# DAFTAR PUSTAKA


[1] Ginting, Hidayat Depema. 2013. "RANCANG BANGUN PERANGKAT PEMANTAU SHELTER BTS." Jurnal Teknik Komputer Unikom – Komputika – Volume 2, No.2 - 2013 10-14.

[2] www.baktikominfo.id, 22 Mei 2019, Pengertian, Macam dan Komponen pada Tower BTS yang Sebaiknya Anda Tahu, 7 Juli 2024, https://www.baktikominfo.id/id/detail-berita/pengertian,-macam-dan-komponen-pada-tower-bts-yang-sebaiknya-anda-tahu.

[3] Rachmadi, Tri, and S. Kom. Mengenal apa itu internet of things. Vol. 1. Tiga Ebook, 2020.

[4] E. A. Prasty. (2019, Juli 24). Arsitektur dan Fitur ESP32 (Module ESP32) IoT [online]. Available:https://www.edukasielektronika.com/2019/07/arsitektur-dan-fitur-esp32-module-esp32.html

[5] Marzuki, Imam. "Perancangan dan pembuatan sistem penyalaan lampu otomatis dalam ruangan berbasis Arduino menggunakan sensor gerak dan sensor cahaya." Jurnal Intake: Jurnal Penelitian Ilmu Teknik Dan Terapan 10.1 (2019): 9-16.

[6] Ardiansyah, M., Febryan, A., Adriani, A., & Rahmania, R. (2023). Rancang Bangun Sistem Keamanan Rumah Berbasis Telegram Menggunakan ESP 32 CAM. Vertex Elektro, 15(1), 64-71.

[7] Muzaki, A. S., Saptadi, A. H., & Pamungkas, W. (2011). Aplikasi Sensor Cahaya Untuk Alarm Anti Pencuri. Jurnal Infotel, 3(2), 50-59.

[8] Wahyudi, R., & Edidas, E. (2022). Perancang dan Pembuatan Sistem Keamanan Rumah Berbasis Internet of Things Menggunakan ESP32-CAMERA. Jurnal Pendidikan Tambusai, 6(1), 1135-1141.

[9] Utami, D. P., & Subali, S. (2014). Miniatur Pengaman Tower Terhadap Pencuri Berbasis Mikrokontroler Atmega8 Menggunakan Sensor PIR (Passive Infra Red) dan Limit Switch dengan Sistem Scada. Gema Teknologi, 17(4). [10]

[10] Ardiansyah, M., Febryan, A., Adriani, A., & Rahmania, R. (2023). Rancang Bangun Sistem Keamanan Rumah Berbasis Telegram Menggunakan ESP 32 CAM. Vertex Elektro, 15(1), 64-71.

[11] Ipanhar, A., Wijaya, T. K., & Gunoto, P. (2022). Perancangan Sistem Monitoring Pintu Otomatis Berbasis IoT Menggunakan ESP32-CAMERA. Sigma Teknika, 5(2), 333-350.

[12] Hutauruk, S., Pangaribuan, T., & Sinaga, J. H. (2020). Rekayasa Sistem Data Logger Temperature Berbasis Arduino Uno R3. Jurnal ELPOTECS, 3(2), 15-21.

[13] Suhendi, H., & Sofyan, I. (2022). SISTEM KEAMANAN RUMAH MENGGUNAKAN RFID, SENSOR PIR DAN MODUL GSM BERBASIS MIKROKONTROLER ATMEGA328. Journal of Innovation Research and Knowledge, 2(7), 2989-3000.

[14] Aulia, G. F. S., & Arissantoso, K. (2024). RANCANG BANGUN SISTEM KEAMANAN KENDARAAN RODA EMPAT (MOBIL) BERBASIS THINGSBOARD DENGAN MIKROKONTROLER ESP32. Jurnal SISKOM-KB (Sistem Komputer dan Kecerdasan Buatan), 7(2), 175-181.

[15] Kadir, A. (2023). Membuat Aplikasi Web dengan PHP dan *Database* MySQL. Penerbit Andi.

[16] Tyas, U. M., & Buckhari, A. A. (2023). IMPLEMENTASI APLIKASI ARDUINO IDE PADA MATA KULIAH SISTEM DIGITAL. TEKNOS: Jurnal Pendidikan dan Teknologi, 1(1), 1-9.

[17] www.arduinoindonesia.id 20 Juli 2019, Memulai Pemrograman ESP32 menggunakan Arduino IDE, 8 Juli 2024, https://www.arduinoindonesia.id/2019/07/memulai-pemrograman-esp32-menggunakan.html

[18] www.kompas.com 17 Juni 2023, Belasan Baterai Tower BTS Diembat Pencuri, Provider Telekomunikasi Rugi Puluhan Juta Rupiah, 10 Juli 2024. https://yogyakarta.kompas.com/read/2023/06/17/115708578/belasan-baterai-tower-bts-diembat-pencuri-provider-telekomunikasi-rugi.html

[19] Erma Standsyah, R., & Restu NS, I. S. (2017). Implementasi phpmyadmin pada rancangan sistem pengadministrasian. *Unisda Journal of Mathematics and Computer*, *3*(2), 39-44.

