Penerapan Algoritma String Matching dalam Sensor Ultrasonik menggunakan Arduino Uno untuk Deteksi Pola Gerakan

Hugo Sabam Augusto- 13522129[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522129@std.stei.itb.ac.id

***Abstract*— Penerapan algoritma string matching dalam sensor ultrasonik menggunakan Arduino Uno menawarkan pendekatan inovatif dalam deteksi pola gerakan. Sistem ini dirancang untuk mengidentifikasi dan menganalisis gerakan berdasarkan data jarak yang dihasilkan oleh sensor ultrasonik dan dikonversi menjadi karakter. Algoritma string matching digunakan untuk membandingkan pola data gerakan dengan pola referensi yang telah disimpan sebelumnya, memungkinkan deteksi pola yang presisi dan real-time. Penelitian ini mengeksplorasi implementasi algoritma ini, menguji keakuratannya, serta mengevaluasi kinerjanya dalam berbagai skenario gerakan. Hasil dari eksperimen menunjukkan bahwa penggunaan algoritma string matching dapat meningkatkan efisiensi dan keandalan dalam aplikasi deteksi gerakan berbasis sensor ultrasonik.**

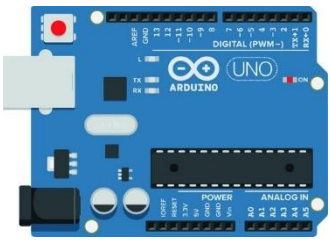
***Kata Kunci*— arduino, sensor ultrasonik, deteksi pola gerakan,string matching**

I. PENDAHULUAN

Deteksi pola gerakan merupakan teknologi yang berkembang pesat dan memiliki berbagai aplikasi praktis dalam kehidupan sehari-hari dan industri. Teknologi ini melibatkan identifikasi dan analisis pola gerakan objek atau manusia dalam lingkungan tertentu, yang dapat diterapkan dalam bidang keamanan, otomasi, kesehatan, olahraga, dan interaksi manusia-komputer. Pada intinya, deteksi pola gerakan bertujuan untuk mengenali urutan atau bentuk spesifik dari gerakan berdasarkan data yang dikumpulkan oleh berbagai jenis sensor. Misalnya, dalam aplikasi keamanan, pola gerakan dapat digunakan untuk mendeteksi aktivitas mencurigakan di area yang diawasi. Dalam otomasi industri, pola gerakan dapat membantu dalam pemantauan dan kontrol mesin. Sementara itu, di sektor kesehatan, deteksi pola gerakan dapat digunakan untuk memantau aktivitas fisik pasien dan mendeteksi potensi masalah kesehatan.

Salah satu komponen penting dalam sistem deteksi pola gerakan adalah sensor yang digunakan untuk mengumpulkan data gerakan. Sensor ultrasonik adalah pilihan populer karena kemampuannya untuk mengukur jarak dengan akurasi tinggi. Sensor ini bekerja dengan memancarkan gelombang ultrasonik dan mengukur waktu yang dibutuhkan untuk gelombang tersebut kembali setelah memantul dari objek. Data jarak yang diperoleh ini dapat digunakan untuk menganalisis pergerakan objek atau manusia. Untuk meningkatkan efisiensi dan akurasi dalam deteksi pola gerakan, data mentah yang dikumpulkan dari sensor perlu dianalisis menggunakan algoritma yang tepat. Salah satu algoritma yang dapat digunakan adalah algoritma string matching. Algoritma string matching, yang umumnya digunakan dalam pemrosesan teks, dapat diterapkan untuk mencocokkan pola gerakan dengan pola referensi yang telah ditetapkan sebelumnya. Dengan membandingkan data gerakan yang diterima dengan pola-pola yang dikenal, sistem dapat mengidentifikasi jenis gerakan dengan cepat dan tepat.

