UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO DE UN SISTEMA DE FICHAJE AUTÓNOMO PARA AULAS Y LABORATORIOS

Trabajo Fin de Grado

Héctor Pous Casas

Tutor: Roberto Capilla Lladró

Grado en Ingeniería Electrónica Industrial y Automática

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Curso 2018-2019

Valencia, Junio 2019

Resumen

El siguiente proyecto trata sobre el proceso de programación, diseño, montaje e instalación de un sistema portátil que permita realizar el registro de la asistencia a aulas o laboratorios, mediante la Tarjeta Universitaria Inteligente de la Universitat Politècnica de València. Este registro permitirá a profesores constatar, con la base de datos de la universidad, la asistencia a sus clases. Para ello, se analizarán las necesidades del proyecto y de las tecnologías utilizadas. El sistema se sustentará en el módulo Arduino.

Seguidamente, se tratará la programación, se utilizará el entorno Arduino IDE, realizando el código necesario para el funcionamiento de cada una de las funcionalidades del producto final. Además, se programará paralelamente una aplicación para Android desarrollada mediante la herramienta del MIT (*Massachusetts Institute of Technology*) llamada AppInventor2. Mediante esta aplicación se habilitará el registro de asistencia, se obtendrá el archivo y se podrá editar el nombre del profesor y asignatura del registro. En cuanto al diseño electrónico. Se creará una placa *shield*, que se conectará al Arduino, y contendrá los componentes electrónicos necesarios para el funcionamiento del sistema.

A continuación se tratará el montaje físico e instalación en una ubicación manejable y portátil, centrándose en los problemas que se han encontrado y en las soluciones propuestas.

Por último, se comentarán las futuras posibles mejoras del sistema, tanto en las fases de programación, diseño y montaje, y se finalizará exponiendo las conclusiones alcanzadas.

Palabras clave: Arduino, RFID, MIFARE, tarjeta universitaria, Android, asistencia, Bluetooth, AppInventor2, registro.

Resum

El present projecte tracta sobre el procés de disseny, programació, muntatge i instal·lació d'un sistema que permeta realitzar el registre de l'assistència a aules o laboratoris, mitjançant la Targeta Universitària Intel·ligent de la Universitat Politècnica de València. Aquest registre permetrà a professors constatar, amb la base de dades de la universitat, l'assistència a les seues classes. Per a això, s'analitzaran les necessitats del projecte i de les tecnologies utilitzades. El sistema se sustentarà en el mòdul Arduino.

A continuació es tractarà la programació, s'utilitzara l'entorn Arduino IDE, realitzant el codi necessari per al funcionament de cadascuna de les funcionalitats del producte final. A més, es programarà paral·lelament una aplicació per a Android desenvolupada mitjançant l'eina del MIT (Massachusetts Institute of Technology) cridada AppInventor2. Mitjançant aquesta aplicació s'habilitarà el registre d'assistència, s'obtindrà l'arxiu i es podrà editar el nom del professor i assignatura del registre. Quant al disseny electrònic. Es crearà una placa shield, que es connectarà al Arduino, i contindrà els components electrònics necessaris per al funcionament del sistema.

Seguidament, es tractarà el muntatge físic i instal·lació en una ubicació manejable i portàtil, centrant-se en els problemes que s'han trobat i en les solucions proposades.

Finalment, es comentaran les possibles millores futures del sistema, tant en les fases de programació, disseny i muntatge, i es finalitzarà exposant les conclusions aconseguides.

Paraules clau: Arduino, RFID, MIFARE, targeta universitària, Android, assistència, Bluetooth, AppInventor2, registre.

Abstract

This project deals with the process of design, programming, assembly and installation of a system that allows the registration of assistance to classrooms or laboratories, through the Smart University Card of the Universitat Politècnica de València. This registry will allow teachers to note, with the database of the university, the attendance to their classes. To do this, the needs of the project and the technologies used will be analyzed. The system will be supported by the Arduino module.

Next, the programming will be treated, the Arduino IDE environment will be used, making the code necessary for the operation of each of the functionalities of the final product. In addition, an Application for Android developed using the MIT (Massachusetts Institute of Technology) tool called AppInventor2 will be programmed in parallel. Through this Application the attendance record will be enabled, the file will be obtained and the name of the professor and subject of the record can be edited. As for the electronic design. A shield plate will be created, which will be connected to the Arduino, and will contain the electronic components necessary for the operation of the system.

Next, the physical assembly and installation will be treated in a portable and portable location, focusing on the problems that have been encountered and on the proposed solutions.

Finally, the possible future improvements of the system will be discussed, both in the programming, design and assembly phases, and will be finalized by presenting the conclusions reached.

Keywords: Arduino, RFID, MIFARE, university card, Android, assistance, Bluetooth, AppInventor2, registration.

Agradecimientos

Quiero dar gracias a mi tutor, Roberto Capilla, por su interés en el proyecto, su excepcional trato hacia mí y por solucionar cualquier duda que aparecía durante el trabajo.

También agradecer a los técnicos del departamento de electrónica, José González y María Recasens, por su ayuda en el diseño y fabricación de la PCB. A mi compañero Pablo Roig Monzón, por su ayuda con el código de lectura de tarjetas el cual ha facilitado el arranque del proyecto. También agradecer su ayuda para orientarme para realizar la memoria de este mismo proyecto.

A la familia, la cual siempre confía en mí y son un pilar fundamental de mi vida, sin los cuales todo este viaje no hubiera sido posible.

A mis amigos, con los que podía desconectar del trabajo y siempre están apoyándome. Particularmente a Joaquín Gumbau Copoví por su ayuda en el apartado financiero.

Por último, a mis compañeros de carrera, que tras 4 años más que compañeros ahora son amigos para toda la vida. A todos, muchas gracias.

"La función de un buen software es hacer que lo complejo aparente ser simple"
-- Grady Booch

Índice

Índice de figuras

Índice de tablas

1. Introducción

En la actualidad la tecnología está presente en casi todos los lugares de nuestras vidas. Esta nos ayuda y facilita gran cantidad de tareas, mejora el rendimiento y clasificación en muchos ámbitos. La tecnología tiene una influencia muy importante en la sociedad, y el desarrollo tecnológico ha ayudado a realizar avances en otra aéreas importantes para la humanidad.

El registro se podría definir como la acción de registrar o el documento donde se relacionan ciertos acontecimientos o cosas, especialmente aquellos que deben constar permanentemente de forma oficial [1]. La tecnología nos ayuda con el registro en cuanto al modo de realizarlo, el tamaño de almacenamiento, seguridad y muchos más factores.

Una de las aplicaciones más importante del registro, es tener control e información sobre la asistencia de un grupo de personas, ya sea en un ámbito laboral o educacional como es nuestro caso. Este proyecto se basa principalmente en esta premisa.

1.1 Objetivo del proyecto

El objetivo del proyecto es diseñar e implementar un sistema que permita al usuario tener un control sobre la **asistencia** a laboratorios o clases de la **Universitat Politècnica de València** (UPV), en principio en el edificio de la Escuela Técnica Superior de Ingeniería del Diseño (ETSID) o departamento de informática. Para llevar a cabo el proyecto, se necesita un sistema que funcione con la **Tarjeta Universitaria Inteligente** (TUI) de la propia UPV, se utiliza este modo ya que todos los alumnos disponen de su propia tarjeta. Esto evitará posibles falsos fichajes por diferentes personas ajenas a la clase o entre los mismos alumnos.

El proyecto requiere de una **base de datos del fichaje o asistencia de las tarjetas** de los alumnos, con la finalidad de comparar el registro de fichaje con la lista de alumnos para comprobar su asistencia. También, se requiere de una aplicación Android para Smartphone para la gestión y obtención de la base de datos, el usuario podrá configurar el nombre del archivo con la clase perteneciente a la lista de alumnos, habilitar el registro, obtener el archivo o eliminarlo.

Por otro lado, el proyecto tiene también como objetivo **fortalecer y enseñar** conocimientos relacionados con la electrónica que se estudiaron en el grado, pero con un enfoque más práctico o cara al mundo laboral. Algún ejemplo seria, la programación con Arduino, diseño y fabricación de PCB y programación de App para Android.

Por último, este proyecto se plantea como el comprobante y verificación de este sistema para su posible instalación en otros departamentos de la UPV. También es el inicio de un proyecto mayor en el que se podrá ir mejorando el rendimiento del sistema por futuros alumnos.

1.2 Antecedentes

El control de la asistencia ha estado presente desde hace miles de años. Los seres humanos siempre han querido controlar la cantidad de gente que acudía a un determinado lugar, más aun si era de carácter profesional o restringido. Por ello en la antigüedad en diferentes actos pasaban lista para entrar a actos de la nobleza o simplemente para entrar a trabajar. Esto empezó a ganar importancia con la revolución industrial, en la cual, con la entrada de las fábricas, debían tener un control de la gente que entraba y salía de la misma. Así comenzó el sistema de fichaje que actualmente utilizan las empresas en sus actividades profesionales.

En la antigüedad, este control de la asistencia era realizado por personas que manualmente apuntaban la lista de asistentes a un lugar. Según ha pasado el tiempo, se ha evolucionado en los métodos para realizar esta tarea, desde las antiguas tarjetas de fichaje hasta las aplicaciones digitales de la actualidad.

Hoy en día, se han desarrollado métodos de fichaje como las tarjetas de banda magnética, aplicaciones software y hasta mediante la huella dactilar del individuo, esto por mencionar algún ejemplo.

En la edad moderna se utilizan métodos automatizados para el control de la asistencia: sistemas con una gran eficacia, los cuales permiten realizar el fichaje de forma automática. También permiten el almacenamiento de este fichaje en formato digital, mejorando notablemente a los métodos anteriores.

En estos últimos métodos se basará el sistema del proyecto: se utilizará la Tarjeta Universitaria Inteligente de la UPV, disponible por cualquier integrante de la universidad. En la UPV ya existe un sistema parecido que utiliza las Tarjetas Universitarias Inteligentes como el acceso a lo aparcamientos, pero el sistema del proyecto será a menor escala y por gestión propia del usuario.

1.3 Motivación

La principal motivación para realizar este proyecto ha sido la gran variedad de campos de la electrónica que se podía trabajar al realizar este Trabajo Final de Grado. También porque la mayor parte del proyecto está orientado para la práctica y aplicación de conocimientos que durante el grado no se pudo profundizar. Otra motivación es lo práctico que resulta el sistema y su posibilidad de poder implantarlo en diferentes ámbitos, como el laboral ya que muchas empresas podrían interesarle este tipo de sistema y más todavía con el reciente cambio del artículo 34 del Estatuto de los Trabajadores que obliga a estas a llevar un registro de la jornada laboral de sus trabajadores [2].

Por otro lado, el factor didáctico de aprendizaje y fortalecimiento de conocimientos de electrónica, programación en la plataforma Arduino, diseño de PCB, etc. ha sido una

gran motivación para realizar este proyecto. El proyecto ha exigido poner en práctica y dominar todos estos conceptos para aplicarlos conjuntamente de forma correcta.

Por último, el factor humano ha sido determinante como motivación por poder trabajar con profesionales que se dedican continuamente a estas disciplinas y aprender de ellos, tanto técnicos como profesores.

1.4 Requisitos del sistema

Los requisitos del sistema provienen mayoritariamente de los siguientes puntos:

- Necesidad de manejabilidad.
- Facilidad de la interfaz de usuario, y registro automático de la asistencia.
- Características técnicas del sistema.

El sistema debe cumplir las siguientes funciones o requerimientos:

- **Guardado en la base de datos de la asistencia.** Al detectar cada tarjeta del alumno, el sistema debe guarda su ID en la base de datos.
- **Eliminar la base de datos cuando se requiera.** Se debe poder eliminar el archivo con la asistencia para poder realizar nuevos registros.
- **Registro de fecha y horas del fichaje.** Cada base de datos debe de incorporar la fecha y hora de realización de la asistencia de la clase.
- **Obtención de la base de datos mediante App Android o tarjeta SD.** El archivo debe de poder obtenerse mediante la App de móvil o directamente de la SD.
- **Alimentación mediante batería.** Para poder tener un sistema portátil y manejable se necesita de una fuente de alimentación fácil de transportar.
- **Lectura de Tarjetas Universitarias Inteligentes (TUI).** Las tarjetas inteligentes de la universidad serán las que se usarán para el proyecto por eso se necesita poder realizar la lectura de las mismas.
- **Edición del nombre del archivo de la base de datos.** Se debe de poder editar el nombre del archivo para diferenciar entre profesores y clases que se pasa la asistencia.

2. Diseño y desarrollo electrónico

A continuación, se describirá la elección de los componentes utilizados y el proceso del diseño del hardware para el sistema. Esta sección se divide en varias partes, primero componentes, después el diseño del hardware y software.

2.1 Diseño de hardware y elección de componentes

En este apartado se analizará cada unas de las partes del sistema por separado para justificar el diseño del hardware y la elección de sus componentes, para ello se compararán las diferentes alternativas y características de los elementos.

2.1.1 Microcontrolador

El **microcontrolador** es un computador pequeño en un solo circuito integrado. Un microcontrolador contiene una o más CPU (*Central Processing Unit* o Unidad Central de Procesamiento) junto con memoria y periféricos de entrada / salida programables. La memoria puede ser RAM, NOR flash u OTP ROM que se incluye a menudo en el chip, así como una pequeña cantidad de RAM. Es el núcleo del sistema, se encarga de manejar la información, procesarla y actuar respecto a su programación. Por lo cual, todo microcontrolador necesita de una programa que le indique los procesos que debe realizar. Con un mismo circuito es posible modificar la programación para realizar diferentes tareas. El programa es editable por un programador, por tanto esto permite al sistema ser modular y flexible [3].

En este proyecto se requiere del uso de diferentes componentes electrónicos como lector de tarjetas, reloj en tiempo real, módulo bluetooth, etc. Debido a esto, el microcontrolador debe ser capaz de comunicarse e interactuar con todos los elementos del sistema.

El programador que se utilizará para el proyecto será **Arduino.** Arduino es una plataforma open-source (código abierto) de electrónica, desarrollada por Massimo Banzi, David Mellis y David Cuartielles entre otros. Arduino está basado en una placa de circuito física programable y en el software imprescindible para su programación, también conocido como IDE (*Integrated Development Environment* o Entorno de Desarrollo Integrado). El IDE de Arduino se ejecuta en el PC, permitiendo al usuario programar y cargar el código en la placa programable [3].

Arduino es uno de las placas más utilizadas, a diferencia de otras marcas, porque no necesita de hardware adicional (programador) para cargar el código en la placa, solo necesita de un cable USB. Otro punto fuerte de esta plataforma es su IDE que utiliza una variante simplificada de C++, lo cual resulta sencillo el aprendizaje del programa. Por último, el diseño de las placas físicas de Arduino separa y ayuda al acceso a las entradas/salidas del microcontrolador.

En el mercado se pueden encontrar varios modelos de placas de Arduino, de las cuales las más utilizadas son las Arduino Uno, Mega y Nano. Para discernir entre las diferentes placas y exponer el criterio de elección de la misma se mostrará una tabla comparativa de las características de las diferentes placas.

Arduino	Uno	Nano	MEGA
Procesador	ATmega328P	ATmega328P	ATmega2560
Voltaje de entrada/ operación	5 V/ 7-12 V	5 V / 7 - 9V	5 V / 7 – 12 V
Velocidad CPU	16MHz	16MHz	16MHz
Analógicas In/Out	6/0	8/0	16/0
Digitales IO/PWM	14/6	14/6	54/15
EEPROM [kB]	1	1	4
SRAM [kB]	2	2	8
Flash [kB]	32	32	256
USB	Regular	Mini	Regular

Tabla 1: Comparativa entre Arduino Uno, Nano y Mega. Fuente: *https://www.Arduino.cc/en/products.compare* (adaptado)

Los principales criterios para la elección de una placa de Arduino son:

- **Número de pines que utilizará el sistema.** Principalmente los pines se utilizan para las entradas y salidas de la placa. El Arduino Uno es el más habitual por tener una cantidad razonable de pines. Por otro lado, si se requiere el uso de muchos pines de entradas y salidas se escogerá el Arduino MEGA. Por último, si la aplicación requiere pocos dispositivos se podría utilizar el Arduino Nano.
- **Memoria Flash.** Es la memoria de programa que almacena el código. Dependiendo de la cantidad de código se elige un tamaño de Flash, esta cantidad se puede observar en la IDE de Arduino antes de compilar el programa. El tamaño del código es posible reducirlo mediante funciones y herramientas de optimización.
- **Memoria SRAM.** Almacena las variables durante el tiempo de funcionamiento, si se desconecta la alimentación se borran estos datos. El uso de esta memoria SRAM suele ir ligado a la memoria Flash, ya que a mayor código suele haber más cantidad de variables.
- **Memoria EEPROM.** (*Electrically Erasable Programmable Read-Only Memory,* memoria de solo lectura programable y borrable eléctricamente), que almacena información sin que sea borrada al interrumpir el suministro eléctrico.
- **Tensión de funcionamiento.** Algunos Arduino usan tensiones de 3.3V, esta tensión no es compatible con todos los componentes electrónicos. Por otro lado, si utiliza una tensión de 5V, es más frecuente y se podría adaptar la señal a componentes que requieran menor voltaje.
- **Tensión de alimentación.** Un factor a tener en cuenta si se debe utilizar componentes con mayor voltaje como podría ser un adaptador AC/DC de 12V.
- **Formato físico.** El MEGA es el mayor en tamaño de los 3 expuestos, después el Uno y el más pequeño es el Nano, el cual no puede incorporar shields (placa superpuestas al Arduino para añadir funcionalidades).

De los criterios expuestos, este proyecto prevé los siguientes puntos:

- **Número de pines que utilizará el sistema.** Se utilizará un número medio de pines, el Arduino Uno es el más indicado por su número moderado de pines.

- **Memoria Flash.** Se utilizará un tamaño medio de programa.
- **Memoria SRAM.** Se usará una cantidad normal de memoria.
- **Memoria EEPROM.** Se utilizará poco ya que las tarjetas leídas se guardarán en un archivo en una tarjeta SD extraíble en un módulo externo al Arduino.
- **Tensión de alimentación.** 5V es la tensión más estandarizada para los componentes electrónicos.
- **Tensión de alimentación.** Se prevé utilizar una batería externa de 9 V.
- **Formato físico.** Se utilizará el formato de Arduino Uno, en el cual se incorporará una shield. El MEGA también puede llevar la shield pero su tamaño es mayor.

Tras ver los criterios, se escoge el Arduino Uno, la cual es la placa más utilizada y popular de las placas. También tiene librerías que para otras placas no están del todo adaptadas.

