CHAPTER 3 - METHODOLOGY

3.1. Methodology Overview

The overview of the implemented research methodology is described as attached on Figure 3.1.

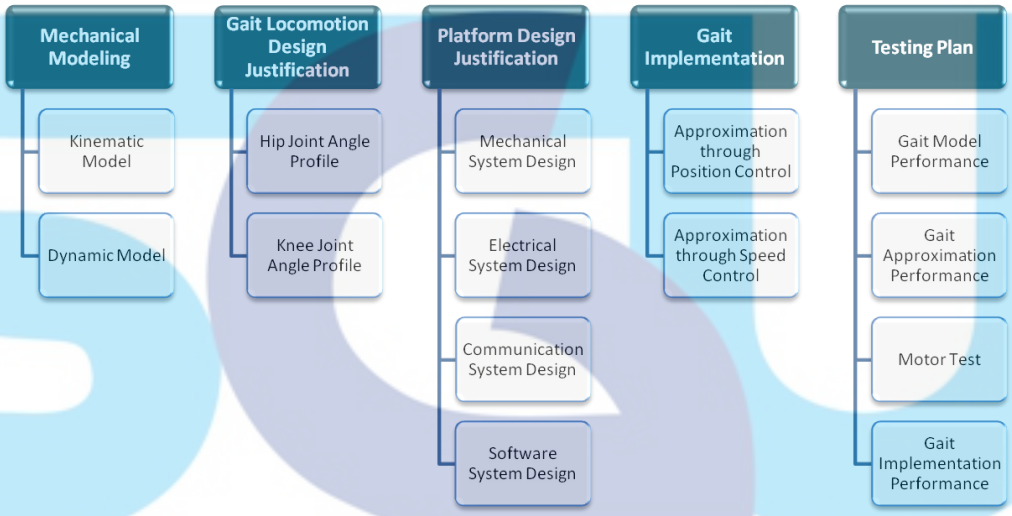

Figure 3.1. Methodology Overview of the Research

The primary step was to create the mechanical model of the quadruped robot, consisting of the kinematic model and the dynamic model of the robot. Kinematic model of the quadruped robot includes the geometric relationships involved in the motion of the system without considering the forces and moments causing the motion. While investigating forces and torques applied to the joints and the toes of the quadruped legs in static equilibrium, having a kinematic model of the system would also be advantageous. On the other hand, dynamic model defines the motion of the system by referring to the forces and moments acting on the system. These mechanical models were parameterized and used to calculate the mechanical torque applied on the actuators and the exerted spring forces due to the load in static equilibrium.

Designing the sinusoidal gait for the quadruped robot was the next step. Similarly to the mechanical modeling, the gait modeling also required parameterization on defining the hip and knee joint angle profiles. The generated gait could be evaluated by examining the stick figures simulation result.

The mechatronics platform design is composed of the mechanical, electrical, communication and software system design of the newly-developed quadruped robot. Design considerations regarding mechanical constraints, selection of electrical components, and methods of communicating and control of the quadruped robot were made.

Implementation of the preferred sinusoidal gait on the actual mechatronics platform of the quadruped robot was done in two ways. The first one was by only approximating the joint angles to reach certain angular position setpoints during some particular time setpoints, thus relying on the position control mode of the servos. The other way was by also approximating the angular velocity setpoints in addition to the angular position setpoints, hence exploring the possibility of attaining more control of the resulting angular position by means of continuous rotation speed control mode.

Finally, both the design and implementation of the sinusoidal gait were tested. The experiments to be conducted revolved around gait model performance, gait approximation performance, motor test, and gait implementation performance.

3.2. Mechanical Modeling

As briefly mentioned on Section 3.1, two mechanical models were made: the kinematic model and the dynamic model.

3.2.1. Kinematic Model

The kinematic model of the quadruped leg is displayed on Figure 3.2. Points labeled with numbers indicate the center of mass position of the particular leg segment. Point 1, 2, 3 and 4 respectively mark the center of mass position of leg segment hip-knee,

knee-ankle, ankle-toe and s1-s3. The other points are tagged with alphabets. Point h, s1, k, a, s3and t respectively represent hip joint, linear spring attachment point, knee joint, ankle joint, parallel segment attachment point and toe tip of the leg.

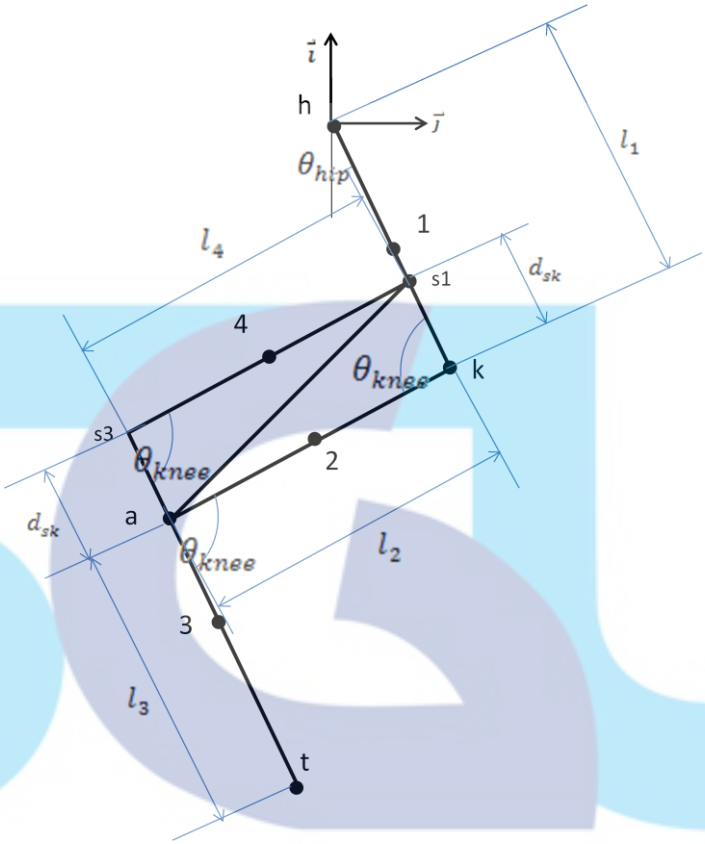

Figure 3.2. Kinematic Model of the Quadruped Leg

With the assumption that the hip-knee and ankle-toe leg segments are parallel to each other, two angles are to be defined. Hip joint angle (θ_{hip}) is the angle of the hip-knee segment to an axis on the hip joint which is parallel to the transverse plane of the platform. Knee joint angle (θ_{knee}) is the angle between hip-knee and knee-ankle leg segments and between knee-ankle and ankle-foot segment.

The kinematic model of the leg is drawn on two-dimensional parasagittal plane which coincides with the hip of the particular leg. The position vectors of the points are represented by the following equations:

$$\overrightarrow{r_1} = \overrightarrow{r_h} + \frac{1}{2} l_1 \left\{ \left[\sin(\theta_{hip}) \right] \vec{\imath} - \left[\cos(\theta_{hip}) \right] \vec{\jmath} \right\} \quad \text{..... Eq. (3.1)}$$

$$\overrightarrow{r_{s1}} = \overrightarrow{r_h} + (l_1 - d_{sk})\{[\sin(\theta_{hip})]\vec{\imath} - [\cos(\theta_{hip})]\vec{\jmath}\} \quad \text{..... Eq. (3.2)}$$

$$\overrightarrow{r_4} = \overrightarrow{r_{s1}} - \frac{1}{2} l_4 \{[\sin(\theta_{hip} + \theta_{knee})]\vec{\imath} - [\cos(\theta_{hip} + \theta_{knee})]\vec{J}\} \quad \text{. Eq. (3.3)}$$

$$\overrightarrow{r_{s3}} = \overrightarrow{r_{s1}} - l_4 \{[\sin(\theta_{hip} + \theta_{knee})]\vec{\imath} - [\cos(\theta_{hip} + \theta_{knee})]\vec{J}\} \quad \text{.... Eq. (3.4)}$$

$$\overrightarrow{r_k} = \overrightarrow{r_h} + l_1 \{[\sin(\theta_{hip})]\vec{\imath} - [\cos(\theta_{hip})]\vec{J}\} \quad \text{..... Eq. (3.5)}$$

$$\overrightarrow{r_2} = \overrightarrow{r_k} - \frac{1}{2} l_2 \{[\sin(\theta_{hip} + \theta_{knee})]\vec{\imath} - [\cos(\theta_{hip} + \theta_{knee})]\vec{J}\} \quad \text{... Eq. (3.6)}$$

$$\overrightarrow{r_a} = \overrightarrow{r_k} - l_2 \{[\sin(\theta_{hip} + \theta_{knee})]\vec{\imath} - [\cos(\theta_{hip} + \theta_{knee})]\vec{J}\} \quad \text{..... Eq. (3.7)}$$

$$\overrightarrow{r_3} = \overrightarrow{r_a} + \frac{l_3 - d_{sk}}{2}\{[\sin(\theta_{hip})]\vec{\imath} - [\cos(\theta_{hip})]\vec{\jmath}\} \quad \text{..... Eq. (3.8)}$$

$$\overrightarrow{r_t} = \overrightarrow{r_a} + l_3 \{[\sin(\theta_{hip})]\vec{\imath} - [\cos(\theta_{hip})]\vec{J}\} \quad \text{..... Eq. (3.9)}$$

Where l_1 = length of the hip-knee segment

d_{sk} = distance from the linear springs attachment point to the knee joint, or from the parallel segment attachment to the ankle joint

l_2 = length of the knee-ankle segment

l_3 = length of the ankle-toe segment

l_4 = length of the s1-s3 segment

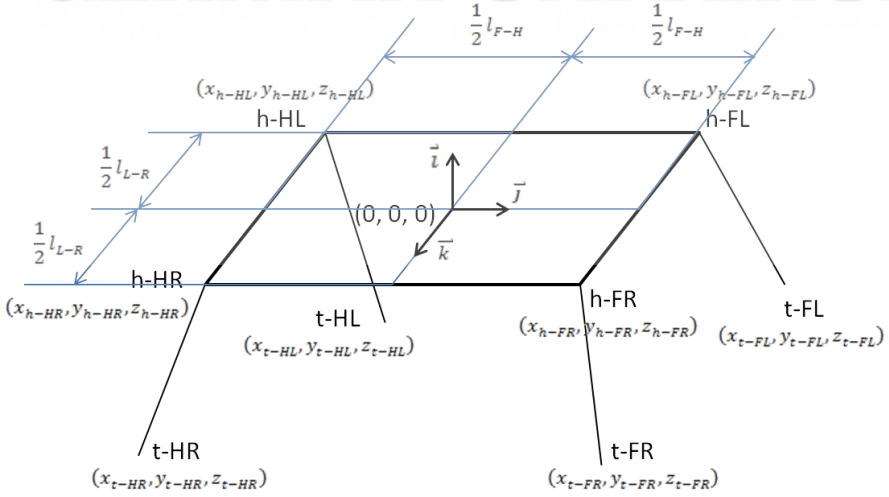

Figure 3.3. Kinematic Model of the Quadruped Body

Meanwhile, the kinematic model of the body of the quadruped robot is shown in Figure 3.3. As depicted in the figure, the reference coordinate system is placed on the centroid of the transverse plane of the quadruped robot. Point h-FL, h-FR, h-HL, and h-HR indicate the position of the hip joint of the front-left leg, front-right leg, hind-left leg, and hind-right leg respectively. Point t-FL, t-FR, t-HL, and t-HR mark the location of the toe of the front-left leg, front-right leg, hind-left leg, and hind-right leg correspondingly. The distance between the front hip joint and the hind hip joint of the same parasagittal plane is symbolized by l_{F-H}, while the distance between the left hip joint and the right hip joint of the same paracoronal plane is denoted by l_{L-R}.

3.2.2. Dynamic Model

The dynamic model of the leg segment of the quadruped robot is illustrated as free body diagrams. These diagrams assisted the mechanical torque calculation.

3.2.2.1. Free Body Diagrams

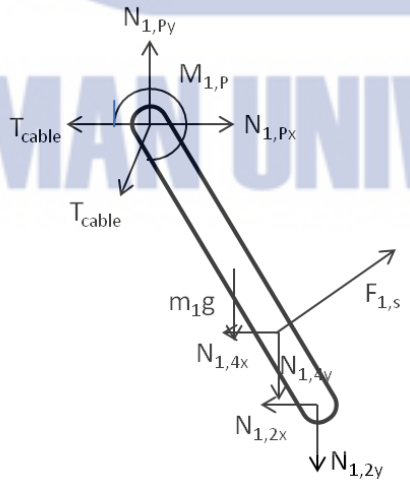

Figure 3.4. Free Body Diagram of the Front Right Leg, First Segment

Figure 3.4, 3.5, 3.6 and 3.7 contain the free body diagram of the front quadruped leg. The free body diagram of the hind quadruped leg differs slightly on the direction of

the horizontal cable force at the hip joint. On the front quadruped legs, the horizontal cable force at the hip joint is facing backwards, while on the hind quadruped legs it is facing forwards.

Figure 3.4 is the free body diagram of the first leg segment of the front-right quadruped leg. Torque $M_{1,P}$ is applied by the hip actuator on the hip joint of the leg. The hip joint also receives supporting force $N_{1,P}$ from the platform. The direction of the x and y components of $N_{1,P}$ shown on Figure 3.4 is the direction of the force components when the toe of the leg is not in contact with the ground during its swing phase. When the toe is in contact with the ground during its stance phase, $N_{1,Px}$ points backward instead of forward since the standing leg is pulling the platform to move forward, and $N_{1,Py}$ points downward instead of upward since the standing leg is supporting the platform instead of being supported. The force $F_{1,s}$ is the force exerted by the compression spring on the first leg segment. There is also reaction force $N_{1,4}$ of the fourth leg segment acting at the same point. Force $N_{1,2}$ is the reaction force of the second leg segment at the knee joint of the leg.

