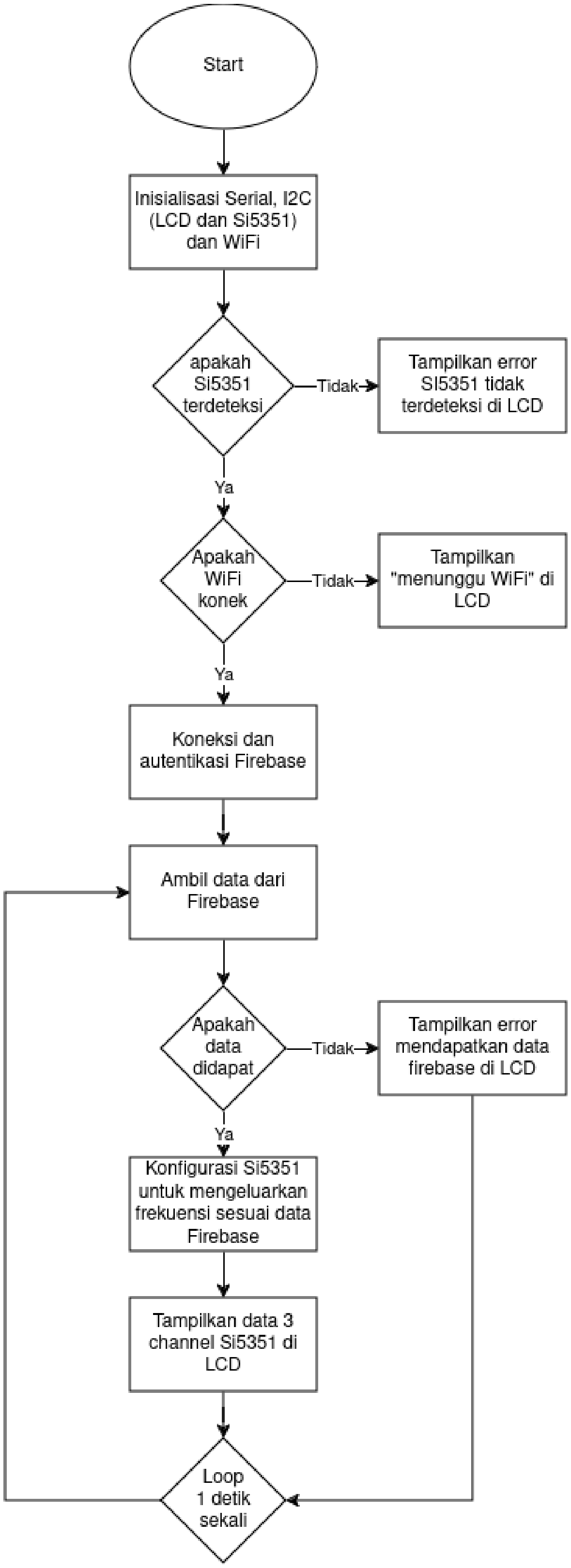
Figo Arzaki Maulana/5022221041

Direct Digital Synthesizer IoT dengan ESP32 dan Firebase
Flowchart Sistem


Start
Inisialisasi Serial, I2C (LCD dan Si5351) dan WiFi
apakah Si5351 terdeteksi
Tidak
Tampilkan error SI5351 tidak terdeteksi di LCD
Ya
Apakah WiFi konek
Tidak
Tampilkan "menunggu WiFi" di LCD
Ya
Koneksi dan autentikasi Firebase
Ambil data dari Firebase
Apakah data didapat
Tidak
Tampilkan error mendapatkan data firebase di LCD
Ya
Konfigurasi Si5351 untuk mengeluarkan frekuensi sesuai data Firebase
Tampilkan data 3 channel Si5351 di LCD
Loop 1 detik sekali

Figo Arzaki Maulana/5022221041

Penjelasan Kode

Mula-mula kode mengimport library yang dibutuhkan untuk project ini, setelah itu didefinisikan kredensial WiFi dan Firebase yang berupa SSID, Password untuk WiFi. Lalu API_KEY, USER_EMAIL, USER_PASSWORD dan DATABASE URL untuk Firebase. Setelah itu kode membuat objek-objek dan variabel-variabel global yang akan digunakan nantinya (Firebase, LCD, Si5351, definisi pin dan lain-lain).

Setelah itu ESP32 akan masuk ke program setup.

- **Program Setup**

Pada tahap setup, program memulai dengan menginisialisasi komunikasi serial untuk debugging, koneksi Wi-Fi menggunakan nama jaringan dan kata sandi yang telah ditentukan, serta protokol I2C untuk berkomunikasi dengan perangkat seperti LCD dan DDS. Kemudian, program memeriksa apakah modul DDS Si5351 terdeteksi. Jika tidak terdeteksi, sistem akan berhenti dan menampilkan pesan "DDS Not FOUND" pada LCD. Selanjutnya, sistem mencoba terhubung ke jaringan Wi-Fi dengan menampilkan pesan "Wait Wi-Fi" secara berulang hingga koneksi berhasil. Setelah terhubung, Firebase dikonfigurasi dengan API Key, kredensial pengguna, dan URL database. Firebase juga disiapkan untuk menangani aturan keamanan berbasis pengguna. Lalu kode akan masuk ke program loop.

