ALAT PENDETEKSI LAMA WAKTU DUDUK DAN POSISI KEBUNGKUKAN TUBUH BERBASIS IOT

PROYEK AKHIR

Laporan akhir ini dibuat dan diajukan untuk memenuhi salah satu syarat kelulusan
Diploma IV Politeknik Manufaktur Negeri Bangka Belitung

Disusun Oleh:

Rizky Daffa Pratama	NIM : 1052225
Tio Pratama	NIM : 1052230

POLITEKNIK MANUFAKTUR NEGERI
BANGKA BELITUNG
TAHUN 2025

LEMBAR PENGESAHAN

ALAT PENDETEKSI LAMA WAKTU DUDUK DAN POSISI KEBUNGKUKAN TUBUH BERBASIS IOT

Oleh :

Rizky Daffa Pratama/1052225

Tio Pratama/1052230

Laporan akhir ini telah disetujui dan disahkan sebagai salah satu syarat kelulusan Program Diploma IV Politeknik Manufaktur Negeri Bangka Belitung

Menyetujui,

Pembimbing 1	Pembimbing 2
Aan Febriansyah , S.ST., M.T.	Misri Yanda , S.Pi., M.Si.
Penguji 1	Penguji 2
Surojo , S.T., M.T.	Enggar Hero Istoto ,S.Si., M.En.

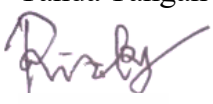
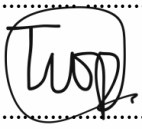
PERNYATAAN BUKAN PLAGIAT

Yang bertanda tangan dibawah ini :

Nama Mahasiswa 1	: Rizky Daffa Pratama	NIM. 1052225
Nama Mahasiswa 2	: Tio Pratama	NIM. 1052230
Dengan Judul	:Alat Pendeteksi Lama Waktu Duduk Dan Posisi Kebungkukan Tubuh Berbasis IoT	

Menyatakan bahwa laporan akhir ini adalah hasil kerja kami sendiri dan bukan merupakan plagiat. Pernyataan ini kami buat dengan sebenarnya dan bila ternyatadi kemudian hari ternyata melanggar pernyataan ini, kami bersedia menerima sanksi yang berlaku.

Sungailiat, 17 Juli 2025

Nama Mahasiswa	Tanda Tangan
1. Rizky Daffa Pratama	 .....
2. Tio Pratama	 .....

ABSTRAK

Penulisan makalah ini bertujuan untuk memaparkan alat pendeteksi lama waktu duduk dan posisi kebungkukan tubuh berbasis IoT guna meningkatkan kesadaran akan pentingnya menjaga postur ergonomis selama bekerja. Postur tubuh yang buruk, seperti duduk terlalu lama atau membungkuk, dapat memicu masalah kesehatan serius seperti nyeri punggung, leher, dan gangguan muskuloskeletal. Penelitian ini merancang sistem yang menggabungkan sensor flex untuk mengukur kemiringan tulang belakang dan sensor ultrasonik untuk memantau jarak antara perut pengguna dengan box yang dilengkapi sensor ultrasonik. Data dari kedua sensor diproses oleh mikrokontroler ESP32 dan dikirim secara real-time ke platform IoT (Firebase). Sistem ini tidak hanya memberikan notifikasi langsung melalui buzzer dan tampilan LCD saat terdeteksi durasi duduk melebihi batas atau posisi tubuh tidak ideal, tetapi juga memungkinkan pemantauan jarak jauh melalui web. Integrasi IoT memungkinkan pengguna menganalisis data historis untuk mengevaluasi kebiasaan duduk mereka secara berkala. Alat ini dirancang dengan antarmuka yang user-friendly, sehingga cocok digunakan di lingkungan kerja atau studi yang menuntut aktivitas duduk intensif. Diharapkan, inovasi ini dapat menjadi solusi preventif untuk mengurangi risiko gangguan kesehatan akibat postur duduk tidak ergonomis, sekaligus meningkatkan kesadaran pengguna melalui pendekatan teknologi yang interaktif dan terukur.

Kata kunci : *Duduk ergonomis; lama waktu duduk; posisi kebungkukan; Sensor flex; IoT*

ABSTRACT

This paper aims to present an IoT-based device for detecting sitting duration and body slouching posture to raise awareness of the importance of maintaining ergonomic posture during work. Poor posture, such as sitting for prolonged periods or slouching, can lead to serious health issues like back pain, neck strain, and musculoskeletal disorders. This research designs a system that integrates a flex sensor to measure spinal inclination and an ultrasonic sensor to monitor the distance between the user's abdomen and a box equipped with the ultrasonic sensor. Data from both sensors are processed by the ESP32 microcontroller and transmitted in real-time to an IoT platform (Firebase). The system not only provides immediate notifications via a buzzer and an LCD display when prolonged sitting or improper posture is detected but also enables remote monitoring through a web. IoT integration allows users to analyze historical data to periodically evaluate their sitting habits. The device is designed with a user-friendly interface, making it suitable for workplaces or study environments that require intensive sitting activities. It is hoped that this innovation can serve as a preventive solution to reduce health risks caused by non-ergonomic sitting postures while enhancing user awareness through interactive and measurable technological approaches.

Keywords: Ergonomic sitting; length of time sitting; bent position; flex sensor ; IoT

KATA PENGANTAR

Puji syukur kami panjatkan kepada Allah swt atas segala rahmat dan hidayah-Nya, sehingga kami dapat menyelesaikan karya tulis Proyek Akhir ini dengan baik. Proyek Akhir ini merupakan salah satu syarat yang harus dipenuhi untuk menyelesaikan program pendidikan Diploma IV di Politeknik Manufaktur Negeri Bangka Belitung.

Selama empat tahun menempuh pendidikan, kami telah belajar banyak hal, dan melalui Proyek Akhir ini, kami mencoba menerapkan ilmu yang telah kami dapatkan, baik dari perkuliahan maupun pengalaman selama melaksanakan Program Kerja Lapangan. Proyek Akhir ini tidak hanya berupa pembuatan alat, tetapi juga penulisan makalah yang menjadi bukti perjalanan akademis kami.

Pada kesempatan ini, kami ingin mengucapkan terima kasih sebesar-besarnya kepada semua pihak yang telah membantu dan mendukung kami dalam menyelesaikan Proyek Akhir ini. Secara khusus, kami berterima kasih kepada:

1. Terima kasih Kepada Allah SWT,telah memberikan kelancaran serta kemudahan dalam melaksanakan Tugas Akhir ini.
2. Bapak I Made Andik Setiawan , M.Eng., Ph.D. selaku Direktur Politeknik Manufaktur Negeri Bangka Belitung.
3. Bapak Aan Febriansyah , S.ST., M.T. selaku Pembimbing 1 dan Bapak Misri Yandi , S.Pi., M.Si. selaku Pembimbing 2, yang telah meluangkan waktu, tenaga, dan pikiran untuk membimbing kami. Beliau memberikan banyak arahan, saran, dan solusi yang sangat membantu dalam penyelesaian Proyek Akhir ini.
4. Seluruh staf pengajar di Politeknik Manufaktur Negeri Bangka Belitung yang telah membagikan ilmu dan pengalaman selama kami menempuh pendidikan.
5. Keluarga kami yang selalu memberikan dukungan dan semangat hingga Proyek Akhir ini selesai.
6. Rekan-rekan mahasiswa yang telah membantu dan mendukung selama proses pengerjaan Proyek Akhir ini.
7. Semua pihak yang telah memberikan bantuan, baik secara langsung maupun

tidak langsung, yang tidak dapat kami sebutkan satu per satu.

Kami menyadari bahwa Proyek Akhir ini masih jauh dari sempurna karena kami adalah manusia biasa yang tidak luput dari kesalahan. Oleh karena itu, kami sangat mengharapkan saran, kritik, dan masukan yang membangun dari pembaca untuk perbaikan di masa mendatang. Semoga Proyek Akhir ini dapat bermanfaat dan menambah wawasan bagi semua pihak, khususnya rekan-rekan mahasiswa. Terima kasih dan mohon maaf jika ada kekurangan dalam penulisan karya tulis ini.

Salam hormat,

Tim Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR ISTILAH

Istilah	Pengertian Singkat
ADC (Analog to Digital Converter)	Pengubah sinyal analog menjadi digital agar dapat dibaca oleh mikrokontroler.
Buzzer	Komponen *output* suara yang memberikan notifikasi saat kondisi tertentu terdeteksi.
Derajat Flex	Nilai sudut kelengkungan yang dihitung dari perubahan resistansi pada sensor *flex*.
Diagram Blok	Representasi visual yang menunjukkan hubungan antar komponen dalam sistem.
ESP32	Mikrokontroler dengan modul *Wi-Fi* dan *Bluetooth* yang digunakan sebagai otak sistem.
Firebase	Layanan *cloud* berbasis *real-time database* yang digunakan untuk menyimpan dan memantau data secara online.
IoT (Internet of Things)	Teknologi yang memungkinkan perangkat terhubung dan bertukar data melalui jaringan internet.
JSON (JavaScript Object Notation)	Format data ringan yang digunakan untuk menyimpan dan mentransfer data antar perangkat.
LCD 16x2 I2C	Modul tampilan yang menampilkan status postur dan waktu secara langsung di alat.
Realtime Database	Sistem penyimpanan data yang memungkinkan sinkronisasi data langsung antara alat dan aplikasi.
RTC DS3231	Modul *Real-Time Clock* yang mencatat dan menjaga waktu secara akurat walau sistem mati.
Sensor Flex	Sensor yang mengubah resistansi berdasarkan tingkat kelengkungan untuk mendeteksi postur.
Sensor Ultrasonik	Sensor jarak yang menggunakan gelombang suara frekuensi tinggi untuk mengukur keberadaan objek.
Visual Studio Code	Editor kode sumber yang digunakan untuk membuat tampilan web dan pemrograman mikrokontroler.
Wi-Fi	Teknologi nirkabel untuk menghubungkan perangkat ke internet atau jaringan lokal.

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Organisasi Kesehatan Dunia (*WHO*) menggambarkan aktivitas fisik sebagai segala bentuk gerakan tubuh yang dihasilkan oleh otot rangka yang membutuhkan energi. Aktivitas ini mencakup berbagai gerakan, baik selama waktu senggang, saat bepergian ke atau dari suatu tempat, atau saat melakukan pekerjaan atau tugas rumah tangga. Aktivitas fisik sedang hingga berat telah terbukti bermanfaat bagi kesehatan. Beberapa cara umum untuk tetap aktif meliputi berjalan kaki, bersepeda, berolahraga, melakukan aktivitas rekreasi yang melibatkan gerakan aktif, dan bermain. Aktivitas ini dapat dilakukan oleh siapa saja, terlepas dari tingkat kemampuan, dan dapat dinikmati oleh semua kelompok.

Aktivitas fisik berperan penting dalam mendukung kesehatan dan kesejahteraan secara keseluruhan. Sebaliknya, kurangnya aktivitas fisik atau gaya hidup yang tidak banyak bergerak dapat meningkatkan risiko timbulnya penyakit tidak menular (PTM) dan dampak kesehatan negatif lainnya. Kurangnya aktivitas fisik dan kebiasaan tidak banyak bergerak dalam jangka waktu yang lama merupakan faktor utama yang berkontribusi terhadap meningkatnya prevalensi PTM dan memberikan tekanan tambahan pada sistem perawatan kesehatan[1]. Contohnya seperti terlalu lama duduk ketika bekerja.

Duduk terlalu lama sering kali mengakibatkan satu kondisi bermasalah yang dikenal sebagai nyeri punggung bawah (NPB). Menurut sebuah penelitian yang dilakukan oleh *American Chiropractic Association* (*ACA*), sekitar 80% orang di dunia akan mengalami nyeri punggung pada suatu waktu dalam hidup mereka; penyebab utamanya adalah postur tubuh yang buruk dan duduk terlalu lama[2].

Di Indonesia, prevalensi nyeri punggung bawah (NPB) belum diketahui secara pasti. Namun, hasil penelitian yang dilakukan oleh Perhimpunan Dokter Spesialis Saraf Indonesia (PERDOSSI) di 14 kota menunjukkan sekitar 18,1% responden mengalami NPB. Di Jakarta, angka prevalensinya tercatat sebesar 50,4%. Mayoritas penderita berada pada rentang usia 50 hingga 60 tahun, yakni sebesar 23,7%, dan sekitar 51,9% kasus NPB terjadi secara tiba-tiba. Oleh karena itu, diperlukan jawaban yang dapat membantu pengguna menjaga postur tubuh dan mengendalikan waktu duduk secara otomatis[3].

Menurut penelitian ergonomi kerja, durasi duduk yang disarankan tanpa jeda adalah maksimal 30 hingga 60 menit, setelah itu seseorang dianjurkan untuk berdiri atau melakukan peregangan selama beberapa menit agar sirkulasi darah tetap lancar dan otot tidak tegang[4]. Jika seseorang duduk lebih dari 1 jam tanpa istirahat, risiko mengalami ketegangan otot dan nyeri punggung meningkat secara signifikan. Oleh karena itu, sistem pendeteksi ini dirancang untuk memberikan peringatan setiap 60 menit duduk berturut-turut. Selain itu, berdasarkan hasil pengujian sensor *flex* yang digunakan dalam penelitian ini, postur duduk normal memiliki nilai sudut antara 9° hingga 15°, sedangkan posisi membungkuk ditandai dengan nilai sudut mulai dari 32° hingga 80°, sesuai dengan tabel pengujian pada Bab IV. Ambang batas kemiringan ini ditentukan berdasarkan perbandingan data manual dengan pengukuran sensor untuk membedakan antara postur ergonomis dan tidak ergonomis secara objektif.

Sejalan dengan perkembangan teknologi *Internet of Things*, postur tubuh dan durasi waktu duduk dapat dilacak dengan akurasi yang tinggi. *Internet of Things (IoT)* telah memberikan dampak yang signifikan di berbagai bidang, terutama di bidang kesehatan. Salah satu hal yang menonjol dari penerapan *IoT* di dunia medis adalah kemampuannya dalam mengumpulkan, memproses, dan menganalisis data secara efisien untuk mendukung peningkatan kualitas layanan dan perawatan pasien[5]. Oleh karena itu, sistem deteksi postur dan durasi waktu duduk berbasis *IoT* telah

dikembangkan untuk memberikan peringatan kepada pengguna jika mereka duduk dalam posisi yang salah atau terlalu lama.

Dalam pembuatan alat sebelumnya, hanya digunakan mikrokontroler Arduino . Sebelumnya, alat tersebut hanya memanfaatkan arduino tanpa memiliki fitur pemantauan data melalui aplikasi atau *website*, tidak menyediakan notifikasi melalui *smartphone*, serta tidak dilengkapi dengan data *logger*[6]. Oleh karena itu, kami mengembangkan instrumen yang memanfaatkan *Internet of Things* (*IoT*) untuk meningkatkan kemudahan penggunaan alat ini.

Pada penelitian ini, penulis menggunakan ESP32 sebagai mikrokontroler utama karena memiliki konektivitas *Wi-Fi* sehingga data dapat dikirim ke *Firebase* untuk pemantauan jarak jauh. Sensor *flex* digunakan untuk mendeteksi perubahan postur tubuh sedangkan sensor ultrasonik digunakan untuk mendeteksi keberadaan pengguna pada kursi. Apabila sensor mendeteksi postur yang salah atau durasi yang duduk melebihi batas yang disarankan, buzzer akan berbunyi serta data akan ditampilkan pada *LCD* dan dikirim ke *Firebase* untuk dianalisis lebih lanjut.

Dengan adanya sistem ini, diharapkan pengguna dapat lebih sadar pentingnya untuk menjaga postur tubuh yang baik dan mengatur durasi duduk yang lebih bijak maka dapat mengurangi risiko gangguan kesehatan akibat kebiasaan tidak sehat tersebut.

1.2. Perumusan Masalah

Adapun Perumusan masalah dari proyek akhir ini yaitu :

1. Bagaimana merancang alat yang dapat mendeteksi durasi duduk pengguna?
2. Bagaimana alat dapat memberikan peringatan jika posisi tubuh membungkuk?
3. Bagaimana data hasil deteksi dapat disimpan ,dimonitor secara *online* dan data riwayat bisa di simpan di *smarthphone*?

1.3. Tujuan Proyek Akhir

Sesuai dengan uraian latar belakang, maka tujuan yang ingin dicapai dari penelitian ini yaitu:

1. Mengembangkan alat yang mampu mendeteksi postur tubuh pengguna saat duduk untuk memastikan posisi yang benar dan sehat.
2. Memberikan peringatan saat pengguna duduk dalam posisi yang salah atau melebihi durasi yang disarankan.
3. Memanfaatkan teknologi *IoT* untuk menghubungkan alat dengan aplikasi atau perangkat lain, sehingga data dapat dianalisis dan ditampilkan secara interaktif.
4. Membantu pengguna meningkatkan kesadaran akan pentingnya menjaga postur tubuh yang baik dan mengatur waktu duduk guna mencegah risiko kesehatan jangka panjang.

1.4. Manfaat

1. Meningkatkan kesadaran pengguna terhadap postur dan kebiasaan duduk.
2. Mengurangi risiko gangguan postur dan penyakit akibat duduk berlebihan.
3. Memberikan data historis untuk evaluasi kebiasaan duduk pengguna.