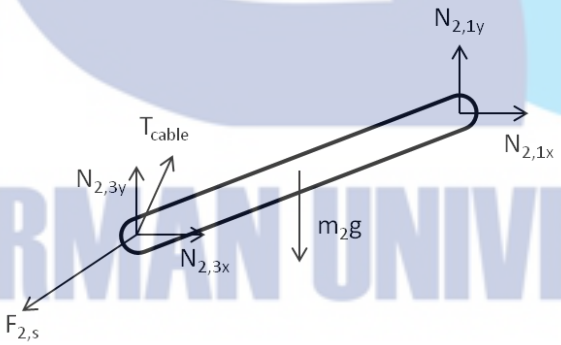

Figure 3.5. Free Body Diagram of the Front Right Leg, Second Segment

Free body diagram of the second leg segment of the same front right leg is shown in Figure 3.5. Force $N_{2,1}$ at the knee joint is the reaction force of the first leg segment. The ankle joint, on the other hand, has reaction force $N_{2,3}$ (reaction force of the third leg segment), tension force of the cable T_{cable}, and spring force $F_{2,s}$ acting on it.

The third segment of the front right leg as shown in Figure 3.6 is the part which has contact with the ground during the leg stance phase, at the toe specifically. When the

toe touches the ground, a normal force $N_{3,g}$ is exerted by the ground at the contact point. If the leg segment tends to move along the ground surface, there would be also friction force $f_{3,g}$ acting on the coinciding surface. However, when the toe is in the air, both the normal ground reaction force and the friction force would not exist. Similarly to the first and second leg segment, the third leg segment also sustains reaction force at its joints. Reaction force $N_{3,4}$ acts on the connecting joint with the fourth leg segment, while reaction force $N_{3,2}$ acts on the connecting joint with the second leg segment.

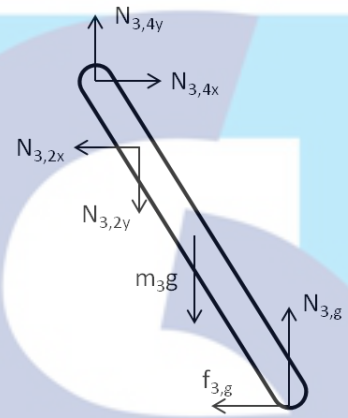

Figure 3.6. Free Body Diagram of the Front Right Leg, Third Segment

Lastly, displayed on Figure 3.7 is the free body diagram of the fourth segment of the front right leg. Force $N_{4,1}$ is the reaction force acting at the joint which connect the fourth leg segment with the first leg segment. Force $N_{4,3}$ is the reaction force acting on the joint which connect the fourth leg segment with the third leg segment.

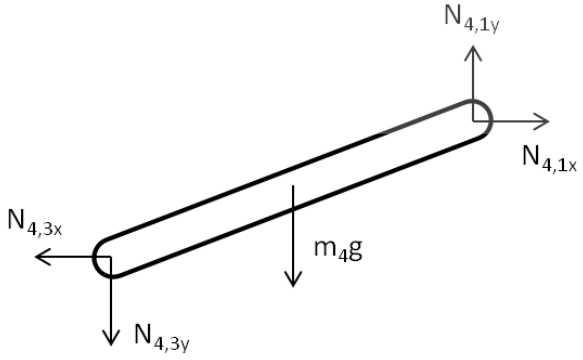

Figure 3.7. Free Body Diagram of the Front Right Leg, Fourth Segment

3.2.2.2. Mechanical Torque Calculation

As motion analysis was excluded from the research, calculations of the required motor torque were performed only in the initial static condition of the quadruped robot. For the purpose of simplifying the necessary mechanical torque, the following postulates were implemented:

1) The chassis plane of the quadruped robot has no roll, pitch, and yaw deviation with respect to the ground plane.
2) Length compression on each leg during the initial static condition is caused by the weight of the robot and ground reaction force—not by retraction through Bowden cable mechanism, thus the strings which connect the cams to the ankle joints of the legs during the initial static condition are loose and exert no force.
3) The linear spring is an ideal spring, that is, there is no limitation in spring dimensioning and the mass is neglected.
4) There is no significant friction on the joints.
5) There is no slip between the motor shaft and the hip joint.
6) Each leg segment is a homogeneous rigid body.
7) The center of gravity of the chassis loaded with the attached components is located at the centroid of the chassis.
8) The weight of individual leg segments is considered as really small compared to the weight of the chassis and the forces exerted on the leg segments, thus the leg segments are mass-less.

Before calculating the mechanical torque, the initial static condition and the ground reaction forces in that condition had to be determined first.

3.2.2.2.1. Determining the Initial Static Condition

Since it was desired that the chassis plane of the robot is parallel to the ground surface on its intial static condition, consequently each hip joints had to be on the same height. The height of the legs h_{leg} is equal to the vertical normalized distance between the hip

joint to the toe of the leg, that is the absolute value of the y component of the toe position vector relative to the hip position ($|r_{t/h}|_y$).

$$h_{leg} = |r_{t/h}|_y \quad \text{..... Eq. (3.10)}$$

Implementing Equation 3.5, 3.7, 3.9 into Equation 3.10 results on Equation 3.11:

$$h_{leg} = (l_1 + l_3)\cos(\theta_h) - l_2\cos(\theta_h + \theta_k) \quad \text{..... Eq. (3.11)}$$

So that the height of the front legs is equal to the height of the hind legs:

$$h_{legF} = h_{legH}$$

$$(l_{F1} + l_{F3})\cos(\theta_{hF}) - l_{F2}\cos(\theta_{hF} + \theta_{kF}) = (l_{H1} + l_{H3})\cos(\theta_{hH}) - l_{H2}\cos(\theta_{hH} + \theta_{kH})$$

where l_{F1} = length of the first segment of the front legs
l_{F2} = length of the second segment of the front legs
l_{F3} = length of the third leg segment of the front legs
θ_{hF} = hip joint angle of the front legs in the initial static condition
θ_{kF} = knee joint angle of the front legs in the initial static condition
l_{H1} = length of the first segment of the hind legs
l_{H2} = length of the second segment of the hind legs
l_{H3} = length of the third segment of the hind legs
θ_{hH} = hip joint angle of the hind legs in the initial static condition
θ_{kH} = knee joint angle of the hind legs in the initial static condition

With assumption that the knee joint angles of all legs are equal ($\theta_{kF} = \theta_{kH} = \theta_k$), the calculation continues as follow:

$$(l_{F1} + l_{F3})\cos(\theta_{hF}) - l_{F2}\cos(\theta_{hF} + \theta_k) = (l_{H1} + l_{H3})\cos(\theta_{hH}) - l_2\cos(\theta_{hH} + \theta_k)$$

$$(l_{F1} + l_{F3})\cos(\theta_{hF}) - (l_{H1} + l_{H3})\cos(\theta_{hH}) = l_{F2}\cos(\theta_{hF} + \theta_k) - l_{H2}\cos(\theta_{hH} + \theta_k)$$

$$(l_{F1} + l_{F3})\cos(\theta_{hF}) - (l_{H1} + l_{H3})\cos(\theta_{hH}) = [l_{F2}\cos(\theta_{hF}) - l_{H2}\cos(\theta_{hH})]\cos(\theta_k) - [l_{F2}\sin(\theta_{hF}) - l_{H2}\sin(\theta_{hH})]\sin(\theta_k)$$

The term on the right hand side has the form of $a\cos x - b\sin x$ and can be converted into the form of $k\cos(x+\varphi)$.

$$a\cos x - b\sin x = k\cos(x+\varphi)$$

$$k = \sqrt{a^2 + b^2}$$

$$\varphi = \tan^{-1}\left(\frac{a}{b}\right)$$

Therefore, for this case:

$$k = \sqrt{[l_{F2}\cos(\theta_{hF}) - l_{H2}\cos(\theta_{hH})]^2 + [l_{F2}\sin(\theta_{hF}) - l_{H2}\sin(\theta_{hH})]^2}$$ .... Eq. (3.12)

$$\varphi = \tan^{-1}\left(\frac{l_{F2}\sin(\theta_{hF}) - l_{H2}\sin(\theta_{hH})}{l_{F2}\cos(\theta_{hF}) - l_{H2}\cos(\theta_{hH})}\right)$$ ..... Eq. (3.13)

The requirement for solving φ is to ensure that the denominator of the equation is not equal to zero.

$$l_{F2}\cos(\theta_{hF}) - l_{H2}\cos(\theta_{hH}) \neq 0$$ ..... Eq. (3.14)

Continuing the calculation:

$$(l_{F1} + l_{F3})\cos(\theta_{hF}) - (l_{H1} + l_{H3})\cos(\theta_{hH}) = k\cos(\theta_k + \varphi)$$

$$\cos(\theta_k + \varphi) = \frac{1}{k}\left((l_{F1} + l_{F3})\cos(\theta_{hF}) - (l_{H1} + l_{H3})\cos(\theta_{hH})\right)$$

Another requirement to solve for θ_k is that the term on the right hand side is inbetween -1 and 1.

$$\left|\frac{1}{k}\left((l_{F1} + l_{F3})\cos(\theta_{hF}) - (l_{H1} + l_{H3})\cos(\theta_{hH})\right)\right| \leq 1$$ ..... Eq. (3.15)

Finally, the value of the knee joint angle of all legs can be determined:

$$\theta_k = \cos^{-1}\left(\frac{1}{k}\left((l_{F1} + l_{F3})\cos(\theta_{hF}) - (l_{H1} + l_{H3})\cos(\theta_{hH})\right)\right) - \varphi \text{.....}$$

..... Eq. (3.16)

The height of the front legs and hind legs can be checked using Equation 3.11.

In conclusion, the sequence of methods for determining the joint angles for the initial static condition is as follows:

1. A combination of hip joint angle of the front legs θ_{hF} and hip joint angle of the hind legs θ_{hH} is determined.
2. The condition of Equation 3.14 is examined; if the expression is not fulfilled, the combination of hip joint angles has to be re-determined.
3. The value of φ is calculated using Equation 3.13.
4. The value of k is calculated using Equation 3.12.
5. The condition of Equation 3.15 is examined; if the expression is not fulfilled, the whole steps starting from the beginning have to be re-done.
6. The knee joint angle is calculated using Equation 3.16.
7. The height of front legs and hind legs is ensured using Equation 3.11.

3.2.2.2.2. Determining Ground Reaction Forces

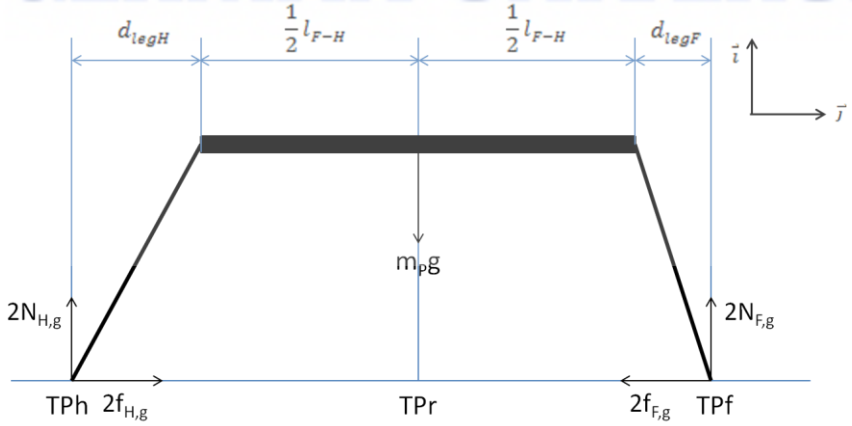

Figure 3.8. Forces Acting on the Quadruped Robot in Static Condition

With the hip and knee joint angles in static condition already determined, the reaction force of the ground at the contact points could be calculated. Since the legs located on the same paracoronal plane have the same hip and knee joint angles, determining ground reaction forces may only consider the sagittal plane of the quadruped robot.

Firstly, the horizontal distance d_{leg} of the toe of a leg to its hip joint is calculated as the value of the x component of the toe position vector relative to the hip position ($\pm|r_{t/h}|_x$).

$$d_{leg} = \pm|r_{t/h}|_x \quad \text{..... Eq. (3.17)}$$

The matter of choosing either positive or negative value of d_{leg} depends on the actual horizontal position of the toe relatively to the hip joint. The value of d_{legF} should be positive if the toe is located in front of the hip joint, and should be negative if the toe is located in back of the hip joint. It is the reverse way for the case of d_{legH}. Positive sign is taken for the value of d_{legH} if the toe is positioned behind the hip joint, while negative sign is taken for the value of d_{legH} if the toe is positioned ahead of the hip joint.

Implementing Equation 3.5, 3.7, 3.9 into Equation 3.17 results on Equation 3.18:

$$d_{leg} = (l_1 + l_3)\sin(\theta_h) - l_2\sin(\theta_h + \theta_k) \quad \text{..... Eq. (3.18)}$$

Applying moment equation $\Sigma M_P = \Sigma \bar{I}\alpha + \Sigma m\bar{a}d$ (Equation 2.1) in static condition would have α and $\bar{a}$ equal to 0, and thus leads to general moment equation in static condition $\Sigma M = 0$.

