UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO UAEM TEXCOCO

“PROTOTIPO AUTOMATIZADO DE REGISTROS PARA EL INSTITUTO MEXIQUENSE DE CULTURA FÍSICA Y DEPORTE”

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA

GARCÍA LÓPEZ DAVID MISAEL

ASESOR

I. EN C. GERARDO OSMANY OLGUÍN GONZÁLEZ

REVISORES

M. EN C.C GERARDO ALEXIS VILLAVICENCIO PÉREZ

M. EN C.C. BENITO SAMUEL LÓPEZ RAZO

TEXCOCO, ESTADO DE MÉXICO, MARZO DE 2019.

Universidad Autónoma del Estado de México
Centro universitario UAEM Texcoco

Texcoco, México a 7 de diciembre 2018

Asunto: Etapa de digitalización

M. EN C. ED. VIRIDIANA BANDA ARZATE
SUBDIRECTORA ACADEMICA DEL
CENTRO UNIVERSITARIO UAEM TEXCOCO
PRESENTE.

AT´N: L. EN I.A. CINTHYA TERESITA ISLAS RODRIGUEZ
RESPONSABLE DEL DEPARTAMENTO DE TITULACION

Con base en las revisiones efectuadas al trabajo escrito titulado "PROTOTIPO AUTOMATIZADO DE REGISTROS PARA EL INSTITUTO MEXIQUENSE DE CULTURA FÍSICA Y DEPORTE" que para obtener el título de Licenciado en Ingeniero en Computación presenta el (la) sustentante García López David Misael, con número de cuenta 1014934, se concluye que cumple con los requisitos teórico-metodológicos por lo que se le otorga el voto aprobatorio para su sustentación, pudiendo continuar con la etapa de digitalización del trabajo escrito.

ATENTAMENTE

M. en C. [illegible] Samuel López
NOMBRE Y FIRMA DEL REVISOR

Gerardo Alexis Villavicencio Pérez
NOMBRE Y FIRMA DEL REVISOR

Gerardo Osmany Olguín González
NOMBRE Y FIRMA DEL DIRECTOR

TITULACION
RECIBIDO
Por Leopoldo
Texcoco, Méx., a 14 de 12 del 18

C.C.P. Sustentante: García López David Misael
C.C.P. Asesor de trabajo terminal: Gerardo Osmany Olguín González
C.C.P. Titulación. CINTHYA TERESITA ISLAS RODRIGUEZ

Agradecimientos

A mis padres y hermanas…

Por la paciencia, dedicación y preocupación a lo largo de la etapa de avance y desarrollo de esta tesis. A mi madre, López Catalina, la gratitud hacia ella nunca será lo suficiente, gracias a su confianza, creer en mí y en mis expectativas. El apoyo que recibí de su parte para poder hacer esto posible, y concluir esta etapa tan importante para un estudiante y un hijo. A mi padre, Garcia David, por siempre guiarme por el buen camino, desear y anhelar lo mejor para mi vida. A mis hermanas, Garcia Dallely y Garcia Damaris, por las experiencias, consejos y palabras compartidas que me guiaron a finalizar este trabajo.

A mis maestros…

Por poder aprender de ellos a lo largo de la estancia en la universidad, y en especial, a los ingenieros Olguín Osmany, Villavicencio Alexis y *López Benito, que me apoyaron en la realización y entrega de la tesis. El tiempo que invirtieron enseñando y orientándome no será mal gastado, ya que gracias a sus enseñanzas y experiencias académicas fui capaz crecer como persona e ingeniero, gracias a esto, será posible presentar un trabajo de tesis. Y de igual manera, mis maestros pueden estar confiados que entregarán estudiantes de calidad al campo laboral.*

Contenido

Tabla de ilustraciones

Índice de tablas

Introducción

En la actualidad a consecuencia del aumento de la población, muchos comercios o empresas, han tenido la necesidad de obtener métodos o diferentes tecnologías para tener un control del personal autorizado a entrar a sus instalaciones, por seguridad o simplemente tener el registro de quien se encuentra dentro para así poder evitar futuros problemas.

En el caso de empresas grandes, han optado por usar la tecnología como una solución a este problema, y con esto, poder identificar plenamente a todos sus trabajadores o visitantes. Muchas de estas compañías usan credenciales, o códigos personales para poder ingresar a los establecimientos, dependiendo de la seguridad que sea deseada será el tipo de admisión necesaria.

La presente investigación, da a conocer el problema de un gimnasio, donde no se cuenta y no se da a conocer con la suficiente seguridad o fiabilidad la información de todos los clientes de este establecimiento. Los datos actualmente son obtenidos por personas, y esto siempre puede generar algunas fallas en el sistema.

Se pretende, con este trabajo abordar los temas de seguridad y fiabilidad de los datos, pertenecientes a los clientes o visitantes de las instalaciones, para posteriormente dar una solución que limite y solucione el problema de ingreso del personal. Este proyecto se compone por un total de cuatro capítulos.

En el primer capítulo, se abordan antecedentes sobre cómo fue posible crear una solución a un problema similar acerca de la conglomeración de gente; así como objetivos planteados, justificación, viabilidad del prototipo, métricas de calidad, etc.

En el segundo capítulo, se dan a conocer tipos de almacenamiento, el funcionamiento, tipos, y justificación de la placa base Arduino. Así como la estructura de programación y características de algunos modelos.

El tercer capítulo se centra en los módulos, como el Global System for Mobile, identificador de radio frecuencia y pantalla LCD, que son los tres módulos utilizados y aceptados por Arduino. Se da a conocer el funcionamiento, configuración, especificaciones y programación de estos dispositivos.

El cuarto y último capítulo, es utilizado para explicar el funcionamiento en conjunto de todos los módulos presentados. Back-end y Front-end son mostrados con el debido código fuente y ejemplos. Se presentan los diagramas utilizados, como el diagrama de la placa de cobre y modelos de instalación, así como los resultados obtenidos y posibles mejoras a futuro.

1. Antecedentes

La conglomeración de personal y usuarios es uno de los principales problemas, hoy en día en la mayoría de los establecimientos públicos por lo cual, es necesario tener el conteo de todo el personal, ya sea por seguridad o simplemente una buena organización. En la mayoría de los casos se han ayudado de la tecnología que, gracias a esta es posible mantener un orden constante.

Con base en lo que María (2012) dice, en marzo de 2008, el sistema de Transporte Colectivo puso a disposición de los usuarios una “Tarjeta Recargable” para agilizar el ingreso a las estaciones, ahorrar tiempo evitando largas filas y mejorar la programación del gasto.

Cuando una persona entra a un establecimiento, en ocasiones registra su entrada con puño y letra, dichos datos en algunas empresas son utilizados para conocer el flujo de visitantes o trabajadores. Aunque la organización no sea la mejor, en muchas de estas aún es utilizada.

En muchas organizaciones, han tenido un problema similar en lo que a registro de personal se refiere, muchos de estos casos han optado por usar la tecnología como solución al problema ya sea con tarjetas electrónicas o diferentes tipos de lectores que puedan identificar a cada uno de los trabajadores.

2. Planteamiento del problema

2.1. Objetivo General

Prototipo de un sistema de registro de entradas y salidas para usuarios dentro del gimnasio polideportivo de IMCUFIDE (Instituto Mexiquense de Cultura Física y Deporte) notificando a cada uno de los usuarios la fecha del vencimiento de pago.

2.2 Objetivos específicos

1. Tener los conceptos de todos los dispositivos y como obtener la calidad del software.
2. Conocer el funcionamiento y la importancia del almacenamiento de datos, así como conocer, y aprender a utilizar hardware y software relacionados con el trabajo.
3. Saber la estructura, conexión y programación de los módulos de comunicación, módulos de lectura y dispositivos para interactuar con el usuario.
4. Diseño de diagramas y armado en conjunto de todos los dispositivos a utilizar.

2.3. Pregunta de investigación

"¿Es posible hacer uso de mensajes de texto, visualizar información para facilitar tramites y obtener una mejor organización en un establecimiento público?"

2.4. Problema principal

En diversos lugares es necesario tener un registro de personas, generalmente es importante tener un control adecuado, o es necesario por seguridad. En la actualidad hay muchos lugares y métodos para poder hacer este tipo de registro, en el trabajo para conocer la asistencia de cada uno de los empleados, en escuelas para conocer el horario de los maestros, o simplemente

existen lugares que requieren saber quién entra a sus instalaciones y la hora en la que salieron.

En el caso particular del gimnasio IMCUIFIDE (Instituto Mexiquense de Cultura Física y Deporte) donde el sistema de registro no es funcional, son utilizados métodos “antiguos” porque así ha sido usado por muchos años y no se opta por mejorar esta situación, algunos métodos son caros y prefieren quedarse con el más barato, pero no el más eficaz.

En este gimnasio en particular, se emplea una técnica que cuenta con diferentes errores, tanto administrativos como de seguridad, puesto que son usados “carnets” para identificar a los usuarios y donde se muestra con un sello el cual refleja si dicha persona realizó su pago mensual. Este carnet al finalizar la estancia se deja en un escritorio, en la entrada del establecimiento lo que genera una gran desorganización.

En el supuesto caso que el número de personas dentro del gimnasio sea poca esto podría ser útil. Pero cuando la afluencia incrementa y todas usan el gimnasio al mismo tiempo, el escritorio se llena de “carnets”, como consecuencia, no es posible saber con exactitud quien entró al gimnasio, quien permanece dentro, quien salió, o simplemente que persona no recogió el carnet, dejándolos olvidados en el escritorio, generando una enorme desorganización, sin mencionar que el usuario deja al entrar su carnet y en ocasiones no hay nadie que los organice.

Para conocer los horarios de los usuarios es empleada una lista (que también es colocada en el mismo escritorio) donde cada persona tiene la “obligación” de

anotar su nombre y la hora “exacta” de entrada, repitiendo el mismo proceso a la hora de salir. Esto genera problemas, ya que algunos no se anotan en esta lista y muchas veces no es posible identificar a dicho usuario.

Tampoco es confiable la hora que pone cada usuario, ya que de igual manera las instalaciones no cuentan con un supervisor encargado de verificar que todos los datos registrados por el usuario sean verídicos, delegando esta función al instructor del gimnasio, pero debido a las múltiples obligaciones que el cargo genera con los usuarios, no puede atender la lista y los “carnets”.

3. Justificación

Como se mostró anteriormente, el gimnasio IMCUFIDE no tiene la suficiente organización en lo que respecta a los horarios de admisión, el nuevo prototipo tiene el registro de los usuarios que están dentro del establecimiento, los que permanecen, y los que ya no se encuentran en las instalaciones. Guardará los datos en ficheros, y solo por un periodo de tiempo mostrará en pantalla y enviará mensajes de texto a los usuarios.

Prototipo que puede fácilmente ser aplicado a otras áreas del deportivo y no solamente al gimnasio. También puede emplearse a cualquier establecimiento o empresa donde se necesite tener un control de usuarios y tener un sistema de avisos mediante mensajes vía texto.

Así como lo dice el *Curi (2002) en el* “Reglamento interno del instituto mexiquense de cultura física y deporte” en el artículo 18 sección 2. Realizar

supervisiones, así como también auditorías contables, operacionales, técnicas y jurídicas a las unidades administrativas del Instituto, tendientes a verificar el cumplimiento de las normas relacionadas con los prototipos de registro, contabilidad, contratación y pago de personal.

Es de gran importancia obtener todos los datos posibles de cualquier persona dentro es estas instalaciones. Por esto se desarrolló un prototipo que facilite la organización de dichos datos, y de igual manera facilidad de visualización de los mismos. Como en muchos establecimientos públicos, es necesaria una cuota y notifica al usuario cuando su pago (que anteriormente fue realizado) estará próximo a vencer.

4. Hipótesis

Poniendo en ejecución este prototipo en el gimnasio IMCUFIDE, el flujo de los usuarios que utilicen estas instalaciones será realizada de una manera más organizada y ordenada, no se perderá tiempo en registros que no son útiles. Ayudará a recordar el corte de pago mensual, y finalmente se podrá hacer menos uso de material primas como lo son hojas, carnés y recibos.

5. Diagrama de flujo

La metodología que es presentada en la Ilustración 1, representa los pasos a seguir de este prototipo para su correcta ejecución.

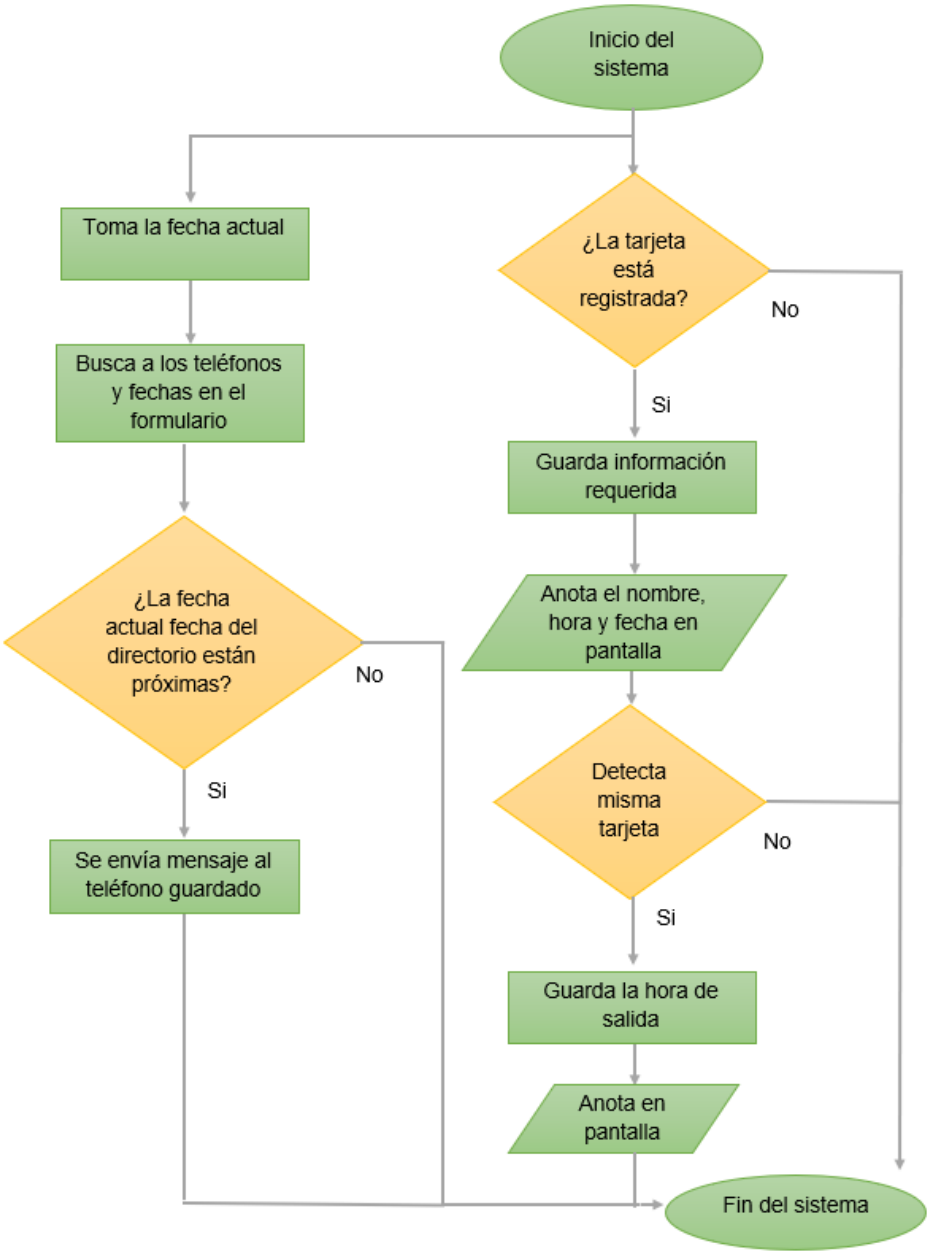

Ilustración 1.Diagrama de flujo, funcionamiento del prototipo

6. Viabilidad

Este prototipo es viable gracias a la velocidad de la lectura de las tarjetas, y gracias a esto, reduce la conglomeración de gente que pudiera existir en esta actividad. Lo que impulsa este tema, es a dar a conocer y mostrar un claro ejemplo de cómo la tecnología puede aplicarse a actividades cotidianas como estas.

El costo de este prototipo, en realidad es muy por debajo de lo que se pensaba antes de realizar las investigaciones necesarias, sin embargo, una mala organización podría generar un problema de pérdida de algún material dentro de las instalaciones, y que no sería viable para la organización reponer módulos que generarían un gasto excesivo. Solo es necesario el pago de cuatro dispositivos (GMS SIM900, RFID, TFT LCD TOUCH 2.4” Y ARDUINO MEGA).

Para poder emplear este prototipo, solamente es necesario saber la correcta utilización de todos los dispositivos ya mencionados, gracias a Arduino estos pueden trabajar en conjunto sin problema alguno. Existen algunos prototipos en la actualidad que hacen el registro del personal, pero muchos de estos son costosos. Lo innovador de este proyecto es que el sistema en conjunto puede notificar a cualquier usuario el corte de su pago.

6.1. Deficiencias

Es necesario saber todas las deficiencias que podrían generarse en el transcurso del prototipo, una de estas es la utilización de una tarjeta SIM con suficiente crédito, puesto que es esencial para poder enviar todos los mensajes de texto que sean necesarios, y la memoria de almacenamiento de la pantalla TFT es limitada.

Capitulo I. Definiciones, métricas de calidad y ciclo de desarrollo.

7. Conceptos generales

7.1. Registro y recolección de datos

La recolección de información debe realizarse utilizando un proceso planeado, para que de forma viable puedan obtener resultados que contribuyan favorablemente al logro de los objetivos propuestos (PARADA, 1999). Es necesario seguir los pasos definidos para obtener los datos requeridos, en este caso los datos de cada usuario, para poder cumplir con los objetivos establecidos.

Ilustración 2. Recolección de datos.

7.2. Arduino

Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware/software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos. (Ojeda, Arduino, 2018)

Ilustración 3. Logo Arduino. Fuente: https://arduino.cc

7.3. Hardware

El termino hardware se refiere a cualquier parte de la computadora que se puede tocar. Consiste en dispositivos electrónicos interconectados que podemos usar para controlar la operación, así como la entrada y salida de la computadora; además, se refiere a los dispositivos físicos que conforman el sistema de computación. (Alonso, 2003)

Ilustración 4. Chip Arduino mega 2560. Fuente: https://oliversmith.io

7.4. Arduino Mega 2560

Arduino Mega es una tarjeta de desarrollo open-source construida con un microcontrolador modelo Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales. Esta tarjeta es programada en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede comunicarse a un PC a través del puerto

serial (conversión con USB) utilizando lenguajes como Flash, Processing, MaxMSP, etc. Las posibilidades de realizar desarrollos basados en Arduino tienen como límite la imaginación. (Ojeda, Arduino, 2018)

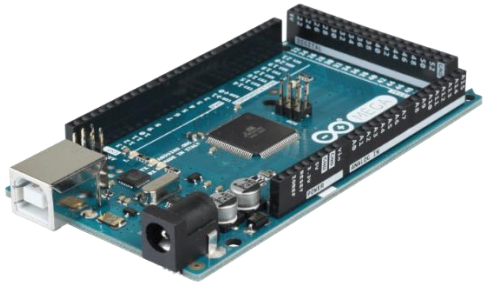

Ilustración 5. Modulo Arduino mega 2560

7.5. Módulo GSM SIM 900

Módulo de comunicación GSM/GPRS capaz de enviar y recibir SMS, datos GPRS y llamadas telefónicas, es compatible con Arduino y cualquier plataforma de desarrollo de microcontroladores. Este permite mediante comandos AT configurar la comunicación de datos vía remota por IP desde cualquier punto con señal GSM. Es posible marcar a cualquier teléfono fijo o móvil y puede recibir y contestar llamadas. Ideal para aplicaciones de Telemetría móvil. Comunicación con el microcontrolador a través del puerto UART (Universal Asynchronous Receiver-Transmitter). (S.A, 2017)

Ilustración 6. Modulo GSM SIM 900.