BAB II
DASAR TEORI

2.1. Dampak Duduk Terlalu Lama terhadap Kesehatan

Duduk dalam durasi yang lama, terutama dengan posisi tubuh yang tidak tepat, dapat memicu berbagai gangguan kesehatan. Organisasi Kesehatan Dunia (*WHO*) mengidentifikasi kekurangan aktivitas fisik sebagai salah satu faktor risiko utama munculnya penyakit tidak menular, seperti obesitas, diabetes, dan gangguan sistem muskuloskeletal[1]. Salah satu kondisi yang sering muncul akibat duduk berkepanjangan adalah nyeri punggung bawah (*Low Back Pain/LBP*). Menurut *American Chiropractic Association (ACA)*, sekitar 80% populasi global akan mengalami nyeri punggung setidaknya sekali seumur hidup, di mana postur tubuh yang buruk dan kebiasaan duduk terlalu lama menjadi penyebab utama [2]. Keluhan ini paling banyak ditemukan pada kelompok usia dewasa. Sejumlah penelitian menunjukkan bahwa sekitar 23% orang dewasa di seluruh dunia mengalami nyeri punggung bawah kronis. Selain itu, tingkat kekambuhan dalam kurun waktu satu tahun berkisar antara 24% hingga 80%[3].

2.2. *Internet of Things* (*IoT*)

Perangkat *Internet of Things (IoT)* yang ditunjukkan pada Gambar 2.1. adalah teknologi yang memungkinkan berbagai perangkat fisik terhubung ke jaringan internet. Melalui konektivitas ini, perangkat dapat secara otomatis mengumpulkan, mengirimkan, dan menerima data tanpa interaksi manusia secara langsung. Dengan kemampuan ini, perangkat *IoT* menciptakan efisiensi dan kemudahan dalam kehidupan sehari-hari.*IoT* telah menjadi solusi cerdas yang membantu meningkatkan produktivitas dan kualitas hidup anda. Teknologi ini memberikan kemudahan, seperti kontrol jarak jauh, yang membuat aktivitas sehari-hari lebih praktis dan efisien[7]. Dalam alat ini, *IoT* memungkinkan sistem untuk mengirimkan data lama duduk dan postur tubuh

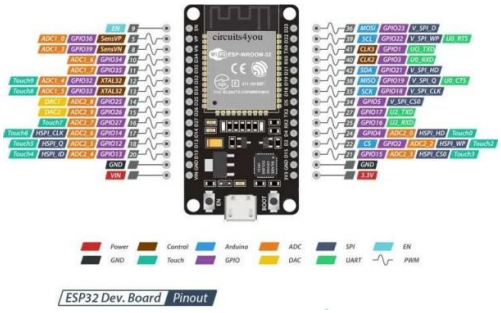
ke perangkat seperti *smartphone* secara *real-time* .

Gambar 2. 1 *Internet of Things*
(Sumber : https://www.microthings.id/peran-internet-of-things-dalam-smart-healthcare/)

Internet of Things (IoT) adalah sebuah konsep di mana berbagai perangkat fisik terintegrasi dengan jaringan internet dan mampu berkomunikasi serta berbagi data satu sama lain. *IoT* telah banyak diterapkan dalam bidang kesehatan untuk memantau kondisi fisik pengguna secara *real-time*. Dengan hadirnya teknologi berbasis internet, proses pemantauan kondisi kesehatan menjadi lebih mudah bagi dokter maupun pasien. Pemantauan secara *real-time* kini dapat dilakukan kapan saja dan dari mana saja, tanpa harus bertatap muka.[8]. *IoT* memungkinkan pengumpulan data secara otomatis, penyimpanan data di *cloud*, dan pemantauan jarak jauh melalui aplikasi atau *platform online*.

2.3. Mikrokontroller ESP 32

ESP32 adalah mikrokontroler yang dikembangkan oleh *Espressif Systems* sebagai generasi berikutnya dari ESP8266. Mikrokontroler ini dilengkapi dengan modul *Wi-Fi* terintegrasi dalam *chip*-nya, sehingga ideal untuk digunakan dalam pengembangan aplikasi berbasis *Internet of Things* (*IoT*)[9]. Berikut dibawah komponen ESP32 pada Gambar 2.2.



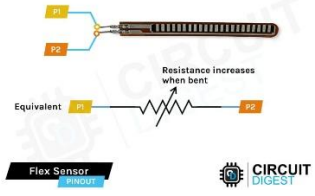
Gambar 2. 2 ESP32

(Sumber : https://circuits4you.com/2018/12/31/esp32-hardware-serial2-example/#google_vignette)

Mikrokontroler ini dipilih karena memiliki konektivitas *Wi-Fi* yang memungkinkan pengiriman data ke *platform cloud* seperti *Firebase*. ESP32 juga memiliki kemampuan pemrosesan yang cukup untuk mengintegrasikan sensor dan perangkat output seperti *buzzer* dan *LCD*.

2.4. Sensor *Flex*

Flex Sensor merupakan salah satu jenis sensor yang bekerja dengan mengubah nilai resistansinya ketika terjadi perubahan sudut atau lengkungan pada permukaannya. Sensor ini menghasilkan output berupa resistansi, dan agar dapat berfungsi dengan baik, diperlukan pasokan tegangan sebesar +5V. Nilai resistansi yang dihasilkan kemudian diubah menjadi tegangan, yang selanjutnya dapat dibaca dan diproses oleh mikrokontroler.[10]. Berikut bentuk fisik sensor *flex* pada Gambar 2.3. dibawah ini.



Gambar 2. 3 Sensor *flex*

(Sumber:https://circuitdigestcom.translate.goog/)

Sensor *Flex* disini digunakan untuk mendeteksi perubahan postur tubuh dengan mengukur fleksibilitas atau lenturan. Sensor *flex* dapat dipasang pada daerah depan tubuh, semacam perekat yang di tempelkan di bawah dada untuk mendeteksi apakah pengguna duduk dalam posisi tegak atau membungkuk.

2.5. Sensor Ultrasonik HC-SR04

Sensor Ultrasonik digunakan untuk mendeteksi keberadaan pengguna di kursi dengan mengukur jarak antara sensor dan objek (pengguna). Jika pengguna duduk,sensor akan mendeteksi keberadaannya dan memulai penghitungan durasi duduk. Berikut bentuk fisik sensor ultrasonik pada Gambar 2.4 dibawah.

2. 4 Sensor Ultrasonik

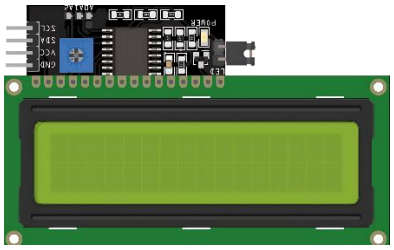
(Sumber : https://www.andalanelektro.id/)

Cara kerja sensor *ultrasonic* dengan cara (*transmitter*) memantulkan gelombang suara *ultrasonic*,lalu gelombang di tangkap oleh *(receiver*) pada sensor ultrasonik.Sensor ultrasonik memiliki frekuensi yang sangat tinggi yaitu dengan gelombang frekuensi 20.000 *Hz*[11].

2.6. *LCD* 16x2 I2C

Lcd 16x2 I2c digunakan untuk menampilkan informasi yang bisa di lihat antar muka,yang berisi tentang status postur tubuh apakah normal atau bungkuk dan berisi *timer* yang telah kita atur di *web.LCD* juga menampilkan jika waktu telah melewati 1 jam maka *lcd* akan memunculkan " 1 Jam Berlalu " dan jika waktu habis dia akan menampilkan tulisan " waktu habis ". Berikut

bentuk fisik *LCD I2C* pada gambar 2.5 dibawah.



2.5 *LCD* 16x2 I2C

(Sumber : https://docs-cirkitdesigner-com.translate.goog/component/)

I2C LCD adalah modul *LCD* yang dikontrol melalui komunikasi serial sinkron menggunakan protokol *I2C (Inter-Integrated Circuit)*, juga dikenal sebagai *TWI (Two Wire Interface)*.[12]. Pada umumnya, modul *LCD* bekerja dengan sistem kendali paralel, baik pada jalur data maupun sinyal kontrolnya.

2.7. *RTC DS3231*

RTC DS3231 (Real Time Clock) merupakan rangkaian elektronika yang berfungsi untuk menyimpan dan menghitung waktu secara akurat, mulai dari detik, menit, jam, hari, tanggal, bulan, hingga tahun. Komponen ini memiliki tingkat akurasi yang tinggi dalam pelacakan waktu, tahan terhadap perubahan suhu lingkungan, serta dilengkapi dengan rangkaian kapasitor tuning yang dapat mengatur secara otomatis untuk menjaga kestabilan frekuensi detaknya. . *RTC* biasanya berupa *IC* yang mempunyai *clock* sumber sendiri dan internal *batery* untuk menyimpan data waktu dan tanggal. Jika *system* komputer/*microcontroller* mati maka waktu dan tanggal didalam memori *RTC* tetap *update* [13]. Bentuk fisik *RTC* dapat ditunjukkan pada gambar 2.6 berikut.

Gambar 2. 6 *RTC* DS3231

(Sumber : https://blog.indobot.co.id/)

2.8. *Visual Studio Code*

Visual Studio Code (VS Code) adalah editor teks ringan namun tangguh yang dikembangkan oleh Microsoft dan tersedia untuk berbagai sistem operasi seperti *Linux, macOS, dan Windows*. Editor ini secara langsung mendukung bahasa pemrograman seperti *JavaScript, TypeScript,* dan *Node.js,* dan dapat diperluas untuk mendukung bahasa lain seperti *C++, C#, Python, Go,* dan *Java* melalui plugin yang tersedia di pasar resminya.[14]. Pada alat ini ,*software* disini berfungsi untuk memonitoring *set timer* ,menampilkan nilai adc dari sensor *flex* dan status kebungkukan tubuh secara *real-time*. *Software Visual Studio Code* bisa dilihat pada Gambar 2.7 berikut.

Gambar 2. 7 *Visual Studio Code*

(Sumber : https://en.wikipedia.org/wiki/Visual_Studio_Code)

2.9. *Firebase*

Firebase Realtime Database adalah layanan basis data berbasis *cloud* yang bekerja secara *real time* dan mendukung berbagai platform seperti *Android, iOS,* dan *Web*. Data disimpan dalam format terstruktur *JSON* (*JavaScript Object Notation*). *Firebase* secara otomatis menyinkronkan data dengan aplikasi klien yang terhubung, sehingga aplikasi *multi-platform* yang menggunakan *SDK Android, iOS,* atau *JavaScript* akan segera menerima pembaruan data setiap kali terjadi perubahan saat terhubung ke server.[15]. *Firebase* disini digunakan untuk menyimpan data *logger* lama waktu duduk dan juga *timer* secara *real-time*. *Software firebase* bisa dilihat pada Gambar 2.8 berikut.

Gambar 2. 8 *Firebase*

(Sumber : https://firebase.google.com/brand-guidelines?)

2.10. Sistem Peringatan dan Notifikasi

Sistem ini menggunakan *buzzer* sebagai peringatan audio ketika pengguna duduk dalam postur yang salah atau durasi duduk yang terlalu lama. *Buzzer* akan berbunyi untuk mengingatkan pengguna agar memperbaiki postur tubuh atau beristirahat sejenak.Selain itu, data postur tubuh dan durasi duduk akan ditampilkan pada *LCD* dan dikirim ke *Firebase* untuk pemantauan lebih lanjut.

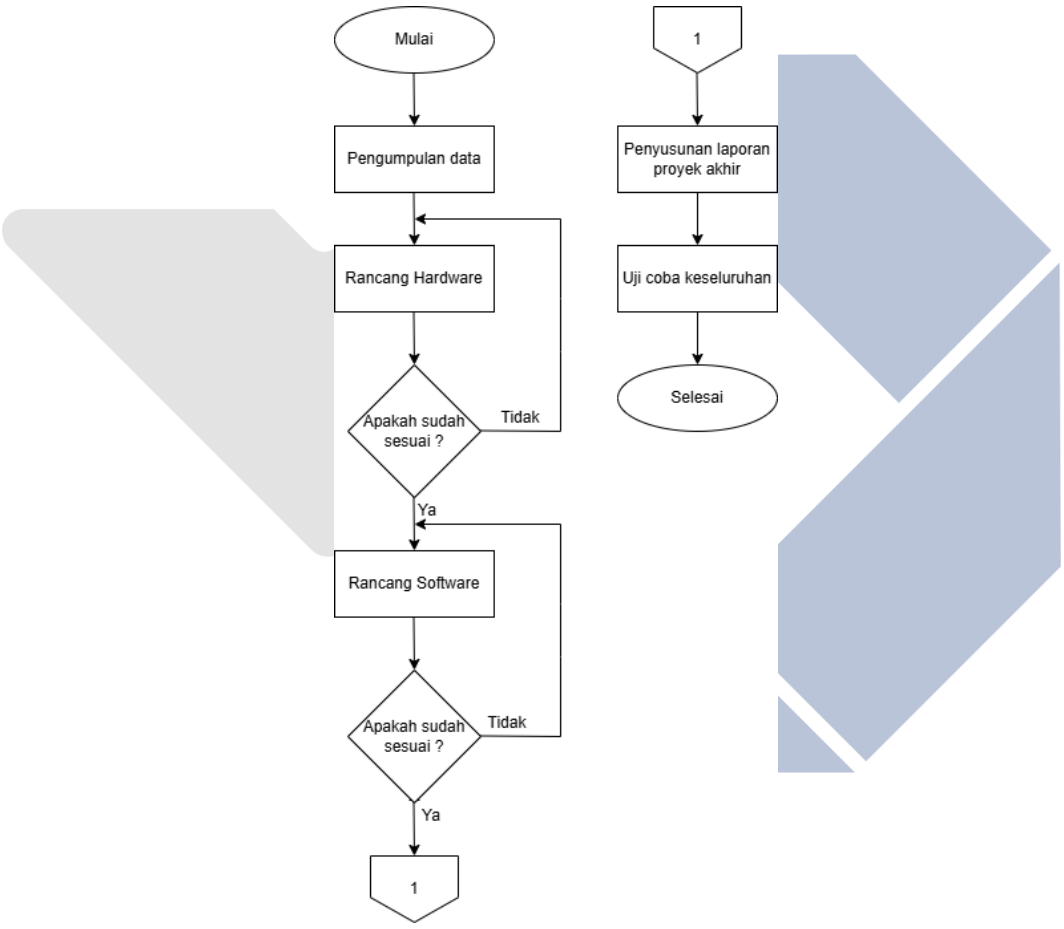
2.11. Studi Terkait Sebelumnya

Sebelumnya, penelitian tersebut menggunakan Arduino untuk mendeteksi postur tubuh dan durasi duduk,juga sensor *flex* digunakan untuk mengukur kelengkungan tubuh bagian perut atas[6][16]. Namun, pada sebelumnya tidak bisa setting waktu yang kita inginkan dan sistem tersebut belum terintegrasi dengan *web* atau penyimpanan data untuk pemantauan jangka Panjang. Penelitian ini mengembangkan sistem yang lebih canggih dengan memanfaatkan *IoT* untuk memungkinkan pemantauan jarak jauh dan penyimpanan data di *cloud.* Jadi, disini juga menggunakan perut bagian atas sebagai patokannya dari penelitian sebelumnya untuk mengukur kelengkungan tubuh dengan sensor *flex*, tetapi menambahkan sabuk sebagai pemasangannya.

BAB III
METODE PELAKSANAAN

3.1. Tahapan Perancangan dan Pembuatan Alat

Tahapan yang dilakukan meliputi studi literatur, perancangan sistem, pembuatan sistem, pengumpulan data, dan pengujian. Alur proses selengkapnya dapat dilihat pada Gambar 3.1 di bawah ini.



Gambar 3. 1 Tahapan-Tahapan Pelaksanaan

(Sumber : Dokumentasi Pribadi)

3.2. Teknik Pengumpulan Data

Pengumpulan data merupakan suatu proses sistematis dalam mengumpulkan dan mengukur informasi yang berkaitan dengan variabel yang diteliti, dengan tujuan untuk menjawab pertanyaan penelitian, menguji hipotesis, dan mengevaluasi hasil yang berkaitan dengan proyek akhir. Data yang telah diperoleh kemudian dianalisis dan diaplikasikan dalam pelaksanaan proyek. Teknik pengumpulan data yang digunakan adalah sebagai berikut.

3.2.1. Studi Pustaka

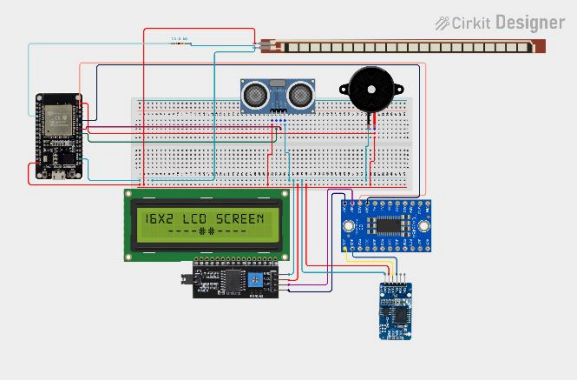
Studi pustaka yang dilakukan yaitu mencari, membaca dan mempelajari referensi jurnal dan buku-buku mengenai penelitian-penelitian terdahulu dan beberapa peneliti yang relevan dengan sensor *flex*, berkaitan dengan *software* arduino, ESP32 dan kontrol *IoT*, juga referensi-referensi lain terkait hal-hal yang mendukung dasar teori.

3.3. Rancang *Hardware*

Untuk rancang *hardware* nya yaitu ada 2, rangkaian elektrik dan rancang *box*. Berikut dibawah ini penjelasannya.

