Quadrupedal walking robot

Martin Metal, *statically balanced crawler*

Quadpod - quadrupedal walking robot or four legged walking robot - is a mechanical construction using 4 legs to move at slow speed. This project demonstrates utilization of various crawling gaits to achieve synchronized movement of the robot. Each of the tested gaits is statically balanced, the dynamically balanced gaits are outside the possibilities of this project. The mechanical construction uses 3 mm plywood and low budget 9 gram servos. The electronics is fitted on a single board having 2 conducting layers. Design further uses Atmel Mega168, servo shield, IR receiver and 2 independent power sources.

1 Mechanical Construction

The quadpod is a robot having the smallest possible number of legs that allows to use statically balanced walking pattern. Statically stable walking pattern is generally very slow, usually used by insect to crawl. The walking cycle is however stable in any moment of the cycle and therefore very useful for analysis and study. The statically stable walking pattern requires that 3 legs are resting on the supporting surface and just one leg may be lifted at time. Additionally, the centre of gravity (CoG) of the robot must reside inside the virtual triangle formed by the foot of legs resting on the walking surface.

1.1 Mechanics of Crawlers

The designed robot is capable of statically balanced walking pattern only. It has neither necessary mechanical construction nor suitable systems of actuators to achieve dynamically balanced movement. The used electronics does not have enough potential to calculate the parameters of dynamically balanced movement in real time either. Therefore the further description is reduced to statically balanced walking pattern only. The details of the statically balanced walking pattern follows.

Locomotion of legged robots is based on lifting and placing legs in a coordinated manner. This is called a gait. Gait is usually cyclic in the sense that the same sequence of lifting and placing the legs is repeated. A complete cycle of leg movements, where all the legs have been lifted and placed exactly once, is called a stride, and the time duration of one stride is called the cycle time.

> ***Duty factor*** **(β): The fraction of the duration of the stride for which a foot is on the ground (in the support phase).**

The cycle time is normalized, begins at time 0 and cycle end at time 1. Next important parameter is called duty factor. The duty factor β_i for leg i is the fraction of the cycle time for which the foot is in ground contact, so the duty factor is a number between 0 and 1.

Relative phase **(φ_i): The time elapsed from the setting down of a chosen reference foot until the foot of leg *i* is set down, given as the fraction of the *cycle time*.**

The relative phase of leg i is the time elapsed from the setting down of an arbitrarily chosen reference foot until the foot of leg i is set down, given as the fraction of the cycle time as explained in the gray box. The convention used here is that the left front leg is number 1, right front leg is number 2, right hind leg is number 3, and left hind leg is number 4, as depicted in figure 2, figure 3 respectively. The gait event sequence can now be specified using the duty factors and the relative phases, where the first event, and the start of the stride, is chosen as the event when the reference leg is set down.

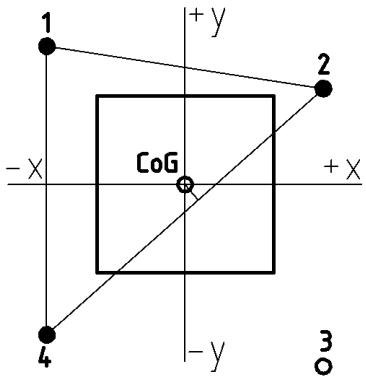


Figure 1: *The CoG is withing the support area, which is defined by the foot of legs being in support.*

In a statically stable gait, the vertical projection of the center of gravity (CoG) onto a horizontal plane, is kept within the support area at all times as shown in the figure 1. The support area can be described as the minimum convex polygon in a horizontal plane, with its vertices formed by the tips of the legs being in support. This description does not take into account moment of inertia and other dynamic properties of the system. It assumes zero mass of the legs as well. While these simplification are helpful when calculating the motion commands it might on the other end result in errors and instabilities in the resulting kinematics. Therefore is important to include *stability margin* in the characteristics of the system. The higher stability margin the less effect of adopted simplifications. The stability margin is defined as the shortest distance from the vertical projection of the CoG of the vehicle onto a horizontal plane, to the boundary of the support area. It is defined as positive if the center of gravity is within the support polygon and negative otherwise. Negative values of stability margin cannot result in statically balanced crawl. The robot tips over due to the effect of gravity. Various strategies are available that increases the stability margin. This projects discuss *swaying* and *tilting* as some of the strategies.

This section established the nomenclature required for further description of the walking pattern of the robot. Most of the theoretical foundation originates in the work [1]. Interesting and complementary information were sourced in the works [3], [2] respectively.

Gait variations were subject to test in this project. Basically, two geometrical configurations were used. First, a **crab-like geometry**. The details of the geometry are depicted in the figure 2.

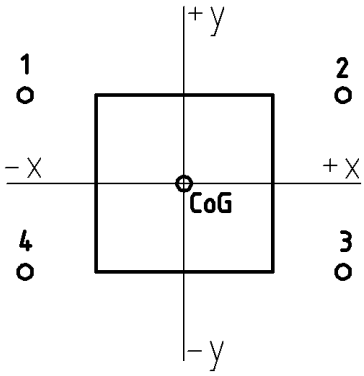


Figure 2: *Crab-like geometry of the legs. The leg numbering starts from left front and continues clockwise along the body.*

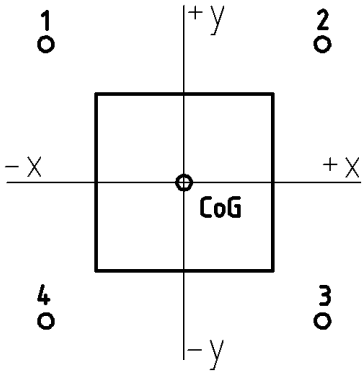


Figure 3: *Frog or Lizard like geometry of the legs.*

This geometry shows that the mid of a stride is put on the line connecting the shoulder joints of both front legs, hind legs respectively. This geometry brings relatively small support area. On the other end **titling** is more effective compared with other geometries as a strategy to increase the stability

margin.

Frog / Lizard like geometrical configuration brings larger support area though the stability margin is exactly the same compared to *crab* like configuration. The mid of a stride is set on a connecting line between left front and right hind leg, as shows in figure 3. This configuration is less sensitive to the *titling*. The tilting input must be much larger to achieve reasonable shifting of CoG. This configuration is however more suitable for emulated **lizard-like gait**, where the CoG gets shifted by the swaying motion of the body.

Basic Crawl is a gait with β=0.75. This is the basic crawling pattern that utilizes the minimum possible *duty factor* allowed for statically balanced gait. Any value bellow this level would mean that a dynamic stability must be included in the gait, because just two legs are supporting the construction for defined fraction of a cycle. The *duty factor* β=0.75 means that the robot is supported by three leg at each moment, one leg is always lifted and changing its positions. Basic crawl - using the nomenclature defined if chapter 1.1 - can be described as follows:

$$\varphi_1 = 0 \quad \varphi_2 = 0.5 \quad \varphi_3 = \beta \quad \varphi_4 = \beta - 0.5 \qquad (1)$$

This gait was tested with both the *crab* and *frog* like geometry. It has been combined in both cases with tilting mechanism in oder to achieve higher positive values of stability margin.

Tilting is a strategy helping to achieve higher positive values of stability margin. This strategy is tilting the body of the robot for specific relative phase of a stride where the stability margin approaches zero. The minimal stability margin occurs short before the leg 1 or leg 3 lifts as visible in figure 5. Expressed in the model the conditions occur at:

$$\varphi_1 = 0, \varphi_3 = \beta \qquad (2)$$
$$\varphi_2 = 0, \varphi_4 = \beta \qquad (3)$$

Conditions described in equations 2 and 3 generates zero stability margin as the CoG projected in the support area is exactly on the connection line between leg 2 and 4 resting on the support area. By extending the leg 3 and simultaneously retracting the leg 1 the robot tilts forward-left, virtually shifting the projected CoG deeper inside the support area,

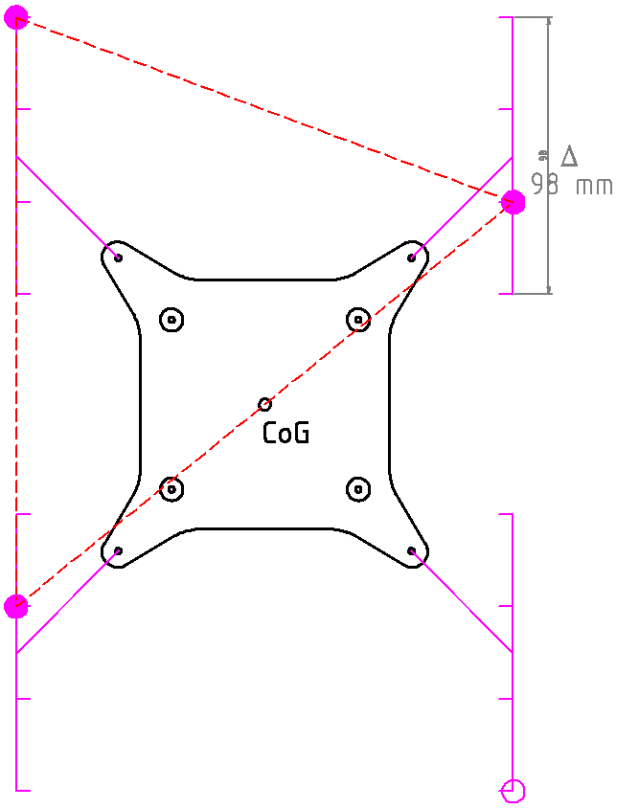


Figure 4: *The position of the CoG in standard walking pattern. The drawing represents the situation where the leg 1 gets placed on the surface and leg 3 just gets lifted ($\varphi_1 = 0$, $\varphi_3 = \beta$). The stability margin is zero.*

short before the leg 3 gets lifted into non-supportive position. In other word the distance increases between projected CoG and the edge of the support area shaped by the connection between legs 2 and 4. The absolute value of this distance depends on the height of the robot and size of the support area. Small titling movement generates larger shift of CoG for smaller support area and vice versa, more tilting must be added to achieve same shift of CoG in the case of larger support area. The same situation occur for conditions captured in equation 3 as well.

Swaying is another strategy of increase the stability margin. Reptiles are a good example of well articulated swaying. Lizards clearly bend the body and shift its head and tail in order to place the CoG before a stride is taken.

This robot is having a rigid body construction, the only joins and actuators are placed in the legs. Therefore the swaying pattern must be simulated by changing the projection of the centre of mass of robot onto the walking plane. This in effect results in distorting

the trajectory of each step within the stride. The trajectory is under normal circumstances straight. The emulated lizard-like walking pattern results in a trajectory shown in the figure 5, 6 respectively. The first mentioned swaying model is not fully statically stable. The CoG is clearly outside the support area for two phases of the step. More specifically the static balance is not achieved shortly before the phase 1[1] and 3 end.

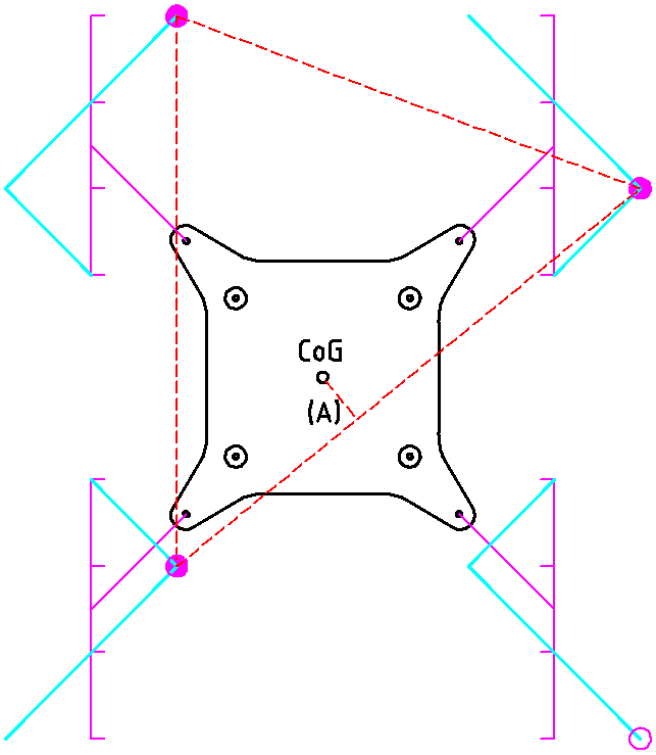


Figure 5: *The position of the CoG in lizard walking pattern. It corresponds the situation where the leg 1 gets placed on the surface and leg 3 just gets lifted. The symbol (A) stands for the stability margin.*

This is compensated by "tilt" function and by the dynamics of the movement. The robot has certain mass and thus momentum, which is used here to achieve semi-stable walking pattern.

The second referred model, shown in the picture 6, compensates fthe problem described in the first swaying model and thus it brings the complete static stability for any relative phase of a stride. The penalty is slowing down the walking itself by including two section where the robot is supported by all 4 legs and is shifting its CoG only. This happens in phase 1, 3 respectively. Both are divided into *(a)* and *(b)* sections. While the *(a)* is the "leg shifting" phase, the *(b)* is the "CoG shifting" phase, where all legs are supporting the body.