# LAMPIRAN

## LAMPIRAN 1 Kode Program ESP32-Utama

```
// Nama file terakhir: FINALISASI_WROOM_100H_2254_01_[13 FEB 2025 22544]_OK_BACKUP
// Update terakhir: [13 FEB 2025 22:54]
// Kontributor: [A.Amrulloh]

//Library yang digunakan
#include <WiFi.h>
#include <WebSocketsServer.h>
#include <WebSocketsClient.h>
#include <SD.h>  // Pastikan SD card sudah terinisialisasi
#include <SPI.h>
#include <Wire.h>
#include <WiFiUdp.h>
#include <HardwareSerial.h>
#include <DFRobotDFPlayerMini.h>
#include <TinyGPS++.h>
#include <time.h>
#include <Bounce2.h>
#include <WiFiClient.h>
#include <HTTPClient.h>
#include <WiFiClientSecure.h>
#include <FS.h>

// Definisi Pin
const int SIREN_RELAY_Pin = 26;
const int LIMIT_SW_Pin = 35;
const int MAGNET_SW_Pin = 36;
const int LED_PIN = 2;
const int TX_DF_PLAYER_Pin = 16;
const int RX_DF_PLAYER_Pin = 17;
const int BUSY_DF_PLAYER_Pin = 25;
const int SD_CS_PIN = 5; // Konfigurasi SD Logger
const int RX_GPS_Pin = 22; // default
const int TX_GPS_Pin = 21; // default
const int SIGNAL_LASER_Pin = 13;
const int SIGNAL_LDR_Pin = 33;
const int PIR_SENSOR_Pin = 32;
// Delay debounce dalam milidetik def :50 >> 1000ms 1x trigger alarm.
const unsigned long debounceDelay = 1000;

// #define ldrThd1 1200
#define ldrThd2 3000

// BluetoothSerial SerialBT;     // Membuat objek Serial Bluetooth
DFRobotDFPlayerMini dfPlayer; // Membuat Objek DFPlayer dan GPS
TinyGPSPlus gps;              // Membuat Objek GPS
Bounce debouncer = Bounce();  // Membuat instance debouncer untuk sensor Limit

// Variabel untuk menyimpan data GPS terakhir yang valid
float lastGpsLat = 0.0;  // Latitude terakhir
float lastGpsLon = 0.0;  // Longitude terakhir
float currentLat = 0.0;  // Variabel Latitude saat ini
float currentLon = 0.0;  // Variabel Longitude saat ini
unsigned long lastGpsUpdate = 0;
const unsigned long gpsUpdateInterval = 60000; // 1 menit dalam milidetik

// Variabel status alarm
// Flag untuk alarm
bool alarmActive = false;
// Flag untuk sedang menerima gambar dari eSp32-CAM
bool isReceivingFile = false;
// Flag tambahan untuk menandai bahwa file sudah lengkap diterima
bool fileComplete = false;
// Flag untuk mengecek status inisialisasi
bool dfPlayerInitialized = false;
// Flag untuk ditulis ke Logs
bool gpsPrinted = false;
// Flag untuk init GPS
```

```
68| bool gpsInitiated = false;
69| // Flag untuk nama file yang dikirimkan ESP32-CAM
70| String incomingFileName = "";
71| // data biner berupa gambar yang dikirim dari cam ke wroom
72| File incomingFile;
73| // Flag untuk cek berapa besar file gambar yang dikirimkan esp32-cam
74| unsigned long totalBytesReceived = 0;
75| // Variabel baru untuk menyimpan ukuran file yang dikirim
76| unsigned long expectedFileSize = 0;
77| // Reset Flag untuk kondisi Magnet Switch terbuka --> HIGH
78| bool magnetHigh = false;
79| // Reset Flag untuk kondisi Limit Switch terbuka -- HIGH
80| bool limitHigh = false;
81| // Flag untuk sensor PIR ketika terdeteksi gerakan
82| bool pirDetected = false;
83| // Flag untuk sensor LDR ketika sinar laser terhalang
84| bool ldrHigh = false;
85| // Flag untuk nilai LDR untuk di compare dengan ldrThd2
86| int ldrValue = analogRead(SIGNAL_LDR_Pin);
87| //→ Modul DFPlayer tidak memutar file audio idle (HIGH pada pin BUSY)
88| bool mp3Play = false;
89| // Deklarasi Mode
90| enum Mode { OPERATIONAL, MAINTENANCE };
91| // Default ke MAINTENANCE Mode
92| Mode currentMode = MAINTENANCE;
93| // Default ke Operational Mode
94| // Mode currentMode = OPERATIONAL;
95| // Deklarasi Connenction
96| enum Connection {internetWifi, internetSIM900A};
97| // Deklarasi variabel connectionType Variabel untuk tipe koneksi aktif
98| Connection connectionType = internetWifi;
99| // Flag untuk memastikan WebSocket server hanya dimulai sekali
100| bool serverInitialized = false;
101| // Flag untuk mendeteksi disconnect
102| bool wasDisconnected = false;
103| // Flag untuk mencegah reconnect saat pertama kali
104| bool setupComplete = false;
105| // Deklarasi objek logFile
106| // ini mengacu pada file logger txt --> simonbts_logs.txt
107| File logFile;
108| // mendefinisikan nama file log --> simonbts_logs.txt
109| String logFileName;
110| // site_id yang digunakan untuk membedakan lokasi fisik BTS
111| const char* site_id = "X001_CISAUK_AMELIA";
112|
113| // Konfigurasi WiFi
114| // const char* ssid = "NC_AQM12";
115| // const char* password = "MN200808";
116| const char* ssid = "Python 12";
117| const char* password = "Dimxp1223?";
118| WebSocketsServer webSocketServer = WebSocketsServer(81);
119|
120| // UART bawaan ESP32
121| HardwareSerial dfPlayerSerial(1);
122| HardwareSerial gpsSerial(2);
123|
124| // Fungsi untuk cek mode dari server
125| // Variabel untuk menyimpan mode sebelumnya
126| String lastMode = "";
127|
128| // Deklarasi fungsi yang digunakan
129| void handleWebSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t
length);
130| void updateLogFileName();
131| void initTime();
132| void initSensors();
133| void inisialisasiMicroSD();
134| void captureSaveImage();
135| String getTimeStamp();
136| void alarmTriggered();
137| void resetFlagNotifikasi();
138| void cekStatusVariable();
```

```
139| void webSocketInit();
140| void resetActuators();
141| void triggerActuators();
142| void resetFlags();
143| void triggerFlags();
144| void playResetNotif();
145| void playAlertNotif();
146| // Deklarasi fungsi yang digunakan
147| void changeMode(Mode mode);
148| void handleAlarmActions();
149| void logAlarmEvent();
150| void sendAlarmParameters();
151| void connectionCheck();
152| // Fungsi tambahan optimasi GPS
153| void initGPS();
154| void updateGPS();
155| void checkGPSQuality();
156| void checkModeFromServer();
157|
158| //=========================space================
159|
160|     // Fungsi validasi PIR untuk menghindari false trigger
161| bool isPIRStable() {
162|     int count = 0;
163|     for (int i = 0; i < 10; i++) {  // Cek 3x dalam 500ms
164|         if (digitalRead(PIR_SENSOR_Pin) == HIGH) count++;
165|         // delay(100); //default_Val
166|         delay(50); // trial_val
167|     }
168|     return (count >= 8);  // Minimal 8 dari 10 pembacaan harus HIGH
169| }
170|
171|
172| // Fungsi untuk cek mode dari server
173| void checkModeFromServer() {
174|     if (WiFi.status() == WL_CONNECTED) {
175|         HTTPClient http;
176|               String  url  =  "http://www.simonbts.my.id/get_mode.php?site_id="  +
String(site_id) + "&t=" + String(millis());
177|
178|         http.begin(url);  // Memulai koneksi HTTP
179|
180|         int httpResponseCode = http.GET();
181|         if (httpResponseCode == 200) {
182|             String newMode = http.getString();
183|             newMode.trim(); // Menghapus spasi ekstra atau karakter tak perlu
184|
185|             if (newMode != lastMode) {
186|                 Serial.println("[INFO]  Mode diterima dari server: " + newMode);
187|                 lastMode = newMode; // Update mode terakhir
188|             }
189|
190|             if (newMode == "operation" && currentMode != OPERATIONAL) {
191|                 changeMode(OPERATIONAL);
192|                 dfPlayer.playMp3Folder(79);
193|             } else if (newMode == "maintenance" && currentMode != MAINTENANCE) {
194|                 changeMode(MAINTENANCE);
195|                 dfPlayer.playMp3Folder(65);
196|             }
197|         } else {
198|             Serial.println("[ERROR] Gagal mengambil mode dari server. HTTP Code: " +
String(httpResponseCode));
199|         }
200|         http.end();
201|     } else {
202|         Serial.println("[ERROR] Tidak terhubung ke WiFi!");
203|     }
204| }
205|
206|
207| // Implementasi fungsi tambahan untuk GPS
208| void initGPS() {
```

```
209|     gpsSerial.begin(9600, SERIAL_8N1, RX_GPS_Pin, TX_GPS_Pin);
210|     Serial.println("[DEBUG] Menunggu GPS warm-up (120 detik)...");
211|
212|     unsigned long startTime = millis();
213|     while (millis() - startTime < 120000) { //default 60s //sementara 120 detik
214|         if (gpsSerial.available()) {
215|             gps.encode(gpsSerial.read());
216|             if (gps.location.isUpdated()) {
217|                 Serial.println("[DEBUG] GPS Ready. Lokasi awal ditemukan.");
218|                 break;
219|             }
220|         }
221|         delay(100);  // Cek GPS setiap 100 ms untuk efisiensi
222|     }
223|     if (!gps.location.isValid()) {
224|         Serial.println("[ERROR] GPS gagal inisialisasi dalam 120 detik. Melanjutkan
dengan data sebelumnya.");
225|     }
226| }
227|
228| void updateGPS() {
229|     if (millis() - lastGpsUpdate >= gpsUpdateInterval) {
230|         lastGpsUpdate = millis();  // Update waktu terakhir pembaruan GPS
231|         if (gpsSerial.available() > 10) {  // Baca hanya jika buffer GPS cukup penuh
232|             while (gpsSerial.available() > 0) {
233|                 gps.encode(gpsSerial.read());
234|             }
235|             if (gps.location.isValid()) {
236|                 currentLat = gps.location.lat();
237|                 currentLon = gps.location.lng();
238|                 lastGpsLat = currentLat;
239|                 lastGpsLon = currentLon;
240|                 Serial.println("[INFO]  GPS diperbarui:");
241|                 Serial.println("[INFO]  Latitude : " + String(currentLat, 7));
242|                 Serial.println("[INFO]  Longitude: " + String(currentLon, 7));
243|             } else {
244|                    Serial.println("[WARN]   Data GPS tidak valid. Menggunakan data
terakhir:");
245|                 Serial.println("[INFO]  Last Latitude : " + String(lastGpsLat, 7));
246|                 Serial.println("[INFO]  Last Longitude: " + String(lastGpsLon, 7));
247|             }
248|         } else {
249|             Serial.println("[INFO]  Tidak ada data baru dari GPS.");
250|         }
251|     }
252| }
253|
254| void checkGPSQuality() {
255|     if (gps.hdop.isValid()) {
256|         int hdop = gps.hdop.value();
257|         Serial.println("[DEBUG] HDOP: " + String(hdop) + " (Semakin kecil, semakin
baik)");
258|         if (hdop > 200) {
259|             // Serial.println("[WARN] HDOP tinggi. Akurasi GPS dapat rendah.");
260|             Serial.println("[WARN] HDOP masih tinggi. Akurasi GPS dapat rendah.");
261|         } else {
262|             Serial.println("[WARN] HDOP sudah bagus. Akurasi GPS dapat tinggi.");
263|         }
264|     } else {
265|         Serial.println("[WARN] HDOP tidak tersedia.");
266|     }
267| }
268|
269|
270| void connectionCheck() {
271|     // Periksa apakah ESP32 terhubung ke WiFi
272|     if (WiFi.status() == WL_CONNECTED) {
273|         connectionType = internetWifi;
274|         // Serial.println("[INFO] Connection type set to WiFi");
275|         return;  // Keluar dari fungsi jika sudah terhubung
276|     }
277|
```

```
278|     // Jika tidak terhubung, lakukan percobaan untuk koneksi WiFi
279|     int maxRetries = 10;
280|     int retryCount = 0;
281|     while (WiFi.status() != WL_CONNECTED && retryCount < maxRetries) {
282|         delay(2000);
283|         retryCount++;
284|         Serial.print(".");
285|     }
286|