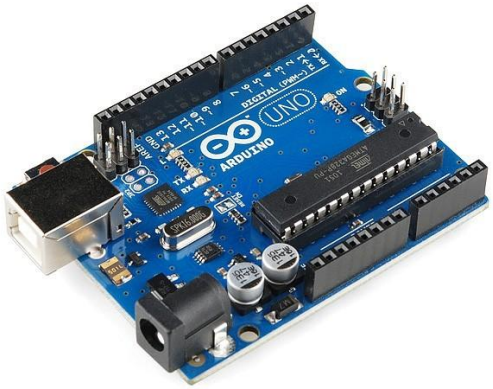


Figura 1: Arduino Uno. Fuente: https://www.google.com/search?q=Arduino+Uno&rlz=1C1CHBF_esES835ES835&source=lnms&tbm=isch&sa=X&ved=0ahUKEwixjJDswd_iAhUR9BoKHXwaAzsQ_AUIECgB&biw=1680&bih=907#imgrc=beVctn8gXtURpM:

Por el hecho de que Arduino es una hardware libre [4], existen muchas placas clonadas, es decir que fabricantes ajenos a Arduino han desarrollado sus propias placas, normalmente a precio inferior al original. Se pueden utilizar estas placas pero hay que tener en cuenta que a lo mejor los componentes no son de la calidad de las originales, ya que de este lugar se suele recortar para mejorar el precio.

2.1.2 Módulo lector de tarjetas RFID / MIFARE

El **lector de tarjetas RFID** es el elemento principal en el que se basa este proyecto, este actúa como nexo entre el usuario y el sistema del microcontrolador. Para poder elegir correctamente el componente electrónico, se examinará la relación entre las dos partes con las que tiene que interactuar, el microcontrolador y la tarjeta RFID.

La tarjeta RFID que se utilizará será la Tarjeta Universitaria Inteligente (TUI) de la Universidad Politécnica de Valencia (UPV) [5]. Estas tarjetas se basan en una tecnología sin contacto llamada **MIFARE,** desarrollada por NXP Semiconductors. Esta tecnología utiliza el estándar de comunicación ISO/IEC 14443-A, de 13.56 MHz, concretamente las partes 1, 2 y 3 de esta especificación. [6]

Cada tarjeta de la UPV posee un **código MIFARE único**, el cual está formado de 8 dígitos hexadecimales. Esto será reflejado por 4 bytes en el código. El sistema registrará cualquier TUI de la UPV en la base de datos del sistema, para más tarde comparar el código de cada alumno con los registrados. Esto se realiza de esta forma porque por la ley de protección de datos no se puede obtener cada código MIFARE relacionado con los alumnos para proteger sus datos personales.

El lector de tarjetas RFID más común y utilizado junto el Arduino es el MFRC522, el cual se encuentra en formato módulo:

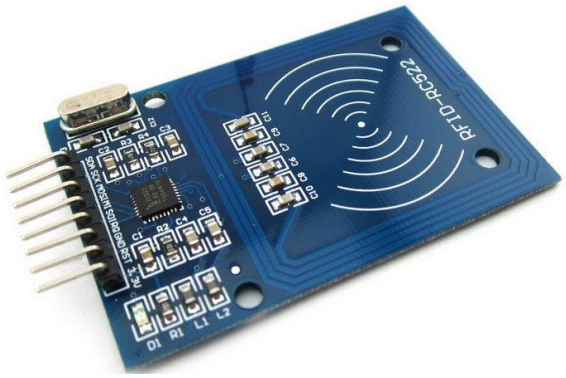


Figura 2: Módulo MFRC522.
Fuente:https://www.google.com/search?q=rfid+mfrc522&rlz=1C1CHBF_esES835ES835&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjzgdiggOHiAhULzhoKHTNrAOcQ_AUIECgB&biw=1680&bih=907#imgrc=8ayrY-80t9EDKM:

Existen diferentes librerías libres para este tipo de módulo, desarrolladas por miembros activos de la comunidad de Arduino, las cuales facilitan su programación. Esta gran variedad de librerías es debido a la popularidad del sensor por sus diversas opciones de compra disponible a bajos precios. El lector RFID utiliza una frecuencia de 13.56 MHz, que es la frecuencia que usan las tarjetas MIFARE y las TUI de la UPV. Además, incluyen comunicación por UART, bus SPI y bus I2C. También es un sensor de tamaño reducido idóneo para nuestra aplicación, por estas características del módulo es el elegido para el sistema del proyecto.

2.1.3 Módulo reloj en tiempo real (RTC)

Un registro como puede ser un fichaje de los trabajadores de una empresa o la asistencia a un aula necesita de control del instante en que se realiza ese fichaje, es decir, tener constancia de la fecha y hora del registro para saber si se ha producido a tiempo. Al leer la tarjeta mediante el MFRC522, la fecha y hora es obtenida gracias a un módulo RTC (Real Time Clock). El módulo RTC debe estar alimentado por una batería externa, una pila de botón, para mantener el reloj funcionando incluso sin la alimentación principal del sistema.

Los diferentes modelos de RTC contienen un cristal resonador, una batería para mantener el tiempo, un chip para conseguir la fecha y hora, y demás componentes electrónicos.

Dentro de las opciones que se disponen de módulos de RTC para Arduino en el mercado, se destacan los 3 siguientes:

- **DS1302**
- **DS1307**
- **DS3231**

A continuación, se procede a exponer las diferencias entre los modelos mencionados.

Las principales diferencias entre los módulos se refieren al protocolo utilizado, tamaño y tipo de memoria interna, dimensiones y consumo de energía. Los módulos DS1307 y DS3231 utilizan el protocolo I2C, mientras que el DS1302 utiliza un protocolo serie. El protocolo I2C utiliza 2 pines: el SDA (datos) y el SCL (reloj de sincronismo). El DS1302 utiliza tres pines: SCLK (Reloj de sincronismo), CE (Chip Enable - para habilitar la transmisión de datos) e I / O (datos).

La gran diferencia es que el I2C señala el inicio y fin de la transmisión de datos por medio de cambios en el estado lógico en los pines SCL y SDA. Básicamente, si SCL es a nivel alto y hay una transición de alto a bajo en la SDA, el DS1307 / 3231 reconoce como inicio de la comunicación. Ahora bien, si SCL es a nivel alto y hay una transición de bajo a alto en la SDA, el DS1307 / 3231 reconoce como el fin de la comunicación.

En el DS1302, el papel de estas combinaciones lógicas es desempeñado por el pin CE. Si se encuentra en nivel alto, la comunicación se inicia, si está a nivel bajo, la transferencia de datos termina. Es decir, siempre que un procedimiento de lectura o escritura se ejecute con el DS1302, el pin CE debe colocarse a un nivel alto.

Figura 3: Módulo DS1302. Fuente:https://www.dx.com/p/rtc-ds1302-real-time-clock-module-for-Arduino-avr-arm-pic-smd-2062445#.XQAKWogza70

El DS1302 dispone de una batería recargable que se carga mientras este alimentado con una tensión externa, mientras que el DS1307 no dispone de ella. Además, el DS1307 posee más RAM que el DS1302, también tiene una salida de onda cuadrada programable. Para este proyecto solo se necesita la salida con la fecha y la hora.

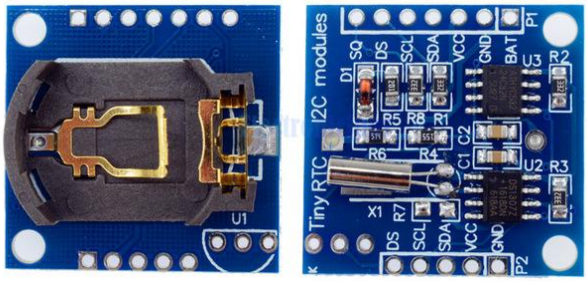


Figura 4: Vista anterior (izquierda) y posterior (derecha) RTC DS1307. Fuente:https://es.aliexpress.com/item/I2C-RTC-DS1307-AT24C32-Real-Time-Clock-Module-For-AVR-ARM-PIC/2037925259.html

Otro factor a tener en cuenta es la precisión de los módulos, el DS1302 y el DS1307 tienen una precisión parecida, que puede oscilar entre aumentar/disminuir unos segundos al día y aumentar/disminuir unos 3-5 minutos al día. Estos errores son acumulativos día tras día.

Por otro lado, el DS3231presenta una precisión mayor a sus hermanos, ya que aumenta/disminuye menos de 1 segundo cada 5/6 días. Esto es gracias a un sensor de temperatura (precisión de ±3ºC) incorporado en el DS3231 que es capaz de compensar los cambios de temperatura en el ambiente. [7]

El control de la temperatura es importante porque los cambios de temperatura afectan al cristal resonador que se encarga de obtener el tiempo. También afectan a la precisión la misma calidad del cristal. [8]

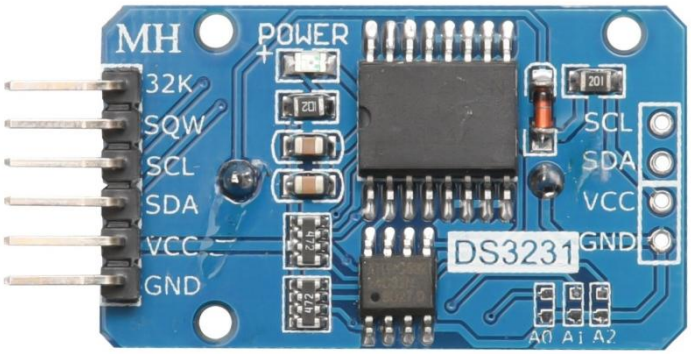


Figura 5: Módulo RTC DS3231. Fuente:https://e-radionica.com/en/rtc-module-ds1307.html

En las hojas de características antiguas del DS1302, el pìn del CE fue erróneamente llamado Reset (RST). Pero la funcionalidad era la misma, es decir, era un error técnico en la redacción del datasheet.

Las tensiones de alimentación también son diferencias importantes, con el DS1302 operando con un mínimo de 2V, mientras que el DS1307 opera con un mínimo de 4.5V y el DS3231 funciona con un mínimo de 3.3V. Todos ellos tienen un máximo de 5.5V de tensión de alimentación. [9]

Para una mejor visualización se compararán mediante una tabla:

	DS1302	DS1307	DS3231
Interfaz	SPI	I2C	I2C
Carga de batería	Algún modelo	No	No
Alimentación	2-5 V	4.5-5.5 V	3.3-5.5 V
Dimensiones	44 x 23 x 8 mm	28 x 26 x 8 mm	38 x 22 x 14 mm
Precisión	Entre unos segundos a nos minutos al día	Entre unos segundos a nos minutos al día	Menos de 1 segundo cada 5/6 días
Precio [11/06/2019]	1.99 € [10]	1.27 € [11]	2.25 € [12]

Tabla 2: Características diferentes módulos RTC. Fuente: realización propia

El sistema de fichaje debe tener una cierta precisión en la determinación de la fecha y hora, porque se debe comparar el registro con la hora a la cual se lleve a cabo las clases del tutor que utilice el sistema. Aunque no es requerida una precisión tan grande como la de un reloj en tiempo real, se debe tener en cuenta.

Por otra banda, el sistema esta diseñado para funcionar autónomamente. El usuario utilizará el sistema solo para obtener el registro de a asistencia y no actualizará la hora del módulo de reloj, por lo costoso que seria la tarea de cargar el código cada vez. Aunque en el código, como se verá más adelante, esta la opción de actualizar la hora.

Por todo lo expuesto anteriormente, se opta por el módulo DS3231 por su mayor precisión respecto a los demás, ya que el usuario no debería actualizar la hora del reloj hasta en un año que es cuando el reloj presentaría un error mayor de un minuto. Aunque un minuto se podría considerar aceptable para el proyecto.

2.1.4 Módulo Bluetooth

Bluetooth es una tecnología de red utilizada como estándar industrial para conexiones inalámbricas de corto alcance para intercambiar datos. Esta tecnología utiliza una banda ISM sin licencia con longitudes de onda corta (UHF), con una frecuencia entre los 2,402 GHz y los 2,480 GHz. Su alcance ronda los 10 menos aproximadamente, suficiente para este proyecto.

Para el reconocimiento de las tarjetas y gestión del fichaje se utiliza la aplicación móvil. En la cual se podrá visualizar la ID de cada tarjeta detectada, enviar el archivo de registro, eliminarlo o deshabilitar el registro, también cambiar el nombre del archivo con el aula y profesor correspondientes a ese fichaje. Para obtener el registro se necesita una comunicación entre el Arduino y el móvil, la cual se realiza mediante un módulo bluetooth, ya que está presente en la gran mayoría de dispositivos móviles.

En cuanto a módulos bluetooth existen dos principalmente comunes:

- Bluetooth HC-05
- Bluetooth HC-06

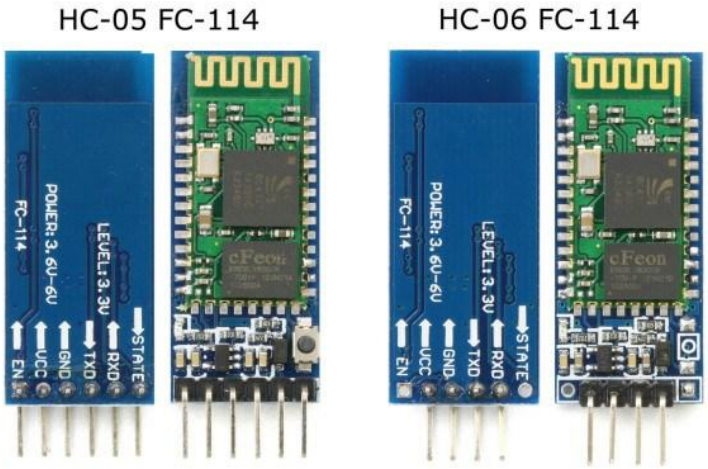


Figura 6: Módulos HC-05 y HC-06. Fuente:https://aprendiendoArduino.wordpress.com/tag/hc-05/

Estos Módulos HC-05 y hc-06 son muy parecidos. La principal diferencia es que el HC-05 puede funcionar en modo ***master*** y en modo ***slave,*** es decir, que puede ser receptor de enlaces y además puede crear esos enlaces con otros dispositivos bluetooth. Mientras el HC-06 únicamente puede ser esclavo (slave), por tanto solo puede ser receptor de los enlaces. Otra forma de diferenciarlos y la más rápida es por el aspecto físico externo, el HC-05 dispone de 6 pines mientras que el HC-06 dispone de 4 pines. Los dos pines que tiene demás el HC-05 son para comandos específicos del módulo, por ejemplo para programar el bluetooth mediante comandos AT. El pin KEY debe estar en HIGH al encender el módulo. En el caso del HC-06 entra en modo programación mientras no haya nadie conectado al bluetooth al encenderlo.

En este proyecto, la comunicación es entre el Arduino y el dispositivo móvil. El módulo bluetooth del Arduino permanece en modo esclavo, a la espera de que el dispositivo móvil del usuario se conecte para comenzar con el fichaje. Por ello, se escoge el módulo HC-06, el cual es más sencillo de utilizar y mediante la puesta en práctica ha resultado con menos problemas que el HC-05.

A continuación, se explica brevemente como configurar el módulo HC-06 mediante comandos AT. Estos comandos son unas instrucciones enviadas por comunicación serie al HC-06. Para entrar en esto modo de programación se debe establecer comunicación con el módulo y no conectar ningún otro dispositivo bluetooth al mismo. [13]

Principales comandos AT utilizados:

- **AT:** si la comunicación es correcta, devuelve un OK.
- **AT+NAME<nombre>:** cambia el nombre del módulo Bluetooth que será visible a otros dispositivos. En el proyecto se ha programado el nombre “FichajeBluetooth” mediante la línea “AT+NAMEFichajeBluetooth”.

- **AT+PIN<pin>:** configura como contraseña para vincularse el indicado en <*pin*>. Número de 4 a 6 dígitos. Por defecto viene "1234". En este proyecto se configura la contraseña "1936".
- **AT+VERSION:** devuelve la versión de firmware del dispositivo.

Otro aspecto del módulo HC-06 a tener en cuenta es que funciona a **3.3V**, en varias fuentes de información se expone que podría funcionar igualmente con 5V pero se hará como indica en las características del módulo que expresa su funcionamiento con 3.3V. Para ello, se debe realizar un **divisor de tensión** para adaptar la tensión de 5V procedente de la salida TX del Arduino a los 3.3V requeridos por el pin RX del HC-06 del bluetooth.

El divisor de tensión es circuito que permite reducir una tensión de entrada a una menor de salida para poder adaptarla a diferentes situaciones:

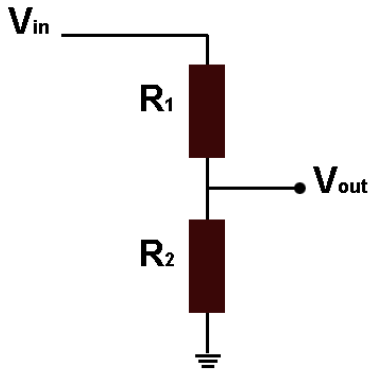


Figura 7: Divisor de tensión. Fuente:https://programarfacil.com/blog/divisor-de-tension-en-Arduino-multiplica-tus-entradas-digitales/

A partir de la ley de ohm, y sabiendo que la intensidad que circula por R1 y R2 es la misma, se obtiene la siguiente ecuación:

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

Se necesita 3.3V en V_{out} , sabiendo que en V_{in} se dispone de 5V.

$$3{,}3V = 5V\frac{R_2}{R_1 + R_2} \rightarrow\ R_2 = 0{,}66(R_2 + R_1) \rightarrow\ R_2 = 2R_1$$

Observando la formula anterior, es necesario que R2 tenga el doble de resistencia que R1. Para ello, se colocan dos resistencias equivalentes en serie iguales a R1. Se debe tener en cuenta la disipación de potencia de estas, se escoge un valor estándar de **1 KΩ** porque así la corriente que circule por ellas sea baja.

Utilizando nuevamente la ley de Ohm se calcula la intensidad de corriente en las resistencias.

$$I = \frac{V}{(R_1 + R_2)} \rightarrow I = \frac{5V}{(1\ K\Omega + 2K\Omega)} = 1{,}66mA$$

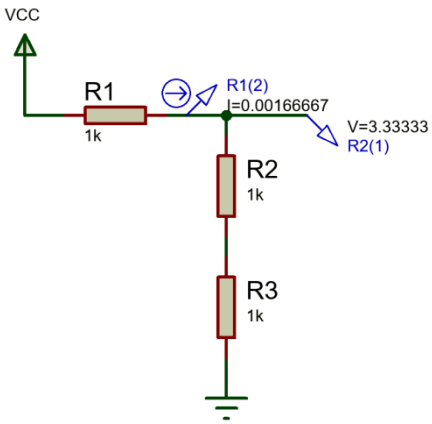


Figura 8: Divisor de tensión pin RX del HC-06. Fuente: propia.

