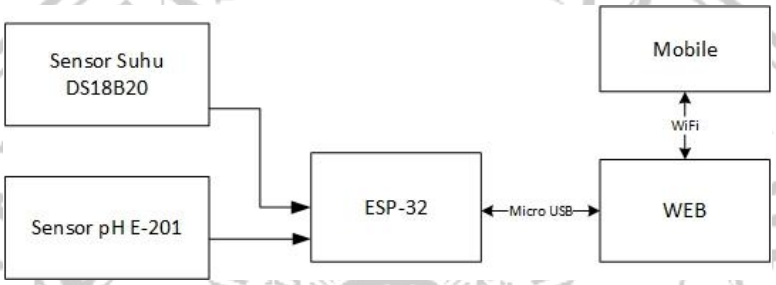
BAB III
PERANCANGAN SISTEM

Metode penilitian menggunakan ESP-32 sebagai modul *development board* yang memiliki *dual core processor* dengan komponen diantaranyasensor suhu tipe DS1820B dengan fitur kedap air dan sensor pH tipe pH E-201. kedua sensor tersebut untuk mikrokontroler ESP-32. untuk pemprosesan program LSTM terdapat pada web dan berjalannya sistem dapat dilihat dari blok diagram di bawah ini



Gambar 3.1 Blok Diagram Sistem

Dalam banyak aplikasi elektronika dan IoT (Internet of Things), penggunaan rangkaian yang terdiri beberapa komponen menjadi hal yang umum. Salah satu kombinasi yang digunakan adalah ESP32, sensor suhu DS18B20, dan sensor pH-4502C. Rangkaian ini akan melakukan pengukuran suhu dan pH dalam suatu sistem dengan menggunakan ESP32 sebagai pengendali utama.

Gambar 3.1 menunjukkan metode sistem dari sensor suhu DS18B20 dan sensor pH E-201 terhubung ke ESP-32 lalu input yang diperoleh dari kedua sensor akan di teruskan di web agar diproses dan diprogramkan dengan LSTM lalu untuk hasil prediksi dan monitoring dapat dikirim ke mobile dengan aplikasi whatsapps.

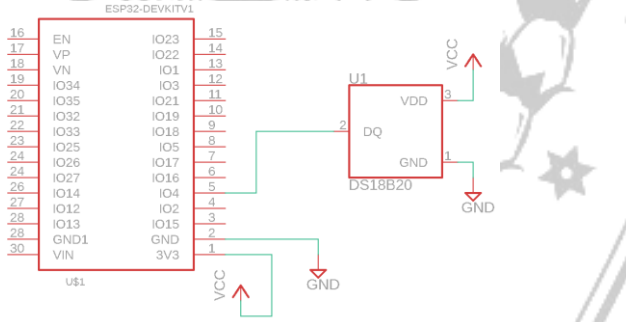
3.1 SENSOR SUHU DSB18B20

Rangkaian sensor suhu DS18B20 dengan ESP32 melalui pin IO4 merupakan kombinasi antara sensor suhu digital DS18B20 dan mikrokontroler ESP32. Sensor suhu DS18B20 adalah sensor suhu digital

yang dapat mengukur suhu dengan akurasi tinggi dan kemampuan komunikasi satu kawat. Sementara itu, ESP32 adalah modul mikrokontroler yang memiliki kemampuan Wifi dan Bluetooth.

Dalam rangkaian ini, pin IO4 pada ESP32 digunakan sebagai pin koneksi antara ESP32 dan sensor suhu DS18B20. Pin ini akan digunakan sebagai jalur komunikasi satu kawat (One-Wire) untuk menghubungkan ESP32 dengan sensor suhu DS18B20. Berikut adalah langkah-langkah umum untuk menghubungkan rangkaian sensor suhu DS18B20 dengan ESP32 melalui pin IO4:

1) Menghubungkan pin GND (ground) pada sensor suhu DS18B20 ke pin GND pada ESP32 untuk menyamakan ground kedua perangkat.
2) Menghubungkan pin VCC pada sensor suhu DS18B20 ke pin 3.3V atau 5V pada ESP32 untuk memberikan daya ke sensor suhu.
3) Menghubungkan pin IO4 pada ESP32 ke pin DQ (data) pada sensor suhu DS18B20. Ini adalah pin komunikasi satu kawat yang akan digunakan untuk mentransmisikan data suhu dari sensor ke ESP32.



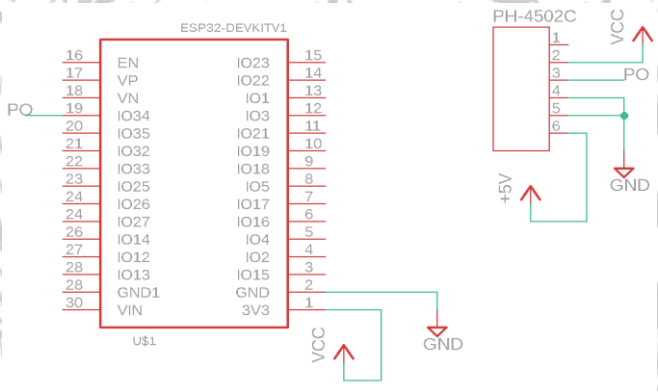
Gambar 3.2 Rangkaian ESP-32 dan Sensor Suhu DS18B20

3.2 SENSOR PH E-201

Rangkaian sensor pH-4502C dengan ESP32 melalui pin IO34 menghubungkan sensor pH-4502C untuk mengukur tingkat pH dan modul mikrokontroler ESP32. ESP32 adalah mikrokontroler yang memiliki kemampuan Wifi dan Bluetooth, sementara pH-4502C adalah sensor pH analog yang digunakan untuk mengukur tingkat pH dalam larutan.

Dalam rangkaian ini, pin IO34 pada ESP32 digunakan sebagai pin koneksi antara ESP32 dan sensor pH-4502C. Pin ini akan digunakan sebagai input analog untuk membaca sinyal pH dari sensor. Berikut adalah langkah-langkah umum untuk menghubungkan rangkaian sensor pH-4502C dengan ESP32 melalui pin IO34:

1) Hubungkan pin GND (ground) pada sensor pH-4502C ke pin GND pada ESP32 untuk menyamakan ground kedua perangkat.
2) Hubungkan pin VCC pada sensor pH-4502C ke pin 3.3V atau 5V pada ESP32 untuk memberikan daya ke sensor pH.
3) Hubungkan pin OUT pada sensor pH-4502C ke pin IO34 pada ESP32. Pin ini akan digunakan untuk mengirimkan sinyal pH analog dari sensor ke mikrokontroler.



