Project 2 Report: Adaptive Cruise Control and Autonomous Lane Keeping

1. Adaptive Cruise Control

1.1 Problem Statement

- To design and implement the micro-controller Arduino UNO on Ultrasonic Sensor HC-SR04 run down the hill as fast as possible while achieving the following function.

a. Adaptive Cruize control:

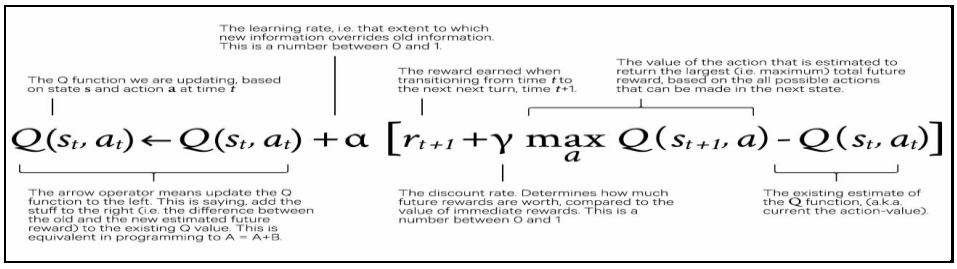
- To stop the RC vehicle 30cm away from the object in-front.
- If the vehicle is in motion it needs to maintain 30cm ahead.

1.2 Technical Approach:

- We used two approaches below are used for the project to achieve

a. Reinforcement Learning approach:

- The reinforcement learning approach enables the RC (agent) to learn the significances (like collecting observations from the environment, decision and receiving strategies, from experiences and iterate accordingly) of actions in a particular environment.
- The Q-Learning is the value-based learning algorithm where the value function receive different updates from value-based learning algorithm based on the equation of Bellman.



Equation 1 (Bellman's Equation)

Reference link:

- https://www.google.com/search?q=bellman+equation+reinforcement+learning&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiMt8eEhtj0AhU6SzABHSuwDqAQ_AUoAnoECAEQBA&biw=767&bih=832&dpr=1.25

b. PID Controller:

- The PID controller acts as the feedback mechanism used in a control system. The PID control stands for Proportional-Integrative-Derivative control. The PID controller is the non-modal based controller.

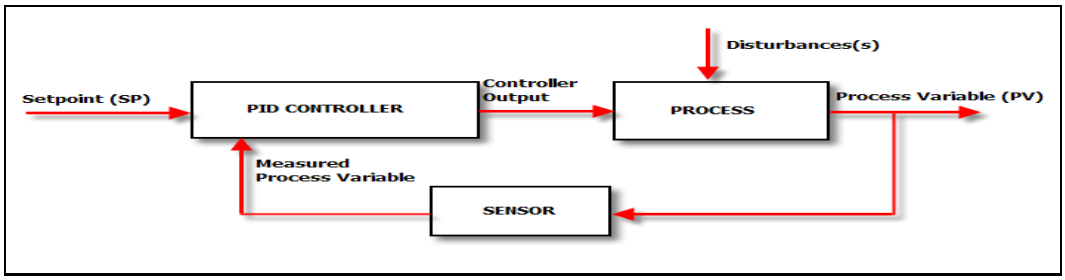


Figure 1

Reference Link:

- https://www.csimn.com/CSI_pages/PIDforDummies.html

1.3 Hardware and Software Implementation

Hardware:

- The hardware components used for the project are

a. Arduino UNO Rev3:

- The Arduino UNO is a microcontroller board which consists of 14 digital input/output pins which is used for variety of electronics projects.

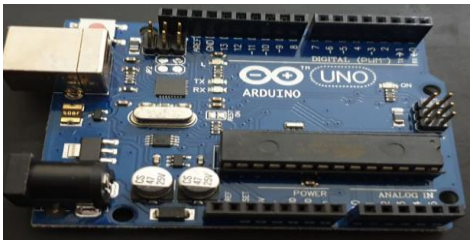


Figure 2

b. Bread board:

- The bread board is the construction base of electronics where it is used to build and test circuits quickly before finalizing the any circuit design further.

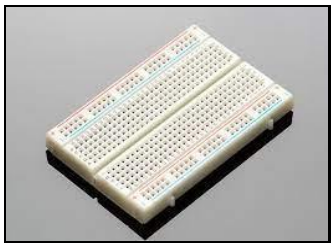


Figure 3

c. HC-SR04 ultrasonic sensor:

- The HC-SR04 ultrasonic sensor is the sensor used for detecting the distance from the object. The sensor uses the non-contact ultrasound sonar to measure the distance to the object.

