BAB 4
HASIL DAN PEMBAHASAN

4.1 Deksripsi Sistem

Sistem yang dirancang adalah sistem kontrol utama untuk SPBKLU di BRIN Serpong. Sistem kontrol utama ini berfungsi sebagai *master* yang berfungsi untuk mengontrol SPBKLU dengan memanfaatkan subkontrol yang berada pada setiap loker yaitu ESP32 mengontrol loker sebagai *slave*. Sistem kontrol utama dapat mengetahui keadaan SPBKLU secara menyeluruh dan mengontrolnya sehingga SPBKLU dapat melakukan transaksi *battery swap*. Hal ini dimungkinkan karena setiap subkontrol mengirimkan data sensor, baterai, hingga status solenoid yang dimilikinya ke sistem kontrol utama. Data tersebut akan menjadi parameter untuk menentukan apakah SPBKLU dapat melakukan transaksi *battery swap* atau tidak serta dalam keadaan tertentu menentukan apakah SPBKLU perlu dimatikan atau tidak dan lain-lain. Beberapa data yang diterima oleh sistem kontrol utama seperti keadaan loker dan juga baterai akan ditampilkan pada HMI untuk memberikan informasi kepada pengguna terkait keadaan SPBKLU.

Komunikasi data antara sistem kontrol utama, subkontrol, dan pengguna dilakukan menggunakan protokol MQTT. Namun, dikarena penggunaan server sebagai broker kurang stabil, maka untuk komunikasi data yang dilakukan pada internal SPBKLU yaitu antara sistem kontrol utama dengan subkontrol dilakukan dengan menggunakan broker lokal yang memanfaatkan device sistem kontrol utama yaitu Raspberry pi sebagai broker. Oleh karena itu, untuk menghubungkan

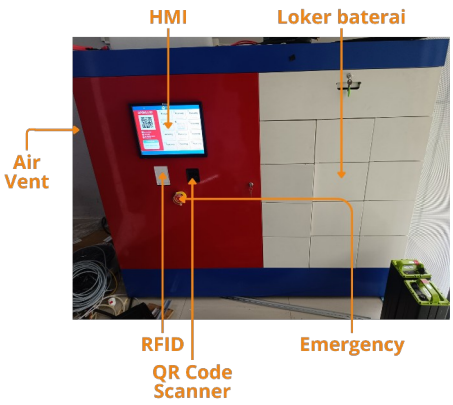
antara broker lokal dan broker server agar sistem kontrol utama dapat tetap terhubung dengan *back-end* dan pengguna namun tetap mempertahankan penggunaan broker lokal untuk komunikasi data internal diperlukan MQTT *bridge*.

4.2 Perancangan *Hardware*

Pada tahap ini akan dibahas mengenai rancangan *hardware* SPBKLU yang dibagi dalam rancangan mekanik dan elektronik.

4.2.1 Perancangan Mekanik SPBKLU

Hasil rancangan mekanik SPBKLU dapat dilihat pada gambar di bawah. Pada bagian samping kiri gambar terdapat 2 vetilasi udara yang berfungsi untuk menjaga suhu SPBKLU agar tidak terlalu panas. Pada bagian depan kiri, terdapat HMI yang berfungsi untuk komunikasi antara SPBKLU dengan pengguna serta RFID dan QR Scanner yang berfungsi untuk autentifikasi pengguna ketika proses *battery swap*. Pada bagian depan kanan terdapat loker baterai dengan total 12 loker yang berfungsi untuk *charge* baterai.



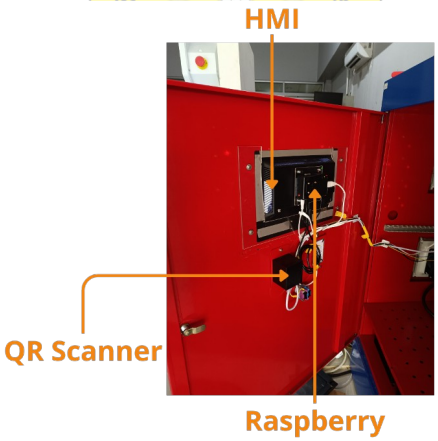
Gambar 4.1: Desain SPBKLU

Bagian merah berisikan sistem kelistrikan SPBKLU,seperti MCCB, SPD, *Power Supply*, WiFi, serta sistem kelistrikan lainnya. Hal ini juga yang

menyebabkan ventilasi udara terdapat di bagian kiri gambar atau bagian kanan SPBKLU.

Gambar 4.2: Tampilan sistem kelistrikan SPBKLU dan ventilasi udara

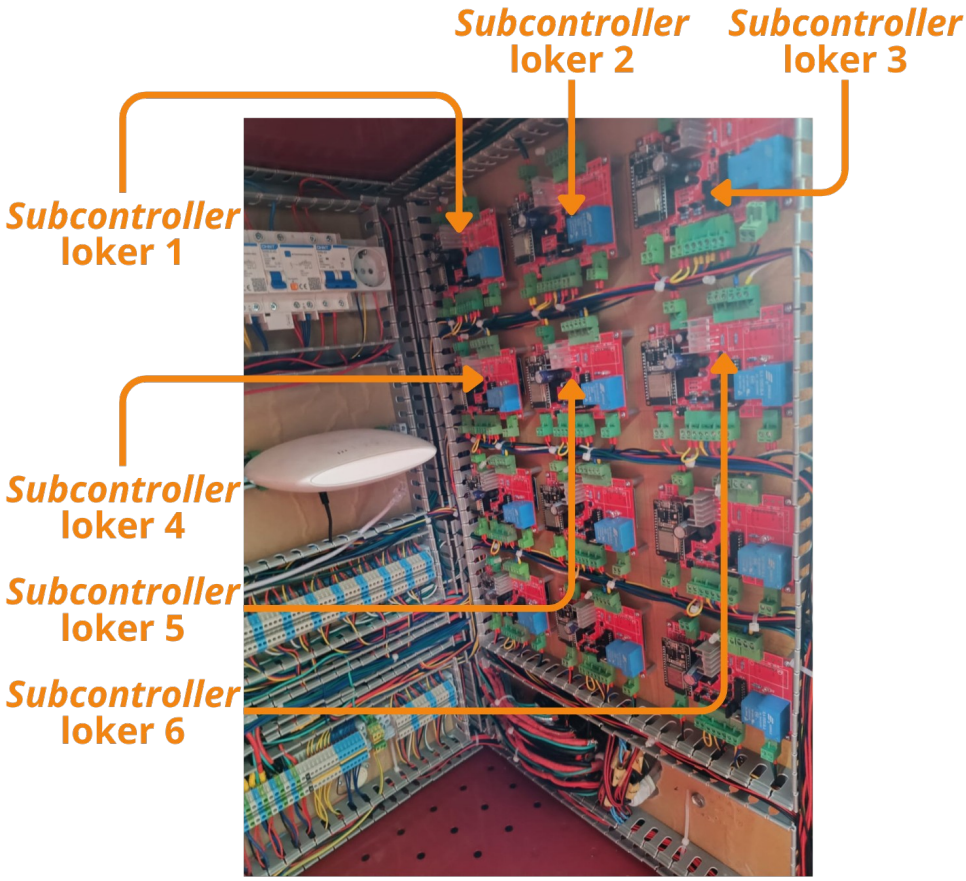
Adapun untuk Raspberry yang menjadi sistem kontrol utama terletak tepat di belakang HMI.



Gambar 4.3: Tampilan Raspberry dan HMI dari dalam

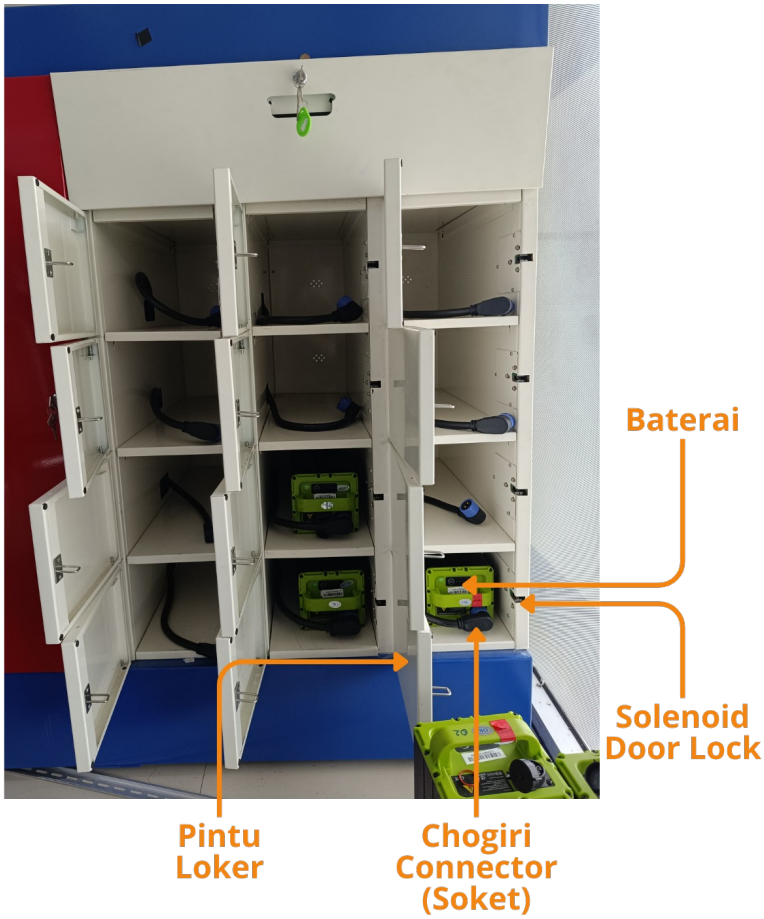
Sedangkan untuk *subcontroller* tiap loker terdapat pada sisi kiri bagian dalam dari SPBKLU. *Subcontroller* tersusun secara berurutan dari atas kiri hingga bawah kanan mulai dari *subcontroller* loker 1 hingga *subcontroller* loker 12.

Charger unit terletak di bagian belakang dari loker baterai sehingga tidak tampak pada gambar.



Gambar 4.4: Tampilan subcontroller loker pada bagian dalam SPBKLU

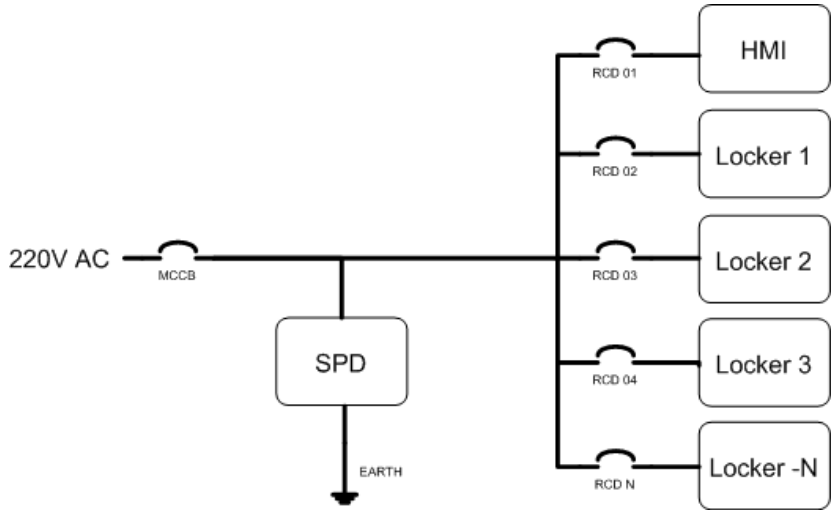
Subcontroller terhubung dengan loker yang sudah ditentukan dan berfungsi untuk mengontrol loker tersebut. Pada setiap loker terdapat soket baterai dengan tipe chogiri, *solenoid door lock*, tempat baterai, dan pintu loker.



Gambar 4.5: Tampilan bagian dalam loker

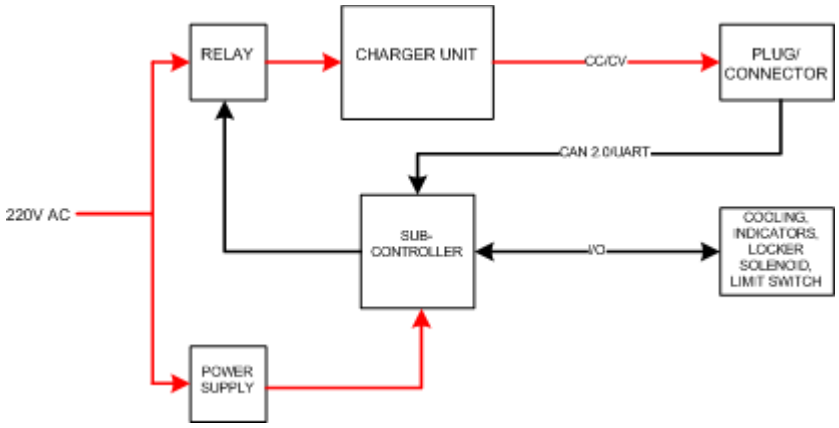
4.2.2 Wiring SPBKLU

SPBKLU memiliki *wiring* yang dapat dilihat pada gambar di bawah. Listrik 220V dalam bentuk listrik AC sebagai catu daya. Terdapat beberapa sistem proteksi sebelum catu daya terhubung ke HMI dan locker-locker, yaitu MCCB (*Molded Case Circuit Breaker*), RCD (*residual current device*), dan SPD (*Surge Protection Device*). Catu daya terhubung secara seri dengan MCCB dan terhubung secara paralel dengan SPD. Nantinya listrik dari catu daya ini akan dibagi menjadi 14 cabang, yaitu 1 cabang untuk HMI, 1 cabang untuk *protection controller*, dan 12 cabang untuk setiap loker. Untuk setiap cabang akan diproteksi dengan RCD.



Gambar 4.6: Wiring SPBKLU

Untuk wiring pada loker SPBKLU dapat dilihat pada gambar di bawah. Listrik 220V akan digunakan sebagai suplai listrik untuk *charge* baterai dan juga *Subcontroller*. Dikarenakan *subcontroller* dan *charge* baterai membutuhkan suplai listrik DC berbeda dimana *subcontroller* membutuhkan listrik DC 12V sedangkan *charge* baterai membutuhkan listrik 84V, maka untuk *subcontroller* akan disuplai oleh *power supply* dengan *output* 12V/10A dan *charge* baterai disuplai oleh *charge unit* dengan *output* 84V/5-10A.



Gambar 4.7: Wiring Loker SPBKLU

Connector merupakan soket tempat baterai dihubungkan ke SPBKLU. Ketika baterai dihubungkan ke soket, maka *subcontroller* akan membaca

informasi dari BMS yang ada pada baterai seperti suhu, tegangan, arus, SoC, dan SOH baterai.

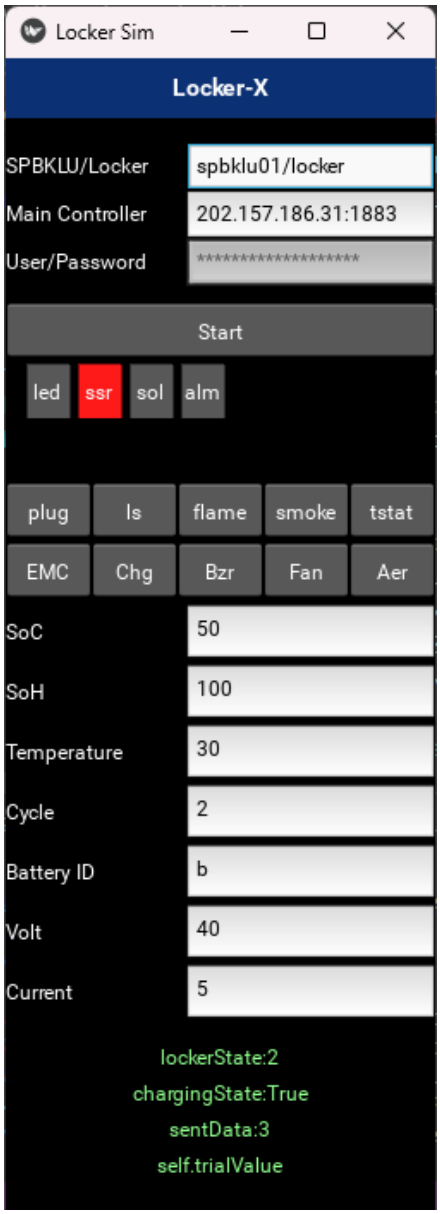
Subcontroller terhubung dengan relay, solenoid *door lock*, dan sensor-sensor. Relay akan menutup ketika SoC baterai kurang dari 100, pintu tertutup, dan temp baterai di bawah 45 °C.

4.3 Pemrograman *Software*

Pada subab ini akan dibahas mengenai perancangan *software* dari sistem kontrol utama SPBKLU. Pembahasan akan dimulai dengan pembahasan mengenai program locker simulator, perancangan HMI, kemudian dilanjut ke perancangan komunikasi data, *battery swap*, data log, dan terakhir adalah *eror handling*. Pembahasan akan dilakukan dengan mencantumkan *source code* program kemudian membahasnya pada baris selanjutnya menggunakan dengan memanfaatkan *flowchart*, tulisan, serta gambar.

4.3.1 Program Locker Simulator

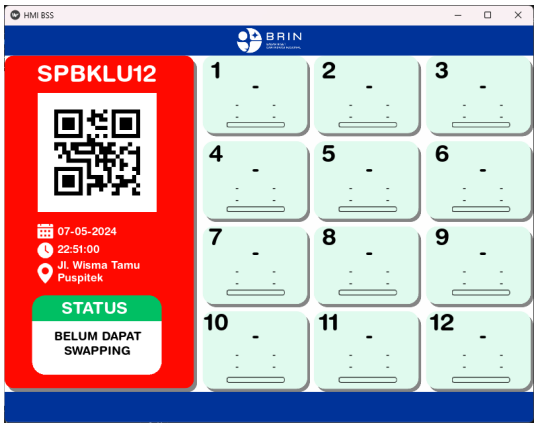
Program locker simulator seperti namanya digunakan untuk mensimulasikan loker pada SPBKLU. Program ini berfungsi sebagai pengganti loker SPBKLU ketika penulis tidak dapat menguji coba alat SPBKLU secara langsung. Fitur yang ada dan cara kerja dari loker simulator ini sama persis dengan cara kerja loker SPBKLU asli sehingga penggunaan loker simulator ini sangat membantu dalam proses penelitian dan uji coba SPBKLU terutama saat penulis tidak dapat secara langsung melakukan uji coba pada SPBKLU asli. Berikut adalah tampilan dari loker simulator.



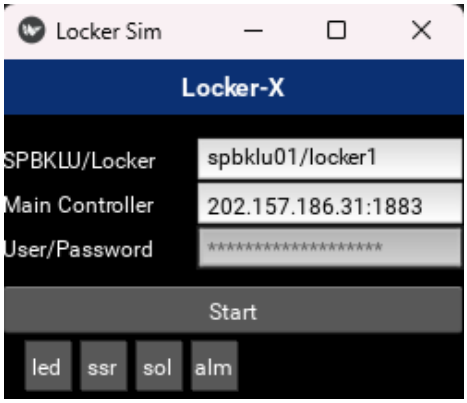
Gambar 4.8: Tampilan loker simulator

Pada bagian atas pada loker simulator terdapat text yang dapat di edit. Text paling atas akan menentukan loker yang ingin disimulasikan, text baris kedua merupakan address dari broker yang digunakan, sedangkan yang berbintang merupakan username dan password. Jika penulis menuliskan “spbklu01/locker1” pada baris pertama dan address broker server SPBKLU kemudian menekan

tombol “start” maka loker akan mensimulasikan loker 1 pada SPBKLU. Dapat dilihat pada gambar di bawah ketika locker simulator dijalankan, tampilan HMI akan berubah. Pada 4 buah kotak yang terletak di bawah tombol “Start” terdapat kotak dengan tulisan “led”, kotak itu merupakan perumpamaan LED yang ada pada loker dimana akan berubah warna sesuai dengan status loker, hal yang sama juga berlaku pada HMI.

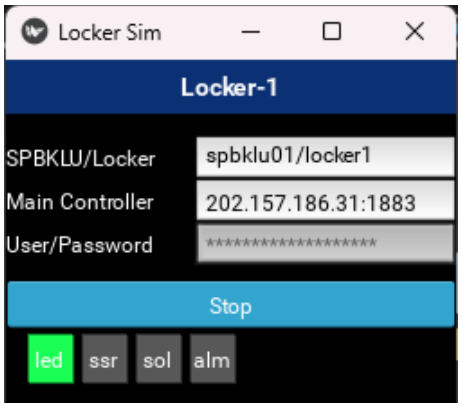


Gambar 4.9: Tampilan HMI sebelum locker simulator dijalankan



Gambar 4.10: Tampilan locker simulator sebelum dijalankan

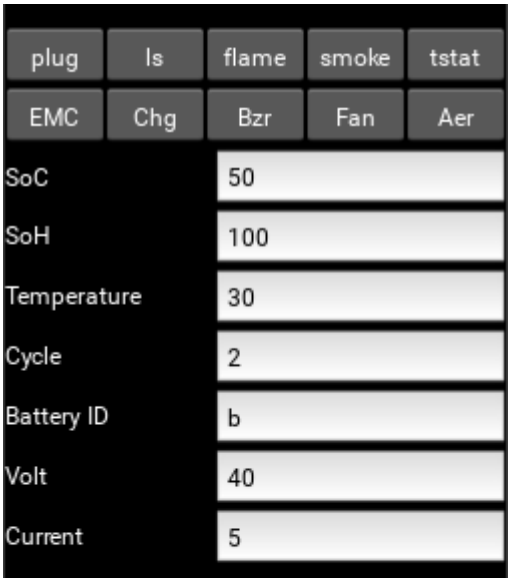
Gambar 4.11: Tampilan HMI ketika locker simulator dijalankan untuk mensimulasikan loker 1



Gambar 4.12: Tampilan locker simulator setelah dijalankan

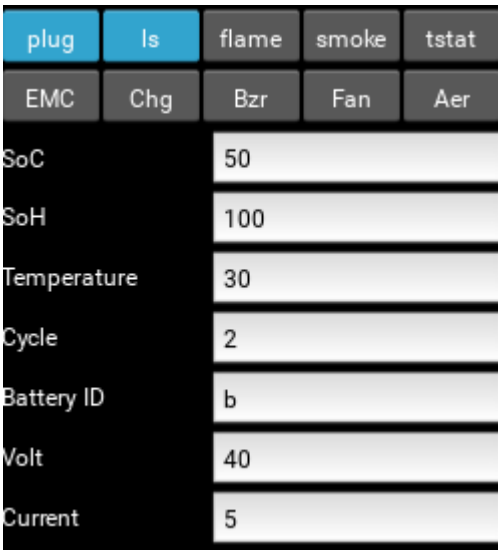
Locker simulator juga memiliki fitur yang memungkinkan untuk melakukan simulasi seperti membuka pintu loker, loker terhubung dengan baterai, sensor api menyala, sensor asap menyala, tomobol emergency, serta menentukan SoC baterai yang terhubung, SoH baterai, suhu baterai, cycle baterai, ID baterai, tegangan *charge*, dan arus *charge*. Jika tombol “ls” yang mensimulasikan limit switch pada pintu loker dan tombol “plug” yang mensimulasikan terhubungnya

baterai dengan loker ditekan, maka tampilan loker 1 pada HMI akan berubah dari yang sebelumnya “TERBUKA” menjadi menampilkan nilai-nilai baterai yaitu SoC bernilai 50, SoH bernilai 100, tegangan bernilai 40, serta arus yang bernilai 5.

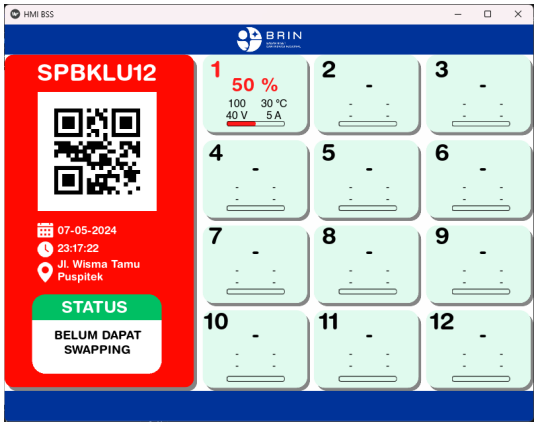


Gambar 4.13: Tampilan locker simulator sebelum tombol “plug” dan "ls" pada locker simulator ditekan

Gambar 4.14: Tampilan HMI sebelum tombol “plug” dan "ls" pada locker simulator ditekan



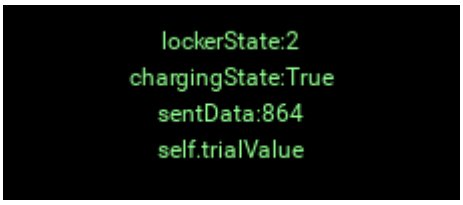
Gambar 4.15: Tampilan locker simulator setelah tombol "plug" dan "ls" pada locker simulator ditekan



Gambar 4.16: Tampilan HMI setelah tombol "plug" dan "ls" pada locker simulator ditekan

Adapun untuk bagian paling bawah pada locker simulator menampilkan status loker saat ini, status *charge* saat ini, serta jumlah pengiriman data yang dilakukan oleh locker simulator. Gambar di bawah menunjukan bahwa status locker 1 adalah 2 yang berarti terdapat baterai yang sedang di *charge*, status

charge adalah *true* menandakan bahwa SoC baterai masih di bawah 100, dan senData bernilai 864 menandakan bahwa lockerSim telah mengirimkan data sebanyak 864 kali.



Gambar 4.17: Tampilan bagian bawah locker simulator

4.3.2 Perancangan HMI

Pendahuluan tentang file kivi

File kv merupakan file yang digunakan untuk mendefinisi aturan, template, dan widget root untuk pemrograman menggunakan kivi. Dengan menggunakan file kv, penulisan program dapat lebih rapi sehingga dapat lebih mudah dipahami dan dimodifikasi. Berikut adalah program yang ada di dalam file HMI.kv yang berfungsi mengatur tampilan HMI.

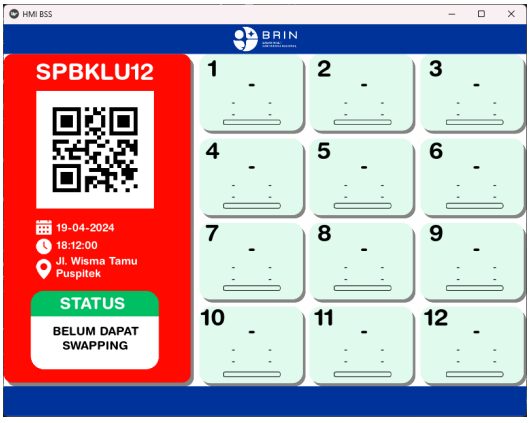
Pemprograman file kv

```
<Background>:
   name: "Background"
   FloatLayout:
      pos: 0,0
      size: root.width, root.height
      #Background
      Image:
          source:'Assets/hmi3.png'
          allow_stretch: True
      #Barcode
      Image:
          id: img_barcode
          size_hint: (0.3, 0.3)
          pos_hint: {'x': .025, 'y': .53}
          source: self.source_img
```

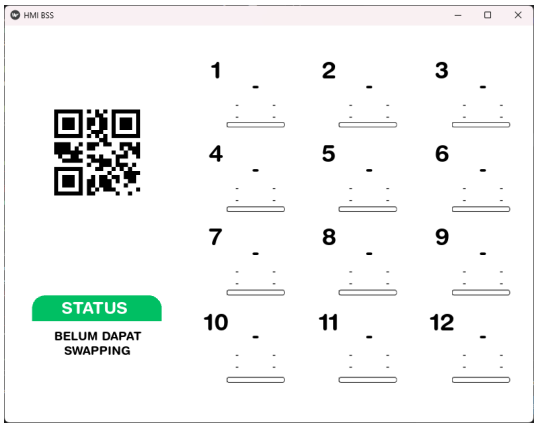
```
            source_img: ''
        #Icon date, hour, dan location
        Image:
            size_hint: (0.15, 0.15)
            pos_hint: {'x': 0.702, 'y': 0.35}
            source:'Assets/Icon Gabung.png'
```

Pada awal program terdapat *class rule* dengan nama “Background” yang berfungsi untuk mengatur aturan dari *class* “Background”, penulisan *class rule* ditandai dengan penggunaan “<>”. Kemudian “FloatLayout” merupakan *root rule* yang berfungsi untuk memanggil modul “kivy.uix.floatlayout”, *root rule* dideklarasikan secara langsung tanpa simbol tambahan. Untuk membuat “FloatLayout” berada di tengah layar dan juga besarnya dapat disesuaikan, maka ditetapkan “pos : (0,0)” dan “size:root.width,root.height”.

Rancang bangun tampilan HMI selanjutnya adalah *background*. Untuk menampilkan gambar yang akan digunakan sebagai *background* HMI digunakan *widget* “Image”, *widget* ini yang berfungsi untuk menampilkan gambar. Gambar yang akan digunakan ada dalam file “hmi3.png” pada folder “Assets”, untuk menampilkan gambar dari file tersebut dapat dituliskan “source:'Assets/hmi3.png'”. “a*llow_strecth = True”* berfungsi untuk membuat ukuran dari gambar akan menyesuaikan dengan ukuran bingkai floatlayout.Untuk ukuran dan posisinya karena tidak ditentukan maka panjang dan lebar gambar akan memiliki perbandingan yang sesuai dengan gambar aslinya dan posisinya akan berada di tengah bingkai.

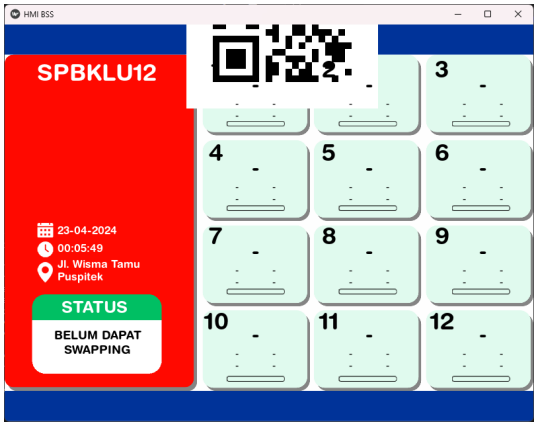


Gambar 4.18: Tampilan HMI normal

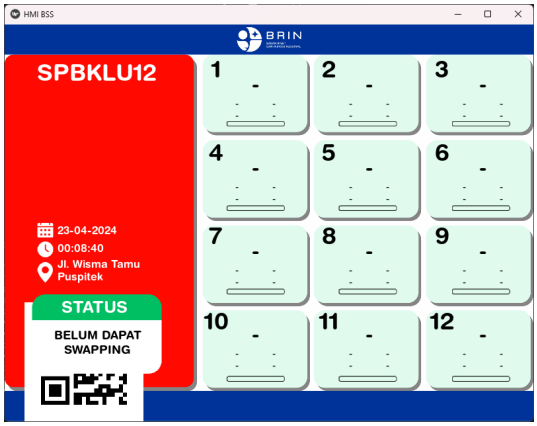


Gambar 4.19: Tampilan HMI tanpa background

Setelah background diterapkan pada tampilan HMI selanjutnya adalah menampilkan barcode. Sama halnya dengan menampilkan gambar *background* dimana kita memanggil gambar menggunakan *widget "image"* tapi nama barcode dikosongkan karena barcode akan terus berganti dalam selang beberapa waktu dengan menggunakan barcode yang memiliki kode berbeda, maka nama gambar yang dipanggil akan menyesuaikan dengan nama gambar barcode yang dibuat oleh program utama. Setelah itu, untuk menyesuaikan ukuran dan posisi barcode dengan *background*, maka ukuran panjang dan lebarnya akan di setel ke 0.3 kali dari panjang dan lebar gambar aslinya dan posisinya akan berada di 0.25 dari panjang bingkai dan 0.53 dari lebar bingkai. Berikut adalah tampilan HMI jika ukuran maupun posisi gambar barcode tidak diatur.

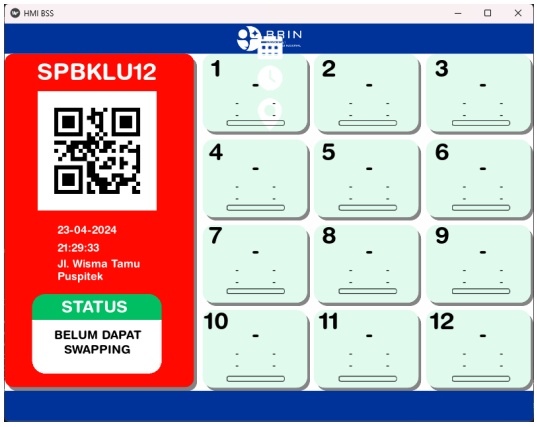


Gambar 4.20: Tampilan HMI ketika ukuran barcode tidak ditentukan

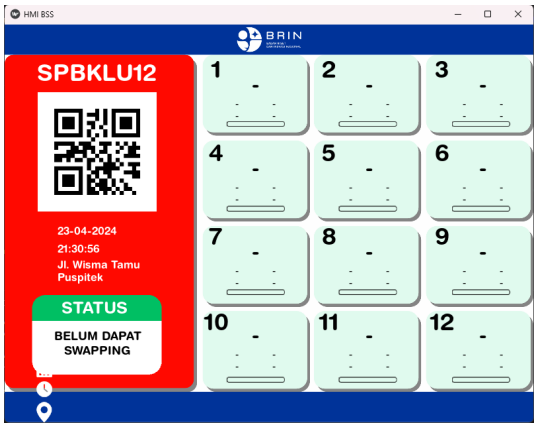


Gambar 4.21: Tampilan HMI ketika posisi barcode tidak ditentukan

Tahap selanjutnya adalah memberikan gambar ikon tanggal, waktu, dan tempat. Gambar dipanggil menggunakan *widget "image"*, kemudian ukuran dan posisinya ditentukan. Ukuran akan ditentukan dengan memberikan nilai panjang dan lebar sebesar 0.15 dari ukuran aslinya dan posisi 0,702 dari panjang bingkai dan 0,35 dari lebar bingkai.