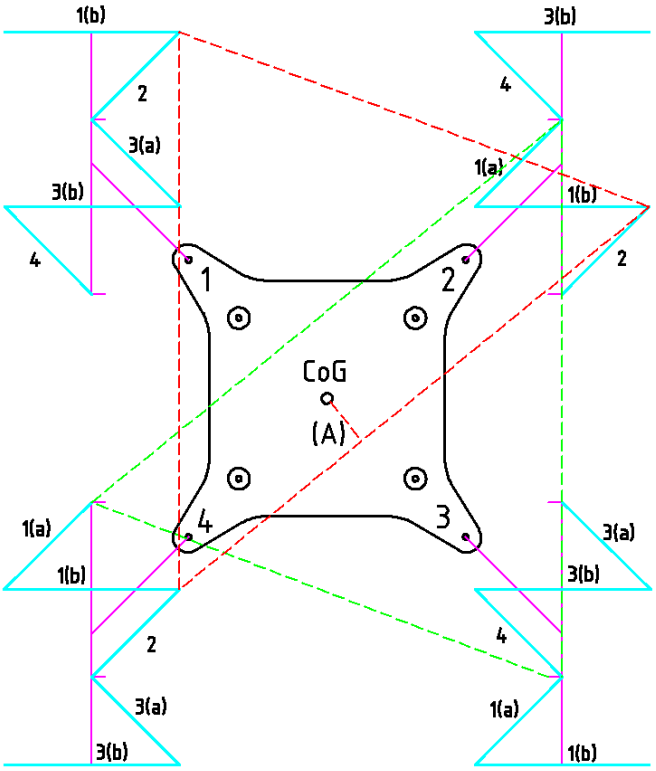


Figure 6: *The position of the CoG in advanced swaying walking pattern. The green outlined support area is valid for the situation where the leg No.1 gets moved in its non-duty part of cycle, while the red color is used to outline the support area when the leg no.3 travels through its non-duty cycle. The symbol (A) stands for the stability margin.*

The phases are outlined for all legs in the drawing 6 in small letters next to the turquoise trajectory of a step. The schema can be used for forward movement only. The trajectories of legs must be flipped (mirrored) along the longitudinal axis for reverse movement. The change in the direction cannot happen at any arbitrary phase of a stride. It only can happen when the trajectory of the step crosses the trajectory of the linear step. That is satisfied for $\varphi = 0.25$ and $\varphi = 0.75$ only.

The figures 5 and 6 demonstrate the effect of the swaying pattern. The color code in the first figure uses black color of the body of the robot, the CoG clearly indicated in the middle. The turquoise color shows the step trajectory. The red triangle shows

[1] End of phase 1 is the situation where the leg No.1 is lifted but it is shortly before it gets placed back on the support plane

the support area at the beginning of the cycle, just after the leg 3 got lifted and the robot is supported by legs 1,2 and 4 only. The drawing 5 documents the negative stability margin occuring short before the phase 1 and 3 end, which may result in tipping the whole robot[2].

The lizard-like movement the trajectory of a stride is specific for each leg as shown in the figures 5 and 6. The trajectory of step is depicted in cyan color. The magenta color shows the linear trajectory of foots having outlined segment for φ at positions 0=1, 0.25, 0.50 and 0.75. The drawing is set for the same cycle of the stride as the figure 4. The CoG is clearly moved much deeper inside the triangle of the support area, depicted in red color again.

Swaying is increasing the stability margin considerably and is well suited strategy to achieve statically stable crawling pattern for this robot.

Advanced Crawling is the last tested strategy in the project. Advanced crawling step is an artificial construct that increases the β to the value as high as 0.83. The larger duty factor helps to increase the minimal positive value of the stability margin.

The situation depicted in the figure 7 corresponds with the situation captured in the figure 4. One can see that the stability margin is a positive number in the case of the figure 7, while the standard walking pattern delivers zero value of stability margin (figure 4). The presented figure demonstrates the advantage of the gait denoted as "$\beta = 0.83$".

The schema of this gait is however more complicated compared to the basic walking pattern. It is still based on the sequence 1-3-2-4, though it utilizes a model of non-equally long segments within the whole cycle. The details are discussed later in next chapter.

1.2 Kinematics of a Crawler

Kinematics is the study of the geometry of a mechanical system, where the motion of the system

[2]The length of the stride in the drawing corresponds with 98 mm, which is unrealistically long stride given the mechanical construction of legs and size of the body, distances between shoulder pivot respectively. This stride is just demonstrative in oder to visualize the effect of swaying more clear. In reality the mechanical constraints allow the stride 72 mm at maximum.

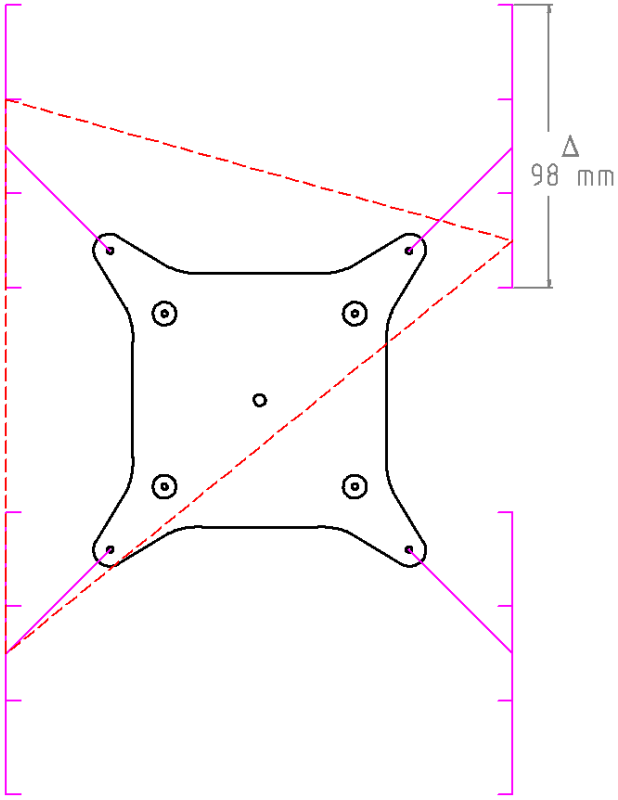


Figure 7: *The position of the CoG in advanced crawling pattern. The drawing represents the situation at the beginning of the cycle, where the leg 1 is already supporting the robot and leg 3 gets just lifted. ($\varphi_1 = 0$, $\varphi_3 = \beta$).*

can be described in terms of the velocity and acceleration of all its components. The components can be connected through different types of joints, which limit how the components can move relative to each other. The interconnection of the components implies that the motion of the components relative to each other is constrained. A kinematic motion of the mechanism is determined by specifying all the generalized coordinates as function of time, and thereby generating a curve for the motion of all points in the mechanism. The coordinates have to be given relative to a frame, usually a Cartesian coordinate system $\Re^3$, that is fixed relative to the walking plane, also known as walking frame or earth frame. Symbol fW denotes the walking frame in the text and is depicted in the figure 8.

As explained in detail in source [1] is the motion of a single component through three dimensional space $\Re^3$ has a geometric degree of freedom of six and can be described with a set of six generalized coordinates. Three coordinates are needed to define the position of an index point of the component,

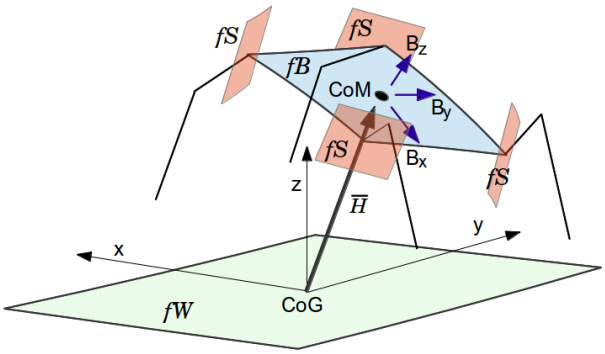


Figure 8: *The position of the CoG is the projection of robot's CoM onto the walking plane. The walking plane is denoted as* fW *known as "walking frame", the orientation of the robot's body denotes* fB*, known as "body frame". Each leg has its own "servo frame",* fS*. The feets of leg are actually positioned withing the servo frames.*

and three coordinates to define its orientation. First of all we define the frames and fixed points for this construction. The example in the figure 8 shows the CoM, CoB respectively as the chosen fixed point for the body frame *fB*. The projection of the CoM of the robot into the walking frame *fW* is the base index point for the walking frame. The projection is done for idle state of the robot, having all legs in neutral position, supporting the robot. The next frame is the servo frame *fS*. The index point of the servo frame is the shoulder joint of the leg. Therefore we have 4 servo frames. The relationship among the frames can be established by chain of affine rotation matrices. We can determine position of each foot by using this system of chained matrices.

By applying the described schema onto the mechanical construction of the robot the kinematics is calculated as follows. First, we establish the relationship between the fixed points of walking frame and body frame. This relationship is depicted in the drawing 8 by vector $\vec{\mathcal{H}}$. The radial distance between the index points of *fW* and *fB* is the length of $\|\vec{\mathcal{H}}\|$. Technically, the length of this vector is the base height of the robot's body over the walking frame, known as h_z^0. Further we need polar angle and azimuthal angle to determine the position of index point of robot's body. It is convenient to use projection of the CoM onto $[x, z]$ and describe it by polar angle $\theta_{(xz)}$ accompanied by the projection into $[y, z]$, which yields polar angle $\theta_{(yz)}$. This transformation simplifies the rotation and eliminates the need for azimuthal angle.

$$M_{(xy)} = \begin{bmatrix} cos(\theta_{(xy)}) & -sin(\theta_{(xy)}) & 0 \\ sin(\theta_{(xy)}) & cos(\theta_{(xy)}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$M_{(xz)} = \begin{bmatrix} cos(\theta_{(xz)}) & 0 & -sin(\theta_{(xz)}) \\ 0 & 1 & 0 \\ sin(\theta_{(xz)}) & 0 & cos(\theta_{(xz)}) \end{bmatrix} \quad (5)$$

$$M_{(yz)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_{(yz)}) & -sin(\theta_{(yz)}) \\ 0 & sin(\theta_{(yz)}) & cos(\theta_{(yz)}) \end{bmatrix} \quad (6)$$

$$\vec{\mathcal{V}}_{fB} = \vec{\mathcal{V}}_{fW} M_{(xy)} M_{(xz)} M_{(yz)} \quad (7)$$

We construct the vector $\vec{\mathcal{H}} = [0, 0, h_z^0]$, where the coordinate h_z^0 is the length of the vector - $\|\vec{\mathcal{H}}\|$. The vector gets rotated by $M_{(xz)}$ (equation 5) and $M_{(yz)}$ (equation 6). The coordinates x and y are interesting in this step, the coordinate z is neglected. The calculated coordinates *x,y* defines the "sway" function in the movement of the robot. In the next phase we calculate the relationship between the index points of body frame and servo frames. The index point of body frame is CoM. The index point of servo frame is shoulder joint. The robot frame has a rigid construction, therefore both *x,y* coordinates are constant. The only relevant output of this rotation is then change in z coordinates, combining the effect of function "roll" and "pitch". So again, we take vector $[x, 0, 0]$ and rotate it for "roll" angle using the matrix shown in equation $M_{(xz)}$ (5). Following that we take vector $[0, y, 0]$ rotate for the "pitch" angle using the matrix from equation $M_{(yz)}$ (6). The resulting coordinate z is sum of both calculation. In the next step we determine the effect of "yaw" function on the robot. This function affects coordinates *x,y* only. Generally, the "yaw"can be calculated for index point in the walking frame or for the index point of servo frames projected into walking frame. The input for the "yaw" function is always vector $[0, y, 0]$, where the coordinate y varies based on the gait style input. However the origin of y is irrelevant for outlined procedure and will be discussed in next section. We determine 4 tuples $[0, y_i, 0]$ where the y is different for all 4 legs, as described in the chapter about the mechanic of the robot. The 4 vectors gets rotated for the "yaw" angle using the matrix $M_{(xy)}$ given in equation (4). Again, "yaw" function rotates the robot in plane $[x, y]$, therefore we can discard the coordinate x in the last rotation. The last step combines the results of all the preceding steps. Function "yaw" contributes to $[x, y]$ while both functions

"pitch" and "roll" impacts the coordinate z only. We use combined vectors for the final step, which is the rotation in the $[x, y]$ plane. This rotation expresses the relationship between the body frame and servo frames. The rotation takes the cumulated vector describing the index points of shoulder frame plus the yaw input for each leg. This vector is first translated for $[x, y]$ coordinates given by vector $\vec{\mathcal{L}} = [x, y]$, whose length corresponds with the physical construction of a leg. The foot of a leg has an initial position, a "sweet" point. It is an ideal position of foot within an action radius of a leg. The value of translation in direction x and y is given by the topology of the construction and can be either crab-like or frog-like as described in the figure 2, figure 3 respectively. Using the 4 leg vectors, translating them for input given by geometry we finally rotate the vectors to receive the coordinates for all feet in a phase of a step. This procedure is repeated for every change to "yaw", "pitch", "roll" input as well as for change in y coordinate of a step.

