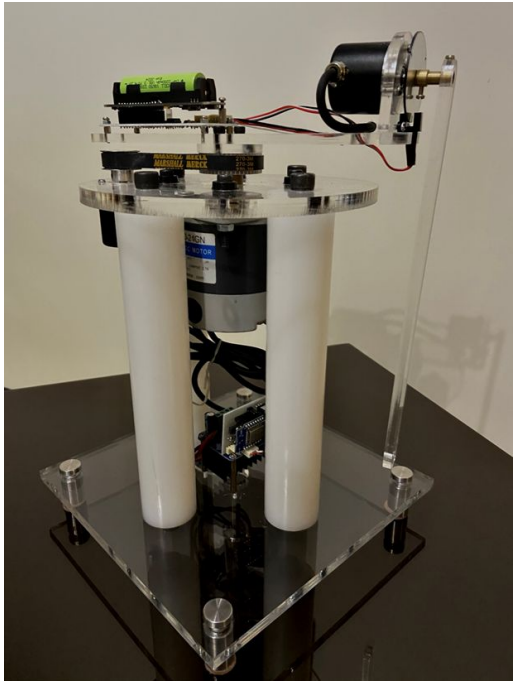
Controlling the Rotary Inverted Pendulum: A Reinforcement Learning Approach

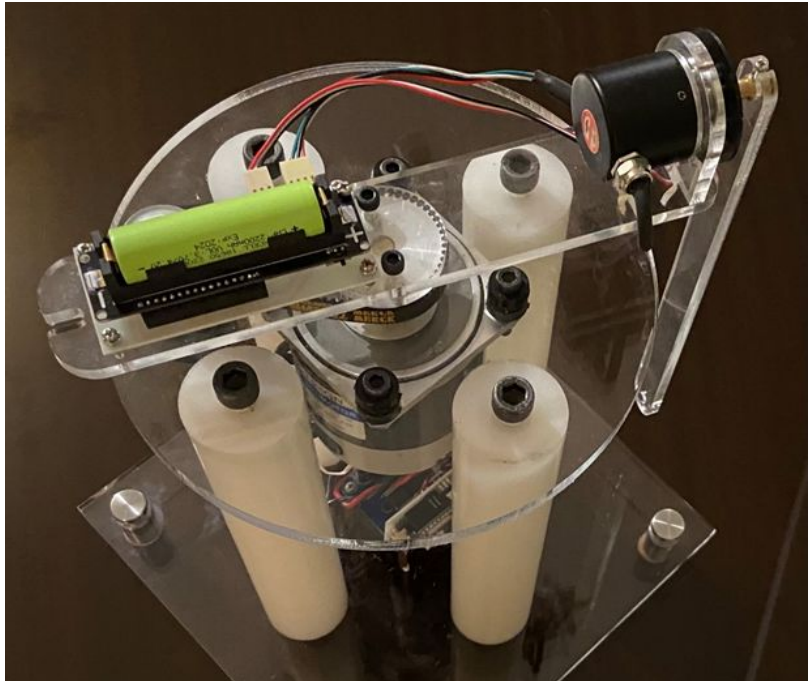
---

Presented By: Arash Hatefi

Supervisor: Dr. Masoud Shariatpanahi

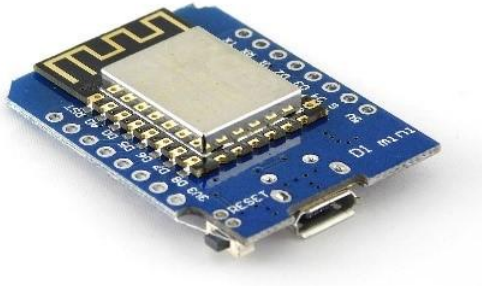
Rotary Inverted Pendulum: The Complete Structure

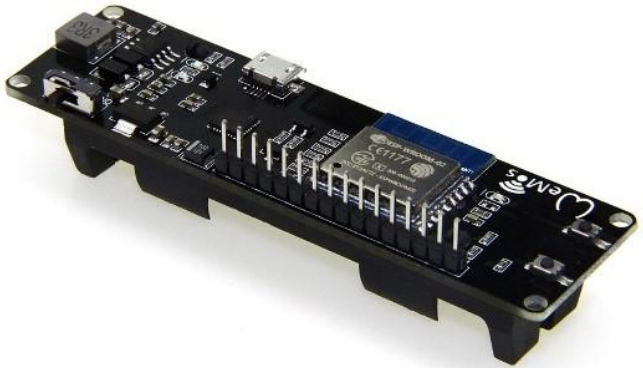




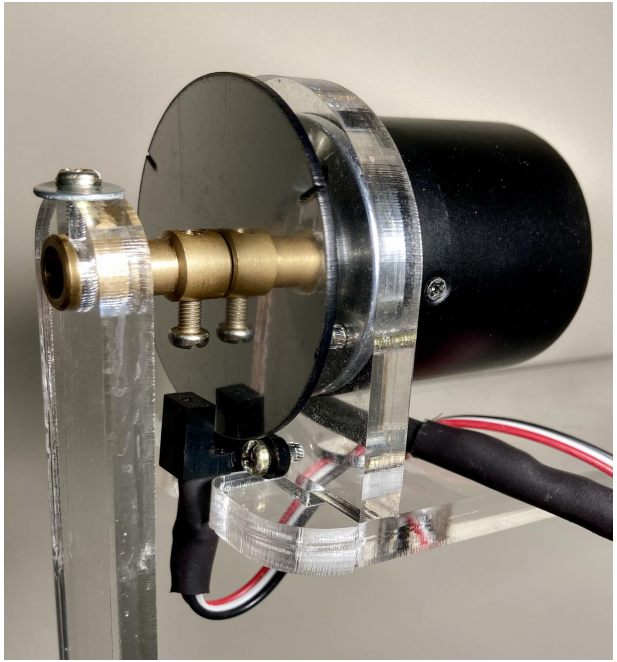
Rotary Inverted Pendulum: Microcontrollers

