

LAPORAN PENELITIAN

SISTEM MESIN PENETAS TELUR OTOMATIS BERBASIS APLIKASI WEB DAN BOARD WEMOS D1 R1 WI-FI ESP8266 ARDUINO COMPATIBLE

Guru Pengampu:

Dalyanta Budi Santosa, S.Pd., M.Eng.



Disusun Oleh:

Abrar Bariq Mu'afa	01 / XII SIJA B
Bintang A'raaf Stevan Putra	10 / XII SIJA B
Eurecsan Dewantoro Putro	13 / XII SIJA B
Hanifah Sinta Nur'Aini	16 / XII SIJA B
Muchhammad Gassa Revaldy Sandy Aji	22 / XII SIJA B
Silfaradila Mei Fizahidah	31 / XII SIJA B

SMK NEGERI 2 KLATEN

TAHUN 2023

KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Tuhan Yang Maha Esa, karena berkat rahmat dan hidayah-Nya laporan ini dapat terselesaikan dengan baik dan selesai tepat pada waktunya. Adapun tujuan penulisan laporan ini adalah untuk memenuhi tugas mata pelajaran SIoT (*Sistem Internet of Things*), dengan judul **“Sistem Mesin Penetas Telur Otomatis Berbasis Aplikasi Web dan Board WeMos D1 R1 Wi-Fi ESP8266 Arduino Compatible”**. Dengan membuat tugas ini kami berharap untuk lebih mengenal tentang alat mesin penetas telur otomatis, aplikasi web, dan mikrokontroler.

Kami sadar, sebagai seorang pelajar yang masih dalam proses pembelajaran, penulisan laporan ini masih terdapat banyak kekurangan. Oleh karena itu, kami berharap adanya kritik dan saran yang bersifat positif guna penulisan karya ilmiah yang lebih baik lagi di masa yang akan datang.

Harapan kami, semoga laporan yang sederhana ini, dapat memberi manfaat tersendiri bagi siapapun yang membacanya.

Klaten, 28 Maret 2023

Penyusun

DAFTAR ISI

HALAMAN JUDUL	ii
KATA PENGANTAR.....	ii
DAFTAR ISI	iii
BAB I PENDAHULUAN	1
A. Latar Belakang Masalah	1
B. Rumusan Masalah	2
C. Maksud dan Tujuan Penelitian.....	2
D. Manfaat Penelitian	3
BAB II LANDASAN TEORI	4
A. Pengertian Penetasan Telur	4
B. Pengertian Mikrokontroler dan IoT	5
C. Pengertian HTTP dan REST API	6
D. Pengertian WebSocket.....	7
BAB III METODE PENELITIAN	8
A. Alat dan Bahan Pembuatan Mesin	8
B. Flowchart dan Alur Kerja Sistem.....	11
C. Perancangan Hardware Sistem	13
D. Perancangan Software Sistem.....	24
BAB IV HASIL PENELITIAN.....	79
A. Kendala dan Solusi.....	79
B. Hasil Pengujian	84
BAB V PENUTUP	85
A. Kesimpulan	85
B. Saran.....	85
DAFTAR PUSTAKA.....	86
LAMPIRAN	

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Budidaya peternakan menjadi salah satu peluang usaha yang menjanjikan, terutama ternak unggas. Ini merupakan dampak positif dari meningkatnya angka permintaan masyarakat terhadap kebutuhan protein hewani seperti daging dan telur. Namun, tingginya angka tersebut perlu diimbangi dengan adanya ketersediaan yang terus-menerus. Kenyataannya saat ini produktivitas unggas dapat dikatakan masih sangat rendah, sekitar 30-60 butir/tahun. Rendahnya produktivitas ini dapat disebabkan oleh lamanya periode mengasuh anak dan istirahat bertelur. Pemeliharaan unggas yang masih bersifat tradisional memiliki lebih banyak resiko kegagalan baik dari segi kuantitas maupun kontinuitas.

Untuk itu dirancang sebuah alat berupa mesin penetas telur, yang ditujukan untuk membantu optimalisasi produktivitas unggas. Setiap bagian dari mesin penetas telur diusahakan sesuai untuk perkembangan struktural dan fisiologi embrio anak unggas. Mesin ini memiliki beberapa parameter biologi, seperti temperatur, kelembapan, dan sirkulasi udara, yang dimanfaatkan untuk mengatur kondisi mesin sehingga sesuai dengan kondisi proses biologi penetasan. Proses penetasan dengan menggunakan mesin otomatis memiliki kelebihan dibanding dengan penetasan tradisional, yaitu: dapat dilakukan sewaktu-waktu, mampu melakukan proses dalam jumlah banyak sekaligus, memiliki tingkat resiko kegagalan yang lebih rendah daripada proses tradisional.

Penelitian ini merupakan salah satu bentuk pengembangan mesin penetas telur konvensional yang sudah ada sebelumnya. Sistem penetasan, yang semula masih bersifat konvensional, dikembangkan menjadi sistem penetas

telur non-konvensional berbasis aplikasi *web*. Penggunaan *web* ditujukan agar pengawasan keadaan dan kontrol mesin penetas telur dapat dilakukan secara *real-time*, mudah, efektif dan efisien melalui *web*.

B. Rumusan Masalah

Berdasarkan latar belakang masalah yang ada, maka dapat ditentukan rumusan permasalahan sebagai berikut:

1. Bagaimana cara kerja mesin penetas telur berbasis *web* dan mikrokontroler?
2. Bagaimana cara membangun mesin penetas telur berbasis mikrokontroler?
3. Bagaimana cara membuat aplikasi *web* berbasis *framework* Laravel untuk pengawasan / *monitoring* dan kontrol kondisi mesin penetas telur?

C. Maksud dan Tujuan Penelitian

Dengan adanya mesin penetas telur berbasis *web*, akan lebih memudahkan operator pada saat mengawasi telur yang akan ditetaskan dengan menggunakan *web*. Hal tersebut dapat meningkatkan efektifitas dan efisiensi.

Berdasarkan rumusan masalah di atas, dapat dicapai tujuan berikut:

1. Dapat mengetahui cara kerja dari mesin penetas telur berbasis *web* dan mikrokontroler.
2. Dapat merancang mesin penetas telur otomatis berbasis mikrokontroler.
3. Dapat membuat *web* untuk pengawasan dan kontrol jarak jauh mesin penetas telur.

D. Manfaat Penelitian

1. Meningkatkan persentase keberhasilan telur yang akan ditetaskan dibandingkan dengan penggeraman alami.
2. Penetasan telur dapat dilakukan terus menerus tanpa dipengaruhi oleh kondisi cuaca karena telur ditempatkan di ruang khusus.
3. Kontrol terhadap kualitas telur lebih mudah dilakukan karena sudah dilengkapi fitur pengawasan / *monitoring* dan kontrol jarak jauh berbasis *web* yang dapat diakses di mana saja dan tentunya fleksibel.
4. Daya hidup anak ayam/itik hasil penetasan dengan mesin tetas lebih tinggi karena perubahan suhu tidak terlalu ekstrim.

BAB II

LANDASAN TEORI

A. Pengertian Penetasan Telur

Penetasan telur adalah proses mengeramkan telur untuk memacu pertumbuhan dan perkembangan embrio menjadi anak ayam yang mampu menetas dengan cara memecahkan dan keluar dari kerabang dalam kondisi sehat sehingga layak untuk dipelihara dan dapat diperjualbelikan. Menurut (Dirjen Peternakan (2008), penetasan adalah kegiatan penggeraman (*setter*) dan penetasan (*hatcher*) telur tetas untuk menghasilkan bibit ayam untuk keperluan sendiri atau untuk diperjualbelikan. Penetasan telur dapat dilakukan dengan menggunakan mesin penetas, sehingga manajemen penetas perlu dikuasai oleh pelaku penetas telur.

Keberhasilan penetasan dengan menggunakan mesin penetas telur dipengaruhi oleh banyak faktor yang saling berkaitan, jika salah satu faktor yang berpengaruh kurang diperhatikan tidak memungkinkan hasilnya sesuai dengan harapan yaitu presentase anak ayam layak jual tinggi. Keberhasilan penetasan telur dipengaruhi oleh mesin penetas telur juga tergantung pada kondisi telur yang ditetaskan sehingga manajemen telur tetas sebelum diinkubasi seperti koleksi telur, transportasi telur, penyimpanan telur dalam ruangan dingin, dan penghangatan telur setelah disimpan perlu diperhatikan. Faktor lain yang memegang peranan penting dalam mempengaruhi keberhasilan penetasan adalah manajemen telur tetas selama diinkubasi, seperti stabilitas temperatur dan kelembapan ruangan mesin tetas, pemuturan telur, pengaturan ventilasi, dan posisi penyimpanan telur di dalam rak. Pengeluaran anak ayam yang menetas dari mesin tetas memegang peranan penting untuk memperoleh kualitas anak yang baik. Keterlambatan pengangkatan anak dari

dalam mesin akan menyebabkan anak unggas mengalami kekurangan air (dehidrasi) sehingga anak ayam tidak layak untuk dijual.

B. Pengertian Mikrokontroler dan IoT

Mikrokontroler adalah sebuah komputer kecil yang dikemas dalam bentuk chip IC (*Integrated Circuit*) serta dirancang untuk melakukan tugas atau operasi tertentu. Umumnya sebuah IC mikrokontroler ini terdiri dari satu atau lebih inti prosesor (CPU), memori (RAM dan ROM) dan perangkat input dan output yang dapat diprogram. Dalam pengaplikasiannya, pengendali mikro yang dalam bahasa Inggris disebut dengan *microcontroller* ini digunakan dalam produk atau perangkat yang dikendalikan secara otomatis.

Mikrokontroler sering digunakan dalam aplikasi IoT (*Internet of Things*) karena dapat mengontrol perangkat elektronik dan menghubungkannya ke internet. IoT adalah konsep di mana berbagai perangkat elektronik dihubungkan dan dapat berkomunikasi satu sama lain melalui internet. Misalnya, sebuah sistem *smart home* dapat menggunakan mikrokontroler untuk mengontrol suhu, cahaya, dan perangkat elektronik lainnya di dalam rumah, dan dapat dihubungkan ke internet agar dapat diatur dari jarak jauh melalui smartphone atau tablet.

Salah satu contoh penggunaan mikrokontroler dalam IoT adalah Arduino, sebuah platform *open-source* untuk membuat perangkat elektronik yang terhubung ke internet. Arduino menggunakan mikrokontroler sebagai otak dari perangkat elektronik yang dibuat oleh penggunanya. Dengan menggunakan bahasa pemrograman yang sederhana, pengguna dapat membuat perangkat IoT yang dapat terhubung ke internet dan berkomunikasi dengan perangkat lainnya.

Mikrokontroler juga sering digunakan dalam aplikasi sensor. Sebuah sensor dapat digunakan untuk mengukur suhu, kelembapan, atau cahaya, dan

mikrokontroler dapat digunakan untuk mengambil data dari sensor dan mengirimkan data tersebut ke *server* atau perangkat lain melalui internet. Pada penelitian ini, mikrokontroler juga akan digunakan untuk hal yang sama, yaitu mengirimkan data sensor mesin penetas telur ke *server* yang kemudian ditampilkan pada *interface web*.

C. Pengertian HTTP dan REST API

Pertukaran data melalui internet dibutuhkan sebuah protokol internet yang salah satunya adalah HTTP. HTTP (*Hypertext Transfer Protocol*) digunakan untuk mengirimkan permintaan (*request*) dari *client* ke *server*, dan untuk mengirimkan respon (*response*) dari *server* ke *client*. Protokol ini merupakan dasar dari banyak aplikasi *web*, seperti halaman *web* dan API. Aplikasi *web* sendiri ialah sebuah aplikasi yang memanfaatkan *web browser* dan teknologi *web* untuk melakukan sebuah tugas melalui internet.

Pada umumnya, aplikasi *web* menggunakan sebuah API sebagai perantara pertukaran datanya menuju *server*, salah satunya adalah REST API. REST API (*Representational State Transfer Application Programming Interface*) adalah jenis API yang menggunakan protokol HTTP untuk mengakses dan mengelola data. REST API mengirimkan data dalam format yang dapat dibaca oleh manusia seperti JSON atau XML. REST API juga menggunakan metode HTTP seperti *GET*, *POST*, *PUT*, dan *DELETE* untuk mengakses sumber daya.

Kaitannya dengan IoT dan mikrokontroler, HTTP dan REST API digunakan untuk menghubungkan perangkat IoT dengan *server* atau aplikasi lainnya melalui internet. Misalnya, sebuah sensor yang terhubung ke mikrokontroler dapat mengirimkan data ke *server* melalui REST API. *Server* dapat mengambil data dari sensor, melakukan pemrosesan, dan mengirimkan respon kembali ke mikrokontroler.

D. Pengertian WebSocket

WebSocket adalah protokol komunikasi dua arah (*bi-directional*) yang digunakan untuk mengirimkan data secara *real-time* melalui internet. Protokol ini berbeda dengan HTTP yang hanya mendukung komunikasi satu arah (*uni-directional*) dari *client* ke *server*. Dalam protokol WebSocket, *client* dan *server* dapat saling berkomunikasi secara langsung dan dapat mengirimkan data secara asinkron.

WebSocket memungkinkan aplikasi *web* untuk membangun koneksi yang lebih efisien dan efektif, karena koneksi dapat dijaga terbuka (*persistent*) tanpa perlu membuka dan menutup koneksi seperti pada HTTP. Koneksi terbuka ini memungkinkan aplikasi IoT untuk memperbarui data secara *real-time*. Dalam aplikasi IoT, WebSocket dapat digunakan untuk mengirimkan data secara *real-time* dari *server* ke mikrokontroler dan sebaliknya, seperti mengirimkan perintah untuk mengontrol perangkat elektronik atau menerima data sensor. Metode HTTP dapat digunakan untuk pengiriman data dari mikrokontroler ke *server*, seperti mengirimkan data sensor dari mikrokontroler ke *server* untuk dianalisis atau disimpan.

Penggunaan kedua protokol ini dalam aplikasi IoT dapat memberikan keuntungan yang berbeda. Penggunaan HTTP pada mikrokontroler memungkinkan pengiriman data yang dapat disimpan dan dimuat kembali, sedangkan penggunaan WebSocket pada server dapat memungkinkan aplikasi untuk mengirimkan data secara *real-time* dengan *latency* yang lebih rendah dan lebih efisien dalam penggunaan.

BAB III

METODE PENELITIAN

A. Alat dan Bahan Pembuatan Mesin

1. Board WeMos D1 R1

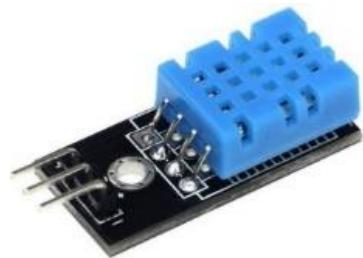
Agar dapat melakukan pertukaran data secara online, pada mesin penetas telur akan dipasang *board* WeMos D1 R1. WeMos D1 R1 merupakan *board* yang menggunakan ESP8266 sebagai mikrokontroler sekaligus modul *Wi-Fi* dan dirancang menyerupai Arduino Uno. Kelebihan dari WeMos D1 R1 ini adalah bersifat *open-source*, kompatibel dengan Arduino, dapat diprogram menggunakan Arduino IDE, *pinout* yang kompatibel dengan Arduino Uno, dapat berdiri sendiri tanpa menggunakan mikrokontroler lain, dan memiliki prosesor 32-bit dengan kecepatan 80 MHz. Pada mesin penetas telur, WeMos digunakan sebagai *board* yang dapat mengirim data suhu dan kelembapan, dan mengendalikan kerja *relay*.



Gambar 3.2 Board WeMos D1 R1

2. Sensor DHT11

DHT11 adalah modul sensor yang berfungsi untuk mendeteksi suhu dan kelembapan suatu ruang yang memiliki output tegangan analog yang dapat diolah lebih lanjut menggunakan mikrokontroler. DHT11 berfungsi untuk mengukur suhu dan kelembapan yang ada di dalam mesin penetas telur.



Gambar 3.3 Sensor DHT11

3. DC Adapter 12V

DC adapter adalah perangkat yang berfungsi mengubah tegangan AC menjadi DC. Maksudnya adalah tegangan arus listrik bolak – balik (AC) akan diubah menjadi tegangan arus listrik yang searah (DC) bertegangan 12V. *DC adapter* di sini berfungsi sebagai sumber tegangan untuk mesin penetas telur.



Gambar 3.4 DC adapter 12V

4. Cooling Fan

Cooling fan adalah perangkat yang memiliki daya putar untuk menghasilkan angin atau untuk mendinginkan suhu. *Cooling fan* berfungsi untuk menurunkan suhu mesin penetas telur jika terlalu tinggi.



Gambar 3.5 Cooling fan

5. Lampu Pijar

Lampu pijar dibutuhkan dalam proses penetasan telur. Kehangatan dari lampu digunakan untuk pengganti induk ayam. Lampu yang dibutuhkan yaitu sebanyak 3 buah lampu pijar.



Gambar 3.6 Lampu pijar

6. Motor Sinkron

Motor sinkron adalah motor AC yang bekerja pada kecepatan tetap pada sistem frekuensi tertentu. Motor sinkron berfungsi untuk menggerakkan rak di dalam mesin penetas telur agar suhu yang didapat merata.

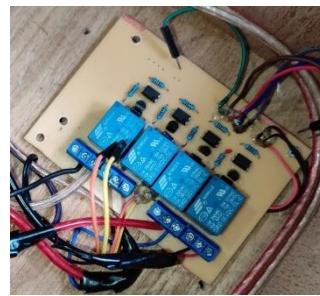


Gambar 3.7 Motor sinkron

7. PCB Custom

PCB (*Printed circuit board*) *custom* di sini adalah PCB yang dibuat dan desain secara mandiri oleh kelompok peneliti. PCB ini memiliki rangkaian untuk empat *relay* 5V yang berfungsi sebagai saklar komponen – komponen yang lainnya. Selain itu ada rangkaian *optocoupler* yang berfungsi sebagai penguat tegangan, karena output dari pin digital WeMos D1 hanya 3.3V. Selanjutnya ada rangkaian dioda yang berfungsi sebagai penyearah

tegangan dalam rangkaian PCB ini. Kemudian ada rangkaian transistor berfungsi sebagai *switching* untuk mengaktifkan kontak *relay*. Terakhir, terdapat rangkaian terminal blok berfungsi sebagai menyambungkan kabel dari PCB ke rangkaian.



Gambar 3.8 PCB custom

8. Boks Kayu

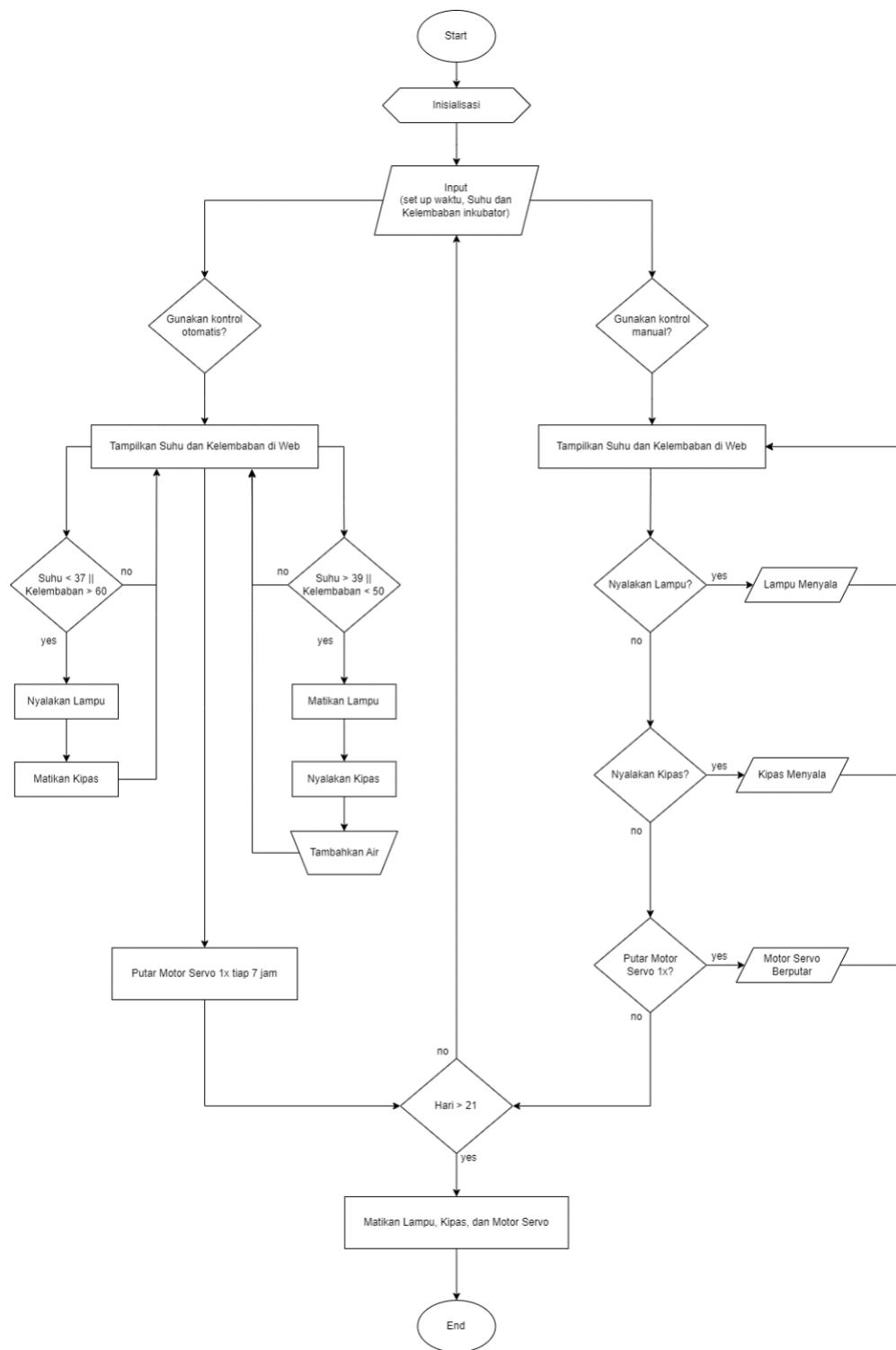
Boks kayu yang dimaksud di sini berfungsi sebagai wadah untuk menyimpan telur dan wadah untuk rangkaian elektronika.



Gambar 3.9 Boks kayu

B. Flowchart dan Alur Kerja Sistem

Flowchart atau bagan alur adalah diagram yang menampilkan langkah – langkah dan keputusan untuk melakukan sebuah proses dari suatu program atau sistem. *Flowchart* dari sistem mesin penetas telur berbasis *web* ini dapat dilihat pada Gambar 3.10.



Gambar 3.10 Flowchart mesin penetas telur berbasis web

Untuk alur kerjanya yaitu dimulai dari input data user dari *web monitoring* penetas telur. User disediakan pilihan mode otomatis dan manual. Berikut adalah penjelasan alur kerja dari setiap mode tersebut.

1. Mode otomatis

Motor sinkron akan berputar 180 derajat sekali setiap 7 jam, sedangkan lampu dan kipas akan menyala maupun mati tergantung pada suhu dan kelembapan di dalam boks penetas telur. Jika suhu kurang dari 37 atau kelembapan lebih dari 60, maka lampu akan hidup dan kipas akan mati. Jika suhu suhu lebih dari 39 atau kelembapan kurang dari 50, maka lampu akan mati dan kipas akan hidup. Hal ini bertujuan untuk menstabilkan suhu dan kelembapan penetas telur agar telur menetas dengan sempurna di kemudian hari.

2. Mode manual

Pada *web*, user disediakan tombol *switch* untuk lampu, kipas, dan motor di mode manual. Dalam hal ini, user bebas untuk menghidupkan dan mematikan komponen – komponen tersebut dan keputusan tersebut harus didasarkan pada pengetahuan user tentang suhu dan kelembapan yang optimal untuk telur – telur di dalamnya. Suhu dan kelembapan sudah ditampilkan secara *real-time* dan online pada panel *monitoring* di *web*.

C. Perancangan Hardware Sistem

1. Skematik Rangkaian Elektronika (oleh: Abrar)

Skema rangkaian elektronika merupakan *blueprint* dari model peralatan elektronika yang ingin kita bangun. Apapun jenis alat yang ingin kita buat, haruslah terlebih dahulu kita buatkan skemanya. Karena dengan adanya skema, kita akan mengetahui apa saja yang kita butuhkan dan apa saja yang perlu kita lakukan untuk membangun alat tersebut.

Dari sanalah kita bisa berpatokan apakah kita sanggup memiliki bahan-bahan komponen pembangunnya. Serta memperkirakan kesanggupan kita untuk merangkai komponen tersebut menjadi alat yang kita inginkan. Sangat penting untuk mengetahui keseluruhan komponen yang ada sekarang ini, karena komponen yang beraneka ragam inilah nantinya yang akan menentukan bagaimana kerja dan fungsinya suatu elektronika setelah dirangkai nanti.

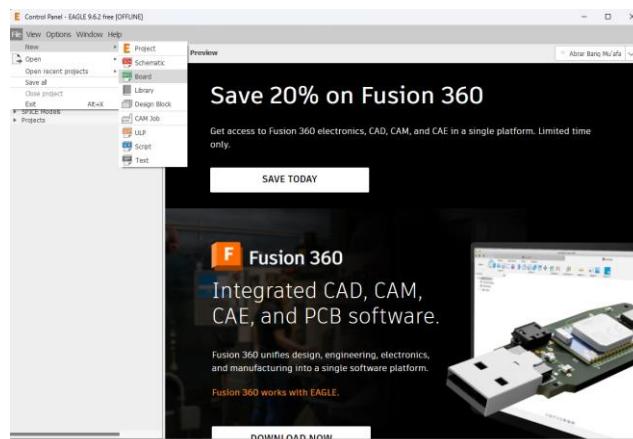
Setelah mengetahui sekilas mengenai skematik rangkaian, langkah selanjutnya adalah membuat desain skematik. Saat ini banyak sekali *software* yang tersedia untuk membuatnya. Salah satunya adalah *software* EAGLE. EAGLE adalah salah satu freeware yang digunakan untuk membantu dalam pembuatan skema rangkaian elektronika. *Software* ini memiliki fitur yang cukup lengkap sehingga cocok untuk membantu dalam menyelesaikan skematik rangkaian elektronika.



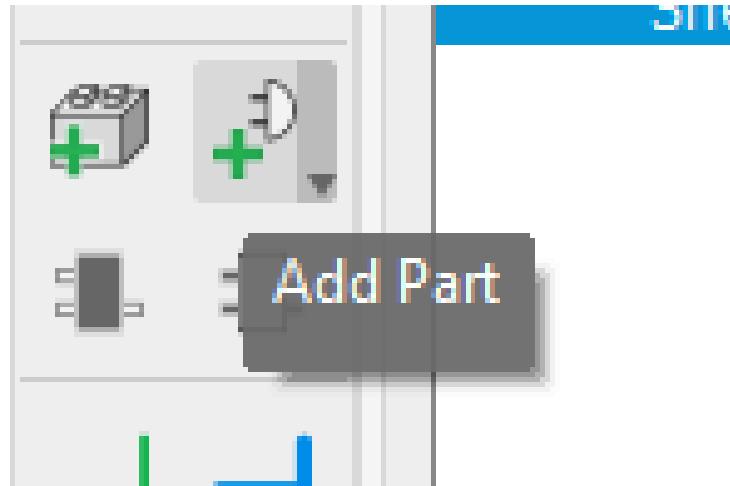
Gambar 3.11 Logo EAGLE

Berikut adalah langkah-langkah membuat skema rangkaian menggunakan *software* EAGLE:

- Buka *software* EAGLE. Kemudian klik menu File > New > Schematic.



