Mustafa Kamal Rabbani, Ajib Hanani, M.T and Dr. Fachrul Kurniawan, M.MT

**Sistem Kendali Pagar Rumah Berbasis *Internet of Things* Dengan Perintah Suara Menggunakan Algoritma *Boyer-Moore***

**[[1]](#footnote-1) *Abstract*—Currently, the development of the Internet of Things in smart homes are not merely controlled using a button but can be controlled through voice commands with speech recognition. The use of commands with a voice can also facilitate user mobility in the smart home. The fence is a very important security component in the house as a security guard for the contents of the house from criminal hazards and road dividers. This study was conducted using the ESP8266 as a microcontroller, driver stepper, and motor stepper as the driving force of the fence. This study uses the Boyer-Moore algorithm as pattern matching, the application using the Boyer-Moore algorithm can perform commands not only with the grammatical sentences but also can perform orders with a non-grammatical sentence in Indonesian that match a predetermined pattern. This study showed that the sentence commands tested are 50 sentences and each sentence is tested 10 times with 4 patterns (buka pagar, tutup pagar, pagar buka and pagar tutup). The test results get an accuracy of 100% with an average processing time of about 0.00672 ms.**

**MATICS** : Jurnal Ilmu Komputer dan Teknologi Informasi ISSN : 1978-161X(p); 2477-2550(e)

Volume 9, No. 1(2017), pp 33-40 DOI : 10.18860/mat.v9i1.XXXX

**Received** : January25th 20206; **Accepted** : March 9th 2020 ; **Avalaible Online** : April 5th 2020

*Index Terms*— Smart home, Boyer-Moore, Internet of Things, *Pattern* *matching*

*Abstrak*–-Perkembangan Internet of Things pada smart home saat ini tidak hanya dapat dikendalikan melalui tombol saja tetapi dapat dikendalikan juga melalui perintah suara dengan Speech Recognition. Penggunaan perintah dengan suara sendiri juga dapat mempermudah mobilitas pengguna dalam smart home. Pagar merupakan komponen keamanan yang sangat penting di rumah sebagai penjaga keamanan isi rumah dari bahaya kriminal dan pembatas jalan. Penelitian dilakukan menggunakan ESP8266 sebagai microcontroller, Driver Stepper dan Motor Stepper sebagai penggerak pagar rumah. Penelitian ini menggunakan algoritma *Boyer-Moore* sebagai *pattern* *matching*, penerapan algoritma *Boyer-Moore* dapat melakukan perintah tidak hanya dengan kalimat baku, tetapi juga dapat melakukan perintah dengan kalimat tidak baku dalam bahasa Indonesia yang sesuai dengan *pattern* yang sudah ditentukan. Berdasarkan penelitian yang sudah dilakukan, kalimat perintah yang diuji berjumlah 50 kalimat dan dilakukan pengujian masing – masing kalimat sebanyak 10 kali pengujian dengan 4 *pattern* (buka pagar, tutup pagar, pagar buka dan pagar tutup). Hasil pengujian mendapatkan akurasi sebesar 100 % dengan rata – rata waktu proses sekitar 0.00672 ms.

*Kata Kunci*— *Smart home, Boyer-Moore, Internet of Things, Pattern matching*.

# PENDAHULUAN

*S*

*martphone* saat ini merupakan barang yang sudah menjadi kebutuhan bagi manusia. Hal tersebut dapat dibuktikan dengan perusahaan platform media sosial dari kanada yaitu *Hootsuite* dan bekerjasama dengan *We are Social* dari Inggris yang merillis perkembangan pengguna internet di seluruh dunia, termasuk Indonesia dengan judul laporan "*Digital 2020: A comprehensive look at the state of the internet, mobile devices, social media, and e commerce*", pada tahun 2020 pengguna internet mencapai 175,4 juta jiwa dari total jumlah penduduk Indonesia yang mencapai 272,1 juta jiwa dan jumlah smartphone yang terkoneksi mencapai 338,2 juta unit, yang artinya hampir rata-rata orang Indonesia mempunyai lebih dari satu *unit* smartphone, sedangkan jumlah pengguna sosial media mencapai 160 juta jiwa [1].

Banyak hal yang dapat kita lakukan menggunakan *smartphone* dengan hanya menekan tombol. Selain *smartphone* digunakan sebagai alat komunikasi dan mencari informasi dengan mudah dan tanpa terkendala dengan jarak, salah satu contoh lainya yaitu melakukan pengontrolan terhadap suatu perangkat atau peralatan. Contoh penerapan sistem tersebut sangat banyak digunakan sekarang yaitu *smarthome* atau rumah pintar. Dimana dulu menghidupkan dan mematikan saklar masih menggunakan *remote*. Hal tersebut dirasa masih kuno dan belum mencerminkan perkembangan teknologi sekarang. Rumah pintar adalah salah satu topik menarik yang sedang trend saat ini. Pagar merupakan komponen keamanan yang sangat penting di rumah sebagai penjaga keamanan isi rumah dari bahaya kriminal dan pembatas jalan. Bahaya kriminal dapat berupa masuknya pencuri, percobaan sadap pesawat telepon, dan lain-lain [2]. Pada *smarthome* saat ini sistem kendali tidak hanya berupa tombol saja sebagai inputan tetapi suara juga dapat digunakan sebagai inputan. Perintah suara yang diberikan oleh pengguna ditafsirkan oleh perangkat *mobile* menggunakan bahasa alami dan perangkat *mobile* bertindak sebagai konsol sentral [3]. Salah satu faktor terpenting dalam *Speech recognition* adalah pengujian dari suara terhadap pola kata. Pengujian pola banyak digunakan dalam berbagai program atau aplikasi yang digunakan sehari-hari, seperti fitur pencarian pada editor teks, *web* *search* *engine* seperti *Google*, analisis citra, *bioinformatics* seperti pengujian rantai asam amino rantai DNA [4], agar memperoleh hasil kecocokan yang benar maka perlu diselesaikan dengan menggunakan algoritma *pattern* *matching*. Dalam pencarian *pattern* berupa string, maka diperlukan algoritma pengujian *string* yang dicari dengan semua string yang terdapat dalam dokumen teks tersebut. Pengujian *pattern* atau *pattern* *matching* adalah proses pencarian semua kemunculan *string* pendek P[0..n-1] yang disebut *pattern* di *string* yang lebih panjang T[0..m-1] yang disebut teks [5].

