BAB 2
TINJAUAN PUSTAKA

2.1. Kajian Pustaka

Pada subbab ini membahas penelitian - penelitian terdahulu yang berhubungan dengan penelitian yang akan dilakukan. Bahan kajian utama yang dibahas adalah penetapan derajat putaran motor untuk per klik pada aplikasi android.

2.2. Internet Of Things (IOT)

Menurut Wikipedia, Internet of Things atau dikenal juga dengan singkatan IoT, merupakan sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas internet yang tersambung secara terus-menerus. Adapun kemampuan seperti berbagi data, remote control, dan sebagainya, termasuk juga pada benda di dunia nyata. Contohnya bahan pangan, elektronik, koleksi, peralatan apa saja, termasuk benda hidup yang semuanya tersambung ke jaringan lokal dan global melalui sensor yang tertanam dan selalu aktif.

Makna serupa yang lain, Internet of Things (IoT) adalah sebuah konsep/skenario dimana suatu objek yang memiliki kemampuan untuk mentransfer data melalui jaringan tanpa memerlukan interaksi manusia ke manusia atau manusia ke komputer.

"A Things" pada Internet of Things dapat didefinisikan sebagai subjek misalkan orang dengan monitor implant jantung, hewan peternakan dengan transponder biochip, sebuah mobil yang telah dilengkapi built-in sensor untuk memperingatkan pengemudi ketika tekanan ban rendah. Sejauh ini, IoT paling erat hubungannya dengan komunikasi machine-to-machine (M2M) di bidang manufaktur dan listrik, perminyakan, dan gas. Produk dibangun dengan kemampuan komunikasi M2M yang sering disebut dengan sistem cerdas atau "smart". (contoh: smart label, smart meter, smart grid sensor).

Meskipun konsep ini kurang populer hingga tahun 1999, namun IoT telah dikembangkan selama beberapa dekade. Alat Internet pertama, misalnya, adalah mesin Coke di Carnegie Melon University di awal 1980-an. Para programer dapat terhubung ke mesin melalui Internet, memeriksa status mesin dan menentukan apakah ada atau tidak minuman dingin yang menunggu mereka, tanpa harus pergi ke mesin tersebut.

Istilah IoT (Internet of Things) mulai dikenal tahun 1999 yang saat itu disebutkan pertama kalinya dalam sebuah presentasi oleh Kevin Ashton, cofounder and executive director of the Auto-ID Center di MIT.

Dengan semakin berkembangnya infrastruktur internet, maka akan menuju babak berikutnya, di mana bukan hanya smartphone atau komputer saja yang dapat terkoneksi dengan internet. Namun berbagai macam benda nyata akan terkoneksi dengan internet. Sebagai contohnya dapat berupa : mesin produksi, mobil, peralatan elektronik, peralatan yang dapat dikenakan manusia (wearables), dan termasuk benda nyata apa saja yang semuanya tersambung ke jaringan lokal dan global menggunakan sensor dan atau aktuator yang tertanam.

Beberapa contoh konkrit dari "wearable" yang mulai dipasarkan di dunia adalah : Google Glass, Google Nest, Nike Fit, dan Samsung Smart Watch. Tidak hanya wearables, Samsung juga mulai merambah dan mengembangkan teknologi IOT di bidang consumer appliances seperti : Smart Air Conditioner, Smart TV, Smart Refrigerator. Pada tahun 2017, menurut CEO Samsung, 90% dari semua produk Samsung akan berupa perangkat IOT, termasuk semua televisi dan perangkat mobile. Dua tahun berikutnya, semua produk Samsung akan siap dengan koneksi IOT. Kompetitor terdekat Samsung, yakni Apple pun memiliki upaya di bidang IOT dengan proyek Homekit, yang merupakan protokol pengontrol rumah pintar melalui sistem operasi iOS. Beberapa produk Apple tersebut antara lain iHome, Incipio, GridConnect, dan iDevices. Semua perangkat Apple Homekit tersebut akan dipasarkan dalam waktu dekat ini.

2.2.1. Manfaat IoT

Berbagai macam implementasi IoT adalah dalam kehidupan sehari-hari kita. Bahkan beberapa mungkin telah kita lakukan, hanya saja tidak terpikir bahwa itu adalah bagian dari IoT. Berikut ini adalah beberapa manfaat dalam beberapa bidang, yakni :

- Sektor Pembangunan
- Sektor Energi
- Sektor Rumah Tangga
- Sektor Kesehatan
- Sektor Industri
- Transportasi
- Perdagangan
- Keamanan
- Teknologi dan Jaringan

2.3. Mikrokontroler

Mikrokontroler adalah sirkuit terintegrasi yang mampu menjalankan program. Ada banyak contoh yang ada di pasaran saat ini dari berbagai produsen. Di pasar hobi, arsitektur open source yang disebut "Arduino" yang menggunakan berbagai prosesor Atmel telah menarik imajinasi banyak orang. Papan yang berisi chip Atmel ini dikombinasikan dengan konvensi untuk koneksi dan juga seperangkat alat pengembangan gratis telah menurunkan titik masuk untuk bermain dengan elektronik hingga hampir nol. Tidak seperti PC, prosesor ini sangat rendah dengan ram dan kemampuan penyimpanan yang rendah. Mereka tidak akan mengganti desktop atau laptop dalam waktu dekat.

Perangkat ini memiliki serangkaian kemampuan luar biasa termasuk input dan output listrik langsung (GPIO) dan dukungan untuk berbagai protokol termasuk SPI, I2C, UART dan banyak lagi, namun, beberapa dari mereka sejauh ini datang dengan jaringan nirkabel yang disertakan.

Arduino didasarkan pada chip Atmel dan memiliki berbagai ukuran fisik kerasnya yang terbuka. Kontroler mikro utama yang digunakan adalah ATmega328.

2.3.1. Mikrokontroler ESP32

ESP32 adalah nama pengontrol mikro yang dirancang oleh Espressif Systems. Espressif adalah perusahaan Cina yang berbasis di Shanghai. ESP32 mengiklankan dirinya sebagai solusi jaringan WiFi mandiri yang menawarkan dirinya sebagai jembatan dari mikrokontroler yang ada ke WiFi, dan juga mampu menjalankan aplikasi mandiri.

Volume produksi ESP32 tidak dimulai sampai akhir 2016 yang berarti bahwa, dalam skema hal, ini adalah entri baru dalam jajaran prosesor. Dan di dunia teknologi, yang baru biasanya sama dengan yang menarik. Beberapa tahun setelah produksi IC, pihak ketiga OEM mengambil chip ini dan membangun "papan breakout" untuk mereka.

OEM membeli IC secara massal, mendesain sirkuit dasar, mendesain papan sirkuit cetak, dan membuat papan dengan IC yang sudah terpasang segera siap untuk digunakan.

2.3.2. Spesifikasi ESP32

Ini adalah serangkaian karakteristik seperti yang dijelaskan oleh pabrikan. Berikut adalah daftar ringkasan item ESP32 inti:

Tabel 2.1 Spesifikasi ESP32

Attribute	Details
Voltage	3.3V
Current consumption	Unknown
Flash memory attachable	Module based
Processor	Tensilica L108 32 bit
Processor speed	Dual 160MHz
RAM	520K
GPIOs	34
Analog to Digital	7
802.11 support	b/g/n/e/i
Bluetooth	BLE
Maximum concurrent TCP connections	16
SPI	3
I2S	2
I2C	2
UART	3

ESP32 adalah prosesor dual core yang menjalankan instruksi Xtensa LX6. Inti disebut "PRO_CPU" dan "APP_CPU".

ESP32 dirancang untuk digunakan dengan modul memori dan ini biasanya merupakan memori flash. Sebagian besar modul dilengkapi dengan beberapa flash yang terkait dengannya. Flash memiliki jumlah penghapusan yang terbatas per halaman sebelum sesuatu gagal. Mereka dinilai sekitar 10.000 penghapusan. Ini biasanya bukan masalah untuk perubahan konfigurasi penulisan atau penulisan log harian tetapi jika

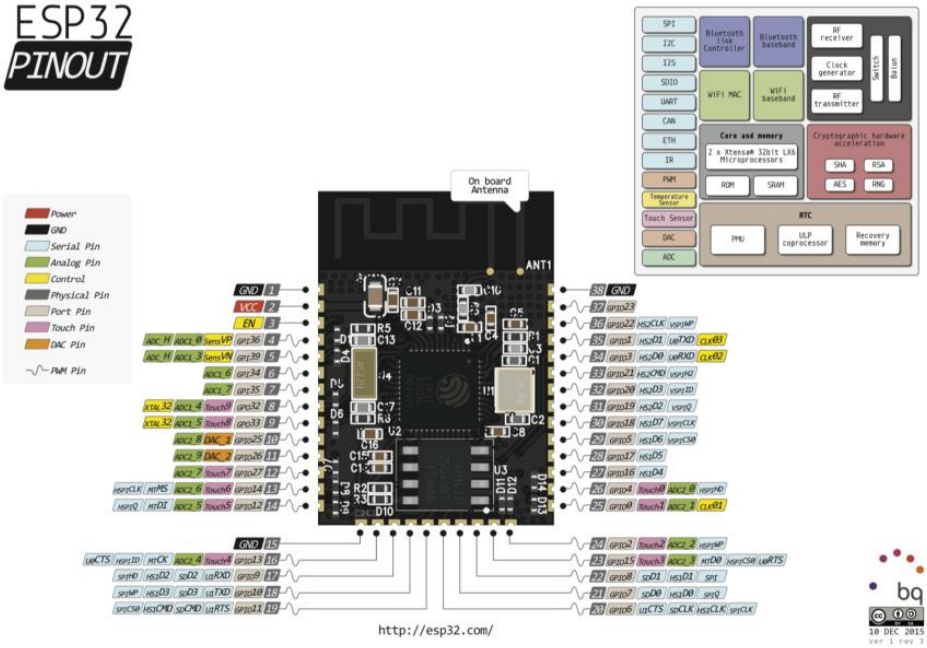
aplikasi terus menulis data baru dengan sangat cepat, maka ini mungkin menjadi masalah dan memori flash akan gagal.

