# **Presentación del prototipo**

Prototipo IoT de Control de Acceso con Arduino UNO y Tecnologías Inalámbricas (RFID, Bluetooth, Ultrasonido)

NOMBRE: Víctor Gangas, Diego Díaz

CARRERA: Ingeniería en informática

ASIGNATURA: Aplicaciones Móviles Para Iot

PROFESOR: Pablo Hernández

FECHA: 23/09/2025

# **Indice**

Introducción…………………………………….……….3

Desarrollo…………………………………….………….4

Problema y solución propuesta……………….……….4

Componentes utilizados……………………..…………6

* Materiales usados en la maqueta……….7
* función de los Componentes utilizados…7

Conexiones (resumen)…………………………………8

Metodología……………………………………………...11

* Link del video del prototipo………….12

Explicación del Script prototipo………………………..13

Conclusión……………………………………………….22

# **Introducción**

El Internet de las Cosas (IoT) permite integrar comunicación inalámbrica, sensórica y control de actuadores para resolver problemas cotidianos.

En este proyecto se implementa la mejora del prototipo anterior, entregando un sistema basado en Arduino UNO que se combina con un lector RFID RC522, un sensor ultrasónico HC-SR04, un servomotor y un módulo Bluetooth HC-06, para simular la apertura y cierre de una puerta de forma segura y flexible. Además, se utilizaron materiales de apoyo como palos de helado, cajas y cinta adhesiva para construir la maqueta y representar de manera visual la idea del proyecto.

El sistema opera en dos modos seleccionables por Bluetooth: modo RFID, donde la puerta se acciona exclusivamente con tarjetas/llaves autorizadas (UIDs), y modo Sensor, donde la apertura/cierre se ejecuta mediante comandos (“Abrir”, “Cerrar”, “M1”, “M2”) validados por presencia en rango mediante el ultrasonido. Se incluye un menú de ayuda invocable desde la app móvil (Serial Bluetooth Terminal) y una arquitectura optimizada para el Arduino UNO: aislamiento del bus SPI del RFID cuando se usa el sensor, asignación de pines sin interferencias y manejo eficiente de memoria.

Este prototipo se enmarca en la unidad de Aplicaciones Móviles para IoT y busca que el estudiante integre sensores y actuadores con comunicación inalámbrica, aplicando programación embebida, electrónica básica y buenas prácticas de conectividad y seguridad (autorización por UID y validación de presencia).

# **Desarrollo**

# **Problema y solución propuesta**

**Alcance de uso:**

Emula el funcionamiento de un portón de estacionamiento o cerradura de edificio: operación rápida y segura con credencial (RFID) o con supervisión por presencia cuando se usa BT, resguardando la seguridad de las personas y evitando aperturas no intencionadas. Su aplicación se proyecta tanto en estacionamientos de centros comerciales como en accesos a condominios y edificios residenciales.

**Problema:** En accesos como estacionamientos y edificios, abrir/cerrar una puerta/portón requiere ir físicamente hasta la cerradura. Este proceso resulta incómodo y poco eficiente, y puede generar inseguridad, especialmente en horarios nocturnos o bajo condiciones climáticas desfavorables. Además, en escenarios de alto flujo vehicular se necesita un control de acceso confiable y una operación ágil para evitar congestiones y vulnerabilidades.

**Solución IoT:**

Se desarrolló un prototipo con Arduino UNO que combina distintos modos de operación:

* **Modo RFID (RC522**): apertura/cierre solo con tarjetas/llaveros autorizados (UID).
* **Modo Sensor (HC‑SR04 + BT)**: apertura/cierre por comandos (“Abrir”, “Cerrar”, “M1”, “M2”) validados por presencia en rango.
* **Conmutación de modo vía Bluetooth (HC‑06)** desde la app Serial Bluetooth Terminal.
* **Servo como actuador del pestillo**. Aislamiento de RFID (SPI) en modo sensor.
* **Menú de ayuda invocable** (“menu”, “help”, “?”) con lista de comandos.

**Objetivos específicos:**

* Integrar comunicación inalámbrica (Bluetooth) con control de actuadores (servo) y autenticación por RFID.
* Implementar validación de presencia por ultrasonido para comandos BT (evitar activaciones sin objeto).
* Demostrar la interacción microcontrolador ⇄ app móvil junto con una arquitectura confiable en recursos limitados.
* Diseñar modos de operación conmutables que emulen casos reales (portón de estacionamiento, acceso a departamentos).

**Brechas de seguridad cubiertas por la solución:**

* Control de acceso por UID en modo RFID: solo credenciales autorizadas accionan el servo.
* Validación de presencia en modo sensor: comandos BT se ejecutan solo si hay objeto dentro del rango permitido.
* Filtrado de comandos: se descartan instrucciones no reconocidas y se separan responsabilidades por modo (en RFID no se aceptan “Abrir/Cerrar” por BT).
* Aislamiento de buses y pines: se reduce interferencia entre módulos, mitigando activaciones erráticas.

**Brechas de seguridad:**

* **Emparejamiento HC-06 con PIN por defecto (1234/0000) → riesgo de acceso no autorizado.**

Solución: cambiar el PIN del módulo HC-06 mediante comandos AT (ejemplo: AT+PIN5678), evitando el uso de credenciales por defecto.

* **Comandos por Bluetooth sin cifrado → riesgo de interceptación/repetición.**

Solución: implementar una capa adicional de autenticación en el software (por ejemplo, aceptar solo comandos con una clave secreta incluida, como “Abrir#123”). Otra opción es migrar a un módulo más avanzado (ej. HC-05 en modo seguro o ESP32 con BLE y cifrado nativo).

