TRABAJO FINAL DE GRADO

TÍTULO: Rediseño y optimización de un sistema electrónico con tecnología NFC para facilitar su producción

AUTOR: ZORRILLA VALENCIA, CHRISTIAN

FECHA DE PRESENTACIÓN: OCTUBRE, 2020

APELLIDOS:	ZORRILLA VALENCIA	**NOMBRE:**	CHRISTIAN
TITULACIÓN:	INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA		
PLAN:	**2013**		
DIRECTOR:	DEL RIO FERNANDEZ, JOAQUIN		
DEPARTAMENT:	INGENIERÍA ELECTRÓNICA		

CALIFICACIÓN DEL TFG

TRIBUNAL

PRESIDENTE	**SECRETARIO**	**VOCAL**
MIRET TOMAS, JAIME	LOPEZ MARTINEZ, ANTONIO MIGUEL	PINEY DA SILVA, JOSE RAMON

FECHA DE LECTURA:

Este Proyecto tiene en cuenta aspectos medioambientales: ☒ Sí ☐ No

RESUMEN

A día de hoy, se nos hace muy extraño el hecho de tener que pagar insertando la tarjeta de crédito en el datáfono o el no poder pagar con el teléfono móvil. Este tipo de aplicación de pago, está basado en la comunicación NFC, «Near Field Communication», comunicación que es sencilla, segura y que, a día de hoy, se usa en muchos dispositivos, ya sean tarjetas, móviles, pulseras, etc.

Este proyecto trata sobre el rediseño y la optimización de un dispositivo de cronometraje deportivo que, previamente desarrollado por un estudiante de la UPC, y que usa esa la tecnología mencionada anteriormente.

Lo que se pretende es rediseñar el sistema, tanto en hardware como en firmware, para que este sea más pequeño y consuma menos energía. Para llevar a cabo estas medidas, se reestructurará el hardware, escogiendo solamente los componentes que sean empleados, quitando así, todo aquel que no aporte nada al sistema. En cuanto a firmware, se modificará el código incluyendo un modo reposos, de tal manera que el usuario pueda, dentro de unos parámetros, poner durante un tiempo el microcontrolador a dormir y así evitar consumos innecesarios. Hay que añadir, que también se implementará una aplicación Android, que permita leer el contenido de las tarjetas NFC.

Como resultado del sistema, se ha podido desarrollar una estructura más sólida que la anterior, cumpliendo los objetivos principales de reducción de tamaño y de disminuir el consumo. También se ha conseguido llevar a cabo la aplicación, mediante la cual podemos leer los datos de las tarjetas.

Debido al nuevo código, la memoria del microcontrolador deja mucho espacio libre que no se usa. Esto podría llevar a utilizar un microcontrolador más pequeño del actual, cosa que podría hacer que el consumo y el espacio que ocupase fuese menor. También se le podría adjuntar una batería de más capacidad, aumentando la autonomía del sistema.

Finalmente se implementará una aplicación que permita mostrar en el dispositivo móvil los datos que contienen las tarjetas NFC.

Palabras clave (máximo 10):

Arduino	NFC	PN532	DS3231
JAVA	APLICACIÓN	ATMEGA328P	RTC

ABSTRACT

Today, we find it very strange to have to pay by inserting the credit card into the dataphone or not being able to pay with the mobile phone. This type of payment application is based on NFC communication, «Near Field Communication», communication that is simple, secure and that, today, is used in many devices, whether they are cards, mobile phones, bracelets, etc. This project deals with the redesign and optimization of a sports timing device that, previously developed by a UPC student, and that uses the technology mentioned above. The aim is to redesign the system, both in hardware and firmware, so that it is smaller and consumes less energy. To carry out these measures, the hardware will be restructured, choosing only the components that are used, thus removing anyone who does not contribute anything to the system. Regarding firmware, the code will be modified including a sleep mode, in such a way that the user can, within certain parameters, put the microcontroller to sleep for a while and thus avoid unnecessary consumption. It should be added that an Android application will also be implemented, which allows reading the content of the NFC cards. As a result of the system, it has been possible to develop a more solid structure than the previous one, meeting the main objectives of reducing size and reducing consumption. The application has also been carried out, through which we can read the card data. Due to the new code, the microcontroller memory leaves a lot of free space that is not used. This could lead to the use of a microcontroller smaller than the current one, which could make the consumption and the space it occupies less. A higher capacity battery could also be attached, increasing the autonomy of the system. Finally, an application will be implemented that allows the data contained in the NFC cards to be displayed on the mobile device.

Keywords (10 maximum):

Arduino	NFC	PN532	DS3231
JAVA	Aplicación	ATMEGA328P	RTC

SUMARIO

SUMARIO DE ILUSTRACIONES

GLOSSARIO DE SIGNOS, SÍMBOLOS, ABREVIATURAS, ACRÒNIMOS I TÉRMINOS

HSU: High Speed UART.

I2C: Inter-Integrated Circuit.

NDEF: NFC Data Exchange Format.

NFC: Near Field Communication.

PCB: Printed Circuit Board.

RFID: Radio Frequency Identification.

SPI: Serial Peripherial Interface.

UHF: Ultra High Frequency.

INTRODUCCIÓN

Actualmente, el sector de la tecnología es un sector que se encuentra en una continua mejora. Este hecho conlleva la implementación de muchas tecnologías que simplifican la manera en la que vivimos, facilitando ciertas tareas cuotidianas o transformando aquellas que antes no lo eran, en tareas que se hacen con un simple clic.

Uno de los tipos de tecnología que nos facilita el día a día, es la comunicación inalámbrica. Esta comunicación está presente en cualquier parte del mundo, ya sea para encender la televisión (infrarrojos) o para enviar un documento a través del correo electrónico (WiFi). Dentro de este tipo de comunicación hay infinidad de ellas, pero hoy en día, la tecnología inalámbrica NFC es de las que más uso tiene. Además, permite, entre otras cosas, pagar con el teléfono o pagar sin introducir la tarjeta de crédito.

Este tipo de comunicación únicamente necesita dos dispositivos para su funcionamiento, un emisor y un receptor. El emisor se encarga de establecer la comunicación con el receptor y maneja el intercambio de datos. Por otro lado, el receptor simplemente responde al primero.

Mediante esta tecnología, se desarrolla la comunicación entre nuestro emisor (Arduino) y nuestro receptor (tarjeta NFC). En este proyecto, se busca mejorar el sistema de cronometraje deportivo en cuanto a hardware y firmware, reduciendo el sistema y optimizando recursos, lo cual implica cambios en el emisor. Para finalizar, se implementará una app que utilice dicha tecnología con el fin de obtener los datos del cronometraje.

1. OBJETIVOS

Como resultado de un previo Trabajo Final de Grado llevado a cabo por la Universidad Politécnica de Cataluña, se ha desarrollado un prototipo electrónico que puede utilizarse para el cronometraje de pruebas deportivas que utiliza tecnología NFC. Actualmente, el sistema se ha diseñado mediante placas de evaluación. El objetivo principal del proyecto será la integración y diseño de un prototipo pre-industrial, teniendo en cuenta las características necesarias para realizar la fabricación y la optimización del consumo: funcionamiento a baterías, cargador universal, caracterización del consumo en los diferentes modos de funcionamiento, documentación, etc.

Para aplicar todas estas medidas, se deben tener en cuenta las finalidades que busca este proyecto, que son:

- Mejorar el código y dividirlo para un microcontrolador más pequeño.
- Rediseño del sistema a nivel físico implementando el nuevo microcontrolador.
- Caracterización del consumo en los diferentes modos de funcionamiento.

Además, se propone como objetivo específico, el desarrollo de una aplicación de interfaz de usuario para dispositivos móviles que permita la comunicación y transferencia de datos del prototipo electrónico, principalmente para la validación del funcionamiento del prototipo y, en segundo lugar, para realizar la descarga de datos.

2. ESTADO DEL ARTE

2.1. RFID

Para poder explicar la tecnología NFC, cabe mencionar una primera tecnología que dio pie a esta, la tecnología RFID. Este tipo de tecnología inalámbrica se presenta inicialmente en 1973, cuando Charles Watson patentó la RFID pasiva. Esta tecnología unidireccional, se basa en un lector enviando una señal continua a una distancia concreta, esperando a que una etiqueta pase por esa señal y se transfiera la información de esta. El RFID trabaja con señales de baja frecuencia, de alta frecuencia o UHF, señales enviadas por una antena. Cabe añadir que, dependiendo de las propiedades de la etiqueta, se puede editar o añadir información en ella (Montero, 2017).

2.1.1. ETIQUETA RFID

Las etiquetas RFID son las encargadas de transmitir su información al lector y así, este puede usarla según su conveniencia. Las etiquetas tienen tres principales componentes, que son el material de fabricación, que será distinto según su aplicación, el circuito integrado, que dependerá de su memoria, robustez y seguridad, y la antena, con la que se debe tener en cuenta, el proceso de lectura para elegirla bien (Medina Delgado, 2017). No obstante, también existe otra clasificación de las etiquetas, donde pueden diferenciarse en etiquetas activas, pasivas o semi-activas (Montero, 2017).

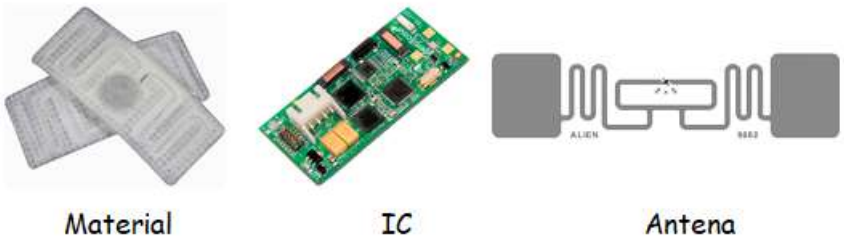


Ilustración 1: Principales componentes de una etiqueta RFID. Fuente: Propia.

Las etiquetas activas son aquellas que tienen una fuente de alimentación propia, transmitiendo siempre una señal constante de forma activa y con un mayor rango de lectura. En cambio, las etiquetas pasivas son aquellas que no disponen de una fuente de alimentación propia, donde la energía que activa el chip proviene del lector y esta limita su rango de lectura, ya que debe estar a cierta distancia para que, mediante la densidad de potencia de la onda del lector, llegue la tensión suficiente que active el chip. (Bin Abdul, 2017). Finalmente, se encuentran las etiquetas semi-activas, las cuales tienen una fuente de alimentación propia que solamente tiene la tensión suficiente para que el chip se encienda. Son de mayor rango de que las pasivas, pero de vida muy limitada por la pequeña fuente de alimentación (Bin Abdul, 2017).

2.1.2. LECTOR RFID

Los lectores son aquellos dispositivos que generan una señal de radiofrecuencia de forma permanente esperando a que una etiqueta entre en contacto y comience la transferencia de datos. Existen dos tipos de lectores, los lectores fijos que están anclados en un lugar permanentemente y los lectores móviles, que son aquellos que

están en movimiento constantemente (Medina Delgado, 2017).

2.2. INTRODUCCIÓN AL NFC

El NFC fue utilizado por primera vez en el año 2002, cuando Philips y Sony necesitaron un protocolo compatible con las tecnologías sin contacto existentes en aquella época (Montero, 2017). Este hecho, hizo que en 2003 fuera aprobado como estándar ISO 18092 y que muchas empresas como Nokia, Google o Visa, entre otras, quisieran formar parte del NFC Forum que ellos crearon (Medina Delgado, 2017). Todos ellos buscaban un sistema de conexión inalámbrico como el bluetooth, pero sin la necesidad de ninguna vinculación entre los dispositivos para poder comunicarse. Esto llevó a que la tecnología a la que imitar fuera la RFID pasiva, de corto alcance, debido a que como se comentó anteriormente, es una tecnología que es sin contacto y que no requiere de ninguna vinculación (Laaroussi, 2019).

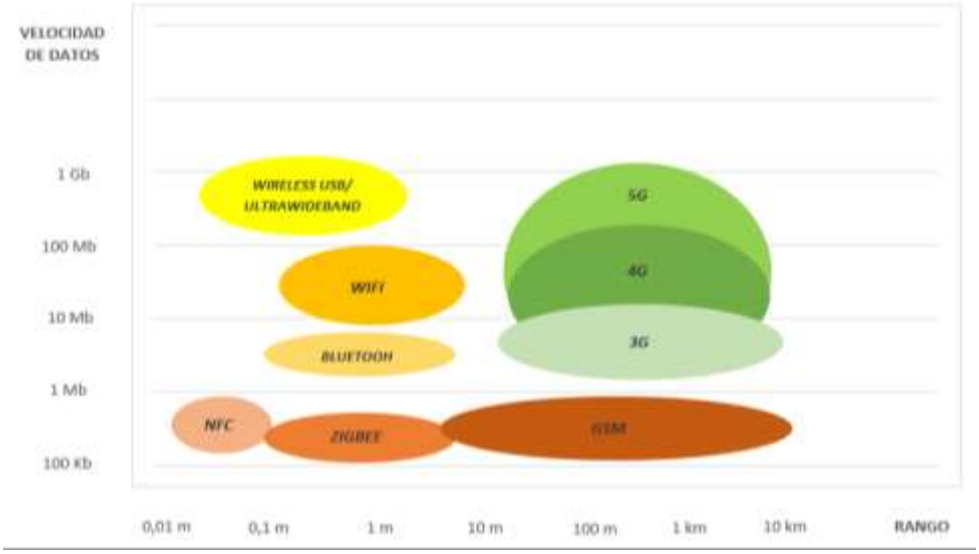


Ilustración 2: Las tecnologías inalámbricas en cuanto a velocidad de datos y rango. Fuente: Propia.

Todo sistema de comunicación NFC necesitará de un iniciador y un objetivo. Por un lado, el dispositivo que inicie y controle la transferencia de datos, será el iniciador. En el caso de la tecnología RFID, corresponde al lector (Medina Delgado, 2017). Por otro lado, el que responda a los requerimientos del iniciador, será el objetivo o etiqueta que en terminología del sistema RFID, corresponde a la etiqueta (Montero, 2017).

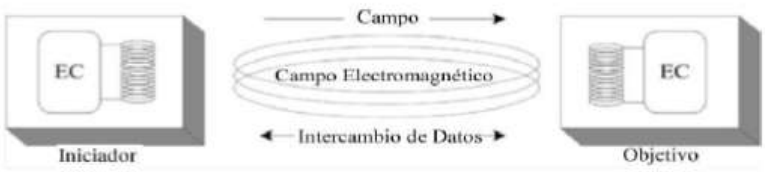


Ilustración 3: Comunicación Iniciador - Objetivo. Fuente: (Laaroussi, 2019)

Más adelante, en el 2010, el NFC se comercializó, mostrando que era una comunicación inalámbrica parecida al RFID pero que, en vez de ser unidireccional, esta era bidireccional (Medina Delgado, 2017). De esta manera, se podía comunicar dos dispositivos de forma inalámbrica con un rango bastante mayor que el de la RFID. Estas mejoras llevan consigo una disminución del precio del sistema (Montero, 2017). Todos estos avances han llevado a que hoy en día, con su precio económico y su reducido

tamaño, se haya implementado esta tecnología en todo tipo de dispositivos, ya que se puede usar en tarjetas, móviles, datáfonos, llaves, transporte público, etc. (Montero, 2017).

2.2.1. OBJETIVO NFC

Dentro del sistema NFC, los objetivos NFC son aquellos que se encargar de almacenar o sobrescribir la información que contienen. Esta acción vendrá dada según el uso que se le quiera dar por parte del iniciador. Actualmente, se dispone de cuatro grupos: Tipo 1, Tipo 2, Tipo 3 y Tipo 4 (Bin Abdul, 2017). En el punto 3.2.2. OBJETIVOS NFC Y CARACTERÍSTICAS PRINCIPALES se verá más detalladamente los tipos de etiquetas NFC con sus características más relevantes. Hay que añadir que los objetivos están basados en 2 normativas: la ISO 14443 y la JIS X 6391-4.

2.2.1.1. PROTOCOLO ISO / IEC 14443

Esta ISO es un estándar para protocolos de modulación y transmisión en comunicaciones RFID de 13.56MHz. Para esta ISO, existen dos tipos, tipo A y tipo B, cuyas diferencias radican en el método de modulación, los esquemas del código y el procedimiento del protocolo de iniciación. Cabe recalcar que comparten el protocolo de transmisión, hecho que hace que sean de la misma ISO (Montero, 2017).

Las de tipo A, son aquellas que mientras están en comunicación, reciben un pico de energía del lector y, cuando este les manda una señal, se activan y envían la información que proceda. Por el contrario, las de tipo B son aquellas que, mientras comunican, reciben energía del lector de forma ininterrumpida y envían la información pertinente (Montero, 2017).

2.2.1.2. PROTOCOLO JIS X 6391-4

Este protocolo, también conocido como FeliCa, está basado en un estándar industrial japones (JIS) que especifica los dispositivos de acoplamiento de proximidad y la trasmisión de datos inalámbricos de alta velocidad. Estas etiquetas vienen pre-configuradas únicamente para ser de lectura o escritura durante todo su ciclo de vida útil (Medina Delgado, 2017).

2.2.2. INICIADOR NFC

Como se ha observado, el sistema NFC debe tener dos dispositivos diferenciados. El iniciador es el instrumento que constantemente genera señales de radiofrecuencia hasta ser respondido por una etiqueta. El iniciador consta de un protocolo especial, el LLCP. Este protocolo es para la transmisión de datos entre dos iniciadores, este tipo de operación se llama P2P y se analiza en el apartado 3.5.2. MODO DE OPERACIÓN P2P (Medina Delgado, 2017).

A día de hoy, existen diversos módulos de lectura NFC en el mercado. Esto es debido a la expansión de esta tecnología y a la versatilidad que tiene. En el punto 3.7. SECTORES DE LA TECNOLOGÍA NFC veremos las diferentes aplicaciones en las que se usa el NFC y veremos cómo se usan diferentes lectores (Medina Delgado, 2017).

2.2.2.1. PROTOCOLO LLCP

El protocolo LLCP (Logical Link Control Protocol), es un protocolo OSI nivel 2 de intercambio de datos P2P entre dispositivos NFC. Este protocolo es esencial debido a que es el único que permite el envío de información de manera bidireccional (Montero, 2017). Este se basa en el estándar industrial IEEE 802.2, mejorando las funcionalidades básicas de este, pero sin impactar en las funciones del chip ni las aplicaciones del NFC (Montero, 2017).

3. NEAR FIELD COMMUNICATION

Como se ha visto en los puntos anteriores, para la comunicación son necesarios un iniciador y una etiqueta. En este punto se explicará el funcionamiento de manera más profunda y señalando, en el caso de las etiquetas, los diferentes tipos que hay. También se analizarán los diferentes modos de comunicación y los tipos de aplicaciones que actualmente usan este tipo de tecnología. De este modo, seremos capaces de observar el gran volumen de actividades que se pueden realizar con ella y la versatilidad de esta.

3.1. INICIADOR

Como se mencionó anteriormente en el punto 2.2.2. INICIADOR NFC, el iniciador está constantemente enviando señales de radiofrecuencia esperando a una etiqueta. Este está constituido principalmente por un IC, una antena y una fuente de alimentación, combinación que provoca que en el mercado exista una gran variedad de productos (Laaroussi, 2019). La antena se analizará en el punto 3.1.1. ANTENA INICIADOR. En cuanto al IC y la fuente es cierto que son dos partes bastante versátiles que normalmente van en función de la antena, dado que esta es el punto diferencial del producto en la mayoría de casos (Medina Delgado, 2017).

3.1.1. ANTENA INICIADOR

Como en toda emisión de radiofrecuencia, la generación del campo magnético es algo a tener en cuenta, dado que a medida que reciba corriente, se creará la inducción magnética (Montero, 2017). Esta inducción será de valor de $B = \mu * H$, siendo B la inducción magnética, μ la permeabilidad absoluta del material y H el campo magnético. Cabe recalcar que esta es proporcional al rendimiento, ya que es cierto que a mayor inductancia mayor rendimiento (Medina Delgado, 2017).

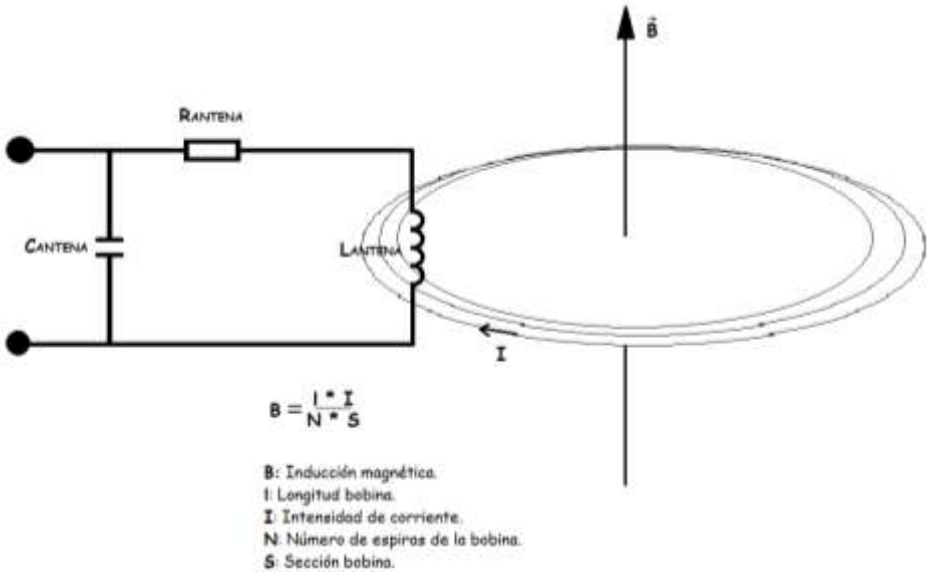


Ilustración 4: Inducción creada por una antena NFC. Fuente: Propia.

Como se puede ver en la Ilustración 4, la inducción magnética (B), dependen de la corriente (I) a más a más del número de espiras de la bobina (N), de la longitud (l) y de su sección (S). Este tipo de antenas suelen ser en bucle, como podemos observar en la

Ilustración 5 (Medina Delgado, 2017).

Ilustración 5: Antena NFC de un móvil. Fuente: (Medina Delgado, 2017)

3.2. OBJETIVO

La etiqueta NFC es la otra parte diferenciada del sistema. Esta se dispone para la transferencia de datos cuando pasa por el campo magnético emitido por el iniciador. Esto viene dado porque cuando pasa por el campo, la etiqueta lo modula para proporcionar una respuesta al iniciador a través de su antena (Laaroussi, 2019). La posición de la antena del objetivo es muy importante porque se necesita un bucle en el campo magnético para capturar el flujo magnético emitido por la antena del iniciador. El mejor rendimiento es cuando la bobina está perpendicular al campo magnético (Bin Abdul, 2017).

3.2.1. ANTENA OBJETIVO

En la ilustración 6, se puede observar que orientación de la antena de la tarjeta tiene mucho que ver con el campo magnético. Los círculos de color rojo, son los que no entran perpendicularmente al campo magnético, no capturan ningún campo magnético y entonces no hay conexión alguna e intercambio de datos (Bin Abdul, 2017). Los de color negro, no entran perpendicularmente, pero sí que capturan parte del campo magnético, lo que conlleva una conexión e intercambio de datos. No obstante, esta manera no es muy eficiente. Finalmente, están los círculos de color verde. Estos entran perpendicularmente al campo generado por la antena del iniciador, conectándose e intercambiando datos. Al contrario de los otros dos casos, este sí que es un método muy eficiente, siendo el mejor de todos los casos que se han analizado anteriormente (Bin Abdul, 2017).

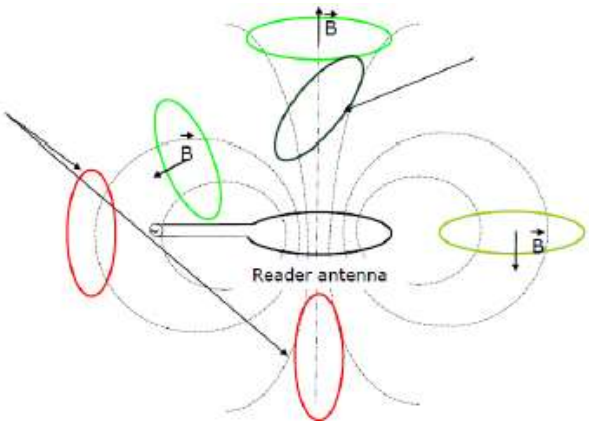


Ilustración 6: Antena de una etiqueta NFC y las diferentes maneras de interactuar con ella. Fuente: (Bin Abdul, 2017).

3.2.2. OBJETIVOS Y CARACTERÍSTICAS PRINCIPALES

Actualmente, se dispone de una gran variedad de tipos de objetivos NFC, pero verdaderamente se pueden separar en cuatro grupos según su tipo.

	Tipo 1	Tipo 2	Tipo 3	Tipo 4
Nombre del chip	Innovision Topaz	NXP MIFARE	Sony FeliCa	NXP DESFire, SmartMX-JCOP
ISO/IEC standard	14433 Type A	14433 Type A	JIS X 6319-4	14433 Type A, Type B
Velocidad de datos	106 kbps	106 kbps	212 kbps 424 kbps	106 kbps 212 kbps 424 kbps
Tamaño memoria	1Kb	2Kb	1Mb	64Kb
Lectura	Sí	Sí	Sí	Sí*
Escritura	Sí	Sí	Sí	Sí*
Soporte anticolisión	No	Sí	Sí	Sí
Precio	Bajo	Bajo	Alto	Medio / Alto
Proveedor	Innovision, Researech and technology	Philips / NXP	Sony	Several

*Solamente se pueden configurar de una manera para su proceso de vida, en cambio las otras pueden ser ambas cosas durante su vida útil.

Ilustración 7: Tipos de objetivos NFC. Fuente: Propia.

Como se puede observar en la Ilustración 7, las etiquetas Topaz son las que menos memoria tienen y, junto a las MIFARE, las de menos velocidad de datos. También se puede ver que se pueden configurar tanto en modo lectura como escritura. Estas características sumadas a no tener soporte anticolisión, hacen que tengan un precio bajo en comparación a las demás (Medina Delgado, 2017).

Las etiquetas de memoria MIFARE son muy similares a las Topaz, siguiendo la misma ISO y teniendo la misma velocidad de datos, pero a diferencia de ellas, estas tienen sistema anticolisión y un 1Kb más de memoria, siendo esta de 2Kb. Cabe recalcar que, a pesar de tener estas dos mejoras en sus características respecto de las Topaz, su precio sigue siendo bajo, bastante similar al de ellas (Medina Delgado, 2017).

Las etiquetas tipo 3, FeliCa, son las más caras del mercado. Este hecho es debido a que son las que mejores prestaciones, siendo así las que más memoria y velocidad de datos tienen. Como los tipos anteriores de etiquetas, estas se pueden configurar tanto para lectura como para escritura (Medina Delgado, 2017).

Finalmente, las etiquetas DESFire son las que permiten más flexibilidad a la hora de la transmisión de datos, debido a sus 3 rangos de velocidad. Dispone de una memoria de 64Kb, menos grande que la de FeliCa, sin embargo, puede servir para una infinidad de aplicaciones (Medina Delgado, 2017). Su precio, medio-alto, varía según la velocidad de datos que requiera la aplicación y según el modo de su configuración, puesto que estas tarjetas se compran ya pre-configuradas en modo escritura o lectura y no pueden cambiarse al otro modo durante su vida útil (Medina Delgado, 2017).