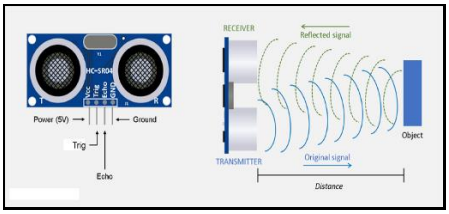


Figure 4

d. Jumper wires (male to male and male to female wires):

- A jumper wire is an electrical wire which is usually used for connecting the components of a breadboard for performing any kind of test or for prototyping.

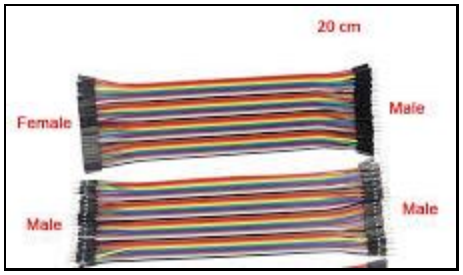


Figure 5

e. Acrylic sheet

- The acrylic sheet is used for the placement of hardware components on it.

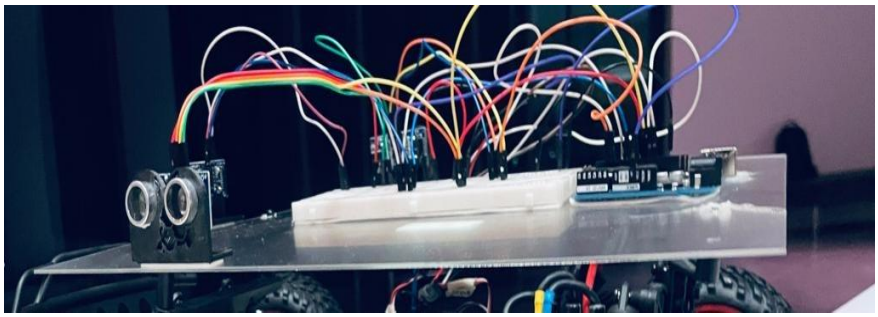


Figure 6

f. 3D-Printed case:

- The 3d printed case is designed to fix the HC-SR04 ultrasonic sensor so that it gives great support to the ultrasonic sensor while the RC vehicle is in motion.

