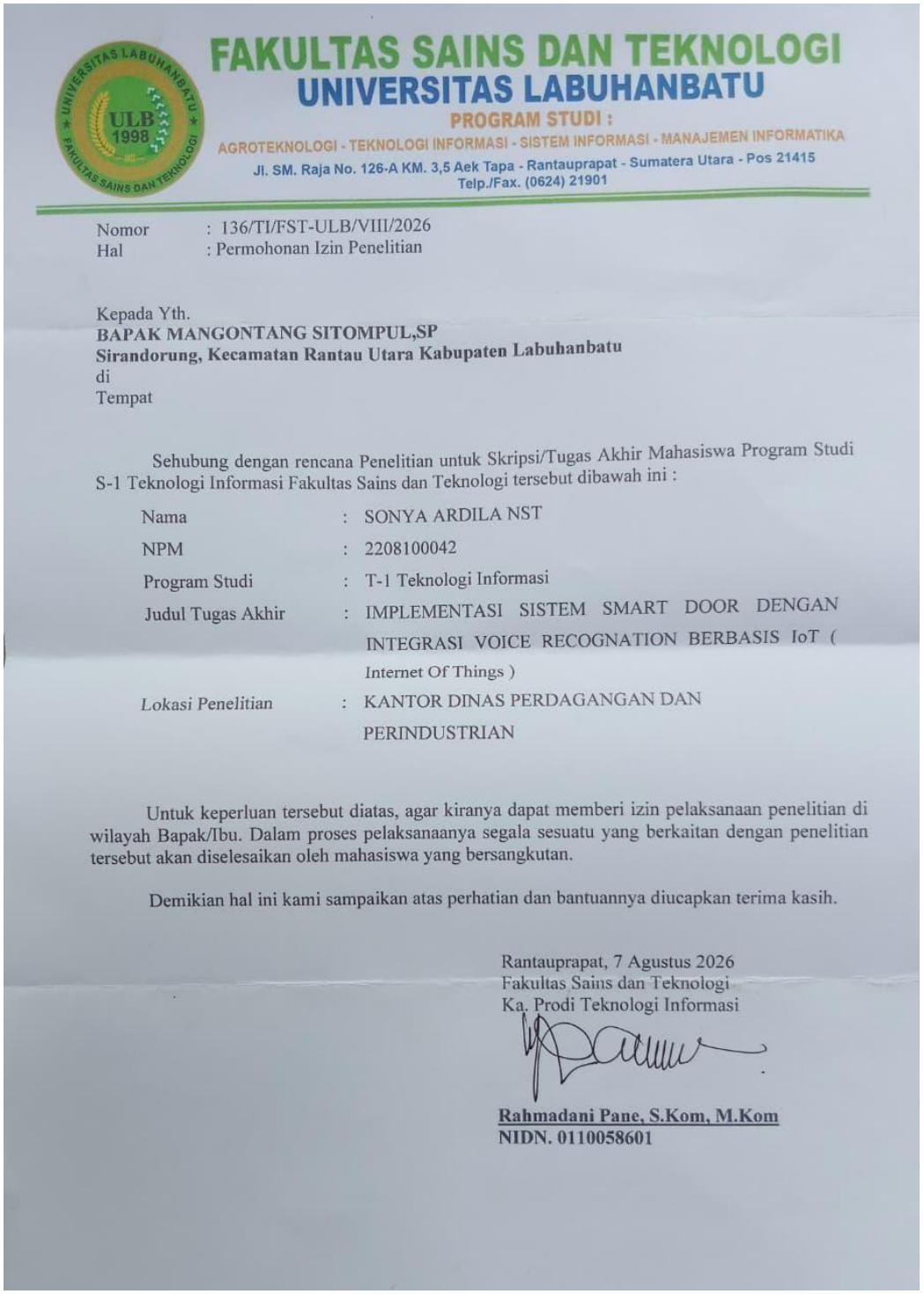
LAMPIRAN DOKUMENTASI

LAMPIRAN SURAT



FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS LABUHANBATU
PROGRAM STUDI :
AGROTEKNOLOGI - TEKNOLOGI INFORMASI - SISTEM INFORMASI - MANAJEMEN INFORMATIKA
Jl. SM. Raja No. 126-A KM. 3,5 Aek Tapa - Rantauprapat - Sumatera Utara - Pos 21415
Telp./Fax. (0624) 21901