3.3. SISTEMA COMPLETO

Como se ha mencionado anteriormente, todo sistema de comunicación NFC necesita de dos integrantes, el iniciador y el objetivo. Para crear este vínculo, se tiene que dar una resonancia de 13.56 MHz (Medina Delgado, 2017). Cuando el iniciador esté en funcionamiento, este creará un campo magnético a través de su antena y esperará a que sea capturado por la antena del objetivo. La tensión AC inducida de la parte electrónica del iniciador se rectificará en una fuente de alimentación CC que tiene integrado el IC del objetivo (Medina Delgado, 2017). Cabe recalcar que cuanto mejor sea el diseño de la antena, mayor rendimiento tendrá la transferencia de datos. El objetivo, por medio de su IC, modulará la corriente que fluirá hacia su antena, modificando el campo magnético total. En la Ilustración 8, se puede observar cómo se constituye el sistema entero y como fluye el flujo del campo magnético, siendo perpendicular a la antena del objetivo (Bin Abdul, 2017).

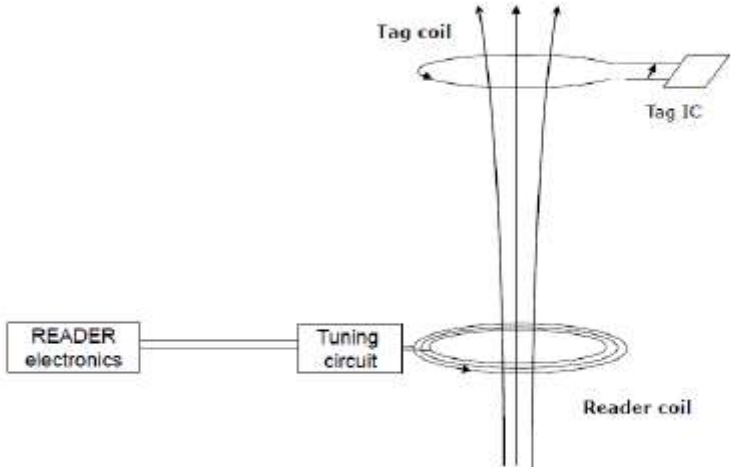


Ilustración 8: Sistema completo de una transferencia de datos vía NFC. Fuente: (Bin Abdul, 2017)

3.4. MODOS DE COMUNICACIÓN ACTIVO - PASIVO

La tecnología NFC dispone de dos únicos modos de comunicación. Estos dos modos de comunicación son similares a los modos de la tecnología RFID. Los modos de comunicación son: activo y pasivo. Este tipo de comunicación varía según si se dispone de fuente de alimentación propia o no (Bin Abdul, 2017). En este modo de comunicación, ambas partes, iniciador y objetivo, disponen de una fuente de alimentación propia. De esta manera, los dos dispositivos pueden enviarse datos generando sus propias señales. En este otro modo de comunicación, solamente el iniciador tiene una fuente de alimentación propia. En este caso, la etiqueta es alimentada por el campo de energía del iniciador (Bin Abdul, 2017).

3.5. MODOS DE OPERACIÓN

A diferencia del sistema de comunicación inalámbrico RFID donde los elementos siempre han sido un lector y una etiqueta, en la comunicación NFC, se pueden encontrar casos en los que la comunicación es entre dos iniciadores o bien, que un iniciador se convierta en objetivo (Bin Abdul, 2017). Estos modos son: Lectura/Escritura, P2P y Emulación de tarjeta (Montero, 2017).

3.5.1. MODO DE OPERACIÓN LECTURA / ESCRITURA

Mediante este modo de operación, el iniciador puede leer los datos que contiene el objetivo y también, escribir en él. En este caso, las etiquetas son de comunicación pasiva, con lo cual no disponen de una fuente de alimentación propia, como se puede observar en la Ilustración 9. Asimismo, en este modo el iniciador es capaz de leer los 4 tipos de etiquetas NFC determinadas por el NFC Forum (Montero, 2017). Esta operación sigue una serie de pasos para poder llevar a cabo la transferencia de datos. En primer lugar, el iniciador solicita la información que desea al objetivo o etiqueta. Después, el objetivo transfiere la información al iniciador. Una vez transferida, el iniciador procesa la información y, dependiendo de la función del sistema, se hace un uso adicional de la información, como podría ser acceder a internet. A continuación, el usuario solicita escribir información en el objetivo al tocarla. Finalmente, la etiqueta transfiere un mensaje al iniciador informando que se ha logrado la operación (Montero, 2017).

Ilustración 9: Modo de operación Lectura / Escritura. Fuente: Propia.

3.5.2. MODO DE OPERACIÓN P2P

En este modo, a diferencia de la tecnología RFID, como puede verse en la Ilustración 10, se establece una conexión bidireccional entre los dos dispositivos que están en comunicación. De esta manera, ambos iniciadores pueden intercambiarse cualquier tipo de datos, ya sean fotos, contactos, aplicaciones, etc. Esta comunicación es muy común, por ejemplo, cuando se transfiere los datos vía NFC del teléfono antiguo al nuevo. Este tipo de operación se hace basándose en el protocolo LLCP, protocolo estandarizado por NFC Forum y que es esencial para cualquier aplicación de comunicación NFC bidireccional (Montero, 2017). El funcionamiento de esta operación es más sencillo que el anterior, siendo dos pasos. Para empezar, se hace una petición y transmisión de datos, donde los dos iniciadores transfieren su información. Para acabar, si la información tiene algún propósito con el uso de internet, se hace ese uso adicional. En caso de que no se requiera internet, este modo de operación únicamente constaría de un solo paso. (Montero, 2017).

Ilustración 10: Modo de operación P2P. Fuente: Propia.

3.5.3. MODO DE OPERACIÓN EMULACIÓN DE TARJETA

En esta última manera de operar, el iniciador se comporta como un objetivo, emulándolo. De esta manera, otro inicializador puede encontrarlo y leerle los datos o reescribirle algún dato. Este método de operación es el más común en cuanto al pago vía NFC debido a su alta seguridad. Para este tipo de operación se usa un plan también corto, estilo al del modo P2P. Primero de todo el iniciador hace una petición al otro iniciador que funciona como etiqueta y, si esta es aceptada, instantáneamente se produce la transmisión de datos. A continuación, el iniciador realiza sus operaciones pertinentes en segundo plano. Por ejemplo, como en la Ilustración 11, en caso de que fuera el pago de una cena, lo que estaría haciendo sería validar la tarjeta de crédito. Finalmente, el iniciador reescribe datos y, en algunos casos, devuelve información (Montero, 2017).

Ilustración 11: Modo de operación Emulación de tarjeta. Fuente: Propia.

3.6. SEGURIDAD

En temas de tecnología con datos de carácter sensible, la seguridad siempre debe estar garantizada pero lamentablemente nunca lo está, a pesar de que se cree esa falsa ilusión. Es por eso que se habla más de fiabilidad que de seguridad. La fiabilidad de un sistema es la realización de su función sin reportar ningún error ni que ocurra algo que no estaba previsto (Medina Delgado, 2017). Generalmente, un sistema que garantice esto, debe cumplir los parámetros de confidencialidad, donde nadie puede acceder a la información si no es autorizado, integridad, en la que la información solamente sea modificada mediante autorización, y disponibilidad, que solo esté disponible dicha información si se tiene la autorización pertinente (Montero, 2017).

Como toda tecnología, la NFC da lugar a problemas con los parámetros mencionados anteriormente, confidencialidad, integridad y disponibilidad (Montero, 2017). Para poder garantizar estos parámetros de la mejor manera, habrá que hacer un diseño muy exhaustivo que disponga de varios mecanismos de seguridad, debido a que, por ejemplo, uno de sus usos es el pago con tarjeta. Para ello, los bienes que deben ser protegidos a través de una gran seguridad, son los que contienen información privada del usuario, la funcionalidad NFC y operatividad del dispositivo, la funcionalidad del controlador host y aplicaciones, la información intercambiada entre dispositivos NFC y la información almacenada en una etiqueta NFC (Medina Delgado, 2017).

Para ello existe un único protocolo de seguridad, el NFCIP-1. Este protocolo tiene definida y estandarizada su seguridad, que se centra en el intercambio en el modo P2P. También se promueve el uso del estándar NFC-SEC (NFCIP-1 Security Services and Protocol) que basa su seguridad en estándares criptográficos. Asimismo, cualquier P2P no requerirá de mecanismos específicos de encriptación ya que estos serán proporcionados por los componentes del NFC (Laaroussi, 2019).

3.6.1. SEGURIDAD ETIQUETAS NFC

Las etiquetas NFC son normalmente los activos que llevan los datos de carácter sensible, siendo así la principal fuente de acción para los atacantes. Está probado que su vulnerabilidad viene dada mediante ataques como el gusano NFC o worm-URL, ocultos en la etiqueta o ataques spoofing, suplantación de identidad (Medina Delgado, 2017). Para mitigar estos ataques y que no vulneren los datos o se realicen copias, las etiquetas NFC se elaboran con la firma de técnicas de cifrado adecuadas según el uso que se le vaya a hacer. Una de las técnicas que se usan para evitar estos ataques es la comunicación mediante los mensajes NDEF (NFC Data Exchange Format). Este estándar de intercambio de datos NFC puede verse en las tarjetas MIFARE Classic o que se usa en el modo de operación P2P. El NFC Forum define este registro de firma como RTD (Record Type Definition) para permitir firmas digitales al mensaje NDEF, protegiendo de esta manera la confidencialidad, la integridad y la disponibilidad (Medina Delgado, 2017).

3.7. SECTORES DE LA TECNOLOGIA NFC

Actualmente, la comunicación NFC está bastante consolidada, aunque la gran mayoría de gente la desconozca (Montero, 2017). Esta tecnología está presente en gran parte de acciones cuotidianas de nuestro día a día, como pagar la compra con el teléfono móvil o con la tarjeta sin que esta sea introducida en el datafono. Pero también, está presente en otros sectores como el sanitario o publicitario (Medina Delgado, 2017).

El sector de pagos o transacciones es el que mueve más intercambio de datos vía NFC debido a que cualquier usuario que disponga de tarjeta de crédito puede hacer uso de esta tecnología (Bin Abdul, 2017). Para realizar una transacción o pago, comunicación de muy corta distancia, se usa el modo de operación Lectura/Escritura si es mediante tarjeta de crédito o el modo de operación Emulación de tarjeta si se realiza con Android o iOS (Medina Delgado, 2017).

Cuando hablamos del sector sanitario, la tecnología NFC se usa primordialmente para la supervisión, el control y el seguimiento de la salud de forma remota a través de la captación de los datos que almacenan las máquinas hospitalarias (Medina Delgado, 2017). El modo de operación que presentan estas máquinas para la obtención de datos puede ser cualquiera de los 3 que existen. En cambio, cuando hablamos del sector publicitario, el modo de operación por excelencia es el de Lectura/Escritura, aunque se han visto casos con el modo P2P. Lo que suele realizar este sector, mediante la tecnología NFC, es introducir una etiqueta NFC en el anuncio y de esta manera, cuando el consumidor quiere actuar con el anuncio, este solamente debe coger su teléfono y leer la etiqueta, la cual le teletransportará a la página donde esté anunciado dicho producto (Laaroussi, 2019).

4. aplicación previa desarrollada

Como se ha comentado anteriormente, este proyecto tiene una base previa debido a que es la continuación del proyecto de Sadik Laaroussi, Sistema para cronometraje deportivo basado en tecnología NFC. En este apartado se analizará el último hardware y software que llegó a implementar y que se seleccionó en este proyecto como base para desarrollar el rediseño y la optimización.

4.1. DESCRIPCIÓN

La idea principal del anterior estudio, fue crear un sistema robusto y seguro para poder cronometrar eventos deportivos. Este cronometraje se hizo mediante la tecnología NFC usando el modo de operación Lectura/Escritura. Para llevar a cabo esto, se dispuso de 4 tipos de bases, las cuales se encargaron de escribir la hora en la que se pasa por la base, leer los datos de la etiqueta, borrar los datos de la etiqueta y verificar que se borraron dichos datos de la etiqueta (Laaroussi, 2019).

En cuanto al hardware, las bases se encuentran formadas por la unión de Arduino, una RTC, un lector de tarjetas NFC y una fuente de alimentación propia. En cambio, para el software se utilizó la programación de Arduino y se estructuró un solo programa para poder usar los 4 métodos. Cabe recalcar que la idea de este proyecto, es una de las que se encuentra en el mercado de la mano de diversas empresas, como por ejemplo SPORTident (Laaroussi, 2019).

En los siguientes apartados de este cuarto punto, se analizará de forma más concreta los diferentes aspectos del sistema implementado por Laaroussi, S.

4.2. SOFTWARE PREVIO

4.2.1. BASE EN MODO ESCRITURA

4.2.1.1. EXPLICACIÓN Y FLUJOGRAMA

Haciendo mención a lo anterior, la base de escritura conlleva un identificador único que hace que se puedan diferenciar las 45 bases. Este identificador es el ID Base, que puede ir desde el número 0 al 44, ocupando 2 bytes de la memoria EEPROM de las etiquetas NFC. Mediante una RTC obtendremos la hora y la fecha para que cuando se acerque una tarjeta NFC al iniciador, este pueda grabar los datos en él, ocupando 14 bytes de la memoria EEPROM de la etiqueta. De esta manera, queda guardado un número como el de la Ilustración 12. En la Ilustración 13 se puede observar como se muestra por el monitor serie un dato escrito en la memoria de la etiqueta (Laaroussi, 2019).

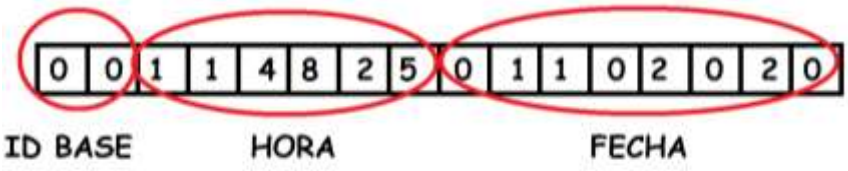


Ilustración 12: Dato guardado tarjeta NFC. Fuente: Propia.

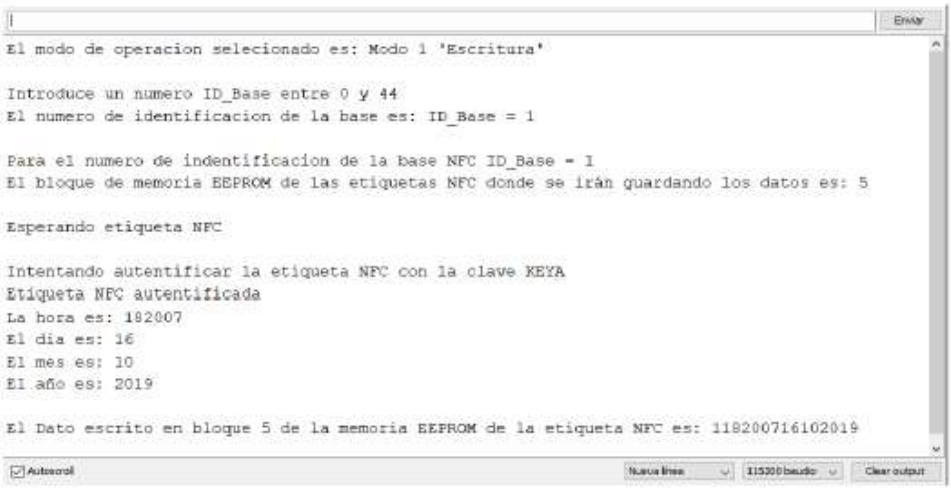


Ilustración 13: Como se escribe un dato en la memoria EEPROM de la etiqueta visto desde el monitor serie. Fuente: (Laaroussi, 2019).

Para poder llevar a cabo la función de escribir en la tarjeta NFC, en primer lugar, se debe inicializar la RTC. Después, se asigna el número de base que será (ID_Base). Este número ID_Base asignará el bloque de la memoria en el que se guardará el dato. Esto es debido a que la memoria EEPROM de la etiqueta tiene ciertos bloques en los que es imposible escribir, por esta razón, mediante el programa obviamos estos bloques, para así, poder utilizar los huecos de la memoria que sí están disponibles. Una vez llegados a este punto, se pasa al bucle de escritura. En este bucle, primero se comprueba que haya una etiqueta NFC. Seguido, si se ha detectado una etiqueta, se escribe el dato. A continuación, se lee el dato escrito para verificar que se ha escrito correctamente, suena un pitido y se escribe el ID_Base más la fecha de la RTC. Toda esta explicación se pude observar en el flujograma de la Ilustración 14 (Laaroussi, 2019).

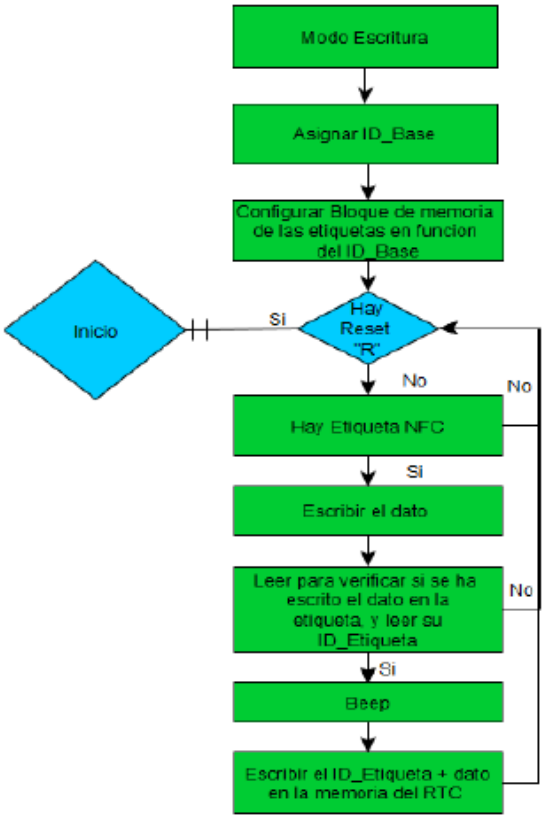


Ilustración 14: Flujograma Modo Escritura. Fuente: (Laaroussi, 2019).

4.2.2. BASE MODO LECTURA

4.2.2.1. EXPLICACIÓN Y FLUJOGRAMA

Este modo es el encargado de que al final de la carrera, se pueden ver la hora a la que se ha pasado por cada una de las bases de escrituras distribuidas a lo largo del evento. El flujograma de este modo, Ilustración 15, es más simple debido a que no es necesario que se introduzca ningún parámetro desde el monitor serie ni haya que inicializar ninguna RTC. Directamente se entra en el bucle de lectura debido a lo mencionado anteriormente. El programa espera pacientemente a que se acerque una tarjeta NFC. Una vez detectada, comienza a leer la memoria y a mostrarla por el monitor serie. Finalmente, suena un pitido y vuelve al inicio del bucle a esperar que pase una nueva tarjeta NFC (Laaroussi, 2019).

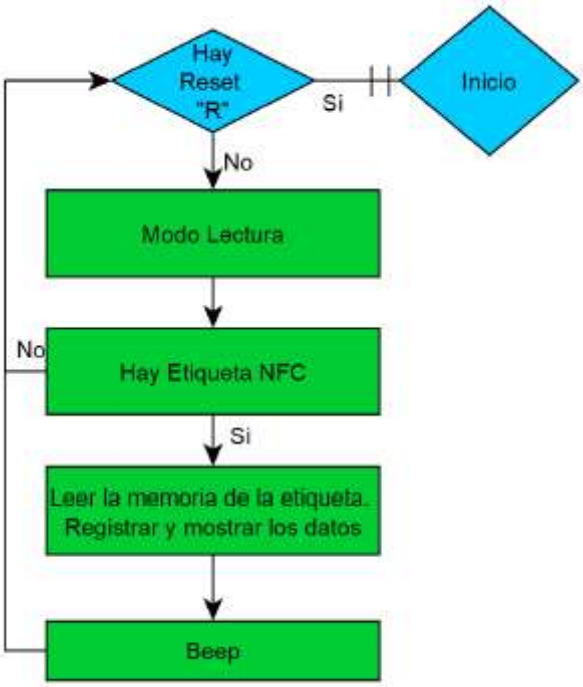


Ilustración 15: Flujograma Modo Lectura. Fuente: (Laaroussi, 2019).

4.2.3. BASE MODO BORRAR

4.2.3.1. EXPLICACIÓN Y FLUJOGRAMA

La idea de este modo es que esté situado al inicio del evento, así se tendrá una tarjeta limpia sin datos en ella que lleven a la confusión. Una confusión podría ser dada a que en la carrera anterior hubiera 45 bases y en la actual solamente 15 y si no se llegasen a borrar todos los datos de esta, el corredor al ir a la base de lectura, se encontraría con sus marcas de la base 0 a las 14 pero de la 15 a las 44 encontraría las marcas de otro corredor, cosa que puede prestar a equívocos. La Ilustración 16, como en el modo de lectura, muestra cómo se entra directamente en el bucle. Cuando se detecta una etiqueta, el programa pasa a borrar la memoria de esta y, cuando haya terminado, sonará un pitido y volverá a esperar a que se acerque una tarjeta NFC (Laaroussi, 2019).

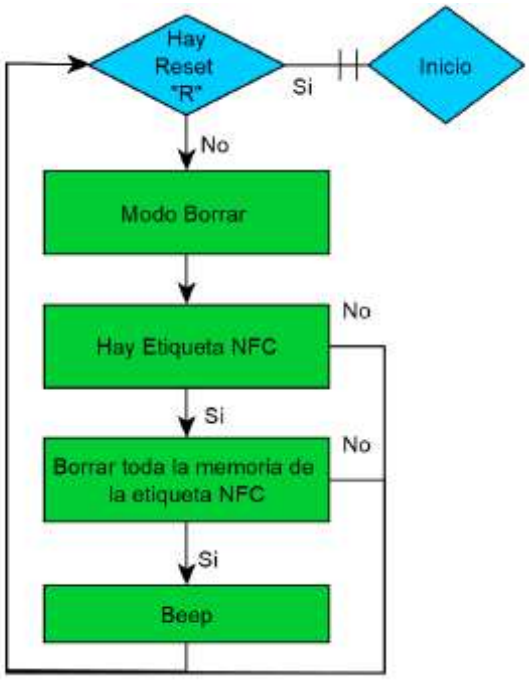


Ilustración 16: Flujograma Modo Borrar. Fuente: (Laaroussi, 2019).

4.2.4. MODO VERIFICAR TARJETA VACÍA

4.2.4.1. EXPLICACIÓN Y FLUJOGRAMA

Este último modo, al igual que los 2 anteriores, no necesita recibir ningún dato del monitor serie. Como se puede ver en la Ilustración 17, esto hace que también se entre directamente en bucle, esperando a detectar una etiqueta. Una vez detectada, este lee la memoria de dicha tarjeta y verifica que está vacía. Una vez acabada la comparación hace un tipo de pitido, dependiendo de si está vacía o no, y vuelve a esperar a una tarjeta (Laaroussi, 2019).

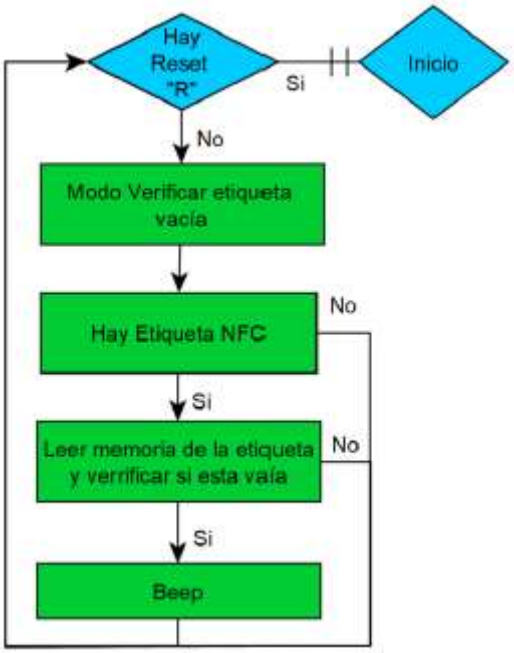


Ilustración 17: Flujograma Modo Verificar tarjeta vacía. Fuente: (Laaroussi, 2019).

4.2.5. SOFTWARE COMPLETO

Una vez visto los cuatro programas por separado, en la Ilustración 18 se muestra el flujograma de todo el conjunto, siendo este el programa definitivo para el control del sistema.

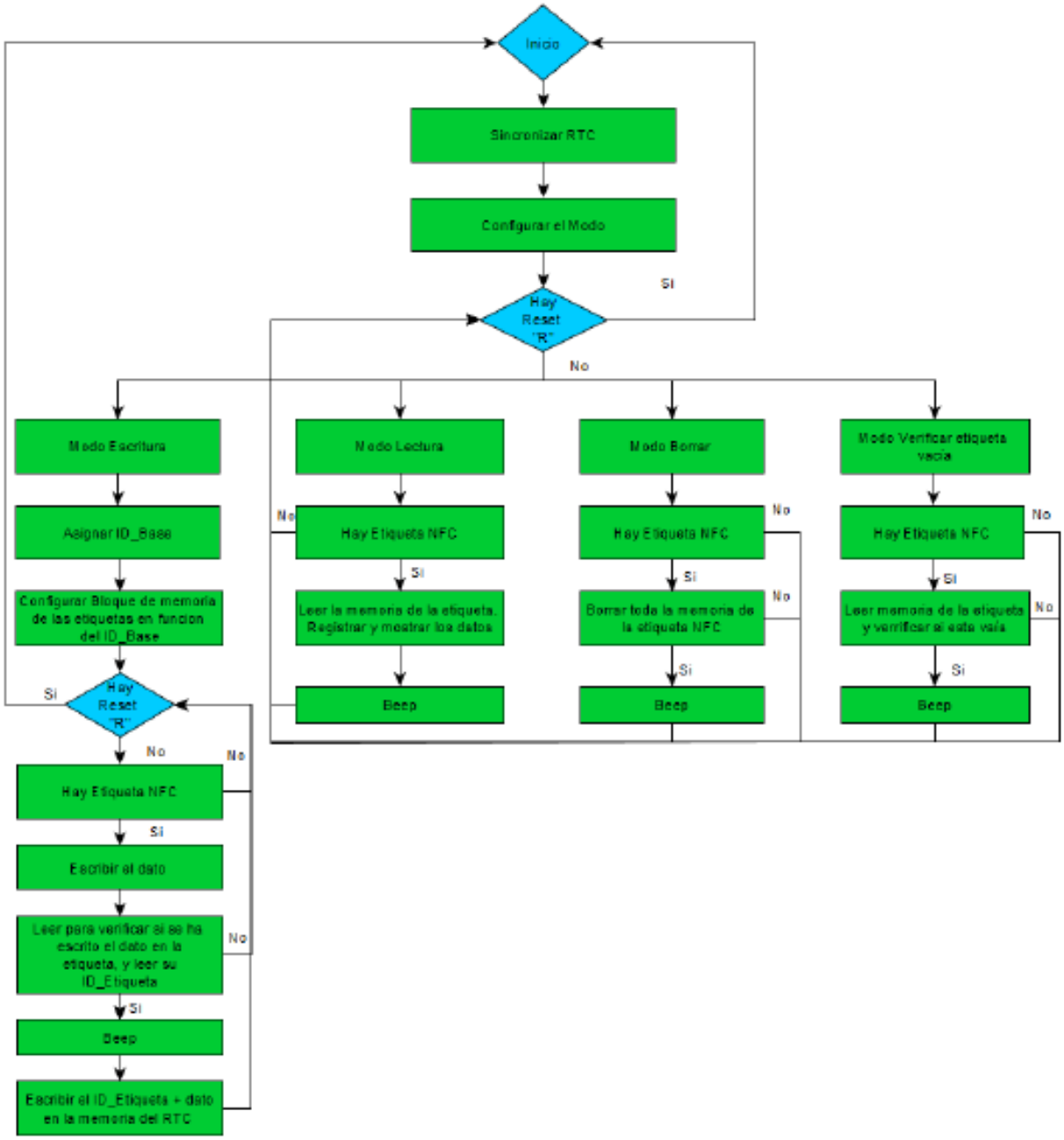


Ilustración 18: Flujograma completo con los 4 modos de trabajo. Fuente: (Laaroussi, 2019).