3.3.1. Rangkaian Elektrik

Rangkaian elektrik ini merupakan langkah awal rancangan antar komponen sebelum dipastikan tidak ada kesalahan sambung antar komponen. Diagram rangkain elektrik ini menggunakan *software* perancang elektronik untuk mempermudah proses perakitan untuk meminimalisir terjadinya salah sambung. Rangkaian elektrik alat ini disusun dengan mengintegrasikan beberapa komponen utama seperti ESP32, sensor ultrasonik, sensor *flex*, *RTC*, dan *LCD I2C*. Semua komponen dirangkai menggunakan skema kabel yang dapat dilihat pada Gambar 3.2 berikut ini.

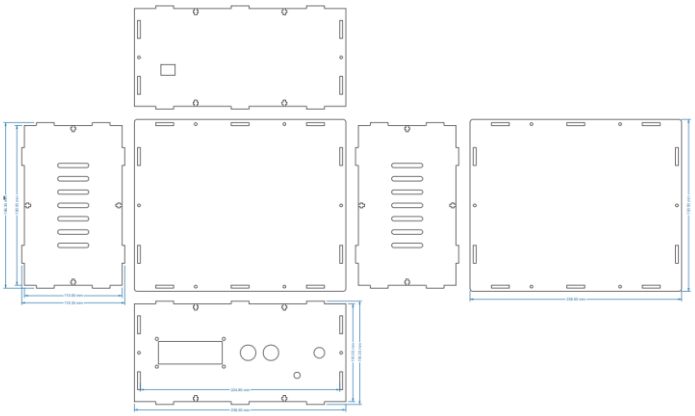


Gambar 3. 2 Rangkaian Elektrik

(Sumber : Dokumentasi Pribadi)

3.3.2. Rancang *Box*

Perancangan desain kontruksi alat merupakan desain untuk gambaran umum atau pedoman dalam pembuatan alat proyek akhir , proses perancangan desain konstruksi *box* menggunakan *software*. Bahan yang digunakan yaitu akrilik. Desain kontruksi *box* dapat dilihat pada Gambar 3.3 berikut.



Gambar 3. 3 Desain *Box*

(Sumber : Dokumentasi Pribadi)

3.4. Rancang *Software*

Perancangan perangkat lunak dilakukan untuk mengatur alur kerja sistem pendeteksi lama waktu duduk dan posisi kebungkukan berbasis *IoT*. Tahapan ini mencakup pembuatan program utama menggunakan *software Visual Studio Code* (*VS Code*) dengan dukungan *PlatformIO* sebagai lingkungan pengembangan. Bahasa pemrograman yang digunakan adalah C++ dengan *library* tambahan untuk mendukung komunikasi dengan sensor *flex*, sensor ultrasonik HY-SRF04, *buzzer*, modul *LCD I2C*, serta koneksi *Firebase*. Logika program dirancang untuk membaca data sensor secara *real-time*, menghitung durasi duduk, serta mendeteksi perubahan postur berdasarkan batas ambang tertentu. Selain itu, sistem juga mengirimkan data ke *Firebase* dan menampilkan peringatan di *LCD* jika postur membungkuk atau waktu duduk telah melewati batas. Semua proses ini diuji dan disimulasikan terlebih dahulu sebelum diimplementasikan secara penuh pada perangkat keras.

3.4.1. Pembuatan Program Arduino

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

//      KREDENSIAL FIREBASE
// Ganti dengan API Key Firebase dan URL Database Anda
#define API_KEY "AIzaSyCSS16XPgn6qhUpCsoCh4yfeRsTiu9KboF"
#define DATABASE_URL "https://fixmonitoring-16ade-default-rtdb.firebaseio.com"

//      PIN HARDWARE
#define TRIG_PIN 5   // Pin Trigger Sensor Ultrasonik
#define ECHO_PIN 18  // Pin Echo Sensor Ultrasonik
#define BUZZER_PIN 23 // Pin Buzzer
#define FLEX_PIN 34  // Pin Flex Sensor (Analog)
#define TCAADDR 0x70 // Alamat I2C TCA9548A I2C Multiplexer

//      OBJEK GLOBAL
LiquidCrystal_I2C lcd(0x27, 16, 2); // Inisialisasi LCD I2C dengan alamat 0x27, 16 kolom, 2 baris
RIC_DS4231 ric;                     // Inisialisasi objek RIC

FirebaseData fbdo;   // Objek untuk data Firebase
FirebaseAuth auth;   // Objek untuk otentikasi Firebase
FirebaseConfig config; // Objek untuk konfigurasi Firebase
bool signupOK = false; // Status pendaftaran Firebase