Por último, se debe tener en cuenta que las conexiones RX/TX han de estar cruzadas, es decir el TX del Arduino con el RX del bluetooth y viceversa.

Bluetooth	Arduino
RX	TX
TX	RX

Tabla 3. Conexiones Bluetooth con Arduino. Fuente: propia

2.1.5 Módulo para almacenamiento de la asistencia

El sistema almacena las ID de las tarjeta detectadas, además almacena también la fecha y hora de la realización del fichaje. Desde un principio se decantó por un formato de almacenamiento con tarjeta SD, por los antecedentes de trabajos anteriores similares a este proyecto. Como es el caso de la memoria EEPROM externa, la cual se descartó por su compleja programación, su conexión electrónica y su limitada capacidad de almacenamiento.

El formato de las tarjetas SD es de los más utilizados actualmente debido a su uso en los dispositivos móviles. Este formato es una tecnología desarrollada por SanDisk, Panasonic y Toshiba [14]. Hoy en día, la relación entre precio y la capacidad de almacenamiento es buena, ya que pueden haber tarjeta desde 1 GB hasta 256 GB. Existen diferentes tamaños de tarjetas: SD, microSD Y miniSD. En la actualidad la microSD es la más utilizada, principalmente, por los teléfonos móviles.

Una opción son los módulos SD O microSD. Por su menor tamaño, los microSD suelen ser más utilizados, y como en este proyecto se tenía disponible una tarjeta microSD, se escoge la microSD.

Figura 9: Módulo microSD. Fuente: https://naylampmechatronics.com/modulos/104-modulo-micro-sd-card.html

Aunque estos módulos necesitan cableado aparte y programación extra. Por tanto se probó utilizar una shield Ethernet para Arduino que incorpora el lector para microSD. Esta shield es una placa de circuito impreso la cual incorpora todo el cableado y programación necesaria para hacer funcionar la SD, también tiene pines en la cara anterior para poder conectarse con el Arduino y pines hembra en la parte superior para permiten realizar más conexiones. Por su sencillez y su precio asequible se elige para el proyecto.

Figura 10: shield Ethernet con lector microSD. Fuente:http://www.maxelectronica.cl/shield-Arduino/4-Arduino-shield-ethernet-w5100.html

El lector del microSD de la *shield* Ethernet se comunica con el Arduino mediante el **bus SPI,** que es el mismo bus que utiliza el MFRC522. Por esto, se debe tener en cuenta los pines del *Slave select* del bus SPI para que no haya interferencias entre los componentes. El lector microSD del shield Ethernet utiliza obligatoriamente el pin 4 y no puede cambiarse al ir así configurado en la placa electrónica.

En este proyecto se utilizará una tarjeta microSD de 8GB al dsiponer de ella, no obstante, podría utilizarse de menor capacidad ya que el registro no suele superar los pocos MB.

2.1.6 Batería del sistema

La batería para la alimentación es la que se encarga de suministrar energía al Arduino y al resto de componentes del sistema. Se ha escogido este componente por su versatilidad y pequeño tamaño, porque la aplicación del sistema requiere de poder trasladar el sistema de forma cómoda para el usuario. Esto es debido a que el proyecto no está diseñado solo para un aula o laboratorio, sino por lo contrario para que cualquier profesor de la UPV pueda gastarlo en sus horas docentes.

En el mercado se disponen de diferentes baterías externas, las cuales se diferencian por su tamaño, tensión de entrada/salida y por su capacidad. En este proyecto se utilizó una de las baterías disponibles en el departamento de electrónica. Esta batería es una *powerbank* de la marca X-one, las características son 2600mAh de capacidad, tensión de entrada 5V/1ª y tensión de salida 5V/2,1ª. [15]

Figura 11: powerbank para alimentación del sistema. Fuente:https://www.discoazul.com/x-one-powerbank-2600mah-azul.html

La batería irá conectada al pin Vin del Arduino. Este pin tiene una doble función:

- **Permite aplicar una fuente de alimentación externa entre el rango de 6 a 12 V** directamente a la entrada del regulador de la tarjeta Arduino. En este caso, no se cuenta con **protección contra inversión de polaridad** ni contra **sobre corriente**. En caso de aplicar voltaje directamente al pin VIN, no se debe aplicar simultáneamente un voltaje en el jack.
- **Funciona como salida de voltaje cuando el Arduino se está alimentando a través del jack de alimentación**. En este caso el voltaje presente en VIN será aquel que estemos aplicando en el jack, restando la caída de tensión en el diodo de protección de inversión de polaridad (alrededor de 0.7 volts). No se recomienda conectar cargas mayores a 1000 mA en este pin, ya que podemos dañar el diodo de protección.

Este método se utiliza para alimentar el Arduino con fuentes no reguladas de corriente directa o conjunto de baterías AA o AAA. Es recomendable la alimentación mediante este pin cuando las baterías proporcionan un voltaje de 6 V, porque no hay diodo de protección que cause caídas de tensión [16]. Aunque la batería del departamento proporciona 5V se ha comprobado que funciona correctamente.

Por su tamaño es idónea para la aplicación del sistema. También se comprobó la durabilidad de la batería, la cual tras estar dos días conectada las 24 horas del día se agotó. Esta capacidad es aceptable para el sistema ya que al ser portátil y no es necesaria su conexión tantas horas al día. Además se comprobó la durabilidad utilizando la aplicación durante su programación y estuvo activa durante varias semanas.

Después de ver las características de la batería externa, se opta por su utilización en el proyecto, aunque cualquier otra batería que se ajuste a estas premisas podría también ser útil.

2.2 Conexión y pruebas en la placaboard

Una ***protoboard*** es una placa o tablero con orificios interconectados eléctricamente entre si por patrones específicos, normalmente conexionados eléctricamente los agujeros de las filas, pero entre filas no están conectados, y todas las dos columnas de ambos lados conexionadas únicamente por la columna, normalmente se utiliza para la masa y VCC. Es muy útil para la creación y comprobación de circuitos, debido a su facilidad de conexión para cables y componentes [17]. Por ello, se utiliza la protoboard para el montaje del sistema y comprobación del código a medida que se va realizando. Posteriormente, una vez funcione el sistema correctamente se procederá al diseño y montaje de la PCB.

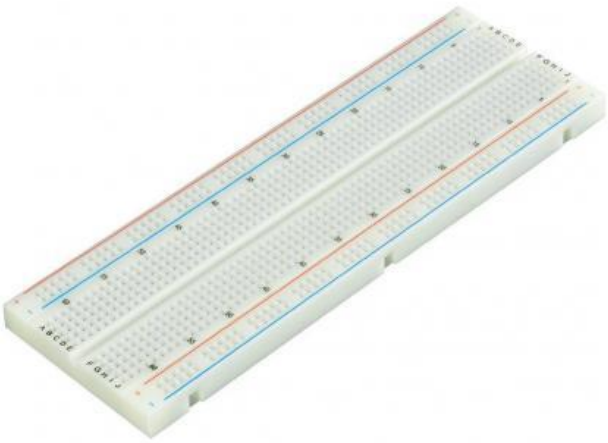


Figura 12: Protoboard para comprobación. Fuente: http://www.tecnovoz.es/CATEGORIA_131

Para facilitar la visualización y el conexionado se expresa el uso de los pines con sus correspondientes componentes en la siguiente tabla:

Pin	Componente	Función
GND	Masa componentes	Masa
VCC	Alimentación componentes	Alimentación
3.3	RC522	3V3 del RC522
5	HC-06	TXD
6	HC-06	RXD con divisor de tensión
12	Bus SPI, MISO	MISO del RC522/ Ethernet so
11	Bus SPI, MOSI	MOSI del RC522/ Ethernet SI
13	Bus SPI, SCK	SCK del RC522/ Ethernet SCK
4	MicroSD (shield Ethernet)	SS (Slave select)
10	W5100(shield Ethernet)	SS (Slave select)
8	RC522	SDA (SS, Slave Select)
9	RC522	RST (reset)
A4	RTC DS3231	SDA
A5	RTC DS3231	SDA

Tabla 4: Conexiones de lso pines del sistema. Fuente: propia

A partir de la información de la tabla se conectan los componentes en la protoboard. Una vez realizado el montaje, se comprueba el funcionamiento de cada componente. La comprobación del sistema es satisfactoria, por lo tanto, se puede comenzar con la parte de diseño de la PCB y programación del sistema.

Por último, se presenta un esquema gráfico de las conexiones para su mejor comprensión y visualización.

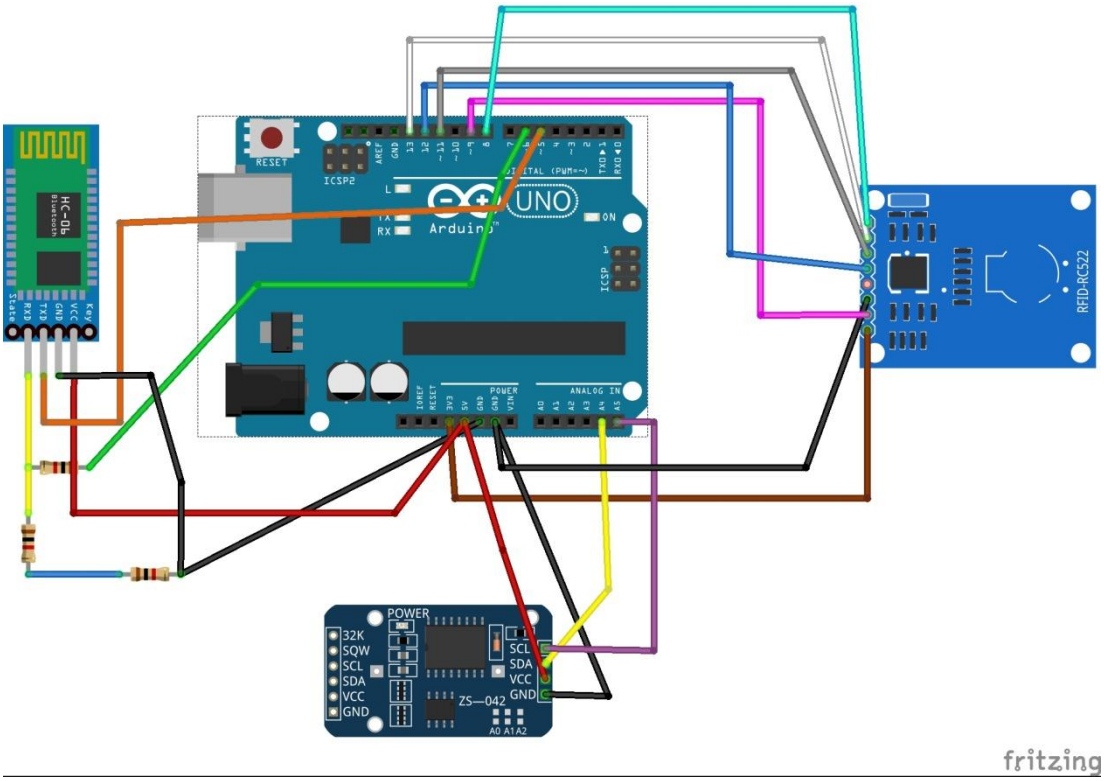


Figura 13: Diagrama de conexión. Fuente: propia

2.3 Diseño de PCB

Una PCB o Printed Circuit Board (placa de circuito impreso) es una placa formada por pistas, caminos o buses de material conductor laminadas sobre una base no conductora o aislante. Su principal función es conectar eléctricamente a través de las pistas conductoras, y aguantar mecánicamente, un conjunto de componentes electrónicos.

La placa que se debe construir ha de contener todos los componentes electrónicos necesarios para la aplicación del sistema. Además, se necesario un tamaño compacto de la placa para poder introducirla dentro de una caja para su fácil transporte. Por consiguiente, se escoge diseñar una PCB en formato shield, como el módulo microSD, para poder conectarlo encima del Arduino y la shield Ethernet. Para realizar la conexión se utilizaran una tiras de pines a la misma altura de las conexiones del Arduino, así evitamos conexiones eléctricas con el Arduino por soldadura o cables.

El diseño de la PCB está programado con el software ISIS Proteus. El primer paso es diseñar la base de la placa para que coincida con las medidas del Arduino Uno. Para ello, se puede incluir una librería en la cual añade una placa en la vista PCB Layout. En este proyecto se midió manualmente las medidas del Arduino Uno y se creó manualmente la base de la placa, también se alinearon correctamente las tiras de pines J1, J2, J3, J4.

Una vez se tiene la base se comienza a añadir el resto de componentes. Para ello, se tiene que añadir el componente tanto en la parte del Esquemático como en el formato PCB. Si no está disponible el paquete para PCB de un componente, se debe crear manualmente siguiendo los siguientes pasos:

1. Se crea la forma mediante la herramienta 2D Graphics Box Mode.
2. Se añaden los pines mediante la herramienta Device Pins Mode en default, y se nombran y se numeran cada pin.
3. Con el botón Make Package se crea el paquete.
4. Se asigna a cada componente su paquete, en la vista de Schematic Capture se hace clic con el botón secundario en el componente, se selecciona Packaging Tool y se asocia el paquete

La vista del esquemático de la PCB es la siguiente:

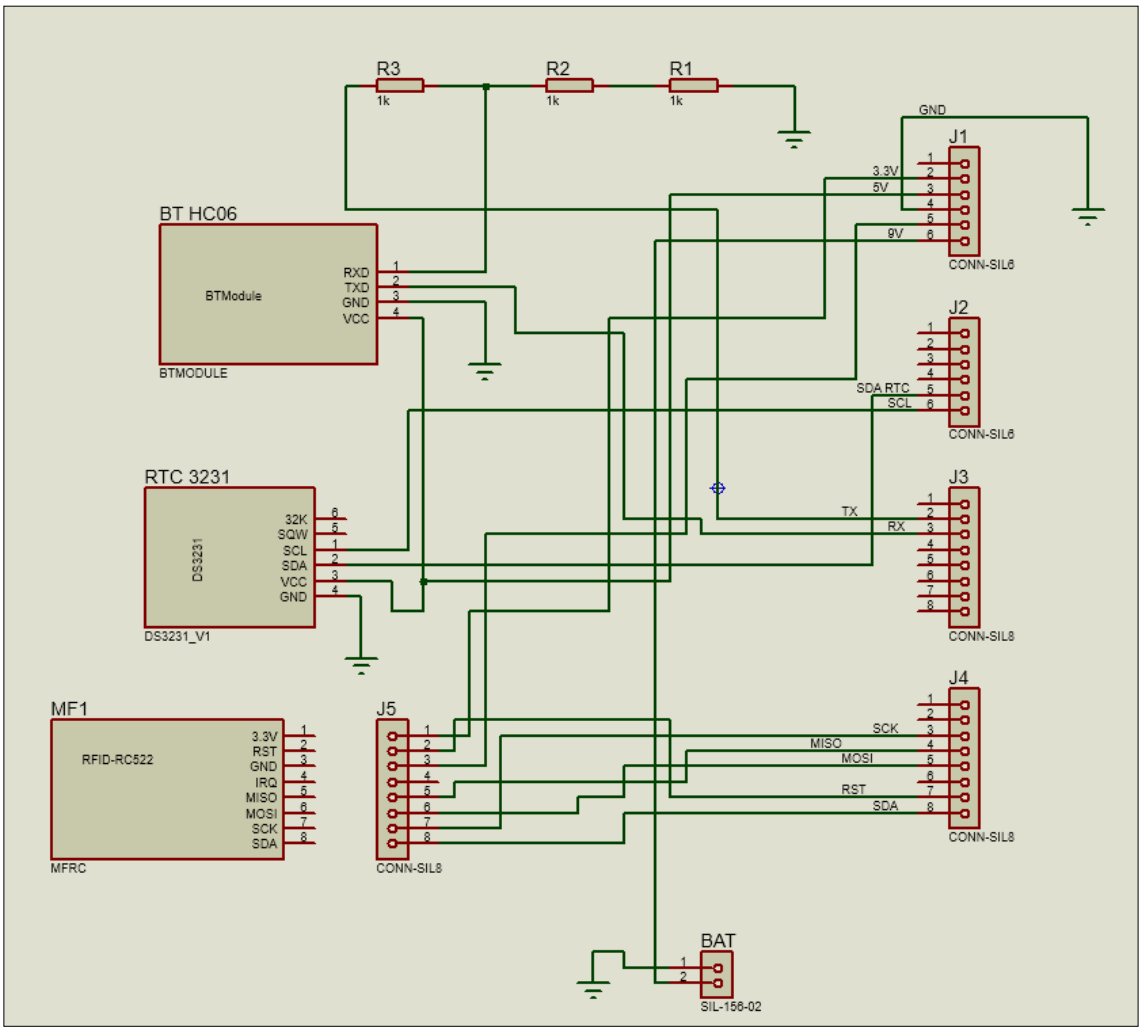


Figura 14: Vista del esquemático de la PCB con proteus. Fuente: propia

A continuación, se explican las elecciones de los paquetes de cada componente del proyecto. Para ello, los factores más importantes son la distancia los agujeros o pads y el diámetro de estos, ya que sin una correcta medida los pines de los componentes no se podrán introducir en los pads de la placa y soldarlos. Se procede a enumerar los diferentes paquetes de cada componente:

- **Resistencias:** se utiliza el paquete por defecto de Proteus, esto se debe a que la distancia entre los pads de las resistencias es indiferente.

- **Bloques de conectores:** se comprueban las diferentes tiras de pines disponibles en el departamento de electrónica y se miden para comprobar si concuerda con los incluidos en la librería de Arduino.

- **Bloque de 6 conectores:** se utilizan dos tiras de 6 pines macho en la parte anterior para realizar la soldadura y en l aparte superior 6 pines hembra para posibles conexiones extra.
- **Bloque de 8 conectores:** en el departamento de electrónica solo están disponible conectores de 10 pines, se corta un conector para eliminar dos pines y se utiliza para los conectores de 8 pines donde van conectados el MFRC522 y el HC-06.
- **Bloque de 2 conectores:** para la alimentación mediante la batería se utiliza un conector de 2 pines con bornes. Por defecto el Proteus tiene un paquete con 4 mm de separación entre los pines del conector, en el departamento has disponibles conectores de 2,5 mm y 5 mm de separación, por tanto, se tuvo que volver a repasar los pads del conector en la placa para que pudieran entrar los pines. Otro solución es crear un paquete nuevo para el conector separando los dos pines los 5 mm ncesarios.