Gambar 4.22: Tampilan HMI ketika ukuran gambar ikon tidak ditentukan

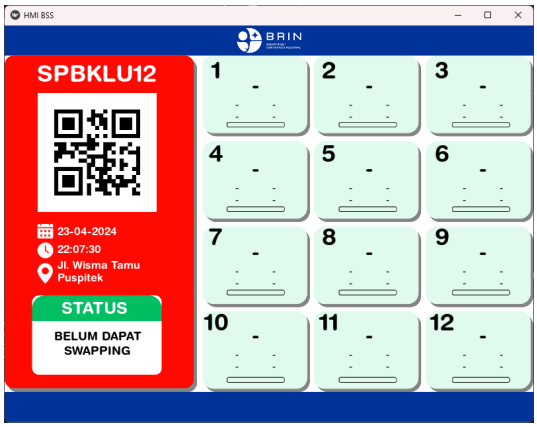


Gambar 4.23: Tampilan HMI ketika posisi ikon tidak ditentukan

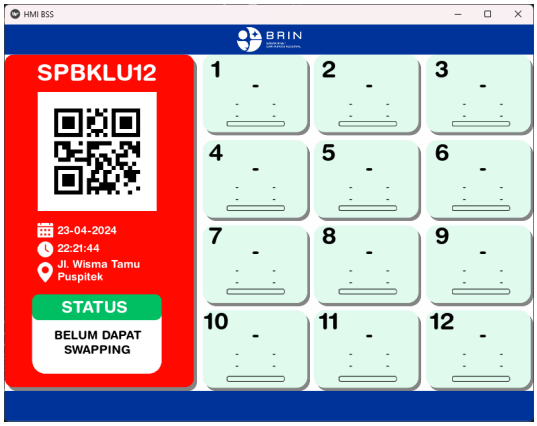
```
Label:
   id: rounded_rectangle_keterangan_status
   pos_hint: {'x': 0.052, 'y': 0.120}
   size_hint: (0.245, 0.2)
   canvas:
      Color:
         rgba: rgba('#ffffff')
      RoundedRectangle:
         pos: self.pos
         size: self.size
         radius: [(20, 20), (20, 20), (20, 20), (20, 20)]
Label:
   id: rounded_rectangle_status
   pos_hint: {'x': 0.052, 'y': 0.255}
   size_hint: (0.245, 0.065)
   col_status: (0/255, 191/255, 99/255, 1)
   canvas:
      Color:
         rgba: self.col_status
      RoundedRectangle:
         pos: self.pos
         size: self.size
         radius: [(20, 20), (20, 20), (0, 0), (0, 0)]
```

Status SPBKLU merupakan suatu yang penting untuk diperhatikan pengguna. Oleh karena itu, tulisan perlu lebih dimunculkan agar dapat lebih diperhatikan. Untuk membuatnya lebih diperhatikan, maka warna di sekitar status SPBKLU akan sedikit dibedakan dengan memberinya bingkai berwarna putih dan hijau. Penambahan bingkai berbentuk persegi panjang dengan dua warna dapat dilakukan dengan menambahkan persegi panjang berwarna putih dengan sudut tumpul kemudian menimpa sebagiannya dengan persegi panjang yang memiliki sudut tajam pada bagian bawah dan sudut tumpul pada bagian bawah. Untuk menambahkan persegi panjang berwarna putih dengan sudut tumpul dapat dilakukan dengan menggunakan “canvas”. Persegi panjang akan ditampilkan dalam warna putih sehingga menggunakan kode “rgba(“#ffffff”)” dan karena setiap sudutnya dibuat sudut tumpul maka radius sudutnya diberi nilai 20, setelah warna dan sudut ditentukan nilainya selanjutnya meletakannya pada posisi dan memberikannya ukuran yang sesuai yaitu dengan posisi pada 0.52 dari panjang bingkai dan 0.12 dari lebar bingkai dan ukuran 0.242 dari panjang bingkai dan 0.2 dari lebar bingkai. Untuk warna default dari gambar adalah putih, penambahan kode “rgba(“#ffffff”)” adalah untuk mempermudah kode dibaca oleh programer.

Untuk menampilkan bingkai berwarna hijau dapat dilakukan seperti menampilkan bingkai putih tapi dibedakan pada bagian pengaturan warna dan radius sudutnya. Warna hijau pada bingkai dapat ditampilkan dengan memasukkan kode warna (0/255, 191/255, 99/255, 1) dan untuk radius sudutnya bernilai 20 untuk sudut bagian atas dan bernilai 0 untuk sudut bagian bawah.



Gambar 4.24: Tampilan bilah status jika radius sudut bingkai putih status tidak ditentukan



Gambar 4.25: Tampilan bilah status apabila radius sudut bingkai hijau status tidak ditentukan

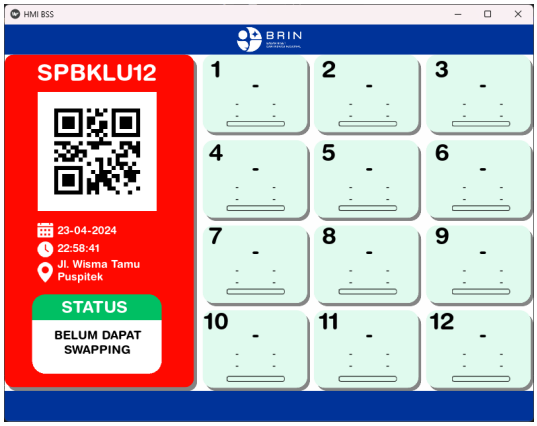
Gambar 4.26: Tampilan bilah status jika bingkai hijau status tidak ditentukan warnanya

```
#Keterangan status
Label:
   id: lbl_keterangan_status
   text: "-"
   font_size: 18
   text_size: self.size
   halign: "center"
   valign: "center"
   size_hint: (0.22, 0.2)
   font_name: "Assets/Helvetica-Rounded.otf"
   pos_hint: {'x': .067, 'y': .1}
   color: rgba('#000000')

#Status
Label:
   id: lbl_status
   text: "STATUS"
   font_size: 25
   font_name: "Assets/Helvetica-Rounded.otf"
   pos_hint: {'x': -0.33, 'y': -0.21}
   color: rgba('ffffff')

#SPBKLU01
Label:
   id: spbklu
   text: "SPBKLU12"
   font_size: 35
   font_name: "Assets/Helvetica-Rounded.otf"
   pos_hint: {'x': -.325, 'y': .380}
#Date
Label:
   id: clock_dt
   text: ""
   halign: "left"
   text_size: self.size
   size_hint: (0.4, 0.04)
   font_size: 16
   font_name: "Assets/Helvetica-Rounded.otf"
   pos_hint: {'x': .1, 'y': .468}
   color: rgba('#ffffff')
#Hour
Label:
   id: clock_h
   text: ""
   font_size: 16
   halign: "left"
   text_size: self.size
   size_hint: (0.4, 0.04)
   font_name: "Assets/Helvetica-Rounded.otf"
   pos_hint: {'x': .1, 'y': .422}
   color: rgba('#ffffff')
#Location
Label:
   id: loc
   text: "Jl. Wisma Tamu \nPuspitek"
   font_size: 16
   halign: "left"
   text_size: self.size
   size_hint: (0.4, 0.1)
   font_name: "Assets/Helvetica-Rounded.otf"
   pos_hint: {'x': .1, 'y': .349}
```

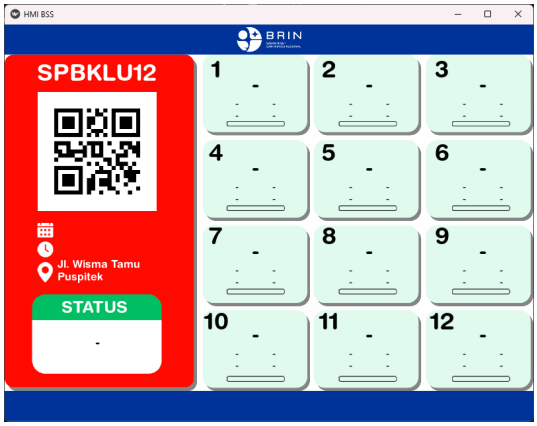
Selanjutnya adalah menampilkan tulisan pada HMI yang dapat dilakukan dengan menggunakan *widget "label"*. Dikarenakan penggunaan *widget* ini cukup

banyak dan penggunaannya yang serupa, maka pembahasan hanya akan dilakukan pada beberapa baris program.

Label digunakan untuk menampilkan tulisan berupa keterangan status yang berada di urutan paling atas dalam program yang dicantumkan di atas. Label untuk keterangan status ini diberikan id “bl_keterangan_status” dengan tulisan yang ditampilkan berupa “-”. Setelah itu, menentukan font yang digunakan, ukuran, posisi, warna, serta *allign* yang ingin digunakan dalam text. Tulisan yang ditentukan untuk ditampilkan adalah “-” tapi pada tampilan HMI dapat berubah-ubah sesuai keadaan SPBKLU. Hal ini dapat terjadi karena *widget “label”* ini telah diberi id yang memungkinkan program utama untuk mengubah text yang ditampilkan oleh *widget “label”* yang berada dalam file kv. Selain text, program utama dapat mengubah pengaturan lain juga dengan mengakses *widget “label”* pada file kv menggunakan id yang telah ditentukan.



Gambar 4.27: Tampilan text pada bilah status setelah dirubah oleh program utama

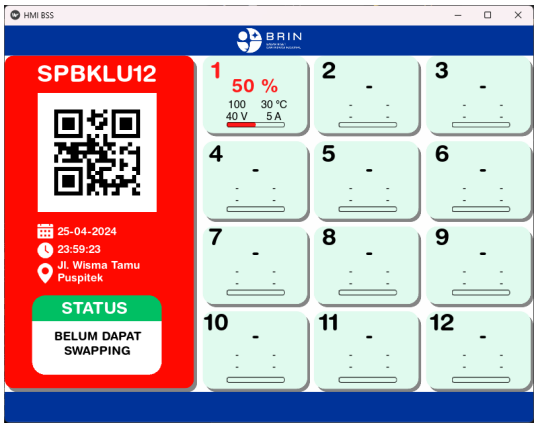


Gambar 4.28: Tampilan text pada bilah status sebelum diubah oleh program utama

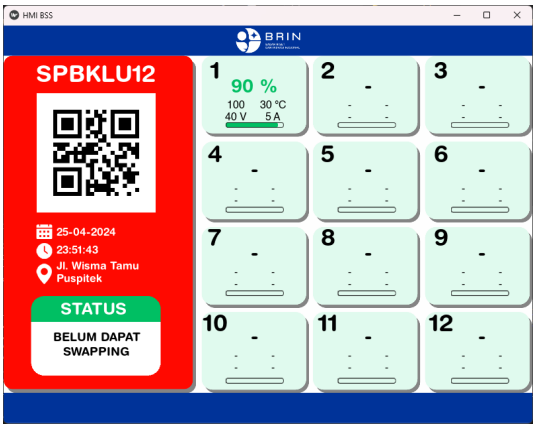
```
Label:
   id: bar1_size
   col:(88/255, 88/255, 88/255, 1)
   pos_hint: {'x': .42, 'y': .743}
   size_hint: (0.0,0.014)
   canvas:
      Color:
         rgba: self.col
      RoundedRectangle:
         size: self.size
         pos: self.pos
         radius: [4]
Image:
   id: outline_bar1
   pos_hint: {'x': .42, 'y': .743}
   size_hint: (0.11, 0.014)
   source:'Assets/Outline_Bar.png'
```

Untuk bagian bar yang menunjukan progres dari SoC baterai dibuat dengan menggunakan *widget "label"* dan *"image"*. Untuk outline dari bar dibuat menggunakan *widget "image"* dengan gambar yang sudah dipersiapkan dengan nama "Outline_Bar.png" pada file "Assets". Selanjutnya progres bar akan diisi menggunakan "*label*" dengan memanfaatkan "*RoundedRectangle*" yang disetting dengan radius 4 serta lebar dan panjangnya telah disesuaikan dengan gambar outline bar. Panjang bar akan sebanding dengan nilai SoC pada baterai. Adapun untuk warna dari bar akan berbeda tergantung dari SoC baterai, apabila SoC baterai di atas atau sama dengan 80 maka bar akan berwarna hijau dan jika SoC di

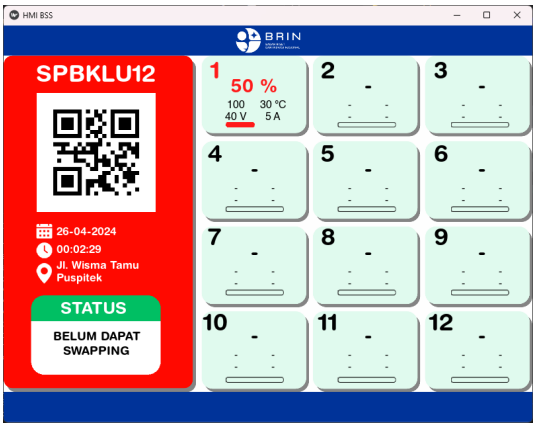
bawah 80 maka bar akan berwarna merah. Perubahan warna dan panjang baterai ditentukan pada program utama.



Gambar 4.29: Tampilan bar pada loker 1 ketika SoC di bawah 80



Gambar 4.30: Tampilan bar loker 1 ketika SoC baterai sama atau di atas 80



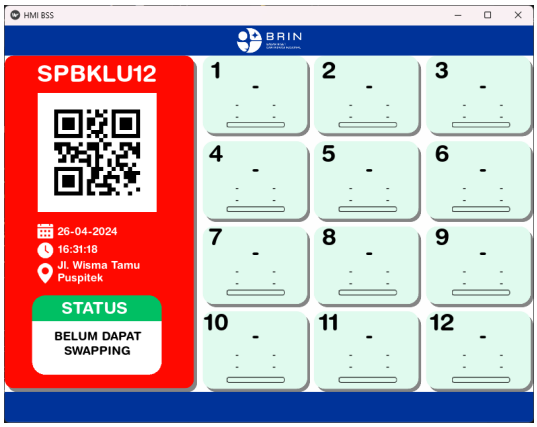
Gambar 4.31: Tampilan bar loker 1 ketika tidak menggunakan outline bar

Python sebagai main program HMI

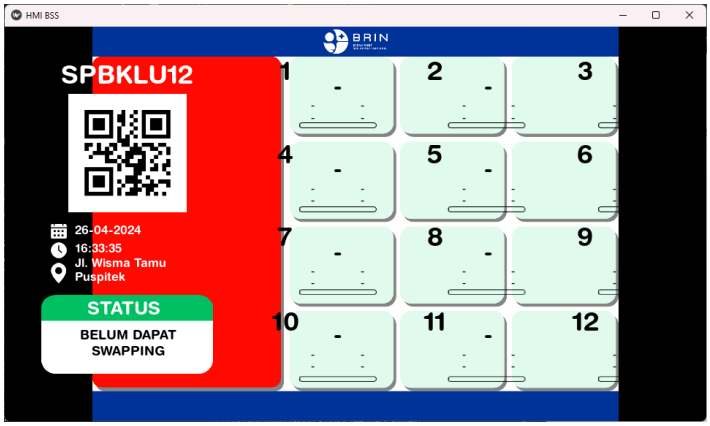
```
import kivy
import datetime
from kivy.app import App
from kivy.core.window import Window
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.uix.image import Image #ubah
from kivy.uix.boxlayout import BoxLayout #ubah
from kivy.uix.label import Label #ubah
from kivy.lang import Builder
from kivy.config import Config
from kivy.clock import Clock
from kivy.graphics import Rectangle, Color
from kivy.config import Config
import paho.mqtt.client as mqtt
import time
import json
import requests #ubah
from custom import printMsg
import urllib
from urllib.request import urlopen
import qrcode
import string #ubah
import random #ubah
import os #ubah
from random import choice

Builder.load_file("HMI.kv")
Window.size = (800, 600)
Window.fullscreen = True
```

Dalam menyusun sistem kontrol utama SPBKLU membutuhkan banyak *library*. Untuk itu, pada bagian awal program dilakukan pemanggilan untuk *library* yang akan digunakan seperti kivy, datetime, json, dan lain-lain. *Library* kivy merupakan *library* yang berfungsi agar program python dapat menggunakan modul kivy.

Untuk memanggil file kv dapat dilakukan dengan menggunakan “Builder.load_file (nama file kv)”. Dikarenakan besar bingkai output dari file kv harus sesuaikan dengan layar ukuran 800X600, besar bingkai perlu disetting ke ukuran yang sama menggunakan “Window.size = (800, 600)” dan “Window.fullscreen = True” agar tampilan ketika program dijalankan *fullscreen.* Namun, “Builder.load_file” hanya digunakan untuk memanggil file kv saja. Untuk menampilkan program diperlukan “ScreenManager” yang merupakan salah satu *method* dari *library* kivy.



Gambar 4.32: Tampilan HMI dengan window size (800, 600)



Gambar 4.33: Tampilan HMI dengan window size (960, 540)

```
sm = ScreenManager

def build(self):
    self.sm.add_widget(Background(name='Background'))
    self.title = 'HMI BSS'
    return self.sm
```

Function build berfungsi untuk menampilkan program kivy dalam satu *screen* menggunakan *method* `ScreenManager()` yang telah dimasukkan dalam variebel sm. Caranya adalah dengan menuliskan `self.sm.add_widget(Background(name='Background'))` kemudian mengembalikan nilainya menggunakan `return`.

```
def __init__(self, **kwargs):
   super(Background, self).__init__(**kwargs)
   self.is_scanned = False

   Clock.schedule_interval(self.clock_rt, 2)
   Clock.schedule_interval(self.actionOfState, 2)
   #func mengirim spbklu status
   Clock.schedule_interval(self.bss_status_send, 180)
   #func mengirim locker status
   Clock.schedule_interval(self.locker_send_status, 120)
   Clock.schedule_interval(self.locker, 1)
   Clock.schedule_interval(self.check_scan_status, 10) # Periksa
status scan setiap 5 detik

   self.unique_code()
```

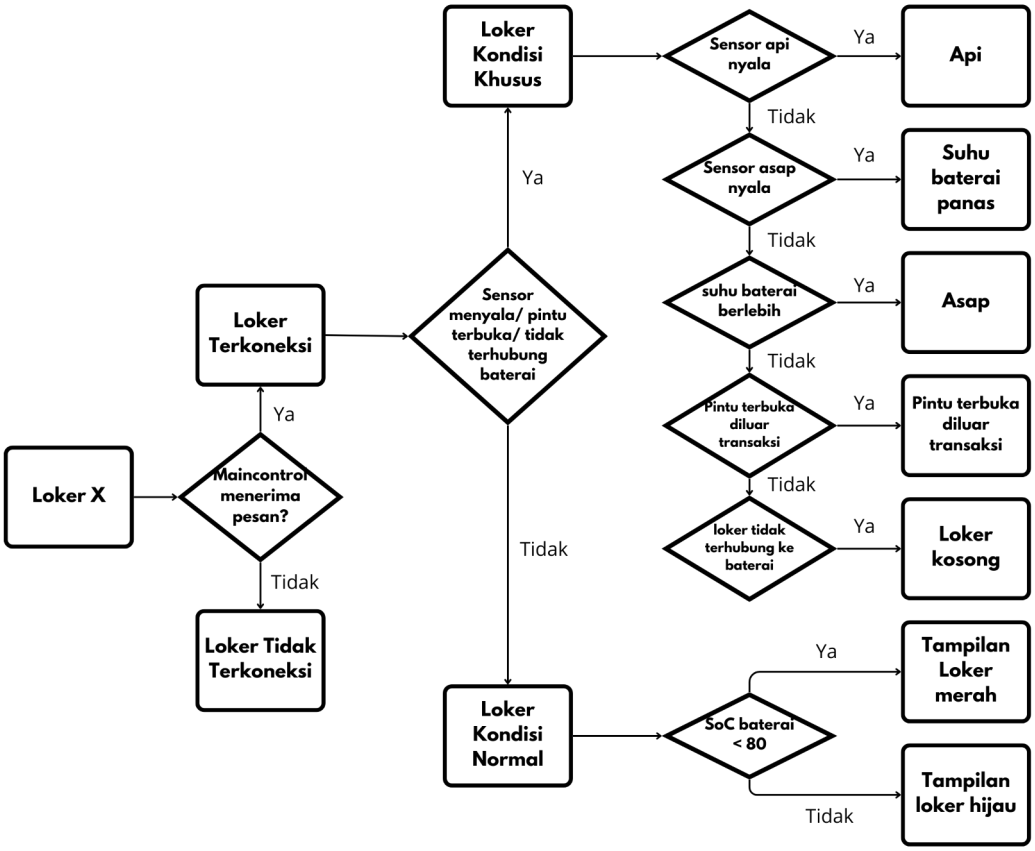
Method __init__ berfungsi untuk menginisialisasi variebel dan menentukan waktu *method* tertentu dijalankan dengan menggunakan modul `Clock.schedule_interval`. *Method* yang dipanggil dengan modul ini akan dijalankan dalam selang waktu tertentu yang telah ditetapkan.

Progam Mengatur Tampilan Loker Pada HMI

```
def locker(self, dt):
  for i in range(MainApp.jumlahLoker):
     MainApp.lockerID = f"locker{i+1}"
     MainApp.mqttTopicUpdate()
     MainApp.timeToDisconnect[i] = MainApp.timeToDisconnect[i] -1
     #Loker terkoneksi
     if MainApp.timeToDisconnect[i] > 0:
        MainApp.lockerConnection[i] = True
        suhuTertinggi = [MainApp.batTemp_cell1[i],
MainApp.batTemp_cell2[i], MainApp.batTemp_cell3[i]]
        #Loker dengan kondisi khusus
        if MainApp.sensor_flame[i] == True or
MainApp.sensor_smoke[i] == True or MainApp.sensor_tstat[i] ==
True or (MainApp.state_plug[i] == False or MainApp.state_door[i]
== False or MainApp.batSOC[i] > 100) :
          self.ids[f"bar{i+1}_size"].size_hint = (0.0008,0.014)
          self.ids[f"batt{i+1}_temp"].text = "-"
          self.ids[f"batt{i+1}_volt"].text = "-"
          self.ids[f"batt{i+1}_ampere"].text = "-"
          self.ids[f"batt{i+1}_SOH"].text = "-"
          self.ids[f"batt{i+1}_SOC"].color = self.ledRed
          #Api terdeteksi
          if MainApp.sensor_flame[i] == True :
            self.ids[f"batt{i+1}_SOC"].text = "Api"
            self.ids[f"batt{i+1}"].col = self.ledOff
          #Asap terdeteksi
          elif MainApp.sensor_smoke[i] == True :
            self.ids[f"batt{i+1}_SOC"].text = "Asap"
```

```
            self.ids[f"batt{i+1}"].col = self.ledOff
          #Termostat mendeteksi kelebihan suhu
          elif MainApp.sensor_tstat[i] == True :
            self.ids[f"batt{i+1}_SOC"].text = "Panas"
            self.ids[f"batt{i+1}"].col = self.ledOff
          #Pintu loker terbuka saat tidak melakukan swapping
          elif MainApp.state_door[i] == False :
            self.ids[f"batt{i+1}_SOC"].text = "Terbuka"
            if(self.ledBlink[1]==0):
               self.ids[f"batt{i+1}"].col = self.ledGreen
               self.ledBlink[1]=1
            elif(self.ledBlink[1]==1):
               self.ids[f"batt{i+1}"].col = self.ledOff
               self.ledBlink[1]=0
          #Loker kosong
          elif MainApp.state_plug[i] == False :
            self.ids[f"batt{i+1}_SOC"].text = "Kosong"
            self.ids[f"batt{i+1}_SOC"].color = self.ledOff
            self.ids[f"batt{i+1}"].col = self.ledGreen
        #Loker kondisi normal
        elif (MainApp.state_plug[i]==True and
MainApp.state_door[i]==True and MainApp.batSOC[i]<=100):
          if MainApp.batSOC[i] >= 0:
            self.ids[f"batt{i+1}_SOC"].text =
f"{MainApp.batSOC[i]} %"
            self.ids[f"batt{i+1}_temp"].text =
f"{max(suhuTertinggi)} °C"
            self.ids[f"batt{i+1}_volt"].text =
f"{MainApp.batVolt[i]} V"
            self.ids[f"batt{i+1}_ampere"].text =
f"{MainApp.batCurr[i]} A"
            self.ids[f"batt{i+1}_SOH"].text =
f"{MainApp.batSOH[i]}"
            self.ids[f"bar{i+1}_size"].size_hint =
((MainApp.batSOC[i])*0.0011,0.014)
            #SOC baterai di bawah 80
            if MainApp.batSOC[i] < 80:
               self.ids[f"bar{i+1}_size"].col = self.ledRed
               self.ids[f"batt{i+1}_SOC"].color = self.ledRed
               if(self.ledBlink[2]==0):
                   self.ids[f"batt{i+1}"].col = self.ledRed
                   self.ledBlink[2]=1
               elif(self.ledBlink[2]==1):
                   self.ids[f"batt{i+1}"].col = self.ledOff
                   self.ledBlink[2]=0
            #SOC baterai di atas 80
            else:
               self.ids[f"bar{i+1}_size"].col = self.ledGreen
               self.ids[f"batt{i+1}_SOC"].color = self.ledGreen
               self.ids[f"batt{i+1}"].col = self.ledOff
     #Loker tidak terkoneksi
     elif MainApp.timeToDisconnect[i] <= 0:
       MainApp.lockerConnection[i] = False
       for attr in ["SOC", "temp", "volt", "ampere", "SOH"]:
         element_name = f"batt{i+1}_{attr}"
         self.ids[element_name].text = "-"
```

```
            self.ids[element_name].color = self.ledOff
            self.ids[f"bar{i+1}_size"].size_hint = (0.0011, 0.014)
```

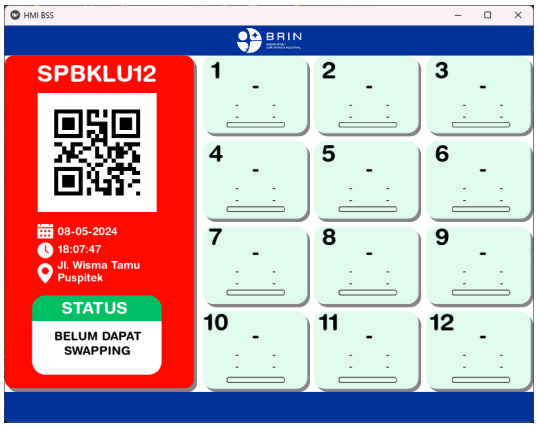


Gambar 4.34: Flowchart method locker untuk identifikasi kondisi loker

Source code di atas merupakan *method* pada program yang berfungsi untuk mengidentifikasi kondisi loker berdasarkan dengan data yang dikirimkan oleh subkontrol pada loker. *Function* ini akan mengidentifikasi setiap loker secara berurutan mulai dari loker 1 hingga 12 menggunakan perulangan. Untuk sistem kerja secara umum dapat dilihat pada gambar di atas.

Pada awalnya loker akan mengidentifikasi apakah loker terkoneksi dengan sistem kontrol utama atau tidak dengan memanfaatkan pesan yang diterima apabila sistem kontrol utama menerima pesan, maka loker akan dinyatakan dalam keadaan terkoneksi dan jika sistem kontrol utama tidak

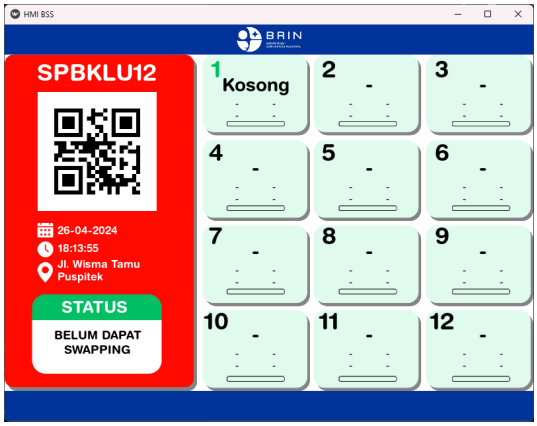
menerima pesan dari loker selama 10 detik, maka loker akan dinyatakan dalam keadaan tidak terkoneksi.



Gambar 4.35 Tampilan loker pada HMI ketika dalam keadaan tidak terkoneksi

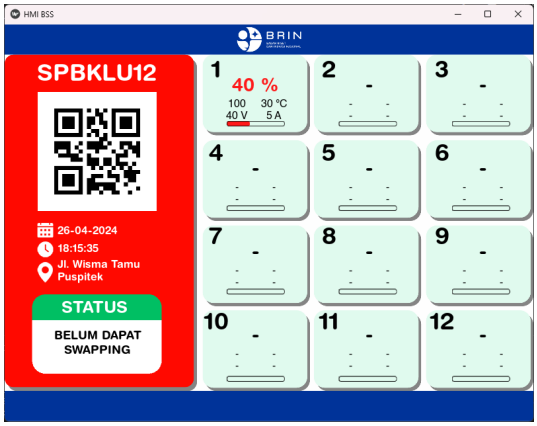
Ketika loker dinyatakan dalam keadaan terkoneksi, maka sistem kontrol utama akan menentukan kondisi loker normal atau khusus. Loker akan dinyatakan dalam kondisi normal ketika terdapat sensor yang menyala, pintu terbuka, atau tidak terhubung dengan baterai dan akan dinyatakan normal jika tidak memenuhi kondisi tersebut.

Jika loker dalam kondisi khusus, maka loker akan diidentifikasi sekali lagi untuk menentukan keadaan khusus yang terjadi pada loker, dimulai dari memeriksa sensor api, sensor asap, suhu baterai, solenoid pintu loker, dan koneksi loker dengan baterai. Kemudian setelah kondisi khusus loker ditentukan, sistem kontrol akan menampilkan kondisi khusus loker tersebut pada HMI.

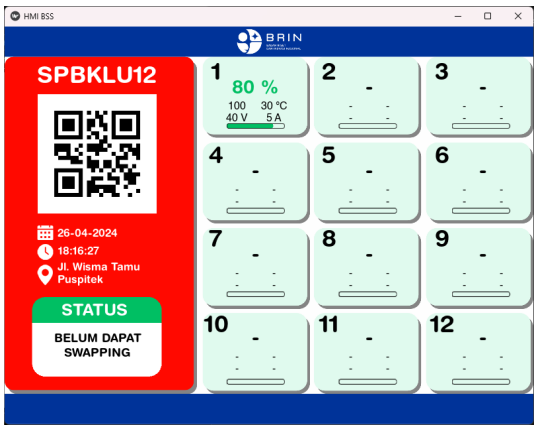


Gambar 4.36: Tampilan loker pada HMI ketika dalam keadaan khusus (kosong)

Jika ternyata loker dalam kondisi normal, maka tahap selanjutnya tinggal menentukan SoC baterai pada loker di bawah 80 atau tidak. Jika iya, maka tampilan SoC dan barnya akan berwarna merah dan apabila diatas atau sama dengan 80, maka tampilan SoC dan barnya akan berwarna hijau.



Gambar 4.37: Tampilan loker pada HMI dalam kondisi normal (SoC<80)



Gambar 4.38: Tampilan loker pada HMI dalam kondisi normal (SoC>=80)

Selain text yang berubah tergantung dari status loker dan data yang dikirimkan oleh subkontrol, angka yang ada pada setiap loker juga akan berubah warna bergantung dari status loker yang melambangkan warna dari LED yang ada pada tiap loker. Untuk lebih jelasnya dapat dilihat pada tabel di bawah.

Tabel 4.1: Warna angka loker pada HMI berdasarkan status loker

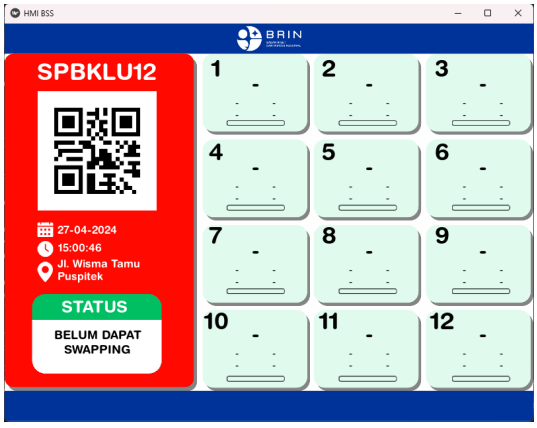
Status Loker	Warna Angka
Loker kosong	Hijau
Loker memiliki baterai dengan SoC di bawah 80	Merah dan hitam bergantian
Loker memiliki baterai dengan SoC di atas 80	Hitam
Pintu loker sedang terbuka	Hijau dan hitam bergantian

Menampilkan Waktu Pada HMI Secara *Realtime*

```
def clock_rt(self, dt):
   now = datetime.datetime.now()
   MainApp.date = now.strftime("%d-%m-%Y")
   hour = now.strftime("%H:%M:%S")
   MainApp.month = now.strftime("%m-%Y")
   self.ids["clock_dt"].text=(f"{ MainApp.date}")
   self.ids["clock_h"].text=(f"{hour}")
```

Pada HMI terdapat tanggal dan waktu yang berubah secara *realtime*. Untuk itu, diprogram *method* `clock_rt` berfungsi untuk mengambil waktu dari

device dengan memanfaatkan modul “datetime”. Variebel `now` berfungsi untuk menyimpan hasil pembacaan tanggal dan waktu saat ini. Kemudian untuk memilah tanggal berupa tahun, bulan, dan hari dilakukan menggunakan `now.strftime("%d-%m-%Y")` yang nilainya dimasukkan kedalam `MainApp.date` dan untuk jam, menit, dan detik dipilah menggunakan `now.strftime("%H:%M:%S")` yang dimasukkan ke dalam variebel `hour`. Selanjutnya untuk menampilkannya pada HMI, nilai tanggal dan waktu yang ada di file kv perlu dirubah dengan `self.ids["clock_dt"].text=(f"{MainApp.date}")` untuk tanggal dan `self.ids["clock_h"].text=(f"{hour}")` untuk waktu. Tanggal dan waktu pada HMI berada tepat di bawah barcode seperti paga gambar di bawah.



Gambar 4.39: Tampilan tanggal dan waktu pada HMI

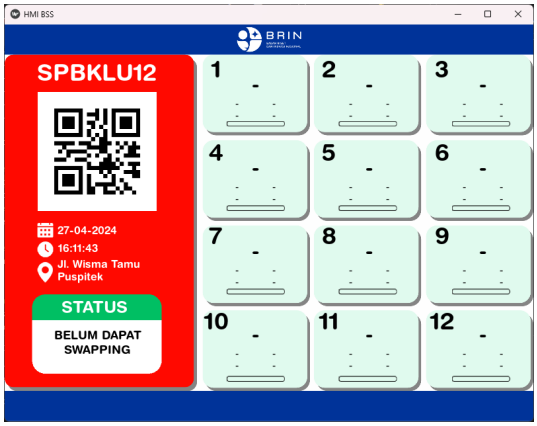
```
#Membuat dan menampilkan barcode secara acak (6 uniq char)
def unique_code(self, dt=0):
  code = ''.join(random.choice(string.ascii_letters +
string.digits) for _ in range(6))
  code = f"{MainApp.spbkluName}_{code}"
```

```
  # Membuat QR Code
  qr = qrcode.QRCode(version=1,
error_correction=qrcode.constants.ERROR_CORRECT_L,
box_size=10, border=4,)
  qr.add_data(code)
  qr.make(fit=True)
  # Membuat image dari QR code
  img = qr.make_image(fill_color="black",
back_color="white")
  filename = f"SPBKLUID_{code}.png"
  filepath =
(f"/home/pi/Documents/spbklu12-hmi/Assets/uniqBarcode/Barco
de{filename}")
  img.save(filepath)
  self.ids["img_barcode"].source =
f'/home/pi/Documents/spbklu12-hmi/Assets/uniqBarcode/Barcod
e{filename}'
  print(filename)
  # waktu penghapusan barcode setelah 1 menit dan mengirim
ke backend
  Clock.schedule_interval(lambda dt:
self.delete_barcode(filepath), 60)
  data = {
    "topic": "spbklu/qrcode",
    "spbkluid": MainApp.spbkluName,
    "qrcode": code
  }
  MainApp.mqttc.publish("spbklu/qrcode", json.dumps(data))

#Men-delete barcode
def delete_barcode(self, filepath):
  try:
    if os.path.exists(filepath):
      os.remove(filepath)
      print(f"File {filepath} dihapus.")
    else:
      print(f"File {filepath} tidak ditemukan.")
  except Exception as e:
    print(f"Error saat menghapus file: {str(e)}")
```

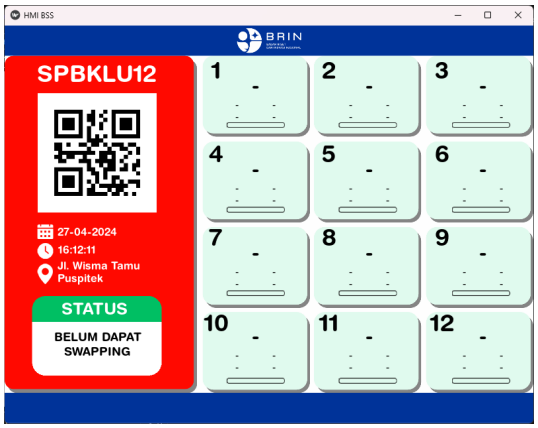
Barcode yang ada pada HMI akan berubah setiap 2 menit untuk mencegah terjadinya kecurangan. Perubahan ini bukan hanya terjadi pada bentuk barcode tapi juga kode unik yang ada di dalamnya. Untuk itu dibutuhkan *method* yang berfungsi untuk membuat barcode yaitu `unique_code` dan *method* yang berfungsi untuk menghapus barcode yang sudah digunakan yaitu

`delete_barcode`. Pertama *method* `unique_code` akan membuat 6 kode unik yang terdiri dari bilangan ASCII dan angka yang dimasukkan dalam variebel `code` menggunakan `random.choice`. Setelah itu program akan menjadikan 6 kode unik tersebut sebagai kode QR menggunakan modul `qrcode.QRCode`. Kode QR menggunakan warna hitam dengan putih sebagai latarnya yang ditentukan menggunakan `qr.make_image`. Kemudian kode QR yang sudah dibuat akan disimpan dalam file uniqBarcode dengan nama "Barcode SPBKLU_(code unik).png". Setelah kode QR disimpan dalam bentuk png barulah kode QR ditampilkan pada HMI. Namun untuk dapat digunakan, kode unik juga perlu dikirimkan ke *back-end* dalam bentuk format JSON. Pengiriman data ini dilakukan menggunakan protokol MQTT dengan topik "spbklu/qrcode".