Algoritma string matching adalah teknik yang digunakan untuk mencari kemunculan pola tertentu dalam sebuah string atau urutan data. Dalam konteks deteksi pola gerakan, algoritma ini dapat digunakan untuk mencocokkan urutan data gerakan yang dihasilkan oleh sensor dengan pola gerakan yang sudah dikenal. Salah satu algoritma string matching yang efisien adalah algoritma Knuth-Morris-Pratt (KMP). Algoritma KMP dirancang untuk mempercepat proses pencocokan dengan cara menghindari perbandingan yang tidak perlu. Algoritma ini menggunakan informasi dari pola yang dicari untuk menentukan langkah berikutnya dalam pencocokan, sehingga mengurangi jumlah perbandingan yang diperlukan. KMP bekerja dengan membangun sebuah tabel yang disebut sebagai "border function" yang membantu menentukan seberapa jauh pola harus bergeser ketika terjadi ketidaksesuaian karakter selama pencocokan.



Gambar 1.1 Arduino Uno
(Sumber : https://stock.adobe.com)

Arduino Uno adalah salah satu varian paling populer dari keluarga Arduino, yang menggunakan mikrokontroler ATmega328P. Papan ini memiliki 14 pin input/output digital, 6 pin input analog, dan koneksi USB yang memudahkan pemrograman dan komunikasi dengan komputer. Arduino Uno

sering digunakan dalam proyek DIY, pendidikan, dan prototipe karena kemudahan penggunaannya, dukungan komunitas yang luas, serta fleksibilitas dalam integrasi dengan berbagai sensor dan modul. Arduino Uno dapat diprogram menggunakan Arduino Software (IDE), yang menyediakan lingkungan pengembangan yang intuitif dan mendukung berbagai perpustakaan untuk memperluas fungsionalitas papan. Keterjangkauan dan kemudahan penggunaan menjadikan Arduino Uno sebagai pilihan utama untuk eksperimen elektronik dan pengembangan cepat.fleksibilitas, menjadikannya populer di kalangan pengembang dan pembuat.

Gambar 1.2 Sensor Ultrasonic HC-SR04
(Sumber : https://www.electronicwings.com/arduino/ultrasonic-sensor-hc-sr04-interfacing-with-arduino-uno)

Deteksi pola gerakan adalah teknologi yang penting untuk berbagai aplikasi, seperti keamanan, otomasi, dan interaksi manusia-komputer. Dengan menggabungkan Arduino Uno dan sensor ultrasonik, kita dapat membangun sistem deteksi pola gerakan yang andal dan efisien. Sensor ultrasonik berfungsi dengan memancarkan gelombang ultrasonik dan mengukur waktu yang diperlukan untuk gelombang tersebut kembali setelah memantul dari objek. Data jarak yang dihasilkan dari sensor ini dapat digunakan untuk mengidentifikasi gerakan objek. Untuk meningkatkan akurasi deteksi, algoritma string matching dapat diterapkan pada data yang dikumpulkan.

II. LANDASAN TEORI

A. String Matching

String matching adalah proses pencarian kemunculan pola tertentu dalam sebuah string teks. Pola, yang juga disebut sebagai pattern, adalah urutan karakter yang ingin dicari dalam teks. Tujuan utama dari string matching adalah untuk menemukan posisi atau lokasi dari setiap kemunculan pola dalam teks. Misalnya, dari definisi diberikan sebuah teks T, sebuah string yang panjangnya n karakter dan pola P, yaitu string dengan panjang m karakter asumsi m < n yang akan dicari dalam teks T. String matching disini bertujuan untuk mencari lokasi pertama di dalam teks yang bersesuaian dengan pola P. Contohnya sebagai berikut

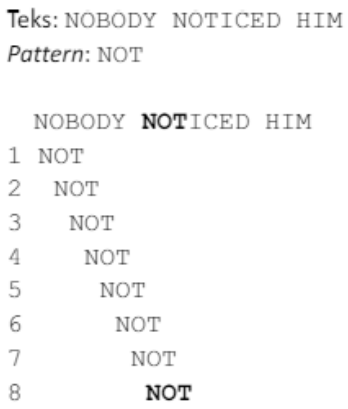
```
T : hari ini aku makan ayam geprek
sambal
P : prek
```

Untuk penerapan deteksi pola gerakan, akan digunakan 2 algoritma string matching yaitu Brute Force dan Knuth-Morris-Pratt (KMP).