2.3.3. Modul ESP-WROOM-32

Modul yang disebut ESP-WROOM-32 tersedia dari Espressif yang berisi ESP32 plus perangkat keras pendukung yang menyertainya seperti memori flash.

Gambar 2.1 ESP-WROOM-32

Ini jauh lebih berorientasi pada mereka yang memiliki keterampilan elektronik yang baik yang ingin menanamkan ESP32 dalam proyek tertentu. Modul ini hanya 18mm x 25.5mm dan memiliki pin spacings di pitch 1.27mm. Berikut adalah skema perangkat saat ini :



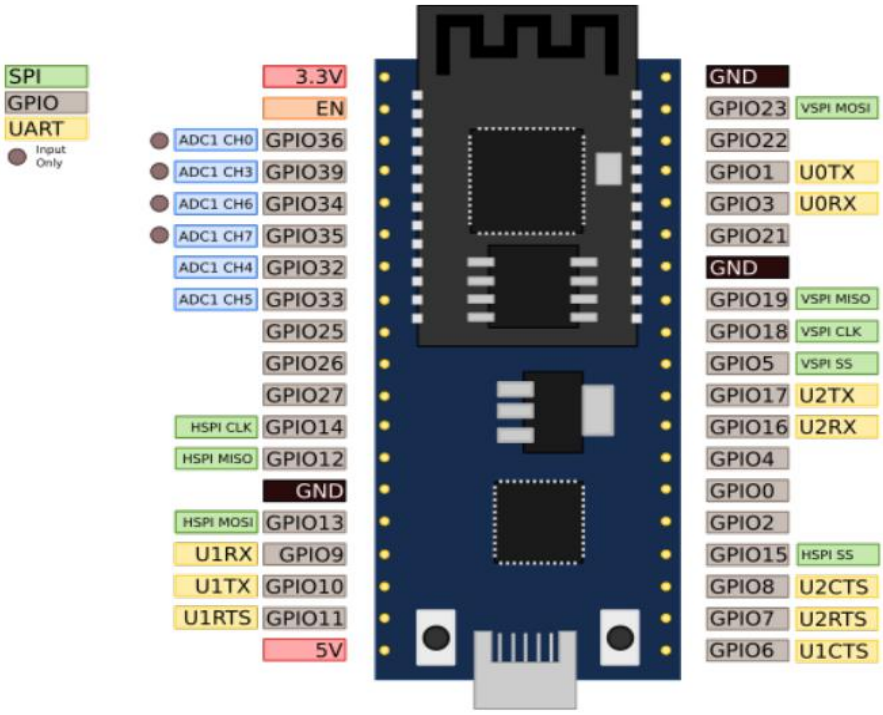
Gambar 2.2 Modul ESP32 Pinout

2.3.4. Modul ESP32-DevKit

Dengan dirilisnya ESP32, Espressif telah merilis modul mereka sendiri untuk mengekspos ESP32 kepada lebih banyak konsumen.

Gambar 2.3 ESP32-DevKit

Papan berisi header untuk ESP32 serta adaptor USB mikro dan dua tombol yang disebut mengaktifkan dan boot. Tombol-tombol ini dapat digunakan untuk "flash" atau "unduh" kode aplikasi baru ke dalam modul. Untuk melakukan tugas ini, tahan tombol "EN" sambil menekan dan melepaskan "Boot". Pin out dari modul ditunjukkan pada gambar berikut:



Gambar 2.4 Board pin ESP32

Untuk me-reboot perangkat, pulsa ID rendah. Jika GPIO0 tinggi, perangkat akan melakukan booting normal sedangkan jika GPIO0 rendah, ia akan mem-boot ke mode flash memungkinkan kita untuk mengunggah aplikasi baru ke dalam penyimpanan flash perangkat.

2.3.5. Menghubungkan ke ESP32

ESP32 adalah perangkat WiFi dan karenanya akhirnya akan terhubung dengannya menggunakan protokol WiFi tetapi beberapa bootstrap diperlukan terlebih dahulu. Perangkat tidak tahu jaringan apa yang harus dihubungkan, kata sandi mana yang digunakan dan parameter lain yang diperlukan. Ini tentu saja mengasumsikan bahwa kita terhubung sebagai stasiun, jika kita ingin perangkat menjadi titik akses atau ingin memuat aplikasi sendiri ke dalamnya, ceritanya semakin dalam. Ini menyiratkan bahwa ada beberapa cara untuk berinteraksi dengan perangkat selain WiFi dan ada jawabannya adalah UART (Serial). ESP32 memiliki antarmuka UART khusus dengan pin berlabel TX dan RX. Pin TX adalah transmisi ESP32 (keluar dari ESP32) dan pin RX digunakan untuk menerima data (masuk ke ESP32). Pin ini dapat dihubungkan ke mitra UART. Sejauh ini cara termudah dan paling nyaman bagi kami adalah konverter USB → UART. Melalui UART, kita dapat melampirkan emulator terminal untuk mengirim penekanan tombol dan data yang diterima dari ESP32 ditampilkan sebagai karakter di layar. Tujuan kedua UART adalah untuk menerima data biner yang digunakan untuk "mem-flash" memori flash perangkat untuk merekam aplikasi baru untuk dieksekusi. Ada berbagai alat teknis yang kami miliki untuk mencapai tugas itu.

Ketika menggunakan UART, kita perlu mempertimbangkan konsep baud rate. Ini adalah kecepatan komunikasi data antara ESP32 dan mitranya. Selama boot, ESP32 mencoba untuk secara otomatis menentukan baud rate pasangan dan mencocokkannya. Ini mengasumsikan default 115200 dan jika memiliki terminal serial terpasang, maka akan terlihat pesan seperti ini:

```
???
```

jika dikonfigurasi untuk menerima pada 115200.

Ketika terhubung ke mesin Windows melalui micro USB, itu muncul sebagai perangkat serial:

Baud rate serial default adalah 115200.

Setelah ESP32 terhubung secara seri, kami biasanya ingin memasang monitor atau terminal emulator terhadap perangkat. Kerangka ESP-IDF menyediakan teknik yang bagus untuk mencapainya. Kita bisa menjalankan perintah "make monitor". Ini meluncurkan alat monitor yang membuntuti keluaran serial ESP32. Alat ini didasarkan pada "miniterm" yang merupakan bagian dari paket "PySerial". Kami dapat mengakhiri miniterm dengan `CTRL +]`.

Dari lingkungan Linux, kita melihat port serial (biasanya) sebagai `/dev/ttyUSB0`. Jika memasang aplikasi "`screen`", kemudian dapat menghubungkan terminal menggunakan:

```
$ screen /dev/ttyUSB0 115200
```

Untuk keluar dari layar, masukkan "`CTRL+A`" diikuti oleh "`: quit`".
Sebagai alternatif untuk layar, kami juga memiliki "cu":

```
$ cu -l /dev/ttyUSB0 -s 115200
```

Untuk menghentikan program, masukkan "~."
Program lain adalah "minicom":

```
$ minicom --baudrate 115200 - device /dev/ttyUSB0
```

Dan tentu saja, karena port serial hanyalah aliran karakter, tidak ada yang mencegah dari hanya menjalankan "`cat`" untuk menampikannya:

```
$ cat /dev/ttyUSB0
```

Jika kami tidak yakin tentang pengaturan port serial, kami dapat menjalankan:

```
$ stty -F /dev/ttyUSB0 -a
```

yang akan mengembalikan pengaturan saat ini. Mengikuti flash, mungkin perlu mengubah beberapa pengaturan untuk melakukan kucing sederhana:

```
$ stty -F /dev/ttyUSB0 ispeed 115200 ospeed 115200
min 100 ixon
```

2.4. Motor Stepper

Motor Stepper adalah sebuah motor langkah adalah motor listrik DC tanpa sikat yang bergerak dalam sudut yang tepat, yang disebut langkah-langkah, dengan mengubah serangkaian pulsa listrik menjadi gerakan rotasi. Mereka tidak akan menghasilkan gerakan kontinu dari tegangan input kontinu, dan itu akan tetap pada posisi tertentu selama daya menyala. Motor langkah dikendalikan dengan menggunakan sinyal pulsa listrik diskrit. Setiap pulsa akan memutar poros motor langkah dengan sudut tetap yang disebut "langkah".

Jika pulsa dilakukan dalam urutan tertentu, motor akan berputar terus menerus; kecepatan dapat dikontrol oleh laju pengiriman pulsa. Sudut langkah alami ini memungkinkan motor langkah diposisikan secara akurat tanpa akumulasi kesalahan. Motor langkah menghasilkan torsi keluaran dari interaksi antara medan magnet di rotor dan di stator. Kekuatan medan magnet sebanding dengan jumlah arus yang diterapkan pada belitan serta jumlah lilitan pada lilitan.