287|     // Jika WiFi masih tidak terhubung, beralih ke koneksi SIM900A
288|     if (WiFi.status() != WL_CONNECTED) {
289|         connectionType = internetSIM900A;
290|         // Serial.println("[INFO] Connection type set to SIM900A");
291|         // Tambahkan logika aktivasi SIM900A di sini, misalnya SIM900A.connect();
292|     } else {
293|         Serial.println("[DEBUG] Connected to WiFi");
294|     }
295| }
296|
297|
298| // Kirim data alarm ke ESP32-CAM
299| void sendAlarmParameters() {
300|     String alarmData = "PARAMS:";  // Awali data dengan "PARAMS:" sebagai penanda
301|     alarmData += "site_id=" + String(site_id) + ";";
302|      alarmData  +=  "mode="  +  String(currentMode  ==  OPERATIONAL  ?  "Operational"  :
"Maintenance") + ";";
303|        alarmData   +=   "connection="   +   String(connectionType   ==   internetWifi   ?
"internetWifi" : "internetSIM900A") + ";";
304|     alarmData += "magnet_s=" + String(magnetHigh) + ";";
305|     alarmData += "limit_s=" + String(limitHigh) + ";";
306|     alarmData += "pir_s=" + String(pirDetected) + ";";
307|     alarmData += "ldr_s=" + String(ldrHigh) + ";";
308|     alarmData += "gps_lat=" + String(currentLat, 7) + ";";
309|     alarmData += "gps_lon=" + String(currentLon, 7) + ";";
310|
311|     // Kirim data parameter alarm ke ESP32-CAM melalui WebSocket
312|     webSocketServer.broadcastTXT(alarmData);
313|     Serial.println("[INFO]  Mengirim parameter alarm ke ESP32-CAM melalui WebSocket");
314|     Serial.println("[INFO]  Param_Dikirim :" + alarmData); // coba_
315|
316| }
317|
318|
319| void logAlarmEvent() {
320|     if (logFile) {
321|         logFile.print(getTimeStamp() + "- Mode= " + String(currentMode == OPERATIONAL
? "Operational" : "Maintenance") +
322|                       ", Alarm terpicu. Magnet: " + (digitalRead(MAGNET_SW_Pin) ==
HIGH ? "HIGH" : "LOW") +
323|                      ", Limit Switch: " + (debouncer.read() == HIGH ? "HIGH" : "LOW")
+
324|                      ", PIR Sensor: " + (digitalRead(PIR_SENSOR_Pin) == HIGH ? "HIGH"
: "LOW") +
325|                       ", LDR_Stats: " + String(ldrHigh) +
326|                       ", LDR_Val: " + String(analogRead(SIGNAL_LDR_Pin) + "."));
327|
328|         if (gps.location.isValid()) {
329|             logFile.print(", Latitude: " + String(currentLat, 7) +
330|                           ", Longitude: " + String(currentLon, 7));
331|         } else {
332|             logFile.print(", Latitude: " + String(lastGpsLat, 7) +
333|                           ", Longitude: " + String(lastGpsLon, 7)); // Jika lokasi
tidak valid, gunakan Longitude dan Latitude yang terakhir
334|             Serial.println("[ERROR] Data GPS menggunakan data terakhir.");
335|         }
336|         logFile.println(".");
337|         logFile.flush();
338|         Serial.println("[INFO]  Event diupdate ke logfile simonbts_logs.txt");
339|     }
340| }
341|
342|
```

```
343| void handleAlarmActions() {
344|     if (magnetHigh || limitHigh || pirDetected || ldrHigh) {
345|         // Alarm akan dipicu ketika ada sensor yang aktif
346|         Serial.println("[INFO]  TimeStamp " + getTimeStamp());
347|
348|         Serial.println("[DEBUG] Alarm terpicu, sirene aktif...");
349|         triggerFlags();
350|
351|         Serial.println("[DEBUG] ON-kan Actuator");
352|         triggerActuators();
353|
354|         Serial.println("[DEBUG] play Alert Notif (01)...");
355|         playAlertNotif();
356|
357|     }
358| }
359|
360|
361| // Fungsi untuk mengubah mode --> Mode Operational dan Maintenance ditambahkan
362| void changeMode(Mode mode) {
363|     currentMode = mode;
364|     if (mode == OPERATIONAL) {
365|         // 100H_1800_07
366|         resetFlags();
367|         resetActuators();
368|         delay(1000);
369|         Serial.println("[INFO]  Mode diubah ke: OPERATIONAL");
370|         dfPlayer.playMp3Folder(97);
371|         // Memutar file 0097.mp3 untuk notifikasi "Mode diubah ke: OPERATION"
372|         delay(2000);
373|     } else if (mode == MAINTENANCE) {
374|         Serial.println("[INFO]  Mode diubah ke: MAINTENANCE");
375|         resetFlags();
376|         resetActuators();
377|         delay(1000);
378|         dfPlayer.playMp3Folder(96);
379|         // Memutar file 0096.mp3 untuk notifikasi "Mode diubah ke: MAINTENANCE"
380|         delay(2000);
381|     }
382| }
383|
384| void webSocketInit() {
385|   // Inisialisasi WebSocket Server
386|   webSocketServer.begin();
387|   // Menggunakan handleWebSocketEvent
388|   webSocketServer.onEvent(handleWebSocketEvent);
389|   Serial.println("[INFO]  WebSocket Server started.");
390|    Serial.println("[INFO]   WebSocket Server (ESP32-Utama) dimulai pada ws://" +
WiFi.localIP().toString() + ":81");
391| }
392|
393| // Definisi fungsi
394| void updateLogFileName() {
395|   logFileName = "/simonbts_logs.txt"; // Definisikan nama file log event
396| }
397|
398| void cekStatusVariable() {
399|   Serial.print("[DEBUG] S_Magnet: ");
400|   Serial.print(magnetHigh);
401|   Serial.print(" | S_LimitSw: ");
402|   Serial.print(limitHigh);
403|   Serial.print(" | S_PIR: ");
404|   Serial.print(pirDetected);
405|   Serial.print(" | S_LDR_Sts: ");
406|   Serial.print(ldrHigh);
407|   Serial.print(" | S_LDR_Val: ");
408|   Serial.println(String(analogRead(SIGNAL_LDR_Pin)));
409|   Serial.print("[DEBUG] S_Alarm: ");
410|   Serial.print(alarmActive);
411|   Serial.print(" | S_Trm_Pic4Cam: ");
412|   Serial.print(isReceivingFile);
413|   Serial.print(" | S_Trf_Pic2UT: ");
```

```
414|   Serial.print(fileComplete);
415|   Serial.print(" | S_PlayMp3: ");
416|   Serial.print(mp3Play);
417|   Serial.println(".");
418|
419|   // Tambahkan debug di sini XLIM
420|   Serial.print("[DEBUG] Limit Switch Status (Debouncer): ");
421|   Serial.println(debouncer.read() == HIGH ? "HIGH" : "LOW");
422| }
423|
424| void resetActuators() {
425|   digitalWrite(SIREN_RELAY_Pin, LOW);
426|   digitalWrite(LED_PIN, LOW);
427|   Serial.println("[INFO]  Actuators (sirene dan LED) berhasil direset.");
428| }
429|
430| void triggerActuators() {
431|   digitalWrite(SIREN_RELAY_Pin, HIGH);
432|   digitalWrite(LED_PIN, HIGH);
433|   Serial.println("[INFO]  Actuators sirene dan LED berhasil di trigger.");
434| }
435|
436| void resetFlags() {
437|   alarmActive = false;
438|   isReceivingFile = false;
439|   fileComplete = false; // cek kacau
440|   mp3Play = false;
441|   ldrHigh = false;
442|   pirDetected = false;
443|   magnetHigh = false;
444|   limitHigh = false;
445|   Serial.println("[INFO]  Status flags berhasil direset.");
446| }
447|
448| void triggerFlags() {
449|   alarmActive = true;
450|   isReceivingFile = false;
451|   mp3Play = false;
452|   ldrHigh = false;
453|   pirDetected = false;
454|   Serial.println("[INFO]  Status flags berhasil di trigger.");
455| }
456|
457| void playResetNotif() {
458|   if (dfPlayerInitialized && !alarmActive && !mp3Play) {
459|     dfPlayer.playMp3Folder(37);
460|     Serial.println("[DEBUG] Memutar file 0037.mp3 sebagai notifikasi alarm reset.");
461|     mp3Play = true;
462|
463|     unsigned long startTime = millis();
464|     while (mp3Play) {
465|       if (!mp3Play || millis() - startTime > 3000) {
466|         mp3Play = false;
467|       }
468|       delay(50);
469|     }
470|     Serial.println("[INFO]  File 0037.mp3 selesai diputar.");
471|   }
472| }
473|
474| void playAlertNotif() {
475|   if (dfPlayerInitialized && alarmActive && !mp3Play) {
476|     dfPlayer.playMp3Folder(1);
477|     Serial.println("[DEBUG] Memutar file 0001.mp3 sebagai notifikasi alert.");
478|     mp3Play = true;
479|
480|     unsigned long startTime = millis();
481|     while (mp3Play) {
482|       if (!mp3Play || millis() - startTime > 7000) {
483|         mp3Play = false;
484|       }
485|       delay(50);
```

```
486|     }
487|     Serial.println("[INFO]  File 0001.mp3 selesai diputar.");
488|   }
489| }
490|
491| void initTime() {
492|     const   char*   ntpServers[]   =   {"time.cloudflare.com",   "asia.pool.ntp.org",
"time.google.com"};
493|   const int maxRetries = 30;  // Batas maksimum percobaan sinkronisasi value_Default
= 10
494|   int retryCount = 0;
495|   bool timeSynced = false;
496|
497|   configTime(25200, 0, ntpServers[0], ntpServers[1], ntpServers[2]); // GMT+7
498|
499|   struct tm timeinfo;
500|   unsigned long startMillis = millis();
501|
502|   while (!timeSynced && retryCount < maxRetries) {
503|     if (getLocalTime(&timeinfo)) {
504|       Serial.println("Waktu berhasil disinkronisasi:");
505|       Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
506|       timeSynced = true;
507|     } else {
508|       Serial.println("[ERROR] Gagal sinkronisasi waktu, mencoba lagi...");
509|       retryCount++;
510|       delay(1000);
511|     }
512|
513|     // Jika waktu tunggu terlalu lama, keluar dari loop
514|     if (millis() - startMillis > 10000) {  // Timeout setelah 10 detik
515|       Serial.println("[ERROR] Timeout: Gagal sinkronisasi waktu setelah 30 detik.");
516|       break;
517|     }
518|   }
519|
520|   if (!timeSynced) {
521|     Serial.println("[ERROR] Gagal sinkronisasi waktu setelah beberapa percobaan.");
522|   }
523| }
524|
525| void initSensors() {
526|   pinMode(LIMIT_SW_Pin, INPUT);
527|   pinMode(MAGNET_SW_Pin, INPUT);
528|   pinMode(BUSY_DF_PLAYER_Pin, INPUT);
529|   pinMode(SIREN_RELAY_Pin, OUTPUT);
530|   pinMode(LED_PIN, OUTPUT);
531|    // Inisialisasi pin untuk sensor PIR dan LDR+Laser
532|   pinMode(SIGNAL_LASER_Pin, OUTPUT); // Laser sebagai output
533|   pinMode(SIGNAL_LDR_Pin, INPUT); // LDR sebagai input
534|   pinMode(PIR_SENSOR_Pin, INPUT); // PIR sebagai input
535|
536|   Serial.println("[DEBUG] All module and sensors initialized.");
537| }
538| // Inisialisasi SD card
539| void inisialisasiMicroSD() {
540|   if (!SD.begin(SD_CS_PIN)) {
541|     Serial.println("[ERROR] Gagal inisialisasi MicroSD.");
542|   } else {
543|     Serial.println("[INFO]  MicroSD berhasil di inisialisasi.");
544|   }
545| }
546|
547| void captureSaveImage() {
548|   // Kirim parameter alarm sebelum meminta ESP32-CAM menangkap gambar
549|   sendAlarmParameters();
550|   // Beri waktu untuk penerimaan parameter
551|   delay(20);
552|   if (webSocketServer.connectedClients() > 0) {
553|     webSocketServer.broadcastTXT("CAPTURE_IMAGE");
554|     // Memberi waktu untuk ESP32-CAM menangkap gambar --> Default 100 Trial 50
555|     delay(100);
```

```
556|     Serial.println("[DEBUG] Capture and Send request sent to ESP32-CAM.");
557|     isReceivingFile = true; // Menandakan bahwa sistem sedang menerima file gambar
558|     Serial.println("[DEBUG] Image capture and send in process.");
559|   } else {
560|      Serial.println("[ERROR] WebSocket not connected, failed to capture and send
image.");
561|     webSocketInit(); // Inisialisasi ulang WebSocket jika tidak terhubung
562|   }
563| }
564|
565| String getTimeStamp() {
566|   struct tm timeinfo;
567|   if (!getLocalTime(&timeinfo)) {
568|     Serial.println("[ERROR] Failed to obtain time");
569|     return "00000000_000000";
570|   }
571|   char timestamp[20];
572|   strftime(timestamp, sizeof(timestamp), "%Y%m%d_%H%M%S", &timeinfo);