* **Sin bloqueo por intentos fallidos → riesgo de fuerza bruta.**

Solución: programar un sistema de bloqueo temporal tras cierto número de intentos fallidos (ejemplo: 3 intentos erróneos → bloquear por 30 segundos y mostrar mensaje de alerta en la app).

* **Servo vulnerable a by-pass mecánico si el pestillo no es robusto.**

Solución: reforzar la parte mecánica con materiales más sólidos (ejemplo: metal o plástico rígido en vez de palo de helado) y diseñar un anclaje que no pueda moverse fácilmente sin la orden del sistema.

* **Caídas de tensión al mover carga → riesgo de reset y apertura involuntaria.**

Solución: usar fuente de alimentación independiente para el servo

# **Componentes utilizados**

Componentes y alimentación

Arduino UNO

Módulo Bluetooth HC‑06 (SPP). Lógica 3.3 V, VCC 5 V

Lector RFID RC522 (VCC 3.3 V)

Sensor ultrasónico HC‑SR04

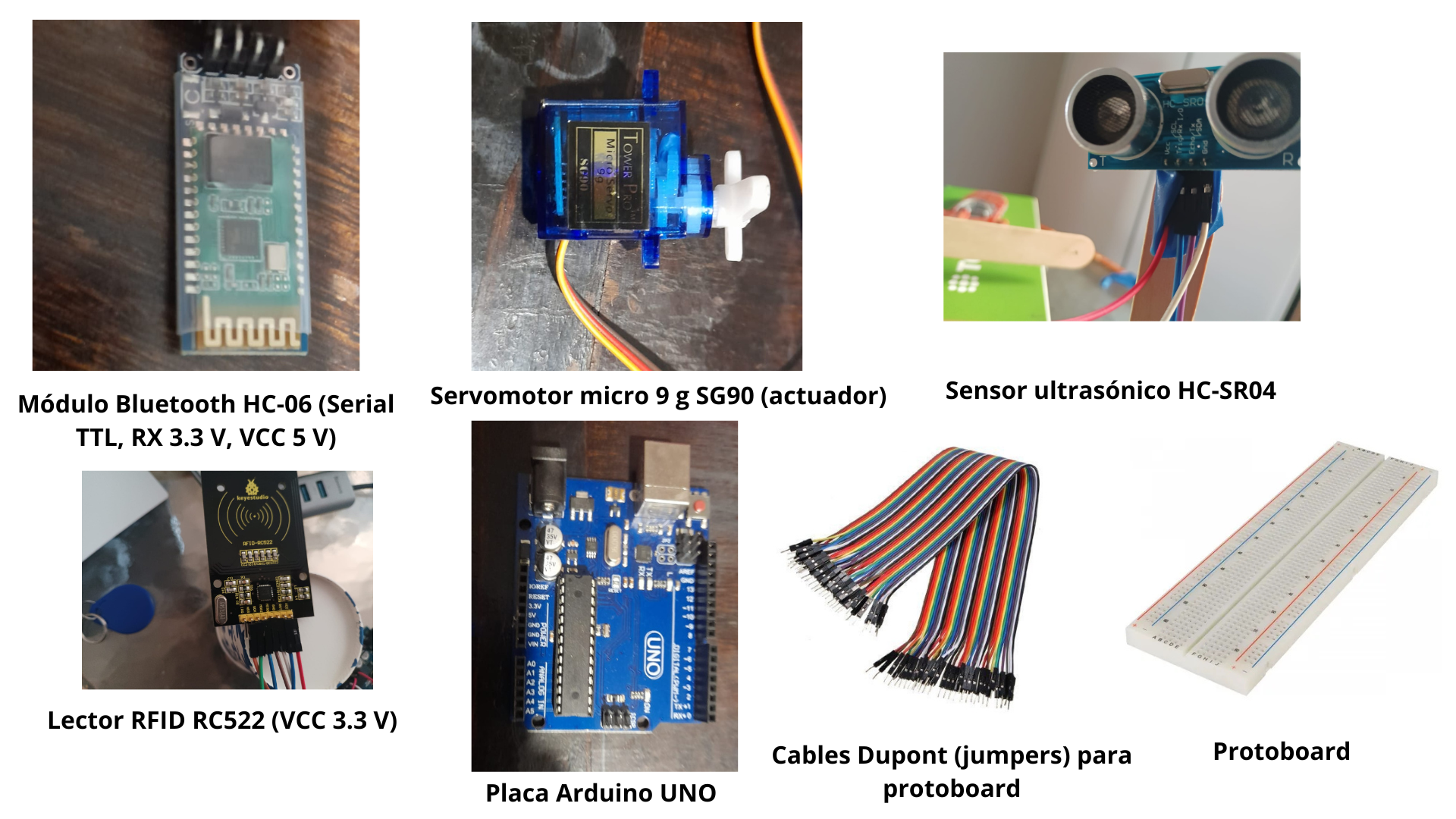
Servomotor SG90

Protoboard y jumpers(20 cables utilizados)

Alimentación por USB (pruebas). Recomendado 5 V externa

Estándar de colores para alimentación:

Rojo = +5 V (riel rojo)

Negro/Blanco = GND (riel azul)

**Imagen de los componentes empleados**

**Tabla con columnas: Componente – Especificación – Función.**

|  |  |  |
| --- | --- | --- |
| Componente | Especificación | Función |
| Arduino UNO | 5 V | Microcontrolador principal |
| HC-06 | Serial TTL, 3.3/5 V | Comunicación inalámbrica Bluetooth |
| RC522 | 3.3 V | Lector de tarjetas RFID |
| HC-SR04 | 5 V | Sensor ultrasónico de proximidad |
| SG90 | 5 V, 9 g | Actuador (pestillo/puerta) |

## **Materiales usados en la maqueta**

Palos de helado – Simulan la barrera de estacionamiento.

