RANCANGAN APLIKASI IDENTIFIKASI KEBAKARAN DENGAN MENGGUNAKAN APLIKASI PWA BERBASIS RASPBERRY PI DI POLITEKNIK PENERBANGAN SURABAYA

TUGAS AKHIR

Oleh :

MOCHAMMAD RIZQI NUR WAHYUDI

NIT. 30218015

PROGRAM STUDI DIPLOMA 3 TEKNIK NAVIGASI UDARA

POLITEKNIK PENERBANGAN SURABAYA

2021

RANCANGAN APLIKASI IDENTIFIKASI KEBAKARAN DENGAN MENGGUNAKAN APLIKASI PWA BERBASIS RASPBERRY PI DI POLITEKNIK PENERBANGAN SURABAYA

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Mendapatkan Gelar Ahli Madya
(A. Md) pada Program Studi Diploma 3 Teknik Navigasi Udara

Oleh :

MOCHAMMAD RIZQI NUR WAHYUDI

NIT. 30218015

PROGRAM STUDI DIPLOMA 3 TEKNIK NAVIGASI UDARA

POLITEKNIK PENERBANGAN SURABAYA

2021

HALAMAN PERSETUJUAN

RANCANGAN APLIKASI IDENTIFIKASI KEBAKARAN DENGAN MENGGUNAKAN APLIKASI PWA BERBASIS RASPBERRY PI DI POLITEKNIK PENERBANGAN SURABAYA

Oleh:

Mochammad Rizqi Nur Wahyudi
NIT. 30218015

Disetujui untuk diujikan pada:

12 Agustus 2021

Pembimbing 1 : NYARIS PAMBUDIYATNO, S.SiT, M.MTr .....
NIP. 19820525 200502 1001

Pembimbing 2 : ROMMA DIANA PUSPITA,S.SIT
NIP. 19820507 200502 2 002

LEMBAR PENGESAHAN

RANCANGAN APLIKASI IDENTIFIKASI KEBAKARAN DENGAN MENGGUNAKAN APLIKASI PWA BERBASIS RASPBERRY PI DI POLITEKNIK PENERBANGAN SURABAYA

Oleh:

Mochammad Rizqi Nur Wahyudi
NIT. 30218015

Telah dipertahankan dan dinyatakan lulus pada Ujian Tugas Akhir
Program Pendidikan Diploma 3 Teknik Navigasi Udara
Politeknik Penerbangan Surabaya
Pada tanggal :

12 Agustus 2021

Panitia Penguji :

1. Ketua : Dr. YUYUN SUPRAPTO, S.SiT, MM
   NIP. 19810529 200812 1001 .....

2. Sekretaris : EFRIED NARA P., S.SiT, MM
   NIK. ASN83380 .....

3. Anggota : NYARIS PAMBUDIYATNO, S.SiT, M.MTr
   NIP. 19820525 200502 1001 .....

Ketua Program Studi
D. 3 Teknik Navigasi Udara

NYARIS PAMBUDIYATNO, S.SiT, M.MTr
NIP. 19820525 200502 1001

PERNYATAAN KEASLIAN DAN HAK CIPTA

Saya yang bertanda tangan dibawah ini:

Nama	: Mochammad Rizqi Nur Wahyudi
NIT	: 30218015
Program Stud	: D3 Teknik Navigasi Udara
Judul Tugas Akhir	: Rancangan Aplikasi Identifikasi Kebakaran Dengan Menggunakan Aplikasi PWA Berbasis Raspberry Pi Di Politeknik Penerbangan Surabaya

dengan ini menyatakan bahwa :

1. Tugas Akhir ini merupakan karya asli dan belum pernah diajukan untuk mendapatkan gelar akademik, baik di Politeknik Penerbangan Surabaya maupun di Perguruan Tinggi lain, serta dipublikasikan, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.

2. Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan Hak Bebas Royalti Non Eksklusif (*Non-Exclusive Royalty-Free Right*) kepada Politeknik Penerbangan Surabaya beserta perangkat yang ada (jika diperlukan). Dengan hak ini, Politeknik Penerbangan Surabaya berhak menyimpan, mengalih media/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya dengan tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya. Apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di Politeknik Penerbangan Surabaya.

Surabaya,
Yang membuat pernyataan

Mochammad Rizqi Nur Wahyudi
NIT. 30218015

KATA PENGANTAR

Puji syukur penulis munajatkan kehadirat Allah SWT, karena atas berkah, rahmat dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir ini sesuai waktu yang telah ditentukan. Penelitian Tugas Akhir dengan judul "RANCANG BANGUN SISTEM PENANGANAN KEBAKARAN BERBASIS RASPBERRY PI DENGAN APLIKASI ANDROID DI POLITEKNIK PENERBANGAN SURABAYA" yang diajukan untuk memenuhi syarat dalam menyelesaikan pendidikan di Politeknik Penerbangan Surabaya dan memperoleh gelar Ahli Madya (A.Md).

