LAPORAN AKHIR

PRAKTIK KERJA LAPANGAN

Kontrol Kecepatan Motor Servo pada Pergerakan *Lower Limb Exoskeleton*

Badan Riset dan Inovasi Nasional (BRIN) Bandung

Disusun oleh:

Shabrina Alvie Aditya	(082011733003)
Mutia Rahman Azzahra	(082011733023)

PROGRAM STUDI S1 - TEKNIK BIOMEDIS

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS AIRLANGGA

2023

LEMBAR PENGESAHAN

1.	Judul Kegiatan	: Kontrol Kecepatan Motor Servo pada Pergerakan *Lower Limb Exoskeleton*
2.	Pelaksana Kegiatan I	
	a. Nama lengkap	: Shabrina Alvie Aditya
	b. NIM	: 082011733003
	c. Jurusan	: Teknik Biomedis
	d. Perguruan tinggi	: Universitas Airlangga
3.	Pelaksana Kegiatan I	
	a. Nama lengkap	: Shabrina Alvie Aditya
	b. NIM	: 082011733003
	c. Jurusan	: Teknik Biomedis
	d. Perguruan tinggi	: Universitas Airlangga
4.	Pembimbing I	
	a. Nama Lengkap dan Gelar	: Endah Purwanti, S.Si., M.T.
	b. NIP	: 197710312009122003
5.	Pembimbing II	
	a. Nama Lengkap dan Gelar	: Dwi Esti Kusumandari, M.T.
	b. NIP	: 197508152000122001
6.	Waktu Pelaksanaan	: 9 Januari – 3 Februari 2023

Surabaya, 20 Februari 2023

Disetujui oleh

Pembimbing I	Pembimbing II
(Endah Purwanti, S.Si., M.T.)	(Dwi Esti Kusumandari, M.T.)
NIP. 197710312009122003	NIP. 197508152000122001

Ketua Program Studi S1 Teknik Biomedis
Fakultas Sains dan Teknologi
Universitas Airlangga

(Dr. Riries Rulaningtyas, S.T., M.T.)
NIP. 197903152003122002

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas kesempatan yang diberikan untuk mengerjakan dan menuntaskan Laporan Praktik Kerja Lapangan yang berjudul **"Kontrol Kecepatan Motor Servo pada Pergerakan *Lower Limb Exoskeleton*"**. Laporan ini dibuat dengan harapan dapat memberikan gambaran kegiatan pelaksanaan yang telah kami laksanakan serta menyajikan hasil penelitian yang telah diperoleh.

Penulis menyadari bahwa laporan ini masih jauh dari kesempurnaan dengan segala kekurangannya. Untuk itu kami sebagai penulis mengharapkan adanya kritik dan saran dari semua pihak demi lebih sempurnanya dari laporan akhir praktik kerja lapangan ini. Akhir kata, penulis berharap semoga dengan adanya laporan akhir ini dapat memberikan manfaat kepada rekan-rekan mahasiswa dan pembaca sekaligus dapat menambah pengetahuan mengenai Praktik Kerja Lapangan.

UCAPAN TERIMA KASIH

Kami mengucapkan terima kasih yang sebesar-besarnya kepada kedua institusi yang telah menaungi kami berdua yaitu Balai Pengembangan Instrumentasi BRIN dan Fakultas Sains dan Teknologi Universitas Airlangga serta para pembimbing kami yaitu Ibu Endah Purwanti, S.Si., M.T. dan Ibu Dwi Esti Kusumandari, M.T. atas dukungan moral dan edukasi yang telah memotivasi kami dalam melaksanakan penelitian ini. Kami juga mengucapkan terima kasih kepada senior yang turut membantu dalam penelitian ini yaitu Jajang Mulyana dan Tasya Claudia serta kolega-kolega dan senior sesama peneliti di Balai Pengembangan Instrumentasi atas bantuan yang telah diberikan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Meningkatnya populasi lansia dalam beberapa tahun terakhir mengakibatkan penurunan kualitas hidup secara signifikan. Dengan adanya kemajuan di bidang robotika dan teknologi kini penggunaan eksoskeleton menjadi pusat perhatian sebagai rehabilitasi bagi penderita penyakit stroke, cedera tulang belakang, kelemahan anggota badan, sendi kaku, dan lainnya yang dapat menyebabkan disfungsi berjalan. Eksoskeleton adalah perangkat elektromekanis, termasuk dalam robot yang dapat dipakai yang memberikan augmentasi atau bantuan kepada manusia. Eksoskeleton juga diklasifikasikan sebagai robot ortotik yang berbeda dari robot prostetik yang digunakan untuk menggantikan anggota tubuh yang hilang dari orang yang diamputasi (Qureshi, Muhammad Hamza, et al. 2018). Penelitian mengenai eksoskeleton ekstremitas bawah telah dilakukan oleh peneliti dari berbagai negara. Salah satunya adalah Robot HAL yang merupakan robot rehabilitasi eksoskeleton pertama yang dikembangkan oleh Universitas Tsukuba di Jepang dan model terbarunya yaitu HAL-5 untuk membantu pasien berjalan normal dan naik turun tangga. Selain itu, reWALK personal dan ReWalk rehabilitasi dikembangkan oleh Israel dimana pasien diperbolehkan berdiri, berjalan, menaiki tangga, memberikan fisioterapi untuk pasien lumpuh (Tian, M., et al, 2019).

Dengan adanya Praktik Kerja Lapangan, mahasiswa dapat mengambil tema "rancang bangun *lower limb exoskeleton* (LLE)" dan mendapatkan wawasan baru yang berkaitan dengan tema tersebut serta pengalaman kerja lapangan yang sesungguhnya. Penggunaan motor servo SPT5435LV yaitu sebagai penggerak dari tiap-tiap joint seperti *hip*, *knee* dan *ankle* pada eksoskeleton *lower limb*. Sistem yang dipakai dalam rangkaian eksoskeleton *lower limb* yaitu sistem dengan mode otomatis dimana menggunakan baterai sebagai penambah daya untuk menggerakan motor servo. Praktik ini dilakukan dengan menggunakan prototipe *lower limb exoskeleton 3D-printing* yang telah disediakan oleh BRIN.

1.2. Rumusan Masalah

Berdasarkan latar belakang diatas maka dapat diambil rumusan masalah dalam praktik kerja lapangan ini sebagai berikut.

1. Bagaimana cara memodifikasi hardware dan software/kode program sebagai sistem kendali kecepatan motor untuk pergerakan eksoskeleton lower limb?
2. Bagaimana cara mengatur pergerakan pada setiap sendi eksoskeleton lower limb?
3. Bagaimana cara mengontrol kecepatan gerakan motor untuk pergerakan eksoskeleton lower limb?
4. Bagaimana hasil simulasi sistem kendali kecepatan motor yang telah dirancang ?

1.3. Tujuan Penelitian

Adapun tujuan dari kerja praktek lapangan ini adalah diantaranya:

1. Memodifikasi hardware dan software/kode program sebagai sistem kendali kecepatan motor untuk pergerakan eksoskeleton lower limb.
2. Mengatur gerakan pada setiap sendi eksoskeleton lower limb.
3. Mengontrol kecepatan gerakan motor untuk pergerakan eksoskeleton lower limb.
4. Mengetahui hasil simulasi kendali kecepatan motor yang telah dirancang.

1.4. Manfaat Penelitian

Adapun manfaat dari kerja praktek lapangan ini yaitu:

1. Mampu mendapatkan informasi terkait perancangan dan cara kerja prototipe eksoskeleton lower limb dengan menggunakan mode otomatis yang sesuai dengan siklus gait normal pada sendi-sendi hip, knee dan ankle, serta
2. mendapatkan perbandingan grafik antara percobaan dengan literatur referensi.

BAB II

GAMBARAN UMUM BPI - BRIN

2.1. Sejarah Singkat BRIN

Gambar 2.1. Logo BRIN

Badan Riset dan Inovasi Nasional merupakan lembaga pemerintah non-kementerian yang berada dibawah dan bertanggung jawab kepada Presiden Indonesia melalui menteri yang membidangi urusan pemerintahan di bidang riset dan teknologi. Lembaga ini pertama kali dibentuk oleh Presiden Joko Widodo melalui Peraturan Presiden Nomor 74 Tahun 2019 yang melekat kepada Kementerian Riset dan Teknologi (kemenristek) sehingga Menteri Riset dan Teknologi juga bertindak sebagai Kepala Brin. Saat ini, BRIN memiliki Ketua Dewan Pengarah dari BPIP yaitu Megawati Soekarnoputri.

Pada 28 April 2021, berdasarkan Peraturan Presiden Nomor 33 Tahun 2021 tentang Badan Riset dan Inovasi Nasional, BRIN menjadi lembaga yang berdiri sendiri dengan mengintegrasikan Kementerian Riset dan Teknologi dan 4 (empat) lembaga pemerintah nonkementerian (LPNK) yakni Badan Pengkajian dan Penerapan Teknologi (BPPT), Badan Tenaga Nuklir Nasional (BATAN), Lembaga Penerbangan dan Antariksa Nasional (LAPAN), dan Lembaga Ilmu Pengetahuan Indonesia (LIPI). Peraturan Presiden Nomor 33 Tahun 2021 tentang Badan Riset dan Inovasi Nasional kemudian dicabut dan digantikan oleh Peraturan Presiden Nomor 78 Tahun 2021 tentang Badan Riset dan Inovasi Nasional.

2.2. Sejarah Singkat BPI BRIN

UPT BPI-LIPI berdiri sejak tahun 1987, berdasarkan SK Ketua LIPI No. 1333/Kep/D.5/1987 pada tanggal 25 November 1987. Di dalam perkembangannya, pada

tahun 2002 mengalami reorganisasi dan berubah nama dari UPT Balai Pemanfaatan hasil Penelitian dan Pengembangan Instrumentasi menjadi UPT Balai Pengembangan Instrumentasi dengan Keputusan Kepala LIPI No. 1025/M/2002 tanggal 12 Juni 2002. Pada pasal 1 disebutkan bahwa UPT Balai Pengembangan Instrumentasi-LIPI (UPT BPI – LIPI) yang dahulu dikenal dengan sebutan UPT Balai LIN-LIPI berada di bawah Deputi Bidang Jasa Ilmiah LIPI dan pembinaan sehari- harinya dilakukan oleh Pusat Penelitian Kalibrasi, Instrumentasi, dan Metrologi LIPI (Puslit KIM-LIPI).

UPT BPI LIPI mempunyai tugas melaksanakan, memanfaatkan, menyebarluaskan dan menerapkan hasil penelitian dan pengembangan di bidang instrumentasi dan kalibrasi, serta mempunyai program pengembangan MSTQ. Dalam menjalankan Tupoksi UPT Balai Pengembangan Instrumentasi pada 5 (lima) tahun mendatang telah menetapkan Visi dan Misi dengan Kompetensi intinya adalah: "Pengembangan dan penerapan bidang instrumentasi teknik pengukuran serta meningkatkan kemampuan SDM untuk menunjang mutu produk industri."

UPT Balai Pengembangan Instrumentasi merupakan unit kerja yang sangat potensial di bidangnya, karena dipengaruhi oleh beberapa faktor yang mendukung dan bisa dijadikan peluang untuk meningkatkan kinerja, di antaranya:

1. UPT Balai Pengembangan Instrumentasi terbentuk atas SK Kepala LIPI Nomor 1025/M/2002 pada tanggal 12 Juni 2002 dan merupakan hasil perjuangan selama 15 tahun UPT Balai LIN-LIPI (SK Kepala LIPI Nomor 1333/Kep/D.5/87, tanggal 25 November 1987) bekerja untuk mengembangkan organisasi.
2. Tupoksi UPT yang merupakan mekanisme penyebarluasan hasil-hasil penelitian di bidang kalibrasi dan instrumentasi serta menunjang pelaksanaan program pengembangan MSTQ.
3. Peningkatan daya saing pada era perdagangan bebas mendatang dimana salah satunya adalah menerapkan standar yang dipersyaratkan.
4. Telah terakreditasinya laboratorium kalibrasi UPT Balai LIN-LIPI sejak tahun 1997 sampai sekarang, dengan kecenderungan meningkatnya rentang ukur yang dimiliki setiap tahunnya**.**

2.3. Visi dan Misi

2.3.1. Visi

Dalam rangka melaksanakan agenda pembangunan RPJMN 2020-2024 dan menjalankan amanah sesuai tugas dan fungsinya, pada tahun 2020-2024 Badan Riset dan Inovasi Nasional menetapkan visi sebagai berikut: “Badan Riset dan Inovasi Nasional yang andal, professional, inovatif dan berintegritas dalam pelayanan kepada Presiden dan Wakil Presiden untuk mewujudkan Visi dan Misi Presiden dan Wakil Presiden : Indonesia Maju yang Berdaulat, Mandiri, dan Berkepribadian berlandaskan Gotong Royong” Kemampuan invensi dan inovasi dimaksudkan untuk menghasilkan produk hasil riset yang dilaksanakan dan inovasi yang berpotensi, sedangkan kemampuan iptek dan inovasi dimaknai sebagai keahlian SDM dan lembaga litbang serta perguruan tinggi dalam melaksanakan kegiatan penelitian, pengembangan, pengkajian dan penerapan iptek yang ditunjang oleh pembangunan faktor input (kelembagaan, sumber daya, dan jaringan). Makna daya saing bangsa yaitu kontribusi iptek dan pendidikan tinggi dalam perekonomian yang ditunjukkan oleh keunggulan produk teknologi hasil litbang yang dihasilkan oleh industri/perusahaan yang didukung oleh lembaga litbang (LPNK, LPK, Badan Usaha, dan Perguruan Tinggi) dan SDM yang berkarakter unggul dan berwawasan kebangsaan.

2.3.2 Misi

Adapun Misi Badan Riset dan Inovasi Nasional sebagai upaya untuk mewujudkan visi tersebut yaitu : Badan Riset dan Inovasi Nasional telah melaksanakan Misi Presiden dan Wakil Presiden yakni :

1. Peningkatan Kualitas Manusia Indonesia
2. Peningkatan Struktur Peningkatan Struktur Ekonomi yang Produktif, Mandiri, dan Berdaya Saing
3. Pembangunan yang Merata dan Berkeadilan
4. Mencapai Lingkungan Hidup yang Berkelanjutan
5. Kemajuan Budaya yang Mencerminkan Kepribadian Bangsa
6. Pengelolaan Pemerintahan yang Bersih, Efektif, dan Terpercaya.