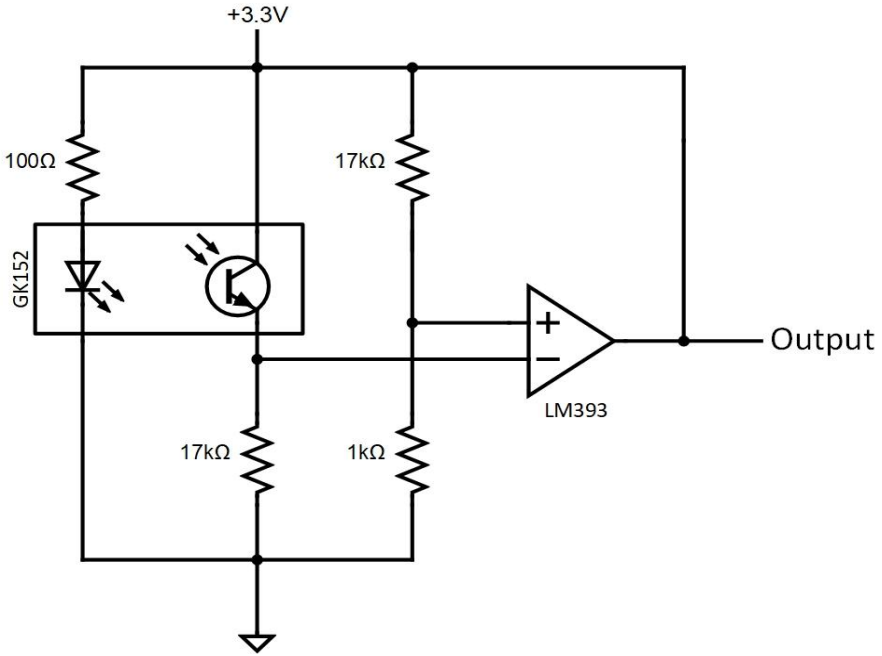
Two Womos-D1 WiFi boards based on ESP-8266 are used to send and receive data to and from the computation server via UDP.



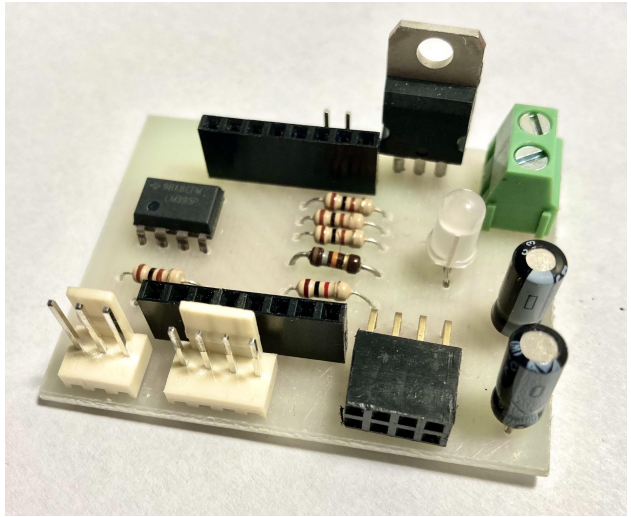


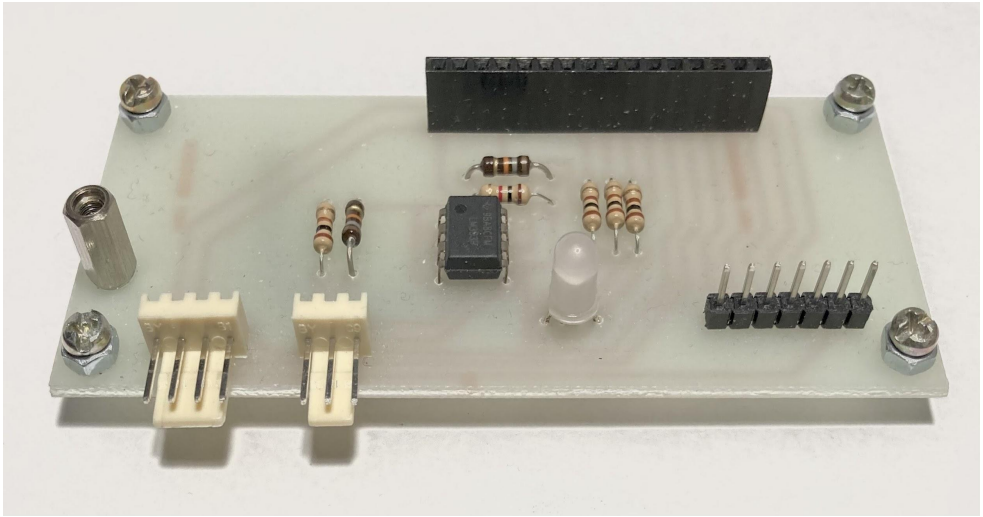
Rotary Inverted Pendulum: Encoder Calibration