7.6. Entorno de desarrollo Arduino

Entorno específico para personas que trabajan en backend, es utilizado para desarrollar todas las instrucciones que sean necesarias para el correcto funcionamiento de cualquier programa.

7.7. RFID

Es un prototipo de auto identificación inalámbrica, el cual consulta de etiquetas que almacenan información y lectores que pueden leer a estas etiquetas a distancia utilizando una frecuencia de onda electromagnética para realizar dicha tarea. Estas frecuencias varían según la aplicación y puede ser de 125Khz, 13.5Mhz, 433-860-900Mhz, 2.4Ghz y 5.8Ghz.

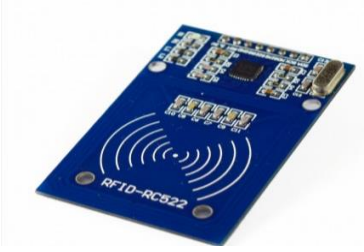

Ilustración 7. Modulo RFID

7.7.1. Tarjeta RFID

Pueden ser presentadas en diferentes formatos, los más comunes son llaveros y tarjetas, son utilizadas para guardar toda información necesaria información para después obtenerla mediante un lector RFID, cabe destacar que cada tarjeta es única de fábrica.

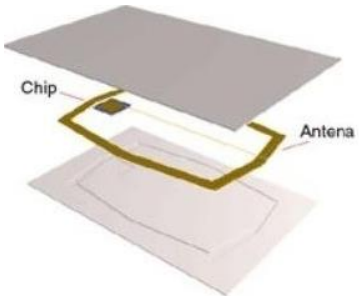

Ilustración 8. Tarjeta RFID

7.8. TFT

Conocido como pantalla de cristal líquido es un tablero o pantalla que trabaja en forma de matriz y dicha matriz trabaja con pixeles, únicos para cada espacio de la matriz y tiene como objetivo mejorar la calidad de la imagen.

Ilustración 9.Pantalla TFT LCD. Fuente: http://www.elocalx.co

7.8.1. TFT LDC Touch 2.4”

Este módulo de visualización es un cristal líquido TFT de matriz activa de color transmisivo pantalla (LCD) que usa TFT de silicio amorfo como dispositivo de conmutación. Este módulo está compuesto por Módulo LCD TFT, un circuito de control y una unidad de retroiluminación. La resolución de 2.4 "contiene 240 (RGB) X320 puntos y puede mostrar hasta 262k colores. (INC, 2013)

Ilustración 10. Pantalla TFT LCD Touch 2.4

8. Variables

- Personas dentro establecimiento.
- Calidad del software
- Calidad de hardware
- Pago al derecho de entrada.

8.1. Variables cuantitativas

Personas dentro del establecimiento: Es posible saber el número de personas que ingresaron, permanecen o ya no hacen uso de las instalaciones

Pago al derecho de entrada: Se necesita saber la fecha de corte para cada usuario, así el prototipo podrá saber cuándo es necesario realizar su pago nuevamente.

8.2. Variables cualitativas

Calidad del software: Es necesario tener una buena calidad de desarrollo de cada una de las partes con conforman al prototipo, hacer el código lo más simplificado posible para así evitar un mal manejo del prototipo y eliminar “bugs” que podrían presentarse.

Calidad del hardware: Se necesitan los mejores dispositivos, hablando de calidad y simplicidad para que el prototipo tenga los mejores resultados y a su

vez que no se entiendan en costos que no son necesarios para el prototipo.

8.3. Variables independientes

Calidad del hardware: Centrado a trabajar con lo tangible.

Calidad del software: Centrado a trabajar con lo intangible.

8.4. Variables dependientes

Personas dentro del establecimiento: Depende directamente de las tarjetas que son entregadas a estos, ya que son el identificador en el prototipo. Pago al derecho de entradas: De igual manera son datos que se necesitan conocer para que el prototipo realice su objetivo.

9. Ciclo de desarrollo

En la Ilustración 11 es mostrado el ciclo que será seguido en este prototipo ya que nos permite entablar una serie de procedimientos para poder llegar al objetivo.

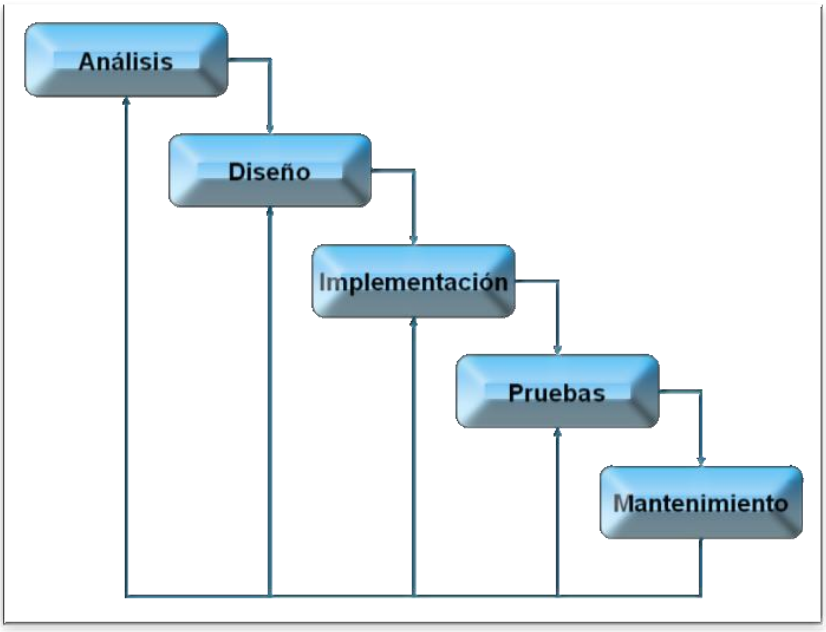

Ilustración 11. Ciclo de desarrollo de software lineal

1. Análisis de los requerimientos del software:

Es la fase en la cual son reunidos todos los requisitos que debe cumplir el software. En esta etapa es fundamental la presencia del cliente que documenta y repasa dichos requisitos. (Leon, 2013)

Es necesario reunir información del campo de estudio, así también, saber que dispositivos son los más útiles para resolver el problema establecido y que tendrán una función única para el desarrollo del prototipo.

2. Diseño:

Es una etapa dirigida hacia la estructura de datos, la arquitectura del software, las representaciones de la interfaz y el detalle procedimental (algoritmo). En forma general se hace un esbozo de lo solicitado y se documenta haciéndose parte del software. (Leon, 2013)

Es importante “dibujar” o diseñar la estructura de todo el armado final para poder detectar errores de ensamblados no previstos.

3. Generación del código:

Es la etapa en la cual se traduce el diseño para que sea comprensible por la máquina. Esta etapa va a depender estrechamente de lo detallado del diseño (Leon, 2013). Se conocen todos los elementos integrados en el prototipo, como la pequeña base de datos almacenada que interactúa con la pantalla TFT y el módulo GSM.

4. Pruebas:

Esta etapa se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en la detección de errores. (Leon, 2013). Para poder identificar los errores que puedan aparecer durante la ejecución de todo el prototipo y poder resolverlos en el momento de falla y si es necesario regresar a alguna etapa y rediseñar e implementar de distinta manera.

5. Mantenimiento:

Debido a que el programa puede tener errores, puede no ser del completo agrado del cliente o puede necesitar, eventualmente acoplarse a los cambios en su entorno. Esto quiere decir que no se rehace el programa, sino que sobre la base de uno ya existente se realizan algunos cambios. (Leon, 2013)

Si el prototipo no cumple con los objetivos establecidos o no se cumplen como se esperaban se podrán hacer los cambios correspondientes, todo esto después de un periodo de prueba.

9.1. Bottom-Up (De abajo arriba)

Junto con *top-down* son estrategias para el procesamiento de información. Sin embargo, cada una es aplicada a distintos prototipos. Bottom-up especifica las partes individuales que incorpora todo el prototipo y los detalla, para posteriormente enlazarlos hasta que sea formado el prototipo completo.

Este método tiene como facilidad las pruebas tempranas y el énfasis en la programación de los dispositivos, aunque va de la mano con un riesgo, puesto que es posible no saber cómo conectar todos los dispositivos al prototipo y esta conexión puede no ser tan fácil como se pensó en un principio, teniendo esto encuentra, es posible la reutilización de código, y este es uno de los mayores beneficios de este enfoque.

En la Ilustración 12 se puede observar una representación de un prototipo bottom-up, donde se tiene diferentes módulos, cada uno independiente de otro, se maneja y programa individualmente, para posteriormente conectar todos los componentes en uno solo.

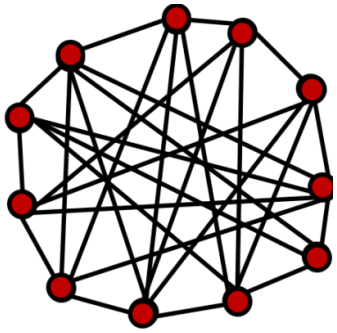

Ilustración 12. Representación del modelo Botton-Up

10. Métricas de calidad

Tener un enfoque correcto y de calidad, hace posible la identificación temprana de errores o algunos defectos que puedan aparecer en algunos sistemas o prototipos, pero que pueden ser corregidos si son detectados a tiempo. Para ellos existen algunas métricas que ayudan a este objetivo.

a) Métricas de exactitud: Información sobre la validez, precisión y estructura del software
b) Métricas de rendimiento: Medir el desempeño del software, tanto de cada uno de sus módulos, como del sistema al completo.
c) Métricas de usabilidad: Eliminar la complejidad y buscar una solución intuitiva y user-friendly.
d) Métricas de configuración: Las limitaciones y el estilo de código.

e) Métricas de eficiencia: minimización de latencias, velocidad de respuesta y capacidad.

Capítulo II. ¿Qué es y para que funciona el almacenamiento de datos? Definición de Arduino, viabilidad, software, hardware y utilización.

11. Almacenamiento de datos

El ser humano, en su época ha tenido la necesidad de actualizar y mejorar sus métodos de almacenamiento a causa del incremento de población y de datos, esto genera varios retos para el manejo de esta información. En la antigüedad, la información era guardada, y se utilizaban tablas de arcilla para llevar un conteo, convirtiéndose en el primer método de almacenamiento.

En el área digital, el almacenamiento surge con la aparición de las primeras computadoras, con ellas surgen los discos duros, y esto ha cambiado nuestra manera de almacenar, obligándonos a tener una mayor capacidad de espacio en estos dispositivos para acceder y recuperar información. Dependiendo la época y la cantidad de datos que son utilizados, es necesario usar diferentes tecnologías de almacenamiento que sea adecuado para completar el trabajo establecido, sin quitarle importancia al resguardo de datos.

11.1. Tipos de almacenamiento

En prototipos, existen tres tipos de almacenamiento, Almacenamiento de Conexión Directa (DAS), Almacenamiento Conectado en Red (NAS) y Red de Área de Almacenamiento (SAN). En este caso es aplicado el almacenamiento DAS, puesto que las unidades están conectadas directamente con un servidor o un host a través de una interfaz. Este método es útil, y tiene muchas ventajas como lo es su instalación, ya que se hace de manera fácil; el software es muy poco complejo; el costo de mantenimiento es bajo; buena compatibilidad.

Aun sabiendo esto, se tienen algunas deficiencias, como la capacidad de almacenamiento, que es limitada por su servidor. Pueden existir algunos problemas que implican una mala ejecución en este prototipo, sin embargo, estos problemas no interfieren en su trabajo.

Todo establecimiento público o privado, necesita un almacén de datos con la información de todos los usuarios que ingresan a sus instalaciones, así como el nombre y hora, para que puedan ser identificados. Varia de la cantidad de datos y la manera de obtenerlos, ya que para cada caso es diferente. Es posible que un establecimiento no tenga tanta concurrencia de personal, o que no sea necesaria mucha seguridad para el acceso al inmueble, pero lo importante es tener conocimiento de ellos.

Existen varios tipos de sensores o lectores que pueden ayudar a esta actividad, por ejemplo:

- Huella
- Iris
- Tarjetas con información escrita
- Firmas
- Código de barras

En el caso de la huella el iris, son sistemas utilizados principalmente para resguardo de datos personales que se usa como control de acceso mediante medios biométricos, y es una buena opción en lo que a seguridad se refiere. Sin embargo, se pueden producir alteraciones en la huella de cada individuo, que podría causar problemas y no reconocer al usuario hasta que coincida de nuevo.

En el prototipo no es necesario esta tecnología ya que, a este tipo de instalaciones, acude una diversidad de personas, muchos de estos menores o personas mayores, donde este tipo de información es clasificada y personal.

Hablando de IMCUFIDE, se cuenta con tarjetones de acceso o también llamados “Carnets, sin embargo, no han dado buenos resultados, ya que la cantidad de usuarios es grande para hacer uso de estos sistemas rudimentarios, y a su vez es difícil acceder a los datos de cada una de las personas.

De igual manera, se usa una firma hecha a mano que genera varios problemas, ya que el instituto no cuenta con el personal capacitado que pueda comprobar la legibilidad de las firmas, y genera un gasto de papelería innecesario.

El código de barras es una tecnología muy usada, pero con varios problemas con la lectura de datos ya que los códigos de barras no pueden leer etiquetas que

estén arrugadas, manchadas o sucias puesto que dependen de estas para ser leídas. De igual manera se busca un ahorro de capital, y se necesitaría una impresión de cada código de barras, cosa que generaría costos grandes comparado al benéfico que puede dar.

Depende de cada caso difiere que se necesite, ya que para una institución donde se requiera una buena seguridad de acceso se puede optar por una huella o iris (que aun así cuentan con algunas deficiencias), pero en otros muy diferentes es posible el uso de otro tipo.

En el caso de gimnasios, donde no se requiere de una seguridad robusta es posible hacer uso de otro tipo de registros. Por ejemplo, tarjetas de radio frecuencia puesto que es una manera más sencilla y cuenta con un grado de seguridad necesario para este tipo de establecimientos. Como cualquier prototipo existen ventajas y desventajas del uso de este dispositivo.

Ventajas:

1) La etiqueta es entregada por el establecimiento.
2) No es posible el uso de tarjetas externas.
3) Se tiene un control total de usuarios activos.
4) Es un dispositivo amigable para todo tipo de persona.
5) Rapidez en la lectura de datos.
6) Menos susceptibles al daño.

Desventajas:

1) Es un poco más caro que otras alternativas como el código de barras, sin embargo, se compensa con la eficiencia.
2) Es posible una colisión en la lectura al estar presente dos o más tarjetas en el lector.

Una vez establecido el método de obtención de datos, es necesario un método para guardar la información obtenida. En este prototipo se usa una pantalla TFT, en la cual está integrada una ranura tarjeta microSD (Secure Digital) explotando al máximo las capacidades de este módulo para almacenar la información previamente obtenida.

12. ¿Qué es Arduino?

Arduino es software y hardware libre que trabajan en conjunto para la realización de sistemas completos y/o prototipos de electrónica. Es una herramienta de fácil utilización, que puede ser de mucha ayuda para estudiantes, trabajadores, diseñadores, profesores o cualquier persona con algún interés electrónico.

Las placas de Arduino pueden ser compradas pre-ensambladas, o bien se puede ensamblar, teniendo todos los componentes necesarios. Y el software puede ser descargado sin costo alguno.

Esta tecnología cuenta con la facilidad de diversos dispositivos externos que pueden ser utilizados para llegar al objetivo de un prototipo determinado, muchos

de estos pueden ser sensores, motores, pantallas, etc. Y que gracias a la plataforma Arduino pueden trabajar en conjunto. En general, podemos decir que Arduino consiste en dos partes, la placa Arduino y el IDE (entorno de desarrollo interactivo) que es ejecutado en una computadora.

13. ¿Por qué Arduino?

Existen muchas otras placas de diferentes fabricantes que, aunque incorporan diferentes modelos de microcontroladores, son comparables y ofrecen una funcionalidad más o menos similar a la de las placas. Todas ellas también vienen acompañadas de un entorno de desarrollo agradable, cómodo y de un lenguaje de programación sencillo y completo. No obstante, la plataforma ofrece una serie de ventajas:

1) Libre y extensible: Es un entorno de desarrollo gratis, y cualquiera que desee ampliar y mejorar tanto el diseño hardware de las placas, como el entorno de desarrollo software y el propio lenguaje de programación, puede hacerlo sin problemas.
2) Hardware barato: Tanto como las placas de Arduino y los módulos que se utilizan en conjunto con este, tienen un precio relativamente barato, si se comparan con otros microcontroladores.
3) Gran cantidad de usuarios usan Arduino: Esto ayuda a saber previas experiencias de otros desarrolladores y saber los errores o fortalezas de distintos dispositivos compatibles con Arduino.

4) Multiplataforma: Es posible instalar y ejecutar en prototipos Windows, Mac OS X y Linux. Esto no ocurre con el software de muchas otras placas.
5) Su entorno y el lenguaje de programación son simples y claros: son muy fáciles de aprender y de utilizar, a la vez que flexibles y completos para que los usuarios avanzados y los que empiezan a desarrollar puedan aprovechar y exprimir todas las posibilidades del hardware.
6) Las placas Arduino son reutilizables y versátiles: Reutilizables porque se puede aprovechar la misma placa para varios proyectos, y versátiles porque las placas Arduino proveen varios tipos diferentes de entradas y salidas de datos, en los cuales se pueden insertar diferentes módulos.

Mucha gente piensa que crear prototipos, o desarrollar, es una tarea solo para los que previamente tienen algún conocimiento en programación, pero gracias a Arduino, esta tarea se ha convertido en algo sencillo, ya que con esta tecnología muchas personas pueden ser introducidas al mundo del desarrollo sin requerir de muchos conocimientos en lo que a programación se refiere.

14. Hardware Arduino

Existen muchas versiones del hardware, ya que Arduino nació como un proyecto el año 2005 sin embargo, algunos años tarde se convirtió en líder del mundo en DIY (Do It Yourself ó hágalo usted mismo). El primer prototipo fue desarrollado en el instituto IVRAE.