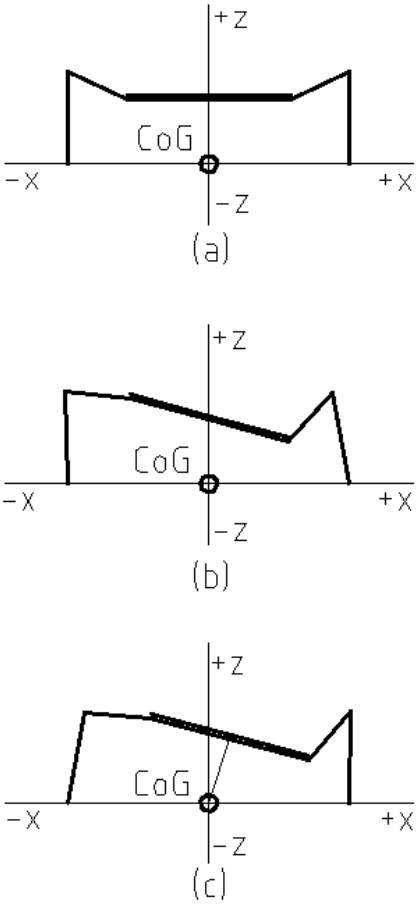


Figure 9: *The project into the plane z-x. The figure* (a) *is the base position, figure* (b) *is infuence of the "roll" and* (c) *is the effect of including the "swaying"*

The drawing in the figure 9(a) shows the basic position of the robot, with no movement function applied. The CoM corresponds with CoG and the normalized length of gravity vector is 1. The gravity vectors of shoulder pivots are all the same, having the same normalized length as the gravity vector of CoG. The figure 9(b) shows effect of the roll input. The robot rolls to the right, the properties of gravity vector for CoG do not change but the gravity vectors of shoulder pivots will change. The normalized length of the vectors in the interval $(0, +x))$ will be < 1 while the normalized length of gravity vectors from interval $(-x, 0)$ will be > 1. The same is true for pitch input, the only change is that coordinate x is swapped for coordinate y. The last section is the section (c) in the figure 9. This sction shows the effect of roll combined with swaying input. In this case the properties of the gravity vector for CoG will differ as well, in the contrary to the section (b). The normalized length of the new gravity vector of CoM will be < 1, while the properties of gravity vectors for the shoulder pivots will remain the same.

Applied Kinematics The kinematics solution for this project adopted several simplifications. First of all, the firmware does not implement roll, pitch and sway properties in a generalized way through rotational matrices. The only generalized function remains then "yaw". The effect of roll, pitch and sway is pre-calculated and stored in tabulated form in the robot's firmware. By adopting this limitations the movement of the robot can be described by a vector $\vec{\mathcal{U}} = [x, y, z]$, which is placed inside an Cartesian system $\mathfrak{R}^3$ described by $[x, y, z]$ as well. In other word, six coordinates are required to describe the movement. First we establish the origin on the vector $\vec{\mathcal{U}}$ in the Cartesian system $\mathfrak{R}^3$ and three other coordinates describes its orientation in the same space.

> Let us set the base position of the robot's body in Cartesian space $\mathfrak{R}^3$. The y characterizes the forward/backward movement, x sideways direction and z defines up/down movement of the robot.

The crawler has a certain height given by the physical dimensions of the construction. The vector $\vec{\mathcal{U}}$ defines the characteristics of the movement relative to the projection of the robot's CoG into walking plane.

The element z is always considered zero and

the motion of the robot is therefore calculated for the plane $[x, y]$ only. This simplification is acceptable for this kind of hobby project. The robot cannot climb and walk side-slopped plane. The change in the z axis is required for tilting and lifting legs and those operations are controlled by adding constant values of z to complete the position vector of each leg. The input is controlled by a tabulated values specific for each leg and phase.

The simplified kinematics approach chosen for this project follows this schema:

1. We assume the robot crawls forward. It means that we identify the new point y, which is different than the current value of y. The change in y-coordinate defines crawling forwards, backwards respectively. New position can be expressed as a vector $\vec{\mathcal{U}} = [x, y, z]$. Let us call it a motion vector of the whole robot. Let us establish that the motion vector is relative to the CoG and is having the coordinates $x{=}z{=}0$. In other words, the motion vector described the movement of the CoG on the plane $[x, y]$. This simplification will allow to use motion vector for robot expressed in polar coordinates by its length $\|\vec{\mathcal{U}}\| = \Delta$ and angle ϕ. The angle direction is clockwise. Zero angle $\phi = 0$ means that the robot crawls straight forwards, $\angle\phi = 45°$ means the robot crawl sideways left-forwards and so on. The angle ϕ is the "yaw" input. The resulting length of step Δ is kept constant for further movement, which is an unnecessary simplification of the solution. That can be easy overcome in next phases of the project. The $\angle\phi$ is variable and thus the only result of this step.

2. The motion vector for all fourth legs $\vec{\mathcal{P}}_i$ of the robot is derived from the robot's motion vector $\vec{\mathcal{U}}$. Generally, each vector $\vec{\mathcal{P}}_i$ is having the same direction ϕ as $\vec{\mathcal{U}}$ but the length is different for all of them $\|\vec{\mathcal{P}}_1\| \neq \|\vec{\mathcal{P}}_2\| \neq \|\vec{\mathcal{P}}_3\| \neq \|\vec{\mathcal{P}}_4\|$. This is because of the gait schema 1-3-2-4, as described in chapter 1.1. First the rules valid for basic walking pattern:

$$\begin{aligned} x &= 0 \\ y &= \begin{cases} +\Delta/2 \\ -\Delta/2 \end{cases} \\ z &= 0 \end{aligned} \tag{8}$$

where the variable Δ is the length of the step (see figure 4 for reference) determined in the previous step. The x,z are set to zero initially. The only variable for the basic calculation is y, which can take value from the interval $y \in \left(-\Delta/2; +\Delta/2\right)$. The exact value for the phases of the individual legs is determined based on the equation 1. As already mentioned, this schema is frequently referred as 1-3-2-4.

Each cycle is split into 4 segment, 3 are duty segments (cycle phases) and one is not. Each segment is further divided into 3 subsections. This fine segmenting helps to achieve smooth stride.

Table 1: *Relative phase for $\beta = 0.75$. The gray cells indicate the non-duty part of the cycle. Dark gray is full non-duty segment, light shade of gray means that a change from duty to non-duty (or vice-versa) is just happening at the given index.*

	0	1	2	3	4	5	6	7	8	9	10	11
φ_1	$\frac{9}{9}$	$\frac{6}{9}$	$\frac{3}{9}$	$\frac{0}{9}$	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{3}{9}$	$\frac{4}{9}$	$\frac{5}{9}$	$\frac{6}{9}$	$\frac{7}{9}$	$\frac{8}{9}$
φ_2	$\frac{3}{9}$	$\frac{4}{9}$	$\frac{5}{9}$	$\frac{6}{9}$	$\frac{7}{9}$	$\frac{8}{9}$	$\frac{9}{9}$	$\frac{6}{9}$	$\frac{3}{9}$	$\frac{0}{9}$	$\frac{1}{9}$	$\frac{2}{9}$
φ_3	$\frac{6}{9}$	$\frac{7}{9}$	$\frac{8}{9}$	$\frac{9}{9}$	$\frac{6}{9}$	$\frac{3}{9}$	$\frac{0}{9}$	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{3}{9}$	$\frac{4}{9}$	$\frac{5}{9}$
φ_4	$\frac{0}{9}$	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{3}{9}$	$\frac{4}{9}$	$\frac{5}{9}$	$\frac{6}{9}$	$\frac{7}{9}$	$\frac{8}{9}$	$\frac{9}{9}$	$\frac{6}{9}$	$\frac{3}{9}$

The table 1 describes the normalized phases of cycle for the basic gait, having $\beta = 0.75$. Variable k is established as the cycle index. The cycle index ranges 0 through 11, which is the product of having 4 main segments further divided into 3 sub-segments. The reference leg is chosen leg 4. The cycle index is set to zero at the moment when the leg No.4 is placed on the surface in accordance with the theory described in chapter 1.1. The cycle begins.

Corresponding equation to determine the element y takes form:

$$y_{i,k} = \frac{\Delta}{2} - \varphi_{i,k}\,\Delta \tag{9}$$

A little more complicated is the swaying pattern, which additionally utilizes changes in the x axis to imitate swaying motion:

$$x = \begin{cases} +\Delta/3 \\ -\Delta/3 \end{cases}$$
$$y = \begin{cases} +\Delta/2 \\ -\Delta/2 \end{cases} \qquad (10)$$
$$z = 0$$

The y coordinate in the vector can again take values from the interval $y \in \left(-\Delta/2; +\Delta/2\right)$. Additionally, the x is not zero but takes values from the interval $x \in \left(-\Delta/3; +\Delta/3\right)$. The exact values are coordinated for each phase of the cycle between both x and y. This schema is as well referred as 1-3-2-4 and the segmenting of a cycle is the same as the basic crawl. The timing table for vector coordinate y is exactly the same as the table 1. The coordinate x is synchronized with the table 1 and takes form captured in the table 2.

Table 2: *Relative phase for swaying component ξ at $\beta = 0.75$.*

	0	1	2	3	4	5	6	7	8	9	10	11
ξ_k	$\frac{0}{9}$	$\frac{3}{9}$	$\frac{6}{9}$	$\frac{9}{9}$	$\frac{6}{9}$	$\frac{3}{9}$	$\frac{0}{9}$	$\frac{-3}{9}$	$\frac{-6}{9}$	$\frac{-9}{9}$	$\frac{-6}{9}$	$\frac{-3}{9}$

and the corresponding look up equation:

$$x_k = \frac{\Delta}{3}\,\xi_k \qquad (11)$$

Table 3: *Relative phase for $\beta = 0.75$, advanced swaying pattern. Legend is identical with table1*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
φ_1	$\frac{9}{9}$	$\frac{6}{9}$	$\frac{3}{9}$	$\frac{0}{9}$	$\frac{0}{9}$	$\frac{0}{9}$	$\frac{0}{9}$	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{3}{9}$	$\frac{4}{9}$	$\frac{5}{9}$	$\frac{6}{9}$	$\frac{6}{9}$	$\frac{6}{9}$	$\frac{6}{9}$	$\frac{7}{9}$	$\frac{8}{9}$
φ_2	$\frac{3}{9}$	$\frac{4}{9}$	$\frac{5}{9}$	$\frac{6}{9}$	$\frac{6}{9}$	$\frac{6}{9}$	$\frac{6}{9}$	$\frac{7}{9}$	$\frac{8}{9}$	$\frac{9}{9}$	$\frac{6}{9}$	$\frac{3}{9}$	$\frac{0}{9}$	$\frac{0}{9}$	$\frac{0}{9}$	$\frac{0}{9}$	$\frac{1}{9}$	$\frac{2}{9}$
φ_3	$\frac{6}{9}$	$\frac{7}{9}$	$\frac{8}{9}$	$\frac{9}{9}$	$\frac{9}{9}$	$\frac{9}{9}$	$\frac{9}{9}$	$\frac{6}{9}$	$\frac{3}{9}$	$\frac{0}{9}$	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{3}{9}$	$\frac{3}{9}$	$\frac{3}{9}$	$\frac{3}{9}$	$\frac{4}{9}$	$\frac{5}{9}$
φ_4	$\frac{0}{9}$	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{3}{9}$	$\frac{3}{9}$	$\frac{3}{9}$	$\frac{3}{9}$	$\frac{4}{9}$	$\frac{5}{9}$	$\frac{6}{9}$	$\frac{7}{9}$	$\frac{8}{9}$	$\frac{9}{9}$	$\frac{9}{9}$	$\frac{9}{9}$	$\frac{9}{9}$	$\frac{6}{9}$	$\frac{3}{9}$

Table 4: *Relative phase for advanced swaying component ξ at $\beta = 0.75$.*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ξ_k	$\frac{0}{9}$	$\frac{-3}{9}$	$\frac{-6}{9}$	$\frac{-9}{9}$	$\frac{-3}{9}$	$\frac{3}{9}$	$\frac{9}{9}$	$\frac{6}{9}$	$\frac{3}{9}$	$\frac{0}{9}$	$\frac{3}{9}$	$\frac{6}{9}$	$\frac{9}{9}$	$\frac{3}{9}$	$\frac{-3}{9}$	$\frac{-9}{9}$	$\frac{-6}{9}$	$\frac{-3}{9}$

The swaying is the same for all the legs within the cycle, therefore a one-row-table is sufficient to describe the dependency. The resulting motion model for each leg looks different (see figure 5) because of the phase shift among the legs. The more complicated swaying pattern, displayed in the drawing 6, requires 18 indices in the "relative phase" table. It must be again synchronized between x and y input, therefore both tables must consists of 18 stops. Tables 3 and 4 documents the extended lookup tables for advanced swaying documented in the drawing (6).

3. The leg-vectors produced in last steps are rotated for the angle ϕ established in the first step. That is the "yaw" function and determine the direction of movement function in the walking frame. As a result we yield the rotated motion vectors $\vec{\mathcal{R}}_i$, where i is the index of the leg.

$$\vec{\mathcal{R}}_i = \vec{\mathcal{P}}_i \times \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (12)$$

4. The rotated motion vector $\vec{\mathcal{R}}_i$ for all legs are still related to body frame of the robot. In this step we apply the transformation to translate the motion vector into specific servo frames. This operation involves translation and rotation. The index point for servo frame is the shoulder joint. The distance between a foot and shoulder joint is the translation constant. The translation follows the geometry from figure 2 and 3. The absolute value of the translation further depends on the physical dimensions of robot's legs. The frog geometry takes $\vec{\mathcal{D}}_i = \left[\delta_x^i, \delta_y^i, \delta_z^i\right] = [\pm 36, \pm 36, +51]$. The crab like geometry takes translation $\vec{\mathcal{D}}_i = \left[\delta_x^i, \delta_y^i, \delta_z^i\right] = [\pm 51, 0, +51]$. The constant shift in the coordinate z is given by the physical construction of the leg and sets the height of the CoG above the walking plane. The figure 10 demonstrates the effect of the z coordinate on the walking geometry.

The coordinate z does not include any input

from either “sway” or “tilt” component yet.

