Report*, jul 2019.

[7] S. M. A. Shah, D. Sundmark, B. Lindström, and S. F. Andler, "Robustness testing of embedded software systems: An industrial interview study," *IEEE Access*, vol. 4, pp. 1859–1871, 2016.

[8] M. Barr, "Embedded systems glossary," *Neutrino Technical Library*, 2007. Online; Available at `https://barrgroup.com/embedded-systems/glossary`.

[9] "Ieee standard glossary of software engineering terminology," *ANSI/ IEEE Std 729-1983*, pp. 1–40, 1983.

[10] N. Zlatanov, "Computer memory, applications and management," 02 2016.

[11] J. Martin, "Programming real-time computer systems," *Englewood Cliffs*, 1965.

[12] J. Wang, "Formal methods in computer science," *CRC Press*, 2019.

[13] E. Moore, "Gedanken-experiments on sequential machines," *Annals of Mathematical Studies*, 1956.

[14] G. Mealy, "A method for synthetizing sequential circuits," *Bell System Technical Jour-*

nal, 1955.

[15] R. Bannatyne and G. Viot, "Introduction to microcontrollers. i," pp. 238 – 248, 11 1998.

[16] Atmel, *8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash*, 2015.

[17] E. Electronics, "5mm infrared led technical datasheet," 2005.

[18] E. Electronics, "5mm silicon pin photodiode technical datsheet," 2005.

[19] Y. Electronics, "Cjmcu-103 rotary angle sensor technical datasheet," 2008.

[20] SynaCorp, "Mc-38 door and window magnetic sensor technical datasheet."

[21] I. Studio, "Hc-sr04 ultrasonic ranging module technical datasheet," 2010.

Apéndice A

A.1. Evolución de la masa en infantes desde los 0 a los 5 años

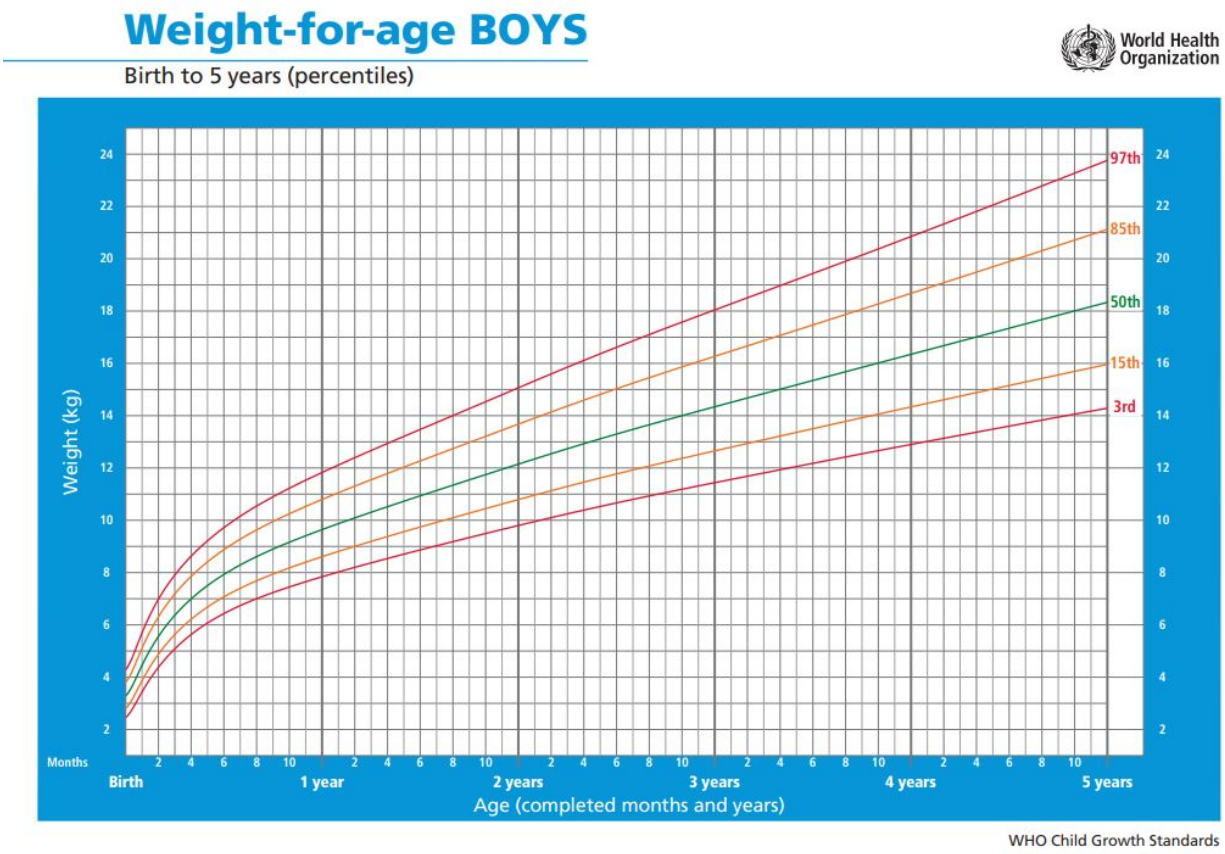

Figura A.1: Evolución de la masa corporal de infantes varones entre los 0 y 5 años. Se denotan los percentiles 97, 85, 50, 15 y 3.