Using the equation $\Sigma M = 0$ at point TPh:

$$\Sigma M_{TPh} = 0$$
$$2N_{F,g}\left(l_{F-H} + d_{legH} + d_{legF}\right) - m_P g\left(\frac{1}{2}l_{F-H} + d_{legH}\right) = 0$$
$$N_{F,g} = \frac{m_P g\left(\frac{1}{2}l_{F-H} + d_{legH}\right)}{2\left(l_{F-H} + d_{legH} + d_{legF}\right)}$$
$$N_{F,g} = \frac{m_P g\left(l_{F-H} + 2d_{legH}\right)}{4\left(l_{F-H} + d_{legH} + d_{legF}\right)} \quad \text{..... Eq. (3.19)}$$

Applying general force equation in static condition $\Sigma F = 0$:

$$\Sigma F_{rx} = 0$$
$$2f_{H,g} - 2f_{F,g} = 0$$
$$f_{F,g} = f_{H,g} \quad \text{..... Eq. (3.20)}$$

$$\Sigma F_{ry} = 0$$
$$2N_{F,g} + 2N_{H,g} - m_P g = 0$$
$$N_{H,g} = \frac{1}{2}(m_P g - 2N_{F,g})$$
$$N_{H,g} = \frac{1}{2}\left(m_P g - 2\,\frac{m_P g(l_{F-H} + 2d_{legH})}{4(l_{F-H} + d_{legH} + d_{legF})}\right)$$
$$N_{H,g} = \frac{1}{2}m_P g\left(1 - \frac{l_{F-H} + 2d_{legH}}{2(l_{F-H} + d_{legH} + d_{legF})}\right)$$
$$N_{H,g} = \frac{1}{2}m_P g\left(\frac{l_{F-H} + 2d_{legF}}{2(l_{F-H} + d_{legH} + d_{legF})}\right)$$
$$N_{H,g} = \frac{1}{4}m_P g\left(\frac{l_{F-H} + 2d_{legF}}{l_{F-H} + d_{legH} + d_{legF}}\right) \quad \text{..... Eq. (3.21)}$$

In order to ensure static condition in x direction, slipping is assumed to be about to happen on the pair of legs which sustain smaller ground reaction force. The reason is that the value of static friction coefficient μ_s is assumed to be identical for the both pair of legs. With same value of μ_s, it is impossible for the pair of legs with smaller ground reaction force to gain friction as big as the pair of legs with bigger ground reaction force.

$$f_g = \mu_s N_{smaller} \quad \text{..... Eq. (3.22)}$$

Implying Equation 3.18:

$$f_{F,g} = f_{H,g} = f_g \quad \text{..... Eq. (3.23)}$$

Therefore, the steps for determining the ground reaction forces acting on each leg is listed as follow:

1. The horizontal distance between the hip joint and the toe of front pair of legs (d_{legF}) and of hind pair of legs (d_{legH}) is calculated using Equation 3.18.

2. The normal force acting on the front pair of legs $N_{F,g}$ is determined by applying Equation 3.19.
3. The normal force acting on the hind pair of legs $N_{H,g}$ is determined via Equation 3.21.
4. Equation 3.22 and 3.23 are implemented to obtain the friction force $f_{F,g}$ on the front pair of legs and $f_{H,g}$ on the hind pair of legs.

3.2.2.2.3. Determining the Required Mechanical Torque on the Hip Joint

Since the quadruped robot is in static condition, when not investigating the internal forces, it is possible to consider a whole leg as a rigid body without separating the leg segments. Figure 3.9 shows the free body diagram of a whole front leg and a whole hind leg. Agreeing to the assumptions in Section 3.2.2.2 that the cable is loose and that the leg is massless, the cable tension forces and the weight forces are on purpose not drawn on the figure.

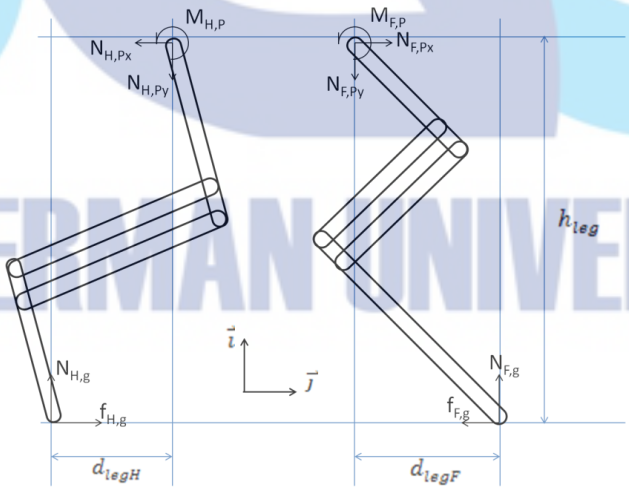

Figure 3.9. Free Body Diagram of One Whole Leg as One Rigid Body: Front Leg (right) and Hind Leg (left)

The remaining forces drawn on the diagram are the ground and platform reaction forces. The torques applied by the hip actuators on the hip joint are also shown in the diagram. The ground reaction forces $N_{F,g}$ and $N_{H,g}$ are acting on the front pair of legs

and hind pair of legs respectively. The friction forces of the front and hind legs are denoted by $f_{F,g}$ and $f_{H,g}$. Values of the ground reaction forces and friction forces are obtained using the method described previously in Section 3.2.2.2.2. Reaction forces $N_{F,P}$ and $N_{H,P}$ are exerted by the platform on the hip joint of the front legs and hind legs. The required mechanical torque on each hip joint is symbolized with $M_{F,P}$ and $M_{H,P}$ for the mechanical torque on front hip joints and hind hip joints.

Applying moment equation in static condition $\Sigma M = 0$ at the hip joint of the front leg is done to determine the required mechanical torque to be exerted by the hip actuator of the front legs:

$$\Sigma M_{Fh} = 0$$
$$M_{F,P} + N_{F,g} d_{legF} - f_{F,g} h_{leg} = 0$$
$$M_{F,P} = f_{F,g} h_{leg} - N_{F,g} d_{legF} \quad \text{..... Eq. (3.24)}$$

Similarly, the equation is employed at the hip joint of the hind leg to calculate the necessary mechanical torque supplied by the hip actuator of the hind legs:

$$\Sigma M_{Hh} = 0$$
$$M_{H,P} + N_{H,g} d_{legH} - f_{H,g} h_{leg} = 0$$
$$M_{H,P} = f_{H,g} h_{leg} - N_{H,g} d_{legH} \quad \text{..... Eq. (3.25)}$$

3.3. Gait Locomotion Design Justification

Each gait of the robot is defined by the parameters of its two profiles: joint angle profile and hip joint height profile. These joint angle profiles are applied by equal means to each limb of the robot. The phase offset differences of each limb compared to each other on different kinds of gait will follow the definition stated in [4].

Walking gait has phase offset differences of a quarter period among the limbs: front-left, hind-left, front-right and hind-right in descending order (the most leading to the most lagging). Trotting gait has one pair of diagonal limbs moving in a phase, while the other pair differs by a half period. Finally, bounding gait has similar phase

differences as trotting gait, but the limbs are paired front-to-front and hind-to-hind instead of diagonally paired.

The phase offset differences of the three common gaits of quadrupedal robot on different limbs is written on Table 3.1 as fraction of one leg cycle period.

Table 3.1. Phase Offset Differences on Common Quadrupedal Gaits

Gait	Front Left	Front Right	Hind Left	Hind Right
Walking	0	0.5	0.75	0.25
Trotting	0	0.5	0.5	0
Bounding	0	0	0.5	0.5

One stride cycle of one quadruped leg is divided into two phase: swing phase and stance phase. While the leg is in the swing phase, the leg is expected to be swinging on the air, not touching the ground. On the contrary, in the stance phase the particular leg should be stepping flat on the ground. Transition from stance phase to swing phase is indicated by the occurrence of toe-off—the leg position when the tip of the leg is just about leaving the ground. Its counterpart, the transition from swing phase to stance phase, is shown by the occasion of touch-down—the leg position when the foot-toe segment is just positioned flat on the ground. As the summary, the sequence of one quadruped leg stride cycle consists of toe-off, swing phase, touch-down, and stance phase. The sequence repeats as the cycle continues.

Joint angle profile consists of the hip joint angle profile and the knee joint angle profile. Referring to the joint angle profile of the Cheetah-cub in [1] and the observation of hip and knee joint angle of cat in [5], the profile is approximated as sinusoidal wave with different frequency between during its swing phase and during its stance phase.

The illustration of the hip and knee joint angle profile along with the parameters defining the profile is shown on Figure 3.10. Time scale and joint angle are represented by the x and y axis respectively. The upper graph corresponds to one cycle of hip joint angle profile, while the lower graph corresponds to one cycle of

knee joint angle profile. Depicted starting from the swing phase, continuing to the stance phase and repeating the same pattern of their cycle, the joint angle profile oscillates similarly to sinusoidal wave.

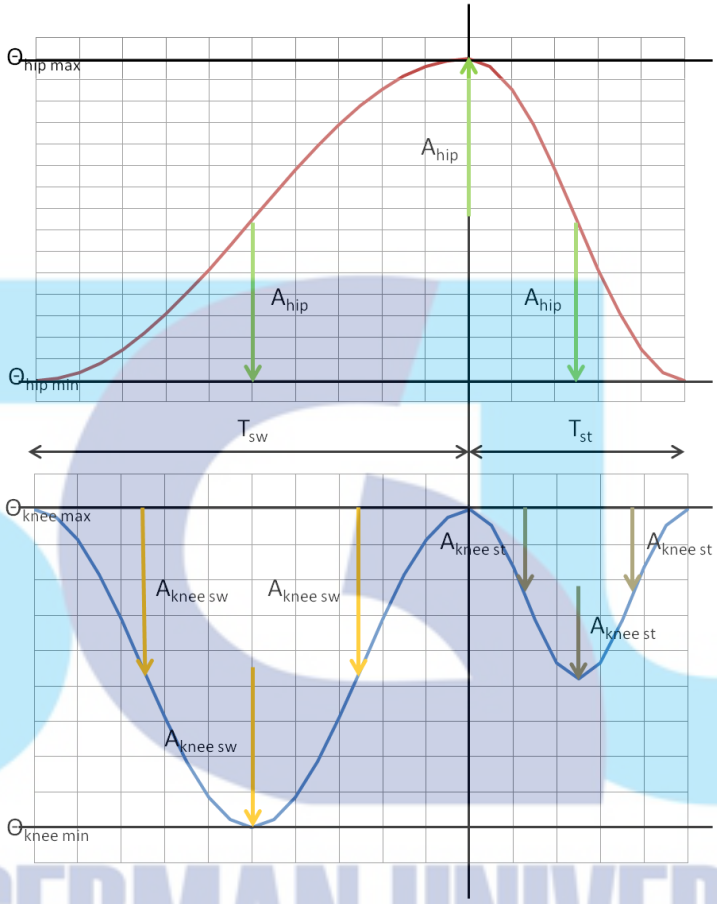

Figure 3.10. Illustration of Hip and Knee Joint Angle Profile

Based on Section 2.2.3, dynamic balance of the quadruped robot may be achieved by ensuring that the tipping rate of the robot is smaller than its stepping rate. Therefore, the gait analysis would be done by examining the toes position of the robot relative to the hips. Vertical length corresponds to the tipping rate and horizontal length corresponds to the stepping rate.

A gait simulator program, *GaitSimulator*, was designed and developed for simulating the resulting gait of the variety of parameters and calculating the toes positions relative to the hips on every time segmentation. Supplied with gait parameters as the input, the program would output the simulated image of the resulting gait represented

by stick figures and text files containing the vertical and horizontal components of hip-toe length. Graphical observation would then be possible. More details on *GaitSimulator* program are included in Section 3.4.5.2.1.

3.3.1. Hip Joint Angle Profile

At the beginning of swing phase, during toe-off, the hip joint angle (θ_{hip}) is at its lowest position. As the particular quadruped leg is swinging forward, the hip joint angle is also increasing gradually until it reaches its peak when the leg touches down the ground. The leg enters stance phase and the leg is swinging backward. The hip joint angle decreases accordingly until it goes back to its lowest position at the end of stance phase. The stride cycle repeats and so does the hip joint angle.

The rise and fall of the hip joint angle during swing and stance phase are the same in magnitude but different in oscillation frequency. The oscillation amplitude of the hip joint angle (A_{hip}) is half of the difference between the maximum and minimum angle position ($\theta_{hip\,max}$ and $\theta_{hip\,min}$). Both the hip joint swing and stance period ($T_{hip\,sw}$ and $T_{hip\,st}$) is two times the leg swing and stance period (T_{sw} and T_{st}), respectively.

Hip joint angle profile as a function of time, along with its first and second time derivatives are defined as follows:

a) During swing phase ($0 \leq t < T_{sw}$):

$$\theta_{hip}(t) = \theta_{hip\,max} - A_{hip}\left[1 + \cos\left(\omega_{hip\,sw}\, t\right)\right]$$

$$\dot{\theta}_{hip}(t) = A_{hip}\, \omega_{hip\,sw}\, \sin\left(\omega_{hip\,sw}\, t\right)$$

$$\ddot{\theta}_{hip}(t) = A_{hip}\, \left(\omega_{hip\,sw}\right)^2\, \cos\left(\omega_{hip\,sw}\, t\right)$$

b) During stance phase ($T_{sw} \leq t < T_{sw} + T_{st}$):

$$\theta_{hip}(t) = \theta_{hip\,min} + A_{hip}\left[1 + \cos\left(\omega_{hip\,st}\, t - \pi\, \frac{\omega_{hip\,st}}{\omega_{hip\,sw}}\right)\right]$$

$$\dot{\theta}_{hip}(t) = -A_{hip}\, \omega_{hip\,st}\, \sin\left(\omega_{hip\,st}\, t - \pi\, \frac{\omega_{hip\,st}}{\omega_{hip\,sw}}\right)$$

$$\ddot{\theta}_{hip}(t) = -A_{hip}\, \left(\omega_{hip\,st}\right)^2\, \cos\left(\omega_{hip\,st}\, t - \pi\, \frac{\omega_{hip\,st}}{\omega_{hip\,sw}}\right)$$

3.3.2. Knee Joint Angle Profile

The knee joint angle (θ_{knee}) is at its maximum value at toe-off. This knee joint angle narrows as the quadruped leg is swinging forward in order to shorten the distance between the hip joint and foot joint, so that the leg does not hit the ground. It reaches the lowest point at the middle of swing phase, and then rises again back to the highest point at touch-down. During stance phase, the knee joint angle dips gradually with the middle of the stance phase as its turning point before returning to the default position at the end of stance phase.

