
Reinforcement learning with arduino using Q-learning

- Gerardo Franco Delgado -

Project Report
gfd.francodelgado@gmail.com

RobotistasMX
15/07/2019



IT
RobotistasMX - www.robotistas.mx

Title:

Reinforcement learning with arduino
using Q-learning

Theme:

Machine learning

Project Period:

Abstract:

This document describes a basic exercise of reinforcement learning using an Arduino. The project aim is to simulate a house with eight rooms, and we want to know the shortest path from one room to another.

Project Group:

RobotistasMX

Participant(s):

Gerardo Franco Delgado

Page Numbers: 17

Date of Completion:

July 17, 2019

The content of this report is freely available.

Chapter 1

Introduction

Reinforcement learning works in a similar way of human learning, the algorithm needs a lot of training to achieve a goal like people (if you want to be the best in something you will need a lot of training). Technically the algorithm with RL chooses the possible action that has more reward and then the algorithm modifies the reward depends on the performance of each action, this is how the algorithm learns the best possible action.

This document describes a basic exercise of reinforcement learning using an Arduino. The project aim is to simulate a house with eight rooms, and we want to know the shortest path from one room to another. The algorithm will start in the main room and it will search in all others rooms until the algorithm can find the target room.

The loop consists of 6 (can be more) iterations, in each iteration, the algorithm always will start in the main room and it will be finished when the target room was found. It can be said that rewards will be given to the algorithm based on the decisions that it takes. To simulate the exercise we'll need eight LEDs because one led represents one room of the house, also we will need protoboard, jumper wires, and Arduino.

1.1 Algorithm

1.1.1 Q-learning

Q-learning is reinforcement learning technique, can be used to find an optimal action-selection policy.

1.1.2 Learning algorithm

Description of RL Algorithm: Q-Learning.

Example 1.1 (How does the algorithm learn?)

Here is the explanation of the algorithm

$$Q(s_t, a_t) \leftarrow -Q(s_t, a_t) + \alpha[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (1.1)$$

The Q function we are updating based on state s and action a at time t.

The arrow operator means update the Q function to the left. This is saying, add the stuff to the right (i.e the difference between the old and the new estimated future reward) to the existing Q value. This is equivalent in programming to $A = A + B$.

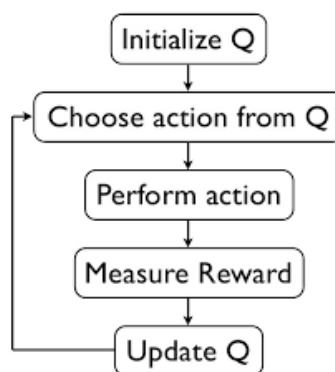
The reward earned when transitioning from time t to the next turn, time t+1.

The discount rate. Determines how much future rewards are worth, compared to the value of immediate rewards. This is a number between 0 and 1.

The value of the action that is estimated to return the largest (i.e maximum) total future reward, based on all possible actions that can be made in the next state.

The existing estimate of the Q function (a.k.a current the action-value)

The above explanation can be summarized in the following image:



We can understand the algorithm function in a easier way with this diagram.

Finally , the algorithm can be described in the next way.

Example 1.2 (Simplify the annotated explanation)

$$(The\ new\ action\ value = The\ old\ value) + The\ learning\ rate(The\ new\ information - the\ old\ information) \quad (1.2)$$