Adapun untuk penghapusan barcode yang sudah tidak digunakan akan dilakukan menggunakan *method* `delete_barcode` dengan menggunakan modul os. Cara kerjanya adalah ketika dipanggil program akan menghapus barcode yang ditentukan. Jika barcode tersebut ada maka barcode akan dihapus, jika tidak ada maka akan ditampilkan "barcode tidak ditemukan", dan jika ternyata file eror maka akan menampilkan "Error saat menghapus file".



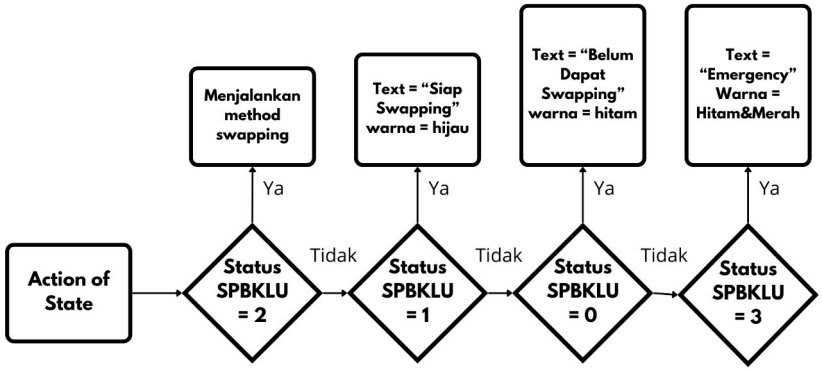
Gambar 4.40: Perubahan barcode pada tampilan HMI (1)



Gambar 4.41: Perubahan barcode pada tampilan HMI (2)

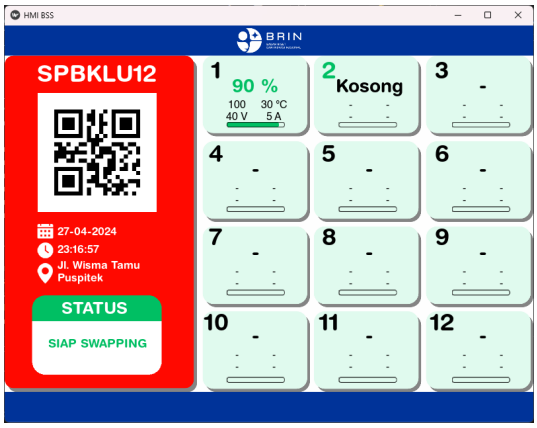
```
def actionOfState(self, dt):
  MainApp.cek_status()
  # Ketika melakukan swapping
  if MainApp.statusSPBKLU == 2 :
     Background.swapping(self)
  # Siap melakukan swapping
  elif MainApp.statusSPBKLU == 1 :
     MainApp.swapping = 0
     self.ids.lbl_keterangan_status.text = f"SIAP SWAPPING"
     self.ids.lbl_keterangan_status.color = self.ledGreen
  # Belum bisa melakukan swapping
  elif MainApp.statusSPBKLU == 0 :
     self.ids.lbl_keterangan_status.text = f"BELUM DAPAT
SWAPPING"
     self.ids.lbl_keterangan_status.color = self.ledOff
  # Kondisi Emergency
  elif MainApp.statusSPBKLU == 3 :
     self.ids.lbl_keterangan_status.text = f"EMERGENCY"
```

```
    if self.ledBlink[0]  == 1:
      self.ids.lbl_keterangan_status.color = self.ledRed
      self.ledBlink[0]  = 0
    elif self.ledBlink[0] == 0:
      self.ids.lbl_keterangan_status.color = self.ledOff
      self.ledBlink[0]  = 1
```

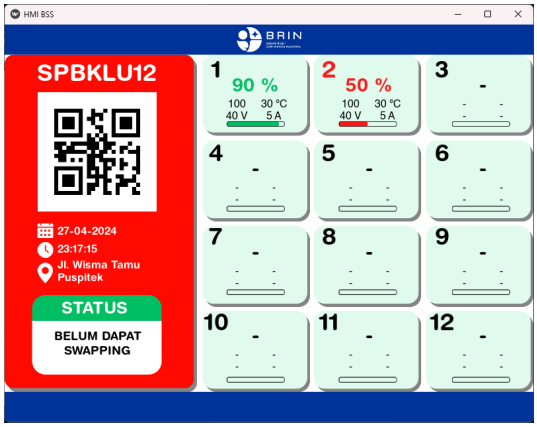


Gambar 4.42: Flowchart method ActionOfState

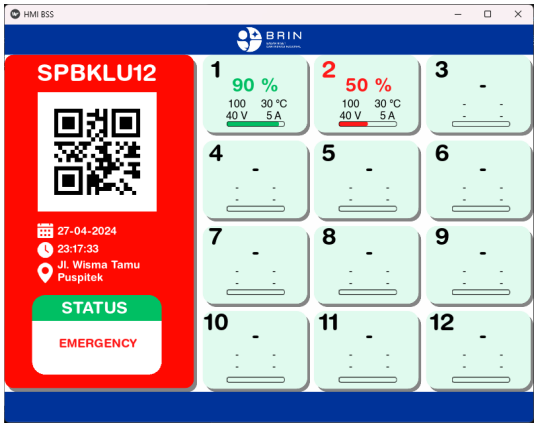
Sebelumnya sudah dibahas mengenai 4 macam status SPBKLU pada BAB 3 dan untuk lebih jelasnya dapat dilihat pada tabel 3.3. *Method* di atas berfungsi untuk menentukan aksi yang dilakukan berdasarkan status SPBKLU saat ini. Aksi yang dilakukan dapat berupa menjalankan *method* tertentu atau mengubah status yang ada pada tampilan HMI. Ketika dipanggil *method* akan memeriksa status SPBKLU, jika status = 2 maka *method* swapping akan dijalankan, jika status = 1 maka status yang ada di HMI akan berisikan text "Siap Swapping" dengan warna hijau, jika status = 0 maka text status yang ditampilkan berisikan "Belum Dapat Swapping" dengan warna hitam, jika status = 3 maka text yang ditampilkan berisikan "Emergency" dengan warna merah dan hitam secara bergantian.



Gambar 4.43: Status SPBKLU = 1



Gambar 4.44: Status SPBKLU = 0



Gambar 4.45: Status SPBKLU = 3

4.3.3 Perancangan Komunikasi Data

Pada subab ini akan dibahas mengenai perancangan komunikasi data pada sistem kontrol utama. Dimulai dengan perancangan komunikasi data

menggunakan protkol MQTT, topik yang digunakan, metode pemilahan data, serta beberapa contoh terkait komunikasi data internal dan komunikasi data eksternal.

***Setting* MQTT Bridge**

Komunikasi data eksternal dengan dilakukan dengan menghubungkan 2 broker, yaitu broker server dan broker lokal. Untuk menghubungkan antara 2 broker dapat dilakukan dengan hanya mengubah konfigurasi salah satu broker. Berikut adalah cara untuk menghubungkan atau melakukan *bridge* antar 2 broker.

Untuk mensetting broker dapat dilakukan dengan mengubah pengaturan file mosquitto.conf. File ini dapat ditemukan pada folder mosquitto. Untuk isi file dari mosquitto.conf setelah di-*setting* adalah sebagai berikut.

```
# Setting broker
pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 1883
allow_anonymous true

# Setting bridge
connection bridge-do
address 202.157.186.31:1883

remote_username microgrid
remote_password m1cr0gr1d

topic # both 0
```

Untuk dapat melakukan bridge antara kedua broker diperlukan address dari broker, username, dan juga password. Setelah ketiga itu diketahui mqtt *bridge* dapat dilakukan. Langkah pertama adalah dengan memberi nama koneksi yang ingin dibuat yaitu “bridge-do” menggunakan “connection”. Setelah itu, address

broker 2 dan portnya dicantumkan dengan menggunakan "`address 202.157.186.31:1883`". Dikarenakan untuk terhubung dengan broker server BRIN membutuhkan username dan password, maka kedua hal tersebut juga perlu disetting dengan "`remote_username microgrid`" untuk username dan "`remote_password m1cr0gr1d`" untuk password. Hal terkahir adalah mensetting topiyang ingin di *bridge* menggunakan "`topic # both 0`" yang berarti bahwa semua topik yang terkirim akan dihubungkan.

Method mqttConnect: *Setting* Komunikasi MQTT

```
def mqttConnect(self):
  MainApp.mqttTopicUpdate()
  MainApp.logSystem("MQTT",f"Connecting...")
  MainApp.mqttc.on_connect = MainApp.onConnect
  MainApp.mqttc.on_message = MainApp.onMessage
  MainApp.mqttc.on_disconnect = MainApp.onDisconnect
  if(MainApp.mqttUser!="" and MainApp.mqttPassword!=""):
    MainApp.mqttc.username_pw_set(MainApp.mqttUser,
password=MainApp.mqttPassword)
  MainApp.mqttc.connect(MainApp.mqttBroker,
MainApp.mqttPort, keepalive=60, bind_address="")
  MainApp.mqttc.loop_start()
```

Perancangan akan diawali dengan perancangan komunikasi MQTT dengan menggunakan *method* mqttConnect. Pada tahap awal dilakukan pemanggilan *method* mqttTopicUpdate yang berisikan topik mqtt yang akan digunakan. Kemudian program akan mencoba membuat koneksi dengan menggunakan `MainApp.mqttc.connect(MainApp.mqttBroker, MainApp.mqttPort, keepalive=60, bind_address="")`. *Method* ini akan terus dipanggil hingga Raspberry Pi berhasil terhubung ke broker. Setelah terkoneksi *method* `MainApp.onConnect` akan dipanggil dan apabila perangkat menerima pesan maka *method* `MainApp.onMessage` akan dipanggil. Adapun

untuk membuat koneksi yang lebih aman ditambahkan *pasword* dengan menggunakan `MainApp.mqttc.username_pw_set(MainApp.mqttUser, password=MainApp.mqttPassword)`. Untuk penjelasan lebih lanjut terkait dengan *method* yang digunakan akan dijelaskan dalam paragraf di bawah.

Method mqttTopicUpdate: Menyimpan Topik yang Akan Digunakan

```
def mqttTopicUpdate():
  MainApp.topicData       =
f"{MainApp.spbkluName}/uc/{MainApp.lockerID}/dataJson"
  MainApp.topicProcess    =
f"{MainApp.spbkluName}/uc/{MainApp.lockerID}/process"
  MainApp.topicSolenoid   =
f"{MainApp.spbkluName}/pi/{MainApp.lockerID}/solenoid"
  MainApp.topicAlarm      =
f"{MainApp.spbkluName}/pi/{MainApp.lockerID}/alarm"
  MainApp.topicToUser     =
f"{MainApp.spbkluName}/pi/{MainApp.userID}"
  MainApp.topicFromUser   = f"{MainApp.spbkluName}/user/#"
  MainApp.topicSubscribe  = f"{MainApp.spbkluName}/#"
```

Method mqttTopicUpdate berisikan topik MQTT yang digunakan dalam komunikasi data sistem kontrol utama. `MainApp.topicData` digunakan untuk menerima data dari ESP32 yang ada pada setiap loker terkait dengan data loker, `MainApp.topicProcess` digunakan untuk menerima respon dari ESP32 ketika diberi perintah membuka pintu loker, `MainApp.topicSolenoid` digunakan untuk mengirimkan perintah ke ESP32 pada loker tertentu untuk membuka pintu loker, `MainApp.topicAlarm` digunakan untuk menyalakan ataupun mematikan alarm, `MainApp.topicFromUser` digunakan untuk komunikasi dengan pengguna, `MainApp.topicSubscribe` digunakan untuk melakukan *subscribe* terkait dengan segala topik yang berkaitan.

Method onConnect: Dipanggil Ketika Sistem Berhasil Terkoneksi ke Broker

```
def onConnect(client, userdata, flags, rc):
  MainApp.mqttc.subscribe(MainApp.topicSubscribe, 1)
  MainApp.logSystem("MQTT", f"Connected to
{MainApp.mqttBroker} rc={rc}")
  MainApp.logSystem("MQTT", f"Subscribed to ->
{MainApp.topicSubscribe}")
```

Method `onConnect` berisikan langkah yang akan dilakukan ketika sistem kontrol utama sudah terkoneksi dengan broker. Ketika dipanggil maka sistem kontrol utama akan melakukan *subscribe* dengan topik "SPBKLU10/#", lambang "#" menandakan bahwa sistem kontrol utama akan menerima semua pesan yang memiliki topik "SPBKLU10". Adapun untuk logSystem merupakan *method* yang dipanggil untuk mencatat data pada log.

```
def onDisconnect(client, userdata, flag):
  MainApp.logSystem("MQTT", f"Disconnected from
{MainApp.mqttBroker}")
```

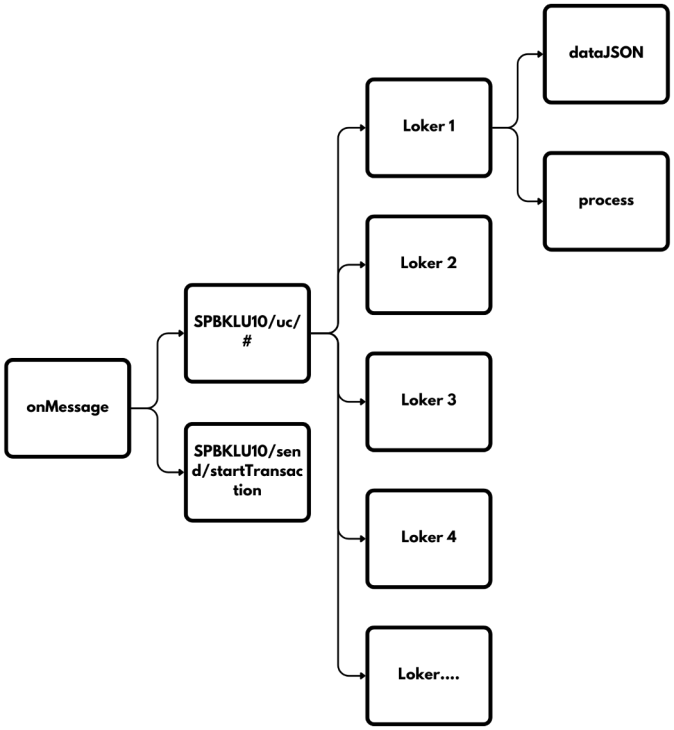
Method `onDisconnect` akan dipanggil ketika koneksi MQTT sistem kontrol utama terputus dari broker. *Method* ini digunakan sebagai penanda sekaligus mencatat ketika sistem kontrol utama mengalami *disconnecting.*

Method onMessage: Pemilahan Data Berdasarkan Topik Pesan

```
def onMessage(self, userdata, message):
  MainApp.data = message.payload.decode("utf-8")

  # pengecekan topic
  #Memulai transaksi
  if message.topic ==
f"{MainApp.spbkluName}/send/startTransaction":
    MainApp.jsondata = json.loads(MainApp.data)
    if MainApp.jsondata["status"] == 200:
      print("Start Transaction Berhasil")
      MainApp.userID = MainApp.jsondata['data']["userID"]
      MainApp.username = MainApp.jsondata['data']['user_name']
      MainApp.transactionID = MainApp.jsondata['data']
['transaksiID']
      MainApp.mqttTopicUpdate()
```

```
       MainApp.logSystem("DATABASE", MainApp.data.split("{")
[1].split("}")[0])
         if MainApp.statusSPBKLU == 1: #kondisi saat 1 loker
kosong dan berganti ke kondisi kedua yaitu proses swapping
           MainApp.statusSPBKLU = 2
           dataToUser = {
           "SPBKLU": MainApp.spbkluName,
           "userID": MainApp.userID,
           "username": MainApp.username,
           "status_SPBKLU": MainApp.statusSPBKLU,
           }
           MainApp.mqttc.publish(MainApp.topicToUser,
json.dumps(dataToUser))
  #Data dari Loker
  elif message.topic.split('/')[1] ==
MainApp.topicData.split('/')[1] :
    MainApp.lockerNumber = int(message.topic.split('/')
[2].split('locker')[1])
    content = message.topic.split('/')[3]
    #Data baterai dan sensor loker
    if content == MainApp.topicData.split('/')[3] :
      MainApp.jsondata = json.loads(MainApp.data)
      MainApp.timeToDisconnect[MainApp.lockerNumber-1]=10
      MainApp.state_door[MainApp.lockerNumber-
1]=MainApp.jsondata['state_door']
      MainApp.state_plug[MainApp.lockerNumber-
1]=MainApp.jsondata['state_plug']
      MainApp.state_locker[MainApp.lockerNumber-
1]=MainApp.jsondata['state_locker']
      MainApp.state_charging[MainApp.lockerNumber-
1]=MainApp.jsondata['state_charging']
      MainApp.state_emg[MainApp.lockerNumber-
1]=MainApp.jsondata['state_emg']
      MainApp.batSOC[MainApp.lockerNumber-
1]=MainApp.jsondata['batSOC']
      MainApp.batSOH[MainApp.lockerNumber-
1]=MainApp.jsondata['batSOH']
      MainApp.batTemp_cell1[MainApp.lockerNumber-
1]=MainApp.jsondata['batTemp_cell1']
      MainApp.batTemp_cell2[MainApp.lockerNumber-
1]=MainApp.jsondata['batTemp_cell2']
      MainApp.batTemp_cell3[MainApp.lockerNumber-
1]=MainApp.jsondata['batTemp_cell3']
      MainApp.batCycle[MainApp.lockerNumber-
1]=MainApp.jsondata['batCycle']
      MainApp.batID[MainApp.lockerNumber-
1]=MainApp.jsondata['batID']
      MainApp.batVolt[MainApp.lockerNumber-
1]=MainApp.jsondata['batVolt']
      MainApp.batCurr[MainApp.lockerNumber-
1]=MainApp.jsondata['batCurr']
      MainApp.sensor_flame[MainApp.lockerNumber-
1]=MainApp.jsondata['sensor_flame']
      MainApp.sensor_smoke[MainApp.lockerNumber-
1]=MainApp.jsondata['sensor_smoke']
      MainApp.sensor_tstat[MainApp.lockerNumber-
```

```
1]=MainApp.jsondata['sensor_tstat']
       MainApp.ctrl_chg[MainApp.lockerNumber-
1]=MainApp.jsondata['ctrl_chg']
       MainApp.ctrl_buzzer[MainApp.lockerNumber-
1]=MainApp.jsondata['ctrl_buzzer']
       MainApp.ctrl_fan[MainApp.lockerNumber-
1]=MainApp.jsondata['ctrl_fan']
       MainApp.ctrl_aerosol[MainApp.lockerNumber-
1]=MainApp.jsondata['ctrl_aerosol']
       MainApp.data_now[MainApp.lockerNumber-1]=MainApp.data
       if MainApp.data_now[MainApp.lockerNumber-1] !=
MainApp.data_prev[MainApp.lockerNumber-1] :
         if MainApp.data_prev[MainApp.lockerNumber-1] == 0 :
           MainApp.data_prev[MainApp.lockerNumber-1] =
MainApp.data_now[MainApp.lockerNumber-1]
           MainApp.logSystem(f"LOCKER
{MainApp.lockerNumber}",MainApp.data_now[MainApp.lockerNumber-
1].split("{")[1].split("}")[0])
         else :
           MainApp.data_now_list =
list(MainApp.data_now[MainApp.lockerNumber-1].split("{")
[1].split("}")[0].split(","))
           MainApp.data_prev_list =
list(MainApp.data_prev[MainApp.lockerNumber-1].split("{")
[1].split("}")[0].split(","))
           for i in range (len(MainApp.data_now_list)) :
             if MainApp.data_now_list[i] !=
MainApp.data_prev_list[i] :

  MainApp.data_update.append(MainApp.data_now_list[i])
           for i in (MainApp.data_update):
             MainApp.data_fix += f"{i},"
           MainApp.data_fix = MainApp.data_fix.strip()
           MainApp.logSystem(f"LOCKER
{MainApp.lockerNumber}",MainApp.data_fix[:-1])
           MainApp.data_prev[MainApp.lockerNumber-1] =
MainApp.data_now[MainApp.lockerNumber-1]
           MainApp.data_fix = ""
           MainApp.data_update.clear()
    #Data feedback locker ketika diberi perintah oleh Rasberry
PI
    elif content == MainApp.topicProcess.split('/')[3] :
      MainApp.dataProcess = MainApp.data
      MainApp.logSystem("MQTT",MainApp.dataProcess)
```



Gambar 4.46: Flowchart pemilahan data untuk method onMessage

Method onMessage akan dipanggil setiap sistem kontrol utama mendapatkan pesan MQTT. Pesan yang diterima memiliki data dan juga topik, data pesan akan disimpan dalam variebel `MainApp.data` dan topik akan digunakan untuk memilah pesan. Berdasarkan topiknya pesan yang diterima akan dibagi menjadi 2, yaitu pesan dari subkontrol pada loker (uc) dan juga pesan untuk memulai transaksi *battery swap*. Pesan dari subkontrol akan dikelompokan berdasarkan lokernya. Selanjutnya untuk setiap loker pesan akan dikelompokan menjadi dataJSON yang berisikan data loker dan Process yang berisikan *feedback* dari loker ketika diperintah untuk membuka loker.

Pesan untuk memulai transaksi berisikan *userID, username, tranctionID,* dan status dari aplikasi pengguna. Jika status aplikasi pegguna 200, maka sistem kontrol utama akan memulai proses transaksi *battery swap*.

Pesan dataJSON berisikan data loker dengan total 23 data yang digabungkan dalam 1 variebel JSON. Untuk itu setelah pesan diterima data tersebut akan dimasukkan ke dalam variebel dalam bentuk list. Terdapat 23 buah variebel list untuk menampung data JSON dari subkontrol dengan panjang index 12 untuk setiap listnya yang berfungsi untuk menampung data dengan jenis yang sama untuk setiap loker. Misalkan data variebel `MainApp.state_door[MainApp.lockerNumber-1]` merupakan variebel yang berfungsi untuk menyimpan data status loker untuk loker 1 hingga loker 12.

Adapun untuk process berisikan pesan berupa *feedback* dari subkontrol ketika diberikan suatu perintah oleh sistem kontrol utama. Misal sistem kontrol utama memerintahkan untuk membuka pintu loker, ketika pintu loker sudah terbuka subkontrol akan mengirimkan pesan "{solenoidState=True}" sebagai balasan. Dengan begitu sistem kontrol utama dapat mengetahui bahwa perintahnya telah dilaksanakan.

4.3.4 Perancangan *Battery Swapping*

Subab ini akan membahas mengenai perancangan *battery swapping* mulai dari menentukan status SPBKLU, memulai *battery swapping,* hingga tahapan dari *battery swapping* itu sendiri.Untuk lebih jelasnya akan dibahas dalam paragraf selanjutnya.

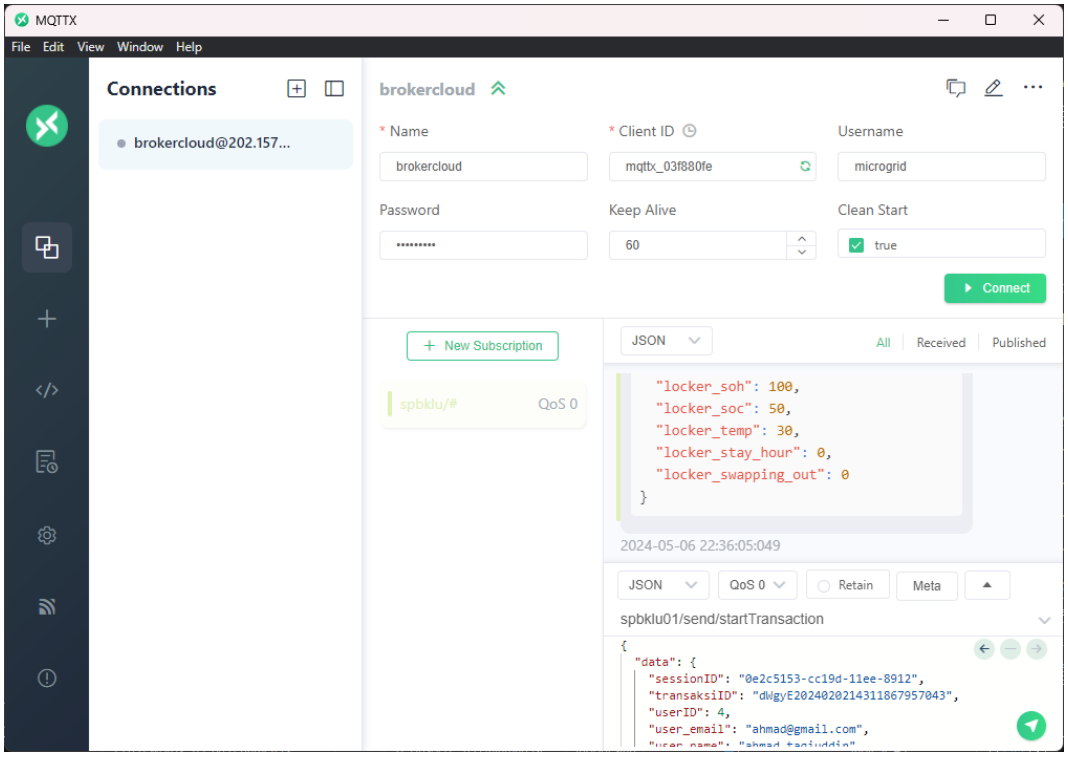
Permintaan Untuk Melakukan Transaksi *Battery Swapping*

```
if message.topic ==
f"{MainApp.spbkluName}/send/startTransaction":
  MainApp.jsondata = json.loads(MainApp.data)
  if MainApp.jsondata["status"] == 200:
    print("Start Transaction Berhasil")
    MainApp.userID = MainApp.jsondata['data']["userID"]
    MainApp.username = MainApp.jsondata['data']
['user_name']
    MainApp.transactionID = MainApp.jsondata['data']
['transaksiID']
    MainApp.mqttTopicUpdate()
    MainApp.logSystem("DATABASE", MainApp.data.split("{")
[1].split("}")[0])
    if MainApp.statusSPBKLU == 1: #kondisi saat 1 loker
kosong dan berganti ke kondisi kedua yaitu proses swapping
      MainApp.statusSPBKLU = 2
    dataToUser = {
      "SPBKLU": MainApp.spbkluName,
      "userID": MainApp.userID,
      "username": MainApp.username,
      "status_SPBKLU": MainApp.statusSPBKLU,
      }
    MainApp.mqttc.publish(MainApp.topicToUser,
json.dumps(dataToUser))
```

Transaksi *battery swapping* akan dimulai ketika sistem kontrol utama menerima pesan JSON dengan topik “SPBKLU01/send/startTransaction”. Ketika pesan yang diterima sesuai, maka sistem kontrol utama akan memeriksa apakah status SPBKLU bernilai 1. Jika benar, maka status akan diubah menjadi 2 yang menandakan bahwa SPBKLU memasuki mode *battery swapping*. Adapun isi dari pesan untuk permintaan melakukan transaksi adalah sebagai berikut.

```
#Pesan Permintaan Melakukan Transaksi
{
  "data": {
    "sessionID": "0e2c5153-cc19d-11ee-8912",
    "transaksiID": "dWgyE2024020214311867957043",
    "userID": 4,
```

```
    "user_email": "ahmad@gmail.com",
    "user_name": "ahmad taqiuddin"
  },
  "message": "Start Transaction Berhasil",
  "status": 200
}
```

Gambar 4.47: Tampilan MQTTX yang digunakan untuk mengirim pesan permintaan Swapping

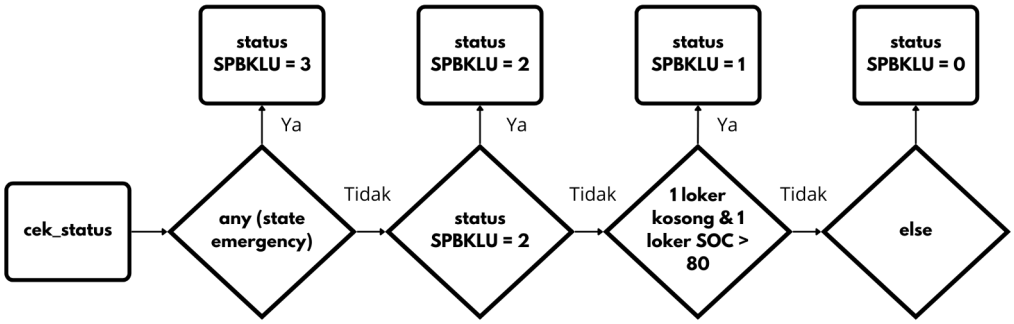
Untuk mengirimkan pesan permintaan transaksi, maka device perlu dihubungkan dengan broker server yang digunakan SPBKLU. Untuk menghubungkan dengan broker server dibutuhkan *address* broker, password, dan username. Setelah MQTTX terhubung dengan broker server, selanjutnya adalah mengirimkan pesan MQTT dalam bentuk JSON dengan format tertentu menggunakan topik “SPBKLU01/send/startTransaction”.

Pesan yang dikirim untuk mentriger mulainya transaksi *battery swapping* berisikan data ID session, ID transaksi, ID pengguna, email pengguna, nama

pengguna, pesan, serta status permintaan transaksi. Nilai 200 merupakan nilai yang akan mentriger sistem kontrol utama untuk memulai transaksi, nilai status permintaan transaksi akan dikirimkan oleh backend.

Method cek_status: Menentukan Status SPBKLU

```
#Menentukan status SPBKLU
def cek_status():
  if any(MainApp.state_emg) :
    MainApp.statusSPBKLU = 3
  else:
    if MainApp.statusSPBKLU == 2 :
      MainApp.statusSPBKLU == 2
    elif any(x == 1 for x in MainApp.state_locker) and
any(x == 3 for x in MainApp.state_locker):
      MainApp.statusSPBKLU = 1
    else:
      MainApp.statusSPBKLU = 0
```



Gambar 4.48: Flowchart method cek_status untuk menentukan status SPBKLU

Transaksi *battery swapping* hanya dapat dimulai ketika terdapat 1 loker kosong dan 1 loker dengan baterai SOC di atas 80 atau status SPBKLU bernilai 1. Sebelumnya sudah dibahas mengenai macam status SPBKLU, *method* di atas berfungsi untuk menentukan status SPBKLU berdasarkan status dari 12 loker yang ada pada SPBKLU. Sistem kontrol utama akan memeriksa data terkait dengan status emergency yang dikirimkan subkontrol, jika terdapat loker dengan

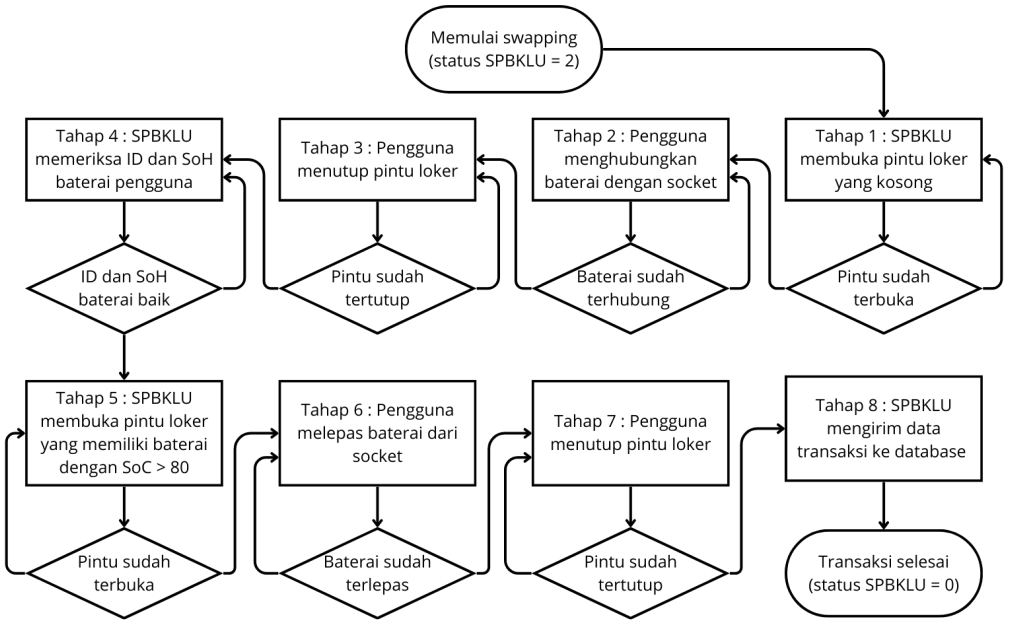
status emergency bernilai true maka status SPBKLU akan bernilai 3, apabila tidak maka SPBKLU akan memeriksa apakah status SPBKLU saat ini bernilai 2, jika benar maka nilai status SPBKLU akan dipertahankan hingga transaksi selesai dan apabila tidak maka sistem kontrol utama akan memeriksa apakah terdepat 1 loker kosong dan 1 loker dengan baterai SOC lebih dari 80, jika benar maka status SPBKLU akan bernilai 1 dan jika tidak status SPBKLU akan bernilai 0.

