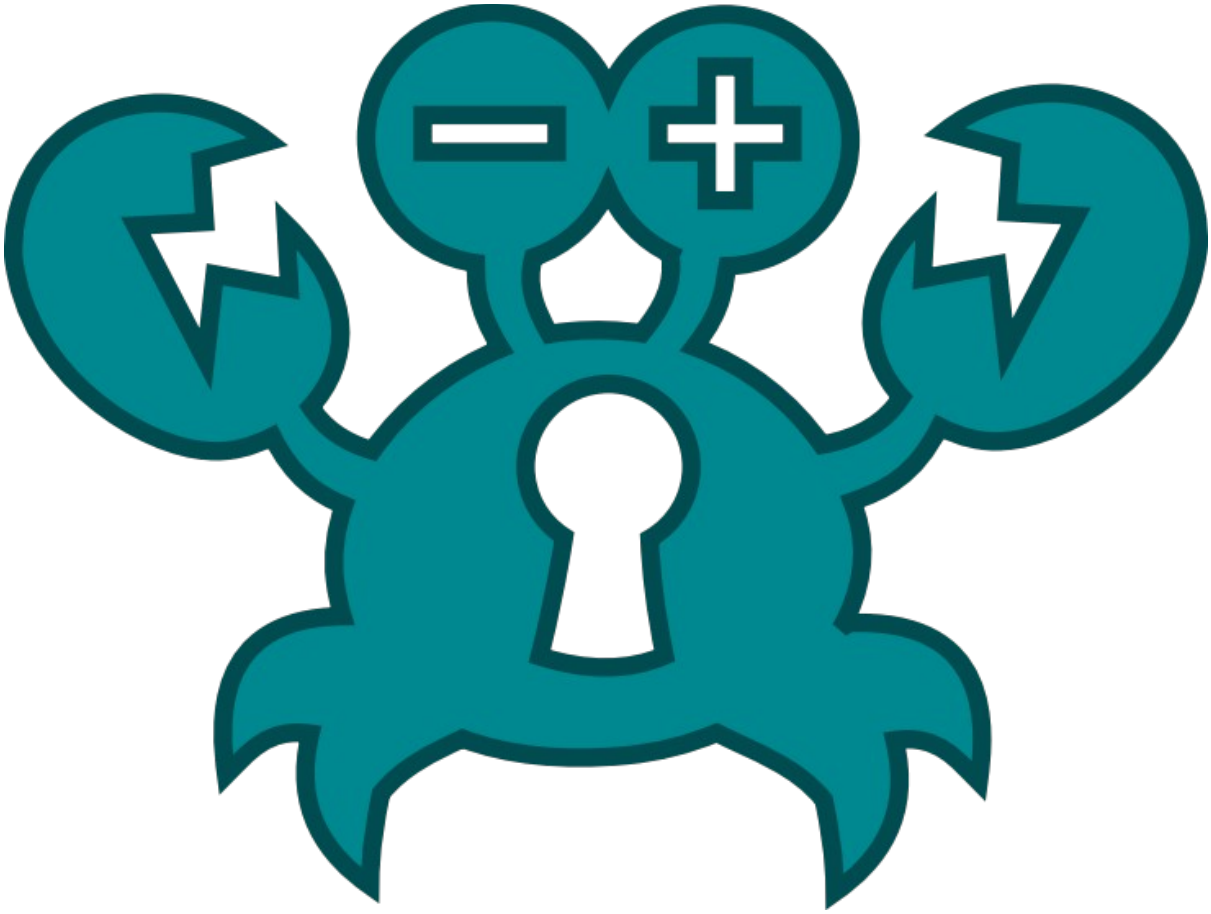
Institución: ESI buceo **Grupo**: 3ºBN

CRAB Lock



Fecha: 17/01/2021

Integrantes: Matías Cardozo, Joaquin Caballero, Ignacio Gomez, Maicol Arezo, Pablo Camargo

Introducción:

Con la idea de hacer un sistema más higiénico para las oficinas o incluso hogares desarrollamos un sistema de control y registro de acceso a través de un módulo lector de tarjetas y tags RFID. la emergencia sanitaria que estamos viviendo nos lleva a una necesidad por tener a mayor higiene posible en todo momento, según estudios realizados se llegó a la conclusión que uno de los objetos con más contacto humano son las puertas (más específicamente los pestillos), con esta idea en mente desarrollamos un sistema que nos permite abrir las puertas minimizando el contacto con la misma.