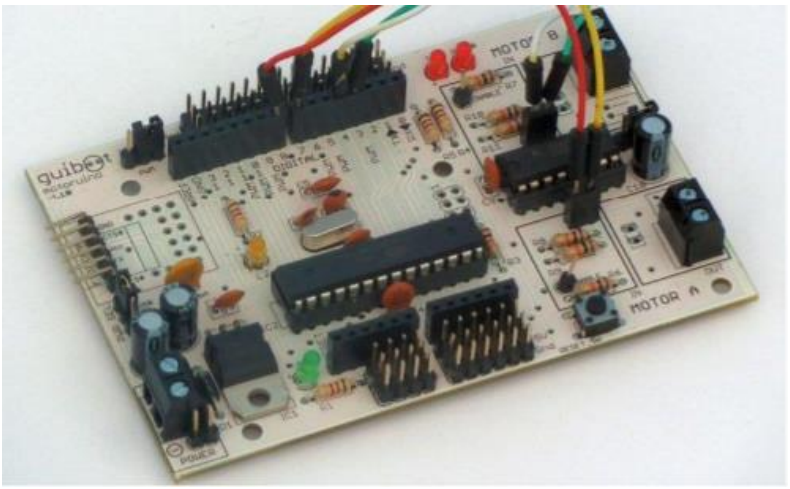

Ilustración 13. Primeros diseños de Arduino.

14.1. Tipos de placas

En la Tabla 1 se muestran las placas donde es posible realizar este prototipo, solamente se puede basar en los números de puertos que cuenta la tarjeta ya que nuestro prototipo necesita de la mayor cantidad de estos.

Tabla 1. Características de las diferentes versiones de la placa de Arduino.

Placa	Procesador	Velocidad	Voltaje	Puertos digitales	Entradas analógicas
Mega2560	ATmega2560	16 MHz	5 V	54	16
Mega	ATmega1280	16 MHz	5 V	54	16
ADK	ATmega2560	16 MHz	5 V	54	16

15. Arduino Mega2560

Se desarrolla este prototipo usando un Arduino Mega 2560, ya que cuenta con la mayor velocidad, almacenamiento y pines disponibles a comparación de otros modelos, cosa que es fundamental para el ensamblado final. Sin contar con la capacidad de almacenamiento, que es mayor con 256 KB de capacidad.

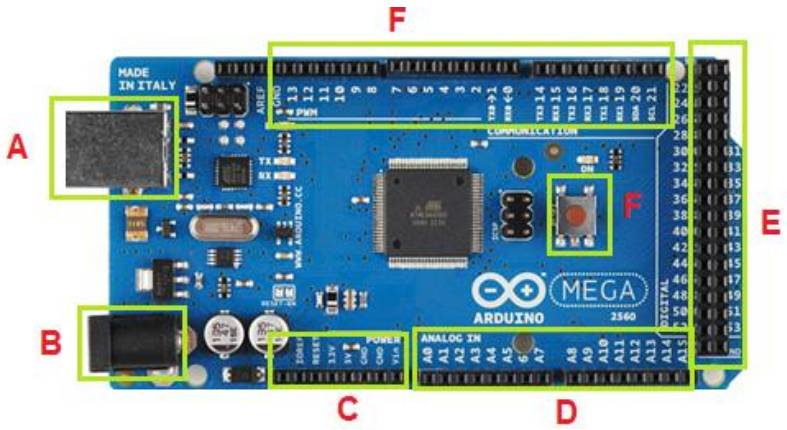

Ilustración 14. Componentes de Arduino MEGA.

En la Ilustración 14 se pueden observar los componentes más importantes de la placa Arduino Mega 2560.

A) Puerto USB

B) Jack alimentación 7 V a 13 V

C) Pines de potencia

D) Pines analógicos

E) Pines digitales

F) Pines digitales

G) Botón de reste

En la Tabla 1 se muestran las características de la placa Arduino Mega 2560.

Tabla 2. Características Arduino Mega2560

Microprocesador	ATmega2560
Tensión de alimentación	7-12V
Integra regulación y estabilización	+5Vcc
líneas de Entradas/Salidas Digitales	54
Entradas Analógicas	16
Máxima corriente continua para las entradas	40 mA
Salida de alimentación	3.3V con 50 mA
Memoria flash	256Kb
Memoria SRAM (Static Random-Access Memory)	8Kb
EEPROM (Electrically Erasable Programmable Read-Only Memory)	4Kb
Velocidad del reloj	16MHz
Dimensiones	100 x 50 mm

15.1. Alimentación

Arduino Mega puede ser alimentando con puerto USB (Universal Serial Bus) o si se desea, con una fuente externa de poder. Si lo que se desea es usar una fuente alterna, es necesario utilizar un convertidor AC/DC, y regular el voltaje de entrada al rango operativo de la placa. Preferiblemente, el rango de voltaje debe estar de los 7V hasta los 12V.

- **VIN:** A través de este pin es posible proporcionar alimentación a la placa.
- **5V:** Se obtiene un voltaje de 5V y una corriente de 40mA desde este pin.
- **3.3V:** Se puede obtener un voltaje de 3.3V y una corriente de 50mA desde este pin.
- **GND:** Pines de tierra.

15.2. Memoria

El Atmega2560 tiene 256 KB de memoria flash para almacenar el código (de la que se utilizan 8 KB para el cargador de arranque), 8 KB de SRAM y 4 KB de EEPROM (que puede ser leída y escrita con la biblioteca EEPROM).

15.3. Entrada y salida

Cada uno de los 54 pines digitales de la Mega se puede utilizar como una entrada o como una salida, utilizando las funciones pinMode(), digitalWrite() y digitalRead(). Operan a 5 voltios, y cada pin puede proporcionar o recibir 20 mA como condición de funcionamiento recomendada y tiene una resistencia de pull-up

(desconectada por defecto) de 20-50 k ohmios. Un máximo de 40 mA es el valor que no debe superarse para evitar daños permanentes en el microcontrolador. Además, algunos pines tienen funciones especiales:

- Serieal: 0 (Rx) y 1 (Tx); Serie 1: 19 (Rx) y 18 (Tx); Serie 2: 17 (Rx) y 16 (Tx); Serie 3: 15 (Rx) y 14 (Tx). Se utiliza para recibir (Rx) y transmitir datos serie (Tx) TTL. Los pines 0 y 1 también están conectados a los pines correspondientes del chip serie ATmega16U2 USB-a-TTL.

- Interrupciones externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3), y 21 (interrupción 2). Estos pines pueden configurarse para activar una interrupción en un nivel bajo, un flanco ascendente o descendente, o un cambio en el nivel. Véase la función attachInterrupt() para más detalles.

- PWM: 2 a 13 y 44 a 46. proporcionan una salida PWM de 8 bits con la función analogWrite().

- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Estos pines soportan la comunicación SPI utilizando la biblioteca SPI

- LED: 13. Hay un LED incorporado conectado al pin digital 13. Cuando el pin está a nivel HIGH, el LED está encendido, cuando el pin está a nivel LOW, está apagado.

- TWI: 20 (SDA) y 21 (SCL). TWI soporte de comunicación utilizando la biblioteca Wire.

15.4. Comunicación

La placa Mega 2560 tiene una serie de facilidades para la comunicación con un ordenador, otra placa, u otros microcontroladores. El Atmega2560 ofrece cuatro UART (Universal Asynchronous Receiver-Transmitter) hardware para TTL (5 V) para la comunicación serie. Una ATmega16U2 (ATmega 8U2 revisión 1 y 2) tiene canales que uno de ellos a través de USB y proporcionan un puerto de comunicación virtual al software del ordenador.

El software de Arduino incluye un monitor serie que permite a datos de texto simple ser enviados a y desde la placa Arduino. Una biblioteca SoftwareSerial permite la comunicación serie en cualquiera de los pines digitales del Mega 2560.

15.5. Programación

La placa Mega 2560 se puede programar con el software de Arduino (IDE). Las Atmega2560 y Mega 2560 vienen preprogramadas con un cargador de arranque, (bootloader) que le permite cargar nuevo código en ella sin el uso de un programador de hardware externo.

15.6. Protección de sobrecarga

El 2560 mega tiene un fusible reajustable que protege a los puertos USB de su ordenador desde cortocircuitos y sobre corriente. Aunque la mayoría de los ordenadores establecen su propia protección interna, el fusible proporciona una capa adicional de protección. Si circulan más de 500 mA por el puerto USB, el fusible interrumpirá automáticamente la conexión hasta que se repara el cortocircuito o se elimina la sobrecarga.

16. Software Arduino IDE (Integrated Development Environment)

El software de Arduino se compone de un IDE, que puede ser ejecutado en cualquier ordenador y puede ser utilizado en cualquier placa de su prototipo, el ambiente de programación el relativamente fácil.

El software Arduino se puede ejecutar en cualquiera de los siguientes Prototipos Operativos:

1) Windows
2) Mac OS X
3) GNU/Linux

El prototipo utilizado en este prototipo fue el prototipo operativo de Windows.

16.1. Estructura de un sketch

La estructura básica del lenguaje de programación de Arduino es bastante simple y con respecto a lo que dice (Banzi, 2007) se organiza en al menos dos partes o funciones que encierran bloques de declaraciones.

```
void setup () {
 statements;
}
void loop () {
 staments;}
```

Ambas funciones son requeridas para que el programa funcione.

Setup () es la parte encargada de recoger la información.

Loop () es la que contiene el programa que se ejecutara cíclicamente.

I. Setup ()

La función setup debe ser incluida en todo código ya que es la primera función que se lee y se invoca solamente una vez. Se utiliza para inicializar los modos de trabajo de los pins o el puerto serie, debe ser incluido, aunque no haya declaración que ejecutar.

```
void setup () {
 pinMode (pin, OUTPUT); // declara un “pin” como salida
}
```

II. Loop ()

La función loop se ejecuta después de setup y contiene el código que se ejecutara cíclicamente lo que posibilita que el programa este respondiendo continuamente.

```
void loop () {
  digitalWrite(pin, HIGH); // pone en uno (on, 5v) el ´pin´
  delay(1000); // espera un segundo (1000 ms)
  digitalWrite(pin, LOW); // pone en cero (off, 0v.) el ´pin´
  delay(1000);
}
```

16.2. Funciones

Una función es un bloque de código que es identificado por un nombre, y puede o no llevar parámetros de entrada y/o salida. La función es mandada a llamar con su nombre, y no pueden existir dos funciones con el mismo nombre.

```
type nombreFunción(parámetros) {
 estamentos;
 }
```

16.3. Variables

Una variable es una forma de almacenar información o un valor numérico para poder modificar ser usada después por el programa. Estas deben ser declaradas y, opcionalmente ser asignado un valor inicial. Es importante que cada

una este declarada antes de ser usadas y se les asigna un tipo de valor, *int, float, etc.*

int variableEntrada = 0; // declara una variable y le asigna el valor 0

variableEntrada = analogRead(2);// la variable recoge el valor analógico del PIN2

16.4. Tipos de datos

I. Byte

Byte almacena un valor numérico de 8 bits sin decimales. Tienen un rango entre 0 y 255.

byte unaVariable = 180; // declara 'unaVariable' como tipo byte

II. Int

Enteros son un tipo de datos primarios, que almacenan valores numéricos de 16 bits sin decimales comprendidos en el rango 32,767 to -32,768.

int unaVariable = 1500; // declara 'unaVariable' de tipo entero

III. Long

El formato de variable numérica de tipo extendido “long” se refiere a números enteros (tipo 32 bits) sin decimales, que se encuentran dentro del rango -2147483648 a 2147483647.

long unaVariable = 90000; // declara 'unaVariable' como tipo long

IV. Float

El formato de dato del tipo “punto flotante” “float” se aplica a los números con decimales. Los números de punto flotante tienen una mayor resolución que los de 32 bits con un rango comprendido 3.4028235E +38 a +38-3.4028235E.

float unaVariable = 3.14; // declara 'unaVariable' como tipo flotante

V. Arrays

Un array es una colección de valores que son accedidos como un índice numérico. Cualquier valor en el array debe llamarse escribiendo el nombre del array y el índice numérico del valor. Los arrays están indexados a cero, con el primer valor en el array comenzando con el índice número 0. Un array necesita ser declarado y opcionalmente asignarle valores antes de que puedan ser usados.

int myArray [] = {value0, value1, value2…};

Capítulo III. Que es, partes, funcionamiento e implementación módulos GSM, RFID y TFT-LCD.

17. GSM SIM900

En inicio de la década de los ochenta, la mayor parte de los países de Europa ya tenían su propio prototipo de telefonía de celular analógica, y esto impedía que las interfaces conocidas funcionen con otros productos fuera de las fronteras de cada país. Fue hasta 1982, donde el CEPT (Conference of European Post and Telecommunications) creo un grupo de trabajo para poder crear un prototipo que funcione con todos los países de Europa, a dicho prototipo se le denomino “GSM-Groupe Speciale Mobile”.

El grupo desarrollo un nuevo prototipo inalámbrico con las siguientes cualidades: itinerancia (roaming) internacional, soporte para la introducción de nuevos servicios, eficiencia espectral y compatibilidad con la RDSI (Red Digital de Servicios Integrados).

En 1989, la responsabilidad por el desarrollo de GSM fue transferida al ETSI (European Telecommunications Standards Institute) que denominó al proyecto como “*Global System for Mobile Communications”*. Se trata de una tarjeta o módulo compatible con cualquier versión de las placas de Arduino, ya que este tipo de tecnologías están fabricadas para esto.

17.1. Seguridad

La seguridad es uno de los elementos clave del éxito de GSM. Esta tecnología asegura la privacidad, integridad y confidencialidad de las llamadas efectuadas por los usuarios. Para ello, incorpora tarjetas inteligentes que contienen la identificación personal del usuario. El elemento clave en la identificación del usuario es el módulo de identidad del suscriptor, más conocido como SIM.

El SIM es una pequeña tarjeta con un chip impreso que va insertada dentro de cada terminal móvil y que, al contrario de otros prototipos celulares contiene toda la información del usuario.

17.2. Partes

En la Ilustración 15. Partes GSM SIM900 se puede observar las partes que conforman el modulo GSM SIM900.

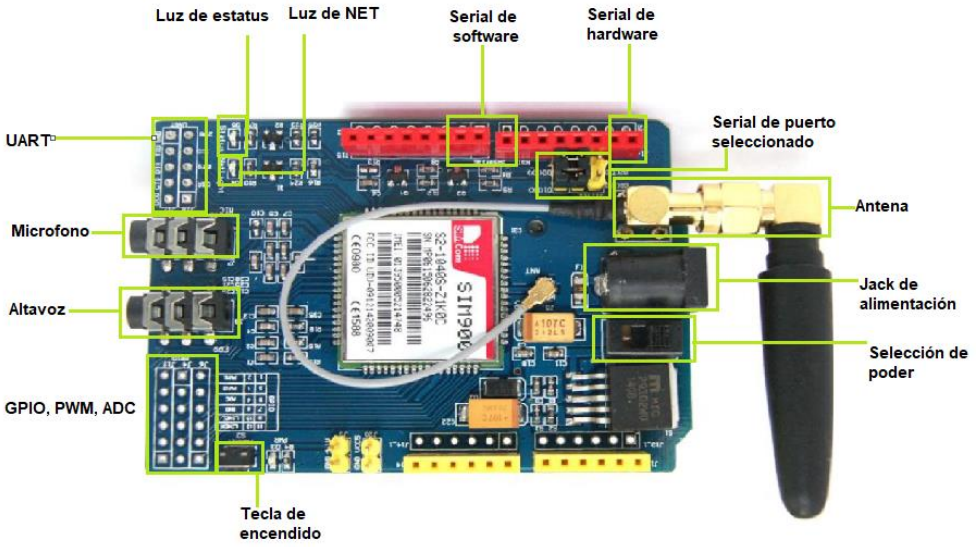

Ilustración 15. Partes GSM SIM900

Los elementos principales que conforman esta tarjeta son:

a) Chip SIM900: En la Ilustración 16 se muestra en pin out del chip.
b) Antena: contiene una entrada directa a la antena por cable coaxial.
c) Zócalo tarjeta SIM: Se encuentra en la parte inferior de la tarjeta donde se inserta una tarjeta SIM. De igual manera contiene un espacio para una batería de 3V para mantener la información cuando la tarjeta con tiene alimentación como se puede ver en la Ilustración 17.
d) Jack de alimentación: la tarjeta puede ser alimentada con 5 V con intensidad de 450 mA.
e) UART: comunicaciones serie asíncronas por hardware mediante los pines D0 y D8.
f) Pines de entrada y salida: Dan la posibilidad de la conexión con Arduino.
g) Micrófono y altavoz: Conectores minijack para llamadas telefónicas.
h) Luz indicadoras: Indican el estado de la tarjeta.

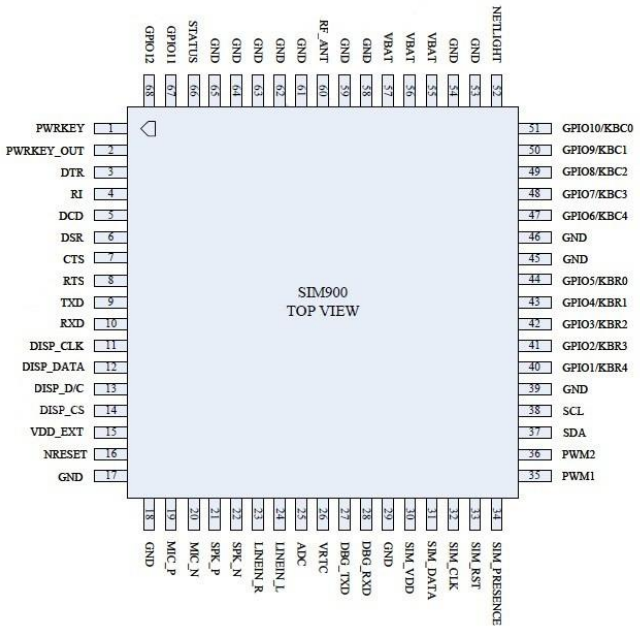

Ilustración 16. Pin out chip SIM900

Ilustración 17. Parte inferior SIM 900

17.3. Comandos AT

La finalidad principal de los comandos AT, es la comunicación con la telefonía móvil GSM usando comandos especiales, y son mandados a la tarjeta para un objetivo en específico. La lista de comando AT son extensos, pero solo se detallan los usados en este prototipo:

AT+CPIN=” XXXX”: Introducir el PIN de la SIM. Cambiar XXXX por el PIN.

AT+CLIP=1: Activamos la identificación de llamada.

AT+CMGF=1: Configura el modo texto para enviar o recibir mensajes.

AT+CNMI=2,2,0,0,0: Configura el módulo para que muestre los mensajes de texto por el puerto serie.

AT+CMGS=” XXXXXXXXX”: Numero al que se le enviara el mensaje de texto.

17.3 Conexión con Placa Arduino

Se recomiendan una serie de pasos para hacer uso del módulo SIM900 ya que de lo contrario podrian surgir errores no esperados.

