Universitas Semarang
Fakultas Teknologi Informasi dan Komunikasi
Teknik Informatika

Jaringan dan Infrastruktur Sistem Cerdas

Modul Praktikum Mahasiswa

Oleh:
Alauddin Maulana Hirzan, S. Kom., M. Kom
NIDN. 0607069401

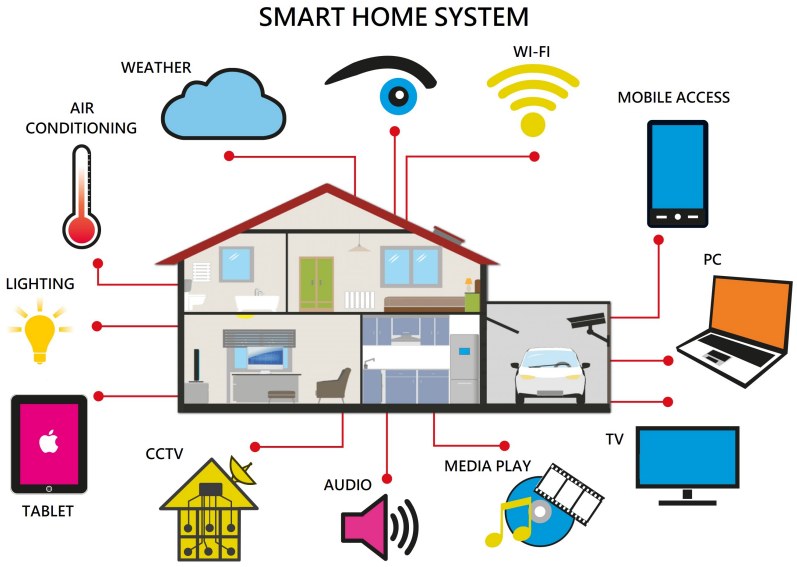
Daftar Isi

Daftar Gambar

Pendahuluan

0.1 Mengenal Jaringan dan Infrastruktur Sistem Cerdas

Jaringan dan Infrastruktur Sistem Cerdas merupakan integrasi antara teknologi jaringan komputer dengan komponen sistem cerdas seperti kecerdasan buatan, Internet of Things (IoT), dan komputasi terdistribusi—yang dirancang untuk mengumpulkan, mentransmisikan, memproses, dan menganalisis data secara otomatis guna menghasilkan keputusan adaptif dan responsif terhadap kondisi lingkungan. Infrastruktur ini mencakup perangkat fisik (sensor, aktuator, server, edge device), protokol komunikasi (misalnya MQTT, HTTP), serta platform komputasi (cloud, fog, dan edge computing) yang bekerja secara terpadu untuk mendukung layanan cerdas seperti smart city, smart agriculture, dan industrial automation, dengan karakteristik utama berupa skalabilitas, interoperabilitas, dan kemampuan pembelajaran dari data secara berkelanjutan.

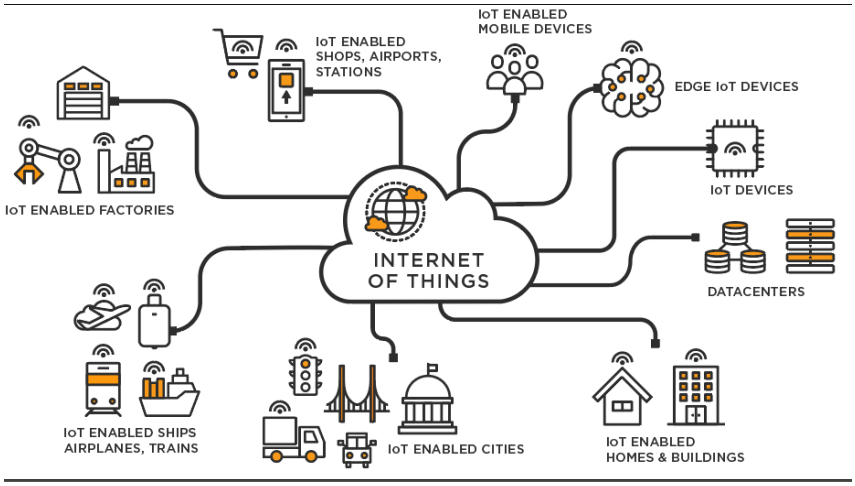


Gambar 1: Sistem Cerdas

0.2 Mengenal Internet of Things

Internet of Things (IoT) merupakan paradigma dalam jaringan komputer yang menghubungkan berbagai perangkat fisik (things)—seperti sensor, aktuator, dan perangkat embedded—ke dalam jaringan internet sehingga mampu mengumpulkan, mengirim, dan bertukar data

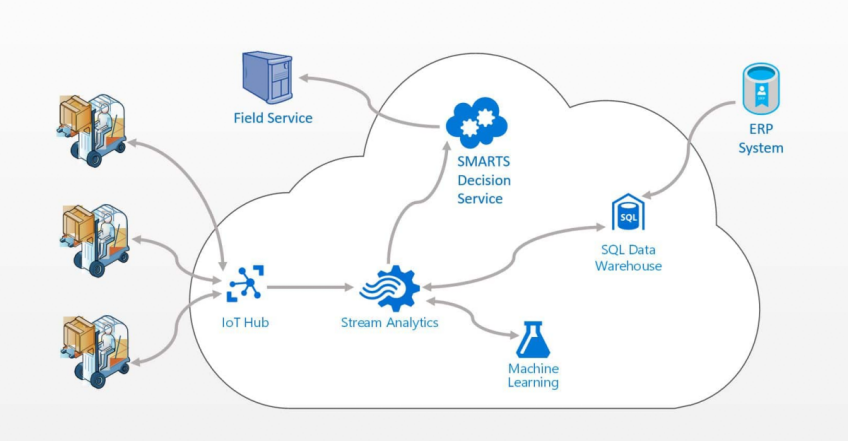
secara otomatis tanpa interaksi manusia secara langsung. Dalam sistem IoT, setiap perangkat dilengkapi dengan identitas unik serta kemampuan komunikasi melalui protokol jaringan (misalnya MQTT, CoAP, atau HTTP), sehingga memungkinkan integrasi dengan platform komputasi seperti cloud, edge, atau fog computing untuk proses analisis data dan pengambilan keputusan. IoT banyak diterapkan pada berbagai domain seperti smart home, smart city, kesehatan digital, dan industri 4.0, dengan karakteristik utama berupa konektivitas tinggi, real-time data processing, serta kemampuan monitoring dan kontrol jarak jauh.



Gambar 2: Internet of Things

0.3 Mengenal Decision System

Decision System atau sistem pendukung keputusan merupakan sistem berbasis komputer yang dirancang untuk membantu proses pengambilan keputusan dengan memanfaatkan data, model analitis, dan algoritma tertentu guna menghasilkan rekomendasi atau alternatif solusi yang optimal. Sistem ini biasanya mengintegrasikan komponen seperti basis data, model perhitungan (misalnya statistik, optimasi, atau machine learning), serta antarmuka pengguna untuk memfasilitasi analisis terhadap berbagai skenario. Dalam konteks sistem cerdas, decision system sering dikombinasikan dengan teknik kecerdasan buatan seperti supervised learning, fuzzy logic, atau reinforcement learning agar mampu menangani ketidakpastian dan meningkatkan akurasi keputusan, serta banyak digunakan dalam bidang bisnis, kesehatan, jaringan komputer, dan manajemen sumber daya.



Gambar 3: Decision System

Persiapan Praktikum

Agar praktikum dapat berjalan dengan lancar, mahasiswa diwajibkan memenuhi persyaratan berikut baik dalam bentuk perangkat keras maupun lunak:

0.4 Perangkat Keras

- Prosesor dengan 4 inti
- RAM minimal 8GB
- HDD 10GB

0.5 Perangkat Lunak

Perangkat lunak berikut ini wajib diinstall oleh mahasiswa demi lancarnya praktikum:

- Browser

Bab 1

Infrastruktur Sistem Cerdas Berbasis Jaringan

1.1 Pendahuluan

Praktikum Infrastruktur Sistem Cerdas Berbasis Jaringan dirancang untuk memberikan pemahaman konseptual dan keterampilan teknis dalam membangun sistem terintegrasi yang menggabungkan jaringan komputer dengan teknologi sistem cerdas, seperti Internet of Things (IoT), komputasi edge, dan kecerdasan buatan. Melalui kegiatan praktikum ini, dilakukan eksplorasi terhadap arsitektur sistem yang melibatkan sensor, aktuator, protokol komunikasi (misalnya MQTT dan HTTP), serta platform pemrosesan data untuk mendukung pengambilan keputusan secara otomatis dan real-time. Pendekatan yang digunakan menekankan pada implementasi langsung, mulai dari akuisisi data, transmisi melalui jaringan, hingga proses inferensi dan respons sistem, sehingga diharapkan terbentuk pemahaman menyeluruh mengenai bagaimana infrastruktur jaringan mendukung operasi sistem cerdas yang adaptif, skalabel, dan efisien dalam berbagai konteks aplikasi.

1.2 Tutorial

1. Untuk memulai praktikum, mahasiswa wajib mengakses website **??**Wokwi terlebih dahulu.

Gambar 1.1: Mengakses Wokwi

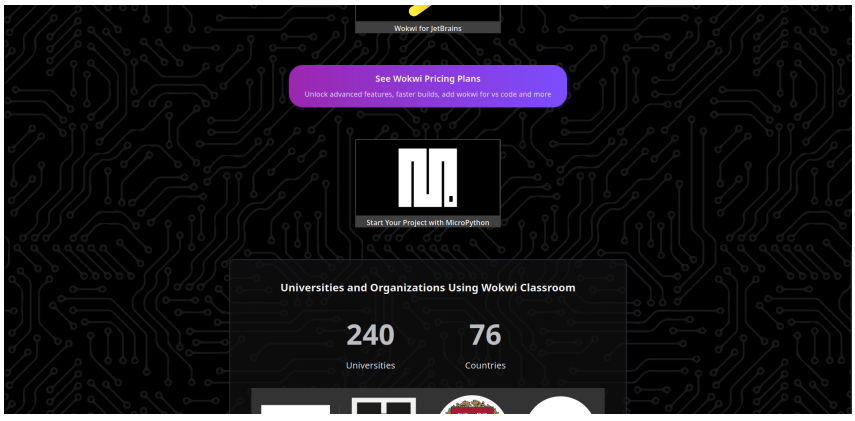
2. Lakukan registrasi akun untuk menyimpan *progress* dengan melakukan klik *Sign Up*. Gunakan *Google* untuk mempermudah dan mempercepat registrasi

Gambar 1.2: Registrasi Akun

3. Jika registrasi berhasil, maka foto profil akun *Google* akan muncul menggantikan tombol *Sign Up*

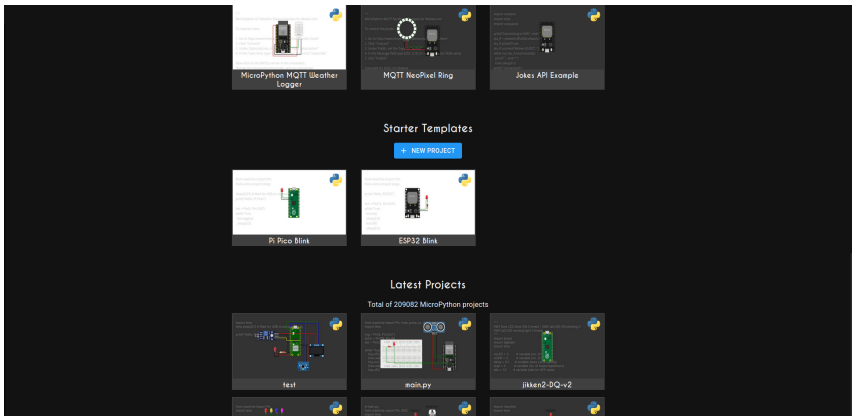
Gambar 1.3: Berhasil Login

4. Untuk memulai proyek, pilih *Start Your Project with MicroPython*



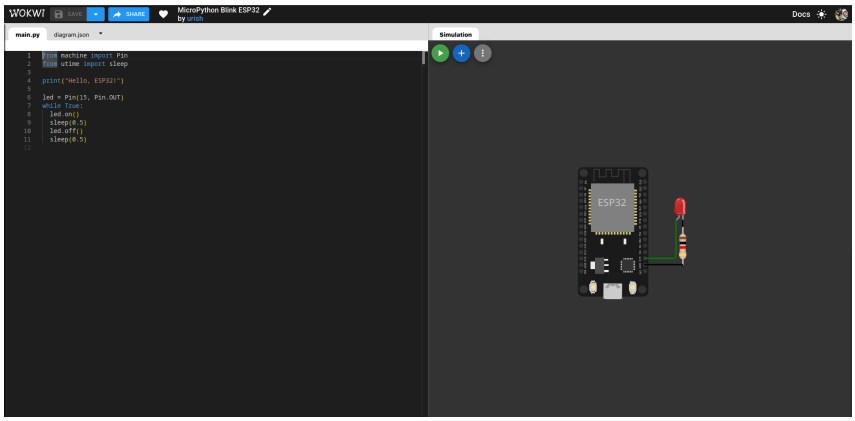
Gambar 1.4: Template MicroPython

5. Pada bagian *Starter Template*, pilih *ESP32 Blink*



Gambar 1.5: Template ESP32 Blink

6. Wokwi akan menampilkan simulasi beserta kode nya



Gambar 1.6: Tampilan ESP32 Blink dan MicroPython

7. Mahasiswa wajib memahami bagaimana kode berjalan. Impor *library* digunakan untuk mengambil pustaka untuk membantu kode, dan kode funsional itu sendiri

   - Bagian import *library*

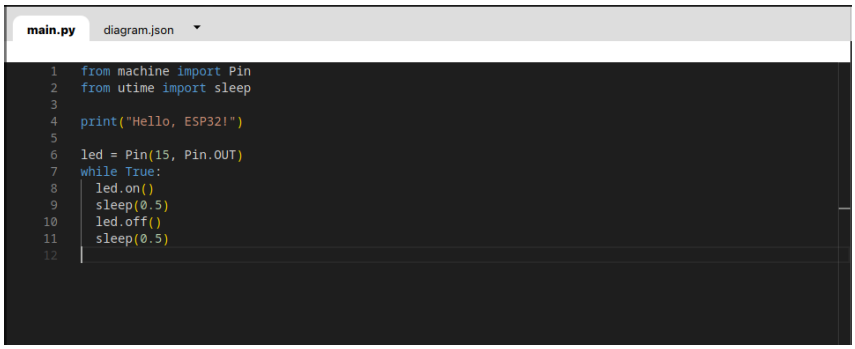
     Potongan Kode

     ```
     from machine import Pin
     from utime import sleep
     ```

   - Bagian fungsional kode

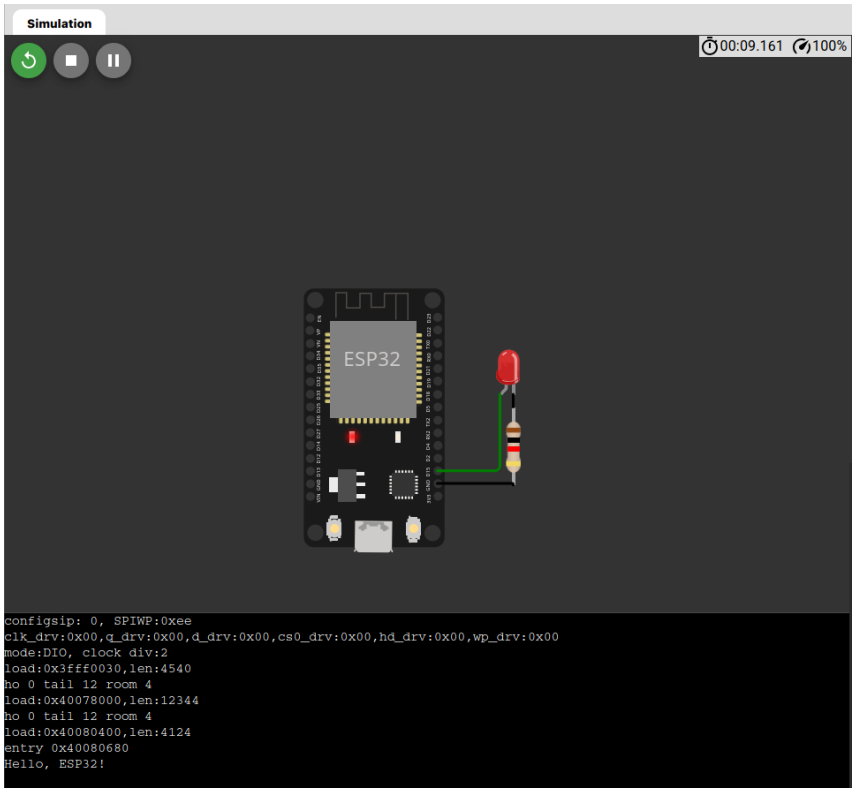
     Potongan Kode

     ```
     print("Hello, ESP32!")

