LAMPIRAN

CODE PROGRAM KERJA ALAT

```
#include <Adafruit_Fingerprint.h>
#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal_I2C.h>
#include <ArduinoJson.h>
#include <CTBot.h>
#include <HardwareSerial.h>

String ssid = "security system";    //nama ssid wifi kalian
String pass = "system iot";  //password wifi kalian
String token = "6221755341:AAExPPGsICSI7ZgSH0HD7L_7kMkA-4sc1kE";
//token bot baru kalian
const int id = 6206773316;      //id telegram kalian

CTBot myBot;
TBMessage msg;
LiquidCrystal_I2C lcd(0x27, 16, 2);
int pb = 4;
#define RST_PIN 13
#define SDA_PIN 12
int u=0;
int relay = 33;
#define buzzer 2
#define exit 27
```

```
#if (defined(__AVR__) || defined(ESP8266)) &&
!defined(__AVR_ATmega2560__)
#else
#endif

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&Serial2);
MFRC522 mfrc522(SDA_PIN, RST_PIN);
MFRC522::MIFARE_Key key;
String strID;

unsigned long currentmillis = 0;
unsigned long millislcd = 0;
unsigned long lastTimeBotRan = 0;

void setup() {
  Serial.begin(9600);
  Serial2.begin(115200);
  pinMode(relay, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(pb, INPUT_PULLUP);
  lcd.begin();
  lcd.backlight();
  SPI.begin();
  mfrc522.PCD_Init();
  lcd.setCursor(0, 0);
  lcd.print(" System Starting ");
  digitalWrite(buzzer, LOW);
  digitalWrite(relay, LOW);
```

```
delay(1000);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("  Fingerprint   ");
lcd.setCursor(0,1);
lcd.print("    starting    ");

// set the data rate for the sensor serial port
finger.begin(57600);
delay(5);
if (finger.verifyPassword()) {
  Serial.println("Found fingerprint sensor!");
  lcd.setCursor(0,1);
  lcd.print("      READY     ");
  delay(500);
} else {
  Serial.println("Did not find fingerprint sensor :(");
  lcd.setCursor(0,1);
  lcd.print("    NOT FOUND!   ");
  while (1) { delay(1); }
}

Serial.println(F("Reading sensor parameters"));
finger.getParameters();
Serial.print(F("Status: 0x")); Serial.println(finger.status_reg, HEX);
Serial.print(F("Sys ID: 0x")); Serial.println(finger.system_id, HEX);
Serial.print(F("Capacity: ")); Serial.println(finger.capacity);
Serial.print(F("Security level: ")); Serial.println(finger.security_level);
```

```
  Serial.print(F("Device address: ")); Serial.println(finger.device_addr, HEX);
  Serial.print(F("Packet len: ")); Serial.println(finger.packet_len);
  Serial.print(F("Baud rate: ")); Serial.println(finger.baud_rate);


  finger.getTemplateCount();
  if (finger.templateCount == 0) {
    Serial.print("Sensor doesn't contain any fingerprint data. Please run the 'enroll' example.");
  }
  else {
    Serial.println("Waiting for valid finger...");
      Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println(" templates");
  }
  lcd.clear();


  Serial.println("Starting TelegramBot...");
  myBot.wifiConnect(ssid, pass);
  myBot.setTelegramToken(token);
  lcd.setCursor(0,0);
  lcd.print("starting TeleBot");
  if (myBot.testConnection()) {
    lcd.setCursor(0,1);
    lcd.print("   CONNECTED    ");
    Serial.println("Koneksi Bagus");
  } else {
    lcd.setCursor(0,1);
    lcd.print("  NOT CONNECTED ");
    Serial.println("Koneksi Jelek");
```

```
  }
   myBot.sendMessage(id, (String)"SECURITY SYSTEM IOT by DEDE ROMANSYAH"
       "\nKirimkan pesan >>> /bukapintu" +
       "\n \nUntuk membuka pintu dari jarak jauh", "");
}

void loop() {
yield();
  newmessage();
 if (digitalRead(pb) == LOW) {
     digitalWrite(relay, HIGH);
     delay(2000);
     delay(2000);
     delay(5000);
     digitalWrite(relay, LOW);
   }
   else {
     digitalWrite(relay, LOW);
   }

 //lcd.clear();
  // button();
  if (millis()-millislcd >= 150) {
    lcd.setCursor(0, 0);
    lcd.print("Scan kartu lalu ");