Nomor : 136/TI/FST-ULB/VIII/2026
Hal : Permohonan Izin Penelitian

Kepada Yth.
BAPAK MANGONTANG SITOMPUL,SP
Sirandorung, Kecamatan Rantau Utara Kabupaten Labuhanbatu
di
Tempat

Sehubung dengan rencana Penelitian untuk Skripsi/Tugas Akhir Mahasiswa Program Studi S-1 Teknologi Informasi Fakultas Sains dan Teknologi tersebut dibawah ini :

Nama	:	SONYA ARDILA NST
NPM	:	2208100042
Program Studi	:	T-1 Teknologi Informasi
Judul Tugas Akhir	:	IMPLEMENTASI SISTEM SMART DOOR DENGAN INTEGRASI VOICE RECOGNATION BERBASIS IoT (*Internet Of Things*)
Lokasi Penelitian	:	KANTOR DINAS PERDAGANGAN DAN PERINDUSTRIAN

Untuk keperluan tersebut diatas, agar kiranya dapat memberi izin pelaksanaan penelitian di wilayah Bapak/Ibu. Dalam proses pelaksanaanya segala sesuatu yang berkaitan dengan penelitian tersebut akan diselesaikan oleh mahasiswa yang bersangkutan.

Demikian hal ini kami sampaikan atas perhatian dan bantuannya diucapkan terima kasih.

Rantauprapat, 7 Agustus 2026
Fakultas Sains dan Teknologi
Ka. Prodi Teknologi Informasi

Rahmadani Pane, S.Kom, M.Kom
NIDN. 0110058601

PEMERINTAH KABUPATEN LABUHANBATU
DINAS PERDAGANGAN DAN PERINDUSTRIAN
JALAN GELUGUR NO.18 TELP. (0624) 7671756 FAX (0624) 7671756
RANTAU PRAPAT

Rantauprapat, 12 Agustus 2026

Nomor : 510 / 498 /Dag.Ind/I/2026
Sifat : Biasa
Lampiran : --
Perihal : Izin Penelitian

Kepada Yth :
Ka. Prodi Teknologi Informasi
Fakultas Sains dan Teknologi ULB
di -
Tempat

Berdasarkan Surat dari Kepala Prodi Teknologi Informasi Fakultas Sains dan Teknologi Universitas Labuhanbatu Nomor : 136/TI/FST-ULB/VIII/2026 tanggal 07 Agustus 2026 dan Rekomendasi dari Badan Kesatuan Bangsa Dan Politik Nomor : 070/0444/BKBP-III/2026 tanggal 11 Agustus 2026 perihal Permohonan Izin Penelitian saudari yang tertera di bawah ini :

Nama : SONYA ARDILA NST
NPM : 2208100042
Program Studi : T-I Teknologi Informasi

Dengan ini menyetujui dan menerima saudari yang tertera tersebut untuk melaksanakan penelitian di instansi Dinas Perdagangan dan Perindustrian Kabupaten Labuhanbatu dengan judul penelitian : Implementasi Sistem Smart Door dengan Integrasi Voice Recognation Berbasis IoT (Internet Of Things). Setelah Melaksanakan penelitian agar melaporkan hasilnya kepada Kepala Dinas Perdagangan Dan Perindustrian Kabupaten Labuhanbatu sesuai dengan ketentuan yang berlaku.

Demikian disampaikan harap maklum.