I. Tarjeta SIM

Es necesario usar una tarjeta SIM con saldo disponible para hacer posible el envio de mensajes de texto, es recomendable usar tarjetas SIM prepagadas, o con un plan, puesto que en la realización de las pruebas podrian enviarse cientos de mensajes de texto sin tener conocimiento de esto. Existen tres tipos de tamaño de SIM, tal como se muestra en la Ilustración 18, sin embargo, la unica compatible con SIM900, es MiniSIM. Algunos módulos vienen con el adaptador a esta medida. Una vez se cuente con el SIM adecuado, se coloca en la parte inferior del modulo SIM900 tal como se muestra en la Ilustración 18. Tipos de SIM. Tambien asegurarse de que la antena este bien conectada. Hhhn

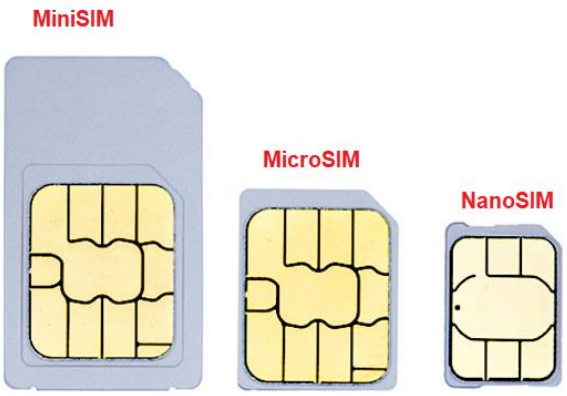

Ilustración 18. Tipos de SIM

II. Alimentación

Existen dos formas de alimentar el módulo, y se puede cambiar el modo de este mediante la selección de poder como se muestra en la Ilustración 15, este prototipo se alimentará mediante el Jack de alimentación que se encuentra a un costado del selector de poder. Se recomienda usar una fuente de alimentacion de 5V con 2A, o bien con 9V a 1ª, 12V a 1A.

III. Encendido

Para encender el módulo, se presiona el botón de encendido aproximadamente 3 segundos, seguido de esto, el LED de estado encenderá y el LED NED comenzará a parpadear.

IV. Conexión

Solamente se hará uso de tres líneas que son las correspondientes a “GND”, que es conectáda a una de las tierras de la placa, y dos para “serial de software”, donde en el módulo SIM900 son los pines 7 y 8, (se puede leer en la parte inferior de la placa)

líneas que son conectadas a la los pins 30 y 31 respectivamente de Arduino como se muestra en la Ilustración 19.

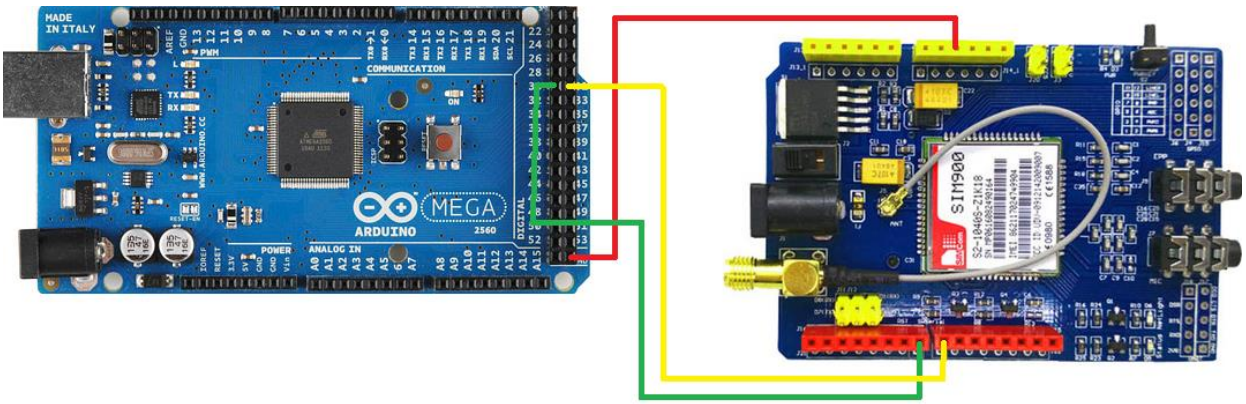

Ilustración 19. Conexión Arduino y SIM900

17.4. En IDE Arduino

Como fue dicho con anterioridad, es necesario incluir las librerías de cada componente que es usado, en este caso es necesario usar “SoftwareSerial.h” de Arduino donde se declaran al inicio de cada código.

```
#include <SoftwareSerial.h>
SoftwareSerial SIM900(30,31); // Configura el puerto serial para el SIM900
```

La mayor parte del prototipo (software), se trabajó con funciones, para tener un mayor orden de todo lo que pase dentro del código, para la utilización del

SIM900, se usaron dos funciones necesarias para el envío de mensajes de texto y, donde la función llamada “inicializaSIM900” recibe un parámetro llamado “telf” y es la encargada de usar los primeros comando AT los que son: “AT + CPIN”, “AT+CLIP=1”,” AT+CMGF=1” y ” AT+CNMI=2,2,0,0,0”. Esta misma función, tiene como trabajo llamar a la segunda función, designada al envío de mensajes.

La segunda función tiene como nombre “enviarSMS”, que es la encargada de recibir el teléfono, y simplemente ejecuta dos comandos AT, necesarios para él envió del mensaje. Cabe destacar, que cada país tiene una lada y que es necesario cambialo dependiendo en donde se aplique, en este caso la lada para México es “+52” que es la que se da como parámetro en el código.

```
void inicializaSIM900(String telf) {

SIM900.begin(19200); //Configura velocidad del puerto serie para el SIM900

Serial.println("OK");

SIM900.println("AT + CPIN = \"1234\""); //Comando AT para introducir el PIN de la tarjeta

Serial.println("PIN OK");

SIM900.print("AT+CLIP=1\r"); // Activa la identificación de llamada

SIM900.print("AT+CMGF=1\r"); //Configura el modo texto para enviar o recibir mensajes
```

```
SIM900.print("AT+CNMI=2,2,0,0,0\r"); // Saca el contenido del SMS por el puerto serie del GPRS

Serial.println("OK");

enviarSMS(telf);// Se manda a llamar la función enviar SMS con el parámetro telf

}

void enviarSMS(String telf){

SIM900.print("AT+CMGF=1\r");                    // AT command to send SMS message

delay(100);

Serial.print("TELEFONO RECIBIDO: ");

Serial.println("+52"+telf+"");

SIM900.println("AT + CMGS = \"+52"+telf+"\"");          // recipient's mobile number, in international format

delay(100);

SIM900.println("En tres días expirara su mensualidad en el instituto IMCUFIDE, se le recomienda hacer el pago correspondiente"); // Mensaje para ser enviado

Serial.println("Mensaje Enviado");

delay(100);
```

```
    SIM900.println((char)26); // End AT command with a ^Z, ASCII code 26
    delay(100);
    SIM900.println();
    delay(3000);                    // give module time to send SMS
                                    // turn off module
}
```

La función “inicializaSIM900” es una de las primeras mandadas a llamar, ya que cuando el prototipo inicie, se enviará el mensaje de texto a los usuarios que así lo requieran.

18. RFID

El origen de RFID, viene de la segunda Guerra Mundial, donde era común el uso de radares, que permitía la detección de aviones a distancia, pero no realizaba identificación exacta. Alemania descubrió que algunos pilotos balanceaban sus aviones al volver a base, y esto cambiaba la señal de radio que transmitían. Con esto, podían identificar los aviones alemanes de los enemigos, y se dio origen al primer dispositivo RFID pasivo.

Los prototipos de radar por radiofrecuencia avanzaron en las décadas de los 50 y los 60 donde querían identificar remotamente. Posteriormente muchas empresas empezaron a usarlas como sistema de antirrobo, usando las ondas de radio, sabían si un objeto estaba pagado o no.

Los prototipos de identificación por radio frecuencia, es una tecnología para identificación de personas u objetos sin necesidad de contacto. La tecnología de RFID es un proceso que integra un módulo físico con un único identificador, este debe seguir un protocolo, que es transferido desde un lector hacia una terminal mediante (como su nombre lo dice) ondas de radio. Funciona como un prototipo, que puede almacenar y obtener información de manera remota usando tarjetas o simplemente etiquetas RFID.

Esta tecnología ha sido desarrollada para convertirse en un prototipo funcional y seguro. En resumen, el funcionamiento no es muy diferente al de código de barras, puesto que permite tener acceso rápido y de manera fiable mediante un lector. No es necesaria completamente la intervención del ser humano para que cumpla su objetivo establecido.

Para poder tener una lectura de un tag o una etiqueta, los lectores emiten señales de radio, y una vez que la etiqueta está lo suficientemente cerca del lector, este lo puede identificar. Los tags pueden ser leídos a distancia, no es necesario un contacto directo al lector o estar alineadas a este.

18.1. Tecnología RFID / Código de barras

RFID y el código de barras han tenido un conflicto siempre preguntándose qué tecnología es la mejor, sin embargo, esto puede variar dependiendo su uso ya que cada una tiene sus ventajas y limitaciones, pero aun sabiendo esto hay algunos

aspectos que pueden ser mencionados, el por qué fue seleccionada RFID para este prototipo y cuáles son sus ventajas en este.

- Los tags de RFID pueden escribirse y modificarse en cualquier momento, solamente actualización la información dentro de los tags. En cambio, esto no es posible en un código de barras.
- Los tags de RFID son más resistentes que un código de barras ya que pueden ser leídos con suciedad o polvo.
- Los tags RFID no se pueden falsificar de forma fácil, puesto que la información en su interior está protegida. En cambio, un código de barras puede ser impreso con cualquier impresora láser.
- El aspecto más importante es que RFID cuenta con más memoria que los códigos de barras, e incluso que un código QR.

18.2. ¿Cómo funciona RFID?

Todo RFID está compuesto de un prototipo de base, también conocido como “interrogador” que lee, escribe y modifica información en los dispositivos. Y un “transportador” que se encarga de responder al interrogador. El interrogador crea un campo de radiofrecuencia que van desde 125 KHz hasta 2.4GHz. El interrogador detecta los datos transmitidos por un tag a un rango de lectura, para la mayoría de los casos, hasta 60 centímetros de distancia entre el interrogador y la tarjeta.

Existen tres componentes básicos que componen un prototipo de RFID:

1) Tag/Etiqueta:

Es la etiqueta RFID, que consiste en un circuito que contiene una pequeña antena que puede transmitir un número identificador único hacia un lector. Existen tres tipos de tags, pasivas, semi-pasivas y activas.

Los tags pasivos no contienen fuente de alimentación, como una batería. Obtienen la corriente de la señal de radio frecuencia, gracias a esta energía son capaces de transmitir una respuesta al lector. Estas etiquetas tienen una distancia de lectura que varían de 10 milímetros hasta los 6 metros, esto depende del tamaño de la antena del tag, en la Ilustración 20 se muestra una etiqueta RFID pasiva. Y en la Tabla 3 se muestra las características de esta etiqueta.

Ilustración 20.Etiqueta RFID pasiva. Fuente: http://www.pandaid.com

Tabla 3.Caracteristicas etiqueta pasiva.

Alcance	**1cm-10m**
Alimentación	Campo magnético
Tiempo de vida	Ilimitado
Costo	Desde $8 MXN
Aplicaciones	Tarjetas inteligentes e inventario

Las etiquetas semi-pasivas, son parecidas a las pasivas, sin embargo, estas integran una diminuta batería, que tiene la función de mantener alimentada la etiqueta, y elimina la necesidad de una antena para obtener la energía de una señal. Estas etiquetas responden más rápido que las pasivas. En la Ilustración 21 se muestra una etiqueta semi-pasiva.

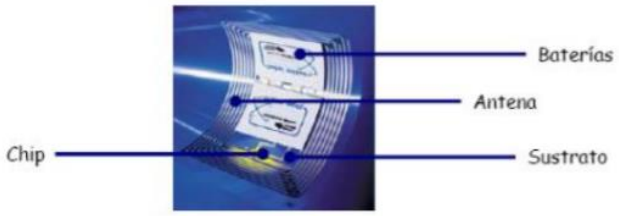

Ilustración 21. Etiqueta RFID semi-pasiva

Las etiquetas activas, de igual manera que las semi-pasivas cuentan con una fuente de energía, pero estas tienen un rango de cobertura mayor y memorias de almacenamiento superiores.

2) Lector

Este puede tener dos funciones, lectura y/o escritura compuesto por un módulo electrónico de radiofrecuencia. Los lectores pueden tener diferentes tamaños, funcionalidad y costo, esto depende del objetivo que se tenga establecido, existen 2 tipos, con prototipo de bobina simple y con prototipos de interrogadores.

Los lectores con prototipos de bobina simple sirven para transmitir la energía y datos, estos son sencillos, económicos y de poco alcance. Este

es el lector utilizado en este prototipo. En la Tabla 4 se muestran las especificaciones del lector RFID RC522 que es el utilizado.

Los lectores con prototipo de interrogadores, depende de la etiqueta, son sofisticados, detectan y corrigen errores y trabajan a mayor frecuencia.

Tabla 4. Características RFID RC522

Corriente de operación	**13-26mA a 3.3V**
Corriente máxima	30mA
Distancia de lectura	0 a 60mm
Conexión	SPI
Dimensiones	40 x 60 mm
Costo	$45 MXN
Estándar	ISO 14443

3) Controlador: Es el encargado de acceder a la base de datos (si existiera una) y controla el software de control.

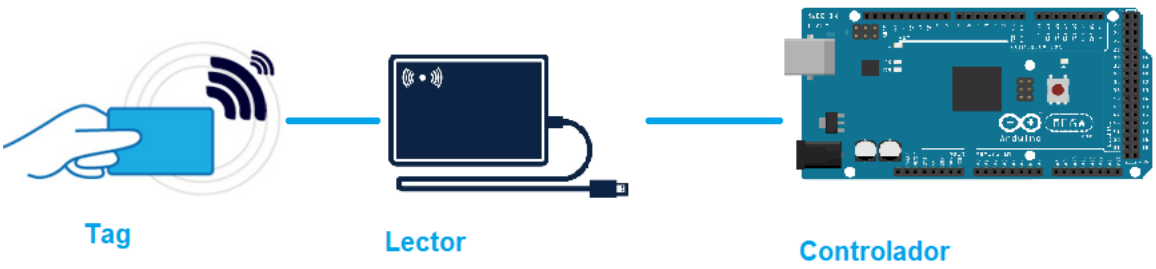

Ilustración 22. Componentes básicos de un prototipo RFID

18.3. Conexión

Es necesario hacer la conexión Arduino Mega-RFID RC522 mediante las terminales como se muestra en la Ilustración 23, de igual manera, se agrega la Tabla 5 de conexiones para este prototipo en específico, en otras fuentes se podrá encontrar que los pines de RST y SDA(SS) podrían estar conectados a otros pines, sin embargo, estos pines son configurables y pueden ser cambiados de posición, y de igual manera es posible conectar nuestra línea de alimentación a otra fuente de Arduino.

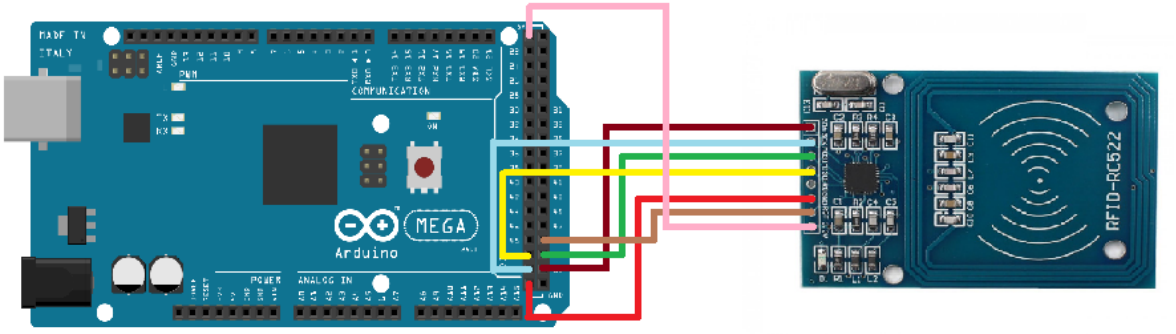

Ilustración 23.Conexión grafica Arduino-RFID

Tabla 5. Conexión Arduino-RFID

RFID	Arduino Mega
RST/Reset	49
SDA(SS)	53
MOSI	51
MISO	50
SCK	52

18.4. Software RFID

Se centró en tres actividades de la tecnología RFID para la realización del prototipo que se presenta, la primera de ellas fue obtener el identificador único de cada etiqueta, la segunda es la escritura de información en estos mismos, y la tercera, obtener dichos datos previamente almacenados. Para comenzar, es necesario mencionar todo apartado utilizado para la correcta ejecución de cualquiera de las tres actividades previamente mencionadas, como librerías y definir variables necesarias para la correcta ejecución.

Como muchos módulos, es necesario de la librería, la mayoría de las librerías que pone a nuestra disposición Arduino servirá para fin, pero es importante hacer las conexiones anteriormente mencionadas para que nuestro modulo RFID funcione con Arduino Mega. Es necesario incluir las siguientes librerías a nuestro código de Arduino:

```
#include <SPI.h>
#include <MFRC522.h>
```

Posterior a este paso, como se mencionó en el apartado 18.3. Conexión, tenemos dos conexiones que pueden ser cambiadas a voluntad del programador, sin embargo, en este caso específico los conectaremos en los pines 49, para RST, y 53 para SS, tal como se muestra a continuación.

```
#define RST_PIN 49 //configurable
#define SS_PIN 53 //configurable
```

Es necesario crear nuestra instancia del módulo MFRC522 con los pines que se han declarado anterior mente:

```
MFRC522 mfrc522(SS_PIN, RST_PIN); // crea la instancia MFRC522
MFRC522::MIFARE_Key key;
```

De igual manera se necesita iniciar el bus utilizado para SPI, es un paso igual al declarar el puerto serial en Arduino, es necesario una comunicación dicho puerto. Al mismo tiempo se inicia la tarjeta MFRC522 para poder tener comunicación.

```
SPI.begin();        // inicia el bus SPI
mfrc522.PCD_Init();   // Inicia tarjeta MFRC522
```

Es importante mencionar un paso para que el código reconozca una etiqueta. MFRC522 contiene una función que nos facilita este trabajo y solamente se necesita mandar a llamar.

```
if ( ! mfrc522.PICC_IsNewCardPresent()) {  //detecta nueva tarjeta
  return;
 }
```

18.4.1. Obtención del identificador unico de etiquetas RFID

En los pasos anteriores se mostró la forma de iniciar la tarjeta MFRC522, con esto, podemos obtener varios parámetros de la etiqueta cuando es detectada por el

lector. En este prototipo fue necesario usar una función para facilitar un poco la liberación de memoria, y como se mencionó, tener más organizado el código. Esta función es llamada “llenaId” que como su nombre lo dice, obtiene el valor del “id” único de nuestra etiqueta y los almacena en una variable para una fácil manipulación.