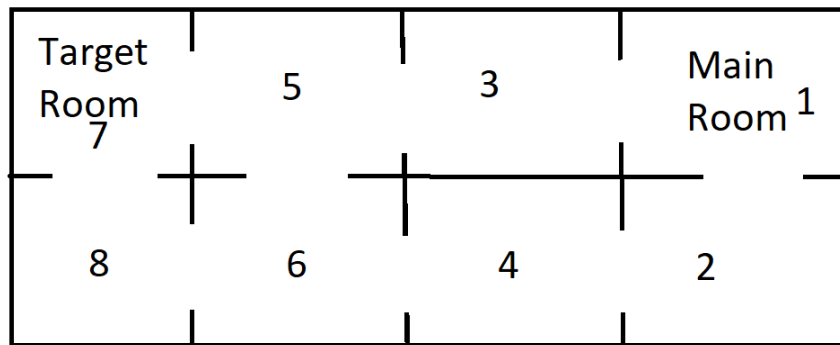
1.1.3 Pseudocode

Once we know the functioning of the algorithm we can understand the pseudo-code, this section shows the core of the code that we are going to use for our algorithm can learn.

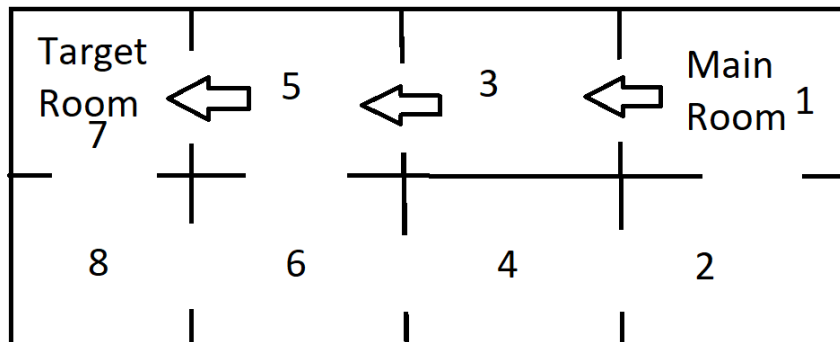
1. loop (i < iterations number)
2. loop (current room != target room)
3. action = random possible actions
4. Q[current room, action] = reward[current room, action] + alpha*max[Q[action,
5. all actions]]
6. current room = action
7. end loop
8. i ++
9. end loop

1.2 Problem

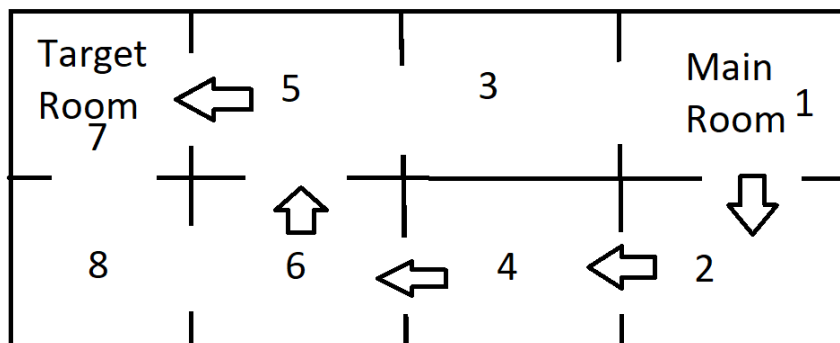
The following image shows a house with 8 rooms, we can see between room[3] and room[4] cannot be passed, so the goal is based on reinforcement learning the algorithm can find the shortest route from the room[1] to [7] which is the target.



Obviously the shortest route is to pass rooms [1], [3], [5] and arrive to [7] as shown in the image below



Although the algorithm will take several routes in each iteration with the aim that it can find the most optimal path, the loop starts again arriving at the target room. The next image shows an alternative route in the process of learning.