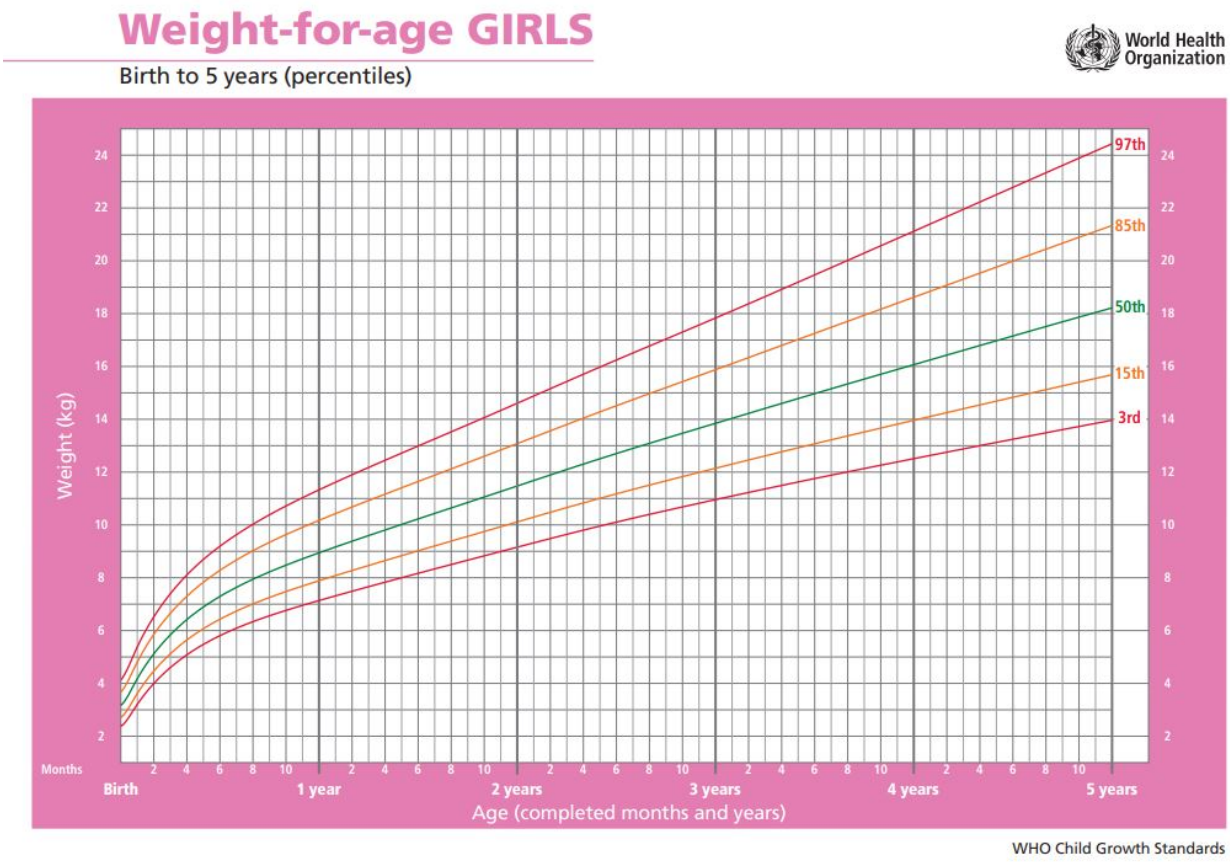

Figura A.2: Evolución de la masa corporal de infantes mujeres entre los 0 y 5 años. Se denotan los percentiles 97, 85, 50, 15 y 3.