Resumen

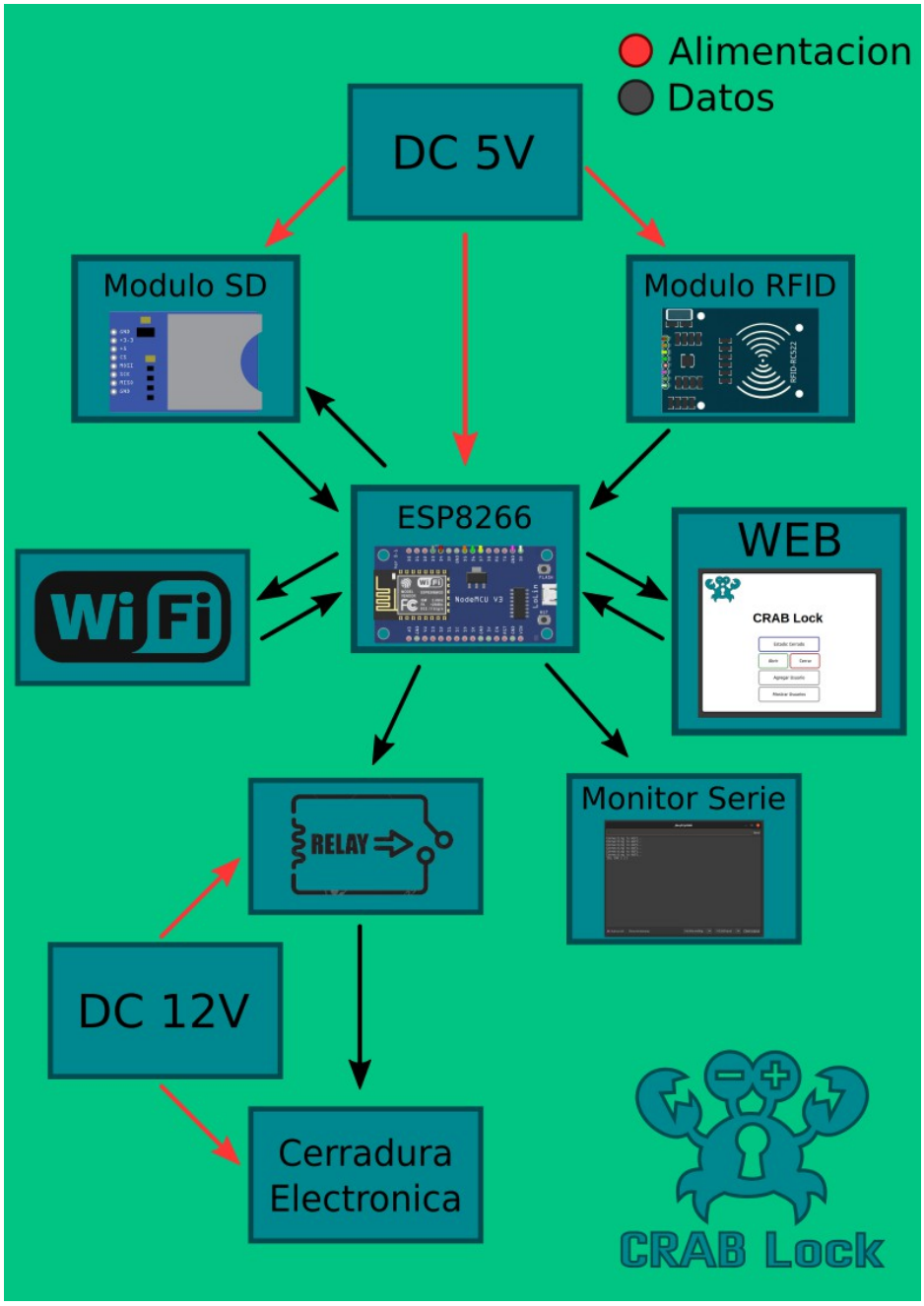
Este proyecto tiene como propósito la construcción en el marco de la cultura OPEN HARDWARE(adaptable) de una “cerradura inteligente” la cual permite un registro de cada usuario y hora de acceso del mismo. Integrando diversas técnicas, dispositivos y herramientas tanto de Hardware, Software, Programación y Robótica.

El **CRAB Lock** (Control and Registry Arduino Based Lock):consiste de un lector de tarjetas RFID que a través de arduino permite acceso a determinados usuarios con un horario establecido; la cerradura para su funcionamiento dispone de dos tipos de alimentación: una batería de 12v en caso de apagones, y una fuente la cual suministra 13v para recargar la batería y 5v como alimentación al arduino. Cabe destacar que los usuarios y los horarios de entrada y salida quedarán registrados en un archivo.

The purpose of this project is the construction within the framework of the OPEN HARDWARE (adaptable) culture of an "intelligent lock" which allows a record of each user and time of access. Integrating various techniques, devices and tools from both Hardware, Software, Programming and Robotics.

The **CRAB Lock** (Control and Registry Arduino Based Lock): consists of an RFID card reader that through arduino allows access to certain users with a set schedule; The lock for its operation has two types of power: a 12v battery in case of blackouts, and a source which supplies 13v to recharge the battery and 5v as power to the arduino. It should be noted that users and entry and exit times will be recorded in a file.

Diagrama de flujo:



Guia Tecnica:

RFID: Mrfc522 RFID o identificación por radiofrecuencia es un sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados etiquetas, tarjetas o transpondedores RFID. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto mediante ondas de radio.

Relay: El relé (en francés, relais 'relevo') es un dispositivo electromagnético. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.

Controlador: Esp8266 Se trata de un SoC o Sistema en Chip. Básicamente consiste en un chip que tiene todo integrado (o casi todo) para que pueda funcionar de forma autónoma como si fuera un ordenador. En el caso del ESP8266 lo único que no tiene es una memoria para almacenar los programas.

SD: Una tarjeta digital segura (SD) es una tarjeta de memoria flash extraíble utilizada para almacenar información digital, como programas y archivos. Las tarjetas SD se utilizan comúnmente en teléfonos celulares y en otros dispositivos portátiles para ampliar el almacenamiento de la memoria ROM.

Cerradura: Funciona de manera invertida, es decir, la posición de reposo del abrepuertas es abierta y cuando recibe electricidad se bloquea cerrando la puerta. En caso de un corte eléctrico la puerta queda abierta permitiendo el franqueo de la misma.

Alimentación: El dispositivo es suministrado con energía eléctrica a través de una fuente modificada la cual está conectada a la red.

Autonomía: En caso de apagarse el suministro eléctrico de la red (alimentación) tenemos una batería que sustenta el dispositivo manteniéndolo en funcionamiento hasta la reconección.