Dengan uraian sebagai berikut :

1. Peningkatan Kapabilitas IPTEK, Budaya Riset, dan Penciptaan Inovasi melalui peningkatan Kualitas SDM IPTEK, Penguatan Transformasi Ekonomi, dan Pembangunan Berkelanjutan berlandaskan Budaya Iptek untuk Peningkatan Daya Saing.
2. Peningkatan Pengelolaan Pemerintahan yang Bersih, Efektif, dan Terpercaya.

Misi ini mencakup upaya menjawab permasalahan pembangunan iptek dan pada periode 2020-2024 dalam aspek kebijakan riset dan inovasi, kerjasama pembangunan dan kemitraan, peningkatan penelitian, pengembangan, pengkajian dan penerapan Iptek pada beberapa fokus prioritas riset dan inovasi nasional, serta peningkatan tata kelola pemerintahan yang baik dalam rangka reformasi birokrasi.

2.4. Tugas dan Fungsi

2.4.1. Tugas

Berdasarkan Peraturan Presiden Nomor 78 Tahun 2021 tentang Badan Riset dan Inovasi Nasional, tugas Brin adalah membantu Presiden dalam menyelenggarakan tugas pemerintahan di bidang penelitian, pengembangan, pengkajian, dan penerapan serta invensi dan inovasi, penyelenggaraan ketenaganukliran, dan penyelenggaraan keantariksaan secara nasional yang terintegrasi, dan evaluasi terhadap pelaksanaan tugas dan fungsi BRIDA sesuai dengan ketentuan peraturan perundang-undangan.

2.4.2. Fungsi

BRIN menyelenggarakan fungsinya dalam menjalankan tugas yakni :

1. pelaksanaan penelitian, pengembangan, pengkajian, dan penerapan serta invensi dan inovasi dalam rangka penyusunan rekomendasi perencanaan pembangunan nasional berdasarkan hasil kajian ilmiah dengan berpedoman pada nilai Pancasila;
2. perumusan dan penetapan kebijakan di bidang riset dan inovasi yang meliputi rencana induk pemajuan ilmu pengetahuan dan teknologi, dan peta jalan penelitian, pengembangan, pengkajian, penerapan, serta invensi dan inovasi, penyelenggaraan ketenaganukliran, dan penyelenggaraan keantariksaan;
3. perumusan, penetapan, dan pelaksanaan kebijakan di bidang pembinaan, pengembangan kompetensi, pengembangan profesi, manajemen talenta, dan pengawasan dan pengendalian sumber daya manusia ilmu pengetahuan dan

teknologi, infrastruktur riset dan inovasi, fasilitasi riset dan inovasi, dan pemanfaatan riset dan inovasi;

4. pengintegrasian sistem penyusunan perencanaan, program, anggaran, kelembagaan, dan sumber daya penelitian, pengembangan, pengkajian, dan penerapan, invensi dan inovasi, penyelenggaraan ketenaganukliran, dan penyelenggaraan keantariksaan;
5. penyelenggaraan penelitian, pengembangan, pengkajian, dan penerapan, serta invensi dan inovasi, penyelenggaraan ketenaganukliran, dan penyelenggaraan keantariksaan;
6. pengawasan dan pengendalian penelitian, pengembangan, pengkajian, dan penerapan, serta invensi dan inovasi, penyelenggaraan ketenaganukliran, dan penyelenggaraan keantariksaan secara menyeluruh dan berkelanjutan;
7. pelaksanaan koordinasi pengabdian kepada masyarakat berbasis penelitian, pengembangan, pengkajian, dan penerapan, serta invensi dan inovasi yang dihasilkan oleh kelembagaan ilmu pengetahuan dan teknologi;
8. pelaksanaan pembangunan, pengelolaan, dan pengembangan sistem informasi penelitian, pengembangan, pengkajian, dan penerapan, serta invensi dan inovasi, penyelenggaraan ketenaganukliran, dan penyelenggaraan keantariksaan;
9. pelaksanaan penelitian, pengembangan, invensi, dan inovasi kebijakan yang mengakui, menghormati, mengembangkan dan melestarikan keanekaragaman pengetahuan tradisional, kearifan lokal, sumber daya alam hayati dan nirhayati, serta budaya sebagai bagian dari identitas bangsa;
10. pemberian fasilitas, bimbingan teknis, pembinaan, dan supervisi serta pemantauan dan evaluasi di bidang penelitian, pengembangan, pengkajian, dan penerapan, serta invensi dan inovasi, penyelenggaraan ketenaganukliran, dan penyelenggaraan keantariksaan;
11. pemantauan, pengendalian, dan evaluasi terhadap pelaksanaan tugas dan fungsi BRIDA.
12. pembinaan dan pemberian dukungan administrasi dan teknis kepada seluruh unsur organisasi di lingkungan BRIN;
13. pengawasan atas pelaksanaan tugas di lingkungan BRIN; dan
14. pelaksanaan fungsi lain yang diberikan oleh Presiden.

2.5. Kelompok Penelitian

Kelompok Penelitian Instrumentasi Biomedis mempunyai fokus kegiatan melakukan penelitian, inovasi dan pengembangan sistem instrumentasi medik yang mempunyai kemampuan dalam meningkatkan kualitas, kecepatan dan akurasi pelayanan kesehatan bagi masyarakat Indonesia. Tujuannya adalah menciptakan suatu alat atau metoda dalam kaitannya dengan kesehatan, dengan cara mengukur, merekam dan mengolah data kesehatan dengan menerapkan metode terbaru yang diperoleh dari penelitian yang berkelanjutan.

Ruang lingkup Kegiatan Kelompok Penelitian Biomedical Instrumentation diantaranya:

1. rancang bangun prototipe eksoskeleton *lower limb* 2:1;
2. pengembangan Telehealth dan Telemedicine;
3. pengembangan metoda dan alat kesehatan dengan cara mengukur, merekam, dan mengolah data kesehatan dengan menerapkan metode terbaru yang diperoleh dari penelitian yang berkelanjutan.

2.6. Lokasi/Unit Pelaksanaan Kerja

Unit pelaksanaan kerja praktik berlokasi di Balai Pengembangan Instrumentasi (BPI), Organisasi Riset Ilmu Pengetahuan dan Teknologi (OR IPT), Badan Riset dan Inovasi Nasional (BRIN), Lantai 3 dan Lantai 4 pada Gedung 80, lantai 3 pada Gedung 10, dan Lantai 2 pada Tower 2, Jalan Cisitu, Sangkuriang, Bandung, Jawa Barat.

BAB III
TINJAUAN PUSTAKA

3.1. *Lower Limb Exoskeleton*

Robot rehabilitasi eksoskeleton ekstremitas bawah, atau *lower limb exoskeleton* yang merupakan kelas utama dari robot rehabilitasi, berhubungan dengan tubuh manusia dengan cara dapat dipakai dan dapat mengontrol pergerakan semua sendi dalam proses training rehab. Robot rehabilitasi *lower limb* bermanfaat dan diperlukan secara signifikan karena dapat mengurangi beban terapis, mendeteksi data selama masa training rehabilitasi, dan membantu evaluasi kuantitatif pemulihan dengan cara yang dapat dikontrol dan diulang. Teknologi robot rehabilitasi *lower limb* menggabungkan penginderaan (*sensing*), kontrol, informasi dan ilmu komputer untuk menghasilkan alat mekanis yang bisa dipakai (*wearable*).

Menurut pengaplikasiannya, robot ini dapat terbagi menjadi dua macam, yakni *treadmill-based* atau berbasis alat treadmill dan aplikasi di atas tanah (*overground*); yang terdiri dari bagian eksoskeleton, sistem penyokong beban tubuh (*body weight system (BWS)*) (Shi, D., et al, 2019).

3.2. Motor Servo SPT5435LV (Aktuator)

Perangkat listrik yang berfungsi untuk mendorong atau memutar objek yang membutuhkan kontrol dengan presisi tinggi dalam hal posisi sudut, akselerasi, dan kecepatan disebut dengan Motor Servo. Biasanya, motor servo digunakan pada mesin-mesin industri karena memiliki sistem closed loop yang berupa encoder. Encoder guna untuk umpan balik posisi yang dapat mengontrol target posisi motor, keluaran torsi, kecepatan rotasi.

Jenis motor yang digunakan untuk penelitian ini yaitu motor servo jenis SPT5435LV. Tampilan fisik dari motor servo SPT5435LV.

Gambar 3.1. Motor Servo SPT5435LV

Servo ini memiliki torsi besar 35 Nm dengan konektor tiga pin standar yang dimana menggabungkan sirkuit kontrol kecepatan internal yang memungkinkan untuk dapat memindahkannya ke arah yang diinginkan dan mengubah kecepatan sesuai dengan yang dibutuhkan. Motor servo ini memiliki roda gigi logam untuk dapat memberikan distribusi gaya (torsi) yang baik. Selain itu, motor ini juga memiliki bantalan bola ganda dan kabel konektor dengan tiga terminal yaitu VCC (Power), GND (Ground), dan Signal (sinyal kontrol). Biasanya, motor ini digunakan pada robot bipedal, mobil R/C, helikopter sayap tetap 50CC/90CC dan 26CC/50CC.

Motor ini memang dirancang untuk dapat menghasilkan torsi yang besar, sehingga tidak direkomendasikan untuk aplikasi sistem yang membutuhkan presisi tinggi, sehingga menjadi pilihan terbaik untuk aplikasi tenaga. Karena memiliki sudut 180 derajat dan bekerja dibawah lebar pulsa PWM 500-2500 atau dikendalikan oleh mainboard arduino, motor ini jika menggunakan pemancar jarak jauh untuk dapat mengontrolnya maka hanya akan mendapatkan setengah sudutnya saja. hal ini dikarenakan peralatan yang di dapat hanya menghasilkan lebar pulsa 1000-2000 PWM. Adapun spesifikasi teknis dari motor servo SPT5435LV antara lain :

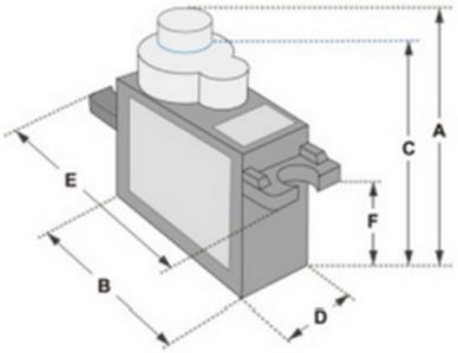
- Brand : SPT Servo
- Neutral point : 1500us
- PWM voltage : 3.3V-5.0V
- Feedback angle : no
- Cycle : 20ms
- Remote control angle: 90/135/150
- Standby current : 100mAh
- Stall current : 3.5 A
- Output gear : Futaba 25T
- Housing material : Aluminium half housing
- Wire Connector : 260MM
- Operating speed length : 4.8 V
- Voltage range : 4.8V
- Signal frequency : 330 Hz
- Yes/ no lock : lock

- Product reference : SPT5435LV

3.3. Motor Servo SG90

Sebuah motor dengan sistem closed feedback dimana posisi dari motor akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo disebut sebagai motor servo. Biasanya motor servo digunakan untuk aktuator yang membutuhkan posisi putaran motor yang presisi dan sudut yang ada pada motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor. Motor servo ini dapat dimodifikasi bergerak secara kontinu. Adapun komponen potensiometer pada motor servo SG90 berfungsi untuk menentukan batas maksimum putar sumbu (axis) motor servo.

Motor servo SG90 merupakan motor kecil dan ringan dengan output daya yang tinggi. Motor servo dapat merotasi 180 derajat (90 derajat ke tiap arah), dan bekerja seperti motor standard pada umumnya tetapi lebih kecil. Motor ini dapat dikendalikan oleh kode servo, perangkat keras, atau library manapun. Dianggap motor yang mudah digunakan untuk pemula yang ingin membuat peralatan dengan pergerakan tanpa feedback dan gear box, dikarenakan dapat termuat dalam tempat-tempat kecil. Motor SG90 terdiri dari 3 lengan dan perangkat keras. Berikut merupakan skema peralatan dengan detail dimensi dan spesifikasi.



Gambar 3.2. Dimensi motor servo

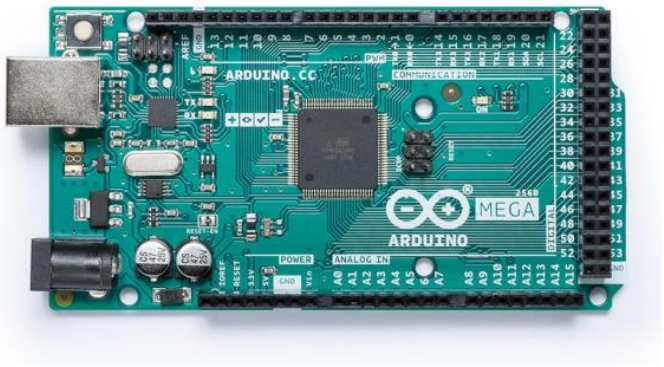
Dimensi dan spesifikasi
A (mm) : 32
B (mm) : 23
C (mm) : 28.5

D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Kecepatan (detik) : 0.1
Torsi (kg-cm) : 2.5
Berat (g) : 14.7
Voltase : 4.8 - 6

Tabel 3.1. Ukuran dimensi dan spesifikasi motor servo SG90

Posisi “0” pada motor servo menunjukkan posisi pulsa 1.5 ms di bagian tengah, “90” berada pada ~2 ms bagian paling kanan, dan “-90” ke arah paling kiri. Pada konfigurasi kabel, warna jingga melambangkan PWM (pulse width modulation), warna merah melambangkan Vcc, dan warna coklat merupakan Ground (Tower Pro, 2021).

3.4. Mikrokontroler (Arduino Mega 2560)



Gambar 3.3. Arduino Mega 2560

Arduino Mega 2560 merupakan board mikrokontroler berbasis ATmega2560. Terdapat 54 pin input/output digital (15 dapat digunakan untuk output PWM), 16 input analog, 1 UART (port serial hardware), sebuah osilator kristal 16 MHz, koneksi USB, *power jack*, sebauh header ICSP, dan tombol *reset*. Arduino Mega 2560 ini dibuat dalam aplikasi dan projek yang memerlukan jumlah pin input dan output yang banyak serta *use cases* yang membutuhkan daya processing yang tinggi. Beberapa contoh aplikasi penggunaan Arduino Mega 2560 diantaranya adalah untuk robotik, yang kompatibel dengan *motor controller shield* sehingga dapat mengendalikan beberapa motor sekaligus. Jumlah pin I/O yang banyak dapat juga mengakomodir sejumlah sensor robotik. Selain

itu, Arduino Mega 2560 mempunyai daya untuk memproses algoritma rumit yang diperlukan dalam 3D printing. Ia juga kompatibel dengan WiFi *shields* sehingga mendukung fitur-fitur *wireless* pada aplikasi 3D printing maupun robotik (Arduino®, 2023).

