CB-08

# PEMBUATAN APLIKASI ANDROID BERBASIS *INTERNET OF THINGS* UNTUK MENGONTROL AGV (*AUTOMATED GUIDE VEHICLE*)

Suhartinah[1], M. Hidayat[2], Angga Putra Hafidzin[3]
Mekatronika, Politeknik Manufaktur Astra, Jakarta, 14330, Indonesia
E-mail : suhartinah@polman.astra.ac.id[1], m.hidayat@polman.astra.ac.id[2], anggaputrahafidzin@gmail.com[3]

***Abstrak*** **-- Salah satu distributor alat berat di Indonesia memiliki gudang untuk menyimpan *Damage Core (DC)* atau komponen yang mengalami kerusakan dari unit alat berat. *DC* tersebut akan dikirim ke pabrik untuk dilakukan investigasi mengenai penyebab kerusakannya. Dalam proses pengiriman terdapat beberapa *DC* yang tidak berhasil dikirim ke pabrik. Kegagalan memenuhi permintaan pengembalian *DC* disebut *DC cancel*. Pada periode bulan Januari sampai dengan Desember 2018 prosentase *DC cancel* mencapai *6,8%* dari total permintaan. Diketahui terdapat 6 permasalahan yang menyebabkan *DC cancel*, yaitu kehilangan komponen, ketidaksesuaian informasi, komponen telah diperbaiki, keluar masa garansi, kerusakan pada saat penyimpanan dan kesalahan pengiriman. Penulis mengambil salah satu masalah dari *DC cancel* yaitu ketidaksesuaian informasi untuk dilakukan perbaikan. Perbaikan yang dilakukan yaitu membuat sebuah aplikasi berbasis *Internet of Things* (*IoT)* yang dapat terintegrasi dengan sistem penyimpanan *DC* di gudang. Aplikasi tersebut digunakan untuk menyimpan data *DC* ke *database,* penentuan lokasi penyimpanan *DC,* pencarian *DC,* konektivitas dengan *PLC* sebagai kontroler *AGV* dan proses pengeluaran *DC*. Pertukaran data antara aplikasi android dengan *PLC* memanfaatkan mikrokontroler nodemcu. Mikrokontroler tersebut terhubung secara serial ke *port* RS232 yang ada pada *PLC* untuk melakukan komunikasi. Dari hasil analisa dan modifikasi alat ini ketidaksesuaian informasi yang menyebabkan DC cancel dapat ditangani**

***Kata kunci*** **: *: Internet of Things,* Mikrokontroler, Aplikasi android, Komunikasi serial.**

## I. PENDAHULUAN

Dalam memberikan pelayanan purna jual, Perusahaan menangani secara langsung permasalahan yang dialami oleh pelanggan. Pelayanan yang diberikan kepada pelanggan meliputi pemeliharaan preventif dan korektif. Untuk unit yang mengalami kerusakan komponen maka harus dilakukan penggantian komponen. Perusahaan wajib menyimpan '*Damage Core'* (komponen rusak) tersebut selama maksimal 8 bulan. Pabrik akan mengirimkan *Part Return Request* kepada perusahaan untuk proses pengembalian *Damage Core* (*DC*).

Dalam prosesnya, ditemukan beberapa masalah yang menyebabkan *DC* tidak dapat dikirim atau tidak sesuai dengan permintaan, kondisi tersebut dikenal dengan istilah *DC cancel*. Pada periode bulan Januari sampai dengan Desember 2018 terdapat 6,8% *DC cancel* dari total 397 *Part Return Request* yang diajukan oleh pabrik. Angka tersebut berada diatas prosentase standar *DC cancel* yaitu 5%. Berdasarkan data dan kondisi di lapangan, ditemukan 6 permasalahan yang menyebabkan *DC cancel*, yaitu kehilangan komponen, ketidaksesuaian informasi, komponen telah diperbaiki, keluar masa garansi, kerusakan pada saat penyimpanan dan kesalahan pengirimann.

Penulis memilih salah satu permasalahan yaitu ketidaksesuaian informasi (*miss information*) untuk dianalisa dan dilakukan perbaikan. Perbaikan dapat dilakukan dengan melakukan digitalisasi proses penyimpanan dan pengeluaran *DC* di gudang. Lalu dibuatlah aplikasi *Damage Core Automation System* (DCAS) berbasis android yang terintegrasi dengan *Automated Guided Vehicle*. DCAS digunakan untuk mengurangi kesalahan yang dapat terjadi pada proses pemasukan data, penentuan lokasi penyimpanan, proses penyimpanan, pencarian, pengambilan dan pengeluran *DC* pada gudang. Aplikasi DCAS terhubung dengan mikrokontroler nodeMCU melalui jaringan internet yang akan meneruskan data ke *PLC* CJ2M (kontroler *AGV*). Dengan melakukan perbaikan tersebut dapat mengurangi *DC cancel* karena faktor ketidaksesuaian informasi pada gudang PT United Tractors.

## II. TINJAUAN PUSTAKA

### 2.1 Pengertian *Internet of Things*

Internet of Things merupakan sebuah perkembangan teknologi dimana terhubungnya sensor dan aktuator pada software melalui jaringan, sehingga dapat mengatur dan mengawasi objek yang terhubung, baik benda mati atau bahkan makhluk hidup.[1] Metode konektivitas yang digunakan pada *Internet of Things* adalah nirkabel atau pengendalian tanpa mengenal jarak karena menggunakan jaringan internet. *Internet of Things* terdiri dari 3 elemen utama, yaitu elemen perangkat keras, perangkat lunak dan komunikasi.

