Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202072337
Vol. 7, No. 5, Oktober 2020, hlm. 1011-1020 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

IMPLEMENTASI QR CODE BERBASIS ANDROID PADA SISTEM PRESENSI

Afif Priyambodo[1], Koredianto Usman[2], Ledya Novamizanti*[3]

[1,2,3]Teknik Telekomunikasi, Universitas Telkom
Email: [1]afifpriyambodo@gmail.com, [2]korediantousman@telkomuniversity.ac.id,
[3]ledyaldn@telkomuniversity.ac.id
*Penulis Korespondensi

(Naskah masuk: 10 Agustus 2019, diterima untuk diterbitkan: 07 Oktober 2020)

Abstrak

Presensi merupakan hal utama dalam suatu kegiatan, karena menjadi bukti dari laporan pelaksanaan. Umumnya, presensi kehadiran dilakukan secara manual, yaitu siswa membubuhkan tanda tangan pada daftar hadir yang diedarkan, atau guru memanggil siswa satu persatu. Namun metode tersebut mengakibatkan terjadinya pemborosan waktu dan sumber daya. Hadirnya teknologi QR-Code berbasis android memberikan solusi agar presensi dapat berjalan dengan efisien. Penelitian ini memiliki tiga konfigurasi sistem, diantaranya sistem encoder, sistem hardware, dan sistem decoder. Sistem encoder melakukan proses encode data berupa Nomor Induk Siswa Nasional (NISN) menjadi QR-Code menggunakan kode Bose, Chaudhuri, Hocquenghem (BCH). Sistem hardware terdiri dari perangkat android dan kartu pelajar. Sistem decoder melakukan proses deteksi QR-Code dengan aplikasi Smart Presence. Sistem diuji dengan pengujian black box, pengujian jarak deteksi, pengujian deteksi berdasarkan cahaya, serta pengujian kartu pelajar bernoda dan rusak. Sistem presensi mampu mendeteksi QR-Code dengan jarak minimal sebesar 3 cm dan jarak maksimal sebesar 45 cm dengan tingkat akurasi sebesar 98 % dan rata-rata waktu komputasi sebesar 1,3 detik.

Kata Kunci: *sistem presensi, android, QR-code, kode BCH*

IMPLEMENTATION OF ANDROID-BASED QR CODE IN THE PRESENCE SYSTEM

Abstract

Presence is the main thing in an activity because it becomes evidence of the implementation report. Generally, attendance is done manually, i.e. students sign on the circulated attendance list, or the teacher calls students one by one. However, this method resulted in a waste of time and resources. The presence of Android-based QR-Code technology provides a solution so that the presence can run efficiently. This research has three system configurations, including the encoder system, hardware system, and decoder system. The encoder system encodes data in the form of a National Student Number (NISN) into a QR-Code using the Bose, Chaudhuri, Hocquenghem (BCH) codes. The hardware system consists of an Android device and a student card. The decoder system carries out the QR-Code detection process with the Smart Presence application. The system was tested with black-box testing, detection distance testing, light-based detection testing, and stained and damaged student card testing. The presence system is able to detect QR-Code with a minimum distance of 3 cm and a maximum distance of 45 cm with an accuracy rate of 98% and an average computing time of 1.3 seconds.

Keywords: *presence system, android, QR-code, BCH code.*

1. PENDAHULUAN

Presensi digunakan sebagai bukti kehadiran dari suatu pelaksanaan kegiatan. Sistem presensi yang banyak diterapkan saat ini adalah presensi secara manual, yaitu siswa membubuhkan tanda tangan di kertas yang diedarkan, atau guru memanggil siswa satu persatu di awal pertemuan. Mengambil dan memeriksa kehadiran peserta oleh penyelenggara di setiap kegiatan adalah proses yang memakan waktu terutama ketika kelas besar. Hal tersebut juga menimbulkan pemborosan kertas dan kurang efisien.

Berbagai penerapan teknologi untuk sistem presensi sudah dilakukan, seperti penggunaan teknologi barcode, RFID, fingerprint, wajah, dan biometrik lainnya. Masalha dan Hirzallah mengusulkan sistem presensi yang menampilkan kode QR ke siswa selama atau di awal setiap kuliah. Para siswa perlu memindai kode untuk mengkonfirmasi kehadiran mereka (Masalha, Hirzallah, 2014). Soleh dan Muharom memanfaatkan *smartphone* untuk melakukan presensi ujian secara online. Nomor Ujian dan NIM mahasiswa ditampilkan menggunakan QR-Code yang telah

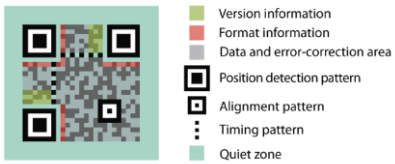
tercetak di HP. Kemudian, pengawas ujian melakukan scanning QR-Code di HP mahasiswa secara satu persatu. Penelitian Norhikmah menggunakan QR Code untuk presensi pelatihan berbasis android. Waktu yang dibutuhkan yaitu 15 detik/idcard. Sedangkan presensi dengan membubuhkan tanda tangan di kertas membubuhkan waktu rata-rata 30 detik/ orang (Noerhikmah, dkk, 2016).

Makalah ini mengusulkan sistem presensi menggunakan Kode QR dengan *smartphone* berbasis android. Kode *QR* dibangkitkan dengan menggunakan program MATLAB, dan di cetak di kartu pelajar. Pengembangan berikutnya, kartu pelajar ini dapat digunakan untuk sistem parkir, transaksi di ATM, eToll, dan keperluan lainnya. Penggunaan kode *Bose, Chaudhuri, Hocquenghem* (*BCH*) sebagai *error correction* menghasilkan sistem yang handal, karena dapat mengoreksi kesalahan ganda (*multiple error correction*). Dengan teknik ini, sistem tidak hanya akan menghemat waktu tetapi juga akan mempercepat proses rekapitulasi kehadiran, mengurangi penggunaan kertas, dan banyak waktu untuk pemaparan materi maupun aktifitas lainnya.

2. METODE PENELITIAN

2.1. *QUICK RESPONSE* (QR) *CODE*

QR-Code adalah jenis simbol dua dimensi yang dikembangkan oleh Denso Wave pada tahun 1994. Setiap simbol *QR-Code* disusun dalam bentuk persegi dan terdiri dari *function patterns* dan *encoding region*. Seluruh simbol dikelilingi oleh batas *quiet zone* pada keempat sisi. Terdapat 4 jenis pola fungsi meliputi *finder pattern, separators, timing patterns*, dan *alignment patterns. Encoding region* berisi data, yang mewakili informasi versi, format informasi, data dan koreksi kesalahan.