Circuito de corte: Este circuito impide que la batería se sobrecargue,cortando la energía y evitando que se desplace a la batería al detectar cuando esta se encuentra cargada.

regulador de voltaje: Un regulador de tensión o regulador de voltaje es un dispositivo electrónico diseñado para mantener un nivel de tensión constante o regulable, el dispositivo utiliza un regulador para alimentar el arduino con su voltaje correspondiente.

Objetivo:

El dispositivo es capaz de bloquear y desbloquear una cerradura con el menor contacto posible sobre la misma para conseguir una mayor higiene, permitiendo el ingreso de usuarios autorizados manteniendo un registro de los mismos.

METODOLOGÍA:

ensayo y prueba.
Investigamos qué diseño de cerradura segura, modificable y más rentable del mercado teniendo en cuenta su durabilidad y gastos energéticos,
para su creación decidimos utilizar sensores y placas de arduino o esp:
sensor RFID(lector de tarjetas), sensor de ultrasonido, una cerradura eléctrica, teclado keypad, fuente ATX, módulo de arduino ethernet, arduino mega, esp8266, reed switch, bateria de gel.

En cada caso indagamos otros componentes de mejor calibre,
Para el lector de tarjetas descubrimos que la placa PN532 tenía mayor delay que MRC522 a la hora de leer.

Para las baterías descubrimos que las baterías de litio tienen mayor costo y menor capacidad frente a las baterías de gel.

Para la cerradura eléctrica investigamos que el mejor método sería usar una cerradura electrónica de perno, sin embargo por motivos de disponibilidad utilizamos un Cerrojo destrabador eléctrico.

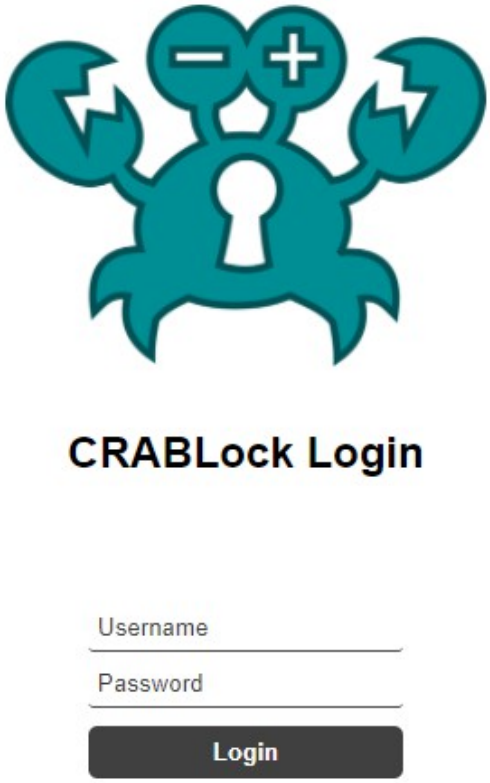
DISCUSIÓN

Se logró realizar la cerradura la cual al pasar una tarjeta con un usuario registrado por un lector se desbloquea y se vuelve a cerrar una vez la puerta se cierra, también registrando la hora y el nombre del usuario.

En cada proceso se hizo una prueba y error, se probó con hacer el sistema fuera de una puerta real, pero por practicidad utilizamos una maqueta a ¼ de escala basada en una puerta de un datacenter.

PÁGINA WEB

Página inicial:



Página principal:

CRAB Lock

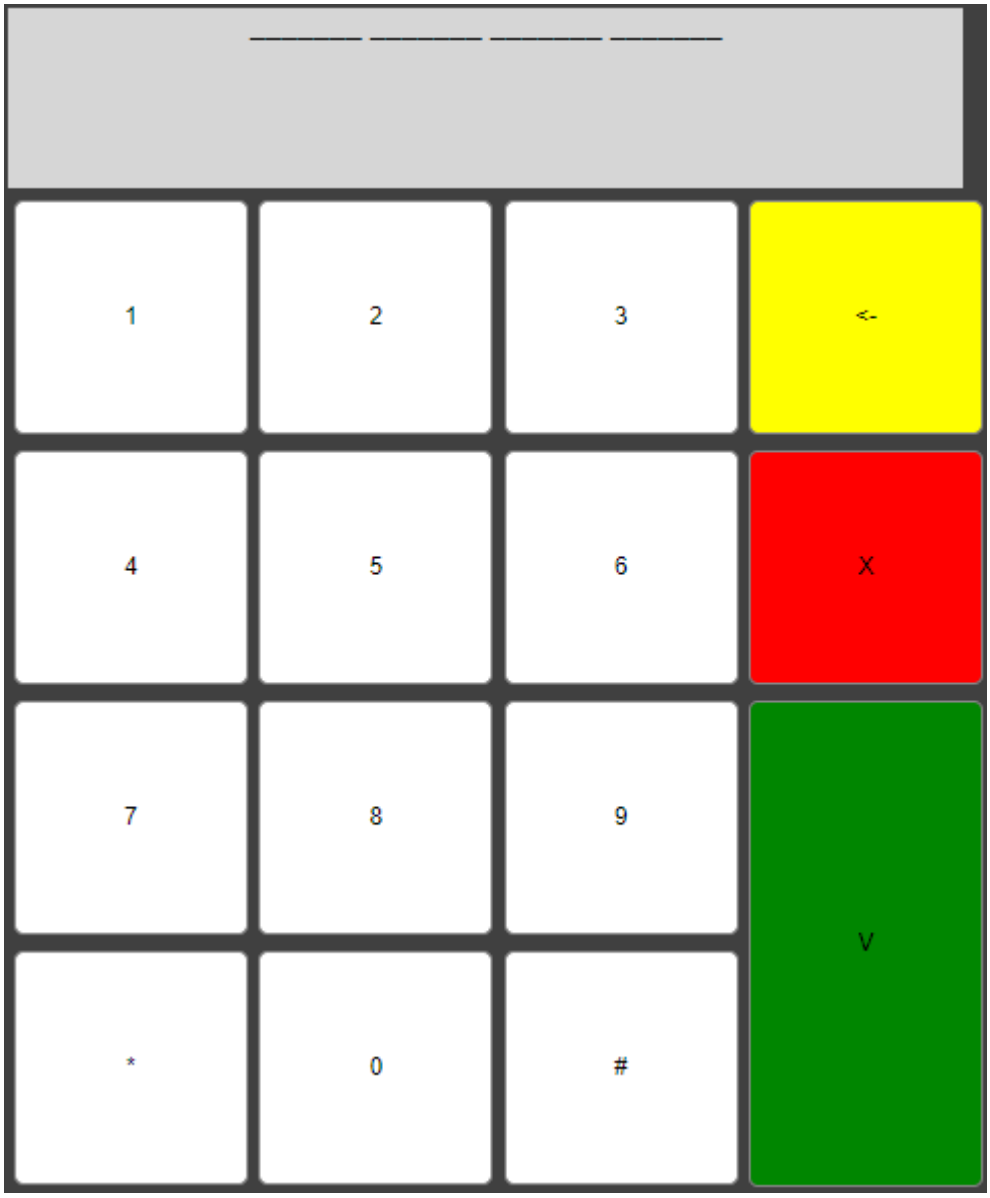
%BUTTONPLACEHOLDER%

Abrir | Cerrar

Agregar Usuario

Administrar Usuarios

Teclado:



Anexo:

DOCUMENTACIÓN

MARCO NORMATIVO

MTB: Ámbito de Aplicación. Las prescripciones contenidas en este Reglamento se refieren a instalaciones definidas por tensiones iguales o inferiores a los 1000 V y particularmente a los servicios suministrados por UTE, que son: En sistemas sin neutro: 50 Hz; 220 V de tensión compuesta. En sistemas con neutro: 50 Hz; 380 V de tensión compuesta y 220 V entre fase y neutro.

Nota: Se aconseja que se utilicen para estos trabajos interruptores termo magnéticos. Si se instalan a la intemperie lo mismo que los fusibles, deben estar colocados en cajas estancas a la humedad.

Medidas de Protección contra Contactos Directos e Indirectos. Las instalaciones eléctricas se establecerán de forma que no supongan riesgo para las personas (y eventualmente para los animales domésticos) tanto en servicio normal como cuando puedan presentarse averías previsibles. En relación con estos riesgos, las instalaciones deberán proyectarse y ejecutarse aplicando las medidas de protección necesarias contra los contactos directos e indirectos. Estas medidas de protección son las señaladas en el Capítulo VI.

Las instalaciones deberán presentar una resistencia de aislamiento mayor a 1.000 × U ohmios, siendo U la máxima tensión de servicio expresada en voltios, con un mínimo de 250.000 ohmios. Este aislamiento se entiende para una instalación en la cual la longitud del conjunto de canalización eléctrica y cualquiera que sea el número de conductores que las componen no exceda de 100 metros de longitud, bien por seccionamiento, desconexión, retirada de fusibles o aperturas de interruptores, cada una de las partes en que la instalación ha sido fraccionada debe presentar el aislamiento que corresponda. Cuando no sea posible efectuar el fraccionamiento citado, se admite que el valor de la resistencia de aislamiento de toda la instalación sea, con relación al mínimo que le corresponda, inversamente proporcional a la longitud de las canalizaciones eléctricas.

Las corrientes máximas admisibles en servicio permanente, para conductores aislados en canalizaciones eléctricas fijas, y a una temperatura ambiente de 25°C, se indican en las distintas Tablas de este reglamento, según sea el tipo de aislamiento, sistema de instalación y medio ambiente.

Corrientes máximas Admisibles. En la Tabla I figuran las corrientes máximas admisibles en régimen permanente para este tipo de cables, en condiciones normales de instalación. Las condiciones normales de instalación se definen como un solo cable tripolar, o tetrapolar, instalado al aire libre en una disposición que permita una eficaz renovación de aire, y a una temperatura ambiente de 25 °C. Para otras condiciones diferentes, en el apartado 2.2 figuran los factores de corrección apropiados.

5.1.- Factores de Corrección. La corriente máxima admisible, deducida de las Tablas X, XI, XII y XIII deberá corregirse teniendo en cuenta las características de la instalación, de forma que el incremento de temperatura provocado por la corriente eléctrica, no dé lugar a una temperatura en el conductor superior a 70°C, en los cables con aislamiento de policloruro de vinilo o de goma y 90°C en los cables con aislamiento de goma butílica, etileno - propileno o polietileno reticulado. Para valores de temperatura ambiente diferente de 25 °C, se aplicarán los factores de corrección de la Tabla XIV, según el tipo de aislamiento. Cuando por un tubo o conducto tengan que pasar más de 3 conductores, normalmente recorridos por la corriente, los valores de la corriente máxima admisible de la última columna, se reducirá aplicando los factores de reducción siguientes: de 4 a 7 conductores = 0.90 más de 7 conductores = 0.70 Para el cómputo de estos conductores no se tendrá en cuenta, en ningún caso, el conductor de protección, ni el neutro, en un suministro trifásico con neutro

Puestas a la Tierra. Definición. La denominación "puesta a tierra" comprende toda la ligazón metálica directa, sin fusible ni protección alguna, de sección suficiente, entre determinados elementos o partes de una instalación y un electrodo, o grupo de electrodos, enterrados en el suelo, con objeto de conseguir que en el conjunto de instalaciones, edificios y superficie próxima del terreno no existan diferencias de potencial peligrosas y que, al mismo tiempo, permita el paso a tierra de las corrientes de falta o la de descarga de origen atmosférico.

