Skip: The learning lamp

Chirstian Sivertsen
S167916
c.sivertsen@student.tue.nl

Mick Haegens
S165003
m.w.heagens@student.tue.nl

Simone Rietmeijer
S153538
s.z.rietmeijer@student.tue.nl

Davide José Nogueia Amorim
S162143
d.j.nogueira.amorim@student.tue.nl

Abstract
In this report the use of reinforcement learning in household items is explored through the implementation of the algorithm in a lamp. The lamp, named Skip, was designed to be able to distinguish different situations with the use of multiple sensors and can be reinforced through voice commands. A learning algorithm inspired by Q-learning was written to train the lamp, resulting in a working and learning prototype. No extended user testing has taken place in actual household settings yet. It therefore is unclear whether the lamp trumps an old fashioned one, but initial experiences are promising.

Authors Keywords
Lamp; reinforcement learning; learning algorithm

Introduction
Artificial Intelligence. In 1956 the first seminar on this topic was organized. A few years earlier, in 1950, a paper by Alan Turing was published in which he explores the notion of machines with the ability to simulate human beings, including the ability to learn (Turing, 1950). A little over sixty years later, artificial intelligence is becoming increasingly present. People using an IPhone can ask their virtual assistant all about the nearest restaurants, upcoming appointments and the fastest way back home. Spotify has great suggestions to expand your music collection based on what you're currently listening and Tesla's autopilot keeps steering the car even when you remove your hands from the steering wheel.

Early artificial intelligence mainly focused on creating so called 'expert systems', in which the machine had access to large amounts of knowledge and rules provided by domain experts in order to solve specific problems (Hawkins & Dubinsky, 2016). One of the fields where the main advantages took place over the last sixty years, is that of machine learning (Smith, 2006). Where fifty years ago people worked on industrial robots, fully pre programmed, the current focus is on self learning robots that can work with humans and understand their intentions (Engels, 2013).

Field pioneer Tom M. Mitchel (1997, p.2) defines machine learning as follows: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." This can be accomplished through various learning algorithms, ranging from supervised to unsupervised learning. Where human beings have a vast amount of control in supervised learning, they have no influence on the learnings of a machine once it starts an unsupervised learning trajectory.

It is important to ask ourselves if unsupervised learning is what we, the users of this ever growing field of technological devices, actually want. Do we trust our devices to classify our behaviour without human intervention? A machine sorting various types of apples in a warehouse may use an unsupervised learning algorithm without any infringement of human autonomy, but what if these types of learning algorithms are used inside our houses?

Supervised learning puts control back into the hands of the teachers, the users in this case. But is that what we want then? For supervised learning, a training session is needed and one would not be able to buy a device and benefit from it immediately. An alternative is reinforcement learning, a learning algorithm that balances the input from the user and the initiative from the device. The device does not learn on it's own, but neither does it require a training session. It proposes a certain action and learns from the teachers reinforcement, that can be either positive or negative.

In this paper the use of reinforcement learning in household items is explored through the implementation of the algorithm in a lamp.

Reinforced learning

To find a learning algorithm that comes natural to people, the idea of teaching a dog came to mind. This project therefore started by exploring Skinner's theory of operant conditioning. For the implementation of operant conditioning in a device, Q-learning was recommended.

Operant conditioning explains how consequences lead to change in voluntary behaviour by using reinforcement and punishment. Reinforcement makes a behaviour more likely to be repeated, while punishment makes it less likely to be repeated (B. F. Skinner Foundation, 2016).

Q-learning is a reinforcement learning algorithm based on a relation between a state of the system and the possible actions that are available. All actions from a state are weighted in relation to how efficient they are at reaching a specific goal (Harmon & Harmon, 1996). Similarly to the behavior of a dog, behavior that was earlier punished in a certain situation is less likely to be repeated. While behavior that received praise is more likely to be repeated.

Why is the dog a good metaphor for a lamp like this?