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan semua pihak yang memberikan arahan dan bimbingannya, untuk itu penulis mengucapkan terima kasih kepada :

1. Allah SWT yang telah memberikan kelancaran dan kesehatan kepada penulis dalam penyusunan penelitian Tugas Akhir.
2. Ayahanda Suparna dan Ibunda Mujiatminingsih yang senantiasa memberikan do'a dan untuk keberhasilan penulis dan menjadi sumber semangat hidup dalam mengikuti pendidikan.
3. Bapak M. Andra Adityawarman, MT. selaku Direktur Politeknik Penerbangan Surabaya.
4. Bapak Nyaris Pambudiyatno, S.SiT, M.MTr selaku Ketua Program Studi Teknik Navigasi Udara sekaligus dosen pembimbing I di Politeknik Penerbangan Surabaya.
5. Ibu Romma Diana Puspita,S.SiT selaku pembimbing II, atas bimbingannya
6. Seluruh dosen dan civitas akademika Program Studi D3 Teknik Navigasi Udara Politeknik Penerbangan Surabaya atas pengajaran.
7. Teman-teman sekelas D3 Teknik Navigasi Udara XI, teman – teman seangkatan dan adik-adik tingkat II dan tingkat I yang selalu memberikan support, motivasi, kebersamaan dan kerjasamanya.

8. Semua pihak yang tidak dapat penulis tuliskan satu persatu yang telah membantu secara sukarela dalam segala keperluan saya selama menyelesaikan Tugas Akhir ini.

Akhirnya, penulis menyadari bahwa dalam pembuatan Tugas Akhir ini masih belum sempurna dan masih banyak hal yang perlu diperbaiki serta terdapat kekurangan dari penulis. Penulis mengharapkan kritik dan saran yang bersifat membangun dari semua pihak, untuk kesempurnaan Tugas Akhir ini. Penulis berharap semoga tugas akhir ini dapat memberikan manfaat bagi semua pihak, khususnya kepada semua Taruna di Politeknik Penerbangan Surabaya.

Surabaya, Agustus 2021

Mochammad Rizqi Nur Wahyudi

ABSTRAK

RANCANGAN APLIKASI IDENTIFIKASI KEBAKARAN DENGAN MENGGUNAKAN APLIKASI PWA BERBASIS RASPBERRY PI DI POLITEKNIK PENERBANGAN SURABAYA

Oleh:

Mochammad Rizqi Nur Wahyudi

NIT. 30218015

Kebakaran adalah musibah yang mengkhawatirkan bagi pihak yang mengalaminya karena dapat mengakibatkan berbagai kerugian. Bangunan gedung, perabotan, dan bahkan dokumen ikut hangus terbakar. Dalam tugas akhir ini, prototipe pendeteksi kebakaran akan didesain dengan hasil output berupa notifikasi pesan yang masuk melalui aplikasi *android* dan pesan khusus yang tersambung langsung dengan pemadam kebakaran.

Hasil tersebut diperoleh dari proses pengambilan gambar melalui *webcam* secara *real time* dan dideteksi oleh ketiga sensor yaitu *Flame Sensor 5 Kanal,* sensor suhu, dan sensor asap kemudian diolah menggunakan Raspberry pi. Setelah diproses menggunakan Raspberry pi, kemudian akan mengaktifkan *relay* untuk menyalakan *valve selenoid 12V*. Dengan begitu notifikasi akan masuk melalui aplikasi *android* berupa pesan yang apabila diklik akan masuk ke aplikasi android dan aplikasi tersebut menampilkan video *realtime* pada titik terjadinya kebakaran.

Rancangan monitoring sistem deteksi dan peringatan nyala api ini menggunakan kamera *webcam*. Data yang di tangkap oleh *Webcam* dan sensor dikirim melalui aplikasi *IoT*. User harus terhubung dengan internet dengan begitu user dapat menerima data tersebut melalui aplikasi *Android*. Dalam desain sistem, pengolahan data dan pertukaran informasi menggunakan salah satu platform *IoT (Internet of Things).*

***Kata kunci** : Webcam, Flame Sensor 5 kanal, Raspberry pi, IoT, dan Sistem Android*

ABSTRACT

DESIGN OF FIRE IDENTIFICATION APPLICATION USING PWA APPLICATION BASED ON RASPBERRY PI IN FLIGHT POLYTECHNIC SURABAYA

By:

Mochammad Rizqi Nur Wahyudi
NIT. 30218015

Fire is a disaster that is worrying for those who experience it because it can result in various losses. Buildings, furniture, and even documents were burned down. In this final project, a fire detector prototype will be designed with the output in the form of notification messages that goes through the android application and special messages that are directly connected to the fire department.

These results are obtained from the process of taking pictures through webcam in realtime and detected by the three sensors, namely the 5 Channel Flame Sensor, temperature sensor, and smoke sensor then processed using Raspberry pi. After processing using Raspberry pi, then it will activate the relay to turn on the 12V solenoid valve. In that way, the notification will go through the android application in the form of a message which when we clicked it, we will go into the android application and the application displays a realtime video at the point of the fire.