- **Módulo Bluetooth HC-06:** se crea un paquete con las medidas del bluetooth para evitar superposiciones con otros componentes y se le incorporan 4 pines a una distancia de 1mm cada uno, la medida estándar para casi todos los componentes.

- **Módulo RTC DS3231.** Se crea un paquete con las medidas del DS3231 para evitar superposiciones y se le incorporan 6 pines para su soldadura.

- **Lector MFRC522:** en un principio se crea el paquete para el lector RFID con las medidas del componente e incorporando 8 pines. Pero finalmente, se utiliza una tira de 8 pines como las de los conectores, para conector el lector mediante una tira de cables macho/hembra. Esto es debido a que el lector debe estar separado de la placa para realizar la lectura de las tarjetas mas cómodamente.

Para el diseño de la PCB se deben tener en cuenta varios apartados:

- En la mayoría de placa con etapa de potencia es recomendable separarla de la electrónica digital, para evitar interferencias en las señales digitales de alta frecuencia. En este caso no existe etapa de potencia como tal pero si se coloca a un extremo de la placa el conector de la batería para evitar posibles interferencias.

- La colocación de los componentes es un factor importante, deben estar próximos al lugar de sus conexiones para evitar pistas largas, ya que esto puede desfavorecer al diseño de las pistas. También se debe tener en cuenta el espacio en la placa para evitar choques entre componentes.

- Los componentes deben colocarse en la **cara superior** o top, cara en la que sueldan principalmente los componentes. Se debe tener en cuenta que algunos componentes permiten la **soldadura en ambas caras**. Como es el

caso de los componentes de patillas largas, en este caso las resistencias. Por el contrario, el resto de componentes y tiras de pines, al ir pegados a la cara de componentes, solo permiten la **soldadura en la cara inferior**, donde se encuentran las patillas.

- La etapa más importante es la fase del trazado de las pistas. Se utiliza la herramienta Auto-router para realizar las pistas, esta función realiza las pistas automáticamente. En este proyecto al constar de pocos componentes solo se utiliza la cara *bottom* para las pistas.

- Si durante el auto-router queda alguna pista sin realizar, se utilizan las vías para hacer unos agujeros en la placa y mediante un cable soldado por la cara superior o bottom se realiza la pista faltante.

Después de estas premisas la placa a realizar es la siguiente:

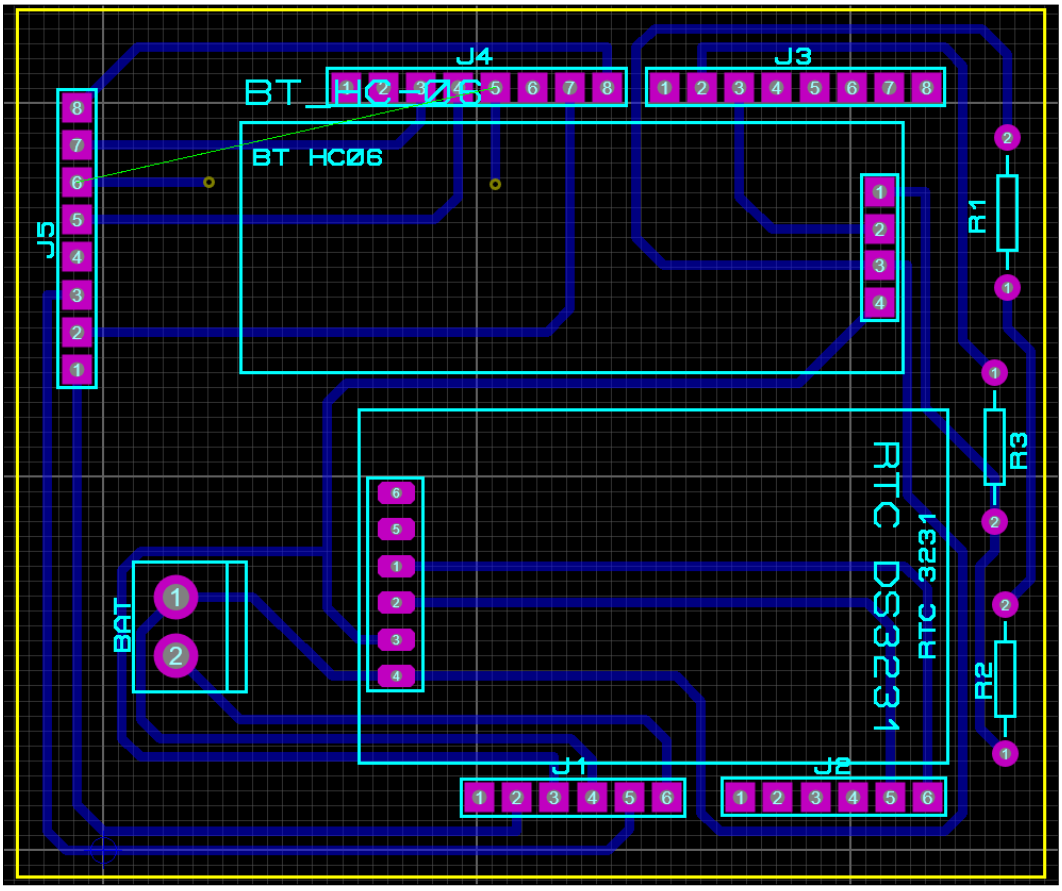


Figura 15: Placa PCB cara bottom en Proteus. Fuente: propia**.**

A continuación, el siguiente paso es la fabricación de la PCB, la cual sigue los siguientes pasos:

- Realización **fotolito.** Es una impresión en la cual se marcan las pistas en las que debe haber cobre. Se realiza imprimiendo en hojas de papel transparente el diseño de las pistas.
- La **insolación** es el procedimiento por el cual se aplica luz a una placa fotosensible. Al aplicar la luz, se elimina el cobre quedando grabado el diseño en la placa.
- **Corte de la placa** al tamaño deseado para el proyecto, sin cortar ninguna pista.
- **Taladrado de *pads* y vías**. Se debe tener en cuenta el tamaño de los pines de los componentes para la elección de la broca. Además para las vías se debe hacer el *pad* lo suficientemente grande para que entre un cable para poder soldarlo.
- **Soldadura de los componentes.** Se colocan los componentes en sus *pads* convenientes y se sueldan por la cara anterior o *bottom* ya que en este diseño solo tenemos componentes en esta cara. Para las resistencias y el cable de la vía se cortan a una distancia corta de la placa para evitar que elementos conductores puedan hacer contacto con lugares no deseados y poder producir un cortocircuito.
- El último paso es conectar los componentes, que no van soldados a la placa, a su respectiva tira de pines. Estos elementos son el bluetooth HC-06 Y DS3231, también el lector MFRC522 pero mediante los cables macho/ hembra.

La placa finalmente resultante es esta:

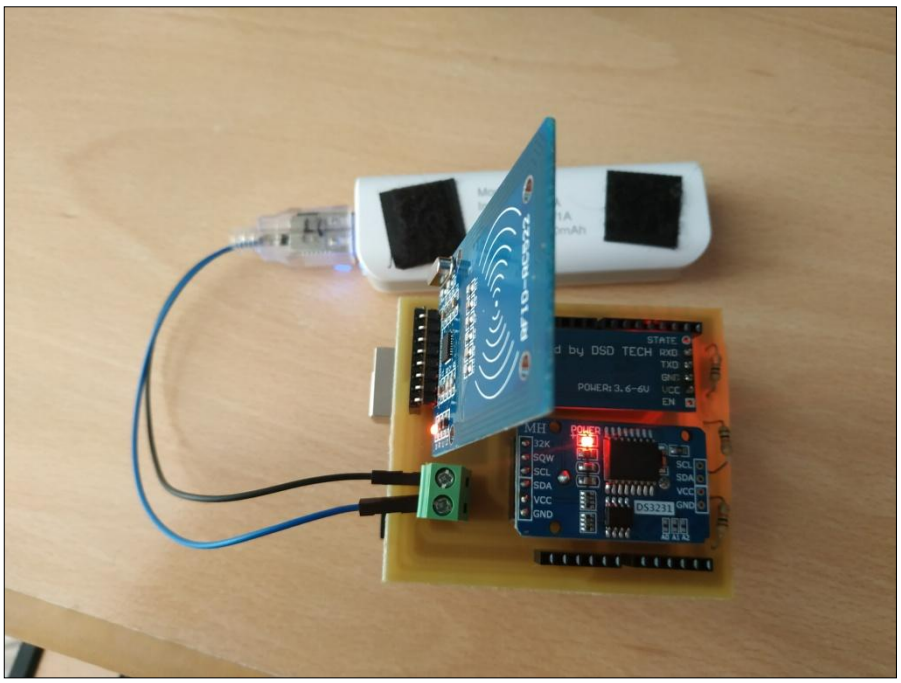


Figura 16: Placa PCB vista arriba. Fuente: propia.

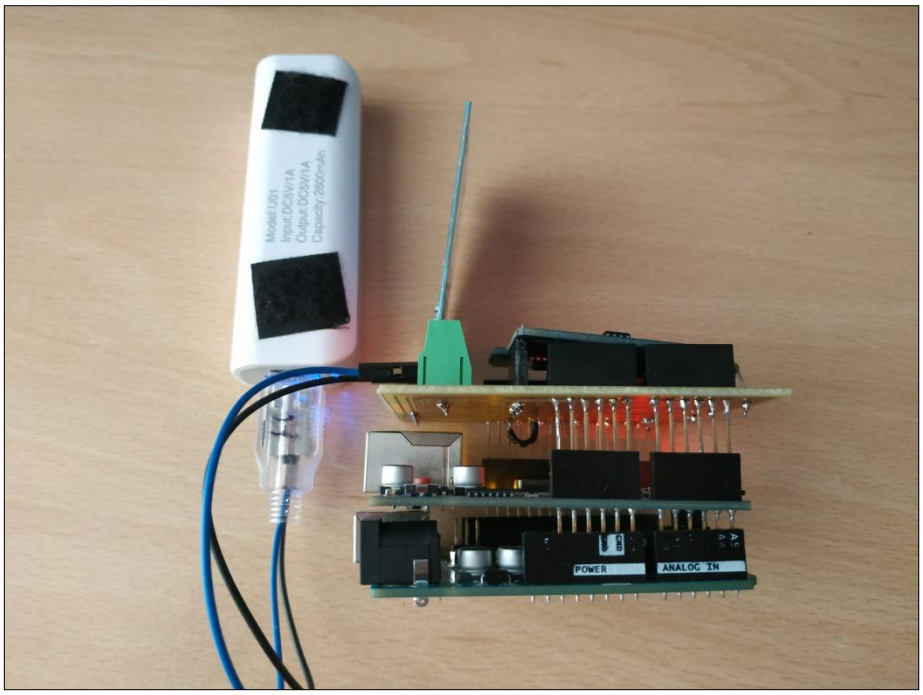


Figura 17: Placa PCB vista perfil. Fuente: propia.

Después de la fabricación de la placa, se encuentra un problema en el diseño. Concretamente en la distancia de los pines en el conector de alimentación para la batería que es de 4 mm, los conectores disponibles en el departamento de electrónica son e 2,5 mm o 5 mm. Para solucionar se volvió a taladrar los agujeros pero con 5 mm de separación.

2.4 Desarrollo de software

Esta parte es la más importante y costosa del proyecto, debido a que sin el software no se podría conseguir un buen funcionamiento de los componentes electrónicos. Para evitar fallos en el código, se recomienda ir programando poco a poco por partes, es decir, programar funciones o diferentes componentes y comprobar su funcionamiento por separado. Este proceso evita errores, también elude mantener un error durante toda la programación, esto es esencial porque una vez programado en su totalidad es muy complicado y tedioso encontrar un fallo en todo el código. Por tanto, la programación se realiza más sencilla y modula

2.4.1 Programación en Arduino

El software elegido para este proyecto es Arduino IDE el cual permite la creación del código para los diferentes Arduino, en este caso el Arduino Uno.

Antes de comenzar a programar se debe tener el software instalado y actualizado a la última versión disponible para ello se puede descargar en la página oficial de Arduino [18]. El primer paso es crear un nuevo programa o sketch, a continuación, seleccionar la placa que se utiliza. Para ello, se selecciona en “Herramientas -> Placa” la opción de “Arduino/Genuino Uno”.

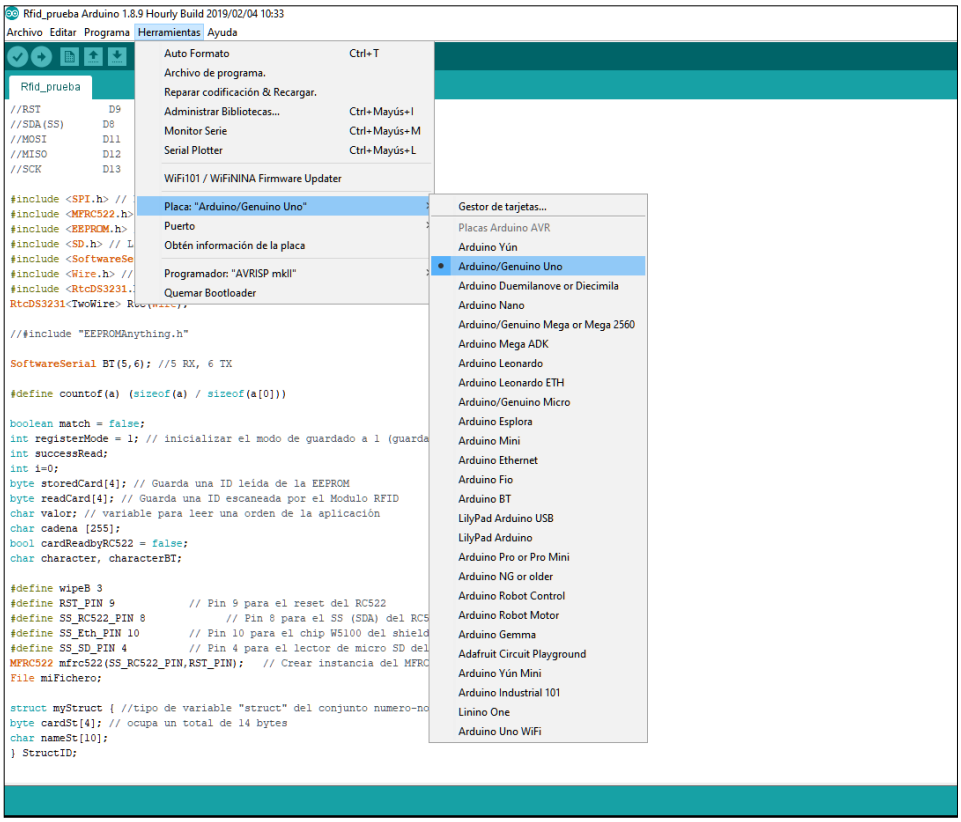


Figura 18: Selección de la placa Arduino. Fuente: propia.

Antes de comenzar a explicar la programación, se debe mencionar que el programa de este proyecto parte del código proporcionado por el alumno Pablo Roig Monzón. En código incorporaba la lectura de las tarjetas inteligentes mediante el lector RFID y su guardado en un archivo de texto en formato “.txt”. A partir de este código se ha trabajado para realizar las diferentes funciones que proyecto requiere. En un primer lugar, la programación se ha realizado y comprobado a través del monitor serial del Arduino IDE y con la conexión del cable USB del Arduino Uno. Una vez comprobado, se incorporó la sección de la App de Android mediante la conexión bluetooth del dispositivo móvil y el módulo HC-06 conectado al Arduino Uno.

A continuación, se presentan las diferentes funciones que incorpora el sistema del proyecto:

- **Conexión inalámbrica por bluetooth:** el sistema se conecta mediante el módulo HC-06 con el Arduino Uno. Este módulo es configurable mediante comandos AT como se ha comentado anteriormente, en este proyecto el nombre y contraseña del módulo bluetooth han sido modificados respecto a los valores por defecto. Concretamente a “FichajeBluuetooth” como nombre y “1936” como contraseña.

- **Lectura de las tarjetas universitarias inteligentes:** una vez conectado el bluetooth, la aplicación estará a la espera continua de detectar una tarjeta, la detección se produce mediante el sensor MFRC522.

- **Guardado de la ID de la TUI en la base de datos:** Cuando se haya detectado la tarjeta, se guardará la ID relacionada con dicha tarjeta en la base de datos del Arduino. Es decir, se guardará cada ID en un documento de texto en la microSD con la fecha y hora en la cual se ha realizado el fichaje.

- **Activación registro:** se puede activar o desactivar el registro de las tarjetas leidas por la aplicación. El principal objetivo de esta función es poder comprobar la correcta lectura de las tarjetas sin registrar las pruebas en la base de datos.

- **Introducir el nombre deseado al documento de la base de datos:** permite cambiar el nombre al documento del fichaje. Esto es debido a que así la aplicación se puede utilizar por diferentes personas sin necesidad de cambiar la programación para cada usuario que requiera utilizarla. Esta función es exclusiva de la aplicación Android.

- **Enviar registro al dispositivo móvil:** se envía el documento del registro de tarjeta al móvil mediante conexión bluetooth para su mejor visualización.

- **Borrar registro de tarjetas de la microSD:** se borra el documento con los registros para poder realizar un nuevo fichaje de otra clase o profesor.

A continuación, se procede a explicar las funciones anteriores dentro del código del programa Arduino IDE:

En primer lugar, se programa el código para el funcionamiento del módulo HC-06. Se debe configurar los pines donde irá conectado el módulo, esto se realiza antes del *"void setup()"* y el *"void loop()"* mediante la instrucción "***SoftwareSerial BT(5,6)".*** El Arduino Uno tiene como soporte incorporado para la comunicación serie los pines 0 y 1 (que son los que utiliza el puerto USB del Arduino para realizar la conexión con el PC), para poder utilizar el bluetooth mientras esté conectado el puerto USB, se utiliza al instrucción *SoftwareSerial* la cual permite la comunicación en serie en otros pines digitales del Arduino [19]. Además, en la función "*void setup(),* se debe inicializar la comunicación serie antes creada mediante la instrucción "***BT.begin(9600)".*** Se configura con una velocidad de 9600 bps (bits por segundo) que es la velocidad por defecto para el puerto serie. Una vez realizado esto, es trabajo del dispositivo móvil conectarse al módulo HC-06.