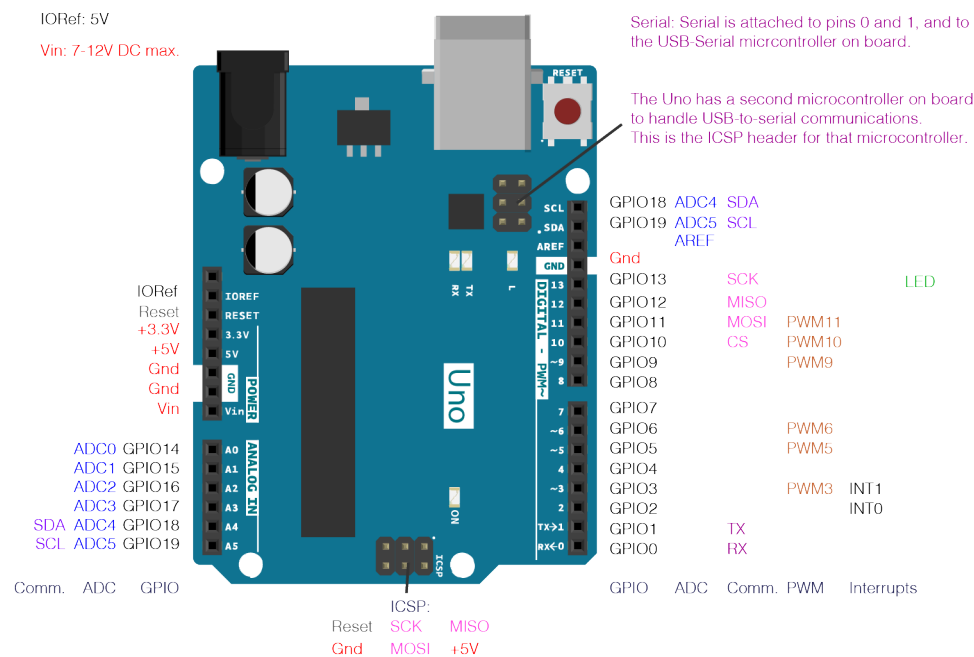
Now we will resolve the problem in a practical way using arduino with Q-learning

Chapter 2

Arduino with Q-learning

2.1 Electronic circuit

The first thing that we have to do is the electronic circuit with Arduino and we will need to connect the LEDs to simulate the eight rooms.



We'll need resistors of 220 ohms and the LEDs will connect to the arduino in the following way:

Room 1 (Main Room) = GPIO 4

Room 2 = GPIO 8

Room 3 = GPIO 5

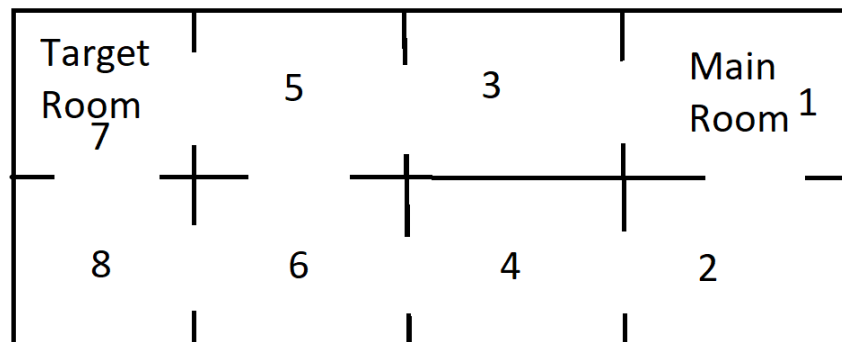
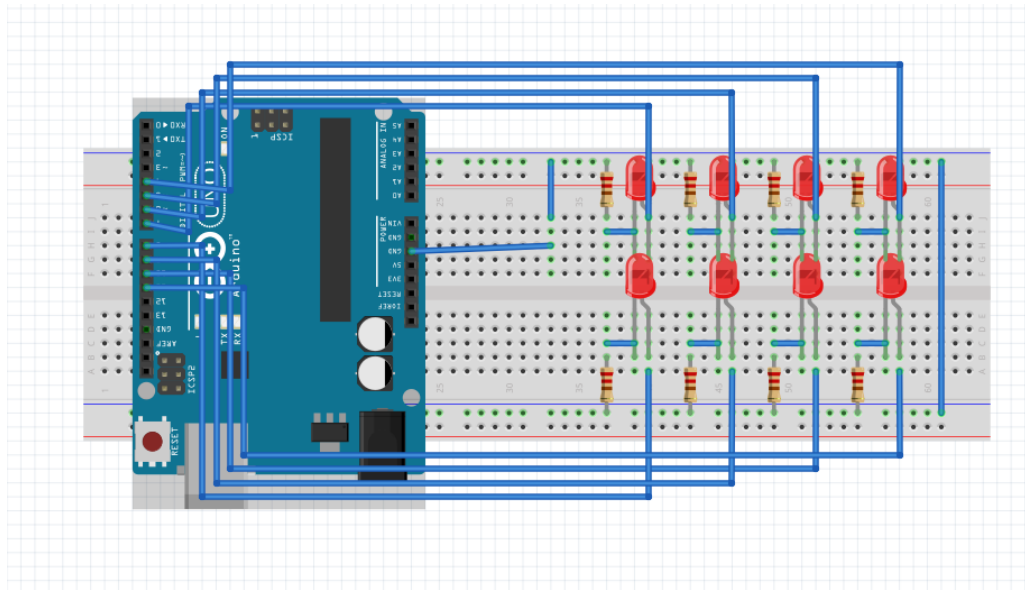
Room 4 = GPIO 9

