LAMPIRAN

Code Program Alat:

```
#include <EEPROM.h>     // membaca dan menulis UID PICC dari/ke EEPROM
#include <SPI.h>        //Modul RC522 menggunakan protokol SPI
#include <MFRC522.h>  // Library untuk Perangkat Mifare RC522
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#define COMMON_ANODE

#ifdef COMMON_ANODE
#define LED_ON HIGH
#define LED_OFF LOW
#else
#define LED_ON LOW
#define LED_OFF HIGH
#endif

constexpr uint8_t redLed = 2;   // Set Led Pins
constexpr uint8_t greenLed = 4;
constexpr uint8_t blueLed = 5;
int groundLed = 3;

constexpr uint8_t relay = A3;     // Set Relay Pin
constexpr uint8_t wipeB = A2;     // Button pin for WipeMode
constexpr uint8_t buzzer = A1;
int pb = A4;   // Pin saklar exit button

boolean match = false;          // initialize card match to false
boolean programMode = false;  // initialize programming mode to false
boolean replaceMaster = false;

uint8_t successRead;    // Variable integer to keep if we have Successful Read from
Reader

byte storedCard[4];   // Stores an ID read from EEPROM
byte readCard[4];   // Stores scanned ID read from RFID Module
byte masterCard[4];   // Stores master card's ID read from EEPROM

// Create MFRC522 instance.
constexpr uint8_t RST_PIN = 9;     // Configurable, see typical pin layout above
```

```
constexpr uint8_t SS_PIN = 10;     // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN);

// Inisialisasi objek LCD

LiquidCrystal_I2C lcd(0x27, 16, 2);  // Alamat I2C dan ukuran LCD (16x2)

////////////////////////////////////// Setup ////////////////////////////////////
void setup() {
  //Arduino Pin Configuration
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(groundLed, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(wipeB, INPUT_PULLUP);   // Enable pin's pull up resistor
  pinMode(relay, OUTPUT);
  pinMode(pb, INPUT_PULLUP);   // Mengatur pin saklar exit button sebagai input
dengan pull-up resistor internal
  //Be careful how relay circuit behave on while resetting or power-cycling your
Arduino
  digitalWrite(relay, LOW);    // Make sure door is locked
  digitalWrite(redLed, LED_OFF);  // Make sure led is off
  digitalWrite(greenLed, LED_OFF);  // Make sure led is off
  digitalWrite(blueLed, LED_OFF); // Make sure led is off
  digitalWrite(groundLed, LED_OFF);
  digitalWrite(buzzer, LOW);

  //Protocol Configuration
  Serial.begin(9600);  // Initialize serial communications with PC
  SPI.begin();           // MFRC522 Hardware uses SPI protocol
  mfrc522.PCD_Init();    // Initialize MFRC522 Hardware


  // Tampilkan pesan selamat datang di LCD
  lcd.backlight();
  lcd.begin(16, 2);
  lcd.print("Sistem Keamanan");
  lcd.setCursor(0, 1);
  lcd.print("Smart Door Lock");
```

```
  //If you set Antenna Gain to Max it will increase reading distance
  //mfrc522.PCD_SetAntennaGain(mfrc522.RxGain_max);

  Serial.println(F("Access Control Example v0.1"));   // For debugging purposes
  ShowReaderDetails();  // Show details of PCD - MFRC522 Card Reader details

  //Wipe Code - If the Button (wipeB) Pressed while setup run (powered on) it wipes
EEPROM
  if (digitalRead(wipeB) == LOW) {  // when button pressed pin should get low,
button connected to ground
    digitalWrite(redLed, LED_ON); // Red Led stays on to inform user we are going to
wipe
    Serial.println(F("Wipe Button Pressed"));
    Serial.println(F("You have 10 seconds to Cancel"));
    Serial.println(F("This will be remove all records and cannot be undone"));
    bool buttonState = monitorWipeButton(10000); // Give user enough time to cancel
operation
    if (buttonState == true && digitalRead(wipeB) == LOW) {    // If button still be
pressed, wipe EEPROM
      Serial.println(F("Starting Wiping EEPROM"));
      for (uint16_t x = 0; x < EEPROM.length(); x = x + 1) {    //Loop end of EEPROM
address
        if (EEPROM.read(x) == 0) {              //If EEPROM address 0
          // do nothing, already clear, go to the next address in order to save time and
reduce writes to EEPROM
        }
        else {
          EEPROM.write(x, 0);       // if not write 0 to clear, it takes 3.3mS
        }
      }
      Serial.println(F("EEPROM Successfully Wiped"));
      digitalWrite(redLed, LED_OFF);  // visualize a successful wipe