Es importante mencionar, ahora que todo dato que se obtenga de nuestro tag usaremos un buffer que trabaja “similar” a un arreglo, sin embargo, este trabaja con datos tipos byte. La función tiene dos parámetros para recibir, una es el buffer que se manda a la función, y el tamaño de este.

```
llenaId(mfrc522.uid.uidByte, mfrc522.uid.size);  //Se llena variable id

void llenaId(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    idTarjeta += buffer[i];
  }
}
```

18.4.2. Escritura etiquetas RFID

Para completar esta tarea, solo es necesario saber cómo trabaja un buffer y como asignarle un valor, ya que el buffer es el encargado de comunicarse con la etiqueta. De igual manera se decidió trabajar con una función, puesto que al

prototipo ingresarán diversas tarjetas y solo se deberá hacer mención de esta función. Como se hizo con anterioridad es necesario iniciar el chip MFRC522 ya que cada operación será independiente.

Se creo una variable “status”, que nos ayuda a saber el estado de nuestro procedimiento y si hay fallo en algún paso poder saberlo. Ahora, es necesario mencionar, que cuando esta función es llamada (no en todos los casos para grabar) ya se tiene un nombre guardado en un array, solo estos pueden igualarse a un buffer, y es lo que se realiza, mediante una iteración “for” de 20 posiciones, se asigna el valor de cada letra de un array al buffer.

Una vez que el buffer está lleno se asigna el tag en la posición anteriormente establecido, (este puede ser cambiado a conveniencia) junto con el buffer para saber qué información deberá ser escrita. Al finalizar estas operaciones, siempre se tiene que limpiar el buffer para que la siguiente entrada no tenga dificultades con información guardada de la anterior etiqueta.

```
void grabaNombre(){
  // Prepare key-Inicialmente chip de fabrica
  SPI.begin();        // Inicia SPI bus
 mfrc522.PCD_Init(); // Inicia MFRC522 card
 MFRC522::MIFARE_Key key;
 for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF; //hasta el bloque6
 if ( ! mfrc522.PICC_IsNewCardPresent()) {  //detecta nueva tarjeta
 return;
}
```

```
  if ( ! mfrc522.PICC_ReadCardSerial()) return;   //selecciona la tarjeta
byte buffer[34];
byte block;
MFRC522::StatusCode status;
for (byte i = 0; i < 20; i++) buffer[i] = nombre[i]; // Asigna el nombre a buffer
block = 4; //Especifica en que bloque escribirá ___------___-----___
//Serial.println(F("Authenticating using key A..."));
status  =  mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
block, &key, &(mfrc522.uid));
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("PCD_Authenticate() failed: PUERTO 4"));
  Serial.println(mfrc522.GetStatusCodeName(status));
  return;
}
status = mfrc522.MIFARE_Write(block, buffer , 16); //escribe el bloque
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("MIFARE_Write() failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
  return;
}
else Serial.println(F("MIFARE_Write() Escritura Realizada Correctamente en sector
4 "));
for (byte a = 0; a < 16; a++) {///Limpiar el buffer
String x;
buffer[a]=x[0];
}
  }
```

18.4.3. Obtener información etiquetas RFID

Este apartado es parecido al de “escribir información”, se necesita especificar de qué bloque se desea obtener la información, y eso lo tenemos con la variable tipo byte “blockAddr” y se delimita en “dataBlock”, de igual manera se obtiene dicha información y es guardada en el buffer, para posteriormente, mediante una función mandar el buffer y poder manipular, o imprimir esta información a gusto.

```
void leerRFID(){
MFRC522 mfrc522(SS_PIN, RST_PIN); // Crea MFRC522 instancia.
MFRC522::MIFARE_Key key;

SPI.begin();        // Inicia SPI bus
mfrc522.PCD_Init(); // Inicia MFRC522 card

// utiliza FFFFFFFFFFFFh que son los datos en chip de fábrica
for (byte i = 0; i < 6; i++) {  //solo el bloque6
key.keyByte[i] = 0xFF;
}
  //Serial.println(F("Ingrese Tarjeta"));
  if ( ! mfrc522.PICC_IsNewCardPresent())  //mira si hay nueva tarjeta
return;

if ( ! mfrc522.PICC_ReadCardSerial())  //selecciona la nueva tarjeta
return;

byte sector = 1;
byte blockAddr = 5; //Lee solo del bloque 5 sus 16bytes
byte dataBlock[] = {
0x01, 0x02, 0x03, 0x04, // 1, 2, 3, 4,
0x05, 0x06, 0x07, 0x08, // 5, 6, 7, 8,
0x08, 0x09, 0xff, 0x0b, // 9, 10, 255, 12,
0x0c, 0x0d, 0x0e, 0x0f // 13, 14, 15, 16
};

byte trailerBlock = 7;
MFRC522::StatusCode status;
byte buffer[18];
byte size = sizeof(buffer);
Serial.println(F("Datos actuales en el sector:"));
mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key, sector);
Serial.println();

Serial.print(F("Cargando dato del bloque ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
```

```
status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(blockAddr, buffer, &size);

if (status != MFRC522::STATUS_OK) {
Serial.print(F("MIFARE_Read() failed: "));
Serial.println(mfrc522.GetStatusCodeName(status));
}

Serial.print(F("Dato en el bloque ")); Serial.print(blockAddr);
Serial.println(F(":"));
imprimir(buffer, 16);//DATO EN DECIMAL

 }
```

19. TFT-LCD

La tecnología TFT-LCD (Transistor de película delgada-Pantalla de cristal líquido), entro en venta en 2006, usa variantes de un simple LCD y como su nombre lo dice, incorpora un transistor de película delgada para mejorar la calidad. Estos dispositivos son normalmente usados en televisores y/o proyectores. En el área de informática, TFT ha desplazado en la competencia a los monitores CRT (tubo de rayos catódicos).

19.1. LCD

La pantalla de cristal líquido es plana, que está formada por un número determinado de pixeles en color, o de un solo color colocados delante de una fuente

de luz. Cada píxel consiste en una capa de moléculas alineadas entre dos electrodos transparentes, y dos filtros de polarización. Si no se coloca un cristal líquido entre el filtro polarizante, la luz del primer filtro no pasaría al polarizado.

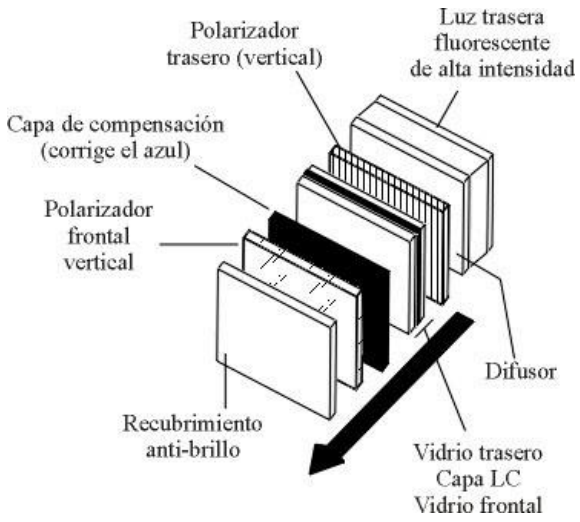

Ilustración 24. Representación en capas de LCD. Fuente: http://www.usmp.edu.pe

19.2. Especificaciones

- Resolución: Es la dimensión tanto horizontal como vertical, que son representadas en los pixeles que estas contienen.
- Ancho de punto: Es la distancia que tienen dos pixeles adyacentes, tanto vertical como horizontal.
- Tamaño: Es el tamaño del LCD, se mide a lo largo de su diagonal, y es expresado en pulgadas.
- Tiempo de respuesta: Tiempo en que le toma a un píxel cambiar de un color a otro.
- Tipo de matriz: Activa, pasiva y reactiva
- Angulo de visión: Es el ángulo en que puede verse el LCD sin que la imagen pierda calidad de visión.

- Soporte de color: Colores soportados.
- Brillo: Cantidad de luz emitida.
- Contraste: Relación entre la intensidad oscura y brillante.
- Aspecto: Anchura y altura.
- Puertos de entrada: DVI (Digital Visual Interface), VGA (Video Graphics Array), HDMI (High-Definition Multimedia Interface). Aunque el más común es el puerto USB (Bus en Universal en Serie)

19.3. Color

Cada píxel que muestra un LCD, es dividido en tres subpíxeles, rojo, verde y azul respectivamente. Cada uno puede manipularse independientemente para poder generar millones de colores posibles para cada píxel. Esto también es usado en los monitores CRT pero estos usan fósforo, y el haz de electrones empleados no dan el número exacto de subpíxeles.

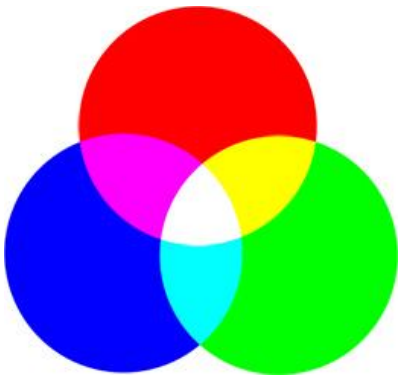

Ilustración 25. Combinaciones RGB.

19.4. Matrices activas y pasivas

Las pantallas LCD contienen sectores que tienen contactos eléctricos individuales para cada segmento. Un circuito, es el encargado de mandar carga eléctrica para cada segmento. Las pantallas de un solo color, también llamadas monocromo, como algunos relojes u organizadores antiguos, son un ejemplo de estos, son ejemplos de matriz pasiva. En cambio, las pantallas a color de alta resolución utilizan una estructura matriz activa TFT, donde cada píxel tiene su propio transistor dedicado,

Ilustración 26. Ejemplo LCD como matriz pasiva / matriz activa.

19.5. Incovenientes

1) Contraste: Los LCD tienen imágenes de mejor calidad que los CRT, y mejor contraste del mundo real, esto quiere decir que mantienen el contraste y la variación del color en ambientes luminosos, pero tiene menor contraste en términos de la profundidad de negros.
2) Tiempo de respuesta: LCD suele tener tiempo de respuesta más lentos que un monitor CRT, o hasta un plasma, en especial, las viejas pantallas que pueden crear pantallas extra cuando se manipulan con mucha rapidez.

3) Angulo de visión: Los LCD tienen un ángulo de visión un poco más limitado que las CRT, esto puede causar que varios usuarios no puedan ver la misma imagen con mejor calidad.
4) Durabilidad: De igual manera, los LCD son más frágiles que sus correspondientes plasma o CRT.

20. TFT

Como sus siglas lo indica, un transistor de películas finas es un transistor de efecto, que se fabrica poniendo películas muy finas de un semiconductor activo y una capa de material dieléctrico (baja conductividad eléctrica) sobre un sustrato. El más común es el vidrio, y que su principal aplicación en las pantallas LCD.

Pueden ser fabricados con varios materiales semiconductores, pero el más común es el silicio. Las cualidades y características de un TFT dependen de su estado cristalino. A partir del 2008, TFT empezó a integrarse a las pantallas LCD, esto ayudó a reducir la diafonía o ruido entre pixeles, y mejorar la estabilidad de la imagen.

La composición de un circuito TFT-LCD, es muy similar a la que se usa en una memoria DRAM (Memoria dinámica de acceso aleatorio) de una computadora, pero en vez de usar los transistores usando silicio, estos son fabricados, depositando una película de silicio sobre un vidrio. Los transistores, solo usan una fracción del área de cada píxel.

20.1. TFT LCD Touch 2.4” para Arduino

Esta pantalla, tiene la función de mostrar datos al usuario, y tiene la facilidad del funcionamiento touch, para así facilitar el uso de esta y dar un aporte “friendly”. Tiene la facilidad que es un módulo echo especial para Arduino, es compatible para Arduino Mega y gracias a esto facilita su manipulación. Sus tres funciones principales son:

1) Pantalla LCD para visualización de gráficos o texto
2) Panel táctil sobre la pantalla LCD
3) Lector de tarjetas MicroSD para datalogging o almacenamiento de imágenes.

20.2. Caracteristicas

- 4 luces de fondo LED blancas, por defecto, pero puede conectar el transistor a un pin digital para controlar la retroiluminación.
- 18 bits 262,000 tonos diferentes.
- Pantalla táctil resistiva de 4 hilos.
- Resolución de 240 × 320.
- Controlador 9325 con buffer de RAM de video incorporado.
- Interfaz digital de 8 bits, más 4 líneas de control.
- Utiliza los pines digitales 5-13 y analógico 0-3. Eso significa que puede usar los pines digitales 2, 3 y los analógicos 4 y 5. El pin 12 está disponible si no usa el microSD.
- Compatible con 5 V, uso con lógica de 3.3V o 5V.

- Tamaño 71 * 52 * 7 mm
- Peso aproximado de 31g.

20.3. Conexión con Arduino Mega

Ya que este módulo, fue hecho especialmente para Arduino, no es difícil conectarlo a una placa de estas, basta con colocarlo como lo muestra en la Ilustración 27 y haciendo coincidir los pines para que la conexión sea la adecuada.

Ilustración 27. Conexión TFT-LCD con Arduino Mega

20.4. Librerías TFT-LCD

Es necesario hacer uso de cuatro librerías de Arduino para poder ejecutar todo el funcionamiento de la pantalla TFT, ya que en este prototipo se hace uso del touch para el usuario, la tarjeta microSD para almacenamiento de información, visualizar texto y/o figuras en la pantalla.

La primera librería es **Ardafruir_TFTLCD,** que es la encargada de incluir los drivers de todas las pantallas ya que, aunque a simple vista, tengan el mismo

aspecto, el modelo puede cambiar y por ende el chip también, los chips más comunes para este tipo de pantallas son **ILI9325**, ILI9341, HX8347G y HX8357, el modelo del chip dependerá del fabricante, sin embargo, esto no afecta el funcionamiento final de la pantalla. Este es un paso muy importante, porque en el código de Arduino, este chip debe ser especificado, de lo contrario, no funcionará el TFT y no visualizará ninguna imagen.

La segunda librería es **Ardafruit_GFX,** es la encargada de dar las herramientas, como palabras reservadas y ejemplos de cómo realizar dibujos y figuras geométricas (puntos, círculos, triángulos, líneas, etc.) dentro del código, y posteriormente poder visualizar esto en la pantalla.

La tercera librería es la librería **TouchScreen,** como su nombre lo dice facilita la tarea de poder manipular la pantalla mediante el cristal, con ayuda del usuario, y que se encarga de traducir las coordenadas para poder trabajar con el táctil resistivo de cuatro hilos que incluye el módulo.

Finalmente, la librería **SD,** ya está incluida en el entorno de Arduino y que, gracias a la pantalla, ya tiene una ranura de SD incluida, podemos acceder a ella fácilmente con esta herramienta. Es importante usar las librerías adecuadas para Arduino mega, puesto que estas mismas librerías, pero con variantes, están hechas para otros tipos de Arduino.

20.5. Software TFT-LCD

Como en los otros módulos que fueron mencionados, se tiene que especificar todas las librerías, y/o variables a declarar para poder hacer uso de la pantalla, lo primero que se necesita, es declarar las librerías que fueron mencionadas. Al mismo tiempo, tienen que estar declaradas cada una de las conexiones que tiene la pantalla con Arduino, y especificar en qué pines se están conectando para posteriormente hacer uso de estas. Primero se mencionarán las librerías.

```
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_TFTLCD.h> // Hardware-specific library
#include "TouchScreen.h"     // Liberia Touch
#include <SD.h>              // Librería de tarjeta SD
```

Para poder hacer uso de la microSD, se tiene que mencionar los pines que se usaron en el Arduino, y que al mismo tiempo están conectadas a la pantalla, así podremos tener una mejor manipulación de estas variables.

```
#define DEBUG
#if defined __AVR_ATmega2560__ // Para Arduino Uno/Duemilanove, conectamos la tarjeta SD en los pines del puerto SPI
#define SD_SCK 13   // que se corresponden con los pines MOSI -> 11, MISO -> 12 y SCK -> 13
#define SD_MISO 12
#define SD_MOSI 11
#define SD_CS 10
#endif
```

Pines para conexión del LCD:

#define LCD_CS A3 // Chip Select - Pin analogico 3

#define LCD_CD A2 // Command/Data - Pin Analogico 2

#define LCD_WR A1 // LCD Write - Pin Analogico 1

#define LCD_RD A0 // LCD Read - Pin Analogico 0

#define LCD_RESET A4 // LCD Reset - Pin Analogico 4

Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET); // Instancia del LCD

Pines necesarios para los cuatro conectores del panel táctil, al mismo tiempo se tiene que delimitar el tamaño donde será posible hacer uso del touch y la presión máxima y mínima posible.

#define YP A1 // Pin analógico A1 para ADC

#define XM A2 // Pin analógico A2 para ADC

#define YM 7

#define XP 6

short TS_MINX = 150; // Coordenadas del panel tactil para delimitar

short TS_MINY = 120; // el tamaño de la zona donde podemos presionar

short TS_MAXX = 850; // y que coincida con el tamaño del LCD

short TS_MAXY = 891;

#define MINPRESSURE 1

#define MAXPRESSURE 500

TouchScreen ts = TouchScreen(XP, YP, XM, YM, 364);

Como se mencionó en el apartado de Color, se pueden hacer las combinaciones de colores necesarias, o las que sean ocupadas, podemos usar su nombre como variable y no el código hexadecimal de cada uno, aquí algunos ejemplos de los colores más usados. Al igual como se mostró, se necesita especificar que chip es usado en nuestra pantalla TFT y podremos especificarlo al iniciar el código.

```
#define BLACK

#define RED     0xF800

#define GREEN   0x07E0

#define WHITE   0xFFFF

#define BLUE    0x001F

#define YELLOW  0xFFE0

#define MAGENTA 0xF81F

#define CORAL   0xF08080

#define BLUE2   0x00BFFF

#define ORANGE  0xFBE0

#define DARCKGRAY 0xBDF7
```

uint16_t identifier = 0x9325; //IDENTIFICADOR "CHIP DE TFT (9325)"

Se necesita iniciar la pantalla para que sea posible visualizar una imagen o texto, gracias a la librería de TFT, es posible hacer estas tareas, como asignar orientación de la pantalla (horizontal o vertical), establecer un color de fondo, el tamaño de la letra, el color de esta misma, colocar el cursor en una posición en específica, pero es importante no olvidar hacer referencia al chip que se está usando.

tft.reset();

tft.begin(identifier);

tft.setRotation(1); // Establecemos la posición de la pantalla Vertical u Horizontal

tft.fillScreen(BLUE); // Colocamos el fondo del LCD en Azul

tft.setTextSize(2); // Definimos tamaño del texto. (Probado tamaños del 1 al 10)

tft.setTextColor(BLACK); // Definimos el color del texto

tft.setCursor(60,0); // Situamos el cursor en la posición del LCD deseada, (X, Y) siendo X el ancho (240 px max.) e Y el alto (320 px max.)

De igual manera tenemos que iniciar cada uno de los componentes que ya hemos declarado.

if (!SD.begin(SD_CS, SD_MOSI, SD_MISO, SD_SCK)) {

return;

}

MFRC522 mfrc522(SS_PIN, RST_PIN); // Crea MFRC522 instancia.