//      VARIABEL KONTROL TIMER & STATUS
unsigned long countdownTime = 0;      // Waktu hitung mundur (detik)
unsigned long previousMillis = 0;     // Millis terakhir untuk update timer (untuk non blocking delay)
bool timerStarted = false;            // Flag apakah timer sudah dimulai
bool timerExpired = false;            // Flag apakah timer sudah habis
bool buzzerState = false;             // Status buzzer (ON/OFF)
unsigned long buzzerMillis = 0;       // Millis terakhir untuk mengontrol kedipan buzzer
```

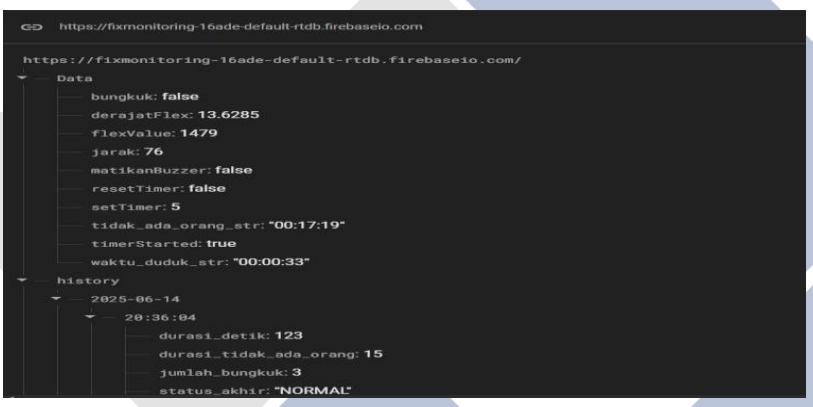
Gambar 3. 4 Program Arduino

(Sumber : Dokumentasi Pribadi)

Pada Gambar 3.4 ini, dilakukan penulisan dan pengujian program menggunakan bahasa C++ di Arduino (melalui *VS Code* dan *PlatformIO*) untuk mengontrol semua komponen sistem. Program dibuat untuk membaca data dari sensor *flex* sebagai pendeteksi postur, sensor ultrasonik HY-SRF04

untuk mengukur keberadaan dan jarak tubuh, serta mengatur logika waktu duduk. Data dari sensor diproses untuk menentukan apakah pengguna duduk terlalu lama atau membungkuk. Hasilnya ditampilkan di *LCD I2C* dan dikirim ke *Firebase*. Selain itu, program juga mengatur aktivasi *buzzer* sebagai alarm peringatan ketika waktu duduk melewati batas atau postur tidak ideal. Fungsi-fungsi seperti koneksi *Wi-Fi*, pembacaan waktu dari *RTC*, pengaturan *timer* dari *Firebase*, dan penyimpanan data juga dikembangkan dalam bagian ini.

3.4.2. Pembuatan *Firebase*



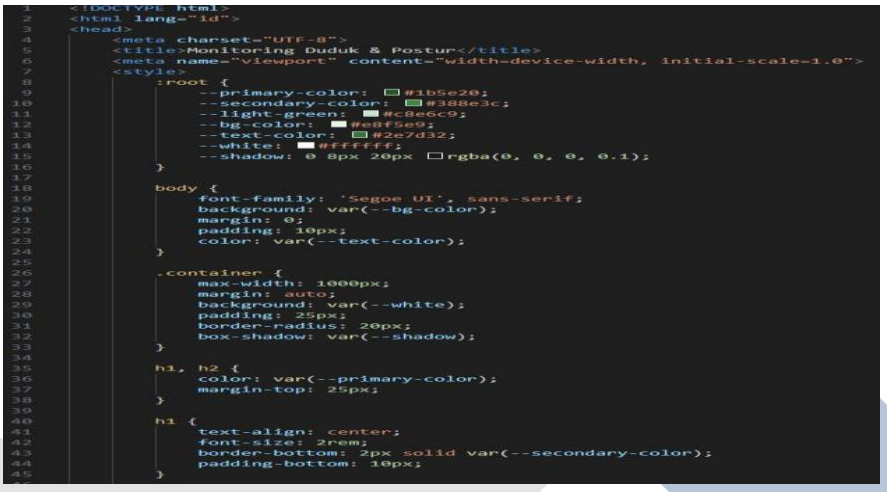
Gambar 3. 5 Perancangan *Firebase*

(Sumber : Dokumentasi Pribadi)

Pada Gambar 3.5 ini, dilakukan perancangan komunikasi antara mikrokontroler *ESP*32 dan *Firebase Realtime Database* sebagai media penyimpanan dan pemantauan data secara *cloud*. *Firebase* digunakan untuk menyimpan data sensor secara *real-time*, seperti nilai *flexValue* dari sensor *flex*, jarak dari sensor ultrasonik, status bungkuk (*true/false*), kontrol *buzzer* melalui matikan *Buzzer*, serta pengaturan waktu duduk dengan *setTimer*. ESP32 diprogram agar secara otomatis mengirim dan membaca data dari *node* tersebut menggunakan *library Firebase_ESP_Client*. Hal ini memungkinkan pengguna untuk memantau kondisi postur tubuh dan waktu duduk dari jarak jauh melalui *Firebase*, serta mengatur *timer* atau mematikan *buzzer* langsung melalui *database*. Dengan sistem ini, alat menjadi lebih

fleksibel dan berbasis *IoT* sepenuhnya, mendukung pemantauan dan kendali secara *online*.

3.4.3. Pembuatan Aplikasi *Visual Studio Code*



Gambar 3. 6 Perancangan *Visual Studio Code*

(Sumber : Dokumentasi Pribadi)

Pembuatan tampilan *web* seperti yang terlihat pada Gambar 3.6 diatas , dilakukan untuk memberikan antarmuka visual kepada pengguna dalam memantau kondisi waktu duduk dan postur tubuh secara *real-time*. *Web* ini dikembangkan menggunakan *HTML* dan *CSS* yang ditulis langsung di *Visual Studio Code* (*VS Code*). Pada halaman *web* terdapat elemen-elemen utama seperti judul, kontainer data, dan input *timer* yang dirancang responsif agar dapat diakses dari berbagai perangkat. Data seperti nilai *flex* sensor, jarak dari sensor ultrasonik, status postur, serta kontrol *buzzer* dapat ditampilkan dan disesuaikan melalui integrasi dengan *Firebase Realtime Database*. Desain *web* menggunakan warna-warna yang nyaman di mata, serta struktur *layout* yang rapi untuk memudahkan pemantauan oleh pengguna. Hal ini mendukung tujuan sistem *IoT* secara keseluruhan, yaitu memberikan informasi dan kendali dari jarak jauh secara praktis dan efisien

3.5. Uji Coba Keseluruhan

Uji coba dilakukan jika sudah menyelesaikan seluruh pekerjaan kontruksi, sistem kontrol dan *monitoring box*. Uji coba ini bertujuan untuk memastikan bahwa kontruksi peralatan, sistem kontrol dan *monitoring* sudah sesuai dengan desain yang telah direncanakan .

3.6. Penyusunan Laporan Proyek Akhir

Penyusunan laporan akhir merupakan tahap terakhir dalam pelaksanaan proyek akhir, yang bertujuan untuk merangkum semua aspek penting dari proyek. Laporan ini meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, landasan teori, metode pelaksanaan, pembahasan, hasil, serta simpulan dan saran.

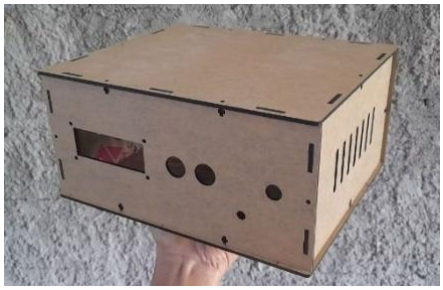
BAB IV
PEMBAHASAN

4.1. Deskripsi Alat

Box atau alat ini memanfaatkan teknologi *Internet of Things* untuk mengukur postur tubuh secara *real-time* dan menyimpan data. Cara kerja alat ini secara simplenya yaitu pengguna duduk di kursi mengenakan sabuk yang sudah ditempel sensor *flex* untuk diikatkan ke pinggangnya. Posisi sensor *flex* harus ditengah atau tepat di bagian perut. Di depan pengguna tepatnya di atas meja , *box* yang sudah dilengkapi dengan sensor ultrasonik sebagai pendeteksi jarak ada tidaknya orang/objek, *lcd* menampilkan *timer* dan status kebungkukan dan *buzzer* memberi tanda peringatan, sudah diletakkan disana. Sebelum memulai perhitungannya ESP32 tersebut harus lebih dulu dihubungkan melalui *Wi-Fi*. Jika sudah kita bisa memulai perhitungan *timer*-nya dan akan terus berjalan. Jika pengguna duduk terlalu bungkuk , maka *buzzer* berbunyi dan lcd memberi notifikasi "bungkuk". Pada posisi normal maka *lcd* menampilkan notifikasi "normal". Waktu *timer* bisa kita *setting* dari *web* PC atau *smartphone* sesuai keinginan kita. Kita juga bisa mematikan *buzzer* atau *reset* lewat *web*-nya. Pada *web* yang ditampilkan yaitu nilai besar derajat kelengkungan sensor *flex*, status kebungkukan, dan jarak terbacanya sensor ultrasonik.

4.2. Pembuatan *Hardware* Mekanik *Box*

Dalam pembuatan *box* , kami menggunakan bahan akrilik untuk dirancang dan kemudian di bikin oleh jasa *custom* akrilik , jadi barangnya datang langsung sudah jadi. Berikut Gambar 4.1 bentuk sudah jadi dari *box*-nya.

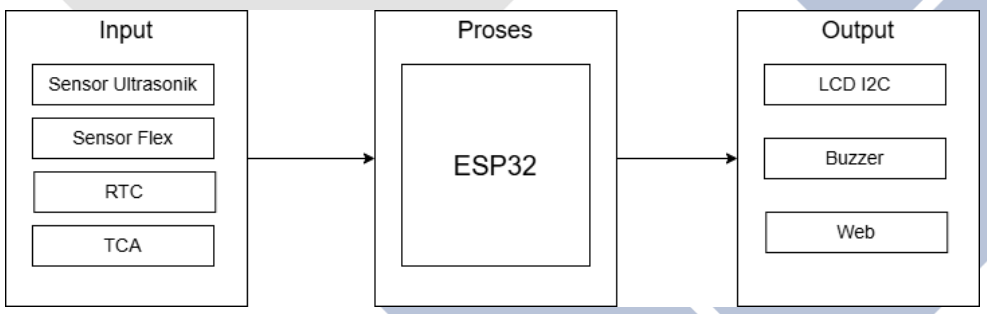


Gambar 4. 1 *Box*

(Sumber : Dokumentasi Pribadi)

4.3. Perancangan dan Pembuatan *Hardware* Elektrik *Box*

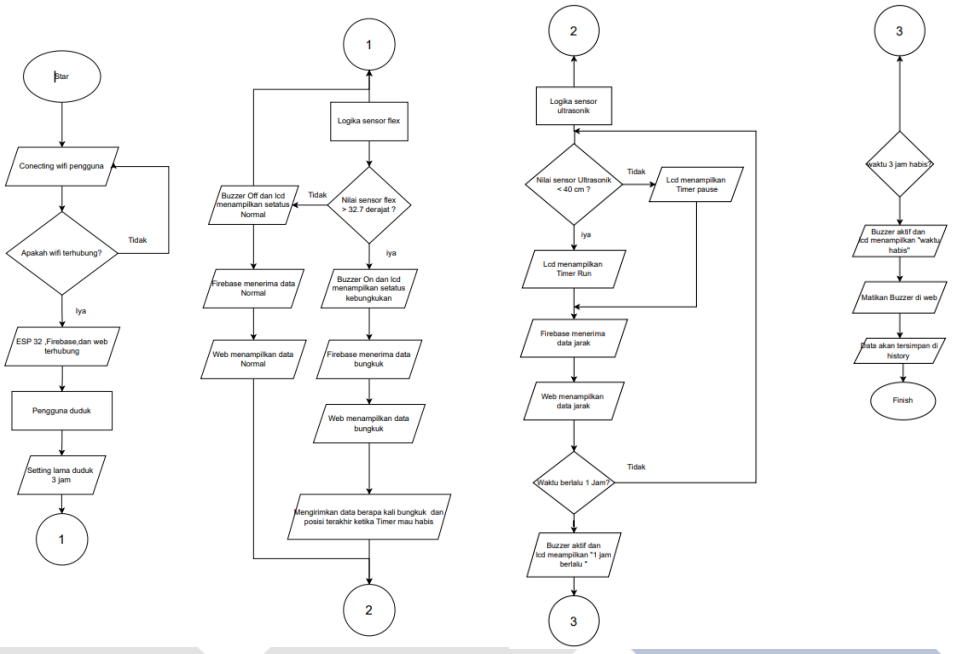
Setelah merancang komponen elektrik dengan baik untuk sistem kontrol *box*-nya,langkah selanjutnya yaitu membuat diagram blok. Diagram blok memiliki macam-macam tujuan, yaitu memberi penjelasan fungsi dari suatu alat, menganalisa rangkaian dan mempermudah merancang kontruksi alatnya. Berikut Gambar 4.2 diagram blok dibawah.



Gambar 4. 2 Diagram Blok

(Sumber : Dokumentasi Pribadi)

Setelah menyelesaikan perancangan komponen elektrik untuk sistem kontrol *box*, langkah selanjutnya adalah menyusun diagram blok. Gambar 4.2 memperlihatkan hubungan antara komponen input (seperti sensor *flex* dan sensor ultrasonik) dengan ESP32 sebagai pusat kontrol, dan output-nya berupa *LCD*, *buzzer*, serta tampilan *web*.



Gambar 4. 3 *Flowchart* Kerja Alat

(Sumber : Dokumentasi Pribadi)

Untuk memahami urutan kerja alat secara sistematis, disusun flowchart seperti pada Gambar 4.3 yang menjelaskan tahapan dari inisialisasi sistem hingga pengiriman data ke Firebase. Berikut cara kerjanya.

1. Inisialisasi Sistem

Alat mulai bekerja saat dinyalakan. ESP32 terhubung ke jaringan *Wi-Fi*. Setelah koneksi berhasil, ESP32 juga langsung terhubung ke *Firebase Realtime Database*. Sistem menunggu input pengguna berupa pengaturan durasi duduk (misalnya 3 jam), yang dikirim dari web ke *Firebase*.

2. Deteksi Keberadaan Pengguna (Sensor Ultrasonik)

Sensor ultrasonik mengukur jarak ke tubuh pengguna. Jika jarak < 40 cm → dianggap duduk, maka *timer* mulai berjalan. Jika jarak ≥ 40 cm → dianggap tidak duduk, maka *timer* akan *pause* sementara.

3. Deteksi Postur Tubuh (Sensor *Flex*)

Sensor *flex* membaca nilai *ADC* dan dihitung menjadi derajat tekukan. Jika derajat > 32.7° → status membungkuk, maka *buzzer* aktif, *LCD*

menampilkan "Membungkuk" , status dikirim ke *Firebase* dan ditampilkan di *web*. Jika ≤ 32.7° → status normal, maka *buzzer* mati, *LCD* menampilkan "Normal", Status dikirim ke *Firebase* dan ditampilkan di *web*.

4. Pengingat Setiap 1 Jam Duduk

Jika pengguna duduk selama 1 jam tanpa jeda, *buzzer* berbunyi sebentar *LCD* menampilkan "1 jam berlalu", ini berfungsi sebagai pengingat untuk istirahat.

5. *Timer* Habis (Misalnya Setelah 3 Jam)

Setelah *timer* mencapai waktu habis, maka *buzzer* menyala ,*LCD* menampilkan "Waktu Habis", *web* juga menampilkan status selesai. Pengguna bisa mematikan *buzzer* melalui tombol di *web*-sistem, selanjutnya menghitung dan mengirim data log. Data yang dikirim ke *Firebase* yaitu Waktu mulai, Waktu selesai, Lama duduk, Jumlah membungkuk, Posisi akhir (normal/bungkuk) , kemudian Data disimpan di path:/history/2025-06-28/10-30-00.

6. *Web Monitoring* Riwayat Per Tanggal

Web hanya menampilkan riwayat berdasarkan tanggal yang dipilih oleh pengguna. Contoh: jika pengguna memilih 2025-06-28, maka data diambil dari:

/*history*/2025-06-28 Tidak bisa melihat semua tanggal sekaligus. Hanya satu hari per tampilan

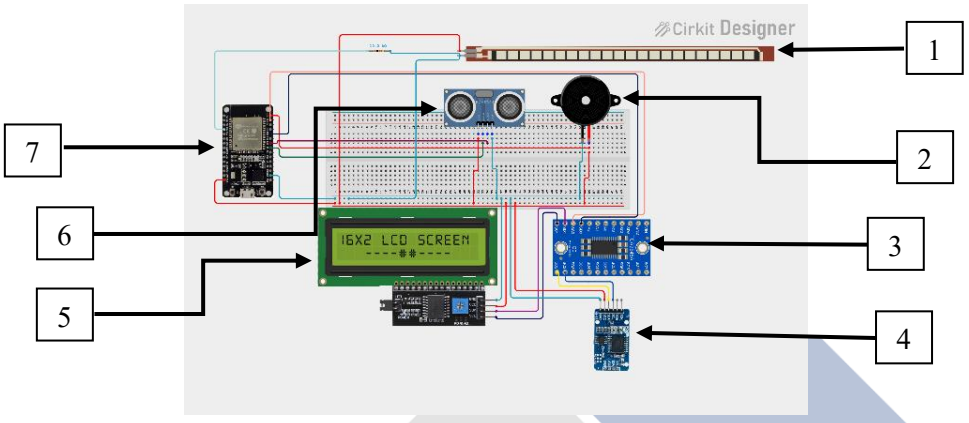
7. *Download JSON* (Per Hari)

Setelah data tampil untuk tanggal tertentu, pengguna bisa klik tombol "Download JSON". Sistem akan mengambil data dari *Firebase* untuk tanggal tersebut dan mengubahnya ke file .json.File otomatis di-*download* dengan nama: history_2025-06-28.json

4.3.1. Perancangan *Hardware* Elektrik *Box*

Tahap ini merupakan perencanaan kerangka pengendalian terkait kerja alat ini. Pada bagian ini ada beberapa input yang berperan penting untuk alat ini, seperti sensor ultrasonik, sensor *flex* ,dan *RTC*. Hasil nilai dari input ini

akan diteruskan ke ESP32 dan keluaran atau *output*-nya yaitu *buzzer*, *lcd* dan *web*. Berikut Gambar 4.4 skema pengkabelan yang dibuat Fritzing untuk rangkaian listrik.



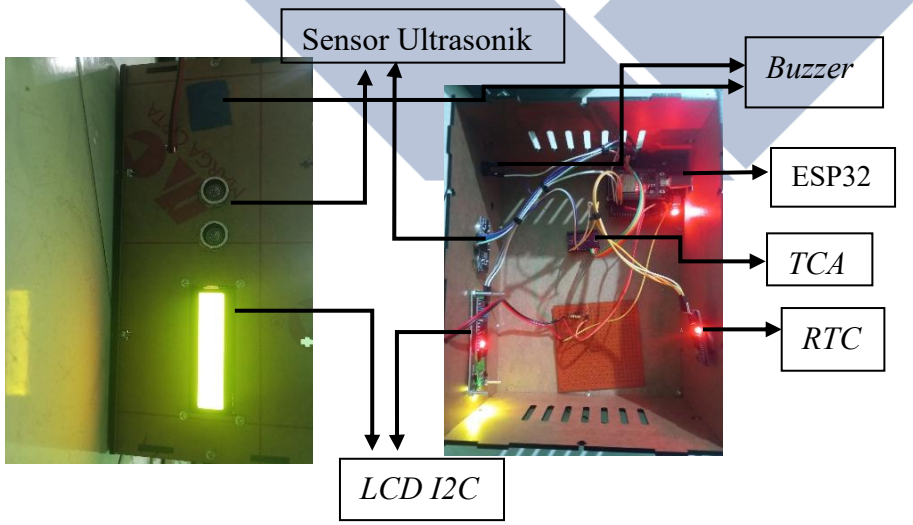
Gambar 4. 4 Rangkaian *Hardware*

(Sumber : Dokumentasi Pribadi)

Ket :

1. Sensor *flex*
2. *Buzzer*
3. *TCA*
4. *RTC DS3231*
5. *LCD* I2C
6. Sensor Ultrasonik
7. ESP32

4.3.2. Pembuatan *Hardware* Elektrik *Box*



Gambar 4. 5 *Box Monitoring*

(Sumber : Dokumentasi Pribadi)

Pada Gambar 4.5 terdapat *box monitoring* yang memiliki berbagai macam komponen yang saling berhubungan, antara lain ,ESP32 ,*buzzer*, *TCA9548A I2C Multiplexer*, *RTC* DS3231, *LCD* I2C, dan sensor ultrasonik. Sensor *flex*-nya diletekkan di bagian luar *box*.

4.4. Pengujian *Hardware* Elektrikal *Box*

Pada pengujian *Hardware* Elektrikal *box* ini berfungsi untuk menguji komponen apakah bisa berfungsi dengan baik. Komponen yang kami gunakan yaitu sensor *flex* dan sensor ultrasonik.

4.4.1. Pengujian Sensor *flex*

Pada pengujian sensor *flex* terdapat dua tabel, yang terdiri dari data kalibrasi sensor *flex* dan pengujian kepekaan sensor *flex*.

4.4.1.1. Kalibrasi Sensor *Flex*

Tabel 4. 1 Hasil Kalibrasi Sensor *Flex*

No.	Postur	Nilai ADC	Sudut Manual	Sudut Terbaca	Selisih Derajat
1	Normal	1472	15	14,1	0,9
2	Bungkuk	1014	45	45,4	-0,4

Pada Tabel 4.1 diatas, merupakan percobaan pengkalibrasian sensor *flex*. Disini kita mencari nilai normal derajat dan dibandingkan dengan pengukuran suduh dengan cara manual menggunakan busur, setelah itu mencari selisih dari nilai yang terbaca dan nilai manual. Berikut rumus selisih perbandingan sudut manual dengan sudut terbaca.

Rumus selisih $=$ Sudut Manual $-$ Sudut Terbaca

Sudut Manual = Hasil pengukuran dengan busur

Sudut Terbaca = Hasil pengukuran dari *lcd*

4.4.1.2. Kepekaan Sensor *Flex*

Tabel 4. 2 Hasil Kepekaan Sensor *Flex*

ADC Value	Derajat Flex (°)	Status
1493	12,67	NORMAL (BUZZER OFF)
1396	19,31	NORMAL (BUZZER OFF)
1367	21,3	NORMAL (BUZZER OFF)
1360	21,78	NORMAL (BUZZER OFF)
1277	27,47	NORMAL (BUZZER OFF)
1120	37,17	BUNGKUK (BUZZER ON)
1085	39,56	BUNGKUK (BUZZER ON)
1052	41,8	BUNGKUK (BUZZER ON)
1023	43,79	BUNGKUK (BUZZER ON)
980	46,68	BUNGKUK (BUZZER ON)

Tabel 4.2 diatas merupakan hasil pengujian kelengkungan sensor *flex* dalam bentuk derajat, nilai *adc* yang dihasilkan dan status normal atau bungkuk. Rumus perhitungan dari derajat yang dihasilkan sebagai berikut.

Derajat *Flex* (°) = $(0{,}0685 \times ADC) + 114{,}94$

ADC : Nilai yang dihasilkan dari sensor *flex*

Hasil pengujian sensor *flex* pada alat ini menunjukkan kemampuan deteksi postur yang konsisten, sejalan dengan prinsip kerja sensor flex yang telah banyak digunakan dalam penelitian deteksi postur tubuh sebelumnya [6] [10].

4.4.2. Pengujian Sensor Ultrasonik

Pada bagian ini, ditampilkan data-data terkait pengujian sensor ultrasonik. Berikut datanya dibawah ini.

Tabel 4. 3 Hasil Perbandingan dan Kepekaan Sensor Ultrasonik

No.	nilai jarak terbaca di lcd (cm)	nilai jarak terbaca penggaris (cm)	Kondisi Timer	Nilai eror
1	9,5	9,7	Normal	2,06%
2	17,1	17,2	Normal	0,58%
3	21,3	20,9	Normal	1,91%
4	25,1	24,9	Normal	0,80%
5	27,4	27,5	Normal	0,36%
6	29,4	29,6	Normal	0,68%
7	33,5	34	Stop	1,47%
8	40,4	41,7	Stop	3,12%
9	52,5	53,6	Stop	2,05%
10	57,2	57,6	Stop	0,69%
		Nilai rata-rata eror		1,37%

Tabel 4.3 diatas merupakan data yang kami dapatkan dari pengukuran jarak sensor ultrasonik dari tampilan *lcd* dan pengukuran langsung menggunakan penggaris juga kepekaan sensornya. Kepekaan sensor ultrasonik terlihat normal dari data ke-1 sampai ke-6, namun dari data ke-7 sampai ke-10 sensor ultrasonik tidak mendeteksi atau menghitung *timer* lagi. Perbandingan tersebut kemudian dicari nilai persentase *error*-nya dan rata-ratanya juga. Persentase *error* paling kecil pada data ke-5 dan paling besar pada data ke-8. Rata-rata persentase *error*-nya yaitu 1.37%. Berikut rumus perhitungan persentase *error*-nya juga rata-ratanya .

$$Error\ (\%) = \left|\frac{\text{Nilai } lcd - \text{Nilai Penggaris}}{\text{Nilai Penggaris}}\right| \times 100\%$$

Nilai lcd = Hasil pengukuran *lcd*

Nilai Penggaris = Hasil pengukuran penggaris

$$\text{Rata} - \text{rata} = \frac{\text{Total } error}{\text{Jumlah data}}$$

Total *error* = Penjumlahan semua total persentase *error*

Jumlah data = Jumlah data pengujian keseluruhan

Data ke-1 sampai 3 kemungkinan kami mengukurnya tidak akurat jadi persen *error* lumayan besar. Data ke-8 dan 9 menunjukkan persen angka *error* yang lumayan besar, dikarenakan kami mengukurnya menggunakan 1 penggaris yang panjangnya 30 cm, jadi untuk jarak lebih dari 30 cm kami mengukurnya dengan penggaris yang di-*double*, kemungkinan adanya kelebihan atau kekurangan dari pengukuran tersebut jadi timbullah *error* yang lumayan besar. Akurasi deteksi jarak sensor ultrasonik sebesar 1.37% (Tabel 4.3) menunjukkan bahwa sensor ini cukup andal untuk mendeteksi keberadaan pengguna, yang merupakan peningkatan dari sistem sebelumnya yang mungkin tidak memiliki fitur deteksi keberadaan pengguna secara spesifik.

4.4.3. Pengujian *LCD*

Pengujian *lcd* dilakukan dengan melihat respon *lcd* ketika sensor *flex* lurus maka *lcd* akan menunjukkan normal dan sebaliknya ketika melengkung makan status membungkuk, dan *timer* berjalan. Status dari *LCD* dapat dilihat pada Gambar 4.6 berikut.

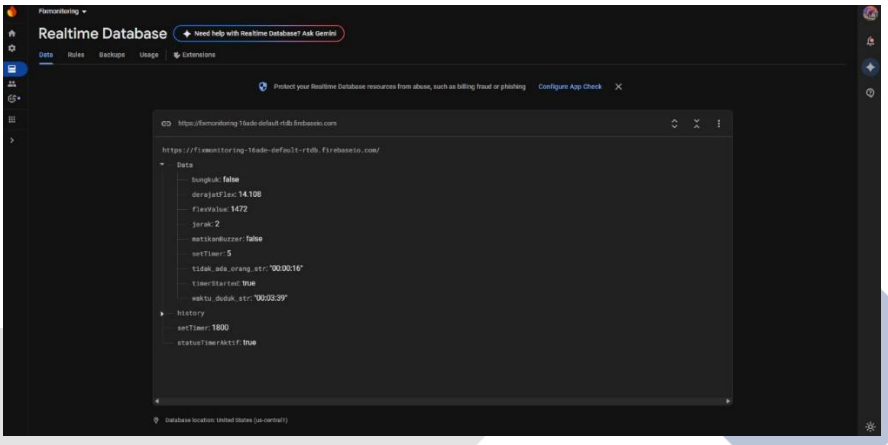
Gambar 4. 6 Status Pada *LCD*

(Sumber: Dokumentasi Pribadi)

4.5. Pengujian *Software IoT* Elektrikal *Box*

Pada pengujian *software* elektrikal *box* ini berfungsi untuk memonitor durasi lama duduk dan menyimpan data secara *real-time*. *Software* yang digunakan yaitu *firebase* dan *visual studio code*.

4.5.1. Analisis *Realtime Database* pada Aplikasi *Monitoring*



Gambar 4. 7 Hasil *Firebase*
(Sumber : Dokumentasi Pribadi)

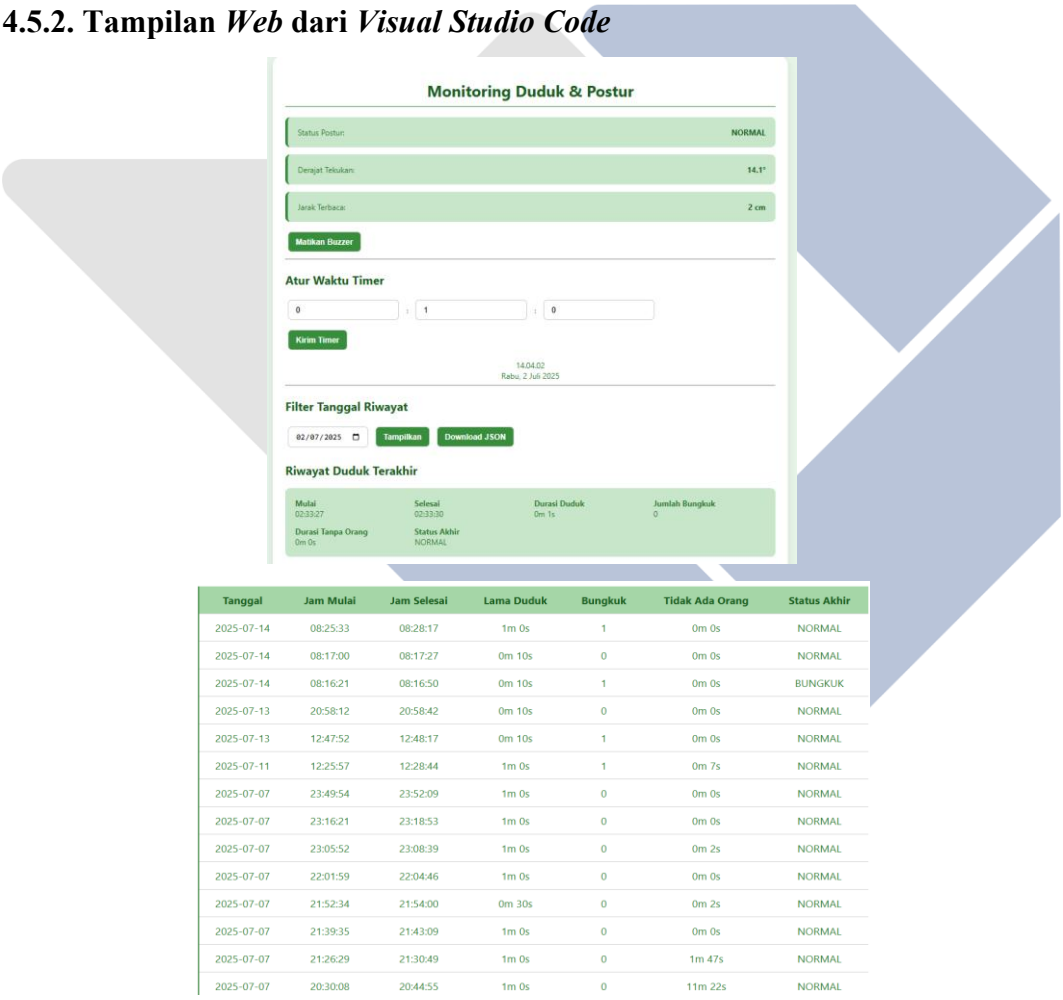
Pada Gambar 4.7 diatas merupakan hasil tampilan dari *firebase*. *Realtime Database* merupakan salah satu layanan *Firebase* yang berfungsi untuk menyimpan dan menyinkronkan data secara langsung (*real-time*), memungkinkan pengguna untuk berinteraksi dengan data yang selalu ter-*update* tanpa memerlukan *refresh* halaman. Pada gambar diatas menunjukkan struktur data yang disimpan dalam *Realtime Database*. Berikut adalah beberapa elemen penting yang tercantum dalam data tersebut.

- bungkuk: (*boolean*) Status apakah bungkuk atau tidak (*false*).
- derajat*Flex*: (*float*) Nilai derajat fleksibilitas (14.108).
- *flexValue*: (*integer*) Jumlah nilai fleksibilitas (1472).
- jarak: (*integer*) Jarak tertentu dalam aplikasi (2).
- matikan*Buzzer*: (*boolean*) Status pengaktifan *buzzer*.
- *setTimer*: (*integer*) Durasi *timer* yang ditetapkan (5 detik).
- tidak_ada_orang_str: (*string*) Waktu tidak ada orang terdeteksi

("00:00:16").

- *timerStarted*: (*boolean*) Status apakah *timer* telah dimulai (*true*).
- waktu_duduk_str: (*string*) Waktu total yang dihabiskan duduk ("00:03:39").
- Riwayat Penggunaan (*History*) :
- *setTimer*: (*integer*) Menunjukkan total *timer* yang ditetapkan dalam detik (1800 detik atau 30 menit).
- status*Timer*Aktif: (*boolean*) Status keaktifan timer (*true*).

4.5.2. Tampilan *Web* dari *Visual Studio Code*



Tanggal	Jam Mulai	Jam Selesai	Lama Duduk	Bungkuk	Tidak Ada Orang	Status Akhir
2025-07-14	08:25:33	08:28:17	1m 0s	1	0m 0s	NORMAL
2025-07-14	08:17:00	08:17:27	0m 10s	0	0m 0s	NORMAL
2025-07-14	08:16:21	08:16:50	0m 10s	1	0m 0s	BUNGKUK
2025-07-13	20:58:12	20:58:42	0m 10s	0	0m 0s	NORMAL
2025-07-13	12:47:52	12:48:17	0m 10s	1	0m 0s	NORMAL
2025-07-11	12:25:57	12:28:44	1m 0s	1	0m 7s	NORMAL
2025-07-07	23:49:54	23:52:09	1m 0s	0	0m 0s	NORMAL
2025-07-07	23:16:21	23:18:53	1m 0s	0	0m 0s	NORMAL
2025-07-07	23:05:52	23:08:39	1m 0s	0	0m 2s	NORMAL
2025-07-07	22:01:59	22:04:46	1m 0s	0	0m 0s	NORMAL
2025-07-07	21:52:34	21:54:00	0m 30s	0	0m 2s	NORMAL
2025-07-07	21:39:35	21:43:09	1m 0s	0	0m 0s	NORMAL
2025-07-07	21:26:29	21:30:49	1m 0s	0	1m 47s	NORMAL