      delay(200);
      digitalWrite(redLed, LED_ON);
      delay(200);
      digitalWrite(redLed, LED_OFF);
      delay(200);
      digitalWrite(redLed, LED_ON);
      delay(200);
      digitalWrite(redLed, LED_OFF);
    }
    else {
```

```
    Serial.println(F("Wiping Cancelled")); // Show some feedback that the wipe button did not pressed for 15 seconds
    digitalWrite(redLed, LED_OFF);
   }
  }
  // Check if master card defined, if not let user choose a master card
  // This also useful to just redefine the Master Card
  // You can keep other EEPROM records just write other than 143 to EEPROM address 1
  // EEPROM address 1 should hold magical number which is '143'
  if (EEPROM.read(1) != 143) {
   Serial.println(F("No Master Card Defined"));
   Serial.println(F("Scan A PICC to Define as Master Card"));
   do {


    successRead = getID();            // sets successRead to 1 when we get read from reader otherwise 0
    digitalWrite(blueLed, LED_ON);    // Visualize Master Card need to be defined
    delay(200);
    digitalWrite(blueLed, LED_OFF);
    delay(200);
    lcd.clear();
    lcd.print("Tempelkan Kartu");
    lcd.setCursor(0, 1);
    lcd.print("  Dijadikan MC  ");
    delay(200);


   }
   while (!successRead);                  // Program will not go further while you not get a successful read
   for ( uint8_t j = 0; j < 4; j++ ) {        // Loop 4 times
    EEPROM.write( 2 + j, readCard[j] );  // Write scanned PICC's UID to EEPROM, start from address 3
   }
   EEPROM.write(1, 143);                  // Write to EEPROM we defined Master Card.
   Serial.println(F("Master Card Defined"));
  }
  Serial.println(F("-------------------"));
  Serial.println(F("Master Card's UID"));
  for ( uint8_t i = 0; i < 4; i++ ) {          // Read Master Card's UID from EEPROM
   masterCard[i] = EEPROM.read(2 + i);    // Write it to masterCard
   Serial.print(masterCard[i], HEX);
```

```
  }
  Serial.println("");
  Serial.println(F("-------------------"));
  Serial.println(F("Everything is ready"));
  Serial.println(F("Waiting PICCs to be scanned"));
  cycleLeds();    // Everything ready lets give user some feedback by cycling leds
}


///////////////////////////////////////// Main Loop ///////////////////////////////////
void loop () {

  lcd.backlight();
  lcd.begin(16, 2);
  lcd.print("Sistem Keamanan");
  lcd.setCursor(0, 1);
  lcd.print("Smart Door Lock");


  do {
    if (digitalRead(pb) == LOW) {
      digitalWrite(relay, LOW);
      digitalWrite(blueLed, LED_OFF);   // Turn off blue LED
      digitalWrite(redLed, LED_OFF);  // Turn off red LED
      digitalWrite(greenLed, LED_ON);   // Turn on green LED
      delay(2000);
      delay(2000);


      delay(5000);
      digitalWrite(relay, HIGH);
    }
    else {
      digitalWrite(relay, HIGH);
    }

    successRead = getID();  // sets successRead to 1 when we get read from reader
otherwise 0
    // When device is in use if wipe button pressed for 10 seconds initialize Master
Card wiping
    if (digitalRead(wipeB) == LOW) { // Check if button is pressed
      // Visualize normal operation is iterrupted by pressing wipe button Red is like
more Warning to user
```

```
      digitalWrite(redLed, LED_ON);  // Make sure led is off
      digitalWrite(greenLed, LED_OFF);  // Make sure led is off
      digitalWrite(blueLed, LED_OFF); // Make sure led is off
      // Give some feedback
      Serial.println(F("Wipe Button Pressed"));
      Serial.println(F("Master Card will be Erased! in 10 seconds"));
      bool buttonState = monitorWipeButton(5000); // Give user enough time to cancel
operation
      if (buttonState == true && digitalRead(wipeB) == LOW) {    // If button still be
pressed, wipe EEPROM
        digitalWrite(redLed, LED_OFF);
        digitalWrite(greenLed, LED_ON);
        digitalWrite(blueLed, LED_OFF);
        delay(300);
        digitalWrite(redLed, LED_OFF);
        digitalWrite(greenLed, LED_OFF);
        digitalWrite(blueLed, LED_ON);
        delay(300);
        digitalWrite(redLed, LED_ON);
        digitalWrite(greenLed, LED_OFF);
        digitalWrite(blueLed, LED_OFF);
        delay(300);
        EEPROM.write(1, 0);                  // Reset Magic Number.