Dalam penelitian [6] menjelaskan *pattern* *matching* dengan menggunakan algortima paling sederhana adalah *bruteforce*, yaitu dengan pengujian dari kiri ke kanan atau *left-to-right*. Sebuah *pattern* akan dicocokkan karakter per karakter dalam sebuah *text*, dan ketika sebuah karakter dalam *pattern* yang ingin dicocokkan tidak sesuai dengan salah satu karakter dalam *text*, maka pencarian akan diulang dan dimulai pada huruf teks selanjutnya. Sedangkan algoritma *Boyer-Moore* adalah *pattern* *matching* dengan pencarian bersifat *right-to-left*, yakni pencarian dimulai dari huruf paling kanan pada *pattern*. Oleh karena itu penggunaan algoritma *Boyer-Moore* jauh lebih cepat daripada *bruteforce* karena pergeseran yang lebih efisien, dengan efisiensi *Boyer-Moore* lebih terlihat ketika menggunakan banyak variasi karakter dalam teks. Oleh karena itu penelitan yang diajukan berjudul “Sistem Kendali Pagar Rumah Berbasis *Internet of Things* dengan Perintah Suara Menggunakan Algoritma *Boyer-Moore*”

# tinjauan pustaka

## Pattern Matchin

Pengujian *pattern* atau *pattern* *matching* yang berupa string adalah proses pencarian jumlah kejadian kemunculan dari *pattern* P dengan panjang m terhadap *text* T dengan panjang n. Pengujian string atau string *matching* adalah proses pencarian semua kemunculan string pendek P[0..m-1] yang disebut *pattern* di string yang lebih panjang T[0..n-1] yang disebut teks. Pengujian string merupakan permasalahan paling sederhana dari semua permasalahan string lainnya, dan merupakan bagian dari pemrosesan data, pengkompresian data, *lexical analysis*, dan temu balik informasi [5].

Prinsip kerja dari algoritma *pattern* *matching* adalah, pertama memindai *text* dengan bantuan sebuah *window* yang ukuran panjangnya sama dengan Panjang *pattern*, lalu menempatkan *window* pada awal *text* agar sejajar, kemudian membandingkan karakter *text* pada *window* dengan karakter dari *pattern*, setelah pengujian dengan hasil cocok atau tidak cocok, maka dilakukan pergeseran *window* ke kanan. Proses ini dilakukan berulang-ulang sampai *window* berada di tepat akhir *text*.

## Boyer-Moore

Algoritma *Boyer-Moore* adalah salah satu algoritma pencarian string, yang diterbitkan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1997 [7]. Algoritma *Boyer-Moore* telah dibuktikan sebagai salah satu algoritma dalam *pattern* *matching* paling efisien dalam aplikasi pencarian string dengan menggunakan natural language. Pada dasarnya algoritma ini memiliki kesamaan dengan algoritma *Knuth-Morris-Pratt* (KMP) dalam segi cara kerjanya dimana kedua algoritma ini melakukan lompatan pengecekan dalam proses pencarian string. Namun berbeda dengan algoritma *Knuth-Morris-Pratt* (KMP),

Algoritma *Boyer-Moore* melakukan perbandingan *pattern* dimulai dari kanan ke kiri atau *right-to-left*, sedangkan algoritma *Knuth-Morris-Pratt* (KMP) melakukan perbandingan *pattern* dimulai dari kiri ke kanan atau *left-to-right*. Dalam algortima *Boyer-Moore* menggunakan dua heuristics untuk memutuskan seberapa jauh lompatan agar mendapat hasil pengujian yang pas yaitu: *bad-character heuristic* atau juga bisa disebut *Occurrence Heuristic* (OH), dan *good-character heuristic* atau juga disebut *match heuristic* (MH). Informasi untuk heuristik setiap dipertahankan dalam sebuah array yang dibangun pada awal operasi yang cocok [8].

## Speech Recognition

*Speech recognition* adalah proses identifikasi suara berdasarkan kata yang diucapkan dengan cara melakukan konversi sinyal akustik yang ditangkap oleh audio device atau proses untuk identifikasi suara seseorang berdasarkan dari kalimat atau kata yang diucapkan [9]. *Speech recognition* yang digunakan oleh penulis adalah *Google* *Speech recognition* dimana digunakan untuk mengubah suara menjadi teks yang dimana aplikasi akan menggunakan hasil tersebut sebagai *text* yang akan dibandingkan dengan *pattern* yang tersedia. Dengan menggunakan *Google* *Speech recognition* akan mempermudah pengguna untuk mengontrol pagar dengan cara pengguna cukup mengucapkan perintah.