3.5. Baterai Sony VTC6A 9V

Gambar 3.4. Baterai Sony VTC6A

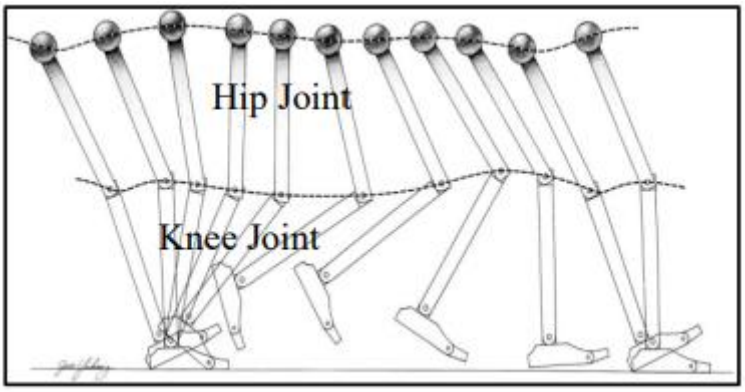
Baterai VTC6A merupakan baterai yang memiliki ampere lebih tinggi sehingga dapat lebih stabil untuk bermain watt besar secara terus-menerus. Selain itu, VTC6A dapat menahan turunnya voltase lebih stabil jika dibandingkan dengan versi sebelumnya. Dengan kapasitasnya dan kombinasi amperenya yang sangat tinggi baterai VTC6A menjadi baterai yang sangat cocok untuk digunakan oleh sejuta umat bahkan untuk high-wattage vape (minimal dua baterai) sekalipun.

3.6. Grafik Gait Cycle

3.6.1. Siklus Gait

Saat tubuh seseorang bergerak maju, satu bagian tubuh bekerja sebagai sumber penyokong selagi bagian tubuh lain maju ke tempat sokong berikutnya, kemudian kedua bagian tubuh akan bertukar peran. Serangkaian pergerakan ini diulang oleh tiap bagian tubuh dengan waktu yang timbal-balik hingga orang tersebut mencapai tempat tujuannya. Satu sekuens dari fungsi ini dinamakan dengan *gait cycle* atau siklus gait (Kharb, Ashutosh, et al. 2011). Dengan kata lain, siklus gait didefinisikan sebagai kombinasi pergerakan rotasional dan berirama dari bagian tubuh untuk memberikan tubuh keseimbangan dan kesinambungan gerak. Kegiatan berjalan umumnya terbagi menjadi dua tahapan, yakni tahap stance dan tahap swing. Bagian yang menyokong beban tubuh

pada tiap siklus gait merupakan tahap stance, sementara tahap swing dimulai ketika kontak antara kaki dan tanah terlepas. Adapun *trajectory* atau lintasan dari sudut-sudut ekstremitas bawah (pinggul dan lutut) saat terjadinya siklus gait seperti yang digambarkan di bawah ini.

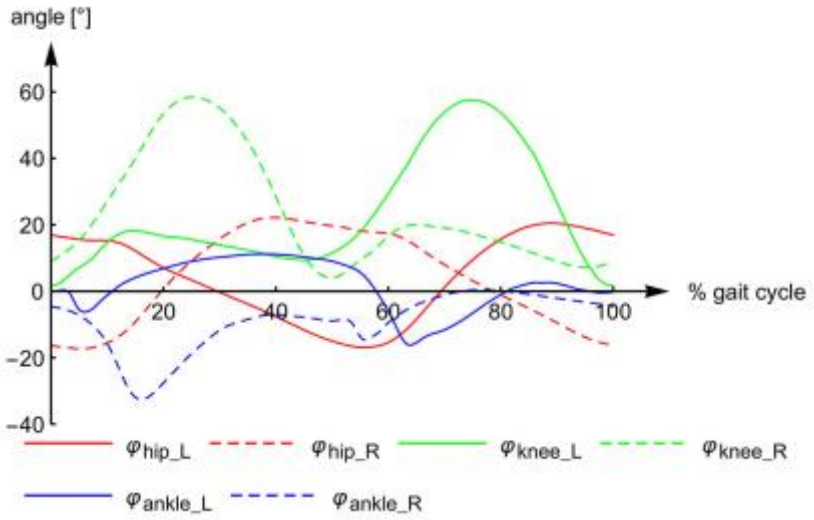


Gambar 3.5. Pergerakan ekstremitas bawah selama satu stride

Mayoritas pergerakan dari alat prostetik ekstremitas bawah dalam pergerakan berjalan berfokus pada bidang sagital saja (pergerakan maju dan mundur) (Haider, Saif M. J., et al. 2021).

3.6.2. Lintasan Pergerakan Gait

Lintasan atau *trajectory* didapatkan dari pengukuran data kinematis sebuah ekstremitas dalam hal lokasi dan orientasi, seperti fleksi/ekstensi, kecepatan sudut, serta percepatan dari paha dan kaki. Pada praktik kerja ini, penulis menggunakan referensi *gait trajectory* dari Jarzyna, Olga, et al (2020). Jarzyna et al berfokus pada gait reguler (mengabaikan trajectory gait saat inisiasi dan terminasi orang berjalan), dan mendapatkan pola pergerakan ekstremitas bawah dari rata-rata time-history posisi sudut pada beberapa perekaman siklus gait individu, periode waktu antara dua heel strike, yang mereka lakukan. Gambar di bawah merupakan grafik dari pola pergerakan yang didapatkan dari eksperimen pengamatan oleh Jarzyna et al.



Gambar 3.6. Sudut-sudut sendi pada bidang sagital selama satu siklus gait - hasil eksperimen Jarzyna et al (2020)

Kurva garis mewakili pola pergerakan untuk ekstremitas bawah bagian kiri, sementara garis putus-putus mewakili ekstremitas bawah tubuh bagian kanan. Adapun bagian kanan dan kiri menunjukkan pola yang mirip tetapi dilakukan shifting atau pergeseran fasa sebesar 50%, sehingga bisa dianggap simetris.

Dari grafik hasil eksperimen ini, dibuatkan perumusan pendekatan pola pergerakan tiap sendi (pinggul, lutut, dan pergelangan kaki) sebagai berikut, dengan grafik pendekatan ini adalah pada Gambar 3.6.:

a. Rumus lintasan sendi *hip*

$$S_{1L}(t) = \phi_1(t), S_{1R}(t) = S_1(t - 0.5T),$$

$$S_1(t) = \phi\big(mod(t - T_1, T)\big),$$

$$\phi_1(t) = \begin{cases} \phi_{10} - (\phi_{10} - \phi_{11}) \sin^2\left(\dfrac{\pi}{2t_{11}} t\right) & if\ t \in < 0, t_{11}), \\ \phi_{11} + (\phi_{12} - \phi_{11}) \sin^2\left[\dfrac{\pi}{2(t_{12} - t_{11})}(t - t_{11})\right] & if\ t \in < t_{11}, t_{12}), \\ \phi_{12} - (\phi_{12} - \phi_{10}) \sin^2\left[\dfrac{\pi}{2(T - t_{12})}(t - t_{12})\right] & if\ t \in < t_{12}, T >. \end{cases}$$

Dengan nilai t_{11} = 0.554T, t_{12} = 0.894T, ϕ_{10} = 18.32, ϕ_{11} = -17.09, dan ϕ_{12} = 21.31.

b. Rumus lintasan sendi *knee*

$$S_{2L}(t) = \phi_2(t), S_{2R}(t) = S_2(t - 0.5T),$$

$$S_2(t) = \phi\big(mod(t - T_2, T)\big),$$

$$\phi_2(t) = \begin{cases} \phi_{20} - (\phi_{21} - \phi_{20}) \sin^2\left(\frac{\pi}{2t_{21}} t\right) & if\ t \in < 0, t_{21}), \\ \phi_{21} + (\phi_{21} - \phi_{22}) \sin^2\left[\frac{\pi}{2(t_{22} - t_{21})}(t - t_{21})\right] & if\ t \in < t_{21}, t_{22}), \\ \phi_{22} - (\phi_{23} - \phi_{22}) \sin^2\left[\frac{\pi}{2(t_{23} - t_{22})}(t - t_{22})\right] & if\ t \in < t_{22}, t_{23}), \\ \phi_{23} - (\phi_{23} - \phi_{20}) \sin^2\left[\frac{\pi}{2(T - t_{23})}(t - t_{23})\right] & if\ t \in t_{23}, T >. \end{cases}$$

Dengan nilai t_{21} = 0.148T, t_{22} = 0.451T, t_{23} = 0.751T, ϕ_{20} = 2.86, ϕ_{21} = -19.07, ϕ_{22} = 8.30, dan ϕ_{23} = 58.01.

c. Rumus lintasan sendi *ankle*

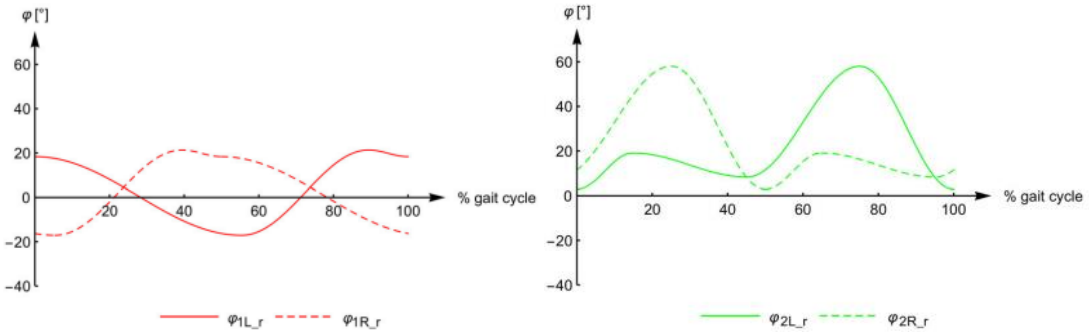
$$S_{3L}(t) = S_{1L}(t) - S_{2L}(t) - \beta(t) + S_3(t)$$

$$S_{3R}(t) = S_{1R}(t) - S_{2R}(t) - \beta(t) + S_3(t - 0.5T)$$

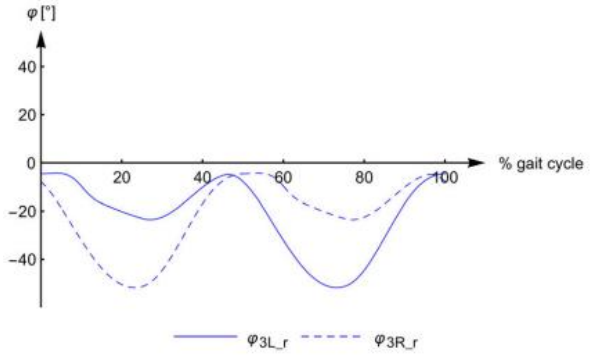
$$S_3(t) = \phi\big(mod(t - T_3, T)\big),$$

$$\phi_3(t) = \begin{cases} \phi_{30} - (0 - \phi_{30}) \sin^2\left(\frac{\pi}{2t_{31}} t\right) & if\ t \in < 0, t_{31}), \\ 0 & if\ t \in < t_{31}, t_{32}), \\ 0 + (\phi_{31} - 0) \sin^2\left[\frac{\pi}{2(t_{33} - t_{32})}(t - t_{32})\right] & if\ t \in < t_{32}, t_{33}), \\ \phi_{31} - (\phi_{31} - \phi_{30}) \sin^2\left[\frac{\pi}{2(T - t_{33})}(t - t_{33})\right] & if\ t \in t_{33}, T >. \end{cases}$$

Dengan nilai t_{31} = 0.10T, t_{32} = 0.25T, t_{33} = 0.5T, ϕ_{30} = -10, ϕ_{21} = -19.07, dan ϕ_{31} = 30. Semua nilai T_1, T_2, T_3 sama, yaitu nol.



(a) (b)



(c)

Gambar 3.7. Lintasan aproksimasi sendi *hip* (a), *knee* (b), dan *ankle* (c)

3.7. Software Proteus

Software yang digunakan untuk mendesain PCB yang dilengkapi dengan simulasi pspice pada level skematik sebelum rangkaian skematik diupgrade ke PCB yaitu Proteus. Software ini banyak sekali kegunaannya yaitu dapat digunakan untuk mendesain rangkaian mikrokontroler, sebagai tempat untuk belajar elektronik dasar-dasar elektronika sampai pada aplikasi mikrokontroler, dan lainnya. Untuk dapat memudahkan dalam pembuatan layout PCB dari skematik yang dibuat, Proteus dapat dikombinasikan dengan program ISIS dan ARES. PCB layout merupakan tampilan kedua yang ada di dalam perangkat lunak. Biasanya digunakan untuk tata letak komponen dan juga penempatan terminal yang divisualisasikan secara 2 dimensi, mengatur ukuran dan bentuk dari PCB.

Adapun fitur-fitur yang ada pada software Proteus yaitu :

1. dapat mensimulasikan rangkaian digital ataupun rangkaian analog;
2. mendukung simbol-simbol alat ukur atau alat tes komponen elektronika seperti Voltmeter, Amperemeter, atau Osiloskop;
3. memiliki desain peripheral yang interactive seperti LED, tampilan LCD, atau RS232;
4. mendukung tampilan berbagai jenis analisis secara grafis seperti transient, frekuensi, noise, distorsi, AC, dan DC;
5. mendukung simulasi berbagai jenis IC seperti ATMega328;
6. mendukung open architecture sehingga kita dapat mengupload program seperti C++ atau arduino untuk keperluan simulasi rangkaian; serta

7. mendukung pembuatan PCB yang di update secara langsung dari program ISIS ke program pembuat PCB-ARES.

BAB IV
METODE PELAKSANAAN

4.1. Waktu dan Tempat Pelaksanaan

Praktik Kerja Lapangan dilakukan selama 4 minggu, dimulai pada tanggal 9 Januari - 3 Februari 2023 dan dilaksanakan di Balai Pengembangan Instrumentasi BRIN (Badan Riset dan Inovasi Nasional), Bandung. Adapun Praktik Kerja Lapangan dilaksanakan di bawah bimbingan Ibu Dwi Esti Kusumandari, M. T.