- **Program Loop**

Tahap main loop adalah inti dari program, di mana fungsi utama dijalankan secara berulang. Pertama, program memeriksa apakah Firebase siap. Jika Firebase tidak siap, pesan "FB not ready" ditampilkan di LCD. Jika Firebase siap, program mengambil data frekuensi untuk tiga kanal (ch0, ch1, ch2) dari database Firebase. Jika data berhasil diambil, nilai tersebut digunakan untuk mengatur frekuensi DDS pada masing-masing kanal. Jika pengambilan data gagal, pesan "Fail to get data" akan ditampilkan pada LCD.

Selain itu, program memperbarui tampilan LCD setiap detik, bergantian antara menampilkan nilai frekuensi masing-masing kanal (dalam kHz atau MHz) dan label kanal (CH0, CH1, CH2). Sistem juga memanfaatkan LED sebagai indikator proses Firebase—LED menyala saat Firebase sedang diakses, dan mati jika Firebase tidak aktif atau pengambilan data gagal.

Figo Arzaki Maulana/5022221041

Pseudocode

- **Program Setup**

```
BEGIN
  Initialize Input and Output (LED)
  Initialize Serial communication (baud rate: 115200)
  Initialize Wi-Fi with WIFI_SSID and WIFI_PASSWORD
  Initialize I2C communication
  Initialize LCD (I2C address: 0x23)
  Turn on LCD backlight

  // Cek apakah SI5351 terdeteksi
  IF DDS initialization fails THEN
          Display "DDS Not FOUND" on LCD
          STOP execution
  ENDIF

  // Tunggu hingga konek WiFI
  WHILE Wi-Fi is not connected DO
          Display "Wait Wi-Fi" on LCD
          Wait 300 milliseconds
          Clear LCD
          Wait 300 milliseconds
  ENDWHILE

  Display "Connected" and "Wait FB Auth" on LCD

  Initialize Firebase with API_KEY, USER_EMAIL, USER_PASSWORD, and DATABASE_URL
  Enable Firebase network reconnection
  Configure Firebase buffer size and response size

  Setup Firebase security rules for database access
  Clear LCD and display "Wait Firebase" and "Sample Data"
END
```

- **Program Loop**

```
WHILE true DO
  // Update data Firebase setiap 1 detik
  IF time elapsed since last Firebase update >= 1 second THEN
          Reset Firebase data retrieval state

          IF Firebase is not ready THEN
          Display "FB not ready" on LCD
          Turn off LED
          CONTINUE to next iteration of loop program
          ENDIF

          Turn on LED

          // Ambil data frekuensi 3 channel dari firebase
```

```
      Fetch "/dds/ch0" as ch0_val
      Fetch "/dds/ch1" as ch1_val
      Fetch "/dds/ch2" as ch2_val

      IF all fetches are successful THEN
      Set DDS frequency for CH0, CH1, CH2 using ch0_val, ch1_val, ch2_val
      Update Firebase data retrieval state to true
      ELSE
      Display "Fail to get data" on LCD
      ENDIF

      Turn off LED
  ENDIF

  // Update LCD
  IF time elapsed since last LCD update >= 2 seconds AND Firebase data is valid THEN
      Clear LCD
      IF LCD display state is true THEN
      Display frequencies (ch0, ch1, ch2) on LCD (format: kHz or MHz)
      ELSE
      Display channel labels "CH0", "CH1", "CH2" on LCD
      ENDIF
      Toggle LCD display state
  ENDIF
ENDWHILE
```