Figure 7

g. RC vehicle

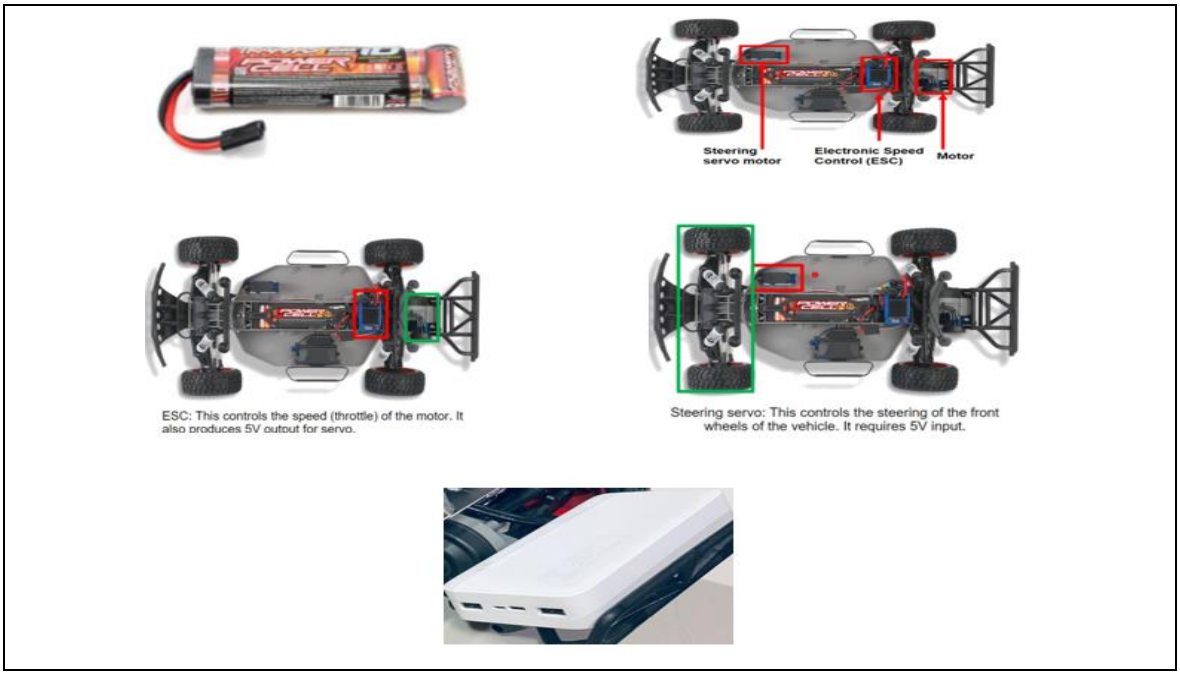


Figure 8 (reference: Lecture Slides)

Software Implementation:

- We used two approaches for the project
  - PID controller
  - Reinforcement Learning

PID CONTROLLER:

- Implementation code for the PID controller:

```
// PID for motor
double kpm =  2.1, kim = 0.00001, kdm = 0.5; // PID Parameters
unsigned long currentTimem, previousTimem;
double elapsedTimem;
double errorm;
double lastErrorm;
double inputm, outputm, setPointm=40.0;
double cumErrorm, rateErrorm;
double computePIDm(double inputm){
        currentTimem = millis();                //get current time
        elapsedTimem = (double)(currentTimem - previousTimem);        //compute time elapsed from previous computation

        errorm = setPointm - inputm;                                // determine error
        cumErrorm = errorm * elapsedTimem;                // compute integral
        rateErrorm = (errorm - lastErrorm)/elapsedTimem;   // compute derivative

        double outputm = kpm*errorm + kim*cumErrorm + kdm*rateErrorm;                //PID output

        lastErrorm = errorm;                                //remember current error
        previousTimem = currentTimem;                        //remember current time

        return outputm;
```

```
double input, output, setPoint=0.0;
double cumError, rateError;
double computePID(double input){
        currentTime = millis();                //get current time
        elapsedTime = (double)(currentTime - previousTime);        //compute time elapsed from previous computation

        error = setPoint - input;                                // determine error
        cumError = error * elapsedTime;                // compute integral
        rateError = (error - lastError)/elapsedTime;   // compute derivative

        double output = kp*error + ki*cumError + kd*rateError;                //PID output

        lastError = error;                                //remember current error
        previousTime = currentTime;                        //remember current time

        return output;
```

REINFORCEMENT LEARNING:

- We can notice from Q-table(table-1) that the rows and columns are formed by states as number of actions.
- So here, we gave the initial value in Q table as zero but later it may change after giving the training further.
- While training, the ten various states and the four various actions (left, right, backward, stop)

```
float Q[STATES][NUMBER_OF_ACTIONS] = {{0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0}};
```

Q Table 1

- ❖ Reward Matrix:
- ➢ Depending upon the RC performing actions the quality of the action (positive or negative) can be observed.

```
int REWARDS[STATES][NUMBER_OF_ACTIONS] = {{10,4,4,6};
                                         {10,4,4,6};
                                         {10,4,4,6};
                                         {10,4,4,6};
                                         {10,4,4,6};
                                         {10,4,4,6};
                                         {10,4,4,6};
                                         {10,4,4,6};
                                         {10,4,4,6};
                                         {10,4,4,6}};
```

Figure 9 (Reward Matrix)

The hyperparameters in the reinforcement learning:

a. Discount rate (γ):

- The discount rate (γ) describes about the importance of future rewards are to the current rate. Basically, the discount factor values are observed range between the 0 - 1. The discount rate (γ) we applied as 0.9.

b. Exploration Rate (Epsilon):

- The exploration rate (epsilon) describes about the probability of the RC vehicle will explore the environment to perform random actions and analyze them through the Q values.
- During the process, we out of 100% we applied 75% actions based on random actions and 25% actions based on the Q table by suing decay function (`ε = ε*0.9`) to reduce the time passing while performing the random actions and to perform based on the Q values.

c. Learning rate (α):

- The learning rate (α) describes about the amount that the weights are reorganized during training period which is referred to as the learning rate.
- The learning rate is a configurable hyperparameter which is used in training period of neural networks. It has a small positive value range between the 0.0 - 1.0.
- During the training period, we applied the learning rate range 0.2 and 0.8 but we are unable to proceed further with optimum learning rate in the process.