Aplikasi seluler merupakan sebuah program komputer yang dirancang untuk piranti bergerak seperti telepon seluler, tablet atau jam tangan pintar.[2] Pada

[1] McKinsey, How we can recognize real power of IoT, (Online), Mei 2019 (https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/how-can-we-recognize-the-real-power-of-the-internet-of-things, diakses 1 Mei 2019).
[2] Wikipedia, Pengertian Aplikasi Seluler (Online), Mei 2019

umumnya terdapat 3 macam tipe aplikasi seluler, yaitu aplikasi web, native dan hybrid. Berikut ini beberapa perbedaan dari ketiga aplikasi seluler tersebut:

| Parameter | Native | Web | Hybrid |
|---|---|---|---|
| Bahasa | Native | HTML5 | JavaScript |
| Multi Platform | Tidak | Ya | Ya |
| Aksesibilitas | *Offline dan online* | *Online* | *Offline dan online* |
| Integrasi Hardware | Ya | Tidak | Ya |

2.2 Sistem Operasi Seluler

Sistem operasi seluler dikenal dengan sebutan platform, perangkat lunak untuk perangkat seluler, sistem operasi seluler adalah sebuah program yang menggerakan suatu perangkat keras seperti telepon seluler, ponsel pintar, tablet dan perangkat informasi lainnya. Ada berbagai jenis sistem operasi (OS) yang dijalankan perangkat seluler, yaiut iOS, Android OS, WindowsPhone OS dan sebagainya. (Yoga Permana, 2012).

Sistem Operasi Android adalah sebuah sistem operasi seluler yang dikembangkan oleh Google untuk digunakan pada perangkat layar sentuh (*touchscreen*). Sistem operasi ini pertama kali dikembangkan oleh Android, Inc. Sistem operasi android bersifat *open-source*, sehingga memungkinkan perangkat lunak untuk dimodifikasi secara bebas. Umumnya sistem operasi android ditulis dalam bahasa pemrograman Java.

iOS merupakan sebuah sistem operasi seluler milik Apple yang dikembangkan dan diaplikasikan hanya untuk perangkat Apple Inc., seperti iPhone, iPad, dan Apple TV. Antarmuka dari iOS didasarkan pada konsep manipulasi langsung melalui sentuhan pada layar. Sistem operasi iOS bersifat *closed-source*, sehingga sangat tidak mungkin untuk mengembangkan dan membuat sistem operasi baru dari iOS.

2.3 Aplikasi Android

Aplikasi Android ditulis dalam bahasa pemrograman Java dan Kotlin. Android SDK (*Software Development Kit*) merupakan perangkat lunak yang dapat mengompilasi kode program, data dan file sumber daya aplikasi menjadi sebuah APK (paket file android yang berbentuk .apk). Satu file APK berisi semua materi aplikasi android dan merupakan file yang digunakan untuk memasang aplikasi. [3]

2.4 Basis Data

Basis data ialah suatu kumpulan dari berbagai data yang saling terkait sedemikian rupa sehingga dapat lebih mudah disimpan, dimanipulasi dan cepat ketika di panggil penggunanya.[4] Basis data bersifat terpadu dan terbagi. Terpadu artinya data yang ada pada database saling terkait. Terbagi artinya data yang sama dapat dipakai oleh sejumlah pengguna dalam waktu yang bersamaan.

2.5 Perangkat Lunak (*Software*)

Menurut Roger S. Pressman (2002), pengertian software adalah suatu perintah program dalam sebuah komputer yang apabila dieksekusi oleh usernya akan memberikan fungsi dan unjuk kerja seperti yang diharapkan oleh user-nya. Dengan kata lain, perangkat lunak berfungsi untuk memberi perintah kepada komputer agar dapat berfungsi secara optimal sesuai dengan perintah user.

2.6 Jaringan Komputer

Jaringan komputer adalah koneksi antara dua perangkat atau lebih, yang terhubung secara fisik maupun secara logika sehingga bisa saling bertukar informasi. Jaringan komputer dapat dikatakan terhubung apabila perangkat yang ada dalam jaringan tersebut bisa saling bertukar data dan berbagi materi yang dimiliki.

Internet (akronim dari *interconnection-networking*) adalah seluruh jaringan komputer yang saling terhubung menggunakan standar sistem global *Transmission Control Protocol/Internet Protocol Suite (TCP/IP)* sebagai protokol pertukaran paket (*packet switching communication protocol*) untuk melayani miliaran pengguna di seluruh dunia.

2.7 Mikrokontroler

Menurut Chamim (2012) Mikrokontroler adalah sebuah sistem komputer yang seluruh atau sebagian besar elemennya dikemas dalam satu chip IC, sehingga sering disebut *single chip microcomputer*. Mikrokontroler merupakan sistem komputer yang mempunyai salah satu atau beberapa tugas yang sangat spesifik.

2.8 Pengertian Gudang

Gudang adalah suatu tempat yang digunakan untuk menyimpan barang-barang berupa *raw material*, barang *work in process* atau *finished good*. Menurut Holy Icun Yunarto dan Martinus Getty Santika (2005) kegiatan tersebut dapat meliputi kegiatan *movement* (perpindahan), *storage* (penyimpanan) dan *information transfer* (transfer informasi).

