MFRC522 RFID Reader

1. Code for getting the card’s code like {0x49, 0x3E, 0xEA, 0x6E} this

#include <SPI.h>

#include <MFRC522.h>

#define SS\_PIN 10

#define RST\_PIN 9

MFRC522 mfrc522(SS\_PIN, RST\_PIN);   // Create MFRC522 instance.

void setup()

{

  Serial.begin(9600);   // Initiate a serial communication

  SPI.begin();      // Initiate  SPI bus

  mfrc522.PCD\_Init();   // Initiate MFRC522

  Serial.println("Approximate your card to the reader...");

  Serial.println();

}

void loop() {

  // Check if a card is present

  if (mfrc522.PICC\_IsNewCardPresent() && mfrc522.PICC\_ReadCardSerial()) {

    // Read the card's UID

    byte cardID[4];

    for (byte i = 0; i < 4; i++) {

      cardID[i] = mfrc522.uid.uidByte[i];

    }

    // Print the byte values of the card ID

    Serial.print("Card ID: {");

    for (byte i = 0; i < 4; i++) {

      Serial.print("0x");

      if (cardID[i] < 0x10) {

        Serial.print("0");

      }

      Serial.print(cardID[i], HEX);

      if (i < 3) {

        Serial.print(", ");

      }

    }

    Serial.println("}");

    // Halt PICC and stop encryption process

    mfrc522.PICC\_HaltA();

    mfrc522.PCD\_StopCrypto1();

  }

}

2. Code for checking the card and its validations:

#include <SPI.h>

#include <MFRC522.h>

#define SS\_PIN 10

#define RST\_PIN 9

int ledPin=4;

MFRC522 mfrc522(SS\_PIN, RST\_PIN);   // Create MFRC522 instance

unsigned long disableTime = 0;      // Stores the time until the card is disabled

unsigned long disableTime2 = 0;      // Stores the time until the card is disabled

unsigned long disableDuration = 20000;  // Time duration (in milliseconds) to disable the card

// Define the IDs of the cards to allow/deny access

byte allowedCardID[] = {0x49, 0x3E, 0xEA, 0x6E};  // Example allowed card ID

byte allowedCardID2[] = {0xF0, 0xE2, 0x25, 0x19};  // Example allowed card ID

byte deniedCardID[] = {0xEE, 0xFF, 0x11, 0x22};   // Example denied card ID

void setup() {

  Serial.begin(9600);   // Initialize serial communication

  SPI.begin();          // Initialize SPI bus

  mfrc522.PCD\_Init();   // Initialize MFRC522

  pinMode(ledPin,OUTPUT);

  // Print a message to indicate the setup is complete

  Serial.println("RFID Reader initialized. Bring an RFID card closer to the reader.");

}

void loop() {

  // Check if a card is present

  if (mfrc522.PICC\_IsNewCardPresent() && mfrc522.PICC\_ReadCardSerial()) {

    // Read the card's UID

    byte cardID[4];

    for (byte i = 0; i < 4; i++) {

      cardID[i] = mfrc522.uid.uidByte[i];

    }

    // Compare the card's UID with the allowedCardID

    if (memcmp(cardID, allowedCardID, sizeof(cardID)) == 0) {

      // Check if the card should be disabled or not

      if (millis() > disableTime) {

        // Access granted

        Serial.println("Access granted!");

        disableTime = millis() + disableDuration;  // Disable the card for a certain time

        digitalWrite(ledPin,HIGH);

        delay(2000);

        digitalWrite(ledPin,LOW);

      } else {

        // Card disabled

        Serial.println("Card disabled. Please wait.");

      }

    }

    if (memcmp(cardID, allowedCardID2, sizeof(cardID)) == 0) {

      // Check if the card should be disabled or not

      if (millis() > disableTime2) {

        // Access granted

        Serial.println("Access granted!");

        disableTime2 = millis() + disableDuration;  // Disable the card for a certain time

        digitalWrite(ledPin,HIGH);

        delay(2000);

        digitalWrite(ledPin,LOW);

      } else {

        // Card disabled

        Serial.println("Card disabled. Please wait.");

      }

    }

    // Halt PICC and stop encryption process

    mfrc522.PICC\_HaltA();

    mfrc522.PCD\_StopCrypto1();

  }

}

FULL project code:

#include <SPI.h>

#include <MFRC522.h>

#define SS\_PIN 10

#define RST\_PIN 9

int ledPin = 4;

int ledpin2 = A0;

// one ultrasonic sensor for tank

const int trigPin1 = 6;

const int echoPin1 = 7;

// another ultrasonic sensor for measering how much rain happen

const int trigPin2 = 2;

const int echoPin2 = 3;

// ledpin3 for all system disableness

int ledpin3 = A2;

MFRC522 mfrc522(SS\_PIN, RST\_PIN);  // Create MFRC522 instance

unsigned long disableTime = 0;              // Stores the time until the card is disabled

unsigned long disableTime2 = 0;             // Stores the time until the card is disabled

unsigned long disableTimeForAllSystem = 0;  // Stores the time until the card is disabled

unsigned long disableDuration = 20000;  // Time duration (in milliseconds) to disable the card

