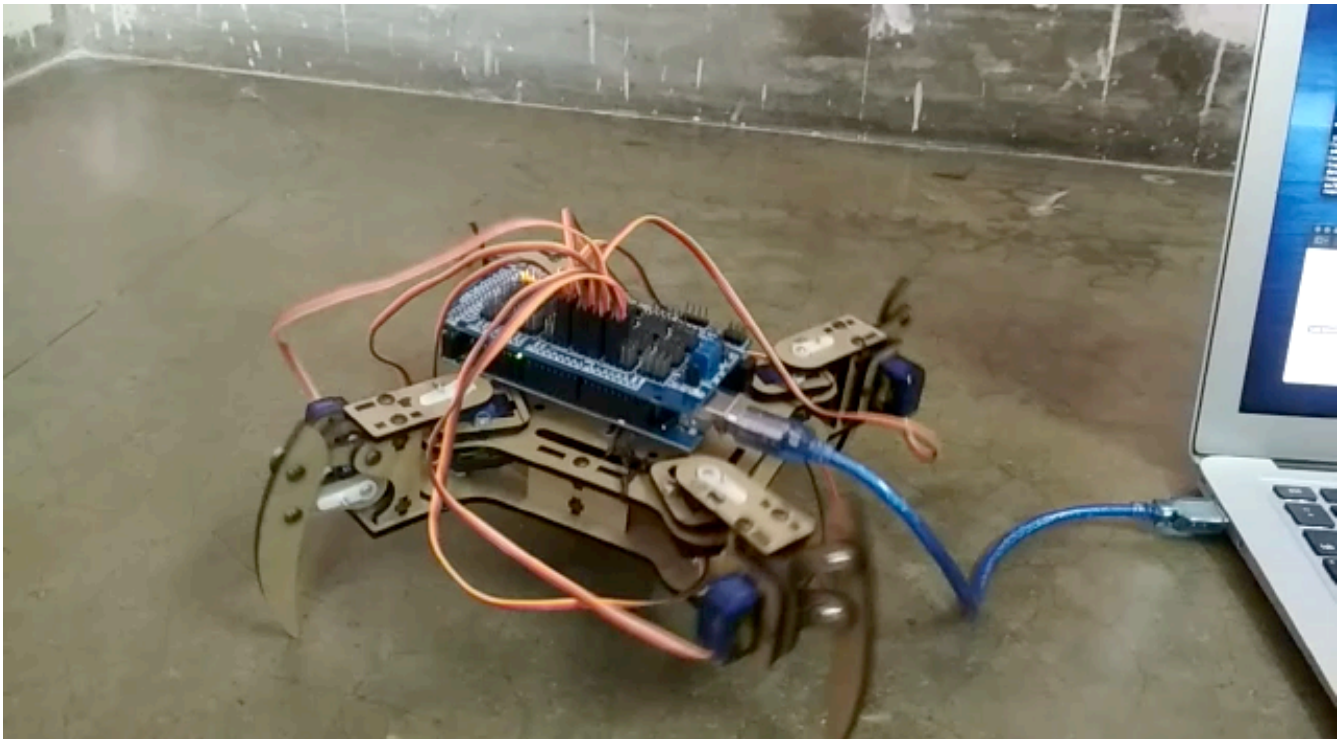
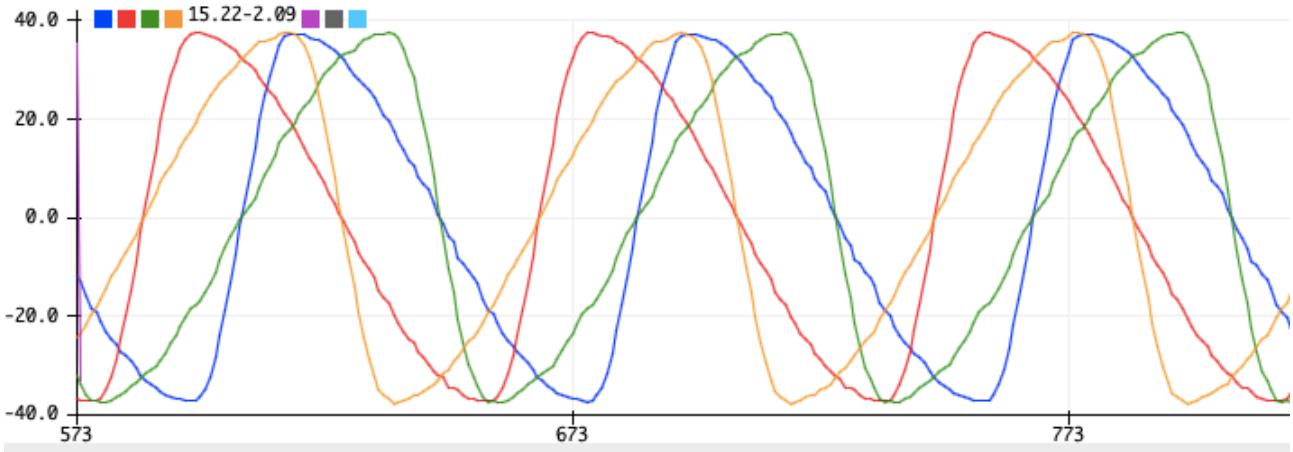
Gait Generation in Quadruped