     led = Pin(15, Pin.OUT)

     while True:
         led.on()
         sleep(0.5)
         led.off()
         sleep(0.5)
     ```



Gambar 1.7: Tampilan Kode MicroPython

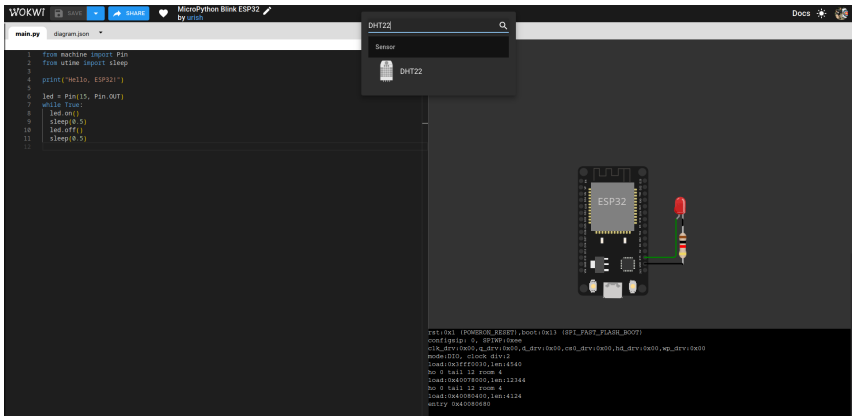
8. Untuk menguji apakah kode berjalan dengan baik atau tidak, klik *Run* di panel kanan, atas perangkat



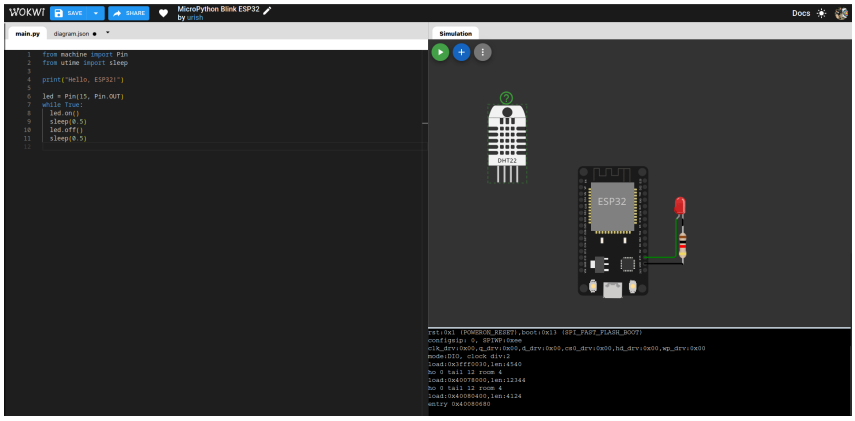
Gambar 1.8: Menjalankan Kode MicroPython

9. Pencet *Stop* untuk menghentikan alat

10. Berikutnya adalah menambahkan sensor DHT22 dengan alat. Klik tombol *Plus*, ketik *DHT22* dan klik sensor tersebut.

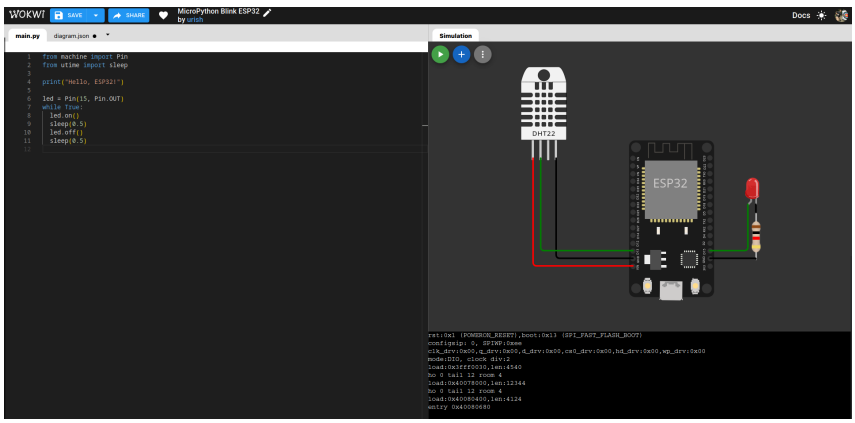


Gambar 1.9: Menambahkan Sensor DHT22



Gambar 1.10: Menambahkan Sensor DHT22

11. Hubungkan kaki-kaki DHT22 ke ESP32 dengan ketentuan berikut:
    - dht1:VCC → esp:VIN
    - dht1:GND → esp:GND.2
    - dht1:SDA → esp:D13



Gambar 1.11: Menghubungkan Kaki DHT22 ke ESP32

12. Agar sensor dapat bekerja dengan baik, di bagian kode perlu ditambahkan potongan berikut:

- Tambahkan impor *library* setelah `from utime import sleep`

Potongan Kode

```
from dht import dht
```

```
from machine import Pin
from utime import sleep
from dht import dht

print("Hello, ESP32!")

led = Pin(15, Pin.OUT)
while True:
  led.on()
  sleep(0.5)
  led.off()
  sleep(0.5)
```

Gambar 1.12: Impor library dht

- Hapus kode `print("Hello, ESP32!")`

```
from machine import Pin
from utime import sleep
from dht import dht

print("Hello, ESP32!")

led = Pin(15, Pin.OUT)
while True:
  led.on()
  sleep(0.5)
  led.off()
  sleep(0.5)
```

Gambar 1.13: Hapus Potongan Kode

- Tambahkan kode berikut sebelum baris `led = Pin()`

Potongan Kode

```
sensor = DHT22(Pin(13, Pin.IN))
```

main.py ● diagram.json ●

```
from machine import Pin
from utime import sleep
from dht import DHT22

sensor = DHT22(Pin(13, Pin.IN))
led = Pin(15, Pin.OUT)

while True:
  led.on()
  sleep(0.5)
  led.off()
  sleep(0.5)
```

Gambar 1.14: Inisialisasi Kode Sensor

- Tambahkan kode berikut sebelum `while True:`. Perhatikan 4 spasi

Potongan Kode

```
def ukur_suhu_lembab():
    data = sensor.measure()
    temp = sensor.temperature()
    humi = sensor.humidity()
    return (temp, humi)
```

```
from machine import Pin
from utime import sleep
from dht import DHT22

sensor = DHT22(Pin(13, Pin.IN))
led = Pin(15, Pin.OUT)

def ukur_suhu_lembab():
    data = sensor.measure()
    temp = sensor.temperature()
    humi = sensor.humidity()
    return (temp, humi)

while True:
  led.on()
  sleep(0.5)
  led.off()
  sleep(0.5)
```

Gambar 1.15: Menambahkan Fungsi DHT22

- Berikutnya adalah membuat fungsi blink. Hapus kode `while True:` seluruhnya

```
from machine import Pin
from utime import sleep
from dht import DHT22

sensor = DHT22(Pin(13, Pin.IN))
led = Pin(15, Pin.OUT)

def ukur_suhu_lembab():
    data = sensor.measure()
    temp = sensor.temperature()
    humi = sensor.humidity()
    return (temp, humi)

while True:
    led.on()
    sleep(0.5)
    led.off()
    sleep(0.5)
```

Gambar 1.16: Hapus kode While

- Lalu tambahkan fungsi baru untuk menyalakan dan mematikan lampu secara dinamis

Potongan Kode

```
def blink(repetisi: int = 1):
    for i in range(repetisi+1):
        led.off()
        sleep(0.5)
        led.on()
        sleep(0.5)
        led.off()
```

```
from machine import Pin
from utime import sleep
from dht import DHT22

sensor = DHT22(Pin(13, Pin.IN))
led = Pin(15, Pin.OUT)

def ukur_suhu_lembab():
    data = sensor.measure()
    temp = sensor.temperature()
    humi = sensor.humidity()
    return (temp, humi)

def blink(repetisi: int = 1):
    for i in range(repetisi):
        led.off()
        sleep(0.5)
        led.on()
        sleep(0.5)
        led.off()
```

Gambar 1.17: Tambahkan Fungsi Blink Dinamis

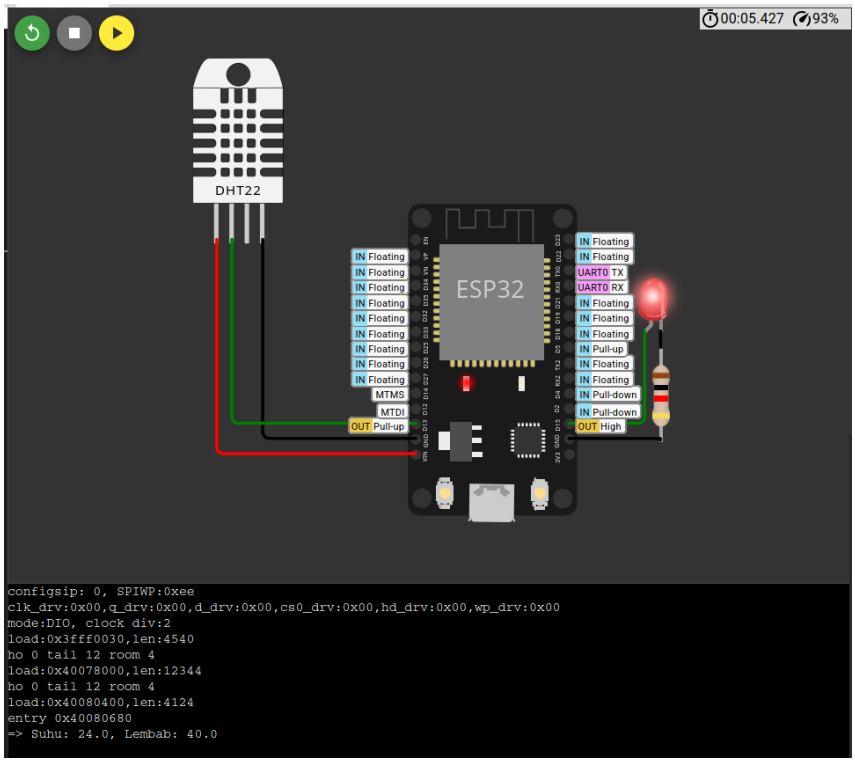
- Di baris paling bawah, tambahkan kode berikut:

Potongan Kode

```
while True:
    temp, humi = ukur_suhu_lembab()
    print(f"=> Suhu: temp, Lembab: humi")
    blink(3)
    sleep(5)
```

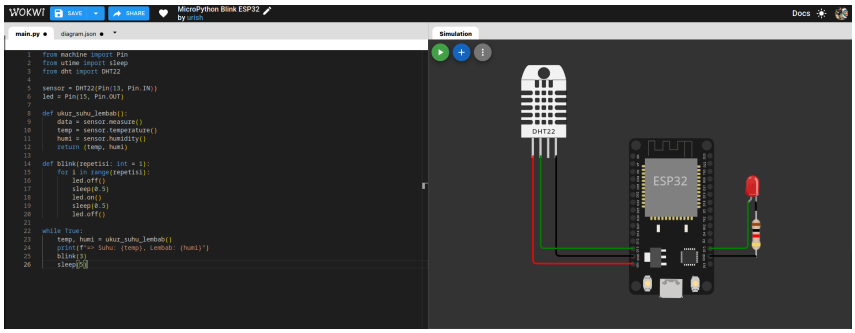
```
from machine import Pin
from utime import sleep
from dht import DHT22

sensor = DHT22(Pin(13, Pin.IN))
led = Pin(15, Pin.OUT)

def ukur_suhu_lembab():
    data = sensor.measure()
    temp = sensor.temperature()
    humi = sensor.humidity()
    return (temp, humi)

def blink(repetisi: int = 1):
    for i in range(repetisi):
        led.off()
        sleep(0.5)
        led.on()
        sleep(0.5)
        led.off()

while True:
    temp, humi = ukur_suhu_lembab()
    print(f"=> Suhu: {temp}, Lembab: {humi}")
    blink(3)
    sleep(5)
```

Gambar 1.18: Tambahkan Kode Runner



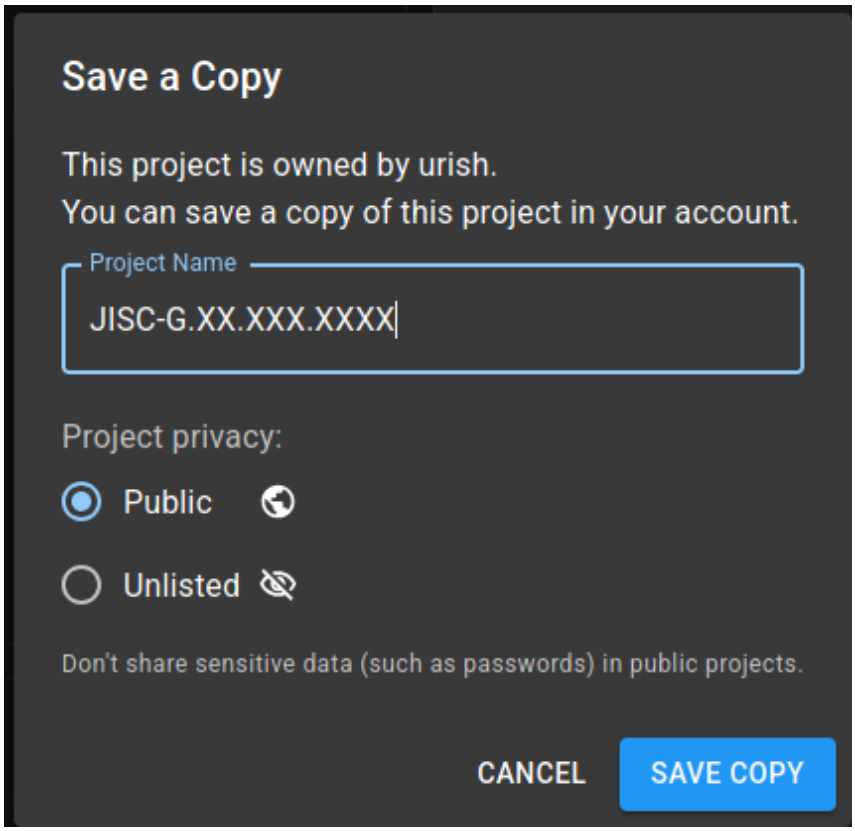
Gambar 1.19: Hasil Akhir Kode

13. Untuk menyimpan permanen, klik *Save* di bagian atas



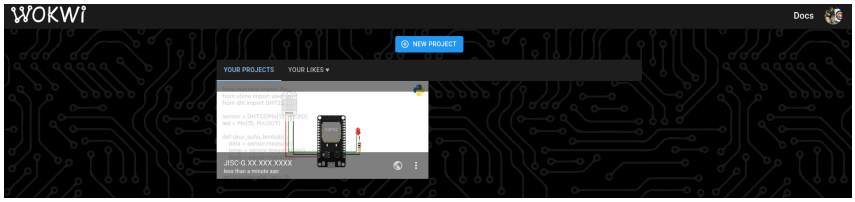
Gambar 1.20: Menyimpan Proyek

14. Masukkan nama proyek dengan nama yang mudah. Contoh: *JISC - NIM*. Lalu klik *Save a Copy*



Gambar 1.21: Ganti Nama Proyek

15. Untuk mengakses kembali proyek, klik *Foto Profil* → *My Projects*



Gambar 1.22: My Projects

Bab 2

Komunikasi Jaringan dari Wokwi ke Internet (HTTPS)

2.1 Pendahuluan

Praktikum ini bertujuan untuk memperkenalkan konsep komunikasi jaringan antara perangkat Internet of Things (IoT) yang disimulasikan menggunakan platform Wokwi dengan layanan di internet melalui protokol HTTPS. Dalam kegiatan ini, mahasiswa akan mempelajari bagaimana sebuah perangkat virtual, seperti mikrokontroler ESP32, dapat mengirimkan data sensor ke server berbasis web menggunakan metode request HTTPS (GET/-POST), serta memahami alur komunikasi client-server secara nyata. Praktikum ini juga menekankan pada implementasi sederhana namun efektif untuk menghubungkan sistem embedded dengan layanan cloud atau API eksternal, sehingga memberikan gambaran dasar mengenai integrasi IoT dengan infrastruktur jaringan global.