B. Brute Force

Pencocokan string dengan brute force adalah salah satu teknik sederhana untuk mencari keberadaan pola atau substring tertentu dalam sebuah teks. Teknik ini bekerja dengan cara membandingkan setiap karakter dari pola dengan setiap karakter dari teks, satu per satu, secara berurutan. Langkah-langkah Metode Brute Force:

1. mengambil satu karakter pertama dari P dan membandingkannya dengan karakter pertama dari T. Jika kedua karakter tersebut cocok, lanjutkan ke langkah berikutnya. Jika tidak, pindah ke karakter kedua dari teks.
2. Setelah karakter pertama dari P cocok dengan karakter pada posisi yang sama di T, lanjutkan dengan membandingkan karakter kedua dari P dengan karakter pada posisi kedua di T. Lanjutkan proses ini sampai semua karakter pada pola telah dibandingkan.
3. Jika semua karakter pada P cocok dengan karakter pada posisi yang sesuai di T, maka pola ditemukan dalam T. Jika tidak, geser posisi P satu karakter ke kanan dan ulangi langkah-langkah tersebut.
4. Proses ini akan terus berlanjut sampai P ditemukan dalam teks atau sampai seluruh T telah diperiksa.



Gambar 2.1 Ilustrasi Brute-Force
(Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf)

Brute Force Tidak efisien untuk penggunaan dalam skenario di mana pola memiliki kemungkinan terjadi secara berulang dalam teks dan untuk teks dan pola yang besar karena memerlukan banyak perbandingan karakter. Misalnya seperti berikut:

```
T : zzzzzzzzzzzbzzcz
P : zzzb
```

Tetapi, Brute Force menjadi sangat efisien ketika karakter pertama pattern P tidak pernah sama dengan karakter teks T yang dicocokan.

C. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) diciptakan oleh Donald Knuth dan Vaughan Pratt pada tahun 1970, dengan penyesuaian oleh James H. Morris.Algoritma KMP merupakan metode algoritma pencarian string yang efisien, digunakan untuk menemukan kemunculan sebuah pola dalam teks. Algoritma ini mencari pola dalam text dari kiri

ke kanan, mirip dengan brute-force namun lebih pintar. Algoritma ini lebih 'pintar' daripada Brute Force karena algoritma ini menghindari pencocokan ulang karakter yang telah dibandingkan sehingga lebih cepat dibanding metode Brute Force.

KMP terdiri dari dua fase utama: pertama, membangun tabel "Border-Function" yang menyimpan informasi tentang panjang prefiks yang juga merupakan sufiks untuk setiap posisi dalam pola; kedua, menggunakan tabel fungsi pinggiran KMP.Berikut Cara kerja KMP dalam pencocokan string:

1. **Membangun tabel border function**

(k = j-1)

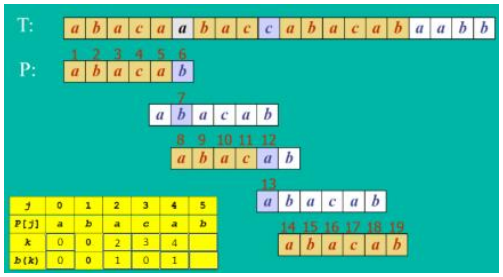
j	0	1	2	3	4	5
P[j]	a	b	a	a	b	a

k	0	1	2	3	4
b(k)	0	0	1	1	2

Gambar 2.2 Tabel border function
(Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf)

- tabel "Border-Function" menyimpan informasi tentang panjang prefiks yang juga merupakan sufiks untuk setiap posisi dalam pola.
- Dilakukan dengan mengiterasi melalui pola dan mengidentifikasi prefix yang juga merupakan suffix terpanjang pada setiap posisi dalam pola. Proses ini dilakukan menggunakan dua indeks, k (posisi sebelum *missmatch / j -1)* dan j (*posisi missmatch)*, yang bertanggung jawab untuk memeriksa kesamaan karakter di antara prefix dan suffix.