Tahapan Pada Proses *Battery Swapping*

```
#Mencari loker kosong
if MainApp.swapping == 0 :
  MainApp.swapping = 1
  for i in range(MainApp.jumlahLoker):
     if(MainApp.state_locker[i] == 3):
       MainApp.selectLockerSwap = i
  MainApp.lockerID = f"locker{MainApp.selectLockerSwap+1}"
  MainApp.mqttTopicUpdate()
  MainApp.mqttc.publish(MainApp.topicSolenoid, "true")
  self.ids.lbl_keterangan_status.text = f"LOKER
{MainApp.selectLockerSwap + 1} TERBUKA"
  MainApp.collectData = True

#SPBKLU membuka pintu loker kosong
elif MainApp.swapping == 1 :
  if MainApp.state_door[MainApp.selectLockerSwap] ==
False :
    MainApp.swapping = 2

#Pengguna menghubungkan baterai ke socket
elif MainApp.swapping == 2 :
  self.ids.lbl_keterangan_status.text =
f"{MainApp.timeLockerOpen}\nPASANG BATERAI"
  if MainApp.state_plug[MainApp.selectLockerSwap] == True :
    MainApp.swapping = 3

#Pengguna menutup pintu loker
elif MainApp.swapping == 3 :
  self.ids.lbl_keterangan_status.text =
f"{MainApp.timeLockerOpen}\nTUTUP LOKER
{MainApp.selectLockerSwap+1}"
  if (MainApp.state_door[MainApp.selectLockerSwap] == True
and MainApp.state_plug[MainApp.selectLockerSwap] == True):
```

```
    MainApp.swapping = 4

#SPBKLU memeriksa kondisi baterai (SOH dan ID baterai)
elif MainApp.swapping == 4 :
  self.ids.lbl_keterangan_status.text = "MEMERIKSA BATERAI
MASUK"
  #SOH baterai baik dan ID baterai terdaftar
  if MainApp.batSOH[MainApp.selectLockerSwap] >=
MainApp.batSOH_Ok and any(x ==
MainApp.batID[MainApp.selectLockerSwap] for x in
MainApp.batteryID) :
    maxBatt = 0
    for i in range(MainApp.jumlahLoker):
      if(MainApp.batSOC[i] > maxBatt):
        maxBatt = MainApp.batSOC[i]
        MainApp.selectLockerSwap = i
    MainApp.swapping = 5
    MainApp.lockerID =
f"locker{MainApp.selectLockerSwap+1}"
    MainApp.mqttTopicUpdate()
    MainApp.mqttc.publish(MainApp.topicSolenoid, "true")
  #SOH baterai rendah atau ID baterai tidak terdaftar
  else :
    MainApp.swapping = 15
    MainApp.lockerID =
f"locker{MainApp.selectLockerSwap+1}"
    MainApp.mqttTopicUpdate()
    MainApp.mqttc.publish(MainApp.topicSolenoid, "true")

#SPBKLU membuka pintu loker yang memiliki baterai dengan
SoC tertinggi
elif MainApp.swapping == 5 :
  self.ids.lbl_keterangan_status.text = f"LOKER
{MainApp.selectLockerSwap + 1} TERBUKA"
  if MainApp.state_door[MainApp.selectLockerSwap] ==
False :
    MainApp.swapping = 6

#Pengguna melepas baterai dengan SoC tinggi dari socket
elif MainApp.swapping == 6 :
  self.ids.lbl_keterangan_status.text =
f"{MainApp.timeLockerOpen}\nLEPAS BATERAI DARI LOKER
{MainApp.selectLockerSwap+1}" #ubah
  if MainApp.state_plug[MainApp.selectLockerSwap] ==
False :
    MainApp.swapping = 7

#Pengguna menutup pintu loker
elif MainApp.swapping == 7 :
  self.ids.lbl_keterangan_status.text =
f"{MainApp.timeLockerOpen}\nTUTUP LOKER
```

```
{MainApp.selectLockerSwap + 1}"
  if (MainApp.state_door[MainApp.selectLockerSwap] == True
and MainApp.state_plug[MainApp.selectLockerSwap] == False):
  MainApp.swapping = 8

#SPBKLU mengirim data transaksi ke database dan transaksi
selesai
elif MainApp.swapping == 8 :
   self.ids.lbl_keterangan_status.text = "MEMUAT"
   MainApp.mqttc.publish(MainApp.topicAlarm, "false")
   Background.stop_transaction()
   MainApp.statusSPBKLU = 0
```



Gambar 4.49: Flowchart tahapan transaksi battery swapping

Transaksi *battery swapping* memiliki 8 tahap, dimulai dari SPBKLU membuka pintu loker, pengguna menghubungkan baterainya dengan socket, pengguna menutup pintu loker, SPBKLU memeriksa ID baterai dan SoH baterai pengguna, SPBKLU membuka pintu loker yang memiliki baterai dengan SoC tertinggi, pengguna melepas baterai dari socket, pengguna menutup pintu loker, SPBKLU mengirimkan data transaksi ke database, dan transaksi selesai. HMI akan membantu pengguna dalam melakukan transaksi *battery swapping* dengan menampilkan perintah sesuai dengan tahapan *battery swapping*.

Tahap 1, sistem kontrol utama akan memeriksa loker yang tidak memiliki baterai di dalamnya kemudian memberi pesan perintah kepada subkontrol loker terkait untuk membuka pintu lokernya dan HMI akan menampilkan pesan "PINTU LOKER X TERBUKA". Tahap ini akan berakhir ketika sistem kontrol utama menerima data dari subkontrol bahwa pintu loker kosong sudah terbuka.

Tahap 2, sistem kontrol utama akan meminta pengguna untuk menghubungkan baterainya ke socket yang tersedia dengan menampilkan pesan "PASANG BATERAI" pada HMI. Tahap ini akan berakhir ketika sistem kontrol mendeteksi bahwa baterai pengguna sudah terhubung dengan socket.

Tahap 3, sistem kontrol utama akan meminta pengguna untuk menutup pintu loker yang menampung baterainya dengan menampilkan pesan "TUTUP LOKER" pada HMI. Tahap ini akan selesai ketika sistem kontrol utama telah mendeteksi bahwa pintu loker sudah tertutup dan baterai pengguna telah terpasang.

Tahap 4, sistem kontrol utama akan memeriksa ID dan SoH baterai pengguna dan HMI akan menampilkan pesan "MEMERIKSA BATERAI". Jika ID baterai terdaftar dan SoH baterai lebih dari 60, maka pengguna dapat melanjutkan transaksi ke tahap selanjutnya. Namun, jika ID baterai tidak terdaftar atau SoH baterai di bawah 60, maka sistem *eror handling* akan berjalan dan transaksi gagal.

Tahap 5, sistem kontrol utama akan memeriksa data loker untuk menemukan loker yang memiliki baterai dengan SoC tertinggi, setelah itu sistem kontrol akan memberi pesan perintah kepada subkontrol loker terkait untuk

membuka pintu loker dan HMI akan menampilkan pesan “PINTU LOKER X TERBUKA”. Tahap ini akan berakhir ketika sistem kontrol utama mendeteksi bahwa pintu loker terkait sudah terbuka.

Tahap 6, sistem kontrol utama akan meminta pengguna untuk melepas baterai dari socket dengan menampilkan pesan “LEPAS BATERAI” pada HMI. Tahap ini akan berakhir ketika sistem kontrol utama mengidentifikasi bahwa baterai dengan SoC tinggi sudah terlepas dari socket.

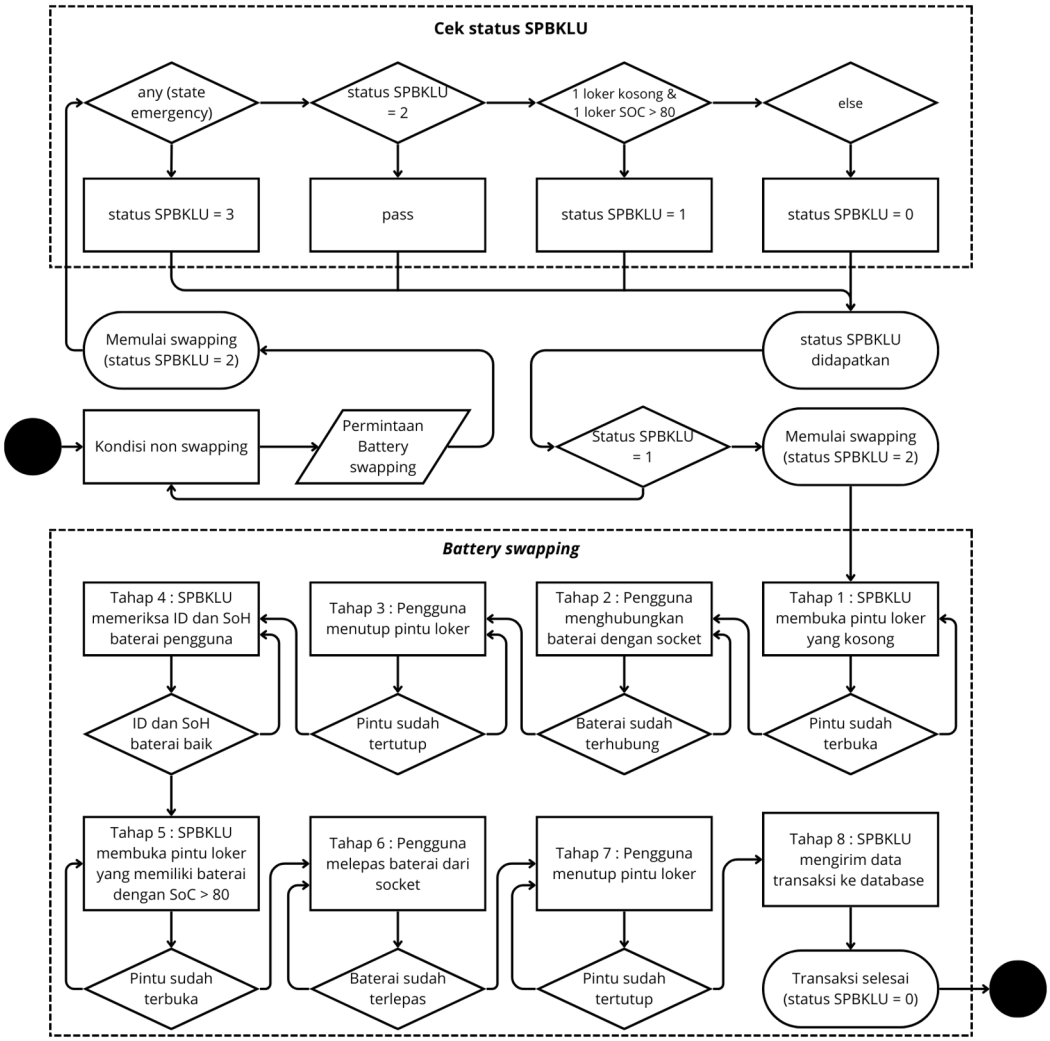
Tahap 7, sistem kontrol utama akan meminta pengguna untuk menutup loker dengan menampilkan pesan berupa “TUTUP LOKER” pada HMI. Tahap ini akan berakhir ketika sistem kontrol utama mengidentifikasi bahwa pintu loker terkait sudah tertutup.

Tahap 8, sistem kontrol utama akan mengirimkan data transaksi *battery swapping* ke database menggunakan komunikasi MQTT dan transaksi *battery swapping* selesai.

Sistem Transaksi *Battery Swapping* Secara Menyeluruh dan Uji Coba Transaksi Menggunakan Program Locker Simulator

Berdasarkan dengan apa yang sudah dibahas sebelumnya, sistem transaksi *battery swapping* dimulai dari sistem kontrol utama menerima permintaan untuk memulai transaksi, kemudian sistem kontrol utama akan memeriksa status SPBKLU menggunakan method cek_status, jika status SPBKLU bernilai 1 yang menandakan memiliki 1 loker kosong dan 1 loker dengan baterai yang memiliki SoC di atas 80 maka transaksi *battery swapping*

akan dimulai dan memasuki proses *battery swapping* yang terdiri dari 8 tahap. Berikut adalah *flowchart* untuk sistem transaksi *battery swapping.*

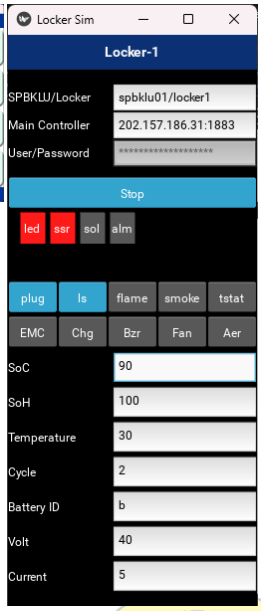
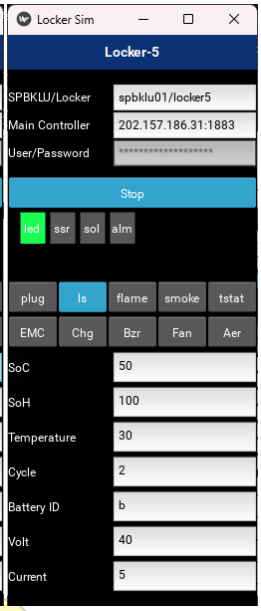
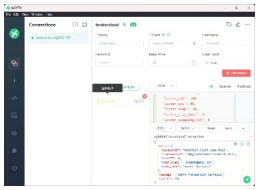


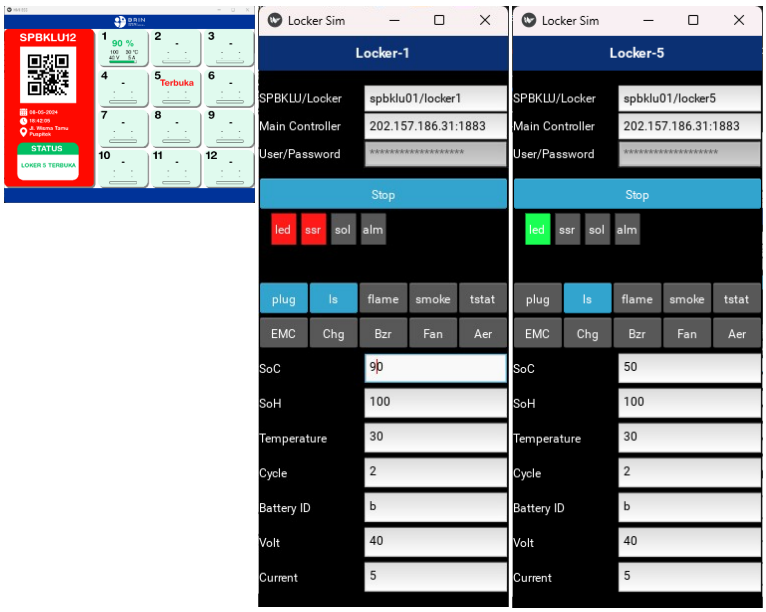
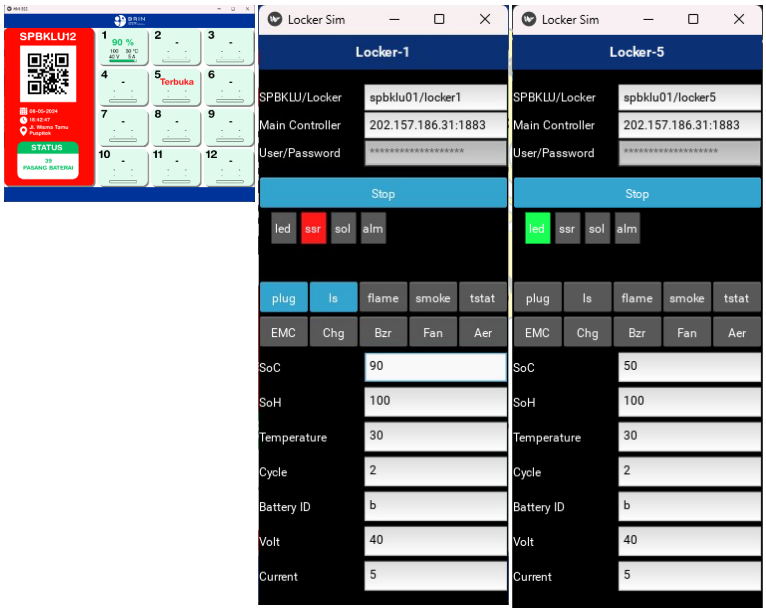
Gambar 4.50: Flowchart transaksi battery swapping secara keseluruhan

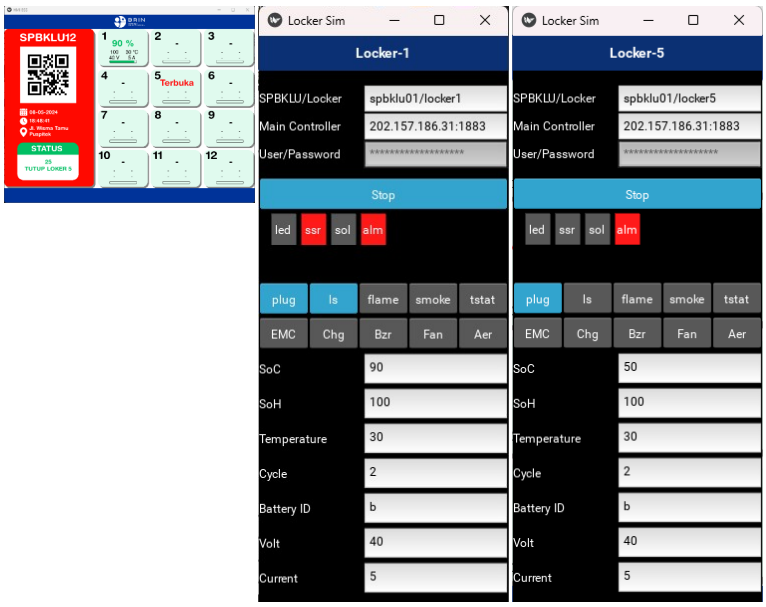
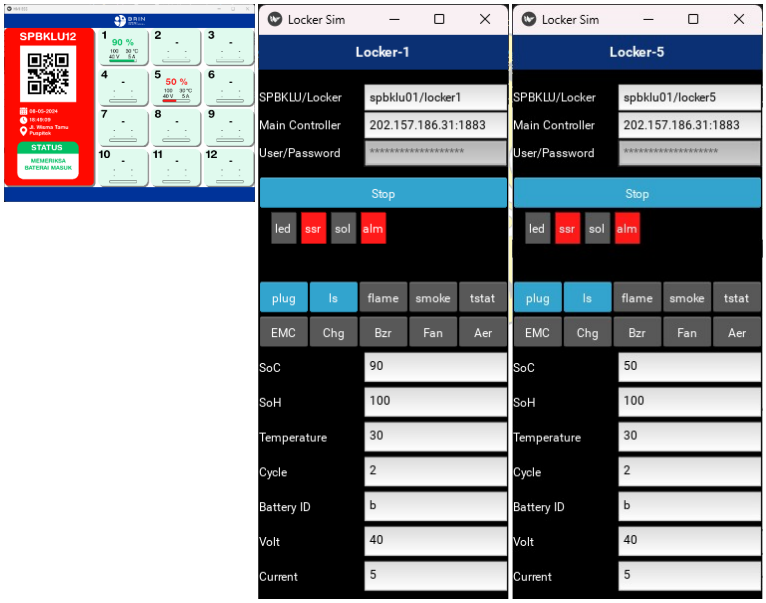
Untuk dapat mempermudah dalam memahami sistem transaksi *battery swapping*, maka akan dilakukan percobaan swapping dengan menggunakan HMI, MQTTX untuk mengirimkan pesan permintaan transaksi *battery swapping* dan juga locker simulator yang akan mensimulasikan loker 1 dan 5.

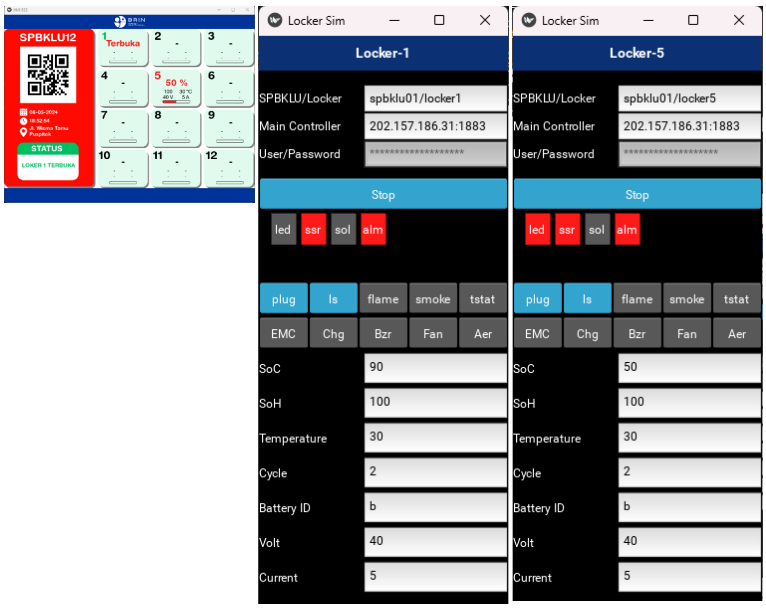
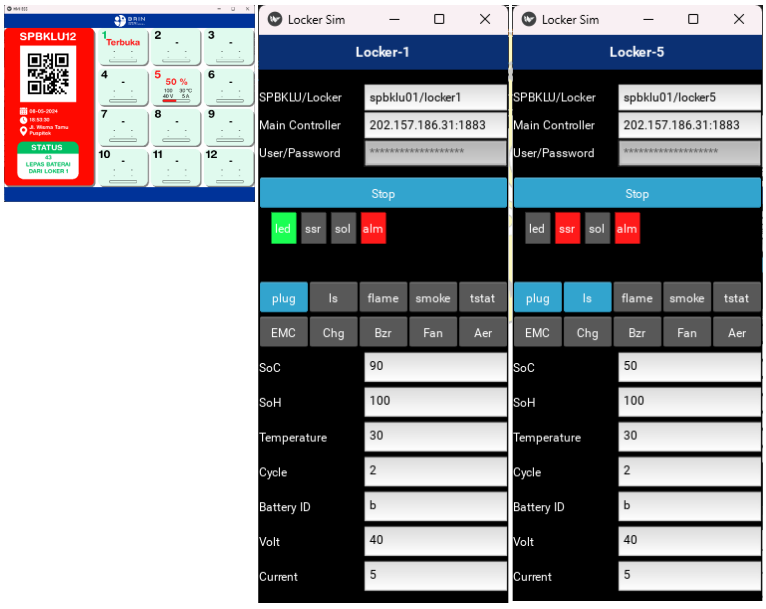
Table 4.2: Simulasi transaksi *battery swapping*

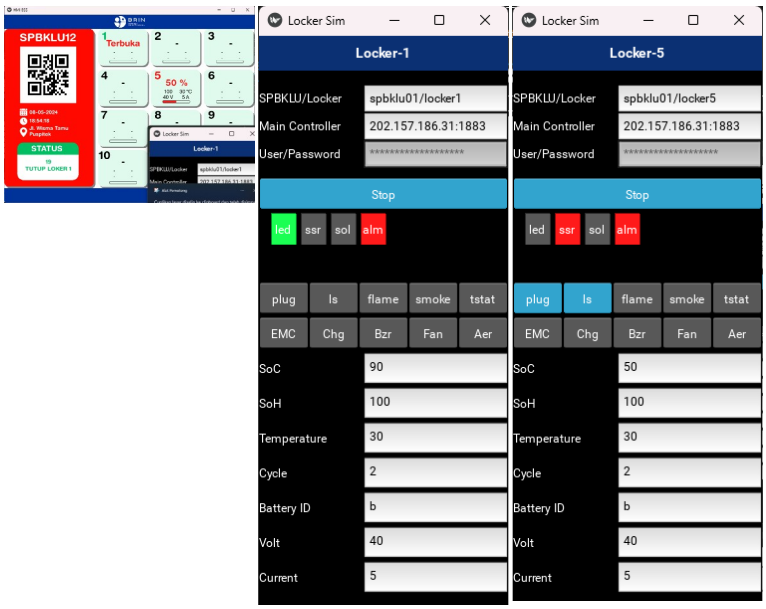
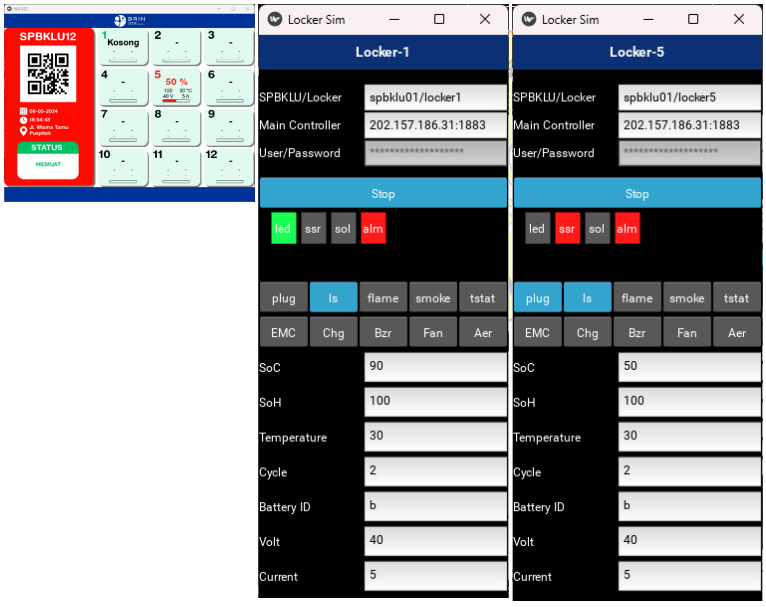
Simulasi Transaksi *Battery Swapping*

No	Tampilan HMI dan MQTTX	Tampilan loker 1	Tampilan loker 5	Penjelasan
1				**Kondisi saat ini:** SPBKLU memiliki 1 loker kosong (loker 5) dan 1 loker dengan baterai yang memiliki SoC 90 (loker 1) sehingga status SPBKLU = 1
2				**Kondisi saat ini:** MQTTX mengirimkan pesan ke sistem kontrol utama sebagai permintaan untuk melakukan transaksi *battery swapping.* **Tindakan:** Selanjutnya sistem kontrol utama memeriksa status SPBKLU dan karena status SPBKLU bernilai 1 maka transaksi *battery swapping* dapat dilakukan.

3		**Kondisi saat ini:** Transaksi *battery swapping* dimulai **Tindakan:** Tahap 1, sistem kontrol utama akan membuka pintu loker kosong yaitu loker 5. **Syarat lanjut:** Tahap ini akan berakhir ketika sistem kontrol utama mengidentifikasi bahwa pintu loker 5 sudah terbuka.
4		**Kondisi saat ini:** Pintu loker kosong sudah terbuka sehingga proses *battery swapping* berlanjut ke tahap 2 **Tindakan:** Tahap 2, Pengguna diminta untuk menghubungkan baterai ke socket yang ada pada locker 5. **Syarat lanjut:** Tahap ini akan selesai ketika sistem kontrol utama mengidentifikasi bahwa baterai sudah terhubung dengan socket pada loker 5.

5		**Kondisi saat ini:** Baterai sudah terhubung dengan socket pada loker 5. **Tindakan:** Tahap 3, Pengguna diminta untuk menutup pintu loker 5. **Syarat lanjut:** Tahap ini akan selesai ketika sistem kontrol utama mengidentifikasi bahwa pintu loker 5 sudah tertutup.
6		**Kondisi saat ini:** Pintu loker 5 sudah tertutup. **Tindakan:** Tahap 4, sistem kontrol utama akan memerikasa ID dan SoH baterai. **Syarat lanjut:** Ttahap ini akan selesai ketika sistem kontrol utama telah selesai memeriksa ID dan SoH baterai.

7		**Kondisi saat ini:** Sistem kontrol utama telah memeriksa ID dan SoH baterai. **Tindakan:** Tahap 5, sistem kontrol utama membuka pintu loker yang memiliki SoC tertinggi yaitu loker 1. **Syarat lanjut:** Tahap ini akan berakhir ketika sistem kontrol utama mengidentifikasi bahwa pintu loker 1 sudah terbuka.
8		**Kondisi saat ini:** Pintu loker 1 sudah terbuka. **Tindakan:** Tahap 6, pengguna diminta untuk melepaskan baterai dari socket loker 1. **Syarat lanjut:** Tahap ini akan selesai ketika baterai telah dilepas dari socket pada loker 1.

9		**Kondisi saat ini:** Baterai telah dilepas dari socket pada loker 1. **Tindakan:** Tahap 7, Pengguna diminta untuk menutup pintu loker 1. **Syarat lanjut:** Tahap ini akan selesai ketika sistem kontrol utama mengidentifikasi bahwa pintu loker 1 sudah tertutup.
10		**Kondisi saat ini:** Sistem kontrol utama telah mengidentifikasi pintu loker 1 sudah tertutup. **Tindakan:** Tahap 8, sistem kontrol utama akan mengirimkan data transaksi *battery swapping* ke *backend* dan transaksi selesai.

4.3.5 Perancangan Log Data

Subab ini akan membahas tentang sistem log data dari SPBKLU. Ada 2 sistem log data, yaitu *log data system* dan *log data transaction* atau juga bisa disebut *log data swapping.* Untuk metode yang digunakan adalah sebagai berikut.

Method logSystem: Membuat *Log Data System*

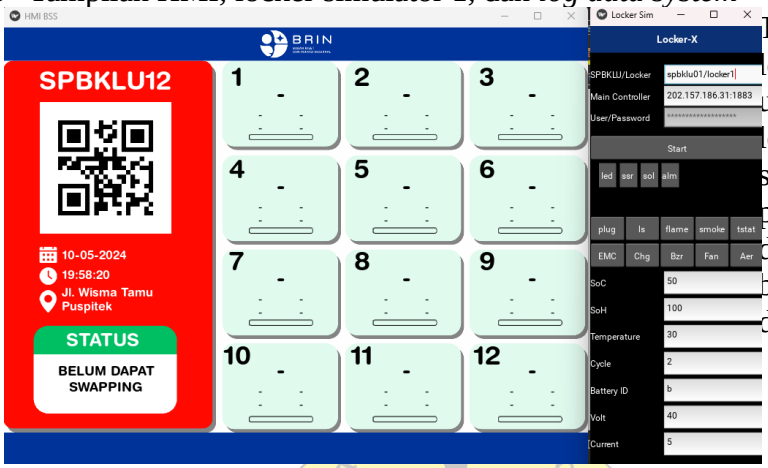
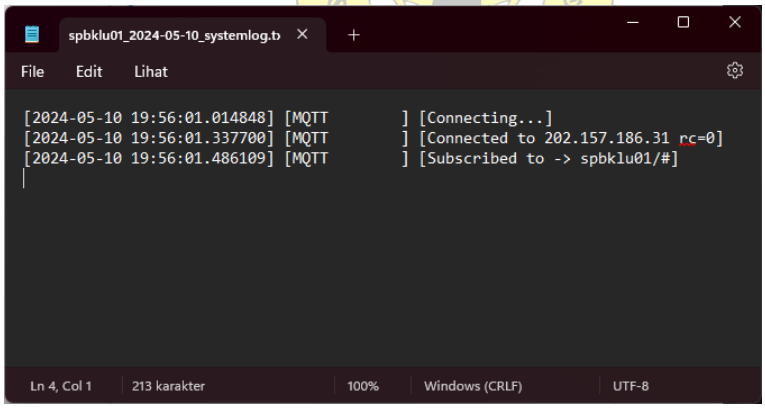
```
#Membuat log system normal
def logSystem(lbl_1, lbl_2):
  tagLength=12
  dt = datetime.datetime.now()
  fileName =
f"/home/pi/Documents/spbklu12-hmi/Log_Data/{MainApp.spbkluN
ame}_{dt.year}-{dt.month:02}-{dt.day}_systemlog.txt" #ysf

  #Ketika tag kurang dari 12 karakter
  if(len(lbl_1)<=tagLength):
    tagString=lbl_1
    for i in range(tagLength-len(lbl_1)):
      tagString = tagString +" "
  #Ketika tag lebih dari 12 karakter
  else:
    tagString = lbl_1[0:12:1] #[start:end:increment]
  msgString = f"[{dt}] [{tagString}] [{lbl_2}]"

  #Membuat atau menyimpan log pada file
  f_data = open(fileName,"a")
  f_data.write(f"{msgString}\n")
  print(msgString)
```

Method ini memiliki 2 parameter yaitu `lbl_1` dan `lbl_2`.Variebel `lbl_1` Berfungsi untuk menampung pesan yang ingin dicantumkan sedangkan variebel `lbl_2` berfungsi untuk menampung data yang ingin disimpan. Tidak ada batasan jumlah karakter terhadap data yang ingin disimpan, tapi pada pesan terdapat batasan yaitu jumlah karakter maksimal 12 buah apabila pesan lebih dari 12 karakter maka hanya 12 karakter pertama yang akan dicantumkan. Hal ini bertujuan untuk membuat log data system lebih rapi sehingga mudah dibaca. Log data system akan disimpan di dalam folder “Log_Data” dengan nama file “SPBKLU01_(Tahun)-(bulan)-(tanggal)_systemlog.txt” yang disesuaikan dengan waktu saat ini. Jika file dengan nama yang sesuai telah tersedia, maka data akan dituliskan pada file tersebut. Namun, jika file dengan nama yang sesuai tidak tersedia, maka akan dibuat file baru dengan nama yang sesuai kemudian data akan

dituliskan di dalamnya. Berikut adalah simulasi untuk mengetahui sistem *log data system.*

Tabel 4.3: Simulasi data log system

Simulasi data log system

No	Tampilan HMI, locker simulator 1, dan *log data system*	Penjelasan
1	 	Tampilan HMI, locker simulator untuk loker 1, dan log data system sesaat setelah program utama dijalankan dan belum terkoneksi dengan loker.
2		Tampilan HMI, locker simulator untuk loker 1, dan log data system setelah loker 1 terhubung dengan sistem kontrol utama.

3 Tampilan HMI, locker simulator untuk loker 1, dan log data system setelah loker 1 dihubungkan dengan baterai dan pintunya ditutup.

Log data system akan digunakan untuk mencatat kerja sistem, mulai dari koneksi dengan broker, topik subcribe, hingga data loker. Untuk data loker, ketika loker tertentu baru saja terhubung ke sistem kontrol utama maka semua data loker

akan dicatat. Namun, selanjutnya sistem kontrol hanya mencatat data-data yang berubah dari data yang sebelumnya dicatat.