SPI.begin(); // Inicia SPI bus

mfrc522.PCD_Init(); // Inicia MFRC522 card

}

20.5.1. Visualización de gráficos en TFT-LCD

Podemos hacer uso de figuras geométricas para hacer botones, gráficos o solo el diseño de la imagen, tenemos la posibilidad de hacer círculos, triángulos, cuadrados o rectángulos, estos pueden ir rellenos, con o sin borde y dependerá del programador.

tft.drawRect(20, 125, 200, 25, YELLOW); // Dibujamos un cuadrado/rectángulo sin color de relleno (Punto inicial X, Punto inicial Y, Longitud X, Longitud Y, Color)

tft.fillRect(20, 165, 60, 60, BLUE); // Dibujamos un cuadrado/rectángulo relleno de color (Punto inicial X, Punto inicial Y, Longitud X, Longitud Y, Color)

tft.drawCircle(120, 195, 30, WHITE); // Dibujamos un circulo sin color de relleno Punto inicial X, Punto inicial Y, Radio del círculo, Color)

tft.fillCircle(120, 195, 20, WHITE); // Dibujamos un circulo relleno de color, Punto inicial X, Punto inicial Y, Radio del círculo, Color)

tft.drawTriangle // Dibujamos un triángulo sin color de relleno

(190, 163, // (Vértice superior X, Vértice superior Y,

160, 225, // Vértice inferior izquierdo X, vértice inferior izquierdo Y,

222, 225, CYAN); // Vértice inferior derecho X, Vértice inferior derecho Y, Color)

tft.fillTriangle // Dibujamos un triángulo relleno de color

(190, 240, // (Vértice superior X, Vértice superior Y,

160, 302, // Vértice inferior izquierdo X, vértice inferior izquierdo Y,

222, 302, MAGENTA); // Vértice inferior derecho X, Vértice inferior derecho Y, Color)

tft.drawRoundRect(20, 245, 130, 60, 20, RED); // Dibujamos un cuadrado/rectángulo con los bordes redondeados sin color de relleno (Punto inicial X, Punto inicial Y, Longitud X, Longitud Y, Radio de los vértices, Color)

tft.fillRoundRect(35, 255, 100, 40, 15, YELLOW); // Dibujamos un cuadrado/rectángulo con los vértices redondeados relleno de color (Punto inicial X, Punto inicial Y, Longitud X, Longitud Y, Radio de los vértices, Color)

20.5.2. Panel tactil

Gracias a las variables que fueron declaradas anteriormente, podemos hacer uso del panel táctil. En este prototipo se puede manipular todo mediante este panel, y se delimitó el touch a coordenadas, dependiendo donde el usuario presione se activará o desactivará una función en específico.

```
TSPoint p = ts.getPoint(); // Obtenemos la lectura del panel

pinMode(XM, OUTPUT);

pinMode(YP, OUTPUT);

uint16_t maxscroll;

if (tft.getRotation() & 1) maxscroll = tft.width();

else maxscroll = tft.height();

// Mapeamos los valores analógicos leídos del panel táctil (0-1023)

// y los convertimos en valor correspondiente a la medida del LCD 320x240

p.x = map(p.x, TS_MAXX, TS_MINX, tft.width(), 0);

p.y = map(p.y, TS_MINY, TS_MAXY, tft.height(), 0);

if (p.z > MINPRESSURE )   // Si se detecta presión sobre el panel

{

//SE DELIMITA EL RANGO DE PIXELES DE CADA BOTON Y SE ACTIVA CADA UNO DEPENDIENDO EL CASO
```

if ((p.x > 210 && p.y > 70) && (p.x < 300 && p.y < 160)) {

tft.print("Se detecto presión en esta área");

}

20.5.3. Manejo SD

Ya se comentó que, en este prototipo, la manera en que los códigos fueron organizados con funciones y por esta razón es que fueron programadas para grabar y al mismo tiempo, obtener la información escrita, pero en funciones diferentes, leer los datos dentro de un fichero tipo .txt. para empezar, se mostrará el procedimiento para poder abrir un fichero y guardar información dentro de estos.

“Procedimiento ingresar información al fichero.”

```
File Archivo; // Se declara la variable tipo File
/* ESCRIBIENDO DATOS EN LA MEMORIA SD DE ARDUINO */
//Se abre el documento sobre el que se va a leer y escribir.
Archivo = SD.open("ID.txt", FILE_WRITE);
//Se comprueba que el archivo se ha abierto correctamente y se procede a
//escribir en él.
if (Archivo) {
Archivo.print("Texto a escribir en .txt");
Archivo.println();
Archivo.close(); // Se cierra la dirección del archivo
Serial.println("Todos los datos fueron almacenados en IDs");
}else{
Serial.println("Error al abrir ID");  }
```

“Procedimiento para acceder a la información del fichero.”

```
/* LEYENDO DATOS EN LA MEMORIA SD DE ARDUINO */

//Se vuelve a abrir el fichero, esta vez para leer los datos escritos.
Archivo = SD.open("ID.txt");
//Si el archivo se ha abierto correctamente se muestran los datos.
if (Archivo) {
//Se muestra por el monitor que la información que va a aparecer es la del
Serial.println("Información contenida en ID.txt: ");
while (Archivo.available()) {
//Se escribe la información que ha sido leída del archivo.
Serial.write(Archivo.read());    }
//Si todo ha ido bien cierra el archivo para no perder datos.
Archivo.close(); }
else {
Serial.println("El archivo datos.txt no se abrió correctamente");  }
```

Capítulo IV. Programación y desarrollo de prototipo

21. Back-end

También conocido como motor, es un término que se refiere a una de las dos separaciones que existen en un prototipo, donde back-end es el encargado de interactuar no con el usuario final, sino que es la parte donde toma funcionalidad un programa, se encarga de manipular los datos y desarrollar sus funcionalidades.

21.1. Registro de usuarios (Back-end)

En los apartados anteriores, ya se demostró cómo es posible hacer uso de cada uno de los módulos a utilizar, aquí se mostrará el desarrollo y funcionamiento de algunas de las funciones de este prototipo, empezando por el Back-end, que es la primera etapa, el registro de usuarios. Para comenzar, es necesario comentar cómo es posible la creación y navegación de cada una de las pantallas que se mostraran a continuación para hacer el recorrido correcto.

Como se muestra en la siguiente Ilustración, es posible viajar entre pantallas gracias a “banderas”, utilizadas en cada una de estas, y dependiendo en donde este posicionado un usuario, dicha bandera se encontrará activada o desactivada. Puesto que cada pantalla es única, es necesario dibujar lo que se quiere mostrar al usuario y dar las restricciones del touch, que anteriormente se mostró la funcionalidad de esto.

```
void loop(void)
{
  /*
   * Cada pantalla tiene su bandera y dependiendo
   * de que vandera este activa entrara el (touch)
   * de cada pantalla (visual)
   */
 if  (listoNombre==false ) { entraNombre(); }
 if  (listoFecha==true ) { entraFecha(); }
 if  (listoMenu==true){entraMenu();}
 if  (listoRenovar==true){entraRenovar();}
}
```

Ilustración 28.Loop de Back-end.

La imagen siguiente, ejemplifica cómo es posible crear una pantalla con el fondo azul, un texto de encabezado “Elije una opción”, un cuadrado con relleno blanco y un marco negro en la posición “60,80” y con un tamaño de “85x85”, y dentro de este se escribe la palabra “Nuevo”. El tamaño, el color, el texto y la posición, depende del desarrollador.

```
tft.fillRect(0,0 ,tft.width(), tft.height(), BLUE2 ); // Se pinta el fondo del mismo color para bor
tft.setCursor(70,20);
tft.println("Elige una opcion");
tft.setTextColor(WHITE);
tft.fillRect(60,80 , 85, 85, BLACK); // Dibujamos un cuadrado/rectangulo relleno de color /(Punto i
tft.drawRect(60, 80, 85, 85, WHITE); // Dibujamos un cuadrado/rectangulo sin color de relleno// (Pu
tft.setCursor(70,110);
tft.print("Nuevo");
```

Ilustración 29. Código dibujar un cuadrado.

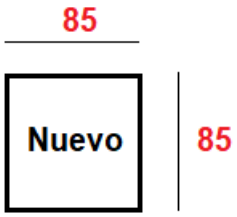

Ilustración 30. Ejemplo cuadrado dibujado con código.

Del mismo modo que se realizó con el cuadrado, es necesario hacer círculos para poder crear, en este caso. teclados numéricos-alfabéticos, esto fue creado con circunferencias blancas con contorno negro, radio, y se introdujo un número o una letra dependiendo el caso, solamente cambiando la dirección de posicionamiento es posible la creación de un teclado.

```
tft.fillCircle(20, 45, 20, WHITE); // Dibujamos un circulo relleno de color//(Punto inicial X, Pu
tft.drawCircle(20, 45, 20, BLACK);// Dibujamos un circulo sin color de relleno//(Punto inicial X,
tft.setCursor(20,45);
tft.print("Q");
```

Ilustración 31. Código dibujar un Círculo.

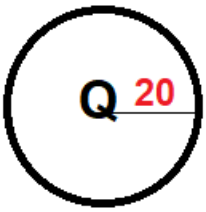

Ilustración 32. Ejemplo círculo dibujado con código.

Para poder almacenar datos en variables con estos teclados que fueron creados, es necesario usar el panel táctil de la pantalla. Se especifica el funcionamiento del touch, y que gracias a esto es posible recolectar información dependiendo en donde se detecte presión.

Por ejemplo, en el siguiente código se muestra un rango de posición, que delimita a un cuadrante de la pantalla, y que si es detectada alguna presión se active esta instrucción. Se muestra una variable llamada “nombre” en donde se agregan valores si este espacio es activado. Estas líneas de código se encuentran en todo el programa, pero variando el tipo de variable que están llenando, y el tipo de teclado que es utilizado.

```
if ( (p.x > 323 && p.y > 170)  &&  (p.x < 360 && p.y < 210) )

  {
```

```
    tft.print("Q");
    nombre += "Q";
}
```

21.2. Muestra de usuarios (Back-end)

El centro de este prototipo es una matriz tridimensional, en donde, se van guardando todos los datos ingresados (nombre, fecha, hora de entrada y hora de salida), en una fila se almacena el nombre, la siguiente después de esta se tendrán los demás datos, y cuando esta matriz es llenada avanza a la siguiente matriz, sin eliminar los datos anteriores.

byte Matriz[9][14][26];// 9 MATRIZ, 14 FILAS(QUE ADMITE LA PANTALLA) 26 COLUNMNAS

Tiene que ser declarada la matriz de tipo byte, puesto que todos los datos que son extraídos de las etiquetas son de este tipo de dato, fueron declaradas nueve matrices de catorce filas, y veintiséis columnas que son las aceptadas por la pantalla. Nueve son el número de matrices aceptadas por el prototipo para mostrar datos, ya que ocupa memoria, sin embargo, los datos nunca se pierden.

En la Ilustración 33, se muestran los datos que son extraídos de las etiquetas, y son almacenados en la matriz ya mencionada, en donde cada espacio representa un carácter que será representado en la pantalla pero que, en la pantalla cada dato es representado con tipo carácter. En la primera línea, hay un total de 16 caracteres, y que son el máximo de caracteres aceptados en un sector de una etiqueta RFID, en donde cada letra, número y carácter especial tiene un numero equivalente al formato byte, por ejemplo:

Tabla 6. Representación tipo Byte a Char

Byte	**68**	**65**	**86**	**73**	**68**	**32**	**77**	**73**	**83**	**65**	**69**	**76**	**32**	**71**	**65**	**82**
Char	D	A	V	I	D		M	I	S	A	E	L		G	A	R

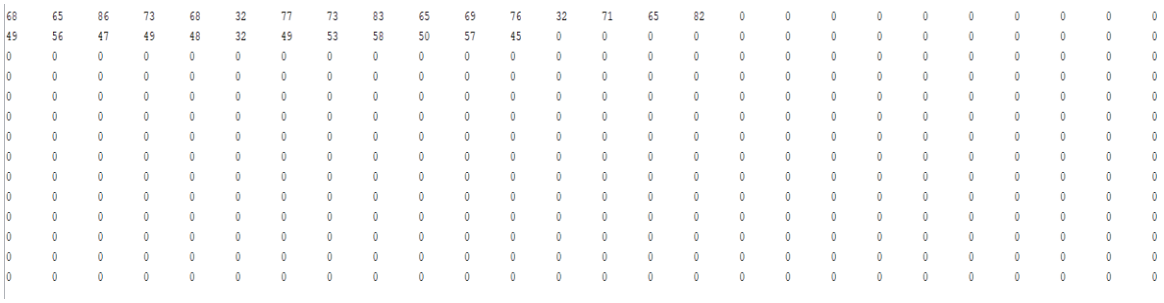

Ilustración 33. Datos tipo byte dentro de la matriz

Para navegar entre pantallas, se usarón dos métodos, y al mismo tiempo la pantalla fue dividida en dos, dependiendo de donde sea presionado cambiará a su posterior o anterior, en este cambio, pasa escribiendo los datos de la matriz en su respectivo lugar tal y como se muestra en la Ilustración 34.

```
void moverPantallaDerecha(){
     numPantalla++;
     tft.fillRect(0, 0, tft.width(), tft.height(), BLUE2);
     tft.setCursor(0,0);

    for (byte i = 0; i < 14; i++) {
     for (byte j = 0; j < 26; j++) {
      if (i%2==0){ tft.setTextColor(RED);}
      else{ tft.setTextColor(BLACK);}
      if(Matriz[numPantalla][i][0]!=vacio[0]){tft.print((char)Matriz[numPantalla][i][j]); }
    }
  }
  }

void moverPantallaIzquierda(){
  if (numPantalla==0){
      numPantalla=0;
      }else{numPantalla--;}

      tft.fillRect(0, 0, tft.width(), tft.height(), BLUE2);
      tft.setCursor(0,0);

    for (byte i = 0; i < 14; i++) {
     for (byte j = 0; j < 26; j++) {
      if (i%2==0){ tft.setTextColor(RED);}
      else{ tft.setTextColor(BLACK);}
          if(Matriz[numPantalla][i][0]!=vacio[0]){tft.print((char)Matriz[numPantalla][i][j]); }
    }
  }
  }
```

Ilustración 34. Funciones avanzar izquierda o derecha.

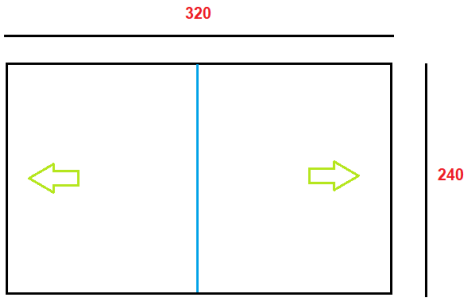

Ilustración 35. Función touch para cambio de pantalla.

La función “void” de este código, solamente contiene dos métodos, “iniciarTFT” y “leerRFID”, en donde la primera hace referencia al funcionamiento táctil de la pantalla, y que en todo momento pueda hacerse uso de este método, la segunda

se encarga, como su nombre lo dice, de leer toda etiqueta que sea detectada por el módulo RFID.

```
void loop(void)

{
  iniciarTFT();
  leerRFID();
}
```

Cuando una etiqueta es detectada tiene que pasar por dos funciones filtro, que son las encargadas de dar acceso al usuario. La primera función, es llamada “tarjetaRegistrada”, donde se busca en los ficheros el identificador único de la etiqueta que está ingresando, de lo contrario, hasta aquí se corta comunicación.

El comportamiento de este método es obtener todos los datos del fichero que contiene los identificadores de las tarjetas que ya fueron registradas, y compararlos con la etiqueta que entra al momento, si esta función no encuentra una similitud envía un booleano “falso”, o de lo contrario un “verdadero”.

```
bool tarjetaRegistrada(){
  Archivo = SD.open("ID.txt");
  if (Archivo) {
    String textoTotal;
    int valor=(Archivo.size());
    byte StringId[valor];
    int s=0;
    while (Archivo.available()) {
     StringId[s] =  Archivo.read();// Se llena variable StringId con los datos de ID.txt
     s++;
    }
    Serial.println();
    String IDs;// String donde se guardan los valores de (StringId) de byte a string
    char charBuf;
    for (byte i=0;i<valor;i++){
       IDs += (char)StringId[i];//Se guardan los valores
        // Serial.print((char)StringId[i]);
      }
   String idArchivado;
   int totales= IDs.length()/12;
   //Se busca hasta encontrar el un id identico
   for (int i=0;i<IDs.length();i++){
     idArchivado += IDs[i];
     if (idArchivado == idTarjeta)
     {
      Serial.println("Encontrado!");
      return true;
      }
      if(idArchivado.length()%12 == 0 ){
      idArchivado="";
      }// Si llega al limite de cada id en txt
      if (i+1==IDs.length()){
        return false;
        }
    }
idArchivado="";
    //Si todo ha ido bien cierra el archivo para no perder datos.
    Archivo.close();
  }
```

Ilustración 36. Código función "tarjetaRegistrada".

El segundo método es “validaFechaTarjeta”, entra en ejecución cuando ya paso la primera antes ya mencionada. Es encargada de dar acceso al usuario, únicamente si la fecha que contiene la tarjeta entrante es la adecuada para ser aceptada por el prototipo, de lo contrario, se corta comunicación.

La funcionalidad de este método es gracias a que en el sector cinco de cada tarjeta es grabada la fecha límite de cada usuario, y tiene como trabajo, obtener estos datos, que son almacenados en los primero cuatro lugares de dicho sector y compararlos con el día y mes actual. Hay tres diferentes casos para poder aceptar, o negar la entrada:

1. Si el mes obtenido es mayor al actual, retornará un booleano “verdadero”.
2. Si el mes obtenido es igual al actual, pero el día obtenido es mayor o igual al actual retornará un booleano “verdadero”.
3. Si el mes obtenido es menor al actual retornará un booleano “falso”.

```
  String dia;
  String mes;
//Se llenan las variabels dia y mes obtenidas del buffer del sector 5 de la tarjeta
for (int a = 0; a < 2; a++) {
 dia+=(char)buffer[a];
}
for (byte a = 2; a < 4; a++) {
 mes+=(char)buffer[a];
}
/*
 * Si el mes es mas grande que el mes actual verifica el pago
 * Si el mes es el mismo que el mes actual verifica que el dia sea menor o igual al actual para ver
 * De lo contrario no pasara la tarjeta
 */
if (mes.toInt() > month()){
  return true;
  }

if (mes.toInt() == month() ){
  if ( dia.toInt() <= day() ){
     return true;
     }else{
      Serial.println("Denegado por fecha");
      return false;
      }
    }
if (mes.toInt()< month()){
   Serial.println("Denegado por fecha");
  return false;
  }
```

Ilustración 37. Código función "validaFechaTarjeta".

22. Front-end

Es también llamado interfaz, y como su nombre lo indica, se refiere a la usabilidad e “interfaz” del prototipo, recogerá instrucciones y entradas que el usuario le da a través de controles, sensores y áreas de control que el mismo front-end habilita y normalmente mostrará el resultado final.