A.2. Código de control del NodeMCU

```
//ESTADOS
//S0 (int 0) = Puerta cerrada
//S1 (int 1) = Puerta abierta
//S2 (int 2) = Estado de Alarma por apertura de puerta
//S3 (int3) = Estado de Alarma por presi n
//Los estados de alarma se diferencian porque el de puerta abierta
    se acaba y vuelve a S0 cuando se cierra la puerta.
//S3 se acaba cuando apreto el bot n y la puerta est  cerrada (
   pasa a S0) o se abre la puerta (pasa a S2).

//TRANSICIONES
//M = Magnetic sensor. M0 = CLOSED, M1 = OPEN
//F = Force sensor. F0 = FREE, F1 = OPEN
//T = Time Countdown. T0 = <DEADLINE, T1 = >DEADLINE
//B = Off Button. B0 = NEVER PRESSED, B1 = PRESSED
//
```

```
// DEFINE PIN NUMBERS
const int magneticSensorPin = D6;
const int pressurePin = A0;
const int buzzerPin = D1;
const int buttonPin = D7;

//ALARM VARIABLES
unsigned long openDoorDeadline = 5000; //5 segundos
int pressureThreshold = 900; //En un rango [0, 1023]

//STATE VARIABLE
int doorState;

//TIME VARIABLE
int openDoorTime;

void setup() {
  Serial.begin(9600);

  //Pin modes
  pinMode(magneticSensorPin, INPUT_PULLUP);
  pinMode(pressurePin, INPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(buzzerPin, OUTPUT);

  //Setting time variables
  openDoorTime = 0;

  //Setting initial state
  doorState = 0; //Cerrada
}

void loop() {
  switch (doorState) {
    case 0: //Closed door
      if (doorIsClosed()) {
        if (doorIsPressed()) {
          Serial.println("S0-S3: Baby is climbing the fence!");
          turnOn(buzzerPin);
          doorState = 3;
        }
        else {  // Door is closed and free of pressure
          Serial.println("S0-S0: Door is closed and free of
              pressure");
```

```
    }
  }
  else { //Door has been opened
    Serial.println("S0-S1: Door has been opened");
    openDoorTime = millis(); //Mark moment it was opened
    doorState = 1;
  }
  break;

case 1: //Door open, but alarm deadline not met yet.
  if (deadlineHasPassed()) {
    Serial.println("S1-S2: Door remained open!");
    turnOn(buzzerPin);
    doorState = 2;
  }
  else {  //Deadline has not been yet met.
    if (doorIsClosed()) {
      Serial.println("S1-S0: Door has been closed on time");
      doorState = 0;
    }
    else {  //Door is still open
      Serial.println("S1-S1: Door is still open");
    }
  }
  break;

case 2: //Alarm because door remained open longer than
   deadline
  if (doorIsClosed()) {
    Serial.println("S2-S0: Door has been closed");
    turnOff(buzzerPin);
    doorState = 0;
  }
  else { //door is not closed
    Serial.println("S2-S2: Door is open! Please close");
  }
  break;

case 3: //Alarm because of baby climbing the fence
  while(!buttonPressed()) {
    Serial.println("S3-S3: Please manually turn off the alarm"
       );
    yield();
  }
  if (doorIsClosed()) {
    if (doorIsPressed()) {
      Serial.println("S3-S3: Please manually turn off the
```

```
            alarm");
        }
        else { //Door not pressured
          Serial.println("S3-S0: Alarm deactivated. Door closed
             and free of pressure");
          turnOff(buzzerPin);
          doorState = 0;
        }
      }
      else { //Door is opened when button was pressed. Alarm
         should not deactivate
        Serial.println("S3-S2: Button pressed when door was open.
           Alarm keeps going");
        doorState = 2;
      }
      break;
  }
}

bool doorIsClosed() {
  int magneticState = digitalRead(magneticSensorPin);
  return (magneticState == LOW);
}

bool doorIsPressed() {
  int pressure = analogRead(pressurePin);
  return (pressure > pressureThreshold);
}

bool deadlineHasPassed() {
  unsigned long timeDelta = millis() - openDoorTime;
  return (timeDelta > openDoorDeadline);
}

bool buttonPressed() {
  int button = digitalRead(buttonPin);
  return (button == LOW);
}

void turnOn(int alarmPin) {
  digitalWrite(alarmPin, HIGH);;
}

void turnOff(int alarmPin) {
  digitalWrite(alarmPin, LOW);
}
```