Table 5: *signs for legs.*

	Leg			
	1	2	3	4
x	−	+	+	−
y	+	+	−	−

The sign in the translation vectors is important and determines in which quadrant the translation takes place. See the drawing in the figure 11 for better orientation regarding the quadrants. The unit used is millimeter.

Using the figure 11 and figures 3, 2 the table 5 is constructed. The table indicates, in which direction the translation of foot relative to shoulder pivot takes place.

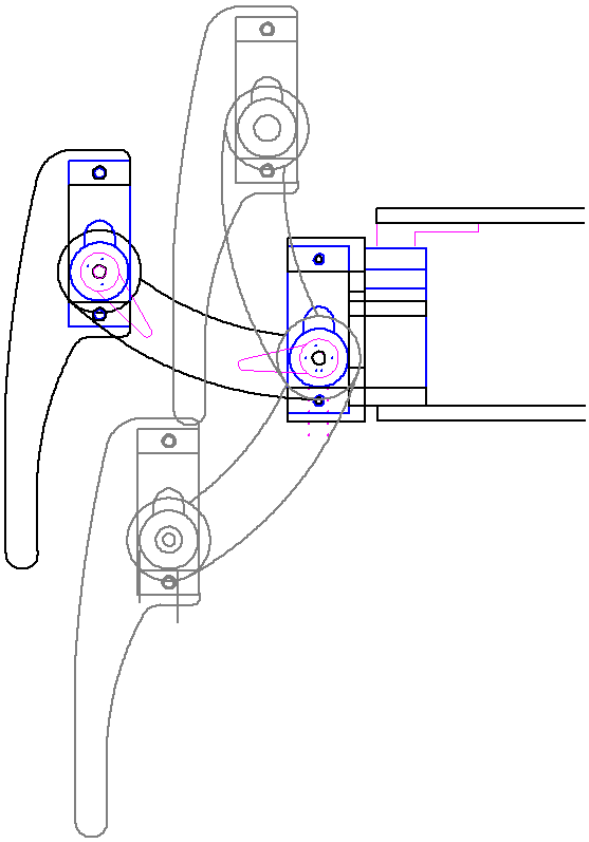


Figure 10: *The height of the CoG above the walking plane. The gray shadows show both minimal and maximal height. The optimal position is depicted in the full color schema. The height of the robot's CoG is controlled by a constant value - $h_0^z = 51\,mm$.*

Resulting translation vectors $\vec{\mathcal{D}}_i$ for frog/lizard like geometry produced by using the input from table 5:

$$\left[\delta_x^1, \delta_y^1, \delta_z^1\right] = [-36, +36, +51]\ mm \quad (13)$$
$$\left[\delta_x^2, \delta_y^2, \delta_z^2\right] = [+36, +36, +51]\ mm \quad (14)$$
$$\left[\delta_x^3, \delta_y^3, \delta_z^3\right] = [+36, -36, +51]\ mm \quad (15)$$
$$\left[\delta_x^4, \delta_y^4, \delta_z^4\right] = [-36, -36, +51]\ mm \quad (16)$$

Similarly for the crab geometry. The resulting vector $\vec{\mathcal{R}}_i^{\mathcal{D}}$ is processed further in next step.

5. In oder to complete the transposition into servo frame the rotation is performed. This rotation reflects the fact that the orthogonal system of each leg (servo frame *fS*) is rotated in relation to the base orthogonal system describing the robot's body (body frame *fB*).

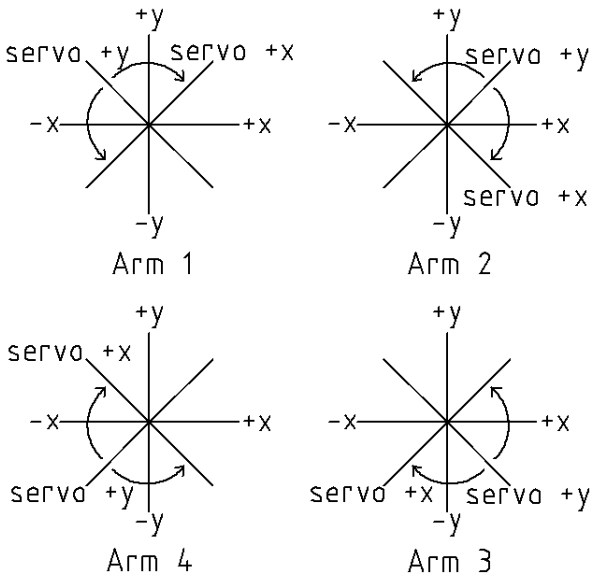


Figure 11: *Mount point of legs on the robot's body. The schema helps to determine the sign for displacement of the motion vectors and further additional rotation specific for shoulder pivot actuators.*

In other words the figures 8 and 11 shows the relationship between the servo frames, body frame and walking frame. The relationship can be described by single angle only, named ψ_i. Based on the figure 11 we can determine the angle for each leg:

$$\begin{aligned}\psi_1 &= 45^\circ \\ \psi_2 &= 315^\circ \\ \psi_3 &= 225^\circ \\ \psi_4 &= 135^\circ\end{aligned} \tag{17}$$

The rotation operation takes form of again the affine matrix. The z is constant, just elements x,z are subject to rotate.

$$\vec{\mathcal{T}}_i = \vec{\mathcal{R}}_i^{\mathcal{D}} \times \begin{bmatrix} cos(\psi_i) & -sin(\psi_i) & 0 \\ sin(\psi_i) & cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

The resulting vector $\vec{\mathcal{T}}_i$ includes direction angle ϕ, displacement contribution $\vec{\mathcal{D}}_i$ and last not least the "mount point angle ψ_i", which describes the relationship between orthogonal system of the robot's body and the plane of a leg.

6. Until now we ignored the effect of "roll" and "pitch" functions. This simplified approach is simulating the input of those two functions though a set of tabulated values. The tabulated values also mix in the effect of change in z-coordinated required to lift and place feet on walking plane. Both lifting/placing and tilting affects in this simplification coordinate z only. The project uses tabulated values again to implement leg lifting. The table 6 was used to capture the lifting and placing the leg.

Table 6: *Complementary data for Relative phase for $\beta = 0.75$. This table defines the change in the z-axis for each leg.*

	0	1	2	3	4	5	6	7	8	9	10	11
1	0	9	9	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	9	9	0	0	0
3	0	0	0	0	9	9	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	9	9

The z element can also include the titling input. This input can be integrated inside the complementary z-table and the resulting table takes forms shown in the table 7.

After including lifting/placing and tilting input we obtain the complete motion vectors $\vec{\mathcal{M}}_i$ for all legs of the robot.

Table 7: *Complementary data for Relative phase for $\beta = 0.75$. This table defines the change in the z-axis for each leg including titling effect.*

	0	1	2	3	4	5	6	7	8	9	10	11
1	−12	9	9	9	12	12	0	0	0	0	0	0
2	0	0	0	0	0	0	−12	9	9	9	12	12
3	9	12	12	−12	9	9	−10	0	0	0	0	0
4	−10	0	0	0	0	0	9	12	12	−12	9	9

Advanced Crawl having the $\beta = 0.83$ requires an extra attention. It basically follows exactly the same logical sequence however the calculation of component y and timing together with z is more complicated.

Table 8: *Relative phase for $\beta = 0.83$. The gray cells indicate the non-duty part of the cycle. Dark gray is full non-duty segment, light shade of gray means that a change from duty to non-duty (or vice-versa) is just happening at the given index.*

	0	1	2	3	4	5	6	7	8	9	10	11
φ_1	$\frac{18}{18}$	$\frac{10}{18}$	$\frac{4}{18}$	$\frac{6}{18}$	$\frac{7}{18}$	$\frac{8}{18}$	$\frac{9}{18}$	$\frac{11}{18}$	$\frac{13}{18}$	$\frac{15}{18}$	$\frac{16}{18}$	$\frac{17}{18}$
φ_2	$\frac{9}{18}$	$\frac{11}{18}$	$\frac{13}{18}$	$\frac{15}{18}$	$\frac{16}{18}$	$\frac{17}{18}$	$\frac{18}{18}$	$\frac{10}{18}$	$\frac{4}{18}$	$\frac{6}{18}$	$\frac{7}{18}$	$\frac{8}{18}$
φ_3	$\frac{12}{18}$	$\frac{14}{18}$	$\frac{16}{18}$	$\frac{18}{18}$	$\frac{8}{18}$	$\frac{2}{18}$	$\frac{3}{18}$	$\frac{5}{18}$	$\frac{7}{18}$	$\frac{9}{18}$	$\frac{10}{18}$	$\frac{11}{18}$
φ_4	$\frac{3}{18}$	$\frac{5}{18}$	$\frac{7}{18}$	$\frac{9}{18}$	$\frac{10}{18}$	$\frac{11}{18}$	$\frac{12}{18}$	$\frac{14}{18}$	$\frac{16}{18}$	$\frac{18}{18}$	$\frac{10}{18}$	$\frac{2}{18}$

The table 8 describes the relative phase withing the cycle for all legs. It again uses 12 stops (0 through 11) and in the contrary to the table 1 it contain sections, where all 4 legs support the robot at time. The transition between stops 2 and 3 is one of the moments, where all 4 legs rest on walking area. The same is true for transitions $5 \rightarrow 6, 8 \rightarrow 9, 11 \rightarrow 0$.

There is one more abnormality in the "$\beta = 0.83$ schema". A varying length of single steps withing a complete stride must be used in order to achieve the cyclic behavior of the walking pattern. In a complete cycle each legs makes two full-length-steps and two half-length-steps. This behavior is clear to detect in the table 8. Looking at the reference leg 4. It starts it first stride-segment at index 0 and it takes thee stops to complete this segment. The length of the corresponding relative trajectory for the first segment (indices 0 though 3) is ${}^{6}/_{18} \equiv {}^{1}/_{3}$, which is the full-length step. The next section starts at the index 3 and takes three stops to complete the second segment.

Table 9: *Position of legs depending on the index* k. *The table indicates length of step for all legs depending on the index value*

	relative phase for Leg				step
Index	1	2	3	4	incr.
0	$^{18}/_{18}$	$^{9}/_{18}$	$^{12}/_{18}$	$^{3}/_{18}$	
⋮	⊓				+1
3	$^{6}/_{18}$	$^{15}/_{18}$	$^{18}/_{18}$	$^{9}/_{18}$	
⋮			⊓		$+^{1}/_{2}$
6	$^{9}/_{18}$	$^{18}/_{18}$	$^{3}/_{18}$	$^{12}/_{18}$	
⋮		⊓			+1
9	$^{15}/_{18}$	$^{6}/_{18}$	$^{9}/_{18}$	$^{18}/_{18}$	
⋮				⊓	$+^{1}/_{2}$
12=0	$^{18}/_{18}$	$^{9}/_{18}$	$^{12}/_{18}$	$^{3}/_{18}$	

The length of the corresponding relative trajectory for this second segment (indices 3 though 6) is $^{3}/_{18} \equiv {}^{1}/_{6}$, which is the half-length step. This pattern is repeated until the whole cycle gets completed. Table 9 shows the same behavior by swapping the axis x and y in the table.

Speaking about the duty factor $\beta = 0.75$, the entire cycle is divided into 4 equal segments, which is very convenient for a 4-legged-robot. Within the cycle must each leg cover the distance twice as long as the length of step Δ. Each leg covers the full length of step in 3 duty segments of the cycle and again the full length of step in lifted position, when being transferred at the start position again. The normalized "distance" between two stops for duty segment is a constant number for all legs. The duty factor $\beta = 0.83$, in the contrary, cannot use such a simple model. The problem is that we still have 4 legs only but we need to combine two different modes. First, all 4 legs are supporting the robot within the fraction of a step and second, just three legs support the robot while the fourth is lifted and moved in start position again. That is why we need a non-linear segmenting, combination of short and long steps, plus different length of step for front and hind legs. Considering the three named method a non-linear dependency is tabulated and stored in table 8.

The figure 12 shows the profile for non-duty segments. Actually, the non-duty segment is

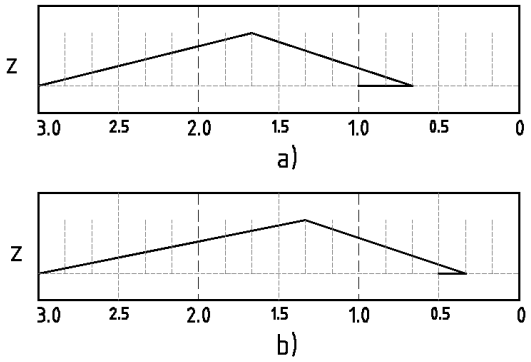


Figure 12: *Profile of a non-duty phase for a leg for duty factor* $\beta = 0.83$. *The figure* a) *shows the trajectory for front leg while the figure* b) *shows the trajectory for hind leg.*

partially a duty segment, as shown in the figure. It is obvious that the full length of step is not utilized for the higher duty factor $\beta = 0.83$. The front legs cover just 78% of the length of step while the hind legs cover 89% of the length of step. The duty factor $\beta = 0.75$ on the other hand uses linear schema, where each arm is lifted once it reaches relative position 3 and placed back int relative position 0.

