Perancangan *Smart Card Reader* Menggunakan STM32F4 Discovery Kit

Agus Bejo[1], Mohamad Faiz Hamzah[2], Addin Suwastono[3]

***Abstract*— Smart card has been a new trend as practical and secure authentication solution in online transactions such as e-payment, net-banking, e-money, and other online services. The increasing of smart card-based applications demands higher supply of smart card readers as well. Many types of smart card readers in the market have been existed. However, their feature and software are mostly closed and can not be modified to satisfy the application developer's requirements to optimize the performance and security of the applications. Therefore, a self-designed smart card reader is needed to offer flexibility and ability to be costumized in order to satisfy application developer's needs. In this research, a smart card reader is designed based on 32-bit microcontroller STM32F407VG which is implemented on STM32F4 Discovery Kit. The proposed smart card reader is evaluated by accessing information resides on the JCOP31 smart card which has been pre-installed by applet with certain APDU. Evaluation results show that the proposed smart card reader is able to access smart card properly, having good portability on different platform machines and having good performance as indicated by the CWT and CBT which are faster than the recommended ones.**

***Intisari*—*Smart card* telah menjadi tren baru sebagai solusi autentikasi yang praktis dan aman pada berbagai transaksi *online* seperti pembayaran *online* (*e-payment*), transaksi perbankan *online* (*net-banking*), transaksi berbasis uang elektronik (*e-money*), maupun layanan *online* lainnya. Meningkatnya jumlah aplikasi berbasis *smart card* berakibat pada meningkatnya pula kebutuhan mesin pembaca *smart card* atau sering disebut dengan *smart card reader*. Ada banyak tipe dan merk *smart card reader* yang tersedia di pasaran, tetapi umumnya fitur dan perangkat lunaknya bersifat tertutup sehingga tidak dapat dimodifikasi apabila pengembang aplikasi memerlukan kustomisasi untuk mengoptimalkan kinerja dan keamanan aplikasinya. Oleh karena itu, diperlukan sebuah perangkat *smart card reader* yang fleksibel dan dapat dikustomisasi untuk memenuhi kebutuhan pengembang aplikasi. Pada makalah ini, sebuah *smart card reader* dirancang menggunakan rangkaian minimalis berbasis mikrokontroler 32-bit STM32F407VG. Rancangan *smartcard reader* diimplementasikan pada STM32F4 Discovery Kit. Pengujian dilakukan dengan membaca informasi pada *smart card* bertipe JCOP31 yang di dalamnya sudah terinstal *applet* dengan perintah APDU tertentu. Perintah untuk membaca dan menampilkan hasil pembacaan *smart card* dikendalikan oleh komputer melalui aplikasi GUI. Hasil evaluasi dan pengujian menunjukkan bahwa rancangan *smart card reader* mampu bekerja mengakses *smart card* dengan baik, memiliki portabilitas pada beberapa platform mesin yang berbeda serta memiliki kehandalan yang baik yang ditunjukkan oleh waktu CWT dan BWT yang singkat dan memenuhi standard spesifikasi yang direkomendasikan.**

***Kata Kunci*— *smart card reader, smart card,* JCOP31, STM32F4DISCOVERY, GUI.**

I. PENDAHULUAN

Teknologi informasi dan elektronika berkembang sangat pesat. Bermula dari penemuan transistor pada tahun 1947, dikembangkanlah teknologi *Integrated Circuit* (IC) yang kemudian terus berevolusi hingga menjadi teknologi *Very Large Scale Integrated Circuit* (VLSI) yang membenamkan jutaan transistor dalam satu *chip*. Salah satu bentuk teknologi IC adalah munculnya *Integrated Circuit Card* (ICC) atau yang lazim disebut dengan *chip card*, yaitu satu *chip* yang berisi ribuan bahkan ratusan ribu transistor dengan skala yang sangat kecil ditanamkan ke sebuah kartu plastik.

Chip card digunakan pertama kali pada tahun 1983, yaitu sebuah kartu yang digunakan secara luas oleh penduduk Perancis sebagai alat pembayaran telepon, *Télécarte*. Pada tahun 1992, untuk kedua kalinya *chip card* digunakan, yaitu integrasi antara *microchip* dengan semua kartu debit di Perancis, *Carté Bleue*. Penggunaan *chip card* meledak pada era 90-an, ketika dikenalkan teknologi *smart card* berbasis *Subscriber Identity Module card* (SIM) yaitu salah satu tipe *chip card* yang digunakan untuk telepon seluler, khususnya di Eropa pada saat itu. Dengan semakin umumnya penggunaan telepon seluler di Eropa, istilah *smart card* menjadi semakin populer dan menyebar ke seluruh dunia.

Akhir-akhir ini, jumlah permintaan *smart card* di dunia terus meningkat, sehingga berdampak pula pada kebutuhan infrastruktur pendukung, yaitu *Card Acceptance Device* (CAD) atau mesin pembaca kartu atau sering disebut dengan istilah *smart card reader*. Menurut Eurosmart, pada tahun 2016 tercatat 9,8 milyar perangkat terintegrasi teknologi *smart card* akan didistribusikan di seluruh dunia atau tumbuh sebesar 13% dari tahun 2015 [1]. Pasar yang semula menggunakan layanan teknologi lama, seperti *barcode* dan *magnetic stripe*, berpindah menjadi menggunakan *smart card*.

Smart card reader adalah sebuah perangkat elektronik yang diperlukan oleh *host* komputer agar dapat berkomunikasi dengan *smart card*. *Smart card* bekerja bersama *smart card reader* untuk dapat menyelenggarakan transaksi informasi dari dan ke kartu dengan aplikasi di dunia luar, misalnya kartu identitas kependudukan, *passport*, kartu asuransi, kartu catatan medis, visa, debit, *e-ticketing*, dan akses keamanan. Umumnya, *smart card reader* yang banyak digunakan adalah

[1, 3] *Dosen, Departemen Teknik Elektro dan Teknologi Informasi, Fakultas Teknik, Universitas Gadjah Mada, Jln. Grafika 2 Yogyakarta 55281 Indonesia . (Telp: 0274-552305; Fax: 0274-552305; e-mail: [1]agusbj@ ugm.ac.id, [3]adyn@ugm.ac.id)*

[2] *Mahasiswa, Departemen Teknik Elektro dan Teknologi Informasi, Fakultas Teknik, Universitas Gadjah Mada, Jln. Grafika 2 Yogyakarta 55281 Indonesia . (Telp: 0274-552305; Fax: 0274-552305; e-mail: mohamad.faiz.h@mail.ugm.ac.id,)*

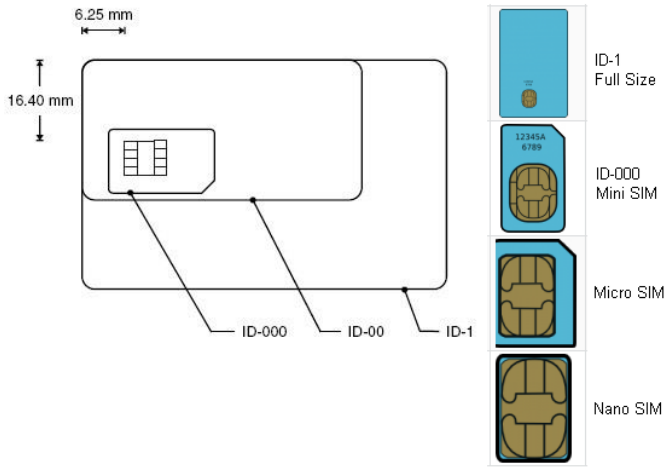
yang bertipe *contacted* atau dengan hubungan kontak langsung karena lebih aman. Selain itu, konektivitasnya stabil karena kartu berada pada posisi tetap di dalam *card slot* dan dapat melakukan transaksi data berukuran besar.