Gambar 3.3 Rangkaian ESP-32 dan Sensor pH-4502C

3.3 ESP32

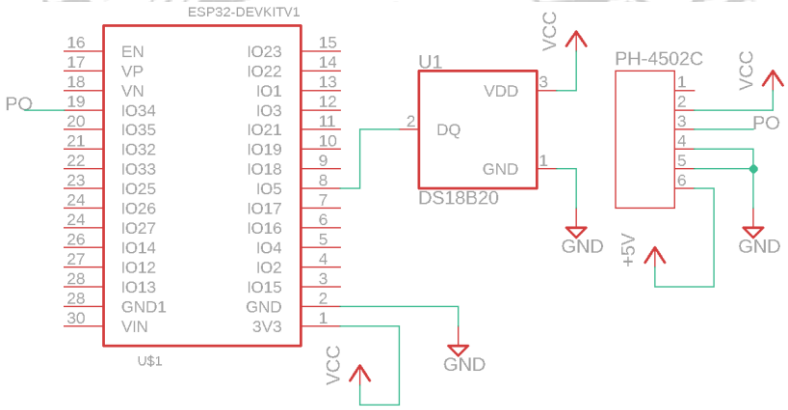
Rangkaian gabungan antara sensor pH dan suhu dengan ESP32 memungkinkan pengukuran dan pemantauan tingkat pH dan suhu dalam suatu sistem menggunakan mikrokontroler. Rangkaian ini memungkinkan ESP32 untuk membaca data dari kedua sensor dan mengintegrasikannya dalam aplikasi yang diinginkan.

Arduino IDE menyediakan berbagai fungsi dan library yang mempermudah penggunaan komponen elektronik dan komunikasi dengan mikrokontroler, seperti ESP32. Dengan menggunakan Arduino IDE, pengguna dapat memprogram ESP32 untuk membaca data suhu dari sensor DS18B20 dan data pH dari sensor pH-4502C,

serta melakukan pemrosesan data dan mengambil tindakan berdasarkan nilai-nilai yang terukur.

Sensor pH, seperti pH E-201, dihubungkan ke ESP32 melalui koneksi fisik yang mencakup penyesuaian ground (GND), daya (VCC), dan output analog (OUT) pada sensor pH. Sementara itu, sensor suhu, seperti DS18B20, juga dihubungkan ke ESP32 melalui koneksi fisik yang sesuai, misalnya menggunakan protokol OneWire.

Setelah melakukan koneksi fisik, program pada ESP32 dapat diatur untuk membaca data dari kedua sensor tersebut. Melalui pin-pin yang ditentukan pada ESP32, seperti IO34 untuk sensor pH dan pin lainnya untuk sensor suhu, ESP32 dapat membaca sinyal analog dan/atau digital dari kedua sensor tersebut.



Gambar 3.4 Rangkaian ESP-32, Sensor Suhu DS18B20 dan Sensor pH-4502C

3.3.1 Program Sensor suhu DS18B20 pada ESP32

Sensor suhu DS18B20 adalah sensor suhu digital yang menggunakan antarmuka OneWire untuk dihubungkan dengan mikrokontroler seperti Arduino atau ESP32. Dalam program menggunakan sensor ini, langkah-langkah umumnya meliputi inisialisasi antarmuka OneWire, pencarian alamat sensor, pengiriman perintah baca suhu, pembacaan dan penguraian data suhu, serta penggunaan data suhu untuk tindakan lebih lanjut. Sensor suhu DS18B20 memberikan kemudahan dalam membaca suhu dengan presisi tinggi dan dapat digunakan dalam berbagai aplikasi pemantauan dan pengendalian suhu.

```
#include <OneWire.h>

#include <DallasTemperature.h>



const int oneWireBus = 4;

OneWire oneWire(oneWireBus);



DallasTemperature sensors(&oneWire);

void setup() {

  Serial.begin(115200);

  sensors.begin();

}

void loop() {

  sensors.requestTemperatures();

  float temperatureC = sensors.getTempCByIndex(0);

  Serial.print(temperatureC);

  Serial.println("°C");

  delay(5000);

}
```

Pembacaan sensor suhu menggunakan dua library utama yakni library OneWire dan DallasTemperature. Library OneWire adalah library yang digunakan untuk komunikasi data sedangkan library DallasTemperature digunakan untuk membaca data suhu dari sensor. Sensor diletakkan di pin GPIO 4 melalui variabel konstanta oneWireBus. Kemudian komunikasi diaktifkan dengan memanggil fungsi `oneWire(oneWireBus)`. Selanjutnya aktivasi pembacaan sensor suhu melalui fungsi `sensors(&oneWire)`. Jika ingin membaca data suhu dalam satuan *Celcius* dapat dilakukan dengan memanggil fungsi `sensors.getTempCByIndex(0)`, di mana 0 adalah indeks sensor jika terhubung

lebih dari satu sensor suhu. Pemberian Jeda atau *delay* selama 5000 *milliseconds* atau 5 detik digunakan untuk memberi jeda sensor sebelum pembacaan berikutnya.

3.3.2 Program Sensor PH E-201 pada ESP32

Sensor pH E-201 adalah sensor pH analog yang umumnya digunakan dalam aplikasi pemantauan dan pengukuran pH. Untuk mengintegrasikan sensor ini dengan mikrokontroler seperti Arduino atau ESP32, langkah-langkah umum dalam program meliputi inisialisasi pin analog, pembacaan nilai tegangan pH, konversi tegangan menjadi nilai pH, dan penggunaan nilai pH untuk tindakan lebih lanjut, seperti penampilan data atau pengendalian sistem berdasarkan pH. Sensor pH4502C dapat memberikan informasi penting tentang tingkat keasaman atau alkalinitas.

```
int pHSense = 39;
int samples = 10;
float adc_resolution = 2048.0;


void setup()
{
    Serial.begin(115200);
    delay(100);
}
float ph (float voltage) {
    return 7 + ((2.5 - voltage) / 0.18);
}


void loop()
```

```
{
    int measurings=0;
    for (int i = 0; i < samples; i++)
    {
        measurings += analogRead(pHSense);
        delay(10);
    }
    float voltage = 5 / adc_resolution * measurings/samples;
    Serial.print("pH= ");
    Serial.println(ph(voltage));
    delay(3000);
}
```

Keluaran dari sensor pH merupakan data tipe analog. Sensor dihubungkan ke pin GPIO 39 pada mikrokontroler diberi nama variabel `pHSense`. Variabel `samples` berfungsi untuk menampung nilai jumlah sampel pH yang diambil. Baris program `adc_resolution = 2048.0` menunjukkan nilai resolusi bit dari tipe mikrokontroler ESP-32. Fungsi `ph (float voltage)` mengembalikan nilai pH yang terbaca berdasarkan perhitungan nilai tegangan dan nilai ADC mikrokontroler.

3.3.3 Program Mengirimkan Data dari Mikrokontroler ke Whatsapp

Program ini menggunakan library ThingESP untuk melakukan pengujian koneksi ESP32 dengan aplikasi WhatsApp. Program melakukan konfigurasi WiFi dan inisialisasi perangkat, dan kemudian secara periodik mengirim pesan melalui WhatsApp untuk memeriksa apakah koneksi berhasil. Program juga menerima pesan balasan dari WhatsApp sebagai tanda bahwa koneksi berhasil. Dengan menggunakan ESP32 dan ThingESP, program ini memungkinkan pengguna untuk menguji koneksi dan komunikasi antara ESP32 dan aplikasi WhatsApp.