573|   return String(timestamp);
574| }
575|
576| //void alarmTriggered//
577|
578| void alarmTriggered() {
579|     if (currentMode == OPERATIONAL) {   // Cek jika dalam mode OPERASIONAL
580|         handleAlarmActions();  // Panggil fungsi untuk menghandle aksi alarm
581|         logAlarmEvent();  // Panggil fungsi untuk menuliskan log
582|     } else {  // Jika mode MAINTENANCE
583|           Serial.println("[INFO] Sistem dalam mode MAINTENANCE, alarm tidak akan
dipicu.");
584|     }
585| }
586|
587|
588| //void resetFlagNotifikasi//
589| // Perbarui `resetFlagNotifikasi` agar sesuai dengan mode
590| void resetFlagNotifikasi() {
591|     if (currentMode == OPERATIONAL && !alarmActive && !isReceivingFile && !mp3Play)
{
592|             Serial.println("[DEBUG] Resetting Flag dan Notifikasi dalam mode
OPERASIONAL...");
593|
594|     // Memecah menjadi tiga fungsi
595|     Serial.println("[DEBUG] Reset aktuator...");
596|     resetActuators();
597|     Serial.println("[DEBUG] Reset Flags...");
598|     resetFlags();
599|     Serial.println("[DEBUG] Play Reset Notif (37)...");
600|     playResetNotif();
601|     } else if (currentMode == MAINTENANCE) {
602|             Serial.println("[INFO] Sistem dalam mode MAINTENANCE, reset tidak
diperlukan.");
603|     }
604| }
605|
606|
607| void setup() {
608|   Serial.begin(115200);
609|
610|   // Inisialisasi DFPlayer
611|   dfPlayerSerial.begin(9600, SERIAL_8N1, TX_DF_PLAYER_Pin, RX_DF_PLAYER_Pin);
612|   for (int i = 0; i < 5; i++) {
613|     if (dfPlayer.begin(dfPlayerSerial)) {
614|       dfPlayerInitialized = true;
615|       Serial.println("[INFO]  DFPlayer Mini siap.");
616|       dfPlayer.volume(26); // vol_default vol_df
617|       dfPlayer.EQ(DFPLAYER_EQ_NORMAL);
618|       dfPlayer.outputDevice(DFPLAYER_DEVICE_SD);
619|       break;
620|     } else {
621|       Serial.println("[ERROR] Gagal menginisialisasi DFPlayer, mencoba lagi...");
622|       delay(1000); //--> Default 1000 Trial 500
```

```
623|     }
624|   }
625|
626|   if (!dfPlayerInitialized) {
627|     Serial.println("[ERROR] DFPlayer Mini gagal diinisialisasi setelah 5 percobaan.
Melanjutkan sistem tanpa DFPlayer.");
628|   }
629|
630|   Serial.println("///////-----Selamat Datang di SiMon SisKa BTS-----///////");
631|   // Memutar file 0081.mp3 untuk notifikasi "Selamat Datang di SiMon BTS"
632|   dfPlayer.playMp3Folder(81);
633|   // delay audio 81 +  Delay singkat untuk stabilisasi Serial
634|   delay (3000);
635|   mp3Play = false;
636|   delay(1000);
637|
638|     // Koneksi ke WiFi
639|   WiFi.setHostname("ESP32_WROOM");  // Set nama perangkat
640|   WiFi.mode(WIFI_STA);
641|   WiFi.begin(ssid, password, false);
642|   int maxRetries = 30;
643|   int retryCount = 0;
644|   while (WiFi.status() != WL_CONNECTED && retryCount < maxRetries) {
645|       delay(2000);
646|       retryCount++;
647|       Serial.print(".");
648|   }
649|   if (retryCount >= maxRetries) {
650|       Serial.println("[ERROR] Gagal terhubung ke WiFi --> Reset.");
651|       // Untuk menghentikan WiFi secara aman sebelum reset
652|       WiFi.disconnect(true);
653|       delay(2000);
654|       ESP.restart();
655|   } else {
656|       Serial.println("[DEBUG] Connected to WiFi");
657|   }
658|
659|   // Inisialisasi WebSocket Server
660|   webSocketInit();
661|
662|   if (!serverInitialized) {
663|         webSocketServer.begin(); // Memulai server WebSocket
664|         webSocketServer.onEvent(handleWebSocketEvent);  // Menetapkan handler event
665|         serverInitialized = true; // Set flag setelah server dimulai
666|         Serial.println("[INFO]  WebSocket Server started.");
667|     }
668|
669|   // Inisialisasi MicroSD
670|   inisialisasiMicroSD();
671|
672|   updateLogFileName(); // Panggil fungsi untuk mengupdate nama file log
673|
674|   // Modifikasi untuk memeriksa keberadaan file sebelum membuka
675|   if (!SD.exists(logFileName)) {
676|       // Buat file baru jika belum ada
677|       logFile = SD.open(logFileName.c_str(), FILE_WRITE);
678|       if (logFile) {
679|           Serial.println("[INFO]  File logger baru dibuat: " + logFileName);
680|           logFile.println("=== LOG FILE STARTED ==="); // Header awal file baru
681|           logFile.flush(); // Pastikan data langsung ditulis ke microSD
682|       }
683|   } else {
684|       // Buka dalam mode APPEND jika sudah ada
685|       logFile = SD.open(logFileName.c_str(), FILE_APPEND);
686|   }
687|
688|   if (!logFile) {
689|       Serial.println("[ERROR] Gagal membuka atau membuat file " + logFileName);
690|       Serial.println("[ERROR] Sistem akan dilanjutkan tanpa logger");
691|   } else {
692|       Serial.println("[INFO]  File logger " + logFileName + " siap digunakan.");
693|   }
```

```
694|
695|   // Inisialisasi NTP
696|   initTime();
697|
698|   // Inisialisasi Sensor
699|   initSensors();
700|
701|   // Inisialisasi debouncer untuk Limit Switch
702|   debouncer.attach(LIMIT_SW_Pin);
703|   debouncer.interval(debounceDelay);
704|
705|   // PIR off (kondisi HIGH) setelah sistem boot
706|   digitalWrite(PIR_SENSOR_Pin, LOW);
707|
708|
709|
710|   // Laser on (kondisi HIGH) setelah sistem boot
711|   digitalWrite(SIGNAL_LASER_Pin, HIGH);
712|
713|   // Inisialisasi Sirine dan LED
714|   digitalWrite(SIREN_RELAY_Pin, LOW);
715|   digitalWrite(LED_PIN, LOW);
716|   cekStatusVariable();
717|
718|   // Inisialisasi mode awal
719|     changeMode(currentMode);
720|     // Memutar file 0091.mp3 untuk notifikasi "Mode awal diatur"
721|     dfPlayer.playMp3Folder(91);
722|     delay (2000);
723|     mp3Play = false;
724|
725|   initGPS();  // Inisialisasi GPS
726|
727|   // Nonaktifkan reconnect otomatis
728|   setupComplete = true;  // Tandai setup selesai setelah inisialisasi
729|   // Jangan sambungkan langsung WebSocket pada startup
730|     delay(1000);  // Delay untuk stabilisasi --> Default 1000 Trial 500
731|   Serial.println("[INFO]  Inisialisasi selesai, SiMon Ready...");
732|    dfPlayer.playMp3Folder(92);   // Memutar file 0092.mp3 untuk notifikasi "Sistem
siap"
733|   delay (2000);
734|   mp3Play = false;
735|
736| }
737|
738|
739|
740| void loop() {
741|
742| // Cek mode dari server setiap 10 detik
743|     static unsigned long lastCheckTime = 0;
744|     unsigned long currentTime = millis();
745|
746|     if (currentTime - lastCheckTime > 5000) { // 10 detik default
747|         lastCheckTime = currentTime;
748|         checkModeFromServer();
749|     }
750|
751| connectionCheck();  // Periksa koneksi
752|
753|
754|  // Cek apakah ada data yang tersedia dari Serial Monitor
755| if (Serial.available()) {
756|   String command = Serial.readStringUntil('\n'); // Membaca perintah hingga newline
757|   command.trim();  // Menghilangkan spasi di awal dan akhir
758|
759|   // Periksa apakah perintahnya untuk mengganti mode
760|             if            (command.equalsIgnoreCase("mode_operation")            ||
command.equalsIgnoreCase("mode_o")) {
761|     Serial.println("Diterima: " + command);
762|     changeMode(OPERATIONAL);
763|     Serial.println("Mode berubah ke operational.");
```

```
764|          }     else     if     (command.equalsIgnoreCase("mode_maintenance")     ||
command.equalsIgnoreCase("mode_m")) {
765|     Serial.println("Diterima: " + command);
766|     changeMode(MAINTENANCE);
767|     Serial.println("Mode berubah ke maintenance.");
768|             }      else      if      (command.equalsIgnoreCase("reset_sys")      ||
command.equalsIgnoreCase("reset")) {
769|     Serial.println("Diterima: " + command);
770|     Serial.println("[INFO]  Sistem akan di-reset.");
771|     dfPlayer.playMp3Folder(95);  // Memutar file 0095.mp3 untuk notifikasi "Sistem
akan restart"
772|     delay(2000);
773|     WiFi.disconnect(true);  // Untuk menghentikan WiFi secara aman sebelum reset
774|     for (int i = 5; i > 0; i--) {
775|       Serial.print("Hitung mundur: ");
776|       Serial.print(i);
777|       Serial.println(" detik");
778|       delay(1000);
779|     }
780|     ESP.restart(); // Lebih baik menggunakan restart dibanding reset
781|             }      else      if      (command.equalsIgnoreCase("last_alarm")      ||
command.equalsIgnoreCase("alarm_last")) {
782|     Serial.println("Diterima: " + command);
783|     Serial.println("Cek Status Alarm dan Flags.");
784|     cekStatusVariable();
785|             }      else      if      (command.equalsIgnoreCase("reset_flag")      ||
command.equalsIgnoreCase("resetflag")) {
786|     Serial.println("Diterima: " + command);
787|     Serial.println("Reset Flags.");
788|     resetFlags(); // ini di trigger dari serial monitor
789|   } else if (command.equalsIgnoreCase("vol_28") || command.equalsIgnoreCase("VOL24"))
{
790|     Serial.println("Diterima: " + command);
791|     dfPlayer.volume(28);
792|     Serial.println("Volume DF Player --> 28.");
793|   } else if (command.equalsIgnoreCase("vol_24") || command.equalsIgnoreCase("VOL24"))
{
794|     Serial.println("Diterima: " + command);
795|     dfPlayer.volume(24);
796|     Serial.println("Volume DF Player --> 24.");
797|   } else if (command.equalsIgnoreCase("vol_22") || command.equalsIgnoreCase("VOL22"))
{
798|     Serial.println("Diterima: " + command);
799|     dfPlayer.volume(22);
800|     Serial.println("Volume DF Player --> 22.");
801|   } else if (command.equalsIgnoreCase("vol_20") || command.equalsIgnoreCase("VOL20"))
{
802|     Serial.println("Diterima: " + command);
803|     dfPlayer.volume(20);
804|     Serial.println("Volume DF Player --> 20.");
805|   } else if (command.equalsIgnoreCase("vol_16") || command.equalsIgnoreCase("VOL16"))
{
806|     Serial.println("Diterima: " + command);
807|     dfPlayer.volume(16);
808|     Serial.println("Volume DF Player --> 16.");
809|                   }      else      if      (command.equalsIgnoreCase("mute")      ||
command.equalsIgnoreCase("S_mute")) {
810|     resetActuators();
811|     // digitalWrite(SIREN_RELAY_Pin, LOW);
812|     Serial.println("Siren: " + command);
813|
814|               }       else       if       (command.equalsIgnoreCase("GPS_Q")       ||
command.equalsIgnoreCase("GPS_Quality")) {
815|     checkGPSQuality();
816|    } else if (command.equalsIgnoreCase("mode") || command.equalsIgnoreCase("cmode")
|| command.equalsIgnoreCase("cmod")) {
817|     if (currentMode == OPERATIONAL) {
818|       Serial.println("Mode: Operational");
819|     } else if (currentMode == MAINTENANCE) {
820|       Serial.println("Mode: Maintenance");
821|     }
822|   } else {
```

```
823|     Serial.println("Diterima: " + command);
824|     Serial.println("[WARN] Perintah tidak dikenali: " + command);
825|   }
826| }
827|
828|   // Update debouncer untuk Limit Switch
829|   debouncer.update();
830|
831|   // Baca sensor PIR dan LDR + Laser
832|   // pirDetected = digitalRead(PIR_SENSOR_Pin) == HIGH;
833|     pirDetected = isPIRStable();
834|   int ldrValue = analogRead(SIGNAL_LDR_Pin);
835|   ldrHigh = ldrValue > ldrThd2;
836|
837|   // Membaca status sensor Magnetik dan Limit Switch
838|   magnetHigh = digitalRead(MAGNET_SW_Pin) == HIGH;
839|   limitHigh = debouncer.read() == HIGH;
840|