About the Q leaning algorithm:

a. Firstly, initializing the Q values.
b. Then, observing the current state(s) values.
c. Choosing the necessary action on state for one of the action selection strategies.
d. After taking the necessary actions, now observe the new state (s) and the reward (r). based on that updating the Q table based on Epsilon Greedy Policy for the state (by observing the "r") and maximize the "r" for obtaining the next state.

```
PROB = RANDOM(EPSILON);
 if (PROB<=EPSILON)      //EXPLORE THE ACTIONS
   {
     ACTION = random(0,3);
     FLAG = 2;
   }
 else                    //EXPLOIT THE ACTIONS FROM Q TABLE
   {
     ACTION = ARGMAX(Q,STATE);
     FLAG = 2;
```

Figure 10 (Epsilon Greedy Policy)

e. Now, the new state is again updated and the process is repeated until reaching the terminal state.

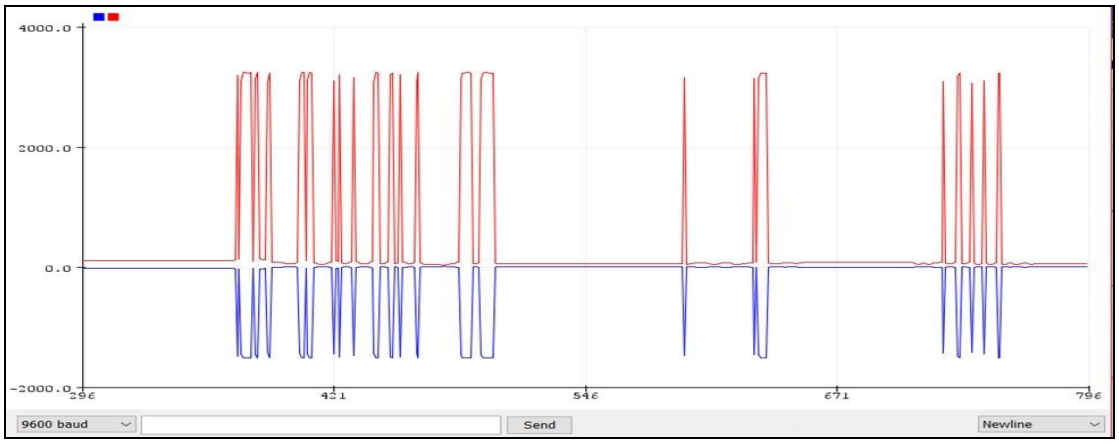
Implementation of Bellmen Equations:

- Based on the Q learning equation (based on Bellman Equation) the Q values in the Q table is updated for defined number of episodes while RC vehicle is performing with new learning approach where the previous Q values in the table is updated with the newer Q values. Finally, the Q values are evaluated and testing is done through reinforcement learning of RC vehicle.

```
Q_OLD = Q_TABLE[S][A];
Q_MAX = MAX(Q_TABLE, NEXT_S);
Q_NEW = (1-LEARNING_RATE)*Q_OLD + LEARNING_RATE*(R + DISCOUNT_FACTOR*Q_MAX);
Serial.print("Q VALUE : ");
Serial.println(Q_NEW);
Q_TABLE[S][A] = Q_NEW;
```

Figure 11 (Implementation of Bellman's Equation)

1.4 Experimental Results;



Plot1(Plot for Throttle)

- **The blue line the plot indicates the error throttle and the re line indicates actual throttle.**

2. Autonomous Lane Keeping

2.1 Problem Statement:

- To design and implement the micro-controller Arduino UNO on Ultrasonic Sensor HC-SR04 run down the hill by performing the autonomous lane keeping function as fast as possible.

2.2 Technical Approach:

- The PID controller acts as the feedback mechanism used in a control system. The PID control stands for Proportional-Integrative-Derivative control. The PID controller is the non-modal based controller.