2.2 Tutorial

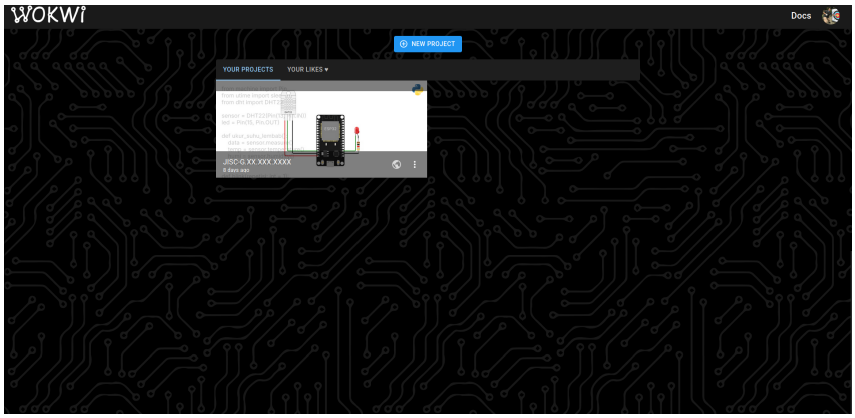
1. Buka website www.wokwi.com, lalu masuk ke akun yang sudah dibaut sebelumnya.

Gambar 2.1: Akses Wokwi

2. Klik Foto Profil, lalu Klik **My Projects**

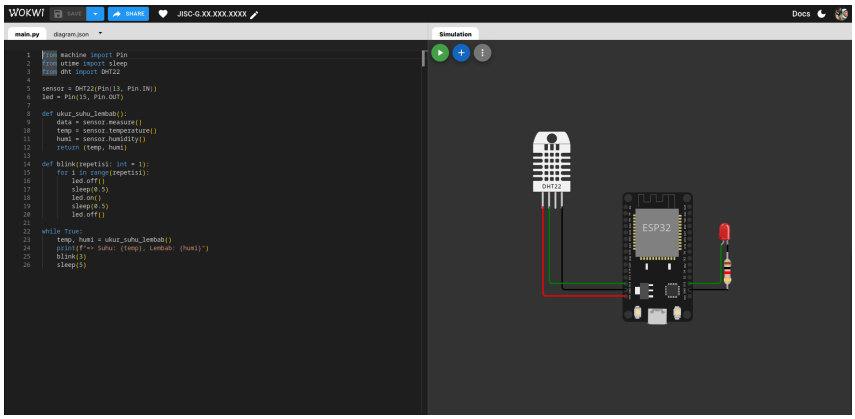
Gambar 2.2: Akses Projek Lama

3. Setelah projek lama ditampilkan, klik projek dengan nama **JISC-G.XX.XXX.XXXX**. Jika tidak ada, ulangi Praktikum 1



Gambar 2.3: Buka Projek JISC

4. Tunggu beberapa saat hingga simulasi perangkat dan kode nya ditampilkan

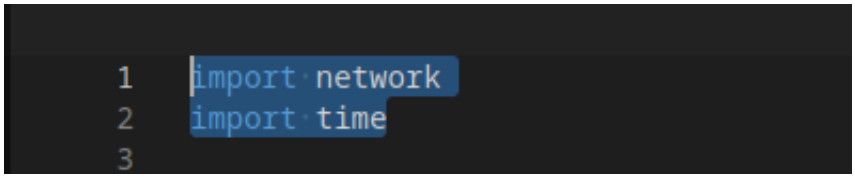


Gambar 2.4: Tampilan Editor Wokwi

5. Hal pertama yang perlu ditambahkana dalah akses Internet melalui jaringan Wokwi. Tambahkan `import network` di baris paling pertama

Potongan Kode

```
import network
import time
```



Gambar 2.5: Menambahkan Import Baru

6. Setelah memasukkan kode untuk jaringan, berikutnya adalah memasukkan variabel global untuk masuk ke jaringan setelah kode import. Untuk jaringan Wokwi menggunakan konfigurasi.
   - SSID : "Wokwi-GUEST"
   - Password : ""

Potongan Kode

```
SSID = "Wokwi-GUEST"
PASSWORD = ""
```

Gambar 2.6: Konfigurasi WiFi SSID dan Password

7. Setelah mengatur variabel SSID dan Password, berikutnya membuat fungsi untuk koneksi WiFi. Letakkan kode ini sebelum kode `while True:`

Potongan Kode

```
def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(SSID, PASSWORD)

    print("Menghubungkan ke WiFi...", end="")
    while not wlan.isconnected():
        print(".", end="")
        time.sleep(1)

    print("\nTerhubung!")
    print("IP Address:", wlan.ifconfig()[0])
```

```
27
28  def connect_wifi():
29      wlan = network.WLAN(network.STA_IF)
30      wlan.active(True)
31      wlan.connect(SSID, PASSWORD)
32
33      print("Menghubungkan ke WiFi...", end="")
34      while not wlan.isconnected():
35          print(".", end="")
36          time.sleep(1)
37
38      print("\nTerhubung!")
39      print("IP Address:", wlan.ifconfig()[0])
40
```

Gambar 2.7: Fungsi untuk menghubungkan ke Internet

8. Untuk menguji apakah bisa terhubung atau tidak, panggil fungsi tersebut dengan memasukkan kode berikut setelah `while True:`

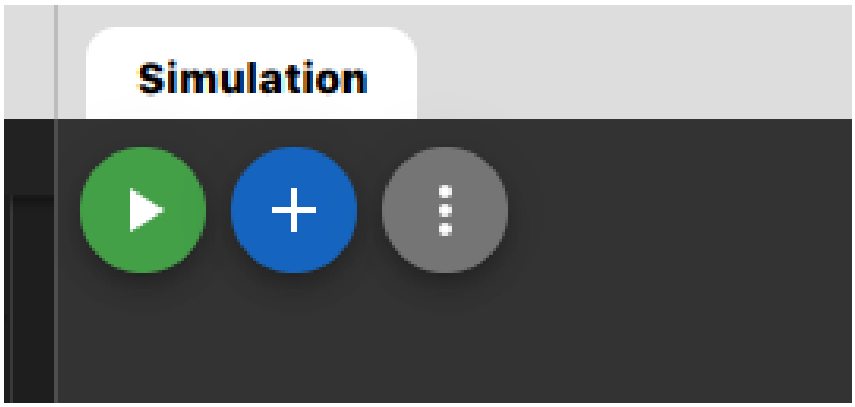
Potongan Kode

```
connect_wifi()
```

```
40
41  while True:
42      connect_wifi()
43      temp, humi = ukur_suhu_lembab()
44      print(f"=> Suhu: {temp}, Lembab: {humi}")
45      blink(3)
46      sleep(5)
```

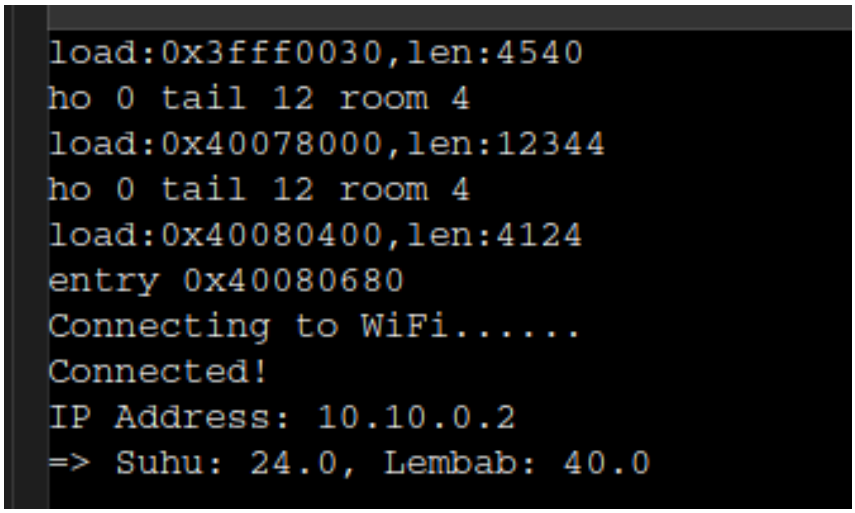
Gambar 2.8: Panggil fungsi connect_wifi

9. Klik **Play** / **Run**



Gambar 2.9: Koneksi WiFi

10. Tunggu hingga muncul tulisan berikut di sebelah kanan bawah

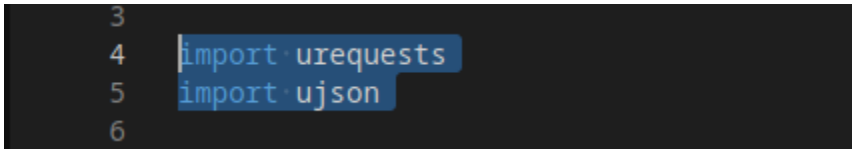


Gambar 2.10: Koneksi Berhasil

11. Klik **Stop** untuk menghentikan simulasi
12. Untuk mempraktekkan pengiriman data melalui ReST API, pertama masukkan import library baru setelah baris kode `import time`

Potongan Kode

```
import urequests
import ujson
```

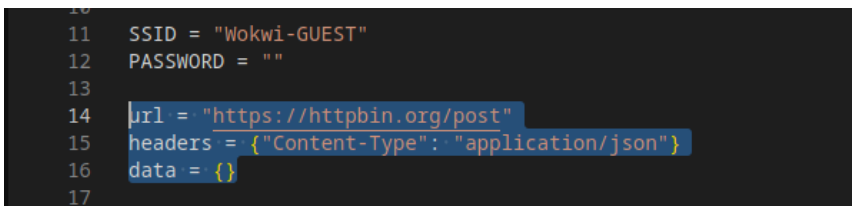


Gambar 2.11: Menambahkan Library Baru

13. Berikutnya adalah menambahkan variabel global untuk url, header dan data. Masukkan kode berikut setelah SSID dan Password

Potongan Kode

```
url = "https://httpbin.org/post"
headers = {"Content-Type": "application/json"}
data = {}
```



Gambar 2.12: Menambahkan Library Baru

14. Langkah berikutnya adalah menambahkan kode untuk membungkus data sensor ke variabel `data` dan diformat ke dalam bentuk json. Tambahkan kode berikut sebelum `blink(3)`

Potongan Kode

```
data = {"temp": temp, "humi": humi}
json_data = ujson.dumps(data)
```

```
47
48  while True:
49      connect_wifi()
50      temp, humi = ukur_suhu_lembab()
51      print(f"=> Suhu: {temp}, Lembab: {humi}")
52
53      data = {"temp": temp, "humi": humi}
54      json_data = ujson.dumps(data)
55
56      blink(3)
57      sleep(5)
```

Gambar 2.13: Bungkus Data Sensor ke data

15. Setelah membungkus data dalam format json, berikutnya adalah mencoba mengirimkan data ke url. Lanjutkan kode sebelumnya dengan kode berikut ini

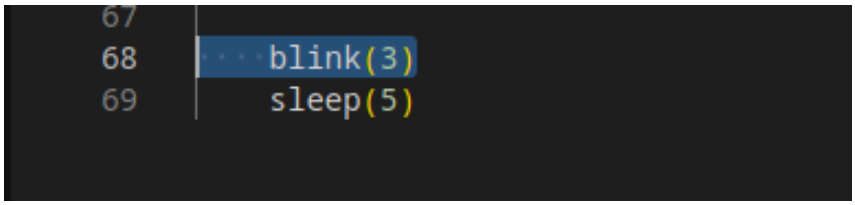
Potongan Kode

```
try:
    print("=> Mengirimkan Data")
    resp = urequests.post(url, data=json_data, headers=headers)
    status = resp.status_code
    resp.close()

    if status == 200:
        print("==> Sukses")
        blink(3)
    else:
        print("=> Gagal")
        blink(1)
except Exception as e:
    print(f"Error : e")
```

```
55
56      try:
57          print("=> Mengirimkan Data")
58          resp = urequests.post(url, data=json_data, headers=headers)
59          status = resp.status_code
60          resp.close()
61
62          if status == 200:
63              print("==> Sukses")
64              blink(3)
65          else:
66              print("=> Gagal")
67              blink(1)
68      except Exception as e:
69          print(f"Error : {e}")
70
71      blink(3)
72      sleep(10)
```

Gambar 2.14: Mengirimkan sampel data via ReST

16. Terakhir adalah hapus baris kode `blink(3)` yang ada di atas `sleep(5)`

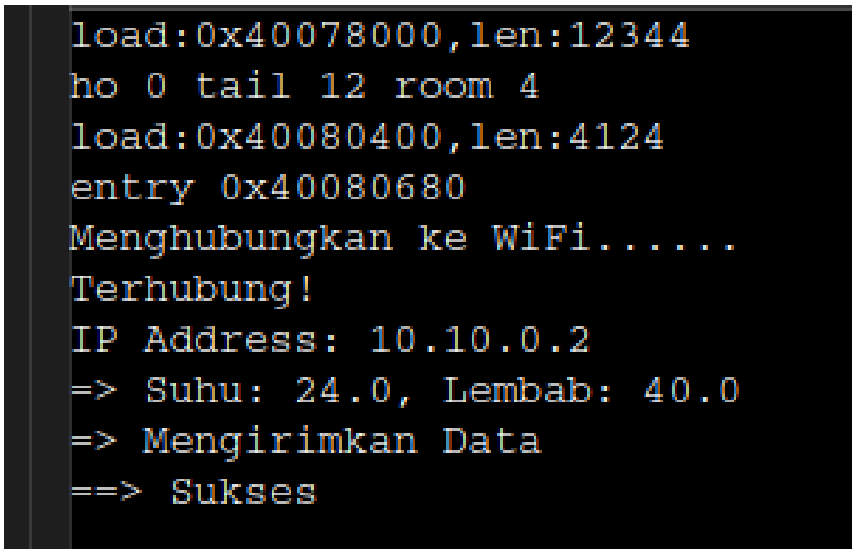


Gambar 2.15: Menghapus kode blink

17. Kemudian naikkan angka sleep dari 5 second ke 10 second

Gambar 2.16: Naikkan Sleep

18. Jalankan simulasi dan tunggu hingga muncul **Sukses**/**Gagal**



Gambar 2.17: Pengiriman Sukses

19. Jika muncul **Gagal** perbaiki kode.
20. Kirimkan hasil ini ke e-Learning, dan simpan projek dengan **CTRL+S**

Bab 3

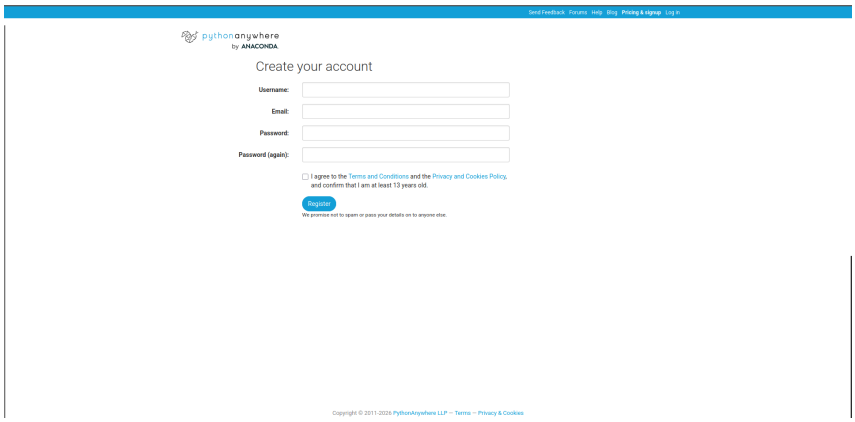
Deployment API di PythonAnywhere

3.1 Pendahuluan

Praktikum ini berfokus pada implementasi dan deployment infrastruktur backend sederhana untuk sistem cerdas melalui pembuatan REST API menggunakan Flask dan penyebarannya ke platform cloud PythonAnywhere. Dalam konteks sistem cerdas modern, backend berperan sebagai komponen utama yang menangani proses inferensi, pengolahan data, serta komunikasi antar perangkat seperti IoT dan aplikasi klien. Melalui kegiatan ini, mahasiswa akan mempelajari bagaimana membangun endpoint API POST /predict yang mampu menerima data input, melakukan proses prediksi sederhana, serta mengembalikan hasilnya secara real-time. Selain itu, proses deployment ke lingkungan cloud free-tier memberikan pemahaman praktis mengenai keterbatasan sumber daya dan strategi optimasi layanan ringan, sehingga dapat digunakan sebagai fondasi dalam pengembangan sistem cerdas berbasis layanan terdistribusi.