The oscillation of the knee joint angle has different oscillation amplitude during its swing and stance phase. The oscillation amplitude of the knee joint angle throughout swing phase ($A_{knee\ sw}$) is half of the difference between the maximum and minimum knee joint angle position ($\theta_{knee\ max}$ and $\theta_{knee\ min}$). No specific criterion has been concluded for determining the oscillation amplitude over the stance phase ($A_{knee\ st}$), but it is typically lower than the one during swing phase. Both the knee joint swing and stance periods ($T_{knee\ sw}$ and $T_{knee\ st}$) are equal to the leg swing and stance periods (T_{sw} and T_{st}), respectively.

Knee joint angle profiles as a function of time, along with its first and second time derivatives are defined as follows:

a) During swing phase ($0 \leq t < T_{sw}$):

$$\theta_{knee}(t) = \theta_{knee\ min} + A_{knee\ sw}[1 + \cos(\omega_{knee\ sw}\ t)]$$

$$\dot{\theta}_{knee}(t) = -A_{knee\ sw}\ \omega_{knee\ sw}\ \sin(\omega_{knee\ sw}\ t)$$

$$\ddot{\theta}_{knee}(t) = -A_{knee\ sw}\ (\omega_{knee\ sw})^2\ \cos(\omega_{knee\ sw}\ t)$$

b) During stance phase ($T_{sw} \leq t < T_{sw} + T_{st}$):

$$\theta_{knee}(t) = \theta_{knee\ max} - A_{knee\ st}\left[1 + \sin\left(\omega_{knee\ st}\ t - \frac{\pi}{2} - 2\pi\ \frac{\omega_{knee\ st}}{\omega_{knee\ sw}}\right)\right]$$

$$\dot{\theta}_{knee}(t) = -A_{knee\ st}\ \omega_{knee\ st}\ \cos\left(\omega_{knee\ st}\ t - \frac{\pi}{2} - 2\pi\ \frac{\omega_{knee\ st}}{\omega_{knee\ sw}}\right)$$

$$\ddot{\theta}_{knee}(t) = A_{knee\ st}\ (\omega_{knee\ st})^2\ \sin\left(\omega_{knee\ st}\ t - \frac{\pi}{2} - 2\pi\ \frac{\omega_{knee\ st}}{\omega_{knee\ sw}}\right)$$

3.4. Platform Design Justification

This section describes the considerations taken in designing the actual mechatronics platform of the quadruped robot. Mechanical, electrical, communication and software system build the mechatronics platform of the robot. The interconnection of the constructing components is described on Section 3.4.1 and shown on Figure 3.11.

3.4.1. Overview of the Whole System

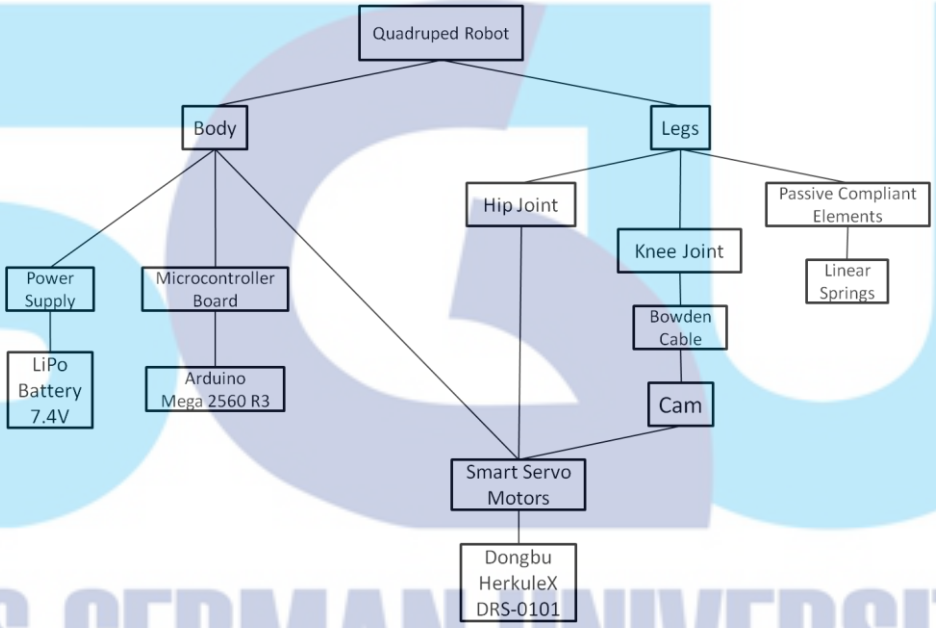

Figure 3.11. Overview of the Whole System

The final result of the project was targeted to be a four legged, trotting quadruped mobile robot similar to Cheetah-cub, but controlled by means of sinusoidal gait instead of Central Pattern Generator (CPG).

The physical form of the quadruped robot consists of the body and the four segmented legs. The body held the electronic components of the robot: a battery as power supply, a microcontroller board, and 8 smart servo motors. Each of the four legs has a hip joint, a knee joint, and a linear spring as passive compliant element. The hip joint is actuated directly by the servo attached onto it, while the knee joint is connected via Bowden cable mechanism to a cam, which is mounted to another servo.

When the robot is turned on, the microcontroller will execute the program for driving the smart servo motors. Smart servo motors, which are connected to the hip and knee joints of the quadruped robot, produce the robot movement. Arrays of pre-calculated setpoints are embedded in the program uploaded to the microcontroller board.

3.4.2. Mechanical System Design

The mechanical structure of the mobile robot was designed based on the mechanical structure of the existing Cheetah-cub Quadruped Robot with some necessary adaptation. The four legs of the robot are segmented. Two of the joints—the hip joint and the knee joint—are actuated for generating the gait of the robot. The hip joint is mounted directly to a servo motor, while the knee joint is connected through a pulley system to a cam mounted to another servo motor. The leg is also equipped with passive compliant element in the form of a linear spring as the antagonist of the pulley system and the load compressing the leg. Electronic components of the robot are loaded on the body of the robot, which consists of a lower platform, a backbone, and an upper platform.

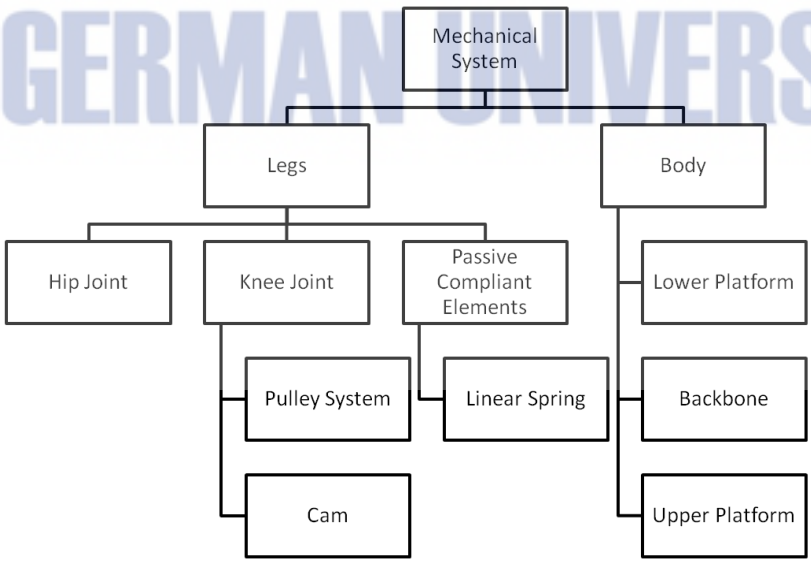

Figure 3.12. Components of the Mechanical System

3.4.2.1. Legs

Each leg of the quadruped robot consists of three main leg segments: hip-knee segment, knee-ankle segment and ankle-foot segment as illustrated in Figure 3.13. The three joints on the leg are hip joint, knee joint and ankle joint. Two actuators—two servo motors—are actively affecting the dynamics of every leg. One motor controls the hip joint angle, while the other controls the knee joint angle.

Hip joint of the leg is mounted to a servo motor. Rotation of the particular servo motor is thus directly transferred to the hip joint. However, this is not the case for the knee joint actuation. Attaching the motor directly to the knee joint would result on considerable inertia on the said joint. Avoiding this inertia problem is accomplished by attaching the knee joint actuator on the robot chassis, and then the motor is connected to the leg via a rope-and-pulley system which operates similarly to Bowden cable mechanism.

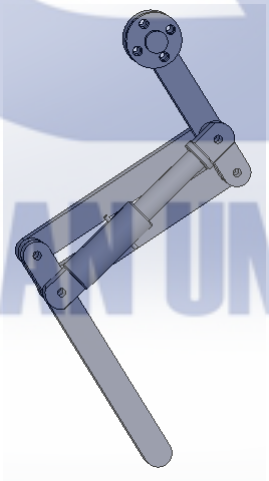

Figure 3.13. Leg Illustration of the Quadruped Robot

The rope-and-pulley system is used to control the knee joint angle. One end of a nylon string is tied to the ankle joint of the leg, while the other one is tied to a cam mounted to the servo motor which act as the actuator. The string is pivoted at the hip joint over

an idler attached on the hip. As the cam rotates with the motor and pulls the string, the string also pulls the ankle joint and alters the knee joint angle.

In addition to the servo motors which actively actuate the legs, the legs are also equipped with passive compliant element, which is a linear spring for each leg. The linear spring is attached at hip-knee segment and at the ankle joint diagonally. It acts as the antagonist of the Bowden cable. The primary role of the Bowden cable is to narrow the knee joint angle, while this spring will be compressed (Figure 3.14). When the spring tries to maintain its relaxed state, it exerts tension force on the hip-knee segment and the ankle joint, and consequently tends to keep the knee joint angle at its default value (Figure 3.15)

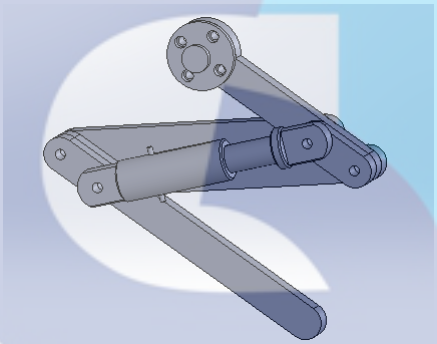

Figure 3.14. Leg in Fully-Compressed Condition

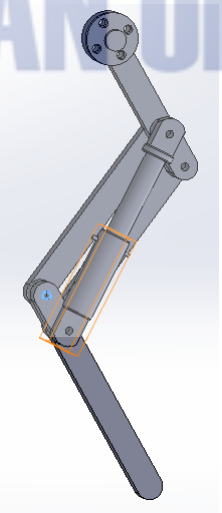

Figure 3.15. Leg in Fully-Extended Condition

On the previous design, another linear spring was intended to be attached at the hip-knee segment at the same point as the linear spring, but the other end is attached to the upper end of the ankle-foot segment. It is supposed to be in parallel with the knee-ankle segment. The function of this parallel spring is to maintain the pantograph between hip-knee segment and ankle-foot segment, that is, to keep the two leg segments parallel to each other. However, after further consideration for achieving less complicated design, the intended usage of parallel spring was then omitted and replaced by a rigid leg segment identical to the knee-ankle segment.

3.4.2.2. Body

The body of the robot is a platform for storing the electrical system of the mobile robot. The components to be supported on the platform are the microcontroller board, servo motors and the battery. It was firstly designed as a single 3mm-thick acrylic sheet with the necessary holes for attaching components. However, the first assembly attempt of the mechanical platform showed that the acrylic sheet was slightly bent. The design of the chassis was then altered to be consisted of a lower platform, a backbone and an upper platform stacked and bolted together, as shown in Figure 3.16. The purpose is to reduce the effect of bending moment on the whole chassis.

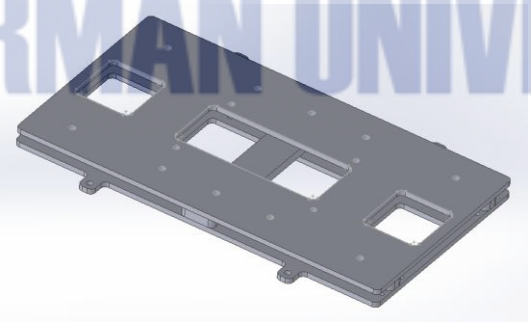

Figure 3.16. Assembled Platform Set

The lower platform (Figure 3.17) is very similar to the initial platform design (a single acrylic sheet, 3mm-thick). There are also holes for attaching the servo motors, string aligners, spacers, and the other parts of the chassis.

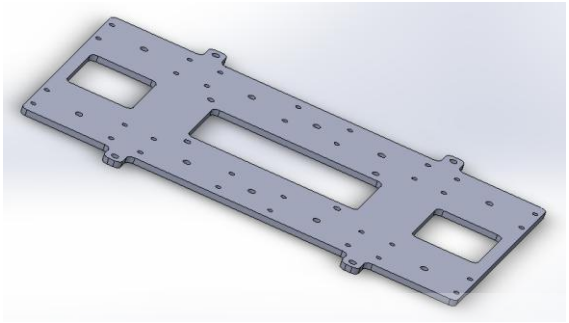

Figure 3.17. Lower Platform

The upper platform (Figure 3.18) is vaguely a simpler cut of the lower platform—also cut of a single 3mm-thick acrylic sheet, but with fewer holes, since the servo motors and string aligners are only attached to the lower platform.