## III. METODOLOGI

3.1 Alur Proses *Damage Core*

(https://id.wikipedia.org/wiki/Aplikasi_seluler, diakses 2 Mei 2019)

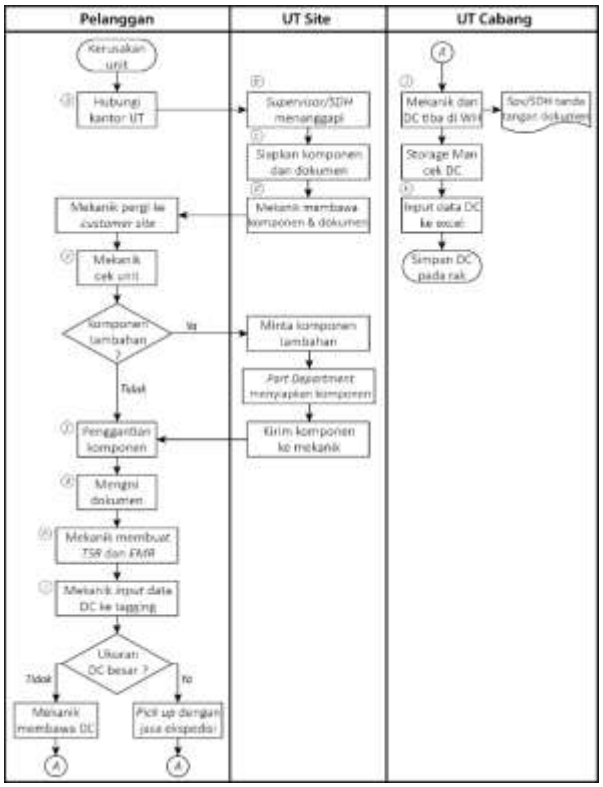
[3] Android, Dasar-dasar Aplikasi Android (Online) Mei 2019 (https://developer.android.com/guide/components/fundamentals?hl=id , diakses 10 Juli 2019)

[4] Ruang Guru, Definisi Basis Data (Online) Juni 2019 (https://www.ruangguru.co.id/pengertian-basis-data-dan-sistem-basis-data-definisi-tujuan-fungsi-dan-komponennya/, diakses 21 Juni 2019)

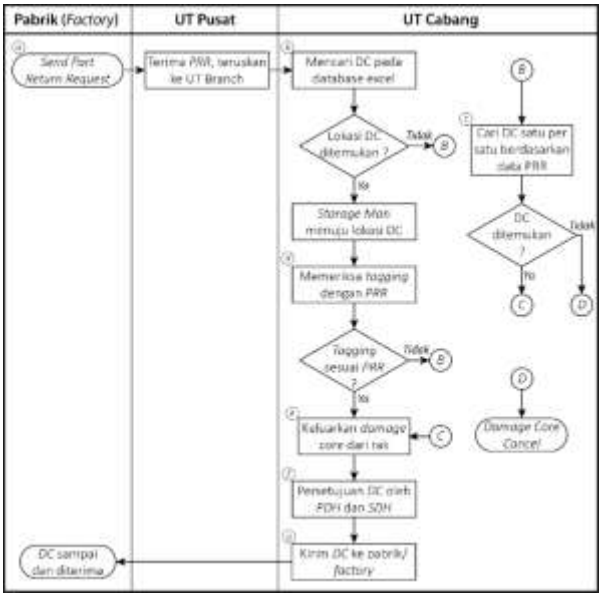
Pada saat unit alat berat produk mengalami kerusakan, Perusahaan akan menindaklanjuti kerusakan tersebut dengan mengirimkan mekanik ke lokasi untuk melakukan perbaikan atau penggantian komponen. Pada proses penggantian komponen terdapat komponen rusak dari unit alat berat, komponen tersebut dinamakan *damage core (DC)*.

Apabila *DC* diambil dari unit yang masih berada dalam masa garansi maka komponen tersebut harus disimpan oleh PT UT di gudang selama maksimal 8 bulan. Dalam masa penyimpanan, *DC* tidak boleh rusak ataupun hilang karena nantinya akan dikembalikan ke pabrik (tempat dimana komponen dibuat) tertentu sesuai dengan permintaan.

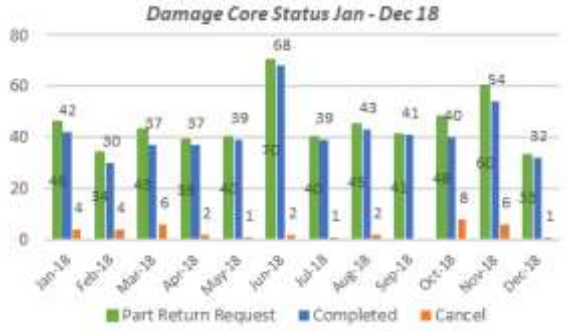
*Take-in DC* merupakan proses untuk menyimpan *DC* ke gudang, sedangkan *Take-out* untuk mengeluarkan *DC* dari gudang. Berikut ini adalah alur proses *take-in DC*:




Setelah *DC* berhasil disimpan ke gudang maka proses *take-in* selesai. Apabila pada masa penyimpanan terdapat *Part Return Request* yang dikirimkan oleh *Komatsu Factory* kepada PT UT, maka dilakukanlah proses *take-out DC*. Berikut ini alur prosesnya:

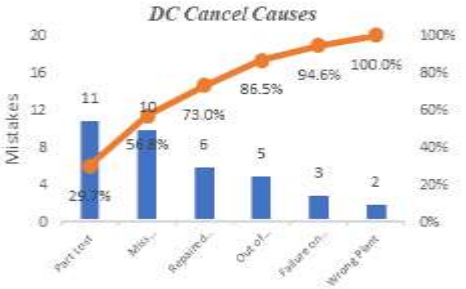



Dalam pelaksanaannya tidak semua *DC* yang diminta oleh *Komatsu Factory* dapat dikirim kembali. Kondisi seperti disebut dengan istilah *Damage Core Cancel*. Prosentase *DC cancel* pada tahun 2019 mencapai angka 6.8% dari total *Part Return Request*. Prosentase *DC cancel* tersebut melebihi standar yang sudah ditentuka oleh PT KMSI dan PT UT yaitu sebesar 5%.