3.2 Tutorial

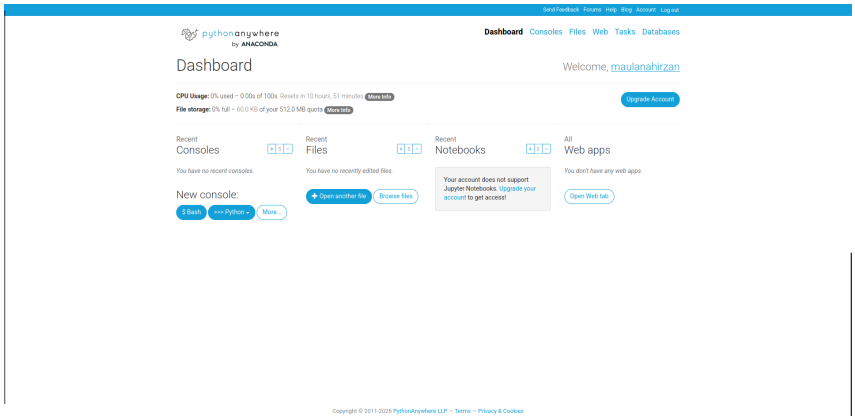
1. Buka website **pythonanywhere.com** terlebih dahulu dan daftarkan akun melalui link https://www.pythonanywhere.com/registration/register/beginner/



Gambar 3.1: Form Registrasi Akun

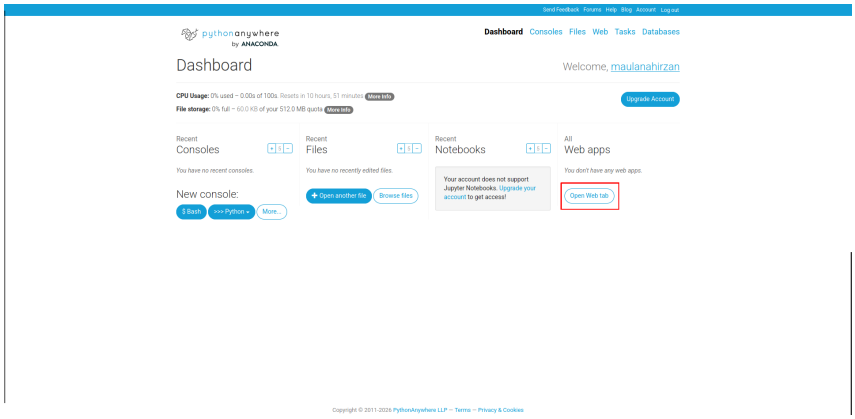
2. Jika akun sudah selesai didaftarkan dan email sudah dikonfirmasi (Cek Email masing-

masing), Log In akun hingga muncul dasbor seperti berikut:



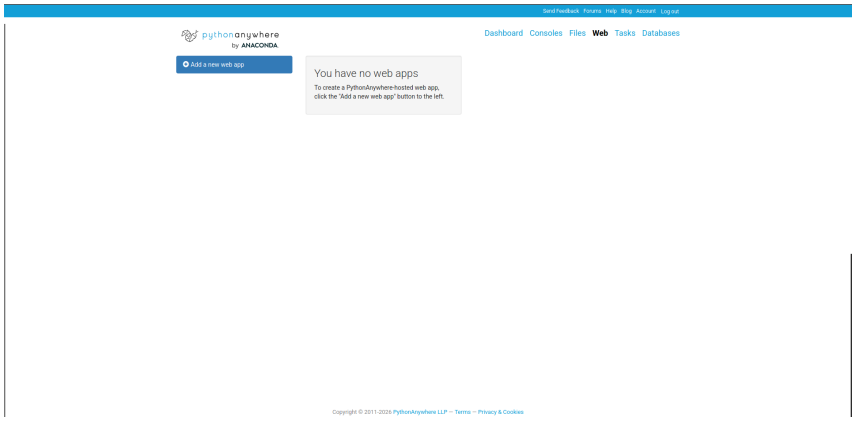
Gambar 3.2: Tampilan Dasbor PythonAnwyhere

3. Hal pertama yang perlu dibuat adalah **Web App** yang memiliki End Point ReST. Dalam kasus ini menggunakan **Python Flask**. Di bagian kanan, klik **Open Web App**



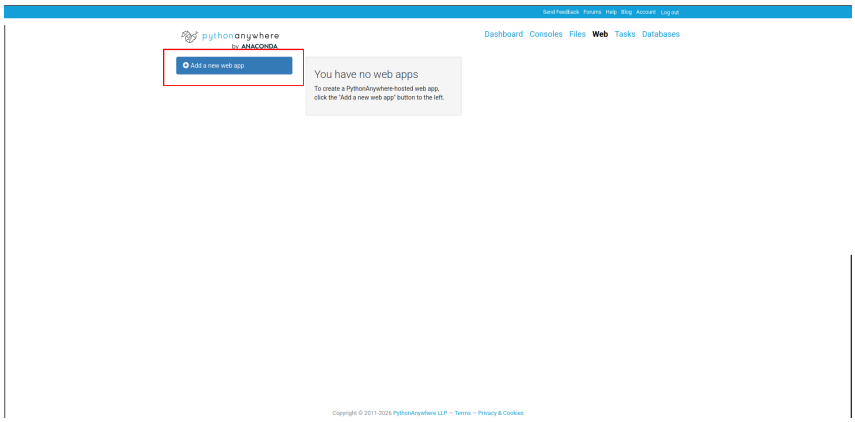
Gambar 3.3: Klik Open Web App

4. Di tampilan berikutnya akan menampilkan daftar Web App. Namun karena masih kosong, perlu membuat app baru



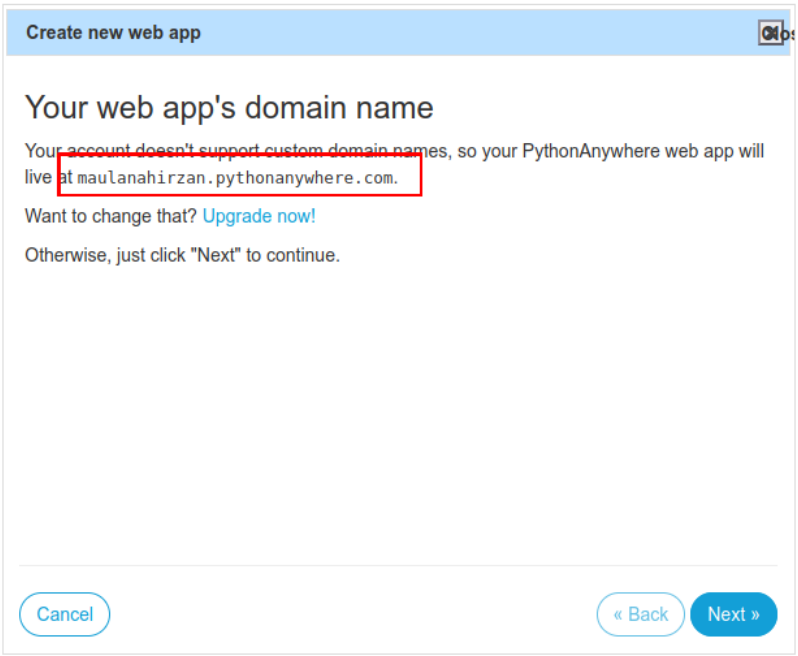
Gambar 3.4: Daftar Web App

5. Klik **Add a new web app** untuk membuat Web App baru



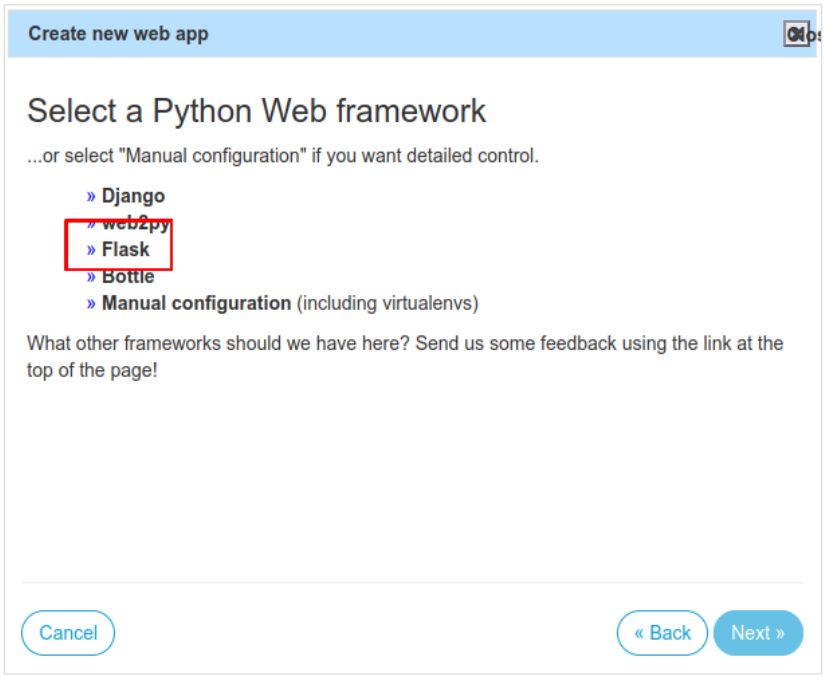
Gambar 3.5: Membuat Web App Baru

6. Dalam membuat Web App, akan ada beberapa opsi yang bisa dipilih. Halaman pertama menampilkan opsi nama domain. Secara default, mahasiswa akan mendapatkan <**username**>**.pythonanywhere.com**. Klik **Next**



Gambar 3.6: Nama Domain Web App

7. Berikutnya adalah memilih **Framework Web** yang akan digunakan. Klik **Flask**

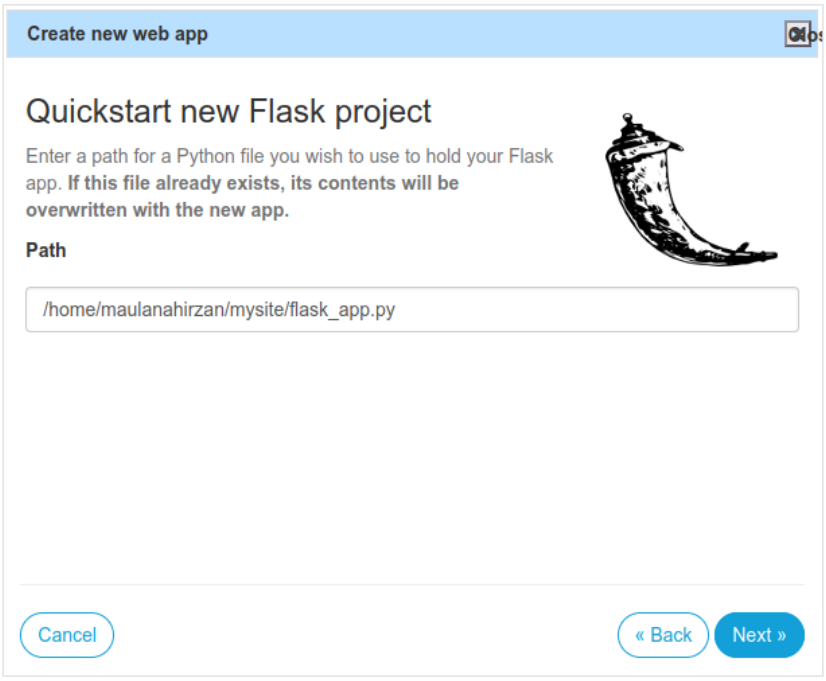


Gambar 3.7: Memilih Framework

8. Berikutnya adalah memilih versi. Pilih versi **Flask Terbaru** dan **Python Terbaru**

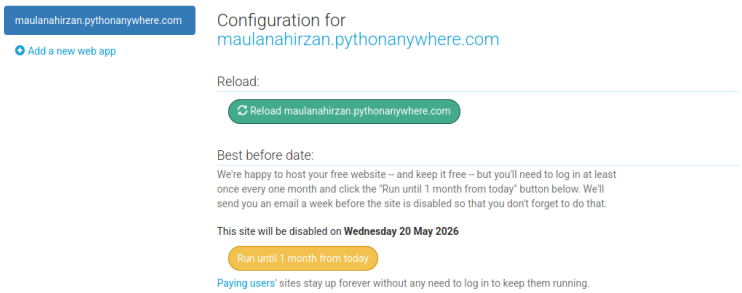
Gambar 3.8: Memilih Versi

9. Setelah itu adalah memilih lokasi projek utama. Cukup klik **Next** untuk lokasi **Default**

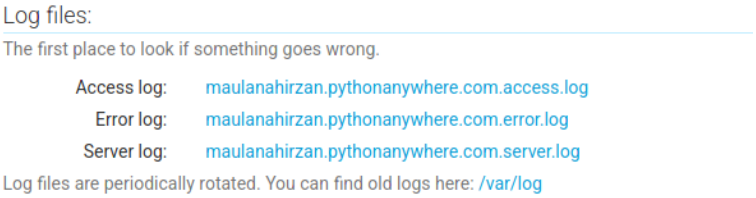


Gambar 3.9: Lokasi Web App

10. Projek **Web App** yang berhasil dibuat akan menampilkan halaman khusus untuk:
    (a) Reload App
    (b) Refresh Activate Time
    (c) Log Debugging

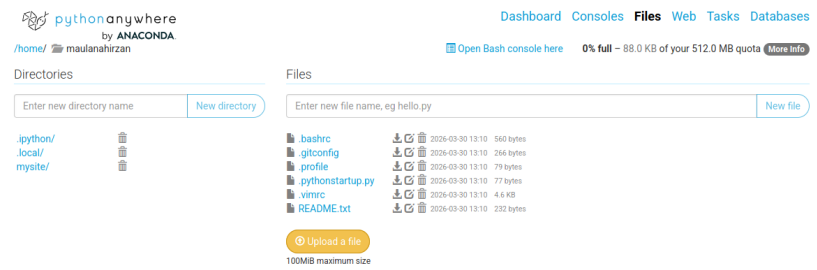


Gambar 3.10: Tombol Reload dan Refresh



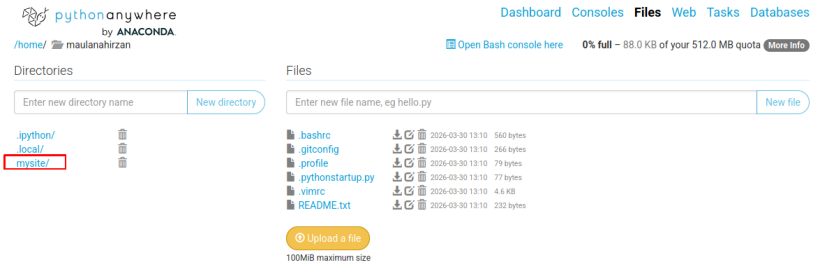
Gambar 3.11: Log Debugging

11. Untuk membuka file kode, **Klik Kanan Files** dan buka new tab (Jangan Tutup Halaman Konfigurasi Web App). Tampilan file seperti berikut