Berikut merupakan jadwal PKL yang telah dilakukan selama 4 minggu:

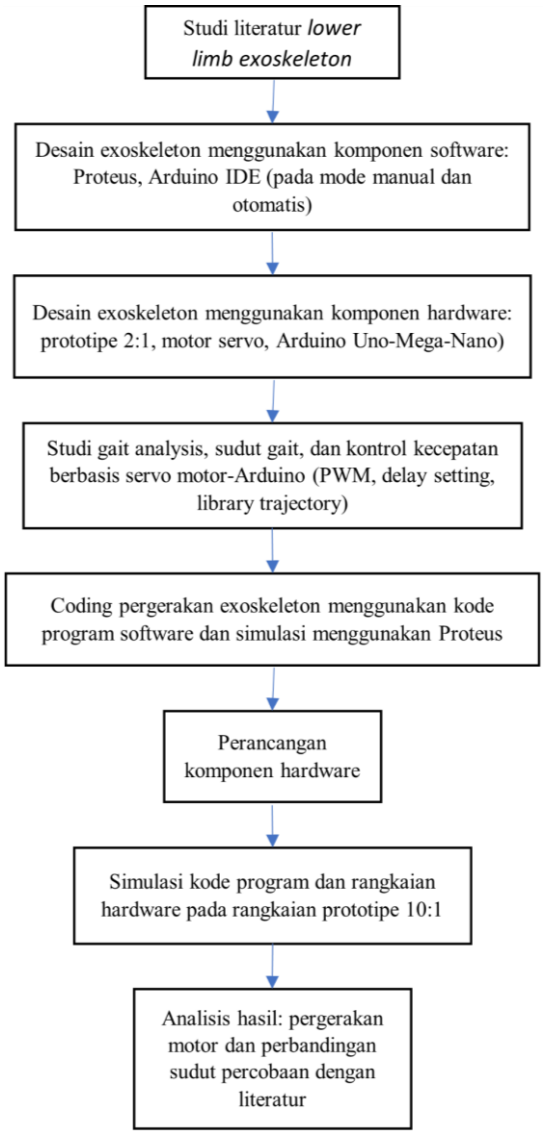
No	Kegiatan	Tanggal Pelaksanaan (9 Januari - 3 Februari 2023)																		
		9	10	11	12	13	16	17	18	19	20	24	25	26	27	30	31	1	2	3
1	Pengenalan institusi, pengurusan administrasi, dan perumusan sasaran kerja	■																		
2	Studi literatur eksoskeleton dan prototype 2:1		■	■	■	■														
3	Simulasi mode manual dan otomatis						■	■	■	■										
4	Presentasi Progress 1										■									
5	Modifikasi motor servo											■	■							
6	Perumusan coding software untuk kontrol kecepatan beserta simulasi di Proteus													■	■	■	■	■		
7	Pengaplikasian kontrol trajectory pada prototype dan perbandingan hasil																		■	■
8	Presentasi akhir																			

4.2. Alat dan Bahan

1. Software Proteus
2. Software Arduino IDE
3. Motor Servo SPT5435LV
4. Motor Servo SG90
5. Mikrokontroler Arduino Mega 2560
6. Baterai Sony VTC6A 9V
7. Kabel Data
8. Kabel Jumper

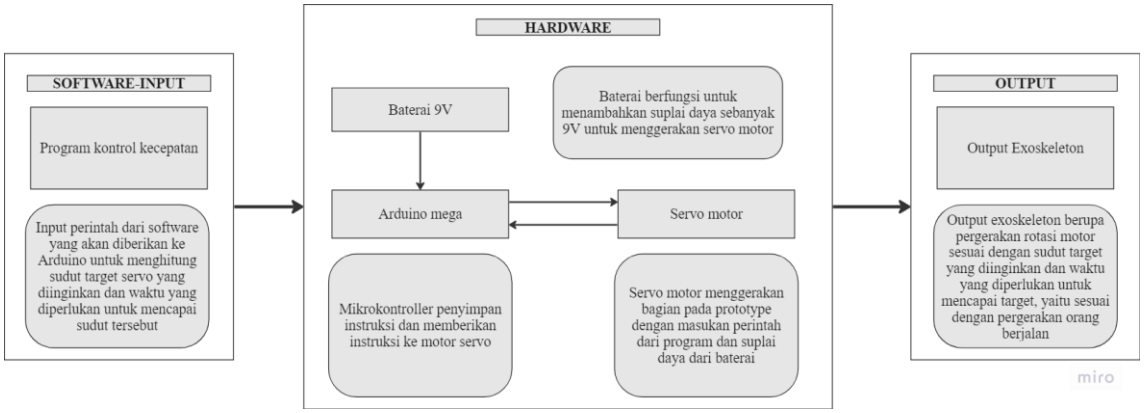
4.3. Tahap Pelaksanaan

Adapun tahapan dalam pelaksanaan praktik kerja ini adalah sebagai berikut:



4.4. Diagram Blok Sistem

Berikut merupakan konsep kerja dari hasil menjalankan kontrol kecepatan prototipe eksoskeleton 2:1 dengan library trajectory.



BAB V
HASIL DAN PEMBAHASAN

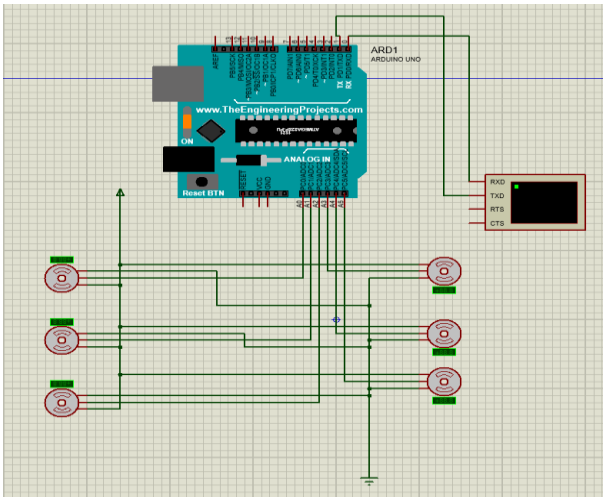
Penelitian pada praktik kerja ini bertujuan untuk mengontrol pergerakan dan kecepatan sebuah aktuator pada sendi ekstremitas bawah (pinggul, lutut, dan pergelangan kaki), agar sesuai dengan pola pergerakan referensi, yakni siklus gait orang sehat yang berjalan secara normal. Penelitian dilakukan dengan membuat pemrograman pergerakan yang kemudian diaplikasikan ke motor servo prototipe 10:1 exoskeleton *lower limb*. Prototipe terdiri dari 6 motor servo yang mewakili 6 sendi yang dihubungkan oleh lengan-lengan hasil 3D-printing milik BRIN dan pembimbing institusi. Kontrol pergerakan yang dibuat diharapkan dapat menghasilkan pergerakan motor dan *limb* prototipe yang lebih halus. Program disimulasikan pada software Proteus terlebih dahulu kemudian dibuatkan grafik pola pergerakan untuk dibandingkan dengan acuan jurnal referensi.

5.1. Perancangan Sistem

5.1.1. Hardware

Hardware yang digunakan pada penelitian ini diantaranya adalah prototipe eksoskeleton *lower limb* yang terdiri dari enam motor servo (motor SPT pada pinggul dan lutut dan motor SG90 pada pergelangan kaki) milik institusi; Arduino ATMega328P sebagai mikrokontroler, bread board, kabel jumper, baterai 9V sebagai penambah daya, serta kabel data.

5.1.1.1. Simulasi Rangkaian Otomatis di Proteus



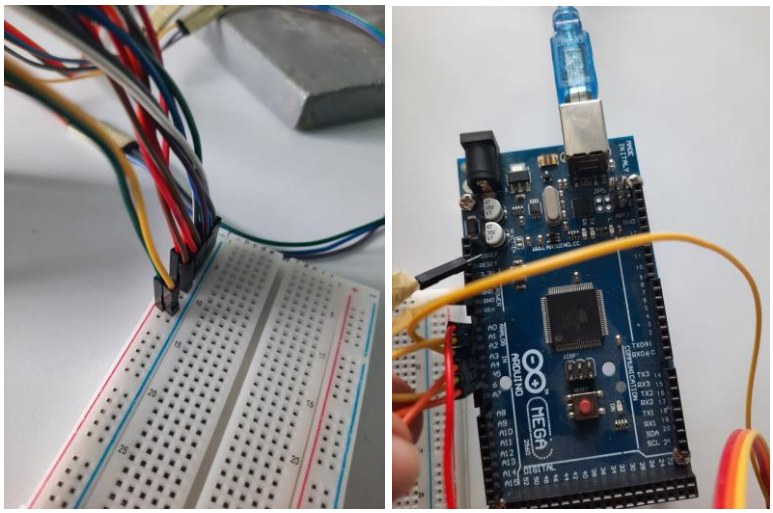
Gambar 5.1. Simulasi rangkaian *exoskeleton lower limb*

Menjalankan simulasi penggerak eksoskeleton *lower limb* di software Proteus menjadi salah satu aktivitas dalam praktik kerja lapangan kali ini sebagai tahap percobaan sebelum diaplikasikan ke prototipe. Pergerakan komponen motor servo dilakukan dengan memberikan program coding penggerak pada software Arduino IDE sehingga target (robot) dapat bergerak seperti orang berjalan dan menghasilkan lintasan sudut yang sesuai dengan referensi. Adapun pin yang digunakan untuk menghubungkan keenam motor servo pada komponen Arduino diantaranya adalah pin A0, A1, A2, A3, A4, dan A5. Kemudian, adanya penggunaan komponen Virtual Terminal yang dihubungkan ke Arduino pada pin RXD dan TXD guna melihat data-data sudut yang diprint menggunakan fungsi Serial.print.

Setelah semua pin telah terhubung di masing-masing komponen, simulasi dapat dijalankan. Tahap selanjutnya adalah mengaplikasikan program coding ke rangkaian hardware atau prototipe 10:1 eksoskeleton *lower limb*.

5.1.1.2. Kontrol Servo Motor

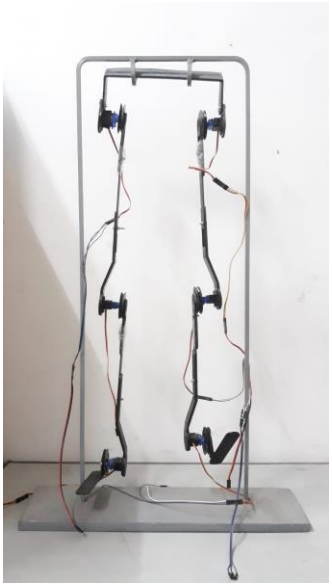
Program penggerak di-*compile* kemudian di-*upload* ke Arduino ATMega328P menggunakan kabel data. Setelah berhasil di-*upload*, kabel data dicabut. Masing-masing motor servo memiliki tiga kabel, yakni Vcc, Sinyal, dan Ground. Kabel sinyal dihubungkan langsung ke pin Arduino yang sesuai dengan sendinya, kemudian Vcc dan Ground serta baterai 9V yang digunakan untuk memberi daya pada keenam motor dihubungkan langsung pada PCB.



(a) (b)

Gambar 5.2. Perkabelan *hardware* PCB (a) dan Arduino Mega 2560 (b)

Beberapa penyesuaian dilakukan untuk dapat memastikan tiap-tiap sendi bergerak dengan semestinya, yakni ke arah yang benar dan juga dengan jangkauan gerakan yang sesuai.



Gambar 5.3. Prototipe *Lower Limb Exoskeleton*

5.1.2. Software

5.1.2.1. Library Trajectory

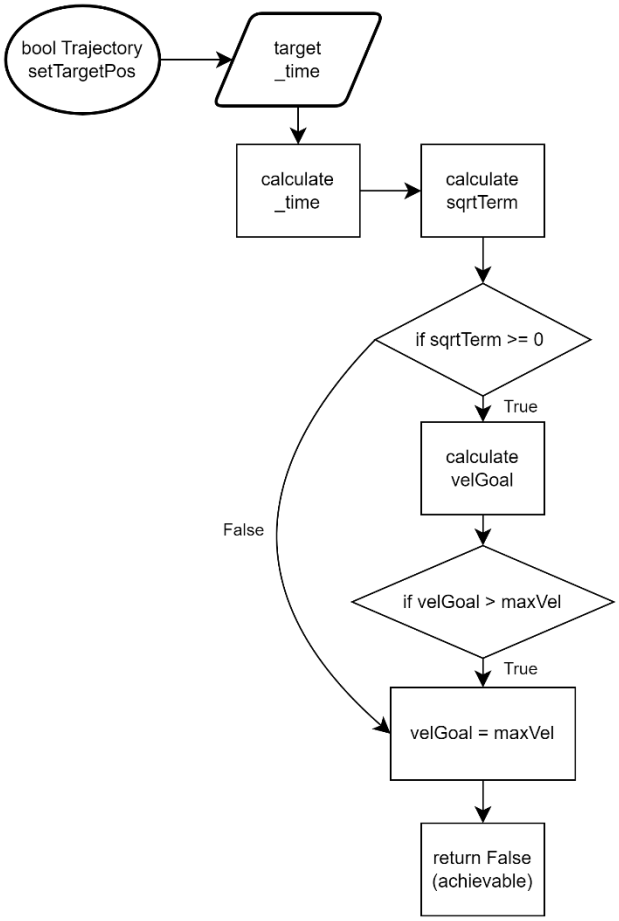
Library ‘Servo Trajectory’ merupakan salah satu library Arduino yang digunakan dalam kontrol pergerakan motor servo proyek robotik. Class pada library ini dapat digunakan untuk mengendali pergerakan dan kecepatan motor servo atau motor DC dengan encoder. Cara kerjanya sendiri adalah ketika diberikan suatu posisi (sudut) target tertentu, motor akan dipercepat dengan laju yang konstan hingga mencapai kecepatan maksimal. Saat motor mendekati target, motor akan diinstruksi untuk melambat sedikit demi sedikit hingga berhenti mencapai sudut target.