Method logSwappingJSON: Membuat *Log Data Transaction*

```
#Membuat log swapping
def logSwappingJSON():
  #Update data tertentu pada tahap tertentu
  if MainApp.swapping == 1 :
    dt = datetime.datetime.now()
    MainApp.startSwapping = f"{dt}"
    MainApp.lockerBatIn = MainApp.selectLockerSwap+1
  elif MainApp.swapping == 4 :
    MainApp.batIn = {
      "batSOC" : MainApp.batSOC[MainApp.lockerBatIn-1],
      "batSOH" : MainApp.batSOH[MainApp.lockerBatIn-1],
      "batVolt" : MainApp.batVolt[MainApp.lockerBatIn-1],
      "batCurr" : MainApp.batCurr[MainApp.lockerBatIn-1],
      "batTemp_cell" :
MainApp.batTemp_cell1[MainApp.lockerBatIn-1],
      "batID" : MainApp.batID[MainApp.lockerBatIn-1],
      "batCycle" :
int(MainApp.batCycle[MainApp.lockerBatIn-1])
    }
  elif MainApp.swapping == 5 :
    MainApp.lockerBatOut = MainApp.selectLockerSwap+1
    MainApp.batOut = {
      "batSOC" : MainApp.batSOC[MainApp.lockerBatOut-1],
      "batSOH" : MainApp.batSOH[MainApp.lockerBatOut-1],
      "batVolt" : MainApp.batVolt[MainApp.lockerBatOut-1],
      "batCurr" : MainApp.batCurr[MainApp.lockerBatOut-1],
      "batTemp_cell" :
MainApp.batTemp_cell1[MainApp.lockerBatOut-1],
      "batID" : MainApp.batID[MainApp.lockerBatOut-1],
      "batCycle" :
int(MainApp.batCycle[MainApp.lockerBatOut-1])
    }
  elif MainApp.swapping == 15 :
    MainApp.lockerBatOut = MainApp.selectLockerSwap+1
    MainApp.batOut = {
      "batSOC" : MainApp.batSOC[MainApp.lockerBatOut-1],
      "batSOH" : MainApp.batSOH[MainApp.lockerBatOut-1],
      "batVolt" : MainApp.batVolt[MainApp.lockerBatOut-1],
      "batCurr" : MainApp.batCurr[MainApp.lockerBatOut-1],
      "batTemp_cell" :
MainApp.batTemp_cell1[MainApp.lockerBatOut-1],
      "batID" : MainApp.batID[MainApp.lockerBatOut-1],
```

```
      "batCycle" :
int(MainApp.batCycle[MainApp.lockerBatOut-1])
    }
  elif MainApp.stop == True :
    MainApp.mqttTopicUpdate()
    MainApp.mqttc.publish(MainApp.topicToUser,
json.dumps(MainApp.dataSwapping))
    MainApp.statusSPBKLU = 0
    data = {
      "topic": "spbklu/updateStatusBss",
      "spbkluid": MainApp.spbkluName,
      "status": MainApp.statusSPBKLU
    }
    MainApp.mqttc.publish(data["topic"], json.dumps(data))
    MainApp.swapping = 0
    MainApp.stop = False
  elif MainApp.swapping == 8 :
    dt = datetime.datetime.now()
    MainApp.endSwapping = f"{dt}"
    MainApp.stop = True

  #Data JSON
  MainApp.dataSwapping = {
    "transactionID": MainApp.transactionID,
    "userID": MainApp.userID,
    "SPBKLU": MainApp.spbkluName,
    "start": MainApp.startSwapping,
    "lockerBatIn": MainApp.lockerBatIn,
    "batIn": MainApp.batIn,
    "lockerBatOut": MainApp.lockerBatOut,
    "batOut": MainApp.batOut,
    "statusSwapping": MainApp.swapping,
    "end": MainApp.endSwapping,
    "stop": MainApp.stop
  }

  #Membuat atau menyimpan log pada file
  try:
    if MainApp.collectData == True:
      with open(f"\home\pi\Documents\spbklu12-hmi\
Log_Data\{MainApp.month} (log swapping).json", "r") as r:
        MainApp.dataSwapPrev = json.load(r)
    MainApp.printDataSwap = MainApp.dataSwapPrev.copy()
    with open(f"\home\pi\Documents\spbklu12-hmi\Log_Data\
{MainApp.month} (log swapping).json", "w") as w:
      MainApp.printDataSwap.append(MainApp.dataSwapping)
      json.dump(MainApp.printDataSwap,w,indent=4)
      MainApp.printDataSwap.clear()
  except:
    with open(f"\home\pi\Documents\spbklu12-hmi\Log_Data\
{MainApp.month} (log swapping).json", "w") as w :
```

```
        w.write("[]")
    if MainApp.collectData == True:
        with open(f"\home\pi\Documents\spbklu12-hmi\
Log_Data\{MainApp.month} (log swapping).json", "r") as r:
            MainApp.dataSwapPrev = json.load(r)
    MainApp.printDataSwap = MainApp.dataSwapPrev.copy()
    with open(f"\home\pi\Documents\spbklu12-hmi\Log_Data\
{MainApp.month} (log swapping).json", "w") as w:
        MainApp.printDataSwap.append(MainApp.dataSwapping)
        json.dump(MainApp.printDataSwap,w,indent=4)
        MainApp.printDataSwap.clear()
```

Method ini berfungsi untuk mencatat data transaksi *battery swapping*. Data yang dicatat di dalam sistem adalah ID transaksi, ID user, nama SPBKLU, waktu mulai transaksi, loker tempat baterai masuk, data baterai yang masuk, loker tempat baterai keluar, data baterai yang keluar, tahapan terakhir *battery swapping*, waktu transaksi selesai, dan status transaksi. *Method* ini hanya dipanggil saat tansaksi *battery swapping* sedang berlangsung, ketika dipanggil *method* ini akan menyimpan data yang telah disebutkan ke dalam variebel JSON `dataSwapping` kemudian menuliskannya dalam file JSON. Data akan diambil secara berkala menyesuaikan dengan tahapan *battery swaaping* yang dilakukan, terdapat data yang hanya diambil sekali selama transaksi dan ada yang diambil secara berkala seperti tahapan *battery swapping* terakhir.

Ketika dipanggil, *method* akan memeriksa tahapan *battery swapping* saat ini dan mengambil data tertentu sesuai dengan tahapannya. Berikut adalah daftar data yang akan diambil berdasarkan tahapan *battery swapping.*

Tabel 4.4 Data yang diambil sesuai dengan tahapan b*attery swapping*

Tahapan *Battery Swapping*	Data yang diambil
1 (SPBKLU membuka pintu loker kosong)	• ID transaksi • ID user • Nama SPBKLU

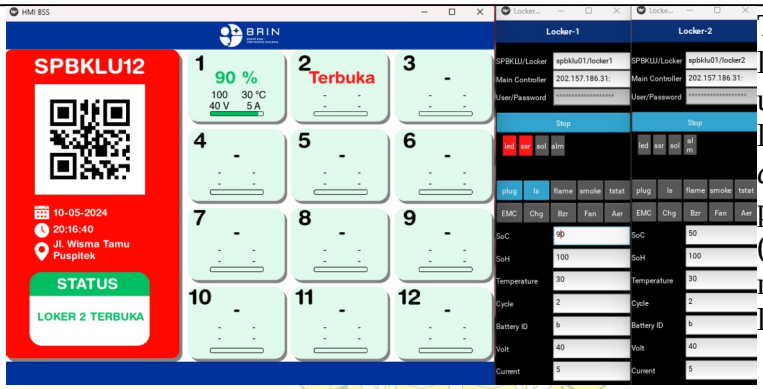
	• Waktu mulai transaksi • Loker tempat baterai masuk • Tahapan b*attery swapping* terakhir
2 (Pengguna menghubungkan baterai dengan socket)	• Tahapan b*attery swapping* terakhir
3 (Pengguna menutup pintu loker)	• Tahapan b*attery swapping* terakhir
4 (Memeriksa baterai)	• Data baterai yang masuk • Tahapan b*attery swapping* terakhir
5 (SPBKLU membuka loker yang memiliki baterai dengan SoC tertinggi)	• Data baterai yang keluar • Loker tempat baterai keluar • Tahapan b*attery swapping* terakhir
6 (Pengguna melepas baterai dari socket)	• Tahapan b*attery swapping* terakhir
7 (Pengguna menutup pintu loker)	• Tahapan b*attery swapping* terakhir
8 (SPBKLU merngirimkan data transaksi ke database dan transaksi selesai)	• Status transaksi • Tahapan b*attery swapping* terakhir

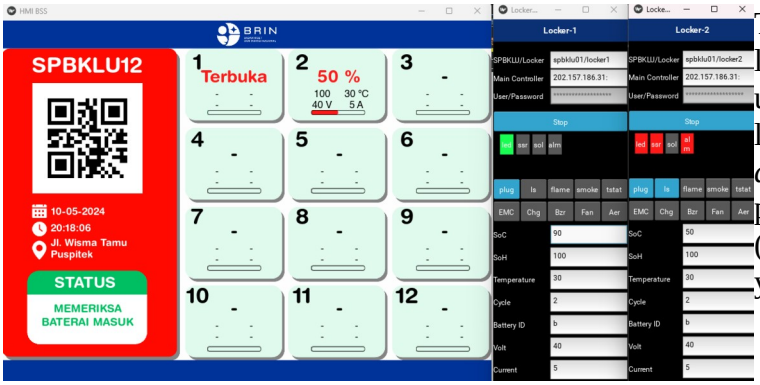
Data akan ditulis dalam sebuah file JSON dengan nama “(bulan)-(tahun) logswapping.json) yang berada di dalam folder “Log_data”. Jika file dengan nama yang sesuai telah tersedia, maka data akan dituliskan pada file tersebut. Namun, jika file dengan nama yang sesuai tidak tersedia, maka akan dibuat file baru dengan nama yang sesuai kemudian data akan dituliskan di dalamnya.

Adapun untuk metode penulisan data dalam sistem *log data transaction* ada 2, yaitu metode update dan tambah. Ketika transaksi dimulai, maka data akan dituliskan dengan metode tambah yaitu menambahkan data yang sebelumnya dengan data yang ada saat ini. Namun, setelah transaksi dimulai data pada file *log*

data transaction hanya akan diupdate. Hal ini bertujuan untuk membuat *data log transaction* menyimpan data secara *realtime* sehingga apabila terdapat gangguan yang menyebabkan transaksi gagal, data terakhir transaksi sudah tertulis pada file *log data transaction.* Untuk dapat lebih memahami mengenai *data log transaction*, berikut adalah simulasi *data log transaction.*

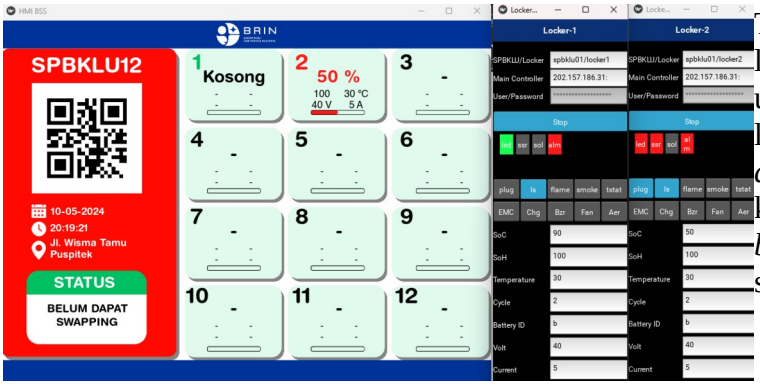
Tabel 4.5: Simulasi *log data transaction*

Simulasi *log data transaction*		
No	Gambar	Penjelasan
1	 	Tampilan HMI, locker simulator untuk loker 1, dan loker 2 serta *log data transaction* pada tahap 1 (SPBKLU membuka pintu loker kosong)

2 

Tampilan HMI, locker simulator untuk loker 1, dan loker 2 serta *log data transaction* pada tahap 4 (Memeriksa baterai yang masuk)


```
{
    "transactionID": "dWgyE2024020214311867957043",
    "userID": 4,
    "spbkluid": "spbklu01",
    "start": "2024-05-10 20:16:38.456791",
    "lockerBatIn": 2,
    "batIn": {
        "batSOC": 50,
        "batSOH": 100,
        "batVolt": 40,
        "batCurr": 5,
        "batTemp_cell": 30,
        "batID": "b",
        "batCycle": 2
    },
    "lockerBatOut": 1,
    "batOut": {
        "batSOC": 90,
        "batSOH": 100,
        "batVolt": 40,
        "batCurr": 5,
        "batTemp_cell": 30,
        "batID": "b",
        "batCycle": 2
    },
    "statusSwapping": 6,
    "end": 0,
    "stop": false
}
```

3 

Tampilan HMI, locker simulator untuk loker 1, dan loker 2 serta *log data transaction* ketika transaksi *battery swapping* sudah selesai.


```
{
    "transactionID": "dWgyE2024020214311867957043",
    "userID": 4,
    "spbkluid": "spbklu01",
    "start": "2024-05-10 20:16:38.456791",
    "lockerBatIn": 2,
    "batIn": {
        "batSOC": 50,
        "batSOH": 100,
        "batVolt": 40,
        "batCurr": 5,
        "batTemp_cell": 30,
        "batID": "b",
        "batCycle": 2
    },
    "lockerBatOut": 1,
    "batOut": {
        "batSOC": 90,
        "batSOH": 100,
        "batVolt": 40,
        "batCurr": 5,
        "batTemp_cell": 30,
        "batID": "b",
        "batCycle": 2
    },
    "statusSwapping": 0,
    "end": "2024-05-10 20:19:15.011735",
    "stop": true
}
```

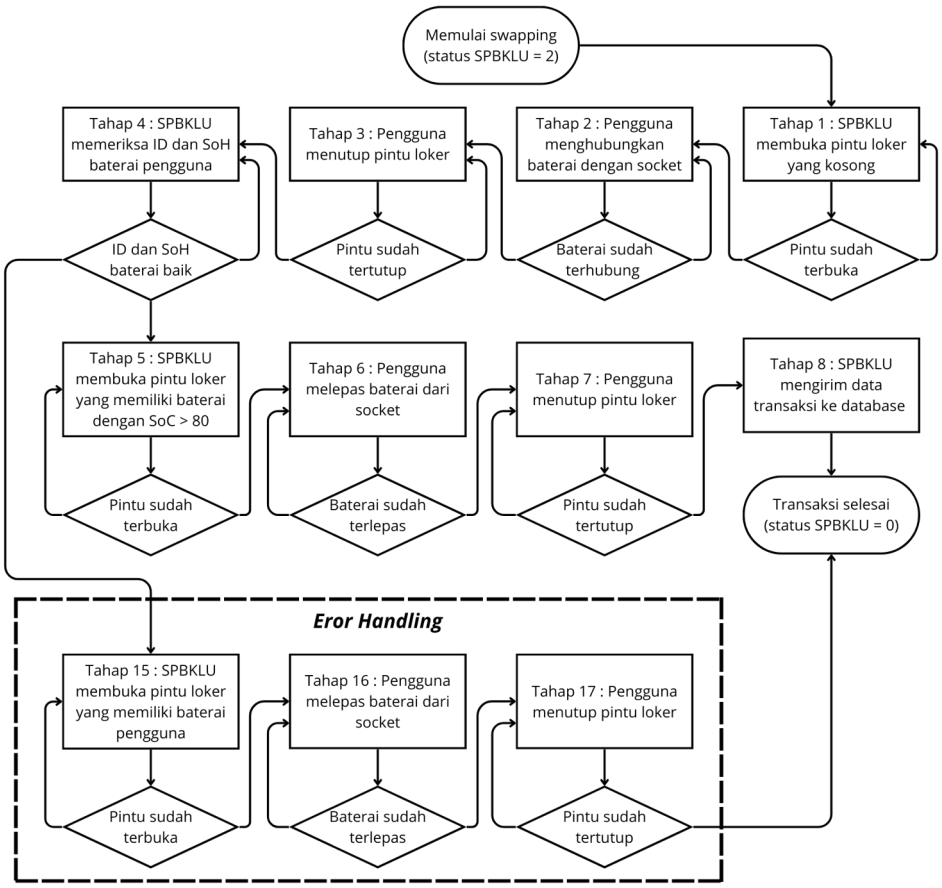
4.3.6 Perancangan *Eror Handling*

Eror handling merupakan penanganan yang dilakukan oleh sistem ketika dihadapkan pada situasi tertentu. Berikut adalah beberapa *eror handling* yang ada pada SPBKLU.

Baterai yang Ditukar Memiliki SoH (Source of Health) di Bawah Standar atau ID Baterai Tidak Terdaftar di Sistem

```
#Memeriksa kondisi baterai (SOH dan ID baterai)
elif MainApp.swapping == 4 :
  self.ids.lbl_keterangan_status.text = "MEMERIKSA BATERAI
MASUK"
  #SOH baterai baik dan ID baterai terdaftar
  if MainApp.batSOH[MainApp.selectLockerSwap] >=
MainApp.batSOH_Ok and
MainApp.batID[MainApp.selectLockerSwap] != "" :
    maxBatt = 0
    for i in range(MainApp.jumlahLoker):
      if(MainApp.batSOC[i] > maxBatt):
        maxBatt = MainApp.batSOC[i]
        MainApp.selectLockerSwap = i
    MainApp.swapping = 5
    MainApp.lockerID =
f"locker{MainApp.selectLockerSwap+1}"
    MainApp.mqttTopicUpdate()
    MainApp.mqttc.publish(MainApp.topicSolenoid, "true")
  #SOH baterai rendah atau ID baterai tidak terdaftar
  else :
    MainApp.swapping = 15
    MainApp.lockerID =
f"locker{MainApp.selectLockerSwap+1}"
    MainApp.mqttTopicUpdate()
    MainApp.mqttc.publish(MainApp.topicSolenoid, "true")

#Loker yang memuat baterai pengguna terbuka kembali
elif MainApp.swapping == 15 :
  self.ids.lbl_keterangan_status.text = f"BATERAI
BERMASALAH, LOKER {MainApp.selectLockerSwap} TERBUKA"
  if MainApp.state_door[MainApp.selectLockerSwap] ==
False :
    MainApp.swapping = 16

#Baterai diambil pengguna
elif MainApp.swapping == 16 :
  self.ids.lbl_keterangan_status.text =
f"{MainApp.timeLockerOpen}\nLEPAS BATERAI DARI LOKER
```

```
{MainApp.selectLockerSwap}"
  if MainApp.state_plug[MainApp.selectLockerSwap] ==
False :
    MainApp.swapping = 17

#Pintu loker tertutup
elif MainApp.swapping == 17 :
  self.ids.lbl_keterangan_status.text =
f"{MainApp.timeLockerOpen}\nTUTUP LOKER
{MainApp.selectLockerSwap + 1}"
  if (MainApp.state_door[MainApp.selectLockerSwap] == True
and MainApp.state_plug[MainApp.selectLockerSwap] ==
False) :
    MainApp.swapping = 8

#SPBKLU mengirim data transaksi ke database dan transaksi
selesai
elif MainApp.swapping == 8 :
   self.ids.lbl_keterangan_status.text = "MEMUAT"
   MainApp.mqttc.publish(MainApp.topicAlarm, "false")
   Background.stop_transaction()
   MainApp.statusSPBKLU = 0
```



Gambar 4.51: Flowchart eror handling ketika ID dan SoH baterai tidak sesuai persyaratan

Dalam melakukan transaksi *battery swapping* terkadang terdapat masalah dimana baterai pengguna memiliki SoH yang buruk (kurang dari 60) atau ID baterainya yang tidak terdaftar. Hal itu dapat membahayakan pengguna selanjutnya dari baterai tersebut, untuk itu diperlukan tindakan untuk hal ini.

Pada tahap 4, baterai pengguna yang telah dimasukkan ke dalam loker akan diperiksa datanya. Jika baterai pengguna tenyata dinyatakan rusak atau memiliki ID yang tidak terdaftar, maka sistem akan menjalankan tindakan khusus

yaitu mengembalikan baterai tersebut ke pengguna dengan menggunakan tahap 15, 16, dan 17.

Tahap 15, sistem kontrol utama akan memerintahakan subkontrol pada loker yang menerima baterai pengguna untuk membuka pintu lokernya kembali dan HMI akan menampilkan “BATERAI BERMASALAH, LOKER X TERBUKA”. Tahap ini akan berakhir ketika sistem konrtol utama mengidentifikasi bahwa pintu loker tersebut telah terbuka.

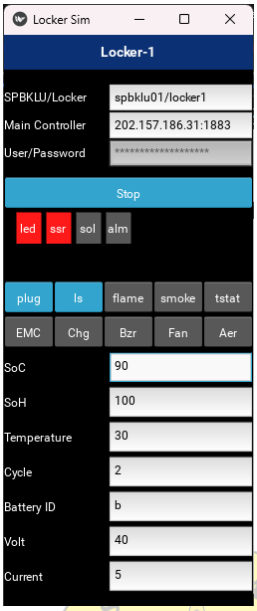
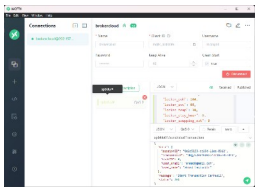
Tahap 16, pengguna akan diminta untuk mengambil kembali baterai yang dimasukkan dengan menampilkan “LEPAS BATERAI” pada HMI. Tahap ini akan berakhir ketika sistem kontrol utama mengidentifikasi bahwa baterai telah dilepaskan dari socket.

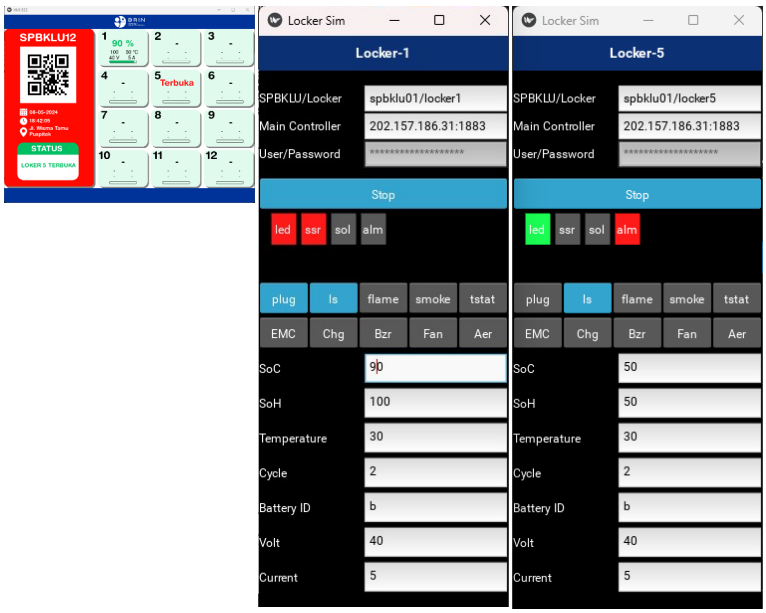
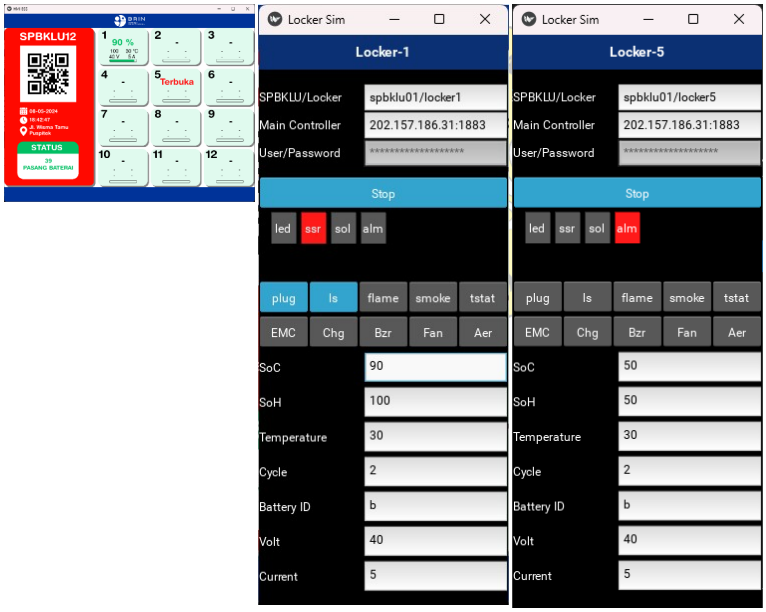
Tahap 17, pengguna akan diminta untuk mengambil kembali baterai yang dimasukkan dengan menampilkan “TUTUP PINTU LOKER” pada HMI. Tahap ini akan berakhir ketika sistem kontrol utama mengidentifikasi bahwa pintu loker tersebut sudah tertutup. Tahap ini akan berakhir ketika sistem kontrol utama mengidentifikasi bahwa pintu loker tersebut telah tertutup. Selanjutnya transaksi akan berlanjut ke tahap 8 untuk menyelesaikan transaksi *battery swapping*.

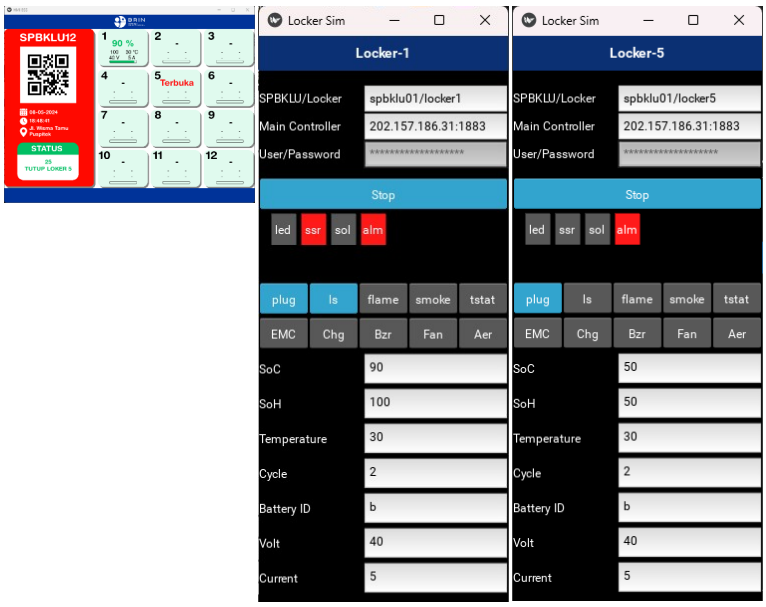
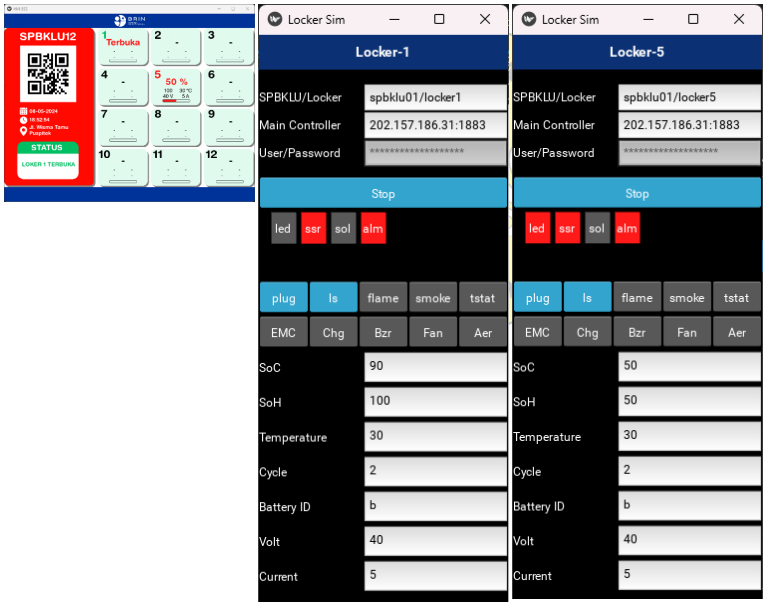
Untuk dapat lebih memahami sistem *eror handling* dalam situasi ini, maka akan dilakukan percobaan menggunakan HMI, MQTTX untuk mengirim pesan permintaan transaksi *battery swapping*, dan program locker simulator untuk mensimulasikan loker 1 dan 5.

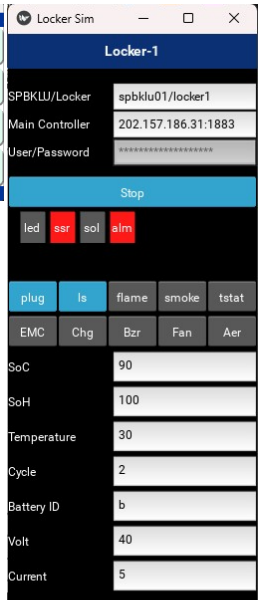
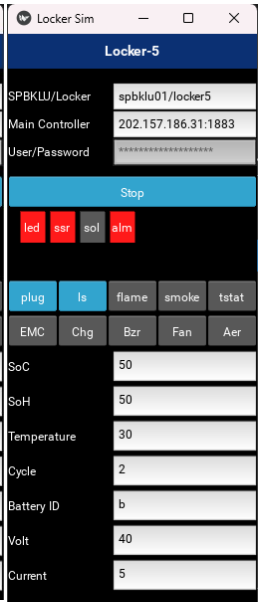
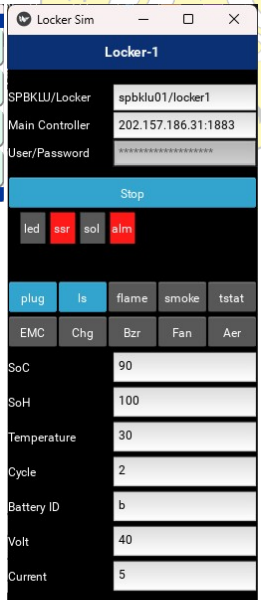
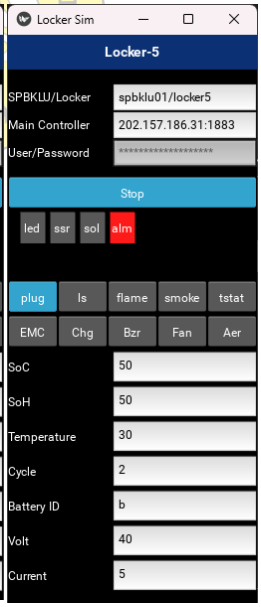
Tabel 4.6: Simulasi *eror handling* ketika ID atau SoH baterai tidak memenuhi syarat

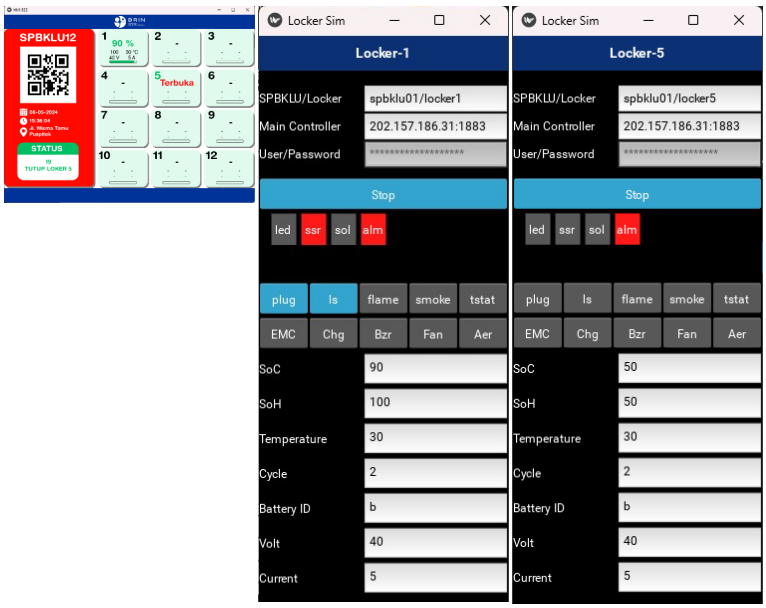
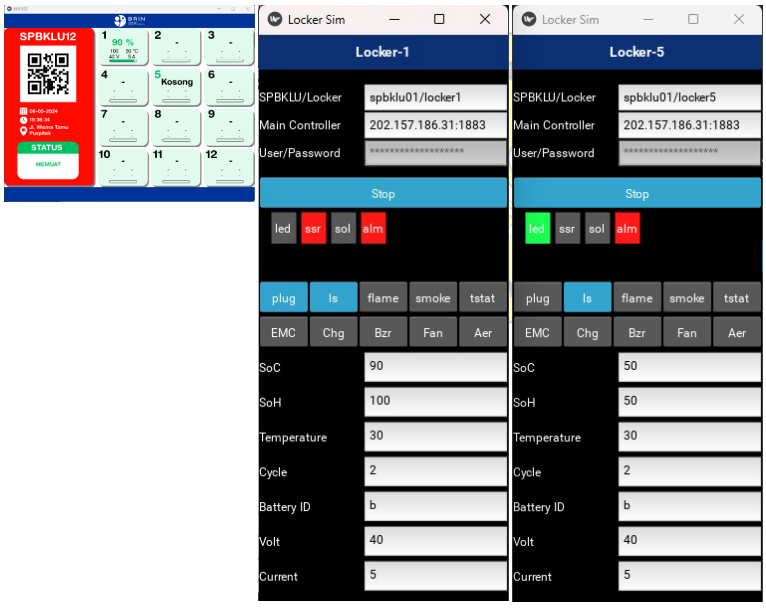
Simulasi *Eror Handling* Ketika ID atau SoH Baterai Tidak Memenuhi Syarat

No	Tampilan HMI dan MQTTX	Tampilan loker 1	Tampilan loker 5	Penjelasan
1				**Kondisi saat ini:** SPBKLU memiliki 1 loker kosong (loker 5) dan 1 loker dengan baterai yang memiliki SoC 90 (loker 1) sehingga status SPBKLU = 1.
2				**Kondisi saat ini:** MQTTX mengirimkan pesan ke sistem kontrol utama sebagai permintaan untuk melakukan transaksi *battery swapping.* Selanjutnya sistem kontrol utama memeriksa status SPBKLU dan karena status SPBKLU bernilai 1 maka transaksi *battery swapping* dapat dilakukan.

3		**Kondisi saat ini:** transaksi *battery swapping* dimulai **Tindakan saat ini:** Tahap 1, sistem kontrol utama akan membuka pintu loker kosong yaitu loker 5. **Syarat lanjut:** Tahap ini akan berakhir ketika sistem kontrol utama mengidentifikasi bahwa pintu loker 5 sudah terbuka.
4		**Kondisi saat ini:** Pintu loker kosong sudah terbuka sehingga proses *battery swapping* berlanjut ke tahap 2 **Tindakan saat ini:** Tahap 2, Pengguna diminta untuk menghubungkan baterai ke socket yang ada pada locker 5. **Syarat lanjut:** Tahap ini akan selesai ketika sistem kontrol utama mengidentifikasi bahwa baterai sudah terhubung dengan socket pada loker 5.

5		**Kondisi saat ini:** Baterai sudah terhubung dengan socket pada loker 5. **Tindakan saat ini:** Tahap 3, Pengguna diminta untuk menutup pintu loker 5. **Syarat lanjut:** Tahap ini akan selesai ketika sistem kontrol utama mengidentifikasi bahwa pintu loker 5 sudah tertutup.
6		**Kondisi saat ini:** Pintu loker 5 sudah tertutup. **Tindakan saat ini:** Tahap 4, sistem kontrol utama akan memerikasa ID dan SoH baterai. **Syarat lanjut:** Tahap ini akan selesai ketika sistem kontrol utama telah selesai memeriksa ID dan SoH baterai.