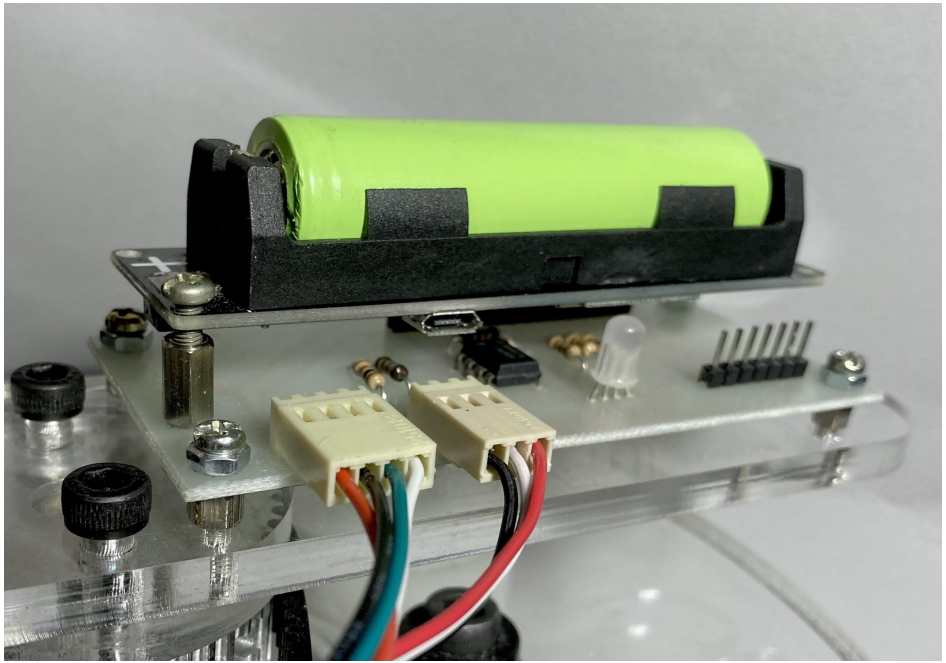


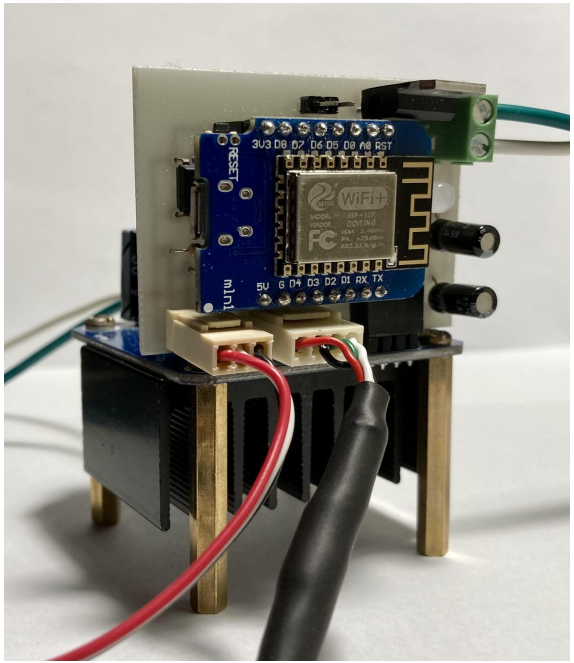
Rotary Inverted Pendulum: The Circuits



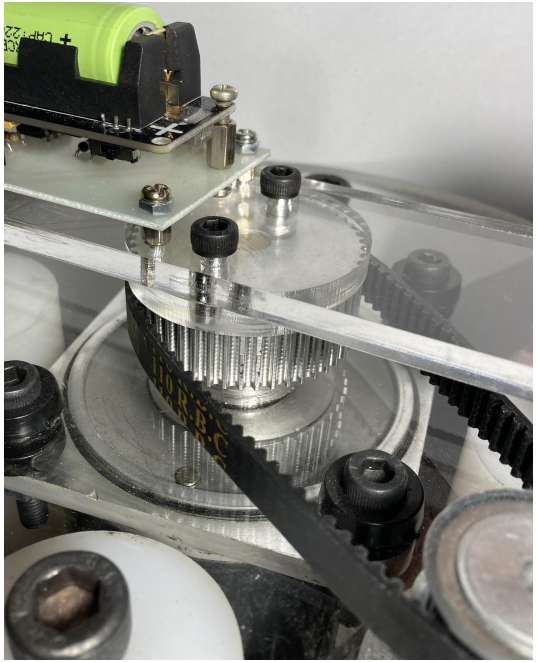


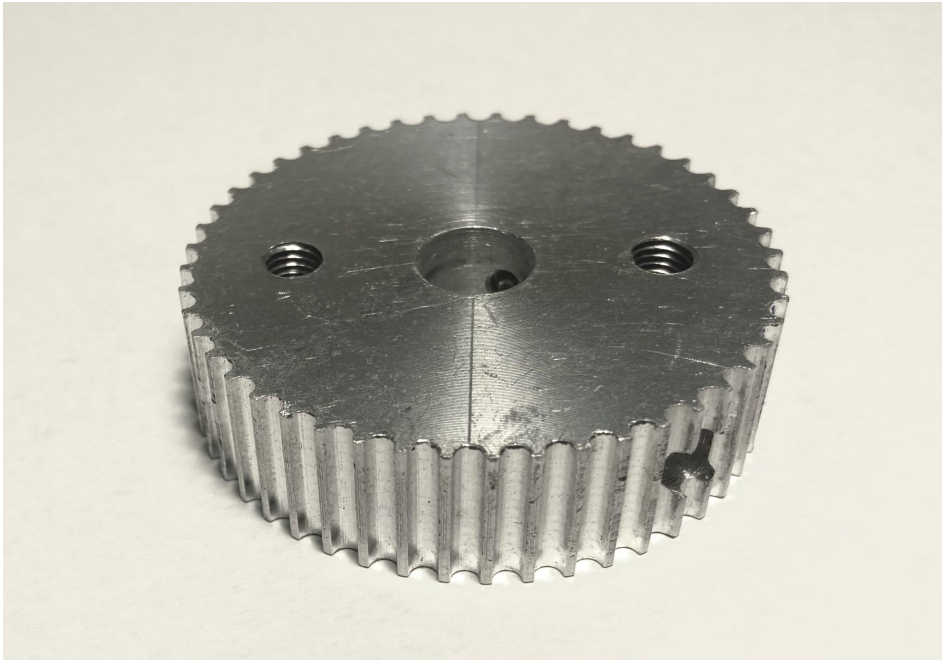