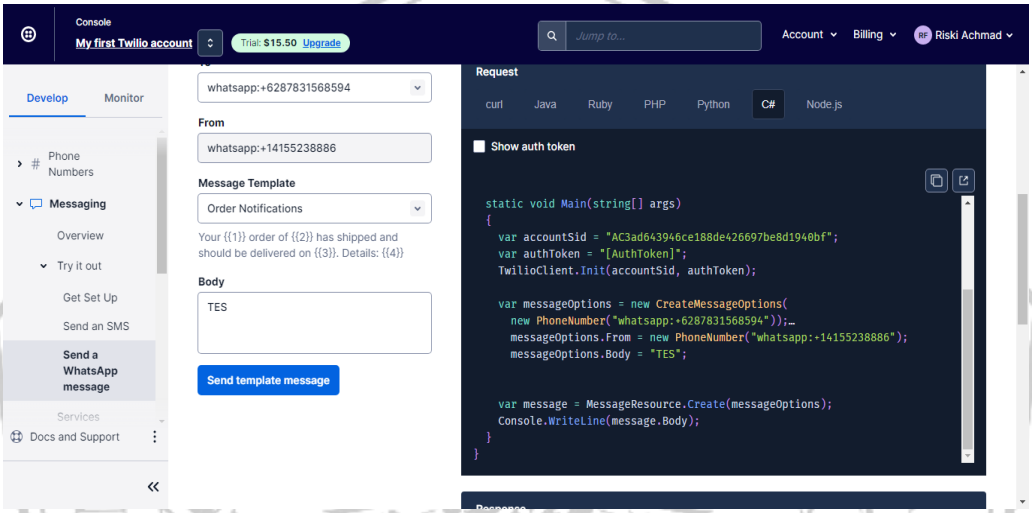
```
#include <WiFi.h>
#include <ThingESP.h>

ThingESP32 thing("Bannna", "ESP32", "1234567890");
int LED = 2;
unsigned long previousMillis = 0;
const long INTERVAL = 6000;
void setup()
{
    Serial.begin(115200);
    pinMode(LED, OUTPUT);
    thing.SetWiFi("NUSANTARA", "Qwerty123");
    thing.initDevice();
}
String HandleResponse(String query)
{
    if (query == "led on")
    {
        digitalWrite(LED, 1);
        return "Done: LED Turned ON";
    }
    else if (query == "led off")
    {
        digitalWrite(LED, 0);
```

```
        return "Done: LED Turned OFF";
    }
    else if (query == "led status")
        return digitalRead(LED) ? "LED is ON" : "LED is OFF";
    else
        return "Your query was invalid..";
}
void loop()
{
    if (millis() - previousMillis >= INTERVAL)
    {
        previousMillis = millis();
        String msg = digitalRead(LED) ? "LED is ON" : "LED is
OFF";
        thing.sendMsg("+6287863780743", msg);
    }
    thing.Handle();
}
```

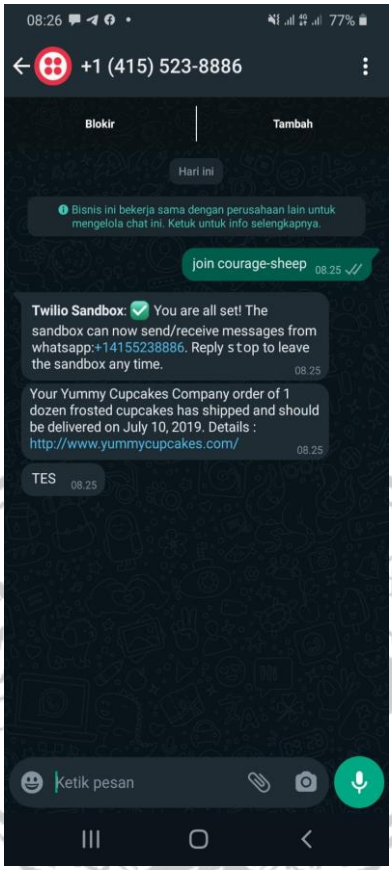
Program tersebut berfungsi melakukan percobaan komunikasi mikrokontroler dengan Whatsapp. Library yang digunakan ialah `WiFi.h` dan `ThingESP.h`. Library WiFi berfungsi menghubungkan mikrokontroler dengan jaringan internet melalui koneksi WiFi. Library ThingESP pada algoritma program tersebut berfungsi sebagai penghubung mikrokontroler dengan platform Twilio agar dapat terhubung ke Whatsapp.

3.4 MOBILE

Komunikasi antara mikrokontroler dengan platform Whatsapp dilakukan melalui perantara aplikasi pihak ketiga yakni platform Twilio. Pengguna mendaftarkan nomor Whatsapp ke platform tersebut untuk mendapatkan kode token agar mikrokontroler dapat berkomunikasi dengan Whatsapp. Pendaftaran dilakukan melalui web resmi twilio.com.



Gambar 3.5 Tampilan Console Konfigurasi Akun Twilio



Gambar 3.6 Tampilan Hasil Tes Pengiriman Chat

Perintah untuk menjalankan sistem terdiri dari 3 jenis perintah. Perintah "Join" diikuti dengan kode token untuk mengaktifkan koneksi Whatsapp API dengan akun Twilio. Perintah "cek tambak" berfungsi untuk mendapatkan nilai monitoring suhu dan pH realtime. Perintah "prediksi" untuk mengambil data prediksi terakhir yang dilakukan cloud computing dan tersimpan di spreadsheet.

3.6 WEB

Metode dalam web yang digunakan adalah metode Metode *Long Short Term Memory* (LSTM) dan Pembuatan program *Deep Learning LSTM* menggunakan *platform* Google Collaboratory dengan basis pemrograman *Python*. Pada tahap ini algoritma program memiliki beberapa tujuan diantaranya membaca data sensor suhu dan pH dari *google spreadsheet*, melakukan normalisasi data, membuat data training dan testing, membuat model dan melatihnya, membuat prediksi data dan menampilkan perbandingan data asli dengan data prediksi.

Metode LSTM digunakan untuk memodelkan dan memprediksi data time series dengan mempertahankan informasi jangka panjang dan mengatasi masalah gradien yang meledak atau menghilang. Dalam proses pelatihan data menggunakan LSTM, beberapa parameter yang perlu diinisialisasi adalah:

1) Jumlah Hidden State atau Hidden Layer

LSTM terdiri dari satu atau beberapa lapisan LSTM yang disebut sebagai hidden state atau hidden layer. Jumlah hidden state yang digunakan dapat bervariasi tergantung pada kompleksitas masalah yang dihadapi. Setiap hidden state LSTM terdiri dari beberapa unit LSTM (neuron) yang berperan dalam mempelajari pola-pola temporal data time series. Jumlah hidden state akan ditentukan berdasarkan hasil iterasi yang akan dihasilkan.