The monitoring design of the flame detection and warning system uses a webcam camera, the results of the image are detected in image processing and processed in the Raspberry pi as the basic concept of the detection system. The data captured by webcams and sensors are sent through the IoT application. The user must be connected to the internet so that the user can receive the data through the Android application. In system design, data processing and information exchange use one of the IoT (Internet of Things) platforms.

Keywords: *Webcam, Flame Sensor 5 channel, Raspberry pi, IoT, and Android System*

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR PUSTAKA

Afrianti, Aninda Nurita (2020) *Rancangan Monitoring Sistem Deteksi Dan Peringatan Nyala Api Terpadu Berbasis Iot Pada Equipment Room Di Bandara Internasional Juanda Surabaya.* Surabaya: Politeknik Penerbangan Surabaya.

Bahrul Ulum, Khabib (2013) *Prototipe Sistem Peringatan Dan Pemadam Kebakaran Ruangan Berbasis Mikrokontroller Atmega16,* Yogyakarta : Universitas Islam Negeri Sunan Kalijaga Yogyakarta

Dwi, Harisna Ade (2017) Image Processing. https://ndoware.com/image-processing.html.

Faisal, A. (2010). *Pendeteksi Kebakaran Dengan Menggunakan Sensor Suhu LM35D dan Sensor Asap*. Yogyakarta: Universitas Gadjah Mada.

Hasan, mahmud. (2014) "*Rancang Bangun Rangkain Pengendali untuk Valve yang digunakan Sebagai Saluran Masuk Gas N2 dan O2 pada Alat Kalibrasi Sensor Oksigen"*. Jurusan Teknik Mesin, Fakultas teknologi Industri, Surabaya : Institut Teknologi Sepuluh Nopember (ITS).

Hidayat, Dody (2018) *Perancangan Proteksi Kebakaran Otomatis Pada Kapal Berbasis Arduino.* Medan : Universitas Harapan Medan.

Jaka, Zaillani M. (2017) *Deteksi Dini Kebakaran Menggunakan Arduino.* Batam : Politeknik Negeri Batam

Jafar, Abdul (2013) *Alat Pemadam Kebakaran Otomatis Arduino Atmega 328P*, Bandung : Universitas Pendidikan Indonesia

Kusnandar, Ni Ketut Hariyawati Dharmi, dan Dwi Ajeng pratika (2019) *Rancang Bangun Prototipe Pendeteksi Kebakaran Menggunakan Konsep Internet of Things.* Cimahi : Universitas Jenderal Ahmad Yani, Indonesia.

Lupita, Sari Dita (2017) *Rancang bangun alat pendeteksi titik kerusakan pada kabel coaxcial dan UTP*. Surabaya : Akademi Teknik dan Keselamatan Penerbangan Surabaya.

Putra Bahari, Widyatmoko (2019) *Rancang Bangun Alat Pendeteksi Kebakaran Berbasis Internet Of Things (Iot),* Yogyakarta : Universitas Teknologi Yogyakarta

Sumarto, (2017) *Sistem Peringatan Dini Deteksi Dan Pemadam Kebakaran Berbasis Raspberry Pi.* Surabaya : Institut Teknologi Sepuluh Nopember.

Wijaya, Indra Dharma (2017) Implementasi Raspberry Pi Untuk Rancang Bangun Sistem Keamanan Pintu Ruang Server Dengan Pengenalan Wajah Menggunakan Metode Triangle Face, Malang : Politeknik Negeri Malang.

Lampiran 1

Koding Modul GSM

```
import serial
import RPi.GPIO as GPIO
import os, time, sys, argparse

parser = argparse.ArgumentParser(description="Send sms message using SIM900.")
parser.add_argument("-d", "--destination", type=str, required=True)
parser.add_argument("-m", "--message", type=str, required=True)

args = parser.parse_args()
print(args);
# sys.exit()

GPIO.setmode(GPIO.BOARD)

# Enable Serial Communication
port = serial.Serial("/dev/ttyS0", baudrate=9600, timeout=1)

# Transmitting AT Commands to the Modem
# '\r\n' indicates the Enter key

port.write(b"AT\r\n")
port.write(b"\x0D\x0A")
rcv = port.read(10)
print(rcv)
time.sleep(1)

port.write(b"AT+CMGF=1\r\n")  # Select Message format as Text mode
```

```
rcv = port.read(10)
print(rcv)
time.sleep(1)

port.write(b"AT+CNMI=2,1,0,0,0\r\n")   # New SMS Message Indications
rcv = port.read(10)
print(rcv)
time.sleep(1)

# Sending a message to a particular Number

dest = str.encode("AT+CMGS=" + "\"" + args.destination  + "\"\r\n")
port.write(dest)
# port.write(b"AT+CMGS=\"082132378313\"\r\n")
rcv = port.read(10)
print(rcv)
time.sleep(1)

msg_bytes = str.encode(args.message + "\r\n")
port.write(msg_bytes)
# port.write(b"Hello User\r\n")  # Message
rcv = port.read(10)
print(rcv)

port.write(b"\x1A") # Enable to send SMS
for i in range(10):
    rcv = port.read(10)
    print(rcv)
```