Cinta adhesiva – Fijación provisional de piezas y refuerzo estructural.

Caja de cartón – Base estructural en la que pasa el vehículo de prueba.

Vaso de cartón donde sostener el modulo RC522

## **función de los Componentes utilizados**

**Arduino UNO:**

Microcontrolador que ejecuta la lógica de modos, lectura de sensores, validación y control del servo.

**Módulo Bluetooth HC‑06:**

Enlace inalámbrico con el celular (Serial Bluetooth Terminal). Recibe comandos de usuario y permite cambiar de modo (modo1, modo2) y enviar acciones en modo sensor (Abrir/Cerrar, M1/M2).

**Lector RFID RC522:**

Autenticación por credencial. En modo RFID lee el UID de la tarjeta/llavero y, si está autorizado, acciona el servo (toggle abrir/cerrar).

**Sensor ultrasónico HC‑SR04:**

Detección de presencia/objeto. En modo sensor valida que haya un objeto dentro del rango antes de aceptar comandos BT.

**Servomotor SG90:**

Actuador del pestillo. Se mueve a ángulos de “abrir” o “cerrar” cuando lo ordenan el RFID (autorizado) o los comandos BT validados.

Protoboard y jumpers: Distribuyen 5 V y GND y conectan señales entre módulos y el UNO.

Alimentación USB 5 V: Fuente principal durante pruebas; entrega energía al UNO y, de forma limitada, a los periféricos. Para cargas mayores del servo se recomienda fuente 5 V externa, en este caso como prototipo el USB cumple la función.

# **Conexiones (resumen)**

**Arduino UNO**

Alimentación: USB 5 V (PC/adaptador).

GND común para todos los módulos. todas las GND unidas (Arduino, HC‑06, RC522, HC‑SR04 y servo) en el riel GND del protoboard.

**Bluetooth HC‑06 (SPP)**

VCC → 5V Cable hembra a macho del módulo al Protoboard

GND → GND Cable hembra a macho del módulo al Protoboard

TX → D2 (RX de SoftwareSerial) Cable hembra a macho Conexión Directo al Arduino

RX → D3 (TX de SoftwareSerial) [conexión directa en este prototipo] Cable hembra a macho

**RFID RC522 (3.3 V, SPI)**

VCC → 3.3V Cable hembra a macho Conexión Directo al Arduino

GND → GND Cable hembra a macho del módulo al Protoboard

RST → D8 Cable hembra a macho

SS (SDA) → D7 Cable hembra a macho Conexión Directo al Arduino

MOSI → D11 Cable hembra a macho Conexión Directo al Arduino

MISO → D12 Cable hembra a macho Conexión Directo al Arduino

SCK → D13 Cable hembra a macho Conexión Directo al Arduino

**Sensor ultrasónico HC‑SR04**

VCC → 5V Cable hembra a macho del módulo al Protoboard

GND → GND Cable hembra a macho del módulo al Protoboard

TRIG → D4 Cable hembra a macho Conexión Directo al Arduino

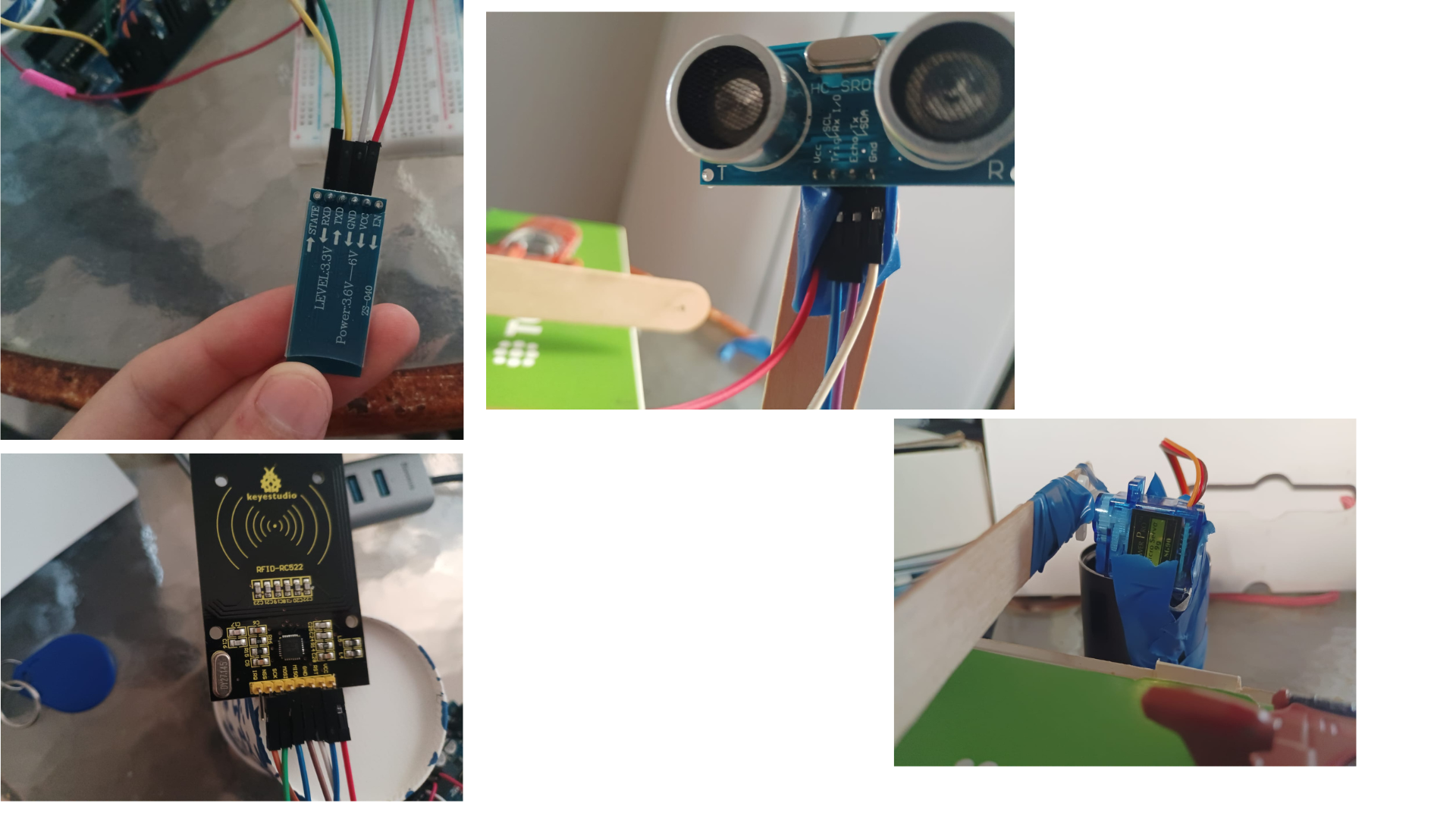
ECHO → D6 Cable hembra a macho Conexión Directo al Arduino