Puestas a tierra en pequeños o medianos suministros eléctricos individuales. En el caso de viviendas, pequeños comercios o industrias individuales, ubicados en predios independientes, y cuyas cargas solicitadas no superen los 15 kW en 220 V, o los 20 kW en 380 V, las secciones de los conductores de protección serán los indicados en el numeral 10.- . En estos casos, si el suelo es de conductividad adecuada y las condiciones obtenidas con un electrodo simple son suficientes para alcanzar los objetivos expresados en el numeral 1.- en lugar de utilizar el anillo de enlace con tierra será suficiente con conectar la línea principal de tierra con el electrodo en la respectiva cámara. En el caso de que esta solución no fuese suficiente se deberá adoptar el criterio reglamentario más adecuado. Si fuera necesaria la instalación de pararrayos para proteger un área de pequeñas dimensiones, se deberá respetar, en todo lo expresado en la figura referida al "esquema de un sistema de puesta a tierra".

NORMAS(ABSU-TÍA/BICST)

PROTOCOLOS

SPI, WiFi, HTTP.

RANGO DE OPERACIÓN DE LOS COMPONENTES

VERSIONES DE LOS S.O

la versión del sistema operativo usada para la programación del proyecto fue Windows 10 ya que es un sistema operativo muy estable y con un mejor acceso a las herramientas de programación como el arduino IDE

CONTRATO DE SERVICIO

UNIT

El conductor es de cobre con un diámetro de 2mm^2, y su resistividad es de 0.0176Ω por mm^2/m, con una aislación de PVC O assl

EMI

El dispositivo puede ser afectado por interferencias de electromagneticas de los conductores que tenga las paredes del comercio y vivienda, estas pueden interrumpir o afectar la comunicación wifi.

RFI

El dispositivo genera cierta interferencia por su campo electromagnético que puede afectar los dispositivos más cercanos

I.P

Este producto está certificado con una protección de ingreso IP20, lo que significa que está protegido contra cuerpos sólidos de hasta 12mm de diámetro (como lo puede ser un dedo), y no está protegido contra entrada de agua

AUTONOMÍA

Su autonomía se basa en una batería de 12v 4ah la cual se mantiene cargada y lista para una emergencia. Esta mantendrá al dispositivo almenos unos

CONDICIONES AMBIENTALES Y OPERATIVAS

Uso exclusivamente interior. No resiste golpes ni sobrecargas o cortocircuito.

CONSUMO

< 5W

-ALIMENTACIÓN

alimentaremos el mecanismo y los dispositivos mediante un transformador conectado a la corriente eléctrica (ampliar un poco mas)

-NIVELES DE SEGURIDAD

-FÍSICO:

Como seguridad física contamos con una caja hecha a medida que protege el dispositivo y a sus componentes, además que la placa que utilizamos cuenta con un sistema que requiere acceso a la placa para cargar el código dentro de ella, el acceso a la placa programable esp32 será restringido para que solo el personal autorizado pueda acceder

-NIVELES DE AJUSTES y/o CALIBRE DE LOS SENSORES(SENSIBILIDAD)

-USABILIDAD

Nuestro dispositivo es muy sencillo y amigable hacia el usuario. Consta de 2 partes fundamentales un lector de tarjetas que simplemente acercando una tarjeta con una ID autorizada accionarias el el sistema o desde una interfaz web dando una serie de opciones de gestión que permiten un fácil control del dispositivo siempre y cuando estés autorizado.

-INSTALACIÓN

La instalación del producto es muy sencilla, simplemente es colocar un cerrojo eléctrico donde iría el cerrojo tradicional y adherir el dispositivo a un lugar cercano y con acceso a corriente y a internet si se quiere usar la interfaz web,

-MANTENIMIENTO

-APLICACIÓN(MANUAL)

-COMPATIBILIDAD

El dispositivo es compatible con casi cualquier puerta siempre y cuando esté en condiciones óptimas para el correcto funcionamiento del dispositivo. además que puede adaptarse fácilmente si es necesario

-GARANTÍA

HTML:

Inicio:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>CRABLock Login</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>
  <main id="main-holder">
    <br>

    <div><img src="https://i.ibb.co/kGJk3RK/CRABLock-logo.png"
height=200> </div>
    <h2 id="login-header">CRABLock Login</h2>

    <div id="login-error-msg-holder">
      <p id="login-error-msg">Usuario o pass<span id="error-msg-second-
line">invalidos</span></p>
    </div>

    <form id="login-form">
      <input type="text" name="username" id="username-field"
class="login-form-field" placeholder="Username">
      <input type="password" name="password" id="password-field"
class="login-form-field" placeholder="Password">
      <input type="submit" value="Login" id="login-form-submit">
    </form>

    <br>
```

```
  </main>
</body>
<script>
  const loginForm = document.getElementById("login-form");
  const loginButton = document.getElementById("login-form-submit");
  const loginErrorMsg = document.getElementById("login-error-msg");

  loginButton.addEventListener("click", (e) => {
      e.preventDefault();
      const username = loginForm.username.value;
      const password = loginForm.password.value;

      if (username === "user" && password === "pass") {
          window.location.href = "main-page.html";
      } else {
          loginErrorMsg.style.opacity = 1;
      }
  })
  </script>