Menurut Badan Pengembangan dan Pengkajian Teknologi (BPPT), pada tahun 2013 tercatat jumlah mesin pembaca kartu atau *smart card reader* yang bersirkulasi di Indonesia mencapai 13.000 unit, yang digunakan baik untuk urusan pelayanan publik, perbankan, keamanan, dan berbagai hal lain. Sayangnya, dari sekian besar jumlah kebutuhan *smart card reader* tersebut, pengembangan dan penelitian di Indonesia masih tergolong sedikit. Kebanyakan *smart card reader* yang ada merupakan produk yang diimpor dari Amerika dan Korea Selatan [2]. Selain harga *smart card reader* di pasaran terbilang cukup mahal, umumnya *smart card reader* yang ada tidak fleksibel atau tidak bisa dimodifikasi sesuai keinginan pengembang aplikasi untuk mengoptimalkan kinerja dan keamanan aplikasinya. Berdasarkan latar belakang tersebut, dalam makalah ini dirancang sebuah *smart card reader* berbasis rangkaian mikrokontroler minimalis yang diimplementasikan pada STM32F4 Discovery Kit. Sebagai tahap awal perancangan *smart card reader* ini difokuskan pada kemampuan membaca informasi dari kartu standar yang banyak ditemui di pasaran yaitu kartu dengan Sistem Operasi Java seperti *JavaCard Open Platform* (JCOP).

II. *SMART CARD* DAN ATRIBUTNYA

A. Smart Card

Smart card merupakan kartu plastik yang ditanamkan IC di dalamnya. Maka dari itu, pada mulanya *smart card* diperkenalkan dengan istilah IC *Card* [3]. *Smart card* bersifat portabel dan *tamper-resistant*. Tidak seperti kartu *magnetic stripe*, *smart card* mampu menyimpan informasi dan melakukan pengolahan lokal pada data yang disimpan. *Smart card* memiliki ukuran 85,6 mm x 53,98 mm x 0,76 mm atau dikenal dengan ID-1 [3]. Dalam perkembangan, *smart card* memiliki beberapa standar yang digunakan secara simultan dan diaplikasikan pada penggunaan yang berbeda yaitu format ID-1, ID-00, ID-000 (*Mini* SIM), *Micro* SIM, dan *Nano* SIM, seperti ditunjukkan pada Gbr. 1.



Gbr. 1 Ukuran format kartu standar ID-1 sampai dengan *Nano* SIM.

B. Card Acceptance Device

Biasanya sebuah *smart card* tidak memiliki suplai daya. Oleh karena itu, untuk dapat melakukan komunikasi dengan dunia luar, harus dipasangkan dengan CAD. CAD dapat diklasifikasikan menjadi dua jenis, yaitu *reader* dan *terminal*. Sebuah *reader* adalah perangkat yang terkoneksi secara serial ke *host* komputer agar dapat melakukan komunikasi dengan *smart card*. *Reader* memiliki *slot* untuk menempatkan *smart card*, memberikan suplai daya, dan menyediakan jalur agar *host* komputer dapat mengakses *smart card*. Sedangkan *terminal* adalah suatu PC khusus yang terinstal aplikasi dan terintegrasi dengan *smart card reader* sehingga bisa melakukan proses transaksi data langsung dengan *smart card*, misalnya *Automatic Teller Machine* (ATM).

C. Protokol Tranmisi

Protokol transmisi yang digunakan *smart card* untuk melakukan transaksi struktur data dikenal dengan istilah *transmission protocol data unit* (TPDU). Dua protokol transmisi yang umum digunakan saat ini ada dua, yaitu protokol transmisi T=0 dan protokol transmisi T=1. Makalah ini fokus pada pengerjaan protokol transmisi T=1.

Protokol komunikasi T=1 merupakan protokol komunikasi blok *half-duplex* asinkron. Protokol ini memiliki model dengan operasi dua *layer*, yaitu *data link layer* dan *physical layer*. Protokol ini memberikan segel di sekitar blok karakter yang memungkinkan untuk *flow control*, *block chaining*, dan *error correction*. Kelebihan protokol T=1 ini adalah kemampuannya untuk mengontrol aliran data pada dua arah dan memiliki sistem pengaturan kesalahan yang lebih bagus. Hal ini memungkinkan penggunaan *error detection code* (EDC) dan kemampuan mentransmisikan ulang blok salah (*error*).

Blok *frame* dalam protokol transmisi T=1 terdiri atas tiga bagian, yaitu *prologue field*, *information field*, dan *epilogue field*. *Prologue field* memiliki tiga *byte*, yaitu *node address* (NAD), *protocol control byte* (PCB) dan *data length* (LEN). Tabel I menjelaskan bagian blok *frame* dalam protokol transmisi T=1 secara lebih rinci.

TABEL I
BLOK *FRAME* PADA PROTOKOL TRANSMISI T=1

Prologue Field			***Information Field***	***Epilogue Field***
Node Address	*Protocol Control*	*Data Length*	(pilihan)	*Error Detection LRC* atau *CRC*
NAD	PCB	LEN	INF	EDC
1 *byte*	1 *byte*	1 *byte*	0--254 *byte*	1 atau 2 *byte*

Byte NAD menggunakan bit 3-1 untuk mengidentifikasi alamat sumber dan bit 7-5 untuk megidentifikasi alamat tujuan. Sedangkan bit 4 dan 8 digunakan untuk kontrol Vpp. *Byte* PCB mengizinkan identifikasi tiga tipe *frame* blok, yaitu *information block* (*I - block*), *receive ready block* (*R - block*) dan *supervisory block* (*S - block*). *Information block* merupakan *frame* yang digunakan untuk mentransfer perintah aplikasi dan data antara *smart card* dengan CAD. *Receive*

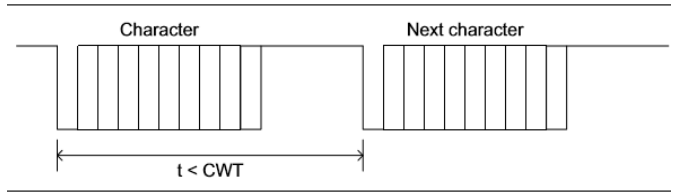
ready block digunakan sebagai tanda ketika protokol sedang mengirimkan data sebagai urutan rangkaian blok. *Supervisory block* digunakan untuk membangun parameter kontrol dan untuk memengaruhi proses sinkronisasi kembali dan status pembatalan sebagai hasil dari beberapa kondisi yang salah. *Byte* LEN mengindikasikan ukuran *information field* dari *frame* dalam *byte*. *Byte* ini mengizinkan *information field* maksimum yaitu sebesar 254 *byte* setara kapasitas maksimum data yang dapat dikirimkan oleh *smart card*. *Information field* digunakan untuk menyampaikan perintah aplikasi dan data. *Epilogue field* mengandung blok kode pendeteksi kesalahan yaitu *longitudinal redundancy check* (LRC) atau *cyclic redundancy check* (CRC).

Protokol T=1 menggunakan tiga karakter untuk membangun *interface character* yang digunakan dalam ATR untuk memulai komunikasi, yaitu sebagai berikut.