4.3. HARDWARE PREVIO

4.3.1. LISTA DE MATERIALES Y FUNCIÓN

Si se observa la Ilustración 19, se pueden detectar los elementos utilizados en el proyecto. En primer lugar, el microcontrolador, un Arduino MEGA 2560 REV3, el cual se encargará de ejecutar el programa. En segundo lugar, se puede ver el módulo PN532 NFC RFID V3, módulo encargado de ser el comunicador entre el microcontrolador y la tarjeta. En tercer lugar, se distingue el DS3231 RTC, este elemento combinado con el que se encuentra en el cuarto lugar, la pila CR2032, se encargan de dar la hora exacta a la que sucede cualquier evento, en el caso de este proyecto, la hora a la que se pasa por una base en modo escritura. En quito lugar, se encuentra un zumbador activo de 5V, que se encarga de dar el pitido para avisarnos de que algún modo ha finalizado. En sexto lugar, se destaca la batería encargada de proporcionar la alimentación necesaria al sistema. En séptimo lugar, se posiciona la PCB, fabricada por Laaroussi, S. que sirve como modo de conexión entre los elementos mencionados anteriormente. En penúltimo, se encuentra la tarjeta NFC, en la cual se almacenarán los datos del evento. Finalmente, se observa la caja en la que se almacenará el sistema (Laaroussi, 2019).

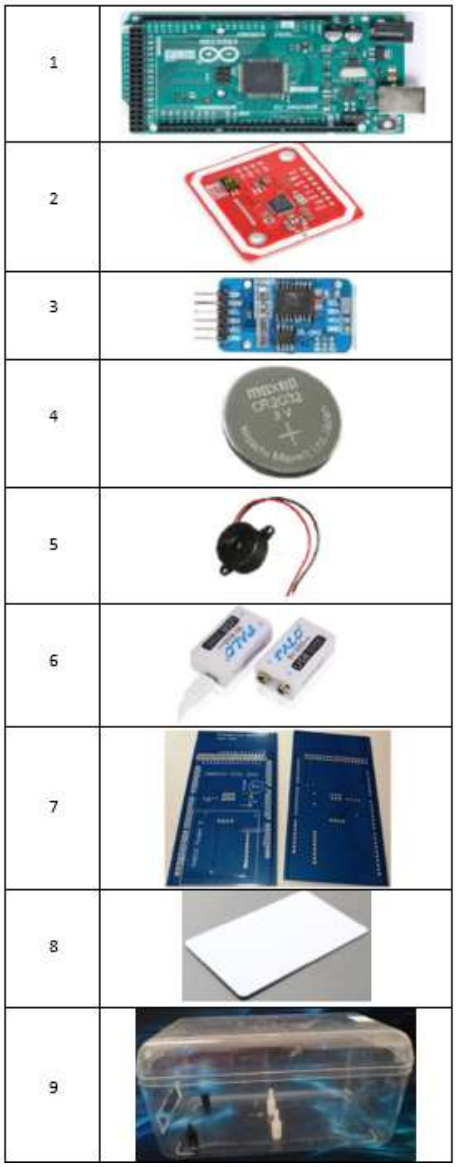


Ilustración 19: Componentes proyecto Larroussi, S. Fuente: Propia.

4.3.2. CARACTERÍSTICAS PRINCIPALES DEL HARDWARE

4.3.2.1. ARDUINO MEGA 2560 REV3

Mediante este microprocesador, se controla que el sistema sea fiable. Este se encarga de controlar el resto de componentes, siendo el que configura el modo y, en caso de ser el modo de escritura, el que configura el ID_Base. Para poder llevar acabo el control de los demás componentes, se utiliza el protocolo I2C, protocolo explicado en el punto 4.4. PROTOCOLO DE COMUNICACIÓN I2C. Cabe decir que, para poder ver los resultados del evento, modo lectura, se conectará el sistema al ordenador por su puerto serie y, mediante el monitor serie, se podrán ver los datos obtenidos de las bases en modo escritura (Laaroussi, 2019).

Sus características principales son:

- Microcontrolador ATmega2560.
- 5V de voltaje de operación.
- 54 pines de E/S digital.
- 16 pines de entrada analógica.
- Memoria Flash de 256Kb.
- Memoria SRAM de 8Kb.
- Memoria EEPROM de 4Kb.

En la ilustración que se muestra a continuación, Ilustración 20, se puede observar el esquema que se usa para todo el conjunto.

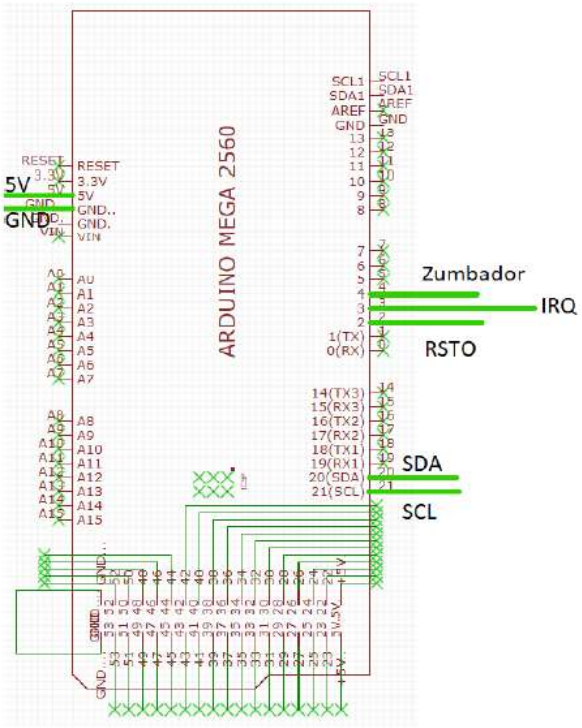


Ilustración 20: Esquema de conexiones del Arduino Mega 2560. Fuente: (Laaroussi, 2019).

Analizando la imagen, a la izquierda se encuentran los pines de alimentación del sistema, GND y 5V. En el lado opuesto están las interrupciones, pines 2 y 3, cuya función es resetear y activar la comunicación del módulo PN532. A su vez, en el pin número 4 se conecta el zumbador que dará las señales de fin de programa. Los pines 20 y 21, son los que comunican el microcontrolador con el PN532 y la RTC DS3231 (Laaroussi, 2019).

4.3.2.2. PN532 NFC MÓDULO V3

Este chip es el encargado en todo momento de leer y escribir las tarjetas NFC. Este dispositivo es capaz de leer todo tipo de etiquetas que implementan la tecnología NFC, siendo así un iniciador versátil. La velocidad de radiofrecuencia es la mencionada anteriormente, 13.56 MHz. La comunicación que tiene es también versátil debido a que, mediante unos jumpers, se puede conectar por SPI, I2C o HSU. Este hecho se puede apreciar en la Ilustración 21. A la hora de escribir, la etiqueta debe estar como máximo a 100 mm del módulo a una velocidad de 424 kbits/s, en cambio, para leer debe estar a una distancia 50 mm con una velocidad de 212 kbits/s (Laaroussi, 2019).

Sus características principales son:

- Compatible con Arduino.
- Uso de la interfaz I2C, SPI o HSU.
- Soporta todo tipo de etiquetas.
- Buen rango de velocidad y de distancia.

En la Ilustración 21, se muestra el jumper que permite cambiar el protocolo de conexión del PN532. Según este modo, se les dará un valor u otro a los bits HIS0 y HIS1, siendo 00 para el protocolo HSU, 10 para el I2C y 01 para el SPI. En la Ilustración 22, se muestra el esquema de conexiones del PN532 (Laaroussi, 2019).

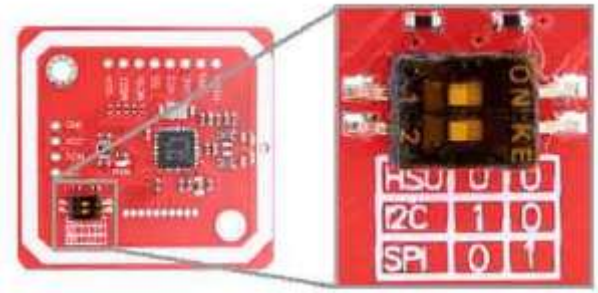


Ilustración 21: Jumper para la configuración del protocolo de comunicación del PN532. Fuente: (Laaroussi, 2019)

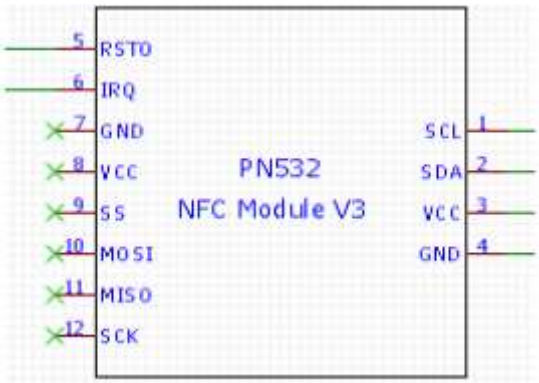


Ilustración 22: Esquema de conexiones I2C del PN532. Fuente: (Laaroussi, 2019)

4.3.2.3. DS3231 MÓDULO RTC

Para poder medir la hora y la fecha a la que se pasa por cada base de escritura, se emplea este módulo junto a una pila de bajo consumo. El I2C es el único protocolo de comunicación que tiene, es por eso que el conjunto del sistema está comunicado con el protocolo I2C (Laaroussi, 2019).

Sus características principales se detallan a continuación:

- RTC de alta precisión con oscilador interno.
- Exactitud reloj: 2ppm.
- Comunicación I2C.
- Salida de onda cuadrada programable.

Como se muestra en la Ilustración 24, este dispositivo consta de 10 pines, de los cuales se usan solamente 4. Estos pines son los de alimentación (GND y 5V) y los de comunicación con el PN532 y Arduino (SCL y SDA). En la Ilustración 24, se observa el esquema de conexiones del DS3231.

Ilustración 23: Dispositivo DS3231 con la selección de los pines utilizados. Fuente: Propia.

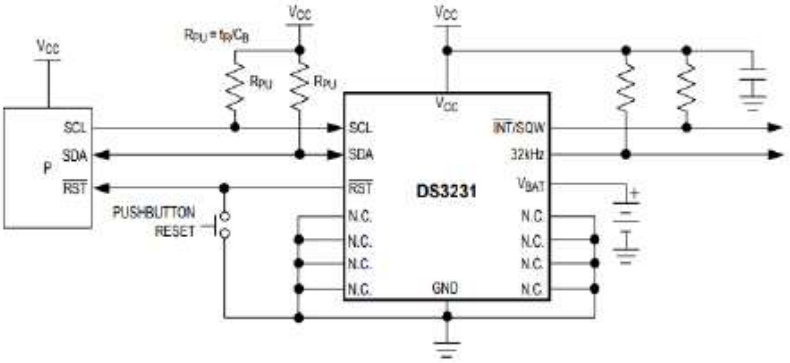


Ilustración 24: Esquema de conexiones del RTC DS3231. Fuente: (Laaroussi, 2019).

4.3.2.4. PILA CR2032

Pila de botón de 3V de litio de la marca Maxell que, junto con el DS3231, sirven para dar la hora exacta cuando esta es demandada.

Ilustración 25: Pila CR2032 de 3V. Fuente: (Laaroussi, 2019).

4.3.2.5. ZUMBADOR ACTIVO 5V

Un zumbador es un transductor electroacústico que produce un sonido o zumbido que puede ser continuo u intermitente según su programación. Como se ha mencionado varias veces en los puntos anteriores, la misión del zumbador en el proyecto es la de hacer un sonido al finalizar cada uno de los programas (Laaroussi, 2019).

Sus características principales son:

- Voltaje de funcionamiento de 3.3V a 5V.
- Salida de sonido a 85 dB.

Ilustración 26: Zumbador activo de 5V. Fuente: (Laaroussi, 2019).

4.3.2.6. BATERÍA

La batería empleada para el proyecto es una recargable de 9V y 650mAh Li-Ion con cable USB, cable mediante el cual alimentamos el sistema.

Ilustración 27: Baterías recargables de 9V y 650mAh. Fuente: (Laaroussi, 2019).

4.3.2.7. PCB

Para lograr la conexión entre todos los integrantes del proyecto, se ha creado una PCB que unifica todas las conexiones y lleva consigo misma una mejor colocación de los elementos, ya que todos están de manera fija en la PCB, en lugar de estar suelta. En la Ilustración 28, se puede observar el esquema de conexiones del sistema y, en las Ilustraciones 29 y 30, se observa la cara Top y Bottom de la PCB.

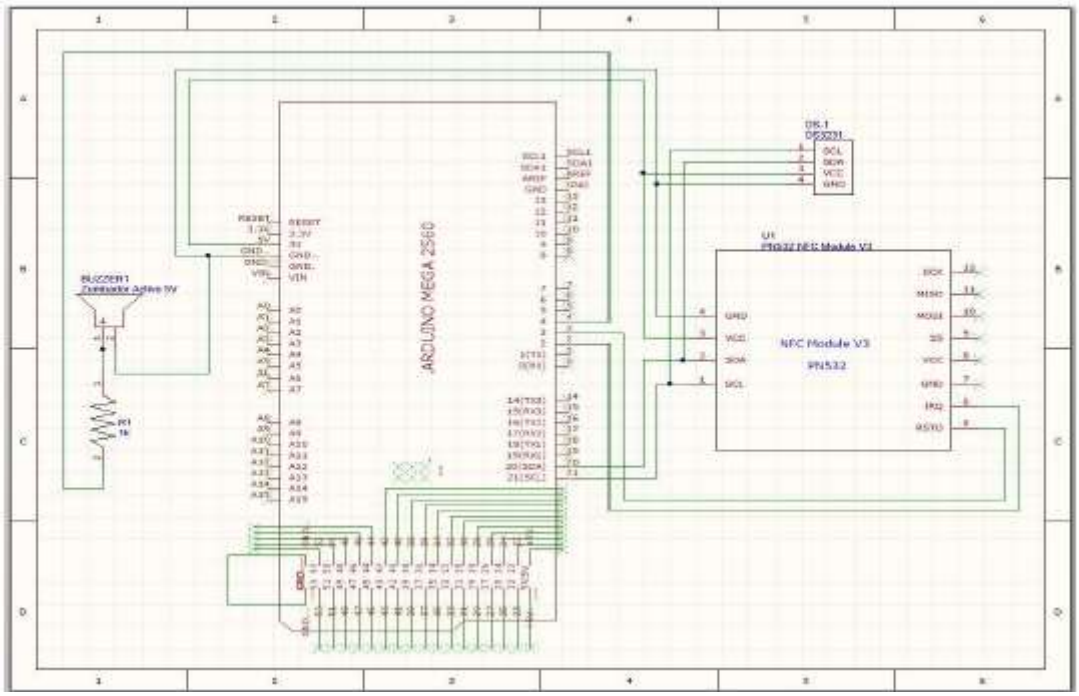


Ilustración 28: Esquema de la PCB. Fuente: (Laaroussi, 2019).

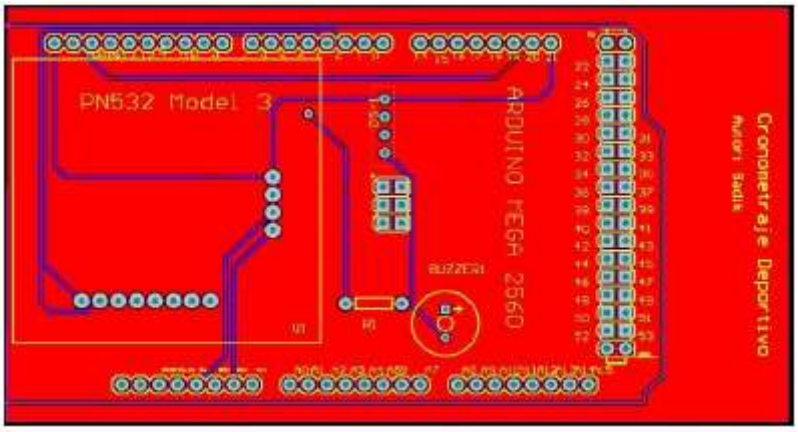


Ilustración 29: Cara Top de la PCB. Fuente: (Laaroussi, 2019).

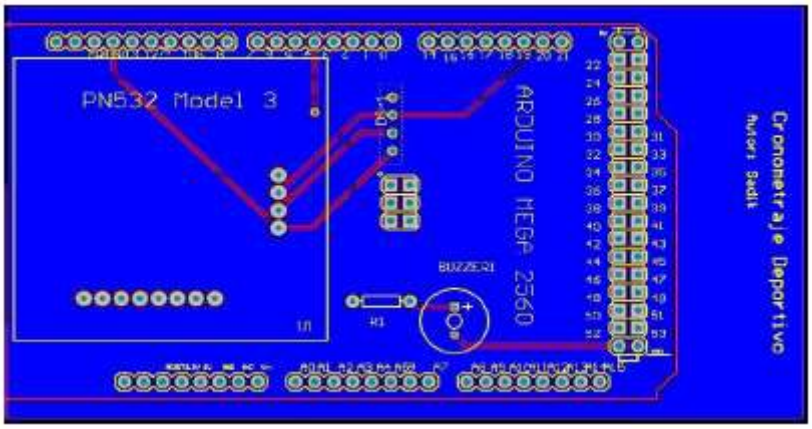


Ilustración 30: Cara Bottom de la PCB. Fuente: (Laaroussi, 2019).

4.3.2.8. TARJETA MIFARE CLASSIC EV1

La tarjeta seleccionada para el proyecto es la MIFARE CLASSIC EV1, que contiene una memoria EEPROM DE 1Kbyte, distribuida en 16 sectores que contienen 4 bloques, en los cuales cada uno tiene 16 bytes (Laaroussi, 2019).

Ilustración 31: Tarjeta de memoria MIFARE CLASSIC EV1. Fuente: (Laaroussi, 2019).

4.3.2.9. CAJA PROTECTORA

La caja es de plástico transparente cuyas medidas son: 75x135x70mm.

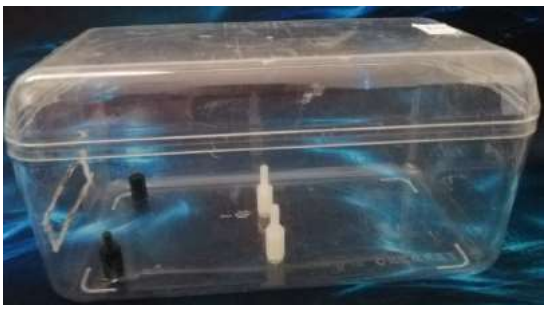


Ilustración 32: Caja protectora. Fuente: (Laaroussi, 2019).

4.4. PROTOCOLO DE COMUNICACIÓN I2C

Este protocolo es el elegido para la comunicación entre los distintos elementos del sistema. Está basado en la comunicación serial desarrollado por la empresa Phillips Semiconductors, de manera que junta lo mejor de las comunicaciones SPI y HSU. Con este protocolo, se puede tener a un maestro controlando a varios esclavos o a varios maestros controlando varios esclavos. En este proyecto, el maestro es el microcontrolador de Arduino Mega y como esclavos se tiene el PN532 y el DS3231. Este tipo de modo de comunicación únicamente necesita de dos vías de comunicación, las cuales vienen indicadas como SCL y SDA y que se puede ver en la Ilustración 33. En la Ilustración 34 se pueden observar sus características principales (Laaroussi, 2019).

Su funcionamiento se Basa en enviar la información mediante mensajes que van divididos en tramas de datos. Cada trama lleva consigo una dirección binaria del esclavo al que va dirigido el mensaje. Este mensaje conlleva las condiciones de inicio y pausa, de lectura y escritura y los bits ACK y NACK. En la Ilustración 35 se observa la estructura de un mensaje de este protocolo (Laaroussi, 2019).

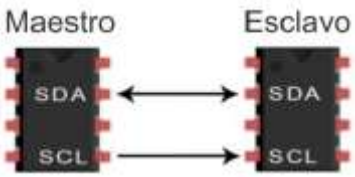


Ilustración 33: Comunicación entre maestro y esclavo. Fuente: (Laaroussi, 2019).

Número de vías o cables	2
Velocidad máxima	Modo estándar (Sm) = 100kbps
	Modo rápido (Fm) = 400kbps
	Modo High Speed (Fm+) = 3.4Mbps
	Modo Ultra Fast (Hs-mode) = 5Mbps
Síncrono o Asíncrono	Síncrono
Paralelo o Serial	Serial
Número máximo de Maestros	Ilimitado
Número máximo de Esclavos	1008

Ilustración 34: Características principales del protocolo I2C. Fuente: (Laaroussi, 2019).

Ilustración 35: Estructura de un mensaje del protocolo I2C. Fuente: (Laaroussi, 2019).

4.5. ESTRUCTURA FINAL DEL HARDWARE

Una vez montado cada componente en su correcto espacio, se configura el modo en el que se quiere que trabaje el sistema, se alimenta con la batería y se introduce en la caja protectora, quedando de la forma en la que se ven en la Ilustración 36.

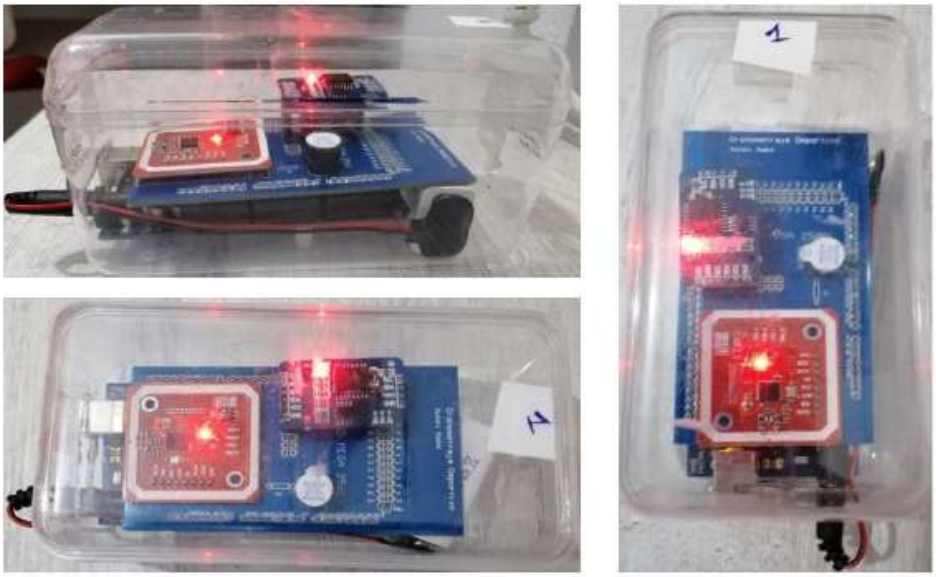


Ilustración 36: Estructura final del hardware. Fuente: (Laaroussi, 2019).

4.6. CONSUMO DEL PROTOTIPO

Una vez montado y comprobado su correcto funcionamiento, se procede a la toma de consumo del prototipo. De esta manera se podrá calcular la autonomía del sistema. En la Ilustración 37 se puede observar el consumo de la placa y se puede hacer el cálculo de la autonomía.

Ilustración 37: Consumo prototipo Laaroussi, S. Fuente: (Laaroussi, 2019).

$$Autonomía\ batería = \frac{Capacidad\ de\ corriente\ de\ batería\ (mAh)}{Consumo\ de\ correinte\ prototipo\ (mA)} = \frac{650\ mAh}{98\ mA} = 7h$$

De este modo, este prototipo creado por Laaroussi, S. tiene una autonomía de 7 horas.

4.7. PRESUPUESTO

En la Ilustración 38 se observa el coste por material y el precio total del prototipo, mientras que en la Ilustración 39, se muestra el precio de la tarjeta NFC.

PRODUCTO	CANTIDAD	COSTES (€)
Arduino Mega 2560 Rev V3	1	35,00
PN532 NFC module V3	1	3,55
RTC DS3132	1	2,25
Zumbador activo 5V	1	0,86
Batería 9V 650mAh	1	6,42
Conector clip de la batería de 9V	1	0,72
Fabricación placa PCB	1	1,82
Diversos materiales	1	1
	Total	**51,62**

Ilustración 38: Presupuesto de los materiales del prototipo. Fuente: (Laaroussi, 2019).

PRODUCTO	CANTIDAD	COSTES (€)
Tarjetas NFC tipo MIFARE Classic EV1 1K	1	0,41

Ilustración 39: Presupuesto unitario de cada tarjeta NFC. Fuente: (Laaroussi, 2019).

4.8. CONCLUSIÓN DEL PROTOTIPO

Como conclusión de este proyecto, se puede decir que se ha conseguido implementar un sistema de lectura y escritura de tarjetas NFC que puede emplearse para el cortometraje de eventos deportivos. Se debe añadir que, a pesar de ser fiable, existen varios puntos que pueden mejorarse:

- Optimización de la memoria del microcontrolador. Como se puede observar en la Ilustración 40, este tipo de sistema, tiene una memoria demasiado grande para el uso que se le da, dejando mucha vacía. Esto podría solucionarse utilizando un microcontrolador más pequeño.

Ilustración 40: Mensaje tras compilar el programa en que se da a conocer la memoria que este ocupa. Fuente: Propia.

- Tamaño grande con elementos que no se usan. Una mejora de reestructuración, eliminando los elementos que este no usan como los pines digitales y analógicos, podría ayudar a disminuir el tamaño.
- Una mejora de la implementación del Modo Verificar tarjeta vacía. Tras probar este modo de funcionamiento, me he percatado de que este siempre da error y, a pesar de que la tarjeta esté vacía, el sistema siempre muestra que está llena.
- Reducción del precio final. Mediante los cambios de los puntos anteriores, se puede llegar a implementar un sistema que sea más económico.
- Una nueva configuración de la RTC. Tras varias pruebas en varios días, se ha podido comprobar que, si se carga un programa y se ejecuta vario días después, la RTC marca la última fecha que tenía. Esto es debido a que se deja de alimentar, no sigue funcionando y no avanza su tiempo, llevando a errores de marca.
- Monitor serie. Si no hay monitor serie, no se puede configurar el modo. Esto implica que pese a grabar el programa en el sistema, cuando este deje de ser alimentado y se vuelva a alimentar, el programa volverá a pedir que se asigne el modo, con lo cual, si no hay monitor serie, no hay funcionamiento del programa.
- Por último, se ha medido el consumo del prototipo, cuando este está en pausa y cuando este está funcionando y lo valores obtenidos han sido diferentes a los mencionados por Laaroussi, S. en la Ilustración 37. En la Ilustración 41, se ve que cuando el sistema está esperando a una tarjeta, su consumo es de 0.118A. Mientras, en la Ilustración 42, se puede observar que el consumo del prototipo cuando se está tomando la lectura de la tarjeta es de 0.166A. Estos datos distan

de los obtenidos por Laarroussi, S.

