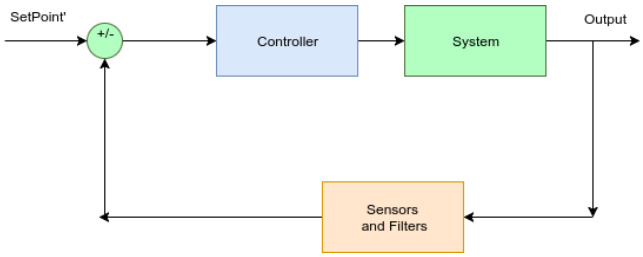

Fig. 3: Controller Block Diagram

IV. Reinforcement Learning Methods as Controllers

In [1], we worked on traditional Controllers like PID, Fuzzy PD,PD+I & LQR . The biggest problem with those methods is that , they need to be tuned manually. So, reaching optimal values of Controllers depends on many trials and errors. Many a times optimum values aren't reached at all. The biggest benefit of Reinforcement learning algorithms as Controllers is that , the model tunes itself to reach the Optimum values. The following two sections discuss Q Learning and Deep Q Network.

A. Q Learning

Q- learning was developed by Christopher John Cornish Hellaby Watkins [7]. According to Watkins, "it provides agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions, without requiring them to build maps of the domains." [8]. In a Markovian domain, Q function- the model to be generated using the algorithm- calculates the expected utility for a given finite state s and every possible finite action a. The agent - which is the robot in this case- selects the optimum action a having the highest value of $Q(s, a)$, this action choosing rule is also called Policy. [8] . Initially, the $Q(s, a)$ function values are assumed to be zero. After every training step , the values are updated according to the following equation

$$Q(s, a_t) \leftarrow Q(s, a_t) + \alpha(r + \gamma maxQ(s_{t+1}, a)) \quad (1)$$

The objective for the Model in our project is to keep it within limits i.e. $\pm 5^0$. At first , the robot model, Q matrix, Policy π are initialized . There are some interesting points to make. The states are not finite. Within limit range , hundreds and thousands of pitch angles are possible. Having thousands of columns is not possible. So, the state values were discretized. We discretized the values to 20 discrete state angles from -10^0 to 10^0. For action value, we chose 10 different velocities. They are $[-200, -100, -50, -25, -10, 10, 25, 50, 100, 200] ms^{-1}$. The Q matrix had 20 columns , each column representing a state and 10 rows each representing every action. Initially ,the Q -values were assumed to be 0 and random actions were specified for every state in the policy π . The training was done for 1500 episodes and in each episode, the training was iterated 2000 times . At the beginning of each episode, the simulation was refreshed. Whenever the robot's state exceeded the limit it was penalized by assigning reward to -100 . The Q Table is updated at each step according to equation 1. The Algorithm 1 shows the full algorithm.

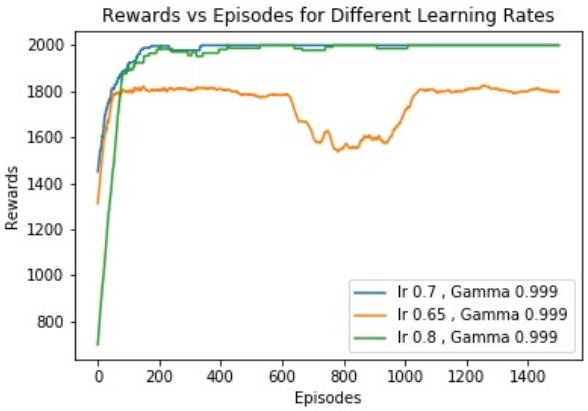

Fig. 4: Rewards for different α

The simulation was run for three different α values $(0.7, 0.65, 0.8)$, with γ value of (0.999). The Fig. 4 shows the Rewards vs Episodes for those αs. It is evident that, the robot couldn't reach the target rewards within the training period for those learning rates. We see that, for the α values 0.7 and 0.8, the robot reaches maximum possible accumulated rewards, 200, within 400 episodes. The curve with α value 0.7 is less stable compared to that of 0.8. But The curve with α value 0.65 never reaches the maximum accumulated reward.

B. Deep Q Network (DQN)

V Mnih et al [9] first used Deep Learning as a variant of Q Learning algorithm to play six games of Atari 2600 , which outperformed all other previous algorithms. . In their paper, two unique approaches were used.

- Experience Replay
- Derivation of Q Values in one forward pass

The technique of Experience Replay, experiences of an agent , i.e. $(state, reward, action, state_{new})$ are stored over many episodes . In the learning period, after each episode random batches of data from experience are used to update the model. [9]. There are several benefits of such approach. According to the paper,

- It allows greater data efficiency as each step of experience can be used in many weight updates
- Randomizing batches breaks correlations between samples
- Behaviour distribution is averaged over many of its previous states

In the classical Q learning approach, one has to give state and action as an input resulting in Q value for that state and action. Replicating this approach in Neural Network is problematic. Because in that case one has to give state and