**Servomotor SG90**

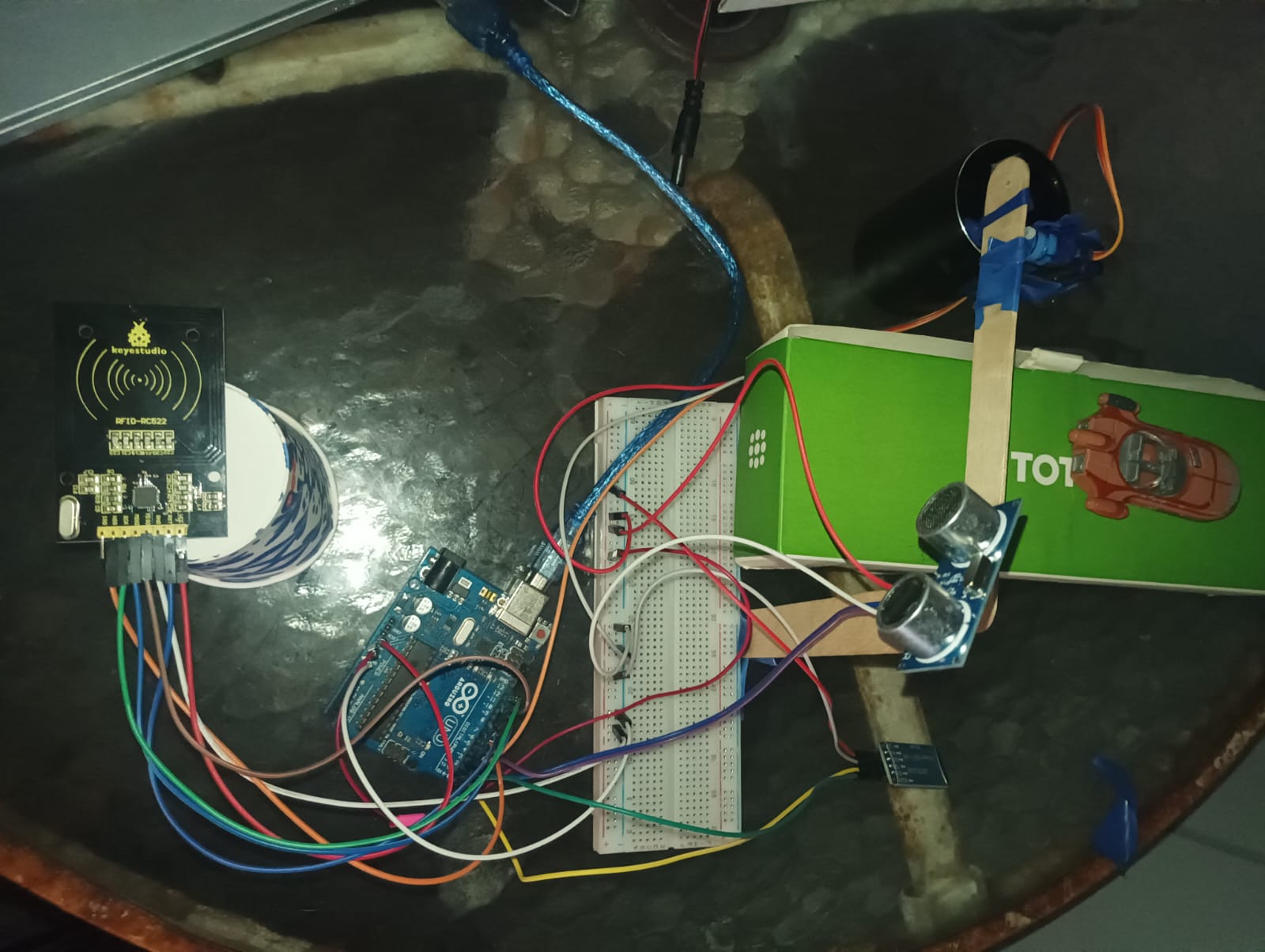
Señal → D5 Cable hembra a macho

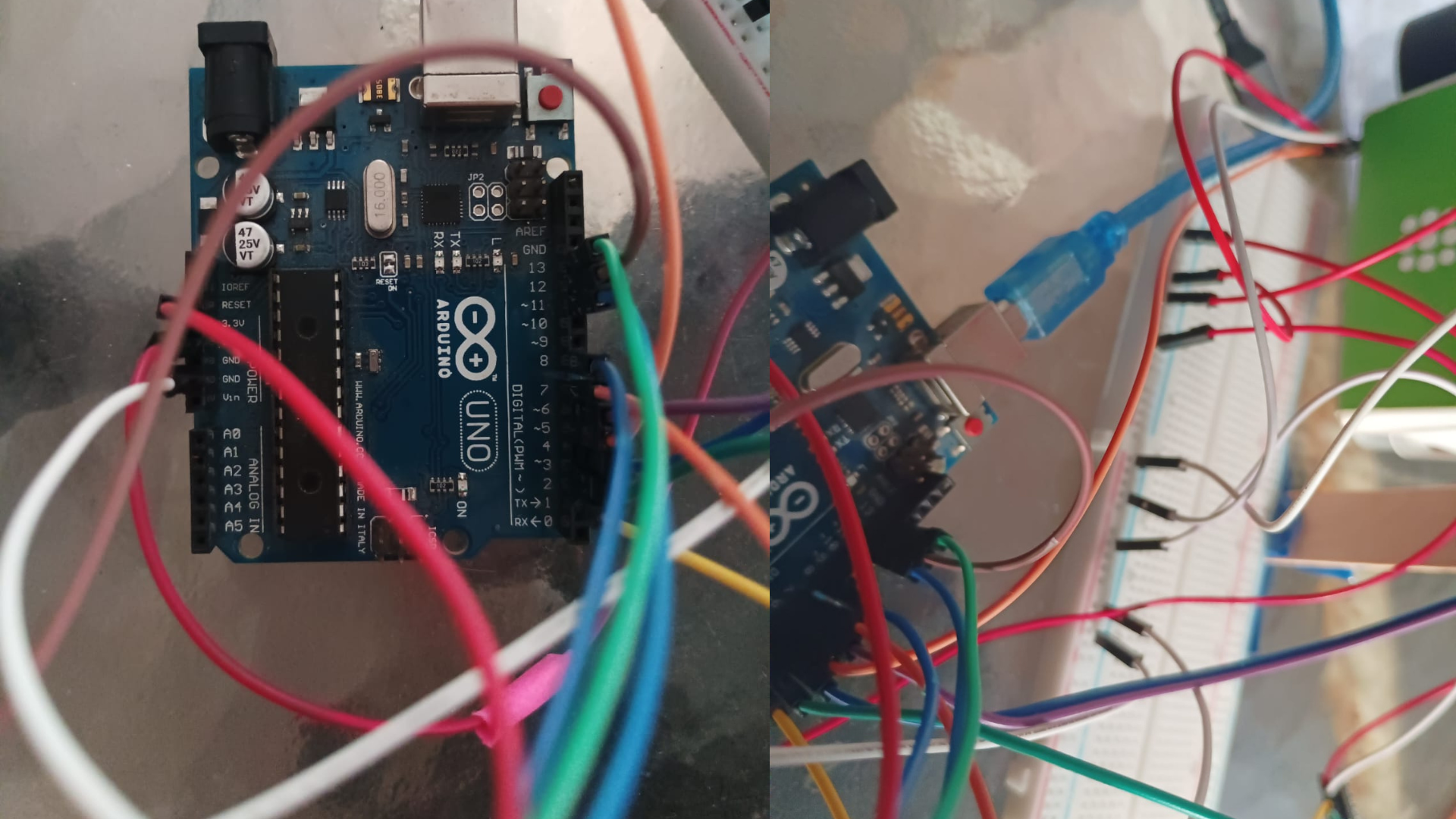
VCC → 5V Cable hembra a macho del módulo al Protoboard

GND → GND Cable hembra a macho del módulo al Protoboard



**Imagen 1 demostrativa de las conexiones**



****

**Imágenes 2 y 3 Demostrativa de las conexiones**

# **Metodología**

**Paso 1 – Problemática**

Accionar puertas/portones a distancia de forma segura, rápida y simple.

**Paso 2 – Retroalimentación docente**

Factibilidad técnica con recursos limitados (Arduino UNO).

Conectividad inalámbrica (HC‑06) + actuador (servo) + autenticación (RFID) / validación de presencia (HC‑SR04).

**Paso 3 – Identificación** **– Diseño funcional (modos)**

**Modo 1 RFID:** apertura/cierre solo con tarjetas/llaveros autorizados (UID).

**Modo 2 Sensor:** apertura/cierre por comandos BT (“Abrir”, “Cerrar”, “M1”, “M2”) válidos solo si hay presencia en rango.

**Conmutación por BT:** comandos “modo1”, “modo2”.

Menú de ayuda: “menu”, “help”, “?”.

**Paso 4 – Adquisición de material**

Se obtienen Arduino UNO, HC‑06, RC522, HC‑SR04, SG90, protoboard y jumpers, cable USB (5 V). Capacitores 100 µF + 100 nF (desacople servo).