</html>
```

Pagina principal:

```
<!DOCTYPE HTML>
<html>

<head>
  <meta content="text/html;charset=utf-8" http-equiv="Content-Type">
  <meta content="utf-8" http-equiv="encoding">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="stylesheet" href="main-page.css">

  <title>CRAB Lock</title>
</head>
<body>
  <div id="adminUsers">
    <span class="helper"></span>
    <div>
        <div id="popupCloseButton"
onclick="mostrarUsers()">&times;</div>

        %MOSTRARUSUARIOS%




    </div>
    </div>


  <main id="main-holder">


    <div id="logo"><a href="/"><img
src="https://i.ibb.co/kGJk3RK/CRABLock-logo.png"
height=100></a></div>
```

```
    <h2><img src="https://i.ibb.co/jGgchqP/CRABlock.png"
height=40></h2>

    %BUTTONPLACEHOLDER%
    <br>
    <button class="button button1" onclick="abrir()">Abrir</button>
    <button class="button button2" onclick="cerrar()">Cerrar</button>
    <br>
    <button class="button button4" onclick="agregarUsuario()">Agregar
Usuario</button>
    <br>
    <button class="button button5" onclick="mostrarUsers()">Administrar
Usuarios</button>
    <br>




  </main>

</body>

<script type="text/javascript">
  function toggle() {
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/update?opcion=2", true);
  xhr.send();
  location.reload();
}

function abrir() {
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/update?opcion=1", true);
  xhr.send();
  location.reload();
}
```

```
function cerrar() {
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/update?opcion=0", true);
  xhr.send();
  location.reload();
}

function agregarUsuario() {
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/update?opcion=3", true);
  xhr.send();
}

function mostrarUsers() {
  var elem = document.getElementById('adminUsers');
   if(elem.style.display === "block"){
   elem.style.display = "none";
 }else{
  elem.style.display = "block";
 }

}

function elimUser(x){
  var xhr = new XMLHttpRequest();
  if (confirm("eliminar usuario " + x + "?")) {
    xhr.open("GET", `/update?eliminar=${x}`, true);
    xhr.send();
    setTimeout(() => {
      location.reload();
    }, 500);
    alert("usuario " + x + " eliminado");
}
}
```

```
</script>
</html>
```

Teclado:

```
<!DOCTYPE HTML>
<html>

<head>
  <meta content="text/html;charset=utf-8" http-equiv="Content-Type">
  <meta content="utf-8" http-equiv="encoding">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="stylesheet" href="tablet.css">

  <title>CRAB Lock</title>
</head>

<body>
  <main class="main-holder">

    <div class="pass">
      _______  _______  _______  _______
    </div>

    <div class="col">
      <button id="row"> 1</button>
      <button id="row"> 2</button>
      <button id="row"> 3</button>
      <button id="del"> <-</button>
    </div>
    <div class="col">
      <button id="row"> 4</button>
      <button id="row"> 5</button>
      <button id="row"> 6</button>
      <button id="X"> X</button>
    </div>
    <div class="col">
      <button id="row"> 7</button>
      <button id="row"> 8</button>
```

```
      <button id="row"> 9</button>
      <button id="ok"> V</button>
    </div>
    <div class="lastcol">
      <button id="row"> *</button>
      <button id="row"> 0</button>
      <button id="row"> #</button>
    </div>

  </main>
</body>

</html>
```

-PROGRAMACION (CODIGO FUENTE)

```
#include <FS.h>
#include <SD.h>
#include <SPI.h>
#include <MFRC522.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ArduinoJson.h>
#include <ESPAsyncWebServer.h>
#include "SPIFFS.h"

#define RST_PIN 22
#define SS_PIN 21

MFRC522 reader(SS_PIN, RST_PIN);
MFRC522::MIFARE_Key key;

File myFile;

String usersJson;
String Users[8];

bool estaAbierto = false;

bool funcionActiva = false;

int cantUsuarios = 6;

byte LecturaUID[4];
byte Usuarios[8][4];

//Servidor Web//////////////////////////////////////////////
/*const char* ssid = "AquiTampoco";
const char* password = "notengoidea";
*/
```

```
const char* ssid = "JoakoMi8";
const char* password = "qqqqqqqq";


const char* PARAM_INPUT_1 = "opcion";
const char* PARAM_INPUT_2 = "eliminar";

AsyncWebServer server(80);

const int cerradura = 15;

String processor(const String& var) {
  //Serial.println(var);
  if (var == "BUTTONPLACEHOLDER") {
    String buttons = "";
    String estadoPuerta = "Estado: ";
    if (estaAbierto == LOW) {
      estadoPuerta = "Estado: Cerrado";
    }
    else if (estaAbierto == HIGH) {
      estadoPuerta = "Estado: Abierto";
    }
    buttons += "<button class=\"button button3\" onclick=\"toggle()\" >" +
estadoPuerta + "</button>";


    return buttons;
  }

  if (var == "FAVICON") {
    String favicon = "<div id=\"topright\"> <img
src=\"https://i.ibb.co/kGJk3RK/CRABLock-logo.png\" height=100>
</div>";

    return favicon;
  }
```

```
  if (var == "MOSTRARUSUARIOS") {
    String alert = "<br>";
    String alertUsers = "";