// Define the IDs of the cards to allow/deny access

byte allowedCardID[] = { 0x49, 0x3E, 0xEA, 0x6E };   // Example allowed card ID

byte allowedCardID2[] = { 0xF0, 0xE2, 0x25, 0x19 };  // Example allowed card ID

byte deniedCardID[] = { 0xEE, 0xFF, 0x11, 0x22 };  // Example denied card ID

void setup() {

  Serial.begin(9600);  // Initialize serial communication

  SPI.begin();         // Initialize SPI bus

  mfrc522.PCD\_Init();  // Initialize MFRC522

  pinMode(ledPin, OUTPUT);

  // pinmode for ultrasonic sensor 1

  pinMode(trigPin1, OUTPUT);

  pinMode(echoPin1, INPUT);

  // pinmode for ultrasonic sensor 2

  pinMode(trigPin2, OUTPUT);

  pinMode(echoPin2, INPUT);

  // pinmode for tank

  pinMode(ledpin2, OUTPUT);

  // pinmode for delay system for a cirtain time

  pinMode(ledpin3, OUTPUT);

  // Print a message to indicate the setup is complete

  Serial.println("RFID Reader initialized. Bring an RFID card closer to the reader.");

}

void loop() {

  int distance2 = ultraSonicDistance2();  //measuring how much rain fallen

  if (distance2 >= 10 && distance2 <= 12) {

    // disableTimeForAllSystem = 10000;

    digitalWrite(ledpin3, HIGH);

    delay(10000);

  }

  if (distance2 >= 8 && distance2 < 10) {

    // disableTimeForAllSystem = 15000;

    digitalWrite(ledpin3, HIGH);

    delay(15000);

  }

  if (distance2 >= 5 && distance2 < 8) {

    // disableTimeForAllSystem = 20000;

    digitalWrite(ledpin3, HIGH);

    delay(20000);

  } else {

    digitalWrite(ledpin3, LOW);

  }

  // Check if a card is present

  if (mfrc522.PICC\_IsNewCardPresent() && mfrc522.PICC\_ReadCardSerial()) {

    // Read the card's UID

    byte cardID[4];

    for (byte i = 0; i < 4; i++) {

      cardID[i] = mfrc522.uid.uidByte[i];

    }

    // Compare the card's UID with the allowedCardID

    if (memcmp(cardID, allowedCardID, sizeof(cardID)) == 0) {

      // Check if the card should be disabled or not

      if (millis() > disableTime) {

        // Access granted

        Serial.println("Access granted!");

        disableTime = millis() + disableDuration;  // Disable the card for a certain time

        digitalWrite(ledPin, HIGH);

        delay(2000);

        digitalWrite(ledPin, LOW);

      } else {

        // Card disabled

        Serial.println("Card disabled. Please wait.");

      }

    }

    if (memcmp(cardID, allowedCardID2, sizeof(cardID)) == 0) {

      // Check if the card should be disabled or not

      if (millis() > disableTime2) {

        // Access granted

        Serial.println("Access granted!");

        disableTime2 = millis() + disableDuration;  // Disable the card for a certain time

        digitalWrite(ledPin, HIGH);

        delay(2000);

        digitalWrite(ledPin, LOW);

      } else {

        // Card disabled

        Serial.println("Card disabled. Please wait.");

      }

      // Halt PICC and stop encryption process

      mfrc522.PICC\_HaltA();

      mfrc522.PCD\_StopCrypto1();

    }

  }

  int distance1 = ultraSonicDistance1();

  // Print the distance

  Serial.print("Distance1 for tank height: ");

  Serial.print(distance1);

  Serial.print(" cm");

  Serial.print("      Distance2 for rain measurement : ");

  Serial.print(distance2);

  Serial.println(" cm");

  if (distance1 > 5 && distance1 < 15) {

    digitalWrite(ledpin2, HIGH);

  } else {

    digitalWrite(ledpin2, LOW);

  }

}

int ultraSonicDistance1() {

  digitalWrite(trigPin1, LOW);

  delayMicroseconds(2);

  // Send a 10us pulse to trigger the sensor

  digitalWrite(trigPin1, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin1, LOW);

  // Measure the duration of the echo pulse

  long duration = pulseIn(echoPin1, HIGH);

  // Calculate the distance in centimeters

  int distance = duration \* 0.034 / 2;

  // Wait for a short delay before taking the next measurement

  delay(500);

  return distance;

}

int ultraSonicDistance2() {

  digitalWrite(trigPin2, LOW);

  delayMicroseconds(2);

  // Send a 10us pulse to trigger the sensor

  digitalWrite(trigPin2, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin2, LOW);

  // Measure the duration of the echo pulse

  long duration = pulseIn(echoPin2, HIGH);

  // Calculate the distance in centimeters

  int distance = duration \* 0.034 / 2;

  // Wait for a short delay before taking the next measurement

  delay(500);

  return distance;

}