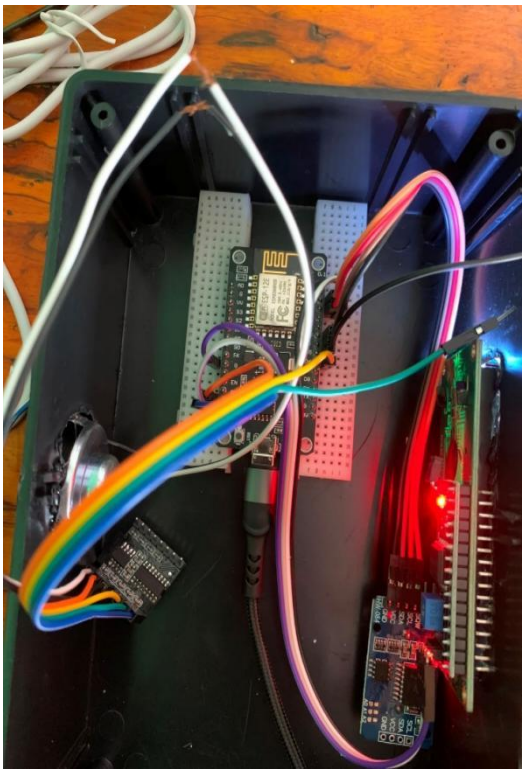
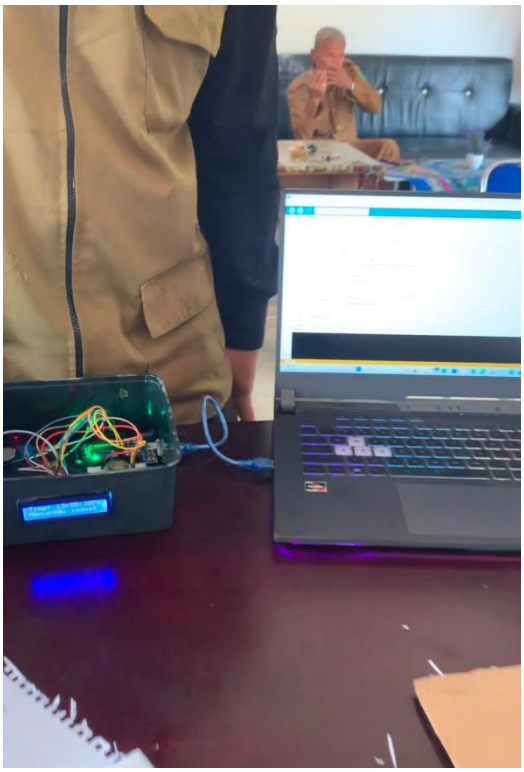
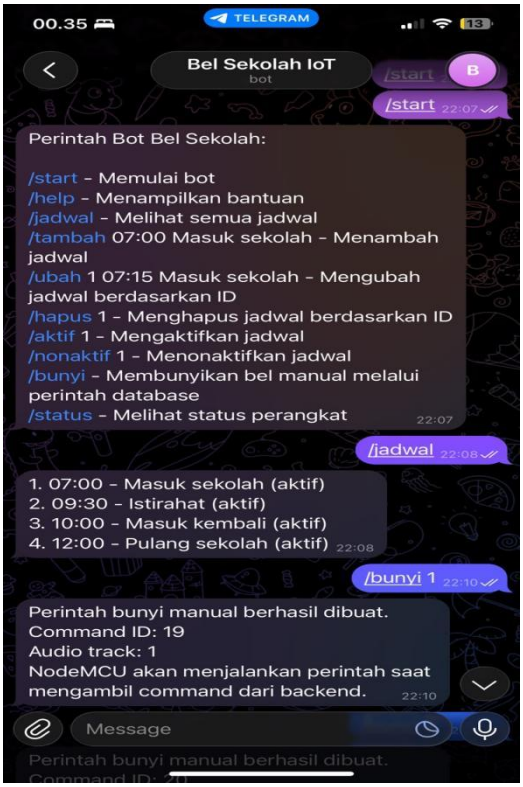
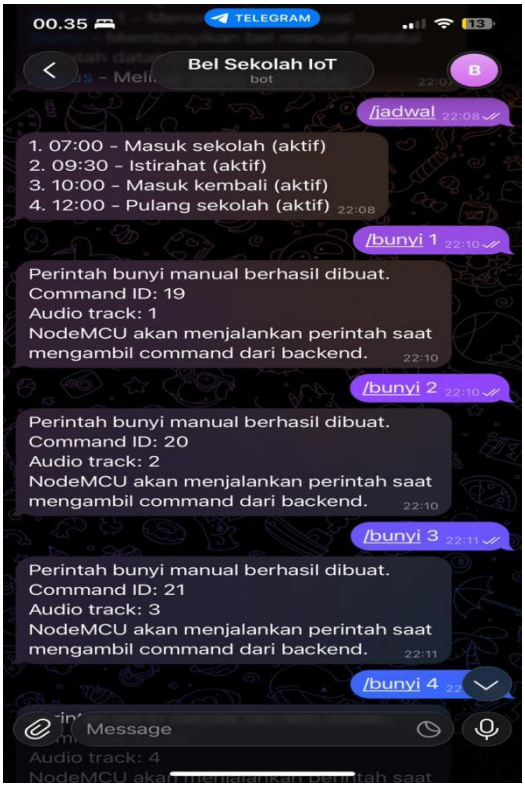
LAMPIRAN