2025-07-07	20:30:08	20:44:55	1m 0s	0	11m 22s	NORMAL

Gambar 4. 8 Hasil *Visual Studio Code*

(Sumber: Dokumentasi Pribadi)

Gambar 4.8 merupakan hasil kode perancangan dari *software Visual Studio Code.* Data yang ditampilkan yaitu status postur (normal/bungkuk),

derajat tekukan dari sensor *flex*nya, dan jarak terbacanya dari sensor ultrasonik. Tombol " Matikan *Buzzer*" berfungsi untuk mereset *buzzer* atau *off*. Kita juga bisa mengatur *timer* dengan jam, menit dan detik yang kita inginkan. Dibawahnya juga ada fitur tanggal riwayat. Kita bisa memilih tanggal berapa yang mau kita lihat data *monitoring*-nya dan juga bisa men-*download history* dengan unduhannya file "*json*". Terakhir ada riwayat *monitoring* duduk terakhir per-tanggal. Jadi , kita bisa lihat detail kapan waktu kita mulai menggunakan alat itu.

4.6. Pengujian Akhir Alat

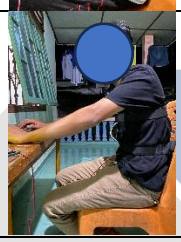
Ada beberapa bagian yang kami uji akhir alat ini. Berikut dibawah ini.

4.6.1. Pengujian Derajat Menggunakan Subjek

Pada bagian ini terdapat tabel / data mengenai pengambilan data per-orangan, meliputi waktu mulai, waktu selesai , lama waktu duduk, durasi tidak ada orang, jumlah bungkuk, status dan *set timer*. Berikut datanya.

Tabel 4. 4 Hasil Pengujian Derajat Menggunakan Subjek

No.	Nama pengguna	Status		Nilai derajat	
		Normal	Bungkuk	Normal	Bungkuk
1.	Orang ke - 1			14	45
2.	Orang ke - 2			13	40

3.	Orang ke - 3			9	32
4.	Orang ke - 4			15	37
5.	Orang ke - 5			13	80

Berdasarkan hasil pengujian pada Tabel 4.4 diatas terhadap lima orang pengguna, diperoleh data perbandingan sudut kelengkungan tubuh saat duduk pada posisi normal dan bungkuk. Pada posisi normal, nilai derajat kelengkungan berada pada kisaran 9° hingga 15° yang menunjukkan postur tubuh relatif tegak dan ergonomis. Sementara itu, pada posisi bungkuk, terjadi peningkatan sudut yang cukup signifikan dengan nilai antara 32° hingga 80° yang menunjukkan postur tubuh duduk kurang ideal dan berpotensi menimbulkan gangguan pada tulang belakang jika berlangsung dalam jangka waktu lama. Perbedaan nilai derajat tersebut menunjukkan bahwa sistem mampu mendeteksi perubahan postur tubuh dengan cukup jelas, sehingga valid untuk dijadikan dasar pemantauan postur tubuh pada pelaksanaan tugas akhir.

Kemampuan alat untuk mendeteksi postur bungkuk dan memberikan peringatan, serta mencatat durasi duduk dan jumlah bungkuk, merupakan pengembangan signifikan dibandingkan dengan alat sebelumnya yang hanya berfokus pada deteksi postur tanpa fitur pemantauan data historis dan notifikasi *real-time* [6].

4.6.2. Pengujian *Timer* Menggunakan Subjek

Pada bagian ini terdapat tabel yang berisi waktu mulai, waktu selesai, *set timer* (detik), selisih akurat (detik) dan error akurat (detik). Berikut data hasilnya ditampilkan pada Tabel 4.5 dibawah.

Tabel 4. 5 Hasil Pengujian *Timer* Menggunakan Subjek

No	Waktu Mulai	Waktu Selesai	Set Timer (detik)	Selisih Akurat (detik)	Error Akurat (detik)
1	23:57:58	00:26:46	600	1728	1128
2	00:29:18	00:44:38	300	920	620
3	00:47:38	00:56:49	120	551	431
4	00:57:39	01:04:07	120	388	268
5	01:06:50	01:49:25	900	2555	1655

Berdasarkan Tabel 4.5 hasil pengujian menunjukkan "*Error* Akurat (detik)". Kolom "*Error* Akurat (detik)" ini merepresentasikan selisih antara "Selisih Akurat (detik)" dengan "*Set Timer* (detik)". Dengan kata lain, nilai pada kolom "*Error* Akurat (detik)" mengindikasikan besarnya kesalahan yang terjadi antara durasi waktu yang terukur secara aktual ("Selisih Akurat") dengan durasi waktu yang telah ditentukan atau disetel pada *timer* ("*Set Timer*"). Semakin besar nilai *error* ini, semakin besar pula ketidaksesuaian antara waktu yang diharapkan dengan waktu yang sesungguhnya tercatat oleh sistem pengujian.

Perbandingan antara waktu duduk aktual dan waktu *set Timer* menunjukkan bahwa semua data memiliki selisih waktu positif yang signifikan, dengan rata-rata kesalahan sekitar 820 detik (±13 menit). Hal ini kemungkinan disebabkan oleh keterlambatan pembacaan sensor ultrasonik akibat proses *trigger-echo* yang tidak selalu *real-time*, serta keterlambatan pengiriman data ke *Firebase* yang menghambat *loop* utama ESP32. Sensor ultrasonik seperti HC-SR04 memerlukan jeda antar pembacaan dan jika tidak dikontrol secara *non-blocking*, pembacaan dapat tertunda atau gagal sehingga

sistem masih menganggap pengguna masih duduk meskipun mereka sudah tidak ada. Kondisi ini menunjukkan bahwa sistem pemantauan masih perlu dioptimalkan, baik dari segi logika pembacaan sensor maupun manajemen pengiriman data, agar dapat menghitung waktu duduk secara akurat dan responsif[17].

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil pengujian yang telah dilakukan dengan judul " Alat Pendeteksi Lama Waktu Duduk Dan Posisi Kebungkukan Tubuh Berbasis *IoT* " , dapat kita simpulkan yaitu :

1. *Realtime Database* memungkinkan penyimpanan dan pengambilan data secara *real-time*, yang sangat penting untuk aplikasi yang memerlukan pembaruan instan.
2. Data yang ditampilkan, seperti derajat *Flex*, *flexValue*, dan *set Timer*, menunjukkan bahwa database ini mendukung berbagai tipe data untuk memfasilitasi *monitoring* secara efektif.
3. Pencatatan riwayat membantu dalam analisis tren dan penggunaan data dari waktu ke waktu, yang berguna untuk pengambilan keputusan.
4. Sistem ini berhasil memberikan peringatan jika duduk dengan bungkuk dengan bunyi dari *buzzer* dan tampilan dari *lcd* bertuliskan "Bungkuk" secara *real-time* berbasis teknologi *IoT*.

5.2 Saran

Berdasarkan data hasil pengujian dengan judul " Alat Pendeteksi Lama Waktu Duduk Dan Posisi Kebungkukan Tubuh Berbasis *IoT* " , dapat ditambahkan beberapa saran yaitu :

1. Bisa ditambahkan sabuk dengan *wireless* agar bisa lebih memudahkan pengguna.
2. Memperkuat keamanan *web* dengan menambahkan kode keamanan.
3. Meningkatkan antarmuka pengguna untuk mempermudah pengguna dalam mengakses dan memahami data yang disimpan.
4. Melakukan lebih banyak uji coba pada sistem untuk mengidentifikasi dan memperbaiki bug atau masalah yang mungkin terjadi dalam proses penyimpanan atau pengambilan data.

5. Menggantikan modul atau *rtc* model lain untuk perhitungan *timer* dan waktu *real* terpisah, yang bisa *run timer* dan *pause timer*.

DAFTAR PUSTAKA

[1] Who.int, “Physical activity,” who.int. Accessed: Apr. 26, 2025. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/physical-activity

[2] American Chiropractic Association, “Back Pain Facts and Statistics,” American Chiropractic Association. Accessed: Jun. 03, 2025. [Online]. Available: https://handsdownbetter.org/health-and-wellness/back-pain-facts-and-statistics/

[3] N. L. P. Ekarini, Y. P. Susman, S. Suratun, N. Yardes, S. Manurung, and W. Wartonah, “Posisi Duduk dan Lama Duduk di Depan Komputer sebagai Faktor Risiko Keluhan Nyeri Punggung Bawah pada Karyawan Kantoran,” *Jkep*, vol. 8, no. 2, pp. 178–194, 2023, doi: 10.32668/jkep.v8i2.1338.

[4] T. A. H. Rashed, “Ergonomic Evaluation of Sitting Posture and Duration in Office Workers,” *Int. J. Occup. Saf. Ergon.*, vol. 28, no. 1, pp. 45–52, 2022, doi: 10.1080/10803548.2021.1872952.

[5] Rita Puspita Sari, “10 Penerapan Internet of Things di Bidang Kesehatan,” cloudcomputing.id. Accessed: Apr. 26, 2025. [Online]. Available: https://www.cloudcomputing.id/pengetahuan-dasar/iot-di-bidang-kesehatan#:~:text=Memantau Pasien Secara Remote,tanpa harus bertemu secara fisik.

[6] S. Salsabila, “Rancang Bangun Alat Koreksi Postur Dan Lama Waktu Duduk Dengan Flex Sensor Berbasis Arduino Uno Program Studi Teknik Informatika,” 2023.

[7] Sean, “Perangkat IoT,” Editor Industri. Accessed: Apr. 26, 2025. [Online]. Available: https://www.fanruan.com/id/glossary/iot/perangkat-iot

[8] Linknet.id, “5 Contoh Penerapan IoT dalam Bidang Kesehatan untuk Efisiensi dan Efektivitas,” linknet.id. Accessed: Apr. 26, 2025. [Online]. Available: https://www.linknet.id/article/penerapan-iot-dalam-bidang-kesehatan

[9] A. Imran and M. Rasul, “Pengembangan Tempat Sampah Pintar Menggunakan Esp32,” *J. Media Elektr.*, vol. 17, no. 2, pp. 2721–9100, 2020, [Online]. Available: https://ojs.unm.ac.id/mediaelektrik/article/view/14193

[10] M. Habib Nurfaizal and Y. M. Djaksana, “Prototype Sistem Kendali Robot ARM Gripper Manipulator menggunakan Flex Sensor Dan MPU6050 Berbasis Internet of Things,” eISSN: 2502-339X, pISSN: 1979-276X, 2020. doi: 10.30998/faktorexacta.v13i4.6598.

[11] T. N. Arifin, G. Febriyani Pratiwi, and A. Janrafsasih, “Sensor Ultrasonik Sebagai Sensor Jarak,” *J. Tera*, vol. 2, no. 2, pp. 55–62, 2022, [Online]. Available: http://jurnal.undira.ac.id/index.php/jurnaltera/

[12] H. Suryantoro, “Prototype Sistem Monitoring Level Air Berbasis Labview dan Arduino Sebagai Sarana Pendukung Praktikum Instrumentasi Sistem Kendali,” *Indones. J. Lab.*, vol. 1, no. 3, p. 20, 2019, doi: 10.22146/ijl.v1i3.48718.

[13] E. A. Pranata, U. Fatimah, and S. Sitorus, “Implementasi Penjadwalan Pakan Ikan Air Tawar Otomatis Pada Kolam Menggunakan RTC,” vol. 3, pp. 139–149, 2024.

[14] M. Z. A. Aristejo, Dimas Waluyo Sejati, “SISTEM INFORMASI ABSENSI BERBASIS WEB DAN MOBILE PADA PT. SINERGI PRIMA TANGGUH,” *J. Sist. Dan Teknol. Inf.*, vol. 07, no. 2, pp. 27–39, 2025.

[15] Ilham Firman Maulana, “Penerapan Firebase Realtime Database pada Aplikasi E-Tilang Smartphone berbasis Mobile Android,” *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 5, pp. 854–863, 2020, doi: 10.29207/resti.v4i5.2232.

[16] A. Widia, “ALAT PENDETEKSI LAMA WAKTU DAN POSISI DUDUK BERBASIS MIKROKONTROLER.” Accessed: Jul. 15, 2025. [Online]. Available: http://scholar.unand.ac.id/78998/

[17] T. Hc-sr, “Non-blocking Ultrasonic Sensor for Arduino Introduction : Non-blocking Ultrasonic Sensor for Arduino Step 1 : Hardware,” no. m, pp. 2–4.

LAMPIRAN 1 DAFTAR RIWAYAT HIDUP

1. Data Pribadi

Nama Lengkap	: Rizky Daffa Pratama
Tempat, Tanggal Lahir	: Pangkalpinang, 11 Desember 2003
Alamat Rumah	: Jl. Usman Ambon,Pangkalpinang
No. HP	: 088269554150
Email	: daffapratama111203@gmail.com
Jenis Kelamin	: Laki-laki
Agama	: Islam

2. Riwayat Pendidikan

1. SD Negeri 1 Belinyu	2010 - 2013
2. SD Negeri 3 Pangkalpinang	2013 - 2016
3. SMP Negeri 2 Pangkalpinang	2016 - 2019
4. SMK Negeri 2 Pangkalpinang	2019 - 2022
5. Politeknik Manufaktur Negeri Bangka Belitung	2022 – Sekarang

3. Pendidikan Non – Formal

-

Sungailiat, 07 Juli 2025

Rizky Daffa Pratama

LAMPIRAN 1 DAFTAR RIWAYAT HIDUP

1. Data Pribadi

Nama Lengkap	: Tio Pratama
Tempat, Tanggal Lahir	: Toboali, 15 Maret 2004
Alamat Rumah	: J. Al Barokah 2 gg. Perumahan, Toboali
No. HP	: 087772476636
Email	: tio22879@gmail.com
Jenis Kelamin	: Laki-laki
Agama	: Islam

2. Riwayat Pendidikan

1. SD Negeri 1 Toboali	2010 - 2016
2. SMP Negeri 2 Toboali	2016 - 2019
3. SMK Negeri 1 Toboali	2019 - 2022
4. Politeknik Manufaktur Negeri Bangka Belitung	2022 – Sekarang

3. Pendidikan Non – Formal

-

Sungailiat, 07 Juli 2025

Tio Pratama

LAMPIRAN 2 PROGRAM

❖ PROGRAM ESP32

||--||

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

// --- KREDENSIAL FIREBASE ---
// Ganti dengan API Key Firebase dan URL Database Anda
#define API_KEY "AIzaSyCSS16XEgn6qhUpCsoCh4yfeBsIiu9KboE"
#define DATABASE_URL "https://fixmonitoring-16ade-default-
rtdb.firebaseio.com"

// --- PIN HARDWARE ---
#define TRIG_PIN 5   // Pin Trigger Sensor Ultrasonik
#define ECHO_PIN 18  // Pin Echo Sensor Ultrasonik
#define BUZZER_PIN 23 // Pin Buzzer
#define FLEX_PIN 34  // Pin Flex Sensor (Analog)
#define TCAADDR 0x70 // Alamat I2C TCA9548A I2C Multiplexer

// --- OBJEK GLOBAL ---
LiquidCrystal_I2C lcd(0x27, 16, 2); // Inisialisasi LCD I2C dengan alamat 0x27,
16 kolom, 2 baris
RTC_DS3231 rtc;                  // Inisialisasi objek RTC

FirebaseData fbdo;   // Objek untuk data Firebase
FirebaseAuth auth;   // Objek untuk otentikasi Firebase
FirebaseConfig config; // Objek untuk konfigurasi Firebase
bool signupOK = false; // Status pendaftaran Firebase

// --- VARIABEL KONTROL TIMER & STATUS ---
unsigned long countdownTime = 0;      // Waktu hitung mundur (detik)
unsigned long previousMillis = 0;     // Millis terakhir untuk update timer (untuk
non-blocking delay)
bool timerStarted = false;           // Flag apakah timer sudah dimulai
bool timerExpired = false;           // Flag apakah timer sudah habis
bool buzzerState = false;            // Status buzzer (ON/OFF)
```

```
unsigned long buzzerMillis = 0;       // Millis terakhir untuk mengontrol kedipan buzzer

// --- VARIABEL SENSOR & POSTUR ---
const int thresholdHigh = 1500;       // Ambang batas atas nilai flex sensor untuk "NORMAL"
const int thresholdLow = 900;         // Ambang batas bawah nilai flex sensor untuk "BUNGKUK"
bool isBungkuk = false;               // Status postur (true jika bungkuk)
int bungkukCounter = 0;               // Penghitung berapa kali postur bungkuk terdeteksi

// --- VARIABEL UNTUK MENGHINDARI UPDATE LCD BERULANG ---
String lastStatus = "";               // Menyimpan status postur terakhir di LCD
String lastTimeDisplay = "";          // Menyimpan tampilan waktu terakhir di LCD
int lastSetTimerValue = 0;            // Menyimpan nilai timer terakhir yang disetting dari Firebase (diinisialisasi ke 0)

// --- VARIABEL SESI & LOGGING ---
DateTime sessionStart;                // Waktu mulai sesi (dari RTC)
DateTime sessionEnd;                  // Waktu selesai sesi (dari RTC)
unsigned long noPersonDuration = 0;   // Durasi tidak ada orang (milidetik)
unsigned long lastNoPersonMillis = 0; // Millis terakhir saat tidak ada orang terdeteksi
bool personDetectedLast = true;       // Status deteksi orang di iterasi sebelumnya
unsigned long sittingDuration = 0;    // Durasi waktu duduk (milidetik)

// --- VARIABEL KONTROL TAMPILAN & PEMBERITAHUAN ---
bool lcdBlinkState = true;            // Status kedipan LCD (untuk "WAKTU HABIS")
unsigned long lastLcdBlink = 0;       // Millis terakhir untuk kedipan LCD
unsigned long oneHourBuzzerStart = 0; // Millis saat buzzer peringatan 1 jam mulai
bool oneHourBuzzing = false;          // Flag apakah buzzer peringatan 1 jam sedang aktif
int hourIntervalNotified = 0;         // Jumlah interval jam yang sudah diberitahukan

// --- FUNGSI TCA9548A MULTIPLEXER ---
// Memilih saluran I2C pada multiplexer
void tcaselect(uint8_t i) {
  if (i > 7) return; // Saluran hanya dari 0 sampai 7
  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i); // Mengirim bitmask untuk memilih saluran
  Wire.endTransmission();
}
```

```
// --- FUNGSI BACA SENSOR ULTRASONIK ---
// Mengukur jarak menggunakan sensor ultrasonik
unsigned int getDistanceCM() {
  // Mengirim pulsa trigger
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Mengukur durasi pulsa echo
  // Timeout 30000 microdetik (sekitar 5 meter) untuk menghindari hang
  unsigned long duration = pulseIn(ECHO_PIN, HIGH, 30000);
  // Menghitung jarak dalam cm (kecepatan suara * waktu / 2)
  return duration * 0.034 / 2;
}

// --- FUNGSI LOGGING SESI KE FIREBASE ---
// Mencatat riwayat sesi duduk ke Firebase Realtime Database
void logSessionHistory() {
  tcaselect(1); // Pilih saluran I2C tempat RTC terhubung
  sessionEnd = rtc.now(); // Ambil waktu selesai sesi

  // Hitung durasi total sesi
  TimeSpan duration = sessionEnd - sessionStart;

  // Format tanggal dan waktu untuk Firebase
  char dateStr[11]; //YYYY-MM-DD
  sprintf(dateStr, "%04d-%02d-%02d", sessionStart.year(), sessionStart.month(),
sessionStart.day());

  char startStr[9]; // HH:MM:SS
  sprintf(startStr, "%02d:%02d:%02d", sessionStart.hour(), sessionStart.minute(),
sessionStart.second());

  char endStr[9]; // HH:MM:SS
  sprintf(endStr, "%02d:%02d:%02d", sessionEnd.hour(), sessionEnd.minute(),
sessionEnd.second());

  // Path dasar untuk menyimpan data riwayat sesi
  // Contoh: /history/2023-10-27/14:30:00
  String basePath = "/history/" + String(dateStr) + "/" + String(startStr);

  // Kirim data ke Firebase
  Firebase.RTDB.setString(&fbdo, basePath + "/waktu_mulai", startStr);
  Firebase.RTDB.setString(&fbdo, basePath + "/waktu_selesai", endStr);
```

```
  Firebase.RTDB.setInt(&fbdo, basePath + "/jumlah_bungkuk", bungkukCounter);
  // Durasi dalam detik
  Firebase.RTDB.setInt(&fbdo, basePath + "/durasi_tidak_ada_orang",
noPersonDuration / 1000);
  Firebase.RTDB.setString(&fbdo, basePath + "/status_akhir", isBungkuk ?
"BUNGKUK" : "NORMAL");
  // Durasi dalam detik
  Firebase.RTDB.setInt(&fbdo, basePath + "/lama_waktu_duduk", sittingDuration
/ 1000);
}

// --- FUNGSI SETUP (Berjalan sekali saat ESP32 pertama kali dinyalakan) ---
void setup() {
  Serial.begin(115200); // Inisialisasi Serial Monitor

  Wire.begin(); // Memulai komunikasi I2C

  // Koneksi WiFi dengan WiFiManager
  Serial.println("Menghubungkan ke WiFi...");
  WiFiManager wm;
  // Ini akan membuat AP dengan nama "MonitoringPostur" dan password
"12345678" jika belum terkoneksi
  bool res = wm.autoConnect("MonitoringPostur", "12345678");
  if (!res) {
    Serial.println("Gagal konek WiFi. Rebooting...");
    delay(3000);
    ESP.restart(); // Restart ESP jika gagal koneksi
  }
  Serial.println("WiFi Terhubung!");

  // Pastikan semua variabel timer dan status direset ke 0/false setelah koneksi
WiFi berhasil.
  // Ini memastikan sistem dalam keadaan bersih sebelum membaca konfigurasi
dari Firebase.
  countdownTime = 0;
  previousMillis = 0; // Penting untuk mereset ini juga
  timerStarted = false;
  timerExpired = false;
  buzzerState = false;