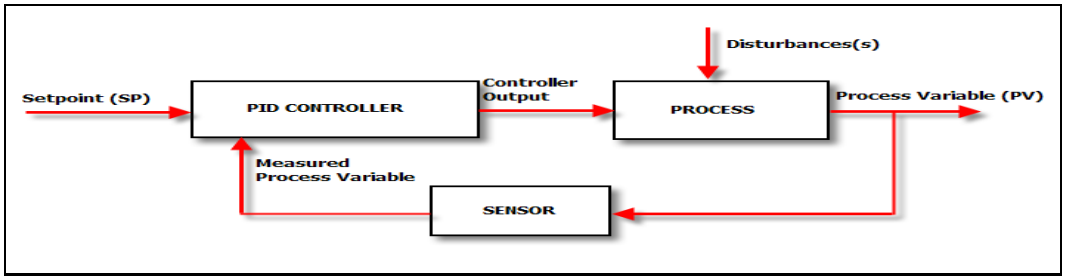


Figure 2

Reference Link:

- https://www.csimn.com/CSI_pages/PIDforDummies.html

2.3 Hardware and Software Implementation:

Hardware:

- The hardware components used for the project are

a. Arduino UNO Rev3:

- The Arduino UNO is a microcontroller board which consists of 14 digital input/output pins which is used for variety of electronics projects.

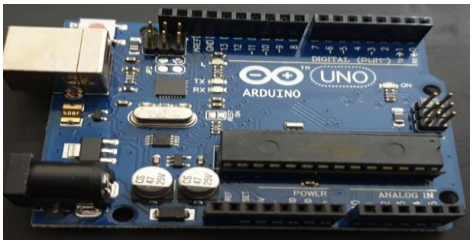


Figure 2

b. Bread board:

- The bread board is the construction base of electronics where it is used to build and test circuits quickly before finalizing the any circuit design further.

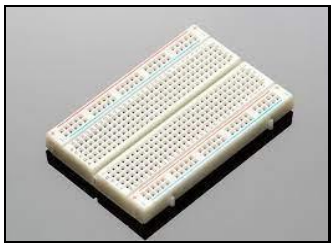


Figure 3

c. HC-SR04 ultrasonic sensor:

- The HC-SR04 ultrasonic sensor is the sensor used for detecting the distance from the object. The sensor uses the non-contact ultrasound sonar to measure the distance to the object.

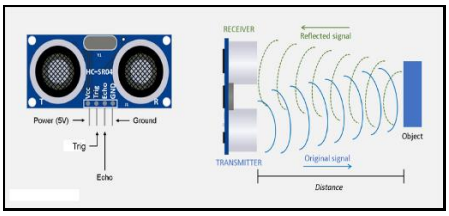


Figure 4

d. Jumper wires (male to male and male to female wires):

- A jumper wire is an electrical wire which is usually used for connecting the components of a breadboard for performing any kind of test or for prototyping.

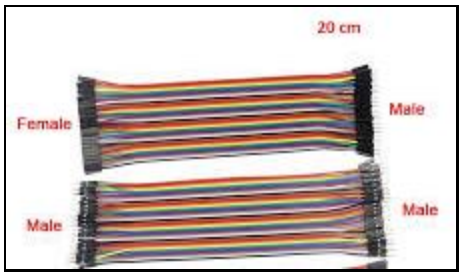


Figure 5

e. Acrylic sheet

- The acrylic sheet is used for the placement of hardware components on it.

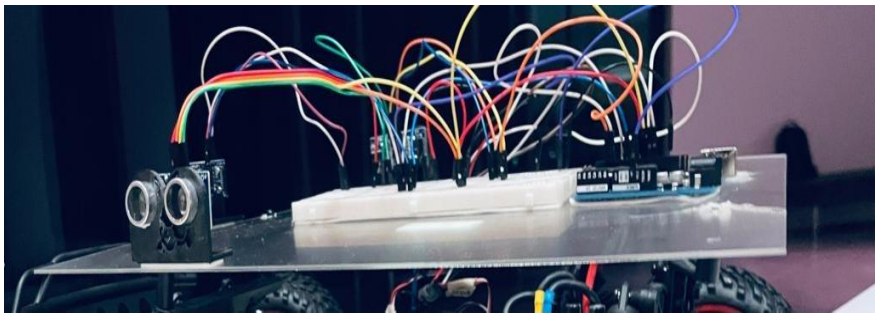


Figure 6

f. 3D-Printed case:

- The 3d printed case is designed to fix the HC-SR04 ultrasonic sensor so that it gives great support to the ultrasonic sensor while the RC vehicle is in motion.