2. **Menggunakan tabel border function untuk pencocokan**
   - Dengan adanya tabel border function, algoritma ini menjadi lebih unggul dibanding algoritma brute force.
   - Iterasi P dari kiri ke kanan dengan bandingkan karakter pada P dengan karakter dengan urutan yang sesuai pada T
   - Jika terjadi *missmatch* di P, maka akan digunakan border function untuk menentukan berapa banyak pergeseran yang harus dilakukan.
   - Apabila ketika perbandingan semua karakter pada pattern cocok, maka pencarian berhasil



Gambar 2.3 Ilustrasi algoritma KMP
(Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Dengan cara ini, KMP mencapai kompleksitas waktu pencarian sebesar O(n + m), di mana n adalah panjang teks dan m adalah panjang pola. Adapaun *cons* dari algoritma ini, yaitu algoritma ini memerlukan kompleksitas ruang yang lebih besar dibanding brute force karena memerlukan pra-pemrosesan tabel border function.

D. Ultrasonik

Sensor ultrasonik memanfaatkan prinsip gelombang ultrasonik untuk mengukur jarak antara sensor dan objek di sekitarnya. Prinsip kerja sensor ini didasarkan pada waktu yang diperlukan gelombang ultrasonik untuk berpropagasi dari sensor ke objek dan kembali lagi. Dalam pengukuran jarak, kecepatan suara dalam medium yang diukur, biasanya udara, sangat penting. Rumus sederhana yang digunakan untuk menghitung jarak adalah

$$Jarak = \frac{Durasi\ x\ Kecepatan\ Suara}{2}$$

$$Kecepatan\ suara : \left(343 \frac{m}{s}\right)$$

$$Durasi : waktu\ pulang\ pergi\ antar\ sensor\ dengan\ objek$$

Dengan menggunakan rumus ini, sensor ultrasonik dapat menghasilkan pengukuran jarak dengan akurasi yang tinggi.

III. IMPLEMENTASI

Untuk Implementasi Algoritma String Matching dalam Sensor Ultrasonik, diperlukan beberapa tahap agar algoritma KMP maupun Brute-Force bisa digunakan. Pertama harus dikonversi terlebih dahulu data jarak menjadi sebuah karakter-karakter T. Setelah itu, pembuatan referensi P yang nantinya akan di cocokan dengan T sehingga didapatkan pola gerakan yang valid. Setelah itu baru implementasi algoritma KMP dan Brute-Force.

Untuk penerapan ini, saya menggunakan 2 algoritma untuk membandingkan juga bahwa KMP lebih unggul daripada Brute-Force, sehingga selain implementasi dilakukan juga optimasi agar deteksi pola gerakan menjadi lebih efisien.

A. Konversi data jarak

Untuk konversi, saya perlu membuat batasan batasan tertentu untuk menyesuaikan cara kerja sensor ultrasonik HC-SR04 dan algoritma string matching nanti. Pertama saya inisialisasi dulu berapa panjang T yaitu 10 dan interval pembacaan jarak. Saya tidak bisa membuat T yang terlalu panjang dikarenakan sumber daya yang terbatas dari Arduino Uno sendiri, lalu saya juga buat interval pengecekan tiap 1 detik sekali. Hasilnya adalah pembacaan 1 jarak tiap 0.1 detik, dihasil 10 data jarak yang dikonversi menjadi karakter untuk T.

```
void loop() {
    // Mengumpulkan data dari sensor ultrasonik
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    jarak = duration * 0.034 / 2;
    if (jarak > 50) {
        jarak = 250;
    }

    int currentCategory = jarak / categoryIncrement;
    category = 'a' + currentCategory;

    charArray[currentIndex] = category;
    currentIndex = (currentIndex + 1) % arraySize;

    if (millis() - lastClearTime >= clearInterval) {
        Cetak array gerakan
        Serial.print("Array Gerakan: ");
        for (int i = 0; i < arraySize; i++) {
            Serial.print(charArray[i]);
        }
        Serial.println();

        // Lakukan pencarian pola menggunakan algoritma KMP
        for (const auto& pattern : patternMovements) {
            const string& pat = pattern.pattern;
            const string& movementName = pattern.movement;
            KMPSearch(pat.c_str(), charArray, movementName.c_str());
            bruteForceSearch(pat.c_str(), charArray, movementName.c_str());
        }

        clearArray();

        lastClearTime = millis();
    }

    delay(100);
}
```