Di dalam software Arduino IDE terdapat library bawaan ‘Servo’ yang dapat digunakan untuk menggerakkan motor ke posisi/sudut tertentu yang diinginkan secara sederhana. Adapun instruksi pergerakan dari library Servo ini tidak mempertimbangkan kecepatan motor servo, sehingga motor bergerak ke sudut target dengan kecepatan maksimalnya. Terdapat pula library ‘Sweep’ yang mengatasi masalah ini dengan memecah pergerakan menjadi pergerakan untuk tiap satu sudut, sehingga pergerakan menuju sudut target lebih lambat dan pergerakan antar sudut dapat diatur dengan

menetapkan waktu delay. Library ‘Servo Trajectory’ yang digunakan pada penelitian ini memodifikasi pergerakan sehingga terdapat percepatan dan juga perlambatan (*acceleration and deceleration*) selama perpindahan antar sudut target, dengan hasil berupa pergerakan motor servo yang lebih halus dan motor tidak berhenti secara mendadak. Sehingga dalam kontrol pergerakan motor servo prototipe *lower limb* ini library ‘Servo Trajectory’ cocok untuk digunakan.

Fitur utama dari library ‘Servo Trajectory’ ini adalah pengguna dapat menentukan sudut target motor servo yang diinginkan serta kecepatan ataupun waktu yang digunakan untuk mencapai target sudut tersebut. Jika waktu yang diperlukan untuk mencapai suatu sudut ditentukan, maka perhitungan kecepatan dilakukan oleh program.

```
bool Trajectory::setTargetPos(float _targetPos, float _time) {
        target = _targetPos;

        // Take into account the cut-off threshold
        _time = _time + sqrt(2 * threshold / dec);

        // Calculate the velocity required to reach the goal within the specified time
        float sqrtTerm = (_time * _time) - 2 * abs(_targetPos - curPos) * (1/acc + 1/dec);

        // If the term is a positive number, then a solution exists
        if (sqrtTerm >= 0) {
                velGoal = (_time - sqrt(sqrtTerm)) / (1/acc + 1/dec);
                if (velGoal > maxVel) velGoal = maxVel;
        } else {
                velGoal = maxVel;
        }
        type = 0;
        noTasks = false;
        if (velGoal == maxVel) return false;
        else return true;
}

// Set a new target velocity
void Trajectory::setTargetVel(float _targetVel) {
        if (_targetVel > maxVel) target = maxVel;
        else if (_targetVel < -maxVel) target = -maxVel;
        else target = _targetVel;
        type = 1;
        noTasks = false;
}
```



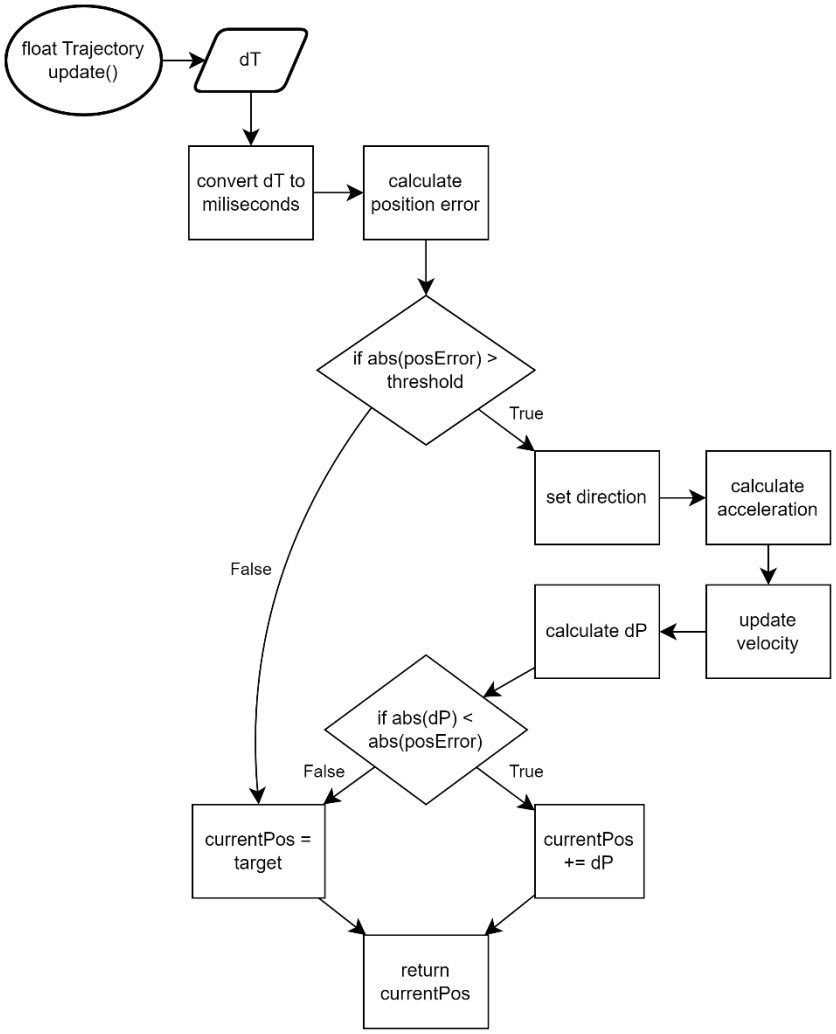
Gambar 5.4. *Flowchart* fungsi setTargetPos pada class Trajectory

Dengan _targetPos merupakan nilai posisi target baru, _time merupakan waktu dalam detik hingga sistem harus mencapai target posisi, _targetVel merupakan nilai kecepatan yang ingin digunakan pada sistem, dan threshold merupakan nilai cutoff threshold atau batas eror posisi yang mana di bawahnya tidak akan ada pergerakan. Threshold diberi nilai default sebesar 0.1.

Selanjutnya untuk mendapatkan pergerakan yang halus, yakni adanya percepatan dan perlambatan gerakan motor antar perpindahan dari satu sudut ke sudut yang lain, maka pada library ini terdapat perhitungan untuk mendapatkan kecepatan linear dengan *parabolic blends*, baik apabila waktu maupun kecepatannya di-*setting*.

```
float Trajectory::update(float dT) {

    // Convert ms to seconds
    dT /= 1000.0;

    // Position Control (Linear velocity with parabolic blends)
```

```
        float posError = target - curPos;

                // If position error is above the threshold
                if (abs(posError) > threshold) {

                        // Determine motion direction
                        bool dir = true;
                        if (posError < 0) dir = false;

                        // Determine whether to accelerate or decelerate
                        float acceleration = acc;
                        if ((curVel * curVel / (2 * dec)) >= abs(posError)) acceleration
= -dec;

                        // Update the current velocity
                        if (dir) curVel += acceleration * dT;
                        else curVel -= acceleration * dT;

                        // Limit Velocity
                        if (curVel > velGoal) curVel = velGoal;
                        else if (curVel < -velGoal) curVel = -velGoal;

                        float dP = curVel * dT;

                        // Prevent overshooting/jittering around target
                        if (abs(dP) < abs(posError)) curPos += dP;
                        else curPos = target;

                        //if (abs(curVel) != velGoal) Serial.println(acceleration * (-1 +
2*dir));

                } else {
                        curVel = 0;
                        curPos = target;
                        noTasks = true;
                }
```



Gambar 5.5. *Flowchart* fungsi update() pada class Trajectory

Coding lengkap library ‘Servo Trajectory’ dapat dilihat pada bagian lampiran.

5.1.2.2. Coding Program untuk Enam Motor Servo

Sudut-sudut lintasan dari referensi didapat dengan mensimulasikan perhitungan rumus di pemrograman Python. Nilai sudut di-print untuk masing-masing sendi. Satu nilai didapatkan untuk satu persentase gait, sehingga didapatkan seratus nilai sudut. Besar sudut menyesuaikan posisi motor servo saat dipasang pada prototipe, sehingga untuk masing-masing sendi berbeda jangkauan geraknya. Namun, untuk membandingkan sudut hasil pergerakan dengan sudut referensi sudut-sudut diberi titik patokan 90^{o}. Tabel sudut-sudut lintasan yang didapatkan dapat dilihat pada Tabel di halaman lampiran.

Dari nilai-nilai tersebut diamati nilai berapa saja yang menjadi titik tolak pada naik atau turunnya grafik, yang kemudian akan menjadi sudut target pada program menggunakan class pada library trajectory.

Pada coding Arduino IDE, program code terbagi menjadi tiga bagian, yakni pemanggilan library dan definisi variabel, definisi fungsi dan void setup, serta void loop. Dua library digunakan dalam percobaan ini yakni ‘trajectory.h’ dan ‘Servo.h’. Enam class trajectory dipanggil dan tiap lintasan motor servo diberi nama; pada percobaan ini penulis menggunakan singkatan H untuk *hip*, K untuk *knee*, A untuk *ankle*, dan huruf berikutnya L/R mewakili *left* atau *right*. Pengaturan default untuk class trajectory ini adalah setiap servo memiliki *max velocity* atau maksimum kecepatan 60, percepatan 40, dan perlambatan 34; tetapi pada percobaan ini penulis menggunakan kecepatan maksimum sebesar 30. Class Servo digunakan untuk memberi nama pada masing-masing motor servo. Variabel global didefinisikan pula, dengan pengaturan controller akan diupdate dengan laju 100 Hz. Diberikan nilai integer sebesar nol untuk moveNumber__ yang merupakan nilai case mula-mula, serta nilai float untuk sudut-sudut akhir yang dihasilkan.

```
#include <trajectory.h>
#include <Servo.h>

Trajectory servoTrajectoryHL(30);
Trajectory servoTrajectoryKL(30);
Trajectory servoTrajectoryAL(30);
Trajectory servoTrajectoryHR(30);
Trajectory servoTrajectoryKR(30);
Trajectory servoTrajectoryAR(30);

Servo hipR;
Servo kneeR;
Servo ankleR;
Servo hipL;
Servo kneeL;
Servo ankleL;

// --- Define global variables ---
// The controller will be updated at a rate of 100Hz
#define UPDATE_FREQUENCY 100
#define UPDATE_TIME (1000 / UPDATE_FREQUENCY)
unsigned long updateTimer = 0;

int moveNumberHipL = 0;
int moveNumberKneeL = 0;
```

```
int moveNumberAnkleL = 0;
int moveNumberHipR = 0;
int moveNumberKneeR = 0;
int moveNumberAnkleR = 0;

float sudutHipL, sudutHipR, sudutKneeL, sudutKneeR, sudutAnkleL, sudutAnkleR;
```

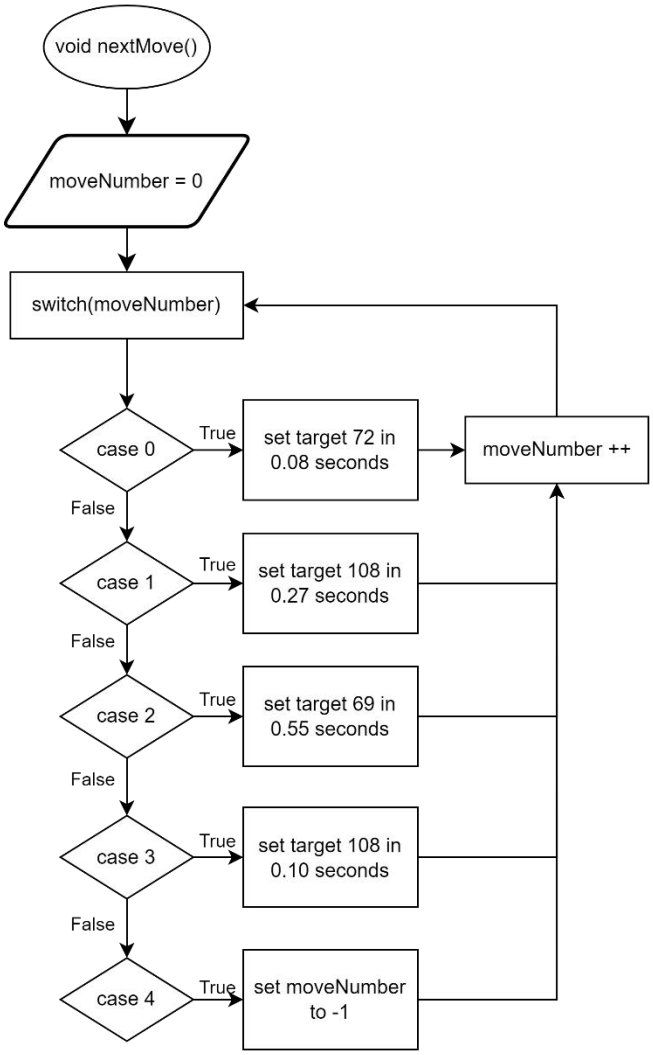
Pada void setup, Serial.begin(9600) akan menginisiasi Serial printer pada software Arduino IDE dengan *baud rate* atau kecepatan transmisi data 9600 bit per detik. Masing-masing motor servo dihubungkan pada pin A0-A5 dan trajectory tiap motor diberikan posisi awal sesuai dengan penempatan motor pada prototipe.