3.2 Analisa Kondisi *DC Cancel*

Terdapat beberapa permasalahan yang menyebabkan terjadinya *DC cancel*, yaitu hilang komponen, informasi tidak sesuai, komponen telah diperbaiki, keluar masa garanasi, kerusakan saat penyimpanan dan kesalahan dalam pengiriman.



3.3 Analisa Masalah

Dari analisa kondisi yang ada penulis memilih salah satu penyebab *DC cancel* yaitu *miss information* untuk dilakukan perbaikan. Berikut analisa penulis terhadap *Damage Core cancel* yang disebabkan oleh faktor kesalahan informasi (*miss information*) :

| ***Tree of Cause*** | | | |
|---|---|---|---|
| **Masalah** | **Penyebab 1** | **Penyebab 2** | **Penyebab 3** |
| *Damage core cancel* melebihi 5% | Salah informasi *tagging DC* | Tulisan tidak terbaca | Tulisan kecil |
| | | | Tulisan tidak rapih |
| | Pencarian DC dilakukan satu per satu | Lokasi DC tidak ditemukan pada database excel | Salah mengisi informasi DC pada database excel |
| | Pengambilan DC terlalu lama | Melakukan 2 pekerjaan secara manual (pencarian dan pengambilan) | Belum adanya sistem pencarian dan pengambilan DC |

3.4 Rencana Perbaikan

Dari tabel analisa masalah diatas dapat diketahui permasalahan utama yang meneybabkan *DC cancel* karena factor kesalahan informasi. Dari permasalahan utama tersebut, penulis membuat beberapa rencana perbaikan untuk menanggulangi terjadinya *DC cancel*.

| Masalah | Perbaikan | Alasan | Sasaran |
|---|---|---|---|
| Tulisan kecil dan tidak rapih | • Input data *DC* secara digital dan *online*<br>• *Digital tagging DC* berupa *QR code* | • Standarisasi penulisan data *DC*<br>• Sarana lain untuk simpan data *DC* | Tidak ada salah data *DC* pada tagging |
| Salah mengisi informasi *DC* pada database excel | • Mengganti *database excel* dengan *cloud database.*<br>• Pencarian lokasi kosong pada rak | • Data DC bisa didapatkan dari *cloud database*<br>• Mudah dalam menyimpan DC | Tidak ada salah data DC pada database |
| Tidak ada integrasi pencarian dan pengambilan DC | • Membuat fitur pencarian DC<br>• Integrasi alat pengambilan dan penyimpanan | • Mudah mencari DC<br>• Penyimpanan dan pengambilan secara otomatis | Mudah ambil dan simpan DC di warehouse |

Berdasarkan tabel diatas terdapat 6 rencana perbaikan yang dapat dilakukan. Rencana-rencana perbaikan tersebut akan dikemas sedemikian rupa dalam sebuah aplikasi. Selain mengacu pada rencana perbaikan, aplikasi yang akan dibuat juga disesuaikan dengan kebutuhan dan atau permintaan perushaan. Berikut ini daftar permintaan dan kebutuhan dari perusahaan :

| No | Daftar Kebutuhan dan Permintaan Perusahaan |
|---|---|
| 1 | Aplikasi dapat dijalankan pada ponsel pintar |
| 2 | Untuk tahap ini aplikasi dapat digunakan oleh 4 posisi jabatan, yaitu mekanik, petugas gudang, kepala departemen servis dan kepala departemen *part* PT UT |
| 3 | Aplikasi bersifat privat dan hanya dapat diakses oleh pengguna yang terdaftar |
| 4 | Metode *sign-in* menggunakan email dan password |
| 5 | Registrasi pengguna aplikasi dilakukan oleh admin aplikasi |
| 6 | Dapat menyimpan informasi pengguna aplikasi meliputi nama, jabatan, foto profil dan email |
| 7 | Menyediakan layanan apabila pengguna lupa terhadap password yang dimiliki |
| 8 | Aplikasi dapat menginput dan menyimpan data *DC* sesuai informasi yang ada pada *tagging* |
| 9 | Proses penulisan data *DC* secara manual pada *tagging* tidak boleh dihilangkan |
| 10 | Diperbolehkan untuk menambahkan kode baris (*barcode*) ataupun kode QR (*QR code*) di belakang *tagging DC* |
| 11 | Aplikasi dapat melakukan pembacaan kode baris ataupun kode QR pada *tagging DC* |
| 12 | Aplikasi memberikan pemberitahuan kepada Kepala Departemen Servis dan Kepala Departemen *Part* terkait permintaan pengeluaran *DC* |

Penulis menyesuaikan permintaan dan kebutuhan serta rencana perbaikan yang sudah penulis buat menjadi beberapa fitur dalam sebuah aplikasi. Berikut ini terdapat 5 fitur utama yang akan disediakan dalam aplikasi manajemen gudang:

1. *Online Input DC Data*
   Pengisian dilakukan mekanik di lokasi kerusakan unit menggunakan ponsel pintar. Data *DC* yang dikirim akan tersimpan pada *cloud database*.
2. *Take-in DC*
   Merupakan fitur untuk menyimpan *DC* pada gudang. Setelah berhasil menuliskan lokasi penyimpanan pada gudang maka DC sudah dapat disimpan baik secara manual ataupun otomatis (menggunakan *Automated Guide Vehicle*). Apabila penyimpanan dilakukan dengan cara otomatis maka aplikasi akan mengirimkan data ke PLC sebagai parameter lokasi bagi *AGV*.
3. *Damage Core List*
   Pada fitur ini menampilkan daftar *DC* yang sudah berhasil disimpan. Daftar tersebut dapat kita pilih untuk melihat informasinya secara keseluruhan.
4. *Approval Take-out DC*
   Fitur ini menampilkan *DC* yang hendak dikirim kembail ke pabrik. Aplikasi akan mengirimkan pemberitahuan kepada Kepala Departemen Komponen dan Servis bahwa terdapat *DC* yang perlu disetujui.
5. *Take-out DC*
   Fitur ini merupakan kelanjutan dari *approval take-out DC*. Pada fitur ini akan muncul *DC* yang sudah mendapat persetujuan dari Kepala Departemen Komponen dan Servis, lalu *DC* bisa dikeluarkan dari *warehouse* baik menggunakan metode manual ataupun otomatis.

3.5 Penentuan Tipe Aplikasi

Perbaikan yang akan dilakukan yaitu membuat aplikasi sistem manajemen gudang yang dapat dijalankan pada perangkat bergerak seperti ponsel pintar. Terdapat 3 tipe aplikasi seluler yang dapat dijalankan pada ponsel pintar, yaitu aplikasi web, aplikasi *native* dan aplikasi *hybrid*. Penentuan tipe aplikasi seluler yang akan dibuat didasarkan pada beberapa aspek, yaitu sebagai berikut:

| Penentuan | Web | *Native* | *Hybrid* |
|---|---|---|---|
| Bahasa pemrograman | HTML | Java, Kotlin, Objective-C, Swift dan lain-lain | JavaScript |
| Aksesibilitas perangkat keras | Tidak tersedia | Tersedia | Tersedia |
| Platform perangkat | Tersedia pada semua platform | Tersedia pada spesifik platform | Tersedia pada semua platform |
| Responsibilitas Aplikasi | Tergantung pada jaringan internet | Berjalan stabil dan dapat dijalankan tanpa ataupun dengan jaringan internet | Tergantung pada jaringan internet |
| Fitur notifikasi | Tidak tersedia | Tersedia | Tersedia |

Berdasarkan tabel diatas penulis memilih untuk mengembangkan aplikasi native karena aspek aksesibilitas perangkat keras, responsibilitas aplikasi dan fitur notifikasi saat ini lebih unggul.

Penentuan *Database*

Dalam melakukan perbaikan terhadap proses penyimpanan dan pengambilan *DC* dibutuhkan *database* yang dapat bekerja secara *real-time*. Selain itu *database* juga harus dapat diakses secara *online*. Berikut beberapa *database* yang umum dipakai untuk aplikasi seluler :

1. MongoDB
   MongoDB adalah salah satu jenis database NoSQL yang berbasis dokumen dengan fomat JSON.
2. Firebase
   Firebase merupakan salah satu NoSQL *database*. Firebase database berjalan secara *real-time*, dimana jika pengguna mengubah atau melakukan tindakan pada suatu data maka pengguna lain akan tersinkronisasi secara langsung
3. SQLite
   SQLite merupakan sebuah *relational database management system* yang memiliki ukuran pustaka kode relatif kecil. SQLite bukanlah sebuah sistem yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan API secara langsung melalui bahasa pemrograman.

Berdasarkan tabel diatas penulis memilih Firebase *Real-time Database* sebagai *database* yang akan digunakan bersama dengan *IDE Android Studio*. Firebase *Real-time Database* mendukung *cloud database* dimana sangat sesuai untuk aplikasi yang akan dijalankan menggunakan jaringan internet.