Dokumentasi Penelitian









Sketch Lengkap

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <WiFiClient.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <RTClib.h>
#include <LiquidCrystal_I2C.h>
#include <DFRobotDFPlayerMini.h>
// ====================================================
// KONFIGURASI WIFI
// ====================================================
#define WIFI_SSID "ABC"
#define WIFI_PASSWORD "abc12345"
// ====================================================
// KONFIGURASI BACKEND
// ====================================================
#define BACKEND_BASE_URL "https://iot-school-bell-
api.vercel.app"
#define BACKEND_API_KEY
"dce83cc697cc56b93bdc8835430defe4769c60a04c12c627d8484aaa8ee0189f"
#define DEVICE_API_KEY
"5a67872694ac897dc85507befde801a470addd981ae4d8346d6502617dfef27f"
#define USE_HTTPS true
// ====================================================
// PIN ESP32
// ====================================================
// I2C untuk LCD + RTC DS3231
#define I2C_SDA 21
#define I2C_SCL 22
// DFPlayer Mini menggunakan UART2 ESP32
// GPIO16 (RX ESP32)  <- TX DFPlayer
// GPIO17 (TX ESP32)  -> RX DFPlayer
// SPK_1 dan SPK_2 DFPlayer langsung ke speaker, bukan ke GPIO
ESP32.
// Disarankan resistor 1K antara GPIO17 ESP32 dan RX DFPlayer.
#define DF_RX 16
#define DF_TX 17
// ====================================================
// OBJECT HARDWARE
// ====================================================
RTC_DS3231 rtc;
LiquidCrystal_I2C lcd(0x27, 16, 2);
// UART2 hardware ESP32, tidak menggunakan SoftwareSerial
HardwareSerial dfSerial(2);
DFRobotDFPlayerMini dfPlayer;
bool dfReady = false;
WiFiClient plainClient;
WiFiClientSecure secureClient;
// ====================================================
// TIMER
// ====================================================
```