Gambar 1. Struktur *QR-Code*

2.2. KODE *BCH*

Kode *BCH* memiliki fungsi mengoreksi kesalahan ganda (*multiple error corretion)*. Kode *BCH* membentuk kode siklik besar yang dapat mengoreksi kesalahan acak dengan kuat. Langkah awal kode *BCH* adalah pembentukan kumpulan *checkbit* yang akan dikirimkan bersama informasi. Adapun proses pembentukan *checkbit* sebagai berikut:

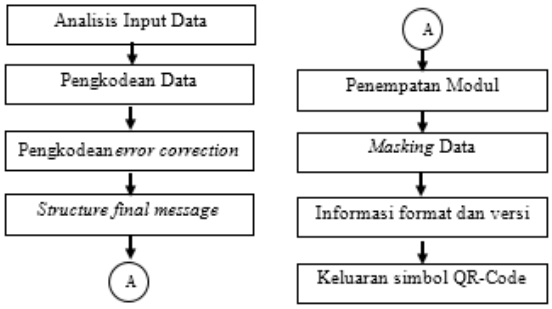
1. Penentuan parameter *BCH Code*.
   Parameter digunakan sebagai input dalam proses *encode-decode BCH Code.* Parameter tersebut meliputi :
   - Variabel m
   - Panjang blok yang dikirim : $n = 2^m - 1$
   - Panjang *bit* informasi : k
   - *Error capability* : t
   - *Checkbit* : $c = m \cdot t$ (ketentuan $n - k \leq mt$)
2. Bentuk *Galois Field, GF* (2^m).
3. Menentukan *minimal polynomial* sebanyak $2t - 1$ buah.
4. Bentuk *Generator polynomial* $(g(x))$.
5. *Bit* informasi telah terbentuk, tambahkan *bit* 0 dibelakangnya sebanyak *checkbit* (c).
6. Gabungkan *bit* biner dengan *bit* 0 dibagi dengan *bit* $g(x)$, diperoleh *checkbit* $(v(x))$.
7. Data informasi $(V(x))$ yang akan dikirim berupa *bit* informasi dan *checkbit*.

Decode BCH Code merupakan proses *error detection* dan *error correction*. *Decode BCH Code* mendeteksi *error* serta mengoreksi jika *error* ditemukan. Data *bit* yang didapatkan pada proses *encode BCH code* menjadi data input pada proses *decode BCH Code.* Adapun proses *decode BCH Code* sebagai berikut :

1. Data *bit* yang akan dikirim $(V(x))$ dibagi dengan *generator polynomial* $(g(x))$. Jika hasil pembagian = 0, maka tidak ada error. Sebaliknya, jika hasil pembagian = 0, berarti terdapat *error* dan harus melalui proses koreksi.
2. Menentukan nilai *minimal polynomial* $(2t)$.
3. Menghitung *syndrome* dari *codeword* $(S_1, \dots, S_{2t})$.
4. Membentuk Tabel BCH (algoritma *Peterson-Berlekamp*).
5. Hasil dari Tabel BCH $(\sigma^{(n)}(x))$ adalah polinomial yang berfungi mendeteksi lokasi jika terdapat *error.*
6. Menentukan akar persamaan polinomial dengan metode *trial* dan *error.*
7. Menentukan nilai kebalikan dari akar persamaan polinomial. Nilai tersebut merupakan posisi dari *bit error*.

2.3. *ENCODING QR-CODE*

Proses *Encode QR-Code* merupakan proses mengubah data masukan menjadi simbol *QR-Code*. Selain itu, proses *encoding* menentukan spesifikasi dari *QR-Code*. Pengkodean *error correction* dilakukan pada proses tersebut.



Gambar 2. Tahapan *encoding QR-Code*

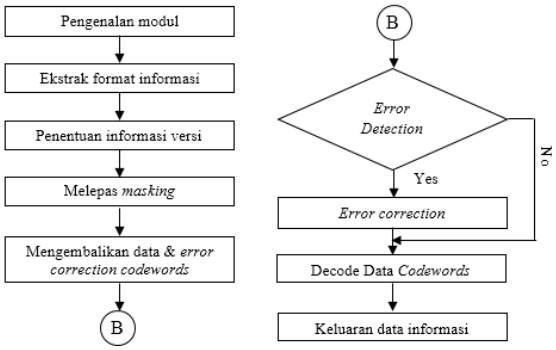
Proses diawali dengan analisis input data. Selanjutnya data dikodekan menjadi *bit.* Pada proses *encoding* dilakukan konfigurasi spesifikasi *QR-Code* berupa format dan versi. *T*ahapan pada proses *encoding QR-Code* sebagai berikut:

1. Analisis data
   QR-Code mengkodekan serangkaian teks dengan empat mode yaitu *numeric, alphanumeric, byte,* dan kanji. Setiap metode pengkodean dioptimalkan untuk mengkodekan data dengan sekumpulan *bit* sesingkat mungkin. Oleh karena itu analisis data digunakan untuk mengetahui apakah data yang dikodekan dalam mode *numeric, alphanumeric, byte,* atau kanji.
2. Pengkodean data
   Data diubah menjadi serangkaian *bit* yang dipecah menjadi data *codewords* yang masing-masing memiliki 8 *bit*.
3. Pengkodean *error correction*
   Pengkodean menghasilkan *string bit* data sebagai *error correction codewords.* Pemindai *QR-Code* membaca *codewords data* dan *error correction codewords*. Dengan membandingkan keduanya, pemindai dapat menentukan benar atau tidak dalam melakukan pemindaian. Jika pemindai tidak membaca dengan benar, maka dapat memperbaiki kesalahan.
4. *Structure Final Message*
   Mengatur data dan *error correction codewords* yang dihasilkan. Data dan *error correction* disisipkan sesuai dengan spesifikasi *QR-Code.*
5. Penempatan modul
   Proses menempatkan *bit* pada matriks *QR-Code. Codewords* disusun dalam matrik dengan cara tertentu.
6. *Masking* data
   Mengubah pola modul sesuai spesifikasi *QR-Code.* Spesifikasi mendefinisikan delapan *mask pattern.*
7. Informasi format dan versi
   Format dan informasi versi ditambahkan ke dalam *QR-Code* dengan menambahkan piksel pada area kode tertentu. Format piksel mengidentifikasi tingkat *error correction* dan *mask pattern* yang digunakan dalam *QR-Code.*