An.Kepala Dinas,
Sekretaris

Hamdi Muhammad Sir, S.Kom, M.AP
Pembina
NIP. 198109112011011006

LAMPIRAN CODINGAN

```
/*
  ============================================================
  SMART DOOR - ESP32 Firmware (Full, Final)
  WiFi + MQTT + Relay + Sensor MC-38 + Voice recognition (dinamis)
  + LCD I2C + Buzzer + Menu Serial untuk tambah/hapus suara
  + Logic Relock Pintar (berdasarkan status fisik sensor pintu)
  ============================================================
  Library dibutuhkan:
    - PubSubClient (by Nick O'Leary) -> Library Manager
    - VoiceRecognitionV3_ESP32 (folder manual di Arduino/libraries)
    - LiquidCrystal_I2C (by Frank de Brabander)
    - Preferences (bawaan ESP32)
    - Wire (bawaan ESP32)
  ============================================================
  LOGIC RELOCK PINTAR:
  - Command OPEN diterima -> solenoid unlock (relay ON)
  - Kalau dalam 3 detik pintu TIDAK dibuka fisik (sensor tetap
    "tertutup") -> auto relock, dianggap tidak dipakai
  - Kalau pintu SEMPAT dibuka fisik (sensor deteksi "terbuka")
    -> solenoid TETAP unlock, menunggu sampai pintu ditutup lagi
    (sensor balik ke "tertutup") -> baru relay OFF (relock)
    Ini mencegah solenoid mencoba mengunci saat pintu masih
    fisik terbuka (yang bisa merusak mekanisme).
  ============================================================
*/

#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Preferences.h>
#include "VoiceRecognitionV3_ESP32.h"

// ==================== WIFI ====================
const char* WIFI_SSID     = "BURHAN";
const char* WIFI_PASSWORD = "1234554321";

// ==================== MQTT ====================
const char* MQTT_BROKER = "broker.hivemq.com";
const int   MQTT_PORT   = 1883;
const char* MQTT_CLIENT_ID = "smartdoor_esp32_sonya_01";

const char* TOPIC_CONTROL = "smartdoor_sonya/control";
```

```
const char* TOPIC_STATUS  = "smartdoor_sonya/status";
const char* TOPIC_ACCESS  = "smartdoor_sonya/access";

// ==================== PIN ======================
const int RELAY_PIN  = 26;
const int SENSOR_PIN = 27;
const int LED_PIN    = 2;
const int BUZZER_PIN = 25;

const bool RELAY_ACTIVE_LOW = true;

// Kalau command OPEN diterima tapi pintu TIDAK dibuka fisik dalam
// waktu ini, otomatis relock (dianggap tidak dipakai)
const unsigned long AUTO_RELOCK_JIKA_TIDAK_DIBUKA_MS = 3000; // 3
detik

// ==================== LCD I2C ====================
LiquidCrystal_I2C lcd(0x27, 20, 4); // Ganti 0x27 -> 0x3F kalau LCD
gak nyala

// ==================== VOICE RECOGNITION ==========
VR myVR(16, 17); // RX=16 (dari TXD modul lewat voltage divider),
TX=17
uint8_t vrBuf[64];

const int MAX_RECORDS = 10;
int openRecords[MAX_RECORDS];
int openRecordCount = 0;
int closeRecords[MAX_RECORDS];
int closeRecordCount = 0;

Preferences prefs;

// ==================== STATE GLOBAL ===============
WiFiClient   espClient;
PubSubClient mqttClient(espClient);

bool unlockedNow = false;
unsigned long unlockStartTime = 0;
bool pintuSudahDibukaFisik = false; // true kalau sensor sempat
deteksi pintu terbuka sejak command OPEN

String lastPublishedStatus = "";
String lastActionText = "Booting...";
```

```
unsigned long deniedMessageUntil = 0;
const unsigned long DENIED_MESSAGE_DURATION = 3000;

int stableSensorState = HIGH;
int lastSensorReading  = HIGH;
unsigned long lastDebounceTime = 0;
const unsigned long DEBOUNCE_DELAY = 200;

unsigned long lastLcdUpdate = 0;
const unsigned long LCD_UPDATE_INTERVAL = 1000;

String serialBuffer = "";

// ==================== BUZZER ============================
void beepPendek() {
  digitalWrite(BUZZER_PIN, HIGH);
  delay(150);
  digitalWrite(BUZZER_PIN, LOW);
}

void beepDitolak() {
  for (int i = 0; i < 2; i++) {
    digitalWrite(BUZZER_PIN, HIGH);
    delay(150);
    digitalWrite(BUZZER_PIN, LOW);
    delay(150);