    lcd.setCursor(0, 1);
    lcd.print(" Scan jari anda ");
```

```
  millislcd = millis();
 }

 if (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial())
  return;

 Serial.print(F("PICC type: "));
 MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
 Serial.println(mfrc522.PICC_GetTypeName(piccType));

  strID = "";
 for (byte i = 0; i < mfrc522.uid.size; i++) {
  strID +=
    (mfrc522.uid.uidByte[i] < 0x10 ? "0" : "") +
    String(mfrc522.uid.uidByte[i], HEX) +
    (i != mfrc522.uid.size - 1 ? ":" : "");
 }
 Serial.println();
 Serial.print("Pesan : ");
 strID.toUpperCase();
 if (strID == "B3:63:45:0E" || strID == "90:5E:4B:11" || strID == "90:5E:4B:11"||
strID=="11:4D:88:4D" ) //penambahan kartu silakan isi tana kutip 2 yang kosong
 {
     myBot.sendMessage(id, "kartu dikenali, menunggu autentikasi kedua", "");
     lcd.setCursor(0,0);
     lcd.print("  Access Card   ");
     lcd.setCursor(1,1);
     lcd.print("    ACCEPTED    ");
```

```
  Serial.println("    ACCESS GRANTED    ");
  delay(1000);
     lcd.setCursor(0,0);
     lcd.print("  Finger  Scan  ");
     lcd.setCursor(1,1);
     lcd.print("                ");
  if (millis()-currentmillis >= 100) {
   getFingerprintID();
   currentmillis = millis();
  }
   delay(1000);
      lcd.setCursor(1,1);
      lcd.print("         ");
      myBot.sendMessage(id, "autentikasi kedua gagal", "");
  lcd.clear();
}
  else {
     myBot.sendMessage(id, "kartu tidak dikenali", "");
   Serial.println(" ACCES DENIED ");
   lcd.setCursor(0,0);
   lcd.print("   Access Card  ");
   lcd.setCursor(1,1);
   lcd.print("     DENIED     ");
   digitalWrite(buzzer, HIGH);
   delay(2000);
   lcd.setCursor(0,1);
   lcd.print("                ");
   lcd.setCursor(0,1);
```

```
      lcd.print(strID);
      digitalWrite(buzzer, LOW);
      delay(1000);
    }
}
uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println("No finger detected");
        lcd.setCursor(1,1);
        lcd.print("     GAGAL      ");
        myBot.sendMessage(id, "autentikasi kedua gagal, Finger tidak terdeteksi!",
"");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }
```

```
// OK success!
p = finger.image2Tz();
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
p = finger.fingerSearch();
if (p == FINGERPRINT_OK) {
  Serial.println("Found a print match!");
      myBot.sendMessage(id, "fingerprint diterima, membuka pintu", "");
```

```
    lcd.setCursor(0,0);
    lcd.print("  FINGERPRINT ");
    lcd.setCursor(1,1);
    lcd.print("    Accepted    ");
    delay(2000);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(" AKSES DITERIMA ");
    lcd.setCursor(1,1);
    lcd.print(" SILAKAN MASUK! ");
    digitalWrite(relay, HIGH);
    digitalWrite(buzzer, LOW);
    delay(10000);
    digitalWrite(relay, LOW);
    lcd.clear();
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_NOTFOUND) {
  Serial.println("Did not find a match");
    myBot.sendMessage(id, "fingerprint tidak dikenali", "");
    lcd.setCursor(0,0);
    lcd.print("  FINGERPRINT ");
    lcd.setCursor(1,1);
    lcd.print("    DENIED    ");
    digitalWrite(relay, LOW);
    digitalWrite(buzzer, HIGH);
    delay(2000);
```

```
      digitalWrite(buzzer, LOW);
      digitalWrite(relay, LOW);
      lcd.clear();
      delay(500);
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }
  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
  return finger.fingerID;
}
// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {
  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK)  return -1;
  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;
  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
  return finger.fingerID;
}
```