**Paso 5 – Estudio de funcionamiento**

HC‑06: VCC 5 V, GND, TX→D2, RX→D3

RC522 (3.3 V): VCC 3.3 V, GND, RST→D8, SS→D7, MOSI→D11, MISO→D12, SCK→D13.

HC‑SR04: VCC 5 V, GND, TRIG→D4, ECHO→

Servo: señal→D5, VCC 5 V, GND

**Paso 6 – Conexión y programación**

Lector RC522 por SPI; aislamiento: en modo sensor se hace SPI.end() y pines SS/RST en alta impedancia; al volver a RFID se reactiva SPI.begin() + PCD\_Init().

Lógica de modos y menú de ayuda.

Validación de presencia: dos lecturas HC‑SR04, promedio y verificación en rango.

Comandos case‑insensitive para “Abrir”/“Cerrar”.

**Paso 7 – Prototipo (maqueta)**

Se entrega junto al informe un video e imagen del funcionamiento del prototipo.

**Link del video del prototipo**: <https://www.youtube.com/watch?v=AmXXXI_lIiI>

En el video del prototipo se evidencian las siguientes pruebas de uso

Prueba Modo 1: presentar tarjeta UID válida → toggle servo; rechazar UID no autorizado.

Prueba Modo 2: verificar mensajes “Detectado a X cm” / “Nada detectado…”, y que comandos solo ejecuten con presencia.

Prueba de confirmación “menu/help/?” muestren la ayuda y que “modo1/modo2” cambien correctamente.

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

Imagen demostrativa del menu

**Nota de proyección:**

Si bien el prototipo se implementó con un único Arduino UNO por limitaciones de recursos, la arquitectura puede ampliarse a dos microcontroladores interconectados mediante comunicación serial por Bluetooth o WiFi. Esto permitiría distribuir funciones (por ejemplo: un microcontrolador dedicado a sensórica y otro a control de actuadores). En este caso con los materiales actuales se cumple las funciones esperadas.

# **Explicación del Script**

**Inclusión de librerías**

* **#include <SoftwareSerial.h>:** comunicación serie por pines D2 (RX) y D3 (TX) para el HC‑06.
* **#include <Servo.h>:** control del servomotor SG90 por el pin D5.
* **#include <SPI.h>:** bus SPI para el lector RFID RC522.
* **#include <MFRC522.h>:** API del RC522 para detectar tarjetas y leer su UID.

**Declaración de objetos y pines**

* SoftwareSerial BT(2, 3); → Enlace Bluetooth (Serial Bluetooth Terminal).
* Servo servo; → Actuador del pestillo (D5).
* MFRC522 mfrc522(SS\_PIN, RST\_PIN); → Lector RFID (SS=D7, RST=D8; MOSI=D11, MISO=D12, SCK=D13).
* const int trigPin = 4, echoPin = 6; → Sensor ultrasónico HC‑SR04.
* UIDs permitidos: uidAbrir y uidCerrar (unsigned long) → credenciales autorizadas.

**Variables principales**

* modo (byte): 1 = RFID, 2 = SENSOR (ultrasonido).
* ultimaPresenciaOK, ultimaDistancia: control de validación de presencia y última distancia medida.
* entrada[16], entradaLen: buffer de texto para comandos BT (pequeño para ahorrar RAM).
* Timers: ultimoAviso (mensajes cada 1000 ms), ultimoRFID (lectura periódica de RFID).
* Configuración inicial (setup)
* Inicializa BT y Serial a 9600 baud.
* Posiciona el servo en 0° y lo desacopla tras el movimiento para evitar consumo/jitter.
* Configura TRIG como salida y ECHO como entrada.
* Inicializa SPI y el RC522.
* Muestra un menú de ayuda y el modo actual.

**Funciones de utilidad**

* btPrint/btPrintln: envían el mismo texto por Bluetooth y por USB (Serial).
* mostrarModo(): imprime el modo actual.
* mostrarMenu(): lista de comandos disponibles y prompt ‘> ’.
* Bucle principal (loop)
* Sin auto‑retorno: el modo se mantiene hasta que el usuario envíe “modo1” o “modo2”.
* Modo SENSOR (modo == 2):
* Desactiva completamente el RFID (SPI.end() y SS/RST en alta impedancia).
* medirPresencia(): realiza 2 lecturas del HC‑SR04, promedia y valida con enRango().
* Cada 1000 ms informa: “Detectado a X cm” o “Nada detectado o fuera de rango”.

**Lectura de Bluetooth:**

* Acumula caracteres en entrada; al recibir fin de línea, llama a procesarComando().
* Modo RFID (modo == 1):
* Lee periódicamente el RC522 (cada ~300 ms). Si hay tarjeta y su UID coincide con los autorizados, hace toggle del servo:
* “OK: RFID ABRIR” si estaba cerrado.
* “OK: RFID CERRAR” si estaba abierto.
* Rechaza tarjetas no autorizadas con “Tarjeta no autorizada”.
* Procesamiento de comandos (procesarComando)

**Cambios de modo:**

* “modo1” → activa RFID (SPI.begin + PCD\_Init, SS/RST a salida).
* “modo2” → activa SENSOR (SPI.end, SS/RST en entrada).
* Ayuda:
* “menu”, “help”, “?” → muestra el menú.

**Comandos en modo SENSOR (con validación de presencia):**

* “Abrir”, “A”, “1” → mueve servo a 90° (case‑insensitive para “Abrir”).
* “Cerrar”, “C”, “2” → mueve servo a 0° (case‑insensitive para “Cerrar”).
* “M1” → Abrir; “M2” → Cerrar (también solo en modo sensor).