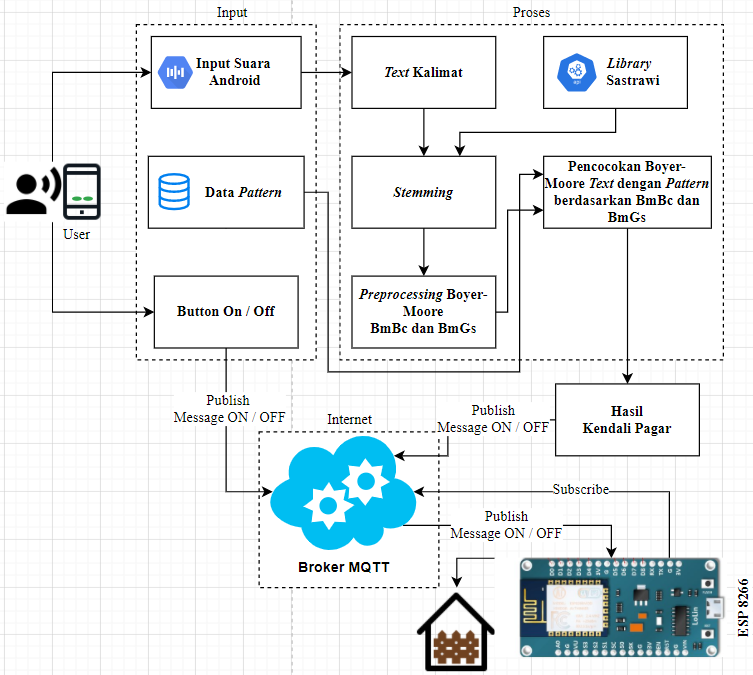
## Stemming

*Stemming* merupakan suatu proses menemukan kata dasar atau root word dari suatu kata yang berimbuhan atau affixed word dengan cara menghilangkan semua imbuhan atau affix yang terdiri dari awalan (*prefix*), akhiran (*suffix*), sisipan (*infix*) dan kombinasi kombinasi dari awalan dan akhiran atau confix. Dalam penelitian [10] menjelaskan bahwa proses steaming ini merupakan cara untuk meningkatkan performa Information Retriveal (IR).

# METODOLOGI PENELITIAN

## Desain Sistem

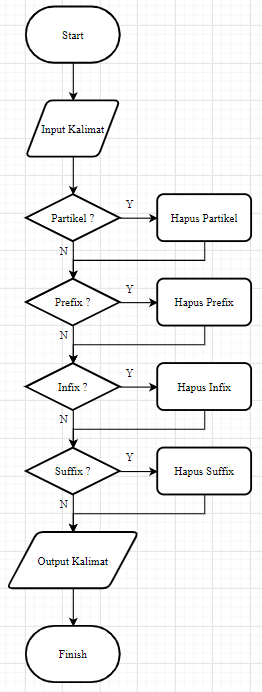
Desain system pada penelitian ini digambarkan pada alur diagram sistem Gambar 1



Gambar 1 Desain Sistem

## Stemming

*Stemming* merupakan proses mengubah kalimat atau kata yang memiliki imbuhan menjadikan kata atau kalimat yang baku jadi kata atau kalimat tersebut yang akan diproses pada sistem adalah kata-kata dasar saja dalam Bahasa Indonesia. Proses steeming ini peneliti menggunakan *library* Sastrawi dengan bahasa pemrograman php. Alur *Stemming* ditunjukan pada Gambar 2.