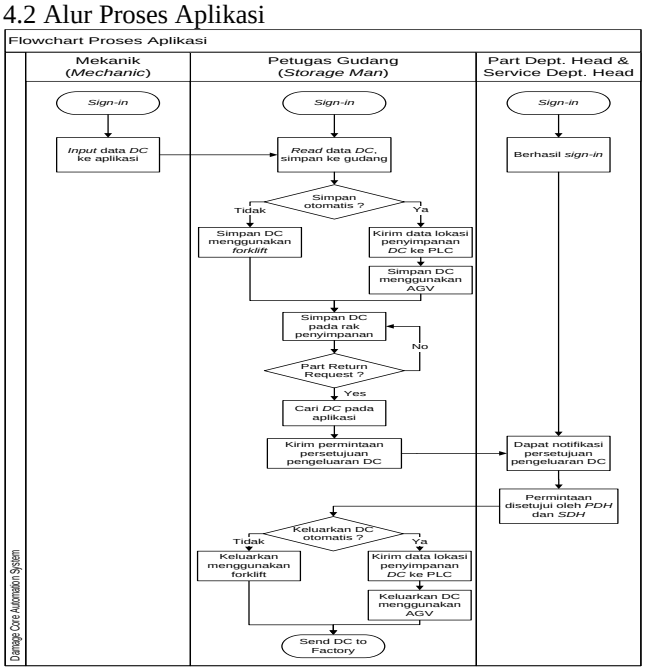
## IV. EVALUASI HASIL

4.1 Pembuatan Aplikasi

Aplikasi yang akan dibuat diberi nama *Damage Core Automation System (DCAS).* Aplikasi akan digunakan oleh Mekanik, Petugas Gudang, Kepala Departemen Servis (*Service Dept. Head*) dan Kepala Departemen *Part* (*Part Dept. Head*). Mekanik menggunakan aplikasi untuk menyimpan dan mengirim data *DC* saat di *site* secara *online*. Data yang sudah dikirim akan disimpan pada firebase *realtime database*. Petugas gudang dapat mengakses data yang sudah dikirimkan oleh mekanik. Data tersebut menjadi informasi *DC* yang nantinya akan disimpan di gudang. Kepala depertemen servis dan kepala departemen part memiliki wewenang untuk memberikan persetujuan terhadap permintaan pengeluaran *DC.*

Pembuatan aplikasi akan dibagi menjadi beberapa tahap, yaitu pembuatan *flowchart* proses aplikasi, pemetaan aplikasi, pembuatan *database* dan implementasi program. Pembuatan *flowchart* proses didasarkan pada alur proses *take-in* dan *take-out DC*. Pemetaan aplikasi digunakan untuk mengetahui fitur apa saja yang ada dalam aplikasi. Dengan pemetaan aplikasi, penulis juga dapat mengklasifikasikan fitur apa saja yang dapat diakses oleh pengguna. Pembuatan *database* untuk menyimpan data aplikasi menggunakan *firebase real-time database*. Implementasi program merupakan tahapan yang paling penting dalam pembuatan aplikasi.

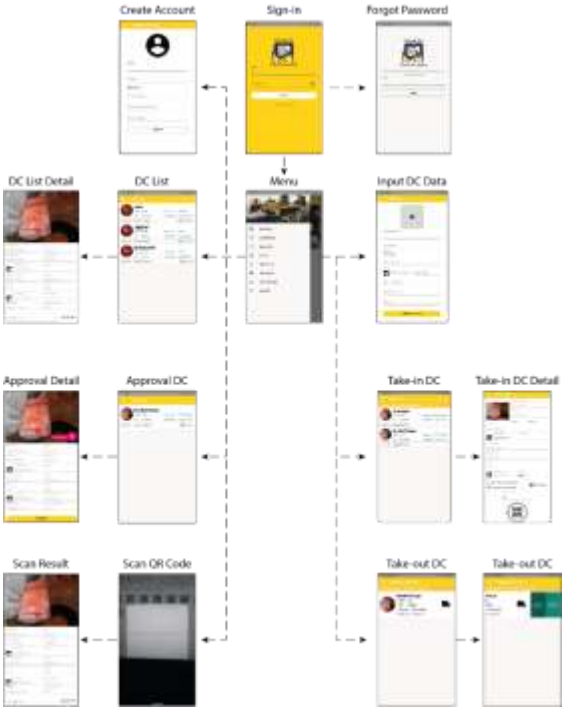
4.2 Alur Proses Aplikasi




Pada flowchart proses aplikasi diatas, dapat diketahui bahwa pengguna aplikasi dibagi menjadi 4, yaitu Mekanik, Petugas gudang, *Part Department Head* dan *Service Department Head*. Untuk jabatan *Part Department Head* dan *Service Department Head* memiliki tugas dan wewenang yang sama, maka dari itu dikelompokan menjadi satu bagian saja. Setiap pengguna memiliki tugas dan wewenangnya masing-masing. Mekanik memiliki tugas untuk memasukan data *damage core* ke dalam aplikasi. Setelah data tersimpan pada *cloud database* dan *DC* sudah tiba di gudang, maka petugas gudang akan mengambil data *DC* yang sesuai. Petugas gudang tidak perlu lagi menuliskan data *DC* yang tertera pada *tagging DC*. Selanjutnya *DC* tersebut dapat disimpan ke dalam rak penyimpanan secara otomatis ataupun manual. Penyimpanan *DC* dengan cara otomatis artinya aplikasi akan menyimpan kode lokasi pada database 'Arduino', arduino atau nodemcu akan membaca kode lokasi tersebut secara *real-time*. Data yang telah berhasil didapatkan oleh arduino akan dikirim ke PLC sebagai parameter pergerakan *AGV*.

Setelah *DC* berhasil disimpan ke dalam rak, informasi mengenai *DC* akan dipindahkan ke *database* '*Rack*'. Durasi maksimal peyimpanan *DC* yaitu 8 bulan. Apabila pada masa penyimpanan terdapat permintaan pengembalian komponen (*Part Return Request*) yang diajukan oleh pabrik Komatsu, maka petugas gudang akan mencari *DC* tersebut pada aplikasi. Pencarian *DC* dilakukan dengan memasukkan infomrasi *DC* seperti *SMR, serial number* dan *part number*. Saat *DC* sudah ditemukan maka petugas gudang akan mengajukan proses permintaan persetujuan kepada *SDH* dan *PDH*. Setelah mendapatkan persetujuan maka data *DC* akan muncul dan sudah siap untuk dikeluarkan dari gudang.