CODE PROGRAM PENAMBAHAN DATA RFID

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

#define SS_PIN 12     // coding esp32
#define RST_PIN 13    //coding esp32

// #define SS_PIN D4    // coding nodemcu
// #define RST_PIN D3   //coding nodemcu

// #define SS_PIN 10     // coding arduino uno
// #define RST_PIN 9    //coding arduino uno

MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class

MFRC522::MIFARE_Key key;

// Init array that will store new NUID
byte nuidPICC[4];

void setup() {
  Serial.begin(9600);
  SPI.begin(); // Init SPI bus
  rfid.PCD_Init(); // Init MFRC522

  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
```

```
  Serial.println(F("This code scan the MIFARE Classsic NUID."));
  Serial.print(F("Using the following key:"));
  printHex(key.keyByte, MFRC522::MF_KEY_SIZE);
}

void loop() {

  // Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.
  if ( ! rfid.PICC_IsNewCardPresent())
    return;

  // Verify if the NUID has been readed
  if ( ! rfid.PICC_ReadCardSerial())
    return;

  Serial.print(F("PICC type: "));
  MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
  Serial.println(rfid.PICC_GetTypeName(piccType));

  // Check is the PICC of Classic MIFARE type
  if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
    piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
    piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("Your tag is not of type MIFARE Classic."));
    return;
  }
```

```
  if (rfid.uid.uidByte[0] != nuidPICC[0] ||
    rfid.uid.uidByte[1] != nuidPICC[1] ||
    rfid.uid.uidByte[2] != nuidPICC[2] ||
    rfid.uid.uidByte[3] != nuidPICC[3] ) {
    Serial.println(F("A new card has been detected."));

    // Store NUID into nuidPICC array
    for (byte i = 0; i < 4; i++) {
      nuidPICC[i] = rfid.uid.uidByte[i];
    }

    Serial.println(F("The NUID tag is:"));
    Serial.print(F("In hex: "));
    printHex(rfid.uid.uidByte, rfid.uid.size);
    Serial.println();
    Serial.print(F("In dec: "));
    printDec(rfid.uid.uidByte, rfid.uid.size);
    Serial.println();
  }
  else Serial.println(F("Card read previously."));

  // Halt PICC
  rfid.PICC_HaltA();

  // Stop encryption on PCD
  rfid.PCD_StopCrypto1();
}
```

```
/**
 * Helper routine to dump a byte array as hex values to Serial.
 */
void printHex(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], HEX);
  }
}

/**
 * Helper routine to dump a byte array as dec values to Serial.
 */
void printDec(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], DEC);
  }
}
```

CODE PROGRAM PENAMBAHAN DATA FINGERPRINT

```
#include <Adafruit_Fingerprint.h>

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&Serial2);

uint8_t id;

void setup()
{
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nAdafruit Fingerprint sensor enrollment");
  // set the data rate for the sensor serial port
  finger.begin(57600);
  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1) { delay(1); }
  }
}

uint8_t readnumber(void) {
  uint8_t num = 0;
  while (num == 0) {
    while (! Serial.available());
    num = Serial.parseInt();
```

```
  }
  return num;
}
void loop()                     // run over and over again
{
  Serial.println("Ready to enroll a fingerprint!");
  Serial.println("Please type in the ID # (from 1 to 127) you want to save this finger as...");
  id = readnumber();
  if (id == 0) {// ID #0 not allowed, try again!
     return;
  }
  Serial.print("Enrolling ID #");
  Serial.println(id);
  while (!  getFingerprintEnroll() );
}
uint8_t getFingerprintEnroll() {
  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println(".");
      break;
```

```
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    break;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    break;
  default:
    Serial.println("Unknown error");
    break;
  }
}

// OK success!

p = finger.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
```

```
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

Serial.println("Remove finger");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
  p = finger.getImage();
}
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image taken");
    break;
  case FINGERPRINT_NOFINGER:
    Serial.print(".");
    break;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
```

```
    break;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    break;
  default:
    Serial.println("Unknown error");
    break;
  }
}

// OK success!

p = finger.image2Tz(2);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
```

```
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
Serial.print("Creating model for #");  Serial.println(id);

p = finger.createModel();
if (p == FINGERPRINT_OK) {
  Serial.println("Prints matched!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
  Serial.println("Fingerprints did not match");
  return p;
} else {
  Serial.println("Unknown error");
  return p;
}

Serial.print("ID "); Serial.println(id);
p = finger.storeModel(id);
if (p == FINGERPRINT_OK) {
  Serial.println("Stored!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
```

```
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not store in that location");
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }
}
```