        Serial.println(F("Master Card Erased from device"));
        Serial.println(F("Please reset to re-program Master Card"));
        while (1);
      }
      Serial.println(F("Master Card Erase Cancelled"));
    }
    if (programMode) {
      cycleLeds();              // Program Mode cycles through Red Green Blue waiting to
read a new card
    }
    else {
      normalModeOn();     // Normal mode, blue Power LED is on, all others are off
    }
  }

  while (!successRead);   //the program will not go further while you are not getting a
successful read
  if (programMode) {
    if ( isMaster(readCard) ) { //When in program mode check First If master card
scanned again to exit program mode
```

```
      Serial.println(F("Master Card Scanned"));
      Serial.println(F("Exiting Program Mode"));
      Serial.println(F("-----------------------------"));
      programMode = false;
      return;
    }
    else {
      if ( findID(readCard) ) { // If scanned card is known delete it
        Serial.println(F("I know this PICC, removing..."));
        deleteID(readCard);
        Serial.println("-----------------------------");
        Serial.println(F("Scan a PICC to ADD or REMOVE to EEPROM"));
      }
      else {                    // If scanned card is not known add it
        Serial.println(F("I do not know this PICC, adding..."));
        writeID(readCard);
        Serial.println(F("-----------------------------"));
        Serial.println(F("Scan a PICC to ADD or REMOVE to EEPROM"));
      }
    }
  }
  else {
    if ( isMaster(readCard)) {    // If scanned card's ID matches Master Card's ID -
enter program mode
      programMode = true;
      Serial.println(F("Hello Master - Entered Program Mode"));
      uint8_t count = EEPROM.read(0);   // Read the first Byte of EEPROM that
      Serial.print(F("I have "));     // stores the number of ID's in EEPROM
      Serial.print(count);
      Serial.print(F(" record(s) on EEPROM"));
      Serial.println("");
      Serial.println(F("Scan a PICC to ADD or REMOVE to EEPROM"));
      Serial.println(F("Scan Master Card again to Exit Program Mode"));
      Serial.println(F("-----------------------------"));
    }
    else {
      if ( findID(readCard) ) { // If not, see if the card is in the EEPROM
        Serial.println(F("Welcome, You shall pass"));
        granted(300);         // Open the door lock for 300 ms
      }
      else {      // If not, show that the ID was not valid
        Serial.println(F("You shall not pass"));
        denied();
```

```
    }
   }
  }

}

//////////////////////////////////////// Access Diterima    ///////////////////////////////////
void granted ( uint16_t setDelay) {

  digitalWrite(blueLed, LED_OFF);   // Turn off blue LED
  digitalWrite(redLed, LED_OFF);  // Turn off red LED
  digitalWrite(greenLed, LED_ON);   // Turn on green LED
  digitalWrite(relay, LOW);     // Unlock door!