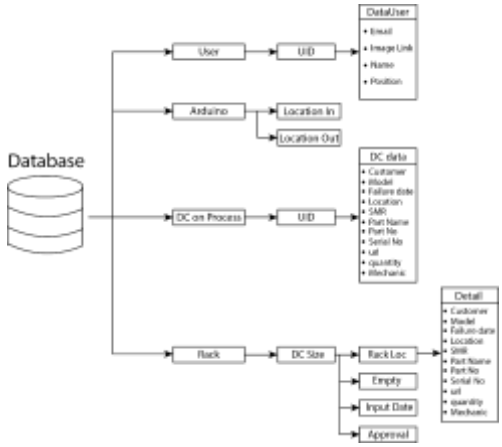
4.3 Pemetaan Aplikasi




Pada gambar diatas berisi fitur-fitur apa saja yang ada dalam aplikasi. Pengguna akan memulai aplikasi dengan melakukan proses *sign-in* terlebih dahulu. Setelah berhasil melakukan proses *sign-in*, pengguna sudah dapat mengakses fitur-fitur yang tersedia. Ketersediaan fitur pada aplikasi disesuaikan dengan jabatan dari pengguna. Berikut ini pembagian fitur pada aplikasi *Damage Core Automation System* :

| *User* / *Menu* | *Mechanic* | *Storage Man* | *SDH & PDH* | *Master* |
|---|---|---|---|---|
| **Input data DC** | **Tersedia** | **Tidak Tersedia** | **Tidak Tersedia** | **Tersedia** |
| **Take-in DC** | **Tidak Tersedia** | **Tersedia** | **Tidak Tersedia** | **Tersedia** |
| **DC List** | **Tersedia** | **Tersedia** | **Tersedia** | **Tersedia** |
| **DC Approval** | **Tidak Tersedia** | **Tidak Tersedia** | **Tersedia** | **Tidak Tersedia** |
| **Take-out DC** | **Tidak Tersedia** | **Tersedia** | **Tidak Tersedia** | **Tersedia** |
| **Create user** | **Tidak Tersedia** | **Tidak Tersedia** | **Tidak Tersedia** | **Tersedia** |
| **QR Scanner** | **Tersedia** | **Tersedia** | **Tersedia** | **Tersedia** |

4.4 Pembuatan Database



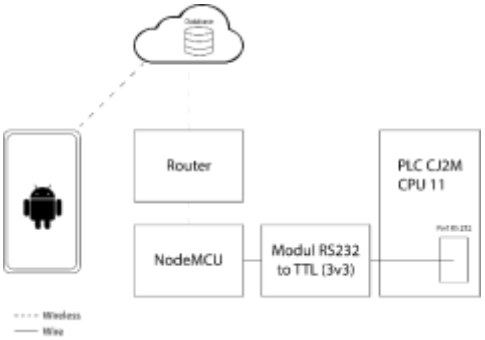

Sesuai dengan rencana perbaikan, *database* yang digunakan adalah Firebase *Real-time Database*. Struktur *database* yang digunakan dapat dilihat pada gambar diatas. Pada tabel *database* 'Arduino' berisi data yang akan dibaca oleh mikrokontroller. Data tersebut akan dikirimkan ke *PLC*.

4.5 Implementasi Program

Aplikasi yang dibuat menggunakan perangkat lunak Android Studio dengan bahasa pemrograman Java. Terdapat beberapa metode yang digunakan untuk membuat fitur-fitur yang ada pada aplikasi, berikut beberapa metode yang digunakan:

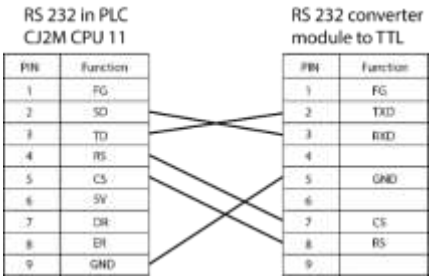
1. Otentikasi Pengguna
   Untuk proses otentikasi pengguna aplikasi penulis menggunakan kelas *FirebaseAuth*, dimana dalam kelas tersebut terdapat metode signInWithEmailAndPassword(String s1,String s2).
2. Mendapatkan Pengguna Aktif
   Untuk mendapatkan pengguna aktif dapat dilakukan dengan mengimplementasi kelas *Firebase Auth* dan *Firebase User*.
   **auth** = FirebaseAuth.*getInstance*();
   FirebaseUser fuser=**auth**.getCurrentUser();
3. Pembacaan dan Penulisan *Database*
   Menulis dan membaca Firebase *database* pada Android Studio menggunakan kelas Firebase *Database* dan *Database Reference*. Metode yang digunakan untuk menulis adalah setValue(value), sedangkan untuk membaca menggunakan get().
4. Kirim Notifikasi/Pemberitahuan
   Penulis menggunakan fitur *Firebase Cloud Messaging* dan juga memerlukan token dari setiap pengguna.

4.6 Pembuatan Sistem Komunikasi




Gambar diatas merupakan diagram blok komunikasi antara aplikasi android dengan PLC. Komunikasi yang digunakan yaitu berupa komunikasi serial melalui port RS232. Data yang diambil oleh mikrokontroler nodeMCU dari Firebase Database merupakan kode lokasi rak penyimpanan *DC*. Selanjutnya data tersebut akan diolah dan dikirimkan ke port RS232 pada *PLC* Omron CJ2M CPU11.