Room 5 = GPIO 6

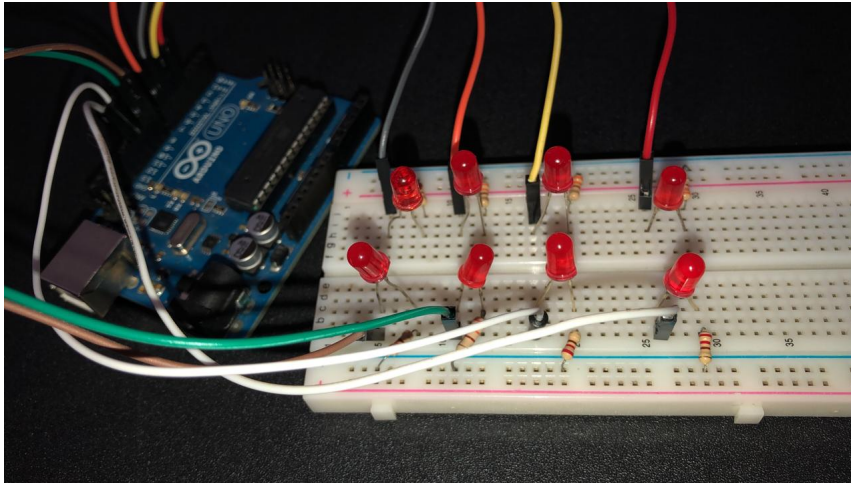
Room 6 = GPIO 10

Room 7 (target room) = GPIO 7

Room 8 = GPIO 11



Once our circuit has been created, we will pass to the part of the programming between arduino and Q-learning