Gambar 3.1 Potongan kode segmen Loop()
(Sumber : dokumentasi pribadi)

Selanjutnya saya mengubah data jarak (saat ini masih berbentuk data mentah berupa angka, misal : 20 cm ,25 cm dst..) menjadi karakter. Untuk konversi data ini saya sudah melakukan percobaan berkali-kali dan hasil yang paling optimal adalah : “Tiap bagi 10 cm, maka dihasilkan kategori baru dan karakter baru”. Misal begini, saya dapat jarak 15 cm maka masuk kategori 1 (karakter ‘a’) karena 15 hanya bisa dibagi 10 1 kali. Selanjutnya 30 cm, maka masuk kategori 3 (karakter ‘c’) karena bisa dibagi 3 kali.

Tak hanya sampai disini, saya juga membatasi jarak maksimal karena kalau terlalu besar, ditakutkan sumber daya arduino uno tidak kuat untuk melakukan algoritma String matching sehingga hasil menjadi tidak efisien. Disini saya buat apabila terdeteksi objek diatas 50 cm, maka akan dibuat jarak menjadi 250 cm. Jadi kesimpulannya di T dan P hanya akan ada 6 karakter yaitu ‘a’,’b’,’c’,’d’,’e’,’z’ dengan z sebagai ‘basis / dasar’ dari T. Dengan pendekatan ini, dihasilkan sebuah T maupun P yang cukup optimal untuk diimplementasikan dengan algoritma KMP maupun Brute-Force, dengan batasan jarak maksimal objek yang terbaca oleh sensor ultrasonik adalah 50 cm, Berikut output nya:

```
Array Gerakan: aazbzbbbbz
Array Gerakan: zzzzzzzzzz
Array Gerakan: czcbccccbb
Array Gerakan: cczzbaabdd
Array Gerakan: ddzzbbbbcz
Array Gerakan: zzdzzzzzbz
Array Gerakan: zbzzzzazzz
Array Gerakan: bzzzzbcbzz
Array Gerakan: zzzbzzzzzb
Array Gerakan: bbcdzdczzc
Array Gerakan: cddzczbzzz
Array Gerakan: zbzbbzzzzc
```

Terlihat bahwa banyak pola, dan ada pola yaitu ‘zzzzzzzzzz’. Pola ini menunjukkan bahwa dalam kronologi pengecekan ultrasonik saat itu, tidak ada objek yang ditangkap oleh sensor ultrasonik.

B. Pembuatan Pola P

```
struct PatternMovement {
    int id;
    string pattern;
    string movement;

};
PatternMovement patternMovements[] = {
    {1,"zaz", "Anda sedang waving"},
    {1,"zbz", "Anda sedang waving"},
    {1,"zcz", "Anda sedang waving"},
    {1,"zdz", "Anda sedang waving"},
    {2,"cccc", "Anda sedang menghalangi"},
    {2,"dddd", "Anda sedang menghalangi"},
    {2,"bbbb", "Anda sedang menghalangi"},
    {3,"bc", "Anda menjauhi sensor"},
    {3,"cd", "Anda menjauhi sensor"},
    {3,"de", "Anda menjauhi sensor"},
    {4,"dc", "Anda mendekati sensor"},
    {4,"ba", "Anda mendekati sensor"},
    {4,"cb", "Anda mendekati sensor"},
    // bisa ditambah lagi, tapi pikirkan
    // juga resource arduino, maksimal 2048 KB

};
```