Figure 7

g. RC vehicle:

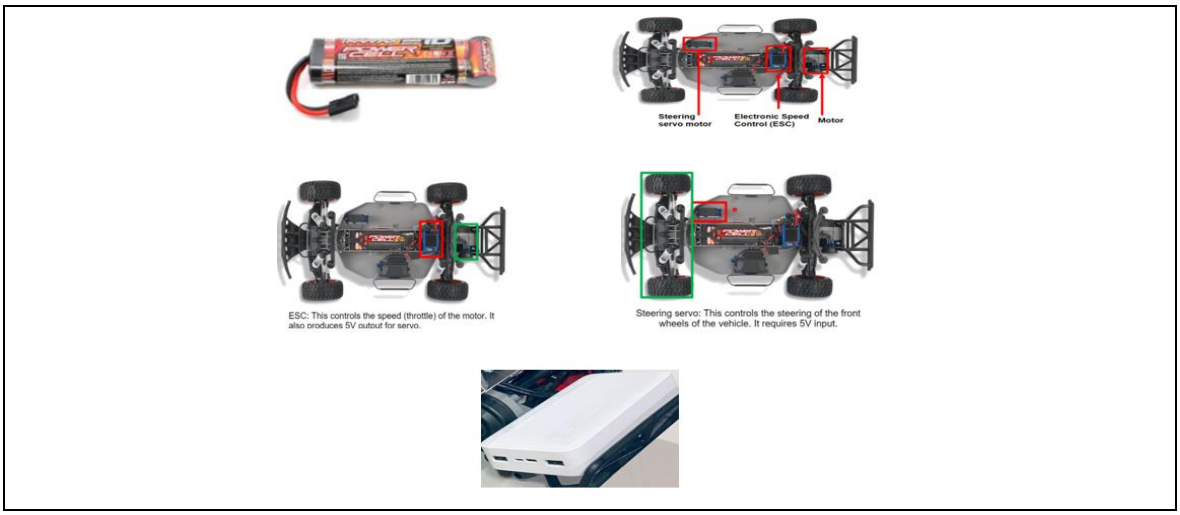


Figure 8 (reference: Lecture Slides)

Software Implementation:

- Implementation code for the PID controller:

```
// PID for motor
double kpm =  2.1, kim = 0.00001, kdm = 0.5; // PID Parameters
unsigned long currentTimem, previousTimem;
double elapsedTimem;
double errorm;
double lastErrorm;
double inputm, outputm, setPointm=40.0;
double cumErrorm, rateErrorm;
double computePIDm(double inputm){
        currentTimem = millis();                //get current time
        elapsedTimem = (double)(currentTimem - previousTimem);        //compute time elapsed from previous computation

        errorm = setPointm - inputm;                                // determine error
        cumErrorm = errorm * elapsedTimem;                // compute integral
        rateErrorm = (errorm - lastErrorm)/elapsedTimem;    // compute derivative

        double outputm = kpm*errorm + kim*cumErrorm + kdm*rateErrorm;                //PID output

        lastErrorm = errorm;                                //remember current error
        previousTimem = currentTimem;                        //remember current time

        return outputm;
```

```
double input, output, setPoint=0.0;
double cumError, rateError;
double computePID(double input){
        currentTime = millis();                //get current time
        elapsedTime = (double)(currentTime - previousTime);        //compute time elapsed from previous computation

        error = setPoint - input;                                 // determine error
        cumError = error * elapsedTime;                // compute integral
        rateError = (error - lastError)/elapsedTime;   // compute derivative

        double output = kp*error + ki*cumError + kd*rateError;                //PID output

        lastError = error;                                //remember current error
        previousTime = currentTime;                        //remember current time

        return output;
```

REINFORCEMENT LEARNING:

- We can notice from Q-table(table-1) that the rows and columns are formed by states as number of actions.
- So here, we gave the initial value in Q table as zero but later it may change after giving the training further.
- While training, the ten various states and the four various actions (left, right, backward, stop)

```
float Q[STATES][NUMBER_OF_ACTIONS] = {{0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0},
                                      {0.0,0.0,0.0,0.0}};
```

Q Table 2

- Reward Matrix:
- Depending upon the RC performing actions the quality of the action (positive or negative) can be observed.

```
int REWARDS[STATES][NUMBER_OF_ACTIONS] = {{10,4,4,6};
                                          {10,4,4,6};
                                          {10,4,4,6};
                                          {10,4,4,6};
                                          {10,4,4,6};
                                          {10,4,4,6};
                                          {10,4,4,6};
                                          {10,4,4,6};
                                          {10,4,4,6};
                                          {10,4,4,6}};
```