```
unsigned long lastLCDUpdate = 0;
unsigned long lastFetchSchedule = 0;
unsigned long lastStatusUpdate = 0;
unsigned long lastCommandCheck = 0;
const unsigned long lcdInterval = 1000;
const unsigned long fetchScheduleInterval = 60000;
const unsigned long statusUpdateInterval = 30000;
const unsigned long commandCheckInterval = 5000;
// ======================================================
// STATUS AUDIO / BEL
// ======================================================
bool bellActive = false;
unsigned long bellStartTime = 0;
// Pengaman jika DFPlayer tidak mengirim event selesai.
// Audio normal tetap dibiarkan selesai secara alami.
const unsigned long audioSafetyTimeout = 60000;
int lastResetDay = -1;
// ======================================================
// DATA JADWAL
// ======================================================
struct JadwalBel {
  int id;
  int jam;
  int menit;
  int audioTrack;
  String label;
  bool sudahBunyi;
};

JadwalBel jadwal[20];
int jumlahJadwal = 0;


// ======================================================
// KHUSUS DFPLAYER MINI
// ======================================================
bool initDFPlayer() {
  Serial.println("[DFPlayer] Memulai UART2...");


  // RX ESP32 = GPIO16, TX ESP32 = GPIO17
  dfSerial.begin(9600, SERIAL_8N1, DF_RX, DF_TX);
  delay(800);


  // ACK aktif dan reset DFPlayer saat begin
  if (!dfPlayer.begin(dfSerial, true, true)) {
    dfReady = false;
    Serial.println("[DFPlayer] GAGAL terdeteksi");
    Serial.println("Periksa wiring, microSD, tegangan, dan
file MP3.");
    return false;
  }
```

```
        dfReady = true;


        // Volume 0 sampai 30
        dfPlayer.volume(25);
        dfPlayer.EQ(DFPLAYER_EQ_NORMAL);
        dfPlayer.outputDevice(DFPLAYER_DEVICE_SD);


        Serial.println("[DFPlayer] Siap");
        Serial.println("Gunakan file: /mp3/0001.mp3, /mp3/0002.mp3,
dst.");
        return true;
      }


      bool putarAudioDFPlayer(int audioTrack) {
        if (audioTrack < 1 || audioTrack > 9999) {
          Serial.print("[DFPlayer] Nomor track tidak valid: ");
          Serial.println(audioTrack);
          return false;
        }
        if (!dfReady) {
          Serial.println("[DFPlayer] Belum siap, mencoba
inisialisasi ulang...");
          if (!initDFPlayer()) {
            return false;
          }
        }
        Serial.print("[DFPlayer] Memutar /mp3/");
        if (audioTrack < 10) Serial.print("000");
        else if (audioTrack < 100) Serial.print("00");
        else if (audioTrack < 1000) Serial.print("0");
        Serial.print(audioTrack);
        Serial.println(".mp3");
        // playMp3Folder(1) = /mp3/0001.mp3
        // playMp3Folder(2) = /mp3/0002.mp3
        dfPlayer.playMp3Folder(audioTrack);
        return true;
      }
      // ======================================================
      // SETUP
      // ======================================================
      void setup() {
        Serial.begin(115200);
        delay(500);
        Serial.println();
        Serial.println("=== BEL SEKOLAH IoT ESP32 ===");
        // LCD dan RTC menggunakan bus I2C ESP32
        Wire.begin(I2C_SDA, I2C_SCL);
        lcd.init();
```

```
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Bel IoT ESP32");
  lcd.setCursor(0, 1);
  lcd.print("Loading...");
  if (!rtc.begin()) {
    lcd.clear();
    lcd.print("RTC Error");
    Serial.println("RTC DS3231 tidak ditemukan");
    while (true) {
      delay(1000);
    }
  }
  if (rtc.lostPower()) {
    Serial.println("RTC kehilangan daya. Set waktu dari waktu
compile.");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
  // Inisialisasi DFPlayer Mini melalui UART2 ESP32
  if (!initDFPlayer()) {
    lcd.clear();
    lcd.print("DFPlayer Error");
    delay(2000);
  }
  koneksiWiFi();
  ambilJadwalDariBackend();
  kirimStatusDevice();
}
// =======================================================
// LOOP
// =======================================================
void loop() {
  DateTime now = rtc.now();
  if (millis() - lastLCDUpdate >= lcdInterval) {
    lastLCDUpdate = millis();
    tampilkanLCD(now);
  }
  if (millis() - lastFetchSchedule >= fetchScheduleInterval) {
    lastFetchSchedule = millis();
    ambilJadwalDariBackend();
  }
  if (millis() - lastStatusUpdate >= statusUpdateInterval) {
    lastStatusUpdate = millis();
    kirimStatusDevice();
  }
  if (millis() - lastCommandCheck >= commandCheckInterval) {
    lastCommandCheck = millis();
    cekPerintahManualDariBackend();
  }
  cekJadwalBel(now);
  cekStatusAudioDFPlayer();
  resetStatusHarian(now);
}
// =======================================================
```