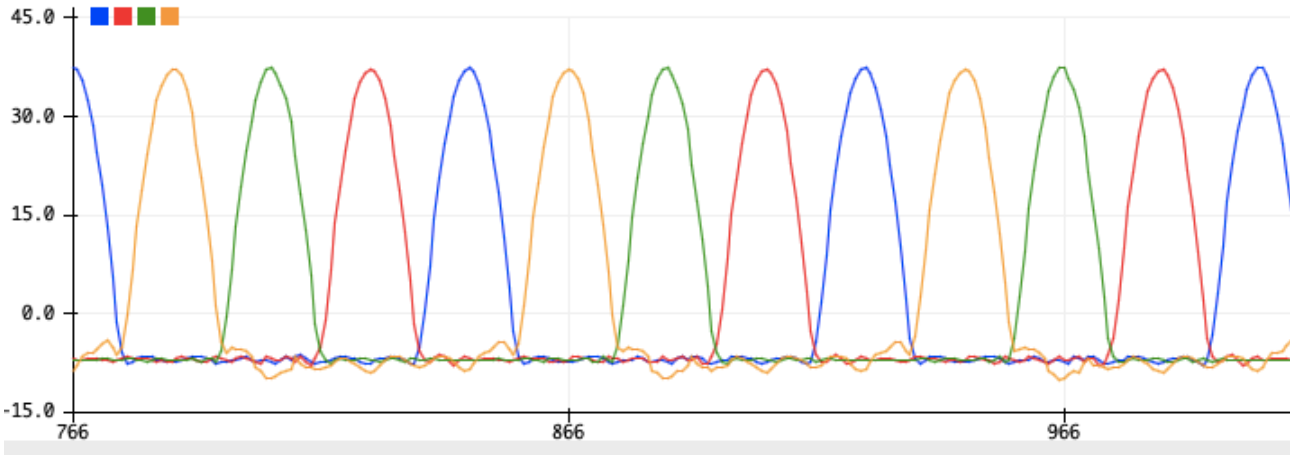


The above is a video of the robot walking uses the following parameters:

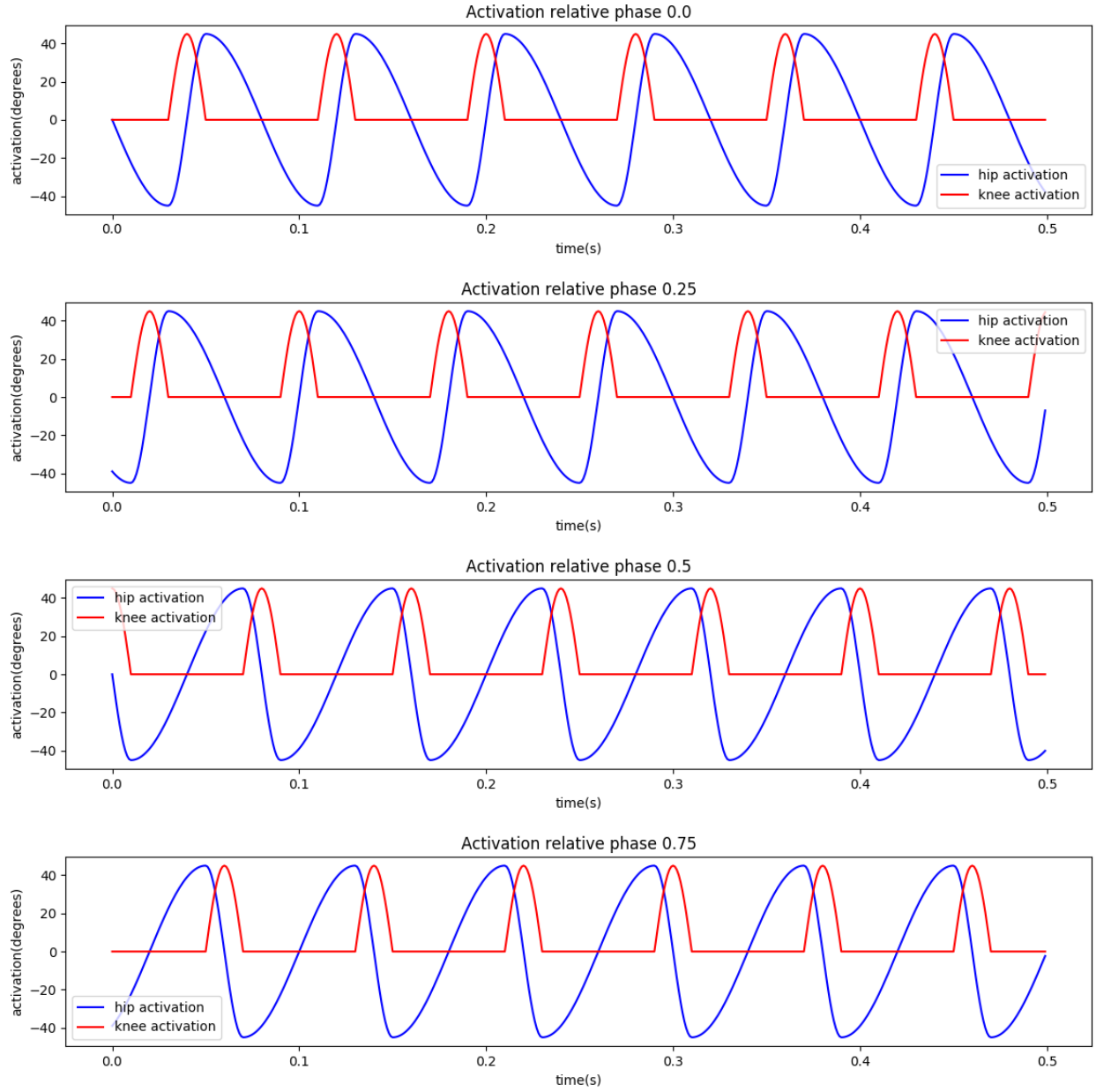
- $T_{sw} = 20 \times 10^{-3} s$
- $T_{st} = 60 \times 10^{-3} s$
- $dt = 10^{-3} s$
- $\theta = 45$

The following output signals were obtained for the hip and the knee joints respectively-





The above diagrams are representative of the signals obtained as output from the CPG model deployed on the Arduino. Refer to the following diagram for more details about the driving signal used to train the CPG.



The above training signal is derived from the following formulae:

Hip Activations:

$$\theta_h(t) = \begin{cases} \theta_h sin(\frac{(t-\frac{iT}{4})\pi}{\beta T} + \pi), & \text{if } 0 \leq t \leq \frac{\beta T}{2} \\ \theta_h sin(\frac{(t-\frac{iT}{4})\pi}{(1-\beta)T} + \frac{(3-4\beta)\pi}{2(1-\beta)}), & \text{if } \frac{\beta T}{2} \leq t \leq \frac{T(2-\beta)}{2} \\ \theta_h sin(\frac{(t-\frac{iT}{4})\pi}{\beta T} + \frac{(\beta-1)\pi}{\beta}), & \text{if } \frac{T(2-\beta)}{2} \leq t \leq T \end{cases}$$