Rotary Inverted Pendulum: The Circuits



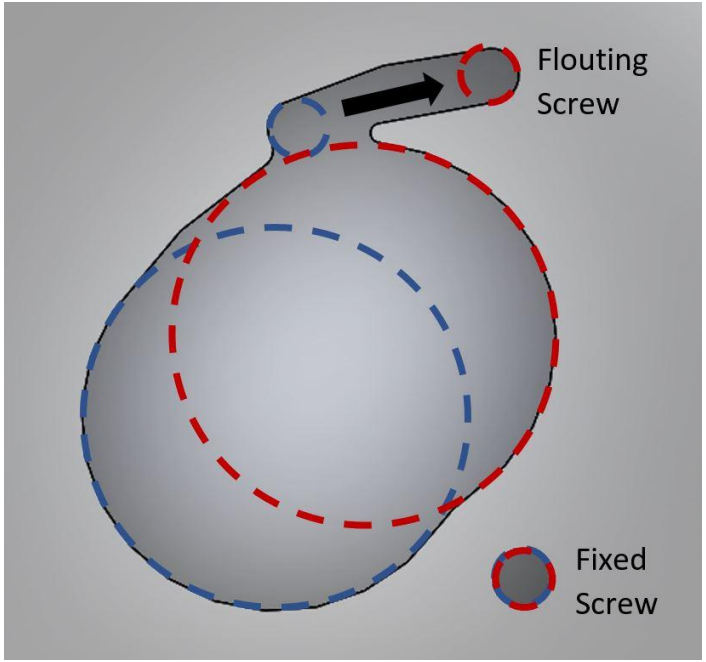


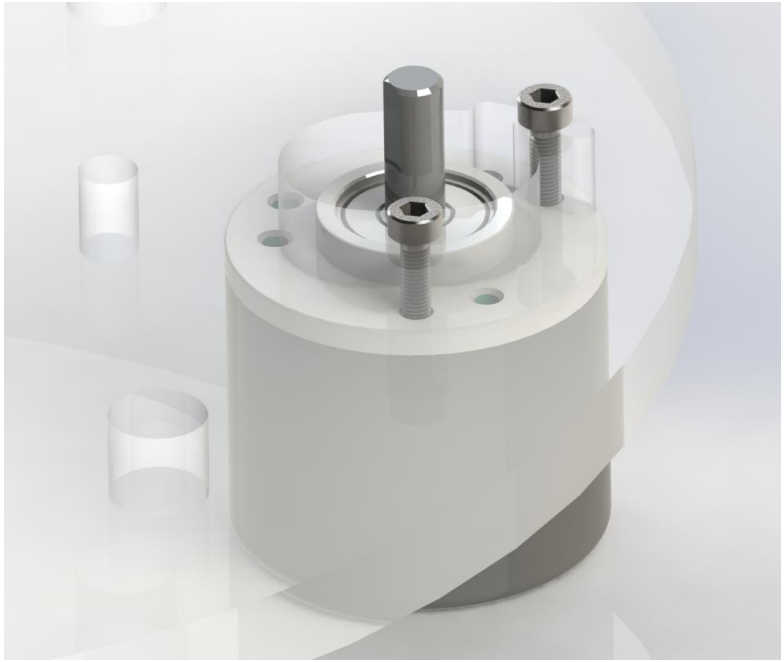
Rotary Inverted Pendulum: Motor to Encoder Connection