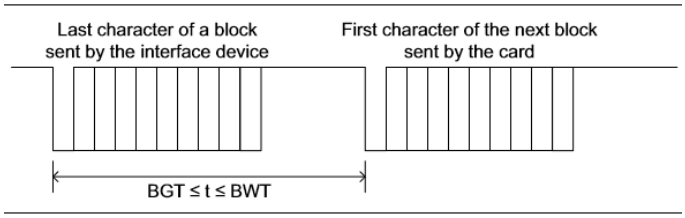
1. TAi = IFSC.
2. TBi, dengan (bit 8 - 5) = BWI dan (bit 4 - 1) = CWI.
3. TCi, jika (bit 1 = 1) = CRC atau (bit 1 = 0) = LRC.

Information Field Size Card (IFSC) merupakan bagian ukuran informasi kartu. Juga ada *Information Size Field Device* (IFSD) yang merupakan bagian ukuran informasi CAD.

Protokol T=1 ini menggunakan dua parameter *waiting time* untuk membantu kontrol aliran, yaitu *character waiting time* (CWT) dan *block waiting time* (BWT). CWT dan BWT masing-masing diilustrasikan pada Gbr. 2 dan Gbr. 3.



Gbr. 2 *Character Waiting Time* (CWT).



Gbr. 3 *Block Waiting Time* (BWT).

CWT dan BWT dihitung dari nilai CWI, BWI, dan Inisial ETU dengan persamaan berikut [4], [5].

$$\text{Inisial etu} = \frac{372}{f} \text{ s (f biasanya = 3,579545MHz)} \quad (1)$$

$$\text{CWT} = (2^{\text{BWI}} + 11) \text{ etu} \quad (2)$$

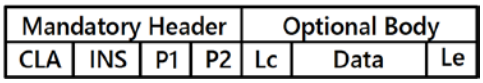
$$\text{BWT} = (2^{\text{BWI}} \times 960 \times 372 / f) \text{ detik} + 11 \text{ etu} \quad (3)$$

dengan f adalah frekuensi clock.

D. Struktur Pesan APDU

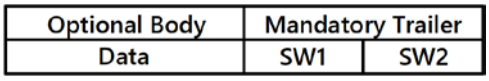
Dalam spesifikasi ISO 7816-4, dijelaskan struktur sebuah pesan yang berlaku, dikenal dengan *aplication protokol data unit* (APDU). Pesan APDU dari dua struktur, yaitu *command* APDU (C-APDU) digunakan oleh CAD untuk mengirim perintah ke kartu, dan *response* APDU (R-APDU) digunakan oleh kartu untuk mengirim respons ke CAD.

Struktur C-APDU terbagi atas *mandatory header* dan *optional body*. Bagian *header* terdiri atas *class of instruction* (CLA), *instruction code* (INS), serta parameter *P1* dan *P2*. Bagian *body* ini bersifat opsional, terdiri atas *Lc*, *Data*, dan *Le* dengan ukuran maksimal 255 *byte*. Ilustrasi struktur C-APDU dijelaskan oleh Gbr. 4.

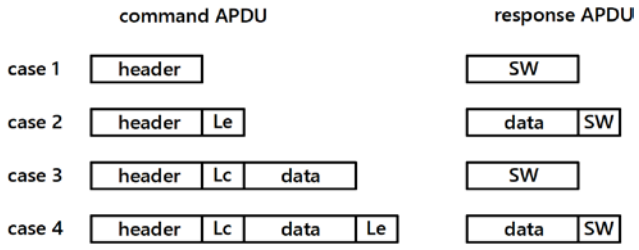


Gbr. 4 Struktur C-APDU.

Struktur R-APDU terbagi atas *optional body* dan *mandatory trailer*. Bagian *body* ini hanya bersifat opsional, hanya terdiri atas data yang panjangnya ditentukan oleh *Le* dalam C-APDU yang bersangkutan. Bagian *trailer* terdiri atas *SW1* dan *SW2*, yaitu *status word*. Ilustrasi struktur R-APDU dijelaskan oleh Gbr. 5.



Gbr. 5 Struktur R-APDU.



Gbr. 6 Pasangan APDU *optional body* pada kasus yang berbeda.

Pasangan *optional body* C-APDU dengan R-APDU dapat memiliki komposisi struktur berbeda-beda, seperti tampak pada kategori kasus Gbr. 6.

a. Kasus 1, tidak ada data dikirimkan ke atau dari kartu, sehingga C-APDU hanya berisi *header* dan R-APDU hanya berisi *SW*.
b. Kasus 2, tidak ada data dikirimkan ke kartu, tetapi ada data dikembalikan dari kartu. *Optional body* C-APDU berisi *Le* 1 *byte* yang menspesifikasikan jumlah data yang akan diterima dari kartu. Sementara R-APDU berisi data yang dikembalikan dari kartu dikuti dengan *SW*.
c. Kasus 3, data dikirim ke kartu, tapi tidak ada data dikembalikan dari kartu sebagai hasil dari pemrosesan C-APDU. *Optional body* C-APDU berisi *Lc* dan data. *Lc* menunjukkan panjang data yang dikirimkan. R-APDU hanya berisi *SW*.
d. Kasus 4, data dikirim ke kartu, dan ada data dikembalikan dari kartu. *Optional body* dari C-APDU berisi *Lc*, data, dan *Le*. *Lc* menunjukkan jumlah data dalam *byte* yang dikirimkan, sedangkan *Le* menunjukkan jumlah data dalam *byte* yang akan diterima. R-APDU berisi data diikuti dengan *SW*.

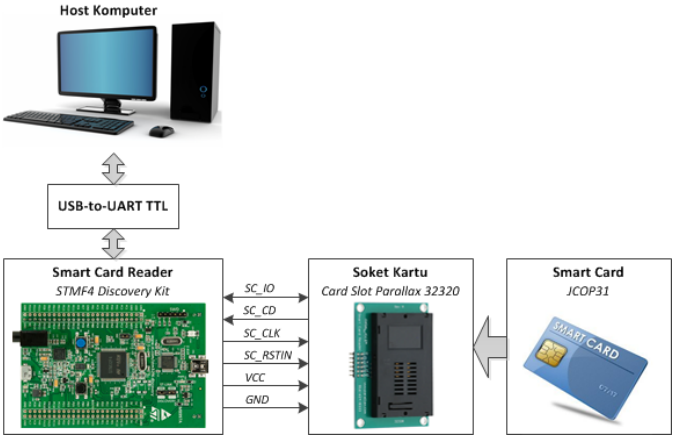
E. JCOP31

JCOP merupakan sistem operasi *smart card* untuk arsitektur *JavaCard*. Salah satu kartu JCOP yang banyak digunakan di pasaran adalah JCOP31 V2.2 dengan spesifikasi *JavaCard* 2.2.1 dari SUN Microsystems. Famili JCOPxx V2.2 diproduksi dengan teknologi canggih lima *layer* metal CMOS 0,18 µm agar diperoleh volume besar dan bisa melayani multi-aplikasi. Kelebihan penggunaan JCOP31 V2.2 antara lain dapat diprogram dengan bahasa pemrograman level tinggi, Java, sehingga mudah membuat *applet* [6], memiliki total ruang data sebesar 140 *kilobyte*, dan mendukung *dual interface*. Operasi untuk *contact interface* mendukung *baudrate* 9.600 bps dengan protokol transmisi *default* T=1 bersifat konvensi langsung. Sedangkan operasi untuk *contactless interface* kompatibel dengan ISO/IEC 14443A dan mendukung komunikasi dengan arus data mencapai 424 kbps.

III. PERANCANGAN SISTEM

A. Desain Perangkat Keras

Tahapan awal perancangan *smart card reader* adalah menyusun rangkaian perangkat keras. Rangkaian perangkat keras *smart card reader* terdiri atas *host* komputer, UART-to-USB TTL, STM32F4 Discovery Kit, dan soket kartu *Card Slot* Parallax 32320. Rangkaian perangkat keras keseluruhan *smart card reader* ditunjukkan pada Gbr. 7.