2.4. *DECODING QR-CODE*

Decoding QR-Code merupakan proses mengubah simbol *QR-Code* menjadi data informasi. Tahapan dalam melakukan proses *decode QR-Code* sebagai berikut:

1. Pengenalan modul
   Mengenali modul gelap dan terang sebagai deretan *bit* "0" dan "1" dengan mencari dan mendapatkan gambar simbol.
2. Ekstrak format informasi
   Decode format informasi, melepaskan pola *masking* dan menerapkan *error correction.*
3. Penentuan informasi versi
   Menentukan versi simbol *QR-Code*.
4. Melepas *masking*
   MenXORkan *decoding region bit* dengan pola *mask* yang referensinya telah diekstras dari format informasi.
5. Mengembalikan data dan *error correction*
   Mengubah simbol menjadi *bit* data dan *bit error correction codewords.*
6. *Error detection* dan *error correction*
   Mengidentifikasi *error* dan memperbaiki dengan memanfaatkan *error correction codewords.*
7. Decode Data *Codewords*
   Membagi data *codewords* menjadi beberapa segmen sesuai indikator mode dan indikator jumlah karakter. *Decode* karakter data sesuai mode yang digunakan dan *output* teks yang diterjemahkan sebagai hasilnya.

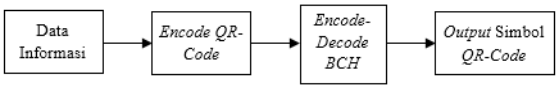


Gambar 3. Proses *decoding QR-Code*

3. HASIL DAN PEMBAHASAN

3.1. SISTEM *ENCODER*

Sistem *encoder* merupakan proses pengkodean data informasi menjadi QR-Code. Proses *encode QR-Code* dilakukan dengan melibatkan kode *BCH* sebagai *error correction.*

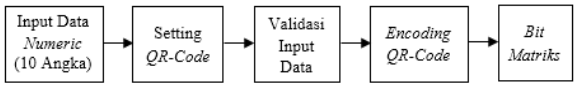


Gambar 4. Blok Diagram Sistem Encoder

Data informasi berupa Nomor Induk Siswa Nasional (NISN) menjadi input pada sistem *encoder*. *Encode QR-Code* merupakan proses pengkodean data menjadi simbol *QR-Code.* Hasil dari proses *encode* menjadi inputan proses *encode-decode BCH Code.* Sistem *encoder* menghasilkan keluaran berupa simbol *QR-Code.*

3.1.1 *ENCODE QR-CODE*

Encode QR-Code mengkodekan data menjadi *bit matriks.* Sebelum itu, sistem mengkonfigurasi *QR-Code* sesuai spesifikasi yang diinginkan. Gambar 5 menunjukan proses *encode QR-Code.*

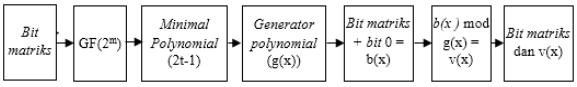


Gambar 5. Blok diagram *encode QR-Code*

Input data berupa text mode *numeric* berupa sepuluh angka (NISN). Kemudian, sistem melakukan konfigurasi *QR-Code*. Konfigurasi tersebut berpengaruh terhadap format *QR-Code.* Pada penelitian ini, paramater konfigurasi *QR-Code* meliputi $characterset = UTF\ 8$, $Versi\ QR - Code = 1$, $initial\ size = 117$ dimana merupakan ukuran citra *QR-Code*, $quitezone\ size = 4$ yaitu menambahkan *quite zone* pada *QR-Code*,$size = [initial\ size + 4 \cdot versi\ QR - Code, initial\ size + 4 \cdot versi\ QR - Code]$ sehingga $size = [117 + 4 \cdot 1, 117 + 4 \cdot 1]$ atau [121 121] dimana merupakan ukuran *QR-Code*. Selanjutnya, input data divalidasi untuk mengetahui mode input berupa *numeric, alphanumeric, byte,* atau kanji. Proses selanjutnya, input data dikodekan menjadi serangkaian *bit* yang dipecah menjadi *codewords* sesuai dengan konfigurasi *QR-Code. Bit-bit* tersebut dipetakan menjadi *bit matriks* dengan ukuran [121 121].

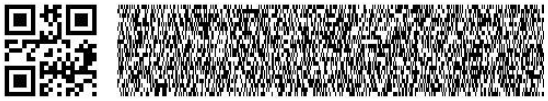
3.1.2 *ENCODE BCH CODE*

Setelah mendapatkan *bit matriks,* proses selanjutnya adalah *error correction coding.* Penelitian menggunakan *BCH Code* sebagai *error correction code*. *Encode BCH Code* membentuk *checkbit* yang akan dikirim dengan *bit matrik*s. Gambar 6 menunjukan proses *encode BCH Code.*



Gambar 6. Blok diagram *encode BCH Code*

Hal pertama yang dilakukan adalah membentuk *Galois Field array* (2^m) dengan parameter yaitu $m = 9, n = 2^m - 1 = 511\ bit$ dimana n adalah *codeword length,* $k = 121$ dimana *k*adalah panjang input, $t = 58$ adalah kemampuan koreksi kesalahan, dan $c = m \cdot t = 522\ bit$ dimana c adalah *checkbit*. Setelah melalui proses pembentukan *GF*, didapatkan nilai $m = 9$ dan $primpoly = 529$. Selanjutnya, menentukan *minimal polynomial* $(2t - 1)$. Untuk $t = 58$, terdapat 115 buah minimal polinomial, yaitu $(m_1(x), \dots, m_{115}(x))$. Kemudian membentuk *generator polynomial* $(g(x))$ dengan perhitungan $g(x) = KPK(m_1(x), \dots, m_{115}(x))$. Langkah selanjutnya, menentukan $b(x)$ dimana merupakan *bit matriks* ditambah *bit* 0 sebanyak 522 buah. Hitung nilai *checkbit* $(v(x))$ dengan rumus $b(x)\ mod\ g(x)$.