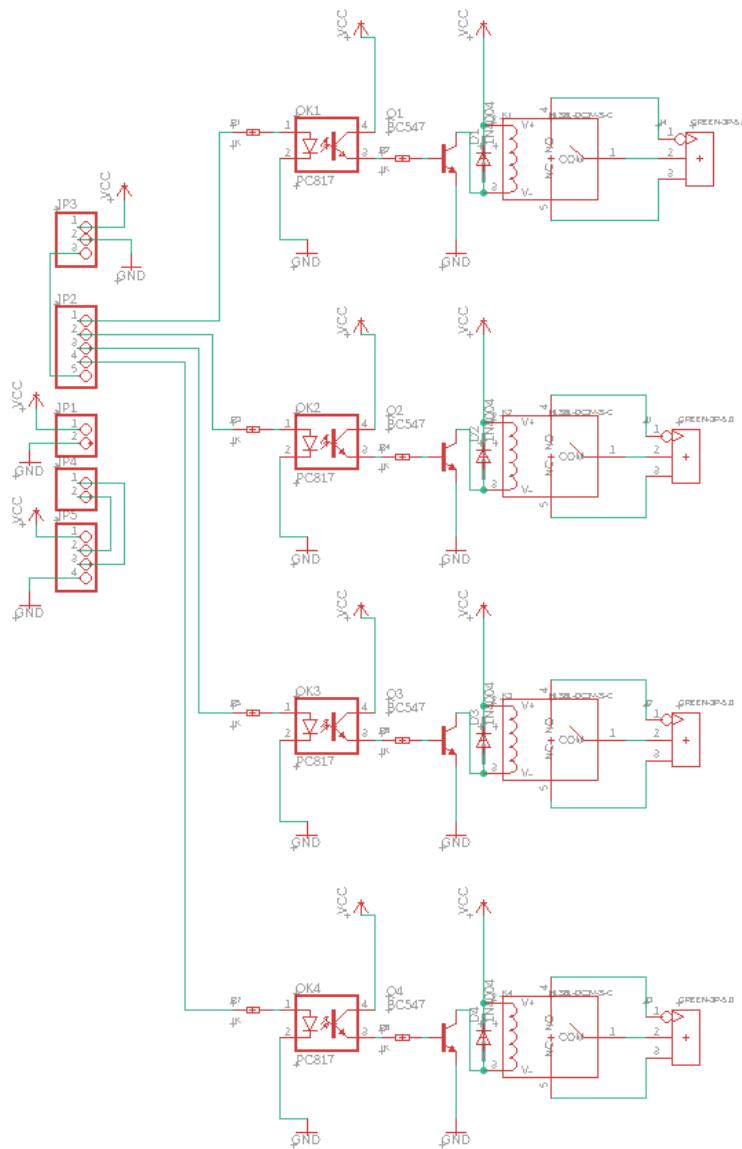
- b. Tambahkan komponen dengan mengeklik Add Part.



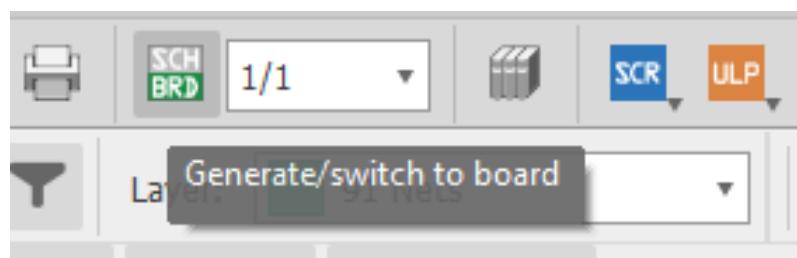
- c. Siapkan *part – part* berikut dari *library* EAGLE:

- 4x Relay 5v
- 4x Optocoupler PC547
- 4x Transistor BC547
- 8x Resistor 1k
- 4x Dioda 1N4007
- 1x Pin header 1x5
- 1x Pin header 1x4
- 1x Pin header 1x3
- 2x Pin header 1x2
- 4x Terminal block 3 pin

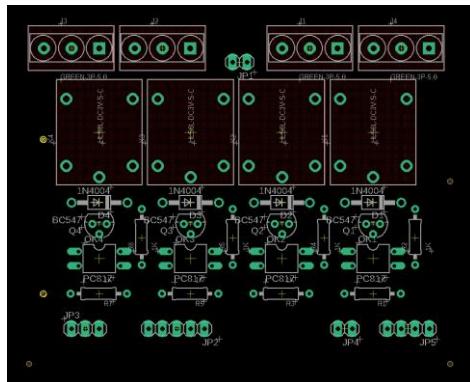
- d. Sambung antar pin menggunakan tool net, dan juga beri power VCC dan GND hingga berbentuk seperti pada gambar di bawah ini.



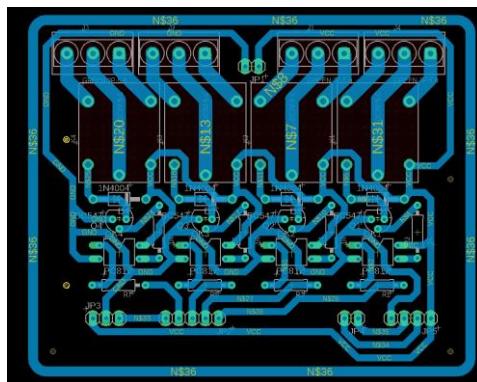
- e. Pada skematik rangkaian yang telah dibuat tadi, klik Generate/switch to board.



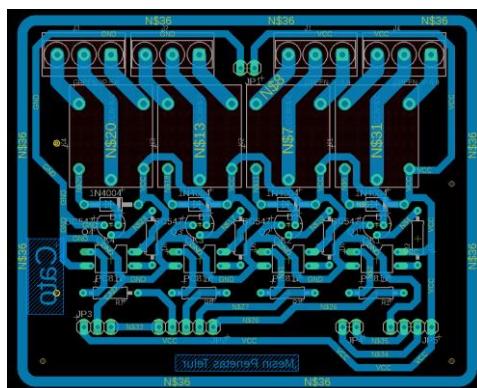
f. Kemudian susun komponen hingga rapi.



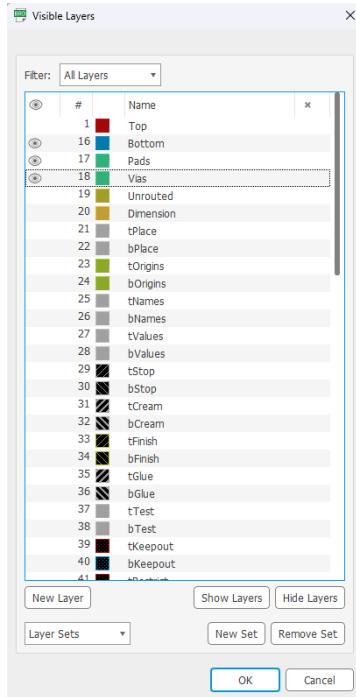
g. Jika sudah rapi, lakukan pembuatan jalur PCB dengan menggunakan *tool* Route Airwire. Untuk ketebalan jalur pada Relay ke Terminal block sebesar 100 mm, untuk tebal jalur komponen lainnya sebesar 50 mm.



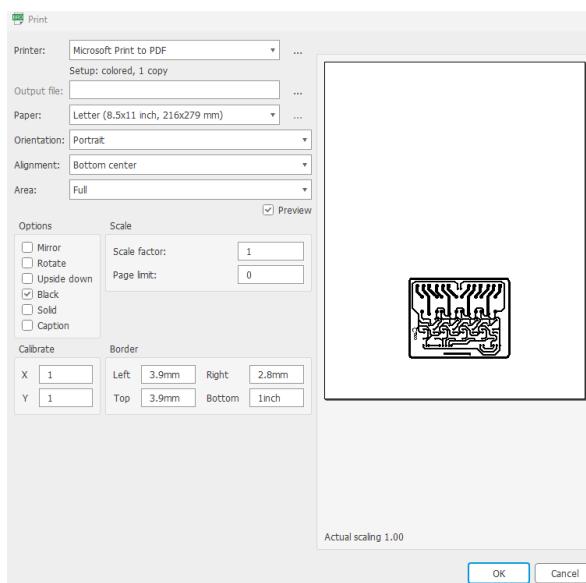
h. Setelah itu beri tulisan agar bisa memberi identitas PCB tersebut dengan menggunakan *tool* Text.



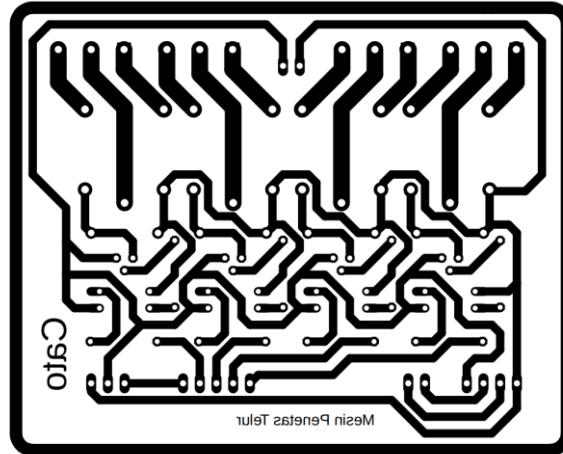
- i. Jika dirasa semua sudah terpasang dengan sempurna, langkah terakhir yaitu print. Masuk ke *tool* Layer Setting, dan *unchecklist* semua *layer* kecuali *layer* Bottom, Pads, dan Vias, kemudian klik OK.



- j. Setelah *layer* di-*setting*, langkah berikutnya print desain PCB dalam bentuk file PDF. Untuk konfigurasi printnya seperti gambar di bawah.

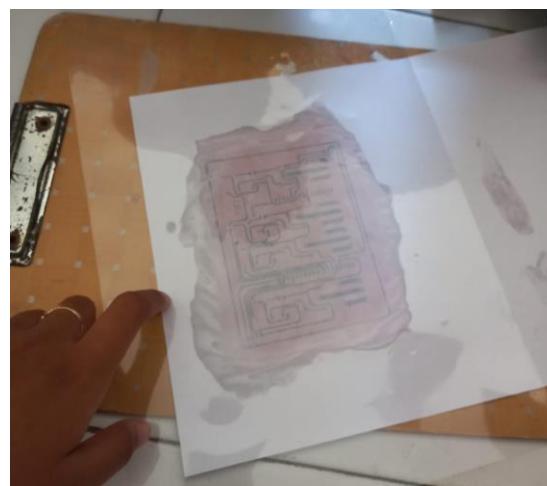


- k. Berikut adalah hasil print-nya.



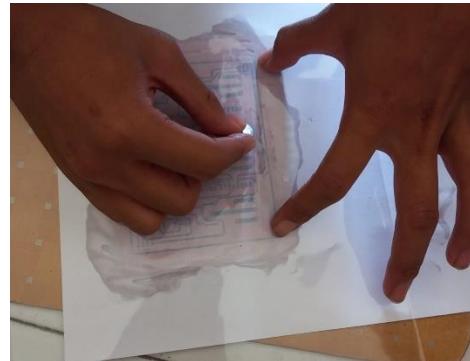
2. Pencetakan PCB Custom (oleh: Abrar, Dewa, Sinta, Gassa, Ida)

- a. Langkah pertama yang kami lakukan yaitu menyiapkan terlebih dahulu PCB polos dan Autan.
- b. Kemudian buka Autan, lalu kuaskan hingga merata pada permukaan lapisan tembaga pada PCB polos.
- c. Tempelkan kertas printing desain yang sudah dipersiapkan pada bagian lapisan tembaga PCB polos.

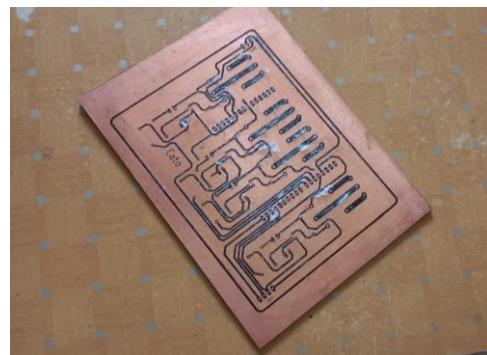


- d. Letakkan di tempat yang rata dengan sisi tembaga menghadap keatas, taruh diatasnya lagi mika plastik yang biasa kita gunakan untuk

menjilid. Gosok menggunakan sisi uang logam sampai merata dengan ditekan supaya jalur bisa menempel namun jangan terlalu keras agar tidak rusak.



- e. Jika proses menggosok sudah selesai, selanjutnya diamkan dulu beberapa menit lalu lepaskan secara perlahan kertas print dengan PCB secara hati-hati. Jika ternyata ada sebagian jalur yang belum menempel, tebalkan jalur dengan menggunakan spidol.
- f. Setelah kertas print diambil biarkan sampai kering betul dengan sendirinya didalam ruangan atau di angin – anginkan selama 30 – 60 menit.
- g. Jika sudah kering, cuci PCB untuk menghilangkan sisa – sisa kertas yang ikut menempel supaya lebih cepat pada proses pelarutan. Gosok pelan-pelan supaya jalur yang sudah menempel tidak mengelupas lagi.
- h. Jika hanya sedikit jalur yang tidak sempurna atau terputus, maka bisa ditambal menggunakan spidol marker permanen.



3. Pelarutan PCB Custom (oleh: Abrar, Dewa, Gassa, Sinta, Ida)

- a. Siapkan wadah plastik, air panas, dan larutan FeCl_3 (*ferric chloride*).
- b. Persiapkan larutan *ferric chloride* pada wadah, kemudian tambahkan air panas secukupnya dan aduk sampai larut.
- c. Masukkan PCB yang sudah di transfer tadi kedalam wadah berisi *ferric chloride*. Kemudian wadah harus di goyang – goyang terus sampai bagian tembaga yang tidak diperlukan hilang atau larut 100% sehingga tidak ada lagi jalur yang konslet antara satu dengan yang lainnya.
- d. Lakukan kira – kira 30 menit, biasanya ditandai dengan warna putih pada bagian tembaga PCB yang sudah larut. Jangan terlalu lama, sebab tembaga bisa habis seluruhnya.



4. Pengeboran PCB Custom (oleh: Abrar, Dewa, Sinta, Gassa, Silfaradila)

Setelah tinta bersih, langkah selanjutnya adalah membuat lubang pada PCB untuk pin – pin untuk komponen yang digunakan. Pada kali ini, kami membuat lubang dengan bor khusus untuk PCB. Kami menggunakan mata bor ukuran 1 milimeter dan 0,8 milimeter.

- a. Langkah pertama yaitu siapkan alat bor yang akan digunakan. Siapkan juga PCB yang telah dilarutkan tadi.

- b. Nyalakan bor dengan menghubungkannya ke stop kontak. Setelah itu arahkan mata bor sesuai dengan jalur yang akan dilubangi. Lakukan sesuai prosedur yang ada.
- c. Jangan lupa pilih ukuran mata bor sesuai kebutuhan. Lakukan berulang hingga selesai.



5. **Penyolderan PCB Custom** (oleh: Abrar, Sinta, Silfaradila)
 - a. Siapkan solder, papan PCB custom, dan komponen – komponen yang akan disolder.
 - b. Panaskan solder selama beberapa menit.
 - c. Pasang komponen – komponen elektronika pada PCB. Letakkan sesuai pengaturan.
 - d. Lakukan penyolderan secara bertahap dengan durasi 3 – 5 detik.



- e. Gunakan penyedot timah apabila ingin melepas solder yang tidak diinginkan.
 - f. Sentuh timah dengan solder panas selama 3-4 detik.
6. **Pemasangan Rangkaian Elektronika** (oleh: Dewa, Sinta, Gassa)

Kegiatan yang kami lakukan selanjutnya yaitu memasang semua rangkaian elektronika ke dalam boks mesin penetas telur. Di sini kami melakukan pengeboran yang nantinya akan menjadi tempat rangkaian elektronikanya.

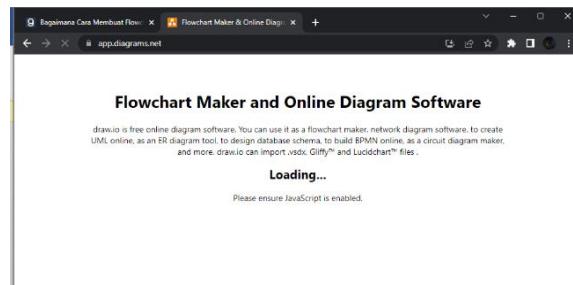


D. Perancangan Software Sistem

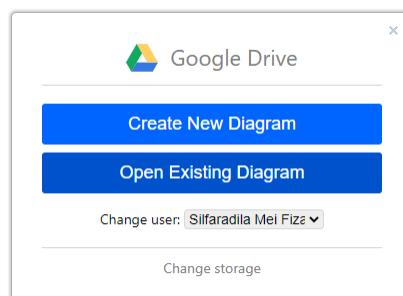
1. Pembuatan Flowchart (oleh: Silfaradila)

Dewasa ini banyak sekali *tools* online yang menyediakan layanan membuat *flowchart* dengan mudah. Salah satu dari beberapa *tools* tersebut adalah draw.io. Draw.io adalah layanan yang dikembangkan khusus untuk membuat diagram secara online, dengan bermodalkan koneksi internet dan browser yang mendukung HTML5 bisa diakses melalui *web*. Berikut merupakan langkah – langkah menggunakan draw.io untuk membuat *flowchart*:

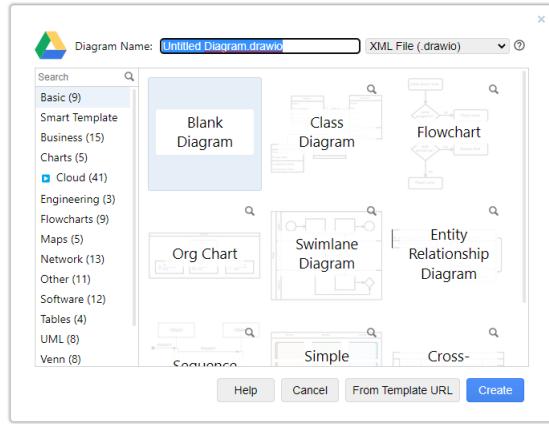
- Membuat skenario untuk *flowchart*. Untuk skenarionya sendiri dapat dilihat lagi pada sub-bab **B. Flowchart dan Alur Kerja Sistem**.
- Kunjungi situs draw.io. Tunggu hingga *loading* selesai. Jika belum pernah menggunakan draw.io sebelumnya, diperlukan melakukan *sign up* terlebih dahulu.



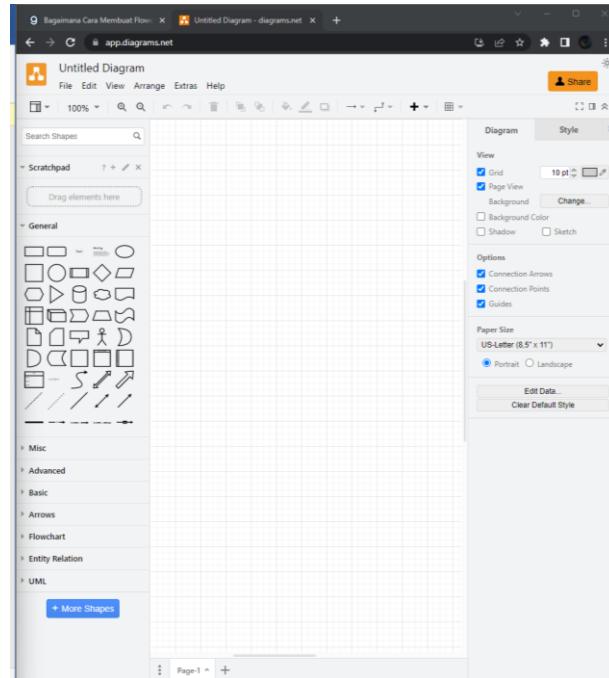
- Pada jendela ini kita ditawarkan apakah ingin membuat diagram *flowchart* baru atau membuka diagram *flowchart* yang sudah ada. Di sini saya akan membuat diagram *flowchart* baru.



- d. Draw.io menyediakan beberapa *template* yang bisa digunakan secara gratis. Jika tidak ingin menggunakan *template* tersebut, maka kita bisa memilih Blank Diagram.

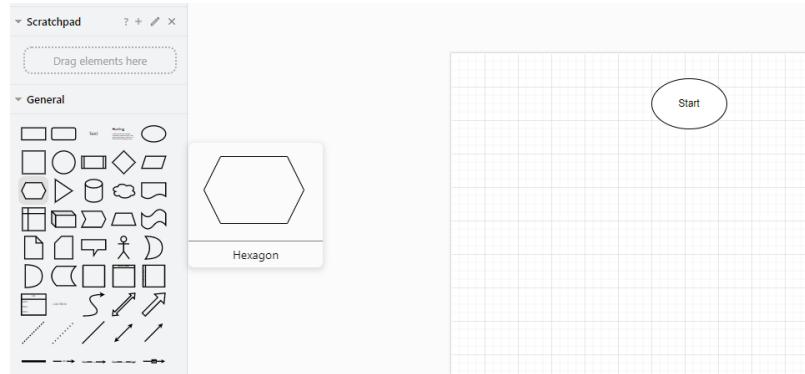


- e. Berikut adalah tampilan *workspace* draw.io yang akan digunakan untuk membuat diagram *flowchart*.

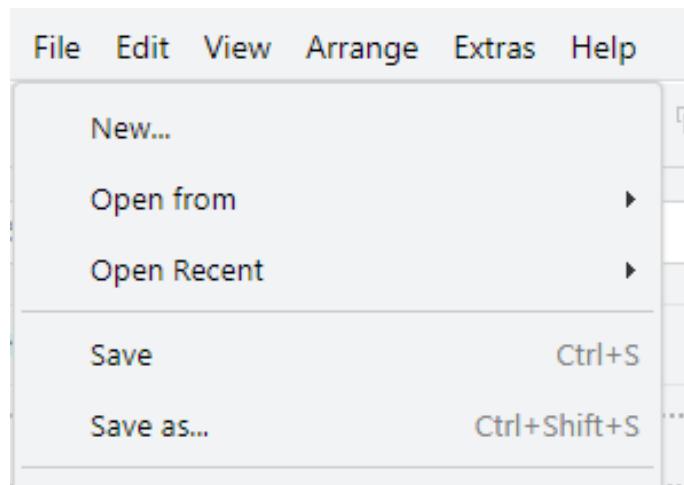


- f. Mulai buat diagram *flowchart* sesuai dengan skenario yang telah kita susun di awal. Manfaatkan *tools* yang ada sesuai dengan fungsi dan

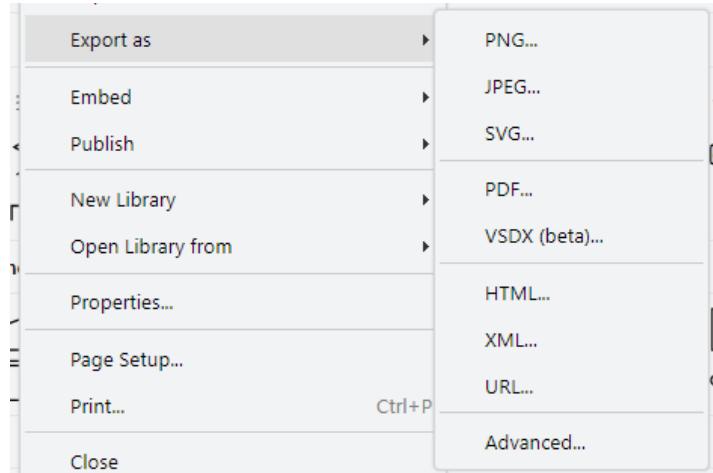
kebutuhan. Misalnya, kita gunakan simbol Terminator untuk menyatakan *start/end program*.



- g. Jika telah selesai membuat diagramnya, simpan file diagramnya pada menu File > Save atau File > Save as...



- h. Selain itu, draw.io memungkinkan kita untuk mengekspor file ke dalam tipe gambar maupun tipe dokumen lain. Pada menu File pilih Export as dan akan ditampilkan beberapa tipe file untuk diekspor seperti PNG, JPEG, SVG, PDF, dan lainnya.



2. Perancangan Desain Web (oleh: Dewa)

Pekerjaan yang saya lakukan dalam projek kelompok ini secara garis besar bekerja dalam hal desain, yaitu mendesain *web* yang akan digunakan dalam *monitoring* dan kontrol mesin penetas telur, dengan memperhatikan segi UI/UX (*User Interface* dan *User Experience*) sehingga *web* dapat nyaman dilihat dan dioperasikan.

Software yang saya pakai adalah Figma. Figma adalah *software* editor grafis vektor dan alat *prototyping* berbasis *web* serta fitur offline tambahan yang diaktifkan oleh aplikasi *desktop* untuk macOS dan Windows. Jadi dalam *software* Figma ini saya mendesain dari awal, mulai dari proses sketsa, wireframe, finalisasi desain, dan *prototyping*.



Gambar 3.12 Logo Figma

Berikut langkah – langkah saya dalam mendesain *web* secara urut:

- Membuat *wireframe* atau rancangan awal terkait dengan fitur – fitur yang akan disertakan. Definisi *wireframe* difungsikan sebagai kerangka awal sebelum membuat desain halaman *web* atau antarmuka sebuah aplikasi. Dalam kasus ini, saya menyertakan 3 panel *monitoring* untuk

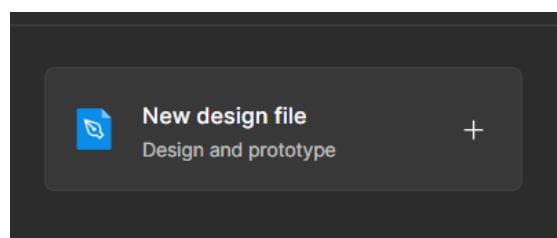
suhu, kelembapan, sisa hari, dan beberapa tombol kontrol untuk komponen kipas, motor, dan lampu, serta mode otomatis dan manual.



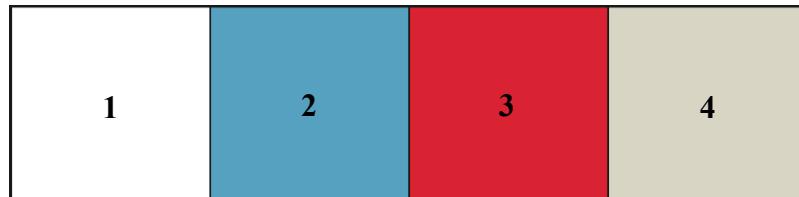
- b. Membuat sketsa awal dari UI *web* dengan memperhatikan UX atau kemudahan dalam penggunaan. Gambar di bawah ini adalah sketsa *layout* panel kontrol.



- c. Buka *software Figma*, kemudian buat file desain baru.

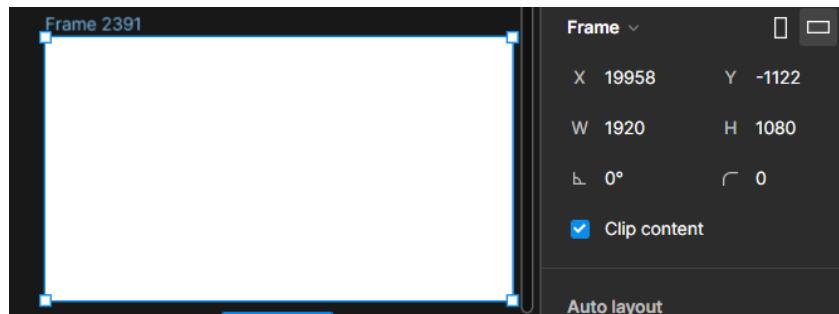


- d. Riset & kumpulkan warna dengan menggunakan konsep hierarki.