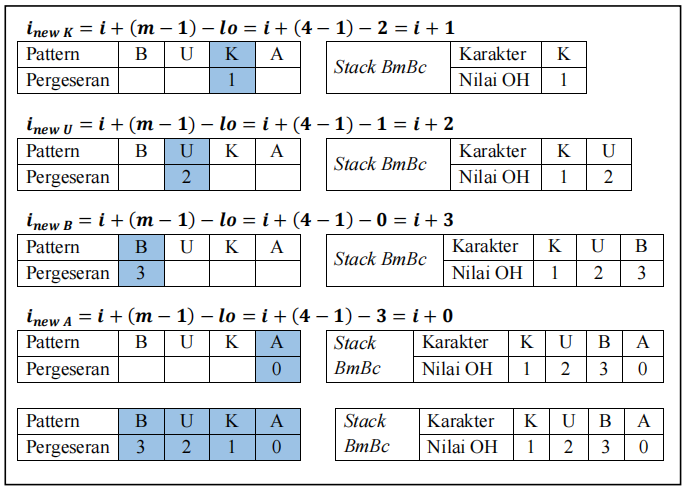
Gambar 2 *Flowchart Stemming*

## Boyer-Moore

### Preprocessing Boyer-Moore

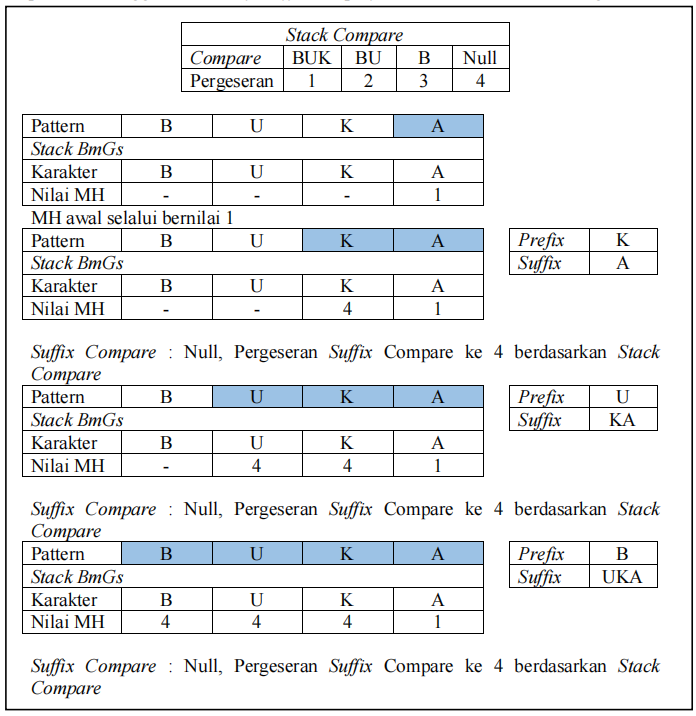
*Boyer-Moore* memiliki tahap Preprocessing terdiri dari 𝒑𝒓𝒆𝑩𝒎𝑩𝒄 dan 𝒑𝒓𝒆𝑩𝒎𝑮𝒔. Prosedur 𝒑𝒓𝒆𝑩𝒎𝑩𝒄 memiliki nilai *Occurrence Heuristic* (OH). Langkah-langkah pada proses nya, pertama membuat *stack* dari BmBc dan diberi nilai kosong, lalu melakukan perhitungan terhadap panjang *pattern* yang akan dimasukan nilai OH kedalam *stack* BmBc.

Contoh kasus *pattern:* Buka



Gambar 3 Penyelesaian prosedur PreBmBc

Pada Gambar 3 dijelaskan bahwa karakter yang di pilih akan ditambahkan ke dalam *stack* BmBc dengan nilai OH sesuai pergeseran. Prosedur 𝒑𝒓𝒆𝑩𝒎𝑮𝒔 memiliki nilai *Match heuristic* (MH). Langkah-langkah pada prosesnya yaitu, pertama membuat *stack* dari BmGs dengan nilai kosong dan tabel *Compare* berisi sejumlah karakter sebelah kanan patter dari pergeseran kanan ke kiri, lalu cacah *pattern* kemudian simpan pada tabel *Compare*. Kemudian cacah *prefix* dan *suffix* dari *pattern* kemudian simpan hasil *prefix* pada tabel *stack* BmGs dan bandingkan hasil *suffix* terhadap *Compare*



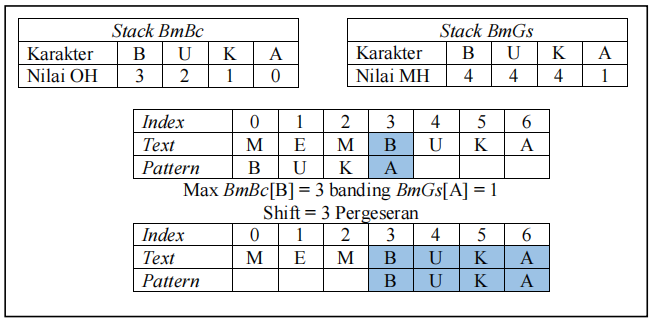
Gambar 4 Penyelesaian Prosedur PreBmGs

Pada Gambar 4 dijelaskan bahwa penentuan nilai dari MH ditentukan dari pergeseran pada *Compare*, nilai MH selalu diberikan kepada semua karakter yang ada pada *pattern* dan penyusunan *stack* BmGs berdasarkan urutan *index* pada *pattern*.

### Pencocokan Boyer-Moore

Pada tahap ini pengujian dimulai dari *index* paling kiri pada *text* dan pengujian per karakter dimulai paling kanan pada *pattern*, lalu setiap kali ditemukan ketidak cocokan maka bandingkan nilai OH dan MH pada *stack* BmBcdan BmGs yang sudah dibuat pada tahap Preprocessing sebelumnya.

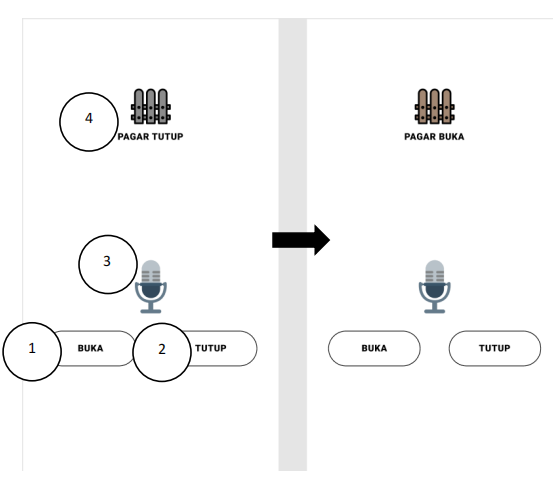
Contoh kasus *Pattern* : buka, *Text* : membuka



Gambar 5 Penyelesaian Pengujian Boyer-Moore

## Perancangan Antarmuka (Interface)

Proses perancangan antarmuka (Interface) pada penelitian ini dirancang agar dapat memberikan kemudahan dalam membangun sistem sehingga dapat dipahamu tujuan dan fungsi dari setiap tampilan pada sistem tersebut. Pada Gambar 6 adalah tampilan awal user.