For $i \in \{0,1\}$

And

$$\theta_h(t) = \begin{cases} -\theta_h sin(\frac{(t-\frac{iT}{4})\pi}{\beta T} + \pi), & \text{if } 0 \leq t \leq \frac{\beta T}{2} \\ -\theta_h sin(\frac{(t-\frac{iT}{4})\pi}{(1-\beta)T} + \frac{(3-4\beta)\pi}{2(1-\beta)}), & \text{if } \frac{\beta T}{2} \leq t \leq \frac{T(2-\beta)}{2} \\ -\theta_h sin(\frac{(t-\frac{iT}{4})\pi}{\beta T} + \frac{(\beta-1)\pi}{\beta}), & \text{if } \frac{T(2-\beta)}{2} \leq t \leq T \end{cases}$$

For $i \in \{2,3\}$

Knee Activations:

$$\theta_k(t) = \begin{cases} \theta_k sin(\frac{t\pi}{T(1-\beta)} - \frac{\beta\pi}{2(1-\beta)}), & \text{if } \dot{\theta}_h(t) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

For $i \in \{0,1\}$

And

$$\theta_k(t) = \begin{cases} \theta_k sin(\frac{t\pi}{T(1-\beta)} - \frac{\beta\pi}{2(1-\beta)}), & \text{if } \dot{\theta}_h(t) \leq 0 \\ 0, & \text{otherwise} \end{cases}$$

For $i \in \{2,3\}$

The above formulation is different from the following previously proposed:

Hip Activation

$$\theta_h(t) = \begin{cases} \theta_h sin(\frac{(t-\frac{iT}{4})\pi}{\beta T} + \pi), & \text{if } 0 \leq t \leq \frac{\beta T}{2} \\ \theta_h sin(\frac{(t-\frac{iT}{4})\pi}{(1-\beta)T} + \frac{(3-4\beta)\pi}{2(1-\beta)}), & \text{if } \frac{\beta T}{2} \leq t \leq \frac{T(2-\beta)}{2} \\ \theta_h sin(\frac{(t-\frac{iT}{4})\pi}{\beta T} + \frac{(\beta-1)\pi}{\beta}), & \text{if } \frac{T(2-\beta)}{2} \leq t \leq T \end{cases}$$