Warna dengan nomor 1 dan 4 saya jadikan sebagai hierarki terendah dan diterapkan sebagai latar belakang atau untuk aksen yang terkesan rendah. Kemudian, warna dengan nomor 2 dan 3 saya jadikan sebagai hierarki tertinggi dan diterapkan sebagai aksen yang penting dan harus diperhatikan.

- e. Siapkan aset – aset desain yang akan digunakan seperti *icon*, *font*, ilustrasi atau gambar.
- f. Buat *frame* dengan dimensi 1920x1080, karena dimensi tersebut adalah dimensi *web* yang ideal.

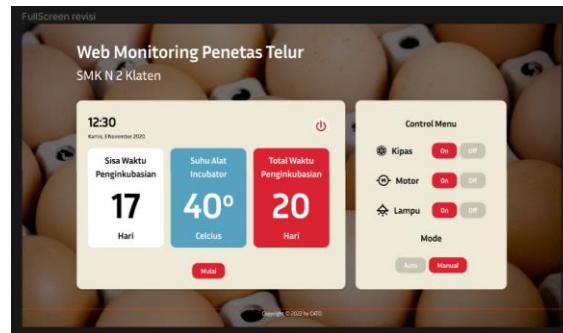


- g. Mulai membuat desain dengan menggunakan informasi yang sudah dipersiapkan sebelumnya. Jangan berhenti dengan konsep pertama, selalu pikirkan bahwa dalam membuat desain harus dengan perspektif pengguna, apakah desain tersebut nyaman dilihat dan digunakan.

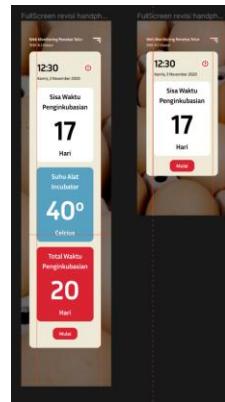
Konsep pertama cenderung sempit dan hanya menonjolkan panel *monitoring* daripada panel kontrol, dan *empty space* terlalu besar dan mengganggu.



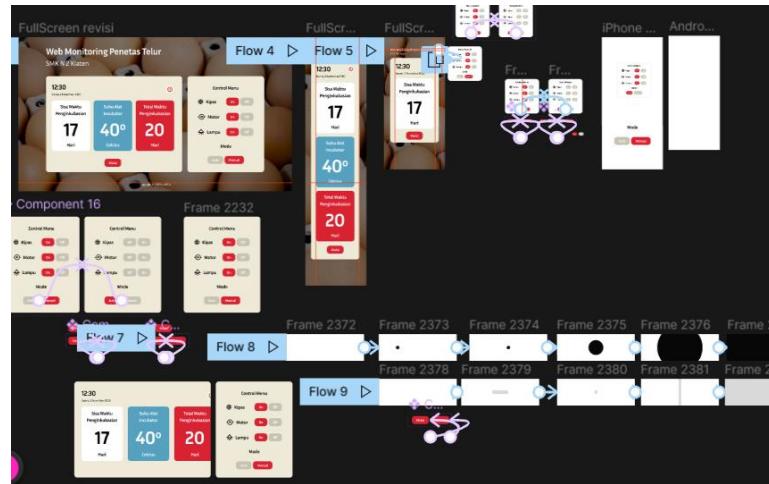
Panel *monitoring* dan kontrol pada konsep kedua terorganisir dengan baik, *empty space* luas dan rata sehingga nyaman dilihat, judul terpisah dengan blok panel sehingga pengguna dapat berfokus pada panel *monitoring* dan kontrol.



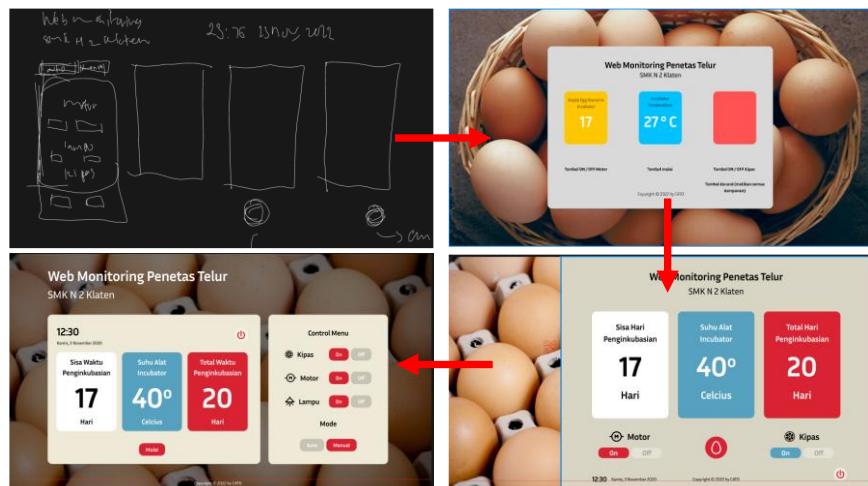
h. Bila diperlukan, buat desain untuk resolusi *mobile*.



- i. Buat *prototype* konsep desain kedua menggunakan fitur Prototype di agar programmer *front-end* tahu cara kerja UI/UX *web* dengan baik.



- j. Gambar di bawah ini adalah tahapan saya dalam mendesain *web*.



3. Instalasi Layanan Server (oleh: Bintang)

Agar aplikasi *web* dapat diakses secara online, dibutuhkan sebuah *server* untuk meng-*hosting* *web* tersebut beserta *database*-nya. Untuk menghemat biaya, karena kebetulan ayah saya adalah seorang pengusaha ISP (*Internet Service Provider*) dan *data center*, saya meminta izin kepada beliau untuk menggunakan salah satu *server*-nya untuk penelitian ini.

Server tersebut berbasis sistem operasi Ubuntu yang merupakan distribusi dari sistem operasi Debian GNU/Linux.

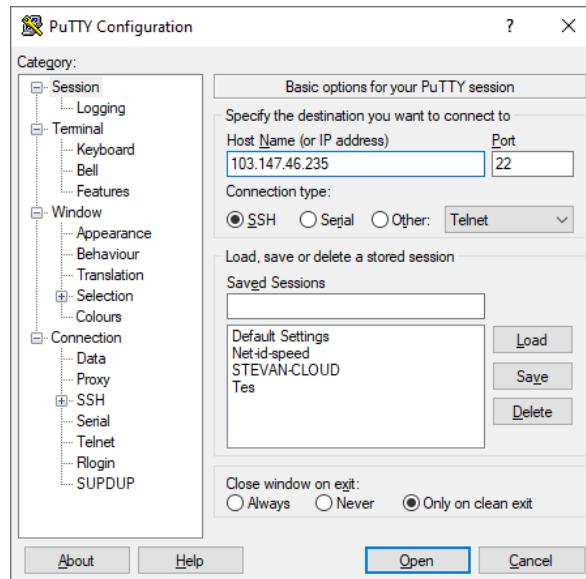
Ubuntu ini menjadi salah satu dari sekian banyak distribusi Linux yang cocok diinstal pada berbagai perangkat, seperti komputer atau laptop, *mobile*, hingga *private server* untuk pengembangan *server*, *desktop*, dan kebutuhan *Internet of Things*.



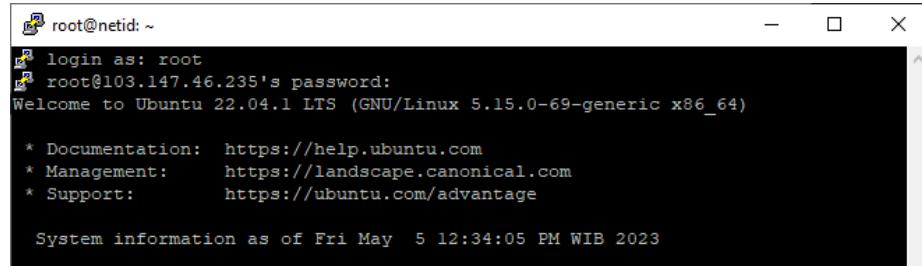
Gambar 3.13 Logo Ubuntu

Secara garis besar, berikut adalah langkah – langkah saya dalam mengkonfigurasi *web* dan *database* agar dapat diakses secara online:

- Login* ke *server* secara online menggunakan *software* PuTTY. *Server* di sini memiliki *IP address* 103.147.46.235.



- b. *Login* sebagai *root user* karena digunakan untuk menginstal *package – package* yang diperlukan aplikasi *web*.

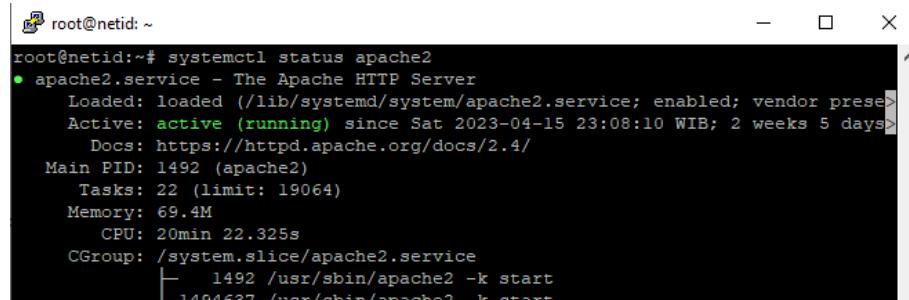


```
root@netid: ~
login as: root
root@103.147.46.235's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-69-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

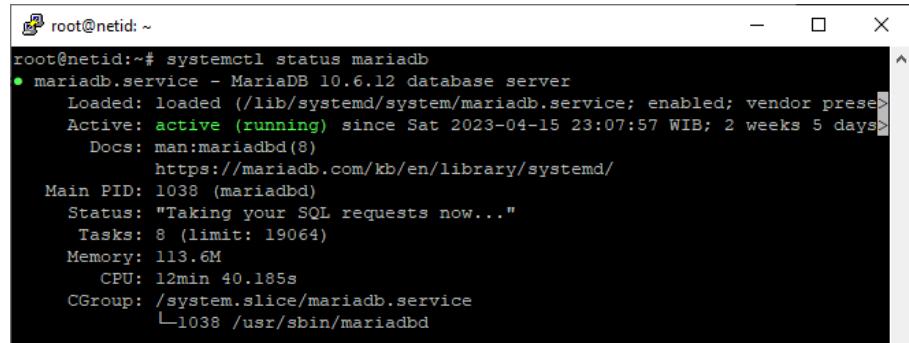
System information as of Fri May  5 12:34:05 PM WIB 2023
```

- c. *Package* pertama yang diinstal adalah Apache2 yang merupakan *package* yang memberikan layanan HTTP. Saya menjalankan perintah `apt-get install apache2`.



```
root@netid:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor pres>
   Active: active (running) since Sat 2023-04-15 23:08:10 WIB; 2 weeks 5 days>
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 1492 (apache2)
      Tasks: 22 (limit: 19064)
     Memory: 69.4M
        CPU: 20min 22.325s
      CGroup: /system.slice/apache2.service
              └─1492 /usr/sbin/apache2 -k start
                  ├─1494637 /usr/sbin/apache2 -k start
```

- d. Setelah *package* tersebut terinstal, jalankan perintah `systemctl status apache2` untuk mengecek apakah layanan Apache2 sudah berjalan. Jika sudah, bisa lanjut ke langkah berikutnya.
- e. *Package* yang selanjutnya diinstal adalah MariaDB. MariaDB adalah sebuah layanan *database* yang merupakan pengembangan mandiri dari MySQL. Jalankan perintah `apt-get install mariadb-server`.



```
root@netid:~# systemctl status mariadb
● mariadb.service - MariaDB 10.6.12 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor pres>
   Active: active (running) since Sat 2023-04-15 23:07:57 WIB; 2 weeks 5 days>
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
   Main PID: 1038 (mariadb)
      Status: "Taking your SQL requests now..."
      Tasks: 8 (limit: 19064)
     Memory: 113.6M
        CPU: 12min 40.185s
      CGroup: /system.slice/mariadb.service
              └─1038 /usr/sbin/mariadb
```

- f. Saya melakukan hal yang sama seperti *package* sebelumnya, yaitu mengecek apakah layanan sudah berjalan atau belum menggunakan `systemctl status`.
- g. Selanjutnya, instal *package* PHP versi terbaru. PHP ini nantinya akan digunakan untuk mengelola bahasa pemrograman PHP yang digunakan *framework* Laravel nanti. Jalankan perintah `apt-get install php`.
- h. Terakhir, instal *package* Composer. Composer adalah alat manajemen *dependency* untuk bahasa pemrograman PHP. Composer ini nantinya akan digunakan untuk menginstal *dependency framework* Laravel. Jalankan perintah `apt-get install composer`.

4. Instalasi Framework Laravel (oleh: Bintang)

Laravel adalah *framework* berbasis bahasa pemrograman PHP yang bisa digunakan untuk membantu proses pengembangan sebuah aplikasi *web* agar lebih maksimal. Laravel menggunakan konsep MVC (*Model-View-Controller*) yang dapat membantu *software developer* dalam pengembangan *software*-nya. MVC adalah sebuah pola arsitektur dalam membuat sebuah aplikasi dengan cara memisahkan kode menjadi tiga bagian yang terdiri dari:

a. Model

Bagian yang bertugas untuk menyiapkan, mengatur, memanipulasi, dan mengorganisasikan data yang ada di *database*.

b. View

Bagian yang bertugas untuk menampilkan informasi dalam bentuk GUI (*Graphical User Interface*).

c. Controller

Bagian yang bertugas untuk menghubungkan serta mengatur *Model* dan *View* agar dapat saling terhubung.



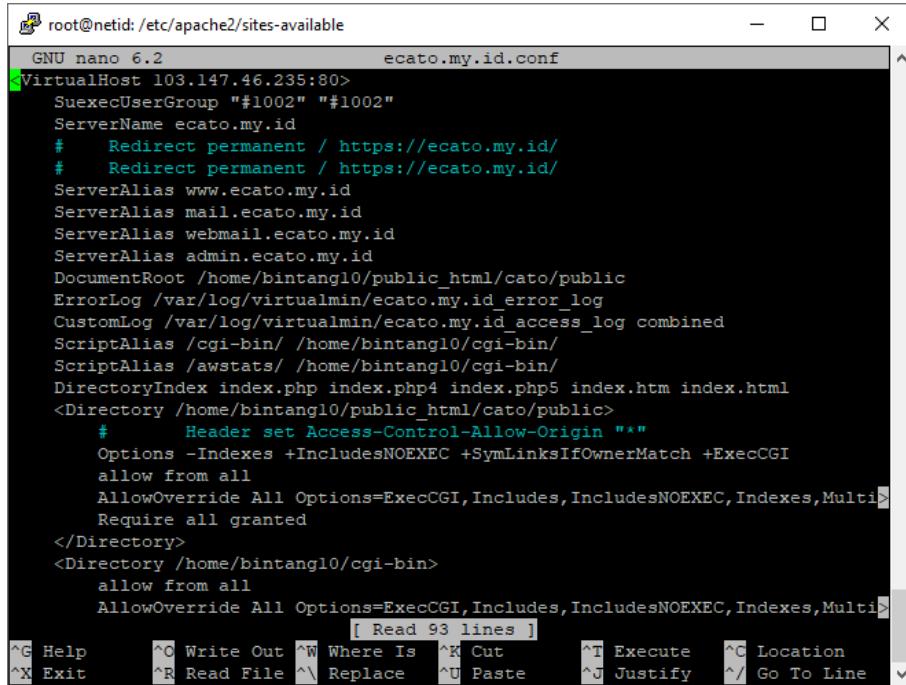
Gambar 3.14 Logo Laravel

Saya akan menggunakan Composer yang telah diinstal sebelumnya untuk menginstal *framework* Laravel. Berikut adalah langkah – langkahnya:

- a. *Login* sebagai *user* biasa di *server* karena di sini saya berpura – pura menjadi *user* yang menggunakan layanan *server* tersebut.
- b. Selanjutnya, masuk ke folder `public_html`. Kemudian saya menjalankan perintah `composer create-project laravel/laravel cato`.

```
Creating a "laravel/laravel" project at "./nama-projek"
Deprecation Notice: Using ${var} in strings is deprecated, use ${var} instead in
/usr/share/php/Composer/Autoload/AutoloadGenerator.php:879
Deprecation Notice: Using ${var} in strings is deprecated, use ${var} instead in
/usr/share/php/Composer/Autoload/AutoloadGenerator.php:884
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v10.1.1)
  - Downloading laravel/laravel (v10.1.1)
  - Installing laravel/laravel (v10.1.1): Extracting archive
Created project in /home/bintang10/public_html/nama-projek
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
```

- c. Sekarang, *login* sebagai *root*. Masuk ke folder `/etc/apache2/sites-available/`. Kemudian, buat *virtual host* Apache2 untuk aplikasi *web* Laravel. Buat file konfigurasi baru dengan nama `ecato.my.id.conf`.
- d. Isikan konfigurasi *virtual host*. Di sini saya menggunakan *domain* `ecato.my.id` yang saya dapat secara gratis beberapa waktu lalu.



```

root@netid: /etc/apache2/sites-available
GNU nano 6.2          ecato.my.id.conf
VirtualHost 103.147.46.235:80>
  SusecUserGroup "#1002" "#1002"
  ServerName ecato.my.id
  #   Redirect permanent / https://ecato.my.id/
  #   Redirect permanent / https://ecato.my.id/
  ServerAlias www.ecato.my.id
  ServerAlias mail.ecato.my.id
  ServerAlias webmail.ecato.my.id
  ServerAlias admin.ecato.my.id
  DocumentRoot /home/bintang10/public_html/cato/public
  ErrorLog /var/log/virtualmin/ecato.my.id_error_log
  CustomLog /var/log/virtualmin/ecato.my.id_access_log combined
  ScriptAlias /cgi-bin/ /home/bintang10/cgi-bin/
  ScriptAlias /awstats/ /home/bintang10/cgi-bin/
  DirectoryIndex index.php index.php4 index.php5 index.htm index.html
  <Directory /home/bintang10/public_html/cato/public>
    #       Header set Access-Control-Allow-Origin "*"
    Options -Indexes +IncludesNOEXEC +SymLinksIfOwnerMatch +ExecCGI
    allow from all
    AllowOverride All Options=ExecCGI,Includes,IncludesNOEXEC,Indexes,Multi
    Require all granted
  </Directory>
  <Directory /home/bintang10/cgi-bin>
    allow from all
    AllowOverride All Options=ExecCGI,Includes,IncludesNOEXEC,Indexes,Multi
    [ Read 93 lines ]
  ^G Help      ^O Write Out  ^W Where Is  ^K Cut      ^T Execute  ^C Location
  ^X Exit      ^R Read File  ^\ Replace   ^U Paste   ^J Justify  ^/ Go To Line

```

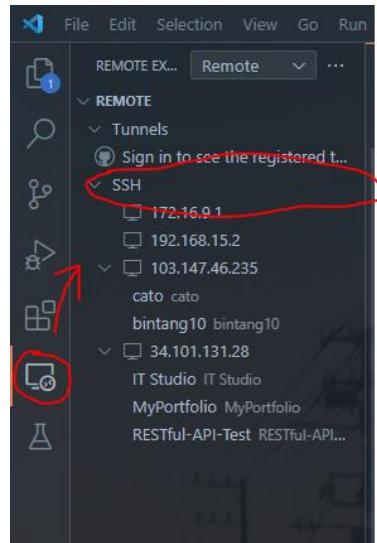
- e. Terakhir, jalankan perintah `a2ensite ecato.my.id` untuk menjalankan *virtual host* tersebut.

5. Pengembangan Back-end Web (oleh: Bintang)

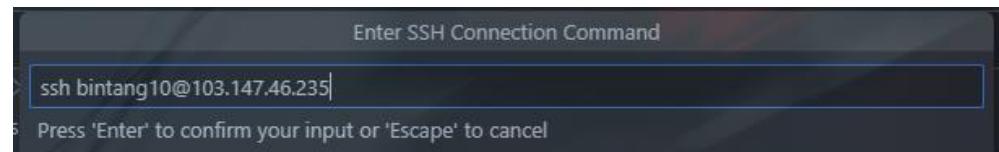
Istilah *back-end* dalam dunia pemrograman dapat diartikan sebagai bagian dari aplikasi yang bertanggung jawab untuk menyediakan kebutuhan yang tak terlihat oleh pengguna. Di sini saya akan menunjukkan langkah – langkah dalam pembuatan REST API dan layanan WebSocket untuk kebutuhan data IoT mesin penetas telur otomatis berbasis *web*.

- a. Di sini saya menggunakan laptop *client* yang istilahnya terpisah jauh dari *server*. Maka dari itu, untuk memprogram bagian *back-end*, saya akan menggunakan fitur Remote Explorer dari *software* Visual Studio Code untuk me-*remote* file – file projek Laravel yang berada di *server*. Visual Studio Code (atau disingkat VS Code) adalah aplikasi editor *source code* buatan Microsoft yang dapat dijalankan di semua perangkat *desktop* secara gratis.

- b. Buka VS Code. Klik menu Remote Explorer, lalu pada *drop down* SSH, klik + New Remote.



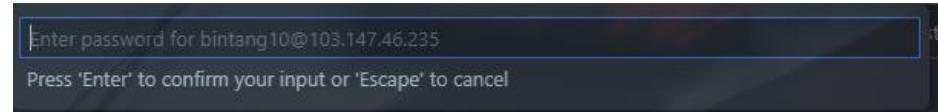
Masukkan *user* dan *IP address* untuk SSH ke *server*. Di sini saya meng (bintang10 sebagai *user login* dan 103.147.46.235 sebagai *IP address server*).



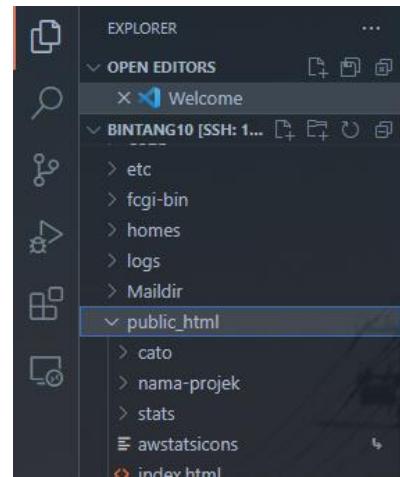
Opsi SSH baru akan muncul dalam list SSH.



Kemudian klik SSH 103.147.46.235, lalu *login* menggunakan *password* untuk *user* bintang10.



Buka folder projek Laravel yang telah dibuat tadi (public_html/cato).



- c. Buka file '.env' untuk membuka konfigurasi projek Laravel. Lalu konfigurasikan semua variabel *environment* yang dibutuhkan projek.

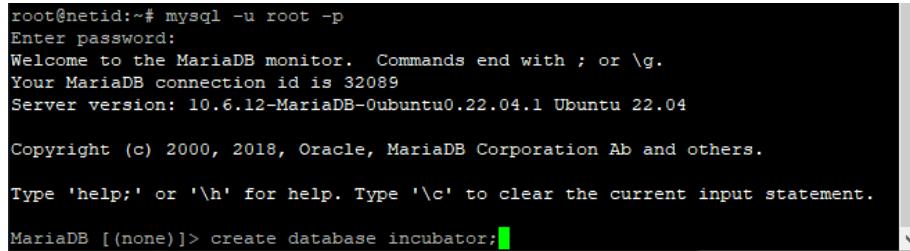
```

.env
1 APP_NAME=Cato
2 APP_ENV=local
3 APP_KEY=base64:N0u1ET7UAZsTrn/p8qvTXfLCDv15XTpRN3bMj8wtHUw=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LARAVEL_WEBSOCKETS_SSL_LOCAL_CERT="/home/bintang10/ssl.cert"
8 LARAVEL_WEBSOCKETS_SSL_LOCAL_PK="/home/bintang10/ssl.key"
9 LARAVEL_WEBSOCKETS_SSL_PASSPHRASE=
10
11 LARAVEL_WEBSOCKETS_PORT=6002
12
13 LOG_CHANNEL=stack
14 LOG_DEPRECATIONS_CHANNEL=null
15 LOG_LEVEL=debug
16
17 DB_CONNECTION=mysql
18 DB_HOST=127.0.0.1
19 DB_PORT=3306
20 DB_DATABASE=incubator
21 DB_USERNAME=bintang10
22 DB_PASSWORD=990705
23
24 BROADCAST_DRIVER=pusher
25 CACHE_DRIVER=file
26 FILESYSTEM_DRIVER=local
27 QUEUE_CONNECTION=sync
28 SESSION_DRIVER=file

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- d. Selanjutnya, sekarang *login* ke *server* menggunakan PuTTY. Buatlah database MySQL baru dengan nama *database* yang sesuai dengan konfigurasi projek tadi (incubator).



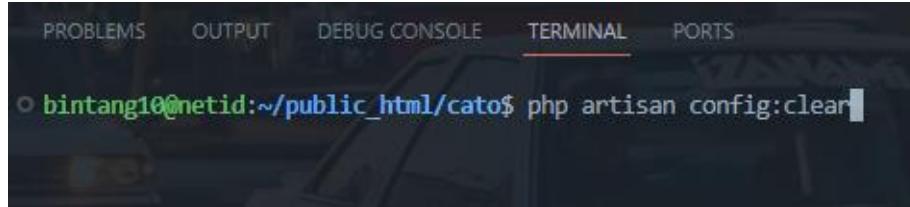
```
root@netid:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 32089
Server version: 10.6.12-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database incubator;
```

- e. Kembali ke VS Code, sekarang buka Terminal (Ctrl + Shift + `) lalu ketikkan `php artisan config:clear` untuk menghapus *cache* konfigurasi projek Laravel.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
bintang10@netid:~/public_html/cato$ php artisan config:clear
```

- f. Sekarang adalah tahap pembuatan *back-end* yang sebenarnya. Buat Model untuk data komponen mesin penetas telur. Jalankan perintah `php artisan make:model Komponen -a` pada Terminal. Kemudian buka folder `database/migrations/`. Folder tersebut berisi file – *file* yang disebut *Migrations*, yang merupakan salah satu fitur Laravel untuk menambah *table* secara otomatis ke *database*.
- g. Buka file *migration* yang telah otomatis dibuat. Isikan kolom – kolom yang diperlukan untuk *table* komponen seperti suhu, kelembapan, dan sebagainya.

```
database > migrations > 2023_05_05_144020_buat_tabel_komponen.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class BuatTabelKomponen extends Migration
8  {
9      /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14      public function up()
15      {
16          Schema::table('komponens', function (Blueprint $table) {
17              $table->integer('suhu');
18              $table->integer('kelembapan');
19              $table->integer('sisa_hari');
20              $table->integer('motor');
21              $table->integer('kipas');
22              $table->integer('lampa');
23              $table->integer('mode');
24              $table->integer('sistem');
25          });
26      }
27
28      /**
29      * Reverse the migrations.
30      *
31      * @return void
32      */
33      public function down()
34      {
35          Schema::table('komponens', function (Blueprint $table) {
36              $table->drop();
37          });
38      }
39  }
```

Kemudian, pada Terminal, jalankan perintah `php artisan migrate` untuk melakukan proses *Migration*. Setelah itu *table* otomatis muncul pada *database*. Selanjutnya saya mengisikan satu *record* secara manual lewat *SSH server*. *Record* inilah yang akan digunakan dan diperbarui secara terus – menerus kedepannya.