22.1. Registro de usuarios (Front-end)

En la Ilustración 38 se muestra la pantalla de inicio del prototipo, se muestran dos opciones, una es para dar de alta a un nuevo usuario, y se requiere, nombre, la fecha de cuando se vencerá su mensualidad, y teléfono, en la segunda, únicamente, es para usuarios que ya fueron dados de alta, únicamente realizaran su pago y se actualiza la fecha de vencimiento de plazo, para poder hacer esto se ingresa la fecha y el teléfono.

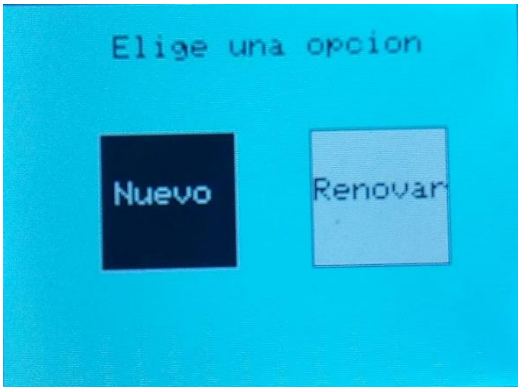

Ilustración 38. Primera pantalla Back-end

Si se selecciona la primera opción, se desplegará la segunda pantalla, en donde aparece un teclado (dibujado con las herramientas ya mencionadas), y se coloca el

nombre de la persona que quiera ser ingresada, una vez ingresado, es posible hacer corrección del nombre o de lo contrario continuar a la siguiente pantalla.

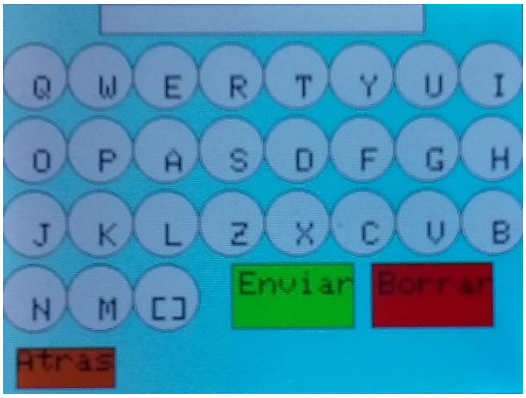

Ilustración 39. Segunda pantalla Back-end (vacía)

Ilustración 40.Segunda pantalla Back-end (llena)

La tercera pantalla, es la encargada de recolectar la fecha (próxima fecha hasta que expire la mensualidad actual), una vez capturados estos datos, es necesario ingresar el teléfono de diez dígitos, que es el estándar para cualquier número telefónico, de igual manera, en la segunda pantalla aparecerá un

teclado, pero en este caso será únicamente numérico, si no se tienen estos datos completos no se podrá avanzar al último paso.

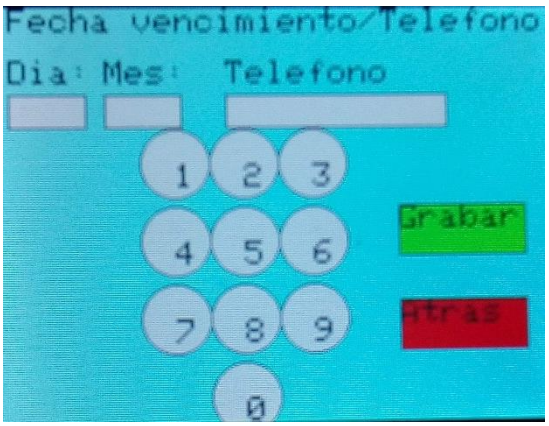

Ilustración 41.Tercera pantalla Back-end (vacía)

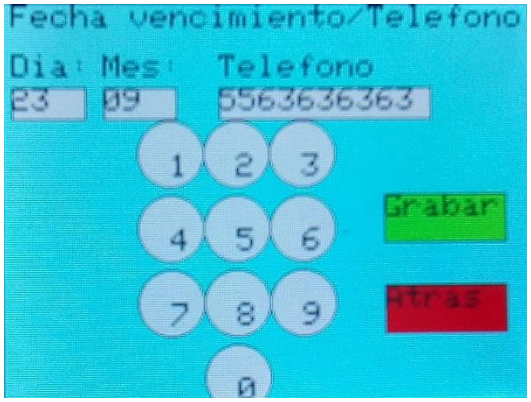

Ilustración 42.Tercera pantalla Back-end (llena)

Al acabar este procedimiento, todos los datos que fueron ingresados son almacenados en ficheros ".txt", que serán utilizados posteriormente. El fichero .txt, tiene la posibilidad de ser modificado en una posterior mejora que pudiera aplicarse en un futuro, y tiene la facilidad de poder visualizar los datos sin ningún software externo. Únicamente, se podrán dar tres diferentes resultados de lo anterior

mostrado, el primero, será un almacenado y grabado correctamente, el segundo se mostrará una leyenda, en el caso de que la tarjeta mostrada ya este registrada en los ficheros, y no dejara guardar en dicha tarjeta, y el tercer resultado, es que el tag no fue colocado en el lector y no fue posible el grabado.

22.2. Muestra de usuarios (Back-end)

La primera pantalla, es de carga, porque este es el momento que el prototipo toma su tiempo para buscar en los ficheros todos los números telefonicos, y al mismo tiempo, compara las fechas para saber a que persona, se le tiene que enviar el mensaje de texto, el procedimiento ya fue especificada en el apartado GSM SIM900. Al finalizar este proceso, el prototipo estará listo para leer etiquetas, y funcionara con normalidad.

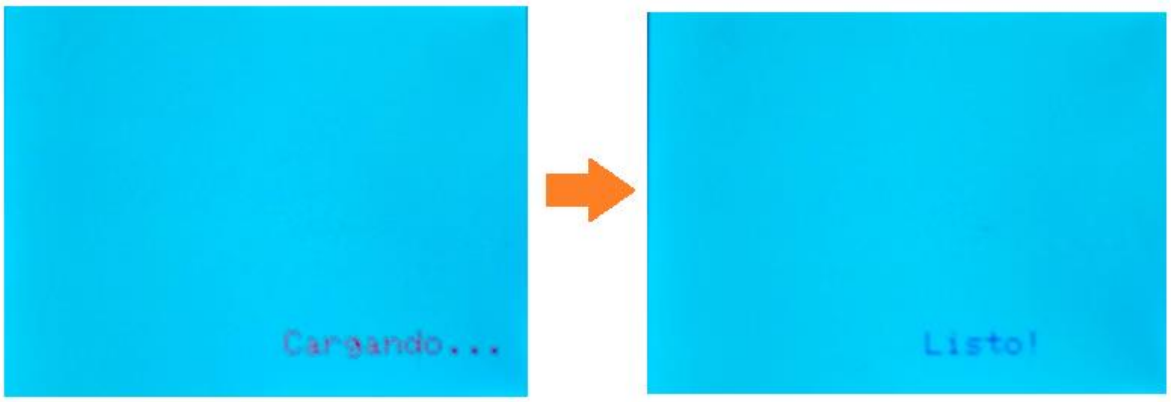

Ilustración 43. Avance de pantalla Front-end

Una vez que la pantalla y el lector están listos, se pueden pasar las etiquetas por el módulo RFID y muestra los datos de esta etiqueta, siempre y cuando, esta

sea aceptada por el prototipo, puesto que este tiene algunas restricciones, si esta persona ya está registrada, saber y verificar que la mensualidad de esta tarjeta esté en tiempo, de lo contrario la pantalla no tiene ningún efecto a la presencia de esta etiqueta. Ahora, si pasa por estos parámetros, se muestra el nombre de la persona junto con la fecha y la hora en la que fue detectada, tal como se muestra en la Ilustración 44.

Ilustración 44. Pantalla con nombre de usuario (entrada).

En este momento el prototipo ya está haciendo su función, puede aceptar a los usuarios que sean detectados por el sensor, pero existe otro momento, cuando la persona que ya estaba anotado en la pantalla termina con su estancia en el establecimiento, y es aquí donde se tiene que detectar la misma etiqueta, anotar la hora de salida para que así de término con este proceso, y si se requiere, volver a iniciar como se muestra en la Ilustración 45.

Ilustración 45. Pantalla con nombre de usuario (salida).

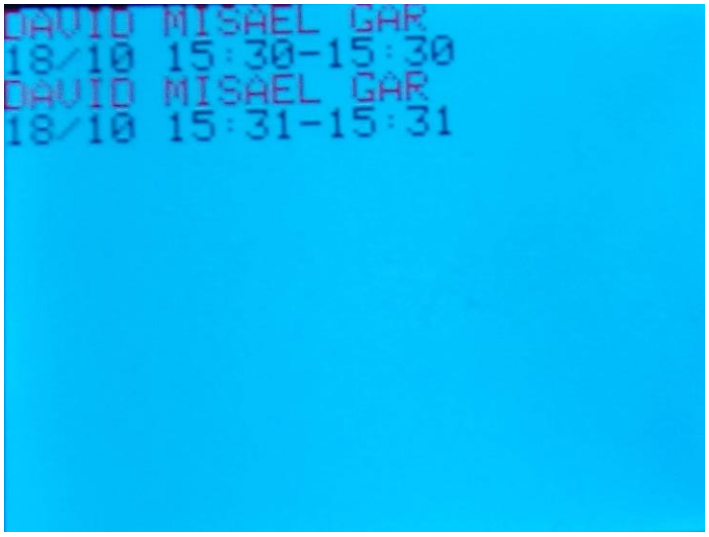

Ilustración 46. Pantalla con nombre de varios usuarios.

Como ya se menciono, se usaron ficheros para almacenar la información guardada y/o comparar a usuarios. En total, se necesitan usar cuatro ficheros distintos, con funciones diferentes cada uno. El primero de estos es “datos”, el encargado de almacenar todos los datos de cada uno de los usuarios que acuden a registrarse, contiene su identificador unico, nombre, fecha, y número de telefono. El segundo es “ID”, donde se guardan únicamente los identificadores ,y de este fichero se puede comparar que usuarios si estan dentro del sistema.

El tercer fichero con el nombre “fechaTel”, es el encargado para enviar los mensajes de texto, ya que este contiene, el día, mes y teléfono de los usuarios, si es seleccionado para enviar el mensaje lo hará. El ultimo fichero, “reg”, es el almacenamiento principal, es donde se guardan todos los movimientos del prototipo, nombre, hora de entrada y hora de salida.

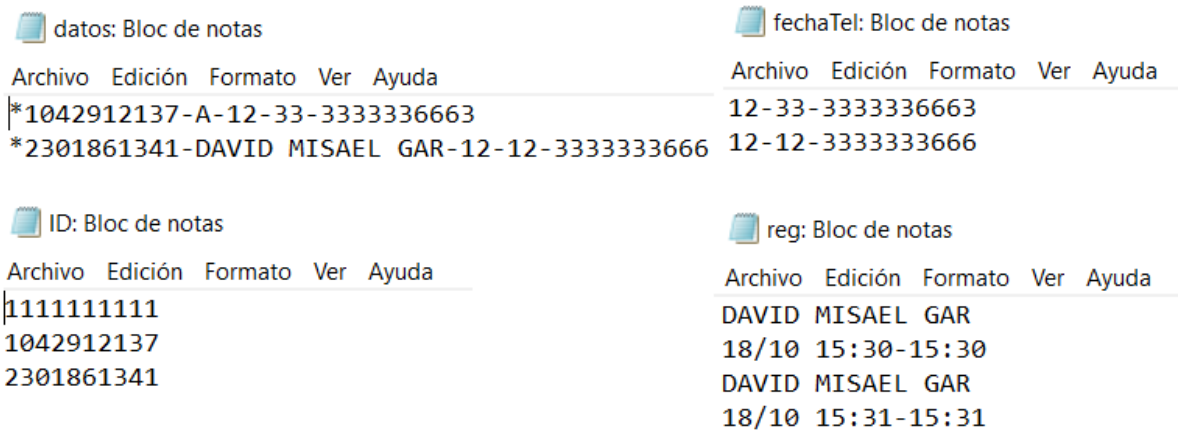

Ilustración 47. Ficheros .txt

23. Diagramas

En la Ilustración 48, es mostrada la conexión que se utilizó en este prototipo en donde, como se estableció con anterioridad, únicamente utiliza seis conexiones, una de estas es la línea de voltaje se conectó a 5V de Arduino mega, y a otra a la línea de GND. De igual manera, tiene que ser conectado el módulo GSM con los dos pines configurables. Sin olvidarse de la pantalla TFT que va ensamblada en la parte de arriba del Arduino.

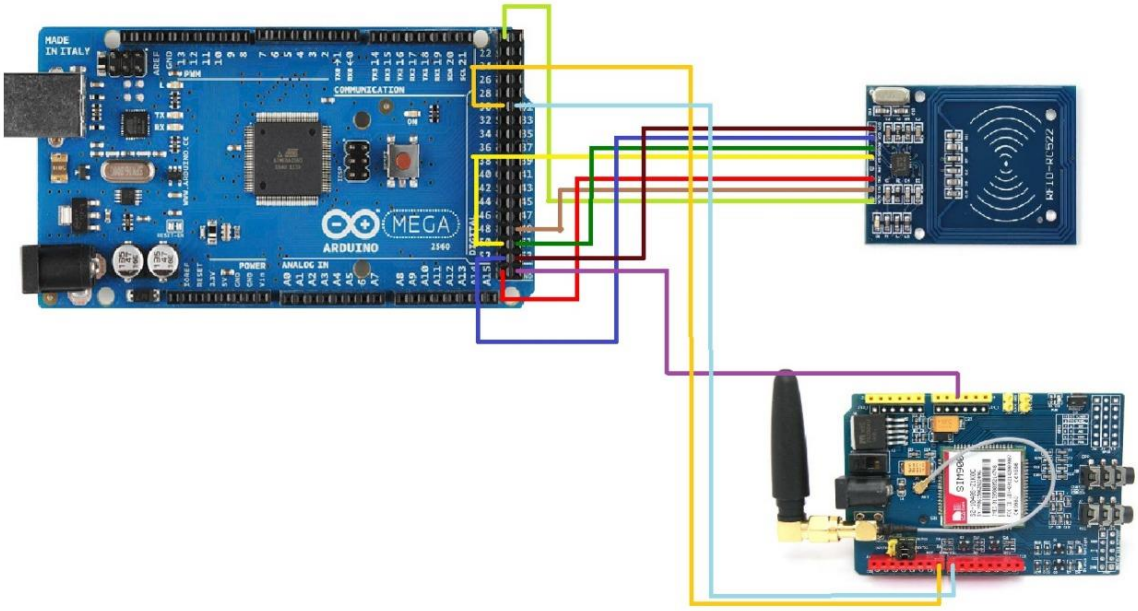

Ilustración 48.Diagrama de conexión Arduino-RFID-GSM.

Gracias a la plataforma PCBwizard, pudo realizarse el diagrama de conexiones para poder usar una placa de cobre y así, eliminar todo cableado que pudiera existir y ocasionar fallas. Únicamente, es necesario hacer las conexiones del módulo RFID y GSM, en la Ilustración 49 se puede mostrar un ejemplo de cómo hacer todas las líneas.

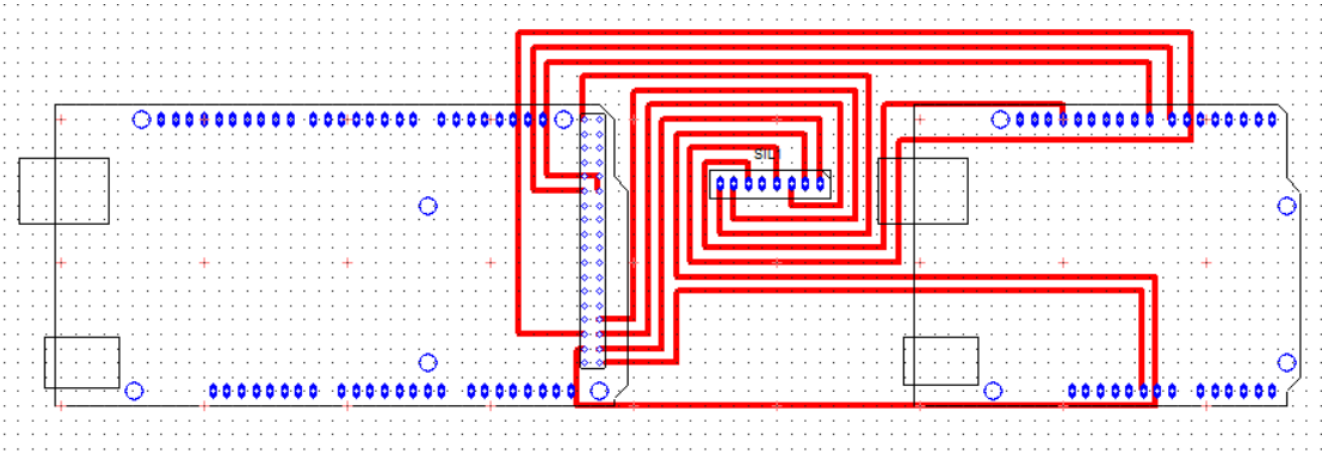

Ilustración 49. Líneas de conexión con PCBwizard.

Para poder hacer el ensamblado en conjunto de todos los módulos y placas ya construidas, se necesita hacer la conexión de las tabletas con ayuda de pines que son soldados en la parte inferior de la tableta de cobre, y que a su vez va conectada con Arduino y GSM.

En la Ilustración 50, se muestra una opción de cómo queda el resultado final de todo el prototipo en donde, las figuras marcadas con color amarillo representan la pantalla TFT y el módulo RFID, de color blanco la placa de cobre, y de color azul Arduino y GSM.

En la Ilustración 51 se muestra una opción para guardar nuestro sistema y poder proteger un poco, en esta ocasión fue elaborado de madera delgada para su fácil manejo.