Figo Arzaki Maulana/5022221041

Kode Lengkap

```
#include "si5351.h"
#include <Arduino.h>
#include <Firebase_ESP_Client.h>
#include <LiquidCrystal_I2C.h>
#include <WiFi.h>
#include <Wire.h>
#include <addons/RTDBHelper.h>
#include <addons/TokenHelper.h>

// I2C device found at address 0x23
// I2C device found at address 0x60
#define I2C_SDA 21
#define I2C_SCL 22
#define DDS_MULTIPLIER 100ULL
#define LCD_PERIOD 2000

#define WIFI_SSID "realme C15"
#define WIFI_PASSWORD "lpkojihu"

#define API_KEY "AIzaSyCWzuvdP0zmkR30zkM6ekgVc2hgGnLlcCg"
#define USER_EMAIL "esp32@esp32.com"
#define USER_PASSWORD "esp32esp32"
#define DATABASE_URL
"https://kohigashi-b72ca-default-rtdb.firebaseio.com/"
#define DATABASE_SECRET "DATABASE_SECRET"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

LiquidCrystal_I2C lcd(0x23, 16, 2);
Si5351 dds;

bool ledState = false;
bool last_get_state = false;
bool lcd_state = true;
uint8_t button_pin = 26;
uint8_t led_pin = 12;
```

```
uint32_t last_millis;
long ch0_val, ch1_val, ch2_val;

int count = 0;

void setup() {
 unsigned long ms = millis();

 pinMode(button_pin, INPUT_PULLUP);
 pinMode(led_pin, OUTPUT);

 Serial.begin(115200);

 WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

 Serial.print("Connecting to Wi-Fi");
 Wire.begin();

 lcd.init(I2C_SDA, I2C_SCL);
 lcd.backlight();
 lcd.clear();
 if (!dds.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0)) {
   lcd.setCursor(0, 0);
   lcd.print("DDS Not FOUND");
   while (1)
     ;
 }

 while (WiFi.status() != WL_CONNECTED) {
   lcd.print("Wait Wi-Fi");
   delay(300);
   lcd.clear();
   delay(300);
 }

 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Connected");
 lcd.setCursor(0, 1);
 lcd.print("Wait FB Auth");

 Serial.println();
 Serial.print("Connected with IP: ");
 Serial.println(WiFi.localIP());
 Serial.println();

 Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);

 config.api_key = API_KEY;
 auth.user.email = USER_EMAIL;
 auth.user.password = USER_PASSWORD;
 config.database_url = DATABASE_URL;

 Firebase.reconnectNetwork(true);

 fbdo.setBSSLBufferSize(4096, 1024);

 fbdo.setResponseSize(4096);

 String base_path = "/UsersData/";

 config.token_status_callback =
     tokenStatusCallback;

 Firebase.begin(&config, &auth);
 String var = "$userId";
 String val = "($userId === auth.uid && auth.token.premium_account ===
true "
              "&& auth.token.admin === true)";
 Firebase.RTDB.setReadWriteRules(&fbdo, base_path, var, val, val,
                                  DATABASE_SECRET);

 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Wait Firebase");
 lcd.setCursor(0, 1);
 lcd.print("Sample Data");
}
```

```
void loop() {
 static uint32_t last_millis_fb;
 if (last_millis_fb - millis() - 1000) {
   last_millis_fb = millis();
   last_get_state = false;
   if (!Firebase.ready()) {
     lcd.clear();
     lcd.setCursor(0, 0);
     lcd.print("FB not ready");
     Serial.println("Firebase not ready");
     digitalWrite(led_pin, LOW);
     return;
   }
   digitalWrite(led_pin, HIGH);
   bool ch0_get = Firebase.RTDB.getString(&fbdo, "/dds/ch0");
   if (ch0_get) {
     String ch0_string = fbdo.stringData();
     ch0_val = ch0_string.toInt();
   }
   bool ch1_get = Firebase.RTDB.getString(&fbdo, "/dds/ch1");
   if (ch1_get) {
     String ch1_string = fbdo.stringData();
     ch1_val = ch1_string.toInt();
   }
   bool ch2_get = Firebase.RTDB.getString(&fbdo, "/dds/ch2");
   if (ch2_get) {
     String ch2_string = fbdo.stringData();
     ch2_val = ch2_string.toInt();
   }
   if (ch0_get && ch1_get && ch2_get) {
     last_get_state = true;
     Serial.printf("Ch0: %d, Ch1: %d, Ch2: %d\n", ch0_val, ch1_val,
ch2_val);
     dds.set_freq(ch0_val * 1000 * DDS_MULTIPLIER, SI5351_CLK0);
     dds.set_freq(ch1_val * 1000 * DDS_MULTIPLIER, SI5351_CLK1);
     dds.set_freq(ch2_val * 1000 * DDS_MULTIPLIER, SI5351_CLK2);
   } else {
     lcd.clear();
```

```
      lcd.setCursor(0, 0);
      lcd.print("Fail to get data");
    }
    digitalWrite(led_pin, LOW);
  }

  if (millis() - last_millis >= LCD_PERIOD && last_get_state) {
    last_millis = millis();
    if (lcd_state) {
      lcd.clear();
      lcd.setCursor(0, 0);
      if(ch0_val < 1000)
        lcd.printf("%ldkHz", ch0_val);
      else
        lcd.printf("%.1fMHz", (float) ch0_val / 1000.);
      lcd.setCursor(9, 0);
      if(ch1_val < 1000)
        lcd.printf("%ldkHz", ch1_val);
      else
        lcd.printf("%.1fMHz", (float) ch1_val / 1000.);
      lcd.setCursor(0, 1);
      if(ch2_val < 1000)
        lcd.printf("%ldkHz", ch2_val);
      else
        lcd.printf("%.1fMHz", (float) ch2_val / 1000.);
    }else{
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("CH0");
      lcd.setCursor(9, 0);
      lcd.print("CH1");
      lcd.setCursor(0, 1);
      lcd.print("CH2");
    }
    lcd_state = !lcd_state;
  }
}
```