Motor langkah digunakan dalam banyak aplikasi berbeda yang membutuhkan penentuan posisi dan kontrol kecepatan yang akurat dan dapat diulang. Mereka digunakan dalam industri seperti: Dirgantara & Pertahanan, Otomasi & Pengemasan, Medis, Percetakan & Ukiran, Sistem Keamanan & Pengawasan, Semikonduktor, dan Teknologi Surya & Hijau.

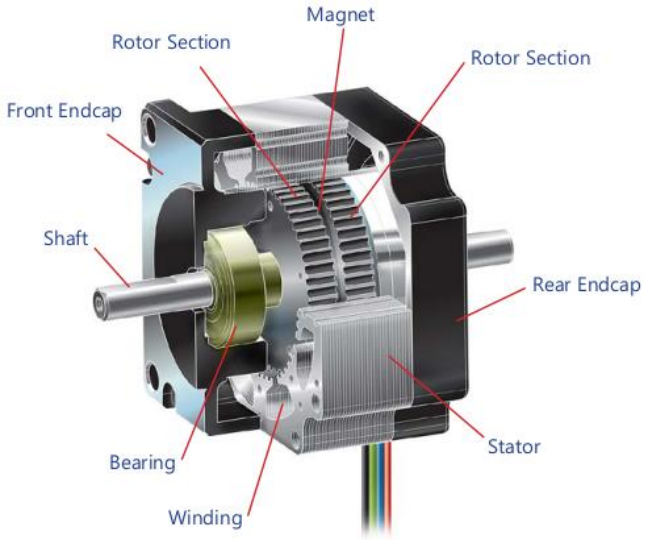
2.4.1. Keunggulan Motor Stepper

- Keandalan tinggi dengan sedikit perawatan.
- Operasi loop terbuka - Tidak diperlukan umpan balik untuk kontrol posisi atau kecepatan.
- Kesalahan posisi non-kumulatif.
- Secara inheren lebih aman daripada motor yang dikendalikan servo.
- Sudut rotasi motor proporsional dengan pulsa masukan sehingga lebih mudah diatur.
- Motor dapat langsung memberikan torsi penuh pada saat mulai bergerak.
- Posisi dan pergerakan repetisinya dapat ditentukan secara presisi.
- Memiliki respon yang sangat baik terhadap mulai, stop dan berbalik (perputaran).
- Sangat reliabel karena tidak adanya sikat yang bersentuhan dengan rotor seperti pada motor DC.
- Dapat menghasilkan perputaran yang lambat sehingga beban dapat dikopel langsung ke porosnya.
- Frekuensi perputaran dapat ditentukan secara bebas dan mudah pada range yang luas.

2.4.2. Fitur dan Konstruksi Dasar

Motor stepper terdiri dari magnet permanen yang terjepit di antara dua rotor (menyebabkan polaritas aksial), yang membentuk bagian pemintalan motor, ditempatkan ke dalam rumah stator di mana gulungan kawat stator membentuk fase motor yang berbeda. Fase dimagnetisasi di

mana fase A dan A- (atau B dan B-) dimagnetisasi pada waktu yang sama, sehingga kedua fase A termagnetisasi sebagai satu kutub, dan kedua fase A termagnetisasi sebagai kutub yang berlawanan karena belitan arah fase A berlawanan dengan arah putaran fase A-.



Gambar 2.5 motor stepper

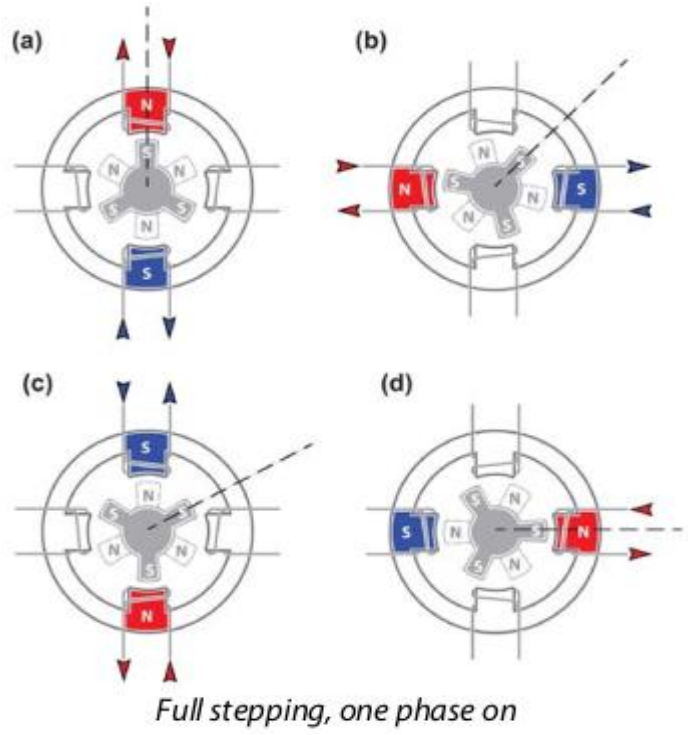
Rotor terhubung ke poros motor, yang memberikan rotasi dan torsi output motor ketika tegangan dan pulsa arus diterapkan ke gulungan motor. Bantalan di kedua sisi rotor memungkinkan rotasi halus dengan sedikit gesekan dan keausan. Bantalan ditempatkan ke ruang yang ditentukan di ujung depan dan belakang, yang memungkinkan konsentrisitas rotor di dalam stator. Pensejajaran yang sempurna antara rotor dan stator sangat penting karena celah udara di antara mereka, di mana torsi motor dihasilkan, harus sama di semua sisi dan hanya beberapa nanometer lebarnya lebih tipis dari sehelai rambut.

2.4.3. Cara Kerja Motor Stepper

Ketika daya terhubung ke kabel, pulsa arus diterapkan ke gulungan motor dan menyalakan "on" fase, yang menyebabkan rotor berputar sampai keengganan magnetik diminimalkan dan rotor mengendap pada posisi magnetik yang stabil. Untuk motor bipolar, pulsa arus mengisi kumparan fase pertama dengan muatan dan menciptakan medan magnet untuk masing-masing kumparan fase pertama, yang memagnetisasi gigi stator di depan kumparan bermuatan.

Keempat kumparan menarik gigi rotor yang bertolak belakang sementara mengusir gigi rotor yang bermuatan sama pada ujung rotor yang berlawanan. Gigi rotor di ujung utara dan selatan rotor diimbangi sedikit dari satu sama lain, sehingga satu gigi stator yang terisi dapat menarik gigi rotor yang bermuatan berlawanan di salah satu ujung rotor sambil memukul mundur dua gigi rotor yang bermuatan serupa di ujung lainnya. rotor hingga motor mengendap pada posisi detent. Ini menciptakan efek "mendorong dan menarik" yang meningkatkan torsi motor.

Torsi yang diperlukan untuk menggerakkan rotor dari posisi stabil ini disebut torsi holding. Turing satu fase aktif akan menahan rotor dalam satu posisi, yang disebut posisi detent. Muatan kemudian dimatikan pada set gulungan pertama, dan dinyalakan pada set gulungan berikutnya, menarik set gigi stator berikutnya. Rotor kemudian berputar sampai gigi rotor berikutnya sejajar dengan salah satu gigi stator yang termagnetisasi. Rotor sekarang dipindahkan satu sudut langkah ke posisi detent berikutnya. Aliran arus "hidup dan mati" ini menyebabkan rotor memutar satu sudut langkah yang tepat, dan gerakan ini diulangi dengan setiap pulsa input.



Gambar 2.6 Putaran Penuh, satu fase aktif

Setengah langkah terjadi ketika pulsa pertama memagnetisasi fase pertama, kemudian pulsa berikutnya memagnetisasi fase pertama dan fase kedua, kemudian pulsa berikutnya memagnetisasi hanya fase kedua, dll. Ini memungkinkan rotor untuk bergerak dalam peningkatan yang lebih kecil pada suatu waktu, memungkinkan untuk gerakan yang lebih halus dan terus menerus dengan resonansi lebih sedikit dan hanya sedikit torsi yang lebih sedikit.

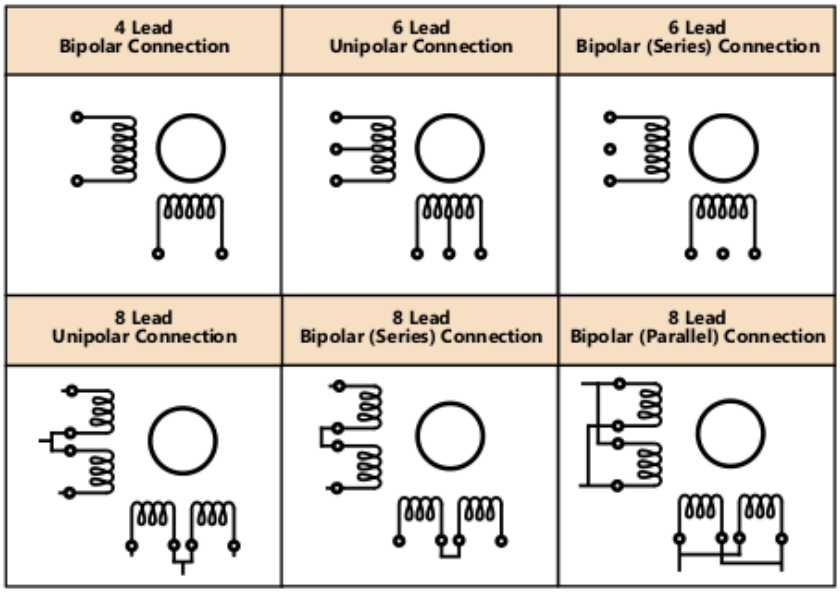
2.4.4. Jenis - Jenis Motor Stepper

Unipolar

Gulungan unipolar memiliki dua lilitan per-fase, dan dapat membalikkan magnetisasi kumparan stator tanpa membalik arah arus. Ini memiliki tap pusat untuk setiap fase, yang dapat digunakan untuk mengubah kabel untuk memasok arus ke bagian yang berlawanan dari masing - masing kumparan.