```
// WIFI
// ======================================================
void koneksiWiFi() {
  if (WiFi.status() == WL_CONNECTED) return;
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  lcd.clear();
  lcd.print("Koneksi WiFi");
  Serial.print("Menghubungkan WiFi");
  int retry = 0;
  while (WiFi.status() != WL_CONNECTED && retry < 30) {
    delay(500);
    Serial.print(".");
    retry++;
  }
  Serial.println();
  if (WiFi.status() == WL_CONNECTED) {
    lcd.clear();
    lcd.print("WiFi Terhubung");
    Serial.print("IP ESP32: ");
    Serial.println(WiFi.localIP());
  } else {
    lcd.clear();
    lcd.print("WiFi Gagal");
    Serial.println("WiFi gagal terhubung");
  }
  delay(1000);
}
// ======================================================
// HARI
// ======================================================
String namaHari(DateTime now) {
  switch (now.dayOfTheWeek()) {
    case 0: return "sunday";
    case 1: return "monday";
    case 2: return "tuesday";
    case 3: return "wednesday";
    case 4: return "thursday";
    case 5: return "friday";
    case 6: return "saturday";
  }
  return "monday";
}
// ======================================================
// HTTP REQUEST ESP32
// ======================================================
bool httpRequest(String method, String url, String jsonBody,
String &response, bool withDeviceKey) {
  if (WiFi.status() != WL_CONNECTED) {
    koneksiWiFi();
  }
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("HTTP dibatalkan: WiFi tidak terhubung");
    return false;
```

```
        }
        HTTPClient http;
        int httpCode = -1;
        if (USE_HTTPS) {
          // Untuk pengujian HTTPS. Untuk produksi lebih baik
gunakan sertifikat CA.
          secureClient.setInsecure();
          if (!http.begin(secureClient, url)) {
            return false;
          }
        } else {
          if (!http.begin(plainClient, url)) {
            return false;
          }
        }


        http.addHeader("x-api-key", BACKEND_API_KEY);
        if (withDeviceKey) {
          http.addHeader("x-device-key", DEVICE_API_KEY);
        }
        if (method != "GET") {
          http.addHeader("Content-Type", "application/json");
        }
        if (method == "GET") {
          httpCode = http.GET();
        } else if (method == "POST") {
          httpCode = http.POST(jsonBody);
        } else if (method == "PATCH") {
          httpCode = http.sendRequest("PATCH", jsonBody);
        }
        if (httpCode > 0) {
          response = http.getString();
        }
        Serial.print("HTTP ");
        Serial.print(method);
        Serial.print(" code: ");
        Serial.println(httpCode);
        http.end();
        return httpCode >= 200 && httpCode < 300;
      }
      // ======================================================
      // AMBIL JADWAL
      // ======================================================
      void ambilJadwalDariBackend() {
        DateTime now = rtc.now();
        String day = namaHari(now);
        String url = String(BACKEND_BASE_URL) +
"/api/device/schedules?day=" + day;
        String response = "";
        bool success = httpRequest("GET", url, "", response, false);
        if (!success) {
          Serial.println("Gagal ambil jadwal");
          Serial.println(response);
```