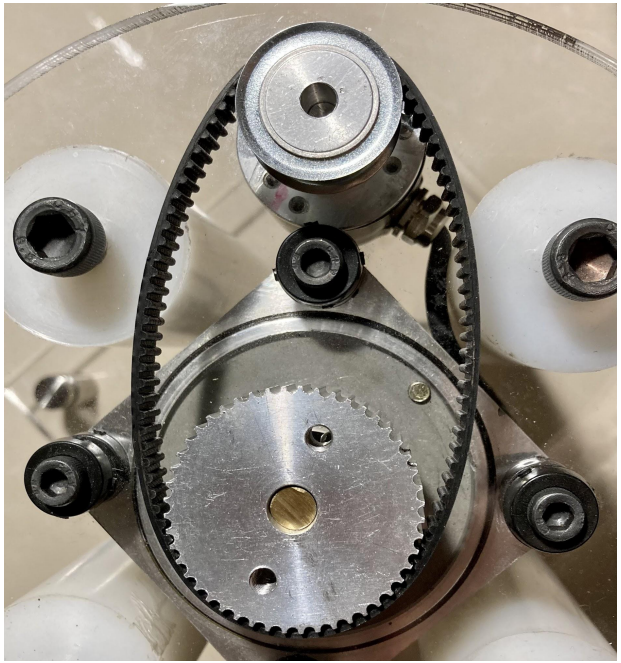


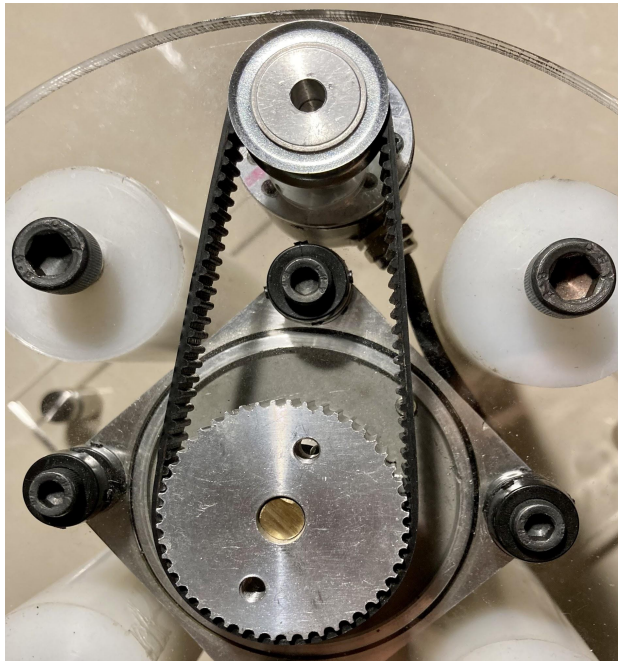
Rotary Inverted Pendulum: Belt Tension Adjustment