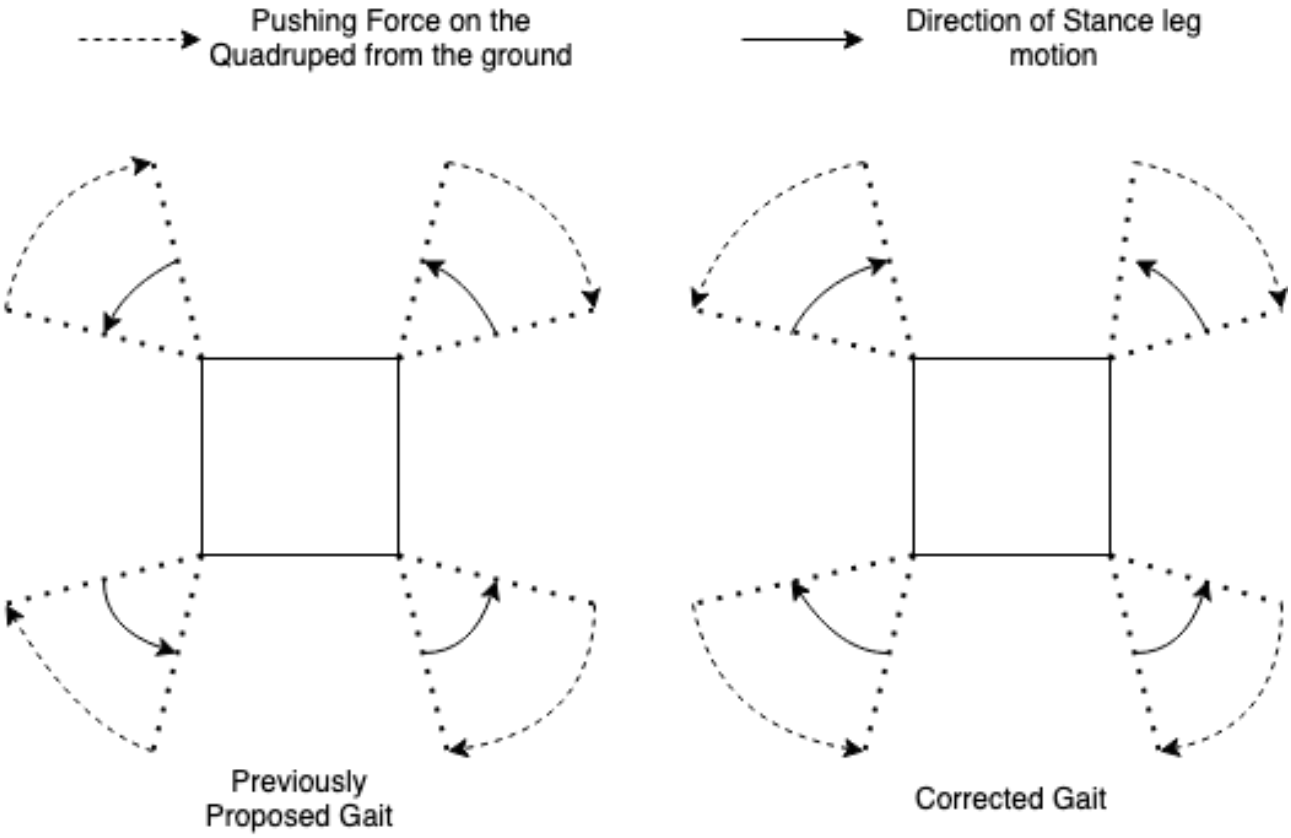
For $i \in \{0,1,2,3\}$

Knee Activation

$$\theta_k(t) = \begin{cases} \theta_k sin(\frac{t\pi}{T(1-\beta)} - \frac{\beta\pi}{2(1-\beta)}), & \text{if } \dot{\theta}_h(t) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

For $i \in \{0,1,2,3\}$

The difference can be understood from the following diagram of direction in which the leg rotates in the stance phase and the resultant force on the quadruped from the ground:



In the case of the previously proposed formulae, the resultant force on the quadruped would be zero as the forward forces generated by the quadruped on side would be cancelled out by the backward forces on the other side. The new formulation fixes this issue and the quadruped walks.

Challenges

An analysis of the walk and the implementation process, reveals the following challenges-

- The COM of the quadruped does not remain within the support polygons on the the remaining three legs when one of the legs is in the swing phase
    - The result of the COM not remaining in the support polygon is that the legs are not in constant contact with the ground during the stance phase and are in contact for sometime during the swing phase. This behaviour destabilises the walk.
    - The above observation was hypothesised to be a result of the asymmetry introduced by the Arduino Mega 2560 controlling the quadruped. An experiment without the Arduino mounted on the quadruped revealed that it was

not the case. A similar experiment was performed with the battery pack as well. Results were still the same
    - This can be solved by modifying the gait generation formulae for training to ensure that the COM does not go out of the support polygon. The only problem with this approach is that the exact COM of different parts of the quadruped is not known. Also the change in the position of the COM of the legs with the movement of the knee joints is also known.
    - Another possible way option would be introduction of visual feedback to stabilise the gait. This needs further investigation to understand how this can be achieved.
- Due to training with python using 32 bit precision and inference on Arduino, that supports a maximum precision of 16 bit, there is a loss of information that can be observed when the generated signal from python and on the Arduino are compared.
    - This can be solved by training directly using 16 bit precision. But the implementation of 16 bit training in python is not possible as complex numbers with 16 bit precisions are not available. Although an attempt was made to write the training code in cpp, which does support 16 bit precision complex numbers, the need for implementation of memory management and other factors make a successful implementation much more difficult and every attempt at such an implementation has failed so far.
    - Another possibility is using a Raspberry pi to control the quadruped. A Raspberry pie has RAM and storage memory in the order of GBs where as an Arduino has RAM and storage memory in the order of KBs. Also Raspberry pi runs a truncated version on Debian, thus is capable of running any python program, unlike the Arduino. The only downside of the Raspberry pi is that it is much more expensive than the Arduino.