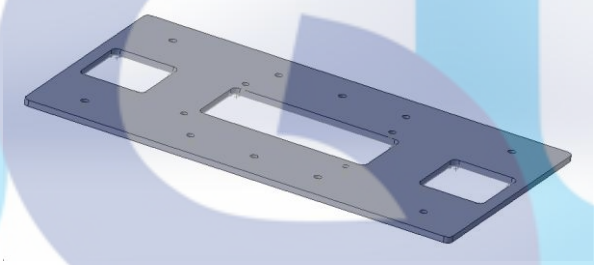

Figure 3.18. Upper Platform

The backbone, as seen on Figure 3.19, has its shape alike an H alphabet. It is cut of a 5mm-thick acrylic sheet, intentionally thicker than the lower and upper platform. Bolting the lower and upper platform to the backbone with more thickness provides forces opposite to the gravitational weight acted on the chassis, thus reduces the critical bending moment acting on the whole chassis.

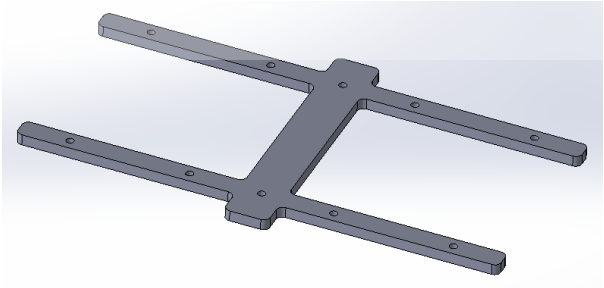

Figure 3.19. Backbone

3.4.3. Electrical System Design

The electrical system of the mobile robot consists of a power supply, a microcontroller board, and smart servo motors.

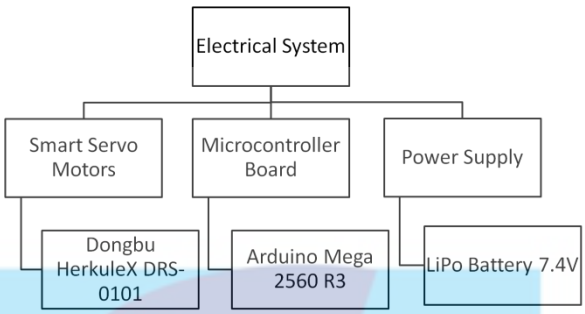

Figure 3.20. Components of the Electrical System

3.4.3.1. Smart Servo Motor Selection

General servo motors receives PWM signals from the actuator controller and will rotate depends on the signal. While rotating, the motors form the angle of the joints in the mobile robot, thus forming the movement of the legs of the mobile robot. As smart servo motors, the motors are capable of providing their internal feedback to be accessed outside the internal system of the motors and communicating with the controller by transmitting and receiving data packets via serial communication. Referring to Section 2.2.5, Dynamixel AX-12A and Dongbu Herkulex DRS-0101 are the two different smart servo motors to be considered.

It is explained on Section 3.4.3.2 that the chosen microcontroller board is Arduino Board. The communication protocol of Arduino Board is Full-Duplex. Therefore, using Dynamixel servos, which communication protocol is Half-Duplex, would require additional circuit to convert from Half-Duplex Protocol into Full-Duplex Protocol before being connected to the Arduino Board. Meanwhile, HerkuleX servos, which communication protocol is Full-Duplex already, can be connected directly to the board.

Table 3.2 shows some basic information regarding the specification of Dongbu HerkuleX DRS-0101:

Table 3.2. Specification of Dongbu HerkuleX DRS-0101

Feature	Description
Size	44.5mm(W) × 24.0mm(D) × 32.0mm(H)
Weight	45g
Input Voltage	7-12V DC, optimum at 7.4V DC
Stall Torque	12kgf·cm@7.4V (equivalent to 1.2N·m)
Maximum Speed	0.166s/60° @7.4V (equivalent to 360°/s)
Operating Angle	320° (for position control mode) or continuous rotation
Communication	Full Duplex Asynchronous Serial (TTL), Multi Drop, Max 0.67Mbps
Feedback	Absolute position, speed (PWM duty cycle), temperature, input voltage

3.4.3.2. Microcontroller Board Selection

Arduino Board was chosen as the microcontroller board to be implemented on the quadruped robot design due to its flexibility and ease of use as have been explained on Section 2.2.6.

Multiple HerkuleX Smart Servo Motors can be connected to one single bus, and this single bus will occupy only one set of UART Serial Port on the Arduino board. Thus, Arduino UNO, which has one set of UART Serial Port seems adequate for controlling the 8 servo motors. However, as will be explained in Section 3.4.4, due to the control complexity of using single communication bus for more than one kind of device, Arduino Mega 2560 R3 was chosen because it is a more powerful microcontroller board with more available communication ports.

Table 3.3 shows the summary of Arduino Mega 2560 R3 specifications:

Table 3.3. Specification of Arduino Mega 2560 R3

Feature	Description
Microcontroller	ATmega2560
Operating Voltage	5V
Recommended Input Voltage	7-12V
Digital I/O Pins	54
Analog Input Pins	16
Hardware Serial Ports	4
Clock Speed	16MHz

3.4.3.3. Power Supply Selection

The power supply is providing electrical power for the microcontroller board and servo motors.

Since the robot was aimed to be mobile, a lithium polymer (Li-Po) battery was chosen as the power supply. Li-Po batteries are relatively light-weighted, small, easy to obtain, provided with high energy capacity and discharge rate. They are commonly used in radio-controlled vehicles. This battery has no memory effect, thus gives no problem in recharging.

Figure 3.21. Gens ace High Discharge LiPo Battery

Output voltage of a single cell Li-Po battery is 3.7 V, thus the Li-Po battery packages available on market have output voltage in multiplication of 3.7 V. A 2-cells Li-Po battery package will provide voltage of 7.4 V, that is optimal for the HerkuleX DRS-0101 servo and still inside the voltage range of Arduino Board Mega 2560 R3 (7 to 12 V).

The specification of the battery is listed on Table 3.4.

Table 3.4. Specification of Gens ace 2200mAh 7.4V 25C 2S1P LiPo Battery

Feature	Description
Size	82 mm × 34 mm × 21 mm
Weight	134g
Rated Output Voltage	7.4V
Capacity	2200 mAh
Maximum Continuous Discharge Rate	25C (equivalent to 55A)
Maximum Burst Discharge Rate	50C (equivalent to 110A)

3.4.3.4. Electrical Circuit Diagram

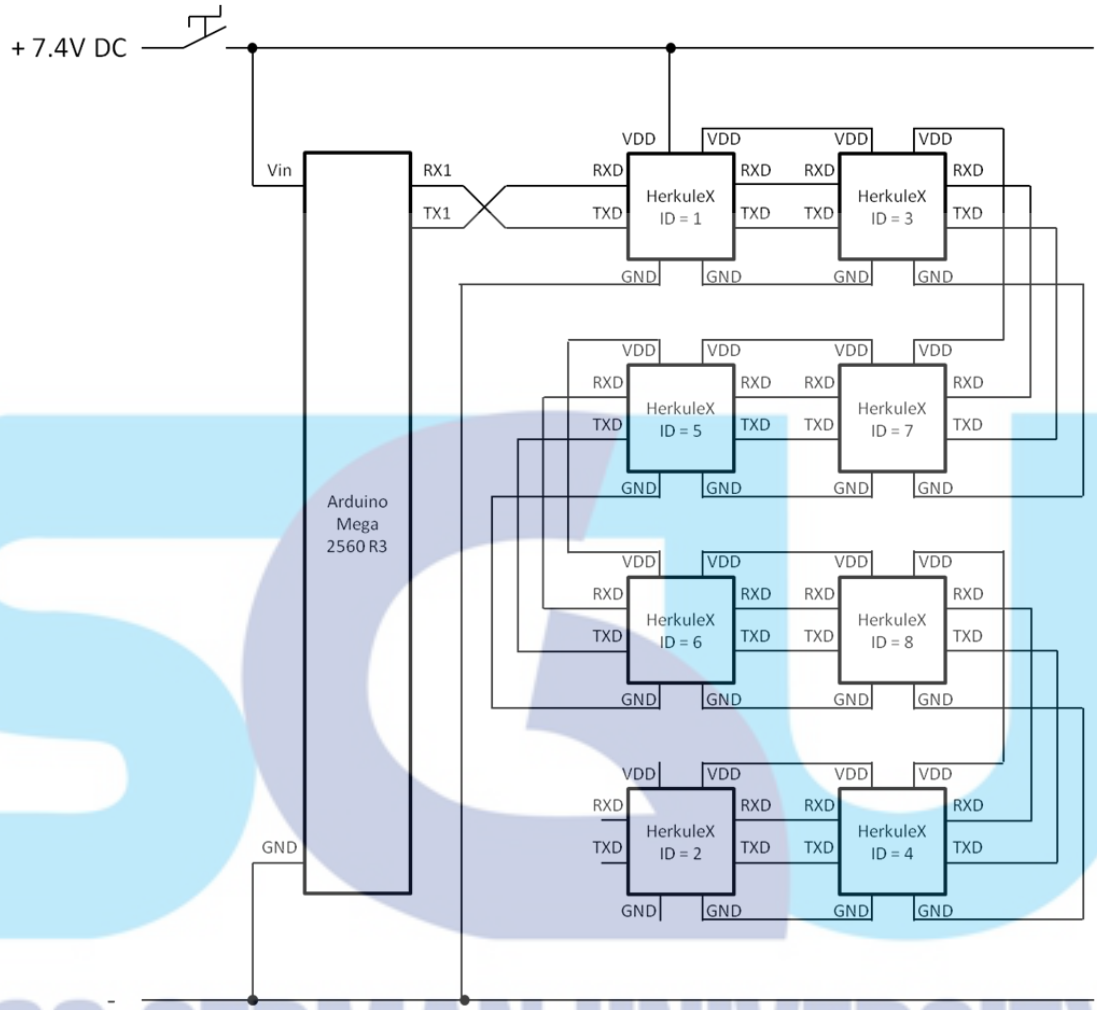

Figure 3.22. Electrical Circuit Diagram of the Quadruped Robot

The electrical circuit diagram of the quadruped robot is shown by Figure 3.24. A 7.4V LiPo Battery supplies electrical power to the Arduino board and the eight HerkuleX servo motors. Connections between the Arduino board and the HerkuleX servos can be done directly since they are operating at the same TTL level with a cross connection (RX pin of the Arduino board to the TX pin of the HerkuleX servo, and TX pin of the Arduino board to the RX pin of the HerkuleX servo). Only one serial port on the Arduino board is needed, since multi-drop connection is possible for the HerkuleX servo motors. The eight servo motors are all connected to this single port as depicted on the figure.

3.4.4. Communication System Design

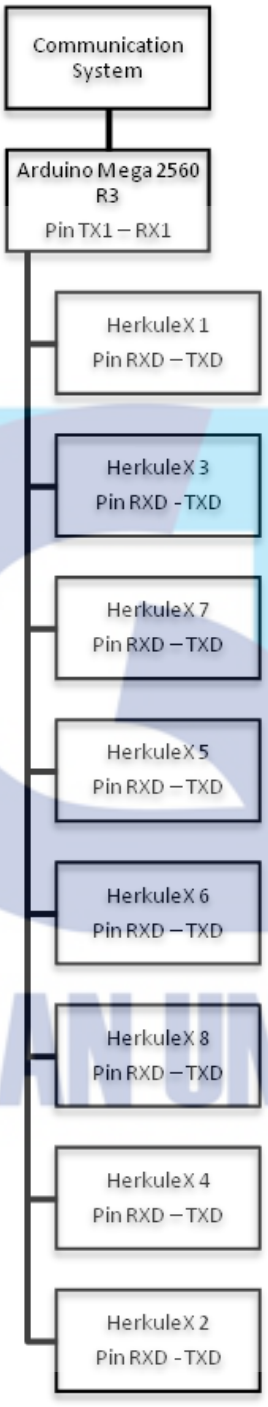

Figure 3.23. Components of the Communication System

The communication system of the quadruped robot is illustrated as a single chain of nodes serially connected (Figure 3.24). The nodes represent the microcontroller board

and HerkuleX servo motors as the hip and knee actuators. Section 3.4.3.4 has stated that the Arduino board and the eight HerkuleX servos are connected on one single bus. HerkuleX already has its own communication protocol, in which data packets are transmitted and received through the communication bus. Some libraries for interfacing the communication between Arduino board and HerkuleX servo by utilizing the aforementioned have been developed and openly distributed on the Internet.

3.4.5. Software System Design

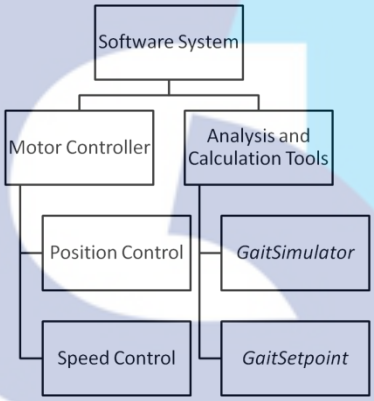

Figure 3.24. Components of the Software System

The motor controller module, which controls the movement of the robot, is packed into the Arduino microcontroller board and consists of position and speed control mode. The analysis and calculation tools for determining the setpoints of the servos to be driven, however, are built in a personal computer and not embedded into the microcontroller board.

GaitSimulator is a tool for simulating gaits with various parameters, thus it assists the process of decide on suitable set of gait parameters. *GaitSetpoint*, on the other hand, aids the process of implementing the preferred gait to the actual quadruped robot. Through different types of approximation, it generates series of setpoints for compensating the ability of servo motors used, but still maintain the gait performance to certain level.

3.4.5.1. Motor Controller

The motor controller is responsible for sending command packets to drive the servo motors. It is provided with an array of time setpoints and another array of either angular position or angular velocity setpoints. Those setpoints are calculated beforehand using *GaitSetpoint* program, which is discussed thoroughly in Section 3.4.5.2.2. One controller is developed for each position control mode and speed control mode, based on the selected gait approximation method. The details regarding gait approximation are explained in Section 3.5.