    for (int i = 0; i < cantUsuarios; i++) {
      alertUsers = Users[i];
      if(alertUsers.length() > 5){
      alertUsers.toUpperCase();
      alertUsers.replace("\"", "");
      alertUsers.replace(",", ", ");
      alertUsers.replace(":", ": ");
      alertUsers.replace("{", "");
      alertUsers.replace("}", "");

      alert += "<a>" + alertUsers + "   </a>";
      alert += "<a style=\"color:#FF0000; font-size: 20px;\",
onclick=\"elimUser(";
      alert += i+1;
      alert += " )\"> x </a>";
      alert += "<br>";
      alert += "<br>";
    }
}

    return alert;
  }



  return String();
}

void setup() {
  /*Users[0] = "{\"id\":1,\"pass\":\"1234\",\"UID\":[211,240,171,002]}";
  Users[1] = "{\"id\":2,\"pass\":\"4321\",\"UID\":[249,116,037,179]}";
  Users[2] = "{\"id\":3,\"pass\":\"1590\",\"UID\":[014,200,190,141]}";
```

```
Users[3] = "{\"id\":4,\"pass\":\"1234\",\"UID\":[211,240,171,002]}";
Users[4] = "{\"id\":5,\"pass\":\"4321\",\"UID\":[249,116,037,179]}";
Users[5] = "{\"id\":6,\"pass\":\"1590\",\"UID\":[014,200,190,141]}";*/



Serial.begin(9600);
while (!Serial) continue;
if (!SPIFFS.begin()) {
  Serial.println("An Error has occurred while mounting SPIFFS");
  return;
}
//Cerradura inicia cerrada
pinMode(cerradura, OUTPUT);
digitalWrite(cerradura, LOW);

SPI.begin();
reader.PCD_Init();
delay(4);
for (byte i = 0; i < 6; i++) {
  key.keyByte[i] = 0xFF;
}

if (!SD.begin()) {
  Serial.println("SD card initialization failed!");
  //return;
}else{
  getSdInfo();
  Serial.println("Listo");
  }



WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi..");
```

```
  }
  //Mostrar IP
  Serial.println(WiFi.localIP());
  obtenerUsuarios(SD);
  iniciarServerWeb();
}

void loop() {
  if (!funcionActiva) {
    leerTarjeta();
  } else {
    agregarUsuario();
  }
}


void iniciarServerWeb() {
  server.on("/", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *
request) {
    request->send(SPIFFS, "/style.css", "text/css");
  });

  server.on("/main-page.html", HTTP_GET, [](AsyncWebServerRequest *
request) {
    request->send(SPIFFS, "/main-page.html", String(), false, processor);
  });

  server.on("/main-page.css", HTTP_GET, [](AsyncWebServerRequest *
request) {
    request->send(SPIFFS, "/main-page.css", "text/css");
  });