Diagram Koneksi Kabel

Pada gambar diagaram dibawah ini menunjukkan perbedaan lilitan kabel arus listrik.



Gambar 2.7 Diagram kabel

Bipolar Half Coil:

Lilitan bipolar memiliki belitan tunggal untuk setiap fase, sehingga arus perlu dibalik untuk membalikkan kutub magnet koil. Karena itu, ia menggunakan 100% belitan. Ini menghasilkan lebih banyak

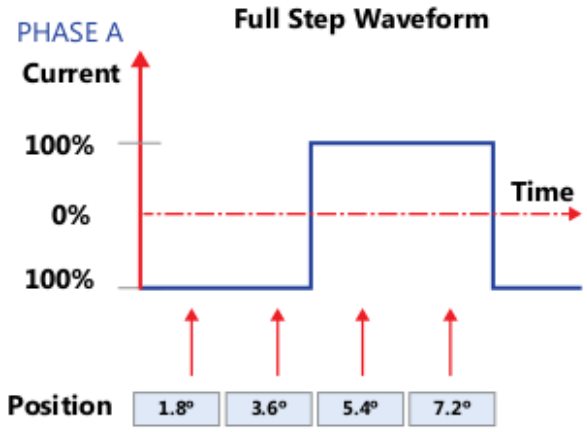
torsi dan memiliki sistem penggerak yang lebih kompleks / mahal. Gulungan bipolar juga dapat ditransfer secara paralel atau seri, tergantung pada torsi yang dibutuhkan dan kecepatan motor.

Bipolar Parallel:

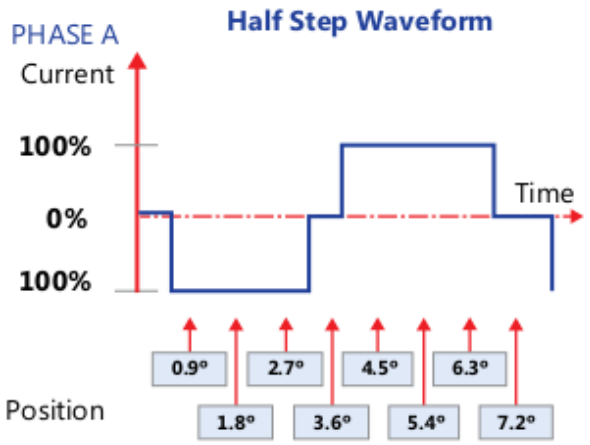
Lilitan paralel bipolar memiliki dua kumparan per-fase yang disatukan secara paralel. Gulungan paralel memungkinkan output torsi yang lebih tinggi pada kecepatan yang lebih tinggi, memiliki induktansi yang sama dengan gulungan setengah kumparan tunggal, memiliki setengah dari resistansi gulungan kumparan setengah, dan membutuhkan dua kali arus untuk mencocokkan putaran ampere seri. luka motor bipolar. Gulungan ini sangat ideal untuk aplikasi kecepatan tinggi yang membutuhkan banyak torsi.

2.4.5. Bentuk Gelombang Listrik Motor

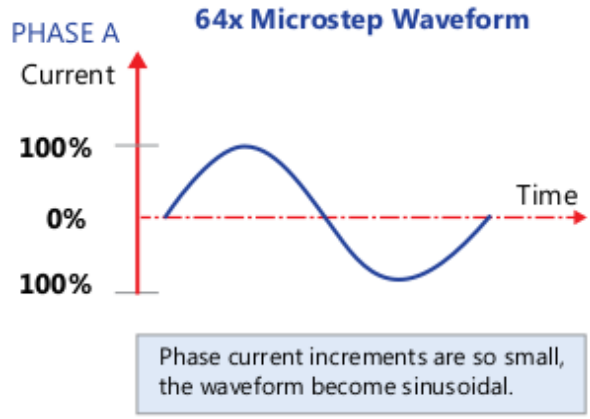
Sebagian besar langkah motor langkah 1,8 ° per langkah penuh. Jika motor digerakkan dalam mode langkah penuh (menggunakan 1,8 ° sebagai contoh), bentuk gelombang saat ini dari salah satu fase akan terlihat seperti berikut:



Gambar 2.8 Full Step Waveform



Gambar 2.9 Half Step Waveform



Gambar 2.10 Microstep Waveform

Arus 100% mengacu pada arus fasa motor atau arus terukur (Amp / Fase). (Microstepping adalah fitur driver, membagi arus menjadi peningkatan yang lebih baik).

2.4.6. 8 POLES vs 12 POLES

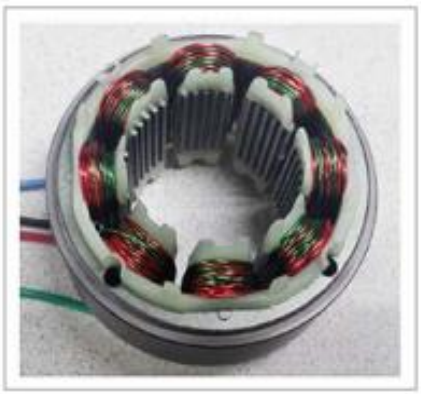
Motor Stepper 2-Fase Hibrida dapat dibangun dengan berbagai cara secara internal. Meskipun ada sejumlah komponen yang berbeda dalam motor stepper yang memiliki dampak besar pada kinerja, dapat dikatakan bahwa komponen yang paling penting adalah stator.

Stator dapat dirancang sedemikian rupa di mana ada 8 kutub, 12 kutub atau bahkan 16 kutub. Kutub adalah tempat kabel dililitkan, pada dasarnya menghasilkan gulungan motor. Perbedaan mekanis antara ketiga jenis desain stator adalah celah udara relatif antara rotor dan stator. Celah

udara antara rotor dan stator selalu non-concentric. Tidak ada langkah produsen motor dapat menghasilkan diameter luar sempurna rotor (rotor OD) dan diameter dalam stator (stator ID) dalam konsentrisitas sempurna di antara satu sama lain.

Ada variasi celah udara yang jelas terlihat di setiap motor. Variasi celah udara ini memiliki efek parah pada ketepatan langkah motor. Mengetahui hal ini, mari kita bandingkan kutub angka pada dua desain motor langkah pertama 0,9 °. Motor 0,9 ° biasanya mengandung 8 kutub atau 16 kutub dalam masing-masing stator.

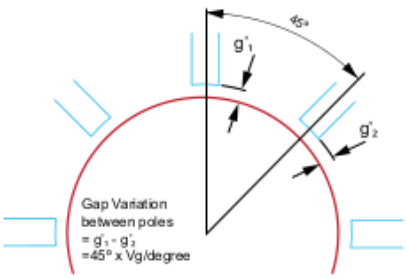
Gambar 2.11 Stator 8-Pole Tradisional



Gambar 2.12 Gulungan di stator

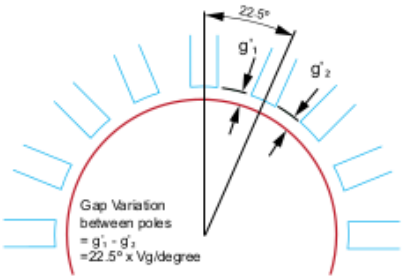
Pada Gambar 1 dan 1 di atas, melihat bahwa sudut antara setiap kutub pada desain 8-Pole adalah 45 °. Desain 16-Pole memiliki sudut 22,5 °. Saat motor melangkah, motor ini memegang dalam satu posisi di mana kutub stator sejajar dengan rotor. Kemudian bergerak ke kutub berikutnya untuk menyelaraskan dengan rotor. Dari satu kutub ke kutub lain, jika

celah udara sedikit lebih kecil atau lebih besar, akan melihat variasi ini dalam bentuk ketepatan langkah yang tidak konsisten, getaran dan resonansi, dan ketidakkonsistenan dalam torsi.



8-pole Design

Figure 1a: 8-Pole Design



16-pole Design

Figure 1b: 16-Pole Design

Perhatikan pada Gambar 1a, bahwa g1 dan g2 berpotensi sangat berbeda dibandingkan dengan Gambar 1b. Semakin dekat kutub, meminimalkan perbedaan variasi dari langkah ke langkah. Namun, ada kekurangan untuk masing-masing desain stator. Karena kendala ruang pada desain 16 kutub, ada lebih sedikit lilitan per kutub (putaran per koil) yang menghasilkan output torsi yang lebih rendah. Desain 8 pole tidak memiliki masalah ini sehingga tidak ada torsi yang relatif turun.