Koding Sensor DS18B

```
import glob
import time

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c
        # return temp_c, temp_f

print(read_temp())

#while True:
```

```
#    print(read_temp())
#    time.sleep(1)
```

RELAY

```
import RPi.GPIO as GPIO
import argparse
import time

parser = argparse.ArgumentParser(description="Toggle relay.")
parser.add_argument("-s", "--state", type=str, required=True)

args = parser.parse_args()
print(args);

try:
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(13, GPIO.OUT)

    if args.state == "ACTIVE":
        print("Turning relay ACTIVE");
        GPIO.output(13, GPIO.LOW)
        # time.sleep(1)
    elif args.state == "INACTIVE":
        print("Turning relay INACTIVE");
        GPIO.output(13, GPIO.HIGH)
        # time.sleep(1)
finally:
    pass
#    GPIO.cleanup()
```

Koding Sensor Suhu MQ7

```
import RPi.GPIO as GPIO
import time

try:
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(15, GPIO.IN, pull_up_down=GPIO.PUD_UP)

    if not GPIO.input(15):
        print("DETECTED")
    else:
        print("NOT_DETECTED")
finally:
    pass
#    GPIO.cleanup()
```

Koding Flame Sensor

```
import RPi.GPIO as GPIO
import time

try:
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)

    pins = [29, 31, 33, 35, 37]
    for pin in pins:
        GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

    results = []
    for pin in pins:
        if not GPIO.input(pin):
            results.append("0")
        else:
            results.append("1")

    print(",".join(results))
finally:
    pass
#    GPIO.cleanup()
```