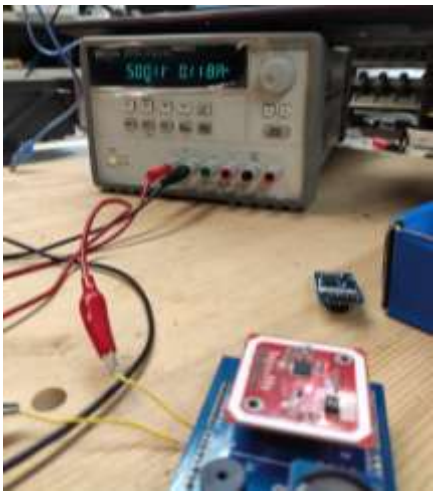


Ilustración 41: Consumo del prototipo cuando está en pausa. Fuente: Propia.

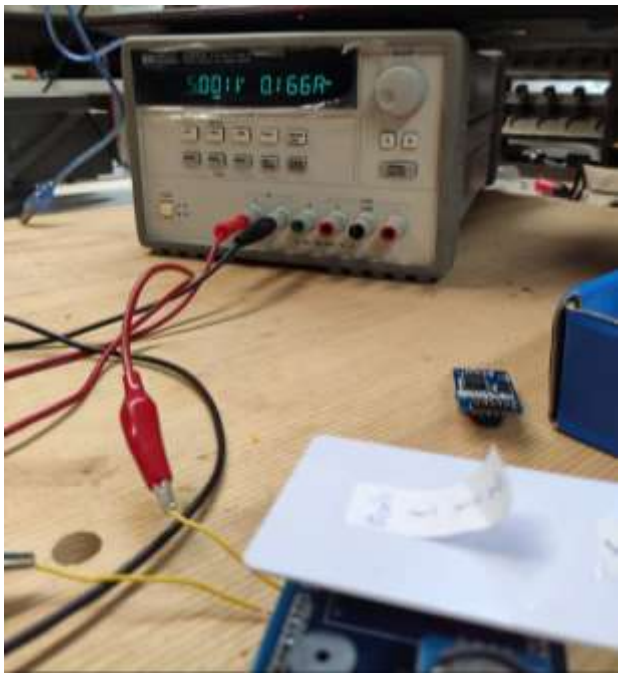


Ilustración 42: Consumo del prototipo cuando está leyendo una tarjeta. Fuente: Propia.

5. PROTOTIPO V1

Partiendo de la base del trabajo realizado por Laaroussi, S. se pretende hacer un nuevo prototipo, el v1, que mejore el conseguido por él. Para ello se va a implementar una serie de cambios en el hardware, basado en implementar en vez de los 4 programa en una sola base, tener cada programa por separado y, de esta manera, reducir la memoria que ocupa el software. De cara a la reducción del consumo, se va a introducir vía software un método de Sleep Mode, método con el que se espera que, cuando nadie interactúe con el sistema, este se ponga a descansar, consumiendo menos.

5.1. CAMBIOS EN EL SOFTWARE

En primer lugar, se ha decidido separar los programas, de esta manera, el tamaño que estos ocupen será mucho menor que estando todos juntos. También se ha decidido implementar un método de Sleep Mode para que así el consumo baje y tenga más autonomía.

5.1.1. CONFIGURACIÓN SLEEP MODE

Para la configuración de este modo, se ha analizado el datasheet de los diferentes microcontroladores de Arduino en los cuales coincide una tabla, la que se puede ver en la Ilustración 43, donde se mencionan los tipos de Sleep Mode que hay.

Table 9-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains					Oscillators		Wake-up Sources							
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	Software BOD Disable
Idle			X	X	X	X	X[2]	X	X	X	X	X	X	X	
ADC Noise Reduction				X	X	X	X[2]	X[3]	X	X[2]	X	X	X		
Power-down								X[3]	X				X		X
Power-save					X		X[2]	X[3]	X	X			X		X
Standby[1]						X		X[3]	X				X		X
Extended Standby					X[2]	X	X[2]	X[3]	X	X			X		X

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

Ilustración 43: Tipos de Sleep Mode de Arduino. Fuente: Arduino.

Entre estos 6 modos disponibles, se ha decidido utilizar el Power-down debido a que, pese a no ser el que reduce más el consumo, es el que nos ofrece una mejor implementación debido a su librería Low Power de Arduino. Esta librería se emplea mediante la instrucción: LowPower.powerDown(SLEEP_Xs, ADC_OFF, BOD_OFF).
ntre estos 6 modos disponibles, se ha decidido utilizar el Power-down debido a que, pese que no ser el que reduce más el consumo, es el que nos ofrece una mejor implementación en su debido a su librería Low Power de Arduino. Esta librería se emplea mediante la instrucción: LowPower.powerDown(SLEEP_Xs, ADC_OFF, BOD_OFF).

Esta instrucción contiene 3 parámetros, el primero, SLEEP_Xs, se decide el tiempo que se quiere poner a descansar el sistema. Este tiempo viene marcado estrictamente y solamente pueden ser de 15ms, 30ms, 60ms, 120ms, 250 ms, 500ms, 1s, 2s 4s u 8s. El segundo parámetro es ADC_OFF, este parámetro apaga los convertidores analógicos-digital, reduciendo así el consumo. Como tercer parámetro está el BOD_OFF, que se encarga de apagar el Brown Out Detection, circuito encargado de detectar los niveles bajos de tensión. Este es el encargado de proteger el sistema de

Arduino de tensiones bajas que puedan dañar los componentes. De esta manera se apagaría el sistema y, mediante el primer parámetro, se despertaría. Este despertar es debido a que ese primer parámetro es el Watchdog. El Watchdog es un temporizador que va disminuyendo continuamente un contador hasta llegar a cero. Cuando este llega a ese valor, reinicia el sistema, reiniciando nuestro programa sin dañar ningún componente. En la Ilustración 44, se puede observar un ejemplo de cómo utilizar la instrucción para que el sistema duerma 8 segundos.

```
LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
```

Ilustración 44: Ejemplo de la instrucción LowPower. Fuente: Propia.

Una vez decidido el modo en el que se dormirá el sistema, se configura vía software un modo Sleep Mode a través del cual se puede decidir cuánto tipo se quiere que se duerma la placa, si 1s, 2s, 4s u 8s. Para ello se implementa una función, Ilustración 45, en cada uno de los modos, llamada Asignar_Tiempo_Sleep_Mode. En esta fusión de manera simple e intuitiva, se podrá asignar dichos tiempos, asegurándose con el while de Ilustración 46 que los segundos introducidos son 1, 2, 4 u 8, si no, se volverá a preguntar cuántos segundos se quiere que duerma el sistema.

```
//Funcion para insertar el tiempo de Sleep Mode
int Asignar_Tiempo_Sleep_Mode() {
    Serial.println("Escribe los segundos del Sleep Mode: ");
    Serial.println();
    Serial.println("Introduce un nuúmero entre 1, 2, 4 u 8:");
    //No hacer nada hasta que se indruduca el tiempo del Sleep Mode
    while(Tiempo_Sleep == false){
        //Para eliminar el error en el buffer y no coger un dato Erroneo
        if (Serial.available()!=0){
            //Leer del puerto serial un dato en decimal
            Error_Sleep = Serial.parseInt();
        }
        delay(500);
        if(Serial.available()!=0){
             //lee un numero decimal
             T_Sleep = Serial.parseInt();
             Serial.print("El tiempo del Sleep Mode seleccionado es:  ");
             delay(500);
             Serial.println(T_Sleep);
             Tiempo_Sleep = true;
             delay(500);
        }
    }
    return T_Sleep;
}
```

Ilustración 45: Función Asignar_Tiempo_Sleep_Mode. Fuente: Propia.

```
//Cuando se haya introducido un Tiempo erróneo, volvemos a configurar el Sleep Mode
while (ValidSM == false){
    Config_Sleep_Mode(T_Sleep);
}
```

Ilustración 46: Código para asegurarse de que el tiempo está dentro de los parámetros definidos. Fuente: Propia.

Una vez hecho esto, se ha implementado la función Config_Sleep_Mode que duerme el sistema. Esta consta de los 4 tiempos que se deben elegir y cada uno de ellos se ejecuta de manera conveniente. En la Ilustración 47, se puede observar la estructura de dicho código. En caso de que no sea un tiempo correcto, se volverá a Asignar_Tiempo_Sleep_Mode.

```
// Funcion para entrar en el Sleep Mode segun se seleccionó
void Config_Sleep_Mode(int T_Sleep) {
    if(T_Sleep == 1){
        Serial.println("Duerme 1 segundo");
        delay(500);
        LowPower.powerDown(SLEEP_1S, ADC_OFF, BOD_OFF);
        //Serial.println();
        Serial.println("Esperando Etiqueta NFC");
        ValidSM = true;
        delay(500);
    }
    else if(T_Sleep == 2){
        Serial.println("Duerme 2 segundos");
        delay(500);
        LowPower.powerDown(SLEEP_2S, ADC_OFF, BOD_OFF);
        //Serial.println();
        Serial.println("Esperando Etiqueta NFC");
        ValidSM = true;
        delay(500);
    }
    else if(T_Sleep == 4){
        Serial.println("Duerme 4 segundos");
        delay(500);
        LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF);
        //Serial.println();
        Serial.println("Esperando Etiqueta NFC");
        ValidSM = true;
        delay(500);
    }
    else if(T_Sleep == 8){
        Serial.println("Duerme 8 segundos");
        delay(500);
        LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
        //Serial.println();
        Serial.println("Esperando Etiqueta NFC");
        ValidSM = true;
        delay(500);
    }
    else{
        Serial.println("Los segundos del Sleep Mode introducidos son incorrectos, debe ser 1, 2, 4 u 8");
        Tiempo_Sleep = false;
        ValidSM = false;
        Asignar_Tiempo_Sleep_Mode();
        delay(500);
    }
}
```

Ilustración 47: Estructura de la función Config_Sleep_Mode. Fuente: Propia.

Estas funciones son para cada uno de los modos, ya que se requiere que en el momento en que no se estén ejecutando, se duerman de manera definida. Se ha previsto que el sistema esté en funcionamiento 3 segundos, tiempo en el que programa es capaz de escribir 3 tarjetas en el modo escritura y, para el resto de modos, tiempo para leer, borrar y verificar una tarjeta, para luego ir al Sleep Mode.

5.1.2. FLUJOGRAMAS PROTOTIPO V1

5.1.2.1. MODO ESCRIBIR

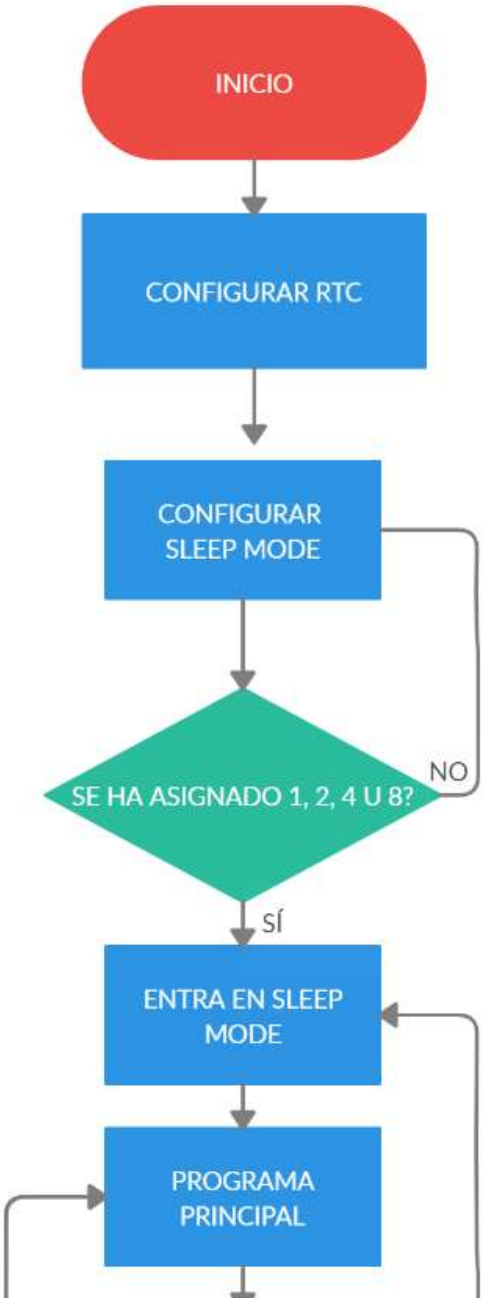
En la Ilustración 48 se puede observar cómo es el flujograma de este programa. Primero, se configura la RTC y se configura un ID_Base. A continuación, si el número es válido,

se entra en la configuración del Sleep Mode que, si también es correcto, se pone a dormir. Después, entramos en el bucle del programa, donde primero si no han pasado 3s desde el último Sleep Mode, se avanza a ver si se ha detectado una tarjeta. Esto se hace para que el programa no se quede esperando la tarjeta durante mucho tiempo. Si se detecta y no han pasado esos 3 s, se autentifica la tarjeta, de ser correcta su autentificación, el programa escribe el dato, suena un pitido para confirmar el fin del proceso, y se vuelve al inicio del bucle.

Ilustración 48: Flujograma Modo Escritura con configuración de Sleep Mode. Fuente: Propia.

5.1.2.2. FLUJOGRAMA MODO LECTURA

Según muestra la Ilustración 49, este modo es igual que el de escritura, pero sin tener que configurar la ID_Base. Sus procesos son iguales, empezando por configurar la RTC. El siguiente paso es configurar el Sleep Mode, verificando que el número seleccionado es uno de los válidos. Entonces, se entra en el bucle principal del programa, donde en primer lugar se mira si han pasado 3 s desde el último Sleep Mode. Si no es así, mientras se espera a la tarjeta y se vuelve a preguntar si han pasado esos 3 s. Si tampoco ocurre esto, y se ha detectado la tarjeta, se avanza a la autentificación de la tarjeta que, de ser correcta, se pasa a leer la memoria de esta. Cuando acaba de leerla, se emite un pitido y vuelve al programa principal, bucle principal.


INICIO
CONFIGURAR RTC
CONFIGURAR SLEEP MODE
SE HA ASIGNADO 1, 2, 4 U 8?
NO
SÍ
ENTRA EN SLEEP MODE
PROGRAMA PRINCIPAL

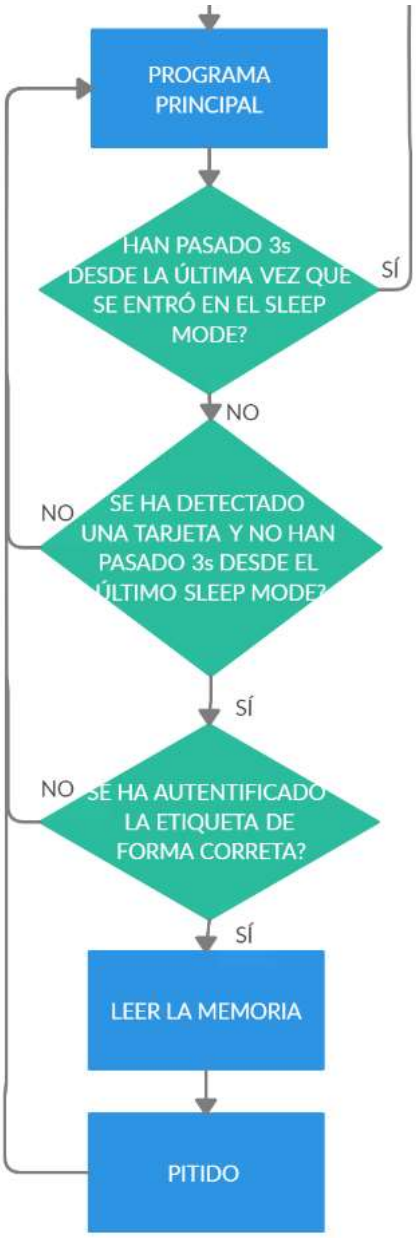


Ilustración 49: Flujograma Modo Lectura con configuración de Sleep Mode. Fuente: Propia

5.1.2.3. FLUJOGRAMA MODO BORRAR

Este tercer modo, como se observa en la Ilustración 50, es idéntico que el de lectura. Su única diferencia es que, en vez de leer los datos de la memoria, este los suprime, suena el pitido y se vuelve al principio del programa, al bucle principal.

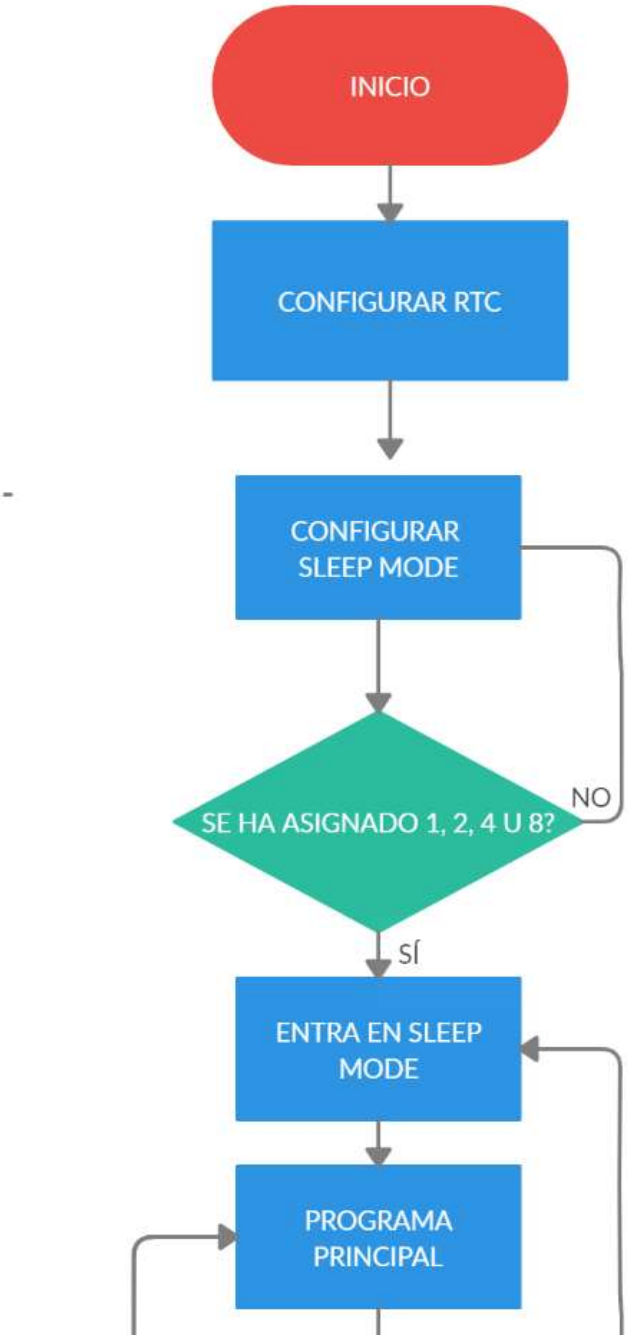
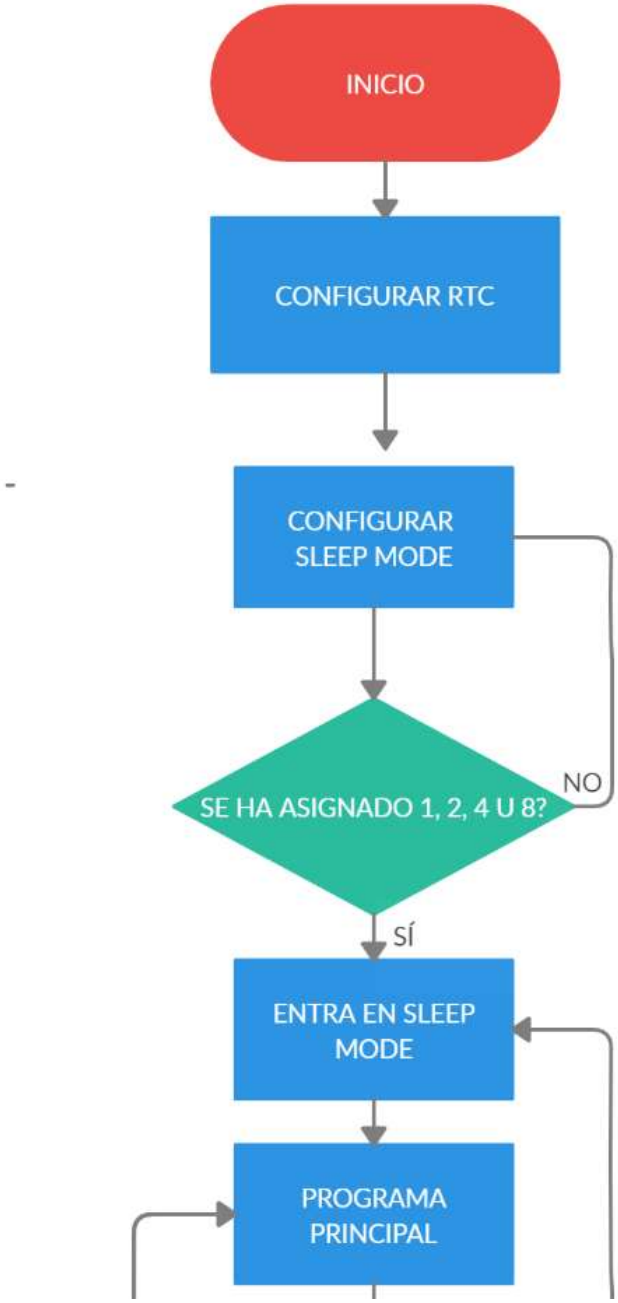

INICIO
CONFIGURAR RTC
CONFIGURAR SLEEP MODE
SE HA ASIGNADO 1, 2, 4 U 8?
NO
SÍ
ENTRA EN SLEEP MODE
PROGRAMA PRINCIPAL

Ilustración 50: Flujograma Modo Borrar con configuración de Sleep Mode. Fuente: Propia

5.1.2.4. FLUJOGRAMA MODO VERIFICAR TARJETA VACÍA

Este último modo, es muy similar a los dos anteriores. En la Ilustración 51, se puede observar que su única diferencia respecto los dos anteriores radica en que, en vez de leer o borrar, verifica que los datos están suprimidos.


INICIO
CONFIGURAR RTC
CONFIGURAR SLEEP MODE
SE HA ASIGNADO 1, 2, 4 U 8?
NO
SÍ
ENTRA EN SLEEP MODE
PROGRAMA PRINCIPAL

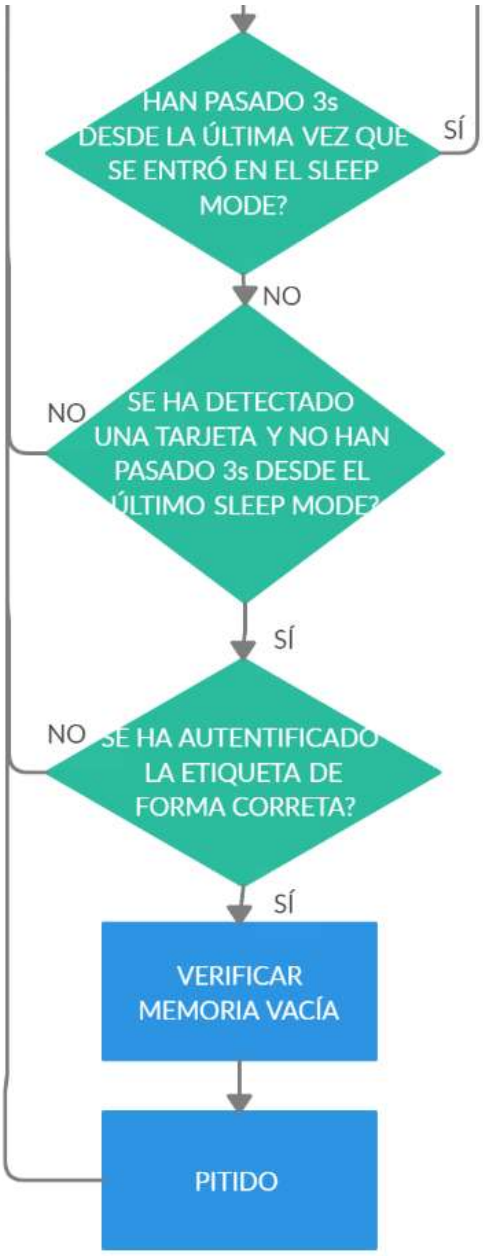


Ilustración 51: Flujograma Modo Verificar tarjeta vacía con configuración de Sleep Mode. Fuente: Propia.

5.2. CAMBIOS EN EL HARDWARE

5.2.1. CAMBIO MICROCONTROLADOR

Como se ha mencionado en uno de los puntos de las conclusiones extraídas del prototipo anterior, el microprocesador es demasiado potente para el uso que se le da, debido a que queda mucha memoria libre como demuestra la Ilustración 40. Para ello se va a sondear el mercado de microprocesadores de Arduino para buscar el ideal para esta aplicación. En la Ilustración 43, se puede observar todos los microprocesadores de Arduino y, en los diferentes tipos de memoria que tiene, qué cantidad de datos pueden almacenar.

Tipo de Arduino	Memoria Flash (kb)	Memoria SRAM (kb)	Memoria EEPROM (kb)
Duemilanove	16 (2kb bootloader)	1	0.512
Nano	16 (2kb bootloader)	1	0.512
Mini	16 (2kb bootloader)	1	0.512
BT	16 (2kb bootloader)	2	1
Esplora	32 (4kb bootloader)	2.5	1
Leonardo	32 (4kb bootloader)	2.5	1
Micro	32 (4kb bootloader)	2.5	1
Yún	32 (4kb bootloader)	2.5	1
Pro Mini	32 (2kb bootloader)	2	1
Ethernet	32 (0.5kb bootloader)	2	1
Uno	32 (0.5 kb bootloader)	2	1
Mega	256 (8kb bootloader)	8	4
Mega ADK	256 (8kb bootloader)	8	4

Ilustración 52: Microcontroladores de Arduino y su memoria. Fuente: Propia.

Tras compilar el nuevo software con el Arduino Mega 2560, controlador utilizado por Laaroussi, S., aparece el mensaje de la Ilustración 53. Hay que recalcar que el programa compilado en esa ilustración, es el de escritura. Esto es debido a que es el programa más pesado de los cuatro.

```
El Sketch usa 12246 bytes (37%) del espacio de almacenamiento de programa.
Las variables Globales usan 1580 bytes (77%) de la memoria dinámica, dejanc
```

Ilustración 53: Memoria Flash y SRAM que utiliza el software. Fuente: Propia.

Mirando el dato obtenido de la compilación, los controladores a los que se podrían optar, según Ilustración 52, son todos los de 32 kb de memoria flash y, de los que tienen 16 kb de memoria flash, solamente al BT debido a que el programa necesita 1580 bytes de memoria SRAM y el resto solamente tienen 1000 bytes. Finalmente se ha optado por el Arduino UNO.

Razones de la elección:

- Similitud con el Arduino Mega 2560. El Arduino Mega y el UNO son dos Arduinos que, en cuanto a componentes y diseño, se parecen bastante.
- Memoria. Es cierto que el microcontrolador BT es el que mejor se adapta a la memoria que requiere el programa. Pero se ha optado por uno de más memoria flash por si en un futuro se requiere de alguna implementación más que deba ir en el código y ocupe más memoria.
- Fácil acceso a compra. Arduino UNO es de los más vendidos y de los que más accesibles, ya que lo puedes encontrar en cualquier tienda de electrónica o informática, mientras que los otros son más difíciles de encontrar.

5.2.1.1. COMPONENTES ARDUINO UNO SELECCIONADOS

Anteriormente se ha decantado por el Arduino UNO, pero si se quiere hacer una mejor optimización del sistema, se deberá eliminar todo aquel componente que no se use. Para ello, hay que fijarse en el esquema del Arduino UNO, Ilustración 54.