Meskipun desain 8 tiang memberikan torsi yang memadai, itu bisa juga memiliki variasi celah udara hingga dua kali lipat jika dibandingkan dengan desain 16 tiang yang menghasilkan akurasi kurang. Ketika memilih antara 8 dan 16 desain stator tiang, pelanggan pasti harus memilih antara motor torsi rendah dengan akurasi tinggi atau motor torsi tinggi dengan akurasi kurang; untuk sementara waktu, hanya itu yang tersedia.

Sebagai contoh, G5709 adalah motor stepper NEMA 23, 0,9 ° yang menggunakan desain 12 kutub dan diresapi dengan Teknologi Seri Signature. Hal ini memungkinkan untuk kinerja tertinggi: akurasi tinggi tanpa mengorbankan jumlah ruang berkelok-kelok.

Gambar 2.13 close-up stator 12-Pole G5709 7 gear/pole

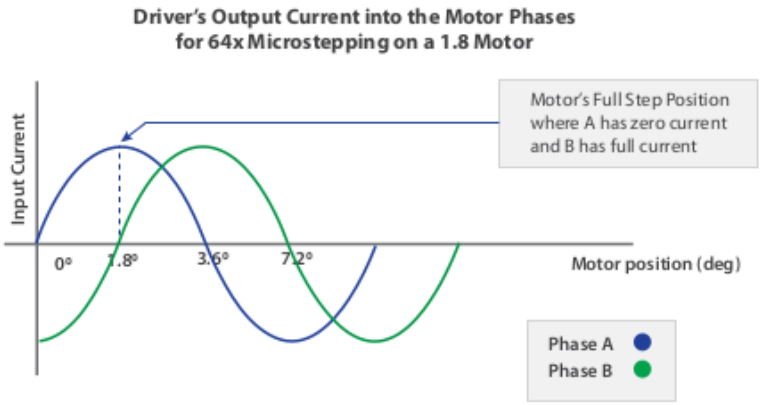
Dalam stepper 0,9 °, ada 100 gigi pada rotor. Oleh karena itu, jumlah gigi stator harus kurang dari 100. Setiap gigi ekstra yang dapat di peras ke dalam desain stator akan memberi lebih banyak torsi, melekat dalam desain, terlepas dari ruang gulungan.

Desain NEMA 23 0,9 ° stepper standar mengandung 8 kutub dan 10 gigi per kutub, dengan total 80 gigi pada stator. Motor G5709 baru berisi 12 kutub dan 7 gigi per kutub, dengan total 84 gigi pada stator. Meskipun ada sedikit ruang berliku dalam desain kutub 12 stator jika dibandingkan dengan desain tiang 8 stator, peningkatan jumlah gigi membuat perbedaan.

2.4.7. Mengukur Langkah Akurasi

Cara tradisional untuk menentukan akurasi langkah adalah dalam bentuk persentase dengan mengambil kesalahan langkah penuh dibagi dengan sudut langkah mendasar motor. Mari pahami ini dan lihat bagaimana bisa meyakinkan industri untuk mengukur dalam hal kesalahan langkah aktual dalam derajat atau lebih baik lagi, dalam menit-menit.

Metode tradisional menggunakan kesalahan persentase berasal dari aplikasi yang sering digunakan motor pada langkah penuh untuk menghasilkan torsi terbaik. Namun, dalam industri saat ini, sebagian besar aplikasi beroperasi pada langkah 1-2 atau langkah mikro. Keakuratan langkah selama melangkah penuh tidak lagi berfungsi untuk kebutuhan hari ini.



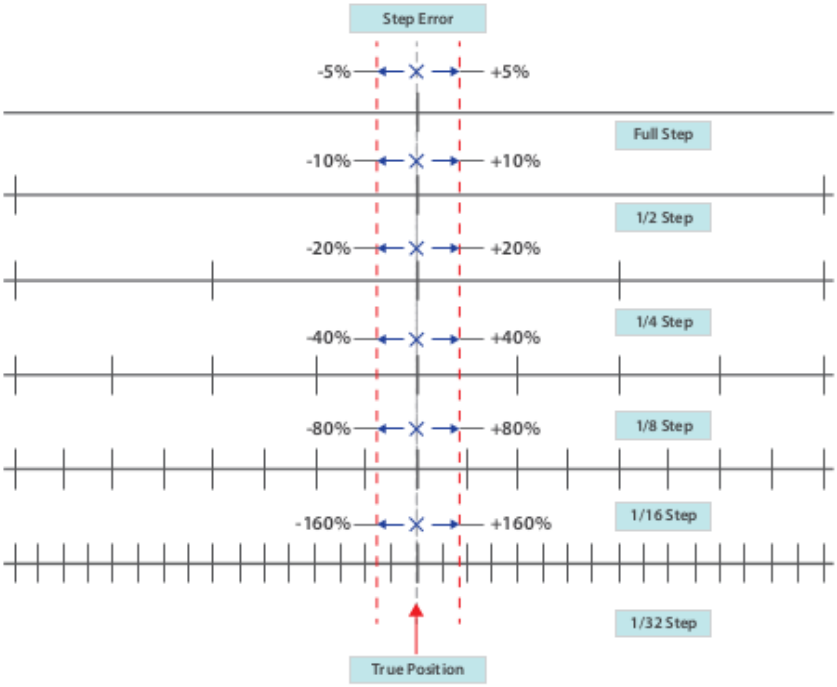
Gambar 2.14 Output Driver saat ini ke dalam Fase Motor

Akurasi langkah melekat dalam desain mekanik motor. Melangkah mikro meningkatkan resolusi langkah tetapi tidak akurasi langkah. Melangkah mikro motor tanpa akurasi langkah yang baik tidak memberikan gerakan yang halus.

Melangkah mikro dikendalikan oleh rasio fase motor saat ini dari driver motor. Tabel rasio saat ini dibuat berdasarkan kurva torsi sinusoidal dari motor step. Lihat Grafik 1 untuk pemahaman visual tentang cara kerja mikro terkait dengan arus input ke dua fase motor.

Motor yang tidak memiliki kurva torsi sinusoidal tidak dapat mengikuti posisi loncatan mikro. Karena tidak setiap motor dapat menghasilkan kurva torsi sinusoidal dan rasio arus tidak dapat sempurna dari kontrol elektronik, motor step tidak dapat mempertahankan akurasi yang sama dengan stepping penuh jika dibandingkan dengan stepping mikro.

Studi laboratorium menunjukkan bahwa kesalahan langkah dapat berlipat dua atau empat kali lipat dari langkah penuh menjadi 1/64 langkah. Motor dengan ketepatan langkah yang baik pada loncatan penuh tidak secara otomatis memberikan akurasi langkah yang baik pada loncatan mikro. Dengan anggapan bahwa motor step-step mikro yang sempurna dapat menjaga error step absolut-kurang dari jumlah step-step mikro, 5% step step error dari langkah penuh menjadi 10% dari langkah 1⁄2 dan 20% dari 1⁄4 langkah. Persentase ukuran langkah mikro akan mencapai 100% atau lebih pada jumlah langkah mikro yang tinggi. Oleh karena itu, melaporkan kesalahan langkah pada% dari langkah penuh tidak ada artinya. Lihat Grafik 2 di bawah ini.



Gambar 2.15 Akurasi versus Resolusi

Tidak tahu apakah aplikasi menggunakan 1⁄2 loncatan, 1⁄4 loncatan, 1/8 loncatan, 1/16 loncatan, 1/32 loncatan atau 1/64 loncatan. Karena apa pun yang lebih tinggi dari 1/64 loncatan memiliki dampak yang sangat kecil untuk aplikasi dalam hal gerakan halus & kebisingan. Sementara itu, motor dengan akurasi langkah yang baik pada 1/64 melangkah secara otomatis memberikan akurasi langkah yang baik pada melangkah lebih besar dari 1/64.

Mode Stepping

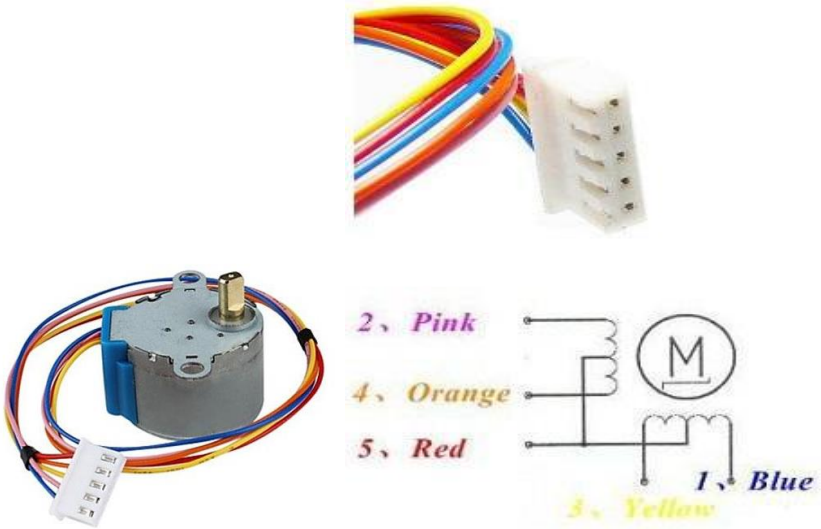
Full Stepping = 1 Microstepping
1/2 Stepping = 2 Microstepping
1/4 Stepping = 4 Microstepping
1/8 Stepping = 8 Microstepping
1/10 Stepping = 10 Microstepping
1/16 Stepping = 16 Microstepping
1/32 Stepping = 32 Microstepping
1/64 Stepping = 64 Microstepping
1/128 Stepping = 128 Microstepping
1/256 Stepping = 256 Microstepping

Kesalahan langkah dapat berkisar dari ± 1,5 arc-menit hingga ± 5 arc-menit untuk ukuran 17, motor 0,9 derajat, dan dari ± 3 arc-menit hingga ± 12 arc-menit untuk motor 1,8 derajat selama stepping mikro.