841|   mp3Play = digitalRead(BUSY_DF_PLAYER_Pin) == LOW;
842|
843|   //KONDISI TRIGGER AWAL
844|    // Trigger alarm jika salah satu sensor aktif, dan kondisi variable lainya -->
false
845|    if ((magnetHigh || limitHigh || pirDetected || ldrHigh) && !alarmActive &&
!isReceivingFile && !mp3Play && currentMode == OPERATIONAL) {
846|     cekStatusVariable();
847|
848|     // Tambahkan debug khusus untuk Limit Switch
849|     if (limitHigh) {
850|         Serial.println("[DEBUG] Limit Switch terpicu! Memicu alarm...");
851|     }
852|     if (magnetHigh) {
853|         Serial.println("[DEBUG] Magnetik Switch terpicu! Memicu alarm...");
854|     }
855|     if (pirDetected) {
856|         Serial.println("[DEBUG] PIR Sensor mendeteksi gerakan! Memicu alarm...");
857|     }
858|     if (ldrHigh) {
859|         Serial.println("[DEBUG] Laser/LDR terhalang! Memicu alarm...");
860|     }
861|
862|
863|     Serial.println("[INFO]  Akses ilegal terdeteksi.");
864|      Serial.println("[DEBUG] Prepare kirim gambar ke Server simon bts dari ESP32-
CAM.");
865|      captureSaveImage(); // Perintah mengambil dan menyimpan gambar ke ESP32-CAM
setelah alarm terpicu
866|     cekStatusVariable();
867|     Serial.println("[DEBUG] Panggil fungsi alarmTriggered().");
868|     alarmTriggered();  // Memanggil fungsi untuk memicu alarm jika salah satu sensor
aktif
869|     // Serial.println("[DEBUG] Keluar dari kondisi trigger awal"); // disable_print
870|     cekStatusVariable();
871|
872|     // Trigger alarm jika salah satu sensor masih aktif,
873|    } else if ((magnetHigh || limitHigh || pirDetected || ldrHigh) && alarmActive &&
!isReceivingFile && !mp3Play && currentMode == OPERATIONAL) {
874|       // Logika untuk tetap menjalankan sistem karena sensor masih HIGH
875|     cekStatusVariable();
876|     Serial.println("[DEBUG] Prepare kirim gambar ke Server dari ESP32-Utama.");
877|     captureSaveImage(); // Mengambil dan Menyimpan gambar setelah alarm terpicu
878|     cekStatusVariable();
879|     Serial.println("[DEBUG] Panggil fungsi alarmTriggered().");
880|     alarmTriggered();   // Memanggil fungsi untuk memicu alarm jika salah satu sensor
aktif
881|     cekStatusVariable();
882|
883|
884|    } else if ((!magnetHigh && !limitHigh && !pirDetected && !ldrHigh) && alarmActive
&& !isReceivingFile && !mp3Play && currentMode == OPERATIONAL) {
885|       // Logika untuk menghentikan sistem jika semua sensor sudah LOW
886|       cekStatusVariable();
```

```
887|       resetFlags();     // ini di trigger jika tidak ada alarm lagi--> clear alarm
888|       resetActuators(); // Tambahkan fungsi untuk menghentikan alarm jika perlu
889|       playResetNotif();
890|       Serial.println("[INFO]  Tidak ada sensor yang aktif.");
891|       cekStatusVariable();
892|   }
893|
894|   updateGPS();        // Perbarui data GPS
895|
896|
897|   if (wasDisconnected && WiFi.status() == WL_CONNECTED) {
898|         Serial.println("[INFO]  Menjalankan reconnect manual ke WebSocket.");
899|         webSocketServer.begin();  // Memulai ulang server jika putus
900|         wasDisconnected = false;  // Reset flag setelah reconnect
901|     }
902|
903|   // Memanggil rutin untuk WebSocket Server
904|   webSocketServer.loop();
905|
906|   // Kirim pesan 'KEEP_ALIVE' jika ada client yang terhubung
907|   if (webSocketServer.connectedClients()) {
908|     static unsigned long lastKeepAlive = 0;
909|     if (millis() - lastKeepAlive > 5000) {
910|       webSocketServer.broadcastTXT("KEEP_ALIVE");
911|       lastKeepAlive = millis();
912|     }
913|   }
914|
915| }
916|
917|
918| // Handler untuk WebSocket Event versi penambahan flag fileIsComplete
919| void handleWebSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t
length) {
920|     switch (type) {
921|         case WStype_PING: {
922|             // Serial.printf("[INFO]  PING received from client %u\n", num);
923|             break;
924|         }
925|
926|         case WStype_DISCONNECTED: {
927|             Serial.printf("[%u] [INFO] Websocket Disconnected!\n", num);
928|             wasDisconnected = true;  // Tandai disconnect
929|             resetFlags(); //20250213_2243
930|             break;
931|         }
932|
933|         case WStype_CONNECTED: {
934|             Serial.println("[INFO]  WebSocket Connected.");
935|             // Memutar file 0098.mp3 untuk notifikasi "WebSocket Connected"
936|             dfPlayer.playMp3Folder(98);
937|             delay(2000);
938|             // Jika reconnect bukan karena restart atau setup awal
939|                 if (wasDisconnected && setupComplete && isReceivingFile &&
incomingFileName != "") {
940|                 wasDisconnected = false;  // Reset flag reconnect setelah reconnect
941|                 int retryCount = 3;
942|                 while (!incomingFile && retryCount > 0) {
943|                     incomingFile = SD.open("/" + incomingFileName, FILE_WRITE);
944|                     if (!incomingFile) {
945|                         Serial.println("[ERROR] Gagal membuka kembali file setelah
reconnect.");
946|                         retryCount--;
947|                         delay(200); // --> Default 200 Trial 100
948|                     } else {
949|                         Serial.println("[INFO] File dibuka kembali untuk menerima
data setelah reconnect.");
950|                     }
951|                 }
952|             }
953|             wasDisconnected = false;  // Reset flag setelah reconnect
954|             break;
```

```
955|         }
956|
957|         case WStype_TEXT: {
958|             String message = String((char*)payload).substring(0, length);
959|                     Serial.printf("[INFO]     WebSocket  Message  Received:  %s\n",
message.c_str()); // disable_print
960|
961|             if (message == "mode_operation") {
962|                 changeMode(OPERATIONAL);
963|             } else if (message == "mode_maintenance") {
964|                 changeMode(MAINTENANCE);
965|             } else if (message == "current_mode") {
966|                 if (currentMode == OPERATIONAL) {
967|                     Serial.println("Mode: Operational");
968|                 } else if (currentMode == MAINTENANCE) {
969|                     Serial.println("Mode: Maintenance");
970|                 }
971|             }
972|
973|             if (message.startsWith("FILE_SIZE:")) {
974|                 String fileSizeStr = message.substring(10);
975|                 if (fileSizeStr.length() > 0 && fileSizeStr.toInt() > 0) {
976|                     expectedFileSize = fileSizeStr.toInt();
977|                           Serial.printf("[INFO]   Expected  File  Size:  %d  bytes\n",
expectedFileSize);
978|                     fileComplete = true; // default --> false --> coba true
979|                     totalBytesReceived = 0;
980|                 } else {
981|                     Serial.println("[ERROR] Invalid file size received.");
982|                 }
983|
984|             } else if (strcmp((const char *)payload, "START_FILE") == 0) {
985|                 if (!incomingFile && incomingFileName != "") {
986|                     int retryCount = 3;
987|                     while (!incomingFile && retryCount > 0) {
988|                         incomingFile = SD.open("/" + incomingFileName, FILE_WRITE);
989|                         if (!incomingFile) {
990|                                Serial.println("[ERROR] Failed to re-open file after
reconnect.");
991|                             retryCount--;
992|                             delay(200); //--> Default 200 Trial 100
993|                         } else {
994|                                Serial.println("[INFO]  File berhasil dibuka kembali
untuk menerima data.");
995|                         }
996|                     }
997|                 }
998|                 incomingFileName = "";
999|                 isReceivingFile = true;
1000|                 Serial.println("[DEBUG] Start receiving file.");
1001|
1002|             } else if (isReceivingFile && incomingFileName == "") {
1003|                 incomingFileName = String((const char *)payload);
1004|                 incomingFile = SD.open("/" + incomingFileName, FILE_WRITE);
1005|                 if (!incomingFile) {
1006|                     Serial.println("[ERROR] Failed to open file.");
1007|                 } else {
1008|                     Serial.println("[DEBUG] Receiving file: " + incomingFileName);
1009|                 }
1010|
1011|             } else if (strcmp((const char *)payload, "END_FILE") == 0) {
1012|                 isReceivingFile = false;
1013|                 if (incomingFile) {
1014|                     incomingFile.flush();
1015|                     incomingFile.close();
1016|                     Serial.println("[INFO]  File berhasil diterima dan disimpan.");
1017|                     if (totalBytesReceived == expectedFileSize) {
1018|                         fileComplete = true;
1019|                          Serial.println("[INFO]  File diterima dengan filesize yang
sesuai.");
1020|                         resetFlags(); // Default tidak ada --> propose add
1021|                         resetActuators(); //20250122_2304
```

```
1022|                     } else {
1023|                       Serial.printf("[ERROR] File size mismatch: Expected %d bytes,
received %d bytes.\n", expectedFileSize, totalBytesReceived);
1024|                     }
1025|                 } else {
1026|                       Serial.println("[ERROR] No file is open to close or invalid
filename.");
1027|                 }
1028|                 Serial.println("[DEBUG] Respon websocket selesai");
1029|                 //Pengiriman gambar dari kamera berhasil.
1030|                  // Memutar file 0076.mp3 untuk notifikasi "Pengiriman gambar dari
kamera berhasil"
1031|                 dfPlayer.playMp3Folder(76);
1032|                 cekStatusVariable();
1033|             }
1034|             break;
1035|         }
1036|
1037|         case WStype_BIN: {
1038|             if (isReceivingFile && incomingFile) {
1039|                 size_t bytesWritten = incomingFile.write(payload, length);
1040|                 totalBytesReceived += bytesWritten;
1041|                   // Serial.printf("[INFO] Menulis %u byte ke MicroSD logger \n",
bytesWritten);
1042|
1043|                 if (totalBytesReceived % 4096 == 0) {
1044|                     incomingFile.flush(); // Serial.println("[INFO] Data flushed ke
MicroSD.");
1045|                 }
1046|
1047|                 if (bytesWritten != length) {
1048|                    Serial.println("[ERROR] Data tidak sepenuhnya ditulis ke file.");
1049|                 }
1050|                 if (totalBytesReceived == expectedFileSize) {
1051|                     fileComplete = true;
1052|                       Serial.println("[INFO]  File diterima dari ESP32-CAM dengan
filesize yang sesuai.");
1053|                 }
1054|             } else if (isReceivingFile) {
1055|                 Serial.println("[ERROR] No file is open to write.");
1056|             }
1057|             break;
1058|         }
1059|     }
1060| }
```