Gambar 6 Perancangan Antarmuka

Keterangan Gambar :

1. Button Buka digunakan untuk membuka pagar secara manual,

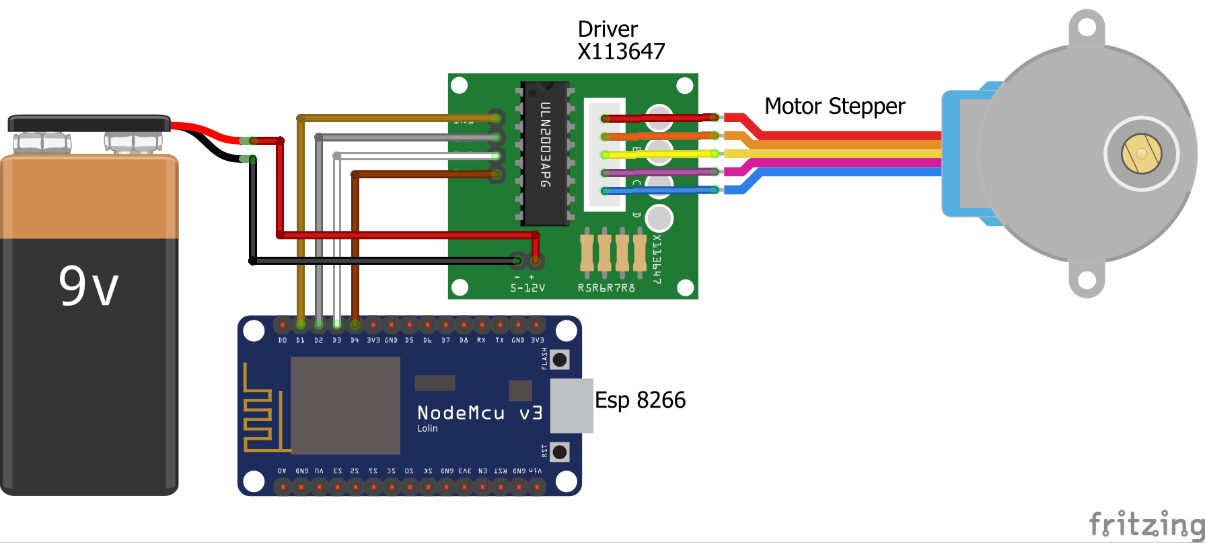
2. Button Tutup digunakan untuk menutup pagar secara manual,

3. Button *Speech recognition* digunakan untuk kendali pagar dengan menggunakan suara.

4. ImageView untuk menampilakn status pada pagar

## Rangkaian Komponen

Rangkaian komponen elektronik ini berperan sebagai blueprint dalam membangun sistem. Dalam penelitian ini menggunakan ESP8266 sebagai microcontroller, Driver Stepper, Motor Stepper sebagai penggerak dan Baterai 9V sebagai power, gambar rangkaian komponen dapat dilihat pada Gambar 7 sebagai berikut.



Gambar 7 Rangkaian Komponen

## Uji coba sistem

Pengujian sistem dilakukan untuk memperoleh hasil apakah sistem yang dibangun sudah akurat atau tidak, untuk mengukur algortima *Boyer-Moore* perlu digunakan *Confussion Matrix*. *Confussion Matrix* terdiri dari *True positive* (TP), *False Positive* (FP), Fal*se Negative* (FN), dan *True Negative* (TN). Dalam penelitian [11] menjelaskan bahwa True *positive* merepresentasikan data yang berada pada kelas positif yang diprediksi secara benar oleh algoritma. *False* *Positive* merepresentasikan data yang seharusnya berada pada kelas positif diprediksi menjadi kelas negatif oleh algoritma. *False Negative* merupakan data yang seharusnya berada di kelas *negative*. diprediksi menjadi kelas positif oleh algoritma. *True Negative* merupakan data yang berada pada kelas *negative* dan diprediksi secara benar oleh algoritma.

(1)

Akurasi adalah proses matematika yang sering digunakan untuk mengevaluasi kinerja suatu algoritma. Akurasi dihitung berdasarkan rasio jumlah data yang diprediksi oleh algoritma dengan benar dan jumlah semua data dalam kumpulan data [11]. Adapun daftar kalimat dan *pattern* yang akan diuji yaitu 50 kalimat perintah dengan 4 *pattern* (buka pagar, tutup pagar, pagar buka, pagar tutup) sebagai berikut:

Kalimat yang cocok dengan *pattern* buka pagar

|  |  |
| --- | --- |
| - Buka pagar  - Tolong buka pagar  - Buka pagar rumah  - Tolong buka pagar rumah  - Tolong dibuka pagar  - Dibuka pagar rumah  - Tolong dibuka pagar rumah  - Tolong dibukakan pagar  - Dibukakan pagar rumah  - Tolong dibukakan pagar rumah | - Tolong bukakan pagar rumah  - Tolong membuka pagar  - Membuka pagar rumah  - Tolong membuka pagar rumah  - Tolong membukakan pagar  - Membukakan pagar rumah  - Tolong membukakan pagar rumah  - Tolong bukakan pagar  - Bukakan pagar rumah |