2) Jumlah Nilai Error dengan Root Mean Squared Error (RMSE)

Selama proses pelatihan LSTM, digunakan matrik evaluasi untuk mengukur sejauh mana prediksi model kita berbeda dengan nilai sebenarnya. Salah satu metrik evaluasi yang umum digunakan adalah Root Mean Squared Error (RMSE). RMSE mengukur rata-rata perbedaan antara nilai prediksi dan nilai sebenarnya dalam skala yang sama dengan data asli. Tujuan utamanya adalah untuk mendapatkan nilai RMSE yang lebih rendah, yang menunjukkan tingkat kesalahan yang lebih kecil dalam prediksi. Metode RMSE dipilih karena memiliki beberapa keunggulan diantaranya:

a. Skala yang Sama dengan Data Asli

RMSE memberikan perhitungan error yang memiliki skala yang sama dengan data asli. Dalam kasus MSE, nilai error yang dihasilkan adalah kuadrat dari satuan data, yang sulit untuk diinterpretasikan secara langsung. Dengan RMSE, hasil evaluasi error akan memiliki satuan yang sama dengan data asli, sehingga lebih mudah dipahami dan dibandingkan dengan data aktual.

b. Sensitivitas terhadap Outlier

RMSE memberikan sensitivitas yang lebih besar terhadap outlier dalam data. Karena RMSE melibatkan operasi akar kuadrat, nilai error yang dihasilkan akan lebih besar ketika ada outlier yang signifikan dalam data. Ini memungkinkan

RMSE untuk memberikan penilaian yang lebih akurat terhadap prediksi yang terlalu jauh dari data asli.

c. Interpretasi yang Lebih Intuitif

RMSE dapat diinterpretasikan sebagai rata-rata jarak antara prediksi dan data asli dalam satuan yang sama dengan data asli. Ini membuatnya lebih mudah dipahami secara intuitif dan memberikan pemahaman yang lebih baik tentang kualitas prediksi.

3) Jumlah Epoch Maksimum:

Epoch mengacu pada satu kali iterasi penuh saat seluruh set data latih digunakan untuk melatih model. Jumlah epoch maksimum merupakan jumlah total iterasi yang akan dilakukan selama pelatihan. Jumlah epoch maksimum dapat ditentukan sebelumnya, dan setelah mencapai jumlah epoch tersebut, pelatihan akan dihentikan. Jumlah epoch yang tepat dapat bervariasi tergantung pada kompleksitas masalah dan ukuran data, dan biasanya dipilih melalui eksperimen untuk mencapai keseimbangan antara waktu pelatihan dan performa model.

Selain parameter-parameter di atas, ada juga parameter lain yang perlu diinisialisasi, seperti learning rate (tingkat pembelajaran), batch size (ukuran batch data), dan algoritma optimasi yang digunakan. Parameter-parameter ini akan mempengaruhi bagaimana model LSTM dipelajari dan diperbarui selama proses pelatihan. Pemilihan parameter yang tepat sangat penting untuk mencapai hasil prediksi yang akurat dan optimal.

a. Training Dataset LSTM

Proses training Long Short Term Memory (LSTM) adalah sebagai berikut:

Menghitung fungsi *Gate* secara berurutan mulai input gates, forget gates, cell states, hidden states dan output gates. Proses perhitungan tersebut dapat dilakukan menggunakan rumus - rumus berikut:

- **Input Gate (i_t)**

$$i_t = \sigma(W_i \cdot [h_{(t-1)}, x_t] + b_i)$$

σ adalah fungsi aktivasi sigmoid, W_i dan b_i adalah parameter yang harus dipelajari, h_(t-1) adalah output hidden state pada timestep sebelumnya, dan x_t adalah input pada timestep saat ini.

- **Forget Gate (f_t)**

$$f_t = \sigma(W_f \cdot [h_{(t-1)}, x_t] + b_f)$$

f_t mengendalikan seberapa banyak informasi yang harus diabaikan dari sel memori sebelumnya. W_f dan b_f adalah parameter yang harus dipelajari.

- **Cell State (C_t)**

$$C_t = f_t * C_{(t-1)} + i_t * \tanh(W_c \cdot [h_{(t-1)}, x_t] + b_c)$$

C_t adalah sel memori pada timestep saat ini. * adalah operasi perkalian elemen-wise, dan tanh adalah fungsi aktivasi tangen hiperbolik.

- **Output Gate (o_t)**

$$o_t = \sigma(W_o \cdot [h_{(t-1)}, x_t] + b_o)$$

o_t mengendalikan berapa banyak informasi yang harus diungkapkan dari sel memori saat ini. W_o dan b_o adalah parameter yang harus dipelajari.

- **Hidden State (h_t)**

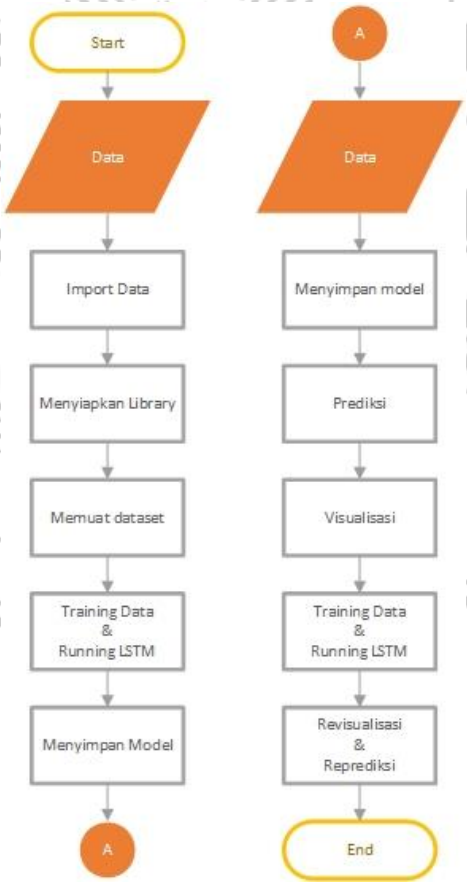
$$h_t = o_t * \tanh(C_t)$$

h_t adalah output hidden state pada timestep saat ini.

Dalam LSTM, setiap gate (input, forget, output) dan sel memori (cell state) memiliki parameter bobot dan bias yang harus dipelajari selama proses pelatihan. Bobot dan bias ini akan disesuaikan untuk meminimalkan fungsi loss saat melatih model.

Setelah menemukan jumlah *epoch* maksimum yang optimal dan menghasilkan RMSE yang optimal pula, selanjutnya melakukan testing data *realtime* dari mikrokontroler, untuk kemudian diproses dan diprediksi suhu dan pH 1 jam kedepan. Hasil baca sensor dari mikrokontroler akan menjadi acuan apabila terjadi data yang diluar batas normal, sistem akan mengirimkan notifikasi peringatan melalui *Whatsapp*.

Langkah-langkah yang dilakukan dalam membuat software system dimulai dengan mengimport dataset ke notebook dan melakukan preposisi data serta pelatihan dan prediksi menggunakan model LSTM.



