#include <Servo.h>

void Mostrar(float Q[][4]); // function to print Q table array on Serial Monitor

float distancia;

float tiempo;

int TRIGGER=7,ECHO=8; // sonar pins

Servo servo1,servo2;

int valor=0; // not used

int act=0;

int ang=40; // servo1 (initial value) angle

int ang2=0; // servo2 angle

int ang\_t=0; // temp holders for servo write routine (slowing down servo)

int ang2\_t=0;

float Q[16][4]={{ 0, 0, 0, 0}, // col 1 and 2 hold servo1 values and col3 and col4 hold servo2 values

{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}

};

int action=0;

int state=0;

int cont=0; // not used

float gamma = 0.8;

float Qmax=0;

float a=0,b=0;

int x=0;

int goal=15; // state 15 is goal

void setup (){

servo1.attach(9);

servo1.write(0); //starting position for servo 1

delay(1000);

servo2.attach(6);

servo2.write(0);

delay(1000);

pinMode(TRIGGER, OUTPUT); // setup sonar

pinMode(ECHO, INPUT);

Serial.begin(9600);

float R[16][4] = { // columns 1 and 2 are for servo1 and columns 3 and 4 are for servo2

{ 0, -1, 0, -1}, // servos in highest position cannot raise first servo nor raise second (servo angle of zero)

{-1, -1, 0, 0}, // can lower or raise 2nd servo

{-1, -1, 0, 0}, // can lower or raise 2nd servo

{-1, -1, -1, 0}, // can only raise 2nd servo

{ 0, 0, 0, -1}, // can lower and raise first servo AND lower 2nd servo

{-1, 0, 0, 0}, // can raise 1st servo AND lower and raise 2nd servo

{-1, 0, 0, 0}, // can raise 1st servo AND lower and raise 2nd servo

{-1, 0, -1, 0}, // can raise 1st servo and raise 2nd servo

{ 0, 0, 0, -1}, // can lower AND raise 1st servo and lower 2nd servo

{-1, 0, 0, 0}, //9 can raise 1st servo and lower and raise 2nd servo

{-1, 0, 0, 0}, //10 can raise 1st servo and lower and raise 2nd servo

{-1, 0, -1, 0}, //11 can raise 1st servo and can raise 2nd servo

{-1, 0, 0, -1}, //12 can raise 1st servo and can lower 2nd servo

{-1, -1, 0, 0}, //13 can raise and lower 2nd servo

{-1, -1, 1000,0}, // can raise and lower 2nd servo

{-1, 0, -1, 0}

}; // servos in their lowest position - can raise 1st servo and can raise 2nd servo

// pos array is valid action table - it contains 3 columns because there can be up to 3 distinct valid actions

int pos[16][3]={ // 4 possible action values: 0=servo1 down -- 1=servo1 up -- 2=servo2 down -- 3=servo2 up rows are states

{0,2,0},

{2,3,2}, //01 230 (old values)

{2,3,3}, //02 - use 233 - the second 3 is just a place holder as must fill all 3 column values with valid value

{3,3,3}, //03 330

{0,1,2},

{2,3,3}, //05 230

{2,3,3}, //06 230

{3,3,3}, //07 330

{0,1,2}, //08

{2,3,3}, //09 230

{2,3,3}, //10 230

{3,3,3}, //11

{1,2,1}, //12

{2,3,3}, //13

{2,3,3}, //14

{1,3,3}, //15

};

int nstate=0;

float diff=0,d\_prom=0,d\_ant=0, d\_new=0;

float point=0;

int cc=0; // not used

for(int d=0;d<20;d++){ // get starting distance - average over 20 mearsurements (probably not necessary to average)

d\_prom=dist()+d\_prom;

delay(100);

}

d\_ant=d\_prom/20;// 20 times

Serial.println(d\_ant);

delay(1000); // exit(0); // exit here to just test SR04 sensor

for (int epoca=0;epoca<2;epoca++) // for 10 episodes (even 5 episodes usually works)

{

randomSeed(analogRead(0));// RandomSeed

// state=random(15); // randomly select a state 0 to 14 - problem with this is that it allows negative

// servo positions to be generated - not too good for servo

state = 0; // start at highest arm position every time

ang=40; // servo 1 starting position

ang2=0; // servo 2 starting position

while(state!=goal){

ang\_t =ang; // used to write from old angle to new one in servo writing function (for slowing down servo speed)

ang2\_t=ang2; // same

cc=0; // not used

cont++; // not used

//x=random(2); //original progam was only using 2 possible actions but this limits the number of states being accessed

x=random(3); // get one of 3 possible action numbers accesses more states

//x=2;

action=pos[state][x]; // choose a valid a ction for that state from pos array

// if action 0 or 1 then select next closest servo 1 position

// if action 2 or 3 then select next closest servo 2 position

if(action==0){ // servo 1 down and servo2 at 0

nstate=state+4; // make next state where servo 1 moves down

ang=ang+20;

ang2=0;