The motor controller is then uploaded to the Arduino microcontroller board. HerkuleX library for Arduino built by Alessandro Giacomel is implemented on the code for the sake of convenience in commanding the servos. The commands supported by the library are listed on Table 3.5.

Table 3.5. List of Commands Supported by HerkuleX Library for Arduino (1)

Function	Description
void Herkulex::begin(long baud, int rx, int tx)	Start the communication between Arduino Uno/2009 and HerkuleX through software serial port
void Herkulex::beginSerial1(long baud)	Start the communication between Arduino Mega and HerkuleX through Serial Port 1
void Herkulex::beginSerial2(long baud)	Start the communication between Arduino Mega and HerkuleX through Serial Port 2
void Herkulex::beginSerial3(long baud)	Start the communication between Arduino Mega and HerkuleX through Serial Port 3
void Herkulex::end()	Stop the communication between Arduino and HerkuleX
void Herkulex::initialize()	Initialize all the motors
byte Herkulex::stat(int servoID)	Request for the status of the servo
void Herkulex::ACK(int valueACK)	Set the behavior of the motors towards a request
void Herkulex::set_ID(int ID_Old, int ID_New)	Change the ID of the motor
void Herkulex::clearError(int servoID)	Clear the motor error register(?)
void Herkulex::torqueON(int servoID)	Set the motor torque to ON
void Herkulex::torqueOFF(int servoID)	Set the motor torque to OFF

Table 3.6. List of Commands Supported by HerkuleX Library for Arduino (2)

Function	Description
void Herkulex::moveAll(int servoID, int Goal, int iLed)	Prepare the motors to execute movement towards certain absolute position in unison
void Herkulex::moveAllAngle(int servoID, float angle, int iLed)	Prepare the motors to execute movement towards certain absolute angle in unison
void Herkulex::moveSpeedAll(int servoID, int Goal, int iLed)	Prepare the motors to execute movement towards certain continuous rotation speed in unison
void Herkulex::actionAll(int pTime)	Execute the unison movement
void Herkulex::moveSpeedOne(int servoID, int Goal, int pTime, int iLed)	Command the motor to execute movement towards certain continuous rotation speed
void Herkulex::moveOne(int servoID, int Goal, int pTime, int iLed)	Command the motor to execute movement towards certain absolute position
void Herkulex::moveOneAngle(int servoID, int Goal, int pTime, int iLed)	Command the motor to execute movement towards certain absolute angle
int Herkulex::getPosition(int servoID)	Get the absolute position of the motor
float Herkulex::getAngle(int servoID)	Get the absolute angle of the motor
int Herkulex::getSpeed(int servoID)	Get the continuous rotation speed of the motor
void Herkulex::reboot(int servoID)	Reboot the motor
void Herkulex::setLed(int servoID, int valueLed)	Set the LED color of the motor
void Herkulex::writeRegistryRAM(int servoID, int address, int writeByte)	Write a value in a register in the RAM of the motor
void Herkulex::writeRegistryEEP(int servoID, int address, int writeByte)	Write a value in a register in the EEPROM of the motor

Additional commands are also added to the library for reading the value in the registers of the motor, listed on Table 3.7.

Table 3.7. List of Additional Commands Added to the Library

Function	Description
void Herkulex::readRegistryRAM(int servoID, int address)	Read a value in a register in the RAM of the motor
void Herkulex::readRegistryEEP(int servoID, int address)	Read a value in a register in the EEPROM of the motor

3.4.5.1.1. Position Control Mode

Position control mode of the motor controller utilizes the position control of the HerkuleX servo. The inputs to the motor controller of this mode are two arrays of setpoints: an array of time setpoints and an array of angular position setpoints. This control mode corresponds to gait approximation through position control (Section 3.5.1).

When the movement of quadruped robot is initiated, the servos are first set to their respective initial position. Their angular positions are adjusted to the first angular position setpoints at the beginning of the gait according to the different phase offsets of each leg. While the continuous condition is fulfilled, the next position setpoints would be executed, otherwise the quadruped robot would stop the movement cycle.

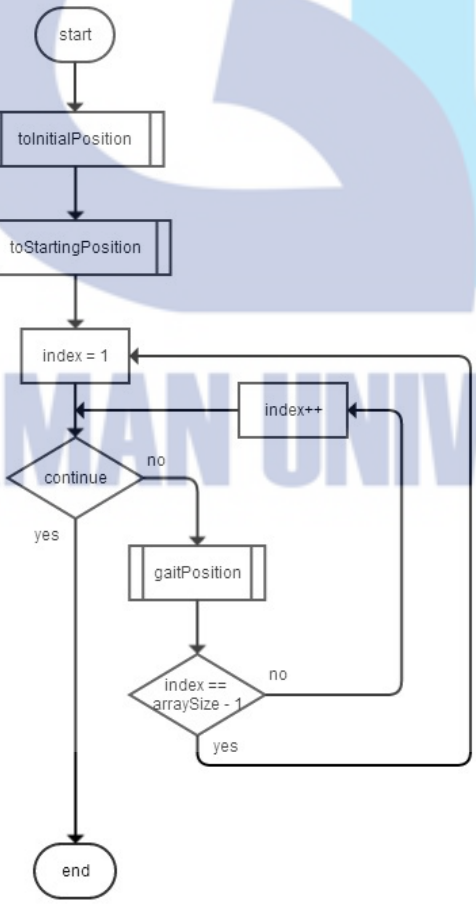

Figure 3.25. Flowchart of the Motor Controller with Position Approximation

3.4.5.1.2. Speed Control Mode

Speed control mode of the motor controller exploits the continuous rotation speed control of the HerkuleX servo. The reasoning of considering rotation speed control in addition to angular position control is explained in Section 3.5.2. Two arrays of setpoints are fed to the controller: an array of time setpoints and an array of angular velocity setpoints (instead of angular position setpoints). This control mode corresponds to gait approximation through speed control (Section 3.5.2).

The continuous rotation speed control mode of HerkuleX DRS-0101 works based on the pulse width modulation (PWM) desired by the user. Duty cycle percentage of 0% to 100% is represented by value of 0 to 1023 for counterclockwise rotation or 0 to -1023 for clockwise rotation. However, the resulting rotation speed will be affected by the input voltage applied on the motor. The quadruped robot will be using battery as its source of electrical power, but the battery voltage may vary at different times, either higher than the rated voltage after being fully charged, or lower than the rated voltage after some period of usage. This might lead to inconsistency of joint angle profile and affect the resulting gait, while it is preferred that the angular velocity setpoints can be achieved despite the change in input voltage. Therefore, it is necessary to investigate the relationship among the input voltage, PWM duty cycle and the resulting angular velocity (in degrees per second or radians per second). Mathematically, the resulting angular velocity ω will be regarded as a function of the input voltage V_{in} and PWM duty cycle p.

$$\omega = f(V_{in}, p)$$

Motor test is conducted to examine the function *f*. Details of the motor test plan are described on Section 3.6.4. Depending on the behavior of the function, given the information of input voltage value and desired angular velocity value, it might be possible to determine the necessary PWM duty cycle to be implemented. The function g is then used as the basis of PWM value calculation.

$$p = g(V_{in}, \omega)$$

Where p = value representing the PWM duty cycle of the HerkuleX servos (from -1023 to 1023) as function g of input voltage and desired angular velocity

V_{in} = the input voltage of the servos (Volt)

ω = the desired angular velocity (degree/s)

The work sequence of the motor controller is displayed on Figure 3.27.

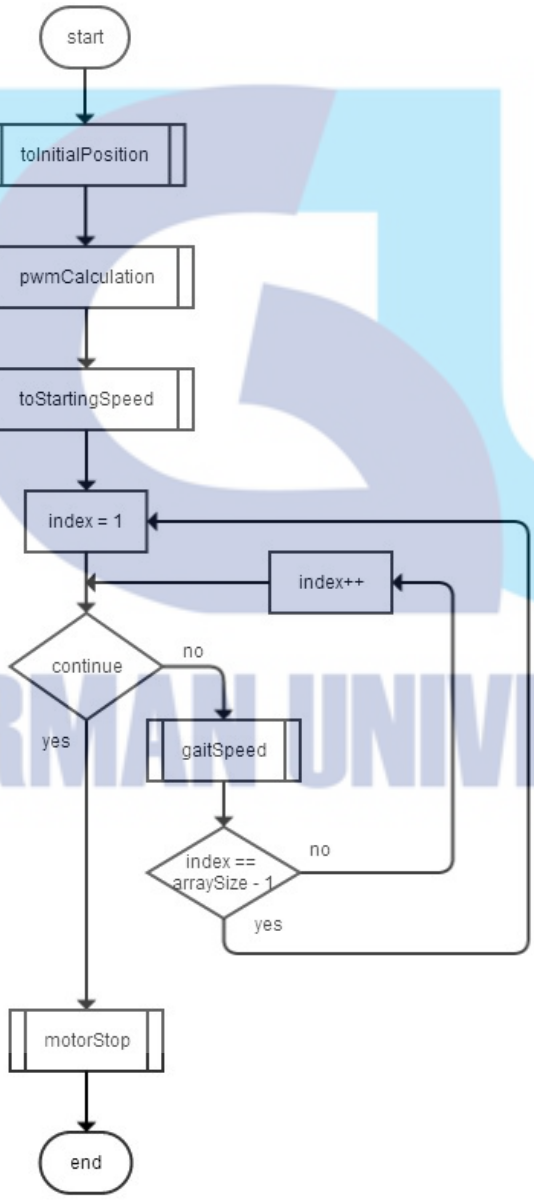

Figure 3.26. Flowchart of the Motor Controller with Speed Approximation

Similarly to the position control mode of the controller, the servos are given commands to reach their respective initial position. However, before continuing to the following steps, an array of PWM value setpoints are calculated based on the input voltage of the servos at that particular time and the array of angular velocity setpoints.

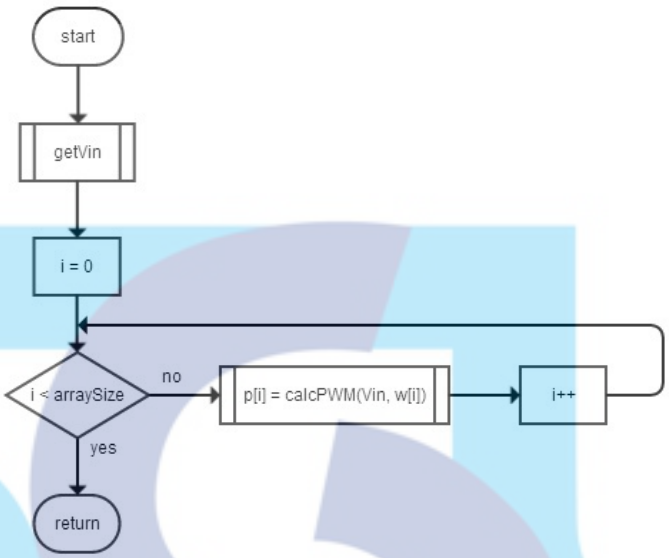

Figure 3.27. Flowchart of the Subroutine "pwmCalculation"

Calculating the PWM values which correspond to the desired angular velocity is done by the subroutine “pwmCalculation”. The input voltage value of the servos is requested from the servos. The subroutine then enters a loop to create an array of PWM value setpoints by converting the array of desired angular velocity setpoints.

Adjustment is done to the angular velocity setpoints instead of angular position setpoints. While the continuous condition is fulfilled, the next velocity setpoints would be executed, otherwise the quadruped robot would stop the movement cycle.

3.4.5.2. Analysis and Calculation Tools

The gait parameters and setpoints to be implemented on the actual quadruped robot are pre-calculated before being embedded to the microcontroller board. Two analysis and calculation software tools are developed to fulfill the mentioned purpose. It is to

be noted that these analysis and calculation tools are used on personal computer, not integrated to the quadruped robot.

3.4.5.2.1. *GaitSimulator*

GaitSimulator program is developed using Qt Creator IDE in C++ programming language. The algorithm of *GaitSimulator* is illustrated on Figure 3.28.