Gambar 3.7 Flowchart proses pembuatan *software system*

a. Import dataset ke notebook

Dataset berasal dari hasil pengukuran suhu dan pH air tambak yang diukur langsung oleh petani tambak udang secara berkala. Dataset ini kemudian diunggah ke akun *google drive* yang selanjutnya di *import* ke *google collabs* melalui perintah berikut:

```
from google.colab import auth

from google.colab import drive

import gspread

from google.auth import default

creds, _ = default()

gc = gspread.authorize(creds)


# Melakukan otentikasi

auth.authenticate_user()


# Menghubungkan Google Drive ke sesi Colab

drive.mount('/content/drive')
```

Library yang dibutuhkan meliputi `google.colab` dan `google.auth`. Dua library tersebut yang digunakan untuk melakukan autentikasi akun *Colab* dengan *Google Drive* dan *Spreadsheet*.

b. Menyiapkan library preposisi, prediksi, dan visualisasi data

Sebelum melakukan preposisi hingga visualisasi data, beberapa library yang diperlukan perlu dipanggil sebelum melakukan eksekusi program. Library yang dimaksud sebagai berikut.

```
import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, LSTM

from keras.models import load_model

import matplotlib.pyplot as plt
```

Library yang dibutuhkan meliputi library pengolah data yakni `numpy`, `pandas`, dan `sklearn`. Library tersebut yang akan melakukan proses preposisi dan normalisasi data. Library lainnya ialah `keras` yakni library yang dapat melakukan operasi *deep learning* termasuk LSTM. Data hasil prediksi divisualisasikan menggunakan library `matplotlib`.

c. Memuat Dataset

Dataset dimuat dari Google Drive ke Colab menggunakan library `pandas` yang diimpor sebagai `pd` seperti pada program berikut.

```
# Memuat data

data = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/Data_Temp_pH.csv', encoding='latin1')



# Mengambil kolom suhu

suhu = data['Temp'].values
```

File dataset diimpor ke variabel data menggunakan perintah `pd.read_csv` sesuai dengan jenis file dataset yakni file `Data_Temp_pH.csv`. Kemudian nilai dari data pada kolom suhu diambil dan disimpan ke variabel suhu melalui perintah `suhu = data['Temp'].values`.

d. Melakukan Preposisi Data

```
# Mengubah bentuk data menjadi array 2 dimensi

suhu = suhu.reshape(-1, 1)


# Melakukan normalisasi data

scaler_suhu = MinMaxScaler(feature_range=(0, 1))

suhu_norm = scaler_suhu.fit_transform(suhu)


# Membuat data latih

lookback = 12  # Jumlah timestep sebelum waktu yang ingin
diprediksi

X_train_suhu = []

y_train_suhu = []

for i in range(lookback, len(suhu_norm)):

   X_train_suhu.append(suhu_norm[i-lookback:i, 0])

   y_train_suhu.append(suhu_norm[i, 0])

X_train_suhu, y_train_suhu = np.array(X_train_suhu),
np.array(y_train_suhu)


# Reshape data menjadi bentuk yang diterima oleh LSTM (jumlah
sampel, jumlah timestep, jumlah fitur)

X_train_suhu = np.reshape(X_train_suhu,
(X_train_suhu.shape[0], X_train_suhu.shape[1], 1))
```

Implementasi dari preposisi data digunakan dalam pelatihan model LSTM (Long Short-Term Memory) pada tugas prediksi suhu maupun pH. Berikut adalah penjelasan langkah-langkahnya:

1. Mengubah Bentuk Data menjadi Array 2 Dimensi:

```
suhu = suhu.reshape(-1, 1)
```

Langkah ini dilakukan untuk mengubah bentuk data suhu menjadi array 2 dimensi, dengan dimensi pertama (-1) menyesuaikan ukuran data yang ada dan dimensi kedua (1) merepresentasikan satu fitur (suhu).

2. Melakukan Normalisasi Data:

```
scaler_suhu = MinMaxScaler(feature_range=(0, 1))

suhu_norm = scaler_suhu.fit_transform(suhu)
```

Normalisasi data dilakukan untuk mengubah rentang data menjadi 0 hingga 1. Pada contoh ini, digunakan MinMaxScaler untuk mengubah setiap nilai suhu menjadi dalam rentang tersebut.

3. Membuat Data Latih:

```
lookback = 12

X_train_suhu = []

y_train_suhu = []
```

Pada langkah ini, kita menentukan jumlah timestep sebelum waktu yang ingin diprediksi (lookback) dan membuat dua list kosong untuk menyimpan data latih (input dan output). Menggunakan loop, setiap sampel data diambil dari `suhu_norm`, dimulai dari indeks lookback. Untuk setiap sampel, diambil sejumlah timestep sebelumnya (berdasarkan lookback) dan menyimpannya dalam `X_train_suhu`. Nilai suhu pada timestep berikutnya (i) akan menjadi nilai target (`y_train_suhu`). Misalnya, jika lookback = 12, maka untuk sampel ke-13,

`X_train_suhu` akan berisi 12 nilai suhu sebelumnya dan `y_train_suhu` akan berisi suhu pada timestep ke-13.

4. Reshape Data:

```
X_train_suhu = np.reshape(X_train_suhu, (X_train_suhu.shape[0],
                    X_train_suhu.shape[1], 1))
```

Langkah ini dilakukan untuk mengubah bentuk data latih menjadi sesuai dengan format yang diterima oleh model LSTM. Reshape dilakukan menjadi bentuk tiga dimensi dengan dimensi pertama menunjukkan jumlah sampel, dimensi kedua menunjukkan jumlah timestep, dan dimensi ketiga menunjukkan jumlah fitur (dalam contoh ini, hanya satu fitur suhu).

Dengan langkah-langkah di atas, data suhu diubah menjadi format yang dapat digunakan untuk melatih model LSTM.

e. Training data dan running LSTM

Proses training data dilakukan dengan menggunakan metode optimasi adam dan *mean squared error* (mse). Proses training dilakukan dengan 5 variasi nilai epoch sebanyak 100, 150, 200, 250, dan 300. Baris program *Training Data* menggunakan kode berikut.

```
# Membuat model LSTM

model = Sequential()

model.add(LSTM(50, return_sequences=True,
input_shape=(lookback, 1)))

model.add(LSTM(50))

model.add(Dense(1))
```

```
model.compile(optimizer='adam', loss='mse')


# Melatih model
model.fit(X_train_suhu, y_train_suhu, epochs=100,
batch_size=32)
```

Program tersebut merupakan implementasi pembuatan dan pelatihan model LSTM (Long Short-Term Memory) menggunakan framework Keras. Berikut adalah penjelasan langkah-langkahnya:

1. Membuat Model LSTM

```
model = Sequential()
model.add(LSTM(50, return_sequences=True,
input_shape=(lookback, 1)))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

Pada langkah ini, dibuat objek model menggunakan `Sequential()` dari Keras. Sequential model digunakan untuk membuat model secara berurutan, dengan lapisan-lapisan ditambahkan satu per satu.

- Pertama, kita menambahkan lapisan LSTM pertama dengan 50 unit dan `return_sequences = True` untuk menghasilkan keluaran sekuensial.
- Selanjutnya, kita menambahkan lapisan LSTM kedua juga dengan 50 unit.

- Terakhir, kita menambahkan lapisan Dense (lapisan fully connected) dengan satu neuron untuk menghasilkan output prediksi suhu.

Setelah menambahkan lapisan-lapisan, kita mengompilasi model dengan optimizer 'adam' dan loss function 'mse' (Mean Squared Error).

2. Melatih Model

```
model.fit(X_train_suhu, y_train_suhu, epochs=100, batch_size=32)
```

Pada langkah ini, kita melatih model menggunakan metode fit() pada data latih yang telah dipreposisi sebelumnya.

- `X_train_suhu` adalah input data latih yang telah direshape sebelumnya dengan bentuk tiga dimensi yang sesuai dengan model.
- `y_train_suhu` adalah target data latih (output yang ingin diprediksi).
- `epochs` menentukan jumlah iterasi pelatihan.
- `batch_size` adalah jumlah sampel yang digunakan dalam satu iterasi pelatihan.

Selama pelatihan, model akan memperbarui parameter-nya dengan mengoptimalkan loss function (mse) menggunakan optimizer (adam) untuk mencapai hasil yang lebih baik.

Dengan langkah-langkah di atas, model LSTM telah dibuat dan dilatih dengan data latih yang telah dipreposisi. Model ini digunakan untuk melakukan prediksi suhu pada data baru.

f. Menyimpan Model

Model yang telah dilatih disimpan ke dalam file dengan menggunakan metode save() dari Kelas.