La lectura de la ID de las tarjetas se realiza mediante la utilización de dos funciones definidas por el usuario, las cuales pueden llamarse múltiples veces, utilizar parámetros de entrada o entregar parámetros de salida [20]. Estas funciones son "***int getID()***"y "***void leadingZeros(int byteLeido)"***. La primera guarda la ID detectada en

una variable y la segunda añade un 0 a la izquierda si solo tiene un digito hexadecimal en uno de los bytes de la variable donde se guarda la ID.

Para el guardado de las IDs de las tarjetas se utilizan 3 funciones de usuario, las cuales son ***void guardarRegistro(byte TarjetaAdmitida[4]), void activarSD()*** y ***void desactivarSD().*** La primera función abre y escribe en un archivo las IDS de las tarjetas leídas, también escribe la fecha y hora del fichaje de la tarjeta junto a su ID. Las otras dos funciones son para activar y desactivar la SD.

Las funciones restantes se activan o desactivan mediante la App Android que dependiendo del mensaje que envié realizará una función o otra. Estos mensajes simplemente cambian el estado de una variable que mediante la instrucción ***if()***, permite realizar la función deseada.

Este apartado se encuentra explicado en más profundidad en el Anexo I, en el cual se encuentra todo el código del Arduino IDE comentado debidamente para su comprensión.

2.4.2 Desarrollo aplicación Android con AppInventor2

Appinventor2 es una plataforma basada en un entorno web para desarrollar aplicaciones Android. Este software esta desarrollado por el MIT (*Massachussetts Institute of Technology)* [21].

Es una aplicación en la cual se programa mediante una interfaz de bloques que es visual y fácil de entender. Con ella se realizará la App Android para realizar el fichaje y la administración de esta. En la interfaz del programa se encuentran dos vistas diferenciadas: el *Designer View* y el *Blocks View.*

La vista del *Designer View* es la siguiente:

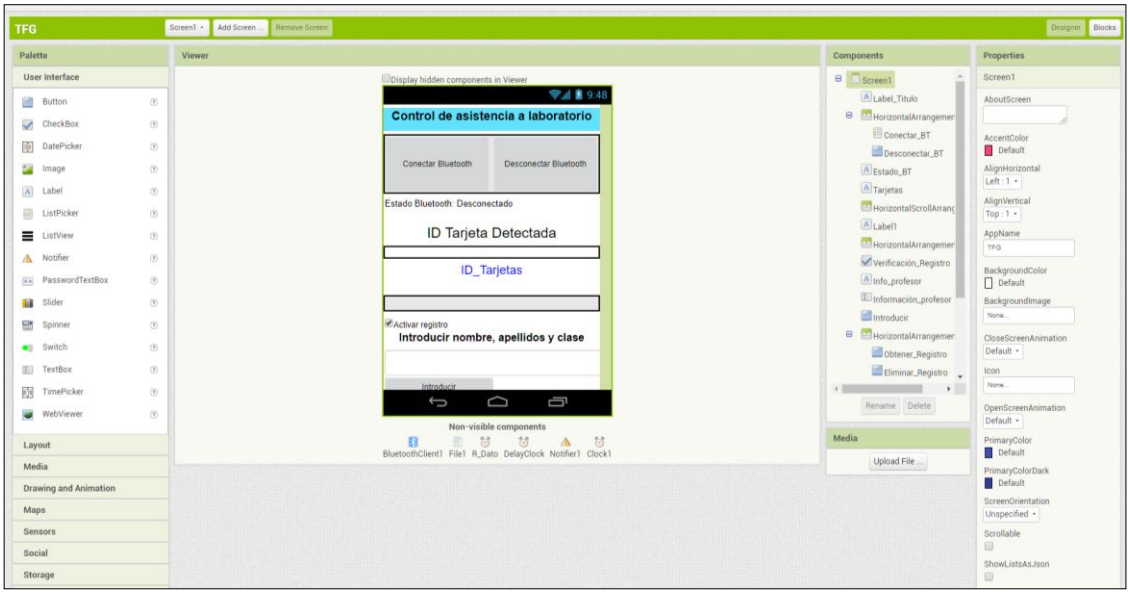


Figura 19: vista del *Designer View* de Appinventor2. Fuente: *http://ai2.Appinventor.mit.edu/*

La vista del *Designer view* tiene 4 partes bien diferenciadas. **Paleta de componentes** (*palette*), **Visor** (**Viewer**), **Componentes** (*components*) y **propiedades** (*properties*).

- La paleta se encuentra en la parte izquierda, en ella se encuentran todos los elementos disponibles, clasificados en diferentes categorías, para poder realizar distintas funciones (existen elementos visibles y no visibles).
- En el visor, que se encuentra en el centro, es muestra el resultado gráfico de la interfaz de usuario que aparecerá en la pantalla de nuestro dispositivo móvil una vez instalada la App.
- Al lado del visor a la derecha, se encuentra la lista de componentes que se han agregado a la aplicación, en la cual se puede cambiar de nombre o eliminar el elemento elegido.
- A la derecha del todo, se encuentran las propiedades de cada elemento una vez se selecciona. Por ejemplo, podemos modificar el tipo, el tamaño y color de la fuente. También cambiar el texto que se muestra o si es visible o no, aparte de muchas más propiedades dependiendo del elemento que sea.

Para comenzar a crear el código de la App, se debe entrar en el apartado de *Blocks,* botón que se encuentra arriba a la derecha. Este software nos permite programar en un lenguaje de bloques juntando diferentes funciones o bloques. Estos bloques se encuentran en la parte izquierda en la vista de *Blocks,* cada elemento añadido en la vista del *Designer* tendrá sus diferentes bloques y funciones. Las diferentes funciones que se encuentran ordenadas por categorías son: control, lógica, matemáticas, texto, listas, colores, variables y procedimientos.

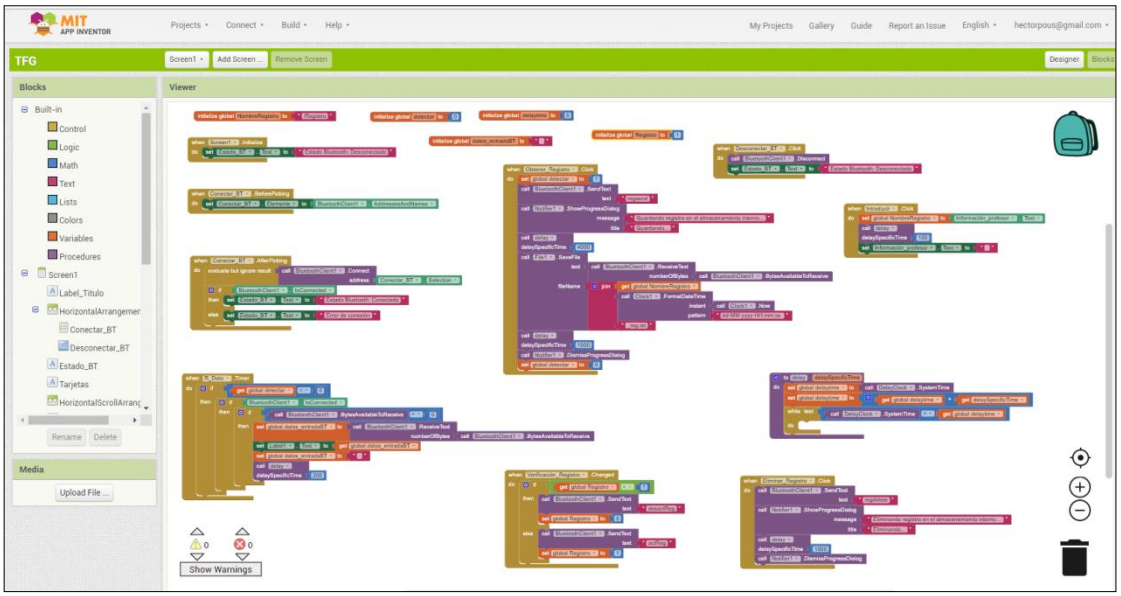


Figura 20: Vista del Blocks View de la Appinventor2. Fuente: *http://ai2.Appinventor.mit.edu/*

A continuación, se procede a explicar la programación de la aplicación Android para el sistema, para ello se utilizarán **diagramas de flujo.** Un diagrama de flujo es una representación gráfica de un algoritmo o proceso, es utilizado en diferentes disciplinas como programación, economía, ingeniería o psicología [22].

Primero se muestra la pantalla principal de la aplicación para Android en el dispositivo móvil:

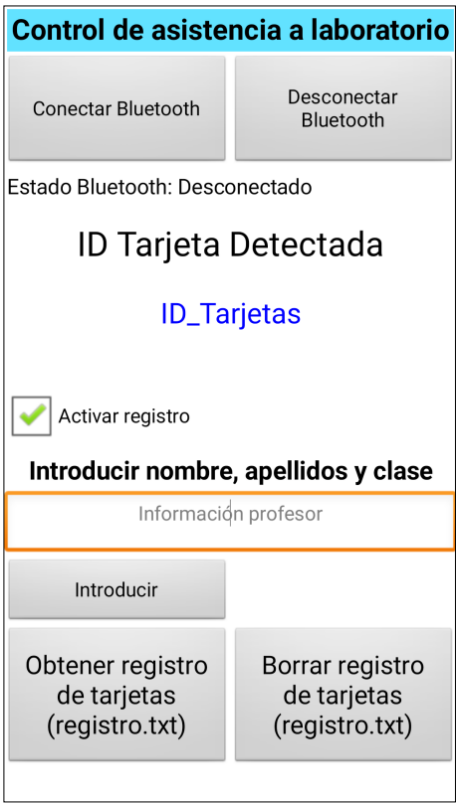


Figura 21: Vista de la App en el dispositivo móvil. Fuente: propia.

A continuación se expondrá el funcionamiento de los botones iniciales de la aplicación **ConectarBluetooth** y **DesconectarBluetooth.**

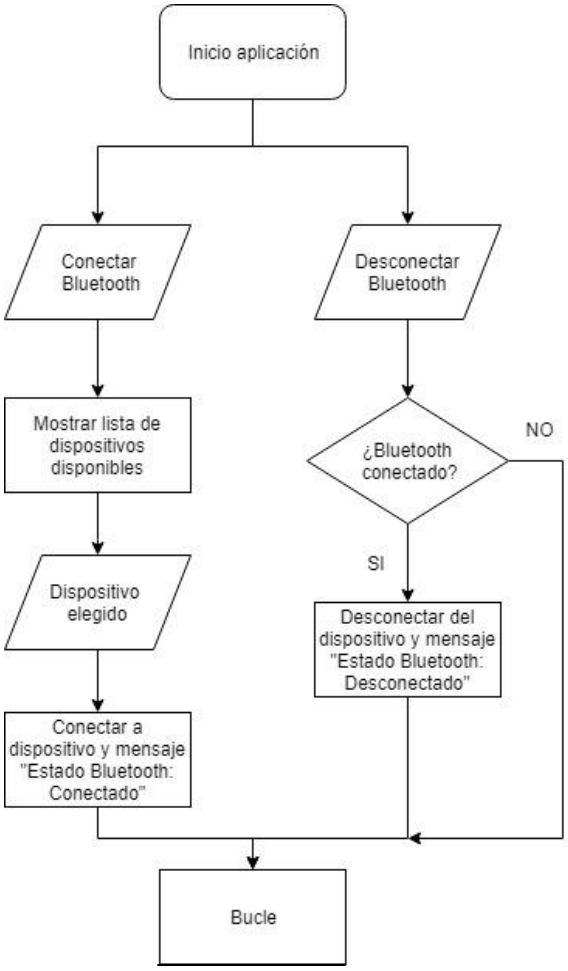


Figura 22: Diagrama de flujo de botones de Blluetooth. Fuente: propia.

En el diagrama se utiliza un bloque con la función bucle porque el sistema permanece a la espera de una acción del usuario, es decir, está a la espera de recibir una entrada para realizar las funciones correspondientes, estas entradas pueden ser tanta detección de tarjetas o acciones realizadas por la interfaz de la App Andorid. Para poder explicar el programa por funciones para su mayor comprensión se utilizará este bloque.

Al pulsar el botón **ConectarBluetooth**, aparece una lista de dispositivos vinculados al dispositivo móvil. En esta lista se debe seleccionar el nombre del módulo bluetooth del sistema. Se debe tener en cuenta que para que el dispositivo aparezca en esta lista primero ha de ser vinculado con el Smartphone, más adelante se explicará paso por paso como realizar esta tarea.

Figura 23: Lista de dispositivos vinculados al Móvil. Fuente: propia

A continuación, se presenta el bucle para la lectura de las tarjetas inteligentes:

Figura 24: Diagrama de flujo de la detección de las tarjetas. Fuente: propia.

El sistema está constantemente a la espera de la detección de una nueva tarjeta, menos durante los periodos de envió y eliminación del archivo del registro y el cambio de nombre del archivo. Para evitar posibles interferencias o errores se realiza un delay (espera sin realizar ninguna instrucción o acción) después de cada detección. También se tratará de explicar la función de delay.

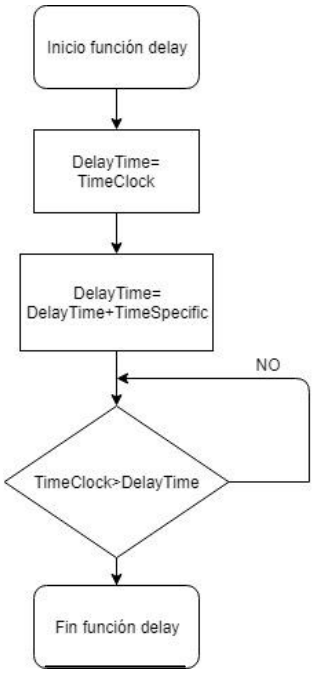


Figura 25: Diagrama de flujo de la función delay. Fuente: propia.

El *specific time* del anterior diagrama de flujo es un tiempo configurado como entrada para la función *delay* para que diferentes aplicaciones puedan tener diferentes valores.

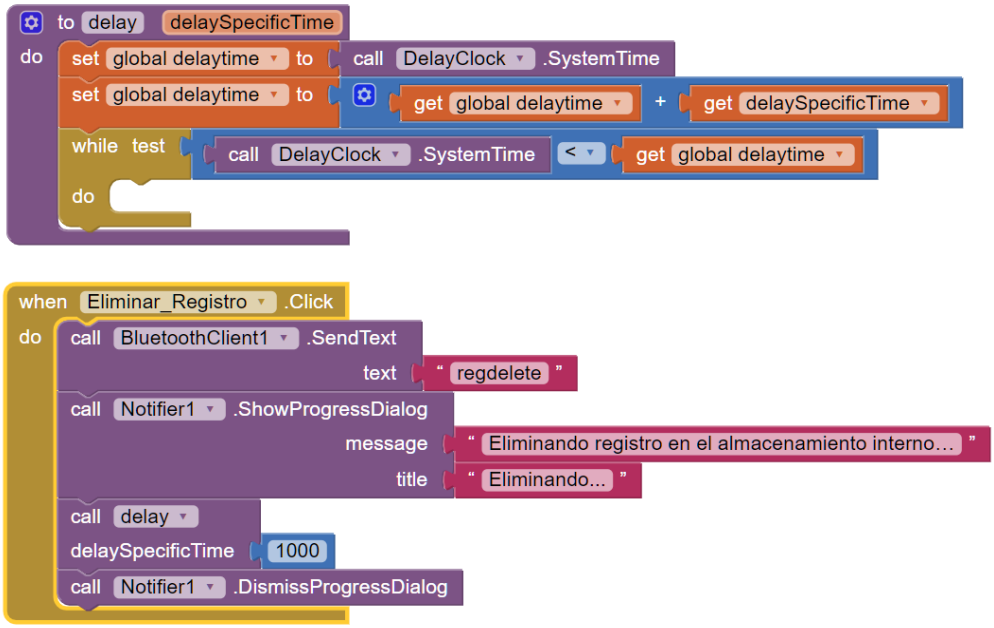


Figura 26: Programación delay y ejemplo en la aplicación. Fuente: propia.

A continuación, se explican las funciones de **activar registro** e **introducir información:**

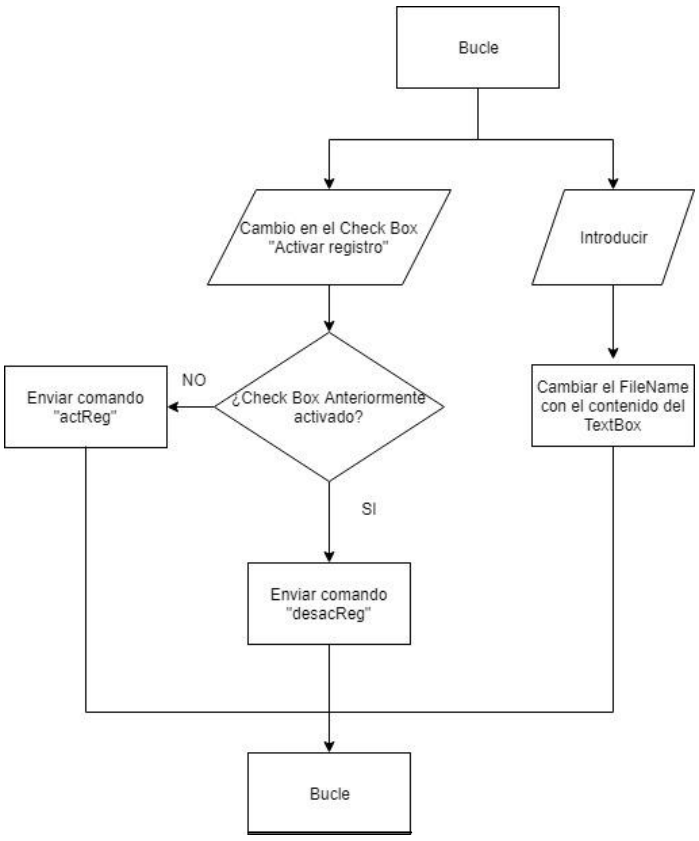


Figura 27. Diagrama de flujo de las funciones activar registro y introducir información. Fuente: propia

Por último, se presentan las dos funciones restantes de la aplicación que son **Obtener registro de tarjetas (registro.txt)** y **Borrar registro de tarjetas (registro.txt):**

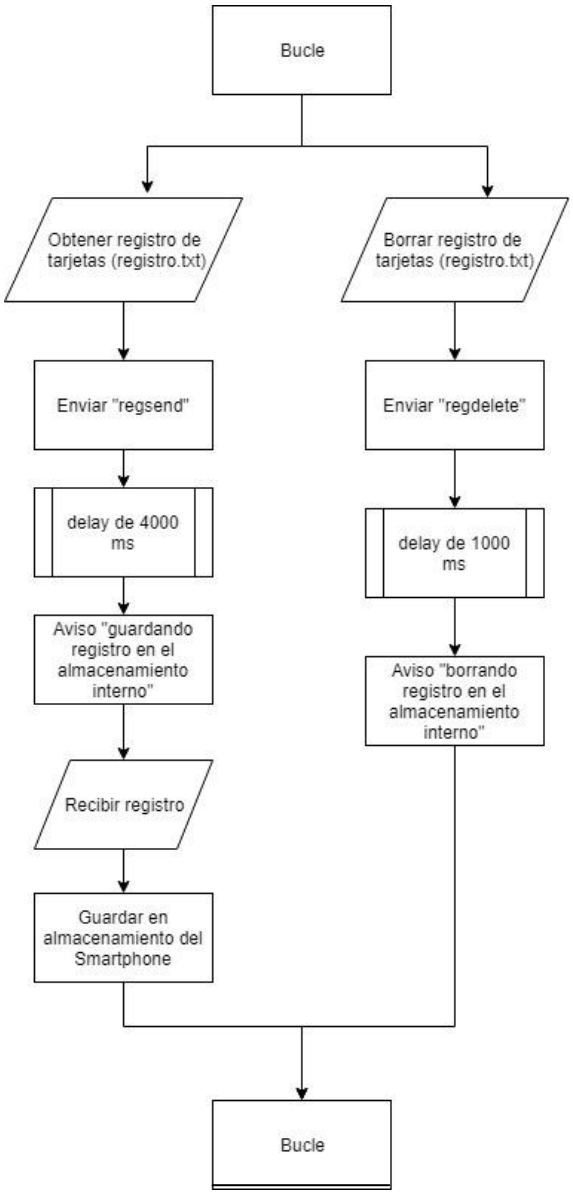


Figura 28: Diagrama de flujo de las funciones obtener y borrar registro de tarjetas. Fuente: propia.