```
        lcd.clear();
        lcd.print("Jadwal gagal");
        delay(1000);
        return;
      }
      StaticJsonDocument<4096> doc;
      DeserializationError error = deserializeJson(doc, response);
      if (error) {
        Serial.println("JSON jadwal error");
        Serial.println(error.c_str());
        return;
      }
      JsonArray data = doc["data"].as<JsonArray>();
      jumlahJadwal = 0;


      for (JsonObject item : data) {
        if (jumlahJadwal >= 20) break;


        String waktu = item["time"].as<String>();
        int titikDua = waktu.indexOf(":");


        if (titikDua == -1) continue;


        jadwal[jumlahJadwal].id = item["id"].as<int>();
        jadwal[jumlahJadwal].jam = waktu.substring(0,
titikDua).toInt();
        jadwal[jumlahJadwal].menit = waktu.substring(titikDua + 1,
titikDua + 3).toInt();
        jadwal[jumlahJadwal].label = item["label"].as<String>();
        jadwal[jumlahJadwal].audioTrack =
item["audio_track"].as<int>();
        jadwal[jumlahJadwal].sudahBunyi = false;
        jumlahJadwal++;
      }
      Serial.print("Jumlah jadwal: ");
      Serial.println(jumlahJadwal);


      lcd.clear();
      lcd.print("Jadwal OK: ");
      lcd.print(jumlahJadwal);
      delay(1000);
    }


    // ====================================================
    // COMMAND MANUAL
    // ====================================================
    void cekPerintahManualDariBackend() {
```

```
        String url = String(BACKEND_BASE_URL) +
"/api/device/commands";
        String response = "";


        bool success = httpRequest("GET", url, "", response, true);


        if (!success) {
          Serial.println("Gagal ambil command");
          Serial.println(response);
          return;
        }


        StaticJsonDocument<4096> doc;
        DeserializationError error = deserializeJson(doc, response);


        if (error) {
          Serial.println("JSON command error");
          Serial.println(error.c_str());
          return;
        }


        JsonArray data = doc["data"].as<JsonArray>();


        for (JsonObject item : data) {
          int commandId = item["id"].as<int>();
          String commandType = item["command_type"].as<String>();
          int audioTrack = item["audio_track"].as<int>();


          if (commandType == "ring") {
            bunyikanBel(audioTrack);
            kirimLogBel(0, "manual", "Bel berbunyi manual dari
Telegram/backend");
            tandaiCommandSelesai(commandId, "done");
          }
        }
      }


      void tandaiCommandSelesai(int commandId, String status) {
        String url = String(BACKEND_BASE_URL) +
"/api/device/commands/" + String(commandId) + "/done";
        String response = "";


        StaticJsonDocument<128> doc;
```

```
  doc["status"] = status;


  String body;
  serializeJson(doc, body);


  httpRequest("PATCH", url, body, response, true);
}


// ======================================================
// STATUS DEVICE
// ======================================================
void kirimStatusDevice() {
  String url = String(BACKEND_BASE_URL) + "/api/device/status";
  String response = "";


  StaticJsonDocument<256> doc;
  doc["wifi_status"] = WiFi.status() == WL_CONNECTED ?
"connected" : "disconnected";
  doc["ip_address"] = WiFi.localIP().toString();


  String body;
  serializeJson(doc, body);


  httpRequest("PATCH", url, body, response, true);
}


// ======================================================
// LOG BEL
// ======================================================
void kirimLogBel(int scheduleId, String eventType, String
message) {
  String url = String(BACKEND_BASE_URL) + "/api/device/logs";
  String response = "";


  StaticJsonDocument<256> doc;


  if (scheduleId > 0) {
    doc["schedule_id"] = scheduleId;
  } else {
    doc["schedule_id"] = nullptr;
  }
```

```
  doc["event_type"] = eventType;
  doc["message"] = message;