## LAMPIRAN 2 Kode Program ESP32-CAM

```
1 | // Nama file terakhir: FINALISASI_CAM_100H_2025_2254_02.txt
2 | // Update terakhir: [13 FEB 2025 22:54]
3 | // Kontributor: [A.Amrulloh]
4 |
5 | //Library yang digunakan
6 | #include "esp_camera.h"
7 | #include "FS.h"
8 | #include "SD_MMC.h"
9 | #include <WiFi.h>
10| #include <WebSocketsClient.h>
11| #include <WebServer.h>
12| #include <HTTPClient.h>
13| #include <WiFiClient.h>
14| #include <WiFiClientSecure.h>
15| #include <MD5Builder.h>  // Tambahkan library untuk menghitung checksum MD5
16|
17| // Konfigurasi WiFi
18| // const char* ssid = "NC_AQM12";
19| // const char* password = "MN200808";
20| const char* ssid = "Python 12";
21| const char* password = " Dimxp1223?";
22|
23| // Konfigurasi WebSocket
24| WebSocketsClient webSocket;
```

```
25| // const char* ws_server_ip = "192.168.174.235"; //tethering python12
26| const char* ws_server_ip = "192.168.139.235"; //tethering python12
27| // const char* ws_server_ip = "192.168.101.71"; // PRIMARY--IP ESP32-UTAMA
28| //  const  char*  ws_server_ip  =  "192.168.101.83";  //  IP  ESP32-Utama  const  char*
ws_server_ip = "192.168.101.74";
29| const uint16_t ws_port = 81;                  // Port WebSocket
30|
31| // Folder di SD_MMC tempat menyimpan gambar
32| const char* imagePath = "/images";
33|
34| unsigned long lastPingTime = 0;
35| // const unsigned long PING_INTERVAL = 8000;  // 8 detik --> default
36| const unsigned long PING_INTERVAL = 5000;  // 5 detik
37| String site_id, alarmMode, connectionType, magnetStatus, limitSwitchStatus, pirStatus,
ldrStatus, gpsLatitude, gpsLongitude;
38|
39| //deklarasi variable
40| bool  imageSaved  =  false;  //  Tambahkan  ini  di  luar  fungsi  untuk  melacak  status
penyimpanan
41|
42|
43| // Pin SD_MMC
44| #define SD_MMC_CLK 14
45| #define SD_MMC_CMD 15
46| #define SD_MMC_D0 2
47| #define CAMERA_MODEL_AI_THINKER // Has PSRAM
48| #define PWDN_GPIO_NUM  32
49| #define RESET_GPIO_NUM -1
50| #define XCLK_GPIO_NUM  0
51| #define SIOD_GPIO_NUM  26
52| #define SIOC_GPIO_NUM  27
53| #define Y9_GPIO_NUM    35
54| #define Y8_GPIO_NUM    34
55| #define Y7_GPIO_NUM    39
56| #define Y6_GPIO_NUM    36
57| #define Y5_GPIO_NUM    21
58| #define Y4_GPIO_NUM    19
59| #define Y3_GPIO_NUM    18
60| #define Y2_GPIO_NUM    5
61| #define VSYNC_GPIO_NUM 25
62| #define HREF_GPIO_NUM  23
63| #define PCLK_GPIO_NUM  22
64| #define FLASH_GPIO_NUM 33
65|
66| String lastCapturedImage; // Simpan path gambar terakhir yang diambil
67| String postData;
68|
69| // FUNGSI CAPTURE DAN KIRIM GAMBAR KE SERVER SIMON
70| void captureAndSendImage() {
71|     // Pastikan parameter dari ESP32-Utama sudah diterima
72|     if (site_id != "" && alarmMode != "" && connectionType != "" && magnetStatus !=
"" && limitSwitchStatus != "" && pirStatus != "" &&
73|         ldrStatus != "" && gpsLatitude != "" && gpsLongitude != "") {
74|
75|         // Ambil dan simpan gambar ke SD_MMC
76|         captureAndSaveImage();
77|
78|         // Kirim gambar dan data ke Server Simon BTS
79|          Serial.println("[INFO] Mengirim gambar dan parameter alarm ke Server Simon
BTS...");
80|          sendImageToServer(lastCapturedImage);  // Kirim gambar terakhir bersama data
parameter alarm
81|
82|         //POS_1
83|         Serial.println("[INFO] Bersiap untuk mengirim gambar ke ESP32-UTAMA");
84|         sendFile(lastCapturedImage.c_str());
85|     } else {
86|         Serial.println("[ERROR] Data parameter alarm belum lengkap. Gambar tidak akan
dikirim.");
87|     }
88| }
89|
```

```
90|
91| // Fungsi untuk menghitung MD5 dari file gambar --> VERIFIKASI FILESIZE GAMBAR
92| String calculateMD5(File file) {
93|     MD5Builder md5;
94|     md5.begin();
95|     while (file.available()) {
96|         uint8_t buffer[512];
97|         size_t bytesRead = file.read(buffer, sizeof(buffer));
98|         md5.add(buffer, bytesRead);
99|     }
100|     file.seek(0);  // Reset posisi baca file kembali ke awal
101|     md5.calculate();
102|     return md5.toString();
103| }
104|
105| //FUNGSI INI UNTUK MENGIRIMKAN DATA ALARM DAN IMAGE/GAMBAR KE SERVER SIMON BTS
106| void sendImageToServer(const String &filePath) {
107|     // WiFiClientSecure client; // default
108|     WiFiClient client;
109|     // const char* server = "simonbts.my.id";  // IP server --> ini kalo pake ssl --
> https://
110|     const char* server = "simonbts.my.id";  // IP server
111|     const int port = 80; // port 80, port standar http, kalo https : 443
112|     const char* device_token = "esp32device-12345";  // Token yang digunakan untuk
autentikasi
113|
114|     // Debugging tambahan sebelum melakukan koneksi ke server
115|     Serial.print("[DEBUG] Mencoba menghubungkan ke server: ");
116|     Serial.print(server);
117|     Serial.print(" pada port: ");
118|     Serial.println(port);
119|
120|     // Mencoba koneksi ke server
121|     if (!client.connect(server, port)) {
122|         Serial.println("[ERROR] Koneksi ke server gagal.");
123|         return;
124|     }
125|     Serial.println("[INFO] Koneksi ke server berhasil.");
126|
127|     File file = SD_MMC.open(filePath, FILE_READ);
128|     if (!file) {
129|         Serial.println("[ERROR] Tidak dapat membuka file di SD_MMC untuk Server Simon
BTS.");
130|         client.stop();
131|         return;
132|     }
133|
134|     // Boundary dan header
135|     String boundary = "ESP32Boundary";
136|     String payloadHeader = "--" + boundary + "\r\n";
137|     payloadHeader += "Content-Disposition: form-data; name=\"site_id\"\r\n\r\n" +
String(site_id) + "\r\n";  // Nama site
138|     payloadHeader += "--" + boundary + "\r\n";
139|     payloadHeader += "Content-Disposition: form-data; name=\"mode\"\r\n\r\n" +
alarmMode + "\r\n";
140|     payloadHeader += "--" + boundary + "\r\n";
141|     payloadHeader += "Content-Disposition: form-data; name=\"connection\"\r\n\r\n" +
connectionType + "\r\n";
142|     payloadHeader += "--" + boundary + "\r\n";
143|     payloadHeader += "Content-Disposition: form-data; name=\"magnet_s\"\r\n\r\n" +
magnetStatus + "\r\n";
144|     payloadHeader += "--" + boundary + "\r\n";
145|     payloadHeader += "Content-Disposition: form-data; name=\"limit_s\"\r\n\r\n" +
limitSwitchStatus + "\r\n";
146|     payloadHeader += "--" + boundary + "\r\n";
147|     payloadHeader += "Content-Disposition: form-data; name=\"pir_s\"\r\n\r\n" +
pirStatus + "\r\n";
148|     payloadHeader += "--" + boundary + "\r\n";
149|     payloadHeader += "Content-Disposition: form-data; name=\"ldr_s\"\r\n\r\n" +
ldrStatus + "\r\n";
150|     payloadHeader += "--" + boundary + "\r\n";
```

```
151|      payloadHeader += "Content-Disposition: form-data; name=\"gps_lat\"\r\n\r\n" +
gpsLatitude + "\r\n";
152|     payloadHeader += "--" + boundary + "\r\n";
153|      payloadHeader += "Content-Disposition: form-data; name=\"gps_lon\"\r\n\r\n" +
gpsLongitude + "\r\n";
154|     payloadHeader += "--" + boundary + "\r\n";
155|     payloadHeader += "Content-Disposition: form-data; name=\"image\"; filename=\"" +
filePath + "\"\r\n";
156|     payloadHeader += "Content-Type: image/jpeg\r\n\r\n";
157|
158|     // Menghitung total ukuran payload untuk HTTP header
159|     String endBoundary = "\r\n--" + boundary + "--\r\n";
160|     size_t totalSize = payloadHeader.length() + file.size() + endBoundary.length();
161|
162|     // Kirim HTTP request header
163|     // client.print("POST /simonbts/log_alarm.php HTTP/1.1\r\n"); // DEFAULT KODE
164|     client.print("POST /log_alarm.php HTTP/1.1\r\n");
165|     client.print("Host: " + String(server) + "\r\n");
166|     client.print("Content-Type: multipart/form-data; boundary=" + boundary + "\r\n");
167|     client.print("Content-Length: " + String(totalSize) + "\r\n");
168|     client.print("Connection: close\r\n\r\n");
169|
170|     // Kirim payload header
171|     client.print(payloadHeader);
172|
173|     // Kirim data gambar dalam bentuk biner
174|     while (file.available()) {
175|         // uint8_t buffer[2048];         // default
176|         uint8_t buffer[3072];         // default
177|         // uint8_t buffer[4096];         // MAx Value for FB 4KB, lower is OK
178|         size_t bytesRead = file.read(buffer, sizeof(buffer));
179|         client.write(buffer, bytesRead);  // Kirim data biner gambar KE SERVER SIMON
BTS
180|     }
181|     file.close();
182|
183|     // Kirim end boundary
184|     client.print(endBoundary);
185|
186|     // Membaca respon dari server
187|     while (client.connected() || client.available()) {
188|         if (client.available()) {
189|             String response = client.readStringUntil('\n');
190|             Serial.println("[SERVER RESPONSE 1] " + response);
191|
192|             // Hentikan setelah mendapatkan "HTTP/1.1 200 OK"
193|             // if (response.startsWith("Expires:")) {
194|             if (response.startsWith("Date:")) {
195|                 break;
196|             }
197|         }
198|     }
199|
200|     client.stop();  // Menutup koneksi
201|     Serial.println("[INFO] Gambar berhasil dikirim ke Server Simon BTS.");
202|     // Serial.println("[INFO] Siap untuk mengirim gambar ke ESP32-UTAMA");
203|
204| }
205|
206|
207| // Fungsi untuk mendapatkan timestamp dalam format YYYYMMDD_HHMMSS
208| String getTimeStamp() {
209|     struct tm timeinfo;
210|     if (!getLocalTime(&timeinfo)) {
211|         return "00000000_000000";  // Timestamp default jika gagal
212|     }
213|     char buffer[20];
214|     strftime(buffer, sizeof(buffer), "%Y%m%d_%H%M%S", &timeinfo);
215|     return String(buffer);
216| }
217|
218|
```

```
// Fungsi untuk capture dan simpan gambar di SD_MMC
void captureAndSaveImage() {
    Serial.println("[INFO] Mengambil gambar sesuai permintaan ESP32-UTama.");

    // Ambil frame pertama dan buang untuk memastikan buffer penuh --> fitur penting
    camera_fb_t *fb = esp_camera_fb_get();
    if (fb) {
        esp_camera_fb_return(fb);
        fb = esp_camera_fb_get();  // Ambil frame kedua yang lebih stabil
    }

    if (!fb) {
        Serial.println("[ERROR] Gagal mengambil gambar dari kamera.");
        return;
    } else {
        // Serial.println("[DEBUG] Gambar HiRes berhasil diambil dari kamera.");
        Serial.println("[DEBUG] Gambar alarm berhasil diambil dari kamera.");
        Serial.println("[INFO] Ukuran gambar (bytes): " + String(fb->len));
    }

    // Buat nama file dengan timestamp
    String timeStamp = getTimeStamp();
    String fileName = "/PICTURE_" + timeStamp + ".jpg";
    String filePath = String(imagePath) + fileName;
    lastCapturedImage = filePath;
    Serial.println("[DEBUG di SD_MMC] Menyimpan gambar di SD_MMC: " + filePath);

    // Buka file untuk penulisan
    File file = SD_MMC.open(filePath, FILE_WRITE);
    if (!file) {
        Serial.println("[ERROR di SD_MMC] Gagal membuka file di SD_MMC.");
        esp_camera_fb_return(fb);
        return;
    }
     Serial.println("[DEBUG di SD_MMC] File berhasil dibuka untuk ditulis: " +
filePath);


    // Tulis data gambar ke file
    size_t bytesWritten = file.write(fb->buf, fb->len);
    file.flush();
    Serial.println("[DEBUG di SD_MMC] Data gambar ditulis ke SD_MMC.");
    file.close();
    Serial.println("[DEBUG di SD_MMC] File ditutup setelah ditulis");
    esp_camera_fb_return(fb);

    if (bytesWritten == fb->len) {
    Serial.println("[INFO] Gambar berhasil disimpan pada : " + filePath + " (Bytes:
" + String(fb->len) + ")");
    delay(300);
    imageSaved = true;
    } else {
    Serial.println("[ERROR] Hanya sebagian gambar yang tersimpan.");
    imageSaved = false;
    }
}


// Fungsi untuk mengirim file gambar ke ESP32-UTama
void sendFile(const char* filePath) {
    // Debug tambahan untuk memeriksa path file
    Serial.printf("[DEBUG] Path file yang akan dikirim: %s\n", filePath);

    // Retry beberapa kali untuk memeriksa apakah file tersedia
    const int maxRetries = 5;
    for (int retries = 0; retries < maxRetries; ++retries) {
        if (SD_MMC.exists(filePath)) {
            break; // Keluar jika file ditemukan
        }
        Serial.println("[ERROR] File tidak ditemukan, mencoba ulang...");
        delay(200);
    }
```

```
289|
290|     // Cek apakah file ada setelah percobaan
291|     if (!SD_MMC.exists(filePath)) {
292|        Serial.printf("[ERROR] File tidak ditemukan setelah beberapa percobaan: %s\n",
filePath);
293|         return;
294|     }
295|
296|     // Buka file jika tersedia
297|     File file = SD_MMC.open(filePath);
298|     if (!file) {
299|         Serial.printf("[ERROR] Gagal membuka file: %s\n", filePath);