  buzzerMillis = 0;
  isBungkuk = false;
  bungkukCounter = 0;
  lastStatus = "";
  lastTimeDisplay = "";
  lastSetTimerValue = 0; // Sangat penting: Set ke 0 agar perubahan dari Firebase
(misal dari 0 ke X) terdeteksi.
```

```
  sessionStart = DateTime(2000, 1, 1, 0, 0, 0); // Inisialisasi dummy
  sessionEnd = DateTime(2000, 1, 1, 0, 0, 0); // Inisialisasi dummy
  noPersonDuration = 0;
  lastNoPersonMillis = 0;
  personDetectedLast = true;
  sittingDuration = 0;
  lcdBlinkState = true;
  lastLcdBlink = 0;
  oneHourBuzzerStart = 0;
  oneHourBuzzing = false;
  hourIntervalNotified = 0;

  digitalWrite(BUZZER_PIN, LOW); // Pastikan buzzer mati

  tcaselect(0); // Pilih saluran 0 untuk LCD
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Waiting timer..."); // Tampilan awal LCD setelah WiFi terkoneksi
  lcd.setCursor(0, 1);
  lcd.print("00:00:00        "); // Tampilan waktu default awal

  // Inisialisasi RTC (melalui TCA9548A - Saluran 1)
  tcaselect(1); // Pilih saluran 1 untuk RTC
  if (!rtc.begin()) {
    Serial.println("RTC tidak ditemukan atau gagal inisialisasi!");
    Serial.println("Pastikan RTC terhubung dengan benar dan saluran I2C sudah
benar (saluran 1).");
    while (1); // Hentikan program jika RTC tidak ditemukan
  }
  // rtc.adjust(DateTime(F(_DATE), F(TIME_))); // UNCOMMENT BARIS INI
HANYA SAAT PERTAMA KALI UNTUK SET WAKTU RTC DARI WAKTU
KOMPILASI, LALU COMMENT KEMBALI

  // Inisialisasi Pin Mode
  pinMode(FLEX_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW); // Pastikan buzzer mati di awal

  // Konfigurasi Firebase
  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;
  config.token_status_callback = tokenStatusCallback; // Callback untuk status
```

```
token Firebase

  // Coba daftar secara anonim ke Firebase
  if (Firebase.signUp(&config, &auth, "", "")) {
    signupOK = true;
    Serial.println("Firebase SignUp OK");
  } else {
    Serial.print("Firebase SignUp Gagal: ");
    Serial.println(config.signer.signupError.message.c_str());
  }

  Firebase.begin(&config, &auth); // Mulai koneksi Firebase
  Firebase.reconnectWiFi(true);   // Aktifkan reconnect otomatis jika WiFi putus

  // Setelah Firebase siap dan terhubung, set nilai awal Firebase di
/Data/matikanBuzzer
  // ini agar konsisten dengan state awal "buzzer mati" di ESP32.
  delay(1000); // Beri sedikit waktu untuk Firebase sinkronisasi awal
  if (Firebase.ready() && signupOK) {
    // Pastikan Firebase matikanBuzzer di-reset ke false saat booting
    Firebase.RTDB.setBool(&fbdo, "/Data/matikanBuzzer", false);
    // Pastikan juga status bungkuk di Firebase direset ke false
    Firebase.RTDB.setBool(&fbdo, "/Data/bungkuk", false);
    // Atur /setTimer di Firebase ke 0, ini akan memicu logikamu di loop() untuk
mengatur timerStarted = false
    Firebase.RTDB.setInt(&fbdo, "/setTimer", 0);
  }
}

// --- FUNGSI LOOP (Berjalan berulang-ulang tanpa henti) ---
void loop() {
  static unsigned long lastFirebaseCheck = 0; // Waktu terakhir cek Firebase
  static bool matikanBuzzerFirebaseStatus = false; // Status lokal untuk
matikanBuzzer dari Firebase

  // --- PEMBACAAN DATA DARI FIREBASE (setTimer & matikanBuzzer) ---
  // Cek Firebase setiap 2 detik untuk menghindari overloading request dan
memastikan responsif
  if (Firebase.ready() && signupOK && millis() - lastFirebaseCheck > 2000) {
    lastFirebaseCheck = millis();

    // Dapatkan nilai timer dari Firebase
    if (Firebase.RTDB.getInt(&fbdo, "/setTimer")) {
      int newTimer = fbdo.intData();
      // Serial.print("Nilai setTimer dari Firebase: "); // Debugging
      // Serial.println(newTimer); // Debugging
```

```
// Jika nilai timer berubah (baik dari >0 ke X, atau dari X ke 0)
if (newTimer != lastSetTimerValue) {
  lastSetTimerValue = newTimer; // Simpan nilai timer yang baru diterima

  if (newTimer > 0) {
    // Logika untuk MEMULAI TIMER (Jika nilai > 0)
    countdownTime = newTimer;
    timerStarted = true;
    timerExpired = false;
    bungkukCounter = 0;
    noPersonDuration = 0;
    sittingDuration = 0;
    lastNoPersonMillis = millis();
    personDetectedLast = true;
    oneHourBuzzing = false;
    hourIntervalNotified = 0;
    digitalWrite(BUZZER_PIN, LOW); // Pastikan buzzer mati saat mulai
timer baru

    tcaselect(1); // Pilih saluran RTC
    sessionStart = rtc.now(); // Catat waktu mulai sesi

    // Perbarui tampilan LCD untuk memulai timer
    tcaselect(0); // Pilih saluran LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Timer dimulai..."); // Pesan saat timer baru dimulai
    lastStatus = ""; // Reset status LCD untuk pembaruan paksa
    lastTimeDisplay = ""; // Reset waktu LCD untuk pembaruan paksa
  } else { // newTimer == 0
    // Logika untuk MENGHENTIKAN TIMER (Jika nilai 0)
    countdownTime = 0; // Pastikan waktu hitung mundur direset ke 0
    timerStarted = false;
    timerExpired = false; // Ini perlu direset agar tidak memicu buzzer terus
    digitalWrite(BUZZER_PIN, LOW); // Pastikan buzzer mati
    buzzerState = false;

    // Reset tampilan LCD ke kondisi awal "Waiting timer..."
    tcaselect(0); // Pilih saluran LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Waiting timer...");
    lcd.setCursor(0, 1);
    lcd.print("00:00:00        "); // Pastikan ada spasi yang cukup
    lastStatus = "";    // Reset status LCD
```

```
        lastTimeDisplay = "";// Reset waktu LCD
        // Tidak perlu log sesi jika dihentikan manual dengan 0 (sesi belum tentu
selesai normal)
      }
    }
  }

  // Baca status matikanBuzzer dari Firebase setiap kali loop berjalan
  if (Firebase.RTDB.getBool(&fbdo, "/Data/matikanBuzzer")) {
    matikanBuzzerFirebaseStatus = fbdo.boolData();
  }
 }

 // --- FUNGSI MATIKAN BUZZER & RESET DARI FIREBASE (prioritas
tertinggi) ---
 // Jika ada perintah 'matikanBuzzer' dari Firebase, Lakukan reset penuh
 if (matikanBuzzerFirebaseStatus) { // Gunakan status lokal yang baru dibaca
  Serial.println("Perintah matikanBuzzer diterima!"); // Debugging

  // Set kembali '/Data/matikanBuzzer' ke false di Firebase SEGERA
  // Ini penting agar kondisi di atas tidak terus menerus melihat TRUE
  Firebase.RTDB.setBool(&fbdo, "/Data/matikanBuzzer", false);
  matikanBuzzerFirebaseStatus = false; // Reset status lokal juga

  digitalWrite(BUZZER_PIN, LOW); // Matikan buzzer
  buzzerState = false;          // Reset status buzzer

  // START: Reset variabel timer dan status lainnya
  countdownTime = 0;            // Penting: Reset countdownTime ke 0
  previousMillis = millis();    // Reset previousMillis agar tidak ada lompatan
waktu
  timerExpired = false;         // Reset flag timer habis
  timerStarted = false;         // Reset flag timer dimulai
  lastSetTimerValue = 0;        // Penting: Pastikan ini 0 agar bisa start timer lagi
  bungkukCounter = 0;           // Reset counter bungkuk
  noPersonDuration = 0;         // Reset durasi tanpa orang
  sittingDuration = 0;          // Reset durasi duduk
  lastNoPersonMillis = 0;       // Reset millis untuk durasi tanpa orang
  personDetectedLast = true;    // Reset status deteksi orang
  oneHourBuzzing = false;       // Matikan peringatan 1 jam
  hourIntervalNotified = 0;     // Reset interval jam
  isBungkuk = false; // Pastikan status bungkuk juga di-reset
  if (Firebase.ready() && signupOK) {
   Firebase.RTDB.setBool(&fbdo, "/Data/bungkuk", false); // Pastikan status di
Firebase juga normal
  }
```

```
    // END: Reset variabel timer dan status lainnya

    // Reset tampilan LCD ke kondisi awal "Waiting timer..."
    tcaselect(0); // Pilih saluran LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Waiting timer...");
    lcd.setCursor(0, 1);
    lcd.print("00:00:00        ");
    lastStatus = "";         // Reset status LCD
    lastTimeDisplay = ""; // Reset waktu LCD
  }


  // --- LOGIKA TIMER UTAMA & DETEKSI ORANG (Hanya berjalan jika
timerStarted dan belum timerExpired) ---
  if (timerStarted && !timerExpired) {
    int distance = getDistanceCM(); // Baca jarak dari sensor ultrasonik
    // Tentukan apakah ada orang (jarak antara 1cm sampai 30cm)
    bool personDetected = (distance > 0 && distance < 40);

    // LOGIKA DURASI TIDAK ADA ORANG
    if (!personDetected) {
      // Tambahkan waktu berlalu sejak terakhir kali tidak ada orang terdeteksi
      if (personDetectedLast || lastNoPersonMillis == 0) {
        lastNoPersonMillis = millis(); // Catat waktu mulai tidak ada orang
        personDetectedLast = false;    // Update flag
      } else {
        noPersonDuration += (millis() - lastNoPersonMillis);
        lastNoPersonMillis = millis(); // Perbarui waktu terakhir tidak ada orang
      }

      // Konversi dan kirim durasi tidak ada orang ke Firebase
      int jamKosong = noPersonDuration / 3600000;
      int menitKosong = (noPersonDuration % 3600000) / 60000;
      int detikKosong = (noPersonDuration % 60000) / 1000;
      char kosongStr[16];
      sprintf(kosongStr, "%02d:%02d:%02d", jamKosong, menitKosong,
detikKosong);
      if (Firebase.ready() && signupOK) {
        Firebase.RTDB.setString(&fbdo, "/Data/tidak_ada_orang_str", kosongStr);
      }
    } else {
      personDetectedLast = true; // Update flag jika ada orang terdeteksi
      lastNoPersonMillis = 0; // Reset lastNoPersonMillis saat ada orang
    }
```

```
  // LOGIKA PENGHITUNGAN WAKTU DUDUK & HITUNG MUNDUR
TIMER
  // Hanya hitung mundur dan waktu duduk jika ada orang dan sudah 1 detik
berlalu
  if (personDetected && millis() - previousMillis >= 1000) {
    previousMillis = millis(); // Perbarui waktu terakhir update
    if (countdownTime > 0) {
      countdownTime--;            // Kurangi waktu hitung mundur
      sittingDuration += 1000; // Tambah durasi duduk

      // Konversi dan kirim durasi duduk ke Firebase
      int jamDuduk = sittingDuration / 3600000;
      int menitDuduk = (sittingDuration % 3600000) / 60000;
      int detikDuduk = (sittingDuration % 60000) / 1000;
      char dudukStr[16];
      sprintf(dudukStr, "%02d:%02d:%02d", jamDuduk, menitDuduk,
detikDuduk);
      if (Firebase.ready() && signupOK) {
        Firebase.RTDB.setString(&fbdo, "/Data/waktu_duduk_str", dudukStr);
      }

      // Peringatan setiap 1 jam duduk
      if ((sittingDuration / 3600000) > hourIntervalNotified && countdownTime >
0) {
        hourIntervalNotified++; // Tambah counter interval jam
        tcaselect(0);          // Pilih saluran LCD
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("1 jam berlalu");
        lcd.setCursor(0, 1);
        lcd.print("Jaga Postur!");
        digitalWrite(BUZZER_PIN, HIGH); // Nyalakan buzzer
        oneHourBuzzing = true;         // Aktifkan flag buzzing 1 jam
        oneHourBuzzerStart = millis();  // Catat waktu mulai buzzing
      }
    } else {
      // TIMER HABIS
      timerExpired = true;          // Set flag timer habis
      buzzerMillis = millis();      // Mulai hitung waktu untuk kedipan buzzer
      lastLcdBlink = millis();      // Mulai hitung waktu untuk kedipan LCD
      logSessionHistory();          // Catat riwayat sesi ke Firebase
      // Pastikan buzzer menyala saat timer habis jika belum aktif
      digitalWrite(BUZZER_PIN, HIGH);
      // Tampilkan 00:00:00 secara permanen di LCD setelah timer habis
      tcaselect(0);
```

```
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("WAKTU HABIS!");
        lcd.setCursor(0,1);
        lcd.print("00:00:00        "); // Pastikan spasi cukup untuk menimpa
        lcdBlinkState = true; // Set ke true agar teks "WAKTU HABIS!" terlihat
      }
    }
  }

  // --- PENGENDALIAN TAMPILAN LCD ---
  if (!timerExpired) { // Jika timer belum habis
    // Tampilkan status postur (NORMAL/BUNGKUK)
    String currentStatus;
    if (timerStarted) { // Hanya tampilkan status bungkuk/normal jika timer
berjalan
      currentStatus = isBungkuk ? "Kondisi: BUNGKUK " : "Kondisi: NORMAL
";
    } else { // Jika timer belum dimulai, tetap tampilkan "Waiting timer..."
      currentStatus = "Waiting timer...";
    }

    if (currentStatus != lastStatus) { // Hanya update jika status berubah
      tcaselect(0); // Pilih saluran LCD
      lcd.setCursor(0, 0);
      lcd.print(currentStatus);
      lastStatus = currentStatus;
    }

    // Tampilkan sisa waktu timer
    int hours = countdownTime / 3600;
    int minutes = (countdownTime % 3600) / 60;
    int seconds = countdownTime % 60;
    char buf[17];
    sprintf(buf, "%02d:%02d:%02d        ", hours, minutes, seconds); // Format
HH:MM:SS dengan spasi padding
    String timeStr = String(buf);

    if (timeStr != lastTimeDisplay) { // Hanya update jika waktu berubah
      tcaselect(0); // Pilih saluran LCD
      lcd.setCursor(0, 1);
      lcd.print(timeStr);
      lastTimeDisplay = timeStr;
    }
  } else { // Jika timer sudah habis, tampilkan 00:00:00 dan "WAKTU HABIS!"
secara statis
```

```
    // Memastikan LCD menunjukkan "WAKTU HABIS!" dan "00:00:00" secara
statis
    tcaselect(0);
    lcd.setCursor(0, 0);
    lcd.print("WAKTU HABIS!");
    lcd.setCursor(0, 1);
    lcd.print("00:00:00        ");
  }

  // --- PENGENDALIAN BUZZER KETIKA TIMER HABIS ---
  // Buzzer tetap berkedip HANYA jika timerExpired dan belum ada perintah
matikanBuzzer dari Firebase
  if (timerExpired && !matikanBuzzerFirebaseStatus) {
    if (millis() - buzzerMillis >= 500) { // Kedip buzzer setiap 500ms
      buzzerMillis = millis();
      buzzerState = !buzzerState; // Toggle state buzzer
      digitalWrite(BUZZER_PIN, buzzerState); // Nyalakan/Matikan buzzer
    }
  } else if (!oneHourBuzzing && !isBungkuk) { // Matikan buzzer jika tidak ada
kondisi yang memerlukannya ON
    digitalWrite(BUZZER_PIN, LOW);
    buzzerState = false; // Reset status buzzer internal
  }


  // --- MENGHENTIKAN BUZZER PERINGATAN 1 JAM ---
  if (oneHourBuzzing && millis() - oneHourBuzzerStart >= 3000) { // Buzzer
berbunyi 3 detik
    digitalWrite(BUZZER_PIN, LOW); // Matikan buzzer
    oneHourBuzzing = false;          // Set flag menjadi false

    // Kembalikan tampilan LCD ke status normal setelah peringatan 1 jam
    tcaselect(0); // Pilih saluran LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    // Tentukan pesan berdasarkan apakah timer masih berjalan
    if (timerStarted && !timerExpired) {
      lcd.print(isBungkuk ? "Kondisi: BUNGKUK" : "Kondisi: NORMAL");
    } else {
      lcd.print("Waiting timer...");
    }
    char buf[17];
    int hours = countdownTime / 3600;
    int minutes = (countdownTime % 3600) / 60;
    int seconds = countdownTime % 60;
    sprintf(buf, "%02d:%02d:%02d            ", hours, minutes, seconds);
```

```
  lcd.setCursor(0, 1);
  lcd.print(buf);
 }

 // --- PENGIRIMAN DATA SENSOR KE FIREBASE ---
 // Sensor dan pengiriman data hanya aktif jika timerStarted dan belum
timerExpired
 if (Firebase.ready() && signupOK && timerStarted && !timerExpired) {
  int distance = getDistanceCM();
  Firebase.RTDB.setInt(&fbdo, "/Data/jarak", distance); // Kirim nilai jarak

  int flexValue = analogRead(FLEX_PIN); // Baca nilai analog dari flex sensor
  Firebase.RTDB.setInt(&fbdo, "/Data/flexValue", flexValue); // Kirim nilai
mentah ke Firebase

  float derajatFlex;
  if (flexValue >= thresholdHigh) { // Jika nilai flex sensor melebihi ambang
batas atas
    derajatFlex = 0.0; // Set derajatFlex menjadi 0
  } else {
    derajatFlex = -0.0685 * flexValue + 114.94; // Gunakan rumus normal
  }
  Firebase.RTDB.setFloat(&fbdo, "/Data/derajatFlex", derajatFlex); // Kirim nilai
derajat ke Firebase

  // Deteksi postur bungkuk
  if (!isBungkuk && flexValue <= thresholdLow) { // Jika belum bungkuk dan
nilai di bawah ambang batas rendah
    isBungkuk = true;            // Set status bungkuk
    bungkukCounter++;            // Tambah counter bungkuk
    Firebase.RTDB.setBool(&fbdo, "/Data/bungkuk", true); // Kirim status
bungkuk ke Firebase
    digitalWrite(BUZZER_PIN, HIGH); // Nyalakan buzzer
  }
  // Deteksi postur normal kembali
  else if (isBungkuk && flexValue >= thresholdHigh) { // Jika sedang bungkuk
dan nilai di atas ambang batas tinggi
    isBungkuk = false;           // Set status normal
    Firebase.RTDB.setBool(&fbdo, "/Data/bungkuk", false); // Kirim status
normal ke Firebase
    // Matikan buzzer hanya jika bukan karena timer habis atau peringatan 1 jam
    if (!timerExpired && !oneHourBuzzing) {
     digitalWrite(BUZZER_PIN, LOW);
    }
  }
```

```
    // --- PENGENDALIAN BUZZER KARENA BUNGKUK ---
    if (isBungkuk) {
      digitalWrite(BUZZER_PIN, HIGH);
    }
    // Jika tidak bungkuk dan tidak ada peringatan 1 jam atau timer habis
    else if (!timerExpired && !oneHourBuzzing) {
      digitalWrite(BUZZER_PIN, LOW);
    }
  } else { // Jika timer belum dimulai atau sudah expired, pastikan buzzer mati dan status bungkuk reset
      digitalWrite(BUZZER_PIN, LOW);
      isBungkuk = false; // Pastikan status bungkuk juga di-reset
      if (Firebase.ready() && signupOK) {
        Firebase.RTDB.setBool(&fbdo, "/Data/bungkuk", false); // Pastikan status di Firebase juga normal
      }
      // Pastikan nilai sensor di Firebase juga direset atau di-stop update
      if (Firebase.ready() && signupOK) {
        Firebase.RTDB.setInt(&fbdo, "/Data/jarak", 0);
        Firebase.RTDB.setInt(&fbdo, "/Data/flexValue", 0);
        Firebase.RTDB.setFloat(&fbdo, "/Data/derajatFlex", 0.0);
        Firebase.RTDB.setString(&fbdo, "/Data/tidak_ada_orang_str", "00:00:00");
        Firebase.RTDB.setString(&fbdo, "/Data/waktu_duduk_str", "00:00:00");
      }
  }
}
```