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("Starting program");

  hipR.attach(A0);    // attaches hip R Servo on pin A0 to the servo object
  kneeR.attach(A1);   // attaches knee R Servo on pin A1 to the servo object
  ankleR.attach(A2);  // attaches ankle R Servo on pin A2 to the servo object
  hipL.attach(A3);    // attaches hip L Servo on pin A3 to the servo object
  kneeL.attach(A4);   // attaches knee L Servo on pin A4 to the servo object
  ankleL.attach(A5);  // attaches ankle L Servo on pin A5 to the servo object

  servoTrajectoryHL.reset(90);
  servoTrajectoryKL.reset(90);
  servoTrajectoryKL.reset(90);
  servoTrajectoryHR.reset(0);
  servoTrajectoryKR.reset(0);
  servoTrajectoryKR.reset(90);
};
```

Fungsi void nextMove() berisi beberapa switch case inkremental untuk tiap motor sendi, dengan tiap case berisi nilai sudut target yaitu sudut titik tolak lintasan beserta waktu untuk mencapai sudut tersebut. Pemanggilan fungsi yang digunakan adalah setTargetPos(sudut_target, waktu). Akumulasi waktu yang diberikan ke case pada masing-masing motor servo adalah 1 detik, yakni penulis menetapkan 1 detik sebagai waktu yang diperlukan untuk melakukan 1 siklus gait. Percobaan pengembangan dapat memodifikasi pengaturan waktu ini, dikarenakan waktu tempuh satu siklus masing-masing orang berbeda. Adapun agar pergerakan siklus ini berulang, ditambah satu case terakhir untuk mengembalikan nilai case (moveNumber__) kembali ke nol, sehingga

siklus akan terulang dari awal. Berikut merupakan pengaturan switch case yang digunakan untuk motor sendi pinggul kiri (*left hip*).

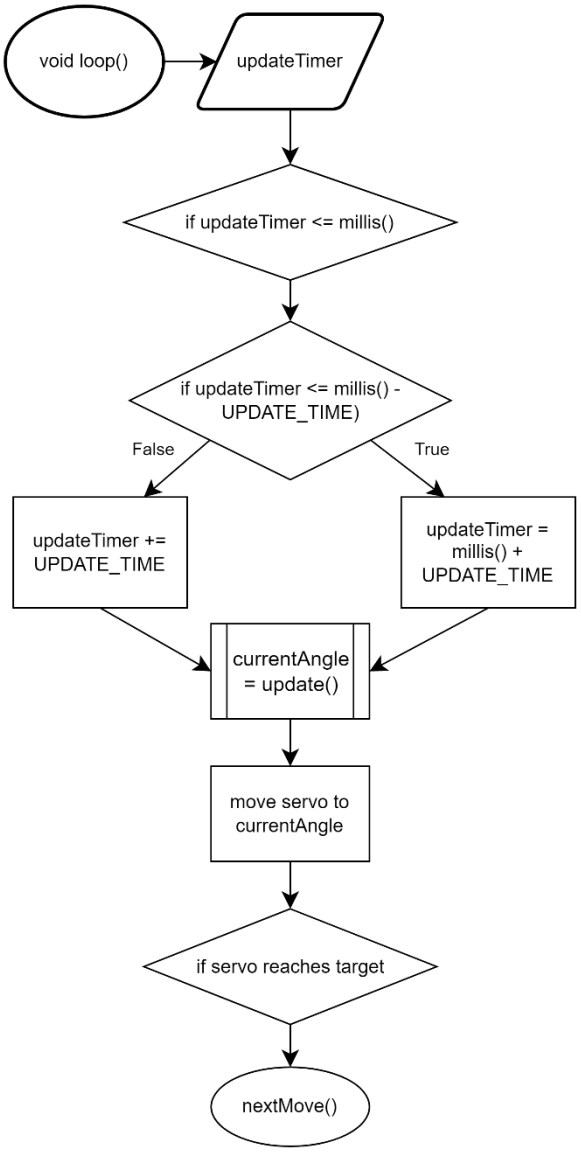
```
void nextMove() {
  switch (moveNumberHipL) {
    case 0:
      servoTrajectoryHL.setTargetPos(72, 0.08);
      break;
    case 1:
      servoTrajectoryHL.setTargetPos(108, 0.27);
      break;
    case 2:
      servoTrajectoryHL.setTargetPos(69, 0.55);
      break;
    case 3:
      servoTrajectoryHL.setTargetPos(108, 0.10);
      break;
    case 4:
      moveNumberHipL = -1;
      break;
   }

  moveNumberHipL++;
  moveNumberKneeL++;
  moveNumberAnkleL++;
  moveNumberHipR++;
  moveNumberKneeR++;
  moveNumberAnkleR++;
}
```



Gambar 5.6. *Flowchart* fungsi void nextMove

Selanjutnya void loop() yang berisi instruksi utama yang berulang berisi perintah untuk meng-*update* posisi sudut servo pada interval yang tetap. Controller akan mengupdate pula terkait posisi sudut servo dan fungsi servoTrajectory.update() akan melihat seberapa dekat posisi sudut saat jalan dengan sudut target. Update inilah yang akan memberikan instruksi penghalusan gerak (percepatan dan perlambatan) untuk tiap motor sekaligus memberi sudut target yang diberikan untuk sendi.write() agar motor servo bergerak menuju nilai sudut tersebut. Nilai-nilai sudut lintasan didapatkan dengan menggunakan Serial.print. Hasil sudut inilah yang akan dibandingkan dengan sudut referensi pada Tabel. Kondisi if servoTrajectory.ready() berfungsi untuk mengecek apakah kontroler telah mencapai sudut target yang ditentukan pada tiap case. Jika iya, maka pergerakan selanjutnya atau case berikutnya akan dijalankan.

```
        void loop() {
          // Update the servo position at regular intervals
          if (updateTimer <= millis()) {
                if (updateTimer <= millis() - UPDATE_TIME) updateTimer = millis()
+ UPDATE_TIME;
                else updateTimer += UPDATE_TIME;

                // Update the controller
                float currentAngleHL = servoTrajectoryHL.update();
                float currentAngleKL = servoTrajectoryKL.update();
                float currentAngleAL = servoTrajectoryAL.update();
                float currentAngleHR = servoTrajectoryHR.update();
                float currentAngleKR = servoTrajectoryKR.update();
                float currentAngleAR = servoTrajectoryAR.update();

                // Set the new servo position; the function only takes integer numbers
                hipL.write(round(currentAngleHL));
                kneeL.write(round(currentAngleKL));
                ankleL.write(round(currentAngleAL));
                hipR.write(round(currentAngleHR));
                kneeR.write(round(currentAngleKR));
                ankleR.write(round(currentAngleAR));

                // Output the target position, along with the current position and velocity
                Serial.print("Target: ");
                Serial.print(servoTrajectoryAL.getTarget());
                Serial.print(", Angle:  ");
                Serial.print(servoTrajectoryAL.getPos());
                Serial.print(", Velocity: ");
                Serial.println(servoTrajectoryAL.getVel());

                // Only once the servo has reached the desired position, complete the
next move
                if (servoTrajectoryHL.ready()) {
                        nextMove();
                }
            }
        }
```



Gambar 5.7. *Flowchart* fungsi void loop

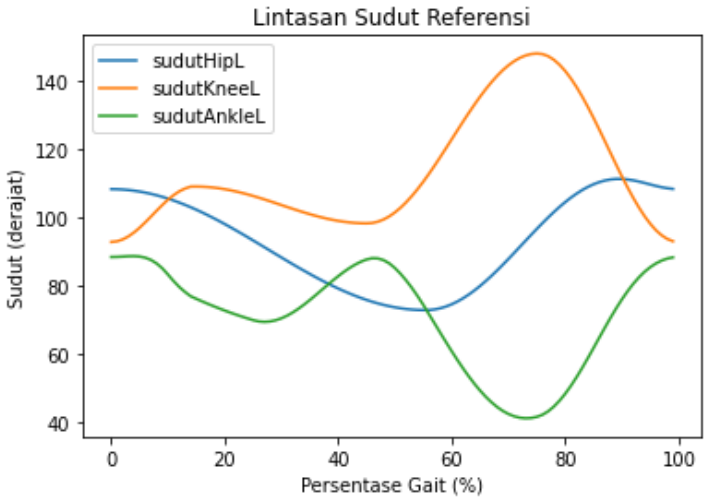
Program coding enam motor servo yang lengkap terdapat pada bagian lampiran.

5.2. Hasil

5.2.1. Lintasan sudut sesuai referensi

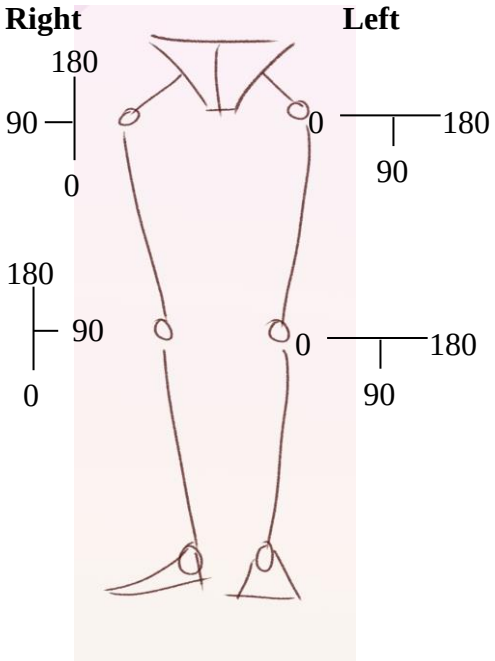
Untuk mendapatkan sudut-sudut pergerakan siklus gait yang ingin digerakkan oleh motor servo, penulis mencari terlebih dahulu sudut berapa saja yang dilalui oleh masing-masing sendi. Hal ini dilakukan dengan merujuk pada grafik *trajectory* yang dipaparkan sebelumnya, dan untuk mendapatkan nilai-nilainya menggunakan perumusan yang ada. Penulis terlebih dahulu mengubah perumusan masing-masing sendi ke dalam program Python kemudian mem-*print* sudut keluaran tiap satu persentase gait. Setiap

sudut yang di-*print* tertera dalam tabel 8.1. pada lampiran. Perhitungan ini dilakukan di Google Collabs yang tersedia secara gratis dan online. Grafik program Python dibuat untuk memastikan sesuai dengan grafik acuan yakni pola pergerakan pada Gambar 3.7. Berikut merupakan grafik hasil perhitungan rumus pada Python.



Gambar 5.8. Lintasan gerak referensi hasil eksperimen Jarzyna et al (2020)

Setelah didapatkan sudut-sudut lintasan pergerakan gait, sudut-sudut disesuaikan dengan posisi masing-masing motor servo yang terpasang pada prototipe, dengan sudut nol motor servo pada bagian kanan tegak lurus dengan lantai sementara bagian kiri paralel dengan lantai. Hal ini akan memengaruhi arah gerak dan juga batasan pergerakan motor servo yang disesuaikan dengan target sudut yang akan dilalui masing-masing servo.



Gambar 5.9. Posisi sudut motor pada *exoskeleton*

Sudut target yang ditentukan sebagai input program merupakan sudut titik tolak pada grafik. Berikut merupakan tabel rangkuman sudut-sudut yang digunakan:

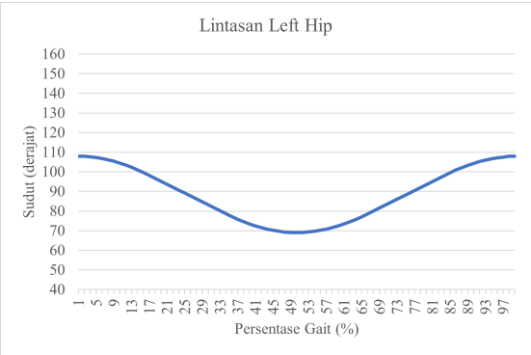
Sendi	*Left Hip*	*Left Knee*	*Left Ankle*
Sudut Target	72	92	88
	108	109	69
	69	98	88
	108	148	41
	72	92	88

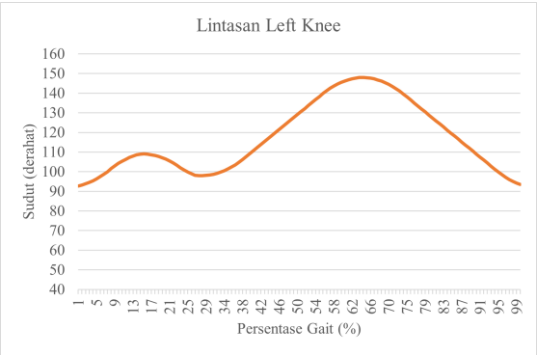
Tabel 5.1. Sudut Target Sendi pada Program

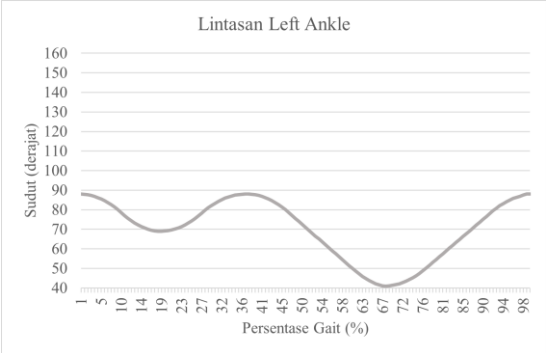
5.2.2. Perbandingan gerakan setiap posisi sudut/sendi dengan referensi

Evaluasi pergerakan dari coding program dilakukan dengan mem-*plotting* sudut-sudut lintasan kemudian membandingkannya dengan grafik referensi. Penulis menggunakan komponen Virtual Monitor pada Proteus yang bisa digunakan selayaknya Serial Monitor pada Arduino IDE. Adapun penggunaan Virtual Monitor dilakukan berulang untuk masing-masing sendi yang diamati. Penulis tidak sempat melakukan pengamatan dengan Serial Monitor dikarenakan sudah kembali ke Surabaya sehingga tidak bisa langsung menghubungkan kode program ke peralatan *hardware* dan prototipe. Sudut-sudut yang dihasilkan disesuaikan agar berpatok pada sudut 90 derajat, seperti pada referensi agar lebih mudah membandingkannya.

Masing-masing motor servo dijalankan secara terpisah terlebih dahulu untuk melihat apakah pergerakan motor tiap sendi sudah sesuai. Adapun pada program coding diberi pengaturan *self-initiation*, yakni masing-masing motor akan mengulang pergerakan siklusnya apabila telah selesai menjalankan satu siklus. Didapatkan hasil grafik sebagai berikut.



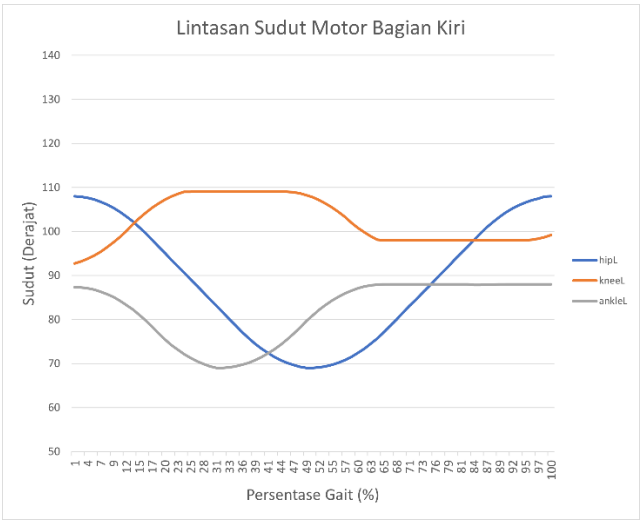




Gambar 5.10. Lintasan gerak sendi bagian kiri dengan *self-initiation*

Pada grafik diamati bahwa untuk masing-masing sendi didapatkan bentuk yang kurang lebih mengikuti pola pada referensi dan tidak adanya pergerakan yang berhenti mendadak. Namun, untuk satu siklus gait, waktu yang ditempuh motor sendi berbeda-beda. Hal ini akan terlihat lebih jelas ketika enam motor servo dijalankan secara bersama-sama.

Berikut merupakan grafik lintasan satu siklus gait motor sendi bagian kiri (*left hip, left knee*, dan *left ankle*) yang dijalankan secara bersamaan. Pergerakan ini memiliki pengaturan inisiasi pada *left hip*, sehingga semua motor akan mengulang siklusnya hanya jika motor *left hip* telah selesai satu siklusnya.



Gambar 5.11. Lintasan gerak satu siklus program coding (dengan inisiasi *left hip*)

Disini terlihat lebih jelas bahwa tiap sendi kurang sinkron. Saat *left hip* sudah menyelesaikan satu siklus gait, motor sendi *left knee* masih separuh jalan. Adapun motor sendi *left ankle* berhenti di lembah pertama dan kemudian konstan berada pada sudut 88°

dan tidak sampai pada akhir siklus. Hal ini mungkin terjadi karena pengaturan inisiator yang kurang tepat.

Sinkronisasi keenam motor servo dapat dicoba dengan memodifikasi kondisi servoTrajectory.ready() ketika semua sendi servo sudah selesai semua pergerakan satu siklus gaitnya, dan masing-masing motor servo dapat diatur besar kecepatannya masing-masing; karena diamati tiap motor sendi menggerakkan satu siklus gait dengan waktu tempuh yang berbeda-beda.

5.2.3. Pergerakan gait prototype exoskeleton 3D

Program coding diberikan ke Arduino ATMega328P untuk dijalankan oleh prototipe. Pada pergerakan prototipe, diamati bahwa keenam motor servo telah berhasil bergerak secara bersamaan dengan arah dan jangkauan yang sesuai. Pergerakan motor sudah halus, yakni tidak adanya pergerakan motor antar sudut yang mendadak atau berhenti dengan mendadak. Terlihat bahwa pergerakan motor lebih lambat dari pengaturan, tidak sesuai dengan pengaturan satu siklus selama satu detik. Hal ini bisa saja terjadi karena *baud rate* yang kurang cepat. Selain itu, terlihat bahwa antar sendi masih kurang sinkron, seperti yang dijelaskan sebelumnya. Berikut merupakan link video rekaman pergerakan prototipe: bit.ly/ExosPKLBRINBdg2023.

BAB VI
PENUTUP

6.1. Kesimpulan

1. Modifikasi hardware dilakukan dengan mengganti motor dengan motor DC SPT5435LV dengan torsi yang lebih besar pada sendi hip dan knee. Modifikasi software yang dilakukan adalah menggunakan library trajectory yang dapat menghitung kecepatan pergerakan antar sudut target motor secara otomatis dari input sudut dan waktu target.
2. Pengaturan pergerakan dilakukan dengan mengcompile main program kemudian diupload ke Arduino melalui kabel. Kabel-kabel sinyal, Vcc, serta Ground tiap-tiap motor servo dihubungkan ke pin Arduino. Baterai 9V terhubung dengan Arduino melalui Breadboard sebagai sumber daya sehingga exoskeleton dapat digerakkan.
3. Pengaturan kecepatan dan pergerakan motor servo dilakukan dengan menentukan sudut target gerak masing-masing sendi kemudian diberikan pada sistem software Arduino yang menghitung kecepatannya secara otomatis. Main program mengatur pergerakan keenam sendi dengan inisiasi left hip yakni perulangan siklus akan dimulai ketika left hip menyelesaikan satu siklus gait.
4. Hasil simulasi pada sistem kendali yang telah dirancang sudah sesuai dengan lintasan sudut referensi. Namun ketika dijalankan bersamaan terlihat antar sudut kurang sinkron dan masih lambat.

6.2. Saran

1. Memperbaiki kode program untuk mendapatkan sinkronisasi antar sendi dan mencapai waktu tempuh siklus yang sama, dengan mengatur besar kecepatan untuk setiap sendinya (menggunakan PWM);
2. Pergerakan setiap sendi dapat dipertimbangkan sesuai siklus gait referensi dalam pengaturan inisiasi gerak siklus pada kode program
3. Menggunakan driver motor dc untuk menghasilkan data pergerakan sudut setiap sendi lebih cepat dan akurat.
4. Suplai daya yang kurang stabil yang dapat menjadi salah satu faktor dari lambatnya pergerakkan eksoskeleton.

DAFTAR PUSTAKA

Arduino®, “Arduino® MEGA 2560 Rev3”. Product Reference Manual SKU: A000067, February 2023. Accessed at https://docs.arduino.cc/static/366335ec602afb5c5ee02a1306852f71/A000067-datasheet.pdf

Haider, S.M., Takhakh, A.M. and Al-Waily, M. (2021) “*A review study on measurement and evaluation of prosthesis testing platform during gait cycle within Sagittal Plane*,” 2021 14th International Conference on Developments in eSystems Engineering (DeSE) [Preprint]. Available at: https://doi.org/10.1109/dese54285.2021.9719575.

Kharb, A, Saini, V, Jain, YK, & Dhiman, S. 2011, '*A review of gait cycle and its parameters*', IJCEM International Journal of Computational Engineering & Management, Vol. 13, July 2021. ISSN (Online): 2230-7893. researchgate.net, <https://www.researchgate.net/profile/Surender-Dhiman/publication/268423123_A_review_of_gait_cycle_and_its_parameters/links/582f259108ae138f1c035005/A-review-of-gait-cycle-and-its-parameters.pdf>

Olga Jarzyna, Dariusz Grzelczyk, Bartosz Stańczyk & Jan Awrejcewicz (2020): *Generation of a gait pattern for a lower limb rehabilitation exoskeleton*, Mechanics Based Design of Structures and Machines, DOI: 10.1080/15397734.2020.1858868

Qureshi, Muhammad Hamza, et al. 2018. *Biomechanical Design and Control of Lower Limb Exoskeleton for Sit-to-Stand and Stand-to-Sit Movements*. Pakistan: Air University, Islamabad

Shi, D., Zhang, W., Zhang, W. et al. (2019). *A Review on Lower Limb Rehabilitation Exoskeleton Robots*. Chin. J. Mech. Eng. 32, 74. https://doi.org/10.1186/s10033-019-0389-8

Tian, M., Wang, X., Wang, J., & Gan, Z. (2019). *Design of A Lower Limb Exoskeleton Driven by Tendon-sheath Artificial Muscle**. 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO). doi:10.1109/robio49542.2019.8961802

Tower Pro, “TowerPro SG90 - Micro Servo” SG90 9 g Micro Servo datasheet, Apr. 2021. Accessed at: https://datasheetspdf.com/pdf-file/791970/TowerPro/SG90/1

LAMPIRAN

Coding Library ‘Servo Trajectory’