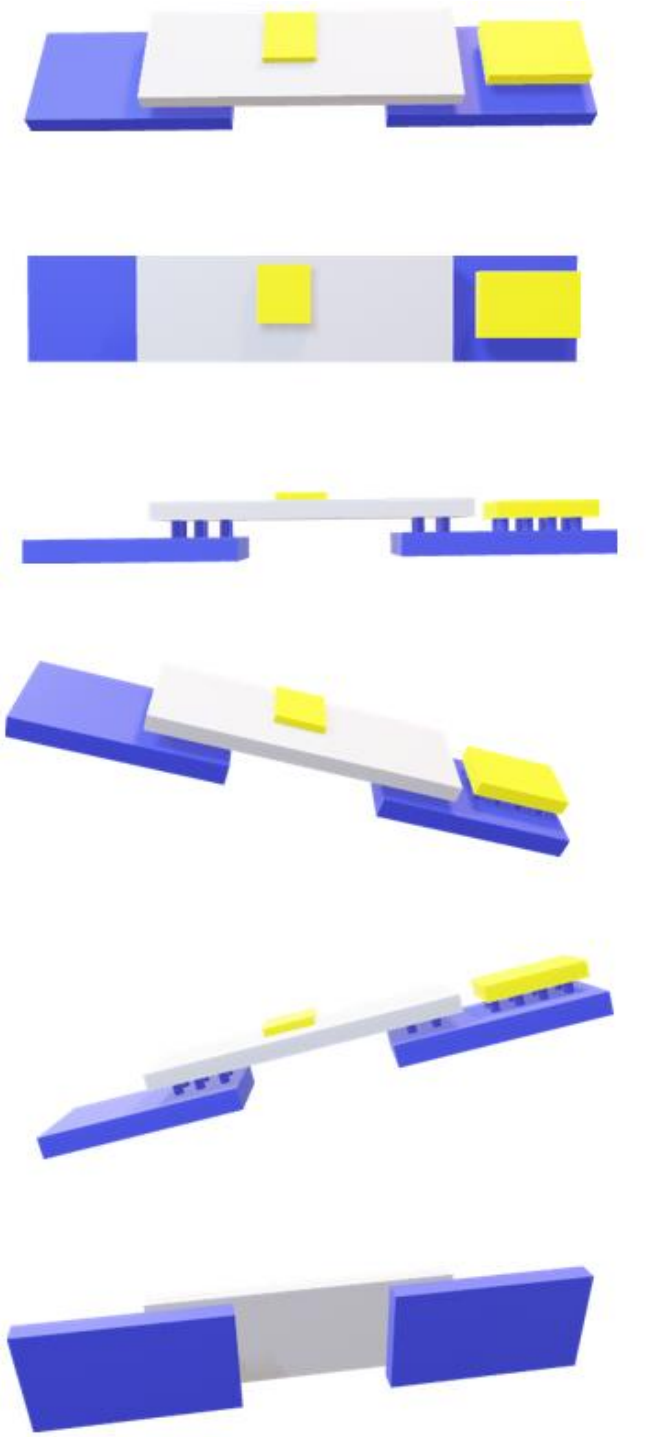

Ilustración 50. Ensamblado de los módulos GSM, Arduino, TFT y placa de cobre.

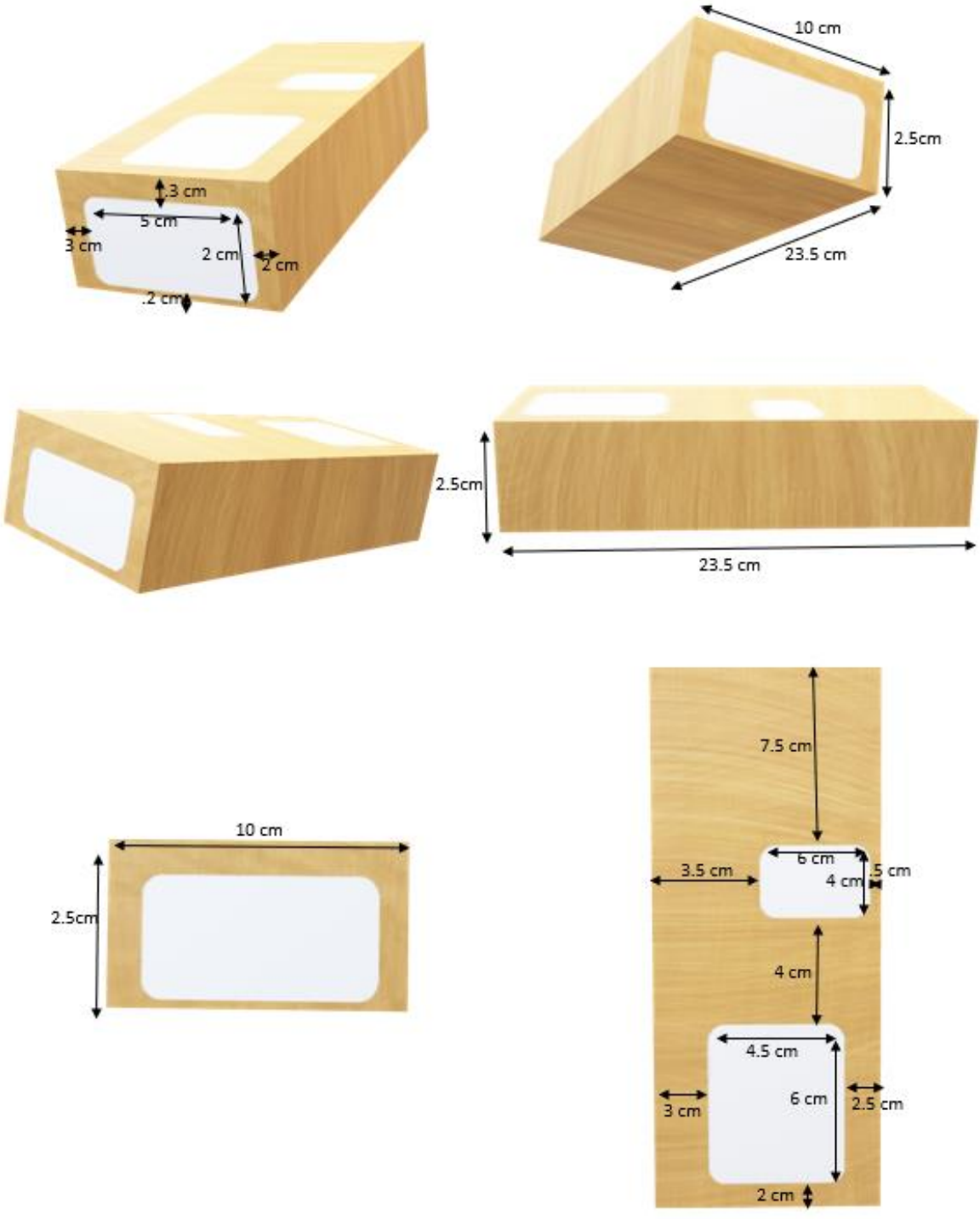

Ilustración 51.Caja del prototipo.

24. Mejoras

Todo modulo utilizado, o programación manejada en este prototipo puede tener algunas mejoras y escalas que podrían ser aplicadas para tener un sistema completo basado únicamente en Arduino, que incluya diversos aspectos como un LCD amplio, si se habla de hardware, o un programa desarrollado en otro lenguaje para interactuar con el responsable del establecimiento y poder ver todos los movimientos realizados.

Existen varios tipos de LCD que pueden ser utilizados para poder tener una mejor visualización de todos los usuarios, en particular en un gimnasio es posible hacer uso de pantallas que muestren las listas de las personas en uso y así saber la información necesaria de cada uno.

De igual manera, en la entrada pueden ser aplicados “torniquetes”, donde solo serán activadas con las tarjetas aceptadas. Así mismo, pueden ser utilizadas en la seguridad de los casilleros, donde almacenan objetos personales, y gracias a las tarjetas y sus identificadores podrían ejecutarse en conjunto.

Capitulo V. Resultados

El fichero con el nombre “reg” muestra los resultados obtenidos de este prototipo almacenando todos los registros de los usuarios que presentaron su etiqueta y fueron aceptados por este, tal y como se muestra en la pantalla TFT, mostrando el nombre de la persona, en el siguiente renglón, la fecha, hora de entrada/salida. Como ya se mencionó con anterioridad, la memoria del sistema es limitada y los espacios para mostrar lo nombres también lo es, y esto obliga a tener un número limitado de nombre mostrados en pantalla.

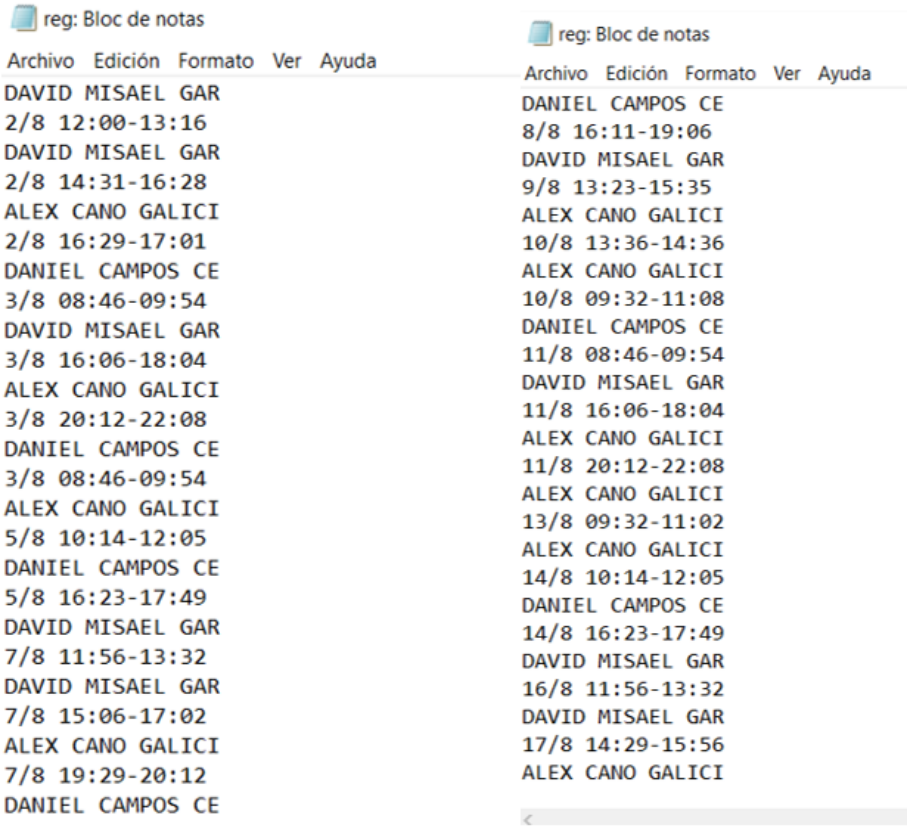

Ilustración 52. Fichero con resultados obtenidos.

Se realizaron pruebas con el prototipo funcionando durante un mes, en donde las primeras dos semanas únicamente fue probado con un total de diez personas registradas en la base de datos. La fecha límite del pago fue cambiado con un día de diferencia para poder comprobar el funcionamiento del envío de mensajes de texto comenzando el día 2 de agosto.

Las últimas dos semanas fueron agregadas cinco personas más, dando un total de 15 personas usando el prototipo, y con esto, probar distintos eventos que podrían ocasionar fallas o deficiencias durante la ejecución. Estos usuarios empezaron el día 15 de agosto y de igual manera, asignando las fechas por un día de diferencia. En la siguiente tabla se muestra con un rango del 1 al 5 el funcionamiento del prototipo, basando los valores del apartado “Métricas de calidad”.

Como conclusión general de la Tabla 7, hablando de exactitud, cumple la función general del prototipo, creando todos los registros necesarios. En el caso de rendimiento, todos los módulos trabajaron con normalidad y sin anomalías, siempre y cuando este no pierda energía. En la parte de usabilidad todo fue correcto hasta la entrada de los nuevos usuarios a mitad de mes, en donde la memoria tiene que ser transferida al administrador para poder dar de alta.

Hablando de limitaciones, es el espacio de almacenamiento en pantalla, en donde, hay un total de 63 registros que pueden ser mostrados al mismo tiempo, pero que en ningún momento sobrepaso dicha cantidad de registros. Y finalmente en eficiencia, gracias a la rápida lectura del lector RFID, todo proceso fue realizado

con rapidez, y no existió problema de almacenamiento, puesto que el único límite es el tamaño de almacenamiento de la memoria SD.

Tabla 7. Resultados de un mes.

	Semana 1	Semana 2	Semana 3	Semana 4
Exactitud	5	4	5	5
Rendimiento	4	4	4	5
Usabilidad	5	5	2 Nota: Cambio de memoria al crea nuevo registro.	5
Configuración	4	5	5	4
Eficiencia	5	5	4	4

En la Ilustración 53, se da a conocer el prototipo en su totalidad finalizado, con las características que fueron establecidas en este trabajo, una pequeña caja donde almacena todos los componentes, y placas ya mostradas. En la Ilustración 54, se observa la parte posterior del prototipo, donde es posible notar el espacio para poder introducir la tarjeta SIM. En la Ilustración 55, el prototipo está en funcionamiento, se puede observar, la pantalla TFT y el módulo RFID trabajando.

Ilustración 53.Prototipo final (Frontal)

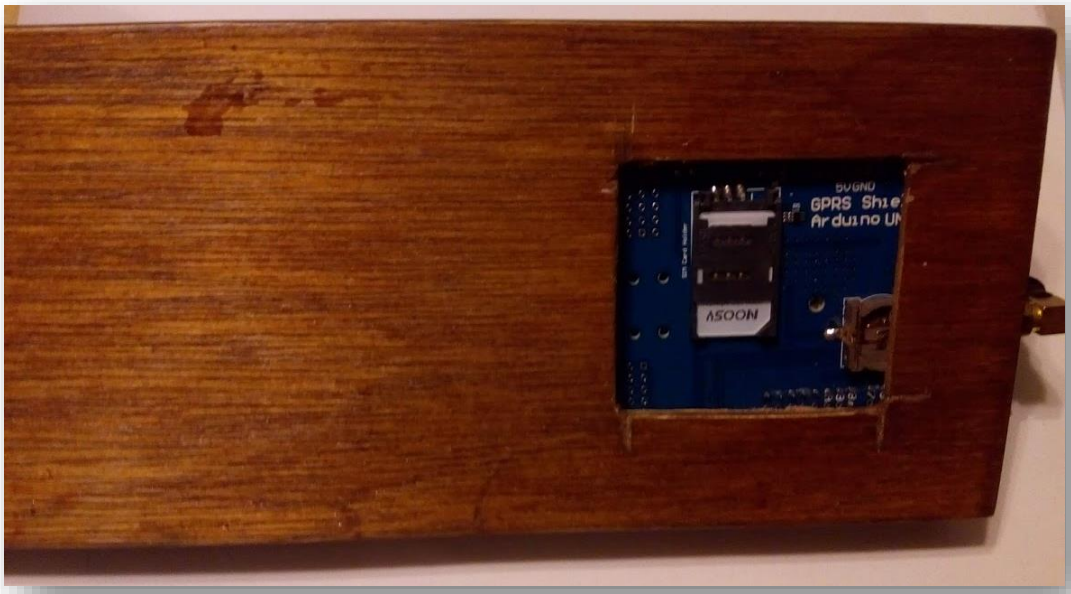

Ilustración 54. Prototipo final (Posterior)

Ilustración 55. Prototipo final (Funcionando)

25. Conclusiones

En el presente trabajo de tesis, se investigaron diversas formas de mantener un control en el registro de usuarios, en donde, en este caso, fue aplicado a un gimnasio que necesitaba un cambio en esta área. Así como poder avisar o recordar a estos usuarios su pago mensual. De igual manera fueron investigados los módulos empleados en este prototipo, su funcionamiento, características y el porqué de dichos módulos.

Mas allá de los conceptos principales que engloban este trabajo, se logró entender de manera más profunda como es el funcionamiento y aplicación de un sistema, de almacenamiento y de registros. De igual manera, se obtuvo conocimiento pleno sobre los módulos o instrumentos que componen este prototipo, tales como el módulo del sistema global para más comunicaciones móviles (GSM), sabiendo programar y configurar para poder enviar mensajes de texto.

Un sensor de radio frecuencia (RFID), donde nos permite leer tarjetas compatibles con este módulo, de igual manera, saber el funcionamiento de este, y poder dar una idea como una tarjeta trabaja en conjuntos con diversas áreas de trabajo o empresas, en donde usan este tipo de tecnología para tener un control de mercancía, o personal. Una pantalla TFT-LCD fue mostrada en este trabajo, enseñando su funcionamiento y programación mediante el entorno Arduino, y que a comparación de lo que muchas personas piensan, este tipo de código se facilita, gracias a la librería que esta pantalla contiene.

Por lo tanto, en el desarrollo de este trabajo de tesis, se logró demostrar que se puede automatizar un sistema de registros remplazando uno obsoleto y poco fiable, gracias al uso de hardware y software libre, cumpliendo con los requerimientos necesarios a un costo accesible. El prototipo cumple con todos los aspectos requeridos por los administradores del lugar, pero queda abierto a obtener mejoras.

En lo personal, fue grato trabajar con este tema, y poder demostrar que las nuevas tecnologías de código o hardware libre pueden sustituir un sistema como este, pero que generaba registros redundantes y poco fiables, que al paso del tiempo podría generar problemas de mayor gravedad.

El resultado obtenido, fue el esperado desde el inicio de este proyecto, en donde, después de muchas fases de experimentación de programación y conexión de componentes, por fin se obtuvo un producto que puede ser usado en diversas empresas, y que cumple con los requerimientos establecidos, logrando con esto, reducir errores que podrían ser provocados por personas.

26. Bibliografía

Alberto, C. N. (24 de Septiembre de 2002). *Reglamento interno del instituto mexiquense de cultura fisica y deporte. Ordenjuridico.* Obtenido de http://www.ordenjuridico.gob.mx/Documentos/Eliminados/wo30474.pdf

Banzi, M. (2007). *Arduino: Manual de Programación.* San Francisco: Attribution-Noncommercial-Share Alike 3.0 License.

Buioli, I. (2016). *Processing, un lenguaje al alcance de todos.*

Castro, A. (s.f.). Comunicaciones Móviles. *mplementación de mobile IP entre redes móviles y WLAN.* E.U.I.T.TELECOMUNICACION., Chile.

INC, I. (21 de 03 de 2013). *Inteltronic*. Obtenido de http://www.inteltronicinc.com/files/DRIVER/201411040007381.pdf

Leon, M. Z. (13 de Mayo de 2013). *Modelos del proceso del software*. Obtenido de http://modelosprocesosdesoftware.blogspot.mx/p/modelo-en-cascada-o-lineal-secuencial.html

María, M. A. (17 de Octubre de 2012). *Tarjeta Recargable*. Obtenido de http://www.metro.cdmx.gob.mx/tramites-y-servicios/servicios/tarjeta-recargable

México, G. d. (20 de Agosto de 2015). *Dirección general de Cultura Física y Reporte*. Obtenido de http://culturaydeporte.edomex.gob.mx/antecendentes

Ojeda, L. T. (02 de 02 de 2018). *Arduino*. Obtenido de http://arduino.cl/que-es-arduino/

PARADA, Y. G. (1999). *RECOLECCIÓN DE LA INFORMACIÓN.* Bogotá: ARFO EDITORES LTDA.

Portillo, J. I. (2007). *Tecnología de identificación por radiofrecuencia (RFID): Aplicaciones en el ámbito de la salud.* España: 76. E-28001 Madrid.

S.A, E. (09 de 01 de 2017). *http://www.agelectronica.com*. Obtenido de http://www.agspecinfo.com/pdfs/A/A6.PDF

Valdivia, A. Á. (2004). NTP 678: Pantallas de visualización: tecnologías (I). *Pantallas de visualización: tecnologías (I).* Ministro de trabajo y asuntos sociales españa, España. Obtenido de http://www.insht.es/InshtWeb/Contenidos/Documentacion/FichasTecnicas/NTP/Ficheros/601a700/ntp_678.pdf