Kalimat yang cocok dengan *pattern* tutup pagar

|  |  |
| --- | --- |
| - Tutup pagar  - Tolong tutup pagar  - Tutup pagar rumah  - Tolong tutup pagar rumah  - Tolong ditutup pagar  - Ditutup pagar rumah  - Tolong ditutup pagar rumah  - Tolong ditutupkan pagar  - Ditutupkan pagar rumah  - Tolong ditutupkan pagar rumah | - Tolong tutupkan pagar  - Tutupkan pagar rumah  - Tolong tutupkan pagar rumah  - Tolong menutup pagar  - Menutup pagar rumah  - Tolong menutup pagar rumah  - Tolong menutupkan pagar  - Menutupkan pagar rumah  - Tolong menutupkan pagar rumah |

Kalimat yang cocok dengan *pattern* pagar buka

|  |  |
| --- | --- |
| - Pagar buka  - Tolong pagar buka  - Pagar dibuka | - Tolong pagar dibuka  - Pagar dibukakan  - Tolong pagar dibukakan |

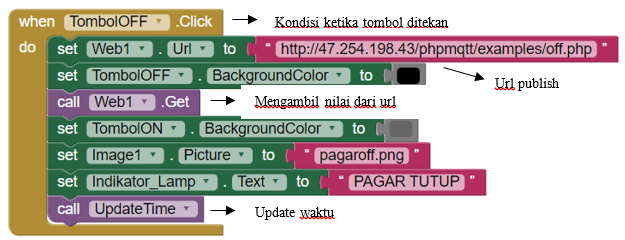
Kalimat yang cocok dengan *pattern* pagar tutup

|  |  |
| --- | --- |
| - Pagar buka  - Tolong pagar buka  - Pagar dibuka | - Tolong pagar dibuka  - Pagar dibukakan  - Tolong pagar dibukakan |

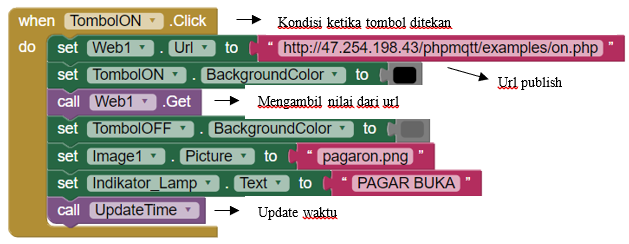
# UJI COBA DAN PEMBAHASAN

## Kendali dengan button

Pada proses pengendalian dengan button ini, *Block code* pada *App Inventor* dapat dilihat pada Gambar 8 (a) dan (b) sebagai berikut.



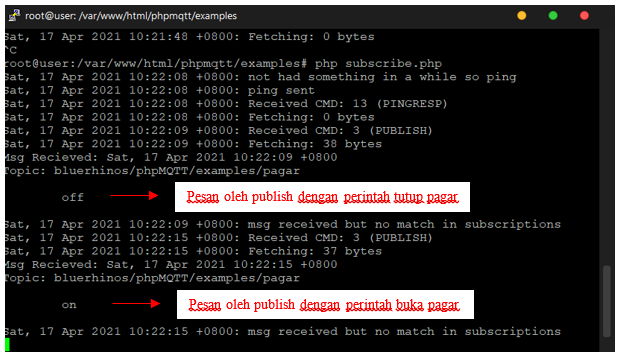
(a)



(b)

Gambar 8 (a) *Block code* button off (b) *Block code* button on

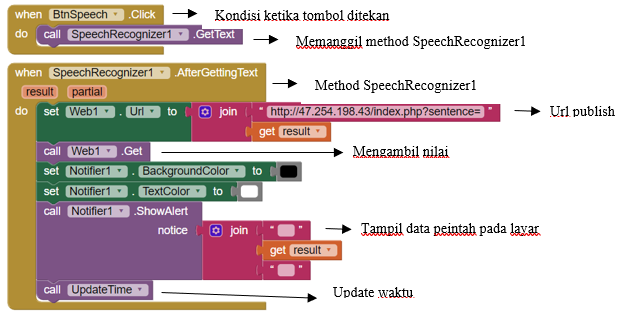
Pada Gambar 8 (a) merupakan *Block code* dari button off dimana ketika button off ditekan makan akan memanggil link pada gambar tersebut, *link* tersebut merupakan code dari PHPMQTT yang sudah tersimpan pada *Alibaba Cloud ECS*. Pada Gambar 8 (b) sama dengan Gambar 8 (a) *Block code* dari button on memanggil link sesuai pada gambar tersebut. Ketika setiap button ditekan maka akan mengirikan pesan sesuai button yang ditekan dan bertindak sebagai *publish*, pesan kemudian di terima oleh *subscribe* dan akan menampilkan pesan, topic, dan waktu yang dikirim oleh *publish*, dapat dilihat pada Gambar 9 sebagai berikut.



Gambar 9 Pesan yang diterima oleh *subscribe*

## Kendali dengan Speech Recognition

Pada proses dengan *Speech recognition* ini, *Block code* pada *App Inventor* dapat dilihat pada Gambar 10 sabagai berikut.

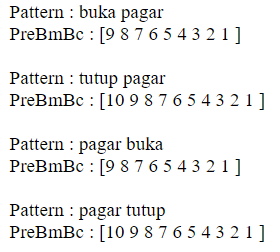


Gambar 10 *Block code* Speech Recognition

Pada Gambar 10 tersebut Ketika button ditekan dan mendapat perintah dalam Bahasa Indonesia akan memanggil link seperti pada gambar, dimana link tersebut akan menerima variabel “sentence” berupa *text* untuk diproses ke dalam algoritma Boyer-Moore.