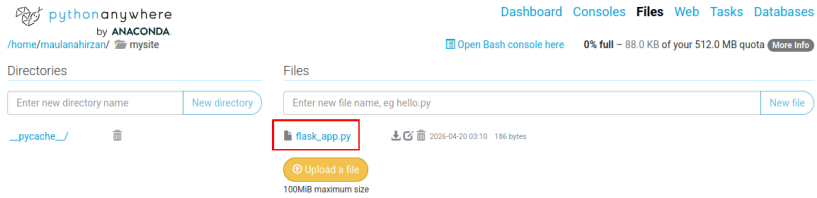
Gambar 3.12: Tampilan Files

12. Lokasi projek ada di **Folder mysite**. Klik Folder tersebut



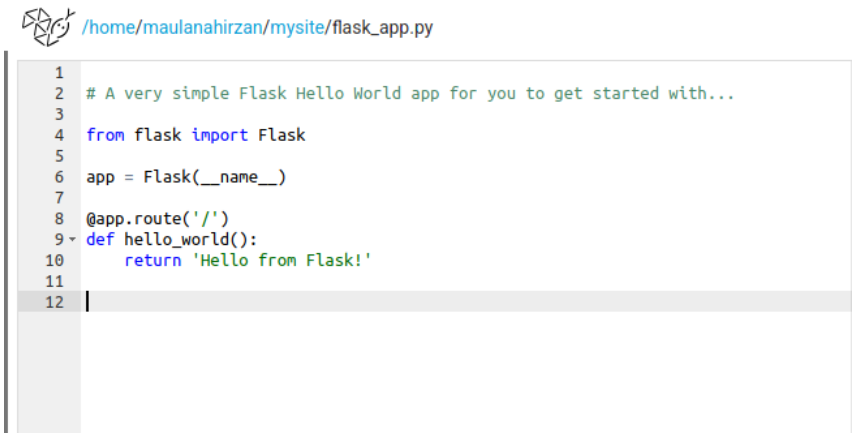
Gambar 3.13: Akses Folder mysite

13. File utama **Web App** ada di file **flask_app.py**. Klik file tersebut



Gambar 3.14: Edit file flask_app.py

14. Tampilan edit file flask_app.py



Gambar 3.15: Tampilan file flask_app.py

15. Hapus bagian kode berikut:

/home/maulanahirzan/mysite/flask_app.py

```
1
2  # A very simple Flask Hello World app for you to get started with...
3
4  from flask import Flask
5
6  app = Flask(__name__)
7
8  @app.route('/')
9  def hello_world():
10     return 'Hello from Flask!'
11
12
```

Gambar 3.16: Hapus Bagian Kode

16. Sebagai gantinya masukkan kode berikut untuk menerima data sensor dari Wokwi dan dikembalikan ke perangkat

Potongan Kode

```
@app.route('/kirim', methods=['GET'])
def kirim_data():
    temp = int(request.args['temp'])
    humi = int(request.args['humi'])

    if temp < 50 or humi < 100:
        return {"status": 1}
    else:
        return {`"status": 0}
```

/home/maulanahirzan/mysite/flask_app.py

```
1
2  # A very simple Flask Hello World app for you to get started with...
3
4  from flask import Flask
5
6  app = Flask(__name__)
7
8  @app.route('/kirim', methods=['GET'])
9  def kirim_data():
10     temp = request.args['temp']
11     humi = request.args['humi']
12
13     if temp < 50 or humi < 100:
14         return {"status": 1}
15     else:
16         return {"status": 0}
```

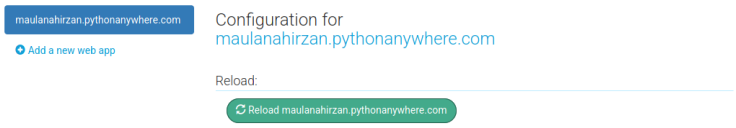
Gambar 3.17: Potongan Kode Baru

17. Di bagian `from flask import Flask`, tambahkan `request` dipisahkan dengan koma

```
1
2  # A very simple Flask Hello World app for you to get started with...
3
4  from flask import Flask, request
5
6  app = Flask(__name__)
7
8  @app.route('/kirim', methods=['GET'])
9  def kirim_data():
10     temp = request.args['temp']
11     humi = request.args['humi']
12
13     if temp < 50 or humi < 50:
14         return {"status": 1}
15     else:
16         return {"status": 0}
```

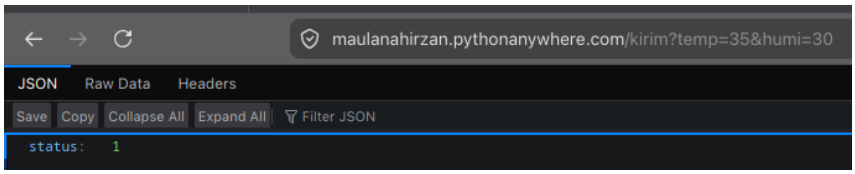
Gambar 3.18: Tambah Kode Import

18. Pastikan kode sudah tersimpan. Kembali ke Tab konfigurasi Web App, dan klik Tombol **Reload**



Gambar 3.19: Klik Reload

19. Cek apakah web berjalan dengan baik dengan cara kopikan link berikut ke browser, dan ubah bagian <username> dengan domain mahasiswa masing-masing
    - `https://<username>.pythonanywhere.com/kirim?temp=35&humi=30`
    - Ubah nilai **temp** dan **humi** untuk melihat hasil yang berbeda



Gambar 3.20: Tes API

Bab 4

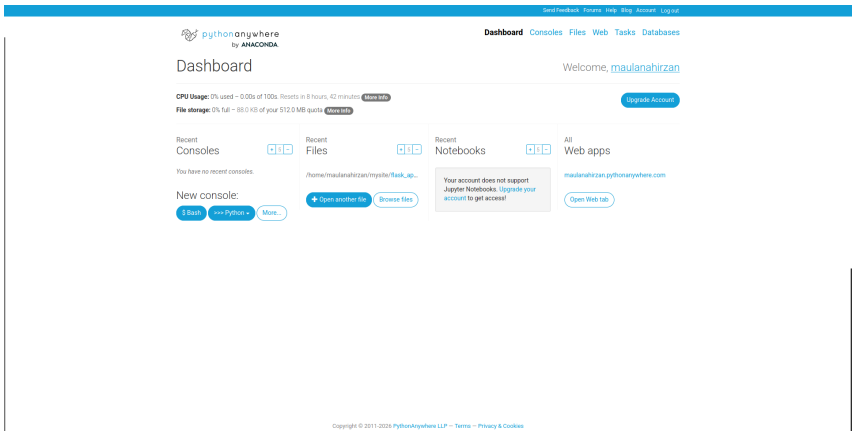
Implementasi Algoritma AI Lightweight

4.1 Pendahuluan

Pada praktikum ini, Decision Support System (DSS) dikembangkan menggunakan pendekatan Logistic Linear Model atau lebih dikenal sebagai Logistic Regression, yang merupakan algoritma klasifikasi lightweight dan sangat sesuai untuk implementasi pada lingkungan terbatas seperti PythonAnywhere free tier. Model ini bekerja dengan menghitung kombinasi linear dari fitur input yang kemudian ditransformasikan menjadi nilai probabilitas, sehingga memungkinkan sistem memberikan keputusan berbasis nilai ambang (threshold). Keunggulan metode ini terletak pada kompleksitas komputasi yang rendah, kebutuhan memori yang minimal, serta kemudahan implementasi tanpa ketergantungan pada library berat, sehingga sangat cocok untuk pembelajaran konsep dasar AI sekaligus penerapan praktis pada sistem berbasis web sederhana atau integrasi IoT.

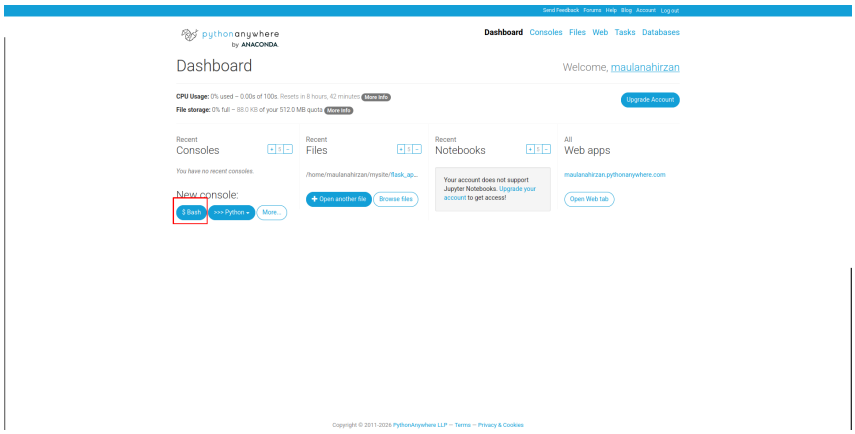
4.2 Tutorial

1. Buka kembali **PythonAnywhere** dan pastikan dashbor ditampilkan



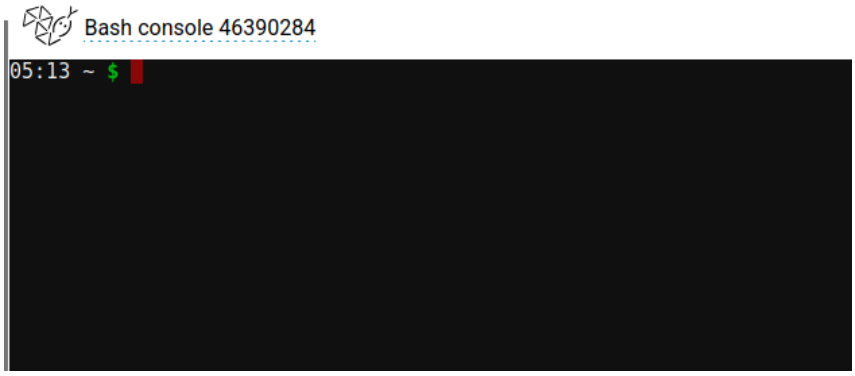
Gambar 4.1: Buka Dasbor PythonAnywhere

2. Di bagian **Console** sebelah kiri. Klik tombol **$ Bash** di bawah **New Console**



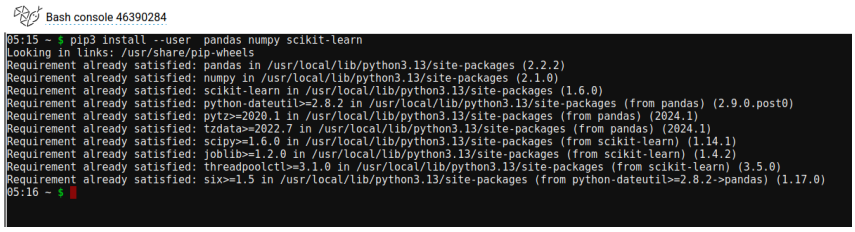
Gambar 4.2: Akses **Bash**

3. **PythonAnywhere** akan membuat konsol baru. Perhatikan gambar berikut



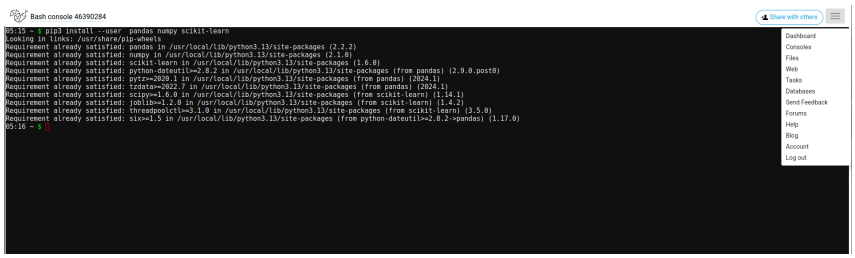
Gambar 4.3: Tampilan **Bash**

4. Install library berikut dengan perintah:
   - `pip3 install --user  pandas numpy scikit-learn`



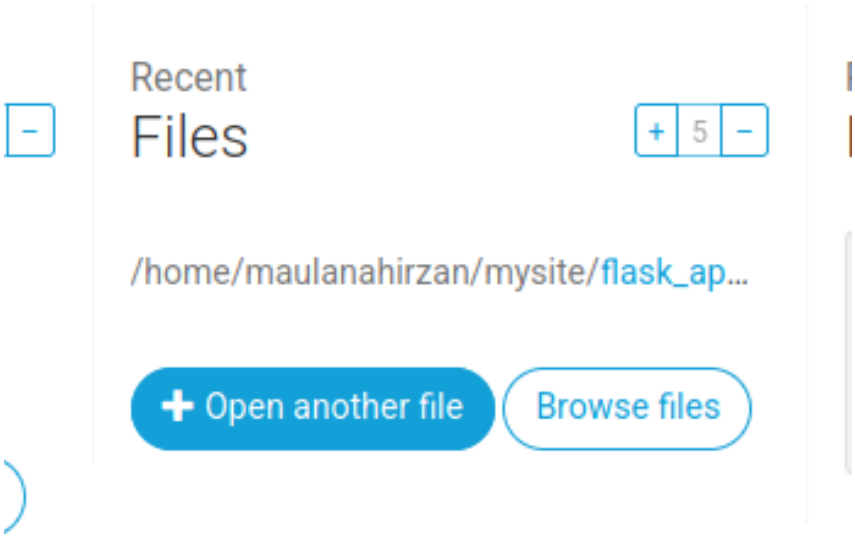
Gambar 4.4: Tampilan Instalasi Library

5. Setelah selesai, kembali ke dasbor dengan mengklik **Strip Tiga** di Kanan Atas, dan klik **Dashbaord**



Gambar 4.5: Kembali ke Dasbor

6. Di bagian **Files** klik **Browse Files**



Gambar 4.6: Membuka File

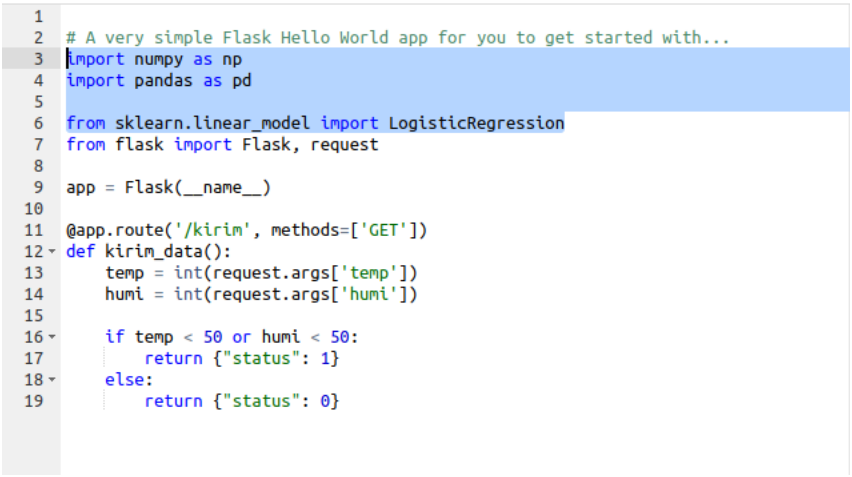
7. Klik **Folder mysite** lalu klik **flask_app.py** untuk memunculkan kode yang telah dibuat sebelumnya

Gambar 4.7: Membuka File

8. Di bagian atas kode `from flask import Flask, request`, masukkan kode berikut

Potongan Kode

```
import numpy as np
import pandas as pd

from sklearn.linear_model import LogisticRegression
```



```
1
2  # A very simple Flask Hello World app for you to get started with...
3  import numpy as np
4  import pandas as pd
5
6  from sklearn.linear_model import LogisticRegression
7  from flask import Flask, request
8
9  app = Flask(__name__)
10
11 @app.route('/kirim', methods=['GET'])
12 def kirim_data():
13     temp = int(request.args['temp'])
14     humi = int(request.args['humi'])
15
16     if temp < 50 or humi < 50:
17         return {"status": 1}
18     else:
19         return {"status": 0}
```