One more aspect must be taken into consideration for the schema with duty factor $\beta = 0.83$. The angular velocity of each leg must be adjusted for short and long segment within a step so that the linear velocity of the whole robot remains constant. Just reminding here that the design is using 12 stops to split the cycle. The first 9 stops cover the duty segment, which is part of cycle that affects the movement of the robot. Theoretically, we can use the above outlined schema to calculate the coordinates for the foot of any leg at any position within the Cartesian system. Realistically, we use the quantisation, where the smallest discreet quantum is $^{\Delta}/_{9}$. The design must be however able to control the duration of the operation of moving a leg for the $^{\Delta}/_{9}$ discreet distance. Only then the linear movement of the whole robot can be kept smooth and constant. The design harnesses linear regression to achieve controlled transition of a foot for the distance $^{\Delta}/_{9}$. The linear regression is adequate approximation method to calculate the immediate positions for a foot within two calculated points on a motion vector $\vec{\mathcal{M}}_i$.

Inverse Kinematics Each leg constitutes of three links (Coxa, Femur, Tibia) and three actuators (Body-Coxa, Coxa-Femur, Femur-Tibia). The first actuator connects the leg to the body. It is a shoulder

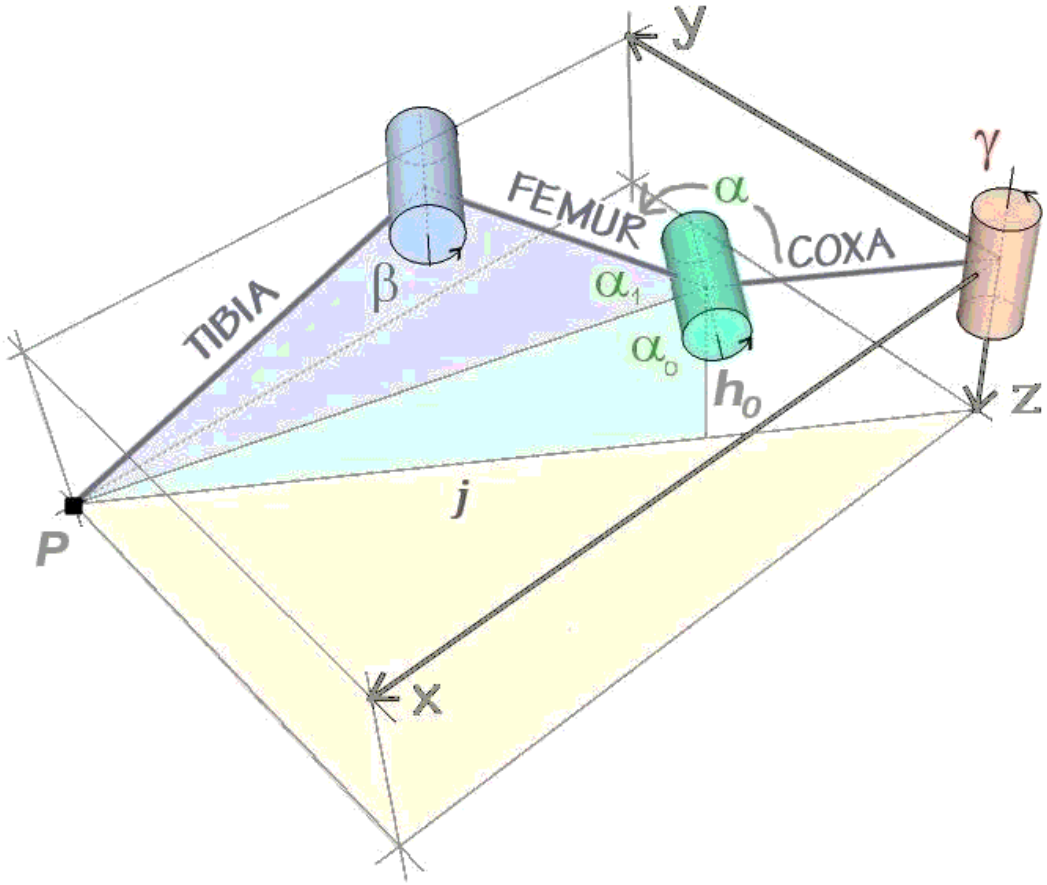


Figure 13: *Model of the leg. The model shows the conversion of the Cartesian system into angles α, β, γ applied to the actuators in order to place foot of a leg into position $P\,[x, y, z]$.*

$$\gamma = tan^{-1}\left(\frac{x}{y}\right) \tag{19}$$

$$\alpha_0 = tan^{-1}\left(\frac{\sqrt{x^2+y^2} - COXA}{h_0}\right) \tag{20}$$

$$\alpha_1 = cos^{-1}\left(\frac{FEMUR^2 + \left(\sqrt{x^2+y^2} - COXA\right)^2 + h_0^2 - TIBIA^2}{2\,FEMUR\sqrt{\left(\sqrt{x^2+y^2} - COXA\right)^2 + h_0^2}}\right) \tag{21}$$

$$\beta = cos^{-1}\left(\frac{TIBIA^2 + FEMUR^2 - h_0^2 - \left(\sqrt{x^2+y^2} - COXA\right)^2}{2\ FEMUR\ TIBIA}\right) \tag{22}$$

pivot and is referred as γ. This angle controls the rotation of the whole leg in the plane parallel to the $[x, y]$ place of the robot's body. The first link is called Coxa, is usually rather short and connects shoulder pivot with the elbow. Another actuator is placed between Coxa and Femur. The angle is called α and allows vertical movement of the leg. The Femur is connected to the Tibia with another actuator. It allows the set the angle between them. The angle is called β and operates in vertical plane as well. This angle allows to place the foot near or far to the CoG. Both angles α, β operate in the same plane, which is perpendicular to the robot's base plane. All the links in the leg has a fixed length. The actuators connecting the links are standard servos. The output horn of a servo can be set for angle $0 - 180°$. Therefore a coordinate conversion between $[x, y, z]$ and angles α, β, γ must be provided.

The conversion can be based on standard trigonometric functions and law of cosines. The details of the conversion process are outlined in the figure 13 and equations 19 though 22.

By establishing the conversion mechanism between the Cartesian system of motion vector and angles required by servos used in legs is the chapter of kinematics complete. The next section describes the details of the construction itself.

1.3 Mechanical Design

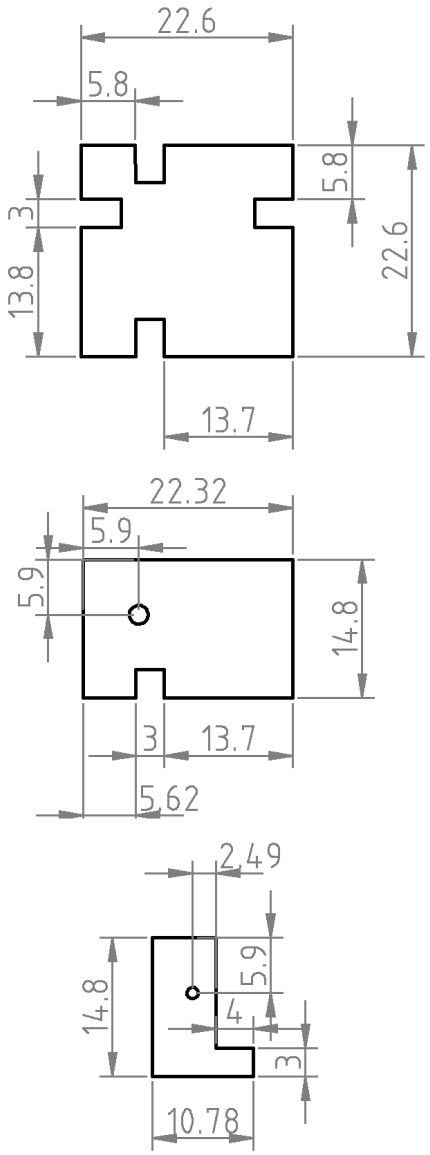


Figure 14: *Design of the servo compartment that forms Coxa link.*

3 mm thick plywood was used in this project. The mechanical properties are adequate for the size of the construction. Plywood is rather light material compared to aluminium or plastic, which positively contributes to the overall weight of the construction. The pillars connecting top and bottom plate of the robot are from Nylon (polyamide) rod, having the diameter of 8 mm. This material is strong enough to allow thread cutting. I used M2 screws to connect the top and bottom plates to the supporting Nylon pillars. The size of the robot was kept as minimal as possible in order to reduce the weight. Low overall weight allows usage of low-budget hobby servos, which will be described in chapter 1.4. The limiting factors are the size of the main board and the size of the servos used in legs.

The tool LibreCAD, formerly known as QCAD. This tool provides semi-professional level for technical drawings, is free under GPLv2 license. The drawings were printed out and transferred to the plywood.

The small mechanical jigsaw was used to cut the parts. The joins were made from a graphic rod revolving in a plywood dry bearing. No classic bearing were necessary, the carbon properties combined with smooth surface of used 2 mm trick rods were meeting the requirements for this application. The complete construction in shown in the drawing depicted on the page 16, figure 16. The black color is used for plywood parts, main board and servos are outlined in blue and mechanical links (servo horns) are violet. The size of the main board exactly fits within the supporting pillars interconnecting the top and bottom plate.

The details of the leg, showing all the construction parts, are in the figure 15. The color-code is identical with the drawing in the picture 16. The longest part of the leg is Tibia, which is connected to the servo mounts by two screws. This joint between Tibia and Femur forms the servo output shaft only. That installs considerably high shear on the output shaft of the servo. The shear can be minimized by shortening up the distance between the mounting point of tibia on the servo body and the servo horn. The shortest distance can be achieved by mounting the tibia on the top of the servo mounts. The price is however that the rotation angle γ must be shifted for certain constant. The reason is that the theoretical model assumes that the projection of the connecting line between tip of tibia P and the output shaft of the shoulder pivot onto the walking plane is perpendicular to the longitudinal axis of the servo casing (top view).

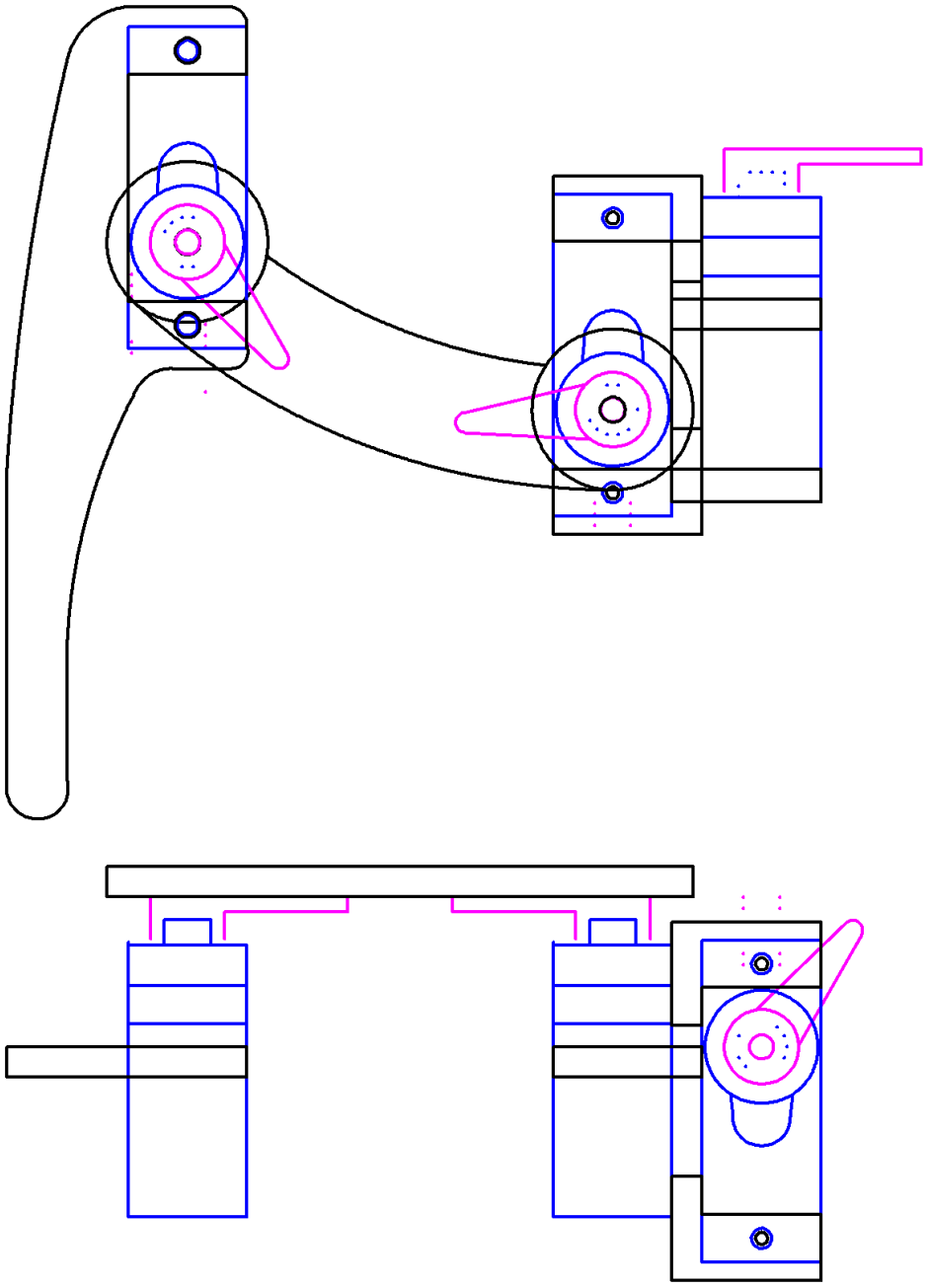


Figure 15: *The construction of the leg. Black is plywood, blue color for servos, magenta for servo horns.*