La programación de bloques completa se encuentra en el apartado Anexo 2, en la cual están las funciones de cada uno de los diagramas de flujo expuestos.

3. Manual de usuario

El manual de usuario es un documento que incluye los aspectos fundamentales de una materia. Es decir, una guía que ayuda a entender el funcionamiento de programa o proceso.

A continuación se explican los pasos que debe realizar el usuario para utilizar la aplicación de forma adecuada:

- **Vincular** el dispositivo móvil al módulo bluetooth HC-06.
- **Conectarse** al módulo HC-06 mediante la App Android.
- **Activar** o **desactivar** el registro.
- **Introducir** nombre, apellidos y aula del usuario que va a realizar el fichaje.
- **Borrar** registro previo, si hubiera.
- **Enviar registro** al Smartphone.

Para vincular el dispositivo móvil al módulo HC-06 se debe seguir los siguientes pasos:

- Ir a la aplicación bluetooth del móvil y activar esta función del Smartphone.
- En dispositivos disponibles, seleccionar dispositivos raramente usados.
- Una vez dentro, seleccionar “FichajeBluetooth” e introducir la contraseña “1936”.

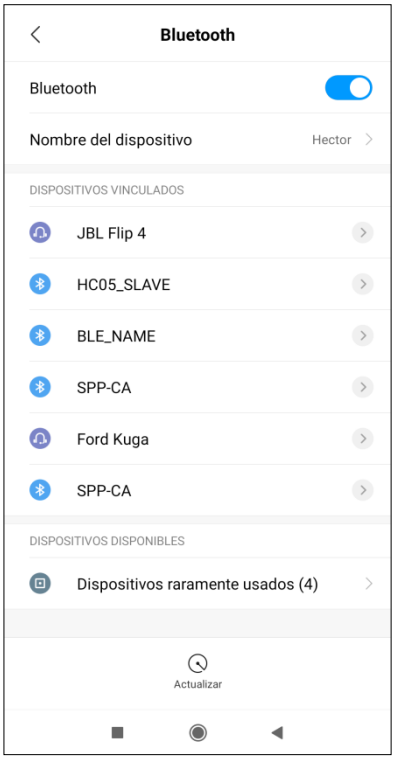


Figura 29: Dispositivos vinculados con el dispositivo móvil. Fuente: propia.

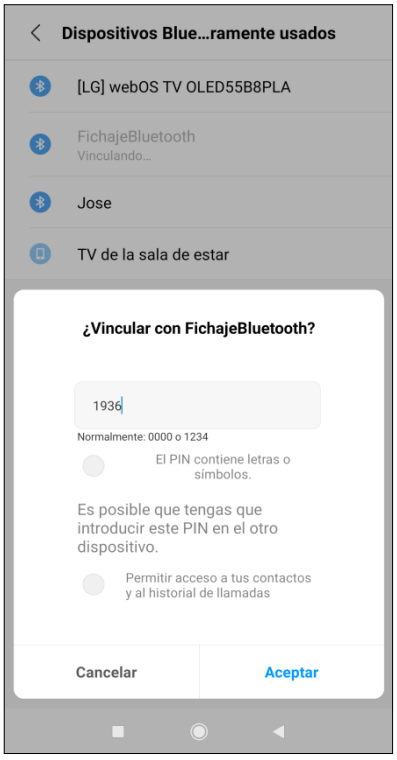


Figura 30: Ventana de introducción del pin del módulo HC-06. Fuente: propia.

El siguiente paso será conectar el dispositivo móvil al módulo bluetooth HC-06 del sistema para ello se debe seleccionar el botón **Conectar Bluetooth,** una vez aparezca la lista de dispositivos vinculados, seleccionar **FichajeBluetooth.** Para facilitar la comprensión se resalta en rojo los componentes a seleccionar.

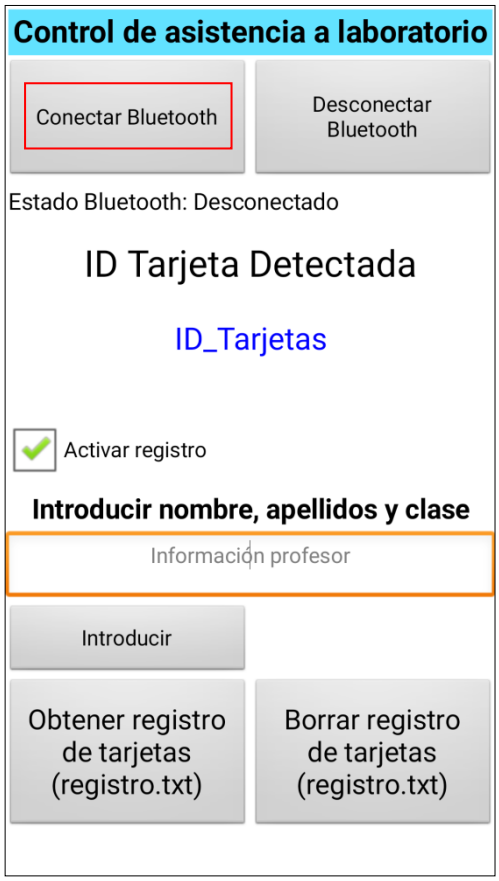


Figura 31: Conexión bluetooth con el módulo HC-06. Fuente: propia.

Figura 32: Selección del Módulo bluetooth HC-06. Fuente: propia.

A continuación se presenta como **activar y desactiva el registro**, esta función se puede realizar en cualquier momento. También durante el mismo fichaje de los alumnos, esta función se utiliza para comprobar la lectura de las tarjetas sin guardar su ID en la base de datos o simplemente si no se quiere realizar el registro. En este apartado se utiliza también para ver la visualización de una tarjeta leída en la pantalla.

Figura 34: Registro activado en App Android. Fuente: propia.

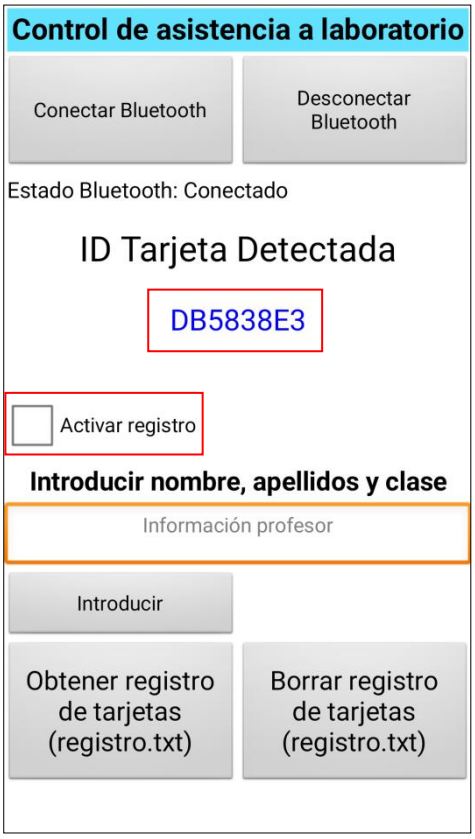


Figura 33: Desactivación de registro y muestra de ID de tarjeta detectada. Fuente: propia.

A continuación se explica cómo cambiar el nombre del archivo del registro. En el lenguaje de bloques de Appinventor2 para poder introducir un nombre al archivo, antes de dicho nombre se debe escribir una barra diagonal como la siguiente, "***/***". Se intentó, mediante el código de bloques, añadir esta barra siempre en la variable que se utiliza para cambiar el nombre pero con las herramientas de la aplicación no se pudo realizar satisfactoriamente, por lo tanto, el usuario deberá introducir esta barra al principio del nombre que quiera introducir al archivo. Para introducir el nombre se deberá seleccionar el *textbox* y emergerá el teclado del Smartphone para escribir dicho nombre.

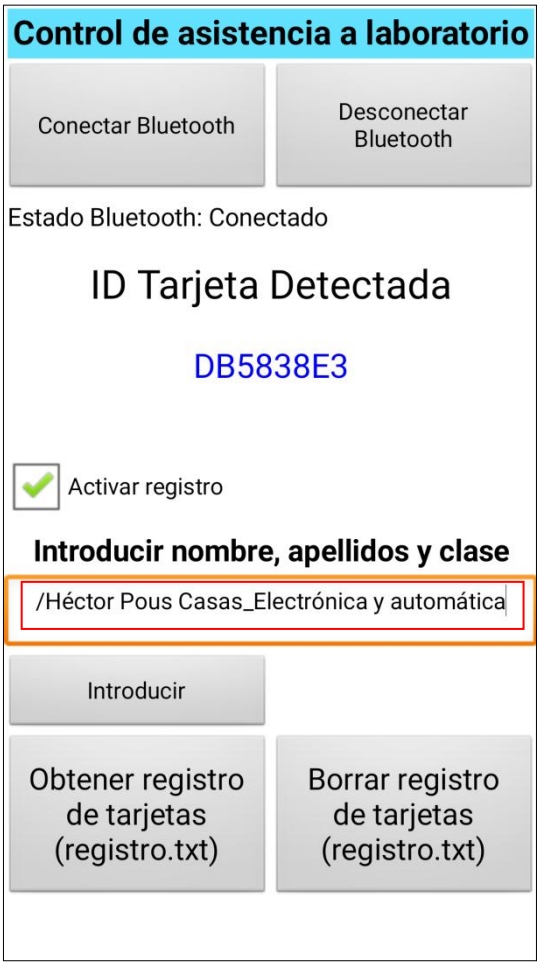


Figura 35: Introducción del nombre del archivo del registro mediante App Android. Fuente: propia.

A continuación, se exponen las dos últimas funciones del sistema que son **Obtener el registro de tarjetas** y **Borrar el registro de tarjetas.** Para ello, se debe seleccionar el botón correspondiente a cada función y esperar el tiempo de la transmisión o borrado del archivo, esto puede durar unos pocos segundos. Mientras se esté realizando estas funciones los botones quedarán iluminados para indicar que se está en proceso.

Figura 36: Función obtener registro de la App Android. Fuente: propia.

Figura 37: Función borrar registro de la App Android. Fuente: propia.

Una vez enviado el archivo para poder visualizarlo en el Smartphone, se debe buscar en el almacenamiento interno del mismo. Para ello, se debe entrar en el gestor de archivos del dispositivo, ir al almacenamiento interno, bajar por debajo de las carpetas del dispositivo móvil y buscar el archivo correspondiente. A continuación se muestra paso a paso como buscar el archivo de forma gráfica. Hay que tener en cuenta que el Smartphone utilizado para esta demostración es un Xaomi Redmi Note 5, esta demostración puede cambiar dependiendo de casa de dispositivo móvil pero será equiparable.

Figura 38: Selección del gestor de archivos del Smartphone. Fuente: propia.

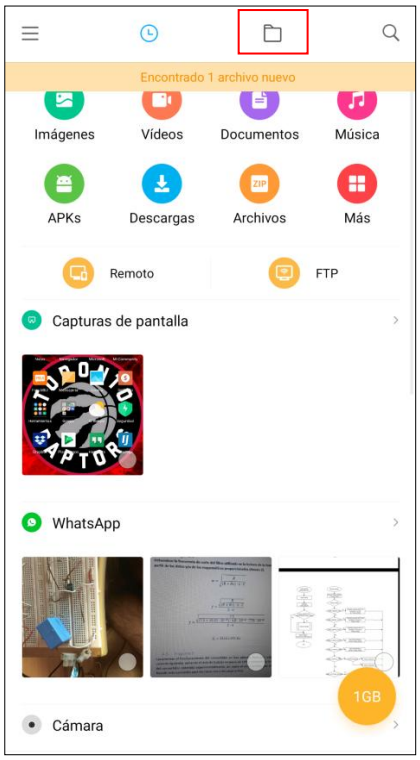


Figura 39: Selección del almacenamiento interno del Smartphone. Fuente: propia.

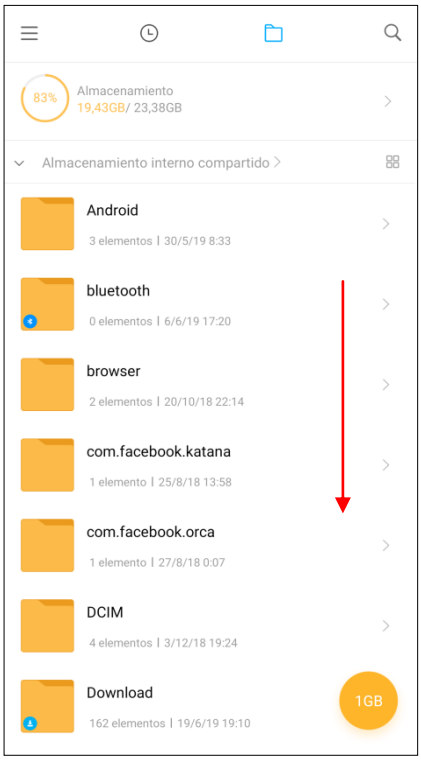


Figura 41: Búsqueda archivo del registro en el almacenamiento interno. Fuente: propia.

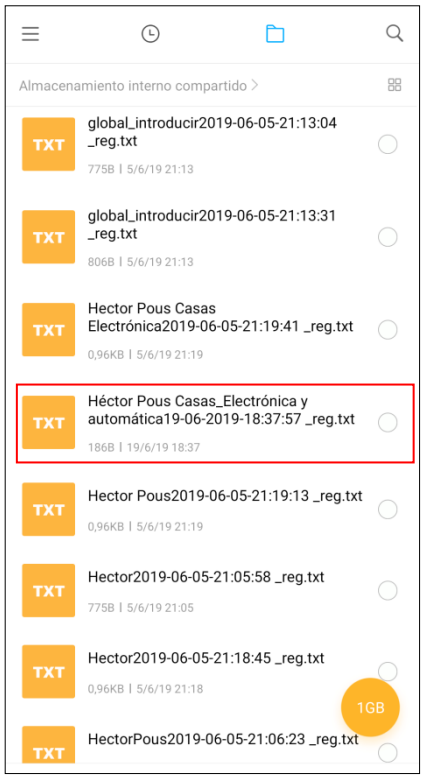


Figura 40: Archivo del registro encontrado con el nombre correspondiente. Fuente: propia.

Por último, si la aplicación muestra un error repetido en conectarse al Módulo bluetooth y se aprecio todo correcto, es posible que sea problema del Smartphone. Para solucionarlo, se debe cerrar totalmente la App y volver a iniciar. Cuando ocurre este problema se deberá también desconectar la alimentación del Arduino (batería) y volver a conectarla, porque al producirse este error el módulo Bluetooth HC-06 está constantemente intentado conectarse con un dispositivo que ya no esta disponible es ese momento.

4. Líneas futuras

En este apartado se trataran posibles mejoras que se puedan aplicar a este sistema tanto en la parte de programación de Arduino, Appinventor o en el diseño y montaje de la PCB. También se expondrán diferentes fallos del sistema tanto en el software como en la parte hardware de la placa, para poder corregirlas en futuros proyectos. Además se tratarán alternativas o mejoras del proyecto a partir del mismo.

Las mejoras en el software son las más importantes y más interesantes ya que tienen un mayor potencial para mejorar la eficiencia del sistema. Esto es posible sin modificar la placa PCB realizada porque el Arduino Uno todavía tiene disponible el puerto USB para su conexión. Primero se expondrán las posibles mejoras en la App Android:

- **Mejorar la interfaz de usuario de la App y la programación de algunas funciones del sistema:** se podría mejorar la estética de la aplicación para hacerla más atractiva y manejable para el usuario. Además incorporar avisos cuando el sistema este realizando las diferentes funciones.
- **Incorporar nuevas funciones al sistema:** en este proyecto se utiliza las tarjetas universitarias inteligentes (TUI) que todo alumno o persona asociada a la UPV debe tener. Pero si resulta que no dispone de ella por cualquier motivo, se podría incluir la funcionalidad de pasar el fichaje mediante la lectura de la huella dactilar. Aunque para realizar esta función se debería de cambiar de entorno de programación ya que el Appinvetor2 tiene sus limitaciones. La mejor alternativa es programar en JAVA o XML debido a que este ámbito se aleja de los conocimientos adquiridos en el grado, aunque existen otro software para la programación en Android sin la necesidad de dominar los lenguajes antes mencionados. Por ejemplo, **Android studio, Basic4Android** o **Mono para Android** [23], estos programas utilizan un lenguaje gráfico o C# los cuales serian interesantes para la realización de este tipo de proyecto, más otentes que Appinventor2 pero con menor complejidad o menos abstractos que los lenguajes JAVA o XML.

A continuación se tratarán las mejoras posibles en la programación de Arduino IDE:

- **Optimizar el código:** Se puede disminuir la utilización de la memoria Flash del Arduino ya que el código ocupa el 81% de esta. Esto se puede conseguir mediante la reducción de las líneas de código, no utilizar bytes innecesarios, tener en cuenta el tamaño de los buffers, utilización de la EEPROM, etc.