300|         return;
301|     }
302|
303|     // Kirim sinyal `START_FILE` hanya jika gambar berhasil disimpan
304|     if (imageSaved) {
305|         webSocket.sendTXT("START_FILE");
306|         Serial.println("[INFO] Mengirim sinyal START_FILE ke ESP32-Utama.");
307|     } else {
308|             Serial.println("[ERROR]  Gambar  belum  tersimpan,  tidak  bisa  mengirim
START_FILE.");
309|         file.close();
310|         return;
311|     }
312|
313|     // Kirim nama file
314|     webSocket.sendTXT(String(filePath).c_str()); //ori
315|     delay(100); // Delay singkat untuk memastikan pesan terkirim
316|
317|     // Kirim ukuran file sebagai pesan tambahan
318|     size_t fileSize = file.size();
319|     webSocket.sendTXT("FILE_SIZE: " + String(fileSize) + " bytes");
320|      Serial.printf("[INFO]  FileSize  gambar  yang  akan  dikirim  ke  ESP32-Utama:  %s
Bytes\n", String(fileSize).c_str());
321|     delay(100); // Delay singkat untuk memastikan pesan terkirim
322|
323|     // Kirim file dalam bentuk biner
324|     uint8_t buffer[2048];  // Buffer 2 KB untuk mengirim data gambar --> OPTIM_VALUE
325|     while (file.available()) {
326|         size_t bytesRead = file.read(buffer, sizeof(buffer));
327|         if (bytesRead > 0) {
328|             webSocket.sendBIN(buffer, bytesRead);
329|             delay(5);
330|         }
331|     }
332|
333|     file.close();
334|     Serial.println("[INFO] File gambar berhasil dikirim ke ESP32-Utama.");
335|     webSocket.sendTXT("END_FILE"); // panggil end_file disini
336| }
337|
338|
339| // ESP32-CAM WebSocket Callback
340| void webSocketEvent(WStype_t type, uint8_t * payload, size_t length) {
341|     if (type == WStype_TEXT) {
342|         String msg = String((char *)payload);
343|
344|         if (msg.startsWith("PARAMS:")) {
345|               // Periksa apakah pesan adalah data parameter DARI ESP32-UTAMA ?
346|               // Parsing data parameter
347|               int site_idIndex = msg.indexOf("site_id=") + 8;
348|               int modeIndex = msg.indexOf("mode=") + 5;
349|               int connectionTypeIndex = msg.indexOf("connection=") + 11;
350|               int magnetIndex = msg.indexOf("magnet_s=") + 9;
351|               int limitSwitchIndex = msg.indexOf("limit_s=") + 8;
352|               int pirIndex = msg.indexOf("pir_s=") + 6;
353|               int ldrIndex = msg.indexOf("ldr_s=") + 6;
354|               int latIndex = msg.indexOf("gps_lat=") + 8;
355|               int lonIndex = msg.indexOf("gps_lon=") + 8;
356|
357|               site_id = msg.substring(site_idIndex, msg.indexOf(";", site_idIndex));
```

```
358|                alarmMode = msg.substring(modeIndex, msg.indexOf(";", modeIndex));
359|                 connectionType = msg.substring(connectionTypeIndex, msg.indexOf(";",
connectionTypeIndex));
360|                        magnetStatus  =  msg.substring(magnetIndex,  msg.indexOf(";",
magnetIndex));
361|                 limitSwitchStatus = msg.substring(limitSwitchIndex, msg.indexOf(";",
limitSwitchIndex));
362|                pirStatus = msg.substring(pirIndex, msg.indexOf(";", pirIndex));
363|                ldrStatus = msg.substring(ldrIndex, msg.indexOf(";", ldrIndex));
364|                gpsLatitude = msg.substring(latIndex, msg.indexOf(";", latIndex));
365|                gpsLongitude = msg.substring(lonIndex, msg.indexOf(";", lonIndex));
366|
367|                // Debugging: Menampilkan parameter yang diterima
368|                Serial.println("[INFO] Parameter alarm diterima dari ESP32-Utama:");
369|                Serial.println("[PARAMS] Site Id: " + site_id);
370|                Serial.println("[PARAMS] Mode: " + alarmMode);
371|                Serial.println("[PARAMS] Connection: " + connectionType);
372|                Serial.println("[PARAMS] Magnet Status: " + magnetStatus);
373|                Serial.println("[PARAMS] Limit Switch Status: " + limitSwitchStatus);
374|                Serial.println("[PARAMS] PIR Status: " + pirStatus);
375|                Serial.println("[PARAMS] LDR Status: " + ldrStatus);
376|                Serial.println("[PARAMS] GPS Latitude: " + gpsLatitude);
377|                Serial.println("[PARAMS] GPS Longitude: " + gpsLongitude);
378|
379|              // Mempersiapkan data untuk dikirim ke server log_alarm.php
380|              if (!imageSaved) {
381|                  String postData = "site_id=" + site_id +
382|                                    "mode=" + alarmMode +
383|                                    "&connection=" + connectionType +
384|                                    "&magnet_s=" + magnetStatus +
385|                                    "&limit_s=" + limitSwitchStatus +
386|                                    "&pir_s=" + pirStatus +
387|                                    "&ldr_s=" + ldrStatus +
388|                                    "&gps_lat=" + gpsLatitude +
389|                                    "&gps_lon=" + gpsLongitude;
390|
391|              }
392|
393|          } else if (msg == "CAPTURE_IMAGE") {
394|              captureAndSendImage();
395|              delay(200);
396|          } else if (msg == "SEND_IMAGEx") {  // propose disable
397|              sendFile(lastCapturedImage.c_str());
398|              delay(200);
399|              webSocket.sendTXT("END_FILE");
400|          }
401|
402|      } else if (type == WStype_CONNECTED) {
403|          Serial.println("[DEBUG] WebSocket Connected.");
404|                   Serial.println("[DEBUG]   ESP32-CAM   mulai   pada   http://"   +
WiFi.localIP().toString());
405|          webSocket.sendTXT("ESP32-CAM Ready!");
406|          delay(200);
407|      } else if (type == WStype_DISCONNECTED) {
408|          // Serial.println("[ERROR] WebSocket Disconnected.");
409|      }
410| }
411|
412|
413| void setup() {
414|      // Inisialisasi Serial
415|      Serial.begin(115200);
416|
417|      // Konfigurasi WiFi
418|      WiFi.setHostname("ESP32_CAM");  // Set nama perangkat
419|      WiFi.begin(ssid, password, false);
420|          Serial.print("[DEBUG] Menghubungkan ke WiFi...");
421|      while (WiFi.status() != WL_CONNECTED) {
422|          delay(2000); // --> Default xxx Trial yyy
423|          Serial.print(".");
424|      }
425|      Serial.println("[DEBUG] WiFi Terhubung!");
```

```
426|
427|     //Inisialisasi Waktu (NTP)
428|     initTime();
429|
430|     // Inisialisasi SD_MMC
431|     if (!SD_MMC.begin()) {
432|         Serial.println("[ERROR] Gagal menginisialisasi SD_MMC.");
433|         return;
434|     }
435|
436|     // Inisialisasi SD_MMC - Cek apakah folder 'images' ada
437|     if (!SD_MMC.exists(imagePath)) {
438|             Serial.println("[DEBUG]  Folder  'images'  tidak  ada!  Mencoba  membuat
folder...");
439|         if (SD_MMC.mkdir(imagePath)) {
440|             Serial.println("[DEBUG] Folder 'images' berhasil dibuat");
441|         } else {
442|             Serial.println("[ERROR] Gagal membuat folder 'images'");
443|         }
444|     } else {
445|         Serial.println("[DEBUG] Folder 'images' sudah tersedia.");
446|     }
447|
448|     // Inisialisasi kamera
449|     camera_config_t config;
450|     config.ledc_channel = LEDC_CHANNEL_0;
451|     config.ledc_timer = LEDC_TIMER_0;
452|     config.pin_d0 = Y2_GPIO_NUM;
453|     config.pin_d1 = Y3_GPIO_NUM;
454|     config.pin_d2 = Y4_GPIO_NUM;
455|     config.pin_d3 = Y5_GPIO_NUM;
456|     config.pin_d4 = Y6_GPIO_NUM;
457|     config.pin_d5 = Y7_GPIO_NUM;
458|     config.pin_d6 = Y8_GPIO_NUM;
459|     config.pin_d7 = Y9_GPIO_NUM;
460|     config.pin_xclk = XCLK_GPIO_NUM;
461|     config.pin_pclk = PCLK_GPIO_NUM;
462|     config.pin_vsync = VSYNC_GPIO_NUM;
463|     config.pin_href = HREF_GPIO_NUM;
464|     config.pin_sccb_sda = SIOD_GPIO_NUM;
465|     config.pin_sccb_scl = SIOC_GPIO_NUM;
466|     config.pin_pwdn = PWDN_GPIO_NUM;
467|     config.pin_reset = RESET_GPIO_NUM;
468|     config.xclk_freq_hz = 20000000;
469|     // Resolusi Gambar FRAMESIZE_XGA(1024 x 768) // --> hasil file 93 KB
470|     config.frame_size = FRAMESIZE_XGA;
471|     config.pixel_format = PIXFORMAT_JPEG;
472|     config.fb_location = CAMERA_FB_IN_PSRAM;
473|     config.grab_mode = CAMERA_GRAB_LATEST;
474|     // kualitas JPEG config.jpeg_quality = 8; // maximal untuk FRAMESIZE_UXGA
475|     config.jpeg_quality = 8;
476|     config.fb_count = 1; // single buffer
477|
478|     // Inisialisasi kamera - Konfigurasi Sensor Kamera
479|     esp_err_t err = esp_camera_init(&config);
480|     if (err != ESP_OK) {
481|         Serial.printf("[ERROR] Gagal menginisialisasi kamera: 0x%x", err);
482|         return;
483|     }
484|
485|
486|     // Konfigurasi sensor kamera
487|     sensor_t * s = esp_camera_sensor_get();
488|     s->set_vflip(s, 1); // Gambar tidak terbalik.
489|     s->set_gain_ctrl(s, 1);
490|     s->set_exposure_ctrl(s, 1);
491|     s->set_awb_gain(s, 1);
492|     s->set_brightness(s, -1);
493|     s->set_saturation(s, 1);     // -2 to 2
494|     s->set_contrast(s, 1);       // -2 to 2
495|     s->set_gainceiling(s, (gainceiling_t)2);  // 0 to 6
```

```
496|     // 0 to 4 - if awb_gain enabled (0 - Auto, 1 - Sunny, 2 - Cloudy, 3 - Office, 4
- Home)
497|     s->set_wb_mode(s, 2);
498|     s->set_awb_gain(s, 1);       // 0 = disable , 1 = enable
499|     s->set_dcw(s, 1);            // 0 = disable , 1 = enable
500|     s->set_hmirror(s, 1);        // 0 = disable , 1 = enable
501|     s->set_vflip(s, 1);          // 0 = disable , 1 = enable
502|
503|     pinMode(FLASH_GPIO_NUM, OUTPUT);
504|     digitalWrite(FLASH_GPIO_NUM, LOW);
505|
506|
507|     // Inisialisasi WebSocket
508|     webSocket.begin(ws_server_ip, ws_port, "/");
509|     webSocket.onEvent(webSocketEvent);
510|     webSocket.setReconnectInterval(5000);
511|         Serial.println("[DEBUG] WebSocket server aktif");
512|
513| }
514|
515| void loop() {
516|
517|     webSocket.loop(); // Tangani event WebSocket
518|
519|     handleWebSocketPing();  // Pisahkan penanganan PING ke fungsi khusus
520|
521|     // Cek apakah ada data yang tersedia dari Serial Monitor
522|   if (Serial.available()) {
523|     String command = Serial.readStringUntil('\n'); // Membaca perintah hingga newline
524|     command.trim();
525|     // Periksa apakah perintahnya untuk mengganti mode
526|     if (command.equalsIgnoreCase("reset_sys") || command.equalsIgnoreCase("reset"))
{
527|       Serial.println("Diterima: " + command);
528|       Serial.println("Sistem akan di-reset.");
529|       WiFi.disconnect(true);  // Untuk menghentikan WiFi secara aman sebelum reset
530|         for (int i = 5; i > 0; i--) {
531|           Serial.print("Hitung mundur: ");
532|           Serial.print(i);
533|           Serial.println(" detik");
534|           delay(1000); // --> Default xxx Trial yyy
535|         }
536|       delay(100); // --> Default xxx Trial yyy
537|       ESP.restart();
538|                 }     else     if     (command.equalsIgnoreCase("captureSave")     ||
command.equalsIgnoreCase("capturesave")) {
539|       Serial.println("Diterima: " + command);
540|       captureAndSaveImage();
541|       Serial.println("[INFO] Jalankan " + command);
542|                 }     else     if     (command.equalsIgnoreCase("captureSend")     ||
command.equalsIgnoreCase("capturesend")) {
543|       Serial.println("Diterima: " + command);
544|       captureAndSendImage();
545|       Serial.println("[INFO] Jalankan " + command);
546|     } else {
547|       Serial.println("Diterima: " + command);
548|       Serial.println("[WARN] Perintah tidak dikenali: " + command);
549|     }
550|   }
551| }
552|
553| void handleWebSocketPing() {
554|     // Cek apakah sudah waktunya mengirimkan PING
555|     if (millis() - lastPingTime >= PING_INTERVAL) {
556|         // Serial.println("[INFO] Sending WebSocket PING...");
557|         webSocket.sendPing();  // Kirim PING ke server WebSocket
558|         lastPingTime = millis();
559|     }
560| }
561|
562|
563| void initTime() {
```

```
564|           configTime(25200,    0,    "time.cloudflare.com",    "asia.pool.ntp.org",
"time.google.com"); // GMT+7
565|     // configTime(25200, 0, "time.google.com", "asia.pool.ntp.org", "pool.ntp.org");
// GMT+7
566|     struct tm timeinfo;
567|     while (!getLocalTime(&timeinfo)) {
568|         Serial.println("[ERROR] Gagal sinkronisasi waktu, mencoba lagi...");
569|         delay(500); //DefValue 1000 --> 500
570|     }
571|     Serial.println("[DEBUG] Waktu berhasil disinkronkan");
572| }
```