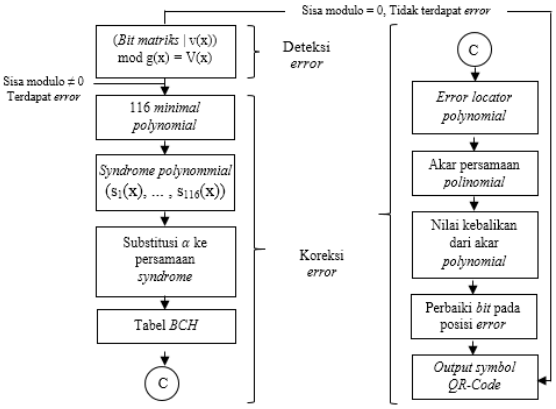


Gambar 7. Contoh *bit matriks* dan *checkbit* (*v(x)*)

Bit matriks dan *checkbit* merupakan hasil proses *encode BCH Code.* Gambar 7 menunjukan *bit matriks* dan *checkbit* $(v(x))$. Hasil tersebut kemudian menjadi inputan pada proses *decode BCH Code.*

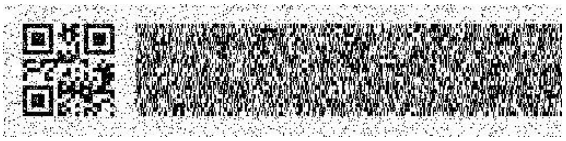
3.1.3 *DECODE BCH CODE*

Decode BCH Code merupakan proses deteksi *error* dan koreksi *error*. *Bit matriks* dan *checkbit* menjadi inputan pada proses ini. *Bit matriks* dan *checkbit* dicek terlebih dahulu agar diketahui apakah terdapat *error* atau tidak. Gambar 8 menunjukan diagram alir proses *decode BCH Code.*



Gambar 8. Diagram alir *decode BCH Code.*

Proses pertama pada *decode BCH Code* adalah cek apakah terdapat *error* atau tidak. Pengecekan tersebut dilakukan dengan menghitung $bit\ matriks \mid checkbit\ (v(x)) mod\ g(x)$. Hasil *mod* dinamakan $V(x)$. Jika $V(x) = 0$, maka tidak terdapat *error* dan *bit matriks* dikonversi menjadi *barcode format QR-Code.* Jika $V(x) \neq 0$, maka terdapat *error* kemudian melanjutkan proses koreksi *error*.



Gambar 9. Contoh sisa modulo $V(x) \neq 0$

Gambar 9 menunjukan nilai $V(x) \neq 0$, artinya *bit matriks* terdapat *error.* Langkah selanjutnya menghitung nilai *minimal polynomial* $(2t)$ yang berjumlah 116. Kemudian hitung *syndrome* dari 116 *minimal polynomial* $(V(x)\ mod\ m_1(x), \dots, V(x)\ mod\ m_{116}(x))$. Selanjutnya, substitusi α ke persamaan tiap *syndrome*. Bentuk Tabel *BCH* dengan algoritma *Peterson-Berlekamp*. Hasil dari tabel tersebut merupakan polinomial pendeteksi lokasi *error* $(\sigma(x))$. Cari akar dari polinomial pendeteksi lokasi *error* dengan mengganti nilai $\sigma(x)$. Persamaan polinomial lokasi *error* dengan *field element* pada GF (2^m), (α^k), dengan ketentuan $0 \leq k \leq 511$, dan

$\sigma(\alpha^k) = 0$). Posisi *bit error* adalah nilai kebalikan dari akar yang diperoleh. Selanjutnya, perbaiki *bit* pada posisi yang *error* dengan membalikan nilai *bit*. *Bit matriks* berhasil diperbaiki dan dikonversi menjadi *barcode format QR-Code*. Gambar 10 menunjukan keluaran simbol *QR-Code* yang merupakan hasil proses sistem *encoder*.

Gambar 10. Keluaran simbol QR-Code

3.2. SISTEM *HARDWARE*

Smartphone berbasis *android* digunakan sebagai alat *scanning* pada proses deteksi *QR-Code.* Selain itu, penggunaan kartu pelajar dimana terletak *QR-Code* itu sendiri berfungsi sebagai media dalam proses deteksi *QR-Code*. Objek penelitian yaitu pada SMA Negeri 1 Kebumen. Desain kartu pelajar disesuaikan dengan format pada umumnya. Data siswa terletak pada bagian depan kartu pelajar. Simbol *QR-Code* terletak pada bagian belakang kartu pelajar.

(b)

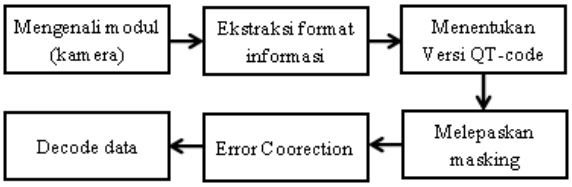
Gambar 11. Desain Kartu Pelajar (a) tampak depan, (b) tampak belakang.

3.3. SISTEM *DECODER*

Sistem *decoder* merupakan proses deteksi *QR-Code* menjadi data informasi. Perangkat *smartphone* digunakan sebagai alat pendeteksi *QR-Code* dengan aplikasi berbasis *android.* Aplikasi sistem presensi ini didesain menggunakan Android Studio sebagai aplikasi pendeteksi *QR-Code.* Selain itu, sistem *decoder* menggunakan kartu pelajar sebagai media dalam proses deteksi *QR-Code.*

3.3.1 DETEKSI *QR-CODE*

Deteksi *QR-Code* mengubah *QR-Code* menjadi data informasi. Secara garis besar, proses deteksi diawali dengan identifikasi *QR-Code.* Kemudian, sistem mengidentifikasi konfigurasi *QR-Code*. Proses selanjutnya, sistem mendekode *QR-Code* menjadi data informasi.