}

else if(action==1){ // servo 1 up and servo2 at 0

nstate=state-4; // make next state where servo 1 moves up

ang=ang-20;

ang2=0;

}

else if(action==2){ // servo 2 down

nstate=state+1; // make next state where servo2 moves down

ang2=ang2+45;

}

else if(action==3){ // servo 2 up

nstate=state-1; // make next state where servo2 moves up

ang2=ang2-45;

}

// move servos //

Serial.print(" episode = ");Serial.print(epoca);

Serial.print(" state = ");Serial.print(state);

Serial.print(" action= ");Serial.print(action);

Serial.print(" ang = ");Serial.print(ang);

Serial.print(" ang2 = ");Serial.println(ang2);

servoVelocidad(servo1,ang\_t,ang,5); // // move servo1 ----5 is just delay speed for writing servo(0=full speed)

servoVelocidad(servo2,ang2\_t,ang2,5); // move servo2

d\_new=dist(); // get distance

diff=d\_new-d\_ant; // see how much moved from last distance

d\_ant=d\_new;

if(diff>=1.9 ){ // if more than 1.9 cm then...

point=map(diff,1,4,5,10); // maps from 1-4 to 5-10

R[nstate][action]=point; // increase reward for movement forward in next state (future reward)

Serial.println(point);

}

Serial.println(" ");

a = -10;

for (int i = 0; i < 4; i++) {

if (a < Q[nstate][i]) { // get max value of next state

a = Q[nstate][i];

}

}

Qmax = a \* gamma; // get percentage of max value

Q[state][action] = R[state][action] + Qmax; // calculate and store Q value

state = nstate;

} // while not goal

} // end each epoch

Mostrar(R); // show reward table

Serial.println(" ");

Serial.println(" ");

Mostrar(Q); // show q table

} // end setup

void loop(){ // main loop reads Qtable and performs actions

//state = random(3); // randomly choose state from 0 to 3 so that goal state is not chosen first I assume

state=0; // start out at zero state every time (highest position of arm)

ang=40; // starting angle of servo1

ang2=0; // starting angle of servo2

Serial.println("Starting main loop... ");

while(state!=goal){//goal=15

b = -10;

for (int i = 0; i < 4; i++) { // find highest value in Qtable for selected state and get that action number

if (b <= Q[state][i]) {

b = Q[state][i];

act = i;

}

}

ang\_t=ang;

ang2\_t=ang2;

Serial.print(" b = ");Serial.print(b);

Serial.print(" state = ");Serial.print(state);

if(act==0){ // servo1 down and servo2 at 0

state=state+4;

ang=ang+20;

ang2=0;

}

else if(act==1){ // servo1 up and servo2 at 0

state=state-4;

ang=ang-20;

ang2=0;

}

else if(act==2){ // servo2 down

state=state+1;

ang2=ang2+45;

}

else if(act==3){ // servo2 up

state=state-1;

ang2=ang2-45;

}

Serial.print(" act= ");Serial.print(act);

Serial.print(" ang = ");Serial.print(ang);

Serial.print(" ang2 = ");Serial.println(ang2);

servoVelocidad(servo1,ang\_t,ang,25); // move servo1 at speed 25

servoVelocidad(servo2,ang2\_t,ang2,25); // move servo2 at speed 25

} // end while not goal

Serial.println("End main loop. ");

} // end loop

void Mostrar(float Q[][4]){ // routine to print Qtable on monitor

for (int i=0;i<16;i++){

for(int j=0;j<4;j++){

Serial.print(Q[i][j]);

Serial.print(" ");

}

Serial.println(" ");

}

} // end show Q table

float dist() { // routine to measure distance

digitalWrite(TRIGGER, LOW);

delayMicroseconds(5);

digitalWrite(TRIGGER, HIGH);

delayMicroseconds(10);

digitalWrite(TRIGGER, LOW);

// Calcula la distancia midiendo el tiempo del estado alto del pin ECHO

tiempo = pulseIn(ECHO, HIGH);

distancia = tiempo / 58.00;

Serial.print(tiempo);Serial.print(" ");

Serial.print(distancia);

Serial.println("cm");

delay(100);

return distancia;

}// end get sonar distance routine

void servoVelocidad(Servo servo, int anguloA, int anguloB, int velocidad) { // routine to control speed of servo

if (anguloA){

for (int angulo = anguloA; angulo <= anguloB; angulo=angulo+2) {

servo.write(angulo);

delay(velocidad);

}

}

else {

for (int angulo = anguloA; angulo >= anguloB; angulo=angulo-2){

servo.write(angulo);

delay(velocidad);

}

}

} // end servo write routine