  String body;
  serializeJson(doc, body);


  httpRequest("POST", url, body, response, true);
}


// ======================================================
// LCD
// ======================================================
void tampilkanLCD(DateTime now) {
  lcd.clear();


  lcd.setCursor(0, 0);
  if (now.hour() < 10) lcd.print("0");
  lcd.print(now.hour());
  lcd.print(":");
  if (now.minute() < 10) lcd.print("0");
  lcd.print(now.minute());
  lcd.print(":");
  if (now.second() < 10) lcd.print("0");
  lcd.print(now.second());


  lcd.setCursor(0, 1);


  if (WiFi.status() == WL_CONNECTED) {
    lcd.print("WiFi OK ");
  } else {
    lcd.print("WiFi OFF ");
  }


  if (bellActive) {
    lcd.print("BEL");
  } else if (dfReady) {
    lcd.print("DF OK");
  }
}
// ======================================================
// JADWAL BEL
// ======================================================
void cekJadwalBel(DateTime now) {
  for (int i = 0; i < jumlahJadwal; i++) {
    if (now.hour() == jadwal[i].jam &&
```

```
            now.minute() == jadwal[i].menit &&
            now.second() == 0 &&
            jadwal[i].sudahBunyi == false) {
          bunyikanBel(jadwal[i].audioTrack);
          jadwal[i].sudahBunyi = true;
          kirimLogBel(jadwal[i].id, "auto", "Bel otomatis: " +
jadwal[i].label);
        }
      }
    }
    // ======================================================
    // BUNYIKAN BEL + DFPLAYER
    // ======================================================
    void bunyikanBel(int audioTrack) {
      if (bellActive) {
        Serial.println("Audio masih diputar, perintah baru
diabaikan.");
        return;
      }


      bool audioOK = putarAudioDFPlayer(audioTrack);


      if (!audioOK) {
        bellActive = false;
        Serial.println("DFPlayer gagal memulai audio.");


        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("DFPlayer Gagal");
        return;
      }


      bellActive = true;
      bellStartTime = millis();


      Serial.print("Audio track: ");
      Serial.print(audioTrack);
      Serial.println(" mulai diputar melalui SPK_1 dan SPK_2.");


      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Bel Berbunyi");
      lcd.setCursor(0, 1);
      lcd.print("MP3 ");
      lcd.print(audioTrack);
    }
    void selesaiAudioDFPlayer() {
```

```
  if (!bellActive) return;
  bellActive = false;
  Serial.println("Audio selesai diputar.");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Audio Selesai");
}
// Tidak menggunakan pin BUSY.
// Status selesai dipantau dari event serial DFPlayer Mini.
void cekStatusAudioDFPlayer() {
  if (!dfReady) return;


  if (dfPlayer.available()) {
    uint8_t type = dfPlayer.readType();
    int value = dfPlayer.read();


    switch (type) {
      case DFPlayerPlayFinished:
        Serial.print("[DFPlayer] Track selesai: ");
        Serial.println(value);
        selesaiAudioDFPlayer();
        break;


      case DFPlayerError:
        Serial.print("[DFPlayer] Error code: ");
        Serial.println(value);
        bellActive = false;
        break;


      default:
        break;
    }
  }


  // Fallback agar status tidak terkunci jika event selesai
tidak diterima.
  if (bellActive && millis() - bellStartTime >=
audioSafetyTimeout) {
    Serial.println("[DFPlayer] Safety timeout, status audio
di-reset.");
    bellActive = false;
  }
}


// ====================================================
// RESET STATUS HARIAN
// ====================================================
```

```
      void resetStatusHarian(DateTime now) {
        if (now.hour() == 0 && now.minute() == 0 && now.day() !=
lastResetDay) {
          for (int i = 0; i < jumlahJadwal; i++) {
            jadwal[i].sudahBunyi = false;
          }


          lastResetDay = now.day();
          ambilJadwalDariBackend();
        }
      }
```