## Preprocessing Boyer-Moore

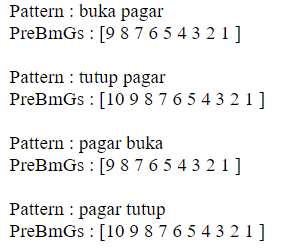
Proses preBmBc dilakukan agar mendapatkan nilai OH dimana nilai OH dinilai mulai dari *index* paling kanan pada *pattern*. Dalam proses preBmBc dilakukan pencacahan masing-masing karakter pada *pattern* dimulai dari *index* kanan sampai *index* paling kiri. Nilai pada setiap *index* diberikan sesuai rumus. Adapaun hasil nilai OH pada masing-masing *pattern* dapat dilihat pada Gambar 11 sebagai berikut.



Gambar 11 Nilai OH pada masing-masing *pattern*

Pada Gambar 11 tersebut merupakan hasil dari nilai OH pada masing-masing *pattern* yang sudah melalui proses pergeseran dan pencacahan.

Proses preBmGs dlakukan agar mendapat nilai MH, agar mendapat nilai MH perlu dilakukan dengan membentuk *suffix* dari kanan ke kiri dari *pattern* dan perbandingan *suffix* kiri ke kanan dari *pattern*. Kemudian pemberian nilai MH dilakukan pencacahan dimulai dari *index* paling kanan *pattern* ke *index* paling kiri *pattern*, pada *index* paling kanan *pattern* selalu diberi nilai MH=1, selanjutnya dilakukan pengujian terhadap *suffix* *Compare*. Apabila dalam proses pengujian terdapat kesamaan terhadap *suffix*, maka nilai MH karakter pada *pattern* sama dengan nilai pergeseran terhadap *suffix*, dan jika tidak terdapat kesamaan maka nilai MH karakter pada *pattern* bernilai sebanyak jumlah karakter ada patteen. Adapun hasil nilai MH dari proses preBmGs pada masing – masing *pattern* dapat dilihat pada Gambar 12 sebagai berikut.

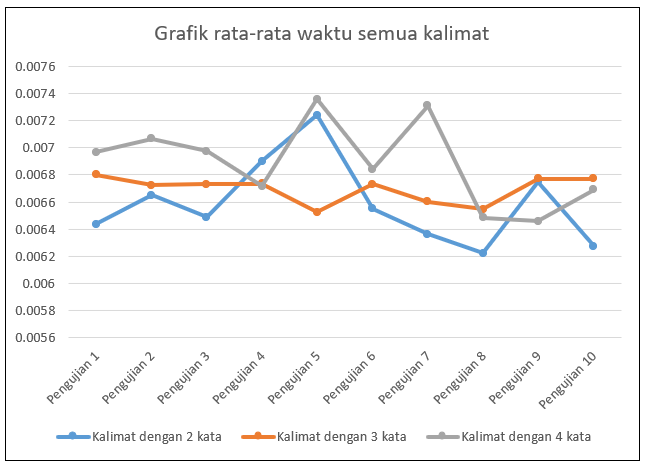


Gambar 12 Nilai MH pada masing-masing *pattern*

Pada Gambar 12 tersebut merupakan hasil dari nilai MH pada masing-masing *pattern* yang sudah melalui proses pergeseran dan pencacahan

## Pengujian Algoritma Boyer-Moore

Algoritma *Boyer-Moore* pada sistem kendali pagar rumah ini diimplementasikan pada pengujian kalimat perintah *text* terhadap *pattern*, jumlah kalimat pengujian yang digunakan pada penelitian ini adalah 50 kalimat atau *text* dengan 4 *pattern*. Perhitungan akurasi pada sistem kendali pagar rumah dengan algoritma *Boyer-Moore* ini dilakukan dengan 10 kali pengujian pada setiap kalimat perintah menggunakan *Confussion Matrix*. Berikut hasil grafik waktu dari semua kalimat perintah yang terdiri dari 2 kata, 3 kata dan 4 kata dapat dilihat pada Gambar 13 sebagai berikut.



Gambar 13 Grafik rata – rata waktu semua kalimat

Pada Gambar 13 tersebut merupakan grafik waktu dari pengujian semua kalimat yang terdiri dari 2 kata, 3 kata dan 4 kata, rata – rata yang dibutuhkan dalam pengujian untuk kalimat yang terdiri dari 2 kata adalah 0.00657 ms dan kalimat yang memiliki waktu terlama dalam 2 kata adalah pada kalimat ke 20 yaitu “Tutup pagar” pada pengujian 4 dan pengujian 5 dengan lama waktu 0.0099 ms, rata – rata yang dibutuhkan dalam pengujian untuk kalimat yang terdiri dari 3 kata adalah 0.00659 ms dan kalimat yang memiliki waktu terlama dalam 3 kata adalah pada kalimat 17 yaitu “Tolong membukakan pagar” pada pengujian 1 dan pengujian 9 dengan lama waktu masing – masing 0.01 ms dan 0.0103 ms, rata – rata yang dibutuhkan dalam pengujian untuk kalimat yang terdiri dari 4 kata adalah 0.00689 ms dan kalimat yang memiliki waktu terlama dalam 4 kata adalah pada kalimat 16 yaitu “Tolong membuka pagar rumah” pada pengujian 1 dengan lama waktu 0.0104 ms. Pada Gambar 13 tersebut dilakukan 10 kali pengujian pada sertiap kalimat dan memiliki selisih perubahan selisih waktu yang tidak sangat jauh. Dapat dilihat pada grafik garis warna abu – abu yaitu kalimat yang terdiri dari 4 kata memiliki waktu yang lebih lama dari kalimat yang terdiri dari 2 kata dan 3 kata. Lama waktu pegujian tersebut tergantung dari panjang *text* dari kalimat yang diuji, dapat dilihat pada rata – rata waktu dalam 2 kata yaitu 0.00657 ms sedangkan 3 kata yaitu 0.00659 ms dan 4 kata yaitu 00.00689 ms. Berdasarkan hasil uji coba dengan 10 kali pengujian pada 50 kalimat perintah dan hasil yang tepat dimasukan ke dalam rumus akurasi, maka didapatkan hasil akurasi sebesar 100% dengan rata – rata waktu 0.00672 ms.