  lcd.clear();
  lcd.print(" Akses Diterima");
  lcd.setCursor(0, 1);
  lcd.print(" Pintu Terbuka");
  delay(2000);
  delay(setDelay);          // Hold door lock open for given seconds
  delay(5000);
  digitalWrite(relay, HIGH);    // Relock door
  delay(250);           // Hold green LED on for a second

}

//////////////////////////////////////// Access Ditolak  ///////////////////////////////////
void denied() {

  digitalWrite(buzzer, HIGH);
  digitalWrite(greenLed, LED_OFF);  // Make sure green LED is off
  digitalWrite(blueLed, LED_OFF);   // Make sure blue LED is off
  digitalWrite(redLed, LED_ON);   // Turn on red LED
  lcd.clear();
  lcd.print("  Akses Ditolak ");
  lcd.setCursor(0, 1);
  lcd.print(" Pintu Terkunci ");
  delay(2000);
  delay(2000);

}
```

```
//////////////////////////////////////// Get PICC's UID ///////////////////////////////////
uint8_t getID() {
  // Getting ready for Reading PICCs
  if ( ! mfrc522.PICC_IsNewCardPresent()) { //If a new PICC placed to RFID reader
continue
    return 0;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) {   //Since a PICC placed get Serial and
continue
    return 0;
  }
  // There are Mifare PICCs which have 4 byte or 7 byte UID care if you use 7 byte
PICC
  // I think we should assume every PICC as they have 4 byte UID
  // Until we support 7 byte PICCs
  Serial.println(F("Scanned PICC's UID:"));
  for ( uint8_t i = 0; i < 4; i++) {  //
    readCard[i] = mfrc522.uid.uidByte[i];
    Serial.print(readCard[i], HEX);
  }
  Serial.println("");
  mfrc522.PICC_HaltA(); // Stop reading
  return 1;
}

void ShowReaderDetails() {
  // Get the MFRC522 software version
  byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
  Serial.print(F("MFRC522 Software Version: 0x"));
  Serial.print(v, HEX);
  if (v == 0x91)
    Serial.print(F(" = v1.0"));
  else if (v == 0x92)
    Serial.print(F(" = v2.0"));
  else
    Serial.print(F(" (unknown),probably a chinese clone?"));
  Serial.println("");
  // When 0x00 or 0xFF is returned, communication probably failed
  if ((v == 0x00) || (v == 0xFF)) {
    Serial.println(F("WARNING: Communication failure, is the MFRC522 properly
connected?"));
    Serial.println(F("SYSTEM HALTED: Check connections."));
    // Visualize system is halted
```

```
    digitalWrite(greenLed, LED_OFF);  // Make sure green LED is off
    digitalWrite(blueLed, LED_OFF);   // Make sure blue LED is off
    digitalWrite(redLed, LED_ON);   // Turn on red LED
    while (true); // do not go further
  }
}

//////////////////////////////////////// Cycle Leds (Program Mode) ///////////////////////////////////
void cycleLeds() {
  digitalWrite(redLed, LED_OFF);  // Make sure red LED is off
  digitalWrite(greenLed, LED_ON);   // Make sure green LED is on
  digitalWrite(blueLed, LED_OFF);   // Make sure blue LED is off
  delay(200);
  digitalWrite(redLed, LED_OFF);  // Make sure red LED is off
  digitalWrite(greenLed, LED_OFF);  // Make sure green LED is off
  digitalWrite(blueLed, LED_ON);  // Make sure blue LED is on
  delay(200);
  digitalWrite(redLed, LED_ON);   // Make sure red LED is on
  digitalWrite(greenLed, LED_OFF);  // Make sure green LED is off
  digitalWrite(blueLed, LED_OFF);   // Make sure blue LED is off
  delay(200);
}

//////////////////////////////////////// Normal Mode Led  ///////////////////////////////////
void normalModeOn () {

  digitalWrite(blueLed, LED_ON);  // Blue LED ON and ready to read card
  digitalWrite(redLed, LED_OFF);  // Make sure Red LED is off
  digitalWrite(greenLed, LED_OFF);  // Make sure Green LED is off
  digitalWrite(relay, HIGH);    // Make sure Door is Locked
  digitalWrite(buzzer, LOW);
}

//////////////////////////////////////// Read an ID from EEPROM //////////////////////////////
void readID( uint8_t number ) {
  uint8_t start = (number * 4 ) + 2;    // Figure out starting position
  for ( uint8_t i = 0; i < 4; i++ ) {     // Loop 4 times to get the 4 Bytes
    storedCard[i] = EEPROM.read(start + i);   // Assign values read from EEPROM to
array
  }
}

//////////////////////////////////////// Add ID to EEPROM   ///////////////////////////////////
```

```
void writeID( byte a[] ) {
  if ( !findID( a ) ) {     // Before we write to the EEPROM, check to see if we have seen this card before!