Figure 9 (Reward Matrix)

The hyperparameters in the reinforcement learning:

d. Discount rate (γ):

- The discount rate (γ) describes about the importance of future rewards are to the current rate. Basically, the discount factor values are observed range between the 0 - 1. The discount rate (γ) we applied as 0.9.

e. Exploration Rate (Epsilon):

- The exploration rate (epsilon) describes about the probability of the RC vehicle will explore the environment to perform random actions and analyze them through the Q values.
- During the process, we out of 100% we applied 75% actions based on random actions and 25% actions based on the Q table by suing decay function (`ε = ε*0.9`) to reduce the time passing while performing the random actions and to perform based on the Q values.

f. Learning rate (α):

- The learning rate (α) describes about the amount that the weights are reorganized during training period which is referred to as the learning rate.
- The learning rate is a configurable hyperparameter which is used in training period of neural networks. It has a small positive value range between the 0.0 - 1.0.
- During the training period, we applied the learning rate range 0.2 and 0.8 but we are unable to proceed further with optimum learning rate in the process.

About the Q leaning algorithm:

h. Firstly, initializing the Q values.
i. Then, observing the current state(s) values.
j. Choosing the necessary action on state for one of the action selection strategies.
k. After taking the necessary actions, now observe the new state (s) and the reward (r). based on that updating the Q table based on Epsilon Greedy Policy for the state (by observing the "r") and maximize the "r" for obtaining the next state.

```
PROB = RANDOM(EPSILON);
 if (PROB<=EPSILON)      //EXPLORE THE ACTIONS
   {
     ACTION = random(0,3);
     FLAG = 2;
   }
 else                    //EXPLOIT THE ACTIONS FROM Q TABLE
   {
     ACTION = ARGMAX(Q,STATE);
     FLAG = 2;
```

Figure 10 (Epsilon Greedy Policy)

l. Now, the new state is again updated and the process is repeated until reaching the terminal state.

Implementation of Bellmen Equations:

- Based on the Q learning equation (based on Bellman Equation) the Q values in the Q table is updated for defined number of episodes while RC vehicle is performing with new learning approach where the previous Q values in the table is updated with the newer Q values.

```
Q_OLD = Q_TABLE[S][A];
Q_MAX = MAX(Q_TABLE, NEXT_S);
Q_NEW = (1-LEARNING_RATE)*Q_OLD + LEARNING_RATE*(R + DISCOUNT_FACTOR*Q_MAX);
Serial.print("Q VALUE : ");
Serial.println(Q_NEW);
Q_TABLE[S][A] = Q_NEW;
```

Figure 11 (Implementation of Bellman's Equation)

- Finally, the Q values are evaluated and testing is done through reinforcement learning of RC vehicle.

2.4 Experimental Results

- The experimental results obtained are

STATE 1	EPISODE ENDED : 10
Q VALUE : -7.00	START:
EPISODE ENDED : 1	STATE: 1
START :	Q VALUE : - 8.64
STATE 2	EPISODE ENDED: 11
Q VALUE : -3.50	START:
EPISODE ENDED : 2	STATE: 2
START :	Q VALUE : 9.57
STATE 3	EPISODE ENDED: 12
Q VALUE : 5.52	START:
EPISODE ENDED : 3	STATE: 3
START :	Q VALUE : - 6.57
STATE 4	EPISODE ENDED: 13
Q VALUE : 5.52	START:
EPISODE ENDED : 4	STATE: 4
START :	Q VALUE : - 1.56
STATE 5	EPISODE ENDED: 14
Q VALUE : 7.00	START:
EPISODE ENDED : 5	STATE: 5
START :	Q VALUE : - 4.03
STATE 6	EPISODE ENDED: 15
Q VALUE : 5.52	START:
EPISODE ENDED : 6	STATE: 6
START :	Q VALUE : - 9.32
STATE 7	EPISODE ENDED: 16
Q VALUE : 3.50	START:
EPISODE ENDED : 7	STATE: 7
START :	Q VALUE : 5.52
STATE 8	EPISODE ENDED: 17
Q VALUE : -7.00	START:
EPISODE ENDED : 8	STATE: 8