Gambar 3.2 Potongan kode PatternMovement
(Sumber : dokumentasi pribadi)

Untuk pembuatan pola P. Saya sudah membuat beberapa referensi pola yang cukup bagus untuk dijadikan pola gerakan. Untuk saat ini, saya menemukan 4 pola yang sudah cukup optimal untuk dibaca oleh Sensor ultrasonik ini. Saya juga membuat pattern yang beragam untuk gerakan yang sama, misal untuk ‘waving’ (sedang halo halo). Misalnya, ada 3 P yang berbeda yaitu ‘zaz’ , ‘zbz’, dan ‘zcz’. Karakter yang berbeda hanya di tengah P itu. Pendekatan ini saya lakukan agar pendeteksi gerakan lebih fleksibel untuk jarak yang berbeda-beda(saat ini ketika a b c atau d, yaitu 10 ,20,30 atau 40 cm dari sensor ultrasonik).

C. Implementasi Algoritma KMP

Untuk algoritma KMP sendiri, saya sudah membuat 2 fungsi yang nantinya akan di pakai di segmen Loop() yaitu KMPSearch() dan borderFunc(). Sebelum dicari dengan KMP, sesuai dengan teori KMP bahwa diperlukan dulu array border function yang dibuat di dalam fungsi borderFunc().

```
void borderFunc(const char* pat, int M, int* lps) {
    int len = 0;  // Panjang awal dari lps
    int i = 1;

    lps[0] = 0;

    while (i < M) {
        if (pat[i] == pat[len]) {
            len++;
            lps[i] = len;
            i++;
        }
        else {  // (pat[i] != pat[len])
            if (len != 0) {
                len = lps[len - 1];
            }
            else {  // if (len == 0)
                lps[i] = 0;
                i++;
            }
        }
    }
}
```

Gambar 3.3 Potongan kode borderFunc()
(Sumber : dokumentasi pribadi)

Untuk pembuatan border function sendiri, pertama inisialisasi terlebih dahulu isi array lps elemen pertama dibuat 0, lalu selama i kurang dari m(panjang P). maka dicek apabila P di indeks len(awal dari lps) sama dengan P di i, maka elemen lps di indeks i = len + 1, apabila tidak, maka dicek lagi apakah len sama dengan 0, apabila iya maka lps di indeks I dibuat 0 dan i di-*increment*, apabila tidak sama dengan 0 maka nilai len = lps[len – 1].

```
void KMPSearch(const char* pat, char* txt, const char* movementName) {
    unsigned long startTime = millis();

    int M = strlen(pat);
    int N = strlen(txt);

    int lps[M];

    borderFunc(pat, M, lps);

    int i = 0;  // Indeks untuk teks
    int j = 0;  // Indeks untuk pola
    while (i < N) {
        if (pat[j] == txt[i]) {
            j++;
            i++;
        }

        if (j == M) {
            //ditemukan..
            Serial.print("Pola ");
            Serial.print(pat);
            Serial.print(" ditemukan, gerakan: ");
            Serial.println(movementName);
            j = lps[j - 1];
            unsigned long executionTime = millis() - startTime;
            Serial.print("Waktu Eksekusi KMP: ");
            Serial.print(executionTime);
            Serial.println(" ms");
            break;
        }

        else if (i < N && pat[j] != txt[i]) {
            if (j != 0) {
                j = lps[j - 1];
            }
            else {
                i = i + 1;
            }
        }
    }
}
```

Gambar 3.4 Potongan kode KMPSearch()
(Sumber : dokumentasi pribadi)

Pembuatan kode diatas menyesuaikan algoritma KMP, Misalnya i (i = 0)adalah index dari T, j(j= 0) adalah indeks dari P m adalah panjang dari P dan n adalah panjang dari T dan lps[m] adalah array border function yang memiliki panjang m. Maka iterasi terus (while) selama i kurang dari N maka:

1. Apabila karakter di P di indeks J sama dengan karakter T di indeks i, maka j dan i di-*increment*
2. Jika j sudah sama dengan M, maka P ditemukan di T.
3. Jika i kurang dari N dan P di indeks J tidak sama dengan T di indeks i, disini dibuat nilai j = lps[k / j-1] apabila j tidak sama 0, apabila j = 0,maka i ditambah 1 atau next ke karakter T selanjutnya.