    uint8_t num = EEPROM.read(0);     // Get the numer of used spaces, position 0 stores the number of ID cards
    uint8_t start = ( num * 4 ) + 6;  // Figure out where the next slot starts
    num++;               // Increment the counter by one
    EEPROM.write( 0, num );     // Write the new count to the counter
    for ( uint8_t j = 0; j < 4; j++ ) {   // Loop 4 times
      EEPROM.write( start + j, a[j] );  // Write the array values to EEPROM in the right position
    }
    successWrite();
    Serial.println(F("Succesfully added ID record to EEPROM"));
  }
  else {
    failedWrite();
    Serial.println(F("Failed! There is something wrong with ID or bad EEPROM"));
  }
}

//////////////////////////////////////// Remove ID from EEPROM   ////////////////////////////////
void deleteID( byte a[] ) {
  if ( !findID( a ) ) {     // Before we delete from the EEPROM, check to see if we have this card!
    failedWrite();      // If not
    Serial.println(F("Failed! There is something wrong with ID or bad EEPROM"));
  }
  else {
    uint8_t num = EEPROM.read(0);   // Get the numer of used spaces, position 0 stores the number of ID cards
    uint8_t slot;       // Figure out the slot number of the card
    uint8_t start;      // = ( num * 4 ) + 6; // Figure out where the next slot starts
    uint8_t looping;    // The number of times the loop repeats
    uint8_t j;
    uint8_t count = EEPROM.read(0); // Read the first Byte of EEPROM that stores number of cards
    slot = findIDSLOT( a );   // Figure out the slot number of the card to delete
    start = (slot * 4) + 2;
    looping = ((num - slot) * 4);
    num--;      // Decrement the counter by one
    EEPROM.write( 0, num );   // Write the new count to the counter
    for ( j = 0; j < looping; j++ ) {         // Loop the card shift times
```

```
    EEPROM.write( start + j, EEPROM.read(start + 4 + j));   // Shift the array values
to 4 places earlier in the EEPROM
   }
   for ( uint8_t k = 0; k < 4; k++ ) {         // Shifting loop
    EEPROM.write( start + j + k, 0);
   }
   successDelete();
   Serial.println(F("Succesfully removed ID record from EEPROM"));
  }
}

//////////////////////////////////////// Check Bytes   ///////////////////////////////////
boolean checkTwo ( byte a[], byte b[] ) {
  if ( a[0] != 0 )      // Make sure there is something in the array first
    match = true;       // Assume they match at first
  for ( uint8_t k = 0; k < 4; k++ ) {   // Loop 4 times
    if ( a[k] != b[k] )     // IF a != b then set match = false, one fails, all fail
      match = false;
  }
  if ( match ) {      // Check to see if if match is still true
    return true;      // Return true
  }
  else  {
    return false;       // Return false
  }
}

//////////////////////////////////////// Find Slot   ///////////////////////////////////
uint8_t findIDSLOT( byte find[] ) {
  uint8_t count = EEPROM.read(0);       // Read the first Byte of EEPROM that
  for ( uint8_t i = 1; i <= count; i++ ) {    // Loop once for each EEPROM entry
    readID(i);               // Read an ID from EEPROM, it is stored in storedCard[4]
    if ( checkTwo( find, storedCard ) ) {   // Check to see if the storedCard read from
EEPROM
      // is the same as the find[] ID card passed
      return i;         // The slot number of the card
      break;          // Stop looking we found it
    }
  }
}

//////////////////////////////////////// Find ID From EEPROM   ///////////////////////////////////
boolean findID( byte find[] ) {
```

```
  uint8_t count = EEPROM.read(0);     // Read the first Byte of EEPROM that