- h. Konfigurasikan *Model Komponen* yang berada pada 'App/Models/Komponen.php'. Kemudian, tambahkan 'protected \$fillable', agar setiap kolom pada *table* dapat diisi data nanti. Data – data komponen sekarang dapat dimanipulasi.

```
app > Models > Komponen.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Komponen extends Model
9  {
10     use HasFactory;
11
12     protected $fillable = ['suhu', 'kelembapan', 'sisa_hari', 'motor', 'kipas', 'lampa', 'mode', 'sistem'];
13
14 }
```

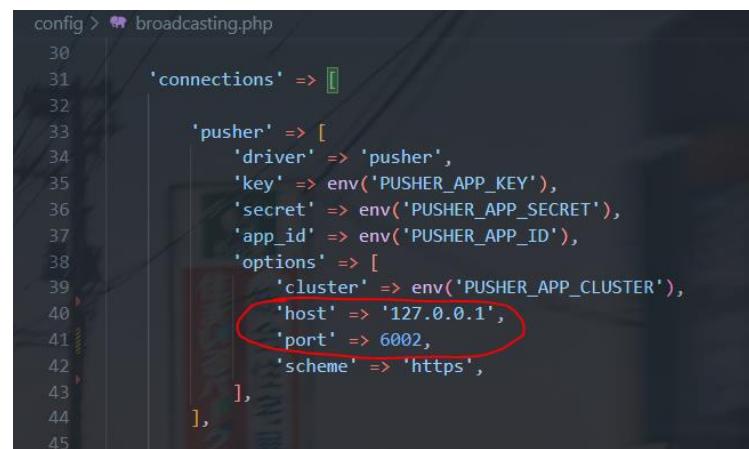
- i. Selanjutnya, saya akan membuat layanan WebSocket. Sebelum itu saya harus menginstal *package* Laravel WebSockets dengan menjalankan perintah `composer require beyondcode/laravel-

websockets` pada Terminal untuk menginstalnya. Setelah itu, jalankan perintah `php artisan migrate` lagi untuk membuat *table* untuk data – data layanan WebSocket. Selanjutnya, untuk membuat file konfigurasi Laravel WebSockets saya menjalankan perintah berikut di Terminal:

```
'php artisan vendor:publish --  
provider="BeyondCode\LaravelWebSockets\WebSocketsServiceProvider" --tag="config"
```

Berikutnya saya menambahkan satu *package* lagi, yaitu Pusher, *package* yang memudahkan untuk menghubungkan *client* WebSocket dengan *server* WebSocket. Saya menginstalnya dengan menjalankan perintah di Terminal: `composer require pusher/pusher-php-server`. Kemudian edit file `'.env'`, pada variabel BROADCAST_DRIVER, isikan `pusher`.

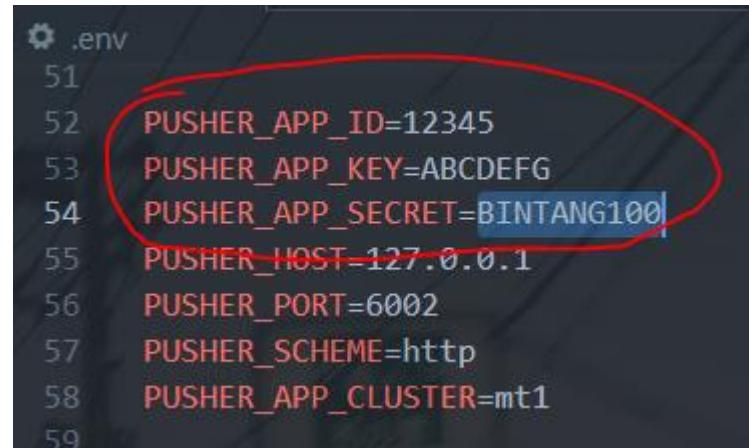
- j. Pada file `config/broadcasting.php`, saya menambahkan *host* 127.0.0.1 dan *port* 6002 pada bagian *options* agar komunikasi WebSocket dapat terhubung ke *server*. Sebelumnya, yang dimaksud dari konsep *Broadcasting* pada Laravel adalah konsep di mana sebuah *Event* atau mudahnya perubahan pada data – data yang ada di *server* disebarluaskan/disiarkan (*broadcast*) secara *real-time* ke seluruh *client* yang terhubung ke *server* tersebut melalui layanan WebSocket.



```
config > broadcasting.php
30
31     'connections' => [
32
33         'pusher' => [
34             'driver' => 'pusher',
35             'key' => env('PUSHER_APP_KEY'),
36             'secret' => env('PUSHER_APP_SECRET'),
37             'app_id' => env('PUSHER_APP_ID'),
38             'options' => [
39                 'cluster' => env('PUSHER_APP_CLUSTER'),
40                 'host' => '127.0.0.1',
41                 'port' => 6002,
42                 'scheme' => 'https',
43             ],
44         ],
45     ],

```

Buka lagi file ` `.env` , kemudian isikan beberapa variabel yang dilingkari di bawah ini dengan nilai yang unik.



```

.env
51
52 PUSHER_APP_ID=12345
53 PUSHER_APP_KEY=ABCDEFG
54 PUSHER_APP_SECRET=BINTANG100
55 PUSHER_HOST=127.0.0.1
56 PUSHER_PORT=6002
57 PUSHER_SCHEME=http
58 PUSHER_APP_CLUSTER=mt1
59

```

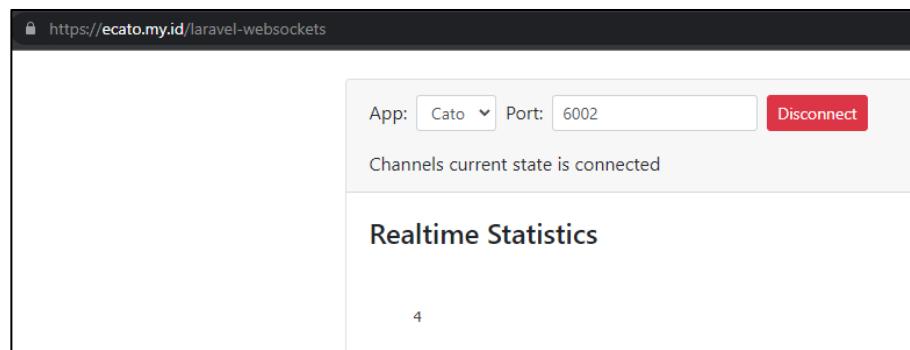
Terakhir dari proses instalasi dan konfigurasi WebSocket ini, jalankan perintah ` `php artisan websockets:serve --port 6002` ` untuk menghidupkan layanan WebSocket pada *port* 6002.

```

bintang10@netid:~/public_html/cato$ php artisan websockets:serve --port 6002
Starting the WebSocket server on port 6002...

```

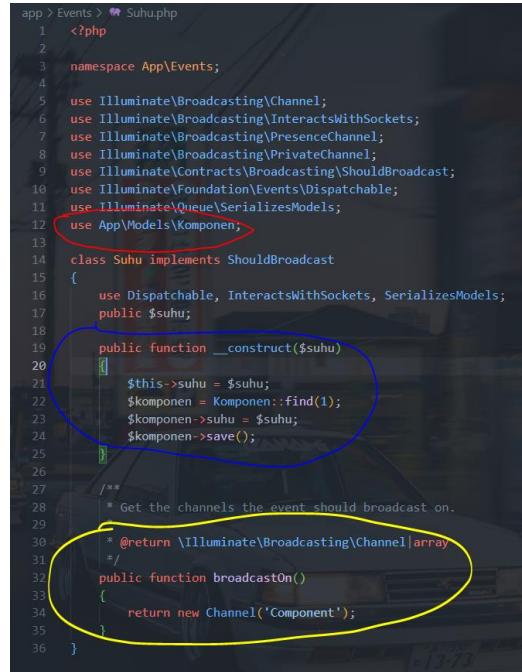
Untuk menguji coba apakah WebSocket benar – benar berjalan, saya menguji URL <https://ecato.my.id/laravel-websockets> dan mencoba untuk terhubung ke layanan WebSocket. Hasilnya, layanan WebSocket berjalan lancar dan dapat dikembangkan lagi.



k. Langkah selanjutnya, saya membuat sebuah program *Event* yang dapat menginformasikan secara *real-time* tentang perubahan data komponen seperti suhu, kelembapan, status lampu dan kipas, dan sebagainya. Berikut adalah perintah – perintah yang saya jalankan di Terminal:

```
'php artisan make:event Suhu'  
'php artisan make:event Kelembapan'  
'php artisan make:event Sistem'  
'php artisan make:event Mode'  
'php artisan make:event Motor'  
'php artisan make:event Kipas'  
'php artisan make:event Lampu'  
'php artisan make:event Penetasan'  
'php artisan make:event SisaHari'
```

Event Sistem di sini adalah data yang menunjukkan apakah sistem *monitor* dan kontrol penetas telur hidup atau mati. Sedangkan Penetasan adalah estimasi tanggal penetasan telur dan SisaHari adalah selisih hari saat *monitoring* dengan estimasi hari penetasan telur. Semua *Event* yang dibuat akan masuk ke folder 'App/Events'. Selanjutnya, saya mengedit file – file tersebut. Berikut adalah yang bagian – bagian yang saya edit.



```

1 <?php
2
3 namespace App\Events;
4
5 use Illuminate\\Broadcasting\\Channel;
6 use Illuminate\\Broadcasting\\InteractsWithSockets;
7 use Illuminate\\Broadcasting\\PresenceChannel;
8 use Illuminate\\Broadcasting\\PrivateChannel;
9 use Illuminate\\Contracts\\Broadcasting\\ShouldBroadcast;
10 use Illuminate\\Foundation\\Events\\Dispatchable;
11 use Illuminate\\Queue\\SerializesModels;
12 use App\\Models\\Komponen;
13
14 class Suhu implements ShouldBroadcast
15 {
16     use Dispatchable, InteractsWithSockets, SerializesModels;
17     public $suhu;
18
19     public function __construct($suhu)
20     {
21         $this->suhu = $suhu;
22         $komponen = Komponen::find(1);
23         $komponen->suhu = $suhu;
24         $komponen->save();
25     }
26
27     /**
28      * Get the channels the event should broadcast on.
29      *
30      * @return \Illuminate\\Broadcasting\\Channel|array
31      */
32     public function broadcastOn()
33     {
34         return new Channel('Component');
35     }
36 }

```

Bagian yang dilingkari merah wajib diberi karena di sini yang dimanipulasi datanya adalah *Model* Komponen. Bagian yang dilingkari biru merupakan program untuk menangkap *input* dari luar kemudian dimasukkan ke *class* Komponen untuk diperbarui. Sedangkan yang dilingkari kuning adalah penempatan *Channel* di mana *Broadcasting* akan dilakukan. Seperti halnya istilah *broadcasting* dalam kehidupan sehari – hari (seperti penyiaran televisi dan sebagainya), *Broadcasting* pada Laravel juga membutuhkan *channel* sebagai tempat penyiaran data.

1. Selanjutnya saya mengedit *Controller* yang berada di `App\Http\Controllers\KomponenController.php`. Pada bawah baris kode *namespace* tambahkan *class* PHP *Illuminate\Http\Request* agar Controller dapat melakukan permintaan (*request*) HTTP. Tambahkan juga *class* Model dan *class – class* Event yang dibuat tadi karena di sini saya menggunakan *Controller* untuk men-*trigger* semua *Event* tersebut.

```
app > Http > Controllers > KomponenController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\Komponen;
6  use Illuminate\Http\Request;
7  use Illuminate\Validation\Rule;
8  use App\Events\Suhu;
9  use App\Events\Kelembapan;
10 use App\Events\SisaHari;
11 use App\Events\Motor;
12 use App\Events\Kipas;
13 use App\Events\Lampu;
14 use App\Events\Mode;
15 use App\Events\Sistem;
16 use App\Events\Penetasan;
```

Selanjutnya saya membuat sebuah *function* yang saya beri nama ‘index’ untuk memuat data – data komponen.

```
class KomponenController extends Controller
{
    public function index(){
        return Komponen::find(1);
    }
}
```

Terakhir, saya membuat *function* ‘komponenUpdate’ untuk memperbarui data komponen. Saya membuat algoritma simpel di mana jika kondisi *request* data ada atau tidak kosong, maka akan dilakukan jenis *Event* yang sesuai dengan kolom *request*-nya, jika tidak, tidak akan terjadi apa – apa namun program akan melanjutkan ke kondisi *if* selanjutnya. Tujuan utama variabel ‘\$compRequest’ di sini hanyalah untuk memberi tahu berapa *request* yang telah dilakukan sekaligus men-debug *Controller* ini.

```

public function komponenUpdate(request $request){
    $compRequest = 0;
    if($request->sistem){
        event(new Sistem($request->sistem));
        $compRequest++;
    }
    if($request->mode){
        event(new Mode($request->mode));
        $compRequest++;
    }
    if($request->suhu){
        if($request->suhu < 0) $request->suhu = 0;
        event(new Suhu($request->suhu));
        $compRequest++;
    }
    if($request->kelembapan){
        if($request->kelembapan < 0) $request->kelembapan = 0;
        event(new Kelembapan($request->kelembapan));
        $compRequest++;
    }
    if($request->sisa_hari){
        if($request->sisa_hari < 0) $request->sisa_hari = 0;
        event(new SisaHari($request->sisa_hari));
        $compRequest++;
    }
    if($request->penetasan){
        event(new Penetasan($request->penetasan));
        $compRequest++;
    }
    if($request->motor){
        event(new Motor($request->motor));
        $compRequest++;
    }
    if($request->kipas){
        event(new Kipas($request->kipas));
        $compRequest++;
    }
    if($request->lampu){
        event(new Lampu($request->lampu));
        $compRequest++;
    }
    $response = "$compRequest data telah terupdate";
    return $response;
}

```

- m. Langkah selanjutnya adalah membuat rute atau *route*. *Route* inilah yang nantinya digunakan untuk mengakses REST API. Letak file-nya terdapat pada `routes/api.php`. Pertama, saya menambahkan *class Controller* yang sudah saya buat tadi.

```

routes > api.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\KomponenController;
5

```

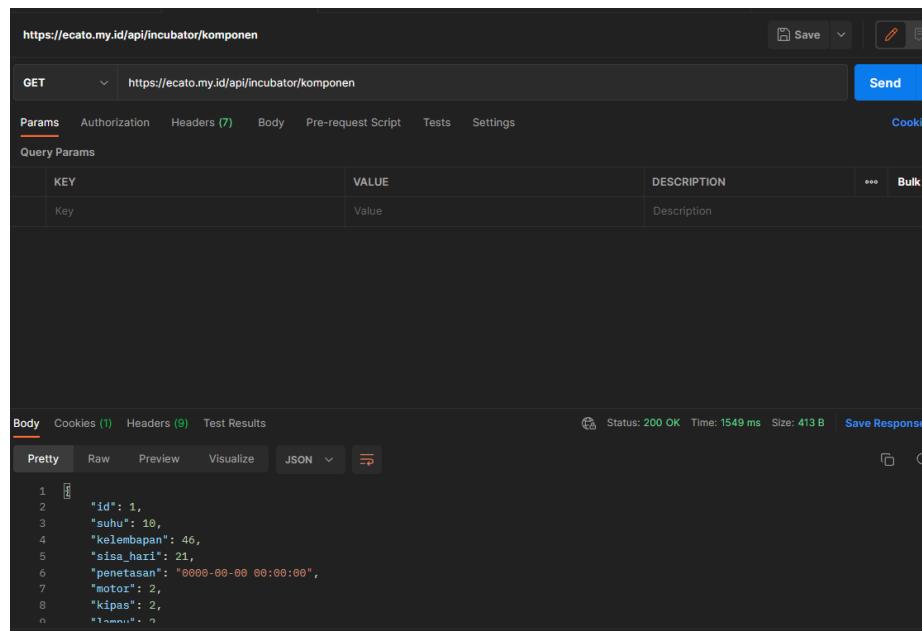
Kemudian, saya membuat rute dengan HTTP *method* *GET* dan *PUT*. Saya memanggil *function* dari *Controller* dan meletakkannya sesuai dengan HTTP *method* pada *route*-nya.

```

Route::get('/incubator/komponen', [KomponenController::class, 'index']);
Route::put('/incubator/komponen', [KomponenController::class, 'komponenUpdate']);

```

- n. Langkah terakhir dari pengembangan *back-end* ini adalah pengujian REST API. Saya menggunakan *software* Postman untuk mengujinya. Saya cukup meng-*input url* REST API dan memilih HTTP *method* yang ingin saya uji. Pertama saya menguji HTTP *GET* yang fungsinya untuk memuat data.



https://ecato.my.id/api/incubator/komponen

GET https://ecato.my.id/api/incubator/komponen

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk
Key	Value	Description		

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON

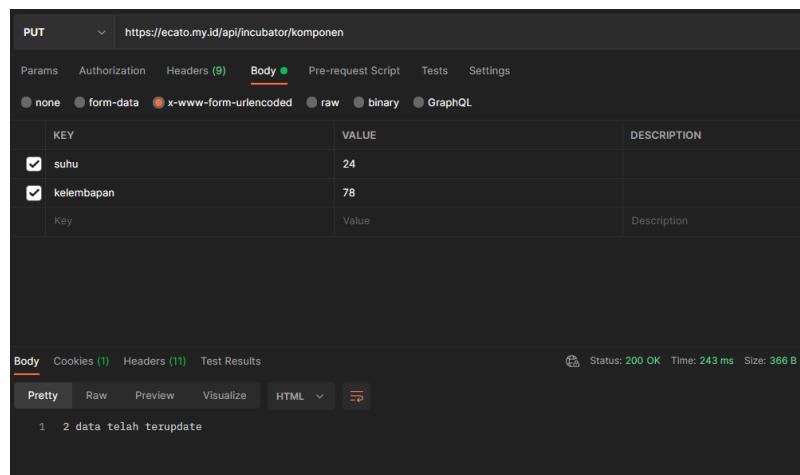
```

1
2   "id": 1,
3   "suhu": 10,
4   "kelembapan": 46,
5   "sisa_hari": 21,
6   "penetasan": "0000-00-00 00:00:00",
7   "motor": 2,
8   "kipas": 2,
9   "lampu": 2

```

Status: 200 OK Time: 1549 ms Size: 413 B Save Response

Pengujian HTTP *GET* sudah berhasil ditandai dengan 'Status: 200 OK' dan *response* data dari *server* dalam bentuk JSON. Terakhir, saya menguji HTTP *PUT* yang tujuannya untuk memperbarui data.



PUT https://ecato.my.id/api/incubator/komponen

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Body

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> suhu	24	
<input checked="" type="checkbox"/> kelembapan	78	
Key	Value	Description

Body Cookies (1) Headers (11) Test Results

Pretty Raw Preview Visualize HTML

```

1 2 data telah terupdate

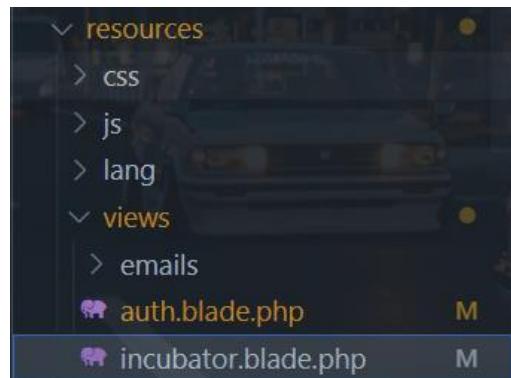
```

Status: 200 OK Time: 243 ms Size: 366 B

6. Pengembangan Front-end Web (oleh: Bintang, Gassa)

Berkaitan dengan proses pengembangan sebuah aplikasi *web*, *front-end* adalah bagian *web* yang berhadapan langsung dengan para pengunjung *web*. Mudahnya, *front-end* adalah bagian yang dapat menciptakan tampilan *web* yang responsif dan menarik. Pada umumnya, bahasa pemrograman yang digunakan untuk membuat *front-end* adalah HTML, CSS, dan JavaScript. Untuk pengembangan *front-end*, kami juga menggunakan ketiga bahasa tersebut, dengan tambahan *library* untuk JavaScript, yaitu jQuery, untuk mempermudah memprogram kode JavaScript. Tahap ini juga merupakan pengembangan *View* dari konsep MVC. Laravel memiliki fitur *templating* PHP tersendiri untuk pengembangan *View* yang disebut Blade. Berikut adalah langkah – langkahnya:

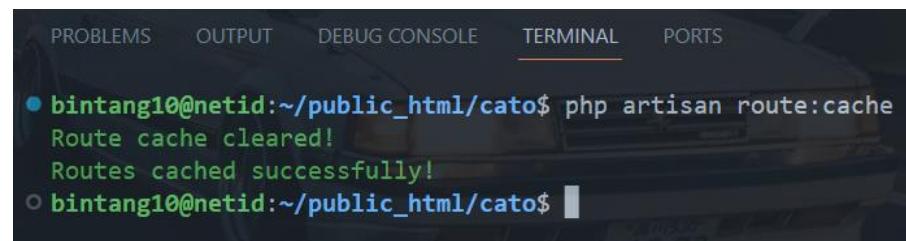
- Untuk membuat file – file *front-end* di projek Laravel, penempatan file harus sesuai aturan. File Blade terletak di `resources/views`, sedangkan file – file CSS dan JavaScript terletak di folder `public`. Buat file Blade baru dengan nama `incubator.blade.php`.



Selanjutnya, tambahkan *route* untuk menampilkan *View* `incubator` pada `routes/web.php`.

```
routes > 🏷️ web.php
26
27     Route::get('/incubator', function() {
28         return view('incubator');
29     );
30
```

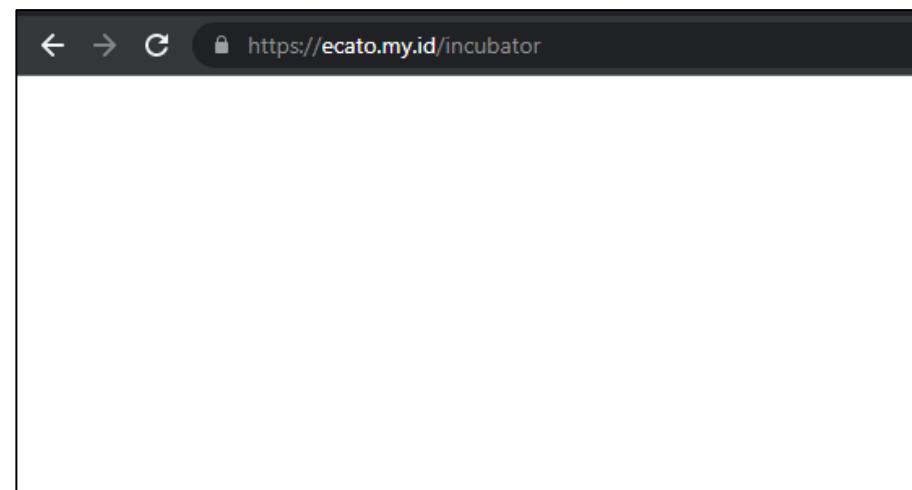
Artinya, ketika URL `https://ecato.my.id/incubator` dibuka, maka akan menampilkan *View* 'incubator'. Program tersebut sudah otomatis mengetahui file Blade mana yang di maksud 'tanpa menambahkan ekstensi file-nya (incubator.blade.php). Untuk melihat perubahan pada *route – route* untuk *View* maupun REST API, selalu jalankan perintah '`php artisan route:cache`' di Terminal untuk menghapus *cache route* Laravel, karena jika tidak, Laravel hanya akan menggunakan *cache* yang lama, alhasil, tidak akan menampilkan perubahan apapun.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● bintang10@netid:~/public_html/cato$ php artisan route:cache
  Route cache cleared!
  Routes cached successfully!
○ bintang10@netid:~/public_html/cato$
```

Selanjutnya, uji coba URL tadi menggunakan *web browser*. Jika berhasil, muncul tampilan dengan *background* putih polos karena *View* belum diisi apapun sama sekali, jika tidak berhasil, maka muncul tampilan *web* dengan tulisan '404 Not Found'. Jika sudah berhasil, maka lanjut ke langkah berikutnya.



- b. Buka file 'incubator.blade.php'. Tambahkan judul dan ikon *web* pada bagian *head*. Untuk ikon *web*, kami menggunakan desain ikon yang

dibuat di Figma sebelumnya. Kami cukup meng-*ekspor* gambarnya lalu meng-*upload* gambarnya ke folder `public/img`.

```
resources > views > 🗂 incubator.blade.php
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="icon" type="image/x-icon" href="/img/favicon.ico">
8      <title>Cato | Incubator Monitor</title>
9  </head>
```

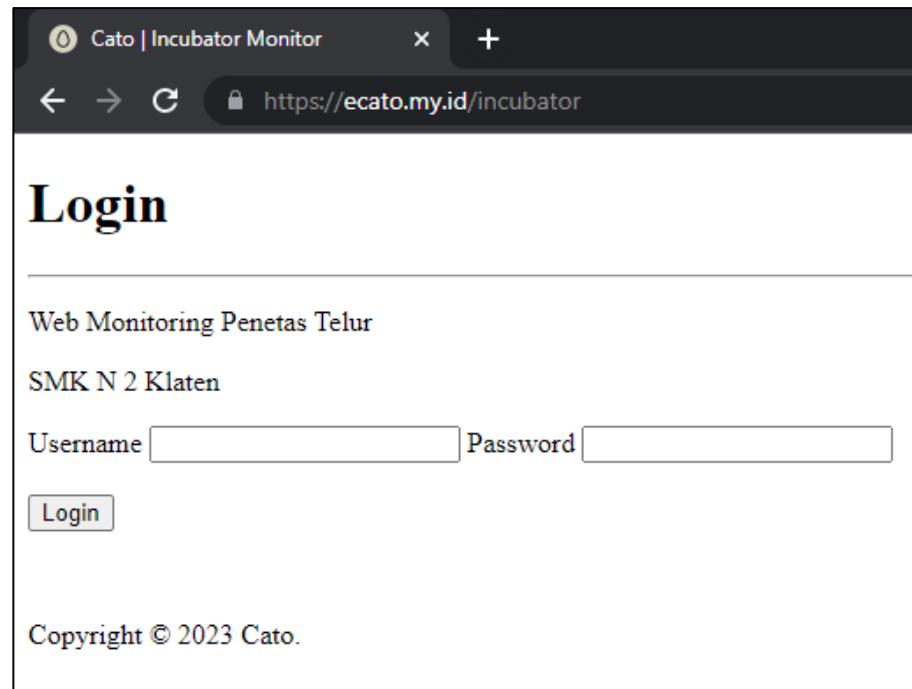
Desain – desain yang sebelumnya dibuat di Figma dijadikan patokan selama pembuatan tampilan *web*, termasuk struktur *web*. Struktur pertama adalah tampilan *loading*. Saat pertama kali dibuka, aplikasi *web* pada umumnya akan memuat data – data yang dibutuhkan yang kemudian dimasukkan ke *element – element web* yang bersangkutan secara dinamis, salah satu contohnya adalah memuat data nama lengkap *user* yang kemudian ditampilkan pada *dashboard*. Secara umum, tujuan dari tampilan *loading* adalah untuk mengindikasikan apakah *web* sudah selesai memuat data atau belum. Selain itu, *loading* juga berfungsi untuk menyembunyikan tampilan *web* yang sedang me-*render* di belakang layar atau perubahan tampilan *web* yang terlihat tidak wajar bagi pengguna *web* yang awam.