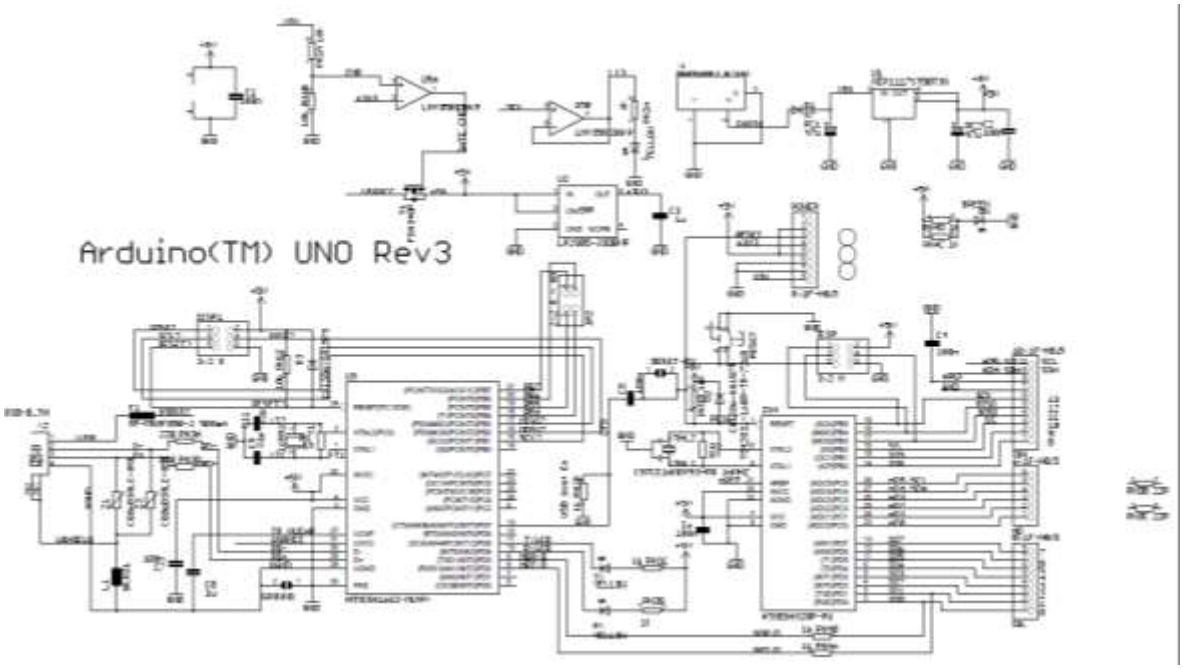


Ilustración 54: Esquema Arduino UNO. Fuente: Arduino.

Puede verse en el esquema que tiene dos microcontroladores, Ilustración 55. El primer controlador, ATMEGA16U2, es el encargado de la transmisión de información con el protocolo USB. De esta manera, cuando se conecta al ordenador la placa por USB, hace de intermediario entre el ordenador y el microcontrolador ATMEGA328P. Este segundo microcontrolador es el encargado de procesar el código de Arduino, ejecutándolo. Este microcontrolador es el que se utilizará en el prototipo v1.

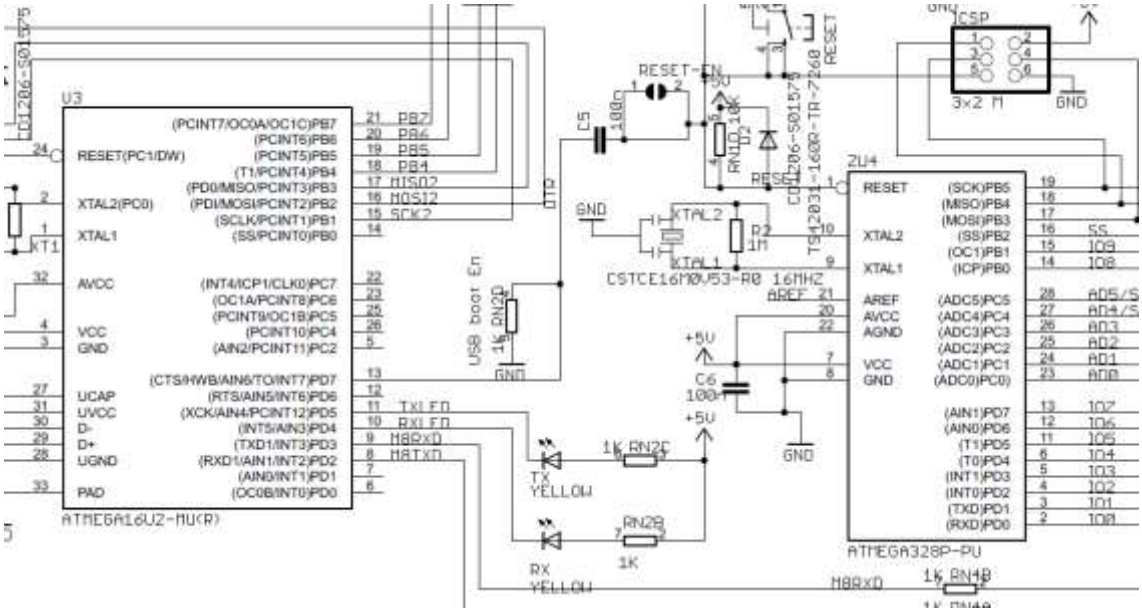


Ilustración 55: Ampliación del Esquema Arduino UNO donde se ven los dos microcontroladores. Fuente: Arduino.

Para su correcto funcionamiento se ha decidido utilizar los componentes que ya tiene como recomienda el Datasheet del microcontrolador. Por tanto, se ha suprimido todo aquello que no es esencial:

- ATMEGA16U2. Este microcontrolador no será necesario debido a que no alimentaremos el prototipo por la entrada USB. Esta medida implica que se

eliminará todo aquello que sea controlado por él. Para cargar el programa en el microcontrolador, primero este se pondrá en un Arduino UNO y, cuando se haya finalizado la carga, este será traspasado al prototipo v1.

- Regulador de potencia LP2985. Este regulador será suprimido debido a que el microprocesador será alimentado directamente de la fuente a 5V.
- Tirapines. Todos lo tirapines que tiene el ATMEGA328P serán eliminados.

Dicho esto, solamente se utilizarán los elementos que garantizan la seguridad y el funcionamiento del microprocesador (Ilustración 56). De los componentes seleccionados, se ha decidido cambiar solamente el cristal, CSTCE16M0V53 de 16MHz, por un resonador (COM-09420) de la misma frecuencia, debido al alto precio que tiene el cristal.

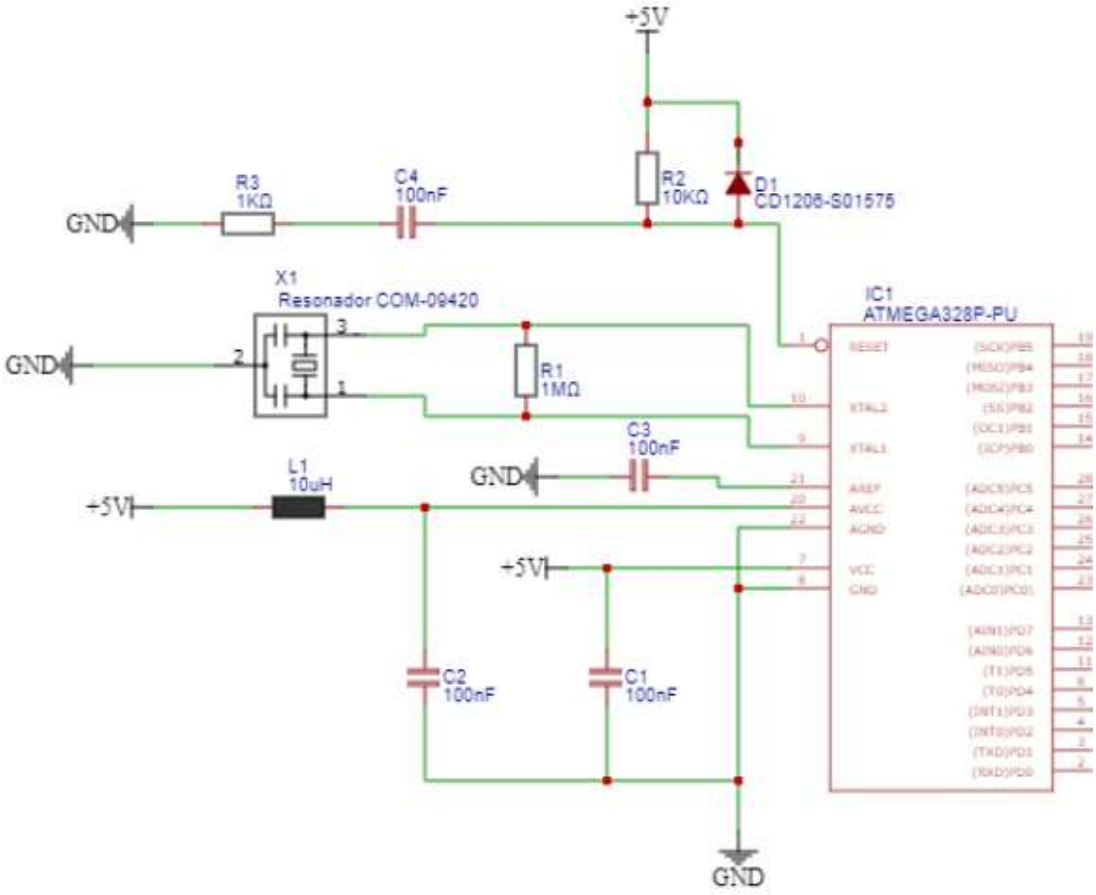


Ilustración 56: Elementos del Arduino UNO que han sido seleccionados. Fuente: Propia.

De esta manera, el esquema resultante del microprocesador es el de ilustración anterior, teniendo los siguientes componentes:

- Microcontrolador ATMEGA328P.
- 3 resistencias de 1 KΩ, 1 KΩ y 1 MΩ.
- 4 condensadores de 100 nF.
- 1 bobina de 10 µH.
- 1 resonador de 16MHz.

5.2.1.2. CAMBIO BATERÍA

De cara a la batería, se ha decidido utilizar una como la de la Ilustración 57, conocida como batería LI-ION 18650 PACK. Sus características principales son sus 3.7 V de tensión, que es recargable y su capacidad de corriente de 2600 mAh. Se ha cambiado la batería debido a que, la utilizada en el anterior proyecto era de 9 V cosa que, según

las características del microprocesador, no son funcionales porque este prototipo estará alimentado a 5 V. También se ha pasado de una capacidad de corriente de 650 mAh a una de 2600 mAh, incrementándola un 400%.

Ilustración 57: Batería de 3.7V y 2600mAh. Fuente: Propia.

Para poder alimentar el sistema a 5 V, se ha decidido utilizar un escudo para la batería (Ilustración 58) con regulador a 3 V y 5 V, de esta manera el circuito será alimentado a la tensión correspondiente.

Ilustración 58: Escudo batería con regulador a 3V y 5V. Fuente: Propia.

5.2.2. CAJA PROTECTORA

Para proteger el sistema contra la lluvia y otros efectos ambientales y contra caídas, se ha decidido utilizar otra caja diferente a la del proyecto de Laaroussi, S. Esta caja es de plástico sólido PVC gris, con la clasificación IP66, lo que significa que la caja protege contra lluvia, salpicaduras de agua y que no se oxida. Sus dimensiones exteriores son de 120x80x50mm mientras que las interiores son de 112x72x45mm. Cabe recalcar que viene con 4 tornillos para atornillar la tapa al resto de la caja y así garantizar que no se pueda abrir.

Ilustración 59: Caja protectora. Fuente: Propia.

5.2.3. HARDWARE NO CAMBIADO

El resto del hardware del sistema, así como la tarjeta NFC, son los mismos utilizados en el proyecto de Laaroussi, S. Para ver sus características y no volver a repetirlas, tan

solo hay que ir a los siguientes puntos:

- PN532 NFC Módulo V3 (Punto 4.3.2.2. PN532 NFC MÓDULO V3).
- DS3231 Módulo RTC (Punto 4.3.2.3. DS3231 MÓDULO RTC).
- Pila CR2032 (Punto 4.3.2.4. PILA CR2032).
- Zumbador (Punto 4.3.2.5. ZUMBADOR ACTIVO 5V).
- Tarjeta NFC (Punto 4.3.2.8. TARJETA MIFARE CLASSIC EV1).

5.2.4. CONEXIONES HARDWARE

El conexionado del sistema, quitando el nuevo microprocesador y sus componentes, queda idéntico debido a las similitudes entre el Arduino Mega y el Arduino UNO. Esto se puede observar en la Ilustración 60. Sus conexiones serán las siguientes:

- Pin 4 del ATMEGA328P: se conectará el pin IRQ del PN532 encargado de la rutina de interrupción 0.
- Pin 5 del ATMEGA328P: se conectará el pin RSTO del PN532 encargado de la rutina de interrupción 0.
- Pin 6 del ATMEGA328P: se conectará el positivo del zumbador mediante una resistencia de 1 KΩ.
- Pin 27 del ATMEGA328P: se conectará el pin SDA del PN532 y del DS3231, encargado de la comunicación I2P.
- Pin 28 del ATMEGA328P: se conectará el pin SCL del PN532 y del DS3231, encargado de la comunicación I2P.
- DS3231: los pines utilizados restantes, VCC y GND, irán conectados al escudo de la batería.
- PN532: los pines utilizados restantes, VCC y GND, irán conectados al escudo de la batería.

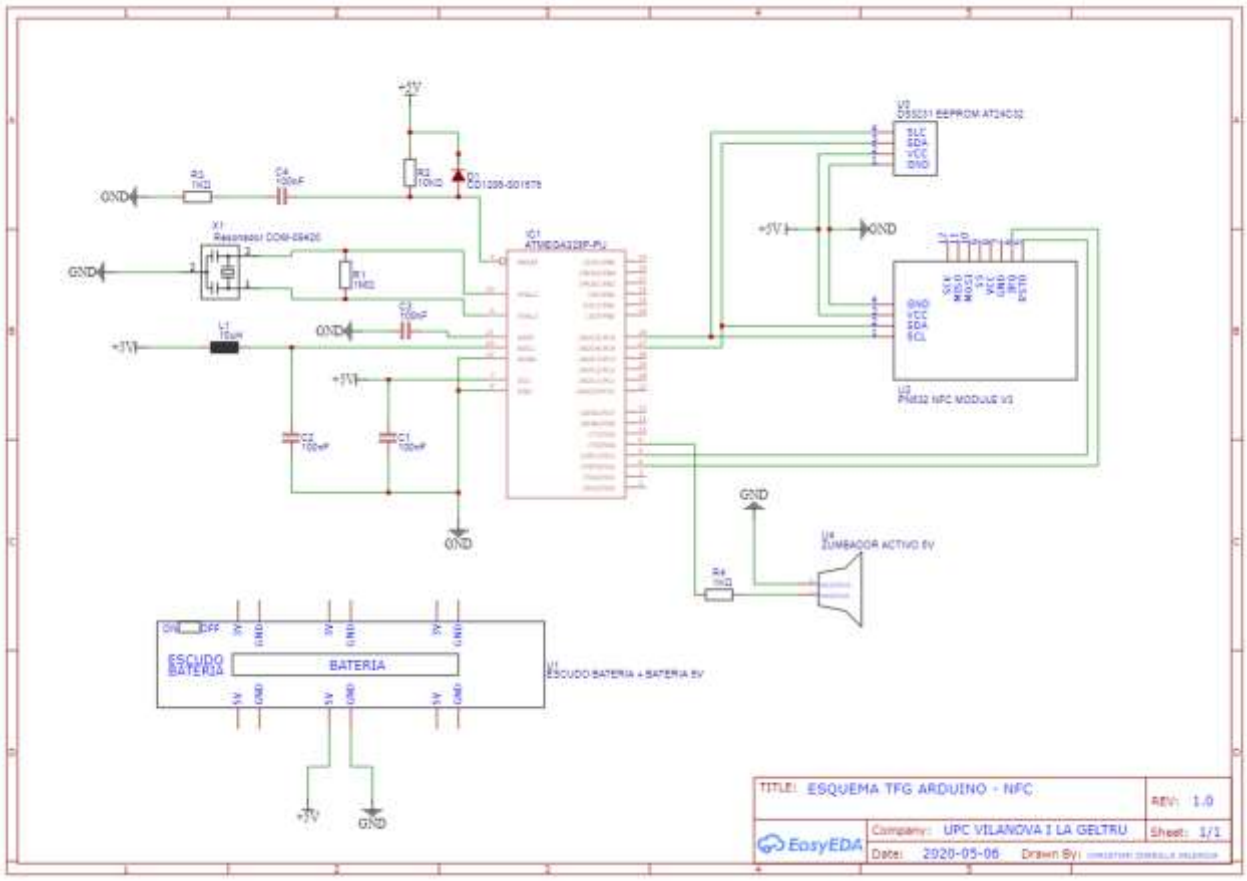


Ilustración 60: Esquema conexiones prototipo v1. Fuente: Propia.

5.2.5. PCB PROTOTIPO V1

Para llevar a cabo las conexiones de la Ilustración 60, se ha decidido implementarlas mediante una PCB. De esta manera, el sistema quedará bien ordenado, ocupando cada componente su espacio. En la Ilustración 61 se puede observar la cara Top del prototipo v1 y en la Ilustración 62, se observa la cara Bottom.

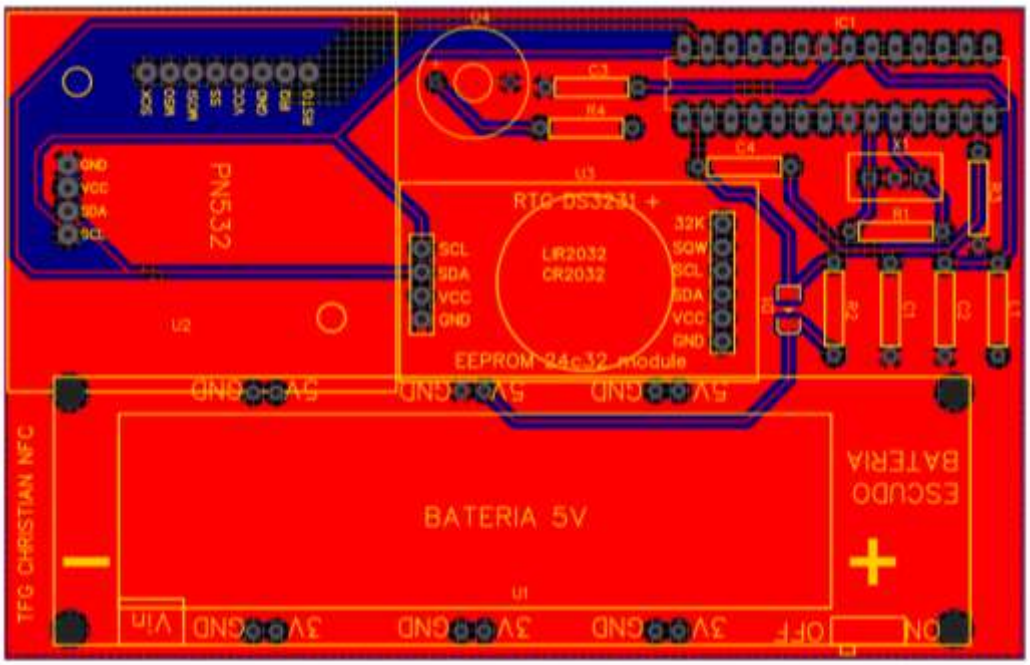


Ilustración 61: Cara top prototipo v1. Fuente: Propia.

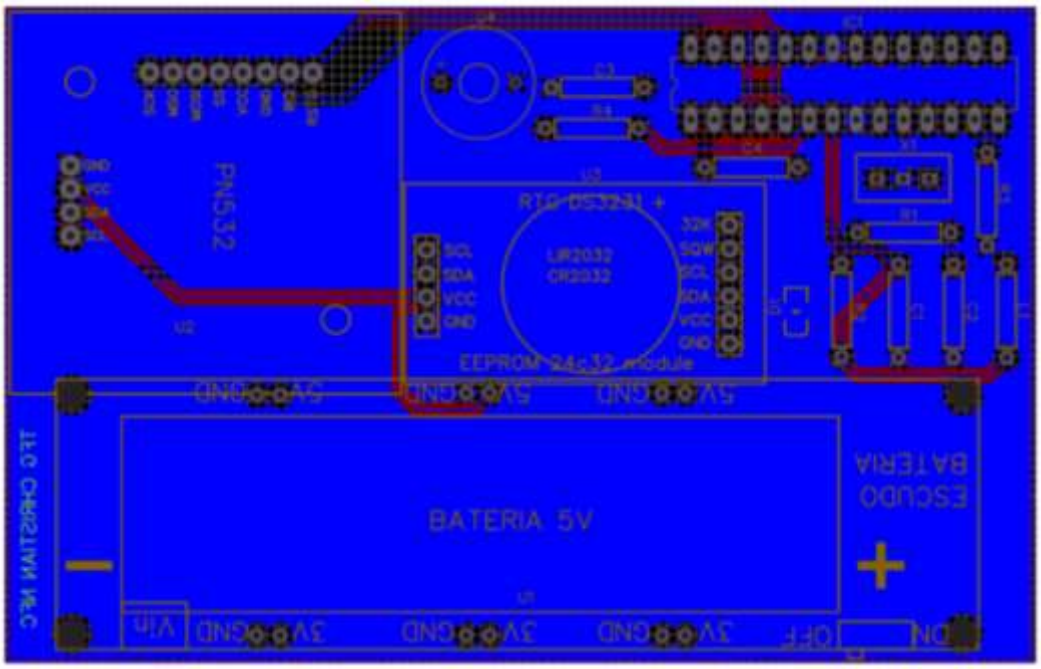


Ilustración 62: Cara Bottom prototipo v1. Fuente: Propia.

5.3. LISTA COMPONENTES

Todos los elementos que componen el sistema del prototipo 1, son los siguientes:

- Microcontrolador ATMEGA328P.
- Soporte ATMEGA328P.
- Escudo batería con regulador a 3 V y 5 V.
- Batería de 3.7 V con capacidad de corriente de 2600 mAh.
- Zumbador activo a 5V.

- PN532 NFC Módulo V3.
- DS3231 MÓDULO RTC.
- Pila CR2032.
- Diodo CD1206-S01575.
- Resonador COM-09420.
- Resistencias de: 1 MΩ, 10 KΩ y dos de 1 KΩ.
- Bobina de 10µH.
- 4 condensadores de 100 nF.

5.4. PUESTA EN MARCHA

Una vez llegados a este punto, llega el momento crucial. Por un lado, se ha verificado que todos los puntos que van unidos estén bien. En segundo lugar, se han soldado todos los componentes y, tras conectar la batería, se ha verificado que llega la tensión correcta a los componentes. Finalmente, se ha cargado el programa al microcontrolador y este ha sido conectado en la PCB, llegando así a obtener los resultados del prototipo v1.

5.4.1. PRIMERAS CONCLUSIONES DEL PROTOTIPO V1

Una vez cargado el programa en el micro y este puesto en la PCB, se ha comprobado que el sistema no funcionaba como debía. Primero de todo se ha observado mediante el osciloscopio si el pin 1 (pin reset) estaba funcionando. Tras observar que este estaba siempre a 5 V, se ha descartado la idea de que este fuera el problema. Una vez hecha esta comprobación, se ha mirado también con el osciloscopio, si el micro está funcionando a la frecuencia de 16 MHz. Al hacer esto, se ha podido comprobar que este no iba a la frecuencia mencionada.

Para corregir este error, se ha mirado el datasheet del resonador (Ilustración 63) y se ha observado el fallo. El fallo viene dado porque se ha supuesto que venían los condensadores incluidos. Al pensar esto, como se puede observar en la Ilustración 60, no se han incluido, pero sí se debería haber hecho eso. Para solucionar este error, se han soldado 2 condensadores de 22 pF entre su pin 1 y GND y entre su pin 3 y GND.

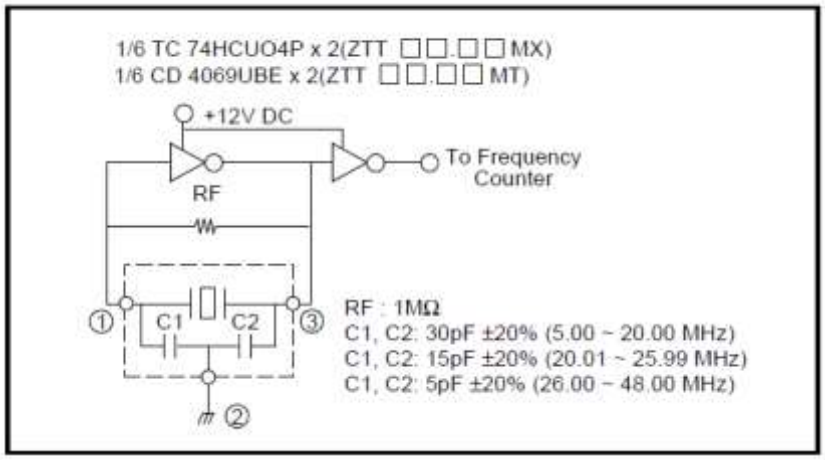


Ilustración 63: Esquema Resonador COM-09420. Fuente:

Una vez solucionado esto, se ha observado que el sistema sí que funciona correctamente a excepción del zumbador. Se ha comprobado mediante el osciloscopio si, cuando el programa de escritura grababa la información en la tarjeta, el pin 6 del microcontrolador, se ponía a 5 V. Tras ver que era así, se ha vuelto a tomar la tensión

del zumbador y se ha podido constatar que la primera vez que se tomó, se hizo mal, ya que la tensión que le llegaba era muy inferior a 5 V. Visto esto, se ha suprimido la resistencia R4, se ha conectado un jumper en sus pads y se ha vuelto a probar su funcionamiento. Finalmente, tras esta corrección la sustitución de la R4 por un cable y el haber añadido dos condensadores en la Bottom de la PCB, se ha comprobado que el sistema funciona perfectamente.

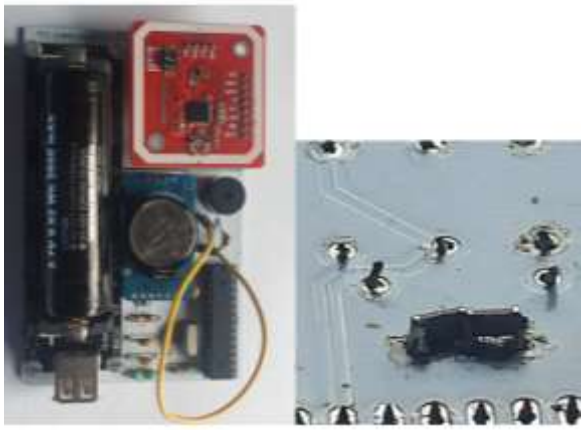


Ilustración 64: Prototipo v1 con sus cambios finales. Fuente: Propia.

5.4.2. FUNCIONAMIENTO, CONSUMO Y ATONOMÍA PROTOTIPO V1

Para probar y demostrar el correcto funcionamiento del sistema, se ha decidido documentar mediante fotos, los diferentes pines del microprocesador para el Modo Escritura con un Sleep Mode de 4 segundos. En la Ilustración número 65, se observa que el pin de reset (pin 1) siempre está activo a 5 V. Esto implica su correcto funcionamiento y, pese a que la placa vaya a dormir, este sigue funcionando.