Gbr. 7 Diagram blok perangkat keras *smart card reader*.

Smart card reader memperoleh suplai tegangan DC 3,3 volt melalui *port* USB *debugger* STM32F4 Discovery Kit yang terhubung dengan *port* USB *host* komputer. Setelah *smart card reader* menyala, *smart card* dimasukkan melalui soket kartu C*ard Slot* Parallax 32320 untuk melakukan proses transaksi data. *Smart card reader* bertugas sebagai perantara untuk menerima perintah dari *host* komputer dan menerjemahkannya dalam runtutan *byte* yang sesuai untuk ditransmisikan ke *smart card*. Data yang ditransmisikan ke/dari *smart card* berbentuk heksadesimal dan dapat dikonversikan ke bentuk ASCII supaya dapat ditampilkan dalam penampil dan mudah dipahami oleh pengguna. STM32F4 Discovery Kit mengirimkan data yang telah dikonversi untuk ditampilkan ke *host* komputer melalui *port* serial USB-to-UART TTL. *Host* komputer akan menerima data hasil pembacaan dari *smart card* kemudian menampilkanya dalam aplikasi GUI.

Interkoneksi antara STM32F4 Discovery Kit dan soket kartu C*ard Slot* Parallax 32320 ditunjukkan pada Tabel II. STM32F4 Discovery Kit mengalokasikan empat buah *pin* I/O, yaitu PD8, PE6, PB12, dan PB15, yang masing-masing terhubung ke *pin* I/O soket kartu *SC_IO*, *SC_CD*, *SC_CLK*, *SC_RSTIN* serta dua *pin* catu daya, masing-masing *VCC* dan *GND*. Mode transmisi *smart card* ditangani oleh USART3 mode *smart card with card clock*. STM32F4 Discovery Kit membagi *clock* 21 MHz untuk disuplai ke *clock smart card*. Oleh karena *clock* ini terlalu cepat dengan pengaturan umum frekuensi *smart card* adalah 3,57 MHz dan batas maksimal 10 MHz, maka *clock* perlu diperlambat dengan koefisien *prescaler = 5*. *SC_CLK* memberikan sinyal *clock* keluaran dari *prescaler* sebesar 21/5 = 4,2 MHz. *Pin SC_IO* atau saluran data diatur agar *baudrate* sama dengan *PCLK1*/3720. *SC_CD* (*card detection*) merupakan *pin* masukan yang digunakan untuk mendeteksi ada tidaknya *smart card*. Ketika *smart card* dimasukkan pada soket kartu, maka sistem mekanik akan terpicu untuk membangkitkan sinyal *high*. *SC_RSTIN* merupakan *pin* keluaran yang digunakan untuk memberikan sinyal bernilai *high* ke *smart card* agar memaksa program yang berada di IC ROM melakukan *reset*, lalu memberikan respons kode unik ATR. Catu daya *VCC* terhubung dengan suplai 3,3V dari pin power STM32F4 Discovery Kit, sedangkan *pin* GND tersambung dengan GND sebagai *grounding*.

TABEL II
PEMETAAN *PIN CARD SLOT* PARALLAX 32320

STM32F4 Discovery Kit		***Card Slot* Parallax 32320**		**Fungsi**
Pin	**Nama**	***Pin***	**Nama**	
PD8	*SC_IO*	1	I/O	Data I/O bidireksional
PE6	*SC_CD*	2	CD	Deteksi kartu
PB12	*SC_CLK*	3	CLK	Sinyal *clock*
PB15	*SC_RSTIN*	4	RST	Sinyal *Reset*
VCC	*VCC*	5	VCC	Catu daya 3,3 V
GND	*GND*	6	GND	*Ground*

TABEL III
PEMETAAN *PIN* USB-TO-UART TTL

STM32F4 Discovery Kit		**USB-to-UART TTL**	**Fungsi**
Pin	**Nama**	**Nama**	
PA3	TXD	RXD	Serial TX_MCU ke RX_host
PA2	RXD	TXD	Serial RX_MCU ke TX_host
GND	GND	GND	*Grounding*

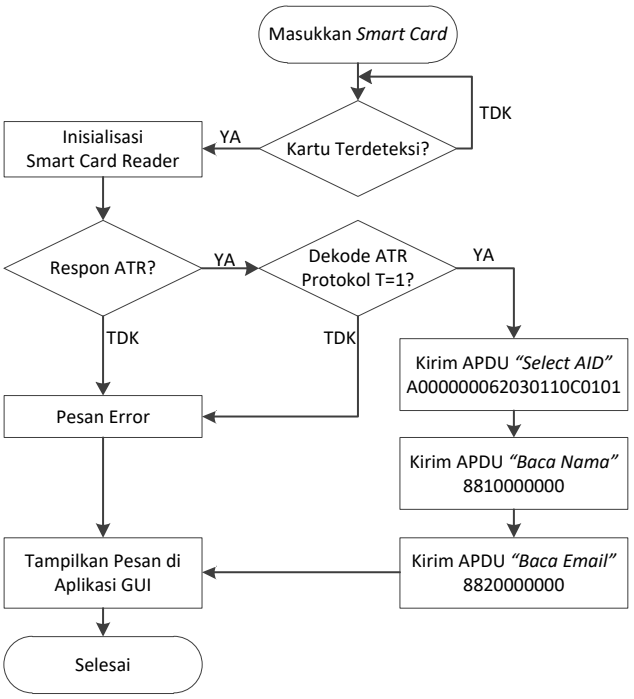
Koneksi perangkat keras antara STM32F4 Discovery Kit dengan USB-to-UART TTL bertujuan untuk mengerjakan tugas komunikasi serial antara *smart card reader* dengan *host* komputer, yang secara rinci disajikan pada Tabel III. Mode

transmisi serial antara STM32F4 Discovery Kit dengan *host* komputer ditangani oleh USART2 mode asinkron dengan *baudrate* 9.600 bps, panjang pesan 8 bit, 1 bit stop, dan tanpa bit paritas.

B. Desain Perangkat Lunak (Firmware)

Secara garis besar, perancangan perangkat lunak atau program yang berada pada *smart card reader* atau disebut dengan *firmware smart card reader* untuk membaca informasi yang disimpan pada *smart card* dapat dijelaskan seperti ditunjukkan pada diagram alir prosedur pembacaan *smart card* pada Gbr. 8.

Alur program *firmware smart card reader* ini diawali dengan pembuatan fungsi deteksi kartu untuk mengetahui ada tidaknya kartu yang masuk ke soket. Setelah *smart card* terdeteksi, program akan memanggil fungsi inisialisasi dengan memberi sinyal *reset* level logika *high* ke *smart card*, kemudian menunggu respons kode ATR, dan menyimpan kode ATR tersebut ke dalam *array buffer*. Program akan mendekode kode ATR untuk mengenali protokol transmisi yang didukung oleh *smart card*. Jika kode ATR tidak dikenali, maka program akan mengirimkan pesan *error* ke *host* komputer. Sebaliknya, jika kode ATR dikenali, maka program akan menginisiasi protokol transmisi berdasarkan parameter-parameter yang diperoleh dari kode ATR. Selanjutnya, *smart card reader* melakukan transaksi data dengan format pesan APDU yang telah ditentukan, antara lain mengirimkan perintah "*select AID*", perintah "*baca nama*", dan "*baca email*". Kemudian, hasil pembacaan akan dikirimkan oleh *smart card reader* ke *host* komputer. Terakhir, aplikasi GUI pada *host* komputer akan menampilkan data hasil pembacaan pada penampil. Program *firmware smart card reader* dibuat menggunakan perangkat lunak IAR Embedded Workbench dengan bahasa pemrograman C.