- **Program Web (Visual Studio Code)**

||----------||

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <title>Monitoring Duduk & Postur</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        :root {
            --primary-color: #1b5e20;
            --secondary-color: #388e3c;
            --light-green: #c8e6c9;
            --bg-color: #e8f5e9;
            --text-color: #2e7d32;
            --white: #ffffff;
            --shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
        }
```

```
body {
    font-family: 'Segoe UI', sans-serif;
    background: var(--bg-color);
    margin: 0;
    padding: 10px;
    color: var(--text-color);
}

.container {
    max-width: 1000px;
    margin: auto;
    background: var(--white);
    padding: 25px;
    border-radius: 20px;
    box-shadow: var(--shadow);
}

h1, h2 {
    color: var(--primary-color);
    margin-top: 25px;
}

h1 {
    text-align: center;
    font-size: 2rem;
    border-bottom: 2px solid var(--secondary-color);
    padding-bottom: 10px;
}

.data-box {
    background: var(--light-green);
    border-left: 6px solid var(--secondary-color);
    border-radius: 10px;
    padding: 20px;
    margin: 15px 0;
    display: flex;
    justify-content: space-between;
}

.data-box span {
    font-weight: bold;
    color: var(--primary-color);
}

input, button {
```

```
    padding: 10px 15px;
    border-radius: 8px;
    border: 1px solid #ccc;
    font-size: 1rem;
    margin: 5px;
}

button {
    background: var(--secondary-color);
    color: white;
    font-weight: bold;
    cursor: pointer;
}

button:hover {
    background: var(--primary-color);
}

.timer-container {
    display: flex;
    flex-wrap: wrap;
    align-items: center;
    gap: 10px;
    margin-bottom: 10px;
}

.riwayat-box {
    background: var(--light-green);
    border-radius: 10px;
    padding: 20px;
    display: grid;
    gap: 15px;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

th, td {
    padding: 10px;
    border-bottom: 1px solid #ccc;
    text-align: center;
}
```

```
        th {
            background: #a5d6a7;
            color: var(--primary-color);
        }

        .time {
            margin-top: 15px;
            text-align: center;
        }

        /* Gaya untuk menyembunyikan elemen */
        .hidden-until-active {
            display: none;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Monitoring Duduk & Postur</h1>

        <div id="activeMonitoringSection" class="hidden-until-active">
            <div class="data-box"><div>Status Postur:</div><span
id="bungkukStatus">–</span></div>
            <div class="data-box"><div>Derajat Tekukan:</div><span
id="derajatFlex">–</span></div>
            <div class="data-box"><div>Jarak Terbaca:</div><span
id="jarak">Memuat…</span></div>
        </div>

        <button onclick="matikanBuzzer()">Matikan Buzzer</button>

        <hr>
        <h2>Atur Waktu Timer</h2>
        <div class="timer-container">
            <input id="jam" type="number" min="0" value="0"> :
            <input id="menit" type="number" min="0" value="1"> :
            <input id="detik" type="number" min="0" value="0">
        </div>
        <button onclick="kirimTimer()">Kirim Timer</button>

        <div class="time">
            <span id="clock">--:--:--</span><br>
            <span id="date">-- -- ----</span>
        </div>
```

```
      <hr>
      <h2>Filter Tanggal Riwayat</h2>
      <input type="date" id="filterTanggal">
      <button onclick="filterTanggal()">Tampilkan</button>
      <button onclick="downloadJSON()">Download JSON</button>

      <h2>Riwayat Duduk Terakhir</h2>
      <div class="riwayat-box" id="riwayatTerakhir">
        <div><strong>Mulai</strong><br><span id="riwayatMulai">–
</span></div>
        <div><strong>Selesai</strong><br><span id="riwayatSelesai">–
</span></div>
        <div><strong>Durasi Duduk</strong><br><span
id="riwayatDurasiDuduk">–</span></div>
        <div><strong>Jumlah Bungkuk</strong><br><span
id="riwayatBungkuk">–</span></div>
        <div><strong>Durasi Tanpa Orang</strong><br><span
id="riwayatTidakAdaOrang">–</span></div>
        <div><strong>Status Akhir</strong><br><span id="riwayatStatus">–
</span></div>
      </div>

      <h2>Seluruh Riwayat Duduk</h2>
      <div style="overflow-x: auto;">
        <table>
          <thead>
            <tr>
              <th>Tanggal</th>
              <th>Jam Mulai</th>
              <th>Jam Selesai</th>
              <th>Lama Duduk</th>
              <th>Bungkuk</th>
              <th>Tidak Ada Orang</th>
              <th>Status Akhir</th>
            </tr>
          </thead>
          <tbody id="tabelRiwayat">
            <tr><td colspan="7">Memuat data…</td></tr>
          </tbody>
        </table>
      </div>
    </div>

    <script>
      const urlBase = "https://fixmonitoring-16ade-default-rtdb.firebaseio.com";
      const activeMonitoringSection =
```

```
document.getElementById("activeMonitoringSection");

        // Fungsi untuk format detik menjadi menit dan detik (misal: 1m 30s)
        function formatDetik(detik) {
            const m = Math.floor(detik / 60);
            const s = detik % 60;
            return ${m}m ${s}s;
        }