Koding System

```
require("dotenv").config({
  path: "../../.env",
});
const scheduler = require("node-schedule");
const shell = require("shelljs");
const dayjs = require("dayjs");
const OneSignal = require("onesignal-node");

const dotenv = require("dotenv");
const fs = require("fs");
const envConfig = dotenv.parse(fs.readFileSync("./.env"));
// console.log({ envConfig });
for (const k in envConfig) {
  process.env[k] = envConfig[k];
}

const FIRE_FORCE_PHONE = "081332188308";
const USER_PHONE = "081332188308";
const MESSAGE_FORMAT =
  "TERDETEKSI API / ASAP PADA RUANGAN..!  LIHAT KAMERA:
https://android-fire-control.tapoltekbangsby.com/webcam.  LOKASI:
bit.ly/3jA1npD";
console.log("> MESSAGE_FORMAT long:", MESSAGE_FORMAT.length);
if (MESSAGE_FORMAT.length > 140) {
  console.warn("! Error: MESSAGE_FORMAT is too long!");
  process.exit();
}

const TEMP_THRESHOLD = 40;
const RISK_THRESHOLD = 4;
```

```
const RISK_DETECTION_CONFIDENT_COUNT = 2;

let latestSMSSent = null;
let riskDetectedCount = 0;
let sensorState = {
  mode: 1, // 1 = AUTO, 0 = MANUAL
  relay: 0,
  ds18b20: 29,
  mq9: 0,
  flame1: 0,
  flame2: 0,
  flame3: 0,
  flame4: 0,
  flame5: 0,
};

const start = () => {
  const oneSignalClient =
    process.env.NODE_ENV === "production"
      ? new OneSignal.Client(
          "c26b25b3-bfa1-463f-a0cc-0a98197c0104",

"YjQzNjBjMmYtMWYxOC00ODM4LWE2NDQtMDQ1NTY0OGY1MzVi"
        )
      : new OneSignal.Client(
          "73ce3d95-db07-4554-94f3-8056b8d64253",
          "ZGZlOGI1ZjItZDI3ZC00ZDg0LTkwNzUtY2E2OTI5N2UyYWRk"
        );

  const WebSocket = require("ws");
  const wsEndpointUrl =
```

```
  process.env.WS_PRODUCTION_ENDPOINT ||
  `ws://localhost:${process.env.WS_PORT}`;
console.log("Connect to", wsEndpointUrl);
const ws = new WebSocket(wsEndpointUrl);

let running = false;
const readSensor = async () => {
  if (running) return;
  running = true;

  try {
    if (process.env.NODE_ENV === "production") {
      console.log("Reading ds18b20...");
      var child = shell.exec(`python3 server/02-test-ds18b20.py`, {
        silent: true,
        async: true,
      });
      var result = await new Promise((resolve, reject) => {
        let latestMessage = "";
        child.stdout.on("data", function (data) {
          data = data.trim();
          // console.log({ data })
          latestMessage = data;
          resolve({ stdout: latestMessage });
        });
        child.on("exit", () => {
          resolve({ stdout: latestMessage });
        });
      });
      console.log("> Result", result.stdout);
      const ds18b20 = parseFloat(result.stdout.trim());
```

```
console.log("Reading mq...");
var child = shell.exec(`python3 server/04-test-mq.py`, {
  silent: true,
  async: true,
});
var result = await new Promise((resolve, reject) => {
  let latestMessage = "";
  child.stdout.on("data", function (data) {
    data = data.trim();
    // console.log({ data })
    latestMessage = data;
    resolve({ stdout: latestMessage });
  });
  child.on("exit", () => {
    resolve({ stdout: latestMessage });
  });
});
console.log("> Result", result.stdout);
const mq9 = result.stdout.trim() === "DETECTED" ? 1 : 0;

console.log("Reading flame-sensor...");
var child = shell.exec(`python3 server/05-test-flame-sensor.py`, {
  silent: true,
  async: true,
});
var result = await new Promise((resolve, reject) => {
  let latestMessage = "";
  child.stdout.on("data", function (data) {
    data = data.trim();
    // console.log({ data })
```

```
    latestMessage = data;
    resolve({ stdout: latestMessage });
  });
  child.on("exit", () => {
    resolve({ stdout: latestMessage });
  });
});
console.log("> Result", result.stdout);
const flameSensor = result.stdout
  .trim()
  .split(",")
  .map((result) => parseInt(result));

let countRisks = 0;
if (ds18b20 >= TEMP_THRESHOLD) {
  countRisks += 1;
}
if (mq9 === 1) {
  countRisks += 1;
}
for (const flameSensorValue of flameSensor) {
  if (flameSensorValue === 1) {
    countRisks += 1;
  }
}

if (countRisks >= RISK_THRESHOLD) {
  riskDetectedCount += 1;
} else {
  riskDetectedCount -= 1;
}
```

```
        if (riskDetectedCount >= RISK_THRESHOLD) {
          riskDetectedCount = RISK_THRESHOLD;
        } else if (riskDetectedCount < 0) {
          riskDetectedCount = 0;
        }

        let relay = sensorState.relay;
        if (sensorState.mode === 1) {
          if (riskDetectedCount >= RISK_DETECTION_CONFIDENT_COUNT) {
            relay = 1;
            if (process.env.NODE_ENV === "production") {
              let result = shell.exec(
                `python3 server/03-test-relay.py -s ACTIVE`,
                {
                  silent: true,
                }
              );

              if (
                !latestSMSSent ||
                dayjs().diff(dayjs(latestSMSSent), "second") > 30
              ) {
                latestSMSSent = dayjs().toISOString();
                const notification = {
                  contents: {
                    en: MESSAGE_FORMAT,
                  },
                  included_segments: ["Subscribed Users"],
                  web_url:
                    "http
s://android-fire-control.tapoltekbangsby.com/connect",
```

```
            };
            console.log("Sending notification:", notification);

            oneSignalClient
              .createNotification(notification)
              .then((response) => {
                console.log("createNotification:", response.body);
              })
              .catch((err) => {
                console.warn("Error createNotification:", err);
              });

            let result = shell.exec(
              `python3 server/01-test-sim900.py -d ${FIRE_FORCE_PHONE} -m
"${MESSAGE_FORMAT}"`,
              {
                // silent: true,
              }
            );
            if (FIRE_FORCE_PHONE !== USER_PHONE) {
              let result = shell.exec(
                `python3 server/01-test-sim900.py -d ${USER_PHONE} -m
"${MESSAGE_FORMAT}"`,
                {
                  // silent: true,
                }
              );
            }
          }
        } else if (riskDetectedCount === 0) {
```

```
    relay = 0;
    if (process.env.NODE_ENV === "production") {
      let result = shell.exec(
        `python3 server/03-test-relay.py -s INACTIVE`,
        {
          silent: true,
        }
      );
    }
  }
}

console.log("> State", {
  ds18b20,
  mq9,
  flameSensor,
  countRisks,
  riskDetectedCount,
  relay,
});

sensorState = {
  ...sensorState,
  relay,
  ds18b20,
  mq9,
  flame1: flameSensor[0],
  flame2: flameSensor[1],
  flame3: flameSensor[2],
  flame4: flameSensor[3],
  flame5: flameSensor[4],
```

```
      };
    }

    let message = `${sensorState.mode},${sensorState.relay},${sensorState.ds18b20},${sensorState.mq9},${sensorState.flame1},${sensorState.flame2},${sensorState.flame3},${sensorState.flame4},${sensorState.flame5}`;
     console.log("> Sending", message);
     ws.send(message);
     console.log(" ");
   } catch (err) {}

  running = false;
 };

 ws.on("open", function open() {
  console.log("> On open!");
  const job = scheduler.scheduleJob("* * * * * *", readSensor);
 });

 ws.on("message", function incoming(data) {
  console.log("> On message", data);

  const values = data.trim().split(",");
  console.log({ values });
  if (values.length !== 2) return;

  const newMode = parseInt(values[0]);
  const newRelay = parseInt(values[1]);
  console.log({
   newMode,
```

```
      newRelay,
    });

    sensorState.mode = newMode;
    sensorState.relay = newRelay;

    let message = `${sensorState.mode},${sensorState.relay},${sensorState.ds18b20},${sensorState.mq9},${sensorState.flame1},${sensorState.flame2},${sensorState.flame3},${sensorState.flame4},${sensorState.flame5}`;
    console.log("> Sending", message);
    ws.send(message);

    if (process.env.NODE_ENV === "production") {
      if (sensorState.relay === 1) {
        let result = shell.exec(`python3 server/03-test-relay.py -s ACTIVE`, {
          // silent: true,
        });
      } else {
        let result = shell.exec(`python3 server/03-test-relay.py -s INACTIVE`, {
          // silent: true,
        });
      }
    }
  });
  ws.on("close", function close() {
    console.log("> On close!");
    process.exit();
  });
};
start();
```