Gambar 4.8: Menambahkan Library

9. Lanjutkan kode dengan menambahkan kode berikut setelah bagian `app = Flask(__name__)`

Potongan Kode

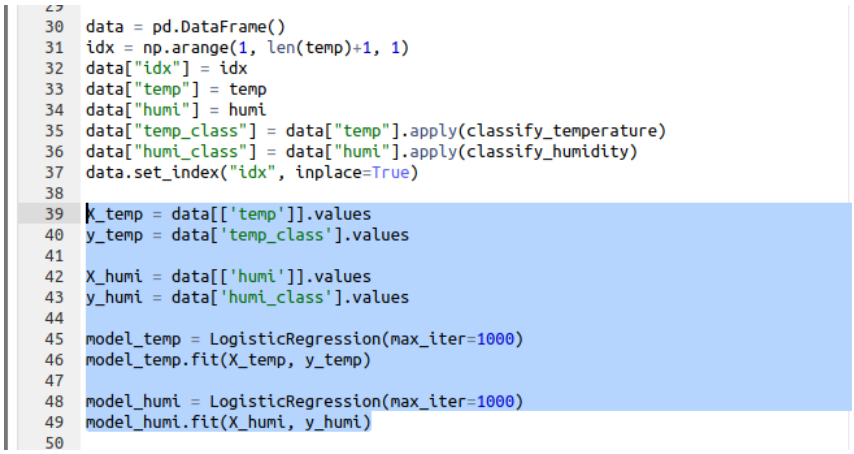
```
temp = np.arange(0, 100, 1)
humi = np.arange(0, 100, 1)

def classify_temperature(temp):
    if temp < 25:
        return 0
    elif temp < 60:
        return 1
    else:
        return 2

def classify_humidity(humi):
    if humi < 25:
        return 0
    elif humi < 60:
        return 1
    else:
        return 2

data = pd.DataFrame()
idx = np.arange(1, len(temp)+1, 1)
data["idx"] = idx
data["temp"] = temp
data["humi"] = humi
data["temp_class"] = data["temp"].apply(classify_temperature)
data["humi_class"] = data["humi"].apply(classify_humidity)
data.set_index("idx", inplace=True)
```

Gambar 4.9: Menambahkan Kode Parameter

10. Lanjutkan kode sebelumnya dengan kode berikut untuk melatih model

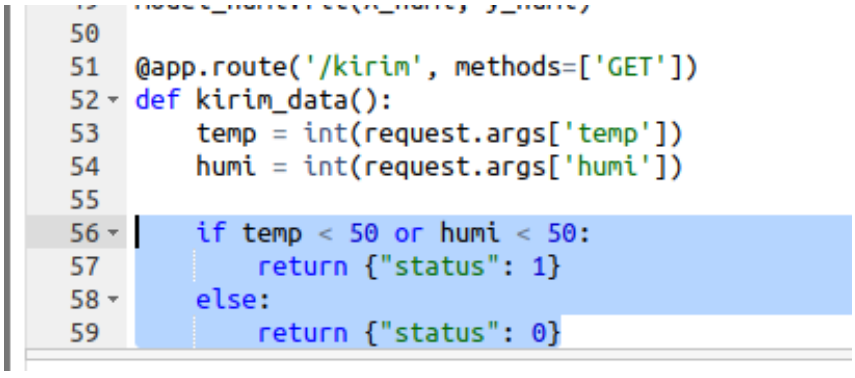
Potongan Kode

```
X_temp = data[['temp']].values
y_temp = data['temp_class'].values

X_humi = data[['humi']].values
y_humi = data['humi_class'].values

model_temp = LogisticRegression(max_iter=1000)
model_temp.fit(X_temp, y_temp)

model_humi = LogisticRegression(max_iter=1000)
model_humi.fit(X_humi, y_humi)
```



Gambar 4.10: Menambahkan Kode Melatih Model

11. Langkah berikutnya adalah mengganti kode **kirim** agar menggunakan model yang sudah dilatih sebelumnya. Hapus bagian **if** dan **else** dari kode **kirim**



Gambar 4.11: Menghapus Kode If Else

12. Masukkan potongan kode di dalam fungsi **Kirim** seperti berikut:

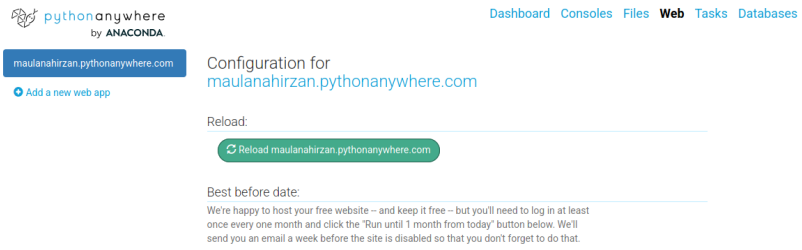
Potongan Kode

```
test_temp = np.array([[temp]])
test_humi = np.array([[humi]])

pred_temp = model_temp.predict(test_temp)[0]
pred_humi = model_humi.predict(test_humi)[0]

return {"temp": str(pred_temp), "humi": str(pred_humi)}
```

```
50
51  @app.route('/kirim', methods=['GET'])
52  def kirim_data():
53      temp = int(request.args['temp'])
54      humi = int(request.args['humi'])
55
56      test_temp = np.array([[temp]])
57      test_humi = np.array([[humi]])
58
59      pred_temp = model_temp.predict(test_temp)[0]
60      pred_humi = model_humi.predict(test_humi)[0]
61
62      return {"temp": str(pred_temp), "humi": str(pred_humi)}
```

Gambar 4.12: Menambahkan Kode Prediksi

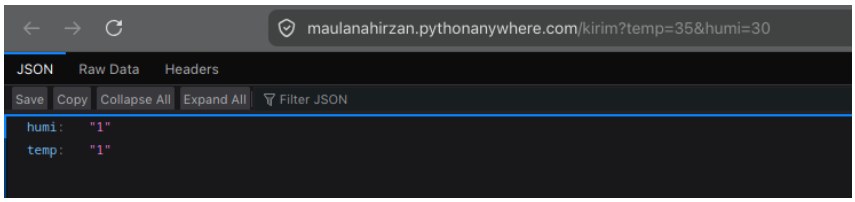
13. Simpan kode, lalu masuk ke dasbor **Web App** dan klik **Reload**



Gambar 4.13: Me-Reload Web App

14. Coba aplikasi dengan membuka URL berikut, atur nilai **temp** dan **humi** sesuka hati. Ubah <username> di url sesuai domain masing-masing mahasiswa

    - `https://<username>.pythonanywhere.com/kirim?temp=35&humi=30`



Gambar 4.14: Hasil Tes Data

Bab 5

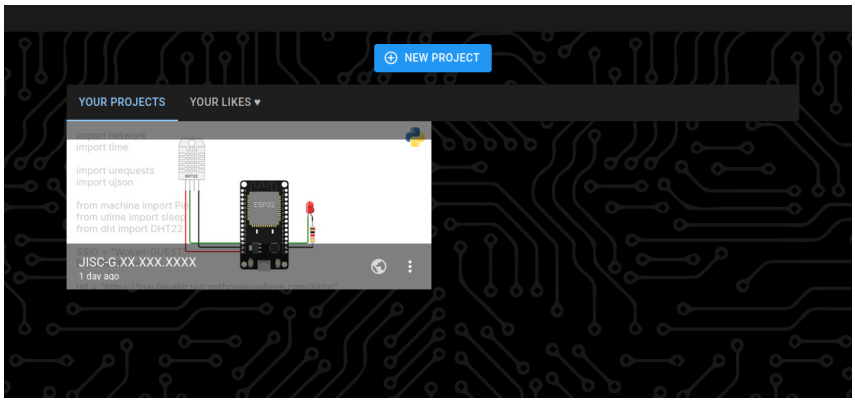
Integrasi Inference API dengan Wokwi

5.1 Pendahuluan

Praktikum pada pertemuan ini difokuskan pada integrasi layanan Inference API dengan simulasi perangkat IoT menggunakan Wokwi. Kegiatan ini bertujuan untuk memberikan pemahaman mengenai alur komunikasi antara perangkat berbasis mikrokontroler dan layanan komputasi berbasis cloud, khususnya dalam mengirimkan data sensor, melakukan proses inferensi menggunakan model lightweight, serta menerima respons untuk mengendalikan aktuator secara real-time. Melalui praktikum ini, diharapkan terbentuk pemahaman konseptual dan implementatif terkait penerapan sistem cerdas terdistribusi dalam skenario Internet of Things (IoT).

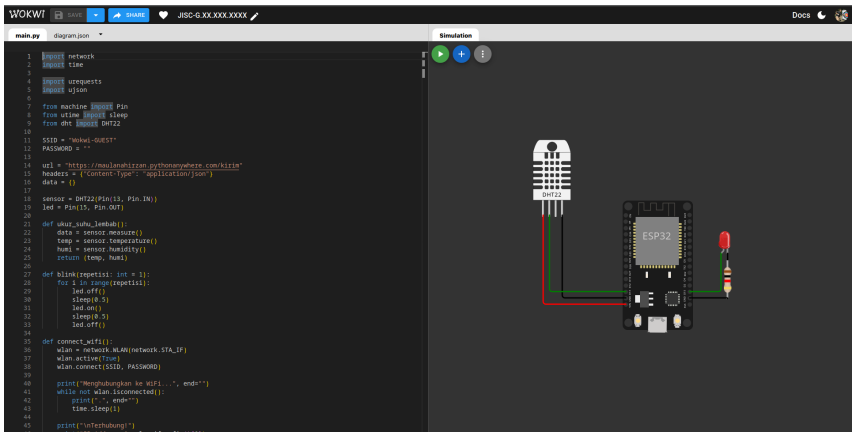
5.2 Tutorial

1. Buka website **wokwi.com** untuk mengakses kembali praktikum sebelumnya melalui **Foto Profil** => **My Projects**



Gambar 5.1: Membuka Projek Sebelumnya

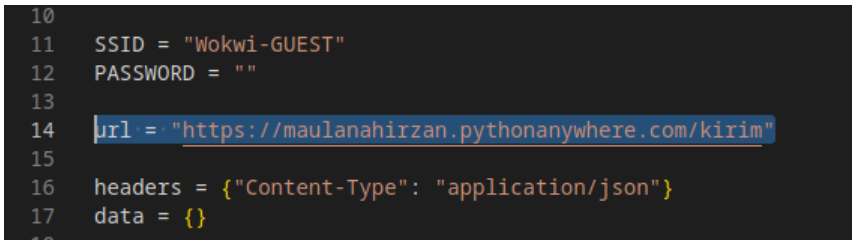
2. Pastikan projek sebelumnya sudah terbuka lengkap



Gambar 5.2: Membuka Projek Sebelumnya

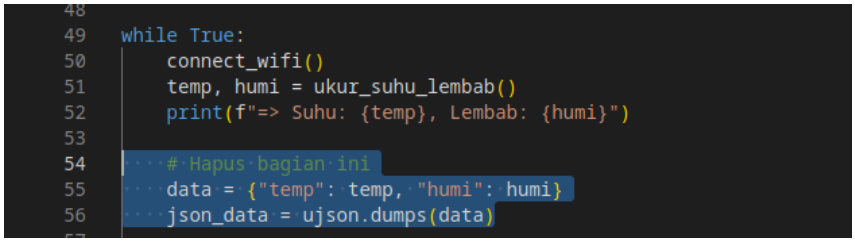
3. Di bagian dalam kode, ada yang perlu dirubah agar dapat melakukan pengiriman data ke **Web App** yang sudah dibuat sebelumnya.

   (a) Bagian **url**, ubah alamat menjadi "`https://<username>.pythonanywhere.com/kirim`". Pastikan ganti <**username**> dengan nama mahasiswa masing-masing



Gambar 5.3: Update URL WebApp

   (b) Di dalam blok **while True:** hapus bagian kode **ujson**. Lihat gambar

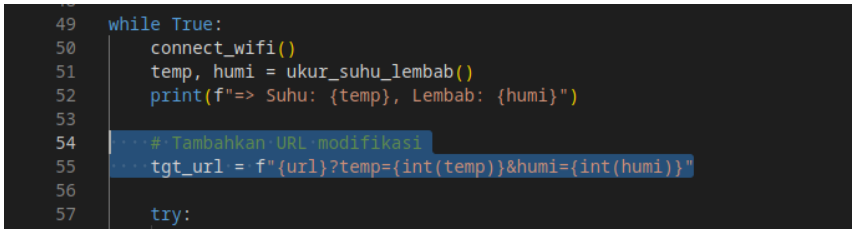


Gambar 5.4: Hapus bagian kode data dan ujson

   (c) Tambahkan variabel baru yang memodifikasi **url** di atas kode **try:**

Potongan Kode

```
tgt_url = f"{url}?temp={int(temp)}&humi={int(humi)}"
```



Gambar 5.5: Menambahkan URL Modifikasi

(d) Di dalam blok kode **try:**, modifikasi `resp = urequests.post(...)` menjadi:

Potongan Kode

```
resp = urequests.get(tgt_url)
```

```
57      try:
58          print("=> Mengirimkan Data")
59          resp = urequests.get(tgt_url)
60
61
```

Gambar 5.6: Ubah kode urequests

(e) Tambahkan baris kode baru yang diletakkan setelah `status = resp.status_code`

Potongan Kode

```
data = resp.json()
```

```
57      try:
58          print("=> Mengirimkan Data")
59          resp = urequests.get(tgt_url)
60          status = resp.status_code
61          # Tambah
62          data = resp.json()
63          resp.close()
64
```

Gambar 5.7: Mengambil data dari ReST API

(f) Berikutnya adalah memodifikasi blok `if else`. Tambahkan kode berikut di blok 1 sebelum `blink(3)`

Potongan Kode

```
# Tambah
class_temp = data['temp']
class_humi = data['humi']
print("==> Sukses")
```

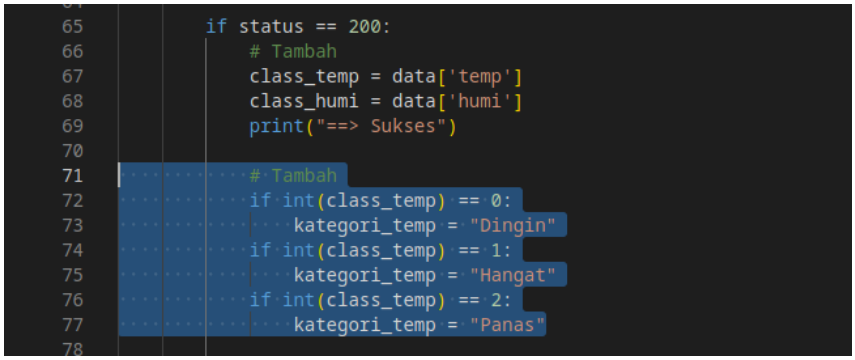
```
64
65          if status == 200:
66              # Tambah
67              class_temp = data['temp']
68              class_humi = data['humi']
69              print("==> Sukses")
70
```

Gambar 5.8: Menambahkan Kode Tampilan

(g) Setelah itu kita tambahkan kode untuk klasifikasi yang mengubah menjadi kategori Suhu : Panas, Hangat, Dingin, dan Kelembaban : Basah, Lembab, dan Kering. Letakkan setelah `print("==> Sukses")`

Potongan Kode

```
# Tambah
if int(class_temp) == 0:
    kategori_temp = "Dingin"
if int(class_temp) == 1:
    kategori_temp = "Hangat"
if int(class_temp) == 2:
    kategori_temp = "Panas"
```



Gambar 5.9: Menambahkan Klasifikasi Suhu

(h) Lanjutkan dengan klasifikasi kelembaban

Potongan Kode

```
# Tambah
if int(class_humi) == 0:
    kategori_humi = "Kering"
if int(class_humi) == 1:
    kategori_humi = "Lembab"
if int(class_humi) == 2:
    kategori_humi = "Basah"
```