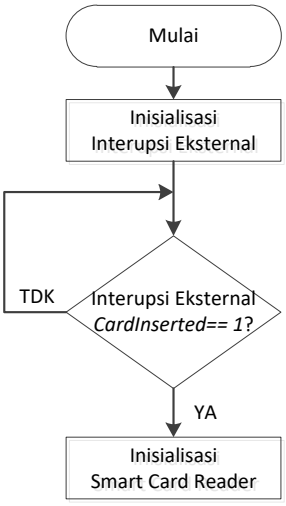
Gbr. 8 Diagram alir prosedur pembacaan *smart card.*

1) Deteksi Kartu: Fungsi deteksi kartu ini dirancang untuk mendeteksi keberadaan kartu yang masuk ke soket kartu *card slot* Parallax 32320. Sesaat setelah kartu masuk ke *card slot* sensor mekanis akan terpicu untuk membangkitkan sinyal level *high* pada *pin SC_CD* yang akan dikenali oleh fungsi deteksi kartu sebagai sinyal interupsi. Prinsip kerja fungsi ini dijelaskan secara rinci oleh diagram alir pada Gbr. 9.

Fungsi deteksi kartu menahan program untuk berada dalam *infinite looping* dengan syarat *CardInserted* = 0. Satu-satunya jalan keluar yaitu sinyal interupsi yang berasal dari pin deteksi kartu *SC_CD*, dengan mengubah nilai *CardInserted* = 1, sehingga program keluar dari *infinite looping* dan melanjutkan tugas berikutnya yaitu inisialisasi *smart card reader* dan siap menginisialisasi *smart card.*

2) Definisi APDU: SC_APDU_t merupakan definisi struktur APDU yang dijalankan oleh program *firmware smart card reader*. *SC_APDU_t* terdiri atas tiga bagian, yaitu *SC_APDU_Cmd_t*, *SC_APDU_Resp_t* dan *buffer[T1_Buffer_Size]*.

SC_APDU_Cmd_t merupakan pesan perintah untuk dikirimkan ke *smart card*, *SC_APDU_Resp_t* merupakan pesan balasan yang dikirimkan oleh *smart card*, dan *buffer[T1_Buffer_Size]* merupakan larik yang menangkap data dari transmisi T1 yang berbentuk blok dengan ukuran larik sebesar 255 *byte*.



Gbr. 9 Diagram alur deteksi kartu.

SC_APDU_Cmd_t tersusun atas *SC_Header_t* dan *SC_Body_t*. *SC_Header_t* mempunyai format standar, yaitu *CLA*, *INS*, *P1*, *P2*, yang masing-masing berukuran sebesar 1 *byte* data. *Byte CLA* berfungsi untuk menunjukkan *applet* yang menyimpan informasi pemilik kartu, *byte* INS menunjukkan instruksi membaca *applet*, sedangkan *byte* P1 dan P2 masing-masing merupakan *byte* yang menunjukkan parameter. *SC_Body_t* mempunyai format standar, yaitu *LC*, *DATA*[*LC_MAX*], *LE*. *LC* berukuran sebesar 1 *byte*, berfungsi untuk menginformasikan besar data dari *SC_Body_t* yang dikirimkan dalam *byte*. *DATA* merupakan isi informasi yang dikirimkan ke *smart card* dengan ukuran maksimal 255 *byte* sesuai ukuran *LC*. Sementara *LE* berukuran 2 *byte*, berfungsi

untuk menginformasikan besar data yang mungkin dibalas oleh *smart card*. Perintah APDU yang digunakan dalam makalah ini ditunjukkan pada Tabel IV.

TABEL IV
PERINTAH APDU

CLA	*INS*	*P1*	*P2*	*LC*	Keterangan
00	A4	00	00	0A	Memilih *applet*
88	10	00	00	00	Membaca nama
88	20	00	00	00	Membaca *email*

Pada program *firmware smart card reader* ini, proses pertama adalah memilih berkas dengan cara menunjuk ke AID *applet*.

Untuk memilih berkas program, *firmware smart card reader* akan mengirimkan c-APDU *select file* lengkap beserta AID *applet* yang diisi di dalam larik *DATA*[*LC_MAX*], seperti yang ditunjukkan pada Gbr. 10. Dalam contoh ini, AID berupa larik 10 *byte*, yaitu 0xA0 0x00 0x00 0x00 0x62 0x03 0x 01 0x0C 0x01 0x01. Setelah pesan c-APDU telah siap, maka fungsi *T1_APDU* dipanggil untuk mengerjakan transaksi data yang berorientasi blok sesuai dengan sifat protokol transmisi T=1. Kemudian, untuk membaca nama dan *email* pemilik, kartu juga terlebih dahulu menyiapkan pesan c-APDU yang dijelaskan secara rinci oleh Tabel V. Fungsi *T1_APDU* tidak dapat dijalankan jika status komunikasi yang dibaca oleh *resp_status* bernilai kurang dari nol, sehingga program aplikasi *smart card reader* akan menampilkan pesan *error.*

```
/* Send Select File Command */
cAPDU1.APDU_S.Header.CLA = 0x00;                /* CLA */
cAPDU1.APDU_S.Header.INS = 0xA4;                /* INS: Select File */
cAPDU1.APDU_S.Header.P1 = 0x00;                 /* P1 */
cAPDU1.APDU_S.Header.P2 = 0x00;                 /* P2 */
cAPDU1.APDU_S.Body.LC = 0x0A;                   /* Lc */
cAPDU1.APDU_S.Body.Data[0] = (uint8_t)(0xA0);   /* Data */
cAPDU1.APDU_S.Body.Data[1] = (uint8_t)(0x00);
cAPDU1.APDU_S.Body.Data[2] = (uint8_t)(0x00);
cAPDU1.APDU_S.Body.Data[3] = (uint8_t)(0x00);
cAPDU1.APDU_S.Body.Data[4] = (uint8_t)(0x62);
cAPDU1.APDU_S.Body.Data[5] = (uint8_t)(0x03);
cAPDU1.APDU_S.Body.Data[6] = (uint8_t)(0x01);
cAPDU1.APDU_S.Body.Data[7] = (uint8_t)(0x0C);
cAPDU1.APDU_S.Body.Data[8] = (uint8_t)(0x01);
cAPDU1.APDU_S.Body.Data[9] = (uint8_t)(0x01);


/* ---Sending APDU:  cAPDU, 7--- */

HAL_Delay(CARD_CHILL_TIME);
/* Send/Receive APDU command/response: Select AID applet */
resp_status = T1_APDU(&SCInterface, NAD, &cAPDU1, &rAPDU1);

/* If the APDU communication has failed */
if (resp_status < 0)
{
  /* ---APDU communication error--- */
  Error_Handler();
}
```

Gbr. 10 *Source code* memilih AID *applet.*

C. *Desain Aplikasi GUI.*

Aplikasi GUI dirancang untuk berjalan di *host* komputer. Tugas aplikasi GUI adalah memberi perintah dan mengendalikan *smart card reader*. Aplikasi GUI dirancang menggunakan perangkat lunak IDE yang berbahasa pemrograman Java. Dalam eksperimen ini, tugas utama aplikasi GUI adalah memerintahkan *smart card reader* untuk membaca informasi nama dan *email* yang sudah disimpan pada *smart card*, kemudian menampilkannya pada penampil GUI. Tampilan aplikasi GUI tampak seperti pada Gbr. 11.