D. Implementasi Algoritma Brute Force

Untuk algoritma Brute-Force, implementasinya cukup *straight-forward*, yaitu geser 1 karakter apabila ditemukan karakter yang tidak sama di indeks P dan indeks T, apabila j sama dengan M maka pola ditemukan.

```
void bruteForceSearch(const char* pat, char* txt, const char* movementName) {
    unsigned long startTime = millis();
    int M = strlen(pat);
    int N = strlen(txt);

    for (int i = 0; i <= N - M; i++) {
        int j;

        for (j = 0; j < M; j++) {
            if (txt[i + j] != pat[j]) {
                break;
            }
        }

        if (j == M) {
            Serial.print("Pola ");
            Serial.print(pat);
            Serial.print(" ditemukan (Brute-force), gerakan: ");
            Serial.println(movementName);
            unsigned long executionTime = millis() - startTime;
            Serial.print("Waktu Eksekusi Brute-force: ");
            Serial.print(executionTime);
            Serial.println(" ms");
            break;
        }
    }

}
```

Gambar 3.5 Potongan kode bruteForceSearch()
(Sumber : dokumentasi pribadi)

E. Percobaan

Untuk logika dari implementasi ini sudah siap, sekarang saatnya untuk melakukan percobaan pada mikrokontroller Arduino Uno yang sudah di *setup* dengan sensor ultrasonik. Untuk percobaan pertama, saya akan melakukan gerakan waving dan gerakan mendekat atau menjauh untuk melihat perbedaan kemampuan kedua algoritma tersebut. Berikut contoh outputnya dan data yang saya kumpulkan dari berbagai macam gerakan

```
Array Gerakan: zzbabczzab
Pola bc ditemukan, gerakan: Anda
menjauhi sensor
Waktu Eksekusi KMP: 28 us
Waktu Eksekusi Brute-force: 20 us
Pola ba ditemukan, gerakan: Anda
mendekati sensor
Waktu Eksekusi KMP: 24 us
Waktu Eksekusi Brute-force: 16 us
Array Gerakan: babzczabzc
Pola zcz ditemukan, gerakan: Anda
sedang waving
Waktu Eksekusi KMP: 28 us
Waktu Eksekusi Brute-force: 16 us
```

No	Waving (ms)		Menjauhi (ms)		Mendekati (ms)		Menghalangi (ms)	
	KMP	BF	KMP	BF	KMP	BF	KMP	BF
1	28	16	28	20	16	12	24	12
2	32	20	32	20	28	20	24	12
3	40	28	16	12	28	12	48	24
4	40	28	28	12	20	12	32	24
5	44	28	28	29	24	16	24	12

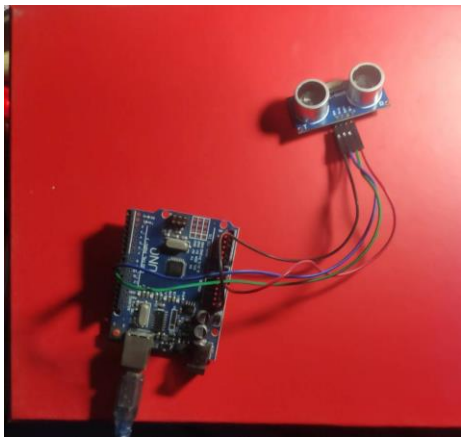
Diatas sudah disajikan data dan contoh outpunya. Terlihat jelas bahwa Brute Force memiliki waktu yang lebih cepat dibandingkan KMP. Algoritma brute-force lebih cepat dalam percobaan yang dilakukan, terutama karena overhead pre-pemrosesan KMP tidak sebanding dengan keuntungan dalam pola pendek dan teks yang pendek.