Lampiran 2

PRELIMINARY

DALLAS SEMICONDUCTOR

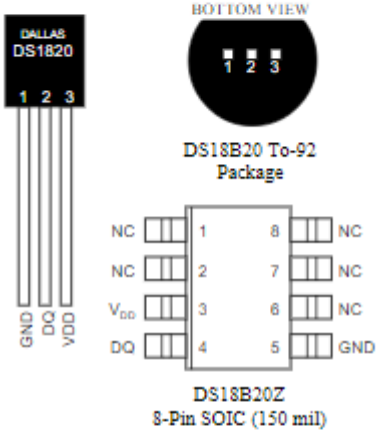
DS18B20
Programmable Resolution 1-Wire® Digital Thermometer

www.dalsemi.com

FEATURES

- Unique 1-Wire interface requires only one port pin for communication
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line. Power supply range is 3.0V to 5.5V
- Zero standby power required
- Measures temperatures from -55°C to +125°C. Fahrenheit equivalent is -67°F to +257°F
- ±0.5°C accuracy from -10°C to +85°C
- Thermometer resolution is programmable from 9 to 12 bits
- Converts 12-bit temperature to digital word in 750 ms (max.)
- User-definable, nonvolatile temperature alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

PIN ASSIGNMENT



DS18B20 To-92 Package

DS18B20Z 8-Pin SOIC (150 mil)

PIN DESCRIPTION

GND - Ground
DQ - Data In/Out
V_{DD} - Power Supply Voltage
NC - No Connect

DESCRIPTION

The DS18B20 Digital Thermometer provides 9 to 12-bit (configurable) temperature readings which indicate the temperature of the device.

Information is sent to/from the DS18B20 over a 1-Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS18B20. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

Because each DS18B20 contains a unique silicon serial number, multiple DS18B20s can exist on the same 1-Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and process monitoring and control.

S

TECHNICAL DATA MQ-7 GAS SENSOR

FEATURES

* High sensitivity to carbon monoxide
* Stable and long life

APPLICATION

They are used in gas detecting equipment for carbon monoxide(CO) in family and industry or car.

SPECIFICATIONS

A. Standard work condition

Symbol	Parameter name	Technical condition	Remark
Vc	circuit voltage	5V±0.1	Ac or Dc
V_H (H)	Heating voltage (high)	5V±0.1	Ac or Dc
V_H (L)	Heating voltage (low)	1.4V±0.1	Ac or Dc
R_L	Load resistance	Can adjust	
R_H	Heating resistance	33Ω±5%	Room temperature
$T_{H(H)}$	Heating time (high)	60±1 seconds	
$T_{H(L)}$	Heating time (low)	90±1 seconds	
PH	Heating consumption	About 350mW	

b. Environment conditions

Symbol	Parameters	Technical conditions	Remark
Tao	Using temperature	-20℃-50℃	
Tas	Storage temperature	-20℃-50℃	Advice using scope
RH	Relative humidity	Less than 95%RH	
O_2	Oxygen concentration	21%(stand condition) the oxygen concentration can affect the sensitivity characteristic	Minimum value is over 2%