The strong connection between dogs and people affects their familiarity with interacting with these animals. Most people have some experience with dogs, and have seen how they can be trained by disciplining or giving them

treats. Thus we consider it an appropriate image for our model of learning. As the goal is to make machine learning transparent, approachable and comprehensible, the familiarity of the dog training might be beneficial. The user will recognize that when unwanted behavior of occurs, the reason is generally a lack of training for the current situation, and that it can be resolved by reinforcing the behavior appropriately.

SKIP

Design
Skip is a smart lamp that is able to adjust its intensity of light and the position of his head. It learns how to behave in certain situations through receiving (negative and positive) feedback from its users. It relates the feedback from the user to certain states of its environment.

Skip's body is made up of multiple 4 mm MDF panels that fit each other like a puzzle. These MDF boards were cut by a laser cutter. All the electronical parts in the lamp are connected to an Arduino Uno microprocessor. It uses sensors to get input from the environment and actuators to respond to it. Skip is made out of three main parts:

The head
This part of the lamp is designed in a way that it is able to point upwards and downwards. When the head is pointing downwards it acts like a desk lamp and when the head is pointing upwards it creates ambient light by reflecting líght off nearby walls and ceiling. The movement of the head is controlled by a 5V analog servo motor that is able rotate 180 degrees. It also consist of 32 programmable RGB LEDs that are programmed to create four different types of light intensities: none, low, medium and high. Furthermore it has an LDR sensor which is placed on the frontside of the head, so that it measures the light of the surrounding environment.


Fig. 1 - SKIP, the physical prototype.

The stem
The stem is designed to cover up the wires and to hold the head of the lamp. For aesthetic reasons, the stem has the same width as the head of the lamp. The four vertically oriented plates of the stem all have a hole in which a PIR sensor is placed. These PIR sensors detect motion around the lamp.

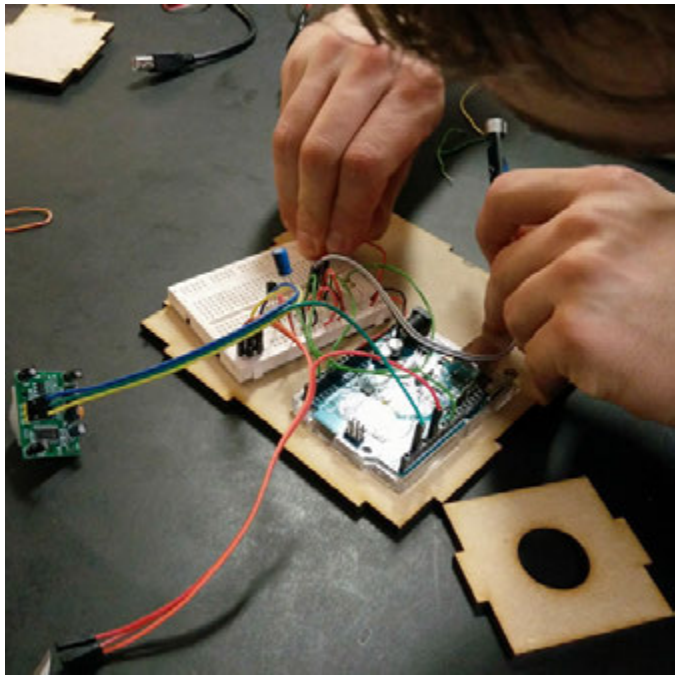

Fig. 2 - Implementation of the electronics in the lamp.

The base
The lower part of Skip is designed in a way that it would be able to fit an Arduino, a breadboard and the necessary electronics (e.g. sensors, capacitors, resistors) and in the back there is a hole for the power cable. The base harbors three sensors. The first is a temperature sensor that is able to measure the temperature of the environment. To be able to properly measure this, three side panels of the base are perforated. The second sensor placed in the base is a microphone sound sensor. It is placed with the microphone facing the front side of the lamp allowing the user to to give feedback with his or her voice.