Mengapa KMP bisa lebih lambat: KMP menghabiskan waktu untuk membangun fungsi border, yang mungkin lebih lama dari pencarian pola langsung jika pola dan teks cukup pendek.Nah, Kapan KMP lebih cepat: Dalam teks

panjang dengan pola kompleks atau pola yang memiliki banyak kesamaan karakter dengan teks tetapi tidak cocok, KMP akan lebih efisien karena mengurangi perbandingan berulang.

F. Rangkaian Arduino

Untuk Rangkaian arduino sendiri, saya hanya menggunakan 2 buah komponen yaitu Arduino UNO dan Sensor Ultrasonic HC-SR04. Untuk rangkaian bisa ditingkatkan dengan penambahan lampu LED yang berbeda warna untuk tiap metode gerakan yang berbeda agar representasi visualnya lebih jelas, namun hal ini tidak sempat saya lakukan karena keterbatasan komponen yang saya punyai sekarang. Berikut untuk foto rangkaian Arduino nya.



Gambar 3.6 Rangkaian Arduino
(Sumber : dokumentasi pribadi)

Untuk detil rangkaiannya sendiri cukup mudah, pertama PIN VCC dari sensor ke pin 5V di arduino,GND to GND,Trig ke pin 9 dan Echo ke pin 10.

IV. KESIMPULAN

Kesimpulan dari penerapan algoritma string matching dalam sensor ultrasonik menggunakan Arduino Uno untuk deteksi pola gerakan menunjukkan bahwa algoritma Brute Force lebih cepat dibandingkan dengan Knuth-Morris-Pratt (KMP) dalam konteks teks dan pola yang pendek, karena Brute Force melakukan pencarian pola langsung tanpa memerlukan pre-pemrosesan tambahan seperti KMP. Namun, KMP akan lebih efisien dalam teks panjang dengan pola kompleks atau pola yang memiliki banyak kesamaan karakter dengan teks tetapi tidak cocok, karena KMP mengurangi perbandingan berulang. Dengan demikian, pemilihan algoritma yang tepat sangat tergantung pada konteks penggunaan, di mana Brute Force lebih unggul untuk teks pendek dan sederhana, sedangkan KMP lebih efektif untuk teks panjang dan kompleks. Penerapan ini memungkinkan kita mendeteksi pola gerakan secara efisien, membuka peluang untuk pengembangan mengenai pengawasan berbasis sensor.

V. UCAPAN TERIMA KASIH

Saya ingin menyampaikan rasa terima kasih yang tulus kepada Tuhan Yang Maha Esa atas bimbingan-Nya selama penulisan karya ilmiah ini. Juga, terima kasih yang sebesarbesarnya untuk Pak Rinaldi Munir ,Pak Monterico Adrian dan Pak Rila Mandala, Dosen Strategi Algoritma, yang telah memberikan bimbingan dan ilmu pengetahuan yang sangat berarti. Keberhasilan penyelesaian karya ilmiah ini tidak terlepas dari kontribusi mereka. Terima kasih atas dedikasi dan bantuan yang diberikan. Semoga Tuhan selalu memberkati langkah-langkah kita semua

LAMPIRAN

Berikut pranala video youtube :
https://www.youtube.com/watch?v=7b6_lEL8X1A

REFERENCES

[1] Arduino.Arduino Uno Rev3 Datasheet. https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf (Diakses 11 Juni 2024 pukul 13.20)

[2] Munir, Rinaldi. Pencocokan String. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf , (Diakses 11 Juni 2024 pukul 13.00)

[3] Getting Started with the HC-SR04 Ultrasonic, https://projecthub.arduino.cc/Isaac100/getting-started-with-the-hc-sr04-ultrasonic-sensor-7cabe1 (Diakses 11 Juni 2024 pukul 16:50)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

Hugo Sabam Augusto
13522129