Gbr. 11 Tampilan aplikasi GUI.

Gbr. 12 menjelaskan secara garis besar kode program aplikasi GUI. Segera setelah proses inisialisasi untuk membuat tampilan, *background*, *setting port* serial dilakukan, maka aplikasi GUI dalam kondisi *standby* untuk menunggu data yang dikirimkan melalui komunikasi serial oleh *smart card reader*. Ketika ada data yang diterima melalui komunikasi serial, maka data akan disimpan pada variabel *myString*. Isi variabel *myString* ini kemudian akan dibandingkan dengan variabel *header* yang telah didefinisikan sebelumnya, yaitu berupa string "*nama*", "*email*", atau "*detect*". Metode *data parsing* adalah dengan mencari kata pengenal yang merupakan *header*, lalu data dipotong antara *substring header* dengan karakter *new line* '/n' yang berada di akhir *myString*. Kemudian, data hasil *parsing* akan ditampilkan pada masing-masing kotak *text label* yang telah disediakan pada aplikasi GUI. Selain itu, aplikasi GUI juga menangkap informasi status yang menyatakan status kartu terdeteksi atau *error*.

```
6  int lf = 10;    // Linefeed in ASCII
7  String myString = null;
8  String nama = "NAMA "; String email = "EMAIL"; String detect = "SC d"
9  Serial myPort;  // The serial port
10 public void setup(){
11   size(480, 320, JAVA2D);
12   createGUI();
13   customGUI();
14   img=loadImage("background.png");
15   printArray(Serial.list());
16   myPort = new Serial(this, Serial.list()[0], 9600);  myPort.clear();
17   myString = myPort.readStringUntil(lf);  myString = null;
18 }
19 public void draw(){
20   background(img);
21     while (myPort.available() > 0) {
22     myString = myPort.readStringUntil(lf);
23     if (myString != null) {
24       if(myString.substring(0,4).equals(detect)){
25       label5.setText(myString.substring(0,myString.length()-6));
26       }
27       if(myString.substring(0,5).equals(nama)){
28       label1.setText(myString.substring(5,myString.length()-1));
29       }
30       if(myString.substring(0,5).equals(email)){
31       label2.setText(myString.substring(6,myString.length()-1));
32       }
33       println(myString);
34     } } }
```

Gbr. 12 Potongan kode program aplikasi GUI.

IV. EVALUASI PENGUJIAN DAN PEMBAHASAN

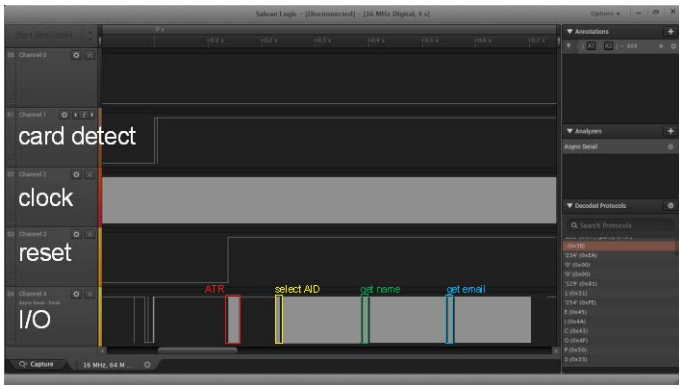
Untuk mengevaluasi keandalan rancangan *smart card reader* yang telah dibuat, dilakukan empat pengujian yaitu: (1) pengujian fungsionalitas, (2) pengujian parameter *waiting time*, (3) pengujian keandalan, dan (4) pengujian portabilitas di beberapa platform mesin yang berbeda.

A. Pengujian Fungionalitas

Pengujian fungsionalitas dilakukan untuk mengetahui apakah transaksi data antara *smart card reader* dengan *smart card* berjalan dengan baik. Dalam uji ini, *smart card reader* menerima perintah dari aplikasi GUI yang dijalankan di *host* komputer yaitu laptop, kemudian *smart card reader* akan membaca informasi yang ada pada *smart card* tipe JCOP31. Di dalam *smart card* JCOP31 sudah terinstal *applet* berisi informasi data nama dan *email* pemilik kartu. Format komunikasi data sesuai dengan aturan APDU ditunjukkan pada Tabel IV. Gbr. 13 menunjukkan kondisi eksperimen pengujian. Pertama, program aplikasi akan mengirimkan pesan *select file* AID. Setelah *applet* berhasil dipilih, selajutnya diirimkan pesan untuk meminta informasi pemilik kartu. Informasi pemilik kartu ini didefinisikan dengan *byte* CLA 0x88. Di dalam *applet* tersimpan informasi nama dan *email* pemilik kartu yang masing-masing diwakilkan dengan *byte* INS 0x10 dan 0x20.

Analisis hasil pengujian dapat dilihat melalui sinyal yang ditangkap pada perangkat lunak *logic analyzer "Salea Logic"* maupun secara langsung melalui penampil aplikasi GUI. Analisis berdasarkan hasil-hasil pembacaan sinyal melalui perangkat *logic analyzer "Salea Logic"* ditunjukkan pada Gbr. 14.

Gbr. 13 *Setup* eksperimen pengujian *smart card reader*.



Gbr. 14 Analisis fungsionalitas *smart card reader* menggunakan perangkat lunak *logic analyzer*.

Tampak pada *channel-01* atau *channel card detect* terlihat bahwa setelah *smart card* dimasukkan pada soket, maka dihasilkan sinyal *high* sehingga akan menginterupsi *smart card reader* dan menghasilkan respons untuk melakukan proses inisialisasi. Beberapa saat setelah proses ininsialisasi selesai, maka *smart card reader* membangkitkan sinyal *reset high* seperti ditunjukkan pada *channel-03* atau *channel reset*. Sinyal reset akan diterima oleh *smart card*, kemudian *smart card* akan mengirimkan kode ATR seperti ditunjukkan pada *channel-04* atau *channel I/O*. Apabila kode ATR yang dikirimkan oleh *smart card* dikenali, maka komunikasi selanjutnya dapat dilakukan berdasarkan paramater yang sudah diketahu melalui kode ATR tersebut. Program *firmware* pada *smart card reader* selanjutnya akan mengirimkan perintah *select AID* 0x00 0x00 0x00 0x62 0x03 0x01 0x0C 0x01 0x01, lalu JCOP31 membalas dengan kode status 0x90 0x00 yang berarti OK. Kemudian, atas perintah dari aplikasi GUI, *smart card reader* mengirimkan APDU 0x88 0x10 0x00 0x00 0x00 yang berarti meminta informasi nama pemilik kartu diikuti dengan APDU 0x88 0x20 0x00 0x00 0x00 yang berarti meminta informasi *email* pemilik kartu. *Applet* pada JCOP31 akan merespons dengan mengirimkan data nama dan *email* yang tersimpan pada *applet* per *byte* karakter yang diakhiri dengan *byte* SW, yaitu 0x90 dan 00 yang berarti OK.

Berdasarkan Tabel V, Tabel VI dan Tabel VII, informasi yang diberikan oleh JCOP31 tersimpan pada larik *rAPDU1.buffer*, *rAPDU2.buffer*, dan *rAPDU3.buffer*, sesuai dengan urutan alamat *buffer*. Dalam program *firmware smart card reader*, *rAPDUx.buffer* memiliki panjang data *IFSD* disesuaikan dengan spesifikasi *IFSC* dari JCOP31, yaitu 254 *byte*. Bila diperhatikan, informasi *status word* untuk *rAPDU2* dan *rAPDU3* muncul pada *buffer[128]* dan *buffer[129]*, sedangkan *rAPDU1* muncul pada *buffer[0]* dan *buffer[1]*. Masing-masing *status word* menunjukkan *SW1* = 0x90 *SW2* = 0x00 yang berarti status OK.