```
resources > views > 🗂 incubator.blade.php
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="icon" type="image/x-icon" href="/img/favicon.ico">
8      <title>Cato | Incubator Monitor</title>
9  </head>
10 <body>
11     <div id="loader" class="center"></div>
12     <div class="loading">
13     </div>
14 </body>
15 </html>
```

- c. Kemudian kami menambahkan struktur halaman *login*. Halaman autentikasi ini bertujuan untuk menghindari *web* dari pengguna tidak bertanggung jawab yang berpotensi merusak sistem.

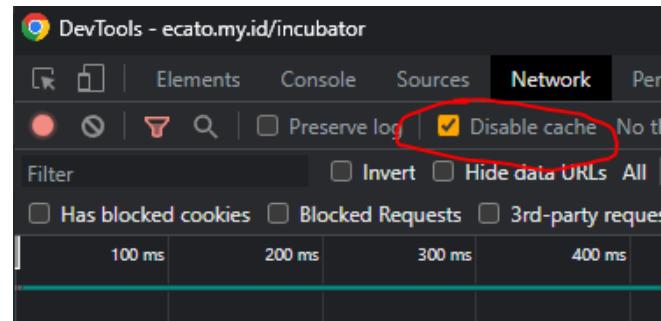
```
<div class="loading">
</div>
<div class="page1">
  <div class="login">
    <h1>Login</h1>
    <hr>
    <p class="login-text-atas">Web Monitoring Penetas Telur</p>
    <p class="login-text-bawah">SMK N 2 Klaten</p>
    <div class="form-input">
      <label class="ipt-txt">Username</label>
      <input class="ipt" type="text" id="user-ipt">
      <label class="ipt-txt">Password</label>
      <input class="ipt" type="password" id="pass-ipt">
    </div><br>
    <button class="login-button" id="signin" type="button">Login</button>
    <p class="login-info">&ampnbsp</p>
    <p class="copyright">Copyright © 2023 Cato.</p>
  </div>
</div>
```

Seperti inilah tampilan sementara dari halaman *login*.

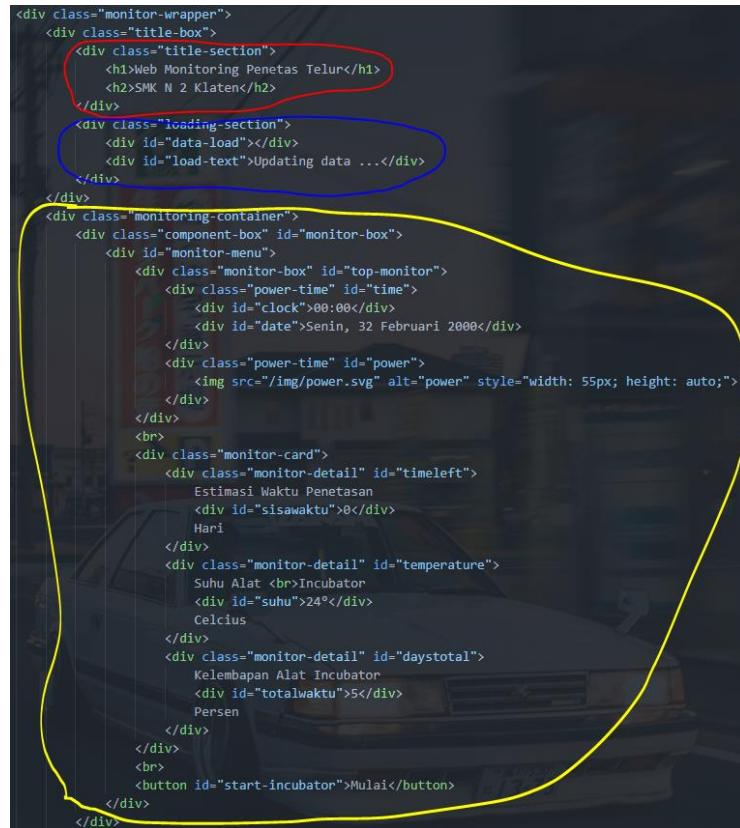


Setiap kali melakukan perubahan pada tampilan *web*, diharuskan untuk menghapus *cache web* tersebut untuk melihat perubahannya secara

total. Cara kami melakukannya adalah dengan mengeklik kanan pada halaman *web*, lalu pilih *Inspect*. Selanjutnya, pada *Network*, centang *Disable cache*. Cara ini dilakukan agar setiap kali me-*refresh* halaman *web*, *cache* akan otomatis terhapus dengan sendirinya.



- d. Selanjutnya kami membuat struktur untuk halaman utama *web*. Halaman utama ini dibagi menjadi dua panel, yaitu panel *monitor* dan panel kontrol. Panel *monitor* terdiri dari suhu, kelembapan, sisa hari penetasan telur, dan tombol untuk menghidupkan dan mematikan sistem. Sedangkan panel kontrol terdiri dari tombol mode dan tombol – tombol untuk mengontrol komponen. Kami membuat panel *monitor* terlebih dahulu. Pada gambar di bawah ini, bagian yang dilingkari warna merah adalah judul halaman. Bagian yang dilingkari warna biru adalah kotak *loading* kecil untuk mengindikasikan data yang diperbarui. Sedangkan bagian yang dilingkari warna kuning adalah panel *monitor*.

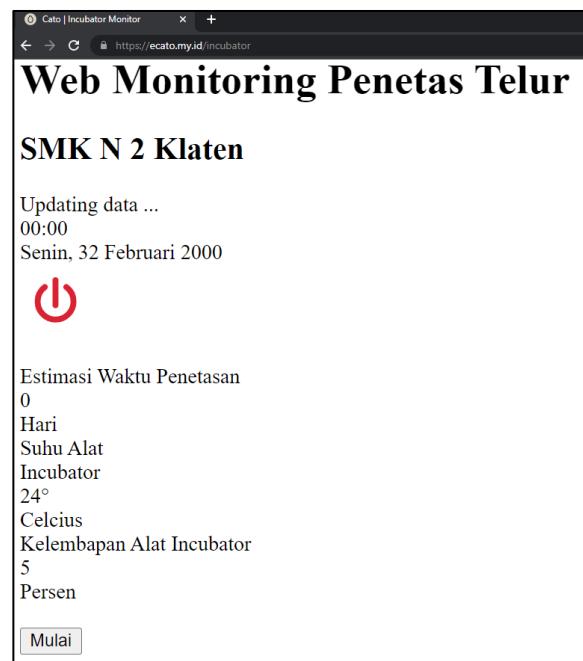


```

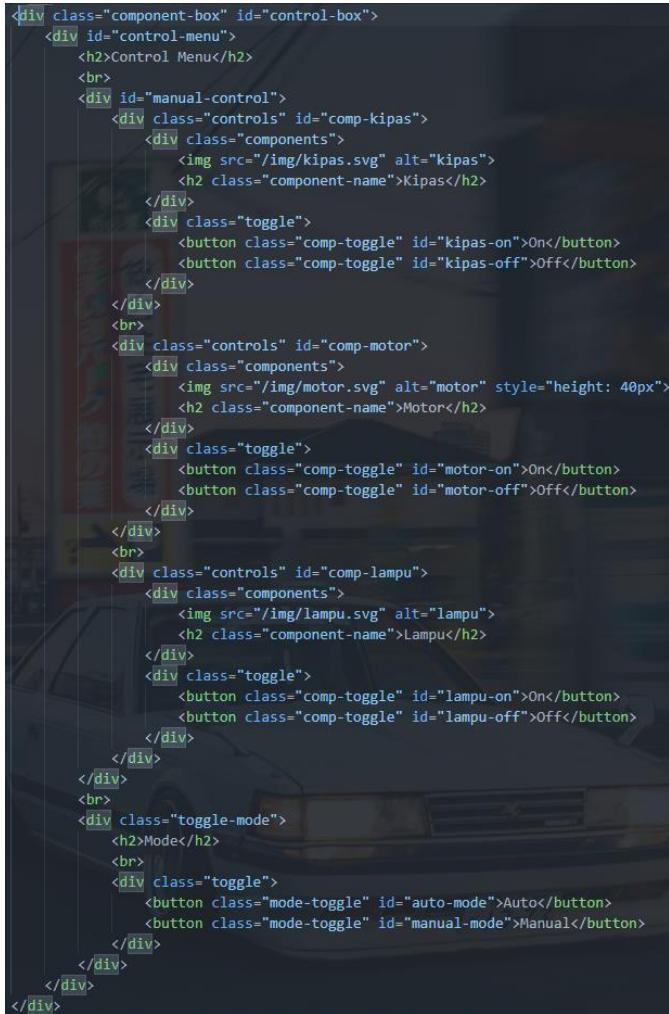
<div class="monitor-wrapper">
  <div class="title-box">
    <div class="title-section">
      <h1>Web Monitoring Penetas Telur</h1>
      <h2>SMK N 2 Klaten</h2>
    </div>
    <div class="loading-section">
      <div id="data-load"></div>
      <div id="load-text">Updating data ...</div>
    </div>
  </div>
  <div class="monitoring-container">
    <div class="component-box" id="monitor-box">
      <div id="monitor-menu">
        <div class="monitor-box" id="top-monitor">
          <div class="power-time" id="time">
            <div id="clock">00:00</div>
            <div id="date">Senin, 32 Februari 2000</div>
          </div>
          <div class="power-time" id="power">
            
          </div>
          <br>
        <div class="monitor-card">
          <div class="monitor-detail" id="timeleft">
            Estimasi Waktu Penetasan
            <div id="sisawaktu">0</div>
            Hari
          </div>
          <div class="monitor-detail" id="temperature">
            Suhu Alat <br>Incubator
            <div id="suhu">24°</div>
            Celcius
          </div>
          <div class="monitor-detail" id="daystotal">
            Kelembapan Alat Incubator
            <div id="totalwaktu">5</div>
            Persen
          </div>
          <br>
          <button id="start-incubator">Mulai</button>
        </div>
      </div>
    </div>
  </div>

```

Seperti inilah tampilan sementara dari panel *monitor*.

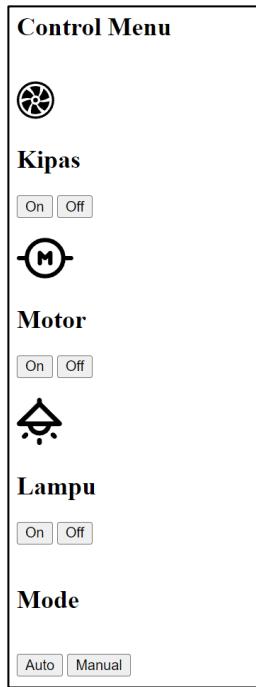


- e. Kemudian, kami membuat struktur panel kontrol. Kami mengambil ikon – ikon yang dibutuhkan dari desain yang telah dibuat di lembar kerja Figma.



```
<div class="component-box" id="control-box">
  <div id="control-menu">
    <h2>Control Menu</h2>
    <br>
    <div id="manual-control">
      <div class="controls" id="comp-kipas">
        <div class="components">
          
          <h2 class="component-name">Kipas</h2>
        </div>
        <div class="toggle">
          <button class="comp-toggle" id="kipas-on">On</button>
          <button class="comp-toggle" id="kipas-off">Off</button>
        </div>
      </div>
      <br>
      <div class="controls" id="comp-motor">
        <div class="components">
          
          <h2 class="component-name">Motor</h2>
        </div>
        <div class="toggle">
          <button class="comp-toggle" id="motor-on">On</button>
          <button class="comp-toggle" id="motor-off">Off</button>
        </div>
      </div>
      <br>
      <div class="controls" id="comp-lampu">
        <div class="components">
          
          <h2 class="component-name">Lampu</h2>
        </div>
        <div class="toggle">
          <button class="comp-toggle" id="lampu-on">On</button>
          <button class="comp-toggle" id="lampu-off">Off</button>
        </div>
      </div>
      <br>
      <div class="toggle-mode">
        <h2>Mode</h2>
        <br>
        <div class="toggle">
          <button class="mode-toggle" id="auto-mode">Auto</button>
          <button class="mode-toggle" id="manual-mode">Manual</button>
        </div>
      </div>
    </div>
  </div>
</div>
```

Gambar di bawah ini adalah tampilan sementara dari panel kontrol.



Begitulah tahap pembuatan struktur *web*. Selanjutnya adalah pemberian *style* menggunakan CSS untuk struktur – strukturnya.

- f. Buat file dengan nama 'style.css' di folder 'public/css'. Kemudian pada 'incubator.blade.php', tambahkan *link* pada bagian *head* untuk CSS *external* yang telah dibuat tadi.

```

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="icon" type="image/x-icon" href="/img/favicon.ico">
  <link rel="stylesheet" href="/css/style.css">
  <title>Cato | Incubator Monitor</title>
</head>

```

- g. Pertama, ubah *margin* dan *padding* untuk semua *element* menjadi 0. Teknik ini sering disebut 'CSS Reset', yang tujuannya agar *web* tidak terikat dengan aturan *default* CSS.

```

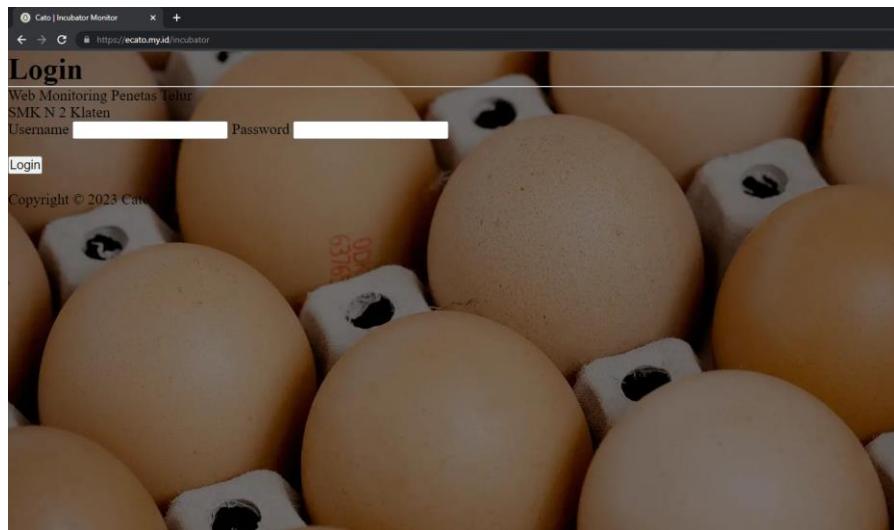
public > css > # style.css > ...
1  *{
2    margin: 0;
3    padding: 0;
4  }

```

- h. Tambahkan gambar *background* sesuai dengan desain Figma.

```
html{
    background-image: url('../img/eggback.webp');
    background-repeat: no-repeat;
    background-size: cover;
    background-color: #EEEAD7;
    background-position: center center;
    background-attachment: fixed;
}
```

Hasilnya, tampilan *web* akan berubah seperti di bawah ini.



- i. Tambahkan *font* pilihan dari desain yang nantinya digunakan dalam *styling* setiap *element* berbentuk teks.

```
@font-face {
    font-family: InriaSans;
    src: url("/fonts/InriaSans-Regular.ttf");
}

@font-face {
    font-family: InriaSans;
    src: url("/fonts/InriaSans-Bold.ttf");
    font-weight: bold;
}

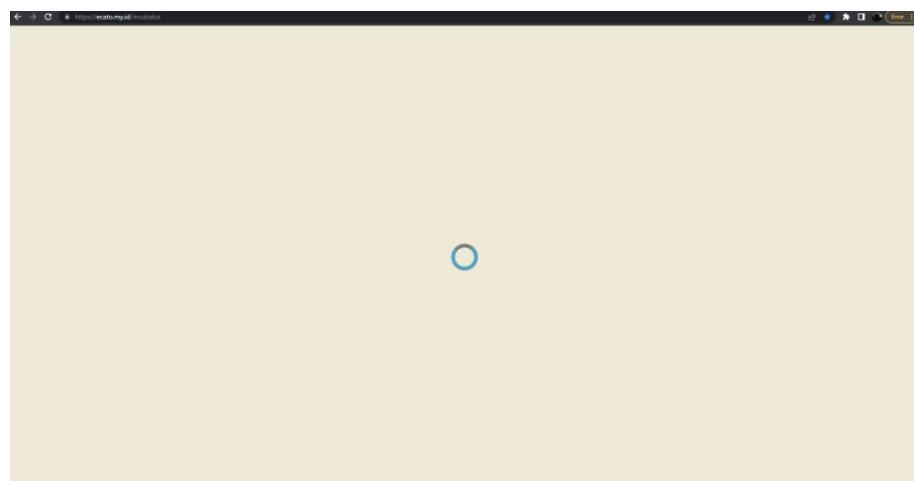
@font-face {
    font-family: InriaSans;
    src: url("/fonts/InriaSans-Light.ttf");
    font-weight: lighter;
}
```

- j. Selanjutnya, kami membuat *style* untuk tampilan *loading* dan halaman *login*. Berikut adalah CSS untuk tampilan *loading*.

```
#loader {  
    border: 8px solid #56A1C0;  
    border-radius: 50%;  
    border-top: 8px solid gray;  
    width: 40px;  
    height: 40px;  
    animation: spin 1s linear infinite;  
    z-index: 10;  
}
```

```
.loading{  
    background-color: #EEEAD7;  
    height: 100%;  
    width: 100%;  
    left: 0;  
    top: 0;  
    position: fixed;  
    overflow: hidden;  
    z-index: 9;  
}
```

Hasilnya, saat pertama mengakses *web* akan muncul tampilan *loading* yang simpel seperti ini.



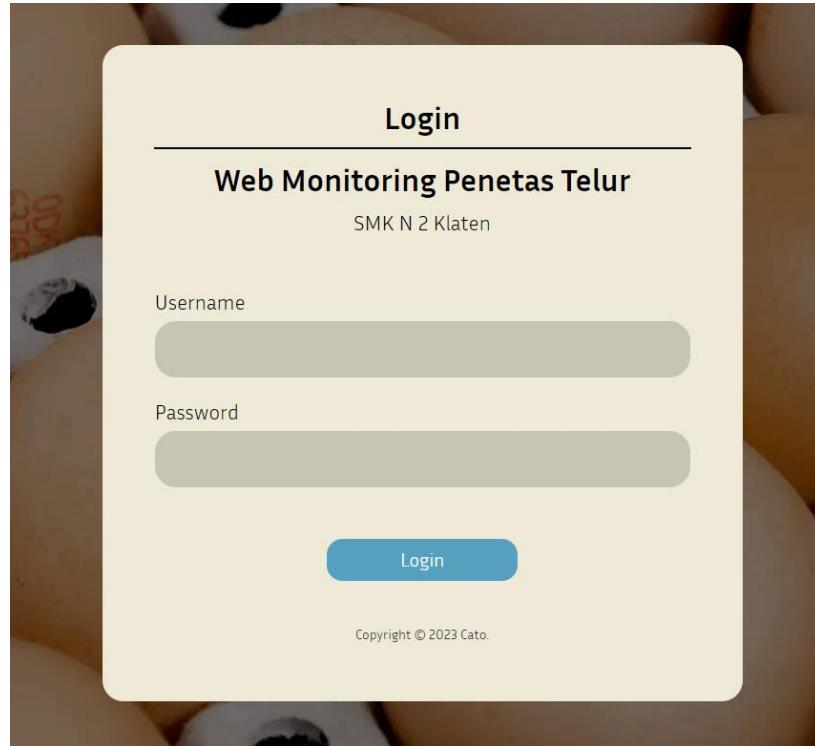
Kemudian kami menambahkan *style* untuk halaman *login*. Pertama, kami menengahkan *element – element* yang ada pada kotak *login* dan memberinya warna *background* dengan CSS berikut.

```
.login{  
    display: flex;  
    flex-direction: column;  
    justify-content: center;  
    align-items: center;  
    background: #EEEAD7;  
    border-radius: 20px;  
    padding: 50px;  
}
```

Kemudian kami beri *style* untuk *input* dan tombol *login* seperti ini.

```
input{  
    border: none;  
    outline: none;  
    border-radius: 20px;  
    background: #C7C4B3;  
    width: 450px;  
    height: 24px;  
    display: block;  
    font-family: InriaSans;  
    font-weight: lighter;  
    margin-top: 5px;  
    margin-bottom: 20px;  
}  
  
input[type=text] [type=password]{  
    padding-left: 20px;  
    padding-right: 20px;  
}  
  
.login-button{  
    background-color: #56A1C0;  
    border-radius: 20px;  
    text-align: center;  
    font-family: InriaSans;  
    color: white;  
    font-size: 20px;  
    border: none;  
    outline: none;  
    line-height: 50px;  
    padding: 0px 100px 0px 100px;  
}
```

Hasil akhirnya, halaman *login* akan terlihat seperti ini.



- k. Selanjutnya adalah *styling* untuk halaman utama. Pertama, kami beri *style* untuk *wrapper* halaman utama sehingga bisa terletak di tengah halaman.

```
.monitoring-container{
  padding-top: 50px;
  display: flex;
  justify-content: center;
  gap: 40px;
  height: 650px;
}
```

Kemudian kami memberi *style* untuk judul halaman dan panel *monitor* seperti ini.

```
.title-box{
  align-self:flex-start;
  display: flex;
  justify-content: space-between;
  align-items: center;
  width: 100%;
}
```

```
#monitor-menu{  
    display: flex;  
    flex-direction: column;  
    justify-content: center;  
    align-items: center;  
    gap: 10px;  
    height: 100%;  
}  
.monitor-card{  
    width: 780px;  
    display: flex;  
    justify-content: space-between;  
}  
.monitor-box{  
    display: flex;  
    justify-content: space-between;  
    align-items: center;  
    gap: -50px;  
    width: 800px;  
}  
.monitor-detail{  
    padding-top: 24px;  
    padding-bottom: 16px;  
    border-radius: 20px;  
    line-height: 1.5;  
    font-family: InriaSans;  
    font-weight: bold;  
    font-size: 28px;  
    color: white;  
    text-align: center;  
    display: inline-block;  
    width: 250px;  
    height: fit-content;  
}  
#sisawaktu, #suhu, #totalwaktu{  
    margin: 7px 0 7px 0;  
    font-size: 110px;  
}
```

Panel *monitor* yang sudah diberi *style* terlihat seperti berikut.



Selanjutnya, kami memberi *style* untuk panel kontrol.

```
#control-menu{
    font-family: InriaSans;
    color: black;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    gap: 20px;
    height: 100%;
}

.controls{
    width: 90%;
    display: flex;
    flex-direction: row;
    justify-content: space-between;
    align-items: center;
    text-align: justify;
}

.components{
    display: flex;
    flex-direction: row;
    justify-content: center;
    gap: 10px;
}

.toggle{
    display: flex;
    flex-direction: row;
    justify-content: center;
    gap: 8px;
}
```

Inilah hasil akhir dari tampilan halaman utama.



- l. Setelah *styling*, langkah selanjutnya adalah membuat halaman *web* menjadi responsif, interaktif, dan dinamis menggunakan JavaScript. Kami membuat file JavaScript bernama 'incubator.js' di folder 'public/js'. Untuk library jQuery, kami sudah mendownloadnya sebelumnya sehingga cukup kami letakkan di folder yang sama juga. Untuk menambahkan *script* JavaScript pada Blade, kami menambahkan *tag script* berikut di bagian paling bawah *body*.

```

<script src="/js/jquery-3.6.1.min.js"></script>
<script src="/js/incubator.js"></script>
</body>
</html>

```

- m. Pertama, kami menambahkan program JavaScript agar tampilan *loading* dan halaman *login* berfungsi. Namun, tampilan *web* belum dapat dimanipulasi sebelum semua *element – element*-nya termuat sepenuhnya. Maka dari itu, awal program JavaScript wajib diberi sebuah *function* yang dapat mengindikasikan bahwa *web* sudah termuat sepenuhnya. jQuery sudah memiliki *built-in function*-nya tersendiri yang memiliki fungsi yang sama. Berikut adalah programnya.

```

public > js > js incubatorjs > ...

```

```

1 | $(document).ready(function() {
2 | // ...
3 | });

```

Setelah itu, kami membuat beberapa baris kode yang bisa menjalankan tampilan *loading* dengan tepat serta menyembunyikan halaman utama sebelum bisa diakses.

```

$(".monitor-wrapper").hide();
$(".loading-section").fadeOut();
$("#loader").delay(1000).fadeOut(400);
$(".loading").delay(1000).fadeOut(400);
$(".page1").delay(1000).fadeIn(400).show();

```

Selanjutnya, kami membuat sebuah *function* untuk tombol *login*. Isi program tersebut yaitu memvalidasi *input login*, jika *input* yang dimasukkan benar maka halaman utama akan dimunculkan dan halaman *login* akan disembunyikan. Di sini kami hanya membuat sistem *login* yang simpel tanpa memerlukan data yang kompleks seperti aplikasi *web* pada umumnya, yaitu hanya dengan mengecek apakah *input username* berisikan 'Bintang' dan *input password* berisikan '12345', jika semua kondisi terpenuhi, maka halaman utama dapat diakses. Walaupun begitu, pendekatan ini sangatlah tidak aman karena informasi *login* tersebut dapat terkuak dengan mudah ketika seorang *user* melakukan **Inspect** terhadap JavaScript yang dimiliki *web*.

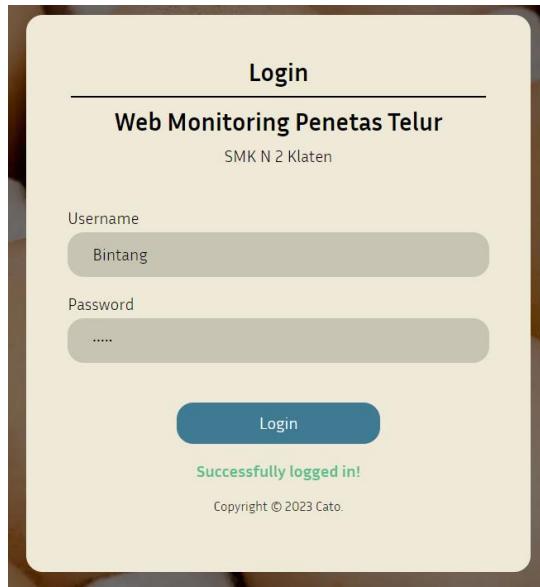
```
function loginButton(){
  let userInput = $("#user-ipt");
  let passInput = $("#pass-ipt");
  let userValue = userInput.value.trim();
  let passValue = passInput.value.trim();

  if(userValue.length == 0 || passValue.length == 0){
    $(".login-info").text("Please insert username and password!");
    return false;
  } else if(userInput.value == "Bintang" && passInput.value == "12345") {
    $(".login-info").css('color', '#56C0BD');
    $(".login-info").text("Successfully logged in!");
    getData();
    $("#loader").delay(500).fadeIn(400);
    $(".loading").delay(500).fadeIn(400);
    setTimeout(function() {
      $(".page1").hide();
      $(".monitor-wrapper").show();
    }, 1000);
  } else {
    $(".login-info").text("Incorrect username or password!");
    return false;
  }
}
```

Kemudian kami memasukkan *function* tersebut ke *attribute onclick* tombol *login*.

```
<button class="login-button" onclick="loginButton()" id="signin" type="button">Login</button>
```

Beginilah hasil akhirnya. Halaman utama sekarang bisa diakses.



- n. Sebelum halaman utama ditampilkan sepenuhnya, kami menambahkan sebuah *function* bernama 'getData' untuk memuat data – data komponen dari REST API menggunakan *method* AJAX. AJAX (*Asynchronous Javascript and XML*) mengacu pada sekumpulan teknis pengembangan *web* yang memungkinkan aplikasi *web* untuk bekerja secara *asynchronous* (tidak langsung) dan memproses setiap permintaan HTTP ke *server* di balik layar. Function yang kami buat ini mengambil data – data dari REST API yang telah dibuat sebelumnya yang kemudian dimuat ke *element – element* HTML. Dalam *function* ini ada juga *function* lain yang dipanggil yaitu 'updateButton'. *Function* tersebut kami buat sebelumnya untuk mengubah status tombol – tombol kontrol, contohnya jika komponen kipas menyala maka tombol kipas juga akan menyala.