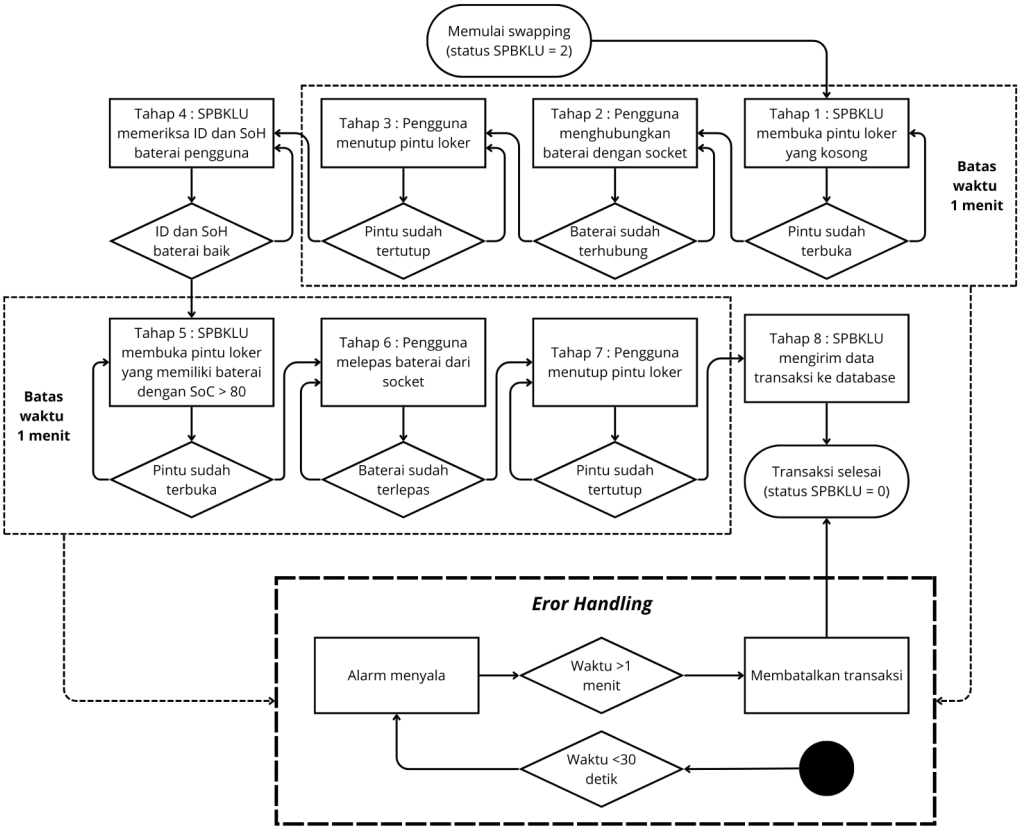
7	  	**Kondisi saat ini:** Sistem kontrol utama selesai memeriksa baterai dan mengidentifikasi bahwa baterai memiliki SoH dengan nilai 50, dikarenakan SoH minimal baterai adalah 60, maka sistem kontrol utama akan menjalankan penanganan khusus untuk situasi ini, **Tindakan saat ini:** Tahap 15, sistem kontrol utama memerintahakan pintu loker 5 terbuka kembali. **Syarat lanjut:** Tahap ini selesai ketika pintu loker 5 sudah terbuka.
8	  	**Kondisi saat ini:** Pintu loker 5 sudah terbuka. **Tindakan saat ini:** Tahap 16, Pengguna diminta untuk mengambil kembali baterai yang sebelumnya dimasukkan pada loker 5 dengan melepaskannya dari socket. **Syarat lanjut:** Tahap ini akan selesai bergitu baterai terlepas dari socket.

9		**Kondisi saat ini:** Baterai pengguna telah terlepas dari socket. **Tindakan saat ini:** Tahap 17, pengguna diminta untuk menutup kembali loker 5. **Syarat lanjut:** Tahap ini akan selesai ketika pintu loker 5 sudah tertutup.
10		**Kondisi saat ini:** Sistem kontrol utama telah mengidentifikasi pintu loker 5 sudah tertutup. **Tindakan saat ini:** Tahap 8, sistem kontrol utama akan mengirimkan data transaksi *battery swapping* ke *backend* dan transaksi selesai.

Pengguna Melakukan *Battery Swapping* lebih dari waktu yang ditentukan

Masih dalam *eror handling* untuk *battery swapping*. Namun, *eror handling* ini dijalakan ketika proses *battery swapping* yang dilakukan pengguna lebih lama dari batas waktu yang telah ditentukan.

```
if MainApp.swapping == 1 or MainApp.swapping == 5 or
MainApp.swapping == 15 :
  MainApp.timeLockerOpen = 60
elif (MainApp.swapping <= 3 or (MainApp.swapping >=5 and
MainApp.swapping <=7) or
  (MainApp.swapping >=15 and MainApp.swapping <=17)):
  if MainApp.timeLockerOpen <= 0:
    MainApp.swapping = 25
    MainApp.mqttc.publish(MainApp.topicAlarm, "false")
  if MainApp.timeLockerOpen > 0 and MainApp.timeLockerOpen
<= 30 :
    MainApp.lockerID =
f"locker{MainApp.selectLockerSwap+1}"
    MainApp.mqttTopicUpdate()
    MainApp.mqttc.publish(MainApp.topicAlarm, "true")
    MainApp.timeLockerOpen -= 1
    printMsg("TIMER SWAP", f"{MainApp.timeLockerOpen}")
```



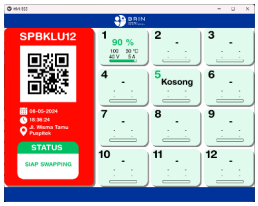
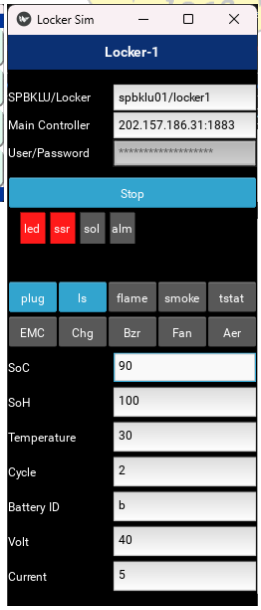
Gambar 4.52: Flowchart eror handling ketika proses battery swapping lebih lama dari batas waktu yang ditentukan

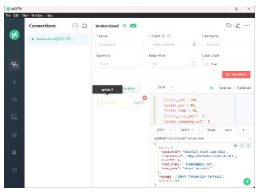
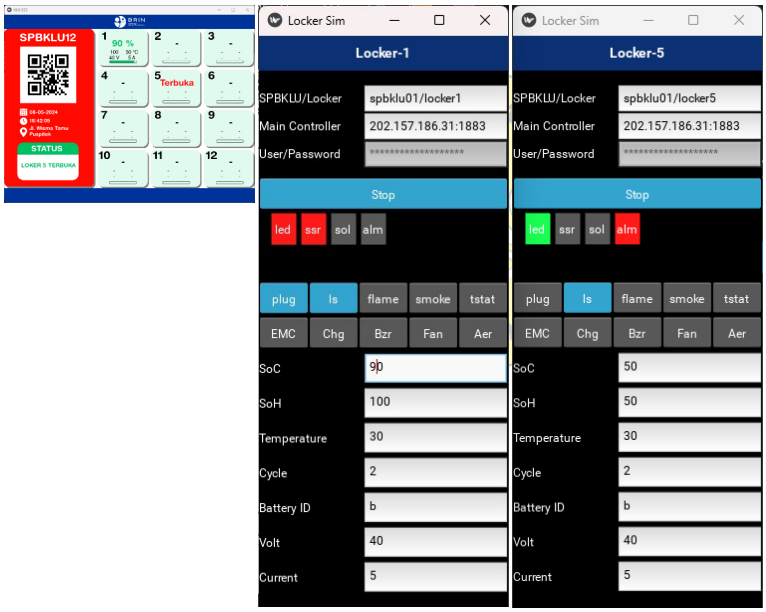
Ketika proses *battery swapping* dilakukan terdapat kemungkinan dimana pengguna melakukan kesalahan seperti meninggalkan SPBKLU ketika transaksi

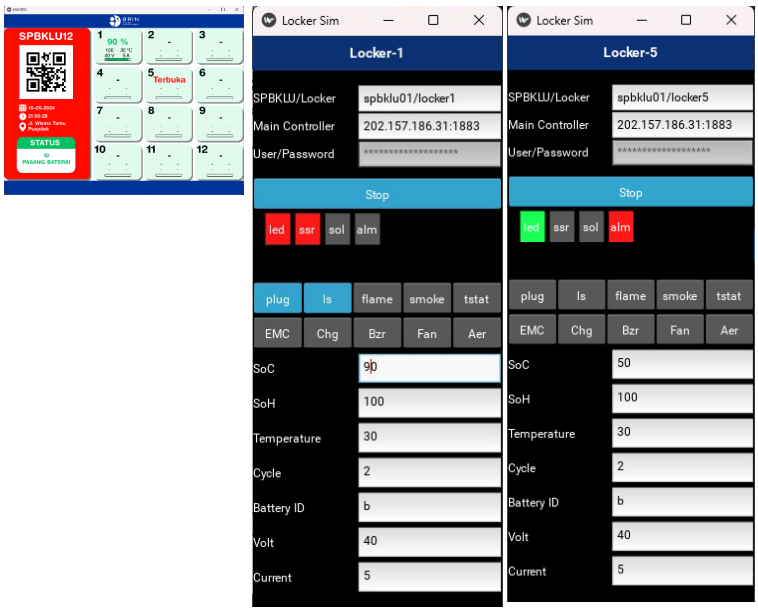
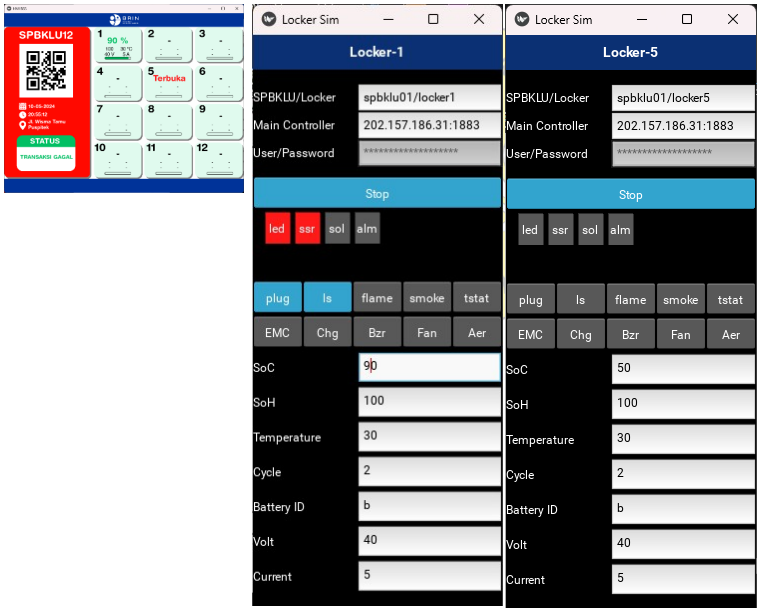
battery swapping belum selesai. Untuk itu dibuat suatu program *eror handling* dalam menangani situasi tersebut.

Pada proses *battery swapping* kesalahan oleh pengguna dapat terjadi hanya pada tahap 1, 2, 3 dan tahap 5, 6, 7 karena tahap lainnya dilakukan oleh mesin. Untuk itu, diberi batasan waktu pada 3 tahap pada 2 waktu tersebut yaitu 1 menit. Dalam tahap 1, 2, 3 atau tahap 5, 6, 7, ketika waktu sudah mencapai lebih dari 30 detik dan akan berhenti ketika pengguna berhasil lanjut ke tahap 4 atau 8. Namun, jika waktu melebihi 1 menit sistem akan membatalkan transaksi dan transaksi selesai.

Tabel 4.7: Simulasi *eror handling* ketika waktu proses *battery swapping* lebih dari 1 menit

Simulasi *eror handling* ketika waktu proses *battery swapping* lebih dari 1 menit				
No	Tampilan HMI dan MQTTX	Tampilan loker 1	Tampilan loker 5	Penjelasan
1				**Kondisi saat ini:** SPBKLU memiliki 1 loker kosong (loker 5) dan 1 loker dengan baterai yang memiliki SoC 90 (loker 1) sehingga status SPBKLU = 1. **Tindakan:** tidak ada, karena belum ada permintaan untuk melakukan transaksi *battery swapping.*

2		**Kondisi saat ini:** MQTTX mengirimkan pesan ke sistem kontrol utama sebagai permintaan untuk melakukan transaksi *battery swapping.* **Tindakan:** Selanjutnya sistem kontrol utama memeriksa status SPBKLU dan karena status SPBKLU bernilai 1 maka transaksi *battery swapping* dapat dilakukan.
3		**Kondisi saat ini:** Tahap 1, sistem kontrol utama akan membuka pintu loker kosong yaitu loker 5. **Tindakan:** Mulai dari tahap ini sistem akan menghitung waktu transaksi *battery swapping* dari loker terbuka hingga tertutup. Waktu maksimal 60 detik

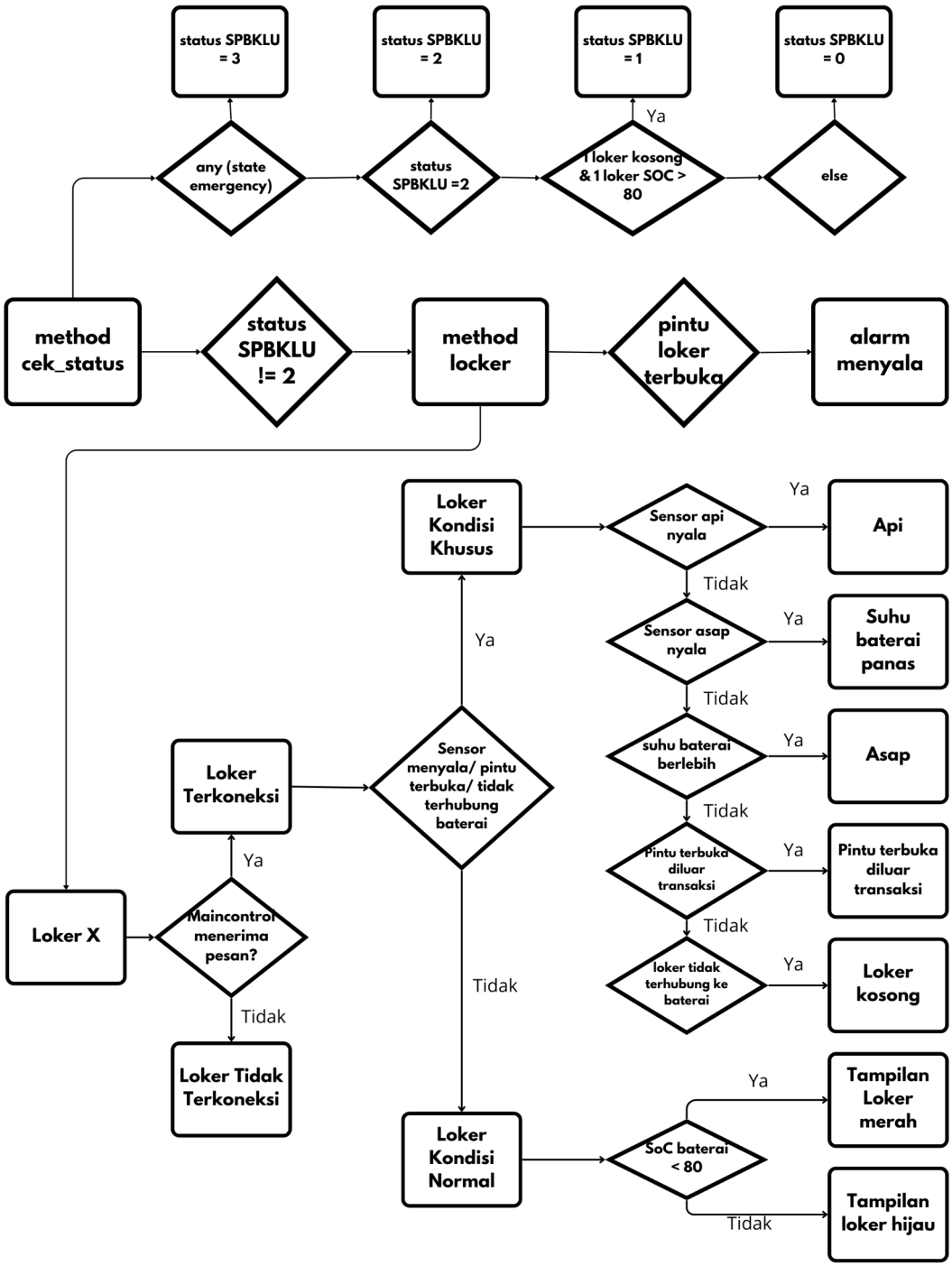
4		**Kondisi saat ini:** Waktu transaksi sudah lebih dari 30 detik. **Tindakan:** Alarm akan menyala hingga baterai dihubungkan ke socket dan pintu loker ditutup.
5		**Kondisi saat ini:** Waktu transaksi *battery swapping* lebih dari 60 detik. **Tindakan:** Alarm dimatikan serta transaksi digagalkan dan transaksi dianggap selesai.

Pintu Loker yang Terbuka di Luar Proses *Battery Swapping*

Berdasarkan gambar 4.32 yang merupakan *flowchart* untuk method locker yang berfungsi untuk mengecek kondisi loker. Terdapat 4 kondisi khusus yang berbahaya, yaitu api, asap, suhu baterai tinggi, dan pintu loker terbuka tapi yang dipilih untuk diberikan penganan khusus berupa *eror handling* justru loker

terbuka. Hal ini disebabkan kondisi khusus seperti terdapat api, asap, suhu baterai yang tinggi tidak memerlukan informasi mengenai status SPBKLU sedang melakukan transaksi atau tidak sehingga ketika kondisi khusus ini terjadi subkontrol pada loker hanya perlu mengirimkan sinyal untuk menghentikan sistem yang berjalan pada lokernya. Sedangkan untuk mengetahui pintu dari suatu loker terbuka di luar kondisi transaksi membutuhkan informasi mengenai status SPBKLU sehingga penangannya dilakukan oleh sistem kontrol utama. Berikut adalah algoritma program dan flowchart dari *eror handling* dalam situasi yang telah disebutkan.

```
"""Sistem peringatan berupa alarm saat pintu terbuka ketika
tidak melakukan swapping"""
if MainApp.statusSPBKLU != 2:
  if MainApp.alarmTime[i] > 0 :
    MainApp.alarmTime[i] -= 1
  if MainApp.state_door[i] == False and
MainApp.alarmState[i] == 0:
    MainApp.mqttc.publish(MainApp.topicAlarm, "true")
    MainApp.alarmState[i] = 1
    MainApp.alarmTime[i] = 10
  elif MainApp.state_door[i] == True and
(MainApp.alarmState[i] == 1 or MainApp.alarmState[i] == 2):
    MainApp.mqttc.publish(MainApp.topicAlarm, "false")
    MainApp.alarmState[i] = 0
  elif MainApp.alarmTime[i] <= 0 and MainApp.alarmState[i]
== 1 :
    MainApp.mqttc.publish(MainApp.topicAlarm, "false")
    MainApp.alarmState[i] = 2
elif MainApp.statusSPBKLU == 2:
  if (MainApp.swapping == 0 or MainApp.swapping == 1) and
MainApp.alarmState[i] == 1 :
    MainApp.mqttc.publish(MainApp.topicAlarm, "false")
    MainApp.alarmState[i] = 2
```

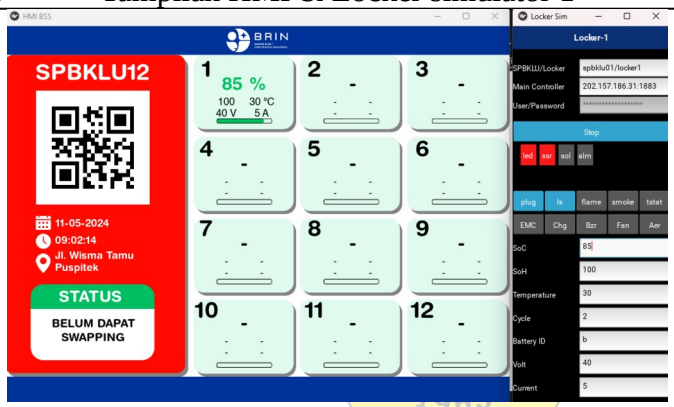


Gambar 4.53: Eror handling ketika pintu loker terbuka di luar kondisi battery swapping

Identifikasi terbukanya loker di luar kondisi transaksi *battery swapping* dapat dilakukan dengan memanfaatkan method yang telah dijelaskan sebelumnya, yaitu method cek_status dan method locker. Method cek_status digunakan untuk

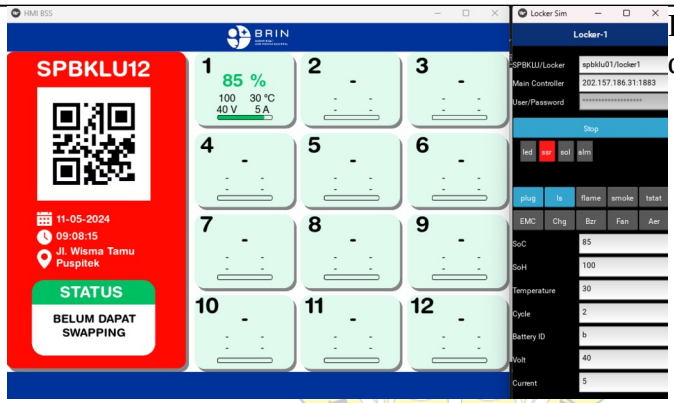
menentukan status SPBKLU saat ini, jika status SPBKLU tidak sama dengan 2 maka dapat diartikan SPBKLU di luar kondisi *battery swapping*. Sedangkan method locker digunakan untuk menentukan pintu suatu loker terbuka atau tidak. Jika status SPBKLU sama dengan 2 dan pintu loker terbuka, maka alarm akan berbunyi.

Tabel 4.8: Simulasi eror handling ketika pintu loker terbuka di luar kondisi *battery swapping*

No	Tampilan HMI & Locker simulator 1	Keterangan
1		• Status SPBKLU != 2 • Pintu loker tertutup • Alarm mati
2		• Status SPBKLU != 2 • Pintu loker terbuka • Alarm menyala

Untuk mematikan alarm dapat dilakukan dengan 2 cara, yaitu menutup pintu loker yang terbuka atau menunggunya selama 10 detik dan alarm akan mati dengan sendirinya.

Tabel 4.9: Cara untuk mematikan alarm

No	Tampilan HMI & Locker simulator 1	Keterangan
1		• Status SPBKLU != 2 • Pintu loker terbuka • Alarm menyala
2		Pintu loker yang terbuka ditutup sehingga • Status SPBKLU != 2 • Pintu loker tertutup • Alarm mati
3		Ditunggu selama 10 detik setelah alarm berbunyi • Status SPBKLU != 2 • Pintu loker terbuka • Alarm mati

4.4 Pengujian

Pada subab ini akan dilakukan pengujian terkait dengan subsistem pada perangkat sistem kontrol SPBKLU dengan tujuan untuk mengetahui keandalan dari sistem yang telah dilakukan rancang bangun.

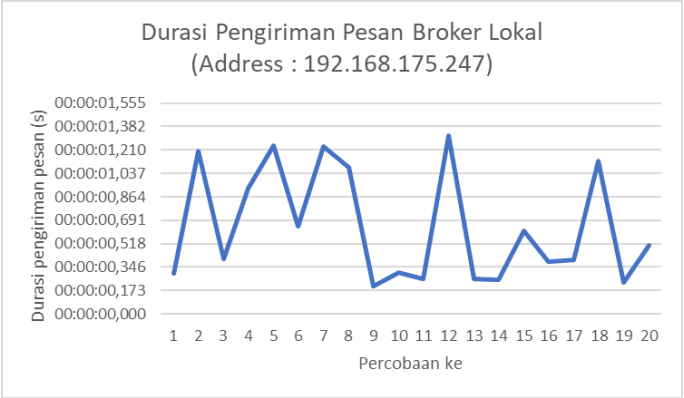
4.4.1 Pengujian Latensi MQTT

Pengujian latensi MQTT dilakukan untuk mengetahui keandalan sistem komunikasi data menggunakan broker lokal pada perangkat Raspberry Pi jika dibandingkan dengan menggunakan broker server yaitu broker ee.unsoed.ac.id atau 202.157.186.31 yang merupakan broker dari backend server. Pengujian akan dilakukan dengan menggunakan Raspberry Pi dan Laptop. Laptop akan mengirimkan pesan MQTT kemudian Raspberry Pi akan menerima pesan tersebut. Waktu pesan terkirim akan dikurangi dengan waktu pesan diterima untuk mencari durasi pengiriman pesan. Jarak antara Laptop dengan Raspberry Pi sekitar 30 cm. Jaringan yang digunakan untuk keduanya adalah wifi dari HP yang menggunakan kartu dengan merek Telkomsel. Pengujian dilakukan di kos rumatoris yang berada di Dusun 2, Blater, kec. Kalimanah, kab. Purbalingga. Berikut adalah data hasil percobaannya.

Tabel 4.10: Hasil Pengujian Latensi Broker Lokal (IP Adress = 192.168.175.247)

Hasil Pengujian Broker Lokal (IP Adress = 192.168.175.247)			
No	Waktu Pesan Dikirim	Waktu Pesan Diterima	Durasi Pengiriman Pesan (s)
1	17:50:20,802	17:50:21,103	0,301
2	17:50:25,208	17:50:26,408	1,200
3	17:50:32,405	17:50:32,807	0,402
4	17:50:37,481	17:50:38,411	0,930
5	17:50:43,161	17:50:44,404	1,243
6	17:50:49,766	17:50:50,413	0,647
7	17:50:55,178	17:50:56,413	1,235
8	17:51:01,327	17:51:02,406	1,079
9	17:51:06,195	17:51:06,402	0,207
10	17:51:10,500	17:51:10,808	0,308
11	17:51:14,140	17:51:14,401	0,261
12	17:51:19,088	17:51:20,405	1,317

13	17:51:24,140	17:51:24,400	0,260
14	17:51:28,958	17:51:29,209	0,251
15	17:51:33,791	17:51:34,403	0,612
16	17:51:38,524	17:51:38,912	0,388
17	17:51:42,905	17:51:43,301	0,396
18	17:51:47,282	17:51:48,407	1,125
19	17:51:52,677	17:51:52,909	0,232
20	17:51:56,897	17:51:57,401	0,504
Rata-rata durasi pengiriman pesan			0,645

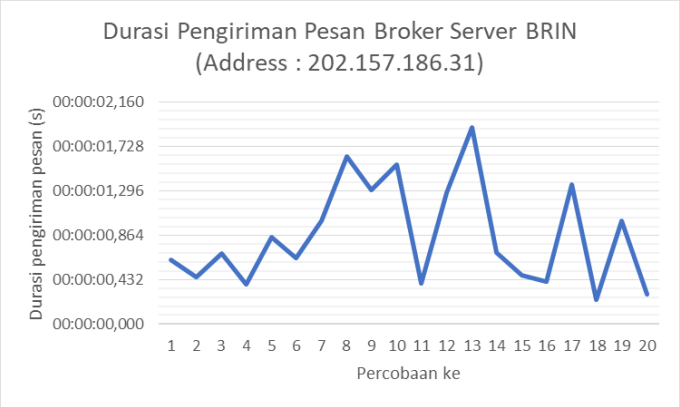


Gambar 4.54: Grafik hasil pengujian broker lokal (IP adress = 192.168.175.247)

Tabel 4.11: Hasil Pengujian Latensi Broker Server (IP Adress = 202.157.186.31)

Pengujian Broker Server (IP Adress = 202.157.186.31)			
No	Waktu Pesan Dikirim	Waktu Pesan Diterima	Durasi Pengiriman Pesan (s)
1	15:28:04,247	15:28:04,868	0,621
2	15:28:08,412	15:28:08,866	0,454
3	15:28:12,188	15:28:12,878	0,690
4	15:28:16,679	15:28:17,068	0,389
5	15:28:20,036	15:28:20,883	0,847
6	15:28:26,220	15:28:26,866	0,646
7	15:28:29,867	15:28:30,868	1,001
8	15:28:35,230	15:28:36,862	1,632
9	15:28:39,579	15:28:40,887	1,308

10	15:28:45,308	15:28:46,860	1,552
11	15:28:50,766	15:28:51,159	0,393
12	15:28:55,598	15:28:56,874	1,276
13	15:29:00,948	15:29:02,866	1,918
14	15:29:06,170	15:29:06,864	0,694
15	15:29:10,406	15:29:10,878	0,472
16	15:29:14,448	15:29:14,860	0,412
17	15:29:19,506	15:29:20,864	1,358
18	15:29:24,747	15:29:24,980	0,233
19	15:29:28,870	15:29:29,875	1,005
20	15:29:34,811	15:29:35,101	0,290
Rata-rata durasi pengiriman pesan			0,860

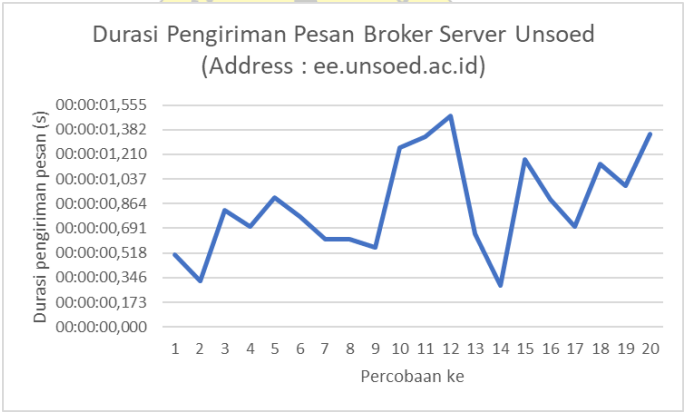


Gambar 4.55: Grafik pengujian broker server BRIN (IP adress = 202.157.186.31)

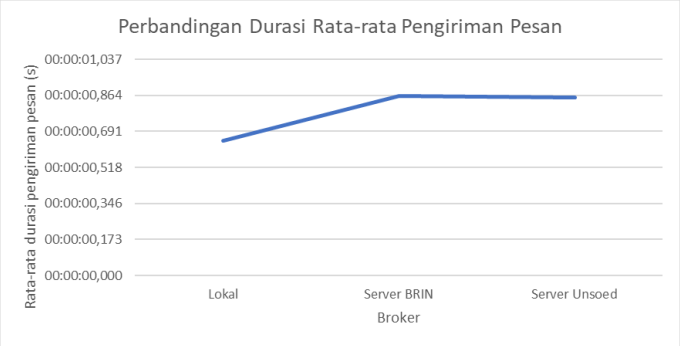
Tabel 4.12: Hasil Pengujian Latensi Broker Server (IP Adress = ee.unsoed.ac.id)

Pengujian Broker Server (IP Adress = ee.unsoed.ac.id)			
No	Waktu Pesan Dikirim	Waktu Pesan Diterima	Durasi Pengiriman Pesan (s)
1	17:47:02,406	17:47:02,912	0,506
2	17:47:06,293	17:47:06,612	0,319
3	17:47:09,586	17:47:10,403	0,817
4	17:47:13,696	17:47:14,402	0,706
5	17:47:17,514	17:47:18,418	0,904
6	17:47:22,628	17:47:23,402	0,774
7	17:47:26,789	17:47:27,406	0,617

8	17:47:30,784	17:47:31,401	0,617
9	17:47:34,803	17:47:35,361	0,558
10	17:47:39,145	17:47:40,402	1,257
11	17:47:43,064	17:47:44,399	1,335
12	17:47:46,926	17:47:48,406	1,480
13	17:47:51,757	17:47:52,407	0,650
14	17:47:56,111	17:47:56,401	0,290
15	17:48:01,224	17:48:02,401	1,177
16	17:48:06,512	17:48:07,407	0,895
17	17:48:09,700	17:48:10,405	0,705
18	17:48:15,254	17:48:16,399	1,145
19	17:48:20,414	17:48:21,405	0,991
20	17:48:27,058	17:48:28,413	1,355
Rata-rata durasi pengiriman pesan			0,855



Gambar 4.56: Grafik hasil pengujian broker server (IP adress = ee.unsoed.ac.id)



Gambar 4.57: Grafik perbandingan durasi rata-rata pengiriman pesan

Berdasarkan hasil pengujian yang dilakukan dengan mencatat waktu pesan dikirim dan pesan diterima kemudian mencari durasi pengiriman pesan dengan mengurangi antara waktu pesan dikirim dengan waktu pesan diterima dimana dapat dilihat pada tabel bahwa komunikasi MQTT yang menggunakan broker lokal pada Raspberry Pi memiliki rata-rata durasi pengiriman pesan terkecil yaitu 0,645 detik sedangkan komunikasi MQTT yang menggunakan broker server cenderung memiliki durasi pengiriman pesan yang lebih lama yaitu 0,860 detik untuk server BRIN dan 0,855 detik untuk server Unsoed. Dari hasil ini dapat diambil kesimpulan bahwa penggunaan broker lokal pada Raspberry Pi memiliki latensi perngiriman data yang lebih kecil dibandingkan dengan menggunakan broker server jika jarak antara Raspberry Pi dengan alat tidak terlalu jauh.

4.4.2 Pengujian Keandalan HMI

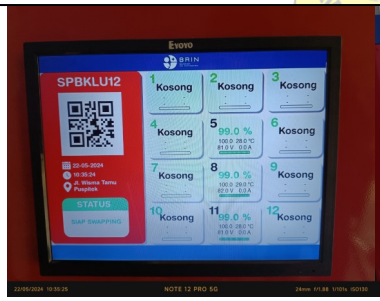
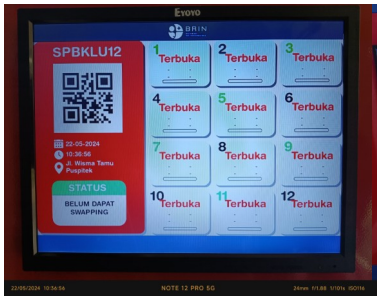
Pengujian HMI dilakukan dengan membandingkan tampilan pada HMI dengan kondisi sebenarnya yang terjadi pada SPBKLU. Pengujian akan dilakukan dalam 2 jenis pengujian. Pengujian pertama dilakukan untuk mengetahui

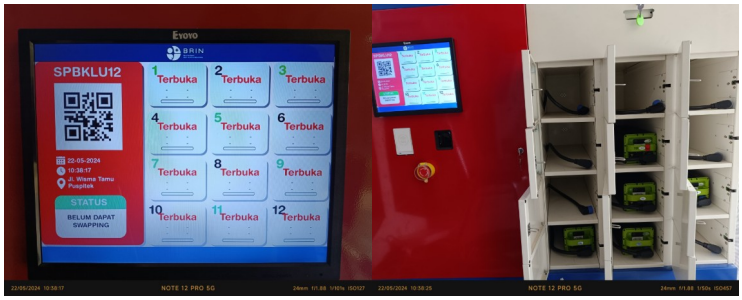
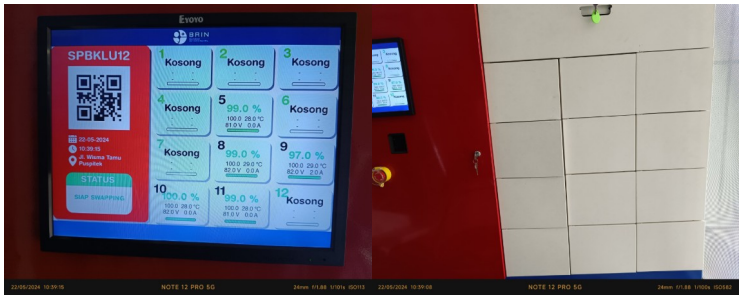
keandalan HMI terhadap perubahan yang terjadi pada loker dan pengujian kedua dilakukan untuk mengetahui keandalan waktu dan kode QR pada HMI.

Pengujian Pertama

Pengujian pertama dilakukan untuk mengetahui keandalan HMI terhadap kondisi yang terjadi pada loker SPBKLU. Pengujian akan dilakukan dengan mengamati perubahan yang terjadi pada HMI ketika kondisi loker diubah. Pengujian akan dimulai dengan mengamati kondisi HMI dan loker dimana setiap loker dalam kondisi tertutup, kemudian setiap loker dibuka, beberapa loker dimasukkan baterai, dan loker ditutup. Untuk data perngujian dapat dilihat pada tabel di bawah.

