```python
import matplotlib.pyplot as plt
import numpy as np


# Plotting the output control variable (Good)
def output_controller_good(x):
    if 0 >= x >= -0.15 / 2:
        return_value = (x * 2 / 0.15) + 1
    elif x <= -0.15 / 2 or x >= 0.15 / 2:
        return_value = 0
    else:
        return_value = -(x * 2 / 0.15) + 1
    return return_value


# Plotting the output control variable (High)
def output_controller_high(x):
    if x >= 0 or x < -0.15:
        return_value = 0
    elif -0.15 / 2 >= x >= -0.15:
        return_value = 2 * x / 0.15 + 2
    else:
        return_value = - (2 * x / 0.15)
    return return_value


# Plotting the output control variable (Low)
def output_controller_small(x):
    if x <= 0 or x > 0.15:
        return_value = 0
    elif 0.15 / 2 >= x >= 0:
        return_value = 2 * x / 0.15
    else:
        return_value = - (2 * x / 0.15) + 2
    return return_value

# X-axis definition for the output partition plot
t = np.linspace(-0.2, 0.2, 200)
counter = 0
# Define intervals for the output control variable partition
good_triangle = np.zeros(shape=t.shape)
high_triangle = np.zeros(shape=t.shape)
small_triangle = np.zeros(shape=t.shape)

for i in t:
    good_triangle[counter] = output_controller_good(i)
    high_triangle[counter] = output_controller_high(i)
    small_triangle[counter] = output_controller_small(i)
    counter += 1

plt.ylabel("Membership")
plt.xlabel("$\delta$")
plt.title('Fuzzy Partition for the control variable')
plt.plot(t, small_triangle, label='Low Infection rate')
plt.plot(t, good_triangle, label='Good Infection rate')
plt.plot(t, high_triangle, label='High Infection rate')
plt.legend(bbox_to_anchor=(0.98, 1), loc="upper left")

plt.show()


# Plotting the input partition for the rate of infected bots (High)
def controller_high(x):
    if 1 >= x >= 0.8:
        return_value = 1
    elif 0.8 >= x >= 0.65:
        return_value = (x / 0.15) - 4.33
    else:
        return_value = 0
    return return_value


# Plotting the input partition for the rate of infected bots (Good)
def controller_good(x):
    if 0.7 >= x >= 0.6:
        return_value = -10 * x + 7
    elif 0.6 >= x >= 0.5:
        return_value = (10 * x) - 5
    else:
        return_value = 0
    return return_value


# Plotting the input partition for the rate of infected bots (Low)
def controller_small(x):
    if 0.4 >= x > 0:
        return_value = 1
    elif 0.55 >= x >= 0.4:
        return_value = -(x / 0.15) + 3.666
    else:
        return_value = 0
    return return_value

# X-axis definition for the input measurement partition
k = np.linspace(-0.05, 1, 400)

counter = 0
good_triangle = np.zeros(shape=k.shape)
high_triangle = np.zeros(shape=k.shape)
small_triangle = np.zeros(shape=k.shape)
for x in k:
    good_triangle[counter] = controller_good(x)
    high_triangle[counter] = controller_high(x)
    small_triangle[counter] = controller_small(x)
    counter += 1

plt.ylabel("Membership")
plt.xlabel("$\pi$")
plt.title('Fuzzy partition of the current percentage of the infected bots')
plt.plot(k, small_triangle, label='Low Infection rate')
plt.plot(k, good_triangle, label='Good Infection rate')
plt.plot(k, high_triangle, label='High Infection rate')
plt.xticks(np.arange(0, 1.1, 0.1))
plt.legend(bbox_to_anchor=(0.98, 1), loc="upper left")

plt.show()

```