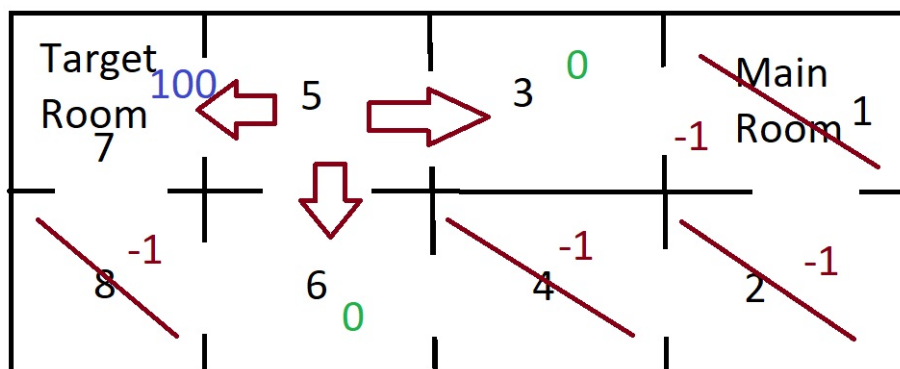


2.2 Code

The first thing that we have to do is create a matrix, the columns represent the possible actions can be taken by the algorithm, and the rows represent the current state where the algorithm is, in this case, represent what is the current room. That matrix is going to have the rewards of all possible options. We can see in the next image that the values with -1 is because the algorithm cannot go for this choice. For example, if we are in room [5] in the first column (`rewards[5][1]`) we have in the matrix -1, (is impossible to go to the room1!, if we are in the room 5, we can't go to room 1 because before we will need to go to the room 3 and now in this room we can go to the room 1). So in the matrix all the possible choices we put 0 and the impossible options we put -1, although it is important to say that if we can move to the target room that is the room 7, in this case, the reward must be major.

```
int rewards [8][8] =
{
  {-1, 0, 0, -1, -1, -1, -1, -1},
  {0, -1, -1, 0, -1, -1, -1, -1},
  {0, -1, -1, -1, 0, -1, -1, -1},
  {-1, 0, -1, -1, -1, 0, -1, -1},
  {-1, -1, 0, -1, -1, 0, 100, -1},
  {-1, -1, -1, 0, 0, -1, -1, 0},
  {-1, -1, -1, -1, -1, -1, -1, -1},
  {-1, -1, -1, -1, -1, -1, 0, 100, -1}
};
```

In the next image shows why in the row five has the values: -1,-1,0,-1,-1,0,100,-1



Also, we need weights matrix that can help algorithm to choose the best action. The choice that has more weight it will be the action that the algorithm runs. It is worth noting that at the begging the matrix starts with 0.

```
int weights [8][8] =
{
    {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}
};
```

It is necessary two functions for the LEDs that can be turn off or turn on, depends on what we need.

```
void turnOn(){
    digitalWrite(cuarto1, HIGH);
    digitalWrite(cuarto2, HIGH);
    digitalWrite(cuarto3, HIGH);
    digitalWrite(cuarto4, HIGH);
    digitalWrite(cuarto5, HIGH);
    digitalWrite(cuarto6, HIGH);
    digitalWrite(cuarto7, HIGH);
    digitalWrite(cuarto8, HIGH);
}
```

```
void turnOff(){
    digitalWrite(cuarto1, LOW);
    digitalWrite(cuarto2, LOW);
    digitalWrite(cuarto3, LOW);
    digitalWrite(cuarto4, LOW);
    digitalWrite(cuarto5, LOW);
    digitalWrite(cuarto6, LOW);
    digitalWrite(cuarto7, LOW);
    digitalWrite(cuarto8, LOW);
}
```

The variables need to be declared, for the rooms we put the number of the GPIO where the LEDs were connected. After that, we need a variable to store the current room, another variable 'target' in this case we put six because in the matrix it starts with 0, so the room 7 is the number 6. Also, we need a variable to store the random number to choice a possible action. Finally, the variables num-max, n, and index are going to be explained in the future.

```
int cuarto1 = 4;
int cuarto2 = 11;
int cuarto3 = 5;
int cuarto4 = 10;
int cuarto5 = 6;
int cuarto6 = 9;
int cuarto7 = 7;
int cuarto8 = 8;

int currentRoom = 0;
int target = 6;
int action = 0;
int num_random = 0;
int num_max = 0;
int n=0;
int index = 0;
int sound = 10;
```