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

```
  //Interpretacion HTTP
  // Send a GET request to <ESP_IP>/update?state=<inputMessage>
  server.on("/update", HTTP_GET, [] (AsyncWebServerRequest *
request) {
    String inputMessage;
    String inputParam;
    // GET input1 value on <ESP_IP>/update?state=<inputMessage>
    if (request->hasParam(PARAM_INPUT_1)) {
      inputMessage = request->getParam(PARAM_INPUT_1)->value();
      inputParam = PARAM_INPUT_1;
      if (inputMessage.toInt() == 0) {
        Serial.println("Cerrar");
        estaAbierto = LOW;
        digitalWrite(cerradura, estaAbierto);
      } else if (inputMessage.toInt() == 1) {
        Serial.println("Abrir");
        estaAbierto = HIGH;
        digitalWrite(cerradura, estaAbierto);
      }
      else if (inputMessage.toInt() == 2) {
        Serial.print("Toggle - ");
        estaAbierto = !estaAbierto;
        digitalWrite(cerradura, estaAbierto);
        if (estaAbierto) {
          Serial.println("Abierto");
        } else {
          Serial.println("Cerrado");
        }

      }
      else if (inputMessage.toInt() == 3) {
        Serial.println("Agregar Usuario");
        agregarUsuario();
      }
      else if (inputMessage.toInt() == 4) {
        Serial.println("Eliminar usuario");
      }
```

```
    }
    else if (request->hasParam(PARAM_INPUT_2)) {
      inputMessage = request->getParam(PARAM_INPUT_2)->value();
      inputParam = PARAM_INPUT_2;

      Serial.println("Eliminar usuario " + inputMessage);

      }
    else {
      inputMessage = "No message sent";
      inputParam = "none";
    }
    //Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
  });
  // Start server
  server.begin();
}

void noAutorizado() {
  Serial.println("no estas autorizado");
}

void usuario () {
  Serial.println("Bienvenido Usuario");
  Serial.println("abierto");
  estaAbierto = true;
  digitalWrite(cerradura, estaAbierto);
  //delay(2000);
}

void admin() {
  Serial.println("Bienvenido administrador ");
  Serial.println("abierto");
  estaAbierto = true;
  digitalWrite(cerradura, estaAbierto);
}
```

```
void dentro () {
  Serial.println("saliendo ");
  estaAbierto = true;
  digitalWrite(cerradura, estaAbierto);
  //delay(2000);
}

void getSdInfo(){
  uint8_t cardType = SD.cardType();
  Serial.println();
  Serial.print("SD Card Type: ");
  if(cardType == CARD_MMC){
    Serial.println("MMC");
  } else if(cardType == CARD_SD){
    Serial.println("SDSC");
  } else if(cardType == CARD_SDHC){
    Serial.println("SDHC");
  } else {
    Serial.println("UNKNOWN");
  }

  uint64_t cardSize = SD.cardSize() / (1024 * 1024);
  Serial.printf("SD Card Size: %lluMB\n", cardSize);
  }


void leerTarjeta() {
  if (!reader.PICC_IsNewCardPresent())
    return;

  if (!reader.PICC_ReadCardSerial())
    return;

  Serial.print("UID:");
  for (byte i = 0; i < reader.uid.size; i++) {
    if (reader.uid.uidByte[i] < 0x10) {
```

```
      Serial.print(" 0");
    }
    else {
      Serial.print(" ");
    }
    Serial.print(reader.uid.uidByte[i], HEX);
    LecturaUID[i] = reader.uid.uidByte[i];
  }

  Serial.print("\t");

  bool userEncontrado = false;

  for (int i = 0; i < cantUsuarios; i++) {
    if (comparaUID(LecturaUID, Usuarios[i])) {
      userEncontrado = true;
    }
  }
  if (userEncontrado) {
    usuario();
  } else {
    noAutorizado();
  }

  // Halt PICC
  reader.PICC_HaltA();
  // Stop encryption on PCD
  reader.PCD_StopCrypto1();

}


boolean comparaUID(byte lectura[], byte usuario[])
{
  for (byte i = 0; i < reader.uid.size; i++) {
    if (lectura[i] != usuario[i])
      return (false);
```

```
  }
  return (true);
}

void obtenerUsuarios(fs::FS &fs) {
  myFile = fs.open("/cards.txt");
  int count = 0;


  //Obtener usuarios guardados en la SD
  if (myFile) {
    while (myFile.available()) {
      String list = myFile.readStringUntil('\r');
      Serial.println(list);
      list.trim();
      Users[count] = list;
      count++;
    }


    myFile.close();
  } else {
    Serial.println("Error abriendo el archivo cards.txt");
  }
  if (count > 0) {
    cantUsuarios = count + 1;
  }
  else {
    cantUsuarios = 0;
  }

  //Imprimir usuarios obtenidos
  Serial.println("Usuarios:");
  for (int i = 0; i < cantUsuarios; i++) {
    Serial.println(Users[i]);
  }
```

```
  //Deserializar y guardar usuarios obtenidos
  for (int i = 0; i < cantUsuarios; i++) {
    parseJson(Users[i], i);
  }
}


void agregarUsuario() {

  if (!funcionActiva) {
    Serial.println("Agregar Usuario");
    Serial.println("Acerque la tarjeta a ingresar");

  }
  funcionActiva = true;
  delay(50);


  if (!reader.PICC_IsNewCardPresent())
    return;


  if (!reader.PICC_ReadCardSerial())
    return;


  Serial.print("UID:");
  for (byte i = 0; i < reader.uid.size; i++) {
    if (reader.uid.uidByte[i] < 0x10) {
      Serial.print(" 0");
    }
    else {
      Serial.print(" ");
    }
    Serial.print(reader.uid.uidByte[i], HEX);
    LecturaUID[i] = reader.uid.uidByte[i];
  }
```

```
  // Halt PICC
  reader.PICC_HaltA();
  // Stop encryption on PCD
  reader.PCD_StopCrypto1();

  Serial.print("\t");

  bool userEncontrado = false;
  for (int i = 0; i < cantUsuarios; i++) {
    if (comparaUID(LecturaUID, Usuarios[i])) {
      userEncontrado = true;
    }
  }
  if (userEncontrado) {
    Serial.println("La tarjeta ya esta en el sistema");
  } else {
    for (int i = 0; i < 4; i++) {
      Usuarios[cantUsuarios][i] = LecturaUID[i];
    }
    cantUsuarios++;
    guardarUsuario(SD ,(cantUsuarios), "0000", LecturaUID);
  }

  funcionActiva = false;

}


void guardarUsuario(fs::FS &fs, int jsonId, String jsonPass, byte
jsonUID[4]) {

  myFile = fs.open("/cards.txt", FILE_APPEND);

  StaticJsonDocument<200> doc;

  String jsonString;
```

```
  doc["id"] = jsonId;
  doc["pass"] = jsonPass;

  JsonArray data = doc.createNestedArray("UID");
  for (int i = 0; i < 4; i++) {
    data.add(jsonUID[i]);
  }

  serializeJson(doc, jsonString);

  Serial.println("Usuario a guardar:");
  Serial.println(jsonString);

  if (myFile) {
    delay(50);
    myFile.println(jsonString);
    Serial.println("Guardando usuario en cards.txt...");
    delay(50);
    myFile.close();
    Serial.println("Guardado.");
  } else {
    Serial.println("Error abriendo cards.txt");
  }
}


void parseJson(String jsonToParse, int userID) {
  StaticJsonDocument<200> doc;

  //Deserializar Json
  DeserializationError error = deserializeJson(doc, jsonToParse);

  //Comprobar errores en la deserializacion
  if (error) {
    Serial.println("");
    Serial.print(F("deserializeJson() failed: "));
```

```
    Serial.println(error.f_str());
    return;
  }

  //Guardar usuario
  int jsonId = doc["id"];
  String jsonPass = doc["pass"];
  for (int i = 0; i < 4; i++) {
    Usuarios[userID][i] = doc["UID"][i];
  }

  /*
    //Imprimir usuario
    Serial.println("");
    Serial.print("ID: ");
    Serial.println(jsonId);
    Serial.print("PASS: ");
    Serial.println(jsonPass);
    Serial.print("UID: ");
    for (int i = 0; i < 4; i++) {
    Serial.print(Usuarios[userID][i],HEX);
    Serial.print(", ");
    }
  */
}
```