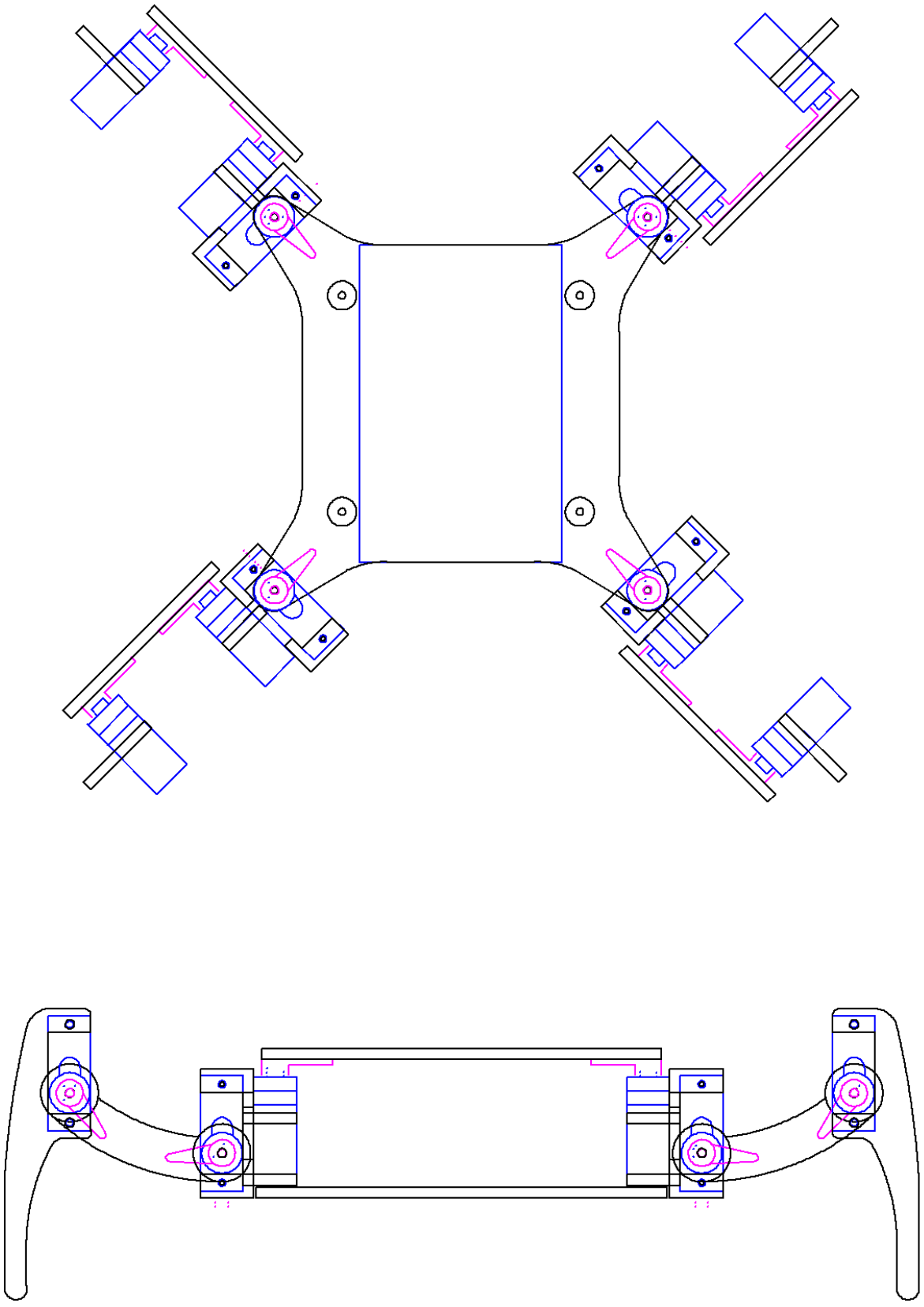


Figure 16: *The robot. This figure shows the horizontal projection and profile of the construction. The blue rectangle in the center of the robot's body is the main board.*

Alternatively an auxiliary pivot can be installed into the servo case opposite to the output shaft for servos on both ends of Femur link and duplicating the Femur itself. This design idea was however not easy to implement with the used low-budget servos (TowerPro 9g), which have very thin and fragile plastic case. My attempt for adding a small pivot made from 2 mm carbon rod were not mechanically firm enough to support the weight of the construction. Although the output shaft of the used servos is not set in bearings it still provides enough strength to withhold the shear and support the whole construction. I believe the lifetime of servos will be rather short, but it is worth the experiment.

The same shear forces applies also the servo connecting the Femur with Coxa. The joint is implemented by means of the servo output shaft only. It is the same design pattern as used in the joint between Tibia and Femur. The Coxa itself is very short, corresponding to the width of the servo (11.8 mm) plus thickness of the plywood (3 mm). The shoulder servo that rotates the legs in the walking plane. This servo is using the pivot virtually extending the longitudinal axis of the output shaft. The pivot is however not mounted directly in the servo case but an additional plywood buffer layer is used to expand the height of the servo case. This can happen because of the sufficient distance between the top and bottom plate of the robot.

The only "complicated" part is the servo compartment forming the Coxa link. The shoulder pivot servo and perpendicular servo lifting the Femur are attached together by means of a light weighted compartment made from plywood. The schema and details of the compartment is documented on the figure 14. All parts are glued together by a common 2-component Epoxy. This technique provides sufficient strong links.

The plywood parts are blue painted.

1.4 Actuators

This project uses servos to implement the joints in the legs of the robot. Servos are controlled by sending them a pulse of variable width. The parameters of the pulse is *minimal width*, *maximum width* and *repetition rate*.

Parameters to control a servo are standardized

Figure 17: *Controlling a servo by a pulse.*

and most of the available servos still uses the same PWM modulation to rotate the output shaft. A typical servo requires a control pulse having the width ranging from 544μs to 2400μs, where 544μs set the servo to the angle position 0° and 2400μs to the angle position 180°. The 1472μs sets the centered position, 90°, for most servos. Sometimes the 1500μs pulse is used to center the shaft instead. The required repetition rate is usually 20 ms for most servos. Refer to figure 17 for overview.

The dependency between the width of the pulse and the angle is linear and thus the equation for ideal servo takes a simple form:

$$y = 590.87x + 544 \tag{23}$$

Where the input x is the angle in radians and output y is the width of control pulse in μs.

Typical servos are designed for control pulses ranging from 800μs through 2200μs. It means that 0 and 180 degree limits are theoretically not achievable. Sometimes the servo sets the maximal or minimal angle even for shorter width of the pulse and still keeps the dependency linear. Sometime simply the maximum and minimum angle cannot be achieved. Each and every made of servos might behave slightly different way.

The quadpod design is aware of this limitation and using a corrected equation to determine the angles based on the known control pulses. The experiment for the used servos (9g TowerPro) yielded the equation in this form:

$$y = 636.4x + 452.4 \tag{24}$$

where the units and semantics is identical with the equation 23. Having the linear dependence between the angle of the output shaft and the width of control pulse. The equation 24 is included in the kinematics calculation in the firmware of the robot.

2 Electronics

The electronic section gives detail about the IR circuit, servo shield, power sources, controller and main board.

2.1 IR Receiver

The infrared protocol is used to control the movement of the quadpod. The reception and demodulation is done by a dedicated IC - OS1838, which is budget alternative to Vishal's TSOP31238. The circuit demodulates the incoming IR signal and sends inverted output to the microprocessor. The modulation frequency is 38 kHz. The signal is intercepted on the external interrupt pin INT0 and the interrupt handler implements a simple finite state machine (FSM) to decode the signal.

The used protocol is Mitsubishi XSat, which has following characteristic:

Figure 18: *Train of pulses characteristic for the Mitsubishi XSat Protocol.*

where the modulation for logical "1" and "0" is shown in figure 19.

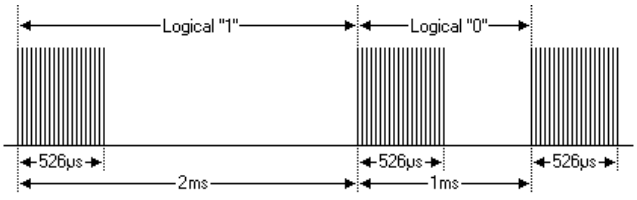


Figure 19: *Modulation of the Mitsubishi Protocol.*

The finite state automata for signal decoding is throwing the address away, it decodes the instruction only. An old remote controller was used in this project, presented on the figure 20. The logic analyser "OpenBench LogicSniffer" in combination with software "LogicAnalyser" was used to identify the IR protocol and interpret the instruction codes send by remote. The finite state automata is delivering just the 8-bit instruction code as a product.

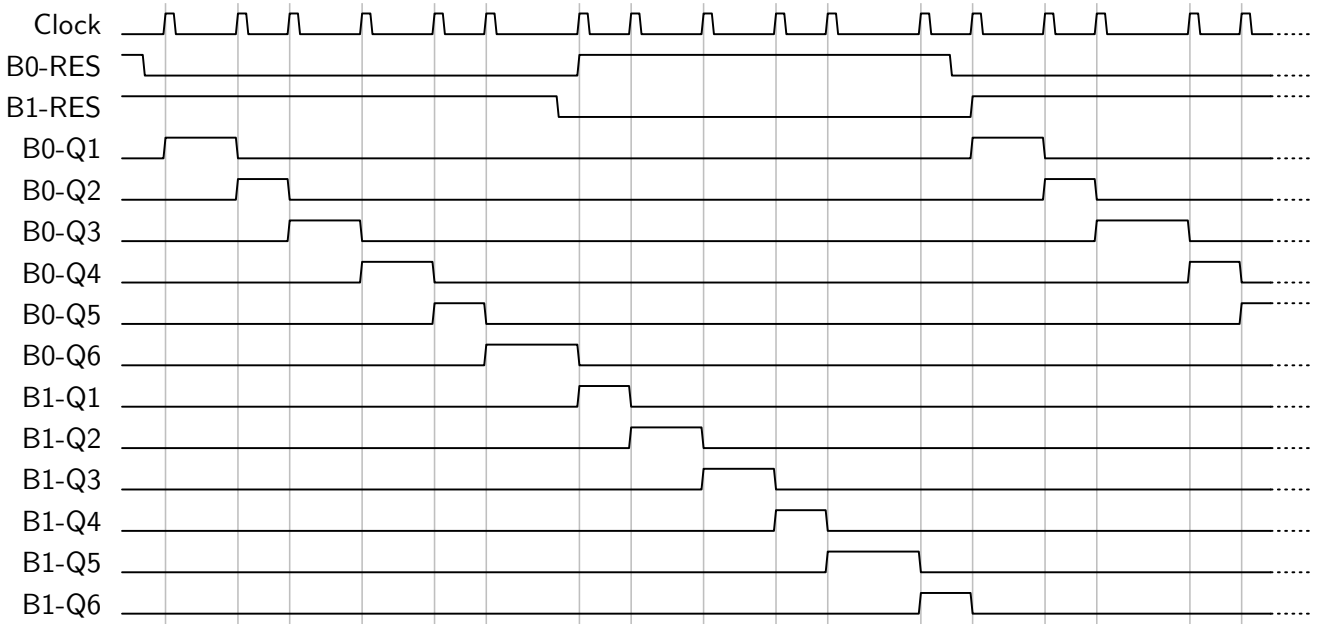
Figure 20: *Remote control.*

2.2 Servo Shield

Servo shield is a specialized circuit that drives the multiple servos in parallel. The 4-legged robot is using 3 servos per a single leg and therefore we need a servo shield that drives at least 12 servos in real time. There is variety of commercial servo driver circuits on the market. They are however delivered on a dedicated PCB and typically requires an external step down high current regulator for proper operation of servos. It would mean that three PCBs must be used in the solution, which is way too many. The outlined mechanical design of the robot however allows to place just one small PCB inside the body. The PCB is expected to host power circuits, controller itself and servo shields. A dedicated design is described in this project that satisfies the requirement.

Johnson decade counter with 10 decoded outputs, available under the code 74HC4017/74HCT4017,

Table 10: *Signal model for servo shield*



delivers a very good match in required functionality. It consists of a 5 D Flip-flop elements linked together and having additional logic that allows to use the circuit as a "shift register". The shift operation happens on rising edge of the clock signal. IC also offers RESET [3] signal, which can be used to cascade multiple ICs together is a coordinated manner. This project uses 2 Johnson decade counters as one large shift register to generate the control pulse for 12 servos. The theoretical maximum of two decade counters is actually 18 servos, practically the race hazard may rise for 12 controlled servos already. The point is that the control pulse to a servo may take 2 milliseconds or even more. Having 12 servo connected to the shift register we need more than 24 milliseconds to complete the cycle and start a new one. Strictly speaking, each servo requires the presence of new control pulse within 20 milliseconds. When the refresh rate drops bellow this rate, the servo may enter a "fail safe" condition, which would break the movement. Therefore the practical limit of this design is somewhere between 10 and 14 servos. The assumption is that there is no real position of all legs in which all servos would be preset to 180°. Therefore the race hazard with 12 servos is acceptable for the project.

The used design requires three output pins on the micro controller to drive both Johnson counters. One pin delivers common clock signal, two pins control the RESET signal for both counters. The clock signal is actually irregular. The controller is just strobing the CLK pin of both counters with a 2 μs short pulse. That is long enough for ICs to shift the register. Basically the time delay between two incoming strobes defines the length of a control pulse for one servo. The output pins Q1-Q6 on Johnson counter generates the pulses, whose length is equal to the time delay between two incoming strobes on CLK pin. This simple effect generates the train of pulses required to control servos. The signal generated on Q1-Q6 resembles the control signal from the figure 17.