# PENUTUP

## Kesimpulan

Penelitian yang terlah dilakukan ini adalah penelitian tentang kendali pagar rumah dengan algortima Boyer-Moore. Berdasarkan hasil uji pada penelitian ini sudah berjalan dengan baik, algortima *Boyer-Moore* menghasilkan akurasi sebesar 100% dari 50 kalimat percobaan dengan 10 kali pengujian setiap kalimat perintah dan 4 *pattern* yang digunakan dengan durasi waktu pengujian keseluruhan kalimat dengan rata-rata 0.00672 ms. Lama waktu pegujian tersebut tergantung dari panjang *text* dari kalimat yang diuji, pada rata – rata waktu dalam kalimat yang terdiri dari 2 kata yaitu 0.00657 ms sedangkan 3 kata yaitu 0.00659 ms dan 4 kata yaitu 00.00689 ms.

## Saran

Sistem kendali pagar rumah dengan algoritma *Boyer-Moore* ini masih memiliki kekurangan, sehingga diperlukannya pengembangan sehingga diperoleh hasil yang lebih baik, berikut saran dari dari peneliti dapat bermanfaat bagi penelitian selanjutnya antara lain:

1. Pengembangan lebih lanjut sistem tidak hanya dapat mengontrol pagar saja, tetapi kedepan dapat mengontrol banyak device.
2. Kendali dengan *Speech recognition* dapat dengan bermacam – macam Bahasa.

DAFTAR PUSTAKA

[1] cyberthreat.id, “Digital 2020: Pengguna Internet Indonesia dalam Angka,” 2020. https://cyberthreat.id/read/5387/Digital-2020-Pengguna-Internet-Indonesia-dalam-Angka

[2] H. S. Utama, J. Setiawan, and P. B. Mardjoko, “Sistem Kontrol Pintu Pagar Rumah Berbasis Arduino dengan Koneksi Nirkabel Bluetooth pada Smartphone Android,” *TESLA*, vol. 21, no. 2, p. 135, Jan. 2020, doi: 10.24912/tesla.v21i2.7184.

[3] M. P. J. Rani, J. Bakthakumar, and S. Kumar, “Voice Controlled Home Automation System Using Natural Language Processing (Nlp) And Internet Of Things (IoT),” *International Conference on Science Technology Engineering & Management (ICONSTEM)*, p. 6, 2017.

[4] Nursyahrina, “Aplikasi Pencocokan Pola pada Asisten Pribadi Cerdas *Google* Now,” *Makalah IF2211 Strategi Algoritma*, 2016.

[5] G. H. Kumara, “Visualisasi Beberapa Algoritma Pencocokan String Dengan Java,” *Institut Teknologi Bandung*, p. 14, 2008.

[6] B. D. Junaidi, “Implementasi *Pattern* *Matching* pada Speech Recognition,” *Institut Teknologi Bandung, Makalah IF2211 Strategi Algoritma*, p. 9, 2018.

[7] R. I. Darmawan and A. H. Setianingrum, “Implementasi Algoritma Boyer Moore Pada Aplikasi Kamus Istilah Kebidanan Berbasis Web,” *Jurnal Sistem Informasi*, vol. 02, no. 01, p. 10, 2018.

[8] J. Orwant, J. Hietaniemi, and J. Macdonald, *Mastering Algorithms R’ith Perl*. O’Reilly United States, 1999.

[9] J. Dirmansyah and D. Hermanto, “Rancang Bangun Aplikasi Penunjuk Arah Berbahasa Indonesia Berbasis *Text* To Speech Dan *Speech recognition* Pada Perangkat Android,” *Fakultas Teknologi Informasi Dan Elektro Universitas Teknologi Yogyakarta*, p. 8, 2014.

[10] J. Wibowo, “Aplikasi Penentuan Kata Dasar Dari Kata Berimbuhan Pada Kalimat Bahasa Indonesia Dengan Algoritma *Stemming*,” *Jurnal Riset Komputer (JURIKOM)*, vol. Volume : 3, Nomor: 5, p. 5, 2016.

[11] O. S. Y. Prakasa and K. M. Lhaksamana, “Klasifikasi Teks Dengan Menggunakan Algoritma K-nearest Neighbor Pada Kasus Kinerja Pemerintah Di Twitter,” *e-Proceeding of Engineering*, vol. Vol.5, No.3, p. 12, 2018.

1. [↑](#footnote-ref-1)