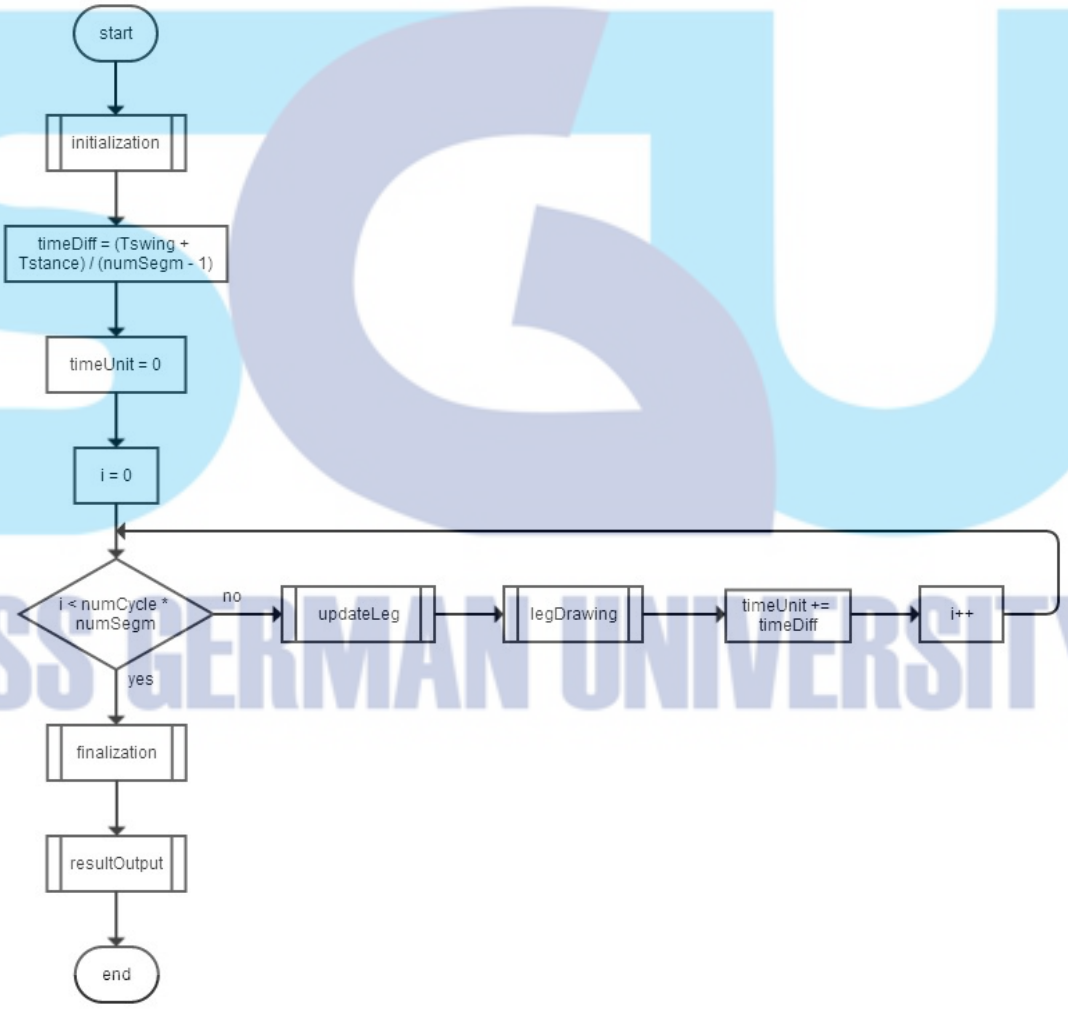

Figure 3.28. Flowchart of *GaitSimulator* Main Program

The gait simulation is calculated with the following limitations:

1) The quadrupedal robot platform has no pitch, roll, and yaw movement.
2) The height of the robot at an instance is equal to the longest vertical distance between the hip joint and foot joint.
3) Until the time this report is written, the stick figures representing the gait generated by the latest version of *GaitSimulator* represent the state of the leg segments, but not the correct position of the leg with respect to the ground.

The whole process starts with initialization. All values on the Graphical User Interface (GUI) are obtained and stored into variables. The scenes for displaying stick figures of the legs are cleared. Afterwards, using the data stored in the variables, a *quadrupedLeg* object is created for every robot legz

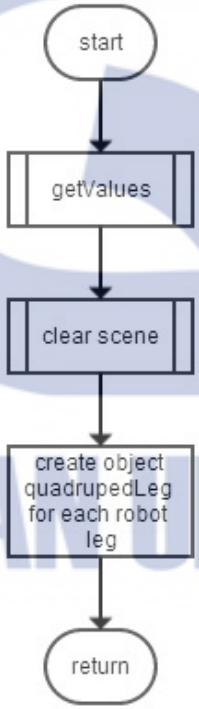

Figure 3.29. Flowchart of the Subroutine "initialization" of *GaitSimulator*

After the initialization is done, the period of one time segment is calculated. The calculation is then done inside a loop. One loop cycle generates one stick figure of the legs.

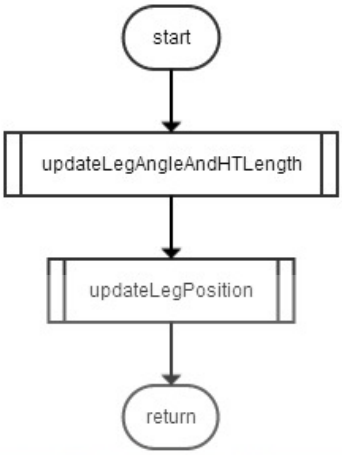

Figure 3.30. Flowchart of the Subrouting "updateLeg"

The first step inside the loop, given the time setpoint, is to update the joint angles and the distance between the hip joint and the toe of the leg. The legs are then drawn.

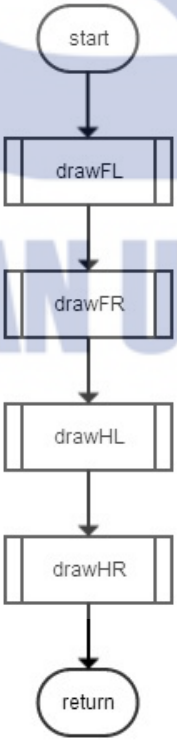

Figure 3.31. Flowchart of the Subroutine "legDrawing"

The loop ends when the calculation during all time setpoints are completed. Subroutine “finalization” is called to destroy all *quadrupedLeg* objects to clear up memory space.

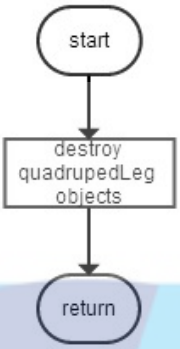

Figure 3.32. Flowchart of the Subroutine "finalization" of *GaitSimulator*

The result of the calculation is shown as the stick figures displayed on a result window and the data is written to one text file for each leg.

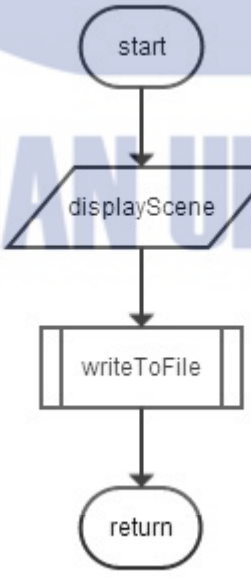

Figure 3.33. Flowchart of the Subroutine "resultOutput"

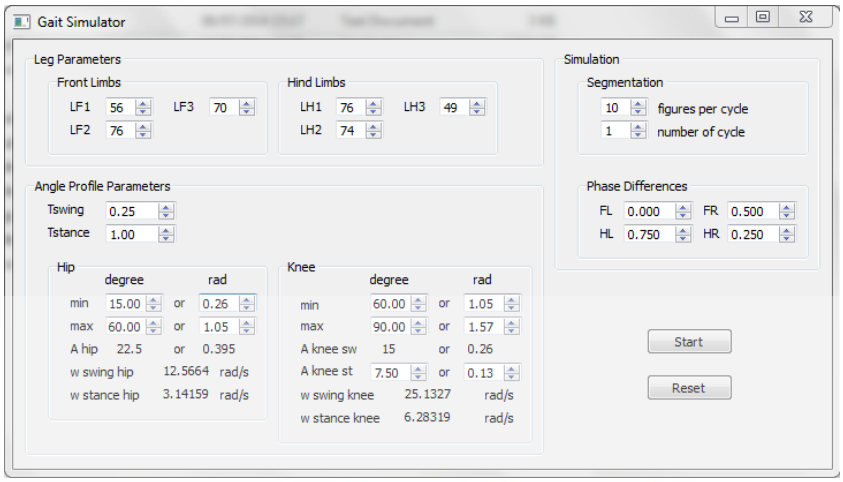

Figure 3.34. User Interface of *GaitSimulator*

Figure 3.34 displays the screenshot of *GaitSimulator* user interface. The user interface is separated into three divisions: "Leg Parameters", "Angle Profile Parameters", and "Simulation".

"Leg Parameters" section contains the length of each leg segmentation. The details of the completed parameters concluded under this group are listed on Table 3.8. When the program is executed, the program comes with default values of leg parameters which have been adapted to the actual newly-developed quadruped robot. However, if the user wishes, the values can be changed by simply typing the desired values on the designated boxes.

Table 3.8. Leg Parameters of *GaitSimulator*

User Interface	Kinematic Model	Description
LF1	l_{F1}	length of the hip-knee segment of the front legs, in millimeters, input by the user, comes with default value
LF2	l_{F2}	length of the knee-ankle segment of the front legs, in millimeters, input by the user, comes with default value
LF3	l_{F3}	length of the ankle-toe segment of the front legs, in millimeters, input by the user, comes with default value
LH1	l_{H1}	length of the hip-knee segment of the hind legs, in millimeters, input by the user, comes with default value
LH2	l_{H2}	length of the knee-ankle segment of the hind legs, in millimeters, input by the user, comes with default value
LH3	l_{H3}	length of the ankle-toe segment of the hind legs, in millimeters, input by the user, comes with default value

"Angle Profile Parameters" section include the sinusoidal gait parameters: swing and stance, minimum and maximum hip joint angles, minimum and maximum knee joint angles, and the oscillation amplitude of the knee joint angle during stance phase. These parameters are defined on Table 3.9.

Table 3.9. Angle Profile Parameters of *GaitSimulator*

User Interface	Kinematic Model	Description
Tswing	T_{sw}	swing period of the legs, in seconds, input by the user
Tstance	T_{st}	stance period of the legs, in seconds, input by the user
Hip min	$\theta_{hip\ max}$	minimum hip joint angle position, in degrees and radians, input by the user
Hip max	$\theta_{hip\ min}$	maximum hip joint angle position, in degrees and radians, input by the user
A hip	A_{hip}	oscillation amplitude of the hip joint angle, in degrees and radians, automatically calculated from the values of *Hip min* and *Hip max*
w swing hip	$\omega_{hip\ sw}$	oscillation frequency of the hip joint angle during swing phase, in radians per second, automatically calculated from the value of *Tswing*
w stance hip	$\omega_{hip\ st}$	oscillation frequency of the hip joint angle during stance phase, in radians per second, automatically calculated from the value of *Tstance*
Knee min	$\theta_{knee\ min}$	minimum knee joint angle position, in degrees and radians, input by the user
Knee max	$\theta_{knee\ max}$	maximum knee joint angle position, in degrees and radians, input by the user
A knee sw	$A_{knee\ sw}$	oscillation amplitude of the knee joint angle during swing phase, in degrees and radians, automatically calculated from the values of *Knee min* and *Knee max*
A knee st	$A_{knee\ st}$	oscillation amplitude of the knee joint angle during stance phase, in degrees and radians, input by the user
w swing knee	$\omega_{knee\ sw}$	oscillation frequency of the knee joint angle during swing phase, in radians per second, automatically calculated from the value of *Tswing*
w stance knee	$\omega_{knee\ st}$	oscillation frequency of the knee joint angle during stance phase, in radians per second, automatically calculated from the value of *Tstance*

As described on Table 3.9, some of the angle profile parameters are automatically calculated by the internal algorithm of *GaitSimulator* program. Such parameters are those which values are derivable from other parameters. As soon as the required parameters are input by the user, the dependant parameters are immediately calculated and displayed on their respective spaces.

The final division, "Simulation" section, sets the number of time segmentations per one gait cycle, the number of gait cycle to be simulated, and the phase offset differences for every legs. Table 3.10 enlists all the simulation parameters of *GaitSimulator* program.

Table 3.10. Simulation Parameters of *GaitSimulator*

User Interface	Kinematic Model	Description
figures per cycle	-	number of stick figures per cycle to be displayed on the result window, in integers, input by the user
number of cycle	-	number of cycle to be simulated, in integers, input by the user
FL	-	phase offset of the front-left leg, in fraction of one cycle period, input by the user
FR	-	phase offset of the front-right leg, in fraction of one cycle period, input by the user
HL	-	phase offset of the hind-left leg, in fraction of one cycle period, input by the user
HR	-	phase offset of the hind-right leg, in fraction of one cycle period, input by the user

After inputting all the necessary parameters, the simulation can be started by clicking the "Start" button. The program then calculates the state of each leg at certain time segmentation. Four series of stick figures, one series representing each leg, are depicted on a result window. Figure 3.35 shows an example of resulting gait generated using *GaitSimulator*.

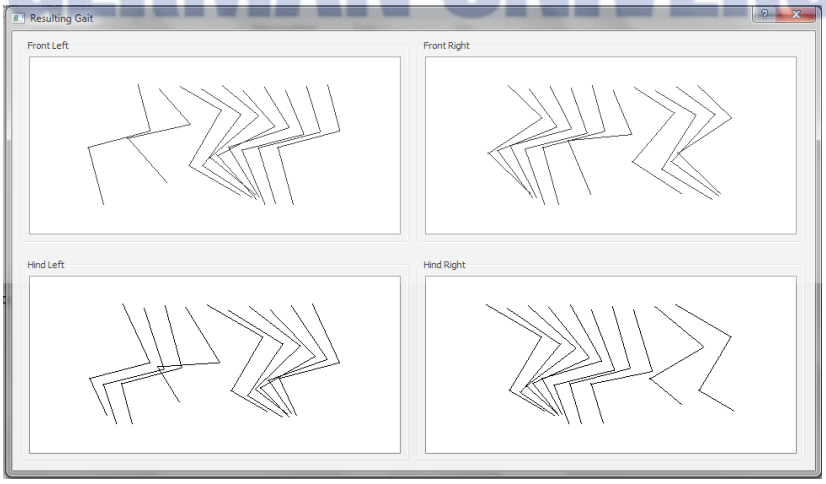

Figure 3.35. Example of the Resulting Gait Generated using *GaitSimulator*

3.4.5.2.2. *GaitSetpoint*

While *GaitSimulator* represents the ideal condition of having the joint angle profile executed perfectly at the time setpoints, *GaitSetpoint* compensates for the limitations of the selected servo motors. The job of *GaitSetpoint* is to generate the arrays containing time, angular position, and angular velocity setpoints. The purpose of these setpoints is to approximate the ideal curve of the gait into a rougher but applicable curve. Only at these setpoints the approximation curve of the gait is obliged to coincide with the ideal curve of the gait.

Figure 3.36. Flowchart of the *GaitSetpoint* Main Program

GaitSetpoint program is also developed using Qt Creator IDE in C++ programming language. The algorithm of *GaitSetpoint* is illustrated on Figure 3.37.