Implementation of the learning algorithm
To implement the learning algorithm in the device, it needs to be programmed in the software of the device. On the internet, various examples of a programmed q-learning algorithm can be found. Most of the examples feature a mouse trying to find a piece of cheese that is placed in a certain place on a grid (Studywolf, 2012). The mouse (or robot in other examples) has a certain state, it is in a certain situation, and tries to reach a piece of cheese (or beer, as in the example of Kunuk Nykjaer, 2012) by trying various actions. If the action leads to the cheese, this action is reinforced. This not only occurs when an action leads directly to the reward, but also when it leads to a 'better' situation, closer to the reward. This is accomplished through backpropagation: once the reward is reached, the previous actions are reinforced as well.

Although Skip also learns through reinforcement and has multiple states in which various actions are possible, there are some differences between the concept of Skip and Q-learning. One of the differences is that in the Q-learning examples, such as that of the mouse, actions lead the mouse from one situation to another. Harmon and Harmon (1996) introduce Q-learning as a learning algorithm in which a given action in a given state leads to a successor state. This differs from Skip, where the state (the situation) is determined by the input sensors and is not influenced by actions from the device itself. It therefore cannot be 'on a right track' and backpropagation is not relevant.

A second difference is that Skip not only learns from direct reinforcement on a certain situation-action pair, whereas this is the case for Q-learning (Harmon & Harmon, 1996).

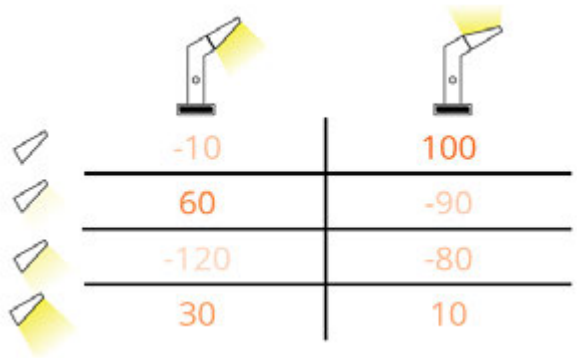

Fig. 3 - Each situation contains all possible actions that the lamp can take. Each action is weighted individually for the specific situation.

Based on the absolute difference in inputs, the reinforcement is partially transferred to the same actions in similar situations. This only occurs in situations close to the current state and the effect decreases with an increase in difference. By transferring the reinforcement, Skip learns faster than otherwise would be the case.

After exploring the various options to alter existing pieces of programming code to fit our purposes, we decided to write our own code instead. We used our knowledge of the learning algorithm and the inspiration we got from reviewing the online examples to program Skip in a manner that exactly aligns with our intentions.

Our code

The full code that defines Skips behaviour can be found in Appendix 1. In the following explanation, the numbers between brackets refer to specific parts of the code in the appendix.
[1] When the program starts, all possible situations are defined. In each situation, there are eight possible actions, since there are two variations in direction and four in brightness. Each situation-action pair has an initial value '0'.
[2] The program listens to the sensor information that the Arduino sends it through the Serial communication line. When the situation changes, this sensor information changes too. This triggers the program to change the action of the lamp, choosing the action with the highest value in the new situation.
[3] Aside from listening to the sensor information, the program also listens to hear when Skip is being reinforced. The moment this happens, the value of the current situation-action pair is altered. In case of a positive reinforcement it is raised by 100, in case of a negative reinforcement it is lowered by 100. The same action is also reinforced in similar situations by going through all possible situations while calculating the absolute difference from the current situation for the properties time, light, temperature and presence. The smaller the difference, the higher the value with which it is reinforced.
[4] When the lamp receives a negative reinforcement, a new action is triggered. This also happens when the action that received the negative reinforcement is still the action with the highest value for the current situation.
Although the lamp functions with solely the use of negative reinforcement, positive reinforcement was added to speed up the learning process and strengthen learned behaviour over time. A convenient side effect is that the possibility to positively reinforce Skip strengthens the metaphor, since praise is also used when training a dog.