```

function getData(){
  $.ajax({
    url:'https://ecato.my.id/api/incubator/komponen',
    type:"GET",
    async:true,
    dataType:"json",
    success: function(data){
      $("#loader").delay(1000).fadeOut(400);
      $(".loading").delay(1000).fadeOut(400);
      suhu = data.suhu;
      kelembapan = data.kelembapan;
      motor = data.motor;
      kipas = data.kipas;
      lampu = data.lampu;
      mode = data.mode;
      sistem = data.sistem;
      penetasan = data.penetasan;
      sisa_hari = data.sisa_hari;

      $("#suhu").text(suhu + "°");
      $("#totalwaktu").text(kelembapan);
      if(sistem == 1){
        updateButton("sistem",1);
      } else if(sistem == 2){
        if(sisa_hari > 0){
          startDate = new Date(penetasan);
          lastEstimation = sisa_hari;
          dayCountdown("enable");
        }
        updateButton("sistem",2);
        $("#sisawaktu").text(sisa_hari);
        if(motor > 0){
          if(motor == 1){
            updateButton("motor",1);
          } else if(motor == 2){
            updateButton("motor",2);
          }
        }
        if(kipas > 0){
          if(kipas == 1){
            updateButton("kipas",1);
          } else if(kipas == 2){
            updateButton("kipas",2);
          }
        }
        if(lampu > 0){
          if(lampu == 1){
            updateButton("lampu",1);
          } else if(lampu == 2){
            updateButton("lampu",2);
          }
        }
        if(mode > 0){
          if(mode == 1){
            updateButton("mode",1);
          } else if(mode == 2){
            updateButton("mode",2);
          }
        }
      }
    });
}

```

Function ini kami masukkan ke *function* untuk *login* tadi, sehingga setelah *login* berhasil, data – data akan dimuat.

- o. Selanjutnya, kami membuat *function* yang membuat jam dan tanggal pada panel *monitor* menjadi berfungsi. *Function* ini menggunakan

object JavaScript bernama Date untuk mendapatkan waktu di kehidupan nyata.

```
function everySec(){
  const timeNow = new Date();

  let weekDay = ["Minggu", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]
  let today = weekDay[timeNow.getDay()];
  let currentDate = timeNow.getDate();
  let months = ["Januari", "Februari", "Maret", "April", "Mei", "Juni", "Juli", "Agustus", "September", "Oktober", "November", "Desember"];
  let currentMonth = months[timeNow.getMonth()];
  let year = timeNow.getFullYear();
  let hoursOfDay = timeNow.getHours();
  let minutes = timeNow.getMinutes();

  hoursOfDay = hoursOfDay < 10 ? "0" + hoursOfDay : hoursOfDay;
  minutes = minutes < 10 ? "0" + minutes : minutes;

  let time = hoursOfDay + ":" + minutes;
}
```

Selanjutnya, kami memanggil *function* ini di dalam *function* login tadi menggunakan *setInterval*. *Method* *setInterval* ini memanggil sebuah *function* setiap milisekon tertentu tanpa henti hingga jendela *web* *close* atau ditutup. Maka, tujuan dari kode di bawah ini yaitu memperbarui waktu setiap detik (1000 milisekon sama dengan 1 detik).

```
setInterval(everySec, 1000);
everySec();
```

- p. Langkah selanjutnya, kami membuat *function* yang memberi fungsionalitas tombol – tombol yang ada pada panel *monitor* (tombol sistem) dan panel kontrol (tombol mode dan tombol komponen). *Function* ini memiliki *parameter* yang berisi apa saja data yang di-*update*. Selanjutnya, dengan AJAX, data di-*update* melalui REST API menggunakan HTTP *method* *PUT*.

```
function updateData($fields){
  let $obj = $fields;
  $.ajax({
    type: 'PUT',
    async: true,
    url: 'https://ecato.my.id/api/incubator/komponen',
    contentType: 'application/json',
    data: JSON.stringify($obj),
    beforeSend: function(){
      request = 1;
      $(".loading-section").fadeIn(400);
    },
    success: function(){
      request = 0;
      $(".loading-section").fadeOut(400);
    }
  });
}
```

Function ini selanjutnya dipanggil di setiap *method click* yang telah kami buat sebelumnya. Data yang di-update tergantung pada tombol atau *button* yang diklik. Jika *button* mode Auto diklik, maka data mode di-update menjadi otomatis. Jika *button* kipas diklik, dan jika mode adalah Manual, maka data kipas di-update. Begitu juga dengan tombol – tombol lainnya.

```

$( "#start-incubator" ).click(function(){
  if(request == 0 && sistem == 1){
    var countDown = new Date();
    countDown.setDate(countDown.getDate() + 22);
    startDate = countDown;
    updateData({sistem:2,penetasan:countDown,mode:mode,motor:motor,kipas:kipas,lampu:lampu});
    dayCountdown("enable");
  }
});
$( "#power" ).click(function(){
  if(request == 0 && sistem == 2){
    dayCountdown("disable");
    updateData({sistem:1,sisa_hari:-1,penetasan:"0000-00-00 00:00:00"});
    lastEstimation = 24;
  }
});
$( "#auto-mode" ).click(function(){
  if(request == 0 && sistem == 2 && mode != 1){
    updateData({mode:1});
  }
});
$( "#manual-mode" ).click(function(){
  if(request == 0 && sistem == 2 && mode != 2){
    updateData({mode:2});
  }
});
$( "#kipas-on" ).click(function(){
  if(request == 0 && sistem == 2 && mode == 2 && kipas != 1){
    updateData({kipas:1});
  }
});

```

- q. Setelah itu, kami menambahkan *client* WebSocket menggunakan *library* Laravel Echo agar *web* dapat berkomunikasi dengan *server* WebSocket. WebSocket di sini berfungsi untuk menangkap data – data secara *real-time*. Kami bisa saja menerapkan AJAX dengan HTTP *GET* yang diberi setInterval setiap detik untuk mendapatkan data setiap waktunya. Namun, hal tersebut akan memakan cukup banyak sumber daya (*resource*) komputer bagi *client-side*, sehingga performa aplikasi bisa saja menurun. Itulah mengapa kami menggunakan WebSocket, yang dapat melakukan komunikasi data secara *real-time* selama koneksi

antar *client* dan *server* tidak terputus. Untuk menginstal Laravel Echo, kami menjalankan perintah berikut di Terminal:

```
'npm install --save-dev laravel-echo pusher-js'
```

Selanjutnya, pada `resources/js/bootstrap.js`, *import* library – library yang telah diinstal tadi.

```
resources > js > JS bootstrap.js > ...
1  import Echo from "laravel-echo";
2  import Pusher from 'pusher-js';
3
```

Masih dalam file yang sama, tambahkan blok kode berikut. Sesuaikan *key* Echo dengan `PUSHER_APP_KEY` pada `*.env`.

```
window.Pusher = Pusher;
window.Echo = new Echo({
  broadcaster: 'pusher',
  key: 'ABCDEFG',
  wsHost: window.location.hostname,
  wsPort: 6002,
  wssPort: 6002,
  forceTLS: true,
  enabledTransports: ['ws', 'wss'],
  disableStats: true,
});|
```

Buka `incubator.blade.php`, tambahkan *tag script* berikut.

```
<script src="/js/jquery-3.6.1.min.js"></script>
<script src="https://ecato.my.id/js/app.js"></script>
<script src="/js/incubator.js"></script>
```

- r. Sekarang *client-side* sudah dapat mengkoneksikan ke *server* WebSocket dengan menggunakan *method* Echo.channel() dan *method* listen untuk menangkap semua **Event** atau semua data yang diperbarui. Langkah ini merupakan langkah terakhir dari pengembangan *front-end*.

Berikut adalah kode – kodenya.

```

    Echo.channel('Component')
      .listen('Suhu', (e) => {
        console.log("[WEBSOCKETS] {#Component} Suhu Event: " + e.suhu);
        $('#suhu').text(e.suhu + "°");
        suhu = e.suhu;
      })
      .listen('Kelembapan', (e) => {
        console.log("[WEBSOCKETS] {#Component} Kelembapan Event: " + e.kelembapan);
        $('#totalwaktu').text(e.kelembapan);
        kelembapan = e.kelembapan;
      })
      .listen('SisaHari', (e) => {
        console.log("[WEBSOCKETS] {#Component} SisaHari Event: " + e.sisa_hari);
        $('#sisawaktu').text(e.sisa_hari);
        sisa_hari = e.sisa_hari;
      })
      .listen('Penetasan', (e) => {
        console.log("[WEBSOCKETS] {#Component} Penetasan Event: " + e.penetasan);
        penetasan = e.penetasan;
      })
      .listen('Motor', (e) => {
        console.log("[WEBSOCKETS] {#Component} Motor Event: " + e.motor);
        motor = e.motor;
        if(sistem == 2){
          if(motor == 1){
            updateButton("motor",1);
          } else if(motor == 2){
            updateButton("motor",2);
          }
        }
      })
      .listen('Kipas', (e) => {
        console.log("[WEBSOCKETS] {#Component} Kipas Event: " + e.kipas);
        kipas = e.kipas;
        if(sistem == 2){
          if(kipas == 1){
            updateButton("kipas",1);
          } else if(kipas == 2){
            updateButton("kipas",2);
          }
        }
      })
      .listen('Lampu', (e) => {
        console.log("[WEBSOCKETS] {#Component} Lampu Event: " + e.lampu);
        lampu = e.lampu;
        if(sistem == 2){
          if(lampu == 1){
            updateButton("lampu",1);
          } else if(lampu == 2){
            updateButton("lampu",2);
          }
        }
      })
      .listen('Mode', (e) => {
        console.log("[WEBSOCKETS] {#Component} Mode Event: " + e.mode);
        mode = e.mode;
        if(sistem == 2){
          if(mode == 1){
            updateButton("mode",1);
          } else if(mode == 2){
            updateButton("mode",2);
          }
        }
      })
      .listen('Sistem', (e) => {
        console.log("[WEBSOCKETS] {#Component} Sistem Event: " + e.sistem);
        sistem = e.sistem;
        if(sistem == 1){
          updateButton("sistem",1);
        } else if(sistem == 2){
          updateButton("sistem",2);
        }
      })
    }
  
```

Dalam pengembangan JavaScript, kami banyak menggunakan **console.log**, karena dengan begitu kami dapat mengecek apakah ada kesalahan dalam program (*debugging*). Dengan begini, semua fungsionalitas aplikasi *web* sudah berjalan sepenuhnya, yang terakhir adalah fungsionalitas pada *board* WeMos.

7. Pemrograman Board WeMos D1 (oleh: Bintang)

Pemrograman untuk WeMos ini dilakukan menggunakan bahasa pemrograman Arduino yang memiliki kemiripan dengan bahasa pemrograman C dan C++. *Software* yang digunakan untuk memprogram adalah Arduino IDE, *software* yang digunakan untuk membuat *sketch* (program) pada *board – board* yang menggunakan bahasa Arduino.



Gambar 3.15 Logo Arduino

Secara keseluruhan, fungsionalitas yang harus ada pada *board* WeMos adalah kemampuan untuk melakukan pertukaran data (menerima dan mengirim data), memproses status komponen – komponen menjadi data – data yang dapat dikirim ke aplikasi *web*, hingga menerima data – data dari aplikasi *web* dan memprosesnya untuk menggerakkan komponen – komponen yang ada. Berikut adalah langkah – langkah saya untuk memprogram WeMos D1.

- WeMos adalah *board* yang berbasis ESP8266. Maka dari itu, saya harus menginstal Boards Manager untuk ESP8266 di Arduino IDE. Buka Arduino IDE, lalu pada *tab* File, klik Preferences. Pada Additional Boards Manager URLs, tambahkan URL berikut:

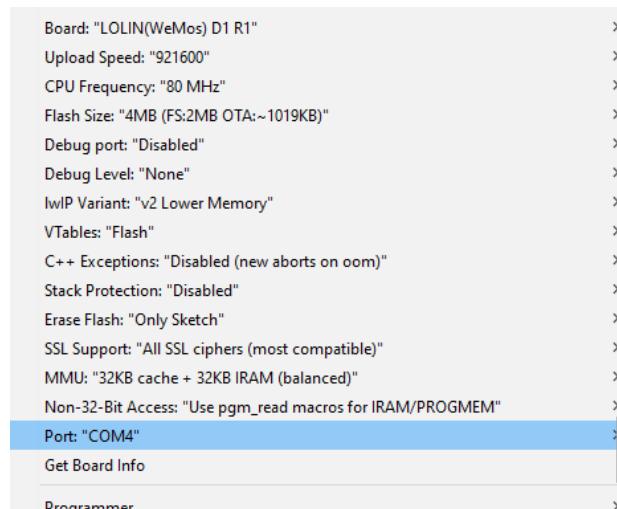
http://arduino.esp8266.com/stable/package_esp8266com_index.json

Setelah itu pada *tab* Tools, klik Board > Boards Manager. Cari 'esp8266', kemudian instal Boards Manager-nya.

- Selanjutnya saya menginstal *library* – *library* yang diperlukan. Ada *library* DHT untuk komponen DHT11, ESP8266WiFi untuk menghubungkan *board* ke internet dan ESP8266HTTPClient untuk

melakukan pertukaran data melalui REST API, ArduinoJson untuk memproses data mentah dari REST API, dan ArduinoWebsockets untuk menghubungkan koneksi WebSocket. Semua *library* dapat ditemukan pada Sketch > Include Library > Manage Libraries. Untuk *library* – *library* ESP8266 sendiri sudah satu paket dengan Boards Manager ESP8266, sehingga tidak perlu menginstal lagi.

- c. Siapkan *board* WeMos D1 dan kabel *micro USB* yang digunakan untuk meng-*upload sketch* ke *board*. Siapkan *Wi-Fi* juga untuk WeMos D1 dan pastikan jaringan *Wi-Fi* lancar. Setelah itu, tancapkan kabel *micro USB* yang sudah terhubung dengan komputer ke WeMos D1.
- d. Buat *sketch* baru di Arduino IDE. Ubah Board pada *tab Tools* > Board menjadi ESP8266 Boards > LOLIN(WeMos) D1 R1 dan ubah Port dengan *port* Serial yang terbuka.



- e. Masukkan atau *include* *library* – *library* yang diperlukan tadi menggunakan Sketch > Include Library. Setelah itu, muncullah baris – baris kode *include* seperti ini.

```

1 #include <dht.h>
2 #include <ESP8266WiFi.h>
3 #include <ESP8266HTTPClient.h>
4 #include <ArduinoJson.h>
5 #include <ArduinoWebsockets.h>

```

- f. Deklarasikan variabel untuk *pin – pin* komponen menggunakan *define*.

```
#define pinDHT D3
#define pinKipas D4
#define pinMotor D5
#define pinLampu D7
```

Buat juga variabel – variabel di bawahnya yang berisi SSID dan *password Wi-Fi* yang akan dikoneksikan nanti.

```
const char* ssid = "hotspot1024";
const char* password = "12345678";
```

- g. Saya ingin menguji koneksi *Wi-Fi* di WeMos D1 terlebih dahulu. Pada *void setup*, di baris pertama, tambahkan *Serial.begin* untuk menentukan kecepatan transmisi data WeMos D1 melalui *port* Serial (gunakan kecepatan 115200 untuk ESP8266). Setelah itu, tambahkan baris kode *WiFi.begin* untuk memulai koneksi ke *Wi-Fi*.

```
void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);
```

Agar tahu jika sudah terkoneksi ke *Wi-Fi* atau belum, saya menambahkan program di bawah ini. Ketika sedang proses menghubungkan, pada Serial Monitor akan tertulis 'Connecting...'. Jika sudah terhubung, maka Serial Monitor akan menampilkan tulisan 'Connected.'

```
void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);

    delay(500);
    Serial.print("Connecting");
    for(int i = 0; i < 10 && WiFi.status() != WL_CONNECTED; i++) {
        Serial.print(".");
        delay(1000);
    }
    Serial.println("Connected.");
}
```

Klik tombol Verify yang berbentuk centang di pojok kanan untuk mengecek apakah terdapat kesalahan *syntax* dalam program. Jika tidak ada masalah, klik Upload untuk mengirim program ke *board*. Pada jendela Output di bagian bawah, jika program berhasil di-*upload*, maka akan muncul tulisan berikut.

```
Writing at 0x00024000... (76 %)
Writing at 0x00028000... (84 %)
Writing at 0x0002c000... (92 %)
Writing at 0x00030000... (100 %)
Wrote 273488 bytes (201053 compressed) at 0x00000000 in 4.6 seconds (effective 478.5 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Buka Serial Monitor (pojok kanan atas). Jika *baud rate* diubah ke 115200, maka akan muncul tulisan koneksi *Wi-Fi* seperti program yang dibuat sebelumnya.

```
Connecting...
Connected.
```

- h. Langkah selanjutnya, saya membuat fungsi *void* yang memuat data dari aplikasi *web* (mode, sistem, kipas, motor, dan lampu) melalui REST API. Sebelum itu, saya membuat variabel – variabel *global* untuk menyimpan data – data yang diambil dari internet. Variabel – variable *global* inilah yang nantinya banyak digunakan di beberapa blok kode.

```
int sistem, _mode, kipas, motor, lampu,
```

Masukkan URL REST API di sebuah variabel.

```
const char* requestUrl = "http://ecato.my.id/api/incubator/komponen";
```

Tambahkan juga *object – object* berikut di atas *void setup*.

```
HTTPClient http;
WiFiClient client2;
```

Selanjutnya, buat fungsi *void* dengan nama `getData` seperti berikut.

```
void getData(){
    http.begin(client2,requestUrl);
    int code = http.GET();
    if(code > 0){
        String response = http.getString();
        Serial.print("[HTTP] GET DATA - Code : ");
        Serial.println(code);
        StaticJsonDocument<512> doc;
        deserializeJson(doc, response);
        sistem = doc["sistem"];
        _mode = doc["mode"];
        kipas = doc["kipas"];
        motor = doc["motor"];
        lampu = doc["lampu"];
    } else{
        Serial.print("[HTTP] GET DATA - Error on sending request, code : ");
        Serial.println(code);
    }
    http.end();
}
```

Inti dari fungsi tersebut adalah mendapatkan data dari URL REST API yang kemudian *response* datanya yang masih berbentuk JSON diolah atau diproses menjadi data yang dapat dimanfaatkan menggunakan *deserializeJson* yang merupakan salah satu fitur dari *library* ArduinoJson. Data yang telah diproses tersebut dimasukkan ke variabel – variabel *global*. Fungsi `getData` ini kemudian saya panggil di *void setup*, di bawah kode *Wi-Fi*. Jadi, setelah *Wi-Fi* terkoneksi, langkah program selanjutnya adalah memuat data – data dari REST API tadi.

```
}
Serial.println("");
Serial.println("Connected.");
getData();
```

- i. Selanjutnya, saya membuat fungsi *void* untuk mengirim data suhu dan kelembapan setiap detiknya. Fungsi ini saya buat lebih optimal dengan hanya memperbolehkan data dapat terkirim jika suhu/kelembapan terakhir (saya masukkan dalam variabel `lastTemp` dan `lastHumi`) tidak sama dengan suhu/kelembapan yang baru saja dibaca oleh sensor. Penjelasan mudahnya, jika nilai suhu/kelembapan yang dibaca sensor

masih sama dengan nilai suhu/kelembapan terakhir, maka data tidak akan terkirim. Sebaliknya, data dapat terkirim jika nilai suhu/kelembapan yang dibaca sensor mulai berbeda dengan nilai suhu/kelembapan terakhir. Alasan saya memakai pendekatan ini adalah untuk mengoptimalkan kinerja *board* dan *server* REST API. Selain itu, tanpa pendekatan ini, *board* bisa saja mengirim data suhu/kelembapan yang sama setiap detiknya saat tidak ada perubahan dari sensor sama sekali, sehingga terkesan sia – sia.

```
void updateDHT(){  
    DHT.read11(pinDHT);  
    String dhtJson; StaticJsonDocument<256> doc;  
    if((lastTemp != DHT.temperature) || (lastHumi != DHT.humidity)) && (!isnan(DHT.temperature)) || (!isnan(DHT.humidity))){  
        http.begin(client2,requestUrl);  
        http.addHeader("Content-Type", "application/json");  
        if(lastTemp != DHT.temperature){  
            lastTemp = DHT.temperature;  
            doc["suhu"] = lastTemp;  
        }  
        if(lastHumi != DHT.humidity){  
            lastHumi = DHT.humidity;  
            doc["kelembapan"] = lastHumi;  
        }  
        serializeJson(doc,dhtJson);  
        int httpCode = http.PUT(dhtJson);  
        if (httpCode > 0){  
            String response = http.getString();  
            Serial.println(response);  
        }  
        http.end();  
    }  
}
```

Fungsi `updateDHT` ini kemudian saya panggil di *void loop* dengan *delay* 2000 milisekon.

- j. Kondisi *board* yang sekarang masih belum menerima data secara *real-time* dari aplikasi *web*. Salah satu cara yang umum adalah *me-request* data dari REST API setiap detiknya menggunakan HTTP *client* dari ESP8266. Namun, menurut saya, cara ini tidak bagus karena dapat memperlambat kinerja *board*. Maka dari itulah saya menggunakan layanan WebSocket yang hanya memerlukan satu kali koneksi. Pertama saya akan menambah *callback function* bawaan dari *library* ArduinoWebsockets. *Callback function* adalah sebuah *function* yang dipanggil di dalam sebuah *function*. Saya mengambil *callback function* ini dari Examples ArduinoWebsockets yang kemudian saya kembangkan lagi untuk menyesuaikan kebutuhan *board*. Fungsi dari

function ini adalah untuk menangkap data – data *Event* yang disiarkan oleh *server*:

```
void onMessageCallback(WebsoketsMessage message) {
    String messageData = message.data();
    StaticJsonDocument<512> doc1, doc2;
    deserializeJson(doc1, message.data());
    if(messageData.indexOf("Sistem") != -1){
        Serial.println("sistem change");
        deserializeJson(doc2,doc1["data"]);
        sistem = doc2["sistem"];
    }
    if(sistem == 1){
        return;
    } else if(sistem == 2){
        if(messageData.indexOf("Mode") != -1){
            Serial.println("Mode change");
            deserializeJson(doc2,doc1["data"]);
            _mode = doc2["mode"];
            if(motor == 1){
                motor = 2;
                updateMotor(motor);
            } else if(motor == 2){
                motor = 1;
                updateMotor(motor);
            }
        }
    }
    if(_mode == 1){
        return;
    } else if(_mode == 2){
        if(messageData.indexOf("Kipas") != -1){
            Serial.println("Kipas change");
            deserializeJson(doc2,doc1["data"]);
            kipas = doc2["kipas"];
        }
        if(messageData.indexOf("Motor") != -1){
            Serial.println("Motor change");
            deserializeJson(doc2,doc1["data"]);
            motor = doc2["motor"];
        }
        if(messageData.indexOf("Lampu") != -1){
            Serial.println("Lampu change");
            deserializeJson(doc2,doc1["data"]);
            lampu = doc2["lampu"];
        }
    }
}
```

Maksud dari program di atas adalah, jika isi data yang disiarkan dari *server* terdapat *string* seperti 'Sistem', 'Mode', 'Kipas', 'Motor', atau 'Lampu', maka data di masukkan ke variabel – variabel *global*. Data – data yang ditangkap masih dalam format JSON. Maka, di sini saya juga menggunakan *deserializeJson* untuk memprosesnya.

- k. Selanjutnya, saya menambahkan beberapa baris kode untuk menghubungkan *board* ke *server* WebSocket menggunakan URL yang saya buat di variabel baru. URL tersebut memiliki *port* wss, jadi saya memerlukan *fingerprint* SSL agar dapat terhubung ke URL WebSocket tersebut. Sebab, *port* wss adalah koneksi WebSocket yang melewati protokol HTTPS.

```
const char* websockets_connection_string = "wss://ecato.my.id:6002/app/ABCDEFG?protocol=7&client=js&version=4.3.1&flash=false";
const char echo_org_ssl_fingerprint[] PROGMEM = "B4 A0 D6 76 1B 68 01 54 43 A7 33 45 AB D5 73 CE 06 09 40 83";
```

Untuk memulai koneksi WebSocket, saya menambahkan kode – kode berikut di bagian *void setup*. Setelah terkoneksi, kode di bawah ini akan mengirim data ke *server* tentang *Channel* yang akan di-*subscribe* atau didengarkan, yaitu 'Component'. *Channel* ini sudah dibuat sebelumnya pada tahap pengembangan *back-end*.

```
client.onMessage(onMessageCallback);
client.setFingerprint(echo_org_ssl_fingerprint);
client.connect(websockets_connection_string);
client.send("{\"event\":\"pusher:subscribe\", \"data\": {\"channel\":\"Component\"}}");
client.ping();
```

Saya juga mengubah *pinMode* untuk komponen kipas, motor, dan lampu menjadi *OUTPUT* semua.

- l. Langkah terakhir, saya membuat program utama di *void loop*. Program ini dibagi menjadi dua bagian. Program pertama bertugas untuk selalu menangkap data – data yang dikirim dari *server* menggunakan *client.poll* selama koneksi WebSocket masih tersambung, jika terputus program akan mencoba untuk mengkoneksikan WebSocket lagi. Sedangkan program kedua yaitu bagian yang menyalakan dan mematikan komponen – komponen berdasarkan mode. Selain itu, saya juga mengimprovisasi program yang memiliki *delay* dengan diganti menggunakan *millis*. Perbedaannya, *delay* digunakan untuk menunda atau menghentikan jalannya semua program selama periode waktu tertentu, sedangkan *millis* juga dapat digunakan untuk membuat jeda

dalam satu atau beberapa program tanpa menghentikan jalannya semua program sepenuhnya. Sebab itulah saya berubah pikiran untuk menggunakan *millis* untuk membuat sebuah jeda. Meskipun begitu, *millis* memerlukan kode yang lebih kompleks daripada *delay*.

```

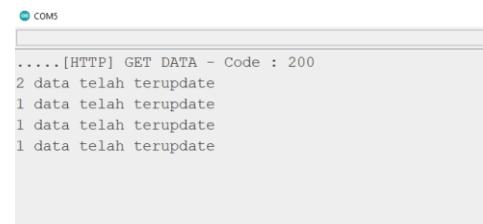
void loop() {
    if(client.available()) {
        client.poll();
    } else {
        client = {};
        client.onMessage(onMessageCallback);
        client.setFingerprint(echo_org_ssl_fingerprint);
        client.connect(websockets_connection_string);
        client.send("{\"event\":\"pusher:subscribe\", \"data\":{\"channel\":\"Component\"}}");
        client.ping();
    }

    if(sistem == 1){
        digitalWrite(pinKipas,1);
        digitalWrite(pinMotor,1);
        digitalWrite(pinLampu,1);
    } else if(sistem == 2){
        unsigned long currentMillis2 = millis();
        if(currentMillis2 - previousMillis2 >= interval2){
            previousMillis2 = currentMillis2;
            updateDHT();
        }
        if(_mode == 1){
            unsigned long currentMillis1 = millis();
            if(currentMillis1 - previousMillis1 >= interval1){
                previousMillis1 = currentMillis1;
                if(motor == 1){
                    Serial.println("Motor hidup!");
                    digitalWrite(pinMotor,0);
                    updateMotor(1);
                    motor = 2;
                } else if(motor == 2){
                    Serial.println("Motor mati!");
                    digitalWrite(pinMotor,1);
                    updateMotor(2);
                    motor = 1;
                }
            }
        }
    }

    if(lastTemp >= 39 || lastHumi <= 50){
        digitalWrite(pinKipas,0);
        digitalWrite(pinLampu,1);
        kipas = 1;
        lampu = 2;
        updateComp(2);
    } else if(lastTemp <= 37 || lastHumi >= 60){
        digitalWrite(pinKipas,1);
        digitalWrite(pinLampu,0);
        kipas = 2;
        lampu = 1;
        updateComp(1);
    }
    else if(_mode == 2){
        if(kipas == 1) digitalWrite(pinKipas,0); else if(kipas == 2) digitalWrite(pinKipas,1);
        if(motor == 1) digitalWrite(pinMotor,0); else if(motor == 2) digitalWrite(pinMotor,1);
        if(lampu == 1) digitalWrite(pinLampu,0); else if(lampu == 2) digitalWrite(pinLampu,1);
    }
}

```

Beginilah hasil akhir dari pemrograman Arduino yang dapat dilihat pada Serial Monitor.