```
/* * * * * * * * * * * * * * * * * * * * * * *
 * TRAJECTORY CONTROLLER CLASS
 *
 * Code by: Simon Bluett
 * Website: https://wired.chillibasket.com
 * Version: 1.2
 * Date:    19th May 2020
 * Copyright (C) 2020, MIT License
 *
 * This class can be used to control the movement and velocity of
 * servo motors or DC motors with encoders. When a new target
 * position is set, it accelerates at a constant rate until a
 * maximum velocity is reached. As the system approaches the
 * target position, it then smoothly decelerates to a stop.
 *
 * This controller uses a simple trapezoidal velocity profile.
 *
 * For more information, please visit my tutorial at:
 * https://wired.chillibasket.com/2020/05/servo-trajectory
 *
 * An example Arduino sketch is also provided with this library
 * to show how the class can be used in practice.
 * * * * * * * * * * * * * * * * * * * * * * */

#ifndef TRAJECTORY_H
#define TRAJECTORY_H

#include <Arduino.h>

// TRAJECTORY CLASS
class Trajectory {

public:
	// Constructors
	Trajectory(float _maxVel = 100, float _setAcc = 50, float _setDec = -1, float
_thresh = 0.1);

	// Set target position/velocity
	void setTargetPos(float _target);
	bool setTargetPos(float _target, float _time);
	void setTargetVel(float _target);

	// Get the current target position or velocity values
	float getTarget() { return target; };
```

```
        // Set max velocity, acceleration or decelleration
        void setMaxVel(float _maxVel) { maxVel = _maxVel; };
        void setAcc(float _acc) { acc = _acc; };
        void setDec(float _dec) { dec = _dec; };

        // Get the current max velocity, accel or decel values
        float getMaxVel() { return maxVel; };
        float getAcc() { return acc; };
        float getDec() { return dec; };

        // Get the current position and velocity
        float getPos() { return curPos; };
        float getVel() { return curVel; };

        // Set the current position
        void setPos(float newPos = 0);

        // Reset the controller
        void reset(float newPos = 0);

        // Check if the controller has reached the desired position
        bool ready() { return noTasks; };

        // Update and get the new position value
        // dT = time since update or reset function was last called in milliseconds
        float update(float dT);
        float update();

        // Default destructor
        ~Trajectory();

private:
        // type = 0: "position" control (Straight line with parabolic blends)
        // type = 1: "velocity" control (Straight line with parabolic blends)
        int type;         // Target type

        float threshold;   // Position error threshold below which no motion occurs
        float target;      // Target Position or Velocity
        float curPos;      // Current Position (units)
        float curVel;      // Current Velocity (units/second)
        float maxVel;      // Maximum Velocity (units/second)
        float velGoal;     // Target velocity to reach goal position at a specific time
        float acc;         // Constant Acceleraction (units/second^2)
        float dec;         // Constant Deceleration (units/second^2)
        unsigned long oldTime;
        bool noTasks;
```

```
};

/**
 * Default constructor
 *
 * @param (_maxVel) Maximum velocity - default = 100
 * @param (_acc) Constant acceleration - default = 50
 * @param (_dec) Constant deceleration - default = same as acceleration
 * @param (_thesh) Cutoff threshold - default = 0.1
 */
Trajectory::Trajectory(float _maxVel, float _acc, float _dec, float _thresh) {
        type = 0;
        target = 0;
        curPos = 0;
        curVel = 0;
        maxVel = _maxVel;
        velGoal = _maxVel;
        acc = _acc;
        if (_dec == -1) dec = _acc;
        else dec = _dec;
        oldTime = millis();
        threshold = _thresh;
        noTasks = true;
}

/**
 * Default destructor
 */
Trajectory::~Trajectory() {
        // Empty
}

/**
 * Set a new target position
 *
 * @param  (_targetPos) New target position value
 * @note   This overides any previous position or velocity commands
 */
void Trajectory::setTargetPos(float _targetPos) {
        target = _targetPos;
        velGoal = maxVel;
        type = 0;
        noTasks = false;
}

/**
 * Set a new target position and time
```

```
 *
 * @param  (_targetPos) New target position value
 * @param  (_time) The time in seconds until system should reach target position
 * @return (True) goal is achievable, (False) goal won't be achieved within specified
time
 * @note   This overides any previous position or velocity commands
 */
bool Trajectory::setTargetPos(float _targetPos, float _time) {
        target = _targetPos;

        // Take into account the cut-off theshold
        _time = _time + sqrt(2 * threshold / dec);

        // Calculate the velocity required to reach the goal within the specified time
        float sqrtTerm = (_time * _time) - 2 * abs(_targetPos - curPos) * (1/acc +
1/dec);

        // If the term is a positive number, then a solution exists
        if (sqrtTerm >= 0) {
                velGoal = (_time - sqrt(sqrtTerm)) / (1/acc + 1/dec);

                if (velGoal > maxVel) velGoal = maxVel;
        } else {
                velGoal = maxVel;
        }

        type = 0;
        noTasks = false;

        if (velGoal == maxVel) return false;
        else return true;
}

/**
 * Set a new target velocity
 *
 * @param  (_targetVel) New target velocity value
 * @note   This overides any previous position or velocity commands
 */
void Trajectory::setTargetVel(float _targetVel) {
        if (_targetVel > maxVel) target = maxVel;
        else if (_targetVel < -maxVel) target = -maxVel;
        else target = _targetVel;
        type = 1;
        noTasks = false;
}
```

```
/**
 * Update controller and calculate new position value
 *
 * @param   (dT) The time change (ms) since function was last called
 * @return  The new current position of the system
 */
float Trajectory::update(float dT) {

	// Convert ms to seconds
	dT /= 1000.0;

	// Position Control (Linear velocity with parabolic blends)
	if (type == 0) {

		float posError = target - curPos;

		// If position error is above the threshold
		if (abs(posError) > threshold) {

			// Determine motion direction
			bool dir = true;
			if (posError < 0) dir = false;

			// Determine whether to accelerate or decelerate
			float acceleration = acc;
			if ((curVel * curVel / (2 * dec)) >= abs(posError)) acceleration
= -dec;

			// Update the current velocity
			if (dir) curVel += acceleration * dT;
			else curVel -= acceleration * dT;

			// Limit Velocity
			if (curVel > velGoal) curVel = velGoal;
			else if (curVel < -velGoal) curVel = -velGoal;

			float dP = curVel * dT;

			// Prevent overshooting/jittering around target
			if (abs(dP) < abs(posError)) curPos += dP;
			else curPos = target;

			//Serial.print(millis()); Serial.print(",");
			//Serial.print(curPos); Serial.print(",");
			//Serial.print(curVel); Serial.print(",");
			//if (abs(curVel) != velGoal) Serial.println(acceleration * (-1 +
2*dir));
```

```
                //else Serial.println(0);

        } else {
                curVel = 0;
                curPos = target;
                noTasks = true;
        }

// Velocity Control (Linear velocity with parabolic blends)
} else if (type == 1) {

        float velError = target - curVel;

        // If velocity error is above the threshold
        if (abs(velError) > threshold) {

                // Determine whether to accelerate or decelerate
                float acceleration = acc;
                if (target < curVel && curVel >= 0) acceleration = -dec;
                else if (target < curVel && curVel < 0) acceleration = -acc;
                else if (target > curVel && curVel < 0) acceleration = dec;

                // Update the current velocity
                float dV = acceleration * dT;
                if (abs(dV) < abs(velError)) curVel += dV;
                else curVel = target;

        } else {
                curVel = target;
                noTasks = true;
        }

        // Limit Velocity
        if (curVel > maxVel) curVel = maxVel;
        else if (curVel < -maxVel) curVel = -maxVel;

        // Update current position
        curPos += curVel * dT;
}

// Error check for NaN
//if (curPos != curPos) {
//      //printf(F("ERROR: Dynamics calculation error"));
//      if (type == 0) curPos = target;
//      else curPos = 0;
//      curVel = 0;
//}
```

```
        return curPos;
}

/**
 * Overloaded "update" function, were the time change has not been specified
 *
 * @return  The new current position of the system
 */
float Trajectory::update() {
        // Calculate Time Change
        unsigned long newTime = millis();
        float dT = float(newTime - oldTime);
        oldTime = newTime;
        return update(dT);
}

/**
 * Reset all class variables
 *
 * @param (newPos) the current position - default = 0
 */
void Trajectory::reset(float newPos) {
        curPos = newPos;
        curVel = 0;
        target = 0;
        velGoal = maxVel;
        noTasks = true;
        oldTime = millis();
}