        // Fungsi untuk mengupdate jam dan tanggal real-time di antarmuka
        function updateClock() {
            const now = new Date();
            document.getElementById("clock").innerText =
now.toLocaleTimeString(); // Tampilkan jam
            document.getElementById("date").innerText =
now.toLocaleDateString("id-ID", { // Tampilkan tanggal
                weekday: "long", year: "numeric", month: "long", day: "numeric"
            });
        }
        setInterval(updateClock, 1000); // Update setiap 1 detik
        updateClock(); // Panggil pertama kali agar langsung tampil

        // Fungsi utama untuk mengambil dan menampilkan data monitoring
        async function updateData() {
            try {
                // Ambil data status timer, jarak, bungkuk, dan derajat flex dari Firebase
secara paralel
                const [setTimerData, jarakData, bungkukData, derajatData] = await
Promise.all([
                    fetch(${urlBase}/setTimer.json).then(r => r.json()),
                    fetch(${urlBase}/Data/jarak.json).then(r => r.json()),
                    fetch(${urlBase}/Data/bungkuk.json).then(r => r.json()),
                    fetch(${urlBase}/Data/derajatFlex.json).then(r => r.json())
                ]);

                // Tentukan apakah timer sedang aktif atau belum diset
                // Timer dianggap aktif jika nilai setTimer > 0 di Firebase (sesuai logika
ESP32)
                const isTimerActive = (setTimerData !== null && setTimerData > 0);

                if (isTimerActive) {
                    // Jika timer aktif, tampilkan bagian monitoring aktif
                    activeMonitoringSection.classList.remove("hidden-until-active");

                    // Update nilai-nilai sensor
                    document.getElementById("jarak").innerText = jarakData + " cm";
```

```
                document.getElementById("bungkukStatus").innerText =
bungkukData ? "BUNGKUK" : "NORMAL";
                document.getElementById("derajatFlex").innerText = derajatData
!== undefined ? derajatData.toFixed(1) + "°" : "–";
            } else {
                // Jika timer tidak aktif, sembunyikan bagian monitoring aktif
                activeMonitoringSection.classList.add("hidden-until-active");
                // Reset tampilan atau beri pesan "menunggu timer"
                document.getElementById("jarak").innerText = "Menunggu timer...";
                document.getElementById("bungkukStatus").innerText =
"Menunggu timer...";
                document.getElementById("derajatFlex").innerText = "Menunggu
timer...";
            }

        } catch (e) {
            console.error("Gagal update data:", e);
            // Tampilkan pesan error jika gagal mengambil data
            document.getElementById("jarak").innerText = "Error memuat!";
            document.getElementById("bungkukStatus").innerText = "Error
memuat!";
            document.getElementById("derajatFlex").innerText = "Error memuat!";
        }
    }

    // Fungsi untuk mengirim nilai timer ke Firebase
    async function kirimTimer() {
        const h = +document.getElementById("jam").value;
        const m = +document.getElementById("menit").value;
        const s = +document.getElementById("detik").value;
        const total = h * 3600 + m * 60 + s; // Total detik

        if (total < 0) return alert("Waktu tidak boleh negatif!");
        // Konfirmasi jika timer diatur ke 0
        if (total === 0 && !confirm("Timer diatur ke 0 detik. Lanjutkan?"))
return;

        try {
            // Kirim nilai total detik ke Firebase di path /setTimer
            await fetch(${urlBase}/setTimer.json, {
                method: "PUT", body: JSON.stringify(total)
            });
            alert(Timer berhasil dikirim: ${total} detik);
            updateData(); // Segera update tampilan setelah mengirim timer
        } catch (e) {
            console.error("Gagal mengirim timer:", e);
```

```
            alert("Gagal mengirim timer. Pastikan koneksi internet stabil.");
        }
    }

    // Fungsi untuk mematikan buzzer melalui Firebase
    async function matikanBuzzer() {
        try {
            // Kirim nilai true ke /Data/matikanBuzzer untuk mematikan buzzer di
ESP32
            await fetch(${urlBase}/Data/matikanBuzzer.json, {
                method: "PUT", body: JSON.stringify(true)
            });
            alert("Perintah matikan buzzer dikirim.");
        } catch (e) {
            console.error("Gagal mematikan buzzer:", e);
            alert("Gagal mematikan buzzer. Pastikan koneksi internet stabil.");
        }
    }

    // Fungsi untuk mengambil dan menampilkan riwayat sesi terakhir
    async function ambilRiwayatTerakhir() {
        try {
            const res = await fetch(${urlBase}/history.json);
            const data = await res.json();
            if (!data) {
                // Jika tidak ada data, tampilkan pesan default
                document.getElementById("riwayatMulai").innerText = "N/A";
                document.getElementById("riwayatSelesai").innerText = "N/A";
                document.getElementById("riwayatDurasiDuduk").innerText =
"N/A";
                document.getElementById("riwayatBungkuk").innerText = "N/A";
                document.getElementById("riwayatTidakAdaOrang").innerText =
"N/A";
                document.getElementById("riwayatStatus").innerText = "N/A";
                return;
            }

            // Mengurutkan tanggal secara descending untuk mendapatkan tanggal
terbaru
            const tanggalTerbaru = Object.keys(data).sort().reverse()[0];
            // Mengurutkan jam secara descending untuk mendapatkan sesi terbaru
pada tanggal tersebut
            const jamTerbaru =
Object.keys(data[tanggalTerbaru]).sort().reverse()[0];
            const riwayatTerakhir = data[tanggalTerbaru][jamTerbaru];
```

```
            // Update elemen HTML dengan data riwayat terakhir
            document.getElementById("riwayatMulai").innerText =
riwayatTerakhir.waktu_mulai || "-";
            document.getElementById("riwayatSelesai").innerText =
riwayatTerakhir.waktu_selesai || "-";
            document.getElementById("riwayatDurasiDuduk").innerText =
formatDetik(riwayatTerakhir.lama_waktu_duduk || 0);
            document.getElementById("riwayatBungkuk").innerText =
riwayatTerakhir.jumlah_bungkuk || 0;
            document.getElementById("riwayatTidakAdaOrang").innerText =
formatDetik(riwayatTerakhir.durasi_tidak_ada_orang || 0);
            document.getElementById("riwayatStatus").innerText =
riwayatTerakhir.status_akhir || "-";

        } catch (e) {
            console.error("Gagal mengambil riwayat terakhir:", e);
            // Tampilkan pesan error jika gagal
            document.getElementById("riwayatMulai").innerText = "Error";
            document.getElementById("riwayatSelesai").innerText = "Error";
            document.getElementById("riwayatDurasiDuduk").innerText = "Error";
            document.getElementById("riwayatBungkuk").innerText = "Error";
            document.getElementById("riwayatTidakAdaOrang").innerText =
"Error";
            document.getElementById("riwayatStatus").innerText = "Error";
        }
    }

    // Fungsi untuk menampilkan seluruh riwayat duduk dalam tabel, dengan
opsi filter tanggal
    async function tampilkanSemuaRiwayat(filterTanggalParam = null) {
        const tbody = document.getElementById("tabelRiwayat");
        tbody.innerHTML = "<tr><td colspan='7'>Memuat data…</td></tr>"; //
Tampilkan loading state

        try {
            const res = await fetch(${urlBase}/history.json);
            const data = await res.json();

            if (!data) {
                return tbody.innerHTML = "<tr><td colspan='7'>Belum ada data
riwayat.</td></tr>";
            }

            let allRecords = [];
            // Iterasi melalui setiap tanggal dan jam untuk mengumpulkan semua
riwayat
```

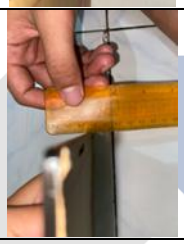
```
            for (const tanggal in data) {
                if (filterTanggalParam && tanggal !== filterTanggalParam) {
                    continue; // Lewati jika ada filter tanggal dan tidak sesuai
                }
                for (const jam in data[tanggal]) {
                    const record = data[tanggal][jam];
                    allRecords.push({
                        tanggal: tanggal,
                        jamMulai: record.waktu_mulai || "-",
                        jamSelesai: record.waktu_selesai || "-",
                        durasiDuduk: record.lama_waktu_duduk || 0,
                        bungkuk: record.jumlah_bungkuk || 0,
                        tidakAdaOrang: record.durasi_tidak_ada_orang || 0,
                        status: record.status_akhir || "-"
                    });
                }
            }

            // Urutkan riwayat dari yang terbaru ke terlama
            allRecords.sort((a, b) => {
                const dateTimeA = new Date(${a.tanggal}T${a.jamMulai});
                const dateTimeB = new Date(${b.tanggal}T${b.jamMulai});
                return dateTimeB - dateTimeA; // Urutkan descending (terbaru di
atas)
            });

            // Bersihkan tabel sebelum mengisi data baru
            tbody.innerHTML = "";
            if (allRecords.length === 0) {
                tbody.innerHTML = "<tr><td colspan='7'>Tidak ada data untuk
tanggal yang dipilih.</td></tr>";
                return;
            }

            // Isi tabel dengan data yang sudah diformat
            allRecords.forEach(record => {
                const tr = document.createElement("tr");
                tr.innerHTML = `
                    <td>${record.tanggal}</td>
                    <td>${record.jamMulai}</td>
                    <td>${record.jamSelesai}</td>
                    <td>${formatDetik(record.durasiDuduk)}</td>
                    <td>${record.bungkuk}</td>
                    <td>${formatDetik(record.tidakAdaOrang)}</td>
                    <td>${record.status}</td>`;
                tbody.appendChild(tr);
```

```
            });

        } catch (e) {
            console.error("Gagal memuat semua riwayat:", e);
            tbody.innerHTML = "<tr><td colspan='7'>Gagal memuat data
riwayat.</td></tr>";
        }
    }

    // Fungsi yang dipanggil saat tombol "Tampilkan" filter tanggal diklik
    function filterTanggal() {
        const selectedDate = document.getElementById("filterTanggal").value;
        if (!selectedDate) {
            alert("Pilih tanggal terlebih dahulu!");
            return;
        }
        tampilkanSemuaRiwayat(selectedDate);
    }

    // Fungsi untuk mengunduh riwayat dalam format JSON
    async function downloadJSON() {
        const selectedDate = document.getElementById("filterTanggal").value;
        if (!selectedDate) {
            alert("Pilih tanggal terlebih dahulu!");
            return;
        }
        try {
            // Ambil data riwayat hanya untuk tanggal yang dipilih
            const res = await fetch(${urlBase}/history/${selectedDate}.json);
            const data = await res.json();

            if (!data || Object.keys(data).length === 0) {
                alert("Tidak ada data untuk tanggal ini.");
                return;
            }

            // Format data menjadi array objek untuk JSON
            const formattedData = Object.keys(data).map(jam => ({ jam,
...data[jam] }));

            const jsonToDownload = {
                tanggal: selectedDate,
                data: formattedData
            };

            // Buat Blob dari data JSON dan unduh
```

```
            const blob = new Blob([JSON.stringify(jsonToDownload, null, 2)],
{type: 'application/json'});
            const url = URL.createObjectURL(blob);
            const a = document.createElement('a');
            a.href = url;
            a.download = riwayat-duduk-${selectedDate}.json;
            document.body.appendChild(a);
            a.click();
            document.body.removeChild(a);
            URL.revokeObjectURL(url); // Bersihkan URL objek
            alert("File JSON berhasil diunduh.");

        } catch (e) {
            console.error("Gagal download JSON:", e);
            alert("Gagal mengunduh file JSON. Pastikan koneksi internet stabil.");
        }
    }

    // Event listener yang berjalan setelah seluruh DOM dimuat
    document.addEventListener('DOMContentLoaded', () => {
        updateData(); // Panggil pertama kali untuk menampilkan data terbaru
        ambilRiwayatTerakhir(); // Panggil untuk menampilkan riwayat terakhir
        tampilkanSemuaRiwayat(); // Panggil untuk menampilkan semua riwayat
        setInterval(updateData, 3000); // Update data setiap 3 detik

        // Set input tanggal filter ke tanggal hari ini secara default
        document.getElementById("filterTanggal").value = new
Date().toISOString().split('T')[0];
    });
  </script>
</body>
</html>
```

LAMPIRAN 3 DOKUMENTASI PENGUJIAN

No.	Sensor Ultrasonik	Penggaris	Kondisi timer
1.			Normal
2.			Normal
3.			Normal
4.			Normal
5.			Normal
6.			Normal
7.			Stop

8.				Stop
9.				Stop
10.				Stop

Dokumen Daffa

ORIGINALITY REPORT

13%	11%	5%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universitas Diponegoro Student Paper	1%
2	repository.polman-babel.ac.id Internet Source	1%
3	ojs.trigunadharma.ac.id Internet Source	1%
4	Submitted to Universitas Muhammadiyah Sumatera Utara Student Paper	1%
5	Submitted to Universitas Negeri Manado Student Paper	1%
6	journal2.unusa.ac.id Internet Source	<1%
7	Submitted to Konsorsium 4 Perguruan Tinggi Swasta Student Paper	<1%
8	Submitted to Universitas Muhammadiyah Surakarta Student Paper	<1%

9	repository.uin-suska.ac.id Internet Source	<1%
10	Submitted to Institut Teknologi Nasional Malang Student Paper	<1%
11	docplayer.info Internet Source	<1%
12	repository.usbypkp.ac.id Internet Source	<1%
13	Submitted to Politeknik Manufaktur Negeri Bangka Belitung Student Paper	<1%
14	nanopdf.com Internet Source	<1%
15	Submitted to Universitas Negeri Jakarta Student Paper	<1%
16	id.dralexjimenez.com Internet Source	<1%
17	nscpolteksby.ac.id Internet Source	<1%
18	repo.unand.ac.id Internet Source	<1%
19	repository.unbari.ac.id Internet Source	<1%

20	Submitted to Universitas Islam Indonesia Student Paper	<1%
21	repository.itk.ac.id Internet Source	<1%
22	sihoooffice.com Internet Source	<1%
23	Submitted to Institut Bisnis dan Teknologi Indonesia (INSTIKI) Student Paper	<1%
24	Muhammad Wildan, Seftian M. Gilang, Novan Wijaya. "PERANCANGAN SMART HOME DENGAN INTERNET OF THINGS (IoT) MENGGUNAKAN CISCO PACKET TRACER", DEVICE : JOURNAL OF INFORMATION SYSTEM, COMPUTER SCIENCE AND INFORMATION TECHNOLOGY, 2025 Publication	<1%
25	ojs.widyakartika.ac.id Internet Source	<1%
26	repository.trisakti.ac.id Internet Source	<1%
27	medium.com Internet Source	<1%
28	repository.umsu.ac.id Internet Source	<1%

29	repozitorij.unios.hr Internet Source	<1%
30	www.microthings.id Internet Source	<1%
31	dspace.uii.ac.id Internet Source	<1%
32	jurnalmedikahutama.com Internet Source	<1%
33	randomnerdtutorials.com Internet Source	<1%
34	www.mql5.com Internet Source	<1%
35	Achmad Solih, Jamaaluddin Jamaaluddin. "Rancang Bangun Pengaman Panel Distribusi Tenaga Listrik Di Lippo Plaza Sidoarjo Dari Kebakaran Berbasis Arduino Nano", JEEE-U (Journal of Electrical and Electronic Engineering-UMSIDA), 2017 Publication	<1%
36	ejournal.ps.fisip-unmul.ac.id Internet Source	<1%
37	eujyux.tatestreetart.com Internet Source	<1%
38	repository.mercubuana.ac.id Internet Source	<1%

39	repository.petra.ac.id Internet Source	<1%
40	repository.ppns.ac.id Internet Source	<1%
41	Feni Anggraeni, Muhammad Rofiq. "Sistem Monitoring Penyiraman Bibit Tanaman Jeruk Siam Dengan Menggunakan Modul Nodemcu ESP 8266 Dan Sensor Kelembaban Tanah Berbasis IOT", JURNAL SISTEM KOMPUTER ASIA, 2024 Publication	<1%
42	Khairi Wilda Prihati, Lailan Lailan, Attiya Istarini. "Evaluasi Bagian Tubuh Saat Duduk Lama dan Kejadian Nyeri Punggung Bawah pada Pegawai Perkantoran di Kota Sungai Penuh", Journal of Medical Studies, 2023 Publication	<1%
43	ejournal.uki.ac.id Internet Source	<1%
44	eprints.itn.ac.id Internet Source	<1%
45	id.123dok.com Internet Source	<1%
46	text-id.123dok.com Internet Source	<1%

47	www.coursehero.com Internet Source	<1%
48	www.fao.org Internet Source	<1%
49	Anita Fira Waluyo, Taufik Rizki Putra. "Peringatan Dini Banjir Berbasis Internet Of Things (IOT) dan Telegram", Infotek: Jurnal Informatika dan Teknologi, 2024 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

Lampiran Nomor : 30/PL28/KM/PA/2025

SURAT PERNYATAAN

Saya/Kami yang bertandatangan dibawah ini telah menyelesaikan Proyek Akhir yang berjudul:

ALAT PENDETEKSI LAMA WAKTU DUDUK DAN POSISI KEBUNGKUKAN TUBUH BERBASIS IOT

Oleh :

1. Rizky Daffa Pratama /NPM 1052225
2. Tio Pratama /NPM 1052230

Dengan ini menyatakan bahwa isi laporan akhir proyek akhir sama dengan *hardcopy.*
Demikian surat pernyataan ini dibuat dengan sebenar-benarnya.

Sungailiat, 22 Agustus 2025

1.  ..... (Rizky Daffa Pratama)
2.  ..... (Tio Pratama)

Mengetahui,

Pembimbing 1	Pembimbing 2
	
(Aan Febriansyah , S.ST., M.T.)	(Misri Yandi , S.Pi., M.Si.)