The program is divided into three main subroutines: "initialization", "writeToFile" and "finalization". Subroutine "initialization" reads the input data from the Graphical User Interface, builds the general time setpoints, offset time setpoints, angular position setpoints, and angular velocity setpoints. The result of calculation is written onto text files by the subroutine "writeToFile". Before the program ends, the memory space is cleared up with the subroutine "finalization".

The user interface of *GaitSetpoint* program is rather similar to that of *GaitSimulator* program. It is also divided onto three main groups: "Leg Parameters", "Angle Profile Parameters" and "Gait".

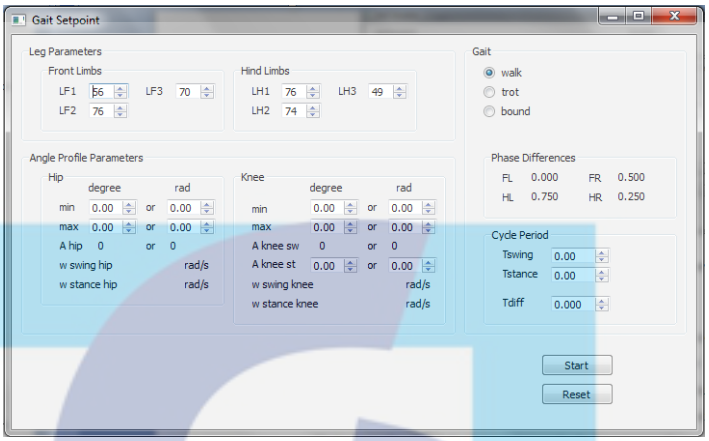

Figure 3.37. User Interface of *GaitSetpoint*

Identically to the "Leg Parameters" of *GaitSetpoint*, "Leg Parameters" of *GaitSetpoint* also represent the length of each leg segment correspond to the actual quadruped robot platform. Explanation of each parameters under "Leg Parameters" group may refer to Table 3.11.

Table 3.11. Leg Parameters of *GaitSetpoint*

User Interface	Kinematic Model	Description
LF1	l_{F1}	length of the hip-knee segment of the front legs, in millimeters, input by the user, comes with default value
LF2	l_{F2}	length of the knee-ankle segment of the front legs, in millimeters, input by the user, comes with default value
LF3	l_{F3}	length of the ankle-toe segment of the front legs, in millimeters, input by the user, comes with default value
LH1	l_{H1}	length of the hip-knee segment of the hind legs, in millimeters, input by the user, comes with default value
LH2	l_{H2}	length of the knee-ankle segment of the hind legs, in millimeters, input by the user, comes with default value
LH3	l_{H3}	length of the ankle-toe segment of the hind legs, in millimeters, input by the user, comes with default value

"Angle Profile Parameters" of *GaitSetpoint* refers to the sinusoidal gait parameters of the hip and joint angle profile, similarly to its counterpart on *GaitSimulator* but without the *Tswing* and *Tstance* parameters. They are included on "Gait" section instead. The entire list of angle profile parameters of *GaitSetpoint* is recorded on Table 3.12.

Table 3.12. Angle Profile Parameters of *GaitSetpoint*

User Interface	Kinematic Model	Description
Hip min	$\theta_{hip\ max}$	minimum hip joint angle position, in degrees and radians, input by the user
Hip max	$\theta_{hip\ min}$	maximum hip joint angle position, in degrees and radians, input by the user
A hip	A_{hip}	oscillation amplitude of the hip joint angle, in degrees and radians, automatically calculated from the values of *Hip min* and *Hip max*
w swing hip	$\omega_{hip\ sw}$	oscillation frequency of the hip joint angle during swing phase, in radians per second, automatically calculated from the value of *Tswing*
w stance hip	$\omega_{hip\ st}$	oscillation frequency of the hip joint angle during stance phase, in radians per second, automatically calculated from the value of *Tstance*
Knee min	$\theta_{knee\ min}$	minimum knee joint angle position, in degrees and radians, input by the user
Knee max	$\theta_{knee\ max}$	maximum knee joint angle position, in degrees and radians, input by the user
A knee sw	$A_{knee\ sw}$	oscillation amplitude of the knee joint angle during swing phase, in degrees and radians, automatically calculated from the values of *Knee min* and *Knee max*
A knee st	$A_{knee\ st}$	oscillation amplitude of the knee joint angle during stance phase, in degrees and radians, input by the user
w swing knee	$\omega_{knee\ sw}$	oscillation frequency of the knee joint angle during swing phase, in radians per second, automatically calculated from the value of *Tswing*
w stance knee	$\omega_{knee\ st}$	oscillation frequency of the knee joint angle during stance phase, in radians per second, automatically calculated from the value of *Tstance*

The descriptions on Table 3.12 show that some of the angle profile parameters are mentioned to be "automatically calculated" instead of "input by the user". Such parameters are indeed automatically calculated by the internal algorithm of *GaitSetpoint* program. Their values are derivable from other parameters. As soon as the required parameters are input by the user, the dependant parameters are immediately calculated and displayed on their respective spaces.

The third main group, "Gait" section, has the three common quadrupedal gait types to be chosen (walking, trotting or bounding). The phase difference of each leg depends on the selected gait option, which values are following the figures on Table 3.1. Cycle periods include the swing phase period, stance phase period, and the time differential period. Descriptions of these parameters are shown on Table 3.13.

Table 3.13. Gait Parameters of *GaitSetpoint*

User Interface	Kinematic Model	Description
walk	-	walking gait option, selected by the user
trot	-	trotting gait option, selected by the user
bound	-	bounding gait option, selected by the user
FL	-	phase offset of the front-left leg, in fraction of one cycle period, automatically referred to the values on Table 3.1 based on *walk*/*trot*/*bound* selection
FR	-	phase offset of the front-right leg, in fraction of one cycle period, automatically referred to the values on Table 3.1 based on *walk*/*trot*/*bound* selection
HL	-	phase offset of the hind-left leg, in fraction of one cycle period, automatically referred to the values on Table 3.1 based on *walk*/*trot*/*bound* selection
HR	-	phase offset of the hind-right leg, in fraction of one cycle period, automatically referred to the values on Table 3.1 based on *walk*/*trot*/*bound* selection
Tswing	T_{sw}	swing period of the legs, in seconds, input by the user
Tstance	T_{st}	stance period of the legs, in seconds, input by the user
Tdiff	-	single time segmentation period, in seconds, input by the user

3.5. Gait Implementation Method

After the desired hip and knee joint angle profiles are known, a simple approach for implementing them to the actual quadruped robot would be approximating the gait by choosing a number of setpoints from the angle profiles. These setpoints are treated as the input for the Motor Controller Module.

The chosen smart servo, Dongbu HerkuleX DRS-0101, provides two types of motor control modes: position control mode and continuous rotation speed control mode. On

the first thought, position control mode seems enough for ensuring the servos to reach the desired angular position setpoints. However, since the position control is done internally by the servo, using this control mode is relatively having less control on parts of the angular position curve which are located between two angular position setpoints. Implementing continuous rotation speed control mode is more laborious, because the servos have to achieve the right rotation speed over the right period of time to reach the accurate angular position as demanded by the joint angle profiles. The reward, though, is expected to be higher controllability on the area between two setpoints.

It is then decided to investigate the performance of these two approximation methods. The conducted test consists of generating the joint angle curve using the two methods, mathematically by calculation and empirically by experiment.

3.5.1. Gait Approximation through Position Control

The position control mode of the servo is able to achieve relatively accurate angular position. A number of time setpoints are determined. The ideal angular positions at those particular time are calculated and become the angular position setpoints. The time setpoints and the respective angular position setpoints are employed to the Arduino board.

3.5.2. Gait Approximation through Speed Control

The continuous rotation speed control mode of the HerkuleX servo is capable of providing linear transition of angular velocity. Integrating this linear function of angular velocity against time results on quadratic function of angular displacement. It is then desired to divide the reference angular position curve, which is sinusoidal, into a set of quadratic functions as approximation to the reference curve, so that the angular velocity curve may be built as a set of linear functions.

$$\omega = at + b$$

In order to ensure the quality of the approximation curve, a number of time setpoints are determined, at which the approximated angular position curve intersects the reference angular position curve. This means that the integration of the angular velocity curve between two of those points must correspond to the reference angular position, as shown by the following equation:

$$\theta_{i+1} - \theta_i = \int_{t_i}^{t_{i+1}} (a_i t + b_i)\, dt$$

$$\theta_{i+1} - \theta_i = \frac{1}{2} a_i ({t_{i+1}}^2 - {t_i}^2) + b_i (t_{i+1} - t_i) \quad \text{..... Eq. (3.26)}$$

It is also preferred that the angular speed profile is continuous, even though it is consisted of a set of linear equations. Therefore, the time setpoints should also mark the time when two linear equations intersect each other (the value of angular speed at a time setpoint has to fulfill two linear equations):

$$\omega_i = a_i t_i + b_i \quad \text{..... Eq. (3.27)}$$

Using Equation 3.26 and 3.27, the set of linear equations forming the angular speed profile could be determined.

The necessary steps to determine the set of linear equations are as stated below:

1. Determine a set of time setpoints consisted of *n* elements.
2. Calculate the desired angular position at those times, resulting on a set of angular position setpoints, also consisted of *n* elements.
3. Assume that at the beginning of the gait cycle, the angular velocity is equal to 0 ($\omega_0 = 0$).
4. Calculate the coefficients of the linear equation for every two time setpoints, resulting on a set of linear equations representing the angular velocity profile consisted of $n - 1$ elements.

As the set of linear equations forming the angular velocity profile is obtained, the set of desired angular velocity during the time setpoints can be determined. The desired angular velocity setpoints are implemented on the motor controller.

3.6. Testing Plan

The following section introduces the different kinds of tests to be conducted and their respective testing method.

3.6.1. Motor Test

In order to implement the speed control mode of the motor controller, it is necessary for being able to control the rotation speed in terms of degrees per second instead of Pulse Width Modulation (PWM) duty cycle. The matter is that electric motors are most frequently controlled by altering the PWM duty cycle. As has been described in Section 3.4.5.1.2, a relationship among the desired angular velocity ω, the input voltage V_{in} and PWM duty cycle value p is required to be investigated.

This motor test includes measuring the resulting angular velocity of the servo motor by supplying the motor with variety of input voltage and applying different PWM duty cycle value. The angular velocity is measured by sampling the angular position of the servo while the servo is running.

In conclusion, the motor test procedure is as follows:

1. The serial port of the servo is connected to the serial port of the microcontroller board.
2. Another serial port of the microcontroller board is connected to Personal Computer (PC) through a Serial-to-USB converter.
3. Certain value of input voltage is supplied to the servo.
4. The microcontroller board is turned on.
5. A serial monitor application is opened.
6. The connecting port is ensured to be opened.
7. A certain command is sent through the serial monitor to drive the servo, so that it rotates on certain PWM duty cycle value.
8. The angular position of the servo is sampled in around 10 seconds.
9. A certain command is sent through the serial monitor to stop the servo.
10. The data samples are collected.

11. Processes 1 to 10 are repeated for various values of input voltage and PWM duty cycle.
12. The resulting angular velocity on each experiment is calculated.
13. The resulting angular velocity is related as function of input voltage and PWM duty cycle value.
14. The equation of the PWM duty cycle value as function of input voltage and desired angular velocity is determined.

3.6.2. Gait Model Performance Test

Gait model performance test is conducted to examine the gait generated from certain set of gait parameters, or to predict whether the particular parameters set being tested is visually acceptable. The test is supported by the *GaitSimulator* program, which outputs graphical representation of leg stick figures and four text files containing data of the joint positions at each time setpoint and joint angle profile. The raw data in the text files can be processed in Microsoft Excel if the numbers are going to be analyzed further.

The following points list the steps for doing the gait model performance test:

1. The set of parameters to be tested is determined.
2. The *GaitSimulator* program is run.
3. The gait parameters and simulation parameters are input on the Graphical User Interface of *GaitSimulator*.
4. The calculation process is started.
5. The graphical representation of the leg motion prediction is observed.
6. If considered as necessary, Microsoft Excel can be used to process the raw data in the four text files.

3.6.3. Gait Approximation Performance Test

The purpose of gait approximation is to consider the ability of servo motors used on the actual quadrupedal robot. Performing this test before implementing the gait would

hinder the risk of putting higher constrain than what can be handled by the servos. Using *GaitSetpoint* program, the time, angular position and angular velocity setpoints will be determined.

The steps of conducting the gait approximation performance test are as follows:

1. The set of parameters to be tested is determined.
2. The *GaitSetpoint* program is run.
3. The gait parameters are input on the Graphical User Interface of *GaitSetpoint*.
4. The interval of the time setpoints is determined and also input on the Graphical User Interface of *GaitSetpoint*.
5. The calculation process is started.
6. Microsoft Excel is used to process the raw data in the text files.
7. For evaluating the position approximation approach, the angular position setpoints are observed, and the biggest average changing rate of angular position is calculated.
8. For evaluating the speed approximation approach, the angular velocity setpoints are observed, and the biggest angular velocity value is examined.

3.6.4. Gait Implementation Performance Test

Through gait approximation performance test, a particular gait is examined whether it is applicable on the actual quadruped robot or not. The gait which passed the test is then trialed on the platform. Arrays of setpoints calculated by *GaitSetpoint* are embedded to the motor controller program in the microcontroller board.

Gait implementation would be examined as stated on the following points:

1. The set of gait parameters to be implemented is determined.
2. The gait model performance test is performed to inspect whether the gait is visually favorable.
3. The gait approximation performance test is performed to evaluate whether the gait can be implemented on the actual quadruped robot. If the gait is proved to

be exceeding the capability of the actual robot, a new set of gait parameters is to be selected before continuing to the next steps.

4. The resulting arrays of setpoints are implemented on the motor controller program.
5. The motor controller is run on the actual quadruped robot.