## LAMPIRAN 3 Kode Program log_alarm.php

// *endpoint* untuk parsing data alarm dari ESP32-CAMERA

```
1 | <?php
2 | // Konfigurasi database
3 | $server = "localhost:3306";
4 | $user = "simx6871_admin";
5 | $pass = "R0cK3t!?C045t8080";
6 | $database = "simx6808871_simonbts";
7 |
8 | // Direktori untuk upload gambar
9 | $uploadDir = 'images/';
10| if (!is_dir($uploadDir)) {
11|     mkdir($uploadDir, 0755, true);
12| }
13|
14| // Path ke font .ttf yang telah diekstrak
15| $fontPath = '/home/simx6871/public_html/font/Roboto-Regular.ttf';  // Ganti dengan
lokasi font yang benar
16|
17| // Koneksi ke database
18| $conn = mysqli_connect($server, $user, $pass, $database);
19| if (!$conn) {
20|     die("Koneksi database gagal.");
21| }
22|
23| // Penanganan request POST
24| if ($_SERVER['REQUEST_METHOD'] == 'POST') {
25|     $contentType = $_SERVER['CONTENT_TYPE'] ?? '';
26|     $textData = $_POST;
27|     $imagePath = '';
28|
29|     // Proses teks (prioritas utama)
30|     $site_id = $textData['site_id'] ?? 'unknown';
31|     $mode = $textData['mode'] ?? 'unknown';
32|     $connection = $textData['connection'] ?? 'unknown';
33|     $magnet_s = isset($textData['magnet_s']) && $textData['magnet_s'] === '1' ? 1 :
0;
34|     $limit_s = isset($textData['limit_s']) && $textData['limit_s'] === '1' ? 1 : 0;
35|     $pir_s = isset($textData['pir_s']) && $textData['pir_s'] === '1' ? 1 : 0;
36|     $ldr_s = isset($textData['ldr_s']) && $textData['ldr_s'] === '1' ? 1 : 0;
37|     $gps_lat = $textData['gps_lat'] ?? null;
38|     $gps_lon = $textData['gps_lon'] ?? null;
39|
40|     // Ambil nama file gambar yang dikirim
41|     $fileName = basename($_FILES['image']['name']);
42|
43|     // Ekstrak timestamp dari nama file
44|     $fileNameWithoutExtension = pathinfo($fileName, PATHINFO_FILENAME);  // Ambil nama
file tanpa ekstensi
45|     $timestampFromFile = str_replace('PICTURE_', '', $fileNameWithoutExtension); //
Hilangkan "PICTURE_"
46|     $timestampFormatted = "Timestamp : " . $timestampFromFile; // Format menjadi
Timestamp : YYYYMMDD_HHMMSS
47|
48|     // Simpan data teks terlebih dahulu
49|     $sql = "INSERT INTO alarm_logs (timestamp, site_id, mode, connection, magnet_s,
limit_s, pir_s, ldr_s, gps_lat, gps_lon, image_path)
```

```
50|              VALUES (NOW(), '$site_id', '$mode', '$connection', $magnet_s, $limit_s,
$pir_s, $ldr_s, '$gps_lat', '$gps_lon', NULL)";
51|     if ($conn->query($sql) === TRUE) {
52|         $lastId = $conn->insert_id; // Dapatkan ID terakhir untuk update gambar nanti
53|
54|         // Proses gambar (jika ada)
55|         if (isset($_FILES['image']) && $_FILES['image']['error'] == UPLOAD_ERR_OK) {
56|             $filePath = $uploadDir . $fileName;
57|
58|             // Cek apakah gambar dapat dipindahkan
59|             if (move_uploaded_file($_FILES['image']['tmp_name'], $filePath)) {
60|                 // Buka gambar
61|                 $img = imagecreatefromjpeg($filePath);
62|
63|                 if ($img) {
64|                     // Tentukan warna untuk teks dan latar belakang
65|                     $textColor = imagecolorallocate($img, 255, 255, 255);  // Putih
66|                     $backgroundColor = imagecolorallocate($img, 0, 0, 0);  // Hitam
67|
68|                     // Tentukan ukuran font dan posisi awal
69|                     $fontSize = 20;
70|
71|                      // Dapatkan bounding box teks untuk mengetahui ukuran area yang
dibutuhkan
72|                    $bbox = imagettfbbox($fontSize, 0, $fontPath, $timestampFormatted);
73|
74|                     // Hitung lebar dan tinggi teks
75|                     $textWidth = $bbox[2] - $bbox[0]; // Lebar teks
76|                     $textHeight = $bbox[1] - $bbox[7]; // Tinggi teks
77|
78|                     // Tentukan posisi teks agar rata kanan
79|                    $x = imagesx($img) - $textWidth - 10; // Rata kanan dengan padding
10px
80|                    $y = imagesy($img) - 10;              // 10px di atas dasar gambar
81|
82|                     // Gambar latar belakang hitam dengan padding
83|                       imagefilledrectangle($img, $x - 5, $y - $textHeight - 5, $x +
$textWidth + 5, $y + 5, $backgroundColor);
84|
85|                     // Tambahkan timestamp ke gambar menggunakan font
86|                      imagettftext($img, $fontSize, 0, $x, $y, $textColor, $fontPath,
$timestampFormatted);
87|
88|                     // Simpan gambar dengan timestamp
89|                     imagejpeg($img, $filePath);
90|                     imagedestroy($img);  // Hapus gambar dari memori
91|                 }
92|
93|                 // Update path gambar di database
94|                 $imagePath = 'http://' . $_SERVER['HTTP_HOST'] . '/' . $filePath;
95|
96|                  $updateSql = "UPDATE alarm_logs SET image_path = '$imagePath' WHERE
id = $lastId";
97|                 if (!$conn->query($updateSql)) {
98|                     error_log("Failed to update image path: " . $conn->error);
99|                 }
100|             } else {
101|                 error_log("Failed to move uploaded file to: $filePath");
102|             }
103|         }
104|
105|         http_response_code(200);
106|          echo json_encode(['status' => 'success', 'message' => 'Data teks berhasil
disimpan.', 'image_path' => $imagePath, 'formatted_timestamp' => $timestampFormatted]);
107|     } else {
108|         http_response_code(500);
109|          echo json_encode(['status' => 'error', 'message' => 'Gagal menyimpan data
teks.', 'error' => $conn->error]);
110|     }
111| }
112| ?>
```