To reduce the code I created a function where I can send the number of the current room and the led will turn on.

```
void room(int n){
    switch(n){
        case 0:
            turnOff();
            digitalWrite(cuarto1, HIGH);
            break;
        case 1:
            turnOff();
            digitalWrite(cuarto2, HIGH);
            break;
        case 2:
            turnOff();
            digitalWrite(cuarto3, HIGH);
            break;
        case 3:
            turnOff();
            digitalWrite(cuarto4, HIGH);
            break;
        case 4:
            turnOff();
            digitalWrite(cuarto5, HIGH);
            break;
        case 5:
            turnOff();
            digitalWrite(cuarto6, HIGH);
            break;
        case 6:
            turnOff();
            digitalWrite(cuarto7, HIGH);
            break;
        case 7:
            turnOff();
            digitalWrite(cuarto8, HIGH);
            break;
    }
}
```

In the setup function is going to be necessary declare the pin outputs for the eight LEDs

```
void setup() {  
    pinMode(cuarto1,OUTPUT);  
    pinMode(cuarto2,OUTPUT);  
    pinMode(cuarto3,OUTPUT);  
    pinMode(cuarto4,OUTPUT);  
    pinMode(cuarto5,OUTPUT);  
    pinMode(cuarto6,OUTPUT);  
    pinMode(cuarto7,OUTPUT);  
    pinMode(cuarto8,OUTPUT);  
    pinMode(sound,OUTPUT);  
}
```

The algorithm will search the target room and later it will start again with a different path during six times, finally, it will decide which path was better. With the function room we can turn on the led to show in what room the algorithm is, later the algorithm needs to choose randomly a number of the possible action, so it's necessary to check in the matrix `rewards[current room][num random]` doesn't exist a -1, if it is the case the algorithm must think of another random number.

Now comes the most important section of the code because now that the algorithm thinks in the possible action, it is necessary to know what choice of this action has the highest weight to storage in num max for determinate the weight of the current room and the action with the formula:

$$\text{weights}[\text{current room}][\text{action}] = \text{rewards}[\text{current room}][\text{action}] + 0.8 * \text{num max.}(\text{Alpha is } 0.8, \text{ you can change the value}).$$

After that, the algorithm moves to another room and it repeats the steps until it can find the target room. To indicate the end of each iteration all the LEDs will turn on and turn off two times.

```
void loop() {  
  while(n < 6){  
    while(currentRoom != target){  
      room(currentRoom);  
      num_random = random(0, 7);  
      while(rewards[currentRoom][num_random] < 0){  
        num_random = random(0, 7);  
      }  
      action = num_random;  
      num_max = weights[action][0];  
      for(int i=1; i<8; i++){  
        if(weights[action][i] > num_max){  
          num_max = weights[action][i];  
        }  
      }  
      weights[currentRoom][action] = rewards[currentRoom][action] + 0.8 * num_max;  
      currentRoom = action;  
      delay(400);  
    }  
    room(target);  
    delay(400);  
    n++;  
    currentRoom=0;  
    for(int i =0; i<2; i++){  
      turnOff();  
      digitalWrite(sound,LOW);  
      delay(500);  
      turnOn();  
      digitalWrite(sound,HIGH);  
      delay(500);  
    }  
    digitalWrite(sound,LOW);  
  }  
}
```

Finally, to test if the algorithm it will move to the choices that have more weight, the variable `index` help us to turn on the LED of the room with the highest reward.