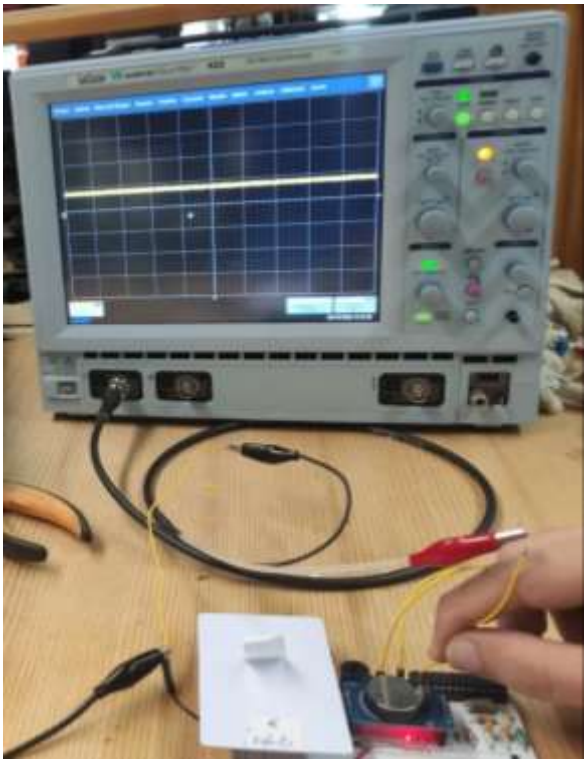


Ilustración 65: Onda del pin 0. Fuente: Propia.

A continuación, se ha comprobado el funcionamiento del resonador. En este se puede observar (Ilustración 66), que cuando el micro está en el Sleep Mode, por el osciloscopio solamente se muestra ruido. Mientras que cuando trabaja (Ilustración 67), se ve que oscila a 16 MHz.

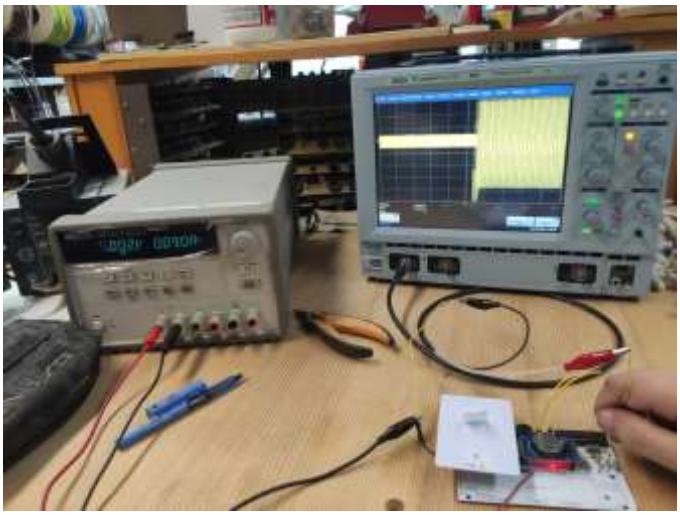


Ilustración 66: Resonador cuando el micro está en Sleep Mode. Fuente: Propia.

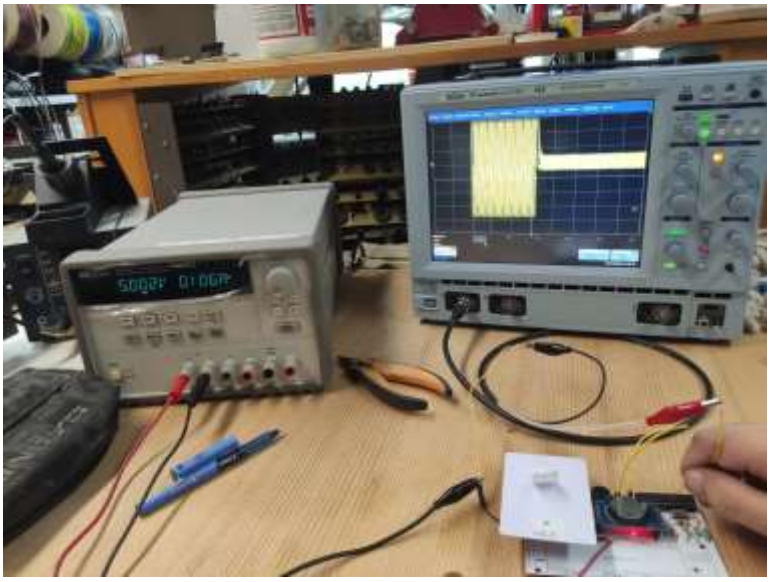


Ilustración 67: Resonador cuando el micro trabaja. Fuente: Propia.

Para acabar de demostrar su funcionamiento y su consumo, se ha decidido pinchar en el pin 6, pin que manda la señal al zumbador para que este pite. En la Ilustración 68 se observa que cuando este está en el Sleep Mode, su consumo es de 0.09 A. En cambio, cuando sí funciona, su consumo asciende a los 0.105 A. Este consumo se puede ver en las Ilustraciones 69, 70 y 71, las cuales indican las tres lecturas que es capaz de hacer el sistema antes de volver a entrar en Sleep Mode.

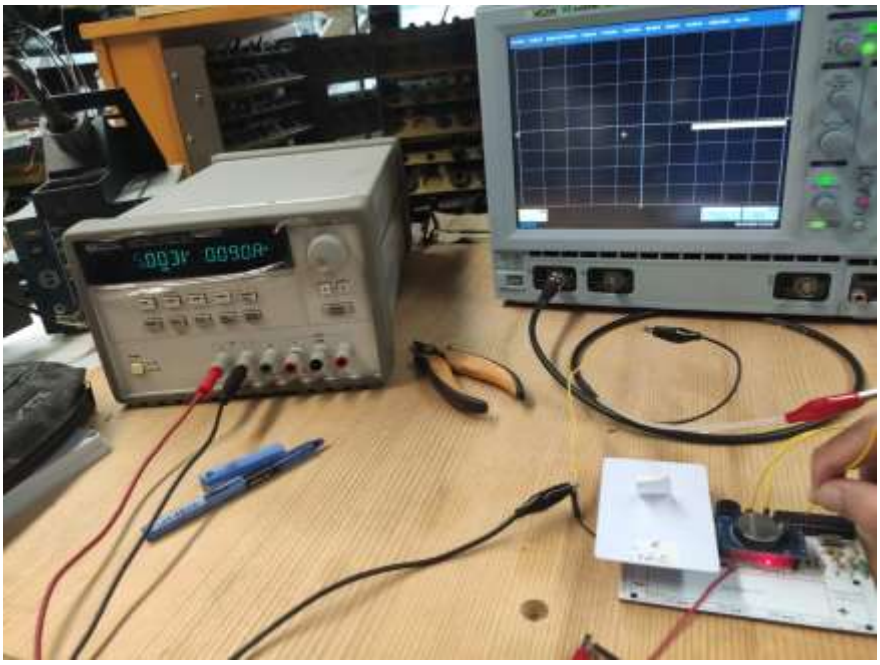


Ilustración 68: Micro en Sleep Mode. Fuente: Propia.

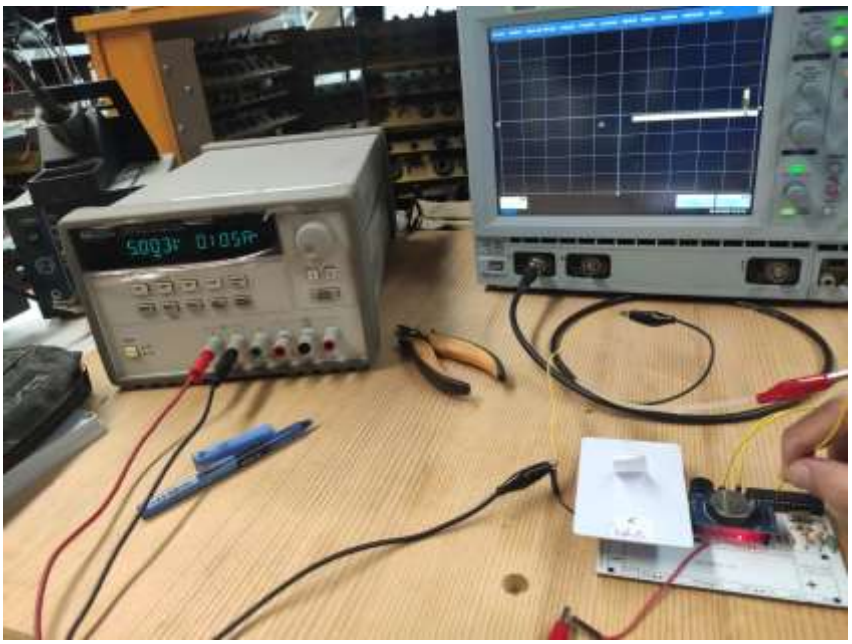


Ilustración 69: Micro en funcionamiento tras detectar una tarjeta. Fuente: Propia.

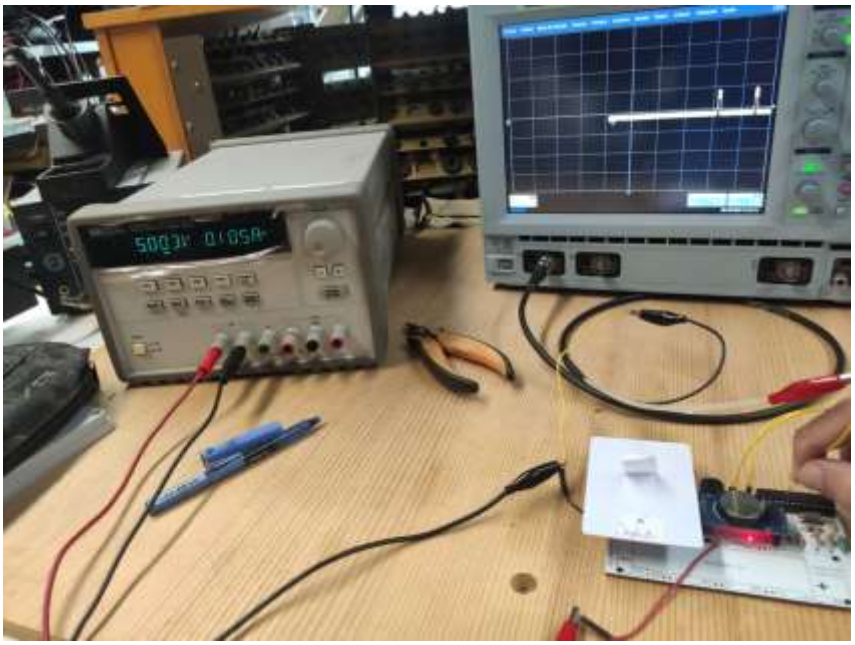


Ilustración 70: Micro en funcionamiento tras detectar una segunda tarjeta. Fuente: Propia.

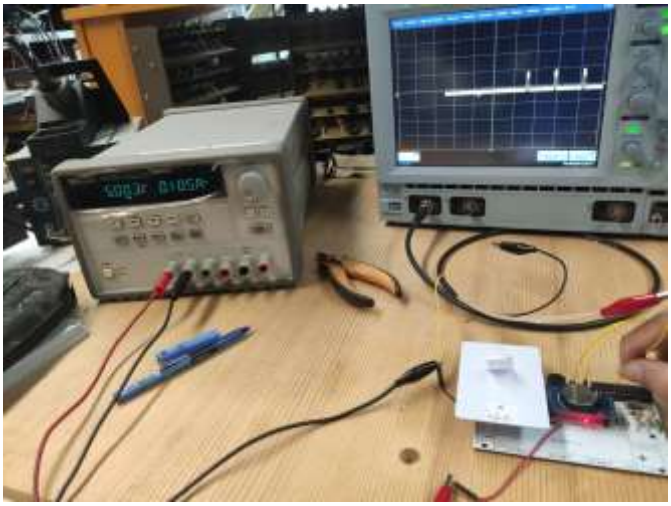


Ilustración 71: Micro en funcionamiento tras detectar una tercera tarjeta. Fuente: Propia.

Finalmente, se procede a calcular su autonomía. Como se mencionó anteriormente, se dispone a que el circuito esté 3 segundos activado y 4 en Sleep Mode, lo que implica que su ciclo es de 7 segundos. Primero, se calcula cuánto consume en los 3 segundos que está activo y, en segundo lugar, los 4 segundos que está en el Sleep Mode. Para concluir, se hace la suma de dicho consumo y se calcula la autonomía final.

$$Consumo\ 3\ segundos\ Activo = 0.105\ A * 3s = \ 0.315\ As$$

$$Consumo\ 4\ segundos\ Sleep\ Mode = 0.09\ A * 4s = \ 0.360\ As$$

$$Consumo\ Total = \frac{C.Activo + C.Sleep}{Tiempo\ ciclo} = \frac{0.315\ A\ s\ + 0.360\ A\ s}{7\ s} = \ 0.096\ A$$

$$Autonomía\ batería = \frac{Capacidad\ corriente\ batería\ (mAh)}{Consumo\ Total\ (mA)} = \frac{2600\ mAh}{96\ mA} = 27h\ (aprox.)$$

5.5. PRESUPUESTO

En la Ilustración 72 se muestra el precio por componente y el precio total del prototipo v1. Mientras, en la Ilustración 73, se observa el precio unitario de la tarjeta utilizada.

	PRODUCTO	PRECIO (€)
	ATMEGA328P	6,5289
	Soporte ATMEGA328P	0,17
	Escudo batería con regulador a 3V y 5V	1,44
	Batería de 3.7 V con capacidad de corriente de 2600 mAh	3,5
	Zumbador activo a 5V.	0,86
	PN532 NFC módulo V3	3,55
	DS3231 MÓDULO RTC	2,25
	Pila CR2032	1,05
	Diodo CD1206-S01575	0,047
	Resonador COM-09420	1,25
	Resistencia de 1 MΩ	0,1
	Resistencia de 10 KΩ	0,1
	Resistencia de 1 KΩ	0,1
	Bobina de 10µH	0,2925
	4 condensadores de 100 nF	0,3
	2 condensadores de 22 pF	0,2
	PCB	0,8
	Caja protectora	8
	Otros materiales	1
Precio total		31,5384

Ilustración 72: Precio total del prototipo v1. Fuente: Propia.

PRODUCTO	PRECIO (€)
Tarjeta NFC MIFARE Classic EV1 1K	0,41

Ilustración 73: Precio tarjeta NFC MIFARE Classic EV1 1k. Fuente: Propia.

5.6. CONCLUSIÓN PROTOTIPO V1

Tras finalizar los ensayos con este prototipo, se ha logrado su perfecto funcionamiento a pesar de los contratiempos producidos. Se ha constatado un prototipo muy sólido en cuanto a las características de hardware y de software. En cuanto a software, se ha podido implementar una mejora con el Sleep Mode, cosa que ha hecho que se consiga un mejor consumo. También, se ha mejorado y se ha podido ejecutar sin problemas en Modo Verificar tarjeta.

En cuanto al hardware, el haber suprimido los elementos innecesarios, ha conllevado una reducción del prototipo de Laaroussi, S. Esta reducción también se ha visto reflejada en el coste de los materiales, pasado de un prototipo de 51.62 € a 31.54 €. El consumo y la autonomía se han visto reflejados con una mejora bastante elevada, ya que instaurando los valores de consumo obtenidos de la medición del prototipo de Laaroussi, S. con su consumo real en pausa y en funcionamiento (Ilustraciones 41 y 42) a un ciclo de 7 segundos, se obtiene el siguiente consumo y la siguiente autonomía:

$$Consumo\ 3\ segundos\ Activo = 0.166\ A * 3s = \ 0.498\ As$$
$$Consumo\ 4\ segundos\ Sleep\ Mode = 0.118\ A * 4s = \ 0.472\ As$$
$$Consumo\ Total = \frac{C.Activo + C.Sleep}{Tiempo\ ciclo} = \frac{0.498\ As\ + 0.472\ As}{7\ s} = \ 0.139\ A$$

$$Autonomía\ batería = \frac{Capacidad\ corriente\ batería\ (mAh)}{Consumo\ Total\ (mA)} = \frac{650\ mAh}{139\ mA} = 5\ h\ (aprox.)$$

De esta manera queda reflejado que el consumo pasa de 0.139 A de media a 0.096 A. mientras, su autonmía cambia de, aproximadamente, 5 horas a 27 horas.

En cuanto a la memoria, se ha optimizado en gran parte debido a que, Ilustración 40, se ha pasado de utilizar solamente un 5% de su memoria Flash y un 32% de la memoria SRAM a un 37% de la Flash y un 77% de la SRAM (Ilustración 53).

A pesar de cumplir estas mejoras que se habían citado en el punto 4.8. CONCLUSIÓN DEL PROTOTIPO, han surgido nuevos problemas que no se han podido mejorar:

- La PCB no entra en la caja. Debido a una mala medición por parte del fabricante, el interior de la caja no hace las medidas proporcionadas por él. Este hecho hace que la PCB no se pueda colocar en ella.
- Una nueva configuración de la RTC. A pesar de haber fraccionado en 4 programas los modos, la RTC sigue sin avanzar su hora cuando pierde la tensión, esto conlleva a errores cuando se usa el Modo Escritura.
- Monitor serie. Mediante el nuevo software, se ha conseguido que el sistema entre en el Sleep Mode y reduzca su consumo. A pesar de ello, se sigue necesitando el monitor serie para Asignar el ID_Base del Modo Escritura y la configuración del Sleep Mode en cada modo. Cabe recalcar que al no haber ninguna conexión con el puerto serie entre el sistema y el monitor serie, el Modo Lectura se vuelve inservible debido a que no se puede leer por ningún medio los valores que lee el programa.

5.7. MEJORAS PARA UN PROTOTIPO V2

Para los problemas mencionados anteriormente, se ha procedido a diseñar un prototipo v2 que pueda garantizar que se cumplen todos los puntos de los apartados 4.8. CONCLUSIÓN DEL PROTOTIPO y 5.4. CONCLUSIÓN PROTOTIPO V1. Para ello se va a proceder a lo siguiente:

- La PCB no entra en la caja. Implementar un nuevo diseño que quepa en la caja de protección.
- Una nueva configuración de la RTC. Averiguar el problema de por qué la RTC no avanza en el tiempo cuando esta pierde la tensión.
- Monitor serie. En el nuevo diseño, implementar unos pines mediante los cuales se pueda ver por el monitor serie qué hace el programa en cada instante para así poder ver los resultados del Modo Lectura. También se procederá a una reestructuración del código de manera que no se tenga que pedir por el monitor serie que se escriba ningún dato.
- Finalmente, también se buscará que el consumo y el precio se no alejen de los valores obtenidos.

6. PROTOTIPO V2

6.1. MEJORAS SOFTWARE

6.1.1. NUEVA CONFIGURACIÓN RTC

Se decidió dejar el sistema sin utilizar una semana y, sin volver a cargar un programa, utilizar el que había (Modo Escritura). Cuando se escribió en la tarjeta el dato y se leyó a través del Modo Lectura, se pudo analizar que la fecha que esta contenía era la que tenía la semana anterior. Tras ver esto, se pensó que tal vez no había grabado nada, aunque el zumbador actuó. Entonces, se decidió utilizar el Modo Borrar y, de esta manera, asegurarse de que el dato que haya en la memoria habrá sido escrito en ese momento. Tras la supresión de los datos y la nueva escritura, se pudo confirmar que la hora no había variado y que era siempre la misma que había antes de que el sistema perdiera la conexión. Esto significa que cada vez que el sistema deja de ser alimentado, la RTC no avanza con el tiempo.

Tras indagar en el código, se observó que en el software del prototipo v1, siempre fuera cual fuera el programa, se inicializaba la RTC, cosa que era totalmente innecesaria para los programas de lectura, borrar y verificar. Por este hecho, se decidió eliminar del código la Lectura, Borrar y Verificar, aunque en estos programas se concluyó mostrar por pantalla la hora de la RTC. Después de esto, se volvió a ejecutar el programa de escritura con algún retoque, pero sin éxito alguno, aunque cuando se usaba otro modo, la lectura que se mostraba por pantalla era correcta pese a no haber inicializado la RTC.

Todo esto llevó a la teoría de que, podía ser, que como el programa configura la hora siempre, cuando se deja de alimentar la RTC esta deja de avanzar con el tiempo debido a que está constantemente se está cogiendo la hora del ordenador para inicializarse. Para ver si esta teoría era correcta, se decidió hacer un programa nuevo donde solamente se inicializa la RTC con la hora del PC (Ilustración 74) .

```
void setup () {

  Serial.begin(115200);
  rtc.begin();
  delay(3000); // wait for console opening

  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }

  if (rtc.lostPower()) {
    Serial.println("Estableciendo Hora y fecha...");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    Serial.println("RTC DS3231 actualizado con la hora y fecha que se compilo este programa:");
    Serial.print("Fecha = ");
    Serial.print(__DATE__);
    Serial.print("  Hora = ");
    Serial.println(__TIME__);
  }
}
```

Ilustración 74: Código para inicializar la RTC. Fuente: Propia.

Una vez realizado esto, se ha procedido a eliminar del Modo Escritura la parte del código que configura la RTC. Entonces, se ha inicializado el programa y se ha comprobado que

pese a perder la tensión, la RTC sí que avanza con el tiempo.

De esta manera, se ha podido solucionar el aspecto de que la RTC no avanzaba con el tiempo, debido a que siempre se le estaba inicializando y, cuando esto era imposible porque no se acaba de cargar el programa del ordenador, seguía con la hora de la última carga del ordenador.

6.1.2. MONITOR SERIE

Como se mencionó en el apartado 5.5. MEJORAS PARA UN PROTOTIPO V2, el monitor serie era necesario para establecer las bases en el Modo Escritura y para asignar el tiempo de Sleep Mode. Para solucionar este problema y que no tener que poner esos valores a través del monitor serie, se ha decido implementar una mejora en los cuatro programas. Para ello, se ha decidido eliminar sus configuraciones y, mediante una explicación en texto comentado, explicar cuáles son los valores que se deben poner. De esta manera el programa funciona de la misma manera, pero no es necesaria una conexión con el monitor serie para la introducción de datos, solamente haría falta para visualizarlos.

```
//Mediante el valor de IDNum, asignamos el valor de la base
//que queremos que sea. Este valor de ser entre 0 y 44.
byte IDNum = 14;
```

Ilustración 75: Nueva forma de configurar el ID_Base (Modo Escritura). Fuente: Propia.

```
//Mediante el SLEEP_4S, coinfiguramos el tiempo de Sleep Mode,
// este tiempo puede ser de SLEEP_1S para 1 segundo,
//SLEEP_2S para 2 segundos, SLEEP_4S para 4 segundos y
//SLEEP_8 para 8 seegundos.
LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF);
```

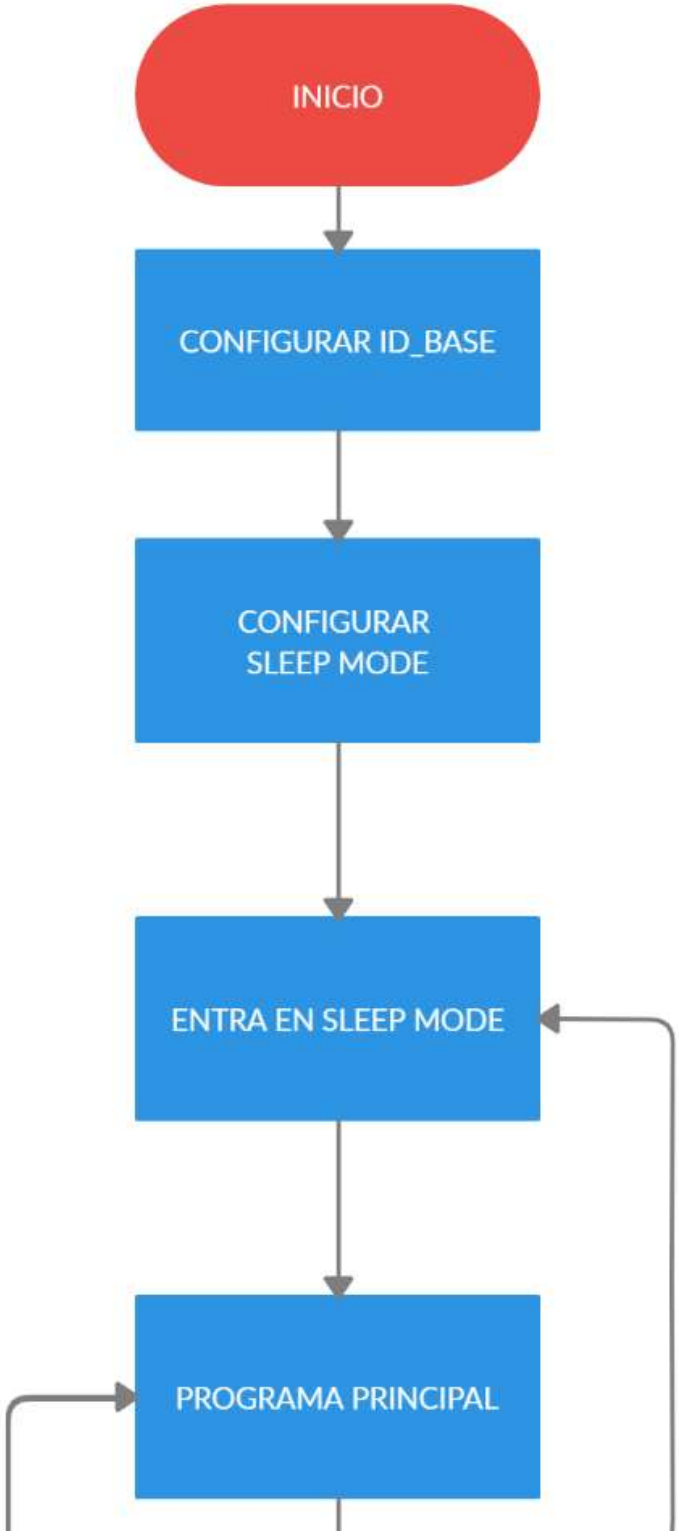
Ilustración 76: Nueva forma de configurar el Sleep Mode de todos los modos. Fuente: Propia.

En caso de que no se introduzcan los valores adecuados, el programa no funcionará de manera correcta, para ello, hay que ir con cuidado introduciendo los valores.

6.1.3. FLUJOGRAMAS PROTOTIPO V2

Los flujogramas del prototipo v2, no distan mucho del prototipo v1, debido a que los únicos cambios han sido la supresión de asignar el ID_Base y el tiempo de Sleep Mode por el monitor serie y que cada vez que se inicie un programa, se configure la RTC. De este modo se configura, en el caso del Modo Escritura primero se configura la ID_Base, el Sleep Mode y duerme el sistema. Entonces se inicia el bucle principal (Programa Principal) y se mira si han pasado 3 segundos desde el último Sleep Mode para, en caso afirmativo volver al Sleep Mode. De no ser así, se avanza en el programa a la detección de tarjeta. Si esta tarjeta ha sido detectada y no han pasado 3 segundo del último Sleep Mode, se avanza a autentificar la tarjeta y, en caso de ser autentificada, se ejecuta la acción principal del modo. Cuando esta finaliza, el zumbador se activa y se vuelve al principio del bucle, a programa principal.