En modo RFID, los comandos distintos de “modo1”/“modo2” se rechazan: “ERROR: En modo RFID solo usar tarjetas o comandos: modo1, modo2”.

**Control del servo**

* moverServo(int ángulo): attach → write → delay (~400 ms) → detach. Se hace el cierre de seguridad para evitar consumo constante, vibraciones (jitter) y que el servo quede forzado si alguien empuja el pestillo.
* toggleServo(): alterna entre 0° y 90° y reporta el estado por BT/Serial.

**Medición de presencia (ultrasonido)**

* medirPresencia():Llama dos veces a medirDistancia(), promedia (si ambas válidas) y guarda ultimaDistancia.
* Retorna true solo si ambas lecturas están en rango [distanciaMin, distanciaMax].
* medirDistancia():Dispara TRIG 10 µs; mide con pulseIn(echoPin, HIGH, 6000 µs) y convierte a cm.
* Devuelve −1 si no hay eco (timeout).
* enRango(int d):
* Verifica límites configurados (por defecto 5–100 cm).

En resumen, el flujo principal es: el usuario selecciona el modo → el sistema valida la entrada (tarjeta en RFID o presencia para comandos BT en Sensor) → si la validación es correcta, se acciona el servo (con detach al finalizar) → se informa el resultado por Bluetooth y por Serial.