TABEL V
RESPONS *SELECT APPLET* AID

Variabel	Alamat	Nilai Terbaca
rAPDU1.buffer[0]	0x20000654	0x90
rAPDU1.buffer[1]	0x20000655	0x00
…	…	0x00
rAPDU1.buffer[254]	0x20000757	0x00

TABEL VI
INFORMASI NAMA PEMILIK KARTU

Variabel	Alamat	Nilai Terbaca
rAPDU2.buffer[0]	0x2000044c	0x61
rAPDU2.buffer[1]	0x2000044d	0x67
rAPDU2.buffer[2]	0x2000044e	0x75
rAPDU2.buffer[3]	0x2000044f	0x73
rAPDU2.buffer[4]	0x20000450	0x20
rAPDU2.buffer[5]	0x20000451	0x62
rAPDU2.buffer[6]	0x20000452	0x65
rAPDU2.buffer[7]	0x20000453	0x6a
rAPDU2.buffer[8]	0x20000454	0x6f
…	…	0x00
rAPDU2.buffer[128]	0x200004cc	0x90
rAPDU2.buffer[129]	0x200004cd	0x00
…	…	0x00
rAPDU2.buffer[254]	0x2000054f	0x00

TABEL VII
INFORMASI EMAIL PEMILIK KARTU

Variabel	Alamat	Nilai Terbaca
rAPDU3.buffer[0]	0x20000550	0x61
rAPDU3.buffer[1]	0x20000551	0x67
rAPDU3.buffer[2]	0x20000552	0x75
rAPDU3.buffer[3]	0x20000553	0x73
rAPDU3.buffer[4]	0x20000554	0x62
rAPDU3.buffer[5]	0x20000555	0x6a
rAPDU3.buffer[6]	0x20000556	0x40
rAPDU3.buffer[7]	0x20000557	0x75
rAPDU3.buffer[8]	0x20000558	0x67
rAPDU3.buffer[9]	0x20000559	0x6d
rAPDU3.buffer[10]	0x2000055a	0x2e
rAPDU3.buffer[11]	0x2000055b	0x61
rAPDU3.buffer[12]	0x2000055c	0x63
rAPDU3.buffer[13]	0x2000055d	0x2e
rAPDU3.buffer[14]	0x2000055e	0x69
rAPDU3.buffer[15]	0x2000055f	0x64
…	…	0x00
rAPDU3.buffer[128]	0x200005d0	0x90
rAPDU3.buffer[129]	0x200005d1	0x00
…	…	0x00
rAPDU3.buffer[254]	0x20000653	0x00

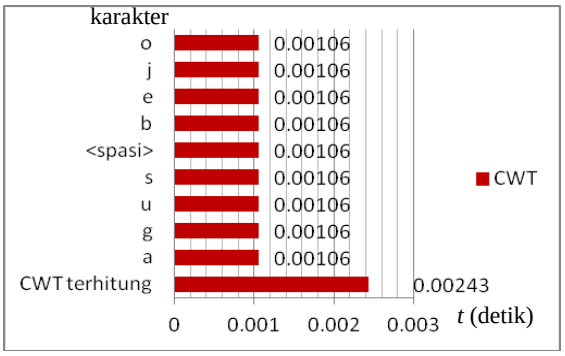
Gbr. 15 Aplikasi GUI pada *host* komputer untuk menampilkan data nama dan *email* pemilik kartu.

Informasi nama kartu yang tersimpan pada larik *rAPDU2.buffer* akan dikonversi dari bentuk heksadesimal ke dalam karakter simbol ASCII, lalu disusun menjadi satu baris yang padu untuk dikirimkan oleh *smart card reader* ke *host* komputer melalui UART. Hal yang sama juga diberlakukan saat mengirim informasi *email* pemilik kartu yang tersimpan pada *rAPDU3.buffer*. Yang membedakan hanyalah tambahan *header* identitas antara keduanya. Ketika mengirimkan informasi nama pemilik kartu, ditambahkan *header* identitas bertuliskan *"nama"*, dan ketika mengirimkan informasi *email* pemilik kartu, ditambahkan *header* identitas bertuliskan *"email"*. Selain itu, *smart card reader* juga mengirimkan status keadaan yang padanya ditambahkan *header* identitas bertuliskan *"status"*.

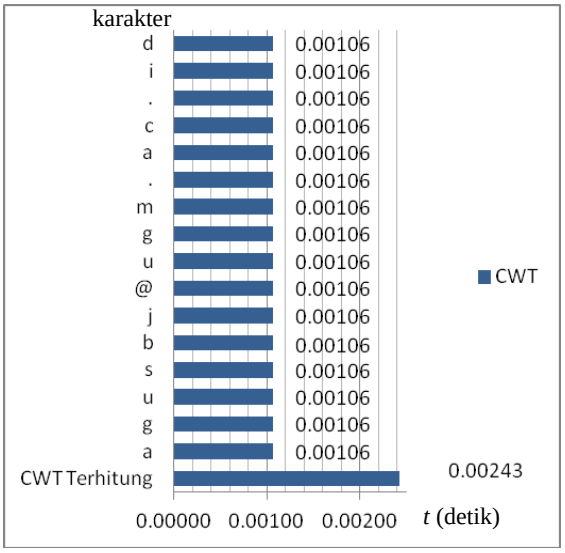
Pada host komputer, aplikasi GUI menerima informasi nama dan *email* pemilik kartu serta status keadaan melalui komunikasi serial, kemudian mengerjakan fungsi *parsing* untuk mencari informasi tersebut dan mengurai masing-masing *header* identitas. Dari hasil *parsing*, diperoleh isi informasi *applet* pemilik kartu dan status keadaan, lalu dipresentasikan pada *text label* masing-masing yang sesuai, seperti tampak pada Gbr. 15.

B. Pengujian Parameter Waiting Time

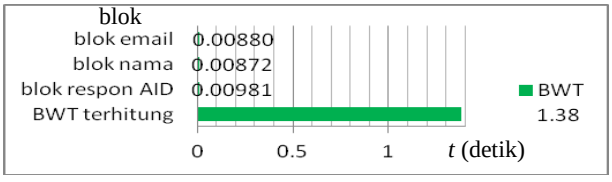
Kinerja program *firmware smart card reader* dipengaruhi oleh faktor kecepatan mengakses JCOP31. Pada operasi membaca *applet* informasi pemilik kartu, proses yang dilakukan adalah mengakses data yang tersimpan pada memori JCOP31. Seluruh data yang ada di dalam memori ditangkap satu per satu dan disimpan di larik *buffer*, ketika komunikasi antara *smart card reader* dengan JCOP31 berlangsung. Kecepatan proses pengambilan data ini dipengaruhi oleh kemampuan mikroprosesor yang ada pada *smart card* JCOP31 dan kecepatan mikrokontroler yang ada pada *smart card reader*, yaitu SMT32F4.



Gbr. 16 Grafik CWT terukur pada akses blok nama.



Gbr. 17 Grafik CWT terukur pada akses blok *email.*



Gbr. 18 Grafik perbandingan BWT terukur terhadap terhitung.