Initially, both counters are reset. Reset condition for the first counter (first bank in shield) is released and shortly after release the first strobe comes. The strobe is shifting the register, which results in high state on Q1 for the first counter. The control pulse is sent to first controller servo. The next strobe on the CLK signal shift the register further, which sets Q1 again to zero and in parallel Q2 goes high. This repeats 6 times. The micro controller releases the reset signal for the second counter and after that resets the first counter after the 6^{th} cycle. The reset for first counter is sent together with the strobe pulse and thus acts as terminator for output pulse present on the 6^{th} output (Q6) of the first counter. The first counter remains reset and its output pins Q1-Q6 remains in low status. The reset condition for second bank is

[3] Certain versions of Johnson counter requires $\overline{\text{RESET}}$ signal to reset the circuit.

released well before the strobe comes and therefore the second counter outputs high level on its Q1 pin in the same clock cycle. See table 10 for timing details.

2.3 Controller

Initially, the brain of the robot was tested with two models of a 8-bit micro controllers. First Atmel ATMega8/16 MHz and second Atmel ATMega168/20 MHz. The final version of the firmware is slightly larger than 8kB and therefore the experiment continued with the second mentioned unit only. The unit corresponds with plenty GIO pins, though this design requires just three of them to control the servo shield. Additionally the TWI and USART and SPI buses are wired with corresponding connectors on the PCB just in case expansion would be required. The USART bus is used to debug the firmware.

2.4 Power Source

The design is using two independent power sources (see the schematic on the figure 22). One power source is used to power up the logic, the second source is powering up servos only. Both power circuits are not galvanic isolated, they share the ground lead in common. Although the servo motors may generate peaks on power line, the experiment did not detect any malfunction of the micro controller and other CMOS based logic. Further description is focused on the power source for servos only. It is built as a linear power source, using 2-stage power amplifier. The power amplifier is formed by resistors R6 through R10, one IC (LM317L) and two transistors, Q2 and Q3. The transistor Q2 is the first stage in the amplifier. I selected BC327-40, a PNP transistor designed for power amplification. It is important to use the version "BC327-40", which delivers DC current gain $h_{FE1} = 250 \sim 630$ and $h_{FE2} = 170$. The high value of h_{FE2} is important for this application. It delivers enough current into base of Q3, which is the second stage in the amplifier. This transistor is MJE3055, a power transistor capable of handling collector current $I_C = 10A$. The h_{FE} is typically between 20 and 100. This value is meeting the requirement for this application. The output voltage is regulated by adjustable voltage stabilizer, LM317L. The circuit is configured by R9 and R10 to deliver 4.8V under no load or little load only. The LM317L is the only source of current under such conditions. The transistor Q2 opens first after the voltage drop on R7 reaches $\sim 0.6V$. This is satisfied once the drawn output current climbs above $\sim 6mA$. The Q2 amplification factor starts working and delivers high current into base of Q3. Assuming the back EMF of load is well under the 4.8V the Q3 opens fully and delivers emitter current into load as high as 7-8A. The tests done with the 12 TowerPro servos under load were made. The drawn current was 5-6A. The power transistor MJE3055 requites active cooler while the thermal loses are ∼20W. That is rather high value but it is a penalty for using a linear power source.

2.5 Board

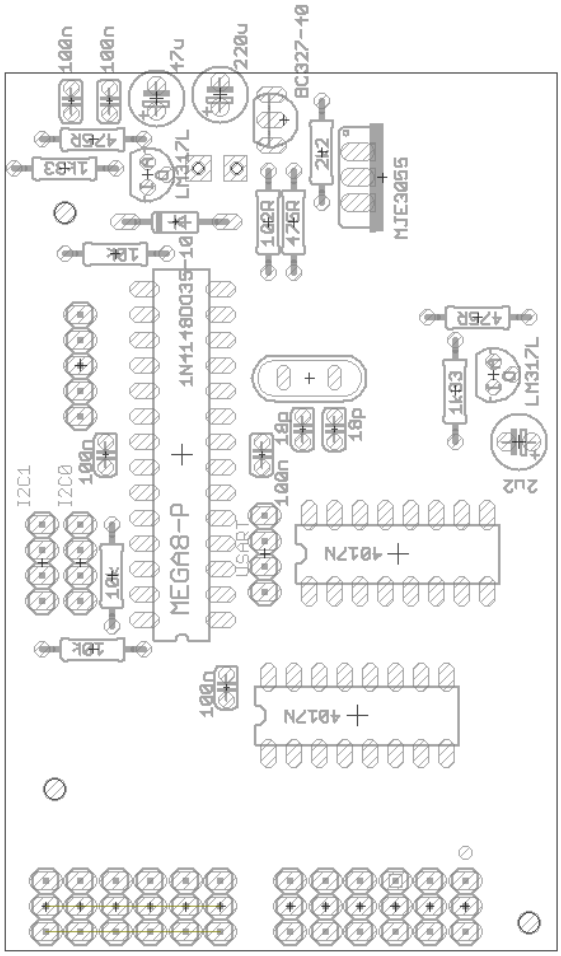


Figure 21: *Component layout diagram for the main board, top view.*

Board was designed in the Eagle 5.12 Light for Linux. The board size is $98 \times 55\,mm$ and it uses 2 conducting layers. The component layout diagram is the figure 21.

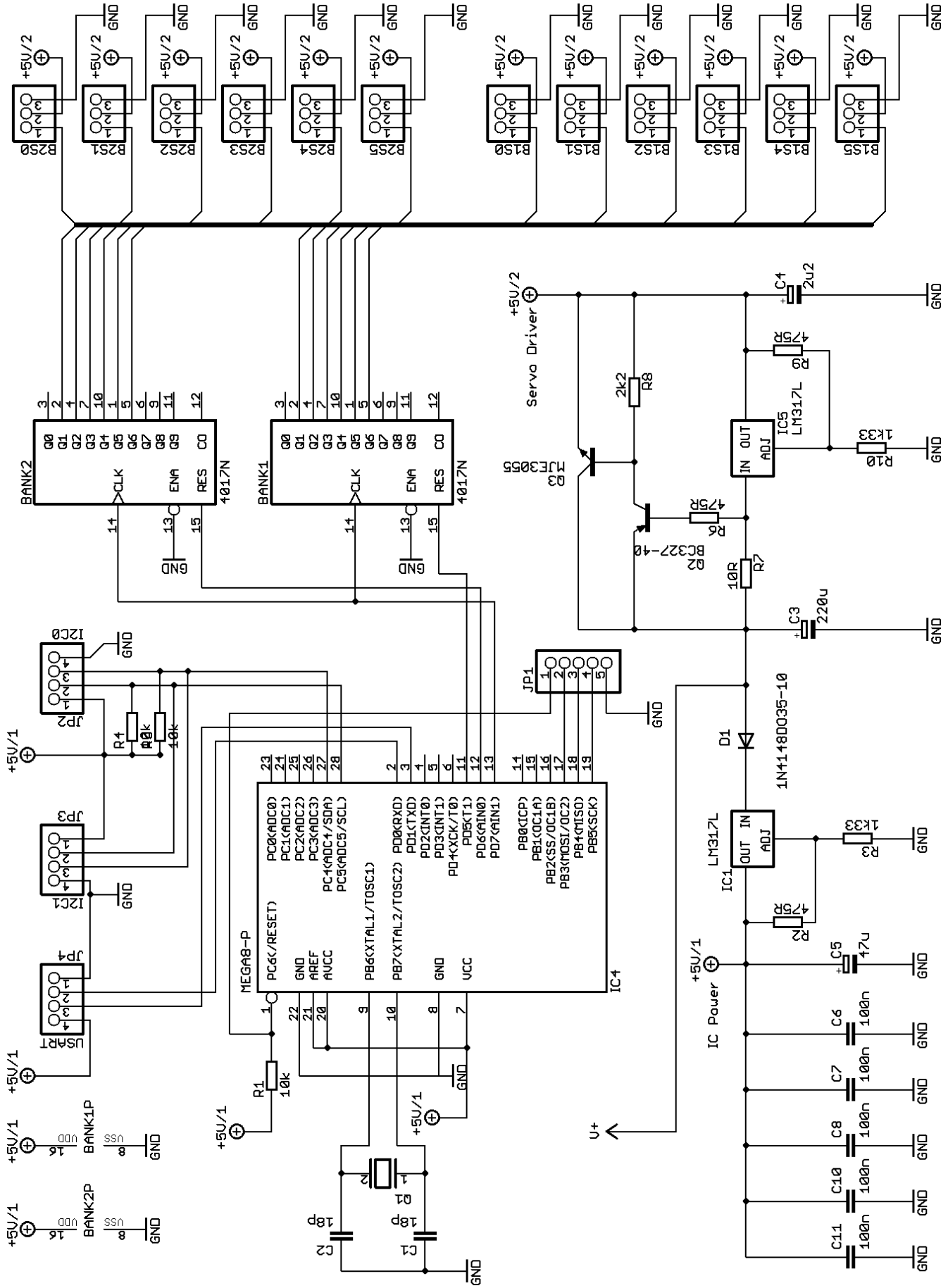


Figure 22: *Main board schematic*

3 Firmware

The firmware is written in C language, compiled with avr-gcc 4.5.3, linked against the AVR Standard C library from the same package. The Standard C library version is 1.8. The Binutils from GNU Toolset (GNU Project) were user to generate the object format flushed into the micro controllers. Tool avrdude was used to transfer the object data into micro controller. The programmer AvrUsb500 from tuxgraphics,org, which is a stk500 V2 compatible programmer.

3.1 Fixed Point Mathematics

The kinematics of the robot uses trigonometric functions and square root. Originally the firmware linked against the mathematical library from the AVR-GCC Toolchain - libm.a. This increased the size of the firmware drastically and the time required to compute the position for all 4 legs took over 100 ms at 14.7MHz. That introduced a problem because the servos require period close to 20 ms. Therefore optimization techniques were tested to resolve this problem.

Linear Regression is used to approximate the trajectory between two points on the trajectory of a step. As described in the chapter 1.2 each step is divided into 9 discreet segments. The kinematics is calculated for those 9 points only (midpoints of segments) and linear regression is further splitting the discreet segments into smaller sub-segments. The number of those sub-segments defines the smoothness of the movement as well as velocity of the movement. The linear regression is great deal less demanding than the kinematics calculation and we still can position the legs while the kinematics calculation for next discreet segment runs. The linear regression typically uses 7 or 9 points and therefore this optimization might be bringing sufficient margin the complete the kinematics for two subsequent segments. Though the processor will be rather busy with kinematics only and there would be no margin left for other operation. Typically debugging via USART port might get expensive while the USART speed is rather low.

Fixed point representation is another tested strategy. The best results were achieved with fixed point representation using 4-bytes-representation, where 2 bytes for integer part and 2 bytes for decimal part. I further found a very compact mathematical library implementing the basic arithmetics for the fixed point numbers plus square root and trigonometric functions. The library is using an interpolation through a cubic function to calculate inverse tangents. The inverse sines and cosines are deduced from inverse tangents.

$$\sin^{-1}(x) = \tan^{-1}\left(\frac{x}{\sqrt{1-x^2}}\right) \tag{25}$$

$$\cos^{-1}(x) = {}^{\pi}/_{2} - sin^{-1}(x) \tag{26}$$

The square root algorithm is quite directly from wikipedia[4]. This method consumes less than 8 ms on the same processor to deliver kinematics coordinates for all 4 legs. The precision of calculated angles α, β, γ is still very good, way better than the precision delivered by the used budget servos TowerPro.