Pengiriman data dari nodeMCU ke PLC menggunakan sebuah modul *converter* RS232 to TTL. Caranya dengan menghubungkan pin RXD, TXD, 5V dan GND yang ada pada nodeMCU ke modul *converter* RS232 to TTL. Pada modul tersebut terdapat port RS232 *female* yang selanjutnya akan dihubungkan dengan kabel RS232 *male to male* ke *port* yang ada di *PLC*. Kabel untuk menghubungkan *PLC* ke modul RS232 to TTL memiliki konfigurasi sebagai berikut:




4.7 Pengujian Aplikasi

Pengujian aplikasi dilakukan untuk mengetahui apakah aplikasi sudah berjalan dengan baik atau belum. Berikut ini merupakan tabel pengujian dari aplikasi yang sudah dibuat

| No | Fitur | Hasil |
|---|---|---|
| 1 | *Sign-in* | Pengguna berhasil masuk ke dalam aplikasi menggunakan *email* dan *password* yang terdaftar |
| 2 | *Forgot Password* | Pengguna mendapatkan pesan *email* yang berisi tautan untuk mengubah *password* |
| 3 | *Online Input DC Data* | Foto berhasil dimasukan |
| | | Pengguna berhasil menyimpan data pada *database* secara *online* |
| 4 | *Take-in DC* | Aplikasi berhasil untuk mengisi lokasi secara otomatis |
| | | Aplikasi berhasil menyimpan data *DC* pada *database* secara *online* |

| | | |
|---|---|---|
| | | Aplikasi berhasil menyimpan lokasi rak pada *database* 'Arduino' |
| | | Aplikasi berhasil membuat *QR code* dan mengunduhnya dalam bentuk gambar pada memori internal ponsel |
| 5 | *DC List* | Aplikasi berhasil menampilkan data *DC* yang sudah tersimpan di rak |
| | | Aplikasi berhasil mengimport file PDF untuk *Part Return Request* |
| | | Aplikasi berhasil mengirim notifikasi kepada *PDH* dan *SDH* |
| 6 | *Approval* | Aplikasi yang dimiliki *PDH* dan *SDH* berhasil menerima notifikasi |
| | | Aplikasi berhasil menampilkan data *DC* yang hendak dikeluarkan |
| | | Aplikasi berhasil melakukan *online approval DC* |
| 7 | *Take-out DC* | Aplikasi berhasil menampilkan data *DC* yang sudah disetujui oleh *PDH* dan *SDH* |
| | | Aplikasi berhasil menyimpan lokasi rak pada database 'Arduino' |
| 8 | *QR Code Scanner* | Aplikasi berhasil menampilkan informasi *DC* sesuai dengan pembacaan QR code |

4.8 Pengujian Sistem Komunikasi

Pengujian sistem komunikasi dilakukan meliputi metode, program dan konfigurasi pengkabelan. Berikut ini tabel pengujian yang sudah dibuat:

| No | Unit Sistem Komunikasi | Hasil |
|---|---|---|
| 1 | NodeMCU | NodeMCU berhasil terhubung ke jaringan internet (WiFi) |
| | | NodeMCU berhasil mengakses Firebase *Realtime Database* |
| | | NodeMCU berhasil membaca data pada Firebase *Database* dan mengirimkan data lokasi rak penyimpanan ke *PLC* |
| 2 | *Modul RS 232* | Modul RS232 berhasil mengkomunikasikan data secara serial antara PLC dan nodeMCU |
| 3 | *PLC CJ2M CPU 11* | PLC Omron CJ2M CPU11 berhasil mengirim dan menerima data secara serial |

## V. KESIMPULAN

Setelah melakukan perencanaan serta pembuatan aplikasi android yang terintegrasi dengan AGV, penulis mendapat beberapa kesimpulan :

1. Perancangan dan pembuatan aplikasi android dibuat menggunakan perangkat lunak Android Studio dengan bahasa pemrograman Java. Aplikasi dipadukan dengan Firebase sebagai penyedia layanan milik Google. Terdapat 5 fitur utama yang mendukung proses penyimpanan dan pengeluaran *DC* pada aplikasi, yaitu *Input DC Data*, *Take-in DC*, *DC List*, *Approval DC* dan *Take-out DC*.
2. Aplikasi dapat melakukan komunikasi dengan PLC Omron CJ2M CPU11 melalui mikrokontroler nodeMCU dan modul RS232 *to TTL*. NodeMCU digunakan untuk membaca data aplikasi dan meneruskannya ke *PLC* secara serial menggunakan modul RS232.

## VI. DAFTAR PUSTAKA

[1] Lase, Knud. 2015. *IoT Basics: Getting started with the Internet of Things*. *IoT Analytics.*
[2] Simpson, Jonathan. 2018. *IT OT We speak both language, Industrial Networks*. Siemens USA.
[3] Adam Gerber dan Clifton Craig. 2015. *Learn Android Studio: Build Android Apps Quickly and Effectively*. Appres.
[4] Android. 2019. Dasar-dasar Aplikasi Android, https://developer.android.com/guide/components/fundamentals?hl=id, diakses 10 Juli 2019.
[5] Sumber Daya Aplikasi. 2019. Dasar-dasar Aplikasi Android, https://developer.android.com/guide/components/fundamentals?hl=id, diakses 10 Juli 2019.
[6] Mikrotik. 2019. Pengertian Jaringan Komputer, http://www.mikrotik.co.id/artikel_lihat.php?id=67, diakses 21 Juni 2019.
[7] NodeMCU, 2019, Pertemuan Pertama: Mengenal NodeMCU, https://embeddednesia.com/v1/tutorial-nodemcu-pertemuan-pertama/, diakses 21 Juni 2019.
[8] OMRON, 2008, CJ2 Unit Hardware: User's Manual.