|  |
| --- |
| #include <SoftwareSerial.h>  #include <Servo.h>  #include <SPI.h>  #include <MFRC522.h>  // ---------- CONFIG ----------  SoftwareSerial BT(2, 3);  Servo servo;  const int SERVO\_PIN = 5;  const int trigPin = 4;  const int echoPin = 6;  // RFID RC522  #define RST\_PIN 8  #define SS\_PIN 7  MFRC522 mfrc522(SS\_PIN, RST\_PIN);  // UIDs permitidos  const unsigned long uidAbrir = 2747323383;  const unsigned long uidCerrar = 2205297957;  // Rango válido  const int distanciaMin = 5;  const int distanciaMax = 100;  // Estado  int estadoServo = 0;  bool servoAdjunto = false;  unsigned long ultimaPresenciaOK = 0;  int ultimaDistancia = -1;  // Buffer simple  char entrada[16];  byte entradaLen = 0;  // Modo (1=rfid, 2=sensor)  byte modo = 1;  // Timing  unsigned long ultimoAviso = 0;  unsigned long ultimoRFID = 0;  // ---------- UTIL PRINT ----------  void btPrint(const char\* s) {  BT.print(s);  Serial.print(s);  }  void btPrintln(const char\* s) {  BT.println(s);  Serial.println(s);  }  // ---------- STRING UTILS ----------  bool equalsIgnoreCase(const char\* a, const char\* b) {  while (\*a && \*b) {  char ca = \*a;  char cb = \*b;  if (ca >= 'A' && ca <= 'Z') ca += 32; // to lower  if (cb >= 'A' && cb <= 'Z') cb += 32; // to lower  if (ca != cb) return false;  ++a; ++b;  }  return (\*a == '\0' && \*b == '\0');  }  // ---------- SETUP ----------  void setup() {  BT.begin(9600);  Serial.begin(9600);  servo.attach(SERVO\_PIN);  servo.write(0);  delay(400);  servo.detach();  servoAdjunto = false;  estadoServo = 0;  pinMode(trigPin, OUTPUT);  pinMode(echoPin, INPUT);  SPI.begin();  mfrc522.PCD\_Init();  mostrarMenu();  }  void mostrarModo() {  btPrint("Modo: ");  if (modo == 1) btPrintln("RFID (Tarjetas + BT)");  else btPrintln("SENSOR (Ultrasonico + BT)");  }  void mostrarMenu() {  btPrintln("\r\n=== Control BT + RFID + Ultrasonico ===");  btPrintln("Comandos:");  btPrintln(" modo1 -> RFID (tarjetas)");  btPrintln(" modo2 -> SENSOR (ultrasonico)");  btPrintln(" Abrir/A/1 -> Abrir (solo modo sensor)");  btPrintln(" Cerrar/C/2 -> Cerrar (solo modo sensor)");  btPrintln(" M1 -> Abrir (solo modo sensor)");  btPrintln(" M2 -> Cerrar (solo modo sensor)");  mostrarModo();  btPrint("\r\n> ");  }  void cambiarModo(byte nuevoModo) {  if (nuevoModo == 1) {  // Reactivar RFID  pinMode(RST\_PIN, OUTPUT);  pinMode(SS\_PIN, OUTPUT);  SPI.begin();  mfrc522.PCD\_Init();  btPrintln("RFID reactivado");  } else {  // Desactivar RFID  SPI.end();  pinMode(RST\_PIN, INPUT);  pinMode(SS\_PIN, INPUT);  btPrintln("RFID desactivado");  }    modo = nuevoModo;  mostrarModo();  }  // ---------- LOOP ----------  void loop() {  // Sin auto-retorno automático - solo cambia con comandos  // 1) Sensor (solo en modo 2)  if (modo == 2) {  // Deshabilitar RFID completamente en modo sensor  SPI.end();  // Desconectar pines RFID físicamente  pinMode(RST\_PIN, INPUT);  pinMode(SS\_PIN, INPUT);    bool presencia = medirPresencia();  if (presencia) ultimaPresenciaOK = millis();  // Avisos cada 1000ms  if (millis() - ultimoAviso >= 1000) {  if (presencia) {  btPrint("Detectado a ");  btPrint(String(ultimaDistancia).c\_str());  btPrintln(" cm");  } else {  btPrintln("Nada detectado o fuera de rango");  }  ultimoAviso = millis();  }  }  // 2) Lectura BT  while (BT.available()) {  char c = BT.read();    if (c == '\n' || c == '\r') {  entrada[entradaLen] = '\0';  if (entradaLen > 0) {  procesarComando();  }  entradaLen = 0;  btPrint("\r\n> ");  } else if (entradaLen < 15) {  entrada[entradaLen++] = c;  }  }  // 3) RFID (solo en modo 1)  if (modo == 1 && (millis() - ultimoRFID >= 300)) {  if (mfrc522.PICC\_IsNewCardPresent()) {  if (mfrc522.PICC\_ReadCardSerial()) {  unsigned long uid = 0;  for (byte i = 0; i < 4; i++) {  uid = (uid << 8) | mfrc522.uid.uidByte[i];  }    btPrint("\r\nRFID: ");  btPrintln(String(uid).c\_str());    if (uid == uidAbrir || uid == uidCerrar) {  toggleServo();  } else {  btPrintln("Tarjeta no autorizada");  }    mfrc522.PICC\_HaltA();  mfrc522.PCD\_StopCrypto1();  }  }  ultimoRFID = millis();  }  delay(10);  }  // ---------- COMANDOS ----------  void procesarComando() {  // Comandos de modo  if (strcmp(entrada, "modo1") == 0) {  cambiarModo(1);  return;  }  if (strcmp(entrada, "modo2") == 0) {  cambiarModo(2);  return;  }  // Mostrar menú  if (strcmp(entrada, "menu") == 0 || strcmp(entrada, "help") == 0 || strcmp(entrada, "?") == 0) {  mostrarMenu();  return;  }  // Validar presencia (solo en modo sensor)  if (modo == 2) {  bool ahora = medirPresencia();  bool ventana = (millis() - ultimaPresenciaOK) <= 2000;    if (!(ahora || ventana)) {  btPrintln("ERROR: Sin objeto en rango");  return;  }  }  // Comandos solo funcionan en modo sensor  if (modo == 2) {  // Comandos de control  if (equalsIgnoreCase(entrada, "Abrir") || strcmp(entrada, "A") == 0 ||  strcmp(entrada, "1") == 0) {  btPrintln("OK: ABRIR");  moverServo(90);  return;  }    if (equalsIgnoreCase(entrada, "Cerrar") || strcmp(entrada, "C") == 0 ||  strcmp(entrada, "2") == 0) {  btPrintln("OK: CERRAR");  moverServo(0);  return;  }  // Comandos M1 y M2 (solo en modo sensor con presencia)  if (strcmp(entrada, "M1") == 0) {  btPrintln("OK: M1 ABRIR");  moverServo(90);  return;  }  if (strcmp(entrada, "M2") == 0) {  btPrintln("OK: M2 CERRAR");  moverServo(0);  return;  }  } else {  // En modo RFID, solo comandos de modo son válidos  btPrintln("ERROR: En modo RFID solo usar tarjetas o comandos: modo1, modo2");  }  btPrintln("Comando invalido");  }  // ---------- SERVO ----------  void moverServo(int angulo) {  if (estadoServo == angulo) return;    servo.attach(SERVO\_PIN);  servo.write(angulo);  delay(400);  servo.detach();    estadoServo = angulo;  }  void toggleServo() {  if (estadoServo == 0) {  btPrintln("OK: RFID ABRIR");  moverServo(90);  } else {  btPrintln("OK: RFID CERRAR");  moverServo(0);  }  }  // ---------- SENSADO ----------  bool medirPresencia() {  int d1 = medirDistancia();  delay(4);  int d2 = medirDistancia();  int best = (d1 > 0 && d2 > 0) ? (d1 + d2) / 2 : (d1 > 0 ? d1 : d2);  ultimaDistancia = best;  return enRango(d1) && enRango(d2);  }  inline bool enRango(int d) {  return (d > 0 && d >= distanciaMin && d <= distanciaMax);  }  int medirDistancia() {  long suma = 0;  int lecturas = 2;  int validas = 0;  for (int i = 0; i < lecturas; i++) {  digitalWrite(trigPin, LOW);  delayMicroseconds(2);  digitalWrite(trigPin, HIGH);  delayMicroseconds(10);  digitalWrite(trigPin, LOW);  // Timeout corto ~ 6 ms (hasta ~100 cm) para no bloquear  unsigned long dur = pulseIn(echoPin, HIGH, 6000UL);  if (dur > 0) {  float cm = (dur \* 0.034f) / 2.0f;  suma += (long)cm;  validas++;  }  }  if (validas == 0) return -1;  return (int)(suma / validas);  } |

**Junto al informe se hace entrega en el archivo comprimido el cual contienen el Script**

# **Conclusión**

La construcción del prototipo permitió consolidar aprendizajes clave sobre la integración de hardware y software en el contexto del IoT

La construcción del prototipo permitió resolver la problemática de apertura/cierre a distancia con criterios de seguridad y usabilidad. Se integraron tres tecnologías clave: autenticación por credencial mediante RFID RC522 (modo RFID), validación de presencia con sensor ultrasónico HC‑SR04 para habilitar comandos por Bluetooth (modo Sensor) y actuación con servomotor SG90. La conmutación de modos vía HC‑06 y el menú de ayuda simplifican la operación desde el celular.

Como resultado, se obtuvo un prototipo funcional que responde a la problemática planteada: abrir o cerrar una cerradura sin necesidad de intervención manual directa, constituyendo un primer paso hacia soluciones más escalables aplicables en estacionamientos, accesos de edificios y entornos residenciales.