BAB IV

HASIL PENELITIAN

A. Kendala dan Solusi

1. Skematik Rangkaian Elektronika (oleh: Abrar)

Saat membuat skematik rangkaian, saya mengalami beberapa kendala. Sebab, saya menggunakan *software* EAGLE untuk yang pertama kalinya. Maka dari itu, saya memerlukan waktu untuk adaptasi. Berikut adalah kendala yang terjadi saat membuat skematik rangkaian.

- a. Memerlukan waktu untuk adaptasi, karena sebelumnya menggunakan *software* Proteus untuk mendesain skematik rangkaian.
- b. Kesulitan untuk mencari beberapa *library* yang dibutuhkan.
- c. Beberapa kali *software* mengalami *crash* dan tidak bisa dibuka.

Dari permasalahan yang dihadapi, berikut adalah yang saya lakukan agar bisa mengatasi masalah – masalah tersebut:

- a. Melakukan riset dan juga mempelajari *tools – tools* yang ada
- b. Mencari *library* di *forum* dan juga di GitHub.
- c. Melakukan instal ulang *software* EAGLE.

2. Pembuatan Desain PCB Custom (oleh: Abrar)

Saat membuat desain PCB, saya mengalami perubahan desain sebanyak 5 kali. Berikut adalah alasan dari perubahan – perubahan tersebut:

- a. Pada desain pertama diperlukan PCB 2 *layer*.
- b. Pada desain kedua tidak diperlukan PCB 2 *layer*, namun harus menggunakan *jumper* yang cukup banyak.
- c. Desain ketiga sudah lumayan bagus, namun ukuran PCB yang dibutuhkan masih sangat besar dan juga ruang yang tersisa masih banyak.

- d. Pada desain keempat, saya sudah bisa membuat desain minimalis. Namun kali ini, saya menghilangkan *optocoupler*, sehingga *relay* tidak bisa berfungsi jika menggunakan WeMos D1 R1.
- e. Pada desain terakhir inilah yang akhirnya kita gunakan untuk mesin penetas telur, bisa dikatakan desain ini sudah cocok.

Dari beberapa kegagalan yang telah dihadapi, saya lebih bisa belajar dan sekaligus melakukan riset terhadap desain – desain yang telah dibuat. Sehingga dari beberapa kali kegagalan yang telah dihadapi, saya dapat menemukan jalan keluar sendiri.

3. Pencetakan PCB Custom (oleh: Abrar, Dewa, Sinta, Gassa, Ida)

Kendala yang kami alami pada saat mencetak desain ke PCB yaitu pemberian autan kurang banyak, sehingga banyak jalur yang terputus. Selain itu, ada juga beberapa jalur bergeser karena proses penggosokan yang terlalu kencang.

Solusi dari permasalahan yang kami hadapi pada tahap ini yaitu dengan cara menambahkan autan yang banyak. Jika jalur terputus, kami bisa menggunakan spidol permanen untuk menyambung jalur yang putus.

4. Pelarutan PCB Custom (oleh: Abrar, Dewa, Gassa, Sinta, Ida)

Pada saat melarutkan kami mengalami beberapa kendala, berikut adalah kendala yang dihadapi:

- a. Saat pertama kali melarutkan, wadah yang digunakan terlalu kecil.
- b. Wadah pelarutan yang digunakan berbahan aluminium sehingga lama – kelamaan, wadah tersebut terkikis dan berlubang.
- c. Terlalu banyak menggunakan larutan *ferric chloride*, sehingga menyebabkan jalur terkikis.
- d. Menggunakan air yang tidak panas, sehingga proses pelarutan sangat lama.

Dari permasalahan yang telah dihadapi, berikut adalah yang kami lakukan agar bisa mengatasi masalah – masalah tersebut:

- a. Mengganti wadah yang besar yang tidak terbuat dari aluminium, besi atau sejenisnya.
- b. Menggunakan air yang panas agar proses pelarutan lebih cepat.
- c. Menggunakan larutan *ferric chloride* secukupnya.

5. **Pengeboran PCB Custom** (oleh: Abrar, Dewa, Sinta, Gassa, Silfaradila)

Kendala yang dihadapi saat melakukan pengeboran PCB yaitu terdapat satu mata bor saja, sehingga ada ukuran yang lebih besar yang harus dilubangi beberapa kali. Kendala lain yang dihadapi yaitu kurang fokus saat mengebor sehingga menyebabkan tangan terkena bor dan terluka.

Solusi dari permasalahan ini yaitu, harus melubangi PCB beberapa kali agar bisa mendapatkan lubang yang pas dengan kaki komponen yang digunakan. Selain itu, kami juga lebih berhati – hati saat proses pengeboran.

6. **Penyolderan PCB Custom** (oleh: Abrar, Sinta, Silfaradila)

Pada saat proses penyolderan ada beberapa kendala yang kami dialami. Berikut adalah kendala – kendala tersebut:

- a. Solder yang digunakan kurang panas.
- b. Tenol atau timah yang digunakan memiliki titik leleh yang tinggi, sehingga sangat sulit meleleh.
- c. Tangan terkena solder.
- d. Mata solder kendor, sehingga beberapa kali copot saat menyolder.
- e. Tenol atau timah sulit menempel di papan PCB.

Dari permasalahan yang telah dihadapi, berikut adalah yang kami lakukan untuk mengatasi masalah – masalah tersebut:

- a. Mengganti solder yang lebih panas.
- b. Mengganti tenol yang memiliki titik leleh rendah.
- c. Memberi *flux* pada papan PCB sebelum menyolder.
- d. Lebih berhati – hati saat melakukan penyolderan.

7. Pemilihan Board

Saat memilih *board*, pilihan kita yang pertama adalah WeMos D1 R1. Kemudian terdapat sebuah kendala, yaitu pin *digital* WeMos hanya mengeluarkan *output* 3.3V sehingga tidak bisa menjalankan *relay* 5V. Kemudian kita memilih Arduino Uno dan juga ESP-01 sebagai *Wi-Fi*, namun tetap saja tidak membuat hasil karena ESP-01 yang digunakan tidak bisa terhubung dengan Arduino Uno.

Solusi dari permasalahan ini adalah kembali menggunakan pilihan pertama, yaitu WeMos D1 R1. *Board* tersebut sudah memiliki *Wi-Fi* dan juga lebih ringkas jika dibandingkan menggunakan Arduino Uno dan juga ESP-01 sebagai modul *Wi-Fi*. Walaupun *output pin digital* WeMos D1 R1 hanya 3.3V, kami tetap bisa mengontrol *relay* 5V.

8. Pemilihan Komponen Mesin Penetas Telur

Saat ingin merancang komponen yang akan digunakan untuk mesin penetas telur, kami mengalami beberapa kendala, yaitu:

- a. Salah memilih *relay*. Saat pertama kali membeli *relay*, kami membeli *relay* 12V, akibatnya, *board* WeMos tidak cocok untuk *relay* tersebut.
- b. Tidak memakai *optocoupler*. Pada desain PCB yang keempat, kami memutuskan untuk membuat desain seminimalis mungkin, sehingga menghilangkan *optocoupler*. Akibatnya, rangkaian *relay* tidak bisa berjalan, karena *output pin digital* pada *board* WeMos yang digunakan hanya 3.3V.

Dari masalah tersebut, berikut adalah yang kami lakukan untuk mengatasi masalah – masalah tersebut:

- a. Kami memutuskan untuk membeli *relay* 5V yang cocok dengan rangkaian yang dibuat.
- b. Dikarenakan *board* yang digunakan hanya mengeluarkan *output* 3.3V pada *pin digital*, kami memutuskan untuk menggunakan *optocoupler*, agar *board* digunakan bisa mengontrol *relay* 5V.

9. Perancangan Desain Web (oleh: Dewa)

Kendala yang saya alami yaitu pada saat melakukan desain *web*, saya berulang kali melakukan perubahan karena peletakan fungsi – fungsi *web* yang kurang relevan apabila diterapkan.

Melakukan riset dan *sketching* dengan berdiskusi bersama kelompok lagi agar dalam tahap desain tidak terjadi banyak revisi.

10. Instalasi Framework Laravel (oleh: Bintang)

Pada tahap ini, saya mengalami sebuah kendala yang cukup merepotkan, yaitu instalasi *framework* Laravel tidak dapat berjalan dengan lancar. Setelah saya melakukan *searching* di Google, saya menemukan masalah yang menyebabkan hal ini terjadi, sekaligus dengan solusinya. Penyebabnya adalah versi PHP pada *server* belum *update*, maka dari itu saya meng-*update* versi PHP-nya menggunakan perintah `apt-get`.

11. Pembuatan Back-end Web (oleh: Bintang)

Kendala pertama yang terjadi pada tahap ini adalah tidak cocoknya versi *library* untuk layanan WebSocket. Solusi dari masalah ini, saya menginstal versi *library* yang cocok untuk layanan WebSocket. Kendala yang kedua hanyalah kesalahan – kesalahan kecil seperti kesalahan *syntax*, kurangnya tanda titik koma (;), dan sebagainya. Solusinya, saya cukup membenarkan kode – kode yang salah ketik.

12. Pembuatan Front-end Web (oleh: Bintang, Gassa)

Kendala yang terjadi pada tahap ini kebanyakan adalah *styling* CSS yang tidak bisa sesuai dengan desain *web* Figma. Solusinya, kami melakukan *searching* di Google untuk mencari solusi *styling* – *styling* yang tepat bagi tampilan *web*.

13. Pemrograman Board WeMos D1 (oleh: Bintang)

Pada tahap ini, kendala yang saya hadapi yaitu tidak berhasilnya koneksi WebSocket. Setelah saya mencari solusi – solusi di Google, ternyata tidak ada satupun yang bisa menyelesaikan masalah ini. Solusinya,

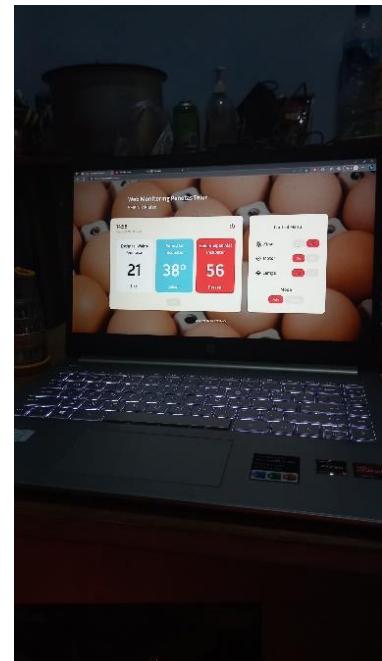
saya mencoba – coba dengan kode – kode yang berbeda hingga saya menemukan penyebab permasalahan dan cara mengatasinya.

B. Hasil Pengujian

Pada gambar *web* di bawah ini, mode yang dipilih adalah otomatis sehingga diperlihatkan komponen – komponen di dalam mesin penetas telur berjalan dengan sendirinya. Saat itu, mesin penetas telur memiliki suhu 38 derajat dan kelembapan 56 persen seperti yang ditampilkan pada *web*, sehingga lampu dan kipas berjalan sesuai dengan *flowchart* yang telah dibuat. Dengan kata lain, sistem mesin penetas telur otomatis berbasis aplikasi *web* sudah berhasil dalam tahap pengujian.



Gambar 3.16 Pengujian mesin penetas telur



Gambar 3.17 Pengujian aplikasi web

BAB V

PENUTUP

A. Kesimpulan

Dalam laporan ini, kami telah memaparkan rancangan sistem mesin penetas telur otomatis berbasis aplikasi *web* dan *board* WeMos D1. Mesin penetas telur menggunakan boks kayu dengan komponen lampu, kipas, dan motor sinkron untuk menstabilkan suhu dan kelembapan yang optimal bagi telur yang akan ditetaskan. *Board* Wemos D1 R1 digunakan untuk melakukan transmisi data melalui REST API dan WebSocket, sementara *framework* Laravel digunakan untuk pengembangan aplikasi *web*.

Secara keseluruhan, dapat disimpulkan bahwa sistem ini merupakan solusi yang efektif dan efisien untuk proses penetasan telur. Selain itu, kemampuan sistem untuk dikontrol melalui *web* memberikan fleksibilitas dalam mengontrol dan memantau kondisi sistem dari jarak jauh. Sistem ini dapat membantu peternak dalam meningkatkan efisiensi dan kualitas hasil penetasan telur. Kami harap penelitian ini dapat memberikan wawasan dan ide untuk pengembangan sistem penetas telur otomatis yang lebih canggih di masa depan.

B. Saran

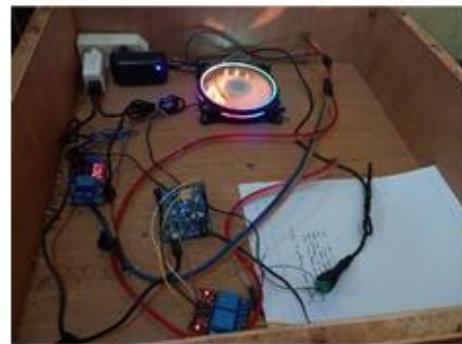
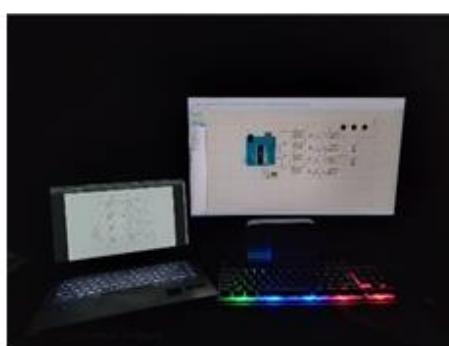
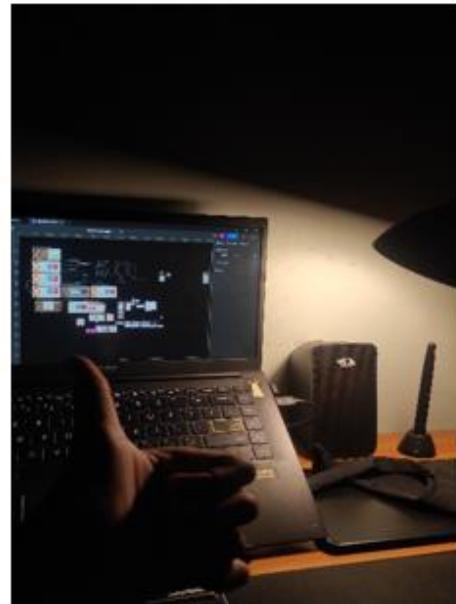
1. Pastikan semua alat dan bahan seperti boks kayu, lampu, kipas, dan motor sinkron memiliki spesifikasi yang sesuai dengan kebutuhan sistem, dan pastikan bahwa semua alat dan bahan tersebut dirawat dengan baik untuk memastikan kinerja yang optimal.
2. Terakhir, pastikan untuk selalu mengikuti pedoman dan peraturan yang berlaku dalam proses penetasan telur, termasuk keamanan dan kesejahteraan hewan, serta kualitas telur yang dihasilkan.

DAFTAR PUSTAKA

- Kmtech.id. (2019, 8 Oktober). *Mengenal Perangkat Lunak Arduino IDE*. Diakses pada 26 Maret 2023, dari <https://www.kmtech.id/post/mengenal-perangkat-lunak-arduino-ide>
- Dewaweb. (2022, 22 November). *Visual Studio Code: Pengertian, Fitur, Keunggulan dan Jenisnya*. Diakses pada 26 Maret 2023, dari <https://www.dewaweb.com/blog/mengenal-visual-studio-code/>
- Dewaweb. (2022, 10 Juni). *Apa itu Laravel? Pengertian, Fitur dan Kelebihannya*. Diakses pada 26 Maret 2023, dari <https://www.dewaweb.com/blog/apa-itu-laravel/>
- BiznetGio. (2015). *Mengenal Apa itu Ubuntu, Jenis, dan Kelebihannya*. Diakses pada 26 Maret 2023, dari <https://www.biznetgio.com/news/apa-itu-ubuntu>
- Ekrut Media. (2022, 27 September). *Figma Adalah: 4 Fitur, Fungsi, Cara Kerjanya, serta Bedanya dengan UI/UX Lainnya*. Diakses pada 26 Maret 2023, dari <https://www.ekrut.com/media/figma-adalah>
- Institut Teknologi Telkom Purwokerto. (2017). *HMDT Selenggarakan Workshop Eagle Software*. Diakses pada 26 Maret 2023, dari <https://ittelkom-pwt.ac.id/hmdt-selenggarakan-workshop-eagle-software/>
- Dicoding Indonesia. (2021, 8 Mei). *Apa itu Back-End dan Back-End Developer?*. Diakses pada 7 Mei 2023, dari <https://www.dicoding.com/blog/apa-itu-back-end/>
- LP2M. (2022, 29 Juni). *Front End dan Back End, Mengenal Definisi dan Perbedaannya*. Diakses pada 7 Mei 2023, dari <https://lp2m.uma.ac.id/2022/06/29/front-end-dan-back-end-mengenal-definisi-dan-perbedaannya/>

LAMPIRAN

A. Dokumentasi Kegiatan



B. Source Code Back-end

1. `Models/Komponen.php`

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Komponen extends Model
{
    use HasFactory;
    protected $fillable = ['suhu', 'kelembapan', 'sisa_hari',
'motor', 'kipas', 'lampu', 'mode', 'sistem'];
    public $timestamps = false;
}
```

2. `App/Http/Controllers/KomponenController.php`

```
<?php
namespace App\Http\Controllers;
use App\Models\Komponen;
use Illuminate\Http\Request;
use Illuminate\Validation\Rule;
use App\Events\Suhu;
use App\Events\Kelembapan;
use App\Events\SisaHari;
use App\Events\Motor;
use App\Events\Kipas;
use App\Events\Lampu;
use App\Events\Mode;
use App\Events\Sistem;
use App\Events\Penetasan;
class KomponenController extends Controller
{
    public function index(){
        return Komponen::find(1);
    }
    public function komponenUpdate(request $request){
        $compRequest = 0;
        if($request->sistem){
            event(new Sistem($request->sistem));
        }
    }
}
```

```
        $compRequest++;
    }
    if($request->mode){
        event(new Mode($request->mode));
        $compRequest++;
    }
    if($request->suhu){
        if($request->suhu < 0) $request->suhu = 0;
        event(new Suhu($request->suhu));
        $compRequest++;
    }
    if($request->kelembapan){
        if($request->kelembapan < 0) $request->kelembapan=0;
        event(new Kelembapan($request->kelembapan));
        $compRequest++;
    }
    if($request->sisa_hari){
        if($request->sisa_hari < 0) $request->sisa_hari = 0;
        event(new SisaHari($request->sisa_hari));
        $compRequest++;
    }
    if($request->penetasan){
        event(new Penetasan($request->penetasan));
        $compRequest++;
    }
    if($request->motor){
        event(new Motor($request->motor));
        $compRequest++;
    }
    if($request->kipas){
        event(new Kipas($request->kipas));
        $compRequest++;
    }
    if($request->lampu){
        event(new Lampu($request->lampu));
        $compRequest++;
    }
    $response = "$compRequest data telah terupdate";
    return $response;
}
}
```

3. `routes/api.php`

```
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\KomponenController;
Route::get('/incubator/komponen', [KomponenController::class,
'index']);
Route::put('/incubator/komponen', [KomponenController::class,
'komponenUpdate']);
```

C. Source Code Front-end

1. `resources/views/incubator.blade.php`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="icon" type="image/x-icon" href="/img/favicon.ico">
    <link rel="stylesheet" href="/css/style.css">
    <title>Cato | Incubator Monitor</title>
</head>
<body>
    <div id="loader" class="center"></div>
    <div class="loading">
    </div>
    <div class="page1">
        <div class="login">
            <h1>Login</h1>
            <hr>
            <p class="login-text-atas">Web Monitoring Penetas
Telur</p>
            <p class="login-text-bawah">SMK N 2 Klaten</p>
    </div>
</body>
```

```
<div class="form-input">
    <label class="ipt-txt">Username</label>
    <input class="ipt" type="text" id="user-ipt">
    <label class="ipt-txt">Password</label>
    <input class="ipt" type="password" id="pass-ipt">
</div><br>
<button class="login-button" onclick="loginButton()" id="signin" type="button">Login</button>
<p class="login-info">&nbsp;</p>
<p class="copyright">Copyright © 2023 Cato.</p>
</div>
</div>
<div class="monitor-wrapper">
    <div class="title-box">
        <div class="title-section">
            <h1>Web Monitoring Penetas Telur</h1>
            <h2>SMK N 2 Klaten</h2>
        </div>
        <div class="loading-section">
            <div id="data-load"></div>
            <div id="load-text">Updating data ...</div>
        </div>
    </div>
    <div class="monitoring-container">
        <div class="component-box" id="monitor-box">
            <div id="monitor-menu">
                <div class="monitor-box" id="top-monitor">
                    <div class="power-time" id="time">
                        <div id="clock">00:00</div>
                        <div id="date">Senin, 32 Februari
2000</div>
                    </div>
                    <div class="power-time" id="power">
                        
                    </div>
                </div>
                <br>
                <div class="monitor-card">
                    <div class="monitor-detail" id="timeleft">
                        Estimasi Waktu Penetasan
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

```
        <div id="sisawaktu">0</div>
        Hari
    </div>
    <div class="monitor-detail"
id="temperature">
        Suhu Alat <br>Incubator
        <div id="suhu">24°</div>
        Celcius
    </div>
    <div class="monitor-detail" id="daystotal">
        Kelembapan Alat Incubator
        <div id="totalwaktu">5</div>
        Persen
    </div>
    <br>
    <button id="start-incubator">Mulai</button>
</div>
</div>
<div class="component-box" id="control-box">
    <div id="control-menu">
        <h2>Control Menu</h2>
        <br>
        <div id="manual-control">
            <div class="controls" id="comp-kipas">
                <div class="components">
                    
                    <h2 class="component-
name">Kipas</h2>
                    </div>
                    <div class="toggle">
                        <button class="comp-toggle"
id="kipas-on">On</button>
                        <button class="comp-toggle"
id="kipas-off">Off</button>
                    </div>
                </div>
            </div>
            <br>
            <div class="controls" id="comp-motor">
                <div class="components">
```

```
          
      <h2 class="component-
name">Motor</h2>
      </div>
      <div class="toggle">
          <button class="comp-toggle"
id="motor-on">On</button>
          <button class="comp-toggle"
id="motor-off">Off</button>
      </div>
      <br>
      <div class="controls" id="comp-lampu">
          <div class="components">
              
          <h2 class="component-
name">Lampu</h2>
          </div>
          <div class="toggle">
              <button class="comp-toggle"
id="lampu-on">On</button>
              <button class="comp-toggle"
id="lampu-off">Off</button>
          </div>
          <br>
          <div class="toggle-mode">
              <h2>Mode</h2>
              <br>
              <div class="toggle">
                  <button class="mode-toggle" id="auto-
mode">Auto</button>
                  <button class="mode-toggle" id="manual-
mode">Manual</button>
              </div>
          </div>
      </div>
  </div>
</div>
```

```
</div>
<div class="footer">
    <p id="copyright-text">Copyright © 2023 Cato.</p>
</div>
</div>
<script src="/js/jquery-3.6.1.min.js"></script>
<script src="https://ecato.my.id/js/app.js"></script>
<script src="/js/incubator.js"></script>
</body>
</html>
```

2. `public/css/style.css`

```
*{
    margin: 0;
    padding: 0;
    -webkit-user-select: none;
    -ms-user-select: none;
    user-select: none;
}
button:focus{
    outline: 0;
}
html{
    background-image: url('../img/eggback.webp');
    background-repeat: no-repeat;
    background-size: cover;
    background-color:#EEEAD7;
    background-position: center center;
    background-attachment: fixed;
}
html, body{
    height: 100%;
}
#loader {
    border: 8px solid #56A1C0;
    border-radius: 50%;
    border-top: 8px solid gray;
    width: 40px;
```

```
    height: 40px;
    animation: spin 1s linear infinite;
    z-index: 10;
}
#data-load {
    margin-left: 10px;
    border: 5px solid #56A1C0;
    border-radius: 50%;
    border-top: 5px solid gray;
    width: 20px;
    height: 20px;
    animation: spin 1s linear infinite;
}
#load-text{
    font-size: 20px;
}
@keyframes spin {
    100% {
        transform: rotate(360deg);
    }
}
@font-face {
    font-family: InriaSans;
    src: url("/fonts/InriaSans-Regular.ttf");
}

@font-face {
    font-family: InriaSans;
    src: url("/fonts/InriaSans-Bold.ttf");
    font-weight: bold;
}

@font-face {
    font-family: InriaSans;
    src: url("/fonts/InriaSans-Light.ttf");
    font-weight: lighter;
}
.center {
    position: absolute;
    top: 0;
    bottom: 0;
```

```
    left: 0;
    right: 0;
    margin: auto;
}
.login-info{
    text-align: center;
    font-family: InriaSans;
    font-weight: bold;
    font-size: 20px;
    color: #D92332;
    margin-top: 20px;
}
.loading{
    background-color: #EEEAD7;
    height: 100%;
    width: 100%;
    left: 0;
    top: 0;
    position: fixed;
    overflow: hidden;
    z-index: 9;
}
.login h1{
    font-family: InriaSans;
    font-weight: bold;
    color: black;
    text-align: center;
    font-style: normal;
    font-size: 30px;
    text-align: center;
    margin-bottom: 10px;
}
.login-button:hover{
    cursor: pointer;
    background-color: #3f7a93;
}
.page1{
    display:flex;
    justify-content: center;
    align-items: center;
    height: 100%;
```

```
}

.login{
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    background: #EEEAD7;
    border-radius: 20px;
    padding: 50px;
}

.login form p h1{
    width: 250px;
}

hr {
    width: 500px;
    border: 1px solid black;
}

body{
    display: flex;
    justify-content: center;
    align-items: center;
}

.login-text-atas{
    font-style: normal;
    font-weight: 600;
    font-size: 30px;
    font-family: InriaSans;
    margin-top: 10px;
}

.login-text-bawah{
    font-family: InriaSans;
    font-weight: lighter;
    font-style: normal;
    font-size: 20px;
    margin-top: 10px;
    margin-bottom: 50px;
}

form label{
    font-family: InriaSans;
}

.form-wrapper{
```

```
    margin: 0 auto;
}

.form-input{
    margin-bottom: 10px;
}
input{
    border: none;
    outline: none;
    border-radius: 20px;
    background: #C7C4B3;
    width: 450px;
    height: 24px;
    display: block;
    font-family: InriaSans;
    font-weight: lighter;
    margin-top: 5px;
    margin-bottom: 20px;
}
input[type=text] [type=password]{
    padding-left: 20px;
    padding-right: 20px;
}
.login-button{
    background-color: #56A1C0;
    border-radius: 20px;
    text-align: center;
    font-family: InriaSans;
    color: white;
    font-size: 20px;
    border: none;
    outline: none;
    line-height: 50px;
    padding: 0px 100px 0px 100px;
}
.ipt{
    padding: 15px 30px;
    font-size: 20px;
}
.ipt-txt{
    text-align: left;
```