  for ( uint8_t i = 1; i <= count; i++ ) {    // Loop once for each EEPROM entry
    readID(i);          // Read an ID from EEPROM, it is stored in storedCard[4]
    if ( checkTwo( find, storedCard ) ) {   // Check to see if the storedCard read from
EEPROM
      return true;
      break;  // Stop looking we found it
    }
    else {    // If not, return false
    }
  }
  return false;
}

/////////////////////////////////////// Write Success to EEPROM   ///////////////////////////////////
// Flashes the green LED 3 times to indicate a successful write to EEPROM
void successWrite() {
  lcd.clear();
  lcd.print(" Kartu Berhasil ");
  lcd.setCursor(0, 1);
  lcd.print("    Ditambah ");
  digitalWrite(blueLed, LED_OFF);   // Make sure blue LED is off
  digitalWrite(redLed, LED_OFF);  // Make sure red LED is off
  digitalWrite(greenLed, LED_OFF);  // Make sure green LED is on
  delay(200);
  digitalWrite(greenLed, LED_ON);   // Make sure green LED is on
  delay(200);
  digitalWrite(greenLed, LED_OFF);  // Make sure green LED is off
  delay(200);
  digitalWrite(greenLed, LED_ON);   // Make sure green LED is on
  delay(200);
  digitalWrite(greenLed, LED_OFF);  // Make sure green LED is off
  delay(200);
  digitalWrite(greenLed, LED_ON);   // Make sure green LED is on
  delay(200);
  delay(1000);
}

/////////////////////////////////////// Write Failed to EEPROM   ///////////////////////////////////
// Flashes the red LED 3 times to indicate a failed write to EEPROM
void failedWrite() {
  digitalWrite(blueLed, LED_OFF);   // Make sure blue LED is off
  digitalWrite(redLed, LED_OFF);  // Make sure red LED is off
```

```
  digitalWrite(greenLed, LED_OFF);  // Make sure green LED is off
  delay(200);
  digitalWrite(redLed, LED_ON);   // Make sure red LED is on
  delay(200);
  digitalWrite(redLed, LED_OFF);  // Make sure red LED is off
  delay(200);
  digitalWrite(redLed, LED_ON);   // Make sure red LED is on
  delay(200);
  digitalWrite(redLed, LED_OFF);  // Make sure red LED is off
  delay(200);
  digitalWrite(redLed, LED_ON);   // Make sure red LED is on
  delay(200);
  delay(1000);
}

//////////////////////////////////////// Success Remove UID From EEPROM
///////////////////////////////////
// Flashes the blue LED 3 times to indicate a success delete to EEPROM
void successDelete() {
  lcd.clear();
  lcd.print(" Kartu Berhasil ");
  lcd.setCursor(0, 1);
  lcd.print("    Dihapus ");
  digitalWrite(blueLed, LED_OFF);   // Make sure blue LED is off
  digitalWrite(redLed, LED_OFF);  // Make sure red LED is off
  digitalWrite(greenLed, LED_OFF);  // Make sure green LED is off
  delay(200);
  digitalWrite(redLed, LED_ON);  // Make sure blue LED is on
  delay(200);
  digitalWrite(redLed, LED_OFF);   // Make sure blue LED is off
  delay(200);
  digitalWrite(redLed, LED_ON);  // Make sure blue LED is on
  delay(200);
  digitalWrite(redLed, LED_OFF);   // Make sure blue LED is off
  delay(200);
  digitalWrite(redLed, LED_ON);  // Make sure blue LED is on
  delay(200);
  delay(1000);
}

//////////////////////// Check readCard IF is masterCard   ///////////////////////////////////
// Check to see if the ID passed is the master programing card
boolean isMaster( byte test[] ) {
```

```
  if ( checkTwo( test, masterCard ) )
    return true;
  else
    return false;
}

bool monitorWipeButton(uint32_t interval) {
  uint32_t now = (uint32_t)millis();
  while ((uint32_t)millis() - now < interval)  {
    // check on every half a second
    if (((uint32_t)millis() % 500) == 0) {
      if (digitalRead(wipeB) != LOW)
        return false;
    }
  }
  return true;

}
```