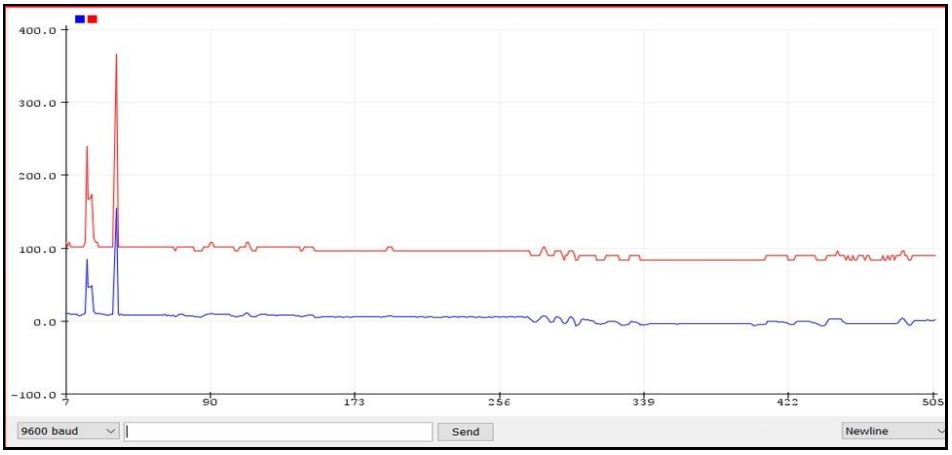
SET VALUES WILL START:	SET VALUES WILL START:
Q VALUE :- 8.54	Q VALUE :- 4.52
Q VALUE :0.00	Q VALUE :7.53
Q VALUE :0.00	Q VALUE :0.00
SET VALUES WILL START:	SET VALUES WILL START:
Q VALUE:- 8.54	Q VALUE :0.00
Q VALUE :0.00	Q VALUE :0.00
Q VALUE :2.43	Q VALUE :9.53
SET VALUES WILL START:	SET VALUES WILL START:
Q VALUE :-7.53	Q VALUE :0.00
Q VALUE :9.20	Q VALUE :7.53
Q VALUE :0.00	Q VALUE: 3.23
SET VALUES WILL START:	SET VALUES WILL START:
Q VALUE :-3.23	Q VALUE :-7.53
Q VALUE :0.00	Q VALUE :12.54
Q VALUE :6.52	Q VALUE :0.00
SET VALUES WILL START:	SET VALUES WILL START:
Q VALUE :-1.12	Q VALUE :0.00
Q VALUE :0.00	Q VALUE :8.54
Q VALUE :3.23	Q VALUE :0.00

figure (Q values) 1

EVALUATION END
ACTION TAKEN: 1
ACTION TAKEN: 1
ACTION TAKEN: 2
ACTION TAKEN: 1
ACTION TAKEN: 2
ACTION TAKEN: 2
ACTION TAKEN: 1
ACTION TAKEN: 1
ACTION TAKEN: 1
ACTION TAKEN: 2

Code evaluation (for 100 episodes)

The plot for the steering angle:



Plot2 (Plot for Steering Angle)

- **The blue line the plot indicates the error steering angle and the re line indicates actual steering angle.**

3. Conclusions and Discussions:

3.1 Conclusions (a summary the results for both the approaches)

- The RC vehicle is kept along the defined center line and stopped at 30 cm away from the object.

- Limitations for both the approaches:
  - The stray data is observed from Ultrasonic sensor.
  - Speed of motor depends on state of charge of battery.
  - By using pulse width modulation, the motor gets stalled below the duty cycle of 22%.
  - During defining the states and actions it was difficult to define the
  - During training the model through reinforcement the complexity of time code is observed.
  - There was a memory limitation observed on the Arduino UNO microcontroller.

3.2 Discussions (a comparison of different approaches, and potential future work to further improve each approach)

- The PID controllers are more common preferred compared to Reinforcement Learning as it is easy to understand because the implementation of reinforcement learning and execute the data is quite difficult.
- During the process of reinforcement learning, we might have only few information about the RC vehicle to realize the rewards.
- We choose the trial-and-error method to tune the PID controller. For the motor controller we assigned the parameters as

$$kpm = 2.1, kim = 0.00001, kdm = 0.5$$

$$kp = 0.2, ki = 0.0, kd = 0.05.$$