```
# Menyimpan model ke file

model.save('model_suhu.h5')
```

Pada langkah ini, model yang telah dilatih disimpan ke dalam file dengan nama "model_suhu.h5". Metode save() digunakan untuk menyimpan seluruh arsitektur dan parameter (weight) dari model ke dalam file yang dapat digunakan kembali di masa depan.

Format file yang digunakan dalam contoh ini adalah HDF5 (H5). File tersebut akan berisi informasi mengenai struktur model (lapisan-lapisan, tipe lapisan, konfigurasi), parameter dari setiap lapisan (weight, bias), dan pengaturan optimasi yang digunakan. Proses menyimpan model ke dalam file dapat memudahkan pengguna memuat kembali model tersebut di waktu yang akan datang tanpa perlu melatih ulang dari awal. Model yang telah disimpan dapat dimuat kembali menggunakan fungsi `load_model()` dari Keras.

g. Melakukan Prediksi

Prediksi suhu 1 jam kedepan menggunakan model yang telah dilatih dilakukan menggunakan perintah program berikut.

```
# Memprediksi suhu 1 jam ke depan

last_sequence_suhu = suhu_norm[-lookback:].reshape(1, -1, 1)

pred_suhu_norm = model.predict(last_sequence_suhu)

pred_suhu = scaler_suhu.inverse_transform(pred_suhu_norm)
```

```
last_sequence_suhu = suhu_norm[-lookback:].reshape(1, -1, 1)
```

Langkah ini mengambil sekuens (urutan) suhu terakhir dengan panjang yang sama dengan `lookback` (jumlah timestep sebelum waktu yang ingin diprediksi). Data suhu ini kemudian di-`reshape` menjadi bentuk tiga dimensi yang diterima oleh model.

- `suhu_norm[-lookback:]` mengambil sekuens suhu terakhir dari data suhu yang telah dinormalisasi.
- `reshape(1, -1, 1)` mengubah bentuk data menjadi tiga dimensi dengan dimensi pertama adalah jumlah sampel (dalam hal ini hanya 1 sampel), dimensi kedua adalah jumlah timestep (sama dengan `lookback`), dan dimensi ketiga adalah jumlah fitur (dalam hal ini hanya 1 fitur suhu).

```
pred_suhu_norm = model.predict(last_sequence_suhu)
```

Langkah ini menggunakan model yang telah dilatih sebelumnya untuk memprediksi nilai suhu berikutnya berdasarkan `last_sequence_suhu`.

- `model.predict()` digunakan untuk memprediksi output berdasarkan input yang diberikan. Dalam hal ini, `last_sequence_suhu` digunakan sebagai input.

```
pred_suhu = scaler_suhu.inverse_transform(pred_suhu_norm)
```

Langkah ini mengembalikan nilai suhu yang telah diprediksi ke dalam rentang awal sebelum normalisasi.

- `scaler_suhu.inverse_transform()` digunakan untuk mengembalikan nilai suhu yang dinormalisasi ke dalam rentang awal. Ini dilakukan dengan membalikkan transformasi yang telah dilakukan sebelumnya.

Dengan langkah-langkah tersebut, dapat diperoleh prediksi suhu 1 jam kedepan (berdasarkan `lookback`) dengan menggunakan model LSTM yang telah dilatih sebelumnya. Nilai prediksi suhu tersebut disimpan dalam variabel `pred_suhu`.

h. Melakukan Visualisasi

Dataset dan hasil prediksi dapat divisualisasikan ke dalam bentuk grafik agar mudah dipahami. Berikut program untuk mengeksekusi proses plotting.

```
# Plot hasil prediksi suhu
plt.plot(suhu, label='Data Asli')
plt.plot(len(suhu)-1, pred_suhu[-1,:][0], 'ro',
label='Prediksi')
plt.xlabel('Waktu')
plt.ylabel('Suhu')
plt.title('Prediksi Suhu 1 Jam ke Depan')
plt.legend()
plt.show()


# Plot hasil prediksi pH
plt.plot(ph, label='Data Asli')
plt.plot(len(ph)-1, pred_ph[-1,:][0], 'ro', label='Prediksi')
plt.xlabel('Waktu')
plt.ylabel('pH')
plt.title('Prediksi pH 1 Jam ke Depan')
plt.legend()
plt.show()
```

Pertama, kode melakukan plot grafik untuk hasil prediksi suhu,

- `plt.plot(suhu, label='Data Asli')` digunakan untuk menggambar garis yang menunjukkan data asli suhu dari waktu ke waktu.
- `plt.plot(len(suhu)-1, pred_suhu[-1,:][0], 'ro', label='Prediksi')` digunakan untuk menandai titik merah `('ro')` yang merupakan hasil prediksi suhu untuk 1 jam ke depan.
- `plt.xlabel('Waktu')` dan `plt.ylabel('Suhu')` digunakan untuk memberi label sumbu x dan y pada grafik.
- `plt.title('Prediksi Suhu 1 Jam ke Depan')` digunakan untuk memberi judul pada grafik.
- `plt.legend()` digunakan untuk menampilkan legenda dengan label `'Data Asli'` dan `'Prediksi'`.
- `plt.show()` digunakan untuk menampilkan grafik suhu.

Selanjutnya, kode melakukan plot grafik untuk hasil prediksi pH,

- `plt.plot(ph, label='Data Asli')` digunakan untuk menggambar garis yang menunjukkan data asli pH dari waktu ke waktu.
- `plt.plot(len(ph)-1, pred_ph[-1,:][0], 'ro', label='Prediksi')` digunakan untuk menandai titik merah `('ro')` yang merupakan hasil prediksi pH untuk 1 jam ke depan.
- `plt.xlabel('Waktu')` dan `plt.ylabel('pH')` digunakan untuk memberi label sumbu x dan y pada grafik.
- `plt.title('Prediksi pH 1 Jam ke Depan')` digunakan untuk memberi judul pada grafik.
- `plt.legend()` digunakan untuk menampilkan legenda dengan label `'Data Asli'` dan `'Prediksi'`.
- `plt.show()` digunakan untuk menampilkan grafik pH.

Grafik-grafik tersebut memberikan visualisasi hasil prediksi suhu dan pH dalam bentuk grafik yang memudahkan pemahaman dan analisis data.

i. Memuat ulang model dan melakukan prediksi ulang

Program berikut digunakan untuk memuat kembali model yang telah disimpan dan melakukan prediksi ulang.

