Table of Contents

[Gait Generator 2](#_Toc4228786)

[Controller Scrip 5](#_Toc4228787)

[Controller Inputs Routine 5](#_Toc4228788)

[Foot Vectors Routine 6](#_Toc4228789)

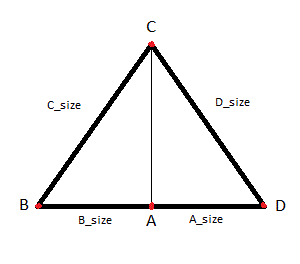
[Gait Script 7](#_Toc4228790)

[Gait Routine 7](#_Toc4228791)

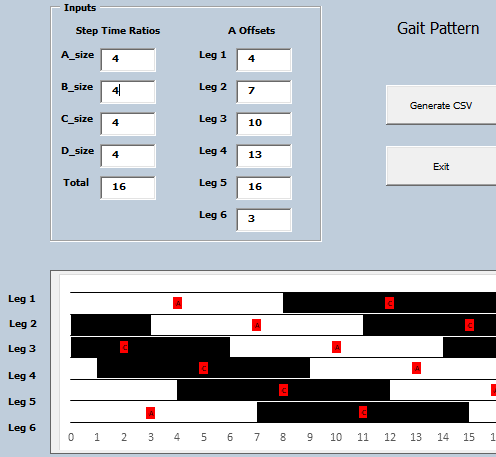
[Output to Motors Routine 8](#_Toc4228792)

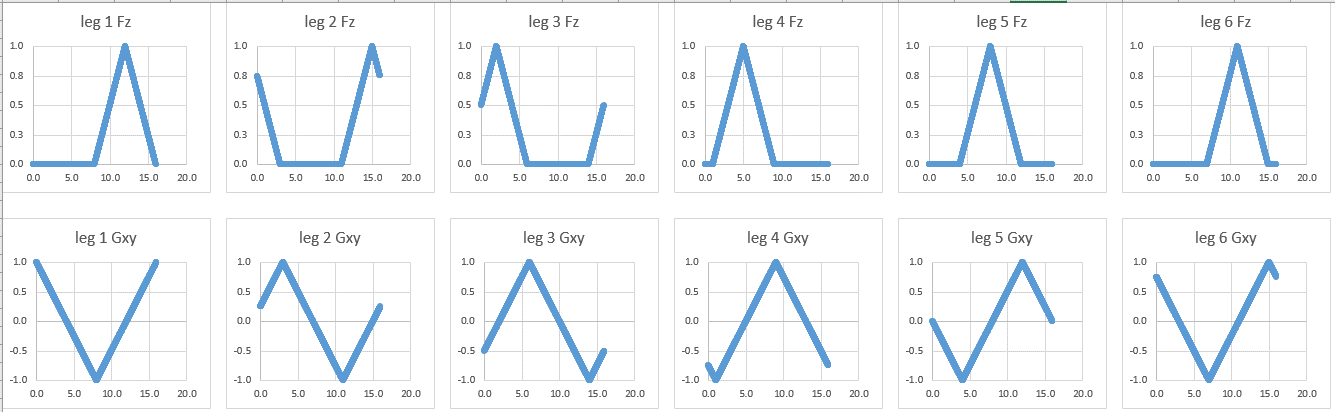
# Gait Generator

The Gait Generator is a VBA program that outputs a .csv file that outlines the general sequence of stepping. Gates are defined by the following step graph:



This is the path a foot will take through one step sequence. The step sequence is the process of moving around the step graph one complete rotation (A→B→C→D→A). The step sequence takes the same amount of time for each leg. The time can be distributed between the sides of the graph whole values a represented by “A/B/C/D\_size” on the step graph. The point on the step graph on which a leg starts at can vary. This make it so that legs are rising off the ground at different times in the sequence. The Gait Generator outputs and graphs for each leg. is a value from 0 to 1 that describes how the leg moves through vertical space and is plotted against time. is a value between -1 and 1 that describes how the leg moves through the lateral space and is plotted against time. The following is an example of the gait input parameters and the associated outputs:





Create two simultaneously running python scripts on the Raspberry Pi.

The **Controller Script** consisting of routines 2 and 3.

The **Gait Script** consisting of routines 4 and 5.

1. Initialize Routine

This will run automatically upon powering up the Raspberry Pi. Calls the **Controller Script** and **Gait Script** simultaneously

# Controller Scrip

## Controller Inputs Routine

Run an Initialize Subroutine on first call to set up the connection to the PS3 controller.