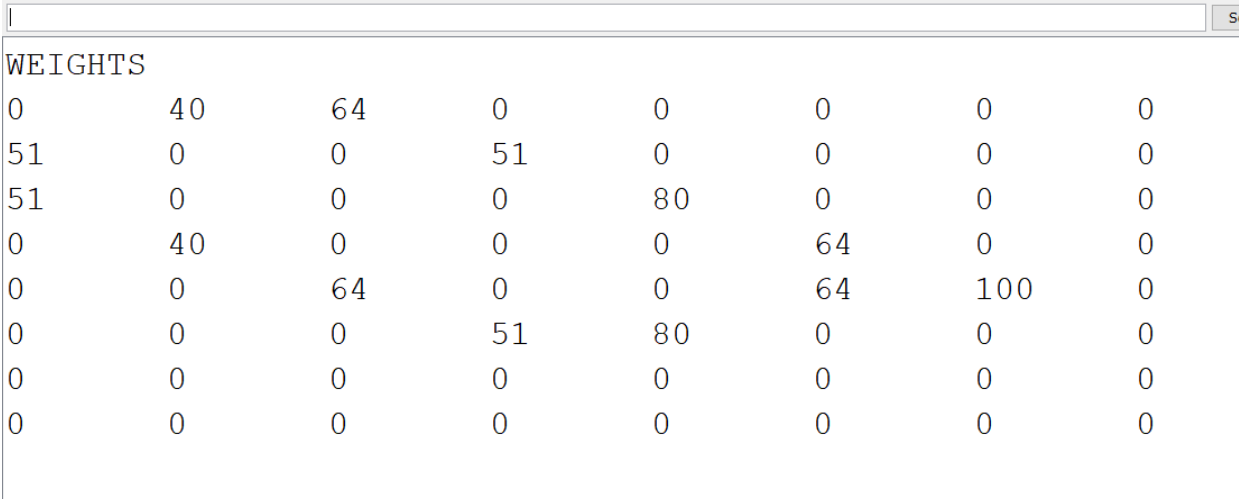
```
currentRoom = 0;
while(currentRoom != target){
    room(currentRoom);
    num_max = weights[currentRoom][0];
    index = 0;
    for(int i=1; i<8; i++){
        if(weights[currentRoom][i] > num_max){
            num_max = weights[currentRoom][i];
            index = i;
        }
    }
    currentRoom = index;
    delay(1000);
}
room(target);
delay(10000);
}
```

Chapter 3

Results and Conclusions

3.1 Results

The weights we can see in the next image:



A screenshot of a terminal window displaying a weight matrix. The window has a title bar with a search icon and the text 'Search'. The content shows the word 'WEIGHTS' followed by an 8x8 grid of numerical values. The values are: Row 1: 0, 40, 64, 0, 0, 0, 0, 0; Row 2: 51, 0, 0, 51, 0, 0, 0, 0; Row 3: 51, 0, 0, 0, 80, 0, 0, 0; Row 4: 0, 40, 0, 0, 0, 64, 0, 0; Row 5: 0, 0, 64, 0, 0, 64, 100, 0; Row 6: 0, 0, 0, 51, 80, 0, 0, 0; Row 7: 0, 0, 0, 0, 0, 0, 0, 0; Row 8: 0, 0, 0, 0, 0, 0, 0, 0.

WEIGHTS							
0	40	64	0	0	0	0	0
51	0	0	51	0	0	0	0
51	0	0	0	80	0	0	0
0	40	0	0	0	64	0	0
0	0	64	0	0	64	100	0
0	0	0	51	80	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

If we are in the first room we can see that the best option is going to go to the room [3]. In this room, the highest weight is 80, so the algorithm will choose to go to the room[5] and finally in this room the algorithm can arrive at the target room.

As you can see, the algorithm worked in the right way.

Here is a video:

Reinforcement learning with arduino (q-learning arduino).

3.2 Conclusions

In case you have questions, comments, suggestions or have found a bug, please do not hesitate to contact me. You can find my contact details below.

Gerardo Franco Delgado
gfdgeras@gmail.com

Bibliography

- [1] Jake Bennett. *The Algorithm Behind the Curtain: Understanding How Machines Learn with Q-Learning*. <https://randomant.net/the-algorithm-behind-the-curtain-understanding-how-machines-learn-with-q-learning/>. 2016.
- [2] Ali Demir. *Q-Learning Algorithm and Basic Implementation on Arduino*. <https://alidemir1.github.io/qlearning-post/>. 2017.
- [3] From Wikipedia the free encyclopedia. *Q-Learning*. <https://en.wikipedia.org/wiki/Q-learning/>. 2017.