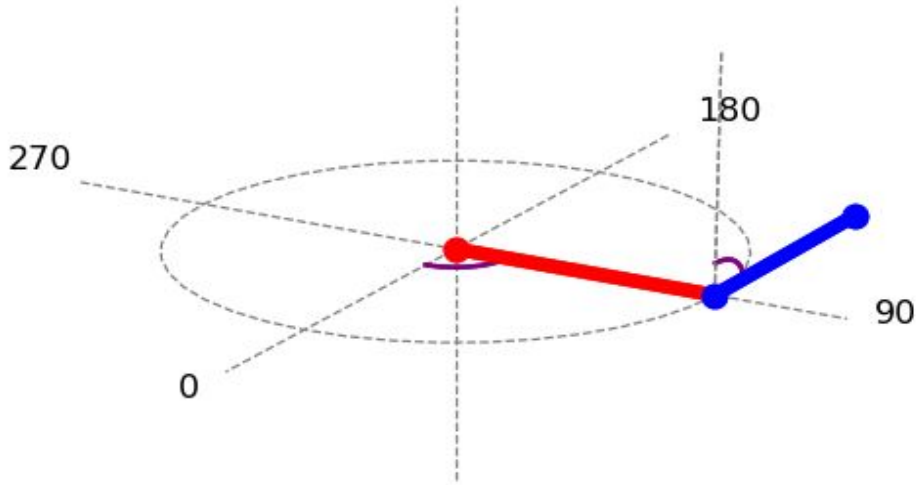


Rotary Inverted Pendulum: Belt Tension Adjustment

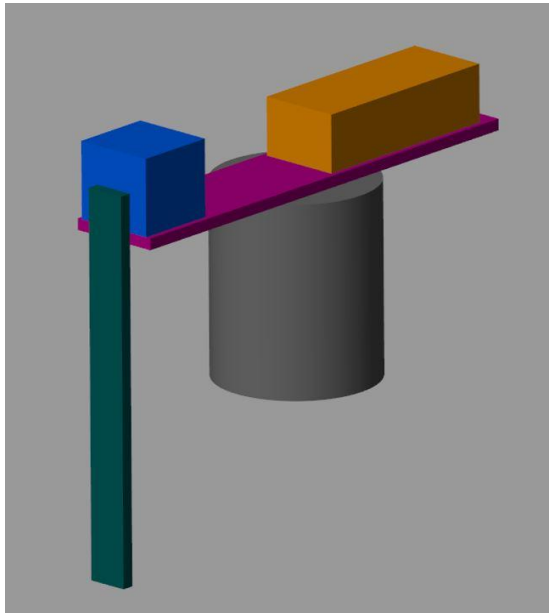




Simulation



Basic simulation from scratch in Python



Simulation in Simulink

Simulation: Simulink Model Assumptions

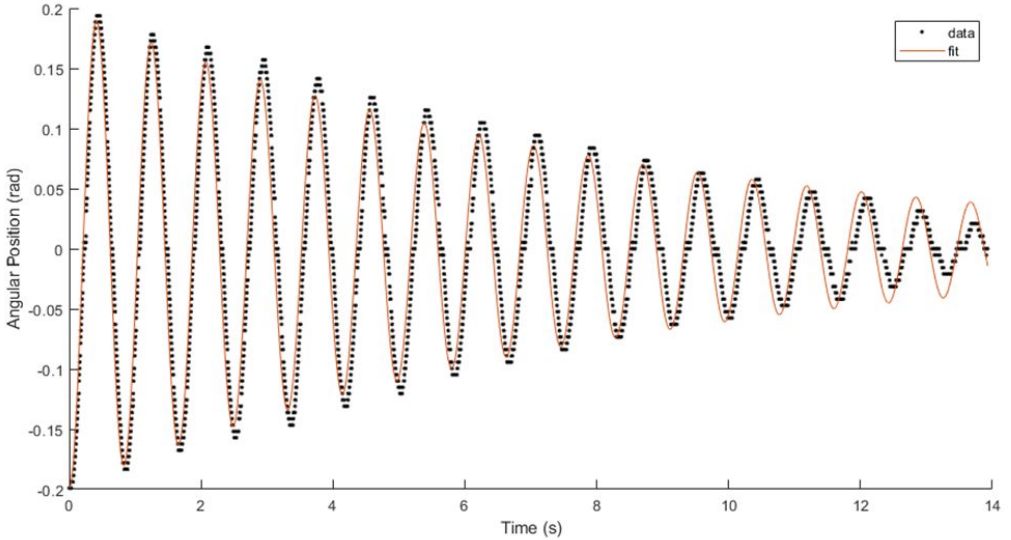
1. The encoder connected to the pendulum and the arm PCB are considered as centered mass.
2. The inertia of the encoder calibration mechanism is assumed to be negligible.
3. The temperature has a little effect on motor parameters.
4. The inertias of wires, screws, and belts are insignificant.
5. Motor and its encoder are considered as a unique component of the system.

Simulation: Viscous Damping in the Encoder Shaft

The pendulum is released from a small angle θ_0 and its position is measured through time. A function in the form of the bellow is then fitted to the measured data to find the viscous damping coefficient.

$$\theta = \frac{\sqrt{c^2 + 4Imgl} - c}{2\sqrt{c^2 + 4Imgl}} \theta_0 e^{-\frac{\sqrt{c^2+4Imgl}+c}{2I}x}$$

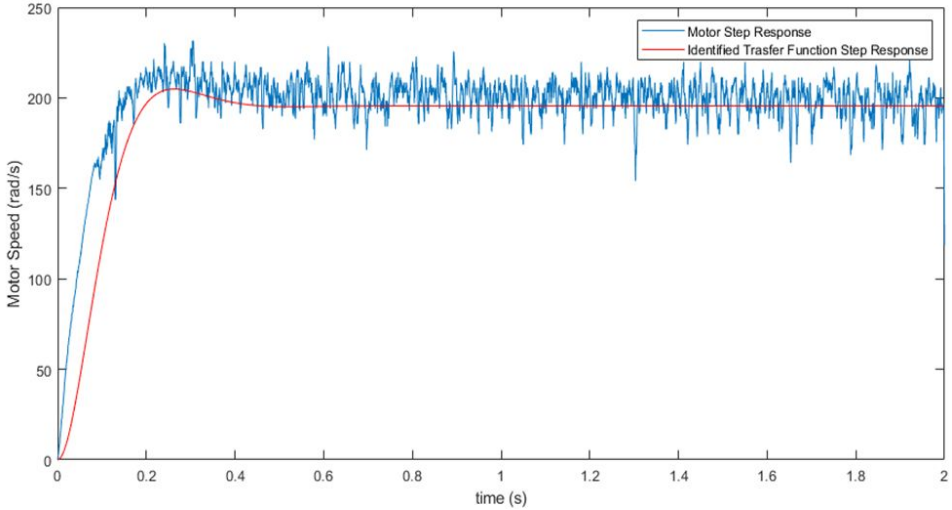
$$+ \frac{\sqrt{c^2 + 4Imgl} + c}{2\sqrt{c^2 + 4Imgl}} \theta_0 e^{\frac{\sqrt{c^2+4Imgl}-c}{2I}x}$$



Simulation: Identifying Motor Parameters

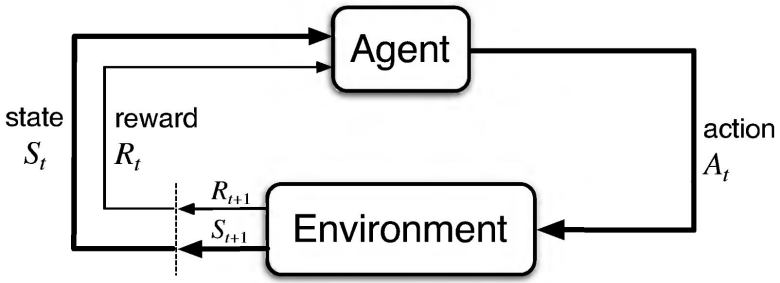
- The resistance and the inductance of the motor are measured
- K_b was measured by rotating the motor shaft and recording the voltage on windings
- Other parameters were determined from the step response of the motor.