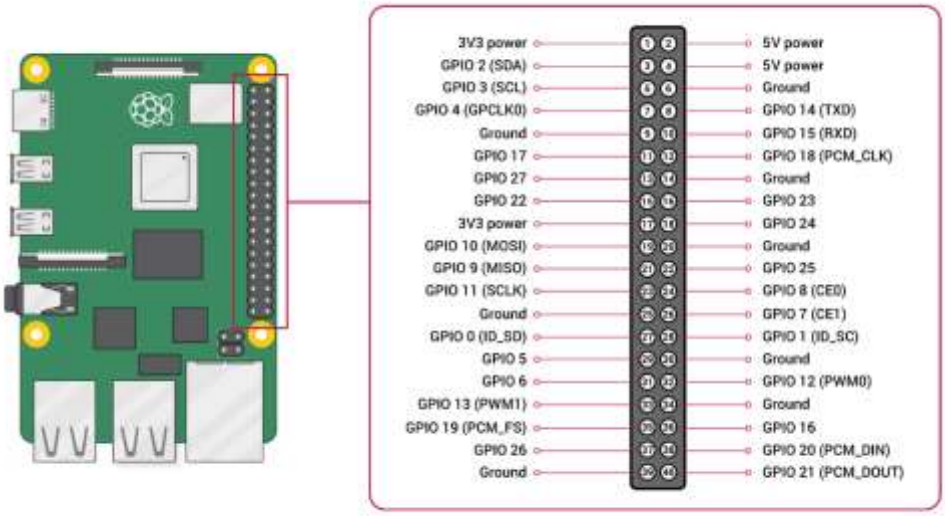
c. Sensitivity characteristic

symbol	Parameters	Technical parameters	Remark
Rs	Surface resistance Of sensitive body	2-20k	In 100ppm Carbon Monoxide
a (300/100ppm)	Concentration slope rate	Less than 0.5	Rs (300ppm)/Rs(100ppm)
Standard working condition	Temperature -20℃±2℃ relative humidity 65%±5% RL:10KΩ±5%		
	Vc:5V±0.1V VH:5V±0.1V VH:1.4V±0.1V		
Preheat time	No less than 48 hours	Detecting range: 20ppm-2000ppm carbon monoxide	

D. Structure and configuration, basic measuring circuit

Structure and configuration of MQ-7 gas sensor is shown as Fig. 1 (Configuration A or B), sensor composed by micro AL_2O_3 ceramic tube, Tin Dioxide (SnO_2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-7 have

T


3V3 power
GPIO 2 (SDA)
GPIO 3 (SCL)
GPIO 4 (GPCLK0)
Ground
GPIO 17
GPIO 27
GPIO 22
3V3 power
GPIO 10 (MOSI)
GPIO 9 (MISO)
GPIO 11 (SCLK)
Ground
GPIO 0 (ID_SD)
GPIO 5
GPIO 6
GPIO 13 (PWM1)
GPIO 19 (PCM_FS)
GPIO 26
Ground
5V power
5V power
Ground
GPIO 14 (TXD)
GPIO 15 (RXD)
GPIO 18 (PCM_CLK)
Ground
GPIO 23
GPIO 24
Ground
GPIO 25
GPIO 8 (CE0)
GPIO 7 (CE1)
GPIO 1 (ID_SC)
Ground
GPIO 12 (PWM0)
Ground
GPIO 16
GPIO 20 (PCM_DIN)
GPIO 21 (PCM_DOUT)

U

1. Introduction

This document describes SIM800L hardware interface in great detail.
This document can help user to quickly understand SIM800L interface specifications, electrical and mechanical details. With the help of this document and other SIM800L application notes, user guide, users can use SIM800L to design various applications quickly.

2. SIM800L Overview

SIM800L is a quad-band GSM/GPRS module, that works on frequencies GSM850MHz, EGSM900MHz, DCS1800MHz and PCS1900MHz. SIM800L features GPRS multi-slot class 12/ class 10 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4.

With a tiny configuration of 15.8*17.8*2.4mm, SIM800L can meet almost all the space requirements in user applications, such as smart phone, PDA and other mobile devices.

SIM800L has 88pin pads of LGA packaging, and provides all hardware interfaces between the module and customers' boards.

- Support 5*5*2 keypads
- One full modem serial port, user can configure two serial ports
- One USB, the USB interfaces can debug, download software
- Audio channel which includes two microphone input, a receiver output and a speaker output
- Programmable general purpose input and output
- A SIM card interface
- Support FM
- Support one PWM

SIM800L is designed with power saving technique so that the current consumption is as low as 0.7mA in sleep mode.

2.1. SIM800L Key Features

Table 1: SIM800L key features

Feature	Implementation
Power supply	3.4V ~4.4V
Power saving	typical power consumption in sleep mode is 0.7mA (AT+CFUN=0)
Frequency bands	• Quad-band: GSM 850, EGSM 900, DCS 1800, PCS 1900. SIM800L can search the 4 frequency bands automatically. The frequency bands can also be set by AT command "AT+CBAND". For details, please refer to document [1]. • Compliant to GSM Phase 2/2+
Transmitting power	• Class 4 (2W) at GSM 850 and EGSM 900 • Class 1 (1W) at DCS 1800 and PCS 1900
GPRS connectivity	• GPRS multi-slot class 12 (default) • GPRS multi-slot class 1~12 (option)
Temperature range	• Normal operation: -40°C ~ +85°C