- **Utilización de *timers:*** El código de Arduino del sistema utiliza una función interna del Arduino IDE llamada delay() la cual permite generar una espera de tiempo. El inconveniente de esta función es que mientras se ejecuta el procesador no realiza ninguna otra tarea, simplemente queda bloqueado. Esto una pérdida de la eficiencia considerable, con los *timers* el procesador podría realizar otras tareas durante el tiempo de espera. Esto mejoraría considerablemente la eficiencia y tiempos del sistema, esto es importante ya que se pretende ampliar el sistema en futuros proyectos y esto implica mayor cantidad de código.

Es importante mencionar algunos errores menores del sistema para su futura corrección:

- En el diseño de la placa PCB para el circuito y los componentes del sistema se encuentra un fallo en las medidas del conector de alimentación donde va conectada la batería. Los conectores estándar tienen las patillas separadas por 2,5 mm o 5 mm, en el caso de este diseño esta distancia es de 4 mm. Esto se comprobó una vez fabricada la placa a la hora de soldar el componente, se subsanó realizando los agujeros más grandes para poder soldar el conector.
- Una vez montada y soldada la PCB, se observó que la tira de pines del sensor RFID hace contacto con el conector Ethernet del módulo de la microSD, este contacto puede causar interferencias o cortocircuitos. Para solucionarlo, se debe cambiar de lugar al conector del sensor en el diseño de Proteus o cambiar la orientación de la PCB. Este contacto no es continuo, es decir, si no se meten las patillas de la PCB del todo no hace contacto, por seguridad se aísla el conector con una espuma aislante.
- En la programación de la App de Appinventor2 se realiza un aviso cuando se envía o borra el archivo del registro. Se ha comprobado el código y supuestamente es correcto pero el aviso sigue sin aparecer. Para futuros proyectos se debería revisar porque sucede.

Este proyecto se implantará se forma experimental en los laboratorios del edifico de la ETSID. Si transcurre correctamente, se implantará en más zonas de la Universidad Politècnica de València. También se puede mejorar en la fabricación de la PCB utilizando componentes SMD, es un factor a estudiar ya que supondría una mejora considerable del espacio y de la versatilidad del sistema. Además herramientas como la pintura protectora para evitar las oxidaciones, ya que el cobre se encuentra en este sistema al aire, o el serigrafiado para mejorar la visualización de la placa. Aunque este sistema incorpora pocos elementos, en futuros proyectos habrá más cantidad.

Por lo tanto, se debería tener en cuenta estas mejoras y correcciones de errores para los futuros proyectos.

5. Presupuesto

En esta sección se presenta el presupuesto de proyecto, contando con el coste de diseño, desarrollo de sistema y coste de los materiales.

5.1. Costes de desarrollo y mano de obra

El salario atribuido a Ingeniero en Electrónica Industrial y Automática (en categoría de Técnico) ha sido consultado en el Convenio Colectivo para la Industria y Servicios del Metal de Valencia.

Concepto	Importe
Salario mensual grupo 2 (Técnicos)	1647,32 €
Cotización a la Seguridad Social (32,6%)	537,026 €
TOTAL MENSUAL	2.184,226 €
TOTAL ANUAL	26.210,712 €
Por jornada mensual (22 días):	99,283 €
Por hora:	12,41 €

El tiempo estimado para la realización del proyecto es de 208 h, por tano el presupuesto de mano de obra es:

Descripción	Precio(€/h)	Tiempo (h)	Ud	Importe (€)
Ingeniero en Electrónica Industrial y Automática	12,41	208	h	2581,28

5.2. Costes de materiales

En este apartado se aportan los costes de los materiales y componentes utilizados para el proyecto. Los precios indicados son sin IVA.

Componentes			
Descripción	Cantidad (ud.)	Precio unitario (€)	Importe(€)
Arduino Uno R3	1	8,29	8,29
Shield Ethernet	1	12,19	12,19
Módulo Bluetooth HC-06	1	4,99	4,99
Módulo RTC DS3231	1	2,25	2,25
Lector RFID MFRC522	1	7,99	7,99
Resistencia 1 kΩ	3	0,02	0,06
Tira de pines hembra	7	0,12	0,84
Conector alimentación batería	1	0,93	0,93
Placa de circuito impreso	1	4,55	4,55
Fabricación de placa de circuito impreso	1	16,53	16,53
Estaño para soldadura (20g)	1	0,047	0,94
		TOTAL:	**59,56 €**

Tabla 5: Costes de componentes electrónicos

5.3. Valoración del presupuesto

Por último, se presenta la siguiente tabla con el presupuesto final:

Descripción	Importe
Costes de desarrollo y mano de obra	2.581,28 €
Costes de materiales	59,56 €
TOTAL (Sin IVA)	2.640,84 €
IVA (21%)	554,57 €
PRESUPUESTO TOTAL	3.195,416 €

Tabla 6: Presupuesto total del proyecto.

El presupuesto total del proyecto asciende a un total de:

TRES MIL CIENTO NOVENTA Y CINCO CON CUATRO

6. Conclusión

Al desarrollar el proyecto se ha conseguido cumplir los objetivos propuestos: se ha creado un sistema para la realización de la asistencia a aulas o laboratorios. El fichaje se realiza bajo el sustento del software Arduino y la App Android, las cuales interactúan de forma satisfactoria y realizan el fichaje adecuadamente. El archivo del fichaje se puede obtener tanto por la vía del dispositivo móvil como por la microSD de la *shield* Ethernet de Arduino.

Durante el proyecto han surgido algunos contratiempos que se han ido corrigiendo poco a poco. Esto ha ayudado a ganar experiencia sobre todo en el campo de la programación la cual requiere de esfuerzo e implicación, tanto en Arduino IDE como en la programación de la App. También se ha conseguido conocimientos prácticos de diseño y fabricación de placas PCB.

Finalmente, se ha conseguido un sistema funcional que se podrá poner a prueba en las diferentes aulas de la ETSID o laboratorios. Este será el comprobante para verificar el proyecto y más adelante implantarlo en diferentes lugares de la Universidad, para poder trasladarlo a otros lugares antes se deberá realizar las correcciones antes mencionadas.

Este trabajo ha puesto todos los conocimientos de la carrera en juego para realizar el proyecto, tanto la programación en Arduino, la programación de la App Android, la electrónica utilizada, el diseño y fabricación de la placa PCB. Otros aspectos muy otiles del proyecto son más los ambientados en el mundo laboral y profesional, como enfrentarse a contratiempos, a plazos de entrega, problemas reales en el montaje, etc. Esta experiencia es muy valiosa y útil para un recién titulado en ingeniería.

El coste del proyecto es considerablemente bajo, este sistema es muy válido como alternativa para el fichaje de cualquier ámbito tanto académico como profesional, con más motivo con la nueva ley (mayo de 2019) que obliga a las empresas a realizar el fichaje de todos sus trabajadores. Aunque en futuros proyectos este presupuesto podría ser todavía más económico, tanto por materiales más baratos o reducción del tiempo de los plazos.

En conclusión, el proyecto ha sido un gran reto para aplicar los conocimientos académicos y mejorar como profesional en el mundo de la electrónica. Por último, sin duda la parte más gratificante del proyecto y de esta profesión es ver como un trabajo realizado por ti mismo después de tanto tiempo, resulta cumplido.

7. Bibliografía

[1] Google, <<Google,>> [En línea] Available: https://www.google.com/search?rlz=1C1CHBF_esES835ES835&ei=Uf73XKLNEaeelwSx6pXAAg&q=registro+definicion&oq=registro+definicion&gs_l=psy-ab.3..0i70i249j0l8.4733.6885..7038...1.0..0.97.854.12.....0....1..gws-wiz.....0i71j0i67j0i67i70i249j0i203j0i20i263i70i249j0i20i263.cTWL69xFOc0 [Último acceso: 9 06 2019]

[2] El País, <<El Pais,>> [En línea] Available: https://elpais.com/economia/2019/05/10/actualidad/1557485213_287287.html [Último acceso: 10 06 2019]

[3] Wikipedia, <<Wikipedia.org,>> [En línea] Available: https://en.wikipedia.org/wiki/Microcontroller

[Último acceso: 10 06 2019]

[4] programarfacil, <<programarfacil,>> [En línea]. Available: https://programarfacil.com/blog/comprar-una-placa-original-o-una-placa-copia-de-Arduino/ [Último acceso: 10 06 2019]

[5] UPV, «Universitat Politècncia de València,» [En linea]. Available: https://www.upv.es/tui/index-es.html [Último acceso: 11 06 2019].

[6] Wikipedia, «Wikipedia.org,» [En linea]. Available: https://es.wikipedia.org/wiki/Mifare.

[Último acceso: 11 06 2019].

[7] Parallax, «Parallax,» 02 2006. [En linea]. Available: https://forums.parallax.com/discussion/83298/difference-between-ds1307-and-ds1302.

[Último acceso: 02 06 2019].

[8] Reuk, «Reuk.co.uk,» [En linea]. Available: http://www.reuk.co.uk/wordpress/accurateds3231-

real-time-clock-as-alternative-to-ds1307/. [Último acceso: 02 06 2019].

[9] Eletrogate, «electrogate,» [En línea]. Available: http://blog.eletrogate.com/rtc-real-time-clock-ds1302-1307-e-3231/ [Último acceso: 11 06 2019].

[10] Ebay, «Ebay,» [En linea]. Available: https://www.ebay.es/itm/Modulo-RTC-DS1302-Reloj-Tiempo-Real-AVR-PIC-Arduino-incluye-pila-CR2032-M0013/201538874189?hash=item2eeca72b4d:g:RYEAAOSwAuNW4Wuo. [Último acceso: 11 06 2019].

[11] Ebay, «Ebay,» [En linea]. Available: https://www.ebay.es/itm/DS1307-AT24C32-con-PILATiny-RTC-RELOJ-TIEMPO-REAL-AVR-ARM-Arduino-Robot-M0101/201878830698?hash=item2f00ea7e6a:g:JT0AAOSw03lY4oe2. [Último acceso: 11 06 2019].

[12] Ebay, «Ebay,» [En linea]. Available: https://www.ebay.es/itm/DS3231-AT24C32-con-PILARTC-RELOJ-TIEMPO-REAL-AVR-ARM-Arduino-Robotica-M0004/201566961566?hash=item2eee53bf9e:g:y50AAOSwYudXGoyM. [Último acceso: 11 06 2019].

[13] Naylamp Mechatronics, «Naylamp Mechatronics,» [En linea]. Available:

https://naylampmechatronics.com/blog/15_Configuraci%C3%B3n--del-m%C3%B3dulobluetooth-HC-06-usa.html. [Último acceso: 11 06 2019].

[14] Wikipedia, «Wikipedia.org,» [En linea]. Available: https://es.wikipedia.org/wiki/Secure_Digital. [Último acceso: 11 06 2019].

[15] discoazul, «discoazul.com,» [En linea]. Available: https://www.discoazul.com/x-one-powerbank-2600mah-negro.html?gclid=Cj0KCQjwxYLoBRCxARIsAEf16-sonQrSJVTlgjmccRxcFewGkLEBaZj2UUsJ2nLDDheEc6uRSmKKs94aAk5NEALw_wcB

[Último acceso: 11 06 2019].

[16] Geek factory, «geekfactory,» [En línea]. Available: https://www.geekfactory.mx/tutoriales/tutoriales-Arduino/alimentar-el-Arduino-la-guia-definitiva/# [Último acceso: 11 06 2019].

[17] Wikipedia, «Wikipedia,» [En linea]. Available: https://es.wikipedia.org/wiki/Placa_de_pruebas. [Último acceso: 11 06 2019].

[18] Arduino, «Arduino,» [En linea]. Available: https://www.Arduino.cc/en/Main/Software. [Último acceso: 11 06 2019].

[19] Arduino, «Arduino,» [En linea]. Available: https://www.Arduino.cc/en/Reference/SoftwareSerial [Último acceso: 11 06 2019].

[20] aprendiendoArduino, «aprendiendoArduino,» [En linea]. Available: https://aprendiendoArduino.wordpress.com/tag/funciones/ [Último acceso: 15 09 2018].

[21] MIT, «AppInventor2,» [En linea]. Available: http://ai2.Appinventor.mit.edu/ [Último acceso: 11 06 2019].

[22] wikipedia, «wikipedia.org,» [En linea]. Available: https://es.wikipedia.org/wiki/Diagrama_de_flujo [Último acceso: 11 06 2019].

[23] yeeply, «yeeply.com,» [En linea]. Available: https://www.yeeply.com/blog/entornos-programacion-desarrollar-apps-android/ [Último acceso: 20 06 2019].

[24] CCOO, «industria.ccoo.es,» [En linea]. Available: http://www.industria.ccoo.es/Pais_Valencia/Convenios/Valencia:_convenios_provinciales/Industria_del_Metal [Último acceso: 20 06 2019].

8. Anexos

8.1. Anexo I: Programación Arduino IDE

En este Anexo se presenta el código utilizado para el proyecto en Arduino IDE. Es el programa principal, Fichaje.ino:

```
//RST            D9
//SDA(SS)        D8
//MOSI           D11
//MISO           D12
//SCK            D13
 #include <SPI.h> // El modulo RC522 usa el protocolo SPI
#include <MFRC522.h> // Libreria para el MIFARE RC522
#include <EEPROM.h> // Para leer y escribir las UIDs de las tarjetas
a la EEPROM
#include <SD.h> // Libreria para SD
#include <SoftwareSerial.h>
#include <Wire.h> // El DS3231 se comunica por protocolo Wire
#include <RtcDS3231.h> / Libreria para el RTC DS3231
RtcDS3231<TwoWire> Rtc(Wire);

//#include "EEPROMAnything.h"

SoftwareSerial BT(5,6); //5 RX, 6 TX

#define countof(a) (sizeof(a) / sizeof(a[0]))

boolean match = false;
int registerMode = 1; // inicializar el modo de guardado a 1 (guardar
registro)
int successRead;
int i=0;
byte storedCard[4]; // Guarda una ID leída de la EEPROM
byte readCard[4]; // Guarda una ID escaneada por el Modulo RFID
char valor; // variable para leer una orden de la aplicación
char cadena [255];
bool cardReadbyRC522 = false;
char character, characterBT;

#define wipeB 3
#define RST_PIN 9            // Pin 9 para el reset del RC522
#define SS_RC522_PIN 8             // Pin 8 para el SS (SDA) del
RC522
#define SS_Eth_PIN 10        // Pin 10 para el chip W5100 del shield
Ethernet, fijo por la placa
#define SS_SD_PIN 4          // Pin 4 para el lector de micro SD del
shield Ethernet, fijo por la placa
MFRC522 mfrc522(SS_RC522_PIN,RST_PIN);   // Crear instancia del
MFRC522
File miFichero;

struct myStruct { //tipo de variable "struct" del conjunto numero-
nombre de cada tarjeta
byte cardSt[4]; // ocupa un total de 14 bytes
char nameSt[10];
} StructID;

////////////////////////////////////////////////////// Void setup
Inicializaciones//////////////////////////////////////////////////////
```

```
void setup() {

   pinMode(SS_Eth_PIN, OUTPUT); //Necesario para que el sistema de
Slave Select funcione
   pinMode(SS_SD_PIN, OUTPUT);
   pinMode(SS_RC522_PIN, OUTPUT);


   digitalWrite(SS_RC522_PIN, LOW); //seleccionar el esclavo "RC522"
   digitalWrite(SS_Eth_PIN, HIGH); //desactivar el esclavo "Ethernet"
   digitalWrite(SS_SD_PIN, HIGH);  //seleccionar el esclavo "SD"

   Serial.print("compiled: ");
   Serial.println(__DATE__ );
   Serial.println(__TIME__);
   Rtc.Begin();

////////////////////////////////////////////////Configuración de la
hora del DS3231, utiliza la hora de
compilación/////////////////////////////////////////////////////////

// Una vez compilado la primera vez comentar este código para que se
quede la misma hora y fecha
/*
RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
printDateTime(compiled);

Serial.println();
if (!Rtc.IsDateTimeValid())
{
// Common Causes:
// 1) first time you ran and the device wasn't running yet
// 2) the battery on the device is low or even missing
Serial.println("RTC lost confidence in the DateTime!");
// following line sets the RTC to the date & time this sketch was
compiled
// it will also reset the valid flag internally unless the Rtc device
is
// having an issue
Rtc.SetDateTime(compiled);
}
if (!Rtc.GetIsRunning()) {
Serial.println("RTC was not actively running, starting now");
Rtc.SetIsRunning(true);
}
RtcDateTime now = Rtc.GetDateTime();
if (now < compiled) {
Serial.println("RTC is older than compile time! (Updating
DateTime)");
Rtc.SetDateTime(compiled);
}
else if (now > compiled) {
Serial.println("RTC is newer than compile time. (this is expected)");
}
else if (now == compiled) {
Serial.println("RTC is the same as compile time! (not expected but
all is fine)");
}
*/
```

```
////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////

Rtc.Enable32kHzPin(false);
Rtc.SetSquareWavePin(DS3231SquareWavePin_ModeNone);

Serial.begin(9600);      //Inicializa la velocidad de Serial



activarSD();
Serial.print(F("Inicializando tarjeta SD..."));
if (!SD.begin(4)) {
Serial.println(F(" fallo! Comprueba la tarjeta SD"));
Serial.println(F("Sin tarjeta SD no puedo continuar"));
} Serial.println(F("Éxito."));
desactivarSD();

SPI.begin();          //Función que inicializa SPI
mfrc522.PCD_Init();     //Función  que inicializa RFID
BT.begin(9600);        // Inicializar Bluetooth
mfrc522.PCD_SetAntennaGain(mfrc522.RxGain_max);

ShowReaderDetails();
//fin del Codigo de Vaciado
Serial.println(F("Todo listo"));
Serial.println(F("Esperando tarjetas de alumnos"));

}


////////////////////////////////////////////////////////////////////
////Bucle Loop Bucle principal/////////////////////////////////


void loop() {
  // Detectar tarjeta

successRead = getID();

if (successRead != 0) { // si se produce una lectura por parte del
MFRC522:

cardReadbyRC522 = true;

for (int i = 0; i < 4; i++) { //
leadingZerosBT(readCard[i]);
BT.print(readCard[i], HEX);
}

delay(250);

if (registerMode == 1) {

guardarRegistro(readCard);// GUARDADO EN LA SD DE LA HORA
}

}
```

```
cardReadbyRC522 = false;

String mensajeBT = "";
String mensaje = "";
bool receivedBT = false; // no se ha recibido el mensaje por
Bluetooth

while (BT.available()) {
characterBT = BT.read();
mensajeBT.concat(characterBT); // a menos que se reciba por Bluetooth
receivedBT = true;
}

if (receivedBT == true) { //si se recibe por Bluetooth

receivedBT = false;
mensaje = mensajeBT; //hacer que el mensaje sea igual al mensaje
recibido por Bluetooth

}

while (Serial.available()) {
character = Serial.read();
mensaje.concat(character);
}
mensaje.trim();

mensaje.toCharArray(cadena, 30); //convertimos el String "mensaje" en
un array de caracteres
//

if (mensaje == "actReg") {
if (registerMode == 0) {
registerMode = 1;
}
Serial.println(F("Registro activado"));
}

if (mensaje == "desactReg") {
if (registerMode == 1) {
registerMode = 0;
}
Serial.println(F("Registro DESactivado"));
}
if (mensaje == "regsend") {
mensaje = "";
Serial.print(F("Enviando el fichero registro.txt vía Bluetooth...
"));
activarSD();
miFichero = SD.open("registro.txt"); //Abierto para lectura
if (miFichero) {
while (miFichero.available()) {
BT.write(miFichero.read());

}
```

```
//BT.write("");
miFichero.close();
Serial.println(F("completado, enviando FinRegistro"));
//BT.write(F("FinRegistro"));
}
else {
Serial.println(F("error abriendo registro.txt"));
//BT.write(F("FinRegistro"));
}
desactivarSD();
}


if (mensaje == "regdelete") {
mensaje = "";
activarSD();
if (SD.exists("registro.txt")) {
Serial.print(F("Eliminando el fichero 'registro.txt'... "));
SD.remove("registro.txt");
}
else {
Serial.println("'registro.txt' no existe en la microSD.");
}
desactivarSD();
}



}

////////////////////////////////////////////////////// Escribir
fecha y hora //////////////////////////////////////////////////////

void printDateTime(const RtcDateTime & dt)
{
char datestring[20];
snprintf_P(datestring,countof(datestring),PSTR("%02u/%02u/%04u
%02u:%02u:%02u"),dt.Month(),dt.Day(),dt.Year(),dt.Hour(),dt.Minute(),
dt.Second() );
Serial.print(datestring);
}
//////////////////////////////////////////////////////////////////
Caracteristicas Sensor RFID ///////////////////////////////////////


void ShowReaderDetails() {
// Obtener la version de software del MFRC522
byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
Serial.print(F("Version de Software de MFRC522: 0x"));
Serial.print(v, HEX);
if (v == 0x91) {
Serial.print(F(" = v1.0"));
}
else if (v == 0x92) {
Serial.print(F(" = v2.0"));
}
```