```
# Memuat model dari file
model_suhu = load_model('model_suhu.h5')

# Memprediksi suhu 1 jam ke depan
last_sequence_suhu = suhu_norm[-lookback:].reshape(1, -1, 1)
pred_suhu_norm = model_suhu.predict(last_sequence_suhu)
pred_suhu = scaler_suhu.inverse_transform(pred_suhu_norm)

# Memuat model dari file
model_ph = load_model('model_ph.h5')

# Memprediksi pH 1 jam ke depan
last_sequence_ph = ph_norm[-lookback:].reshape(1, -1, 1)
pred_ph_norm = model_ph.predict(last_sequence_ph)
pred_ph = scaler_ph.inverse_transform(pred_ph_norm)

#### TAMPILKAN HASIL ####

print("Prediksi suhu 1 jam ke depan:")
print(pred_suhu[-1,:][0])
```

```
print("Prediksi pH 1 jam ke depan:")
print(pred_ph[-1,:][0])

#### KIRIM DATA PREDIKSI KE SPREADSHEET ####

# Buka spreadsheet
worksheet = gc.open('Data_Temp_pH').get_worksheet(1)

# worksheet.update_cell(row, col, value)
worksheet.update_cell(2, 1, str(pred_suhu[-1,:][0]))
worksheet.update_cell(2, 2, str(pred_ph[-1,:][0]))
```

- Load Model

Pertama, program memuat model yang telah disimpan sebelumnya dari file '`model_suhu.h5`' dan '`model_ph.h5`'. Model ini kemungkinan besar sudah dilatih sebelumnya dengan menggunakan data historis untuk mempelajari pola dan hubungan antara input (misalnya suhu dan pH sebelumnya) dengan output (suhu dan pH di masa depan).

- Memprediksi Suhu 1 Jam ke Depan

Berikutnya, program menggunakan model suhu yang telah dimuat untuk memprediksi suhu 1 jam ke depan. Untuk melakukan ini, data suhu terakhir sepanjang periode '`lookback`' diambil dan diformat ulang menjadi bentuk yang sesuai untuk dimasukkan ke dalam model. Kemudian, prediksi suhu dinormalisasi

diubah kembali menjadi skala aslinya menggunakan objek `'scaler_suhu'`. Hasil prediksi suhu kemudian dicetak.

- Memprediksi pH 1 Jam ke Depan

Langkah ini mirip dengan langkah sebelumnya, tetapi menggunakan model pH yang telah dimuat. Data pH terakhir sepanjang periode `'lookback'` diambil dan diformat ulang. Kemudian, prediksi pH dinormalisasi diubah kembali menjadi skala aslinya menggunakan objek `'scaler_ph'`. Hasil prediksi pH dicetak.

- Tampilkan Hasil

Setelah melakukan prediksi, program mencetak hasil prediksi suhu dan pH untuk 1 jam ke depan.

- Kirim Data Prediksi ke Spreadsheet

Langkah terakhir dalam program adalah mengirimkan data prediksi suhu dan pH ke sebuah spreadsheet. Program menggunakan Google Sheets API untuk membuka spreadsheet dengan nama `'Data_Temp_pH'` dan mengakses lembar kerja dengan indeks 1. Kemudian, data prediksi suhu dan pH diperbarui pada sel tertentu dalam lembar kerja menggunakan metode `'update_cell'`.

Jadi, program ini memuat model yang telah dilatih sebelumnya, melakukan prediksi suhu dan pH untuk 1 jam ke depan, menampilkan hasil prediksi, dan mengirimkan data prediksi ke sebuah spreadsheet.

Sistem yang telah dibuat diimplementasikan dengan dilakukan secara lansung pada tambak udang untuk mendapatkan data. Alat diletakkan pada pinggir tambak dengan pengambilan data dilakukan setiap menit dan mengumpulkan data suhu dari sensor Suhu DS18B20 dan data pH dari sensor pH E-201.

```
#include <WiFi.h>

#include <ThingESP.h>

#include <OneWire.h>

#include <DallasTemperature.h>

#include <HTTPClient.h>

#include "time.h"


const char* ntpServer = "pool.ntp.org";

const long  gmtOffset_sec = 25200; //GMT+7 = 25200 seconds

const int   daylightOffset_sec = 0;


RTC_DATA_ATTR int bootCount = 0;


const int potPin = 39; //pin sensor pH

float Value = 0;

float ph    = 0;

float suhu  = 0;

float pred_suhu = 0;

float pred_ph   = 0;


const int oneWireBus = 4; //pin sensor suhu DS18B20


// Setup a oneWire instance to communicate with any OneWire
devices

OneWire oneWire(oneWireBus);
```

```
// Pass our oneWire reference to Dallas Temperature sensor

DallasTemperature sensors(&oneWire);


 ThingESP32 thing("Bannna", "ESP32", "1234567890");


unsigned long previousMillis = 0;

const long INTERVAL = 10000;


String GOOGLE_SCRIPT_ID_POST = "AKfycbwIo-
RfWjvzoLoJLInqhiqKr80qiXrC_oui2a54wgv7GX_TV9f06zvrZV5nw_r_B8yuWQ
";    // change Gscript ID


String GOOGLE_SCRIPT_ID_READ =
"AKfycbyUPCnY1ZXLRhAE4kU6Hg8LKFvch4OfSSqFH7QlxR4V6U9ERBDD1lpI1q3
XDhIYC8frOw";


// Variable to save current epoch time

unsigned long epochTime;


void setup()

{

  Serial.begin(115200);

  sensors.begin();

  pinMode(potPin, INPUT);

  Serial.println ("ESP mulai");

  // if WiFi is down, try reconnecting every CHECK_WIFI_TIME
seconds

  if (WiFi.status() != WL_CONNECTED) {
```

```
    Serial.println("Reconnecting to WiFi...");
    WiFi.disconnect();
    WiFi.reconnect();
  }
  thing.SetWiFi("SSID", "PASSWORD");

  thing.initDevice();
  delay(2500);
  // Init and get the time
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  delay(1000);
}

String HandleResponse(String query) {
  if (query == "cek tambak")
  {
    bacaSensor();
    return "\nNilai suhu : " + String(suhu) + "\nNilai pH : " +
String(ph);
  }

  if (query == "prediksi")
  {
    ReadFromSpreadsheet();
    return "\nPrediksi suhu : " + String(pred_suhu) +
"\nPrediksi pH : " + String(pred_ph);
  }
```

```
  else
    return "Perintah Salah. Tuliskan 'cek tambak' !";
}

void loop()
{
    if (millis() - previousMillis >= INTERVAL)
    {
        previousMillis = millis();
        bacaSensor();
        sendToSpreadsheet();
    }
  thing.Handle();
}
void bacaSensor() {
  sensors.requestTemperatures();
  suhu = sensors.getTempCByIndex(0);
  Value = analogRead(potPin);
  // Serial.print(Value);
  // Serial.print(" | ");
  float voltage = (Value / 4095.0) * 3.0;
  ph = (3.0 * voltage);
  // Serial.println("Suhu:" + String(suhu) + " pH:" +
String(ph));
}
```