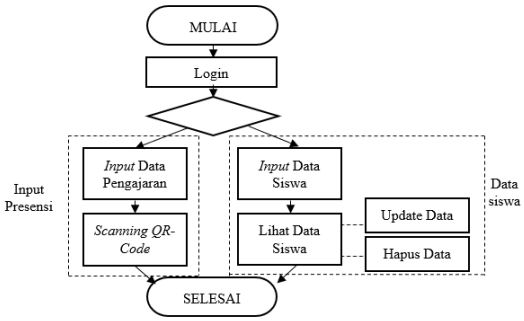


Gambar 12. Diagram alir deteksi *QR-Code*

Kamera pada *smartphone* mengenali modul gelap dan terang sebagai deretan *bit* "0" dan "1" dengan cara *scanning* gambar *QR-Code*. Selanjutnya, sistem mengidentifikasi konfigurasi *QR-Code.* Identifikasi *QR-Code* mendapatkan versi *QR-Code* dan format informasi. Format informasi diekstrak agar mendapatkan pola *masking* dan *error correction.* Sistem mengidentifikasi *error* dan memperbaiki jika terdapat *error.* Keluaran dari proses deteksi *QR-Code* berupa data informasi.

3.3.2 APLIKASI BERBASIS *ANDROID*

Aplikasi presensi berbasis *android* bernama *Smart Presence* melakukan proses deteksi *QR-Code* menjadi data informasi.

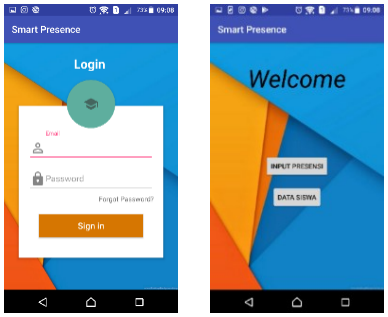


Gambar 13. Diagram alir *Smart Presence*.

Pengajar melakukan proses *login* agar mendapatkan *permission* pada aplikasi *Smart Presence*. Setelah *login,* akan ada dua pilihan yaitu *input* presensi dan data siswa. Jika memilih *input* presensi, proses selanjutnya adalah mengisi data pengajaran meliputi nama pengajar, mata pelajaran, kelas, hari dan tanggal. Kemudian melakukan proses *scanning QR-Code* pada kartu pelajar agar mendapatkan data informasi dari *QR-Code.* Jika memilih data siswa, maka akan muncul *form database* dimana berisi data siswa yang dapat membuat, melihat, menghapus dan merubah data.

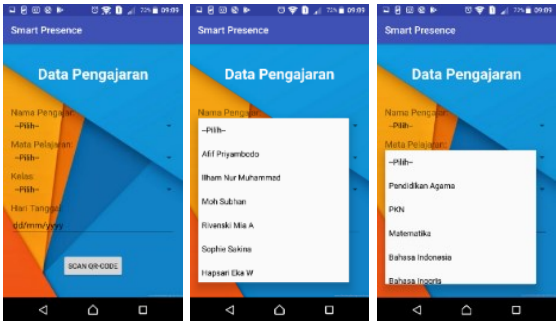
3.3.3 DESAIN APLIKASI *SMART PRESENCE*

Desain aplikasi *Smart Presence* dilakukan dengan program Android Studio. Aplikasi tersebut terintegrasi dengan *Firebase database*. Gambar 14 merupakan tampilan awal aplikasi.

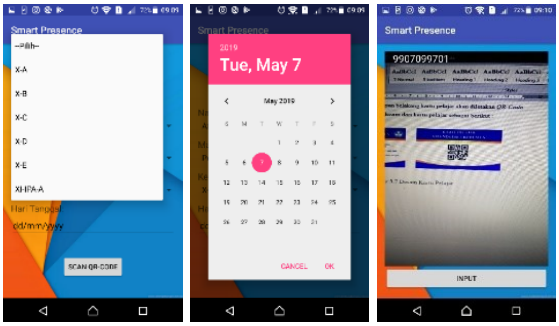


Gambar 14. Tampilan (a) login, (b) awal

Sistem terintegrasi dengan *Firebase database* yang berfungsi sebagai *authentication* dan *database* pada aplikasi *Smart Presence*. Admin akan mendaftarkan *email* dan *password* pada *firebase.* Pengajar melakukan *login* sesuai *email* dan *password* yang didapatkan.



(a) (b) (c)

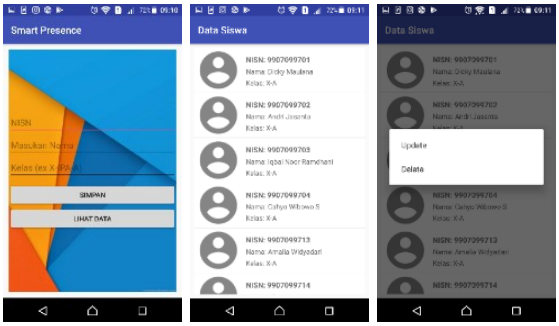


(d) (e) (f)

Gambar 15. Tampilan (a) Data pengajaran, (b) Nama pengajar, (c) Mata pelajaran, (d) Kelas, (e) Tanggal.

Tampilan awal terdiri dari dua pilihan yaitu input presensi dan data siswa. Input presensi dipilih jika akan melakukan proses penginputan siswa dalam presensi di kelas. Sedangkan data siswa dipilih jika akan menginput, melihat, mengupdate dan menghapus data siswa.

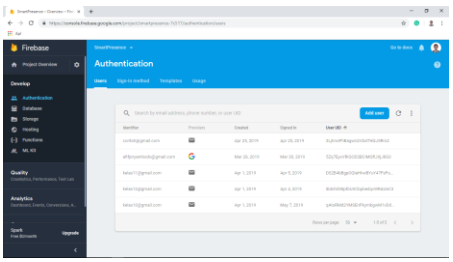
Tampilan data pengajaran berisi informasi mengenai nama pengajar, mata pelajaran, kelas dan tanggal pelaksanaan belajar-mengajar. Kartu pelajar di *scan* dan di *decode* menjadi data informasi berupa NISN. Data informasi ditampilkan pada *layout* ini. Aplikasi meminta *user-permission* untuk menggunakan kamera pada *smartphone.*



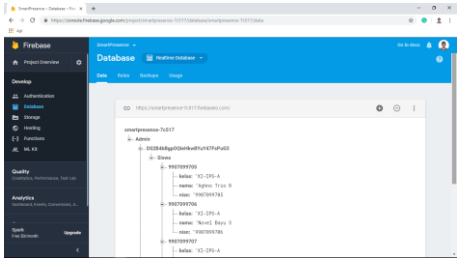
(a) (b) (c)

Gambar 16. Tampilan Database (a) Input data, (b) Lihat data, (c) Hapus data.