  }
}

void beepAlarm() {
  for (int i = 0; i < 5; i++) {
    digitalWrite(BUZZER_PIN, HIGH);
    delay(200);
    digitalWrite(BUZZER_PIN, LOW);
    delay(150);
  }
}

void beepSukses() {
  digitalWrite(BUZZER_PIN, HIGH);
  delay(80);
  digitalWrite(BUZZER_PIN, LOW);
  delay(80);
  digitalWrite(BUZZER_PIN, HIGH);
  delay(80);
```

```
  digitalWrite(BUZZER_PIN, LOW);
}

// ==================== RELAY ============================
void relayOn() { digitalWrite(RELAY_PIN, RELAY_ACTIVE_LOW ? LOW :
HIGH); }
void relayOff() { digitalWrite(RELAY_PIN, RELAY_ACTIVE_LOW ? HIGH :
LOW); }

void bukaPintu(String sumber) {
  Serial.println("[AKSI] Membuka pintu (relay ON)...");
  relayOn();
  unlockedNow = true;
  unlockStartTime = millis();
  pintuSudahDibukaFisik = false; // reset, karena ini command baru
  digitalWrite(LED_PIN, HIGH);
  lastActionText = "Buka: " + sumber;
  beepPendek();
}

void kunciPintu(String sumber) {
  Serial.println("[AKSI] Mengunci pintu (relay OFF)...");
  relayOff();
  unlockedNow = false;
  pintuSudahDibukaFisik = false;
  digitalWrite(LED_PIN, LOW);
  lastActionText = "Kunci: " + sumber;
  beepPendek();
}

void aksesDitolak(String alasan) {
  Serial.println("[VOICE] Akses ditolak: " + alasan);
  lastActionText = "AKSES DITOLAK";
  deniedMessageUntil = millis() + DENIED_MESSAGE_DURATION;
  beepDitolak();

  if (mqttClient.connected()) {
    mqttClient.publish(TOPIC_ACCESS, "VOICE_DENIED");
  }
}

// ==================== LOGIC RELOCK PINTAR
=========================
// Dipanggil terus-menerus di loop(). Menentukan kapan solenoid
// boleh relock berdasarkan status fisik sensor pintu, BUKAN
```

```
// cuma berdasarkan timer tetap.
void cekLogicRelock() {
  if (!unlockedNow) return; // sedang terkunci, tidak perlu dicek

  bool pintuFisikTerbuka = (stableSensorState == HIGH);

  if (!pintuSudahDibukaFisik) {
    if (pintuFisikTerbuka) {
      // Baru terdeteksi terbuka -> tandai, nanti tunggu sampai
ditutup
      pintuSudahDibukaFisik = true;
      Serial.println("[SENSOR] Pintu terdeteksi dibuka fisik,
menunggu ditutup untuk relock.");
    } else if (millis() - unlockStartTime >=
AUTO_RELOCK_JIKA_TIDAK_DIBUKA_MS) {
      // Sudah lewat batas waktu tapi pintu gak pernah dibuka ->
anggap gak dipakai
      Serial.println("[AKSI] Pintu tidak dibuka dalam waktu yang
ditentukan, relock otomatis.");
      kunciPintu("Auto-relock (tidak dibuka)");
    }
  } else {
    // Pintu sudah sempat dibuka fisik, tunggu sampai tertutup lagi
    if (!pintuFisikTerbuka) {
      Serial.println("[AKSI] Pintu terdeteksi tertutup kembali,
mengunci otomatis.");
      kunciPintu("Auto-relock (pintu ditutup)");
    }
  }
}

// ==================== PENYIMPANAN PERMANEN (Preferences/NVS)
==========
void simpanRecordsKePreferences() {
  prefs.begin("smartdoor", false);

  prefs.putInt("openCount", openRecordCount);
  for (int i = 0; i < openRecordCount; i++) {
    String key = "open" + String(i);
    prefs.putInt(key.c_str(), openRecords[i]);
  }

  prefs.putInt("closeCount", closeRecordCount);
  for (int i = 0; i < closeRecordCount; i++) {
    String key = "close" + String(i);
```

```
    prefs.putInt(key.c_str(), closeRecords[i]);
  }

  prefs.end();
  Serial.println("[STORAGE] Daftar record disimpan permanen.");
}

void muatRecordsDariPreferences() {
  prefs.begin("smartdoor", true);

  openRecordCount = prefs.getInt("openCount", -1);
  closeRecordCount = prefs.getInt("closeCount", -1);

  if (openRecordCount < 0 || closeRecordCount < 0) {
    prefs.end();
    Serial.println("[STORAGE] Belum ada data tersimpan, pakai
default.");
    openRecordCount = 2;
    openRecords[0] = 0; // "buka"
    openRecords[1] = 2; // "buka pintu"

    closeRecordCount = 2;
    closeRecords[0] = 1; // "kunci"
    closeRecords[1] = 3; // "kunci pint"