Tabel 4.13: Pengujian Pertama Keandalan HMI

No	Tampilan HMI	Tampilan Loker Ketika Baterai Dimasukkan	Keterangan
1			• Semua loker dalam kondisi tertutup • Loker dengan baterai : 5, 6, 11
2			• Pintu semua loker dibuka • Loker dengan baterai : 5, 6, 11

3		• Pintu semua loker dibuka • Penambahan baterai • Loker dengan baterai : 5, 6, 9, 10, 11
4		• Semua pintu loker ditutup • Loker dengan baterai : 5, 6, 9, 10, 11

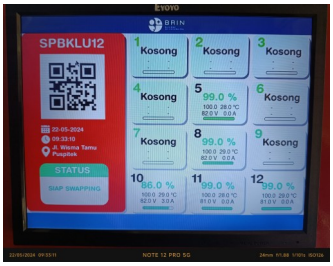
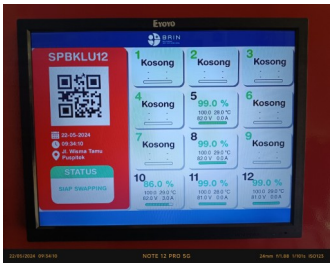
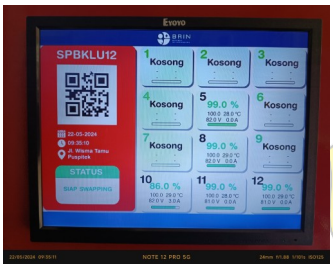
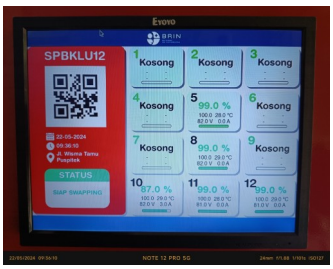
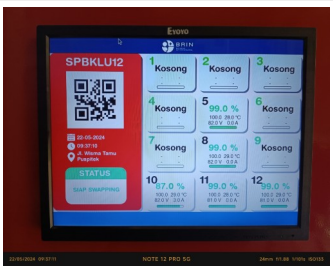
Berdasarkan respon dari HMI terhadap perubahan kondisi yang terjadi pada loker SPBKLU dimana HMI dapat memberikan tampilan yang akurat mengenai keadaan dari setiap loker dapat disimpulkan bahwa HMI memiliki keandalan respon yang baik terhadap kondisi loker.

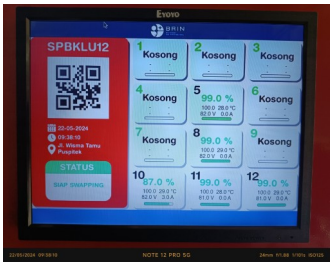
Pengujian Kedua

Pengujian kedua dilakukan untuk mengetahui keandalan HMI ketika terjadi perubahan kode QR dan waktu pada HMI berdasarkan waktu *realtime* dengan menjadikan waktu pada file foto sebagai patokannya. Pengujian akan dilakukan dengan cara mengambil gambar tampilan HMI setiap 1 menit kemudian membandingkannya dengan waktu *realtime* sebanyak 6 kali.

Tabel 4.14: Pengujian Kedua Keandalan HMI

Tampilan HMI & Waktu Dari HP	Waktu *Realtime*	Waktu HMI	Kode QR	Keterangan

	22-05-2024 09:33:10	22-05-2024 09:33:11	spbklu10 _xDJQ7J	Terdapat perbedaan waktu 1 detik
	22-05-2024 09:34:10	22-05-2024 09:34:10	spbklu10 _xDJQ7J	
	22-05-2024 09:35:10	22-05-2024 09:35:11	spbklu10 _xDJQ7J	Terdapat perbedaan waktu 1 detik
	22-05-2024 09:36:10	22-05-2024 09:36:10	spbklu10 _6M5ila	Kode QR berubah
	22-05-2024 09:37:10	22-05-2024 09:37:11	spbklu10 _6M5ila	Terdapat perbedaan waktu 1 detik

	22-05-2024 09:38:10	22-05-2024 09:38:10	spbklu10 _6M5ila	

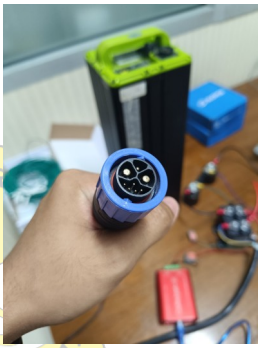
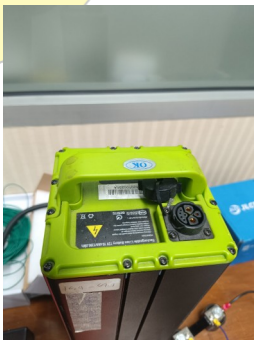
Berdasarkan pengujian yang dilakukan dengan mengambil data setiap 1 menit sebanyak 6 kali terdapat beberapa saat ketika terjadi perbedaan waktu sebanyak 1 detik antara HMI dengan waktu *realtime* dari HP dan juga 1 kali perubahan kode QR. Perbedaan 1 detik terjadi pada menit ke 1, 3, dan 5 dan kode QR berubah pada menit ke-4. Perbedaan waktu antara HMI dan waktu *realtime* dapat terjadi karena adanya delay ketika program dijalankan sehingga terjadi delay persekian detik dari waktu *realtime* yang menyebabkan meskipun waktu dari HMI menunjukan waktu pada detik 10 waktu *realtime* menunjukan detik 11. Adapun untuk kode QR akan berubah setiap beberapa menit, hal ini sesuai dengan data pengujian ketika pada menit ke-4 kode QR berubah dan memiliki data yang berbeda. Dengan hasil pengujian ini, dapat disimpulkan bahwa HMI memiliki keandalan yang baik terhadap perubahan waktu termasuk kode QR di dalamnya.

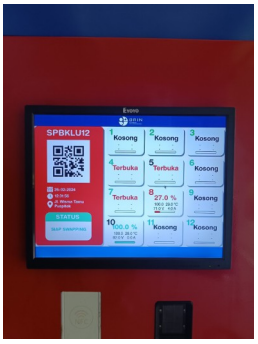
4.4.3 Pengujian Pembacaan Tegangan Baterai Pada SPBKLU

Sebelumnya sudah dilakukan pengujian keandalan HMI terhadap perubahan yang terjadi pada SPBKLU. Namun, selain menguji respon HMI perlu juga dilakukan pengujian keandalan nilai yang ditampilkan HMI terhadap nilai sebenarnya. Pada pengujian ini nilai tegangan yang ditampilkan pada HMI akan dibandingkan dengan nilai tegangan yang diambil menggunakan multimeter. Nilai tegangan dipilih karena pengukuran arus ataupun daya secara langsung tidak

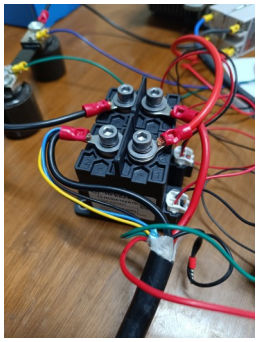
memungkinkan. Adapun untuk tampilan nilai tegangan pada HMI diambil dari baterai secara langsung dengan memanfaatkan BMS pada baterai. Pengujian akan dilakukan dengan membandingkan nilai tegangan pada HMI dengan multimeter. Adapun untuk metode pengukurannya adalah sebagai berikut.

Tabel 4.15: Metode Pengukuran Pada Pengujian Keandalan Tegangan SPBKLU

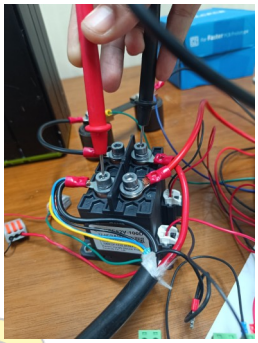
Tahapan Pengukuran			
BMS		Multimeter	
Foto	Dekstripsi	Foto	Dekstripsi
	Socker baterai pada SPBKLU		Socket baterai untuk pengujian
	Socket baterai GESITS		Socket baterai GESITS
	Baterai disambungkan ke socket pada SPBKLU		Baterai disambungkan ke socket



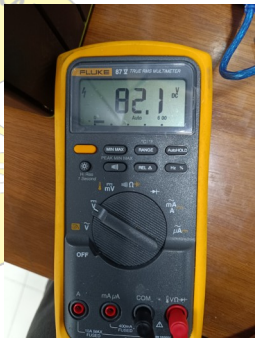
SPBKLU membaca informasi baterai dari BMS dan menampilkannya pada HMI



Relay yang terhubung ke socket



Pengukuran dilakukan dengan menempatkan multimeter pada pin + dan - relay



Hasil pengukuran pada multimeter

Pengujian akan dilakukan menggunakan 2 baterai dengan cara menempatkan baterai tersebut pada loker tertentu untuk diambil nilai pembacaan pada HMI kemudian barulah baterai akan dilihat nilai tegangannya menggunakan multimeter. Percobaan akan dilakukan sebanyak 10 kali untuk setiap loker dengan mengecualikan loker 4 dan 7 karena mengalami rusak pada bagian selonoid pintunya.

Tabel 4.16: Hasil Pengujian Keandalan Pembacaan Tegangan SPBKLU

Pengujian Pembacaan Tegangan (V)						
Loker	Baterai 1			Baterai 2		
	BMS	Multimeter	Selisih	BMS	Multimeter	Selisih

			Pembacaan			Pembacaan
1	72	71.9	0.1	82	82.1	0.1
2	72	71.9	0.1	82	82.1	0.1
3	72	71.9	0.1	82	82.1	0.1
4	72	71.9	0.1	82	82.1	0.1
5	72	71.8	0.2	82	82.1	0.1
6	72	71.8	0.2	82	82.1	0.1
7	71	71.1	0.1	82	81.8	0.2
8	71	71.1	0.1	82	81.8	0.2
9	71	71.1	0.1	82	81.8	0.2
10	71	71.0	0.0	82	81.7	0.3
11	71	71.0	0.0	82	81.7	0.3
12	71	71.0	0.0	82	81.7	0.3

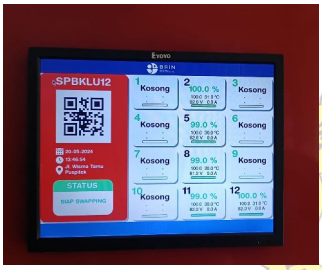
Berdasarkan hasil pengujian yang dilakukan terdapat perbedaan nilai dalam selisih antara nilai pada HMI dan paling besar yaitu 0,3 V. Hal ini dapat terjadi karena nilai yang ditampilkan oleh HMI merupakan bilangan desimal yang tidak memiliki angka dibelakang koma sehingga apabila kita bulatkan nilai pembacaan multimeter ke bilangan desimal terdekat, maka hasilnya akan sama. Oleh karena itu, berdasarkan data pengujian yang didapatkan, nilai tegangan pada HMI dapat dinyatakan akurat jika dibandingkan dengan pembacaan tegangan dengan multimeter.

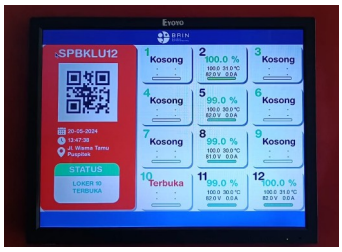
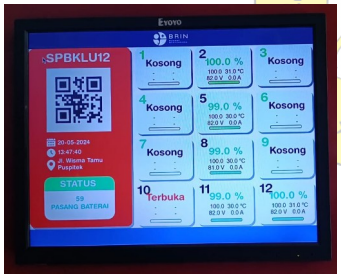
4.4.4 Pengujian *Battery Swapping*

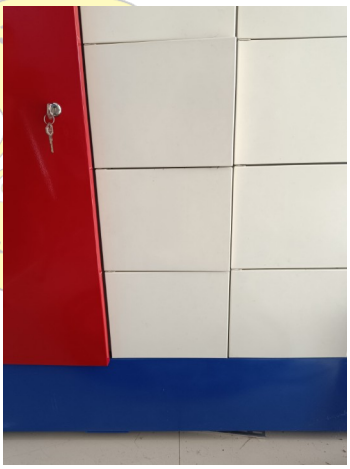
Tujuan utama dari SPBKLU adalah memudahkan pengguna melakukan transaksi dengan menukarkan baterainya dengan baterai yang memiliki SoC tinggi. Untuk itu, pengujian *battery swapping* secara langsung diperlukan untuk mengetahui keandalan SPBKLU dalam melakukan transaksi penukaran baterai. Transaksi akan dimulai dari SPBKLU yang berada dalam kondisi siap melakukan

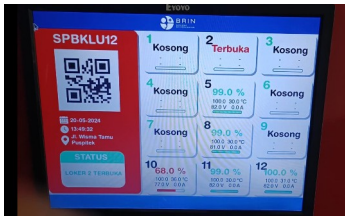
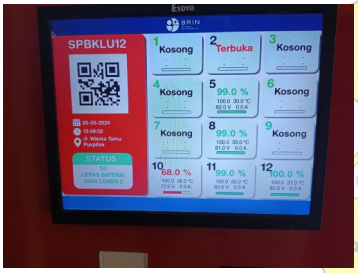
transaksi, *backend* memberikan perintah untuk memulai transaksi, dan transaksi dimulai. Adapun untuk *backend* akan disimulasikan menggunakan MQTTX yang bertugas untuk mengirimkan pesan permintaan untuk melakukan transaksi .Bukti pengujian berupa video dan foto. Berikut adalah data hasil pengujian yang dilakukan.

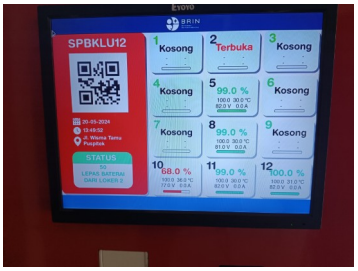
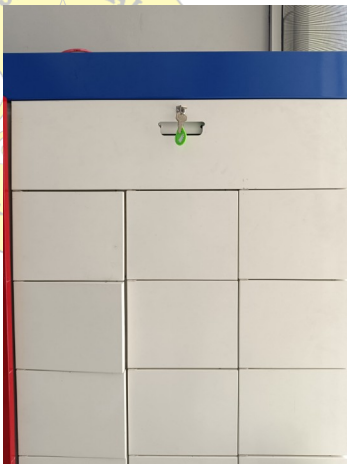
Tabel 4.17: Pengujian Transaksi *Battery Swapping*

<table>
<tr><td colspan="4">Pengujian Transaksi Battery Swapping</td></tr>
<tr><td rowspan="2">No</td><td colspan="2">Foto</td><td rowspan="2">Keterangan</td></tr>
<tr><td>HMI/MQTTX</td><td>Loker</td></tr>
<tr><td>1</td><td> 
</td><td>  </td><td>Status SPBKLU = 2 atau siap melakukan transaksi karena SPBKLU memiliki lebih dari 1 loker kosong dan lebih dari 1 baterai dengan SoC di atas 80</td></tr>
<tr><td>2</td><td>Topic: spbklu/spbklu10/send/startTransaction QoS: 0
<pre>{
  "data": {
    "transaction_id":
"JhTus20240515154851636362254",
    "user_id": 10,
    "user_email": "ahmad@gmail.com",
    "user_name": "ahmad taqiuddin"
  },
  "message": "Start Transaction
Berhasil",
  "status": 200
}</pre></td><td></td><td>MQTTX mensimulasikan backend dengan mengirim pesan permintaan melakukan transaksi ke SPBKLU</td></tr>
<tr><td>3</td><td>Topic: spbklu/updateStatusBss QoS: 0
<pre>{
  "topic": "spbklu/updateStatusBss",
  "spbkluid": "spbklu",
  "status": 2
}</pre></td><td></td><td>Pesan balasan dari SPBKLU ke MQTTX sebagai penanda bahwa transaksi akan dilakukan</td></tr>
</table>

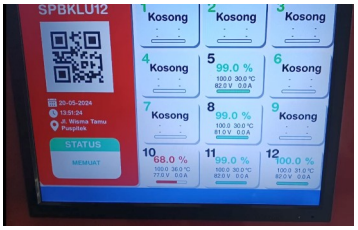
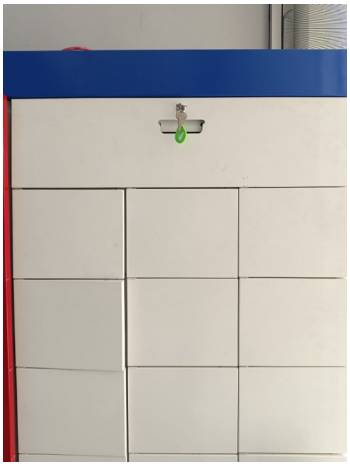
4			**Tahap 1** **Kondisi:** SPBKLU memulai transaksi *battery swapping* **Tindakan:** Sistem memerintahakan loker kosong yaitu loker 10 untuk membuka pintunya **Syarat lanjut:** SPBKLU mendeteksi pintu loker 10 telah terbuka berdasarkan sensor solenoid pada loker
5			**Tahap 2** **Kondisi:** Pintu loker telah terbuka **Tindakan:** Pengguna menghubungkan baterainya ke socket SPBKLU **Syarat lanjut:** Baterai pengguna telah terhubung ke socket SPBKLU

6			**Tahap 3** **Kondisi:** Baterai pengguna telah terhubung dengan socket SPBKLU **Tindakan:** Pengguna menutup pintu loker 10 **Syarat lanjut:** SPBKLU mendeteksi lewat solenoid pada loker bahwa pintu telah tertutup
7			**Tahap 4** **Kondisi:** Pintu loker 10 telah tertutup **Tindakan:** SPBKLU memeriksa ID dan SoH baterai **Syarat lanjut:** - SoH baterai lebih dari 60 - ID baterai terdaftar

8			**Tahap 5** **Kondisi:** ID dan SoH bateraitelah dideteksi dan memenuhi persyaratan **Tindakan:** Sistem membuka loker dengan SoC baterai tertinggi yaitu loker 2 **Syarat lanjut:** Pintu loker 2 telah terbuka
9			**Tahap 6** **Kondisi:** Pintu loker 2 telah terbuka **Tindakan:** Pengguna melepas baterai dari socket SPBKLU **Syarat lanjut:** SPBKLU mendeteksi baterai telah terlepas dari socket berdasarkan data baterai

10			**Tahap 7** **Kondisi:** baterai telah dilepas dari socket **Tindakan:** Pengguna mengeluarkan baterai dan menutup loker **Syarat lanjut:** SPBKLU mendeteksi pintu telah tertutup berdasarkan data sensor solenoid pada loker
11			**Tahap 8** **Kondisi:** Pintu loker 2 telah tertutup **Tindakan:** SPBKLU mengirimkan pesan ke *backend*

12	Topic: spbklu/updateStatusBss QoS: 0
	
	{
	"topic": "spbklu/updateStatusBss",
	"spbkluid": "spbklu10",
	"status": 1
	}
			Pesan dari SPBKLU sebagai penanda bahwa transaksi telah selesai

13			Menyelesaikan transaksi

Berdasarkan dengan pengujian yang telah dilakukan seperti tercantum dalam tabel di atas dapat disimpulkan bahwa SPBKLU dapat melakukan tahapan transaksi *battery swapping* dengan baik sesuai dengan urutan yang ditetapkan. Selain itu, tampilan pada HMI menunjukan pesan dan data yang sesuai dengan tahapan transaksi *battery swapping* juga terhadap kondisi lokernya yang menunjukan bahwa HMI tersinkronisasi dengan baik terhadap subkontrol yang menandakan bahwa komunikasi data berjalan dengan baik.

4.4.5 Pengujian *Eror Handling*

Eror handling merupakan variebel yang penting untuk membuat perangkat SPBKLU dapat mengatasi atau setidaknya menanggapi situasi khusus yang terjadi. Untuk itu diperlukan pengujian langsung pada perangkat untuk memastikan bahwa sistem *eror handling* yang diterapkan pada SPBKLU dapat berjalan dengan semestinya. Terdapat 3 *eror handling* yang akan diuji yaitu ketika baterai pengguna memiliki SoH atau ID yang tidak memenuhi persyaratan saat

melakukan transaksi, transaksi melebihi batas waktu, dan pintu loker yang terbuka di luar kondisi transaksi.

Baterai yang Ditukar Memiliki SoH *(Source of Health)* di Bawah standar atau ID Baterai Tidak Terdaftar di Sistem

Pengujian *eror handling* untuk kondisi baterai pengguna memiliki SoH di bawah 60 atau ID yang tidak terdaftar sehingga tidak memenuhi persyaratan untuk melakukan transaksi, dilakukan mirip seperti pengujian transaksi *battery swapping*. Namun, baterai yang digunakan memiliki ID yang tidak terdaftar pada sistem. Berikut adalah ID baterai yang terdaftar pada sistem ketika pengujian ini dilakukan.

```
batteryID           = ["00859023","00859001","00859002","00859007","00859009"]
```

Gambar 4.58: ID baterai yang terdaftar pada sistem kontrol utama

Sedangkan jika dilihat dari *data log system* dengan cara mencari data yang berasal dari loker 10 pada tanggal 22-05-2024 sekitar jam 13:05:54 seperti yang terlihat pada foto percobaan, baterai yang digunakan untuk percobaan memiliki ID 00859019.

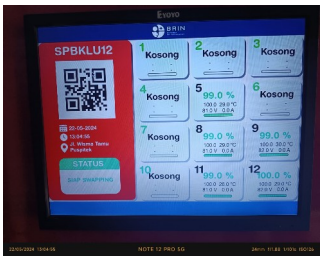
```
[2024-05-22 13:05:16.632604] [LOCKER 10   ] ["state_door":false]
[2024-05-22 13:05:45.473013] [LOCKER 10   ] ["state_plug":true,                "state_locker":4]
[2024-05-22 13:05:47.534967] [LOCKER 10   ] ["batVolt":77.00,              "batTemp_cell1":42.00,              "batID":"00859019"]
[2024-05-22 13:05:49.599702] [LOCKER 10   ] ["state_locker":2,              "batSOC":68.00,              "batSOH":100.00,
```

Gambar 4.59: Data log system loker 10 pada tanggal 22-05-2024 sekitar jam 13:05:54

Ketika baterai pengguna yang memiliki ID tidak terdaftar pada sistem digunakan untuk melakukan transaksi *battery swapping* akan memicu sistem *eror handling* bekerja. Berikut adalah hasil pengujian *eror handling* untuk kasus ini.

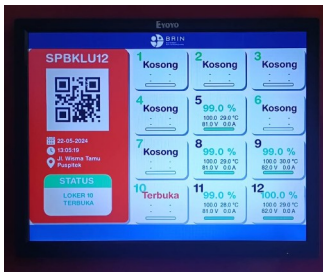
Tabel 4.18: Data pengujian eror handling (SoC atau ID baterai pengguna tidak memenuhi syarat)

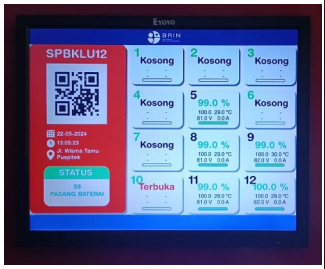
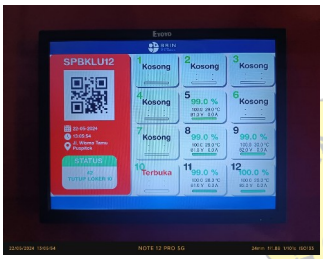
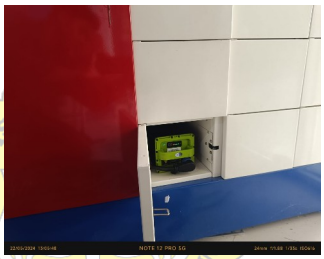
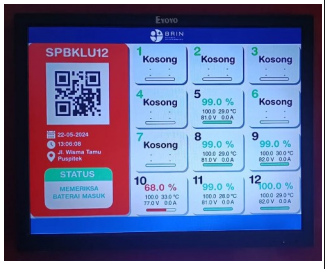
Pengujian Eror Handling (SoC atau ID Baterai Pengguna Tidak Memenuhi Syarat)

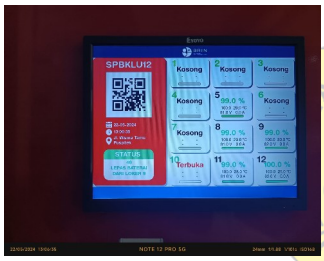
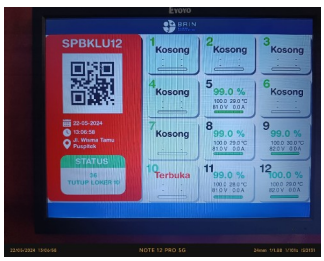
No	Foto		Keterangan
	HMI/MQTTX	Loker	
1			Status SPBKLU = 2 atau siap melakukan transaksi karena SPBKLU memiliki lebih dari 1 loker kosong dan lebih dari 1 baterai dengan SoC di atas 80

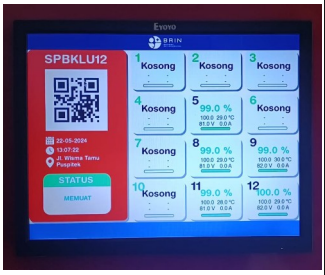
2	Topic: spbklu/spbklu10/send/startTransaction QoS: 0
	{
	"data": {
	"transaction_id": "JhTus20240515154851636362254",
	"user_id": 10,
	"user_email": "ahmad@gmail.com",
	"user_name": "ahmad taqiuddin"
	},
	"message": "Start Transaction Berhasil",
	"status": 200
	}
			MQTTX mensimulasikan *backend* dengan mengirim pesan permintaan melakukan transaksi ke SPBKLU

3	Topic: spbklu/updateStatusBss QoS: 0
	{
	"topic": "spbklu/updateStatusBss",
	"spbkluid": "spbklu",
	"status": 2
	}
			Pesan balasan dari SPBKLU ke MQTTX sebagai penanda bahwa transaksi akan dilakukan

4			**Tahap 1** **Kondisi:** SPBKLU memulai transaksi *battery swapping* **Tindakan:** Sistem memerintahakan loker kosong yaitu loker 10 untuk membuka pintunya **Syarat lanjut:** SPBKLU mendeteksi pintu loker 10 telah terbuka berdasarkan sensor solenoid pada loker

5			**Tahap 2** **Kondisi:** Pintu loker telah terbuka **Tindakan:** Pengguna menghubungkan baterainya ke socket SPBKLU **Syarat lanjut:** Baterai pengguna telah terhubung ke socket SPBKLU
6			**Tahap 3** **Kondisi:** Baterai pengguna telah terhubung dengan socket SPBKLU **Tindakan:** Pengguna menutup pintu loker 10 **Syarat lanjut:** SPBKLU mendeteksi lewat solenoid pada loker bahwa pintu telah tertutup
7			**Tahap 4** **Kondisi:** Pintu loker 10 telah tertutup **Tindakan:** SPBKLU memeriksa ID dan SoH baterai **Syarat lanjut:** - SoH baterai lebih dari 60

			- ID baterai terbaca dan terdaftar
8			**Tahap 5** **Kondisi:** - ID dan SoH baterai telah dideteksi - ID baterai tidak terbaca **Tindakan:** Sistem membuka loker 10 kembali **Syarat lanjut:** Pintu loker 10 telah terbuka
9			**Tahap 6** **Kondisi:** Pintu loker 10 telah terbuka **Tindakan:** Pengguna melepas baterai dari socket SPBKLU **Syarat lanjut:** SPBKLU mendeteksi baterai telah terlepas dari socket berdasarkan data baterai
10			**Tahap 7** **Kondisi:** baterai telah dilepas dari socket **Tindakan:** Pengguna mengeluarkan baterai dan menutup loker

			Syarat lanjut: SPBKLU mendeteksi pintu telah tertutup berdasarkan data sensor solenoid pada loker
11			**Tahap 8** **Kondisi:** Pintu loker 10 telah tertutup **Tindakan:** SPBKLU mengirimkan pesan ke *backend*
12			Pesan dari SPBKLU sebagai penanda bahwa transaksi telah selesai

Berdasarkan pengujian yang dilakukan dimana ketika baterai pengguna yang ID nya tidak terdaftar dihubungkan dengan socket kemudian pintu ditutup membuat sistem *eror handling* bekerja. Hal itu dapat dilihat dari SPBKLU yang membuka kembali loker 10 agar pengguna dapat mengambil baterainya kembali. Dari percobaan ini, dapat disimpulkan bahwa sistem *eror handling* untuk situasi khusus ini dapat berjalan dengan baik pada SPBKLU secara langsung.

Pengguna Melakukan *Battery Swap* Melebihi Waktu yang Ditentukan

Pengujian ini dilakukan untuk mengetahui apakah tindakan *eror handling* ketika pengguna melakukan transaksi *battery swap* melebihi waktu yang ditentukan dimana seharusnya ketika dihadapkan pada situasi ini SPKLU seharusnya akan menyalakan alarm dan membatalkan transaksi yang sedang berlangsung. Pengujian akan dimulai dengan pesan permintaan transaksi

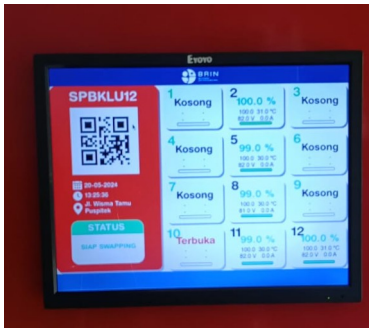
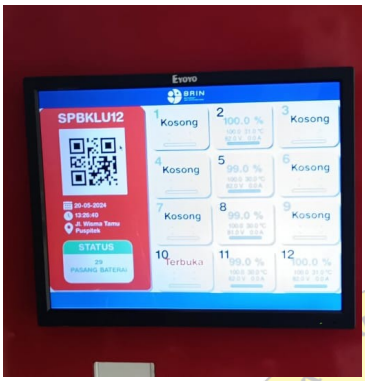
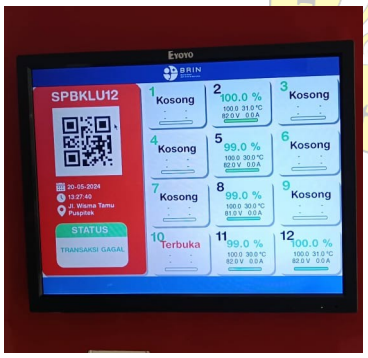
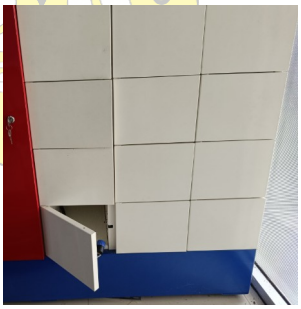
dikirimkan, transaksi dimulai, kemudian transaksi akan didiamkan hingga waktu hitung mundur habis. Berikut adalah data hasil pengujiannya :

Tabel 4.19: Data pengujian *e*ror *handling* (*battery swap m*elebihi waktu yang ditentukan)

Pengujian *E*ror Handling (*Battery Swap* Melebihi Waktu yang Ditentukan)				
No	Foto		Alarm	Keterangan
	HMI	Loker		
1			Off	Status SPBKLU = 2 atau siap melakukan transaksi karena SPBKLU memiliki lebih dari 1 loker kosong dan lebih dari 1 baterai dengan SoC di atas 80

2	Topic: spbklu/spbklu10/send/startTransaction QoS: 0
	{
	"data": {
	"transaction_id": "JhTus202405151548516336225 4",
	"user_id": 10,
	"user_email": "ahmad@gmail.com",
	"user_name": "ahmad taqiuddin"
	},
	"message": "Start Transaction Berhasil",
	"status": 200
	}
			Off	MQTTX mensimulasikan *backend* dengan mengirim pesan permintaan melakukan transaksi ke SPBKLU

3	Topic: spbklu/updateStatusBss QoS: 0
	{
	"topic": "spbklu/updateStatusBss",
	"spbkluid": "spbklu",
	"status": 2
	}
			Off	Pesan balasan dari SPBKLU ke MQTTX sebagai penanda bahwa transaksi akan dilakukan

4			Off	Kondisi: Pintu loker terbuka Tindakan: Perhitungan mundur dimulai
5			On	Kondisi: Waktu penukaran baterai lebih dari 30 detik Tindakan: Alarm berbunyi
6			Off	Kondisi: Waktu penukaran baterai lebih dari 60 detik Tindakan: Transaksi dinyatakan gagal dan alarm dimatikan
7			Off	Pesan dari SPBKLU sebagai penanda bahwa transaksi telah selesai

Berdasarkan pengujian ketika waktu transaksi sudah lebih dari 30 detik maka alarm akan berbunyi, alarm akan mati ketika waktu transaksi sudah

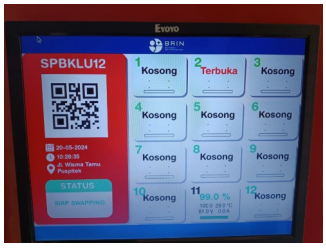
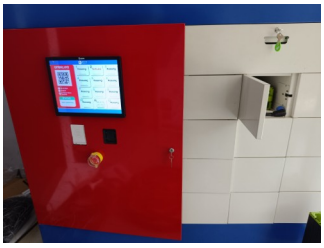
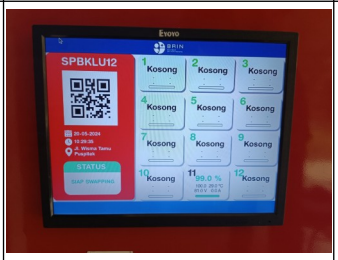
melebihi 1 menit. Ketika waktu melebihi 1 menit transaksi akan dibatalkan dan dianggap gagal. Berdasarkan hal tersebut, sistem *eror handling* untuk situasi dimana transaksi *battery swap m*elebihi waktu yang ditentukan dapat dikatakan dapat berjalan dengan baik.

Pintu Salah Satu Loker Baterai Terbuka di Luar Proses Transaksi

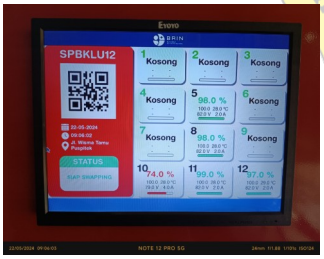
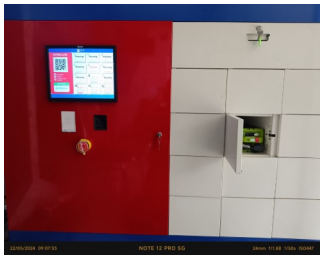
Pengujian ini dilakukan untuk mengetahui apakah tindakan *eror handling* ketika transaksi *battery swapping* dapat berjalan pada SPBKLU dengan baik dimana seharusnya ketika pintu loker terbuka ketika SPBKLU dalam kondisi non transaksi maka alarm akan menyala. Pengujian ini akan dilakukan dengan membuka pintu loker tertentu ketika SPBKLU dalam kondisi non transaksi. Pengujian akan dilakukan sebanyak 2 kali, pada pengujian pertama pintu loker yang terbuka akan ditutup kembali, sedangkan pada pengujian kedua pintu loker akan dibiarkan terbuka. Berikut adalah data hasil pengujian yang dilakukan :

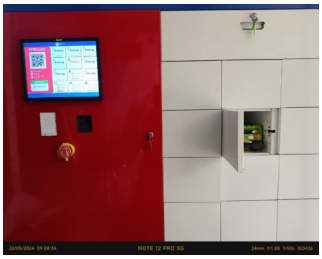
Tabel 4.20: Data pengujian pertama *eror handling* (pintu terbuka di luar kondisi transaksi)

Pengujian Pertama *Eror Handling* (Pintu Terbuka di Luar Kondisi Transaksi)				
No	Foto			Keterangan
	HMI	Loker	Alarm	
1			Off	SPBKLU dalam kondisi normal (tidak melakukan transaksi)

2			On	Pintu loker 2 terbuka ketika SPBKLU tidak melakukan transaksi sehingga alarm menyala
3			Off	Pintu loker ditutup kembali sehingga alarm mati

Tabel 4.21: Data pengujian kedua *eror handling* (pintu terbuka di luar kondisi transaksi)

Pengujian Kedua *Eror Handling* (Pintu Terbuka di Luar Kondisi Transaksi)				
No	Foto			Keterangan
	HMI	Loker	Alarm	
1			Off	SPBKLU dalam kondisi normal (tidak melakukan transaksi)
2			On	Pintu loker 2 terbuka ketika SPBKLU tidak melakukan transaksi sehingga alarm menyala

3			Off	Pintu loker terbuka selama lebih dari 30 detik sehingga alarm dimatikan

Berdasarkan dengan pengujian yang dilakukan ketika pintu loker tertentu yaitu loker 5 terbuka saat perangkat SPBKLU tidak melakukan transaksi *battery swapping* maka alarm akan menyala dan akan berhenti ketika pintu loker ditutup atau alarm telah berbunyi selama 30 detik. Dari pengujian ini dapat diambil kesimpulan bahwa sistem *eror handling* pada situasi pintu loker terbuka di luar kondisi transaksi dapat berjalan dengan baik.