Tampilan *database* berisi perintah membuat, melihat, memperbaharui, dan menghapus data siswa. Aplikasi *Smart Presence* menggunakan *firebase* sebagai *database* dan *authentication login*. Setiap *user* memiliki *database* masing-masing.



(a)



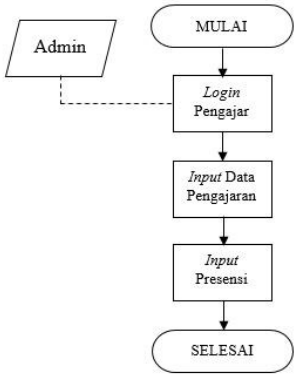
(b)

Gambar 17. Firebase (a) Authentication, (b) Database.

3.3.4 IMPLEMENTASI SISTEM PRESENSI

Penelitian ini mensimulasikan hasil dari konfigurasi sistem. Gambar 18 merupakan diagram alir dari sistem presensi di sekolah. Pengajar melakukan *login* pada aplikasi *smart presence* dengan mengisi *email* dan *password.* Admin memberikan *email* dan *password* kepada pengajar untuk melakukan *authentication login* pada aplikasi *Smart Presence*. Admin sudah terlebih dahulu mendaftarkan akun pada *firebase authentication*. Setelah pengajar mendapatkan *permission login*, pengajar melakukan *input* data pengajaran berupa

nama pengajar, mata pelajaran, kelas, dan hari tanggal pengajaran. Kemudian pengajar melakukan *scanning* kartu pelajar (*QR-Code*) untuk mendapatkan NISN siswa. Hasil *scanning* akan dicocokan terhadap *database* siswa dan ditampilkan ke *web* sekolah sebagai data presensi.



Gambar 18. Diagram alir sistem presensi sekolah.

3.4 PENGUJIAN SISTEM

Pengujian dilakukan untuk mengetahui performansi sistem. Skenario dirancang untuk melakukan beberapa pengujian terhadap aplikasi dan sistem presensi. Adapun parameter pengujian sebagai berikut :

1. Pengujian *Black box*, bertujuan untuk menguji fungsionalitas dari setiap tampilan pada aplikasi *Smart Presence*.
2. Pengujian jarak deteksi *QR-Code*,bertujuan untuk mengetahui jarak minimum dan jarak maksimum pada proses deteksi QR-Code. Jarak yang digunakan pada pengujian adalah 1 cm sampai dengan diluar jangkauan deteksi QR-Code. Pengujian dilakukan terhadap 10 kartu pelajar pada setiap jarak yang ditentukan.
3. Pengujian pengaruh cahaya dalam mendeteksi *QR-Code*, bertujuan untuk mengetahui keberhasilan deteksi QR-Code dalam berbagai intensitas cahaya. Sumber cahaya berasal dari kamera *flash* pada *smartphone*. Terdapat tiga kategori pengujian, yaitu intensitas cahaya yaitu ringan (1.856 lux), sedang (5.283 lux), dan berat (10.470 lux). Pengujian dilakukan terhadap 10 kartu pelajar pada setiap kategori.
4. Pengujian pengaruh noda dan rusak pada *QR-Code* kartu, dimana kartu pelajar diberi noda dan dilakukan proses deteksi *QR-Code.* Terdapat tiga parameter pengujian yaitu, pengujian kotor tinta, pengujian kotor lumpur, dan pengujian gesekan atau goresan.

3.5 HASIL PENGUJIAN SISTEM

3.5.1 PENGUJIAN *BLACK BOX*

Pengujian *black box* dilakukan dengan fokus pada hasil keluaran yang diharapkan dari sistem yang diuji, apakah dapat berjalan sesuai yang diharapkan atau tidak. Tabel 1 merupakan hasil pengujian black box.

Tabel 1. Pengujian *black box*.

No.	Skenario Pengujian	Hasil yang diharapkan	Hasil Pengujian
1	Autentifikasi Password	Jika data benar maka tampilan menuju Layout utama. Jika data salah, maka terdapat info “email tidak valid”/ "wrong password" di layar	Berhasil
2	Memilih tombol Nama Pengajar, Mata Pelajaran, Kelas, dan Jadwal Kelas	Dapat menampilkan data *spinner* Nama Pengajar, Mata Pelajaran, Kelas, dan Jadwal Kelas	Berhasil
3	*Scanning QR-Code*	Jika data benar maka aplikasi melakukan *scanning QR-Code* dan mendapatkan NISN. Jika data salah maka *Text view* menampilkan info “No QR-Code Detected”	Berhasil
4	Mengubah data siswa	Dapat mengupdate data dan menampilkan info “Data berhasil diubah”	Berhasil
5	Menghapus data siswa	Dapat menghapus data siswa dan menampilkan info “Data Berhasil dihapus	Berhasil
6	Menampilkan data siswa yang tersimpan	Dapat menampilkan data siswa	Berhasil
7	Data yang tersimpan masuk ke dalam database server	Dapat terkoneksi ke database server	Berhasil

3.5.2 PENGUJIAN JARAK DETEKSI QR-CODE

QR-Code dideteksi dengan aplikasi *Smart Presence* berdasarkan jarak yang ditentukan. Data yang diambil berupa waktu komputasi, jarak minimum dan jarak maksimum. Waktu komputasi dinilai berdasarkan waktu deteksi QR-Code menjadi data informasi. Tabel 2 merupakan hasil pengujian berdasarkan hjarak deteksi QR-Code.

Tabel 2. Pengujian jarak deteksi QR-Code.

Jarak Deteksi *QR-Code* (cm)	Waktu Komputasi (detik)	Hasil Pengujian
1	-	Tidak berhasil
2	-	Tidak berhasil
3	2.10	Berhasil
4	2.04	Berhasil
5	1.07	Berhasil
10	0.85	Berhasil
15	0.89	Berhasil
20	1.07	Berhasil
25	1.07	Berhasil
30	1.12	Berhasil
35	1.15	Berhasil
40	1.20	Berhasil
45	1.31	Berhasil
50	-	Tidak berhasil
Rata-rata	**1.26**	**-**

Berdasarkan Tabel 1, pengujian jarak deteksi *QR-Code* memiliki jarak minimal sebesar 3 cm dan

jarak maksimal 45 cm. Pengujian menghasilkan waktu komputasi rata-rata sebesar 1,26 detik. Data hasil menunjukkan semakin jauh jarak deteksi semakin besar waktu komputasinya. Namun, pada jarak 10 cm hingga 1 cm, waktu komputasi semakin besar. Hal tersebut didasari oleh berkurangnya fokus pada kamera *smartphone* dalam mendeteksi *QR-Code.* Semakin dekat jarak deteksi, kamera semakin tidak fokus dalam mendeteksi *QR-Code.*

3.5.3 PENGUJIAN PENGARUH CAHAYA PADA DETEKSI QR-CODE

Dilakukan variasi intensitas cahaya saat pengambilan gambar (scanning) *QR-Code.* Performansi sistem berupa tingkat akurasi dan waktu komputasi.