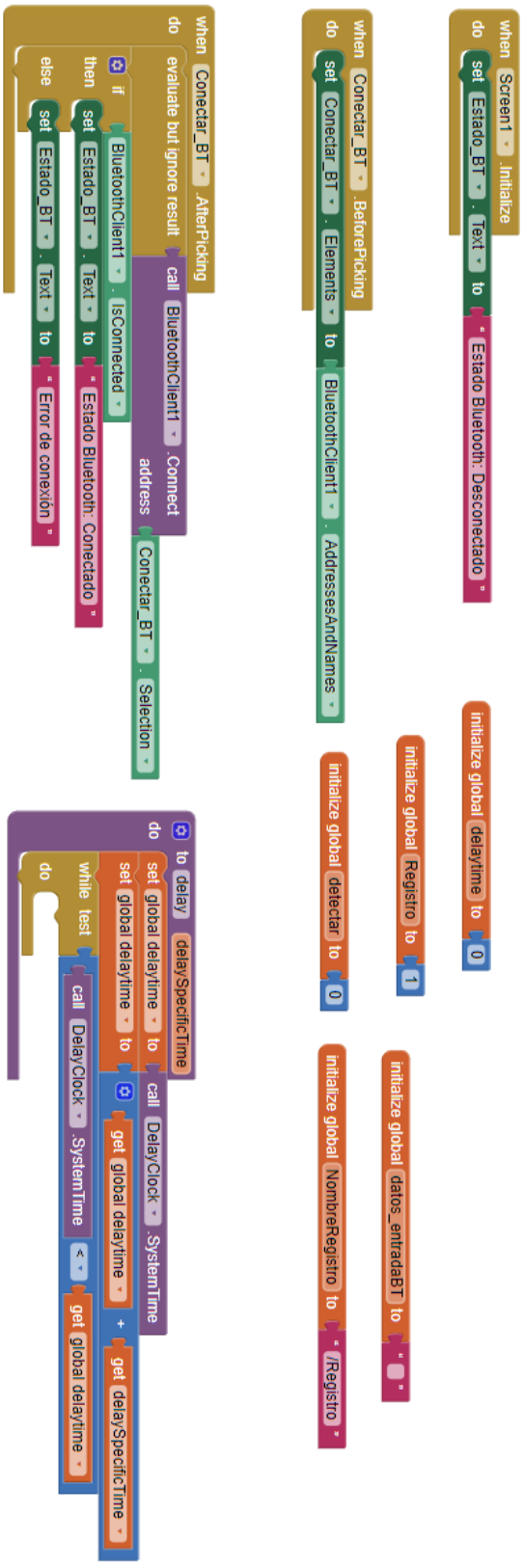
```
else {
Serial.print(F(" (desconocida, comprobar MFRC522 o comunicacion)"));
}
Serial.println("");
// Si se recibe 0x00 or 0xFF probablemente la comunicacion haya
fallado
if ((v == 0x00) || (v == 0xFF)) {
Serial.println(F("ATENCION: Fallo de comunicacion, esta bien
conectado el MFRC522?"));
while (true); // no ir mas alla si no hay lector de tarjetas
}
}

/////////////////////////////////////////Escribir ID puerto serie
//////////////////////////////////////////////////


void printArray(byte *buffer, byte bufferSize) {
   for (byte i = 0; i < bufferSize; i++) {
      Serial.print(buffer[i] < 0x10 ? " 0" : " ");
      Serial.print(buffer[i], HEX);
   }
}



/////////////////////////////////////////////////////////////////////
//////    On-Off SD
   ////////////////////////////////////////////////////////


void activarSD() {
digitalWrite(SS_RC522_PIN, HIGH); //para desactivar el esclavo RC522
digitalWrite(SS_SD_PIN, LOW); //para activar el esclavo SD
}

void desactivarSD() {
digitalWrite(SS_SD_PIN, HIGH); //para desactivar el esclavo SD
digitalWrite(SS_RC522_PIN, LOW); //para activar el esclavo RC522
}
/////////////////////////////////////////////////////////////////////
////////Guardar tarjeta en
SD////////////////////////////////////////////


void guardarRegistro(byte TarjetaAdmitida[4]) {
// CODIGO DEL 3231
RtcDateTime now = Rtc.GetDateTime();
if (!Rtc.IsDateTimeValid()) // si la fecha y tiempo no son validos:
{
// Causas comunes
// 1) la bateria del dispositivo es baja o no esta, y la alimentacion
esta desconectada
Serial.println("El RTC ha perdido confianza en el DateTime!");
}
```

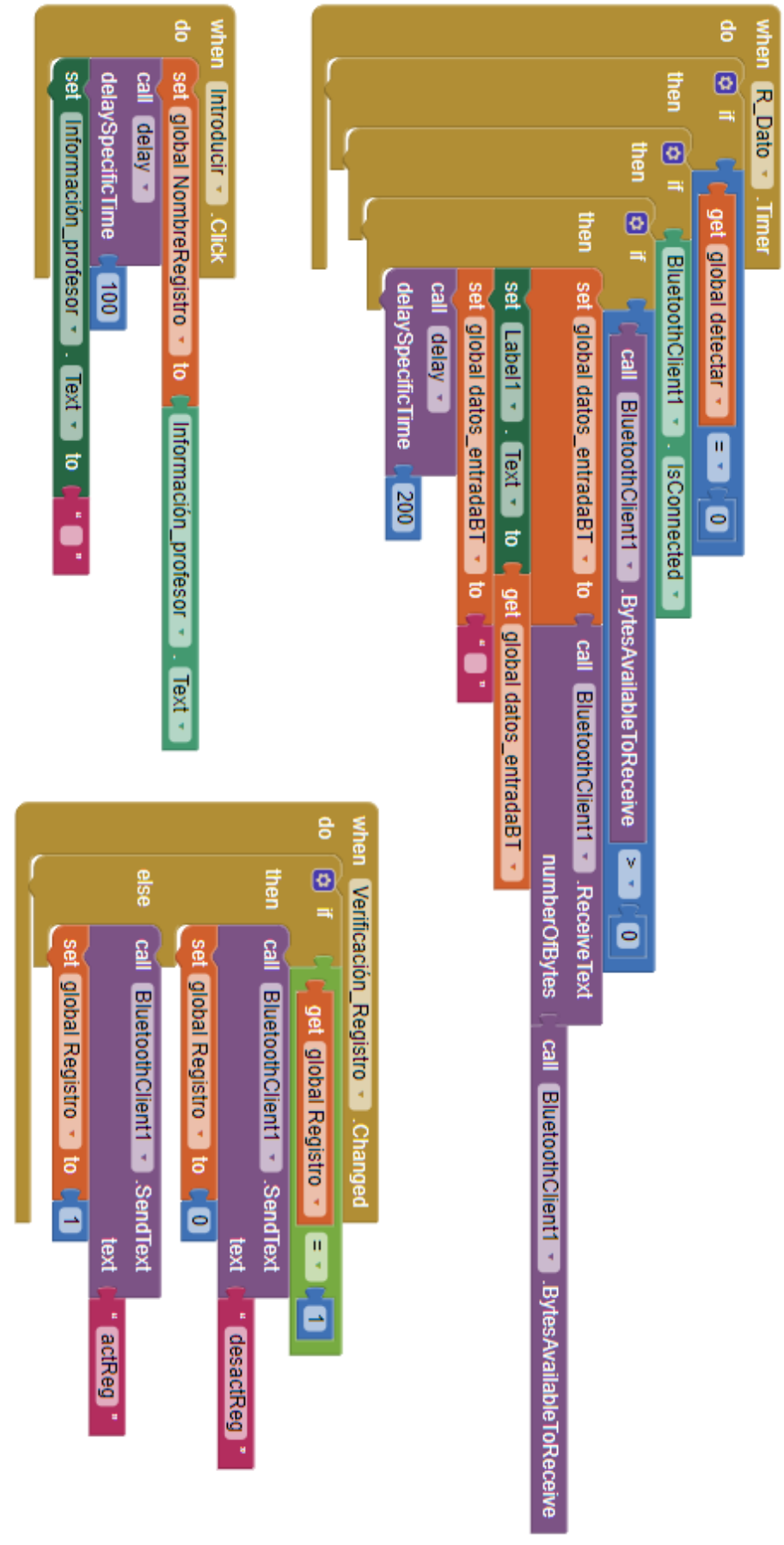
```
char datestring[20];
snprintf_P(datestring,countof(datestring),PSTR("%04u-%02u-%02u
%02u:%02u:%02u"),now.Year(),now.Month(),now.Day()
,now.Hour(),now.Minute(),now.Second());// escritura de la fecha y
hora en la variable
Serial.println(datestring);
//guardado del acceso en la SD
activarSD();
miFichero = SD.open("registro.txt", FILE_WRITE);
if (miFichero) { // si existe la variable miFichero
Serial.print(F("Guardando en registro.txt..."));
miFichero.print(datestring); //fecha y hora 3231
miFichero.print(F("-"));
for (int i = 0; i < 4; i++) { //
if (TarjetaAdmitida[i] < 10) { //para anadir 0s a la izquierda si es
menor que 10 el numero
miFichero.print(F("0"));
}
miFichero.print(TarjetaAdmitida[i], HEX);// escritura ID en el
archivo
}
miFichero.print(F("-"));
miFichero.println(StructID.nameSt); //guarda el nombre
miFichero.close();
Serial.println(F(" Éxito!"));
}
else {
  Serial.println(F("error abriendo registro.txt"));
}
desactivarSD();
}

//////////////////////////////////////////////////////////Poner 0S
antes si hay
huecos//////////////////////////////////////////////////////////////////
//////////////

void leadingZeros( int byteLeido) {
if (byteLeido < 16) { //para anyadir 0s a la izquierda si unicamente
tiene un digito HE
Serial.print(F("0"));
}
}
void leadingZerosBT( int byteLeido) {
if (byteLeido < 16) { //para anyadir 0s a la izquierda si unicamente
tiene un digito HEX, envio por Bluetooth
BT.print(F("0"));
}
}
//////////////////////////////////////////////////////////////Get
PICC's
UID//////////////////////////////////////////////////////////////////
//////////////////
int getID() {

if ( ! mfrc522.PICC_IsNewCardPresent()) { //Si un nuevo PICC se
coloca en el lector RFID continúa
return 0;
}
```

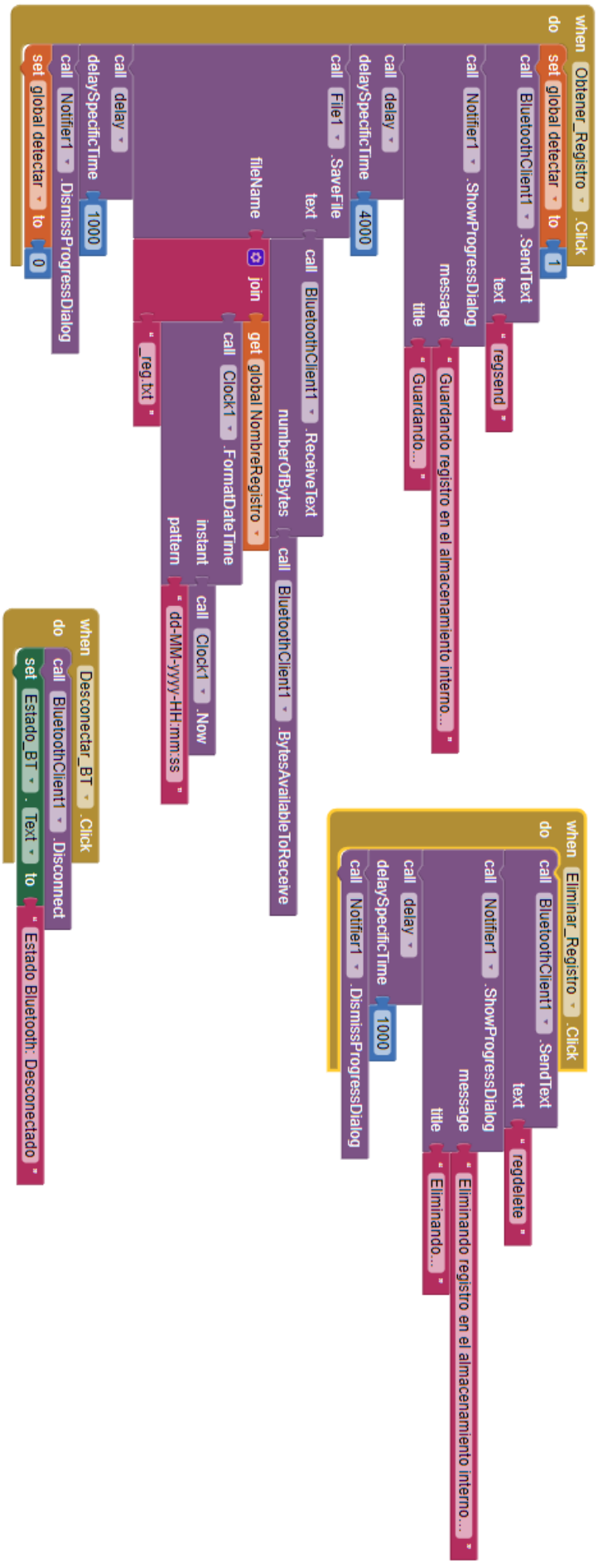
```
if ( ! mfrc522.PICC_ReadCardSerial()) { //Desde que se colocó un PICC
obtener Serial y continuar
return 0;
}

Serial.println(F("Scanned PICC's UID: "));

for (int i = 0; i < 4; i++) { // int de 4 porque las IDs contienen 8
números hexadecimales que se guardan en 4 bytes
readCard[i] = mfrc522.uid.uidByte[i]; //guardado de cada byte leido
en una variable
Serial.print(F("Byte "));
Serial.print(i);
Serial.print(F(": "));
leadingZeros(readCard[i]); // añade 0s  a la izquierda si unicamente
tiene un digito HEX
Serial.print(readCard[i], HEX);
Serial.println(F(""));

}
Serial.println(F(""));
for (int i = 0; i < 4; i++) { //
leadingZeros(readCard[i]);
Serial.print(readCard[i], HEX);// mostrar ID por puerto serie
}
Serial.println("");

mfrc522.PICC_HaltA(); // Stop Lectura
return 1;
}
```

8.2. Anexo II: Programación Appinventor2

En este apartado se presenta el código de programación de bloques de Appinventor2:




when R_Dato .Timer
do if get global detectar = 0
then if BluetoothClient1 .IsConnected
then if call BluetoothClient1 .BytesAvailableToReceive > 0
then set global datos_entradaBT to call BluetoothClient1 .ReceiveText numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
set Label1 . Text to get global datos_entradaBT
set global datos_entradaBT to ""
call delay delaySpecificTime 200
when Introducir .Click
do set global NombreRegistro to Información_profesor . Text
call delay delaySpecificTime 100
set Información_profesor . Text to ""
when Verificación_Registro .Changed
do if get global Registro = 1
then call BluetoothClient1 .SendText text "desactReg"
set global Registro to 0
else call BluetoothClient1 .SendText text "actReg"
set global Registro to 1


when Obtener_Registro .Click
do set global detectar to 1
call BluetoothClient1 .SendText
text "regsend"
call Notifier1 .ShowProgressDialog
message "Guardando registro en el almacenamiento interno..."
title "Guardando..."
call delay
delaySpecificTime 4000
call File1 .SaveFile
text call BluetoothClient1 .ReceiveText
numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
fileName join get global NombreRegistro
"_reg.txt"
call Clock1 .FormatDateTime
instant call Clock1 .Now
pattern "dd-MM-yyyy-HH:mm:ss"
call delay
delaySpecificTime 1000
call Notifier1 .DismissProgressDialog
set global detectar to 0
when Eliminar_Registro .Click
do call BluetoothClient1 .SendText
text "regdelete"
call Notifier1 .ShowProgressDialog
message "Eliminando registro en el almacenamiento interno..."
title "Eliminando..."
call delay
delaySpecificTime 1000
call Notifier1 .DismissProgressDialog
when Desconectar_BT .Click
do call BluetoothClient1 .Disconnect
set Estado_BT . Text to "Estado Bluetooth: Desconectado"