4.4.6 Pengujian *Log Data*

Pengujian *log data system*

Pengujian log data system dilakukan dengan mengamati data log yang ditulis dalam rentang waktu tertentu ketika SPBKLU sedang diuji. Data log system diharuskan mampu untuk menyimpan informasi terkait dengan kinerja SPBKLU untuk itu pengujian keandalan data log system harus mencakup data dari seluruh loker. Oleh karena itu, pengujian keandalan log data akan dilakukan dengan membandingkan HMI, loker, dan juga data log system yang tercatat. Berikut adalah hasil pengujian sebelumnya yaitu pengujian pertama untuk keandalan HMI yang dibagi dalam 4 tahap.

Gambar 4.60: Hasil pengujian keandalan HMI tahap 1

Tabel 4.22: Hasil Pengujian Keandalan HMI Tahap 1

Waktu	Nama Loker	Status Pintu Loker	Status Baterai
10:35:24	1	OFF	-
10:35:24	2	OFF	-
10:35:24	3	OFF	-
10:35:24	4	OFF	-
10:35:24	5	OFF	SoC = 99 SoH = 100 Temp = 28 C Tegangan = 82 V Arus = 0 A
10:35:24	6	OFF	-
10:35:24	7	OFF	-
10:35:24	8	OFF	SoC = 99 SoH = 100 Temp = 29 C Tegangan = 82 V Arus = 0 A
10:35:24	9	OFF	-
10:35:24	10	OFF	-
10:35:24	11	OFF	SoC = 99 SoH = 100 Temp = 28 C Tegangan = 81 V Arus = 0 A
10:35:24	12	OFF	-

Tabel 4.23: Hasil *log data system* tahap 1

Log System Pada Tahap 1

[2024-05-22 10:33:16.038996] [MQTT] [Connecting...]
[2024-05-22 10:33:16.052185] [MQTT] [Connected to 192.168.0.157 rc=0]
[2024-05-22 10:33:16.061133] [MQTT] [Subscribed to -> spbklu10/#]
[2024-05-22 10:33:16.170832] [LOCKER 12] ["state_plug":false,
"state_door":true, "state_charging":false, "state_locker":3,
"batSOC":0.00, "batSOH":0.00, "batVolt":0.00,
"batCurr":0.00, "batTemp_cell1":0.00,
"batTemp_cell2":0.00, "batTemp_cell3":0.00, "batID":"",
"batCycle":0, "sensor_flame":false, "sensor_smoke":false,
"sensor_tstat":false, "state_emg":false, "ctrl_chg":false,
"ctrl_buzzer":false, "ctrl_fan":false, "ctrl_aerosol":false
]
[2024-05-22 10:33:16.436464] [LOCKER 5] ["state_plug":true,
"state_door":true, "state_charging":false, "state_locker":1,
"batSOC":99.00, "batSOH":100.00, "batVolt":81.00,
"batCurr":0.00, "batTemp_cell1":28.00,
"batTemp_cell2":0.00, "batTemp_cell3":0.00,
"batID":"00859023", "batCycle":4, "sensor_flame":false,
"sensor_smoke":false, "sensor_tstat":false,
"state_emg":false, "ctrl_chg":false, "ctrl_buzzer":false,
"ctrl_fan":false, "ctrl_aerosol":false]
[2024-05-22 10:33:17.067178] [LOCKER 8] ["state_plug":true,
"state_door":true, "state_charging":false, "state_locker":1,
"batSOC":99.00, "batSOH":100.00, "batVolt":82.00,
"batCurr":0.00, "batTemp_cell1":29.00,
"batTemp_cell2":0.00, "batTemp_cell3":0.00,
"batID":"00243001", "batCycle":18, "sensor_flame":false,
"sensor_smoke":false, "sensor_tstat":false,
"state_emg":false, "ctrl_chg":false, "ctrl_buzzer":false,
"ctrl_fan":false, "ctrl_aerosol":false]
[2024-05-22 10:33:17.090265] [LOCKER 11] ["state_plug":true,
"state_door":true, "state_charging":false, "state_locker":1,
"batSOC":99.00, "batSOH":100.00, "batVolt":81.00,
"batCurr":0.00, "batTemp_cell1":28.00,
"batTemp_cell2":0.00, "batTemp_cell3":0.00,
"batID":"00859002", "batCycle":11, "sensor_flame":false,
"sensor_smoke":false, "sensor_tstat":false,
"state_emg":false, "ctrl_chg":false, "ctrl_buzzer":false,
"ctrl_fan":false, "ctrl_aerosol":false]
[2024-05-22 10:33:17.420633] [LOCKER 2] ["state_plug":false,
"state_door":true, "state_charging":false, "state_locker":3,
"batSOC":0.00, "batSOH":0.00, "batVolt":0.00,
"batCurr":0.00, "batTemp_cell1":0.00,
"batTemp_cell2":0.00, "batTemp_cell3":0.00, "batID":"",
"batCycle":0, "sensor_flame":false, "sensor_smoke":false,

"sensor_tstat":false, "state_emg":false, "ctrl_chg":false,
"ctrl_buzzer":false, "ctrl_fan":false, "ctrl_aerosol":false
]
[2024-05-22 10:33:17.460821] [LOCKER 7] ["state_plug":false,
"state_door":true, "state_charging":false, "state_locker":3,
"batSOC":0.00, "batSOH":0.00, "batVolt":0.00,
"batCurr":0.00, "batTemp_cell1":0.00,
"batTemp_cell2":0.00, "batTemp_cell3":0.00, "batID":"",
"batCycle":0, "sensor_flame":false, "sensor_smoke":false,
"sensor_tstat":false, "state_emg":false, "ctrl_chg":false,
"ctrl_buzzer":false, "ctrl_fan":false, "ctrl_aerosol":false
]
[2024-05-22 10:33:17.475423] [LOCKER 3] ["state_plug":false,
"state_door":true, "state_charging":false, "state_locker":3,
"batSOC":0.00, "batSOH":0.00, "batVolt":0.00,
"batCurr":0.00, "batTemp_cell1":0.00,
"batTemp_cell2":0.00, "batTemp_cell3":0.00, "batID":"",
"batCycle":0, "sensor_flame":false, "sensor_smoke":false,
"sensor_tstat":false, "state_emg":false, "ctrl_chg":false,
"ctrl_buzzer":false, "ctrl_fan":false, "ctrl_aerosol":false
]
[2024-05-22 10:33:17.498696] [LOCKER 6] ["state_plug":false,
"state_door":true, "state_charging":false, "state_locker":3,
"batSOC":0.00, "batSOH":0.00, "batVolt":0.00,
"batCurr":0.00, "batTemp_cell1":0.00,
"batTemp_cell2":0.00, "batTemp_cell3":0.00, "batID":"",
"batCycle":0, "sensor_flame":false, "sensor_smoke":false,
"sensor_tstat":false, "state_emg":false, "ctrl_chg":false,
"ctrl_buzzer":false, "ctrl_fan":false, "ctrl_aerosol":false
]
[2024-05-22 10:33:17.531282] [LOCKER 9] ["state_plug":false,
"state_door":true, "state_charging":false, "state_locker":3,
"batSOC":0.00, "batSOH":0.00, "batVolt":0.00,
"batCurr":0.00, "batTemp_cell1":0.00,
"batTemp_cell2":0.00, "batTemp_cell3":0.00, "batID":"",
"batCycle":0, "sensor_flame":false, "sensor_smoke":false,
"sensor_tstat":false, "state_emg":false, "ctrl_chg":false,
"ctrl_buzzer":false, "ctrl_fan":false, "ctrl_aerosol":false
]
[2024-05-22 10:33:17.618777] [LOCKER 1] ["state_plug":false,
"state_door":true, "state_charging":false, "state_locker":3,
"batSOC":0.00, "batSOH":0.00, "batVolt":0.00,
"batCurr":0.00, "batTemp_cell1":0.00,
"batTemp_cell2":0.00, "batTemp_cell3":0.00, "batID":"",
"batCycle":0, "sensor_flame":false, "sensor_smoke":false,
"sensor_tstat":false, "state_emg":false, "ctrl_chg":false,

```
"ctrl_buzzer":false,  "ctrl_fan":false,  "ctrl_aerosol":false
]
[2024-05-22 10:33:17.951791] [LOCKER 4  ] ["state_plug":false,
"state_door":true,  "state_charging":false,  "state_locker":3,
"batSOC":0.00,  "batSOH":0.00,  "batVolt":0.00,
"batCurr":0.00,  "batTemp_cell1":0.00,
"batTemp_cell2":0.00,  "batTemp_cell3":0.00,  "batID":"",
"batCycle":0,  "sensor_flame":false,  "sensor_smoke":false,
"sensor_tstat":false,  "state_emg":false,  "ctrl_chg":false,
"ctrl_buzzer":false,  "ctrl_fan":false,  "ctrl_aerosol":false
]
[2024-05-22 10:33:17.957513] [LOCKER 10  ] ["state_plug":false,
"state_door":true,  "state_charging":false,  "state_locker":3,
"batSOC":0.00,  "batSOH":0.00,  "batVolt":0.00,
"batCurr":0.00,  "batTemp_cell1":0.00,
"batTemp_cell2":0.00,  "batTemp_cell3":0.00,
"batID":"",  "batCycle":0,  "sensor_flame":false,
"sensor_smoke":false,  "sensor_tstat":false,
"state_emg":false,  "ctrl_chg":false,  "ctrl_buzzer":false,
"ctrl_fan":false,  "ctrl_aerosol":false
```

Pada tahap 1, SPBKLU akan dinyalakan dalam kondisi semua loker tertutup dengan loker 5, 8, dan 11 berisikan baterai.

Jika dilihat pada *log data system*, dapat dilihat bahwa berusaha membuat koneksi ke broker dan melakukan *subcriber* pada topik “spbklu/#” pada pukul 10:33:16, kemudian data yang didapatkan dari subkontrol tiap loker dicatat seluruhnya. Dari *log data system* tersebut juga dapat dilihat bahwa loker yang memiliki baterai adalah loker 5, 8, dan 11. Hal itu ditandai dengan data variebel “state_plug” bernilai “true” yang menandakan bahwa loker tersebut terhubung dengan baterai. Selain itu juga, terdapat data-data baterai seperti “battVolt”, “battCurr”, “batID”, dan lainnya yang nilainya telah berubah dari nilai defaultnya yang menandakan bahwa loker 5, 8, dan 11 terhubung dengan baterai. Adapun untuk kondisi pintu dari setiap loker dapat dilihat pada variebel “state_door”, dalam tabel yang berisikan *data log system* di atas semua nilai dari variebel

“state_door” untuk tiap loker bernilai “true” yang berarti semua pintu loker tertutup.

Gambar 4.61: Hasil pengujian keandalan HMI tahap 2

Tabel 4.24: Hasil Pengujian Keandalan HMI Tahap 2

Waktu	Nama Loker	Status Pintu Loker	Status Baterai
10:36:56	1	ON	-
10:36:56	2	ON	-
10:36:56	3	ON	-
10:36:56	4	ON	-
10:36:56	5	ON	SoC = 99 SoH = 100 Temp = 28 C Tegangan = 82 V Arus = 0 A
10:36:56	6	ON	-
10:36:56	7	ON	-
10:36:56	8	ON	SoC = 99 SoH = 100 Temp = 29 C Tegangan = 82 V Arus = 0 A
10:36:56	9	ON	-
10:36:56	10	ON	-
10:36:56	11	ON	SoC = 99 SoH = 100

			Temp = 28 C Tegangan = 81 V Arus = 0 A
10:36:56	12	ON	-

Tabel 4.25: Hasil *log data system* tahap 2

Log System Pada Tahap 2
[2024-05-22 10:35:45.771638] [LOCKER 7] ["state_door":false]
[2024-05-22 10:35:45.934216] [LOCKER 1] ["state_door":false]
[2024-05-22 10:35:46.270632] [LOCKER 4] ["state_door":false]
[2024-05-22 10:35:46.274779] [LOCKER 10] ["state_door":false]
[2024-05-22 10:35:47.829022] [LOCKER 7] ["ctrl_buzzer":true]
[2024-05-22 10:35:47.995376] [LOCKER 1] ["ctrl_buzzer":true]
[2024-05-22 10:35:48.330133] [LOCKER 4] ["ctrl_buzzer":true]
[2024-05-22 10:35:48.334346] [LOCKER 10] ["ctrl_buzzer":true]
[2024-05-22 10:36:04.005349] [LOCKER 11] ["state_door":false]
[2024-05-22 10:36:04.281336] [LOCKER 2] ["state_door":false]
[2024-05-22 10:36:05.378973] [LOCKER 5] ["state_door":false]
[2024-05-22 10:36:05.814786] [LOCKER 8] ["state_door":false]
[2024-05-22 10:36:06.070300] [LOCKER 11] ["ctrl_buzzer":true]
[2024-05-22 10:36:06.331100] [LOCKER 2] ["ctrl_buzzer":true]
[2024-05-22 10:36:06.868915] [LOCKER 4] ["ctrl_buzzer":false]
[2024-05-22 10:36:06.913350] [LOCKER 10] ["ctrl_buzzer":false]
[2024-05-22 10:36:07.429489] [LOCKER 5] ["ctrl_buzzer":true]
[2024-05-22 10:36:07.876645] [LOCKER 8] ["ctrl_buzzer":true]
[2024-05-22 10:36:08.427639] [LOCKER 7] ["ctrl_buzzer":false]
[2024-05-22 10:36:08.602105] [LOCKER 1] ["ctrl_buzzer":false]
[2024-05-22 10:36:24.933142] [LOCKER 3] ["state_door":false]
[2024-05-22 10:36:24.953184] [LOCKER 6] ["state_door":false]
[2024-05-22 10:36:24.991701] [LOCKER 9] ["state_door":false]
[2024-05-22 10:36:25.639503] [LOCKER 12] ["state_door":false]
[2024-05-22 10:36:26.663488] [LOCKER 11] ["ctrl_buzzer":false]
[2024-05-22 10:36:26.961203] [LOCKER 2] ["ctrl_buzzer":false]
[2024-05-22 10:36:27.040869] [LOCKER 6] ["ctrl_buzzer":true]
[2024-05-22 10:36:27.044149] [LOCKER 9] ["ctrl_buzzer":true]
[2024-05-22 10:36:27.730417] [LOCKER 12] ["ctrl_buzzer":true]
[2024-05-22 10:36:28.043918] [LOCKER 5] ["ctrl_buzzer":false]
[2024-05-22 10:36:28.474616] [LOCKER 8] ["ctrl_buzzer":false]
[2024-05-22 10:36:29.058696] [LOCKER 3] ["ctrl_buzzer":true]
[2024-05-22 10:36:47.599097] [LOCKER 3] ["ctrl_buzzer":false]
[2024-05-22 10:36:47.614379] [LOCKER 6] ["ctrl_buzzer":false]
[2024-05-22 10:36:47.645597] [LOCKER 9] ["ctrl_buzzer":false]
[2024-05-22 10:36:48.287399] [LOCKER 12] ["ctrl_buzzer":false]

Pada tahap 2, seluruh pintu loker akan dibuka dengan loker 5, 8, dan 11 yang masih sama yaitu terhubung dengan baterai. Pintu loker akan dibuka sekitar pukul 10:36:56 mulai dari loker 1, 4, 7, 10 kemudian berlanjut ke loker 2, 5, 8, 11 dan terakhir loker 3, 6, 9, 12. Hal ini dikarenakan pembukaan secara manual dilakukan menggunakan tombol yang terdapat pada bagian atas SPBKLU tempat loker kosong dimana terdapat 3 tombol yang masing masing terkoneksi dengan beberapa loker di bawahnya sehingga pembukaan dilakukan dengan urutan seperti itu.

Jika dilihat pada *data log system* sekitar pukul 10:36:56, dapat dilihat bahwa pintu loker yang ditandai dengan variebel “state_door” untuk loker 1, 4, 7, dan 11 bernilai “false” yang berarti pintu lokernya terbuka kemudian diikuti oleh variebel “ctrl_buzzer” yang bernilai true yang mengartikan bahwa alarm sedang menyala. Dikarenakan untuk sementara waktu pintu loker akan dibiarkan terbuka maka dalam selang waktu 30 detik alarm akan mati, hal ini sesuai dengan catatan pada *data log system* dimana “ctrl_buzzer” akan bernilai false setelah 30 detik kemudian. Setelah loker 1, 4, 7, dan 10 terbuka akan dilanjutkan dengan loker 2, 5, 8, 11 kemudian loker 3, 6, 9, 12.

Gambar 4.62: Hasil pengujian keandalan HMI tahap 3

Tabel 4.26: Hasil Pengujian Keandalan HMI Tahap 3

Waktu	Nama Loker	Status Pintu Loker	Status Baterai
10:38:17	1	ON	-
10:38:17	2	ON	-
10:38:17	3	ON	-
10:38:17	4	ON	-
10:38:17	5	ON	SoC = 99 SoH = 100 Temp = 28 C Tegangan = 82 V Arus = 0 A
10:38:17	6	ON	-
10:38:17	7	ON	-
10:38:17	8	ON	SoC = 99 SoH = 100 Temp = 29 C Tegangan = 82 V Arus = 0 A
10:38:17	9	ON	SoC = 97 SoH = 100 Temp = 29 C Tegangan = 82 V Arus = 2 A
10:38:17	10	ON	SoC = 100 SoH = 100 Temp = 29 C Tegangan = 82 V

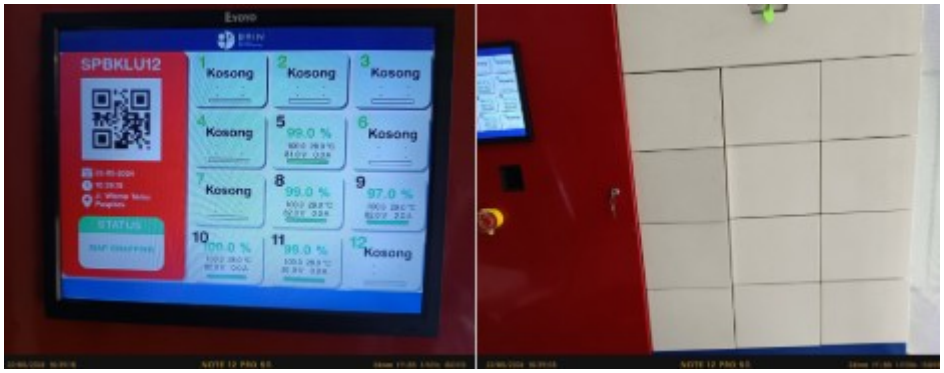
			Arus = 0 A
10:38:17	11	ON	SoC = 99 SoH = 100 Temp = 28 C Tegangan = 81 V Arus = 0 A
10:38:17	12	ON	-

Tabel 4.27: Hasil *log data system* tahap 3

Log System Tahap 3
[2024-05-22 10:37:39.573980] [LOCKER 10] ["state_plug":true, "state_locker":4, "batID":"00859007"]
[2024-05-22 10:37:41.635799] [LOCKER 10] ["state_locker":1, "batSOC":100.00, "batSOH":100.00, "batVolt":82.00, "batTemp_cell1":41.00, "batCycle":5]
[2024-05-22 10:37:43.693841] [LOCKER 10] ["batTemp_cell1":28.00]
[2024-05-22 10:38:01.806022] [LOCKER 9] ["state_plug":true, "state_locker":4, "batVolt":81.00, "batTemp_cell1":42.00, "batID":"00859009"]
[2024-05-22 10:38:03.861347] [LOCKER 9] ["state_locker":1, "batSOC":97.00, "batSOH":100.00, "batTemp_cell1":29.00, "batCycle":5]

Pada tahap 3, dalam kondisi semua pintu loker terbuka diberi tambahan 2 baterai yang dibuhubungkan pada loker 9 dan 10. Baterai dihubungkan sekitar pukul 10:38:17 dimulai dari loker 10 kemudian baru ke loker 9.

Jika dilihat pada *data log system*, pada 10:37:39 nilai variebel “state_plug” dari loker 10 bernilai true yang berarti loker 10 terhubung dengan baterai. Perubahan pada nilai variebel “state_plug” mengakibatkan perubahan pada variebel “state_locker”, "batSOC", "batSOH", "batVolt", "batTemp_cell1", dan "batCycle" yang membuat variebel-variebel tersebut dicatat di dalam *log data system* juga. Kemudian dilanjutkan dengan menghubungkan baterai pada loker 9 yang memiliki catatan pada variebel yang sama dengan loker 10.



Gambar 4.63: Hasil pengujian keandalan HMI tahap 4

Tahap 4.28: Hasil Pengujian Keandalan HMI Tahap 4

Waktu	Nama Loker	Status Pintu Loker	Status Baterai
10:39:15	1	OFF	-
10:39:15	2	OFF	-
10:39:15	3	OFF	-
10:39:15	4	OFF	-
10:39:15	5	OFF	SoC = 99 SoH = 100 Temp = 28 C Tegangan = 82 V Arus = 0 A
10:39:15	6	OFF	-
10:39:15	7	OFF	-
10:39:15	8	OFF	SoC = 99 SoH = 100 Temp = 29 C Tegangan = 82 V Arus = 0 A
10:39:15	9	OFF	SoC = 97 SoH = 100 Temp = 29 C Tegangan = 82 V Arus = 2 A
10:39:15	10	OFF	SoC = 100 SoH = 100 Temp = 29 C Tegangan = 82 V

			Arus = 0 A
10:39:15	11	OFF	SoC = 99 SoH = 100 Temp = 28 C Tegangan = 81 V Arus = 0 A
10:39:15	12	OFF	-

Tabel 4.29: Hasil *log data system* tahap 4

Log System Tahap 4
[2024-05-22 10:38:36.728329] [LOCKER 2] ["state_door":true]
[2024-05-22 10:38:36.790518] [LOCKER 3] ["state_door":true]
[2024-05-22 10:38:38.978111] [LOCKER 1] ["state_door":true]
[2024-05-22 10:38:41.986172] [LOCKER 5] ["state_door":true]
[2024-05-22 10:38:45.029256] [LOCKER 6] ["state_door":true]
[2024-05-22 10:38:45.062713] [LOCKER 9] ["state_door":true, "state_charging":true, "ctrl_chg":true]
[2024-05-22 10:38:46.554199] [LOCKER 8] ["state_door":true]
[2024-05-22 10:38:47.767630] [LOCKER 12] ["state_door":true]
[2024-05-22 10:38:48.878633] [LOCKER 11] ["state_door":true]
[2024-05-22 10:38:51.246203] [LOCKER 9] ["batVolt":82.00, "batCurr":1.00]
[2024-05-22 10:38:53.306139] [LOCKER 9] ["batCurr":3.00]
[2024-05-22 10:38:53.729974] [LOCKER 4] ["state_door":true]
[2024-05-22 10:38:55.279582] [LOCKER 7] ["state_door":true]
[2024-05-22 10:38:57.865693] [LOCKER 10] ["state_door":true]

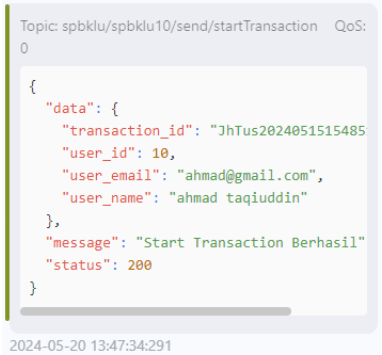
Pada tahap 4, pintu semua loker akan ditutup dengan loker 5, 8, 9, 10, dan 11 terhubung dengan baterai. Pintu ditutup secara acak pada tahap ini.

Jika dilihat pada *data log system*, variebel “state_door” pada setiap loker bernilai true yang berarti semua pintu loker tertutup. Selain itu, terdapat beberapa variebel pada loker 9 yang berubah sehingga dicatat pada *data log system* dikarenakan baterai yang dihubungkan dengan loker 9 memiliki SoC 97 sehingga akan dilakukan charge hingga SoC nya mencapai 99. Hal ini membuat perubahan nilai variebel pada variebel “state_charging” dan “ctrl_charging” menjadi true.

Berdasarkan dari perngujian yang dilakukan dengan membandingkan data yang tercatat pada *data log system* ketika SPBKLU dalam kondisi tertentu dimana *data log system* dapat mencatat dengan akurat waktu, loker, serta variebel yang menggambarkan kondisi loker dapat disimpulkan bahwa *data log system* dapat berfungsi dengan baik untuk mencatat data-data yang berkaitan dengan kondisi SPBKLU.

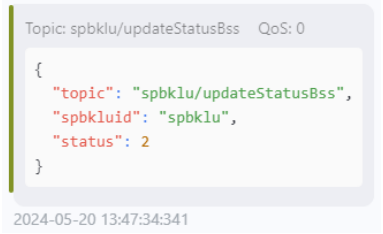
Pengujian *log data transaction*

Pengujian *log data transaction* dilakukan dengan membandingkan *log data transaction* dengan *log data system*, history pesan pada MQTTX, dan kondisi SPBKLU sebenarnya. Untuk data ID transaksi dan ID user didapatkan dari pesan permintaan untuk memulai transaksi yang dikirim oleh MQTTX pada pukul 13:47:34:291, dari pesan tersebut didapatkan data ID transaksi yaitu "JhTus202405151548516336225 4" dan ID user yaitu 10.



Gambar 4.64: Pesan permintaan untuk melakukan transaksi

Untuk data waktu mulai dan waktu selesai didapat dengan melihat pesan diterima pada MQTTX yang sudah terhubung dengan broker dan juga melakukan *subcribe* ke topic “spbklu/#”, dengan melihat waktu pesan diterima didapatkan data waktu mulai yaitu 2024-05-20 13:47:34:341 dan data waktu selesai yaitu 2024-05-20 13:51:22:499.

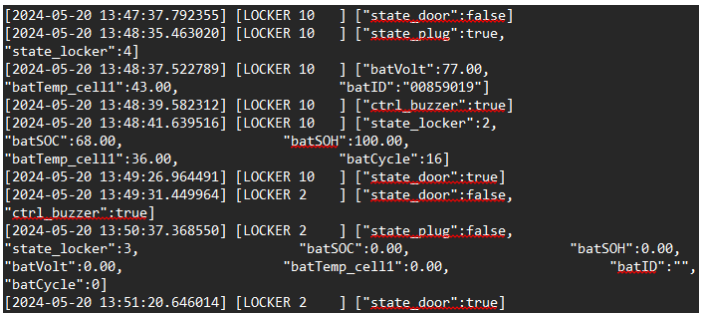


Gambar 4.65: Pesan tanda transaksi dimulai

Gambar 4.66: Pesan tanda transaksi selesai

Adapun untuk data waktu loker masuk dapat dilihat dari *data log system* sekitar waktu mulai dan waktu selesai, berdasarkan *data log system* diantara selang waktu tersebut terdapat 2 loker yang terbuka yaitu loker 10 dan loker 2 dimana loker 10 terbuka dan menutup terlebih dahulu dan loker 2 terbuka dan menutup setelahnya yang berarti loker 10 adalah tempat baterai masuk dan loker 2 adalah tempat baterai keluar. Data baterai 10 juga tercantum pada *data log system* dikarenakan adanya perubahan data loker dari yang sebelumnya kosong menjadi terisi yaitu SoC = 68, SoH = 100, temp = 36, ID baterai = 00859019, dan cycle =

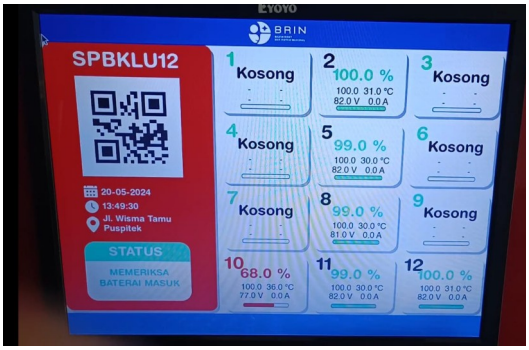
16. Adapun untuk data pada loker 2 bernilai kosong karena *data log system* mencatat perubahan yang terjadi akibat baterai telah terlepas dari socket.



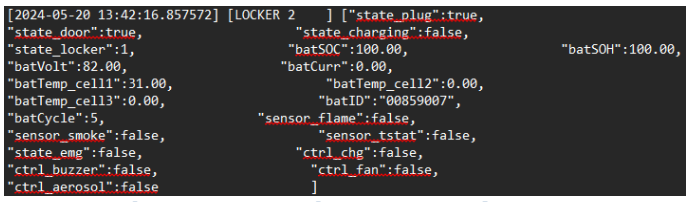
[2024-05-20 13:47:37.792355] [LOCKER 10] ["state_door":false]
[2024-05-20 13:48:35.463020] [LOCKER 10] ["state_plug":true,
"state_locker":4]
[2024-05-20 13:48:37.522789] [LOCKER 10] ["batVolt":77.00,
"batTemp_cell1":43.00, "batID":"00859019"]
[2024-05-20 13:48:39.582312] [LOCKER 10] ["ctrl_buzzer":true]
[2024-05-20 13:48:41.639516] [LOCKER 10] ["state_locker":2,
"batSOC":68.00, "batSOH":100.00,
"batTemp_cell1":36.00, "batCycle":16]
[2024-05-20 13:49:26.964491] [LOCKER 10] ["state_door":true]
[2024-05-20 13:49:31.449964] [LOCKER 2] ["state_door":false,
"ctrl_buzzer":true]
[2024-05-20 13:50:37.368550] [LOCKER 2] ["state_plug":false,
"state_locker":3, "batSOC":0.00, "batSOH":0.00,
"batVolt":0.00, "batTemp_cell1":0.00, "batID":"",
"batCycle":0]
[2024-05-20 13:51:20.646014] [LOCKER 2] ["state_door":true]

Gambar 4.67: Data log system ketika transaksi battery swapping berlangsung

Untuk mendapatkan data dari loker 2 dapat dilakukan dengan melihat *data log system* ketika baterai dimasukkan sebelum pengujian transaksi *battery swapping* dilakukan ataupun dengan melihat pada HMI. Dari HMI didapatkan SoC = 100, SoH = 100, Volt = 82V, Arus = 0A, dan Temp = 31 sedangkan untuk ID baterai dan cycle dapat dilihat di *data log system* pada pukul 13:42:16 yaitu ID baterai = 00859007, cycle = 5. Dari data yang telah dikumpulkan sebelumnya didapatkan data sebagai berikut.



Gambar 4.68: Tampilan HMI sebelum baterai pada loker 2 terlepas dari socket

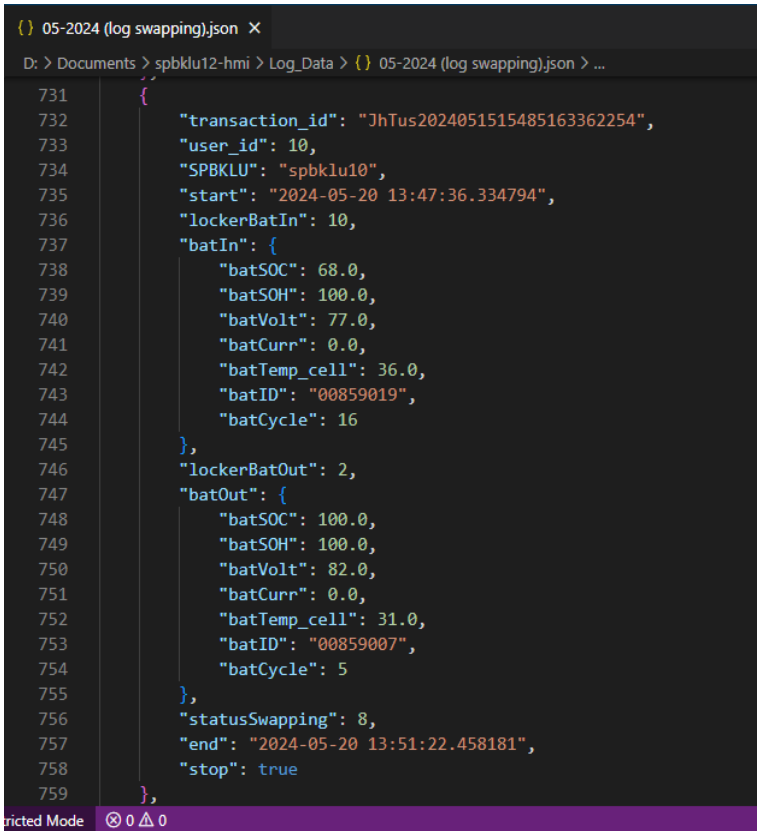


[2024-05-20 13:42:16.857572] [LOCKER 2] ["state_plug":true,
"state_door":true, "state_charging":false,
"state_locker":1, "batSOC":100.00, "batSOH":100.00,
"batVolt":82.00, "batCurr":0.00,
"batTemp_cell1":31.00, "batTemp_cell2":0.00,
"batTemp_cell3":0.00, "batID":"00859007",
"batCycle":5, "sensor_flame":false,
"sensor_smoke":false, "sensor_tstat":false,
"state_emg":false, "ctrl_chg":false,
"ctrl_buzzer":false, "ctrl_fan":false,
"ctrl_aerosol":false]

Gambar 4.69: Data log system pada 13:42:16

Tabel 4.30: Data transaksi *battery swapping*

Waktu Mulai	Transaction ID	User ID	Loker Baterai Masuk	Data Baterai Masuk	Loker Baterai Keluar	Data Baterai Keluar	Waktu Selesai
2024-05-20 13:47:34:341	JhTus20240515154851633622 54	10	10	SoC : 68 SoH : 100 Volt : 77 Arus : 0 Temp : 36 ID : 00859019 Cycle : 16	2	SoC : 100 SoH : 100 Volt : 82 Arus : 0 Temp : 31 ID : 00859007 Cycle : 5	2024-05-20 13:51:22:499



Gambar 4.70: Tampilan data log transaction

Jika dibandingkan antara data yang didapatkan dari HMI, *data log system*, dan riwayat pesan MQTTX dengan data yang didapat dari *data log system* terdapat perbedaan pada data waktu mulai dan data waktu selesai yaitu sekitar

1.993 detik untuk waktu mulai dan 0.041 detik untuk waktu selesai. Adapun untuk data lainnya memiliki nilai yang sama.

Berdasarkan data tersebut, dapat diambil kesimpulan bahwa data pada *data log transaction* memiliki kesesuaian yang tinggi dengan data yang didapatkan dari HMI, *data log system*, dan riwayat pesan pada MQTTX sehingga dapat dikatakan bahwa data pada *data log system* dapat berjalan dengan baik karena memiliki tingkat keakuratan yang tinggi.