Tabel 3. Pengujian deteksi QR-Code berdasarkan cahaya.

Intensitas Cahaya	Akurasi (%)	Waktu Komputasi (detik)
Ringan	100	0,89
Sedang	100	1,267
Berat	100	2,257
Rata-rata	**100**	**1,471**

Berdasarkan Tabel 3, pengujian deteksi *QR-Code* dengan gangguan cahaya memiliki akurasi 100% dan rata-rata waktu komputasi sebesar 1,471 detik. Waktu komputasi terkecil ada pada intensitas cahaya kategori ringan. Waktu komputasi terbesar ada pada intensitas cahaya kategori berat. Intensitas cahaya berpengaruh dalam proses deteksi *QR-Code.* Semakin besar intensitas cahaya yang diberikan, semakin besar waktu komputasi deteksi.

3.5.4 PENGUJIAN KOTOR TINTA

Pengujian menggunakan tinta spidol dengan lima warna, yaitu merah, hitam, biru, hijau dan kuning. Terdapat tiga kategori rusak, yaitu ringan (3 goresan), sedang (5 goresan) dan berat (lebih dari 5 goresan). Setiap warna tinta, dilakukan pengujian terhadap sepuluh kartu pelajar di setiap kategori. Total pengujian sebanyak 150 kartu pelajar.

(a) (b) (c)

Gambar 19. Pengujian kotor tinta merah (a) Ringan, (b) Sedang, (c) Berat.

Berdasarkan Tabel 4, pengujian akibat kotor tinta memiliki akurasi 97% dan rata-rata waktu komputasi sebesar 1,485 detik. Waktu komputasi terkecil, yakni 0,913 detik, pada kondisi tinta warna kuning tingkat ringan.

Tabel 4. Pengujian kotor tinta.

Warna Tinta	Tingkat	Akurasi (%)	Waktu komputasi (detik)
Hitam	Ringan	100	1,527
	Sedang	100	2,174
	Berat	70	2,755
Merah	Ringan	100	1,017
	Sedang	100	1,065
	Berat	100	1,137
Biru	Ringan	100	1,524
	Sedang	100	2,082
	Berat	90	2,615
Hijau	Ringan	100	1,124
	Sedang	100	1,171
	Berat	100	1,327
Kuning	Ringan	100	0,913
	Sedang	100	0,916
	Berat	100	0,934
Rata-rata		**97**	**1,485**

Sedangkan rata-rata waktu komputasi terbesar, yakni 2,755 detik, pada kondisi tinta warna hitam tingkat berat. Warna tinta berpengaruh terhadap hasil waktu komputasi dan tingkat akurasi. Semakin gelap warna tinta semakin besar waktu komputasinya. Tinta warna hitam menghasilkan data waktu komputasi terbesar (tingkat ringan, sedang, dan berat) dibanding dengan warna lain. Kemudian tinta warna biru menghasilkan data terbesar kedua, dilanjut dengan tinta warna hijau, merah, dan kuning. Tingkat akurasi terendah, yakni 70% terjadi pada pengujian tinta hitam tingkat berat. Hal ini disebabkan tinta hitam membuat pola warna gambar pada *QR-Code* menjadi berubah dan timpang tindih yang menyebabkan kesalahan pada proses *scanning QR-Code.*

3.5.5 PENGUJIAN KOTOR LUMPUR

Pengujian dilakukan dengan cara menambahkan lumpur ke kartu pelajar (QR-Code) kemudian diratakan. Terdapat 3 kategori noda, yaitu ringan (1 tetes), sedang (2 tetes) dan berat (3 tetes). Pengujian dilakukan terhadap 10 kartu pelajar di setiap kategori. Total pengujian sebanyak 30 kartu pelajar.

(a) (b) (c)

Gambar 20. Pengujian kotor lumpur (a) Ringan, (b) Sedang, (c) Berat.

Tabel 5. Pengujian kotor lumpur.

Takaran Lumpur	Akurasi (%)	Waktu Komputasi (detik)
Ringan	100	0,89
Sedang	100	1,137
Berat	100	2,164
Rata-rata	**100**	**1,397**

Berdasarkan Tabel 5, pengujian akibat kotor lumpur memiliki akurasi rata-rata sebesar 100% dan rata-rata waktu komputasi sebesar 1,397 detik. Waktu komputasi terkecil, yakni 0,89 detik. Waktu tesebut didapat pada kondisi tingkat takaran lumpur ringan. Sedangkan waktu komputasi terbesar, yakni 2,164 detik. Waktu didapat pada kondisi lumpur tingkat berat. Pengujian kotor lumpur memiliki pengaruh kecil terhadap waktu komputasi dan tingkat akurasi. Hal ini didasari dengan perbedaan data waktu yang sedikit dengan tingkat akurasi 100%.

3.5.6 PENGUJIAN GESEKAN/ GORESAN

Pengujian dilakukan dengan cara menggesekan kartu pelajar (QR-Code) terhadap benda lain. Terdapat 3 kategori rusak, yaitu ringan (50 gesekan), sedang (100 gesekan) dan berat (150 gesekan). Pengujian dilakukan pada 10 kartu pelajar di setiap kategori. Total pengujian sebanyak 30 kartu pelajar.

(a) (b) (c)

Gambar 21. Pengujian gesekan atau goresan (a) Ringan, (b) Sedang, (c) Berat.

Tabel 6. Pengujian gesekan atau goresan.