```
        font-size: 20px;
        font-family: InriaSans;
        font-weight: 400;
    }
    .copyright{
        text-align: center;
        font-family: InriaSans;
        font-weight: lighter;
        font-style: normal;
        margin-top: 20px;
        margin-bottom: 20px;
        vertical-align:text-bottom;
        font-size: 16px;
    }
    .monitor-wrapper{
        display: flex;
        flex-direction: column;
        align-items: center;
    }
    .monitor-wrapper button{
        background-color: #C7C4B3;
        padding: 12px 25px 12px 25px;
        border: 0;
        border-radius: 20px;
        text-align: center;
        font-family: InriaSans;
        color: white;
        width: auto;
        font-size: 20px;
    }
    .title-section{
        display: inline-block;
    }
    .loading-section{
        width: 190px;
        font-family: InriaSans;
        border-radius: 20px;
        background-color: #EEEAD7;
        display: flex;
        justify-content: center;
        align-items: center;
    }
```

```
    text-align: center;
    gap: 10px;
    height: 70px;
    padding: 0 10px 0 10px;
}
.component-box{
    background-color: #EEEAD7;
    height: 600px;
}
#manual-control{
    width: 90%;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    gap: 10px;
    align-items: center;
}
#monitor-box{
    text-align: center;
    border-radius: 20px;
    width: 880px;
}
#control-box{
    text-align: center;
    border-radius: 20px;
    width: 470px;
}
#control-box h2{
    font-size: 28px;
}
.monitoring-container{
    padding-top: 50px;
    text-align: justify;
    display: flex;
    justify-content: center;
    gap: 40px;
    height: 650px;
}
.title-box{
    align-self:flex-start;
    display: flex;
```

```
justify-content: space-between;
align-items: center;
width: 100%;
}
.title-box h1{
    font-family: InriaSans;
    font-weight: bold;
    font-size: 40px;
    color: white;
    margin-bottom: 10px;
}
.title-box h2{
    font-family: InriaSans;
    font-weight: lighter;
    font-size: 30px;
    color: white;
}
#monitor-menu{
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    gap: 10px;
    height: 100%;
}
.monitor-card{
    width: 780px;
    display: flex;
    justify-content: space-between;
}
.monitor-box{
    display: flex;
    justify-content: space-between;
    align-items: center;
    gap: -50px;
    width: 800px;
}
.monitor-detail{
    padding-top: 24px;
    padding-bottom: 16px;
    border-radius: 20px;
```

```
    line-height: 1.5;
    font-family: InriaSans;
    font-weight: bold;
    font-size: 28px;
    color: white;
    text-align: center;
    display: inline-block;
    width: 250px;
    height: fit-content;
}
#sisawaktu, #suhu, #totalwaktu{
    margin: 7px 0 7px 0;
    font-size: 110px;
}
#timeleft{
    color: black;
    background-color: white;
}
#temperature{
    background-color: #56A1C0;
}
#daystotal{
    background-color: #D92332;
}
#time{
    text-align: left;
}
#clock{
    font-family: InriaSans;
    font-weight: bold;
    font-size: 35px;
}
date{
    font-family: InriaSans;
    font-size: 20px;
}
.power-time{
    display: inline-block;
}
.footer{
    text-align: center;
```

```
        display: block;
        width: 80%;
        margin: 0 auto;
    }
    #copyright-text{
        font-family: InriaSans;
        font-weight: lighter;
        color: white;
    }
    #control-menu{
        font-family: InriaSans;
        color: black;
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        gap: 20px;
        height: 100%;
    }
    .controls{
        width: 90%;
        display: flex;
        flex-direction: row;
        justify-content: space-between;
        align-items: center;
        text-align: justify;
    }
    .components{
        display: flex;
        flex-direction: row;
        justify-content: center;
        gap: 10px;
    }
    .toggle{
        display: flex;
        flex-direction: row;
        justify-content: center;
        gap: 8px;
    }
    .components img{
        width: 35px;
```

```

        height: auto;
    }
.components h2{
    display: inline-block;
}

```

3. `public/js/incubator.js`

```

var suhu, motor, kipas, lampu, mode, sistem, kelembapan, sisa_hari,
lastEstimation = 0, penetasan, request = 0, dayctd, loggedIn = 0,
dateInterval = 1000, requestInterval = 2000, startDate;

function everySec(){
    const timeNow = new Date();

    let weekDay = ["Minggu", "Senin", "Selasa", "Rabu", "Kamis",
"Jum'at", "Sabtu"]
    let today = weekDay[timeNow.getDay()];
    let currentDate = timeNow.getDate();
    let months = ["Januari", "Februari", "Maret", "April", "Mei",
"Juni", "Juli", "Agustus", "September", "Oktober", "November",
"Desember"];
    let currentMonth = months[timeNow.getMonth()];
    let year = timeNow.getFullYear();
    let hoursOfDay = timeNow.getHours();
    let minutes = timeNow.getMinutes();

    hoursOfDay = hoursOfDay < 10 ? "0" + hoursOfDay : hoursOfDay;
    minutes = minutes < 10 ? "0" + minutes : minutes;

    let time = hoursOfDay + ":" + minutes;
}

function getData(){
    $.ajax({
        url:"https://ecato.my.id/api/incubator/komponen",
        type:"GET",
        async:true,
        dataType:"json",
        success: function(data){
            $("#loader").delay(1000).fadeOut(400);

```

```
$(".loading").delay(1000).fadeOut(400);
suhu = data.suhu;
kelembapan = data.kelembapan;
motor = data.motor;
kipas = data.kipas;
lampu = data.lampu;
mode = data.mode;
sistem = data.sistem;
penetasan = data.penetasan;
sisa_hari = data.sisa_hari;
$("#suhu").text(suhu + "°");
$("#totalwaktu").text(kelembapan);
if(sistem == 1){
    updateButton("sistem",1);
} else if(sistem == 2){
    if(sisa_hari > 0){
        startDate = new Date(penetasan);
        lastEstimation = sisa_hari;
        dayCountdown("enable");
    }
    updateButton("sistem",2);
    $("#sisawaktu").text(sisa_hari);
    if(motor > 0){
        if(motor == 1){
            updateButton("motor",1);
        } else if(motor == 2){
            updateButton("motor",2);
        }
    }
    if(kipas > 0){
        if(kipas == 1){
            updateButton("kipas",1);
        } else if(kipas == 2){
            updateButton("kipas",2);
        }
    }
    if(lampu > 0){
        if(lampu == 1){
            updateButton("lampu",1);
        } else if(lampu == 2){
            updateButton("lampu",2);
        }
    }
}
```

```

        }
    }
    if(mode > 0){
        if(mode == 1){
            updateButton("mode",1);
        } else if(mode == 2){
            updateButton("mode",2);
        }
    }
}
});

function updateButton($button,$value){
    switch($button){
        case "sistem":
            switch($value){
                case 1:
                    $("#start-incubator").css("background-color","#D92332");
                    $("#start-incubator").hover(
                        function(){
                            $(this).css("cursor", "pointer");},);
                    $("#power").hover(
                        function(){
                            $(this).css("cursor", "default");},);
                    $("#power").css("filter","grayscale(100%)");
                    $(".comp-toggle").css("background-color","#C7C4B3");
                    $(".mode-toggle").css("background-color","#C7C4B3");
                    $(".comp-toggle").hover(
                        function(){
                            $(this).css("cursor", "default");});
                    $(".mode-toggle").hover(
                        function(){
                            $(this).css("cursor", "default");});
                    break;
                case 2:
                    $("#start-incubator").css("background-color","#C7C4B3");
                    $("#power").css("filter","grayscale(0%)");
                    $("#start-incubator").hover(
                        function(){

```

```
        $(this).css("cursor", "default");},));
        $("#power").hover(
            function(){
                $(this).css("cursor", "pointer");},));
        $(".comp-toggle").hover(
            function(){
                $(this).css("cursor", "pointer");});
        $(".mode-toggle").hover(
            function(){
                $(this).css("cursor", "pointer");});
        break;
    }
    break;
case "kipas":
    switch($value){
        case 1:
            $("#kipas-on").css("background-color", "#D92332");
            $("#kipas-off").css("background-color", "#C7C4B3");
            break;
        case 2:
            $("#kipas-off").css("background-color", "#D92332");
            $("#kipas-on").css("background-color", "#C7C4B3");
            break;
    }
    break;
case "motor":
    switch($value){
        case 1:
            $("#motor-on").css("background-color", "#D92332");
            $("#motor-off").css("background-color", "#C7C4B3");
            break;
        case 2:
            $("#motor-off").css("background-color", "#D92332");
            $("#motor-on").css("background-color", "#C7C4B3");
            break;
    }
    break;
case "lampa":
    switch($value){
        case 1:
            $("#lampa-on").css("background-color", "#D92332");
            break;
        case 2:
            $("#lampa-off").css("background-color", "#D92332");
            $("#lampa-on").css("background-color", "#C7C4B3");
            break;
    }
    break;
}
```

```

        $("#lampu-off").css("background-color","#C7C4B3");
        break;
    case 2:
        $("#lampu-off").css("background-color","#D92332");
        $("#lampu-on").css("background-color","#C7C4B3");
        break;
    }
    break;
case "mode":
    switch($value){
        case 1:
            $("#auto-mode").css("background-color","#D92332");
            $("#manual-mode").css("background-color","#C7C4B3");
            $(".comp-toggle").hover(
                function(){
                    $(this).css("cursor", "default");});
                break;
        case 2:
            $("#manual-mode").css("background-color","#D92332");
            $("#auto-mode").css("background-color","#C7C4B3");
            $(".comp-toggle").hover(
                function(){
                    $(this).css("cursor", "pointer");});
                break;
            }
        break;
    }
}

function updateData($fields){
    let $obj = $fields;
    $.ajax({
        type: 'PUT',
        async:true,
        url: 'https://ecato.my.id/api/incubator/komponen',
        contentType: 'application/json',
        data: JSON.stringify($obj),
        beforeSend: function(){
            request = 1;
            $(".loading-section").fadeIn(400);
        },

```

```

success: function(){
    request = 0;
    $(".loading-section").fadeOut(400);
}
});

}

function dayCountdown($value){
switch($value){
    case "enable":
        dayctd = setInterval(function() {
            var now = new Date().getTime();
            var distance = startDate - now;
            var days = Math.floor(distance / (1000 * 60 * 60 * 24));
            if(lastEstimation != days){
                lastEstimation = days;
                updateData({sisa_hari:lastEstimation});
            }
            if (distance <= 0) {
                clearInterval(dayctd);
                updateData({sisa_hari:-1});
            }
        }, 1000);
        break;
    case "disable":
        clearInterval(dayctd);
        break;
}
}

document.onreadystatechange = function () {
    if (document.readyState == "complete") {
        $(document).keypress(function(event){
            if(event.keyCode == 13 && loggedIn == 0){
                $(".login-button").click();
            }
        })
        $(".monitor-wrapper").hide();
        $(".loading-section").fadeOut();
        $("#loader").delay(1000).fadeOut(400);
        $(".loading").delay(1000).fadeOut(400);
        $(".page1").delay(1000).fadeIn(400).show();
        $("#start-incubator").click(function(){

```

```

if(request == 0 && sistem == 1){
    var countDown = new Date();
    countDown.setDate(countDown.getDate() + 22);
    startDate = countDown;
    updateData({sistem:2,penetasan:countDown,mode:mode,motor
:motor,kipas:kipas,lampu:lampu});
    dayCountdown("enable");
}
});
$("#power").click(function(){
    if(request == 0 && sistem == 2){
        dayCountdown("disable");
        updateData({sistem:1,sisa_hari:-1,penetasan:"0000-00-00
00:00:00"});
        lastEstimation = 24;
    }
});
$("#auto-mode").click(function(){
    if(request == 0 && sistem == 2 && mode != 1){
        updateData({mode:1});
    }
});
$("#manual-mode").click(function(){
    if(request == 0 && sistem == 2 && mode != 2){
        updateData({mode:2});
    }
});
$("#kipas-on").click(function(){
    if(request == 0 && sistem == 2 && mode == 2 && kipas !=
1){
        updateData({kipas:1});
    }
});
$("#kipas-off").click(function(){
    if(request == 0 && sistem == 2 && mode == 2 && kipas !=
2){
        updateData({kipas:2});
    }
});
$("#motor-on").click(function(){

```

```

1){
    if(request == 0 && sistem == 2 && mode == 2 && motor != 1){
        updateData({motor:1});
    }
});
$("#motor-off").click(function(){
    if(request == 0 && sistem == 2 && mode == 2 && motor != 2){
        updateData({motor:2});
    }
});
$("#lampa-on").click(function(){
    if(request == 0 && sistem == 2 && mode == 2 && lampu != 1){
        updateData({lampa:1});
    }
});
$("#lampa-off").click(function(){
    if(request == 0 && sistem == 2 && mode == 2 && lampu != 2){
        updateData({lampa:2});
    }
});

$("#start-incubator").hover(function(){
    if(request == 0 && sistem == 1){
        $("#start-incubator").css("background-color", "#db5863");
    } else return false;
},function(){
    if(request == 0 && sistem == 1){
        $("#start-incubator").css("background-color", "#D92332");
    } else return false;
})
$("#power").hover(function(){
    if(request == 0 && sistem != 1){
        $("#power").css("filter", "grayscale(100%)");
    } else {
        return false;
    }
},function(){

```

```

if(request == 0 && sistem != 1){
    $("#power").css("filter","grayscale(0%)");
} else {
    return false;
}
});
$("#kipas-on").hover(function(){
    if(request == 0 && sistem == 2 && mode == 2 && kipas != 1){
        $("#kipas-on").css("background-color","#db5863");
    } else {
        return false;
    }
},function(){
    if(request == 0 && sistem == 2 && mode == 2 && kipas != 1){
        $("#kipas-on").css("background-color","#C7C4B3");
    } else {
        return false;
    }
});
$("#kipas-off").hover(function(){
    if(request == 0 && sistem == 2 && mode == 2 && kipas != 2){
        $("#kipas-off").css("background-color","#db5863");
    } else {
        return false;
    }
},function(){
    if(request == 0 && sistem == 2 && mode == 2 && kipas != 2){
        $("#kipas-off").css("background-color","#C7C4B3");
    } else {
        return false;
    }
});

$("#motor-on").hover(function(){
    if(request == 0 && sistem == 2 && mode == 2 && motor != 1){
        $("#motor-on").css("background-color","#db5863");
    }
});

```

```
        } else {
            return false;
        }
    },function(){
        if(request == 0 && sistem == 2 && mode == 2 && motor != 1){
            $("#motor-on").css("background-color","#C7C4B3");
        } else {
            return false;
        }
    });
    $("#motor-off").hover(function(){
        if(request == 0 && sistem == 2 && mode == 2 && motor != 2){
            $("#motor-off").css("background-color","#db5863");
        } else {
            return false;
        }
    },function(){
        if(request == 0 && sistem == 2 && mode == 2 && motor != 2){
            $("#motor-off").css("background-color","#C7C4B3");
        } else {
            return false;
        }
    });

    $("#lampu-on").hover(function(){
        if(request == 0 && sistem == 2 && mode == 2 && lampu != 1){
            $("#lampu-on").css("background-color","#db5863");
        } else {
            return false;
        }
    },function(){
        if(request == 0 && sistem == 2 && mode == 2 && lampu != 1){
            $("#lampu-on").css("background-color","#C7C4B3");
        } else {
            return false;
        }
    });
}
```

```
});
$("#lampu-off").hover(function(){
    if(request == 0 && sistem == 2 && mode == 2 && lampu != 2){
        $("#lampu-off").css("background-color","#db5863");
    } else {
        return false;
    }
},function(){
    if(request == 0 && sistem == 2 && mode == 2 && lampu != 2){
        $("#lampu-off").css("background-color","#C7C4B3");
    } else {
        return false;
    }
});
$("#auto-mode").hover(function(){
    if(request == 0 && sistem == 2 && mode != 1){
        $("#auto-mode").css("background-color","#db5863");
    } else {
        return false;
    }
},function(){
    if(request == 0 && sistem == 2 && mode != 1){
        $("#auto-mode").css("background-color","#C7C4B3");
    } else {
        return false;
    }
});
$("#manual-mode").hover(function(){
    if(request == 0 && sistem == 2 && mode != 2){
        $("#manual-mode").css("background-color","#db5863");
    } else {
        return false;
    }
},function(){
    if(request == 0 && sistem == 2 && mode != 2){
        $("#manual-mode").css("background-color","#C7C4B3");
    } else {
        return false;
    }
});
```

```

        });
    });
};

function startWebsocket(){
    console.log("[WEBSOCKETS] Ready to receive WebSocket events.");
    Echo.channel('Component')
        .listen('Suhu', (e) => {
            console.log("[WEBSOCKETS] {#Component} Suhu Event: "
" + e.suhu);
            $("#" + e.suhu).text(e.suhu + "°");
            suhu = e.suhu;
        })
        .listen('Kelembapan', (e) => {
            console.log("[WEBSOCKETS] {#Component} Kelembapan
Event: " + e.kelembapan);
            $("#" + e.kelembapan).text(e.kelembapan);
            kelembapan = e.kelembapan;
        })
        .listen('SisaHari', (e) => {
            console.log("[WEBSOCKETS] {#Component} SisaHari
Event: " + e.sisa_hari);
            $("#" + e.sisa_hari).text(e.sisa_hari);
            sisa_hari = e.sisa_hari;
        })
        .listen('Penetasan', (e) => {
            console.log("[WEBSOCKETS] {#Component} Penetasan
Event: " + e.penetasan);
            penetasan = e.penetasan;
        })
        .listen('Motor', (e) => {
            console.log("[WEBSOCKETS] {#Component} Motor Event: "
+ e.motor);
            motor = e.motor;
            if(sistem == 2){
                if(motor == 1){
                    updateButton("motor",1);
                } else if(motor == 2){
                    updateButton("motor",2);
                }
            }
        })
}

```

```
        })
      .listen('Kipas', (e) => {
        console.log("[WEBSOCKETS] {#Component} Kipas Event: "
+ e.kipas);
        kipas = e.kipas;
        if(sistem == 2){
          if(kipas == 1){
            updateButton("kipas",1);
          } else if(kipas == 2){
            updateButton("kipas",2);
          }
        }
      })
      .listen('Lampu', (e) => {
        console.log("[WEBSOCKETS] {#Component} Lampu Event: "
+ e.lampu);
        lampu = e.lampu;
        if(sistem == 2){
          if(lampu == 1){
            updateButton("lampu",1);
          } else if(lampu == 2){
            updateButton("lampu",2);
          }
        }
      })
      .listen('Mode', (e) => {
        console.log("[WEBSOCKETS] {#Component} Mode Event: " +
e.mode);
        mode = e.mode;
        if(sistem == 2){
          if(mode == 1){
            updateButton("mode",1);
          } else if(mode == 2){
            updateButton("mode",2);
          }
        }
      })
      .listen('Sistem', (e) => {
        console.log("[WEBSOCKETS] {#Component} Sistem Event: "
+ e.sistem);
        sistem = e.sistem;
```

```
        if(sistem == 1){
            updateButton("sistem",1);
        } else if(sistem == 2){
            updateButton("sistem",2);
        }
    });
}

function loginButton(){
    let userInput = document.querySelector("#user-ipt");
    let passInput = document.querySelector("#pass-ipt");
    let userValue = userInput.value.trim();
    let passValue = passInput.value.trim();
    if(userValue.length == 0 || passValue.length == 0){
        document.querySelector(".login-info").innerHTML = "Please insert
username and password!";
        return false;
    } else if(userInput.value == "Bintang" && passInput.value ==
"12345") {
        loggedIn = 1;
        document.querySelector(".login-info").style.color = "#56C08D"
        document.querySelector(".login-info").innerHTML = "Successfully
logged in!";
        getData();
        $("#loader").delay(500).fadeIn(400);
        $(".loading").delay(500).fadeIn(400);
        setTimeout(function() {
            $(".page1").hide();
            $(".monitor-wrapper").show();
            setInterval(everySec, 1000);
            everySec();
        }, 1000);
        return true;
    } else {
        document.querySelector(".login-info").innerHTML = "Incorrect
username or password!";
        return false;
    }
}
```

D. Source Code Arduino

```

#include <dht.h>
#include <ArduinoJson.h>
#include <ESP8266HTTPClient.h>
#include <ArduinoWebsockets.h>
#include <ESP8266WiFi.h>

#define pinDHT D3
#define pinKipas D4
#define pinMotor D5
#define pinLampu D7

const char* ssid = "hotspot1024";
const char* password = "12345678";
const char* requestUrl =
"http://ecato.my.id/api/incubator/komponen";
const char* websockets_connection_string =
"wss://ecato.my.id:6002/app/ABCDEFG?protocol=7&client=js&version=
4.3.1&flash=false";
const char echo_org_ssl_fingerprint[] PROGMEM = "B4 A0 D6 76 1B
68 01 54 43 A7 33 45 AB D5 73 CE 06 09 40 83";
unsigned long previousMillis1 = 0;
const long interval1 = 60000;
unsigned long previousMillis2 = 0;
const long interval2 = 3000;
unsigned long previousMillis3 = 0;
const long interval3 = 500;

int sistem, _mode, kipas, motor, lampu, lastTemp, lastHumi,
dummyData;

using namespace websockets;

WebsocketsClient client;
dht DHT;
HTTPClient http;
WiFiClient client2;

void getData(){
    http.begin(client2,requestUrl);
    int code = http.GET();
    if(code > 0){
        String response = http.getString();
        Serial.print("[HTTP] GET DATA - Code : ");
        Serial.println(code);
        StaticJsonDocument<512> doc;
        deserializeJson(doc, response);
        sistem = doc["sistem"];
        _mode = doc["mode"];
        kipas = doc["kipas"];
        motor = doc["motor"];
    }
}

```

```

        lampu = doc["lampu"];
    } else{
        Serial.print("[HTTP] GET DATA - Error on sending request,
code : ");
        Serial.println(code);
    }
    http.end();
}

void updateDHT(){
    DHT.read11(pinDHT);
    String dhtJson; StaticJsonDocument<256> doc;
    if((lastTemp != DHT.temperature) || (lastHumi != DHT.humidity)) && (!isnan(DHT.temperature)) || (!isnan(DHT.humidity))){
        http.begin(client2,requestUrl);
        http.addHeader("Content-Type", "application/json");
        if(lastTemp != DHT.temperature){
            lastTemp = DHT.temperature;
            doc["suhu"] = lastTemp;
        }
        if(lastHumi != DHT.humidity){
            lastHumi = DHT.humidity;
            doc["kelembapan"] = lastHumi;
        }
        serializeJson(doc,dhtJson);
        int httpCode = http.PUT(dhtJson);
        if (httpCode > 0){
            String response = http.getString();
            Serial.println(response);
        }
        http.end();
    }
}

void onMessageCallback(WebsocketsMessage message) {
    String messageData = message.data();
    StaticJsonDocument<512> doc1, doc2;
    deserializeJson(doc1, message.data());
    if(messageData.indexOf("Sistem") != -1){
        Serial.println("Sistem change");
        deserializeJson(doc2,doc1["data"]);
        sistem = doc2["sistem"];
    }
    if(sistem == 1){
        return;
    } else if(sistem == 2){
        if(messageData.indexOf("Mode") != -1){
            Serial.println("Mode change");
            deserializeJson(doc2,doc1["data"]);
            mode = doc2["mode"];
            if(motor == 1){

```

```

        motor = 2;
        updateMotor(motor);
    } else if(motor == 2) {
        motor = 1;
        updateMotor(motor);
    }
}

if(_mode == 1){
    return;
} else if(_mode == 2){
    if(messageData.indexOf("Kipas") != -1){
        Serial.println("Kipas change");
        deserializeJson(doc2,doc1["data"]);
        kipas = doc2["kipas"];
    }
    if(messageData.indexOf("Motor") != -1){
        Serial.println("Motor change");
        deserializeJson(doc2,doc1["data"]);
        motor = doc2["motor"];
    }
    if(messageData.indexOf("Lampu") != -1){
        Serial.println("Lampu change");
        deserializeJson(doc2,doc1["data"]);
        lampu = doc2["lampu"];
    }
}
}

void updateMotor(int state){
    StaticJsonDocument<256> doc; String dhtJson;
    http.begin(client2,requestUrl);
    http.addHeader("Content-Type", "application/json");
    doc["motor"] = state;
    serializeJson(doc,dhtJson);
    http.PUT(dhtJson);
    String response = http.getString();
    Serial.println(response);
    http.end();
}

void updateComp(int state){
    StaticJsonDocument<128> doc; String dhtJson;
    if(dummyData != state){
        dummyData = state;
        http.begin(client2,requestUrl);
        http.addHeader("Content-Type", "application/json");
        if(state == 2){
            doc["kipas"] = 1;
            doc["lampu"] = 2;
        } else if (state == 1){
            doc["kipas"] = 2;
        }
    }
}

```

```

        doc["lampu"] = 1;
    }
    serializeJson(doc, dhtJson);
    http.PUT(dhtJson);
    String response = http.getString();
    Serial.println(response);
    http.end();
}
}

void setup() {
    Serial.begin(115200);
    // Connect to wifi
    WiFi.begin(ssid, password);

    // Wait some time to connect to wifi
    for(int i = 0; i < 10 && WiFi.status() != WL_CONNECTED; i++)
    {
        Serial.print(".");
        delay(1000);
    }

    getData();

    client.onMessage(onMessageCallback);
    client.setFingerprint(echo_org_ssl_fingerprint);
    client.connect(websockets_connection_string);
    client.send("{\"event\":\"pusher:subscribe\", \"data\":{\"channel\":\"Component\"}}");
    client.ping();

    pinMode(pinKipas,OUTPUT);
    pinMode(pinMotor,OUTPUT);
    pinMode(pinLampu,OUTPUT);
}

void loop() {
    if(client.available()) {
        client.poll();
    } else {
        client = {};
        client.onMessage(onMessageCallback);
        client.setFingerprint(echo_org_ssl_fingerprint);
        client.connect(websockets_connection_string);
        client.send("{\"event\":\"pusher:subscribe\", \"data\":{\"channel\":\"Component\"}}");
        client.ping();
    }

    if(sistem == 1){
        digitalWrite(pinKipas,1);
        digitalWrite(pinMotor,1);
    }
}

```

```

        digitalWrite(pinLampu,1);
    } else if(sistem == 2){
        unsigned long currentMillis2 = millis();
        if(currentMillis2 - previousMillis2 >= interval2){
            previousMillis2 = currentMillis2;
            updateDHT();
        }
        if(_mode == 1){
            unsigned long currentMillis1 = millis();
            if(currentMillis1 - previousMillis1 >= interval1){
                previousMillis1 = currentMillis1;
                if(motor == 1){
                    Serial.println("Motor hidup!");
                    digitalWrite(pinMotor,0);
                    updateMotor(1);
                    motor = 2;
                } else if(motor == 2){
                    Serial.println("Motor mati!");
                    digitalWrite(pinMotor,1);
                    updateMotor(2);
                    motor = 1;
                }
            }
            if(lastTemp >= 39 || lastHumi <= 50){
                digitalWrite(pinKipas,0);
                digitalWrite(pinLampu,1);
                kipas = 1;
                lampu = 2;
                updateComp(2);
            } else if(lastTemp <= 37 || lastHumi >= 60){
                digitalWrite(pinKipas,1);
                digitalWrite(pinLampu,0);
                kipas = 2;
                lampu = 1;
                updateComp(1);
            }
            } else if(_mode == 2){
                if(kipas == 1) digitalWrite(pinKipas,0); else if(kipas == 2) digitalWrite(pinKipas,1);
                if(motor == 1) digitalWrite(pinMotor,0); else if(motor == 2) digitalWrite(pinMotor,1);
                if(lampu == 1) digitalWrite(pinLampu,0); else if(lampu == 2) digitalWrite(pinLampu,1);
            }
        }
    }
}

```