Gambar 5.10: Menambahkan Klasifikasi Kelembaban

(i) Di bagian akhir kode sebelum baris `blink(3)`, masukkan kode berikut:

Potongan Kode

```
print("=> Hasil Klasifikasi Suhu dan Kelembaban")
print(f"==> Temp : {class_temp} ({kategori_temp})")
print(f"==> Humi : {class_humi} ({kategori_humi})")
```

```
87          print("=> Hasil Klasifikasi Suhu")
88          print(f"==> Temp : {class_temp} ({kategori_temp})")
89          print(f"==> Humi : {class_humi} ({kategori_humi})")
90          blink(3)
```

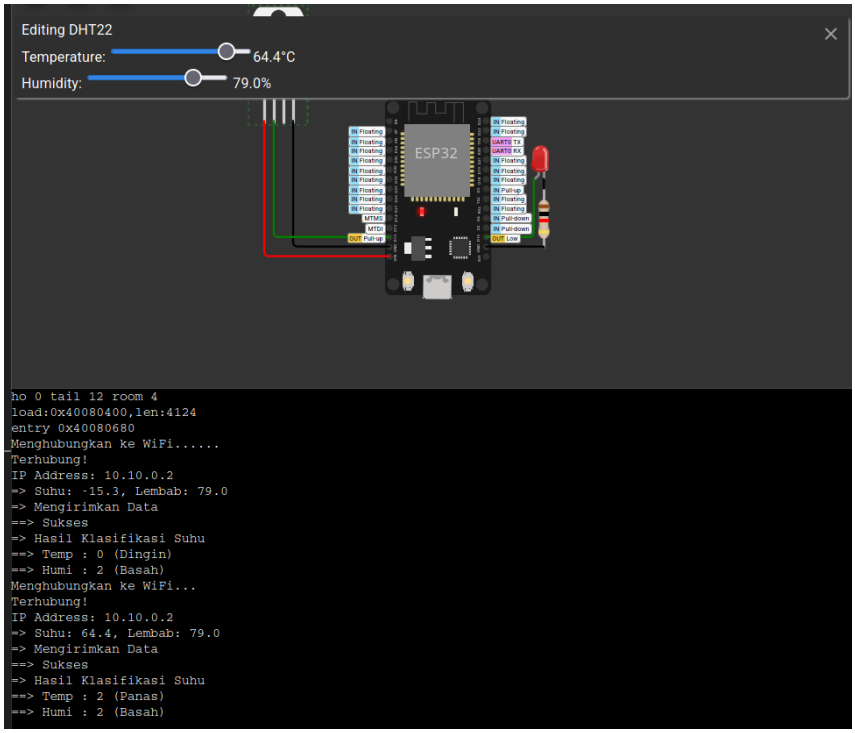
Gambar 5.11: Menampilkan Kode Hasil

4. Jika sudah selesai, coba **Run** kode tersebut dan lihat hasil nya seperti berikut

```
entry 0x40080680
Menghubungkan ke WiFi......
Terhubung!
IP Address: 10.10.0.2
=> Suhu: -15.3, Lembab: 79.0
=> Mengirimkan Data
==> Sukses
=> Hasil Klasifikasi Suhu
==> Temp : 0 (Dingin)
==> Humi : 2 (Basah)
```

Gambar 5.12: Hasil Kode

5. Klik sensor **DHT22**, dan geser Slider **Temperature** dan **Humidity** untuk melihat bacaan sensor yang berbeda, dan hasil prediksi yang berbeda



Gambar 5.13: Ubah Slider Sensor

Bab 6

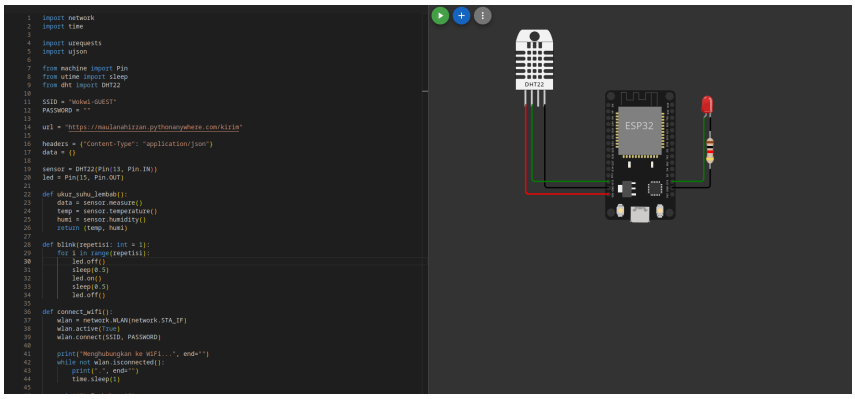
Implementasi Aktuator

6.1 Pendahuluan

Praktikum ini membahas implementasi aktuator pada sistem jaringan dan infrastruktur cerdas dengan memanfaatkan integrasi antara perangkat IoT berbasis ESP32 di Wokwi dan layanan REST API sebagai pusat pengambilan keputusan. Dalam skenario ini, data sensor dikirimkan ke model yang telah tersedia melalui protokol HTTP, kemudian sistem menerima response berupa perintah kontrol yang digunakan untuk mengaktifkan atau menonaktifkan aktuator seperti LED, buzzer, atau relay. Pendekatan ini merepresentasikan konsep closed-loop system pada Internet of Things, di mana proses sensing, decision-making, dan actuation berlangsung secara terintegrasi antara edge device dan layanan berbasis cloud, sehingga memberikan pemahaman praktis mengenai penerapan inference jarak jauh dan kontrol perangkat secara real-time.

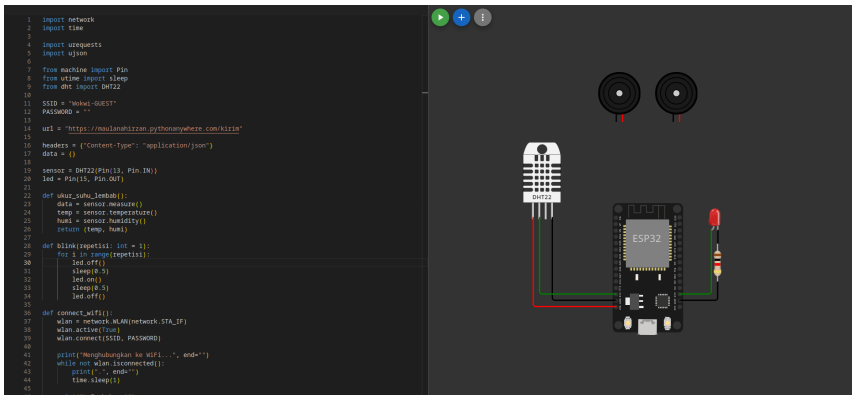
6.2 Tutorial

1. Buka **wokwi.com** dan buka kembali projek yang telah dibuat



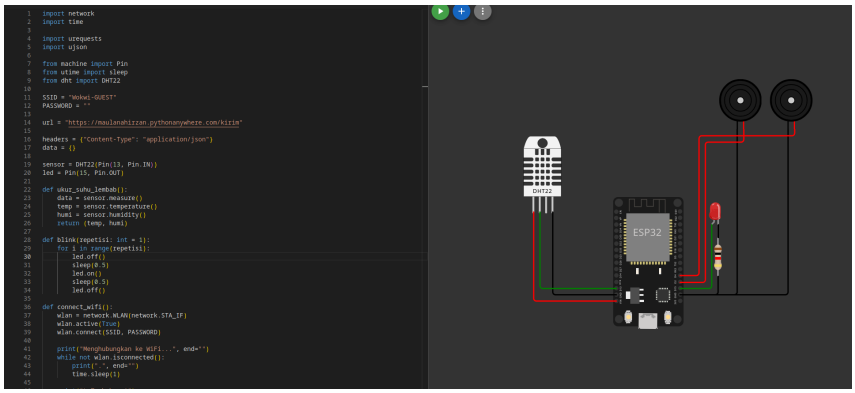
Gambar 6.1: Buka Kembali Projek

2. Masukkan dua komponen baru berupa **Buzzer**



Gambar 6.2: Tambahkan Dua Buzzer

3. Sambungkan komponen kedua buzzer ke ESP32 dengan konfigurasi:
   - **Buzzer 1**
     - bz1:1 (kabel hitam) → esp:GND.1
     - bz1:2 (kabel merah) → esp:D2
   - **Buzzer 2**
     - bz2:1 (kabel hitam) → esp:GND.1
     - bz2:2 (kabel merah) → esp:D4



Gambar 6.3: Koneksi Buzze ke ESP32

4. Di bagian kode editor, letakkan kode baru di bawah kode `led = Pin(15, Pin.OUT)` dan sebelum kode `def ukur_suhu_lembab():`

Potongan Kode

```
bz1 = Pin(2, Pin.OUT)
bz2 = Pin(4, Pin.OUT)
```

```
18
19  sensor = DHT22(Pin(13, Pin.IN))
20  led = Pin(15, Pin.OUT)
21
22  bz1 = Pin(2, Pin.OUT)
23  bz2 = Pin(4, Pin.OUT)
24
25  def ukur_suhu_lembab():
26      data = sensor.measure()
27      temp = sensor.temperature()
28      humi = sensor.humidity()
29      return (temp, humi)
30
```

Gambar 6.4: Kode Inisialisasi Buzzer

5. Buat sebuah fungsi yang digunakan untuk menyalakan buzzer dengan durasi tertentu secara programatik, letakkan kode setelah kode buzzer

Potongan Kode

```
def trigger_buzz(bz: int, durasi: int):
    if bz == 1:
        bz1.on()
        sleep(durasi)
        bz1.off()

    if bz == 2:
        bz2.on()
        sleep(durasi)
        bz2.off()
```

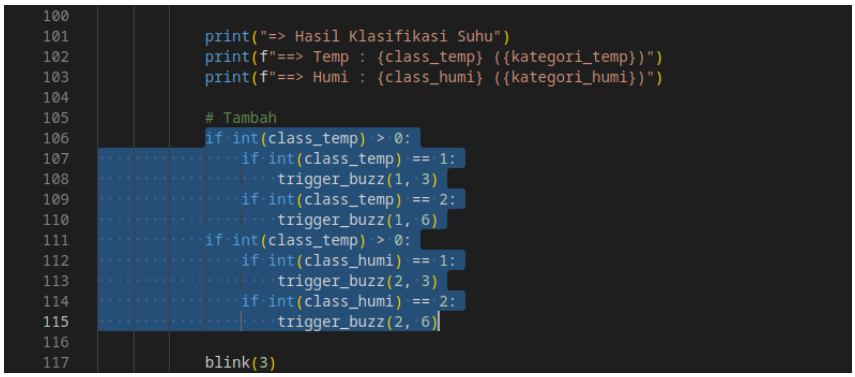
```
24
25  def trigger_buzz(bz: int, durasi: int):
26      if bz == 1:
27          bz1.on()
28          sleep(durasi)
29          bz1.off()
30      if bz == 2:
31          bz2.on()
32          sleep(durasi)
33          bz2.off()
34
35  def ukur_suhu_lembab():
36      data = sensor.measure()
37      temp = sensor.temperature()
38      humi = sensor.humidity()
39      return (temp, humi)
```

Gambar 6.5: Kode Aktivasi Buzzer

6. Kode fungsi ini akan dipanggil dengan ketentuan:
   - Panjang bunyi tergantung jenis kelas
   - Buzzer yang dinyalakan tergantung sensor yang terpicu
7. Untuk menggunakan kode ini, masuk ke blok kode `while True:` dan letakkan kode berikut sebelum kode `blink(3)`

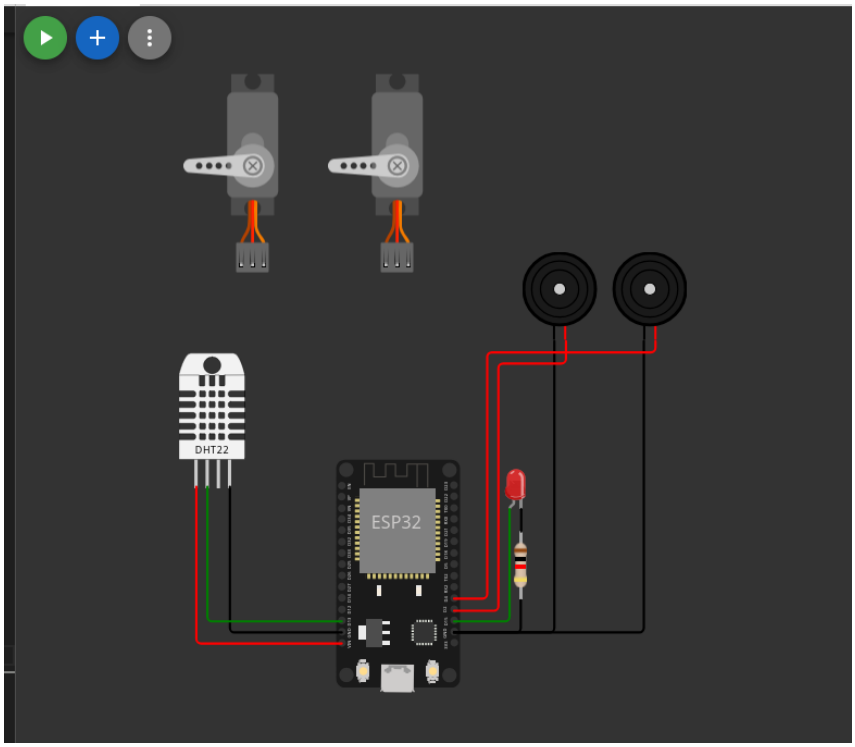
Potongan Kode

```
if int(class_temp) > 0:
    if int(class_temp) == 1:
        trigger_buzz(1, 3)
    if int(class_temp) == 2:
        trigger_buzz(1, 6)
if int(class_temp) > 0:
    if int(class_humi) == 1:
        trigger_buzz(2, 3)
    if int(class_humi) == 2:
        trigger_buzz(2, 6)
```



Gambar 6.6: Mengaktifkan Buzzer

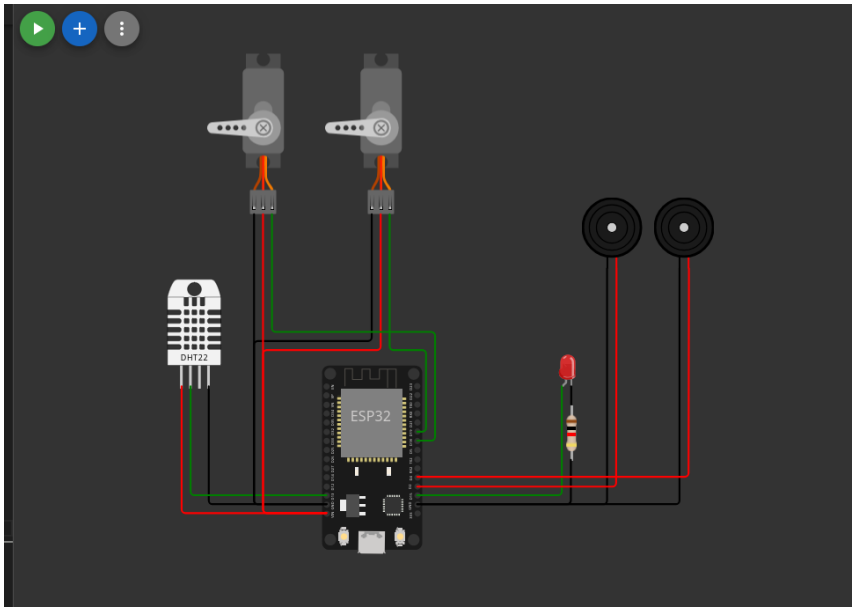
8. Berikutnya adalah menambahkan aktuator berupa motor. Masukkan dua (2) Servo Motor ke Wokwi