Tingkat Goresan	Akurasi (%)	Waktu Komputasi (detik)
Ringan	100	0,86
Sedang	100	0,88
Berat	90	0,92
Rata-rata	**96**	**0,88**

Berdasarkan Tabel 6, pengujian akibat gesekan atau goresan memiliki akurasi sebesar 96% dan waktu komputasi sebesar 0,88 detik. Waktu komputasi terkecil, yakni 0,86 detik pada kondisi tingkat gesekan atau goresan ringan. Sedangkan waktu komputasi terbesar, yakni 0.91 detik pada kondisi tingkat gesekan atau goresan berat. Tingkat gesekan atau goresan berpengaruh terhadap hasil waktu komputasi pengujian. Semakin tinggi tingkat gesekan atau goresan semakin besar waktu komputasinya. Tingkat akurasi terendah, yakni 90% terjadi saat pengujian dengan tingkat gesekan atau goresan berat. Hal ini disebabkan gesekan atau goresan membuat pola gambar pada *QR-Code* menjadi berubah atau hilang yang menyebabkan kesalahan pada proses *scanning QR-Code.*

3.5.7 ANALISIS PERFORMANSI SISTEM

Skenario pengujian dilakukan untuk menghitung rata-rata akurasi dan rata-rata waktu komputasi, dari hasil pengujian cahaya, kotor tinta, kotor lumpur, dan gesekan atau goresan.

Tabel 7. Performansi sistem.

Parameter Pengujian	Akurasi (%)	Waktu Komputasi (detik)
Cahaya	100	1,471
Kotor Tinta	97	1,485
Kotor Lumpur	100	1,397
Gesekan	96	0,88
Rata-rata	**98**	**1,3**

Berdasarkan Tabel 7, sistem memiliki akurasi 100% pada pengujian berdasarkan cahaya. 97% pada pengujian kotor tinta, 100% pada pengujian kotor lumpur, dan 96% pada pengujian gesekan atau goresan. Rata-rata waktu komputasi cahaya sebesar 1,471 detik, kotor tinta sebesar 1,485 detik, kotor lumpur sebesar 1,397 detik, dan gesekan atau goresan sebesar 0,88 detik.

Berdasarkan hasil seluruh pengujian, sistem memiliki tingkat akurasi 98%, dan rata-rata waktu komputasi sebesar 1,3 detik. Tingkat akurasi dan waktu komputasi mengindikasikan sistem bekerja dengan baik.

4. KESIMPULAN

Penelitian ini telah melakukan desain dan implementasi *QR-Code* berbasis *android* dengan kode *Bose, Chaudhuri, Hocquenghem* (*BCH*) sebagai *error correction* untuk sistem presensi. Deteksi *QR-Code* memiliki jarak minimal sebesar 3 cm dan jarak maksimal sebesar 45 cm dengan rata-rata waktu komputasi sebesar 1,26 detik. Secara keseluruhan, sistem yang dibuat pada penelitian ini memiliki akurasi sebesar 98% dan rata-rata waktu komputasi sebesar 1,3 detik.

Sistem dapat mengatasi gangguan cahaya pada kartu pelajar dengan akurasi 100%. Semakin besar intensitas cahaya yang diberikan, semakin besar waktu komputasi deteksi. Sistem dapat mengatasi kerusakan kartu pelajar akibat kotor tinta berwarna hitam, merah, biru, hijau dan kuning pada kondisi ringan, sedang dan berat dengan akurasi 97%. Semakin banyak dan gelap warna tinta yang diberikan, semakin besar waktu komputasi deteksi dikarenakan pola warna gambar pada *QR-Code* berubah dan timpang tindih. Sistem dapat mengatasi kerusakan kartu pelajar akibat kotor lumpur. Pengujian kotor lumpur memiliki akurasi 100%. Kotor lumpur tidak berpengaruh besar terhadap kerusakan kartu pelajar. Pada pengujian gesekan atau goresan, sistem memiliki akurasi 96%. Gesekan atau goresan membuat pola gambar QR-Code menjadi berubah atau hilang.

DAFTAR PUSTAKA

ASARE, I.T., 2015. The Effective Use of Quick Response (QR) Code as a Marketing Tool. IJESS, vol. 2, no. 12.

DE LIMA, N.V., NOVAMIZANTI, L., SUSATIO, E., 2019. Sistem Pengenalan Wajah 3D Menggunakan Icp Dan Svm. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), Vol. 6, No. 6, Desember 2019, hlm. 601-610.

M. LUKMAN, LUTFI A M. 2016. Smart Presence menggunakan QR-Code dengan enkripsi Vignere Cipher. Jember: Universitas Muhammadiyah Jember.

EKO, F S. 2014. Simulasi Kode Hamming, Kode BCH, dan Kode Reed-Solomon untuk Optimalisasi Forward Error Correction. Makalah. Surakarta: Universitas Muhammadiyah Surakarta.

EKO, S. 2009. Analisis Kinerja Kode BCH. Medan: Universitas Sumatera Utara.

HERMANTO, N., NURFAIZAH, BAIHAQI, W. M., SARMINI., 2018. Implementation of QR Code and Imei on Android and Web-Based Student Presence Systems. 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE). 13-14 Nov. 2018, Yogyakarta, Indonesia.

USMAN, K., SEPTIRASYAHYANI, NOVAMIZANTI, L., 2011. Desain dan Implementasi QR-Code Berbasiskan Pengolahan Citra untuk Sistem Parkir di IT Telkom. Bandung: IT Telkom.

INDARTONO, K., JAHIR, A., 2019. Prototype Sistem Keamanan Mobil Dengan Mengunakan Quick Response Code Berbasis Android dan Arduino. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), Vol. 6, No. 3, Juni 2019, hlm. 235-244.

MASALHA, F., HIRZALLAH, N., 2014. A Students Attendance System Using QR Code. International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 5, No. 3, pp. 75-79.

NORHIKMAH, SAFITRI, A. R., SHOLIKHAN, L.A., 2016. Penggunaan QR Code dalam Presensi Berbasis Android. Seminar Nasional Teknologi Informasi dan Multimedia, pp. 97-102. STMIK AMIKOM Yogyakarta, 6-7 Februari 2016.

KRISMANTO, P., USMAN, K., NOVAMIZANTI, L., 2011. Desain dan Implementasi Prototype Sistem Presensi Otomatis Berbasis Barcode Menggunakan Webcam dan Pengolahan Citra Digital di IT Telkom. Bandung: IT Telkom.

SHOLEH, M.L., MUHAROM, L.A., 2016. Smart Presensi Menggunakan QR Code dengan Enkripsi Vigenere Cipher. Limits, vol. 13, no. 2, Nopember 2016, pp. 31-44.

TIWARI, S., 2016. An Introduction to QR Code Technology. India: SITS Educators Society.