6.1.3.1. MODO ESCRIBIR



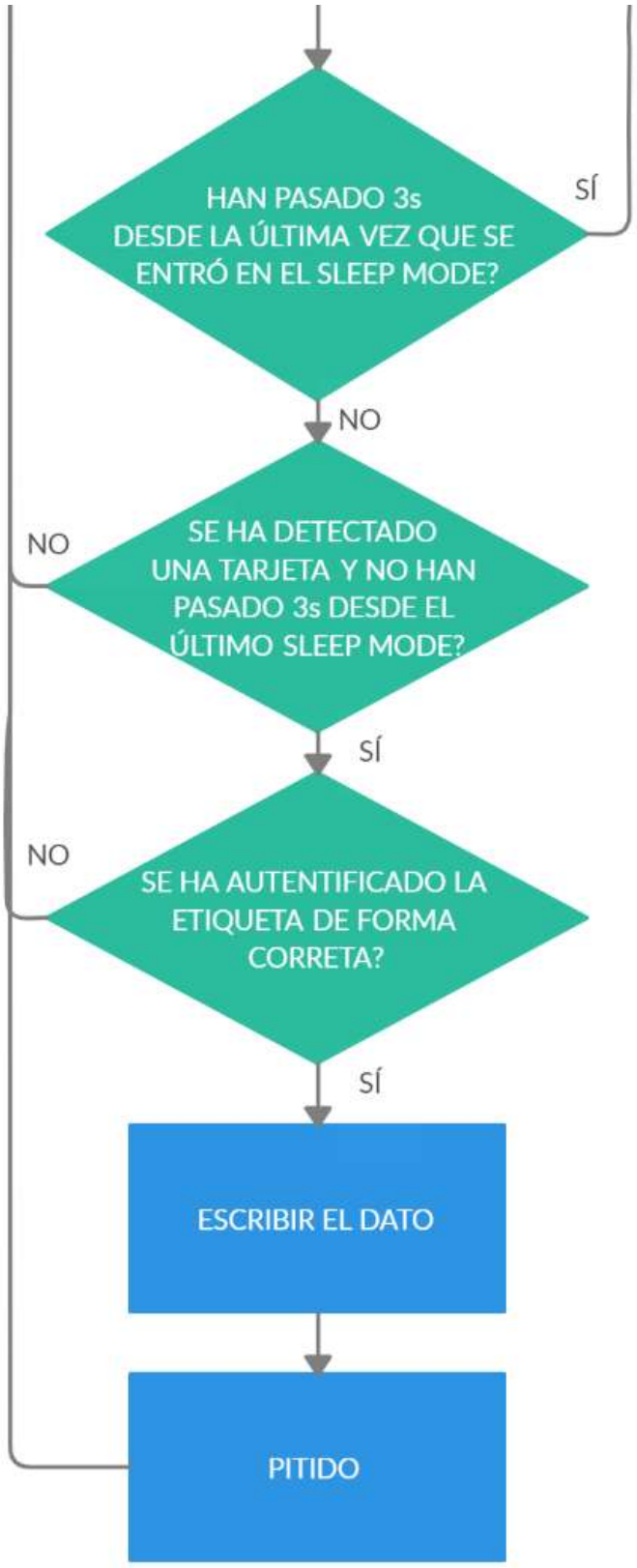
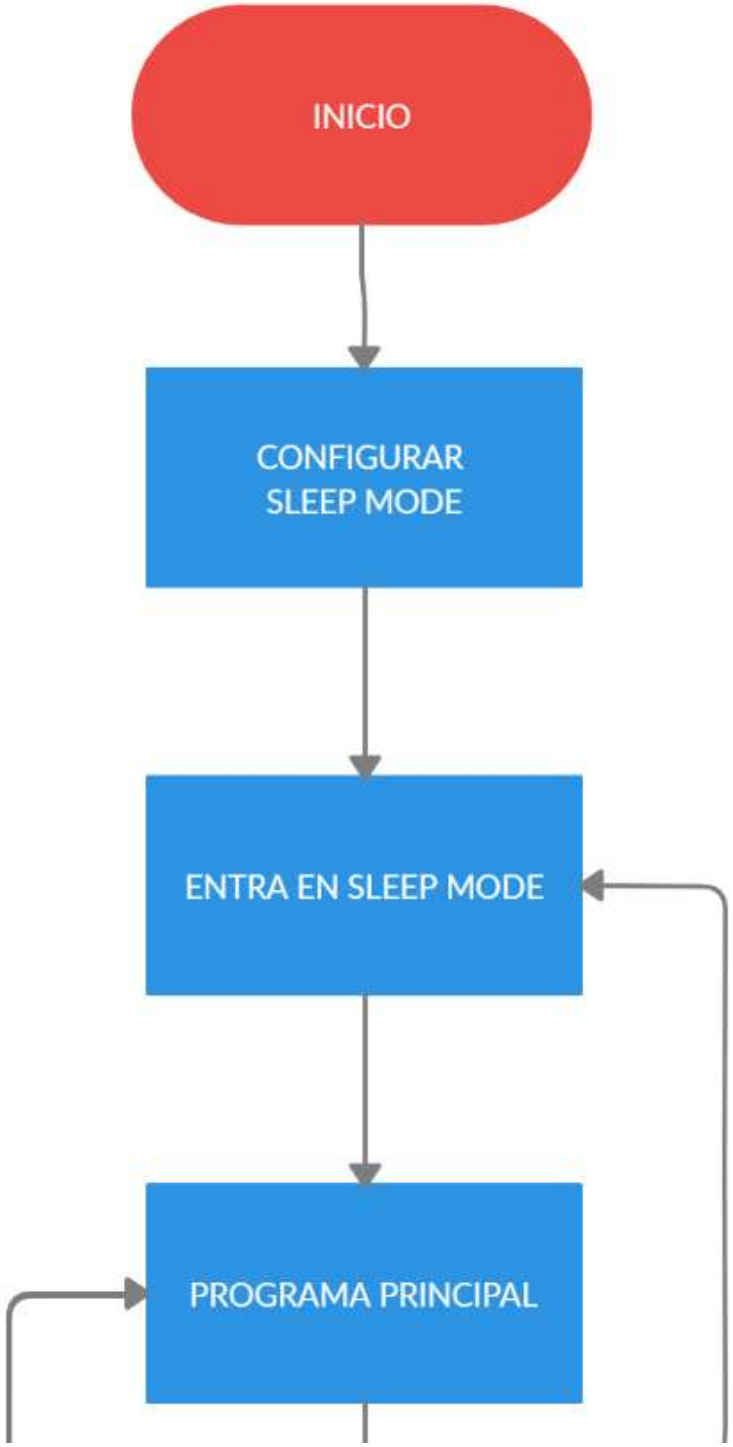


Ilustración 77: Flujograma Modo Escritura prototipo v2. Fuente: Propia.

6.1.3.2. MODO LECTURA



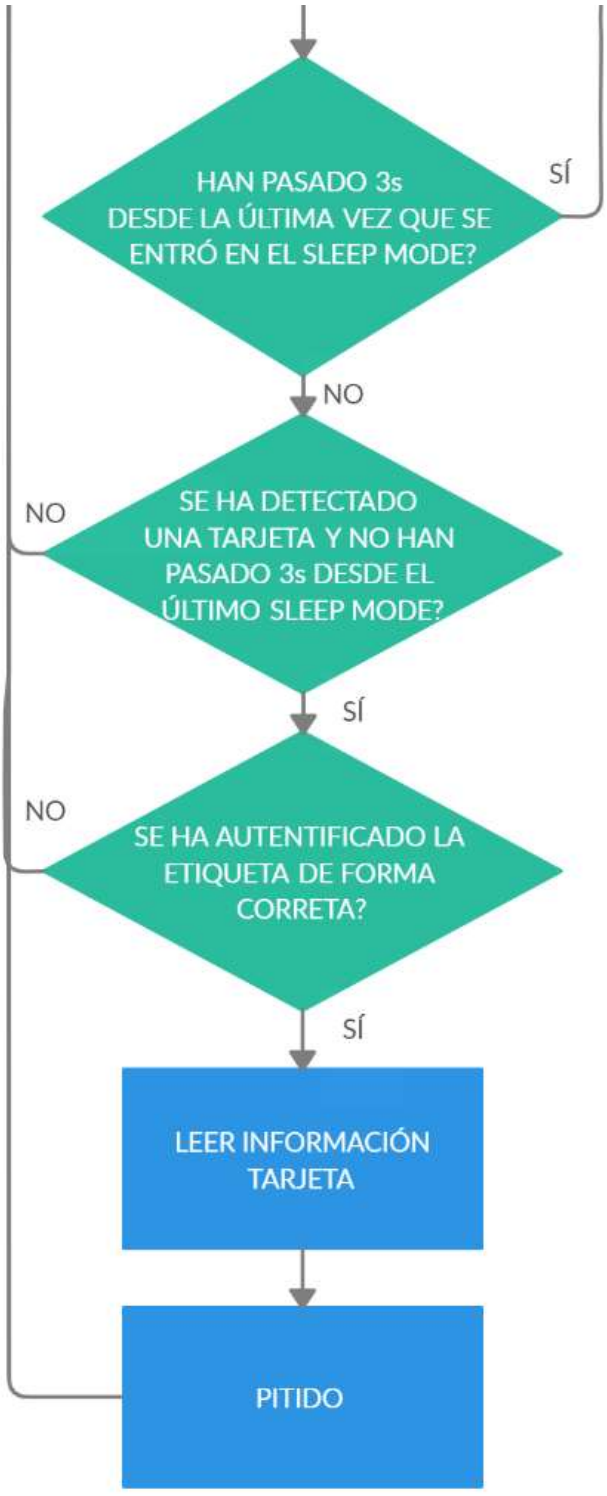
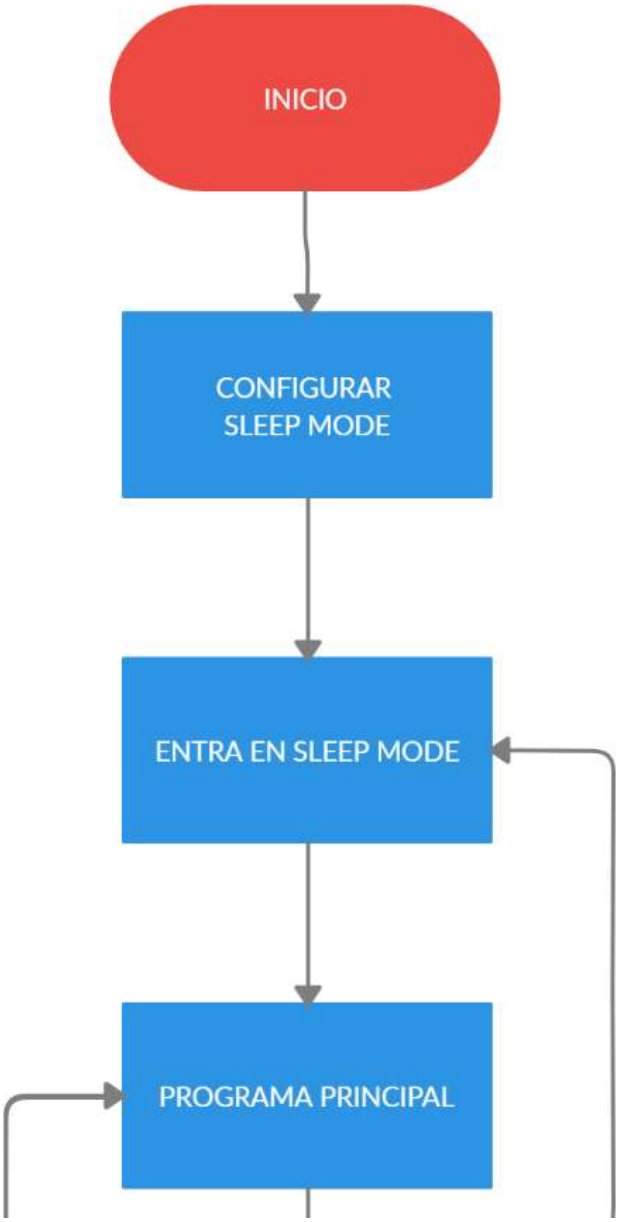


Ilustración 78: Flujograma Modo Lectura prototipo v2. Fuente: Propia.

6.1.3.3. MODO BORRAR



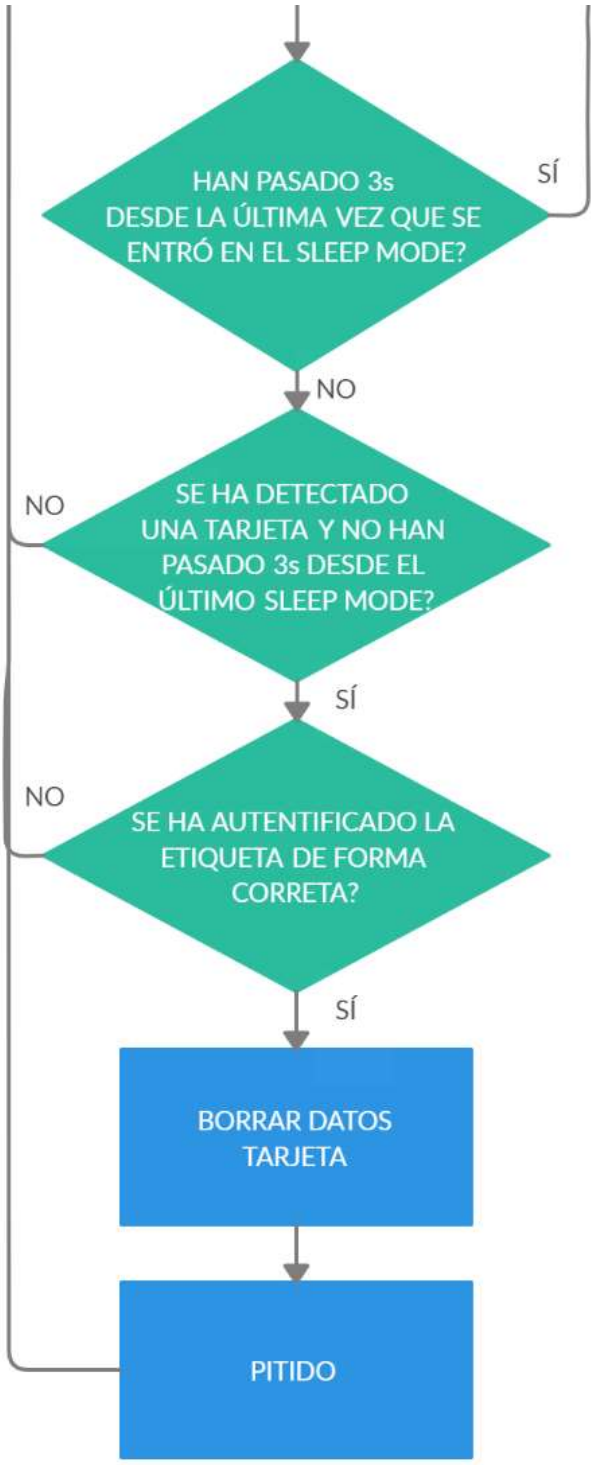
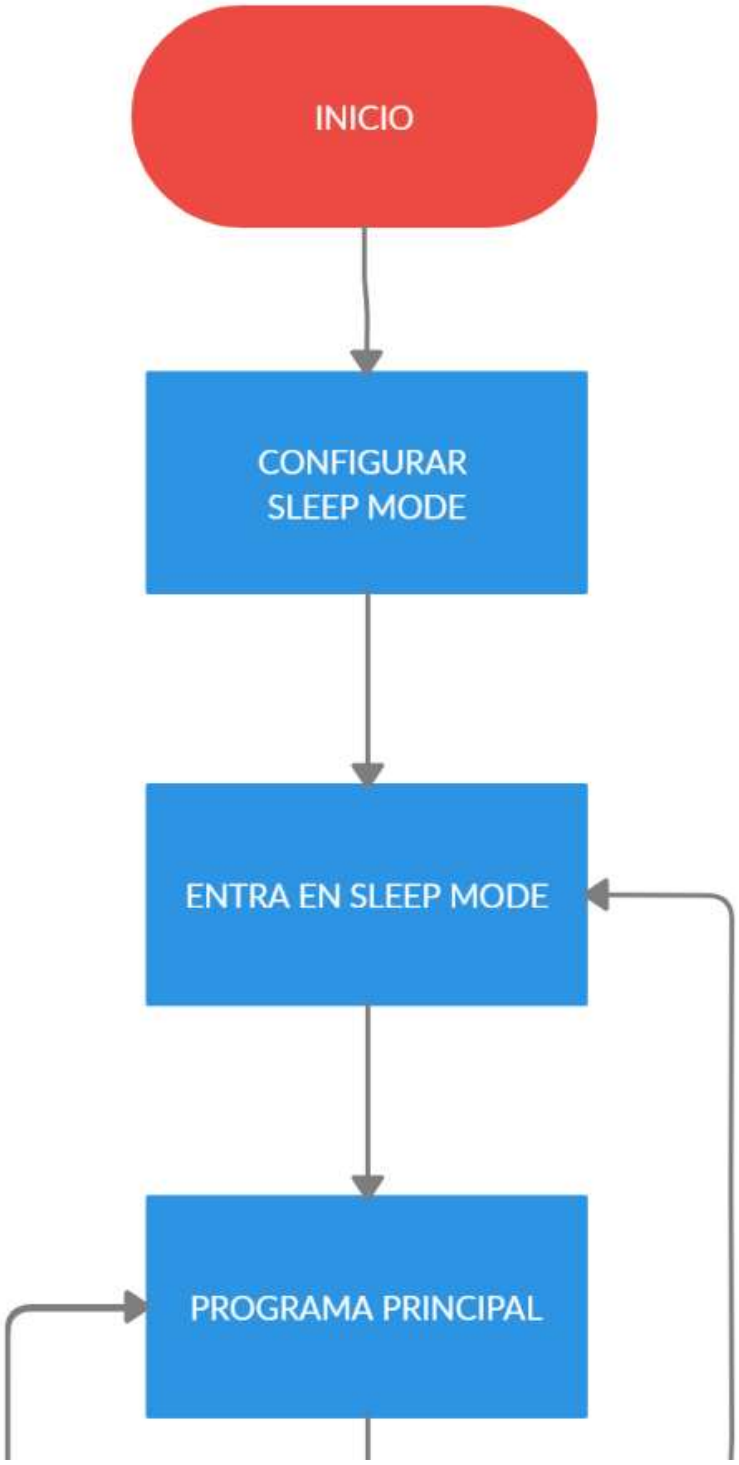


Ilustración 79: Flujograma Modo Borrar prototipo v2. Fuente: Propia.

6.1.3.4. MODO VERIFICAR TARJETA VACÍA



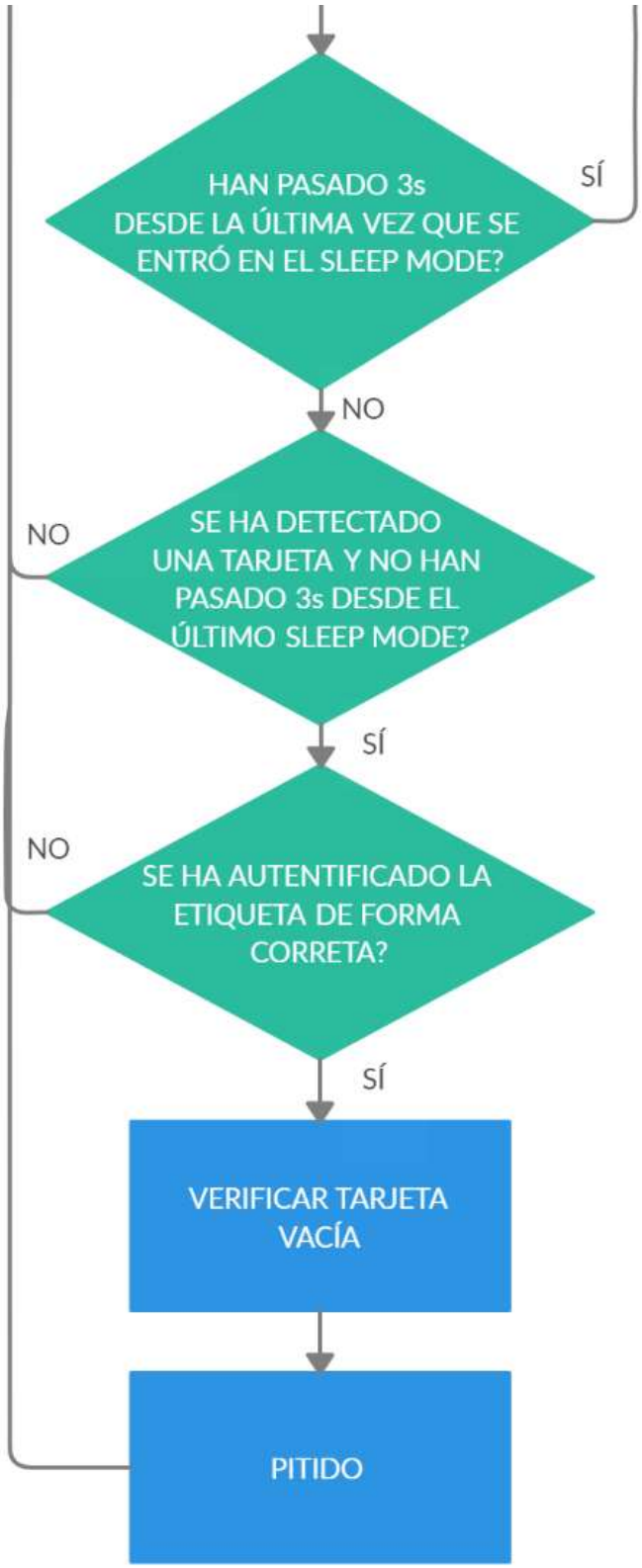


Ilustración 80: Flujograma Modo Verificar tarjeta vacía prototipo v2. Fuente: Propia.

6.2. MEJORA HARDWARE

Como mejoras del hardware, se han introducido varios cambios que se comentarán en los siguientes apartados del proyecto. Se han cambiado las dimensiones de la placa

para que esta pueda caber dentro de la caja protectora y se ha añadido una salida para poder comunicar con el puerto serie y poder ver la información de las tarjetas en cada uno de sus modos.

6.2.1. NUEVAS CONEXIONES HARDWARE

Para poder monitorizar el sistema y ver qué está haciendo en cada momento, se ha decido introducir 3 pines nuevos que vayan del microcontrolador al ordenador. Para que se pueda usar el monitor serie, los pines seleccionados son el RXD y el TXD, siendo estos los pines 2 y 3 del microcontrolador. De esta manera, nuestras conexiones van a ser las siguientes:

- Pin 2 del ATMEGA328P: se conectará el pin RXD creado para poder monitorizar el sistema.
- Pin 3 del ATMEGA328P: se conectará el pin TXD creado para poder monitorizar el sistema.
- Pin 4 del ATMEGA328P: se conectará el pin IRQ del PN532 encargado de la rutina de interrupción 0.
- Pin 5 del ATMEGA328P: se conectará el pin RSTO del PN532 encargado de la rutina de interrupción 0.
- Pin 6 del ATMEGA328P: se conectará el positivo del zumbador mediante una resistencia de 1 KΩ.
- Pin 27 del ATMEGA328P: se conectará el pin SDA del PN532 y del DS3231, encargado de la comunicación I2P.
- Pin 28 del ATMEGA328P: se conectará el pin SCL del PN532 y del DS3231, encargado de la comunicación I2P.
- DS3231: los pines utilizados restantes, VCC y GND, irán conectados al escudo de la batería.
- PN532: los pines utilizados restantes, VCC y GND, irán conectados al escudo de la batería.

Por consiguiente, el esquema de conexiones del prototipo v2 será el mostrado en la Ilustración 81. Este esquema será totalmente idéntico al del prototipo v1 a excepción de la resistencia del zumbador que ha sido suprimida, el resonador ha sido cambiado por un cristal de 16 MHz más económico, se han añadido los dos condensadores del cristal y, por último, se ha añadido las salidas para el monitor serie.

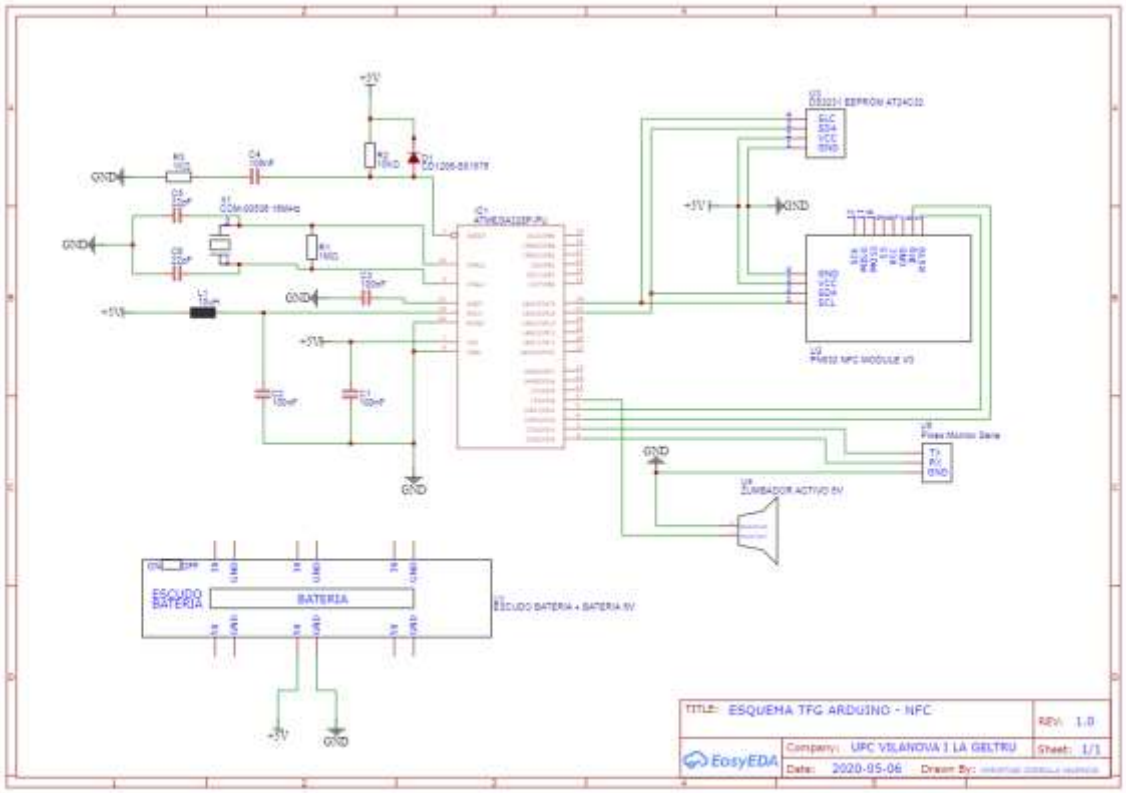


Ilustración 81: Esquema conexiones prototipo v2. Fuente: Propia.

6.2.2. NUEVA PCB

Para llevar a cabo las modificaciones que se han hecho del prototipo v1, se ha creado una nueva PCB (Ilustraciones 82 y 83). Esta, a parte de los cambios mencionados en el apartado anterior, también ha sido modificada de tamaño porque uno de los problemas que se extrajo de las conclusiones del prototipo v1 es que, por fallo en la medición de la caja, la PCB no cabe en ella. Esto es debido a que los datos proporcionados por el fabricante eran erróneos. Para solucionar ese problema, se ha pasado de una PCB de 110x70 mm (Ilustración 84) a una de 100x78 mm (Ilustración 85). Esto es debido a que el problema principal que tiene la PCB del prototipo v1 es que es demasiado largar y se topa con el sitio al que van los tornillos del cierre de la tapa. Para solucionar esto, se ha establecido un diseño en él se diseñan los bordes para que estos no choquen. También se ha hecho más ancha para que de esta manera se aproveche el ancho de la placa.