Motor yang unggul secara akurat juga lebih efisien secara mekanis. Getaran, dari loncatan energi limbah yang tidak akurat dan menghasilkan hilangnya daya dari sistem elektro mekanis. Motor dengan akurasi tinggi menghasilkan lebih banyak energi untuk beban daripada motor yang lebih kuat tetapi kurang akurat.

2.4.8. Motor Stepper 28BYJ-48

Motor Stepper 28BYJ-48 adalah sebuah motor stepper jenis unipolar 4 phase dengan tegangan supply sebesar 5V. Motor Stepper ini ideal digunakan dalam berbagai aplikasi elektronika seperti pembuatan robot, remote control dan perangkat lainnya yang membutuhkan suatu motor yang stepnya dapat diatur secara tepat.



Gambar 2.16 Konfigurasi Pin / Kabel

Spesifikasi dari 28BYJ-48 Stepper Motor :

- Supply Tegangan： 5VDC
- Jumlah Phase 4
- Speed Variation Ratio 1/64
- Stride Angle 5.625° /64
- Frekuensi 100Hz
- DC resistance 50Ω±7%(25℃)
- Idle In-traction Frequency > 600Hz

- Idle Out-traction Frequency > 1000Hz
- In-traction Torque >34.3mN.m(120Hz)
- Self-positioning Torque >34.3mN.m
- Friction torque 600-1200 gf.cm
- Pull in torque 300 gf.cm
- Insulated resistance >10MΩ(500V)
- Insulated electricity power 600VAC/1mA/1s
- Insulation grade A
- Rise in Temperature <40K(120Hz)
- Noise <35dB(120Hz,No load,10cm)

2.4.9. Gear Rasio 28BYJ-48

Pada rasio gigi dalam motor stepper adalah seperti pada gambar dibawah ini.

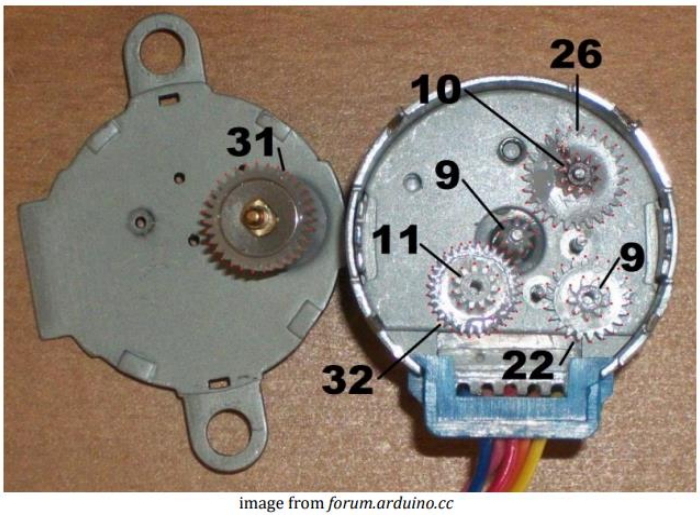


image from *forum.arduino.cc*

Gambar 2.17 Gear rasio

Rasio roda gigi adalah:

$$\frac{31 \times 32 \times 26 \times 22}{11 \times 10 \times 9 \times 9} = 63.68395$$

Putaran Penuh

- Internal motor: 32 langkah per revolusi
- Great reduction ratio: 1 / 63.68395, sekitar 1/64
- Jadi dibutuhkan 32 × 64 = 2048 langkah per revolusi untuk poros output

Perhitungan Putaran

Diketahui putaran penuh 2048 jadi perhitungannya

2048 / 2 = 1024 nilai, 180 putaran derajat
1024 / 2 = 512 nilai, 90 putaran derajat
512 / 2 = 256 nilai, 45 putaran derajat
256 / 2 = 128 nilai, 22,5 putaran derajat
138 / 2 = 64 nilai, 11,25 putaran derajat

Nilai 64 ini yang nantinya digunakan di proses putaran per step motor stepper, dengan bantuan aplikasi Arduino IDE, dan nilai 11,25 yang akan dadi acuan putaran per klik pada aplikasinya.

Switching Sequence

Untuk Switching Sequence (Urutan peralihan) dari pergerakan Motor stepper ditunjukan tabel berikut ini:

Tabel 2.2 Urutan peralihan stepper

Lead Wire Color	-->CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 Orange	-	-						
3 Yellow		-	-	-				
2 Pink				-	-	-		
1 Blue						-	-	-

Tabel diatas berguna untuk mengatur pergerakan motor stepper. Tanda strip (-) nantinya akan diberi masukan HIGH, dan kosong diberi masukan LOW pada program arduino.

2.5. Board ULN2003

Board dengan IC ULN 2003 merupakan IC yang mempunyai 16 buah pin dan sudah dirangkai dengan papan PCB beserta dengan pin dayanya. IC ULN2003 adalah sebuah IC dengan ciri memiliki 7 bit input, tegangan maksimal 50 volt dan arus 500 ma. IC ini termasuk jenis TTL. Didalam IC ini terdapat transistor darlington. Transistor darlington merupakan 2 buah transistor yang dirangkai dengan konfigurasi khusus

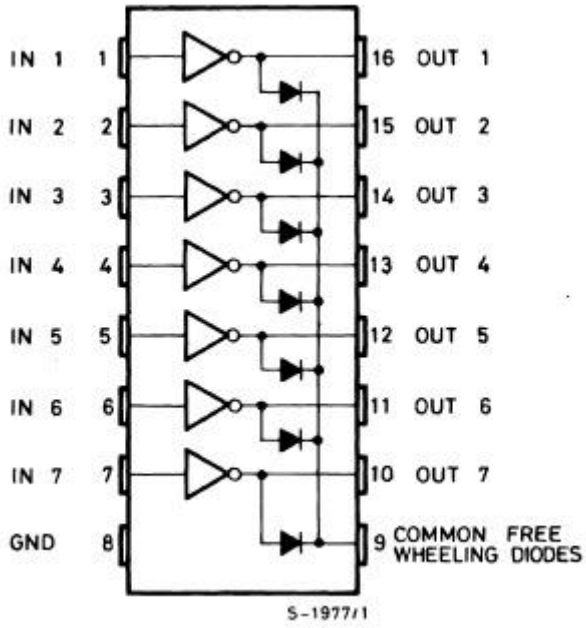
untuk mendapatkan penguatan ganda sehingga dapat menghasilkan penguatan arus yang besar. Fungsi IC ULN 2003 adalah sebagai driver untuk mencatu daya pada relay, karena keluaran dari mikrokontroler tidak dapat mencatu daya yang terdapat pada relay secara langsung. IC ULN idealnya cocok untuk komunikasi sirkuit logic low - level. Prinsip kerja dari IC ULN 2003 ini yaitu bila diberi tegangan inputan pada ic ULN2003 sebesar 3,3 Volt maka pada bagian output ic ULN2003 akan terhubung ke tegangan - (minus).

Gambar 2.18 Board Dengan IC ULN2003

Output yang akan dihubungkan ke motor stepper dan ada yang berfungsi sebagai catu daya. Catu daya ini terdiri dari catu daya (+) dan ground. Besar catu daya yang dihubungkan tergantung pada tipe motor stepper. Bentuk fisik IC ULN 2003 adalah seperti Gambar.

Gambar 2.19 IC ULN2003

Sedang isi dari IC ULN 2003 dan fungsi dari masing-masing pin adalah sebagai berikut :



Gambar 2.20 Pin Out IC ULN2003

2.6. Board LM2596

LM2596 Adjustable Step Down DC - DC Module LM2596 Adjustable DC-DC seperti ditunjukkan pada Gambar dibawah ini menggunakan step-down LM2596S regulator untuk menyediakan pasokan listrik yang stabil bagi pengguna. Tegangan output disesuaikan dan dapat memastikan beban arus keluaran sebesar 3A, modul ini tegangan input antara 3V hingga 40V, dengan tegangan output dapat disesuaikan antara 1,5V hingga 35V.

Gambar 2.21 Board LM2596

2.7. Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) adalah sebuah protokol jaringan lapisan aplikasi yang digunakan untuk sistem informasi terdistribusi, kolaboratif, dan menggunakan hipermedia. Protokol HTTP didefinisikan oleh Tim Berners-Lee dalam RFC 1945 versi 1.0 dan digunakan sejak tahun 1990. Penyempurnaan protokol HTTP menjadi versi 1.1 yang dispesifikasikan oleh IETF dengan RFC 2616. HTTP bersifat request – response, yaitu HTTP client(user agen misalnya) mengirimkan permintaan (request) ke HTTP server dan server merespon sesuai request tersebut. User agen sebagai contoh adalah Mozilla, Netscape, Google Chrome, atau browser berbasis teks contohnya Lynx atau links dan sebagainya.