    simpanRecordsKePreferences();
    return;
  }

  for (int i = 0; i < openRecordCount; i++) {
    String key = "open" + String(i);
    openRecords[i] = prefs.getInt(key.c_str(), -1);
  }
  for (int i = 0; i < closeRecordCount; i++) {
    String key = "close" + String(i);
    closeRecords[i] = prefs.getInt(key.c_str(), -1);
  }

  prefs.end();
  Serial.println("[STORAGE] Daftar record dimuat dari
penyimpanan.");
}

// ==================== HELPER CEK RECORD ==================
bool adaDiArray(int arr[], int count, int value) {
```

```
  for (int i = 0; i < count; i++) {
    if (arr[i] == value) return true;
  }
  return false;
}

void reloadVoiceRecords() {
  uint8_t allRecords[MAX_RECORDS * 2];
  int total = 0;

  for (int i = 0; i < openRecordCount; i++) allRecords[total++] =
(uint8_t)openRecords[i];
  for (int i = 0; i < closeRecordCount; i++) allRecords[total++] =
(uint8_t)closeRecords[i];

  if (total == 0) {
    Serial.println("[VOICE] Tidak ada record untuk di-load.");
    return;
  }

  if (myVR.load(allRecords, total) >= 0) {
    Serial.print("[VOICE] ");
    Serial.print(total);
    Serial.println(" record berhasil di-load.");
  } else {
    Serial.println("[VOICE] Gagal load record!");
  }
}

// ==================== WIFI ====================================
void connectWiFi() {
  Serial.print("[WIFI] Menghubungkan ke ");
Serial.println(WIFI_SSID);
  lcd.setCursor(0, 0);
  lcd.print("Connecting WiFi... ");

  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) { delay(500);
Serial.print("."); }
  Serial.println();
  Serial.print("[WIFI] Terhubung! IP: ");
Serial.println(WiFi.localIP());
}
```

```
// ==================== MQTT CALLBACK ============================
void mqttCallback(char* topic, byte* payload, unsigned int length) {
  String message;
  for (unsigned int i = 0; i < length; i++) message +=
(char)payload[i];

  Serial.print("[MQTT] Pesan dari '"); Serial.print(topic);
Serial.print("': "); Serial.println(message);

  if (String(topic) == TOPIC_CONTROL) {
    if (message == "OPEN") bukaPintu("Whatsapp");
    else if (message == "CLOSE") kunciPintu("Whatsapp");
  }
}

// ==================== MQTT CONNECT =====================
void connectMQTT() {
  while (!mqttClient.connected()) {
    Serial.print("[MQTT] Menghubungkan...");
    if (mqttClient.connect(MQTT_CLIENT_ID)) {
      Serial.println(" berhasil!");
      mqttClient.subscribe(TOPIC_CONTROL);
      publishCurrentStatus(true);
    } else {
      Serial.print(" gagal, rc="); Serial.print(mqttClient.state());
      Serial.println(" coba lagi 5 detik..."); delay(5000);
    }
  }
}

// ==================== SENSOR MC-38 =========================
void bacaSensorDanPublish() {
  int reading = digitalRead(SENSOR_PIN);
  if (reading != lastSensorReading) lastDebounceTime = millis();

  if ((millis() - lastDebounceTime) > DEBOUNCE_DELAY) {
    if (reading != stableSensorState) {
      stableSensorState = reading;
      publishCurrentStatus(false);

      // Alarm cuma bunyi kalau pintu kebuka TANPA sedang dalam mode
unlock
      if (stableSensorState == HIGH && !unlockedNow) {
        beepAlarm();
      }
```

```
    }
  }
  lastSensorReading = reading;
}

void publishCurrentStatus(bool forcePublish) {
  String statusText = (stableSensorState == LOW) ? "DOOR_CLOSED" :
"DOOR_OPEN";
  if (statusText != lastPublishedStatus || forcePublish) {
    mqttClient.publish(TOPIC_STATUS, statusText.c_str());
    Serial.print("[MQTT] Publish status: ");
Serial.println(statusText);
    lastPublishedStatus = statusText;
  }
}

// ==================== VOICE RECOGNITION HANDLER ================
void cekVoiceCommand() {
  int ret = myVR.recognize(vrBuf, 50);

  if (ret > 0) {
    int recordId = vrBuf[1];
    Serial.print("[VOICE] Record terdeteksi: ");
    Serial.println(recordId);