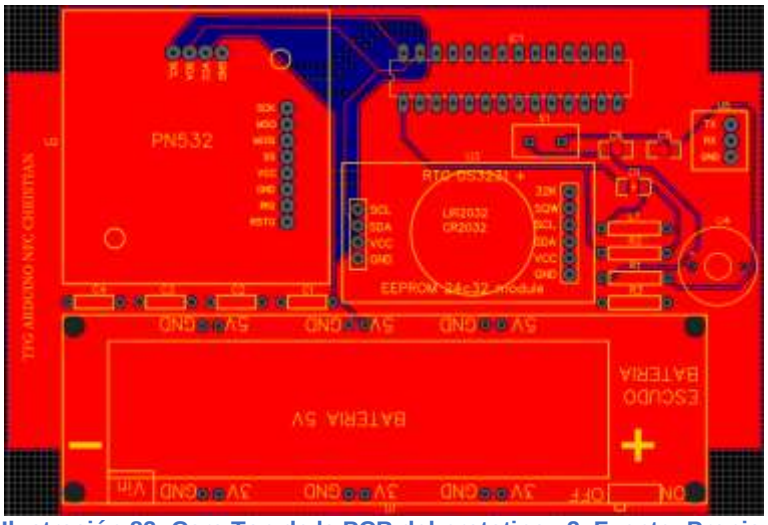


Ilustración 82: Cara Top de la PCB del prototipo v2. Fuente: Propia.

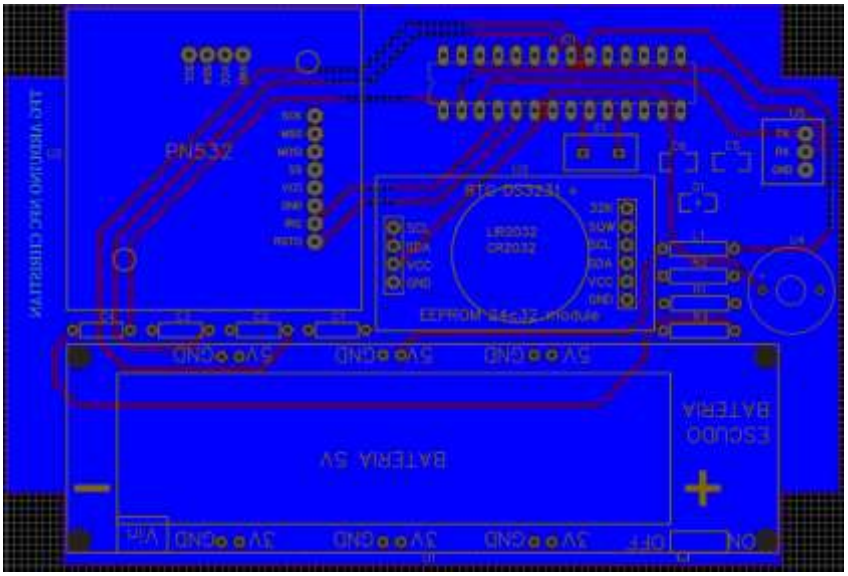


Ilustración 83: Cara Bottom de la PCB del prototipo v2. Fuente: Propia.

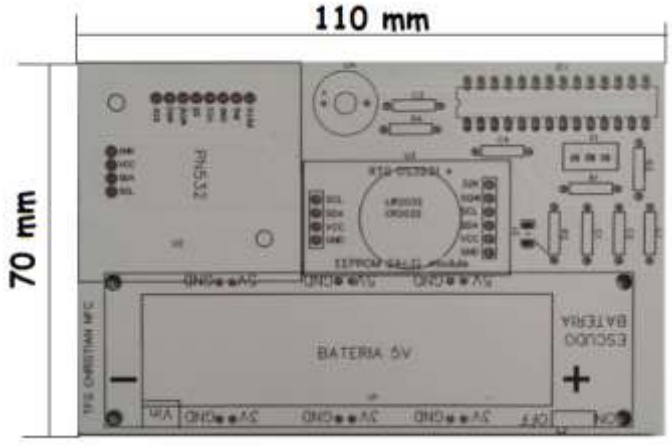


Ilustración 84: PCB prototipo v1 con sus medidas. Fuente: Propia.

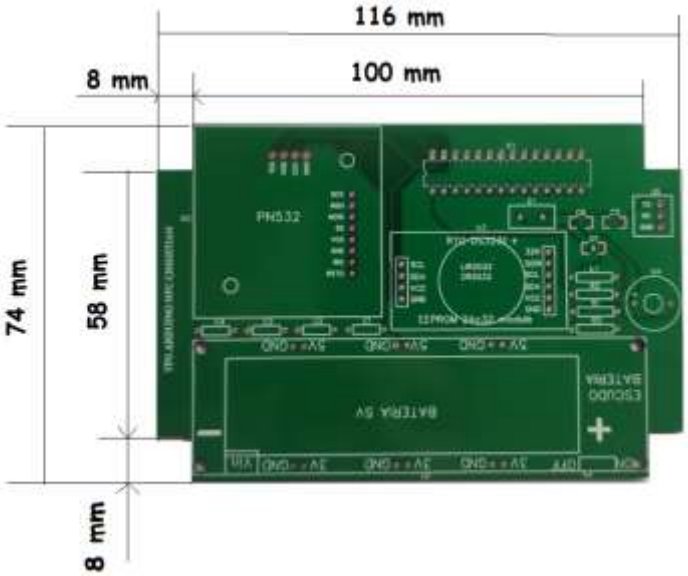


Ilustración 85: PCB prototipo v1 con sus medidas. Fuente: Propia.

6.3. LISTA COMPONENTES A CONECTAR EN LA PCB

Todos los elementos que componen el sistema del prototipo v1, son los siguiente:

- Microcontrolador ATMEGA328P.
- Soporte ATMEGA328P.
- Escudo batería con regulador a 3 V y 5 V.
- Batería de 3.7 V con capacidad de corriente de 2600 mAh.
- Zumbador activo a 5V.
- PN532 NFC Módulo V3.
- DS3231 MÓDULO RTC.
- Pila CR2032.
- Diodo CD1206-S01575.
- Resonador COM-09420.
- Resistencias de: 1 MΩ, 10 KΩ y dos de 1 KΩ.
- Bobina de 10µH.
- 4 condensadores de 100 nF.
- 2 condensadores de 22 pF.

6.4. CONSUMO Y AUTONOMÍA

Tras probar y contrastar la fiabilidad del sistema, se han obtenido los datos de consumo y autonomía. Debido a la similitud de características y de componentes que tiene la placa, el consumo y la autonomía son los mismos que los del prototipo v1. Obteniendo los siguientes valores:

$$Consumo\ 3\ segundos\ Activo = 0.105\ A * 3s = \ 0.315\ As$$

$$Consumo\ 4\ segundos\ Sleep\ Mode = 0.09\ A * 4s = \ 0.360\ As$$

$$Consumo\ Total = \frac{C.Activo + C.Sleep}{Tiempo\ ciclo} = \frac{0.315\ As\ + 0.360\ As}{7} = 0.096\ A$$

$$Autonomía\ batería = \frac{Capacidad\ corriente\ batería\ (mAh)}{Consumo\ Total\ (mA)} = \frac{2600\ mAh}{96\ mA} = 27h\ (aprox.)$$

6.5. PRESUPUESTO

El presupuesto obtenido para este prototipo v2 es casi idéntico al del v1. Esto se debe a que utiliza los mismos componentes a excepción del cambio del cristal. Este cambio conlleva a una mínima diferencia de precio (Ilustración 86) en el que el v2 es más barato a consecuencia de un nuevo proveedor que vende el ATMEGA328P junto con el cristal. El precio de la tarjeta utilizada por unidad (Ilustración 87) sigue siendo el mismo.

	PRODUCTO	PRECIO (€)
	ATMEGA328P + Cristal	6,5289
	Soporte ATMEGA328P	0,17
	Escudo batería con regulador a 3V y 5V	1,44
	Batería de 3.7 V con capacidad de corriente de 2600 mAh	3,5
	Zumbador activo a 5V.	0,86
	PN532 NFC módulo V3	3,55
	DS3231 MÓDULO RTC	2,25
	Pila CR2032	1,05
	Diodo CD1206-S01575	0,047
	Resistencia de 1 MΩ	0,1
	Resistencia de 10 KΩ	0,1
	Resistencia de 1 KΩ	0,1
	Bobina de 10µH	0,2925
	4 condensadores de 100 nF	0,3
	2 condensadores de 22 pF	0,2
	PCB	0,8
	Caja protectora	8
	Otros materiales	1
Precio total		30,2884

Ilustración 86: Presupuesto total del prototipo v2. Fuente: Propia.

PRODUCTO	PRECIO (€)
Tarjeta NFC MIFARE Classic EV1 1K	0,41

Ilustración 87: Precio tarjeta NFC MIFARE Classic EV1 1k. Fuente: Propia.

6.6. CONCLUSIÓN PROTOTIPO V2

Tras finalizar todas las pruebas con este prototipo, se puede asegurar su fiabilidad en todos los modos de funcionamiento. Se ha conseguido mejorar el prototipo v1, arreglando los problemas que este presentaba tanto de software como de hardware. Los problemas de software se han solucionado agregando un programa solamente para la inicialización de la RTC, para así garantizar su perfecto funcionamiento. También se ha cambiado la manera de introducir los valores del ID_Base y del Sleep Mode,

convirtiéndolo en una manera más sencilla y que no reporta problemas.

En cuanto al hardware, se ha diseñado una PCB que encaja a la perfección en la caja mientras que su predecesora no entraba. También, se ha añadido un espacio para los condensadores del cristal, se ha suprimido la resistencia que impedía funcionar al zumbador y se han añadido 3 nuevos pines para poder monitorizar, en caso de que eso sea requerido, los modos de funcionamiento.

Estas medidas, no han hecho cambiar el consumo de la batería de un modelo respecto del otro. Esto se debe a que las medidas no han sido de gran influencia a la hora del este, sino más bien en la distribución de los componentes, teniendo un consumo medio por segundo de 0.096 A y una autonomía de 27 horas.

7. APLICACIÓN ANDROID

Actualmente, como se mencionó en los primeros puntos del proyecto, la tecnología NFC está en auge y, como toda tecnología, esta se exprime para poder sacar su mejor versión. De este modo, en el mercado existen muchos tipos de aplicación, por ejemplo, NFC Tools, para la lectura y escritura de tarjetas NFC. Además, una aplicación puede ser también otro modo de marketing del que se pueden obtener beneficios. En este proyecto se dispone a la creación de una app que se capaz de leer los datos que se escriben en el Modo Escritura.

Para esta programación, se utiliza el lenguaje por excelencia de las aplicaciones Android, Java. Este lenguaje dentro del programa Android Studio, permite que se puedan crear aplicaciones Android gracias a su versatilidad y su fácil interfaz.

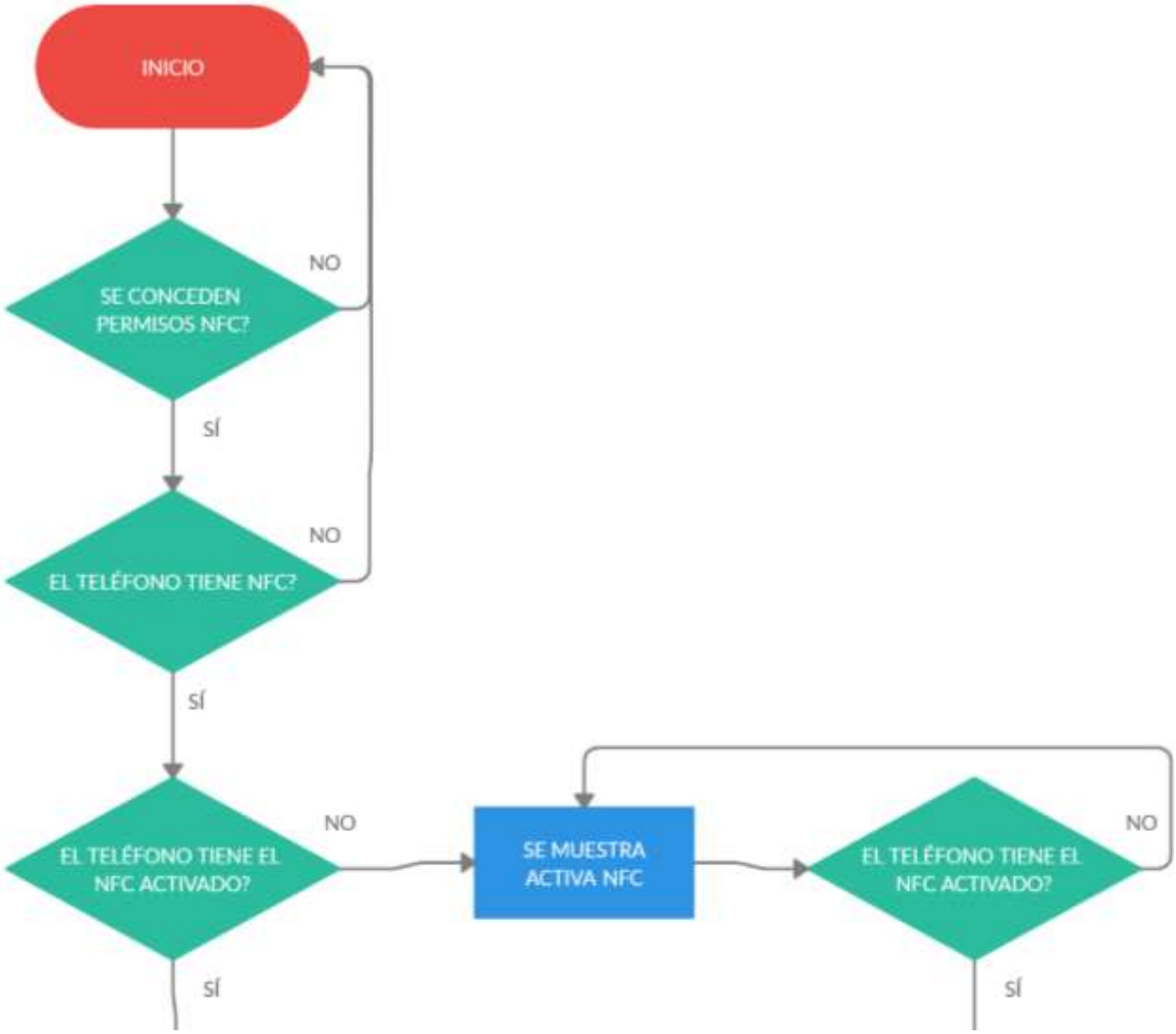
7.1. FUNCIONAMIENTO

En primer lugar, se va a pedir el acceso al sistema NFC del teléfono para poder empezar a ejecutar la aplicación. Una vez permitido, se iniciará la app y se mostrará por pantalla, en caso de que no esté activa, que se active el NFC. Cuando este lo esté, ya podremos pasar la tarjeta por el lector NFC del móvil. Si se mira la interfaz, se puede ver que arriba hay un recuadro llamado Nombre, donde se debe introducir el nombre de la persona o dispositivo. Si se lee una etiqueta y no tiene nombre, la aplicación interactuará diciendo que se añada uno (Ilustración 88).

Una vez pasada la tarjeta, el móvil hará un ruido conforme se ha leído la tarjeta y se mostrará por pantalla que se ha leído y su contenido, ya sea textos, webs, etc.

Ilustración 88: Mensaje de la aplicación cuando no se introduce un nombre. Fuente: Propia.

7.2. FLUJOGRAMA APLICACIÓN



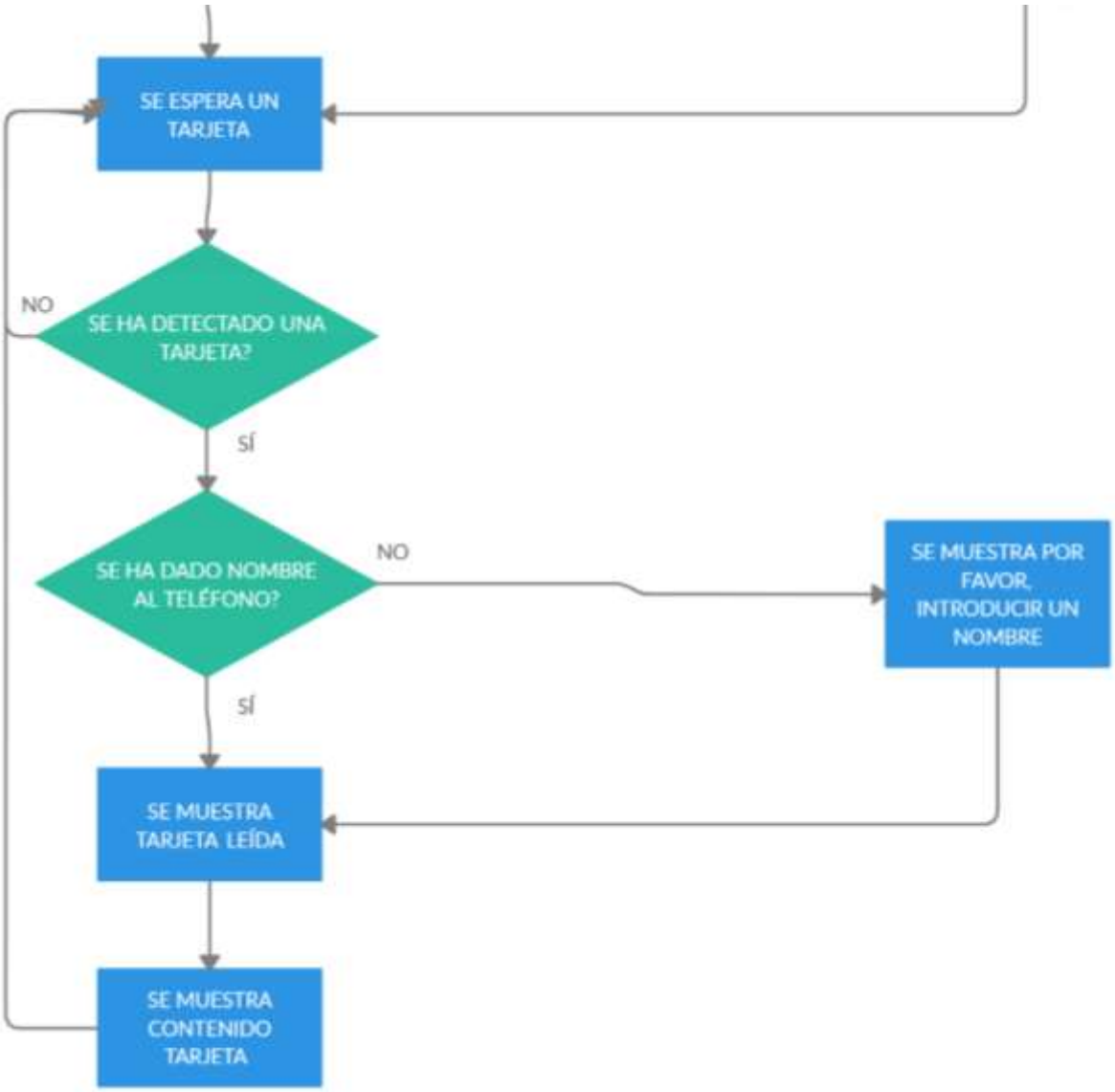


Ilustración 89: Flujograma aplicación. Fuente: Propia.

7.3. PROBLEMÁTICA

Tras probar el funcionamiento de la aplicación, se ha dado con 2 problemas que hacen que no tenga un correcto funcionamiento.

7.3.1. NO MUESTRA EL CONTENIDO GRABADO POR ARDUINO

Cuando se graba una tarjeta con los tiempos de tramo mediante el Modo Escritura, cuando se pasa por el Modo Lectura, se imprimen por pantalla todos los datos que esta contiene. En cambio, cuando esta es pasada por el lector NFC del móvil, se muestra que se ha podido leer la tarjeta, pero no aparece el contenido en ella (Ilustración 90). En cambio, en la Ilustración 91 se puede observar que, si la tarjeta tiene, por ejemplo, un dato escrito mediante la aplicación NFC Tools, sí que se muestra el contenido, una URL en este caso.

Ilustración 90: Aplicación mostrando la tarjeta grabada por el sistema. Fuente: Propia.

Ilustración 91: Aplicación mostrando la tarjeta grabada por NFC Tools. Fuente: Propia

Se puede ver que, en la segunda ilustración, se muestra el contenido cortado ya que le falta parte del texto original (www.epsevg.upc.edu). En un primer caso, se pensó que era porque el texto no estaba centrado, pero tras varias comprobaciones y modificaciones, siempre se mostraba el mismo resultado. Este error puede venir de una mala programación a la hora de la lectura de la tarjeta. Es posible que el puntero con el que se lee a nivel, necesite algún dato más para poder acceder bien a la memoria o que la aplicación NFC Tools, para grabar use algún puntero a nivel bajo que se desconoce. Esta conclusión gana validez cuando se lee por Arduino el valor de la tarjeta grabada por la aplicación mencionada, ya que según se puede ver en la Ilustración 92, el dato es mostrado con unos símbolos que no han sido establecidos a la hora de escribir, números que en ASCII sean los correspondientes al puntero de la dirección de la memoria.

```
El deportista paso por la base número  0 y se guardo el siguiente dato:
Etiqueta NFC autentificada
El deportista paso por la base número  1 y se guardo el siguiente dato: upc.edu?
Etiqueta NFC autentificada
```

Ilustración 92: Tarjeta escrita por NFC Tools leída por el Modo Lectura. Fuente: Propia.

Como se observa, no ha guardado nada en el primer rango de la memoria y en el segundo solamente la mitad de la palabra más un símbolo. Esto puede ser debido a que el sistema de lectura y de grabado NFC con Android sea diferente al de Arduino.

7.4. CONCLUSIÓN APLICACIÓN NFC

Mediante los conocimientos adquiridos durante la programación del modo de lectura, se ha implementado una aplicación que hace lo mismo, pero para el teléfono móvil. Tras acabar la programación y ejecutarla se ha observado que tiene dos problemas a resolver:

- No muestra el contenido de tarjetas escritas por Arduino. Tras varias comprobaciones y pruebas, se ha verificado que la aplicación no es capaz de mostrar ningún dato escrito por Arduino, ya sea el del Modo Escritura, un simple número o una simple letra. Esto puede ser debido a que el protocolo de lectura de Arduino y de Android sea diferente y por eso mismo no se puede implementar la misma forma de lectura.
- Muestra solamente parte del contenido. Tras probar de grabar una tarjeta mediante Android y pasarla por la app de lectura, se muestra parte del contenido. Esto puede ser debido a lo comentado en el punto anterior que tal vez se una implementación errónea plantear la lectura como se plantea en Arduino.

8. CONCLUSIONES

Este proyecto empezó siendo la continuación de otro estudio que había logrado una muy buena implementación de un primer prototipo. A día de hoy, mediante este proyecto, se ha podido obtener un prototipo que cumple las expectativas creadas al inicio de él superándolas en algunos casos.

En cuanto al sistema de Arduino, se ha conseguido un modelo de programación muy simple a la hora de utilizarlo en el que se declara un sistema de gran fiabilidad que no permite errores y de utilización sencilla. Se ha pasado de un programa principal a 5 subprogramas (Configuración RTC, Modo Escritura, Modo Lectura, Modo Borrar, Modo Verificar tarjeta vacía). De esta manera, se ha podido cambiar un sistema rocoso y grande como el del Laaroussi, S. a uno más reducido y, que en cuanto a hardware, aprovecha mejor su memoria. También, se ha introducido un método de reposo llamado Sleep Mode mediante el cual duerme el sistema cuando no se es utilizado, de esta manera el sistema consumirá menos.

Se ha modificado el microcontrolador (Arduino Mega 2560) a uno de menor memoria, (Arduino UNO), dejando memoria libre por futuras mejoras de código, pero sin ser excesiva. También se ha eliminado todo aquello que no se usaba y que ocupaba espacio como el microcontrolador encargado de la comunicación entre el micro y el ordenador y las entradas y salidas digitales y analógicas.

Se han podido solucionar todos los errores que han surgido en cuanto al prototipo v1 en su versión mejorada v2 implementando una nueva PCB. Esta PCB se ha diseñado quitando la resistencia que hacía fallar al zumbador, añadiendo los condensadores para el cristal y los pines para comunicar con el monitor serie.

Se ha podido comprobar que los dos prototipos implementados han sido más eficaces en cuanto a consumo, teniendo así una autonomía mucho mayor a la del prototipo inicial. Estas mejoras también se han visto reflejadas en el precio ya que los prototipos v1 y v2 son mucho más económicos que el inicial.

En cuanto a la aplicación, pese a no poder llegar a leer los datos obtenidos por los prototipos v1 o v2, se ha podido obtener parte la información de las tarjetas grabadas por la aplicación NFC Tools. Se ha obtenido que la implementación del método de lectura de Arduino no ha sido funcional en el de Android debido a que no se ha conseguido mostrar la información de las tarjetas. Cabe decir que a pesar de la existencia de la aplicación NFC Tools y demás, ninguna es capaz de mostrar ningún dato por pantalla mientras sí lo hace de cualquier otro tipo de tarjeta. Es posible que el parte del problema esté en el tipo de tarjeta usado y no la estructura del código.

9. MEJORAS PARA UN FUTURO PROYECTO

Como se ha comentado, se han cumplido las mejoras propuestas por Laaroussi, S. por tanto, se puede decir que todo sistema puede llevarse a un nuevo progreso. De cara a un futuro proyecto se podría hacer una reducción más del código, haciendo que este ocupase menos espacio de la memoria SRAM y así poder utilizar un microcontrolador más pequeño. Esto llevaría a un rediseño tanto del hardware como del software en el que se podría reducir el tamaño de sistema.

También se propondría una batería y un escudo más pequeños. Tal vez una batería que tuviera menos capacidad de corriente pero que fuera más pequeña, esto le iría mejor al sistema y reduciría bastante su tamaño. Otra mejora sería utilizar una caja que se adecue mejor a las proporciones del prototipo, ya que la caja utilizada es demasiado amplia.

De cara a la aplicación se podría implementar otra forma de lectura de la tarjeta e indagar en el tema de lectura de los diferentes tipos de objetivos NFC. Si esta aplicación funcionara, se podría implementar que esta descargara los datos en el móvil para una mejor visualización de estos. Esta medida implicaría que el Modo de Lectura no fuera necesario y así, tampoco lo serían los pines que conectan el ordenador al micro.

10. AGRADECIMIENTOS

Me gustaría agradecer a mi tutor del proyecto, Joaquin Del Rio, toda la ayuda aportada a lo largo de este proyecto. También me gustaría agradecer la ayuda recibida por los compañeros del centro tecnológico.

11. BIBLIOGRAFÍA

Bin Abdul, M. (2017). Development of an NFC-based system for control and remote communication. *UPCommons*, 66.

Laaroussi, S. (2019). Sistema para cronomertraje deportivo basado en tecnología NFC. *UPCommons*, 94.

Medina Delgado, J. (2017). Diseño e implementación de un prototipo para la identificación de ganado bovino mediante la lectura y escritura de etiquetas con tecnología NFC. *Semantic Scholar*, 56.

Montero, R. (2017). Estudio de tecnología de comunicación de campo cercano, NFC. *Archivo Digital UPM*, 155.

12. ANEXOS

Los modos de los programas utilizados se encuentran en el siguiente drive:

https://drive.google.com/drive/folders/1r8WR5qowimBB23mxiqmVi0G1R5l2vNn9