CORDIC method is another technique for calculating the trigonometric functions based on the rotation of a vector. The method is based on the fact that the rotation matrix is built from cos() and sin() trigonometric functions. Rotation of an unit vector then delivers sines and cosines of the angle θ used inside the rotation matrix. Alternatively, trigonometric function $\tan^{-1}$ is obtained by rotating a vector $[1, tan(\theta)]$ until the y coordinate becomes zero. The idea of CORDIC is expressed in equation 27. The smart choice of the base angle α helps speed up final algorithm. E.g. $\tan(\alpha_i) = 2^{-j}$ for integer number j reduces the multiplication to a simple shift operation. The input parameters for the technique were selected based on balance between performance and precision. The delivered data were very good but slightly slower than cubic approximation.

$$\theta = \sum_{i=0}^{n} \alpha_i$$

$$\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} = \prod_{i=0}^{n} \begin{bmatrix} cos(\alpha_i) & -sin(\alpha_i) \\ sin(\alpha_i) & cos(\alpha_i) \end{bmatrix} \tag{27}$$

[4] http://en.wikipedia.org/wiki/Methods_of_computing_square_roots #Binary_numeral_system_.28base_2.29

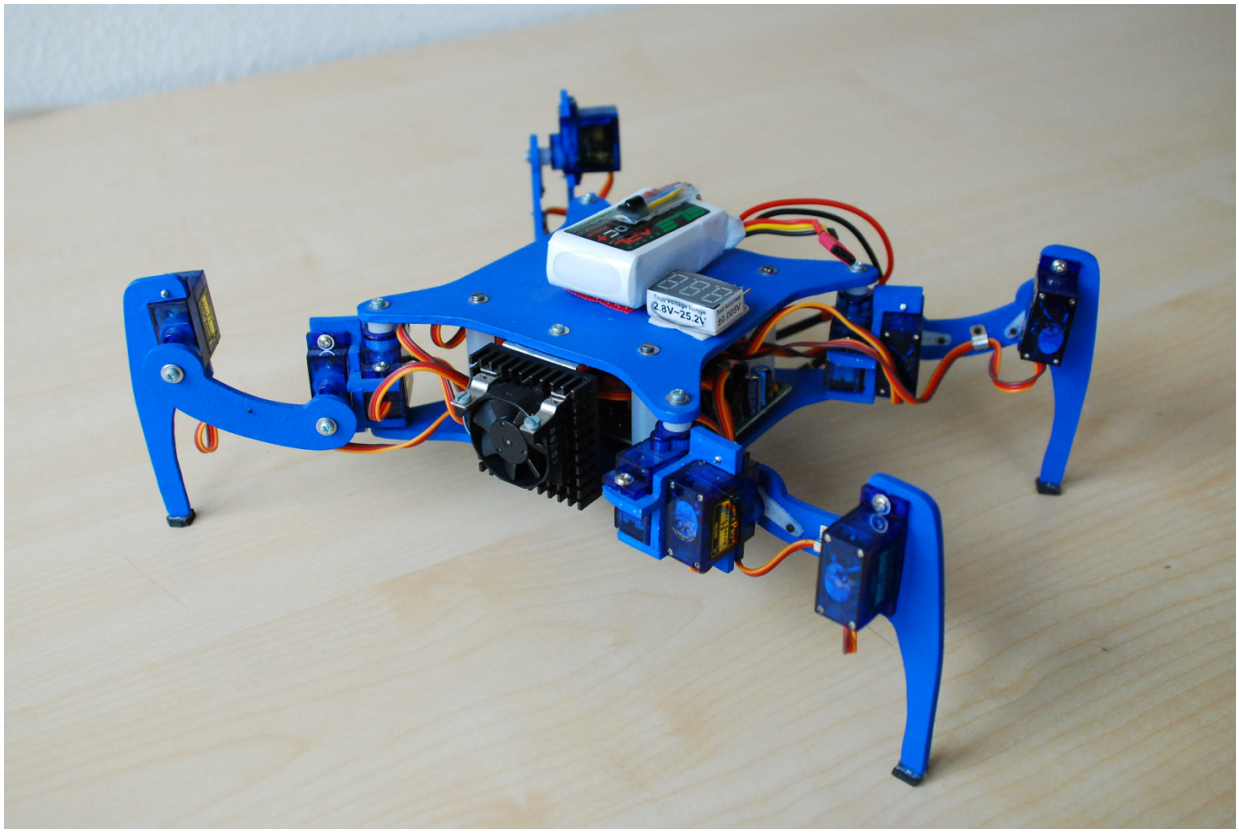


Figure 23: *Front-top picture of the quadrupedal*

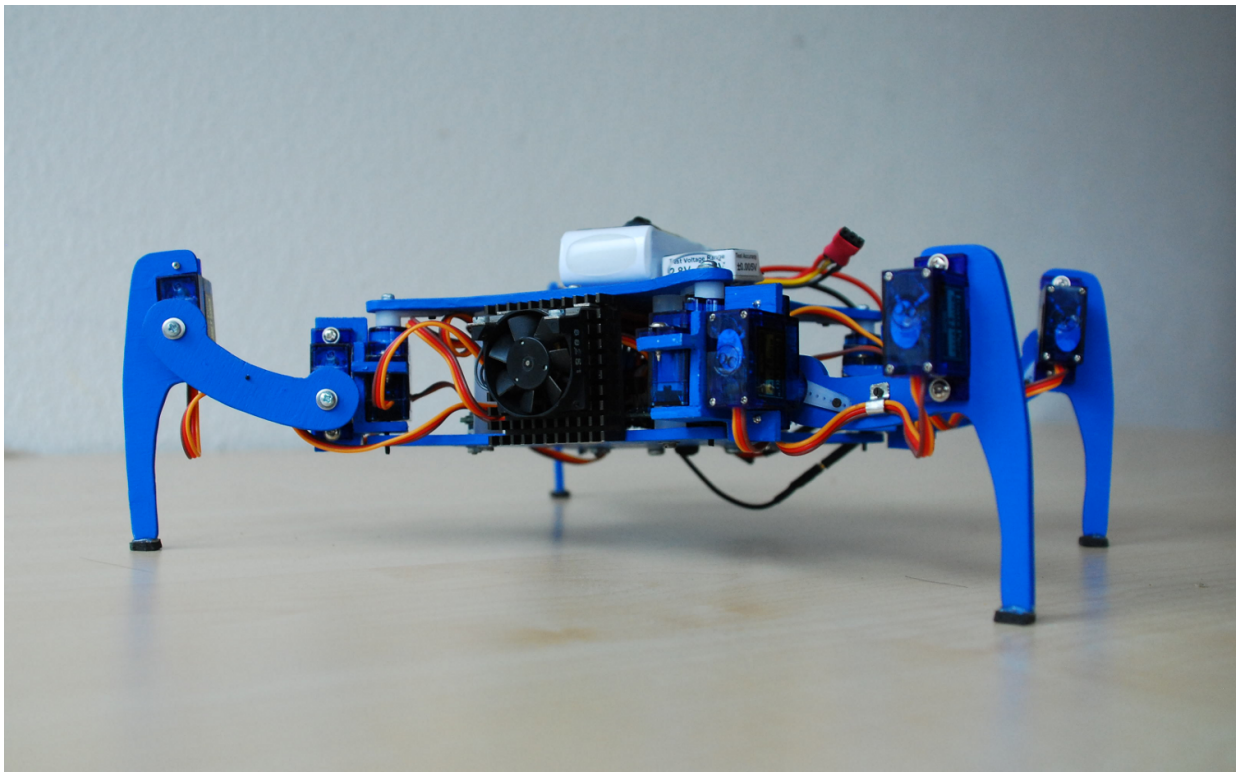


Figure 24: *Front picture of the quadrupedal*

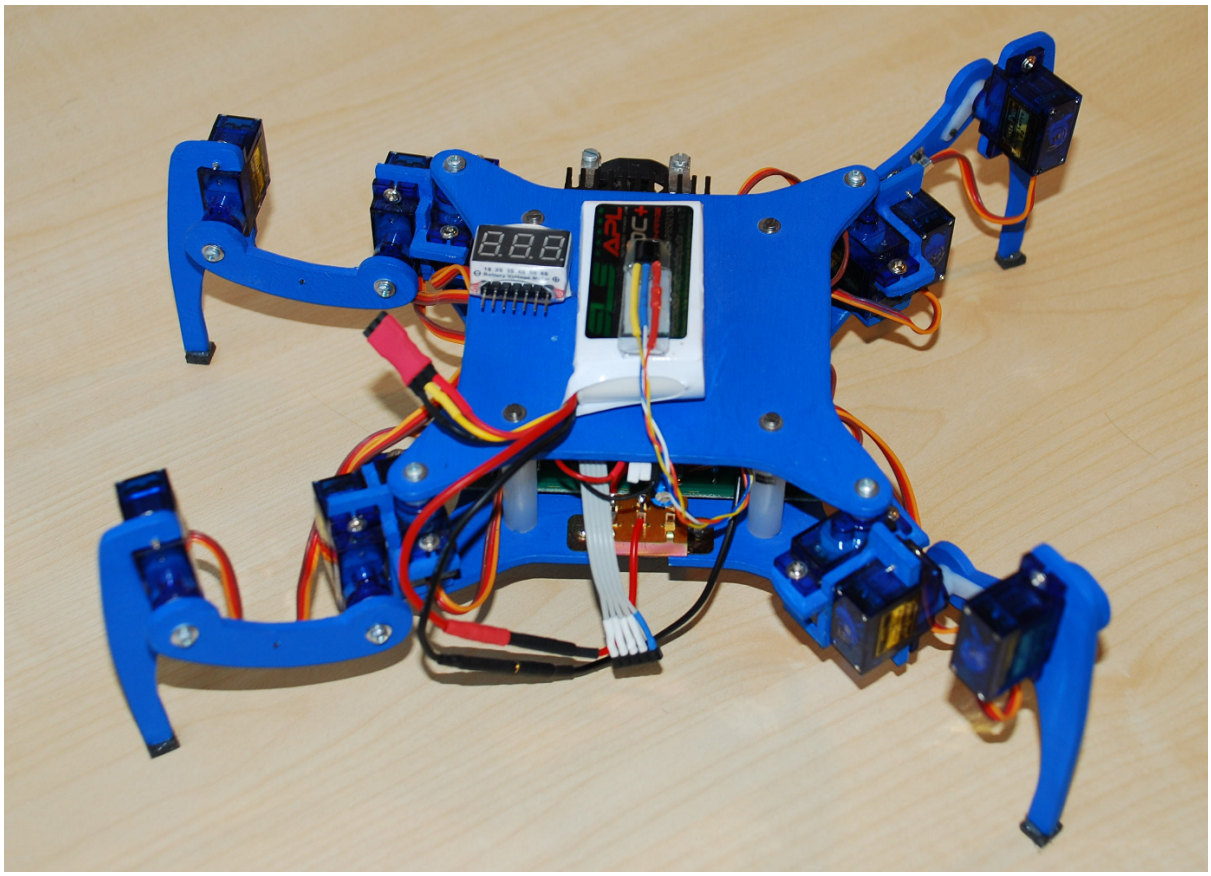


Figure 25: *Rear-top picture of the quadrupedal*

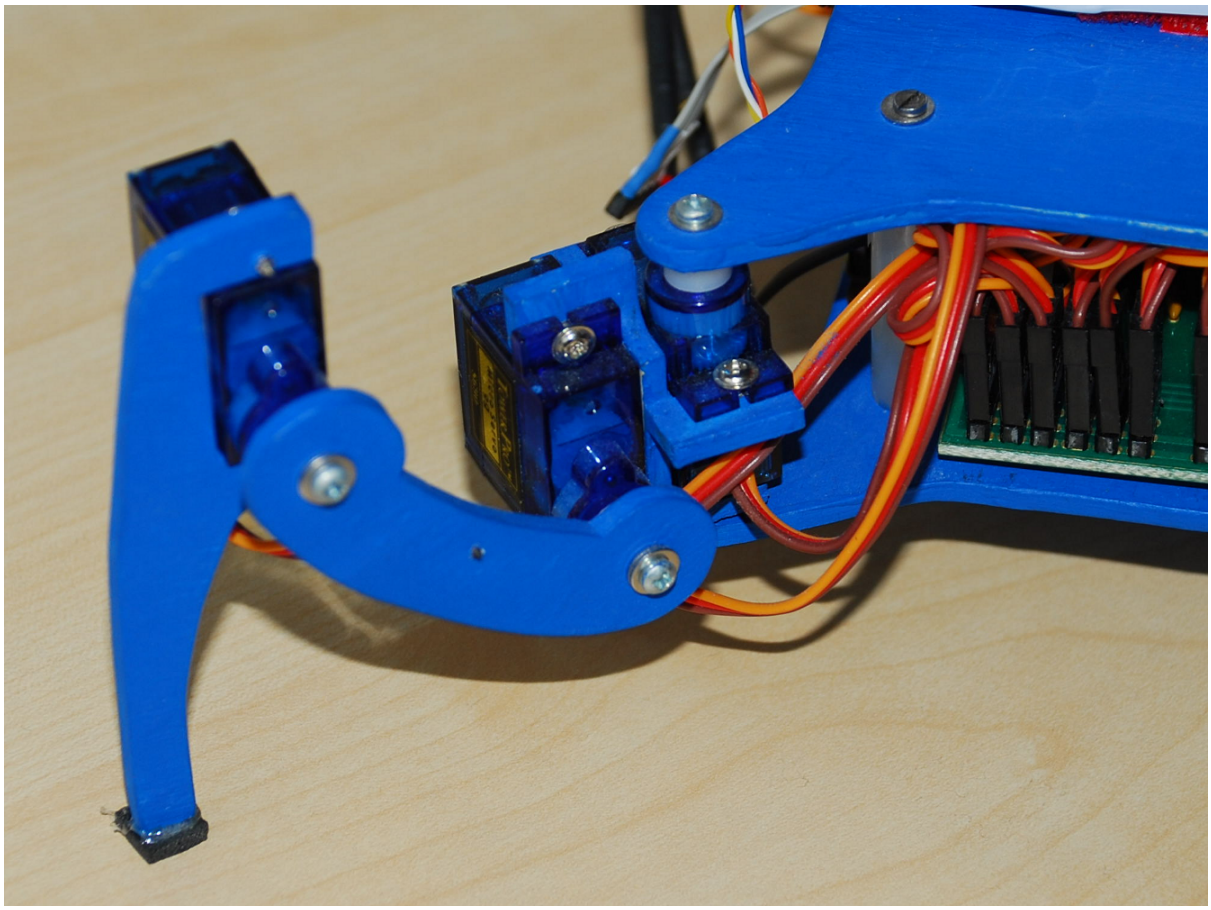


Figure 26: *Leg detail, servo compartment construction.*

References

[1] *Freyr Hardarson*: Stability analysis and synthesis of statically balanced walking for quadruped robots
Royal Institute of Technology, KTH S-100 44 Stockholm, Sweden

[2] *Shigeo Hirose, Kan Yoneda, Riki Furuya, Tatsuo Takagi*: Dynamic and static fusion control of quadruped walking vehicle.
Tokyo Institute of Technology, Department of mechanical Engineering Science, Japan

[3] *S. Parasuraman, F.J. Hang and M.K.A. Ahamed Khan*: Robot Crawler: Statically Balanced Gaits
School of Engineering, Monash University, Malaysia