	• Storage temperature -45°C ~ +90°C
Data GPRS	• GPRS data downlink transfer: max. 85.6 kbps • GPRS data uplink transfer: max. 85.6 kbps • Coding scheme: CS-1, CS-2, CS-3 and CS-4 • PAP protocol for PPP connect • Integrate the TCP/IP protocol. • Support Packet Broadcast Control Channel (PBCCH) • CSD transmission rates: 2.4, 4.8, 9.6, 14.4 kbps
CSD	• Support CSD transmission
USSD	• Unstructured Supplementary Services Data (USSD) support
SMS	• MT, MO, CB, Text and PDU mode • SMS storage: SIM card
SIM interface	Support SIM card: 1.8V, 3V
External antenna	Antenna pad
Audio features	Speech codec modes: • Half Rate (ETS 06.20) • Full Rate (ETS 06.10) • Enhanced Full Rate (ETS 06.50 / 06.60 / 06.80) • Adaptive multi rate (AMR) • Echo Cancellation • Noise Suppression
Serial port and debug port	**Serial port:** • Full modem interface with status and control lines, unbalanced, asynchronous. • 1200bps to 115200bps. • Can be used for AT commands or data stream. • Support RTS/CTS hardware handshake and software ON/OFF flow control. • Multiplex ability according to GSM 07.10 Multiplexer Protocol. • Autobauding supports baud rate from 1200 bps to 57600bps. • upgrading firmware **Debug port:** • USB_DM and USB_DP • Can be used for debugging and upgrading firmware.
Phonebook management	Support phonebook types: SM, FD, LD, RC, ON, MC
SIM application toolkit	GSM 11.14 Release 99
Real time clock	Support RTC
Timing functions	Use AT command set
Physical characteristics	Size:15.8*17.8*2.4mm Weight:1.35g
Firmware upgrade	Main serial port or USB port.

Table 2: Coding schemes and maximum net data rates over air interface

Coding scheme	1 timeslot	2 timeslot	4 timeslot
CS-1	9.05kbps	18.1kbps	36.2kbps
CS-2	13.4kbps	26.8kbps	53.6kbps
CS-3	15.6kbps	31.2kbps	62.4kbps
CS-4	21.4kbps	42.8kbps	85.6kbps

2.2. Operating Mode

The table below summarizes the various operating modes of SIM800L.

Table 3: Overview of operating modes

Mode	Function	
Normal operation	GSM/GPRS SLEEP	Module will automatically go into sleep mode if the conditions of sleep mode are enabling and there is no on air and no hardware interrupt (such as GPIO interrupt or data on serial port). In this case, the current consumption of module will reduce to the minimal level. In sleep mode, the module can still receive paging message and SMS.
	GSM IDLE	Software is active. Module is registered to the GSM network, and the module is ready to communicate.
	GSM TALK	Connection between two subscribers is in progress. In this case, the power consumption depends on network settings such as DTX off/on, FR/EFR/HR, hopping sequences, antenna.
	GPRS STANDBY	Module is ready for GPRS data transfer, but no data is currently sent or received. In this case, power consumption depends on network settings and GPRS configuration.
	GPRS DATA	There is GPRS data transfer (PPP or TCP or UDP) in progress. In this case, power consumption is related with network settings (e.g. power control level); uplink/downlink data rates and GPRS configuration (e.g. used multi-slot settings).
Power down	Normal power down by sending AT command "AT+CPOWD=1" or using the PWRKEY. The power management unit shuts down the power supply for the baseband part of the module, and only the power supply for the RTC is remained. Software is not active. The serial port is not accessible. Power supply (connected to VBAT) remains applied.	
Minimum functionality mode	AT command "AT+CFUN" can be used to set the module to a minimum functionality mode without removing the power supply. In this mode, the RF part of the module will not work or the SIM card will not be accessible, or both RF part and SIM card will be closed, and the serial port is still accessible. The power consumption in this mode is lower than normal mode.	

DAFTAR RIWAYAT HIDUP

Mochammad Rizqi Nur Wahyudi, lahir di Sidoarjo pada tanggal 28 Juni 2000 anak pertama dari pasangan Bapak Suparna dan Ibu Mujiat Miningsih. Sebagai kakak dari Moch. Faza Dwi Alvinanta. Bertempat tinggal di Dusun Jatidukuh, RT 04 RW 01, Desa Jatikalang, Kecamatan Prambon, Kabupaten Sidoarjo Jawa Timur. Menamatkan sekolah Taman Kanak Kanak pada tahun 2006 di Taman Kanak-kanak Dharma Wanita Jatikalang Kabupaten Sidoarjo. Melanjutkan Sekolah Dasar pada tahun 2006-2012 di SDN Jatikalang Kabupaten Sidoarjo. Melanjutkan hingga tamat Sekolah Menengah Pertama pada tahun 2012-2015 di SMP Negeri 2 Krembung Kabupaten Sidoarjo. Melanjutkan sekolah Menengah Atas hingga tamat pada tahun 2015-2017 di SMA Negeri 1 Krembung Kabupaten Sidoarjo. Pada bulan September 2018 diterima menjadi Taruna Politeknik Penerbangan Surabaya pada Program Studi Diploma 3 Teknik Navigasi Udara Angkatan XI. Pengalaman *On the Job Training* (OJT) pada semester 4 selama 5 bulan yang berawal dari bulan Juni 2020 hingga bulan Februari 2021 di Perum LPPNPI Cabang Surabaya.