Continuously run an Input Checker Subroutine that monitors the controller buttons and joysticks values every few milliseconds. Send the analog values through a filter that rounds them to some sensitivity threshold. Compare monitored values against previously logged values.

If the values have changed then push through a Dampener Subroutine that smooths out changes.

Push new the inputs to the Foot Vector Routine.

## Foot Vectors Routine

Take new joystick inputs calculate all six foot-vectors.

Global Foot Datum Vector

Global Foot Datum Radius

Global Final Foot Vector

Joystick 1 x, y input

Joystick 2 rotation input

Body Rotation angle

Log calculated vectors to some where the Gate Scriptcan access it.

# Gait Script

## Gait Routine

Run an Initialize Subroutine on first call to set to resting stance and move motors.

The resting stance is a preset that places the legs close to the HexaPi body but ready to stand up.

Continuously check the foot vectors for value changes. If foot vectors are zeroed and HexaPi is not already in resting stance, then start an Idle Timer.

\*On first foot vector change put HexaPi into datum stance.

Datum stance is the Idle standing position. The feet will all be on their A points.

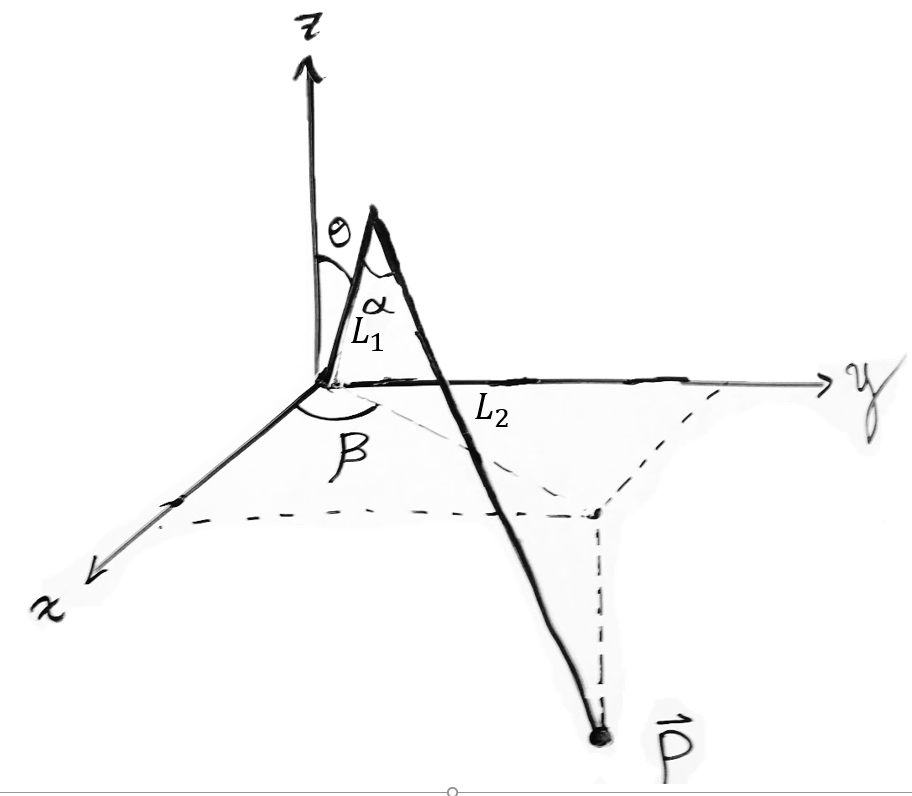
If Idle Timer has lapsed, then run Idle Reset Subroutine. This will bring legs back to point A. Cycle through legs in a tripod gait sequence. If the leg is currently in between (B, C, D) a.k.a. in the air, then take a direct path to A. Add math to calculate this direct path’s . If the leg is currently in between (D, A, B) – a.k.a. on the ground – use a path that lifts off the ground to return to A. Add math to calculate this triangular path’s .

Start the Walk Subroutine which Iterates through the and from the generated \*.csv file. Each iteration will output .

Output to Motors Routine (Make this into a lookup table? Depends which is faster.)

Take the for each leg and calculate the angle required for all 3 servos.

In the graph below: is defined as 0 at the y+ axis. Clockwise is positively increasing.



Take the calculated angles in a PWM Subroutine convert these into PWM signals for each motor.

and are the PWM values when the motor is fully extended in either direction.

is the requested angle for the motor in degrees.

is the total working range of the servo in degrees. Will be set as 180.