$$\frac{\dot{\theta}(s)}{V_a(s)} = \frac{k_t}{IL_a s^2 + (IR_a + cL_a)s + (R_a c + k_b k_t)}$$



Reinforcement Learning

- Involves an **agent**, an **environment**, **actions**, **states,** and **rewards**
- The environment gives a new state and a reward for every action taken by the agent



Source: deepai.org

- The goal is to construct an agent that optimizes a mapping of states to actions to maximize the total reward received. The mapping is called **policy.**

Controlling the Rotary Inverted Pendulum: A Reinforcement Learning Approach

Reinforcement Learning: Q-Learning Algorithm

The action value $Q_\pi(s,a)$ is defined as the expected reward to receive starting from state ***s***, taking action ***a***, and following the policy π.

The **Bellman Optimality Equation** for action values is as follow:

$$Q(s,a) = r(s,a) + \gamma \max_a Q(s',a')$$

- ***r(s,a)*** is the **received reward** for taking action ***a*** in state ***s***.
- ***γ*** is the **discount factor** ($0 \leq \gamma \leq 1$) indicating how much importance is given to future rewards.

The best action to choose is the one that maximizes the right size of the above equation.

Reinforcement Learning: Q-Learning

- The goal of Q-Learning is to find a Q-function mapping each state to an optimal action. The function can be represented using a **Table** or an **Artificial Neural Networks**.
- While using a table, the action values are updated iteratively using this formula:

$$Q^{new}(s,s) = (1-\alpha)Q^{old}(s,a) + \alpha(r + \gamma\, max_{a'} Q(s',a'))$$

Where **α** is the **learning rate** (0 < α ≤1).

- While using a neural network, the network parameters (φ) are updated to minimize:

$$L(\varphi) = \left(Q_\varphi(s,a) - \left(r + \gamma \max_{a'} Q_\varphi(s',a')\right)\right)^2$$

Controlling the Rotary Inverted Pendulum: A Reinforcement Learning Approach

Reinforcement Learning: Deep Deterministic Policy Gradient

- Q-Learning **cannot** be used for solving **continuous-action problems**! DDPG can be considered as the **continuous-action** version of Q-Learning.
- DDPG consists of two separate neural networks:
  - The **actor** network approximates the policy function: $\mu(s) \rightarrow a$
  - The **critic** network approximates the value function: $Q(s, a) \rightarrow q$
- The parameters in both networks (φ and φ) are updated to minimize:

$$L(\varphi) = \left(Q_\varphi(s,a) - \left(r + \gamma \max_{a'} Q_\varphi(s',a')\right)\right)^2 \qquad \max_{a'} Q_\varphi(s',a') = Q_\varphi\left(s', \mu_\psi(s')\right)$$

$$L(\varphi,\psi) = \left(Q_\varphi(s,a) - \left(r + \gamma Q_\varphi\left(s', \mu_\psi(s')\right)\right)\right)^2$$

Controlling the Rotary Inverted Pendulum: A Reinforcement Learning Approach

Reinforcement Learning Approaches to the Problem

Controlling the rotary inverted pendulum can be modeled as both **continuous-action reinforcement learning** and **discrete-action reinforcement learning**.

- In the first approach, the actions can be considered as the continuous input voltage values to the DC motor.
- In the second approach, the voltage domain must be discretized first. The discretized values are as follow:

$$V_1 = -24V\ , V_2 = -18V\ , V_3 = -12V\ , V_4 = -6V\ ,$$

$$V_5 = 0V\ , V_6 = 6V\ , V_7 = 12V\ , V_8 = 18V\ , V_9 = 24V$$

Reinforcement Learning Approaches to the Problem

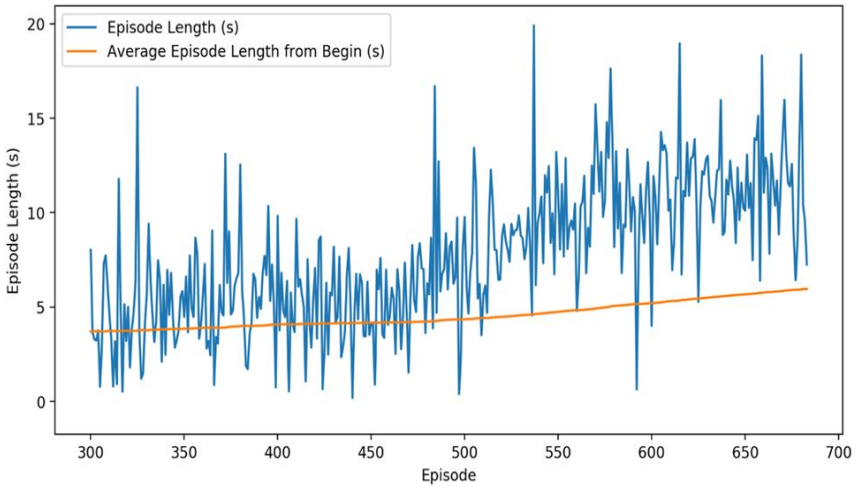
- The reward function used in both algorithms is as below:

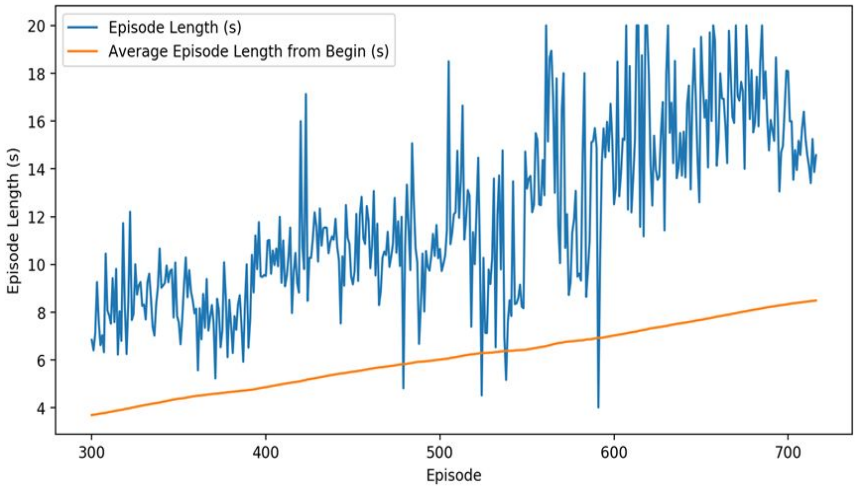
$$R = -\left(\theta_*^2 + 0.1\dot{\theta}_*^{\,2} + 0.0001 * (motor\ voltage)^2\right) \quad , \quad \theta_* \doteq \frac{\theta}{\pi}$$

- Four-layer Feed-Forward neural networks are used in both algorithms.

Results in Simulation

Deep Deterministic Policy Gradient Algorithm (DDPG) marginally outperformed Q-Learning in simulation.



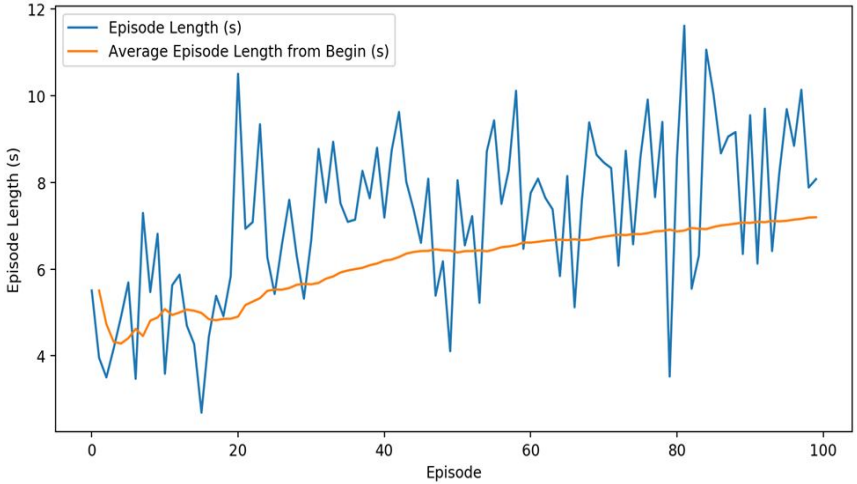


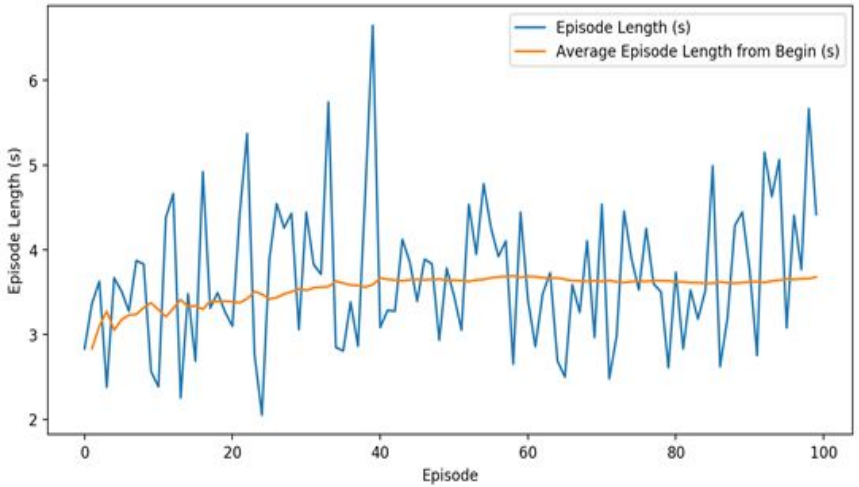
Q-Learning Vs. DDPG

Controlling the Rotary Inverted Pendulum: A Reinforcement Learning Approach

Results in Real Environment

Q-Learning performed considerably better for controlling the physical structure, mainly because it has a lower computation cost than DDPG.





Q-Learning Vs. DDPG

Thank You

for Your Attention!

ahatefi@ut.ac.ir