    if (adaDiArray(openRecords, openRecordCount, recordId)) {
      bukaPintu("Voice");
      mqttClient.publish(TOPIC_ACCESS, "VOICE_OPEN");
    } else if (adaDiArray(closeRecords, closeRecordCount, recordId))
{
      kunciPintu("Voice");
      mqttClient.publish(TOPIC_ACCESS, "VOICE_CLOSE");
    } else {
      aksesDitolak("Record tidak dikenal: " + String(recordId));
    }
  }
}

// ==================== LCD DISPLAY
==================================
void updateLCD() {
  if (millis() - lastLcdUpdate < LCD_UPDATE_INTERVAL) return;
  lastLcdUpdate = millis();
```

```
  String statusPintu = (stableSensorState == LOW) ? "TERKUNCI" :
"TERBUKA ";
  String statusWifi   = (WiFi.status() == WL_CONNECTED) ? "OK" :
"PUTUS";

  lcd.setCursor(0, 0);
  lcd.print("SMART DOOR SYSTEM   ");

  lcd.setCursor(0, 1);
  lcd.print("Status: " + statusPintu + "      ");

  lcd.setCursor(0, 2);
  lcd.print("WiFi: " + statusWifi + "          ");

  if (deniedMessageUntil > 0 && millis() > deniedMessageUntil) {
    lastActionText = "System Ready";
    deniedMessageUntil = 0;
  }

  lcd.setCursor(0, 3);
  String tampil = lastActionText;
  if (tampil.length() > 20) tampil = tampil.substring(0, 20);
  while (tampil.length() < 20) tampil += " ";
  lcd.print(tampil);
}

// ==================== MENU SERIAL UNTUK TAMBAH/HAPUS SUARA
==================
void printMenuSerial() {
  Serial.println(F("======================================="));
  Serial.println(F("  MENU TRAINING SUARA (via Serial)"));
  Serial.println(F("======================================="));
  Serial.println(F("menu                          - tampilkan menu
ini"));
  Serial.println(F("list                          - lihat semua record
aktif"));
  Serial.println(F("trainopen <id> <kata>       - training kata baru
untuk BUKA"));
  Serial.println(F("trainclose <id> <kata>      - training kata baru
untuk KUNCI"));
  Serial.println(F("hapus <id>                  - hapus record dari
daftar aktif"));
  Serial.println(F("======================================="));
  Serial.println(F("Contoh: trainopen 4 halo pintu"));
```

```
  Serial.println(F("Catatan: id harus unik (0-30), belum dipakai
record lain."));
  Serial.println(F("========================================"));
}

void printListRecords() {
  Serial.println(F("--- Daftar Record OPEN ---"));
  for (int i = 0; i < openRecordCount; i++) {
    Serial.print("  Record ");
    Serial.println(openRecords[i]);
  }
  Serial.println(F("--- Daftar Record CLOSE ---"));
  for (int i = 0; i < closeRecordCount; i++) {
    Serial.print("  Record ");
    Serial.println(closeRecords[i]);
  }
}

void trainDanDaftarkan(int recordId, String signature, bool isOpen)
{
  if (recordId < 0 || recordId > 30) {
    Serial.println("[ERROR] ID record harus antara 0-30.");
    return;
  }

  if (adaDiArray(openRecords, openRecordCount, recordId) ||
      adaDiArray(closeRecords, closeRecordCount, recordId)) {
    Serial.println("[ERROR] ID record ini sudah dipakai. Pilih ID
lain atau 'hapus' dulu.");
    return;
  }

  Serial.print("[TRAINING] Mulai training record ");
  Serial.print(recordId);
  Serial.print(" dengan kata '");
  Serial.print(signature);
  Serial.println("'. Bersiap ucapkan kata itu saat diminta...");

  uint8_t trainBuf[64];
  char sigChar[32];
  signature.toCharArray(sigChar, 32);

  int ret = myVR.trainWithSignature((uint8_t)recordId,
(uint8_t*)sigChar, signature.length(), trainBuf);
```

```
  if (ret < 0) {
    Serial.println("[TRAINING] Gagal atau timeout. Coba lagi dengan
command yang sama.");
    return;
  }

  Serial.println("[TRAINING] Berhasil! Mendaftarkan ke
kategori...");

  if (isOpen) {
    if (openRecordCount < MAX_RECORDS) {
      openRecords[openRecordCount++] = recordId;
    } else {
      Serial.println("[ERROR] Daftar OPEN sudah penuh (maks 10).");