Pada protokol HTTP terdapat 3 jenis hubungan dengan perantara proxy, gateway, dan tunnel. Proxy bertindak sebagai agent penerus, menerima request dalam bentuk Uniform Resource Identifier (URI) absolut, mengubah format request dan mengirimkan request ke server yang ditunjukan oleh URI. 4 Gateway bertindak sebagai agen penerima dan menterjemahkan request ke protokol server yang dilayaninya. Tunnel bertindak sebagai titik Relay antara dua hubungan HTTP tanpa mengubah request dan response HTTP. Tunnel digunakan jika komunikasi perlu melalui sebuah perantara dan perantara tersebut tidak mengetahui isi pesan dalam hubungan tersebut.

Perbedaan mendasar antara HTTP/1.1 dengan HTTP/1.0 adalah penggunaan hubungan persistent. HTTP/1.0 membuka satu koneksi untuk tiap permintaan satu URI, sedangkan HTTP/1.1 dapat menggunakan sebuah koneksi TCP untuk beberapa permintaan URI (persistent) (header Conection : keepAlive), kecuali jika client menyatakan tidak hendak menggunakan hubungan persistent (header Conection : close). HTTP port TCP default adalah 80, namun itu bisa diganti dengan nomor TCP lain diantara 1023 – 65535.

2.8. Arduino IDE

IDE itu merupakan kependekan dari Integrated Development Environment, atau secara bahasa mudahnya merupakan lingkungan terintegrasi yang digunakan untuk melakukan pengembangan. Disebut sebagai lingkungan karena melalui software inilah Arduino dilakukan pemrograman untuk melakukan fungsi-fungsi yang dibenamkan melalui sintaks pemrograman. Arduino menggunakan bahasa pemrograman sendiri yang menyerupai bahasa C. Bahasa pemrograman Arduino (Sketch) sudah dilakukan perubahan untuk memudahkan pemula dalam

melakukan pemrograman dari bahasa aslinya. Sebelum dijual ke pasaran, IC mikrokontroler Arduino telah ditanamkan suatu program bernama Bootloader yang berfungsi sebagai penengah antara compiler Arduino dengan mikrokontroler.

Arduino IDE dibuat dari bahasa pemrograman JAVA. Arduino IDE juga dilengkapi dengan library C/C++ yang biasa disebut Wiring yang membuat operasi input dan output menjadi lebih mudah. Arduino IDE ini dikembangkan dari software Processing yang dirombak menjadi Arduino IDE khusus untuk pemrograman dengan Arduino.

2.8.1. Software Serial

Software serial adalah komunikasi serial yang memungkinkan terjadinya pertukaran informasi antara mikrokontroler arduino dan IDE aplikasi pada komputer. Pertukaran informasi tersebut seperti status yang terjadi pada rangkaian mikrokontroler tersebut, tergantung apa yang ingin ditampilkan informasinya melalui script code pada arduino.

Jenis command komunikasi serial Arduino :

- **Serial.begin()** : untuk menentukan kecepatan pengiriman dan penerimaan data melalui port serial. Kecepatan yang umum digunakan adalah 9600 bit per detik (9600 bps). Namun, kecepatan hingga 115.200 didukung oleh Arduino Uno. Contoh yang sering digunakan yaitu Serial.begin(9600).
- **Serial.end()** : digunakan untuk menghentikan program akan perintah komunikasi serial.
- **Serial.available ()** : berguna untuk menghasilkan jumlah byte di port serial yang belum terbaca. Jika port serial dalam keadaan kosong, maka fungsi ini dapat menghasilkan nilai nol.
- **Serial.read()** : berguna untuk membaca satu byte data yang terdapat di port serial. Setelah pemanggilan Serial.read(), jumlah data di port serial berkurang satu.
- **Serial.print(data)** : berfungsi untuk mengirimkan data ke port serial. Apabila argumen format disertakan, data yang dikirim akan menyesuaikan dengan format tersebut. Dalam hal ini, format yang digunakan bisa berupa.
- **Serial.flush()** : berfungsi sebagai untuk pengosongan data pembacaan yang ditaruh pada buffer.
- **Serial.parseFloat()** : berfungsi untuk bilangan titik mengambang atau real.

- **Serial.println(data)** : memiliki fungsi yang hampir sama dengan serial print, yang memberi efek perpindahan baris berikutnya.
- **Serial.parseln()** : untuk menghasilkan nilai bulat.

2.8.2. Library motor Stepper

Berikut fungsi - fungsi pendukung yang ada pada library motor stepper yang akan digunakan :

- `Stepper(steps, pin1, pin2, pin3, pin4)`: fungsi ini menciptakan instance baru dari kelas Stepper yang mewakili motor stepper tertentu yang melekat pada papan Arduino. Digunakan di bagian paling atas sketsa, diatas `setup()` dan `loop()`. Jumlah parameter tergantung pada bagaimana menghubungkan kabel motor, baik menggunakan dua atau empat pin papan Arduino.
- `setSpeed(rpms)`: Mengatur kecepatan motor dalam rotasi per menit (RPM). Fungsi ini tidak membuat motor berputar, cukup atur kecepatannya ketika memanggil `step()`.
- `step(steps)`**:** Mengubah motor sejumlah langkah tertentu, pada kecepatan yang ditentukan oleh panggilan terbaru ke `setSpeed()`. Fungsi ini memblokir ; yaitu, itu akan menunggu sampai motor selesai bergerak untuk melewati kontrol ke baris berikutnya dalam sketsa. Misalnya, jika mengatur kecepatan ke, katakanlah, 1 RPM dan disebut step(100) pada motor 100 langkah, fungsi ini akan membutuhkan waktu satu menit penuh untuk dijalankan. Untuk kontrol yang lebih baik, pertahankan kecepatan tinggi dan hanya lakukan beberapa langkah dengan setiap panggilan ke `step()`.

2.9. MIT App Inventor

MIT App Inventor adalah alat berbasis web untuk membangun aplikasi Android. Ini sering disebut sebagai pemrograman visual, yang berarti pengguna dapat melakukan tugas pemrograman tanpa memasukkan kode komputer apa pun.

Penemu Aplikasi dikelola dan dikembangkan secara aktif oleh Mobile Learning Lab MIT (proyek ini awalnya dibangun oleh Google). App Inventor semakin populer di kalangan pendidik sebagai cara untuk memperkenalkan mereka yang tidak memiliki pengalaman pemrograman dengan prinsip-prinsip ilmu komputer dan pengembangan aplikasi. Ini juga berfungsi sebagai langkah pertama yang bagus bagi mereka yang

berkecimpung dengan pemrograman atau ingin menambah pengetahuan mereka tentang cara kerja aplikasi smartphone.

Pengerjaan berlangsung di dua bagian utama Aplikasi: Desainer dan Editor Blok. Pemrograman berlangsung di Editor Blok. Di sana diberi tahu aplikasi apa yang harus dilakukan dan memberikan instruksi spesifik untuk melakukannya.

Kemampuan khusus diprogram melalui menghubungkan potongan puzzle. Seiring waktu, akan mempelajari apa yang dilakukan masing-masing blok dan menemukan berbagai cara bagi mereka untuk saling berinteraksi. Potongan-potongan yang tidak berinteraksi tidak akan terhubung satu sama lain.

MIT merilis App Inventor 2 pada Desember 2013, menciptakan alat yang lebih kuat dan lebih mudah digunakan. Peningkatan paling signifikan adalah bahwa semua pekerjaan dilakukan di dalam browser (versi sebelumnya memerlukan unduhan perangkat lunak untuk beberapa kemampuan).

Peningkatan ini paling mempengaruhi emulator layar, yang menempatkan layar perangkat Android virtual di komputer. Menggunakan emulator ini memberikan perspektif tentang bagaimana aplikasi akan terlihat dan berfungsi ketika digunakan. Ini sangat berguna bagi mereka yang tidak memiliki perangkat Android atau siapa pun dalam lingkungan pendidikan yang ingin memantau kemajuan siswa dengan melihat aplikasi yang dibuat di layar komputer.

App Inventor juga menawarkan metode untuk menggunakan aplikasi secara real time saat melakukan pekerjaan di atasnya: aplikasi AI Companion. Dengan unduhan gratis ini dari Google Play, dapat dilihat perubahan dan pengembangan aplikasi saat mengerjakannya. Aplikasi Companion juga berfungsi secara nirkabel, jadi tidak perlu menghubungkan ponsel secara fisik ke komputer saat bekerja di App Inventor.

2.9.1. Designer

Pembuatan aplikasi dimulai dari designer. Di sini membuat aplikasi pengguna, atau aplikasi "tampilan dan nuansa". Dapat juga menambahkan komponen yang diperlukan untuk menerima input dari pengguna, serta komponen yang diperlukan untuk menampilkan output atau informasi kepada pengguna. designer juga adalah tempat menentukan komponen yang tidak terlihat yang akan digunakan aplikasi, seperti dialer, GPS, atau SMS.

2.9.2. Blocks Editor

Editor Blok adalah tempat memprogram perilaku suatu aplikasi . Di sini akan menambahkan perintah yang melakukan pekerjaan aplikasi.

MIT App Inventor menggunakan metafora laci yang berisi potongan puzzle untuk pemrograman. Setiap item dalam palet Blok di bawah Bawaan dianggap sebagai laci. Laci berisi potongan-potongan yang tampak seperti puzzle. Pemrograman dilakukan dengan menghubungkan potongan-potongan yang tampak seperti puzzle. Meskipun terlihat sederhana, App Inventor memiliki banyak kemampuan hebat yang memungkinkan pengguna untuk membangun aplikasi yang kompleks.