Gambar 6.7: Mengaktifkan Buzzer

9. Sambungkan kedua servo dengan konfigurasi berikut

   - **Servo 1**
     - servo1:GND → esp:GND.2
     - servo1:V+ → esp:VIN
     - servo1:PWM → esp:D18
   - **Servo 2**
     - servo2:GND → esp:GND.2
     - servo2:V+ → esp:VIN
     - servo2:PWM → esp:D19



Gambar 6.8: Menghubungkan Servo ke ESP32

10. Inisialisasikan masing-masing Servo ke dalam kode. Letakkan kode untuk servo setelah kode inisialisasi buzzer

Potongan Kode

```
srv1 = PWM(Pin(18), freq=50)
srv2 = PWM(Pin(19), freq=50)
```

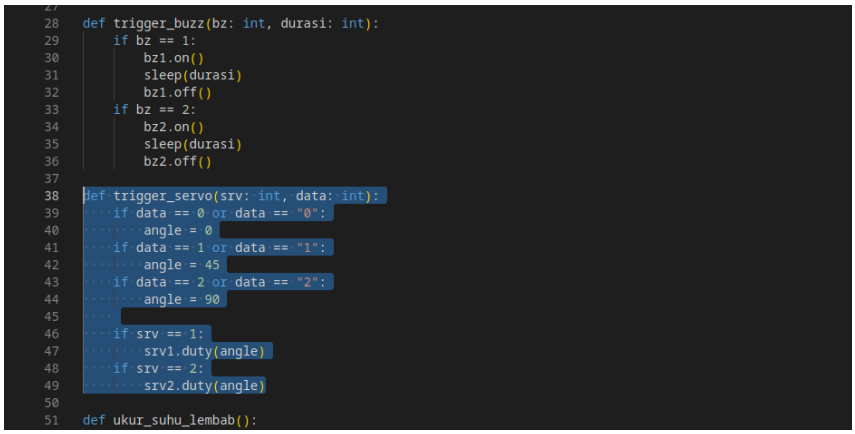
```
21
22  bz1 = Pin(2, Pin.OUT)
23  bz2 = Pin(4, Pin.OUT)
24
25  srv1 = PWM(Pin(18), freq=50)
26  srv2 = PWM(Pin(19), freq=50)
27
28  def trigger_buzz(bz: int, durasi: int):
```

Gambar 6.9: Kode Inisialisasi Servo

11. Untuk mengaktifkan servo, diperlukan kode yang terpisah. letakkan kode berikut setelah blok `def trigger_buzz(bz: int, durasi: int):`

Potongan Kode

```
def trigger_servo(srv: int, data: int):
    if data == 0 or data == "0":
        angle = 0
    if data == 1 or data == "1":
        angle = 45
    if data == 2 or data == "2":
        angle = 90

    if srv == 1:
        srv1.duty(angle)
    if srv == 2:
        srv2.duty(angle)
```

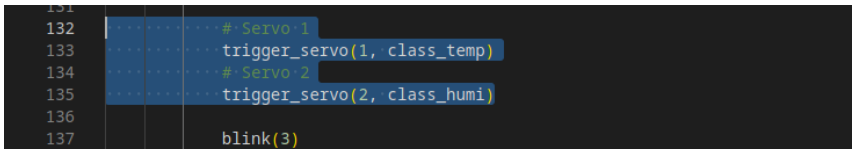


Gambar 6.10: Kode Fuungsional Servo

12. Untuk mengaktifkan kode ini, cukup menambahkan dua baris kode sederhana sebelum `blink(3)`

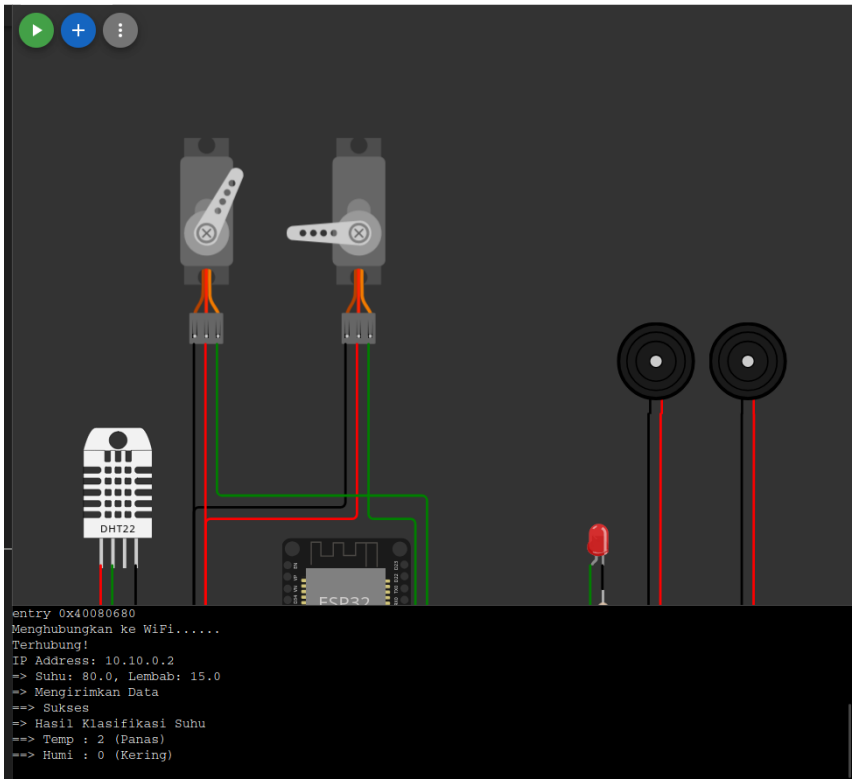
Potongan Kode

```
# Servo 1
trigger_servo(1, class_temp)
# Servo 2
trigger_servo(2, class_humi)
```



Gambar 6.11: Kode Fungsional Servo

13. Jalankan kode dan perhatikan pergerakan servo sesuai indikasi dari sistem cerdas



Gambar 6.12: Memanggil Fungsional Servo

Bab 7

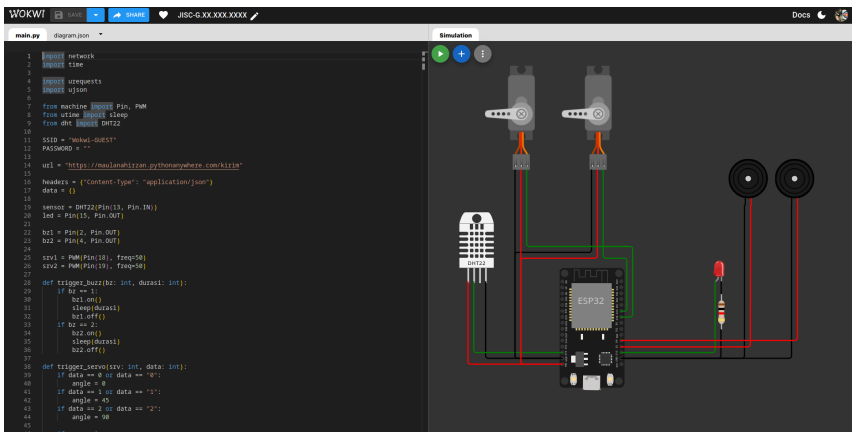
Pengujian dan Pengukuran Kinerja Sistem Cerdas

7.1 Pendahuluan

Praktikum ini bertujuan untuk memberikan pemahaman komprehensif mengenai proses pengujian dan pengukuran kinerja pada sistem cerdas berbasis Internet of Things (IoT) yang diimplementasikan menggunakan MicroPython pada perangkat edge seperti ESP32. Dalam konteks sistem cerdas modern, evaluasi performa menjadi aspek krusial untuk memastikan bahwa proses akuisisi data, komunikasi jaringan, hingga inferensi berbasis kecerdasan buatan dapat berjalan secara efisien, responsif, dan andal. Oleh karena itu, praktikum ini akan memfokuskan pada pengukuran parameter utama seperti latency, throughput, penggunaan memori, serta reliability sistem melalui interaksi antara perangkat MicroPython dan layanan inference berbasis API. Hasil pengujian diharapkan mampu memberikan gambaran nyata mengenai keterbatasan dan potensi sistem cerdas pada lingkungan edge computing, sehingga dapat menjadi dasar dalam pengembangan sistem yang lebih optimal dan adaptif terhadap kondisi operasional yang dinamis.

7.2 Tutorial

1. Buka kembali projek Wokwi yang sudah dibuat sebelumnya.



Gambar 7.1: Membuka Projek Lama

2. Masukkan kode berikut untuk menambahkan variabel baru dan letakkan di bawah kode servo

   - start_time (float)
   - end_time (float)
   - elapsed_time (float)

Potongan Kode

```
start_time: float = 0.0
end_time: float = 0.0
elapsed_time: float = 0.0
```

Gambar 7.2: Kode Variabel Timer

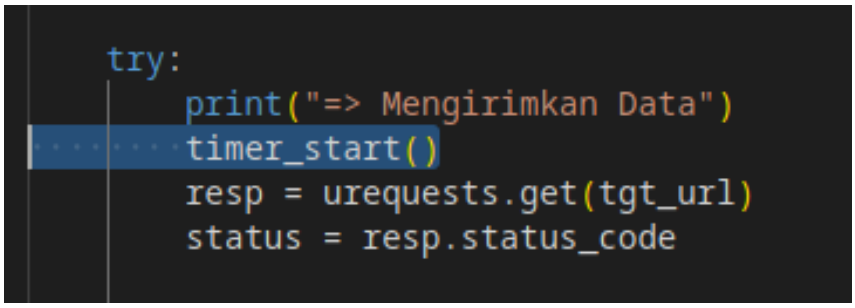
3. Agar variabel ini berfungsi, buat tiga fungsi dan letakkan di bawah kode blok `blink` dan di atas blok kode `connect_wifi`

Potongan Kode

```
def timer_start():
    global start_time
    start_time = time.time()

def timer_stop():
    global end_time
    end_time = time.time()

def timer_elapse():
    global start_time, end_time, elapsed_time
    elapsed_time = end_time - start_time
```

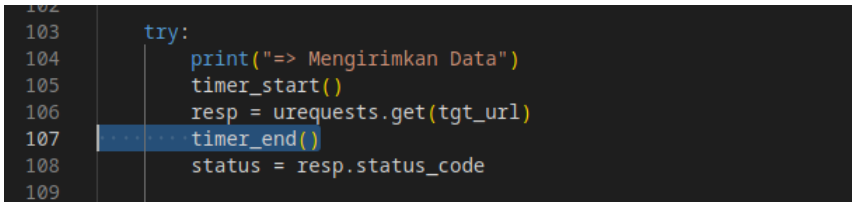
Gambar 7.3: Kode Timer

4. Letakkan fungsi `timer_start()` sebelum baris kode `resp = urequests.get(tgt_url)`



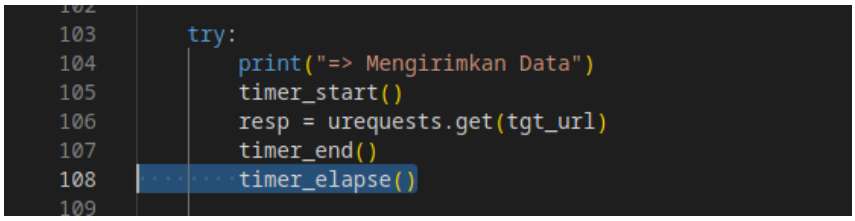
Gambar 7.4: Menambahkan Start Timer

5. Tambahkan kode `timer_stop()` setelah baris kode `resp = urequests.get(tgt_url)`



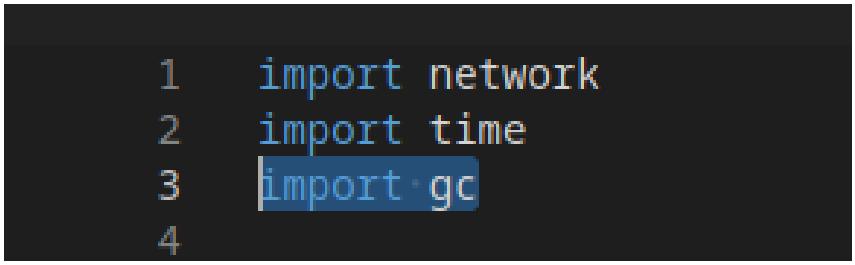
Gambar 7.5: Menambahkan Start Timer

6. Agar timer dikalkulasi, tambahkan baris kode `timer_elapse()` setelah `timer_end()`



Gambar 7.6: Menambahkan Timer Elapse

7. Berikutnya adalah melakukan *benchmark* untuk penggunaan memori. Pertama adalah menambahkan *library* baru dengan `import gc` dan letakkan setelah `import time`

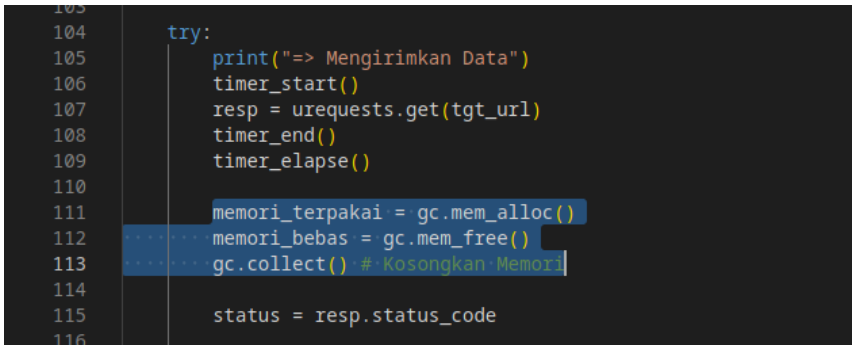


Gambar 7.7: Menambahkan Library Memory

8. Di bawah kode timer, tambahkan fungsi berikut untuk menghitung penggunaan memori dan memori bebas

Potongan Kode

```
memori_terpakai = gc.mem_alloc()
memori_bebas = gc.mem_free()
gc.collect() # Kosongkan Memori
```



Gambar 7.8: Penggunaan Memori dan Memori Kosong

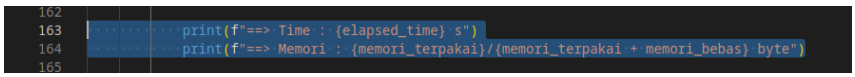
9. Tambahkan `gc.enable()` sebelum baris kode `while True:` untuk memastikan pengumpulan memori diaktifkan

Gambar 7.9: Mengaktifkan Pengumpul Memori

10. Untuk menampilkan waktu yang berjalan dan penggunaan memori ketika melakukan request, tambahkan kode berikut sebelum `blink(3)`

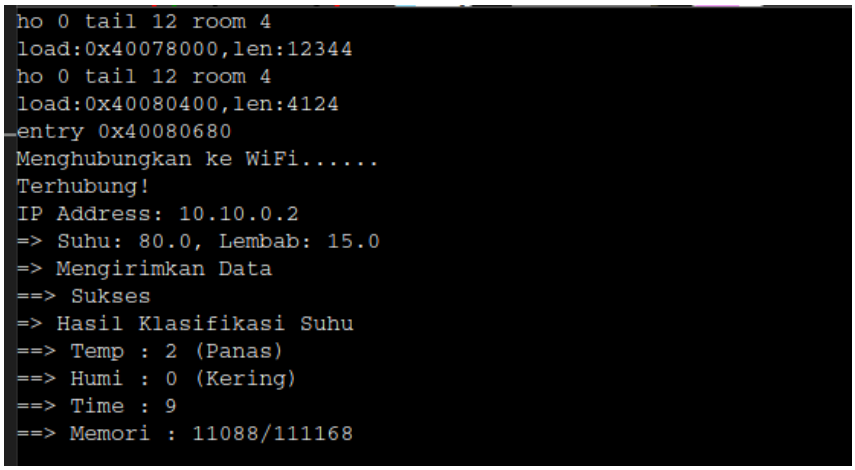
Potongan Kode

```
print(f"==> Time : {elapsed_time}s")
print(f"==> Memori : {memori_terpakai}/{memori_terpakai+memori_bebas}b")
```



Gambar 7.10: Menampilkan Kinerja

11. Jalankan simulasi dan lihat hasil benchmark



Gambar 7.11: Menampilkan Kinerja