```
// Function that gets current epoch time

unsigned long getTime() {

  time_t now;

  struct tm timeinfo;

  if (!getLocalTime(&timeinfo)) {

    //Serial.println("Failed to obtain time");

    return(0);

  }

  time(&now);

  return now;

}

void sendToSpreadsheet()

{

  if (WiFi.status() == WL_CONNECTED) {

    epochTime = getTime();

    Serial.print("Epoch Time: ");

    Serial.println(epochTime);

    String urlFinal =
"https://script.google.com/macros/s/"+GOOGLE_SCRIPT_ID_POST+"/ex
ec?"+"&waktu=" + String(epochTime) + "&suhu=" + String(suhu)+
"&ph=" + String(ph);

    // Serial.print("POST data to spreadsheet:");

    // Serial.println(urlFinal);

    HTTPClient http;

    http.begin(urlFinal.c_str());

    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);

    int httpCode = http.GET();
```

```
    Serial.print("HTTP Post Sheet Status Code: ");
    Serial.println(httpCode);
    //-------------------------------------------------------------
-----------
    //getting response from google sheet
    String payload;
    if (httpCode > 0) {
        payload = http.getString();
        // Serial.println("Payload: "+payload);
    }
    else {
      Serial.println("Error on HTTP request Post To Sheet");
      return;
    }
    //-------------------------------------------------------------
-----------
    http.end();
  }
  delay(1000);
}
void ReadFromSpreadsheet()
{
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = "https://script.google.com/macros/s/" +
GOOGLE_SCRIPT_ID_READ + "/exec?read";
    Serial.println("Making a request");
```

```
    http.begin(url.c_str()); //Specify the URL and certificate
    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
    int httpCode = http.GET();
    Serial.print("HTTP Read Sheet Status Code: ");
    Serial.println(httpCode);
    String payload;
    if (httpCode > 0) {
      /* 27,7 */
      payload = http.getString();
      int koma = payload.indexOf(",");
      pred_suhu = payload.substring(0, koma).toFloat();
      pred_ph   = payload.substring(koma + 1).toFloat();
      Serial.println(httpCode);
      Serial.println(payload);
      Serial.println(String("Pred Suhu: ") + pred_suhu);
      Serial.println(String("Pred pH  : ") + pred_ph);
    }
    else {
      Serial.println("Error on HTTP request Read From Sheet");
      return;
    }
    http.end();
  }
  delay(1000);
}
```

Program tersebut juga berfungsi untuk mengirim data suhu dan pH yang terbaca ke spreadsheet Google Sheets menggunakan HTTP POST request, serta membaca data prediksi suhu dan pH dari spreadsheet menggunakan HTTP GET request.

- Library dan Konstanta

Program menggunakan beberapa library seperti `WiFi`, `ThingESP`, `OneWire`, `DallasTemperature`, `HTTPClient`, dan `time`. Konstanta seperti `ntpServer`, `gmtOffset_sec`, `daylightOffset_sec`, `potPin`, `oneWireBus`, `INTERVAL`, `GOOGLE_SCRIPT_ID_POST`, dan `GOOGLE_SCRIPT_ID_READ` juga dideklarasikan.

- Setup

Fungsi `setup()` berisi konfigurasi awal yang dilakukan saat mikrokontroler dihidupkan. Serial diinisialisasi, sensor suhu DS18B20 diinisialisasi dengan menggunakan OneWire dan DallasTemperature, dan pin sensor pH diset sebagai input. WiFi dihubungkan dengan menggunakan SSID dan password yang telah ditentukan. Kemudian, fungsi `initDevice()` dari `ThingESP32` digunakan untuk menginisialisasi perangkat dengan nama dan deskripsi yang ditentukan. Terakhir, waktu lokal diatur menggunakan `configTime()` dengan menggunakan NTP server dan offset GMT+7.

- HandleResponse()

Fungsi `HandleResponse()` adalah fungsi untuk menangani permintaan yang diterima dari server. Jika permintaan adalah "`cek tambak`", fungsi akan memanggil fungsi `bacaSensor`() untuk membaca nilai suhu dan pH dari sensor. Jika permintaan adalah "`prediksi`", fungsi akan memanggil fungsi `ReadFromSpreadsheet`() untuk membaca data prediksi suhu dan pH dari spreadsheet. Fungsi ini mengembalikan string dengan informasi suhu dan pH.

- Loop

Fungsi `loop()` adalah fungsi yang dijalankan secara berulang. Pada setiap iterasi, program akan memeriksa apakah waktu sejak iterasi sebelumnya telah melebihi `INTERVAL` (10 detik). Jika sudah, program akan memanggil fungsi `bacaSensor()` untuk membaca nilai suhu dan pH dari sensor, dan memanggil fungsi `sendToSpreadsheet()` untuk mengirim data suhu dan pH ke spreadsheet. Selanjutnya, fungsi `Handle()` dari ThingESP32 digunakan untuk menangani permintaan dari server.

- bacaSensor()

Fungsi `bacaSensor()` digunakan untuk membaca nilai suhu dan pH dari sensor. Pertama, sensor suhu DS18B20 dipanggil untuk mengambil data suhu dalam derajat Celcius. Selanjutnya, nilai analog dari pin sensor pH dibaca dan diubah menjadi tegangan. Tegangan tersebut kemudian dikonversi menjadi nilai pH menggunakan rumus yang telah ditentukan. Hasil pembacaan suhu dan pH disimpan dalam variabel global suhu dan ph.

- getTime()

Fungsi `getTime()` digunakan untuk mendapatkan waktu saat ini dalam bentuk epoch time (detik sejak 1 Januari 1970). Fungsi ini menggunakan library time dan `getLocalTime()` untuk mendapatkan waktu lokal.

- sendToSpreadsheet()

Fungsi `sendToSpreadsheet()` digunakan untuk mengirim data suhu dan pH ke spreadsheet Google Sheets menggunakan HTTP POST request. Jika koneksi WiFi terhubung, program akan mendapatkan waktu saat ini menggunakan fungsi `getTime()`. Selanjutnya, URL yang berisi data suhu, pH, dan waktu dikirim melalui HTTP GET request ke script Google Sheets yang telah disediakan. Hasil HTTP request dicetak pada Serial Monitor.

- ReadFromSpreadsheet()

Fungsi `ReadFromSpreadsheet()` digunakan untuk membaca data prediksi suhu dan pH dari spreadsheet Google Sheets menggunakan HTTP GET request. Jika koneksi WiFi terhubung, program akan mengirimkan HTTP GET request ke script Google Sheets yang telah disediakan. Hasil response berupa data prediksi suhu dan pH akan diambil dari payload dan disimpan dalam variabel `pred_suhu` dan `pred_ph`.

- Serial Monitor

Program ini menggunakan Serial Monitor untuk mencetak informasi mengenai koneksi WiFi, waktu epoch, status HTTP request, dan nilai suhu, pH, serta prediksi suhu dan pH.

- delay()

Terdapat beberapa `delay()` dalam program ini untuk memberikan waktu tunggu setelah setiap koneksi atau pemrosesan data.