2.9.3. Komponen Web

Komponen Web di App Inventor memfasilitasi aplikasi yang berkomunikasi dengan layanan web melalui Hypertext Transfer Protocol (HTTP) standar. Protokol itu menyediakan metode Get, Put, dan Post untuk membawa informasi ke dalam aplikasi. Informasi datang bukan sebagai halaman yang dapat ditampilkan, tetapi sebagai data yang dapat ditampilkan atau proses sesuai keinginan.

Komponen ini levelnya cukup rendah, dan menggunakannya membutuhkan keahlian pemrograman. Biasanya mengatur properti Web URL untuk menentukan layanan web yang akan berkomunikasi dengan server, dan kemudian aplikasi memanggil salah satu metode HTTP untuk meminta tindakan. Ini rumit karena perlu memahami API layanan web (protokol untuk komunikasi), dan perlu memahami cara memproses informasi yang dikembalikan oleh layanan web ke aplikasi. Pemrosesan ini dikenal sebagai parsing, dan ini merupakan teknik pemrograman tingkat lanjut. Komponen web yang menyediakan fungsi permintaan HTTP GET, POST, PUT, dan DELETE.

Tabel 2.3 Properti web App Invertor

Properti	Keterangan
AllowCookies	Apakah cookie dari respons harus disimpan dan digunakan dalam permintaan berikutnya. Cookie hanya didukung di Android versi 2.3 atau lebih tinggi.

RequestHeaders	Header permintaan, sebagai daftar sublist dua elemen. Elemen pertama dari setiap sublist mewakili nama bidang header permintaan. Elemen kedua dari setiap sublist mewakili nilai-nilai bidang header permintaan, baik nilai tunggal atau daftar yang berisi beberapa nilai.
ResponseFileName	Nama file tempat respons harus disimpan. Jika SaveResponse benar dan ResponseFileName kosong, maka nama file baru akan dihasilkan.
SaveResponse	Nama file tempat respons harus disimpan. Jika SaveResponse benar dan ResponseFileName kosong, maka nama file baru akan dihasilkan.

Tabel 2.4 Event web App Invertor

Event	Keterangan
GotFile(text url, number responseCode, text responseType, text fileName)	Event yang menunjukkan bahwa permintaan telah selesai.
GotText(text url, number responseCode, text responseType, text responseContent)	Event yang menunjukkan bahwa permintaan telah selesai.

Tabel 2.5 Method web App Invertor

Method	Keterangan
text BuildRequestData(list list)	Mengonversi daftar sub-elemen dua elemen, yang mewakili pasangan nama dan nilai, ke string yang diformat sebagai jenis media aplikasi / x-www-form-urlencoded, cocok untuk diteruskan ke PostText.
ClearCookies()	Menghapus semua cookie untuk komponen Web ini.
Delete()	Melakukan permintaan HTTP DELETE menggunakan properti Url dan mengambil respons. Jika properti SaveResponse benar, respons akan

	disimpan dalam file dan acara GotFile akan dipicu. Properti ResponseFileName dapat digunakan untuk menentukan nama file. Jika properti SaveResponse salah, acara GotText akan dipicu.
Get()	Melakukan permintaan GET HTTP menggunakan properti Url dan mengambil respons. Jika properti SaveResponse benar, respons akan disimpan dalam file dan acara GotFile akan dipicu. Properti ResponseFileName dapat digunakan untuk menentukan nama file. Jika properti SaveResponse salah, acara GotText akan dipicu.
text HtmlTextDecode(text htmlText)	Mendekode nilai teks HTML yang diberikan. Entitas karakter HTML seperti & amp ;, & lt ;, & gt ;, & apos ;, dan & quot; diubah menjadi &, <,>, ', dan ". Entitas seperti & # xhhhh, dan & # nnnn diubah ke karakter yang sesuai.
any JsonTextDecode(text jsonText)	Mendekode nilai enkode JSON yang diberikan untuk menghasilkan nilai AppInventor yang sesuai. Daftar JSON [x, y, z] menerjemahkan ke daftar (xyz), objek JSON dengan nama A dan nilai B, (dilambangkan sebagai A: B terlampir dalam kurung kurawal) menerjemahkan ke daftar ((AB)), yang adalah, daftar yang berisi daftar dua elemen (AB).
PostFile(text path)	Melakukan permintaan HTTP POST menggunakan properti Url dan data dari file yang ditentukan. Jika properti SaveResponse benar, respons akan disimpan dalam file dan acara GotFile akan dipicu. Properti ResponseFileName dapat digunakan untuk menentukan nama file. Jika properti SaveResponse salah, acara GotText akan dipicu.
PostText(text text)	Melakukan permintaan POST HTTP menggunakan properti Url dan teks yang

	ditentukan. Karakter teks dikodekan menggunakan pengkodean UTF-8. Jika properti SaveResponse benar, respons akan disimpan dalam file dan acara GotFile akan dipicu. Properti responseFileName dapat digunakan untuk menentukan nama file. Jika properti SaveResponse salah, acara GotText akan dipicu.
PostTextWithEncoding(text text, text encoding)	Melakukan permintaan POST HTTP menggunakan properti Url dan teks yang ditentukan. Karakter teks dikodekan menggunakan pengkodean yang diberikan. Jika properti SaveResponse benar, respons akan disimpan dalam file dan acara GotFile akan dipicu. Properti ResponseFileName dapat digunakan untuk menentukan nama file. Jika properti SaveResponse salah, acara GotText akan dipicu.
PutFile(text path)	Melakukan permintaan HTTP PUT menggunakan properti Url dan data dari file yang ditentukan. Jika properti SaveResponse benar, respons akan disimpan dalam file dan acara GotFile akan dipicu. Properti ResponseFileName dapat digunakan untuk menentukan nama file. Jika properti SaveResponse salah, acara GotText akan dipicu.
PutText(text text)	Melakukan permintaan PUT HTTP menggunakan properti Url dan teks yang ditentukan. Karakter teks dikodekan menggunakan pengkodean UTF-8. Jika properti SaveResponse benar, respons akan disimpan dalam file dan acara GotFile akan dipicu. Properti responseFileName dapat digunakan untuk menentukan nama file. Jika properti SaveResponse salah, acara GotText akan dipicu.
PutTextWithEncoding(te	Melakukan permintaan PUT HTTP

xt text, text encoding)	menggunakan properti Url dan teks yang ditentukan. Karakter teks dikodekan menggunakan pengkodean yang diberikan. Jika properti SaveResponse benar, respons akan disimpan dalam file dan acara GotFile akan dipicu. Properti ResponseFileName dapat digunakan untuk menentukan nama file. Jika properti SaveResponse salah, acara GotText akan dipicu.
text UriEncode(text text)	Mengkodekan nilai teks yang diberikan sehingga dapat digunakan dalam URL.
text UriDecode(text text)	Mendekodekan nilai teks yang disandikan sehingga nilai-nilai itu tidak lagi disandikan URL.
any XMLTextDecode(text XmlText)	Mendekode string XML yang diberikan untuk menghasilkan struktur daftar. Lihat dokumentasi Penemu Aplikasi pada "Topik, catatan, dan detail lainnya" untuk informasi.

2.9.4. TinyDB

TinyDB adalah komponen tidak terlihat yang digunakan untuk menyimpan data pada suatu aplikasi. Setiap aplikasi memiliki basis data kecil yang diakses komponen TinyDB untuk menyimpan dan mengambil informasi.

Aplikasi yang dibuat diinisialisasi setiap kali dijalankan. Jika aplikasi menetapkan nilai variabel dan pengguna kemudian berhenti dari aplikasi, nilai variabel itu tidak akan diingat pada saat berikutnya aplikasi dijalankan. TinyDB adalah penyimpanan data persisten untuk aplikasi, yang berarti bahwa data yang disimpan di sana akan tersedia setiap kali aplikasi dijalankan. Salah satu contoh adalah skor tinggi sepanjang masa untuk aplikasi game - itu disimpan di perangkat setiap kali pengguna bermain, dan diambil setiap kali aplikasi dibuka.

Item data disimpan di bawah tag. Untuk menyimpan item data, ditentukan tag tempat penyimpanannya. Selanjutnya, dapat mengambil item data yang disimpan di bawah tag yang diberikan. Jika tidak ada nilai yang disimpan di bawah tag, maka nilai yang dikembalikan adalah teks kosong. Akibatnya, untuk melihat apakah tag memiliki nilai yang disimpan di

bawahnya, uji apakah nilai kembali sama dengan teks kosong (mis., Kotak teks tanpa teks yang diisi).

TinyDB dapat digunakan untuk menyimpan informasi yang ingin agar aplikasi mengingat tentang pemilik / pengguna di ponsel itu. Skor tinggi pribadi, siapa yang login, atau lokasi tipe data yang mungkin ingin disimpan. Didalam TinyDB MIT App Inventor perlu diketahui method seperti dibawah ini.

- ClearAll() →Yaitu menghapus semua data yang tersimpan dalam TinyDB
- ClearTaq taq() →Yaitu Mengosongkan Data yang terdapat pada taq yang dipilih.
- GatTaqs() →Yaitu mengembalikan semua taq pada TinyDB.
- GetValue Taq() →Yaitu mengambil nilai yang tersimpan pada taq yang dipanggil, jika tidak ada taq seperti yang dipanggil maka kembalikan nilai taqNotThere.
- StoreValue Taq() →Yaitu menyimpan nilai didalam taq yang diberikan. Dan penyimpananpun tetap ada saat aplikasi dikeluarkan atau telepon direstart.