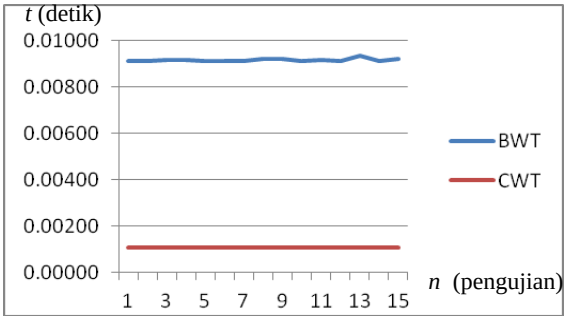
Gbr. 16 menunjukkan kemampuan proses *smart card reader* mengambil informasi nama pemilik kartu dari *smart card* JCOP31 sangat cepat, ditandai dengan waktu CWT yang lebih singkat dari waktu CWT yang direkomendasikan, sesuai dengan perhitungan pada (2). Hal yang sama juga terjadi pada proses *smart card reader* mengambil informasi *email* pemilik kartu, seperti tampak pada Gbr. 17. Dari hasil eksperimen diperoleh bahwa rerata CWT adalah sebesar 0,00106 s, sehingga kecepatan proses pembacaan karakter sebesar 2,3 kali lebih cepat daripada CWT terhitung.

Gbr. 18 menunjukkan kemampuan proses *smart card reader* menangani blok informasi antara karakter terakhir yang dikirimkan oleh sisi *smart card reader* dengan karakter awal pada blok baru yang akan dikirimkan oleh JCOP31. Diperoleh waktu BWT terukur lebih singkat dibanding waktu BWT yang direkomendasikan sesuai dengan perhitungan pada (3). Dari hasil eksperimen diketahui bahwa rerata BWT terukur sebesar 0,00911 s, yang berarti kecepatan proses pembacaan inter-blok sebesar 151,5 kali lebih cepat.

C. Analisis Keandalan

Analisis keandalan digunakan untuk mengetahui keandalan *smart card reader* dalam melakukan pembacaan kartu. Pengujian dilakukan dengan cara mengoperasikan *smart card reader* dengan pemakaian yang sering. Indikator keandalan *smart card reader* adalah upaya membaca kartu berakhir gagal atau berhasil. Hasil pengamatan keandalan kinerja *smart card reader* membaca kartu tampak pada Gbr. 19.

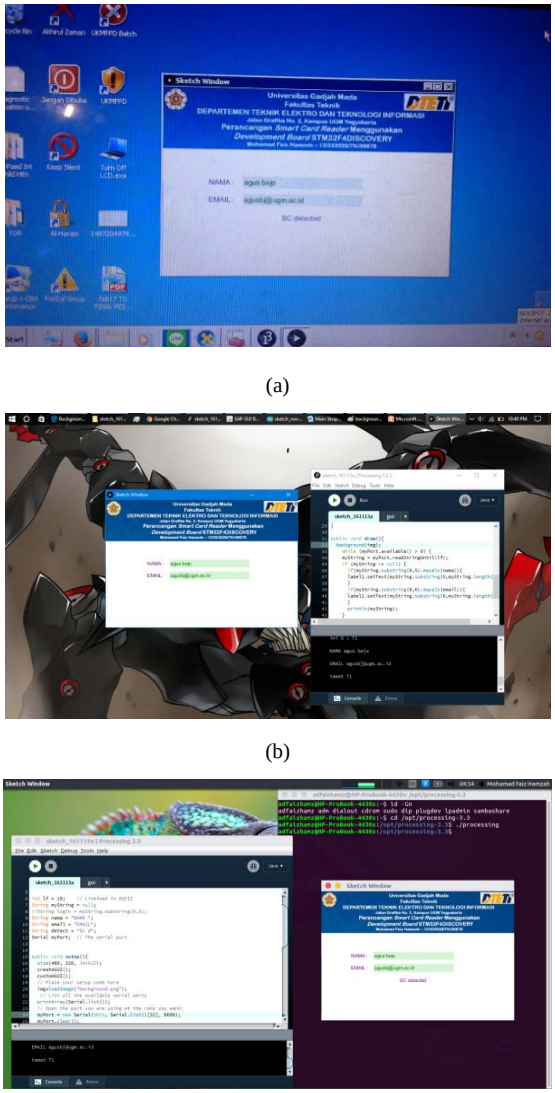
Dari hasil pengujian diketahui bahwa pada lima belas kali pengujian, *smart card reader* berhasil menjalankan tugasnya dengan baik tanpa ditemui upaya membaca kartu gagal. Bila diamati dengan seksama, grafik kinerja *smart card reader* yang diwakilkan oleh rerata CWT dan BWT terlihat cenderung stabil dan linear. Hal ini menunjukkan keandalan *smart card reader* mengatasi pemakaian sering tanpa ditemui kendala.



Gbr. 19 Grafik pengujian kehandalan sistem *smart card reader*.

D. Analisis Portabilitas

Salah satu indikator yang menandakan rancangan *smart card reader* memliki potensi untuk bersaing yakni dinilai dari kemampuan portabilitas yang baik. Pengujian ini bertujuan untuk mengertahui tingkat portabilitas *smart card reader* yang telah dirancang. Setidaknya, *smart card reader* bisa dioperasikan pada beberapa platform *host* komputer yang berbeda.



(a)

(b)

(c)

Gbr. 20 Portabilitas *smart card reader* dijalankan pada beberapa *host* komputer yang berbeda, (a) Windows 7, (b) Windows 10, dan (c) Linux Ubuntu.

Dari hasil pengujian, seperti tampak pada Gbr. 20, diketahui bahwa *smart card reader* bekerja baik di beberapa platform mesin *host* komputer yang berbeda. Hal yang perlu dilakukan adalah menginstal *driver* untuk dapat mengenali STM32F4 Discovery Kit dan mengunduh aplikasi GUI, maka *smart card reader* dapat beroperasi dengan normal semudah *plug-and-play*.

IV. KESIMPULAN

Dalam makalah ini telah dirancang sebuah *smart card reader* berbasis rangkaian minimalis mikrokontroler 32-bit STM32F407VG. *Smart card reader* ini, selain memiliki harga lebih ekonomis, juga memberikan kelebihan berupa fleksibilitas memodifikasi *firmware* sehingga dapat ditanamkan fitur khusus sesuai kebutuhan pengembang aplikasi guna meningkatkan kinerja dan keamanan aplikasinya. Hasil evaluasi dan pengujian menunjukkan bahwa sistem

smart card reader yang dirancang memiliki portabilitas yang baik untuk beberapa platform mesin yang berbeda dan kinerja andal yang ditunjukkan dengan waktu CWT dan BWT yang singkat.

REFERENSI

[1] Eurosmart, "*Eurosmart Forecasts Continued Business Growth For Smart Security Industry Double-Digit Growth In Worldwide Shipments Of Secure Elements In 2015*," Eurosmart, Brussels, 2016

[2] BPPT, "Smartcard Reader E-KTP Buatan Dalam Negeri Siap Di Produksi (Ii)," BPPT, Jakarta, 2013.

[3] Dr. David B Everett, "*ISO/IEC 7816 Part 1 : Physical Charcteristics of Integrated Circuit Cards"*, Geneva, Switzerland: ISO/IEC, 1992.

[4] Dr. David B Everett, "*ISO/IEC 7816 Part 3 : Electronic Signals and Transmission Protocols"*, Geneva, Switzerland: ISO/IEC, 1992

[5] Dr David B Everett, *"Smart Card Tutorial"*, 24 November 2004. [Online]. Available: www.smartcard.co.uk/tutorials/sct-itsc.pdf. [Last-Accessed: 15 Februari 2017].

[6] Zhiqun Chen, *"Java Card™ Technology for Smart Cards: Architecture and Programmer's Guide, California, USA"*: Pearson Education, 2000.