/**
 * Set/reset the current position value
 *
 * @param (newPos) the new current position - default = 0
 */
void Trajectory::setPos(float newPos) {
        curPos = newPos;
}

#endif /* TRAJECTORY_H */
```

Coding programnya itu

```
#include <trajectory.h>
#include <Servo.h>
```

```
#include <math.h>

Trajectory servoTrajectoryHL(60, 40, 34);
Trajectory servoTrajectoryKL(60, 40, 34);
Trajectory servoTrajectoryAL(60, 40, 34);
Trajectory servoTrajectoryHR(60, 40, 34);
Trajectory servoTrajectoryKR(60, 40, 34);
Trajectory servoTrajectoryAR(60, 40, 34);

Servo hipR;
Servo kneeR;
Servo ankleR;
Servo hipL;
Servo kneeL;
Servo ankleL;

// --- Define global variables ---
// The controller will be updated at a rate of 100Hz
#define UPDATE_FREQUENCY 100
#define UPDATE_TIME (1000 / UPDATE_FREQUENCY)
unsigned long updateTimer = 0;

int moveNumberHipL = 0;
int moveNumberKneeL = 0;
int moveNumberAnkleL = 0;
int moveNumberHipR = 0;
int moveNumberKneeR = 0;
int moveNumberAnkleR = 0;

float sudutHipL, sudutHipR, sudutKneeL, sudutKneeR, sudutAnkleL, sudutAnkleR;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("Starting program");

  hipR.attach(A0);    // attaches hip R Servo on pin A0 to the servo object
  kneeR.attach(A1);   // attaches knee R Servo on pin A1 to the servo object
  ankleR.attach(A2);  // attaches ankle R Servo on pin A2 to the servo object
  hipL.attach(A3);    // attaches hip L Servo on pin A3 to the servo object
  kneeL.attach(A4);   // attaches knee L Servo on pin A4 to the servo object
  ankleL.attach(A5);  // attaches ankle L Servo on pin A5 to the servo object

  servoTrajectoryHL.reset(90);
  servoTrajectoryKL.reset(90);
  servoTrajectoryKL.reset(90);
  servoTrajectoryHR.reset(0);
```

```
  servoTrajectoryKR.reset(0);
  servoTrajectoryKR.reset(90);
};

// NORMAL WALKING
void nextMove() {
  switch (moveNumberHipL) {
    case 0:
                // First we move to the 72° position in 0.08 seconds
      servoTrajectoryHL.setTargetPos(72, 0.08);
      break;
    case 1:
      // Then move to 108° position in 0.27 seconds
      servoTrajectoryHL.setTargetPos(108, 0.27);
      break;
    case 2:
      // And so on
      servoTrajectoryHL.setTargetPos(69, 0.55);
      break;
    case 3:
      servoTrajectoryHL.setTargetPos(108, 0.10);
      break;
    case 4:
      moveNumberHipL = -1;
      break;
   }

  switch (moveNumberKneeL) {
    case 0:
      servoTrajectoryKL.setTargetPos(88, 0.01);
      break;
    case 1:
      servoTrajectoryKL.setTargetPos(71, 0.16);
      break;
    case 2:
      servoTrajectoryKL.setTargetPos(82, 0.29);
      break;
    case 3:
      servoTrajectoryKL.setTargetPos(32, 0.31);
      break;
    case 4:
      servoTrajectoryKL.setTargetPos(88, 0.24);
      break;
    case 5:
      moveNumberKneeL = -1;
      break;
  }
```

```
switch (moveNumberAnkleL) {
  case 0:
    servoTrajectoryAL.setTargetPos(88, 0.04);
    break;
  case 1:
    servoTrajectoryAL.setTargetPos(88, 0.08);
    break;
  case 2:
    servoTrajectoryAL.setTargetPos(69, 0.19);
    break;
  case 3:
    servoTrajectoryAL.setTargetPos(88, 0.16);
    break;
  case 4:
    servoTrajectoryAL.setTargetPos(41, 0.25);
    break;
  case 5:
    servoTrajectoryAL.setTargetPos(88, 0.28);
    break;
  case 6:
    moveNumberAnkleL = -1;
    break;
}

switch (moveNumberHipR) {
  case 0:
    servoTrajectoryHR.setTargetPos(0, 0.01);
    break;
  case 1:
    servoTrajectoryHR.setTargetPos(21, 0.40);
    break;
  case 2:
    servoTrajectoryHR.setTargetPos(0, 0.60);
    break;
  case 3:
    moveNumberHipR = -1;
    break;
}

switch (moveNumberKneeR) {
  case 0:
    servoTrajectoryKR.setTargetPos(11, 0.01);
    break;
  case 1:
    servoTrajectoryKR.setTargetPos(54, 0.21);
    break;
```

```
  case 2:
    servoTrajectoryKR.setTargetPos(2, 0.30);
    break;
  case 3:
    servoTrajectoryKR.setTargetPos(19, 0.15);
    break;
  case 4:
    servoTrajectoryKR.setTargetPos(8, 0.26);
    break;
  case 5:
    servoTrajectoryKR.setTargetPos(9, 0.08);
    break;
  case 6:
    moveNumberKneeR = -1;
    break;
}

switch (moveNumberAnkleR) {
  case 0:
    servoTrajectoryAR.setTargetPos(85, 0.03);
    break;
  case 1:
    servoTrajectoryAR.setTargetPos(41, 0.24);
    break;
  case 2:
    servoTrajectoryAR.setTargetPos(88, 0.26);
    break;
  case 3:
    servoTrajectoryAR.setTargetPos(88, 0.08);
    break;
  case 4:
    servoTrajectoryAR.setTargetPos(69, 0.19);
    break;
  case 5:
    servoTrajectoryAR.setTargetPos(88, 0.23);
    break;
  case 6:
    moveNumberAnkleR = -1;
    break;
}

moveNumberHipL++;
moveNumberKneeL++;
moveNumberAnkleL++;
moveNumberHipR++;
moveNumberKneeR++;
moveNumberAnkleR++;
```

```
}

void loop() {
  // Update the servo position at regular intervals
        if (updateTimer <= millis()) {
                if (updateTimer <= millis() - UPDATE_TIME) updateTimer = millis()
+ UPDATE_TIME;
                else updateTimer += UPDATE_TIME;

   // Update the controller
    float currentAngleHL = servoTrajectoryHL.update();
    float currentAngleKL = servoTrajectoryKL.update();
    float currentAngleAL = servoTrajectoryAL.update();
    float currentAngleHR = servoTrajectoryHR.update();
    float currentAngleKR = servoTrajectoryKR.update();
    float currentAngleAR = servoTrajectoryAR.update();

    // Set the new servo position; the function only takes integer numbers
    hipL.write(round(currentAngleHL));
    kneeL.write(round(currentAngleKL));
    ankleL.write(round(currentAngleAL));
    hipR.write(round(currentAngleHR));
    kneeR.write(round(currentAngleKR));
    ankleR.write(round(currentAngleAR));

    /**
    * For more precise servo control, you could use writeMicroseconds.
    * The min and max PWM pulse widths which correspond to the 0° and 180°
    * positions needs to be inserted for MIN_PWM and MAX_PWM.
    */
    //myservo.writeMicroseconds(map(currentAngle, 0, 180, MIN_PWM,
MAX_PWM));

    // Output the target position, along with the current position and velocity
    Serial.print("Target: ");
    Serial.print(servoTrajectoryAL.getTarget());
    Serial.print(", Angle:  ");
    Serial.print(servoTrajectoryAL.getPos());
    Serial.print(", Velocity: ");
    Serial.println(servoTrajectoryAL.getVel());

    // Only once the servo has reached the desired position, complete the next move
    if (servoTrajectoryHL.ready()) {
        nextMove();
    }
  }
}
```

Tabel 8.1. Sudut-sudut lintasan satu siklus gait (disesuaikan dengan posisi sudut motor)

Persentase Gait (%)	Besar Sudut Sendi (L-kiri, R-kanan) (°)					
	Hip L	Hip R	Knee L	Knee R	Ankle L	Ankle R
1	108.32	-16.27	92.86	11.5	88.46	85.23
2	108.29	-16.54	93.04	12.89	88.49	83.52
3	108.21	-16.76	93.58	14.51	88.58	81.57
4	108.06	-16.93	94.45	16.33	88.68	79.39
5	107.87	-17.03	95.61	18.34	88.71	77.0
6	107.61	-17.08	97.01	20.5	88.6	74.44
7	107.31	-17.06	98.59	22.81	88.26	71.73
8	106.94	-16.88	100.28	25.23	87.6	68.99
9	106.53	-16.54	101.99	27.73	86.58	66.25
10	106.06	-16.04	103.67	30.3	85.15	63.55
11	105.55	-15.38	105.21	32.89	83.33	60.9
12	104.99	-14.58	106.57	35.49	81.41	58.34
13	104.38	-13.63	107.68	38.07	79.7	55.88
14	103.72	-12.55	108.49	40.59	78.24	53.56
15	103.03	-11.34	109.05	43.03	76.98	51.38
16	102.29	-10.02	109.07	45.36	76.22	49.38
17	101.52	-8.59	109.03	47.55	75.49	47.57
18	100.71	-7.08	108.93	49.59	74.78	45.97
19	99.87	-5.48	108.78	51.45	74.09	44.58
20	99.0	-3.82	108.57	53.11	73.43	43.43
21	98.1	-2.11	108.31	54.55	72.8	42.52
22	97.18	-0.37	108.0	55.75	72.19	41.85
23	96.24	1.4	107.64	56.71	71.6	41.44
24	95.28	3.17	107.24	57.41	71.04	41.27
25	94.3	4.94	106.8	57.85	70.5	41.35
26	93.32	6.68	106.33	58.01	69.99	41.67
27	92.32	8.38	105.83	57.83	69.61	42.29
28	91.32	10.03	105.31	57.22	69.48	43.3
29	90.31	11.61	104.77	56.18	69.6	44.68
30	89.31	13.11	104.21	54.74	69.95	46.39
31	88.31	14.51	103.66	52.91	70.52	48.43
32	87.32	15.81	103.1	50.72	71.29	50.73
33	86.34	16.99	102.55	48.2	72.23	53.27
34	85.37	18.05	102.01	45.41	73.33	56.0
35	84.42	18.97	101.49	42.38	74.55	58.88
36	83.49	19.74	100.99	39.15	75.87	61.83
37	82.59	20.37	100.52	35.79	77.25	64.83
38	81.7	20.84	100.09	32.35	78.67	67.8
39	80.85	21.15	99.69	28.87	80.1	70.7
40	80.03	21.3	99.34	25.42	81.5	73.47

41	79.24	21.29	99.04	22.05	82.84	76.06
42	78.49	21.15	98.78	18.81	84.1	78.45
43	77.78	20.89	98.58	15.75	85.24	80.61
44	77.11	20.54	98.43	12.93	86.24	82.51
45	76.48	20.12	98.33	10.39	87.08	84.14
46	75.9	19.68	98.3	8.17	87.73	85.49
47	75.37	19.25	98.41	6.3	88.1	86.58
48	74.88	18.87	98.79	4.81	88.04	87.42
49	74.45	18.58	99.44	3.73	87.54	88.0
50	74.06	18.39	100.34	3.08	86.6	88.35
51	73.73	18.32	101.5	2.86	85.23	88.46
52	73.46	18.29	102.89	3.04	83.52	88.49
53	73.24	18.21	104.51	3.58	81.57	88.58
54	73.07	18.06	106.33	4.45	79.39	88.68
55	72.97	17.87	108.34	5.61	77.0	88.71
56	72.92	17.61	110.5	7.01	74.44	88.6
57	72.94	17.31	112.81	8.59	71.73	88.26
58	73.12	16.94	115.23	10.28	68.99	87.6
59	73.46	16.53	117.73	11.99	66.25	86.58
60	73.96	16.06	120.3	13.67	63.55	85.15
61	74.62	15.55	122.89	15.21	60.9	83.33
62	75.42	14.99	125.49	16.57	58.34	81.41
63	76.37	14.38	128.07	17.68	55.88	79.7
64	77.45	13.72	130.59	18.49	53.56	78.24
65	78.66	13.03	133.03	19.05	51.38	76.98
66	79.98	12.29	135.36	19.07	49.38	76.22
67	81.41	11.52	137.55	19.03	47.57	75.49
68	82.92	10.71	139.59	18.93	45.97	74.78
69	84.52	9.87	141.45	18.78	44.58	74.09
70	86.18	9.0	143.11	18.57	43.43	73.43
71	87.89	8.1	144.55	18.31	42.52	72.8
72	89.63	7.18	145.75	18.0	41.85	72.19
73	91.4	6.24	146.71	17.64	41.44	71.6
74	93.17	5.28	147.41	17.24	41.27	71.04
75	94.94	4.3	147.85	16.8	41.35	70.5
76	96.68	3.32	148.01	16.33	41.67	69.99
77	98.38	2.32	147.83	15.83	42.29	69.61
78	100.03	1.32	147.22	15.31	43.3	69.48
79	101.61	0.31	146.18	14.77	44.68	69.6
80	103.11	-0.69	144.74	14.21	46.39	69.95
81	104.51	-1.69	142.91	13.66	48.43	70.52
82	105.81	-2.68	140.72	13.1	50.73	71.29
83	106.99	-3.66	138.2	12.55	53.27	72.23
84	108.05	-4.63	135.41	12.01	56.0	73.33
85	108.97	-5.58	132.38	11.49	58.88	74.55
86	109.74	-6.51	129.15	10.99	61.83	75.87
87	110.37	-7.41	125.79	10.52	64.83	77.25

88	110.84	-8.3	122.35	10.09	67.8	78.67
89	111.15	-9.15	118.87	9.69	70.7	80.1
90	111.3	-9.97	115.42	9.34	73.47	81.5
91	111.29	-10.76	112.05	9.04	76.06	82.84
92	111.15	-11.51	108.81	8.78	78.45	84.1
93	110.89	-12.22	105.75	8.58	80.61	85.24
94	110.54	-12.89	102.93	8.43	82.51	86.24
95	110.12	-13.52	100.39	8.33	84.14	87.08
96	109.68	-14.1	98.17	8.3	85.49	87.73
97	109.25	-14.63	96.3	8.41	86.58	88.1
98	108.87	-15.12	94.81	8.79	87.42	88.04
99	108.58	-15.55	93.73	9.44	88.0	87.54
100	108.39	-15.94	93.08	10.34	88.35	86.6