A.3. Código de control del Arduino Pro Mini

```
#include "Arduino.h"
#include <avr/sleep.h>

#define magneticSensorPin 2 //Interrupt pin
#define pressureSensorPin 3 //Interrupt pin
#define buzzerPin 4
#define ledPin 11    // output pin for the battery warning LED
#define buttonPin 6
#define batteryWarningAnalogPin A0

//OPEN DOOR TIME VARIABLES
unsigned long openDoorDeadline = 5000; //5 seconds
unsigned long openDoorTime;

//PRESSURE BUTTON STATE VARIABLES
const int longPressTime = 2000; //2 seconds
unsigned long pressedTime = 0;
unsigned long releasedTime = 0;
int lastButtonState = LOW;
int currentButtonState;

//DOOR FLAGS
bool doorHasBeenOpened = false;
bool doorHasBeenClimbed = false;

void setup() {
  //Serial.begin(9600);

  //Set pin modes
  pinMode(magneticSensorPin, INPUT_PULLUP);
  pinMode(pressureSensorPin, INPUT_PULLUP);
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(batteryWarningAnalogPin, INPUT);

  //ES System alert with a beep of the buzzer
  digitalWrite(buzzerPin, HIGH);
  delay(500);
  digitalWrite(buzzerPin, LOW);

  //SETUP WATCHDOG TIMER (8 segundos)
  WDTCSR = (24);//change enable and WDE - also resets
  WDTCSR = (33);//prescalers only - get rid of the WDE and WDCE
```

```
    bit
  WDTCSR |= (1<<6);//enable interrupt mode

  //DISABLE ADC (En rutina de interrupci n de bater a baja hay
    que activarlo, leer pin an logo, ejecutar y dormirlo de nuevo
    )
  ADCSRA &= ~(1 << 7);

  //ENABLE SLEEP - this enables the sleep mode
  SMCR |= (1 << 2); //power down mode
  SMCR |= 1; //enable sleep

  //ATTACH INTERRUPTS
  attachInterrupt(digitalPinToInterrupt(magneticSensorPin),
    openDoorAlarmFlag, RISING);
  attachInterrupt(digitalPinToInterrupt(pressureSensorPin),
    climbedDoorAlarmFlag, FALLING);

  //Serial.println("Setup completed");
}

void loop() {

  openDoorAlarmRoutine();

  climbedDoorAlarmRoutine();

  //BOD DISABLE - this must be called right before the __asm__
    sleep instruction
  MCUCR |= (3 << 5); //set both BODS and BODSE at the same time
  MCUCR = (MCUCR & ~(1 << 5)) | (1 << 6); //then set the BODS bit
    and clear the BODSE bit at the same time
  __asm__ __volatile__("sleep");//in line assembler to go to
    sleep

}


//================== INTERRUPT ROUTINES ==================//

//LOW BATTERY TIME ISR
ISR(WDT_vect) {
  //low battery routine
  ADCSRA |= (1 << 7); //Activo el ADC del Arduino
  if (isBatteryLow()) {
    doBlink();
  }
```

```
  ADCSRA &= ~(1 << 7); //Una vez ejecutada la rutina, apago el ADC
}

//OPEN DOOR ISRs
void openDoorAlarmFlag() {
  doorHasBeenOpened = true;
}

void openDoorAlarmRoutine() {
    if (doorHasBeenOpened) {
      openDoorTime = millis();
      while (!deadlineHasPassed()) {
        yield();
      }
      turnOn(buzzerPin);
      while(!doorIsClosed()) {
        yield();
      }
      turnOff(buzzerPin);
      doorHasBeenOpened = false;
    }
}

//CLIMBED DOOR ISRs
void climbedDoorAlarmFlag() {
  doorHasBeenClimbed = true;
}

void climbedDoorAlarmRoutine() {
  if (doorHasBeenClimbed) {
    turnOn(buzzerPin);
    while(!buttonPressed()) {
      yield();
    }
    turnOff(buzzerPin);
  }
  doorHasBeenClimbed = false;
}


//====== AUXILIARY FUNCTIONS =========//
bool doorIsClosed() {
  int magneticState = digitalRead(magneticSensorPin);
  return (magneticState == LOW);
}

bool buttonPressed() {
```

```
  int button = digitalRead(buttonPin);
  return (button == LOW);
}

bool isBatteryLow() {
  int analogInput = analogRead(batteryWarningAnalogPin);
  int voltagePercentage = map(analogInput, 615, 845, 0, 100); //
     Umbrales analogos determinados empricamente en base a
     bater a de 2000 mAh del BM
  //Serial.print("Analogo: ");
  //Serial.println(analogInput);
  //Serial.print("Porcentaje: ");
  //Serial.println(voltagePercentage);

  return voltagePercentage < 15;
}

void turnOn(int alarmPin) {
  digitalWrite(alarmPin, HIGH);
}

void turnOff(int alarmPin) {
  digitalWrite(alarmPin, LOW);
}

bool deadlineHasPassed() {
  unsigned long timeDelta = millis() - openDoorTime;
  //Serial.print("openDoorTime: ");
  //Serial.print(openDoorTime);
  //Serial.print("          timeDelta: ");
  //Serial.println(timeDelta);

  return (timeDelta > openDoorDeadline);
}

void doBlink() {
  digitalWrite(ledPin, HIGH);
  delay(50);
  digitalWrite(ledPin, LOW);
  delay(300);
  digitalWrite(ledPin, HIGH);
  delay(50);
  digitalWrite(ledPin, LOW);
}
```