      return;
    }
  } else {
    if (closeRecordCount < MAX_RECORDS) {
      closeRecords[closeRecordCount++] = recordId;
    } else {
      Serial.println("[ERROR] Daftar CLOSE sudah penuh (maks 10).");
      return;
    }
  }

  simpanRecordsKePreferences();
  reloadVoiceRecords();
  beepSukses();

  Serial.print("[SUKSES] Record ");
  Serial.print(recordId);
  Serial.print(" ('");
  Serial.print(signature);
  Serial.print("') terdaftar sebagai ");
  Serial.println(isOpen ? "OPEN" : "CLOSE");
}

void hapusRecord(int recordId) {
  bool ditemukan = false;

  for (int i = 0; i < openRecordCount; i++) {
    if (openRecords[i] == recordId) {
      for (int j = i; j < openRecordCount - 1; j++) openRecords[j] =
openRecords[j + 1];
      openRecordCount--;
```

```
      ditemukan = true;
      break;
    }
  }

  if (!ditemukan) {
    for (int i = 0; i < closeRecordCount; i++) {
      if (closeRecords[i] == recordId) {
        for (int j = i; j < closeRecordCount - 1; j++)
closeRecords[j] = closeRecords[j + 1];
        closeRecordCount--;
        ditemukan = true;
        break;
      }
    }
  }

  if (ditemukan) {
    simpanRecordsKePreferences();
    reloadVoiceRecords();
    Serial.print("[SUKSES] Record ");
    Serial.print(recordId);
    Serial.println(" dihapus dari daftar aktif.");
  } else {
    Serial.println("[ERROR] Record ID tidak ditemukan di daftar
aktif.");
  }
}

void prosesSerialCommand(String line) {
  line.trim();
  if (line.length() == 0) return;

  int spaceIdx = line.indexOf(' ');
  String cmd = (spaceIdx == -1) ? line : line.substring(0,
spaceIdx);
  cmd.toLowerCase();

  if (cmd == "menu") {
    printMenuSerial();
  } else if (cmd == "list") {
    printListRecords();
  } else if (cmd == "trainopen" || cmd == "trainclose") {
    String rest = line.substring(spaceIdx + 1);
    rest.trim();
```

```
    int spaceIdx2 = rest.indexOf(' ');
    if (spaceIdx2 == -1) {
      Serial.println("[ERROR] Format salah. Contoh: trainopen 4 halo
pintu");
      return;
    }
    int recordId = rest.substring(0, spaceIdx2).toInt();
    String signature = rest.substring(spaceIdx2 + 1);
    trainDanDaftarkan(recordId, signature, cmd == "trainopen");
  } else if (cmd == "hapus") {
    String idStr = line.substring(spaceIdx + 1);
    idStr.trim();
    hapusRecord(idStr.toInt());
  } else {
    Serial.println("[ERROR] Command tidak dikenali. Ketik 'menu'
untuk bantuan.");
  }
}

void cekSerialInput() {
  while (Serial.available() > 0) {
    char c = Serial.read();
    if (c == '\n') {
      prosesSerialCommand(serialBuffer);
      serialBuffer = "";
    } else if (c != '\r') {
      serialBuffer += c;
    }
  }
}

// ==================== SETUP ====================================
void setup() {
  Serial.begin(115200);
  delay(500);

  pinMode(RELAY_PIN, OUTPUT);
  pinMode(SENSOR_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);

  Wire.begin(21, 22);
  lcd.init();
  lcd.backlight();
```

```
  lcd.setCursor(0, 0);
  lcd.print("SMART DOOR SYSTEM");
  lcd.setCursor(0, 1);
  lcd.print("Booting...");

  kunciPintu("Boot");

  connectWiFi();
  mqttClient.setServer(MQTT_BROKER, MQTT_PORT);
  mqttClient.setCallback(mqttCallback);

  stableSensorState = digitalRead(SENSOR_PIN);
  lastSensorReading  = stableSensorState;

  myVR.begin(9600);
  Serial.println("[VOICE] Modul VR3 diinisialisasi.");

  muatRecordsDariPreferences();
  reloadVoiceRecords();

  lastActionText = "System Ready";

  printMenuSerial();
}

// ==================== LOOP
=========================================
void loop() {
  if (WiFi.status() != WL_CONNECTED) connectWiFi();
  if (!mqttClient.connected()) connectMQTT();
  mqttClient.loop();

  cekLogicRelock();

  bacaSensorDanPublish();
  cekVoiceCommand();
  updateLCD();
  cekSerialInput();

  delay(20);
}
```