Since the concept assumes the owner of the lamp knows the possible actions and his or her preference, it does not require explorative actions from the lamp to find an even better solution. The lamp will therefore immediately try the highest valued action in a certain state and only change

when it is negatively reinforced (or when the situation changes). When no positive reinforcement is used, over time this will result in not more than a minor difference in value between the preferred action and the other actions. When in case of an exception the lamp is used differently in the same situation, the new preference will get the highest value for the current situation and the preference for all the time before is 'forgotten'. This does not happen when positive reinforcement is introduced as well. One can positively reinforce the lamp as often as he or she wants, thereby raising the value and widening the gap between the current action and other actions in the same situation.

This way, over time Skip will truly learn one's preferences. It does not only immediately try the highest valued action for the current situation, but also take a smart second guess based on previous experiences when the proposed action is being negatively reinforced. Like a dog, Skip will retain the behavior it has learned before, even when on a special occasion one's preferences are different than usual. It will obey one's orders for this special occasion and the next day, when everything is back to normal, it will go back to the usual pattern.

Discussion

As we have not studied long-term use of the lamp it is hard to say exactly how it would adapt to fit real human lives. We chose four parameters for determining the situation in which the lamp is being used, but it is questionable whether the combination of these factors consistently represent a certain context in relation to lamp use. It might turn out that time of day, ambient light, temperature and presence are irrelevant measures. Maybe it would be more useful to determine who is present and what activity they are currently undertaking, however understanding such complex parameters are outside the scope of this project.

We must also question how voice recognition works as a means for interaction for desk lamps. While the dog metaphor might help making it a meaningful interaction, it is however worth considering if talking to a lamp might turn out to be even more awkward than talking to smart assistants like Amazon Echo or Apple's Siri can be at times.

For a simple application like this lamp, it would be reasonable to argue that the job of controlling the light would have been achieved just as efficiently by controlling it with a dial and a button. These interactions could even have been captured and fed to a unsupervised learning algorithm that could take over control of the lamp after a learning period.

This might have been a better approach from the perspective of creating a good model for the required light setting, however the question we are asking here is two-fold. Do arbitrary household objects need to be intelligent? Should the user be able to affect what and how our intelligent products learn? Neither of the questions are straightforward to answer, but we have proposed a design that takes an approach that does not rely on hidden capturing of user data but provides an understandable model for learning, while making sure that the reigns are in the hands of the user.

Conclusion

This project focused on creating a device to explore reinforcement learning algorithms in household items. During the project, a prototype of a learning lamp named Skip was created. Skip keeps the user in control while having the ability to act autonomously. Through voice recognition Skip's owner can communicate to the lamp what correct and incorrect behavior is for a certain situation. The lamp then uses a learning algorithm inspired by Q-learning to integrate the preferred action in its model for behavior.

Although Skip should be more extensively tested in a household environment to make stronger claims about the

added value, our initial experiences point in a positive direction. The transparency of reinforcement learning as implemented in this project gives the user a feeling of control and the analogy with the training of a dog makes it easy to understand and operate the device for a broad target group.

References

1. B. F. Skinner Foundation. (2016). The B. F. Skinner foundation. Retrieved from http://www.bfskinner.org/
2. Engels, J. (2013, January 24). Een goede robot hoeft niet slim te zijn. Trouw. Retrieved from http://www.trouw.nl/tr/nl/6700/Wetenschap/article/detail/3382326/2013/01/24/Een-goede-robot-hoeft-niet-slim-te-zijn.dhtml
3. Hawkins, J., & Dubinsky, D. (2016, March 24). What is machine intelligence vs. machine learning vs. deep learnign vs. artificial intelligence (AI). Retrieved from http://numenta.com/blog/2016/01/11/machine-intelligence-machine-learning-deep-learning-artificial-intelligence/
4. Mitchell, T. M. (1997). Machine learning. New York, NY: McGraw-Hill.
5. Nykjaer, K. (2012, January 14). Q-learning library example with C#. Retrieved from https://kunuk.wordpress.com/2012/01/14/q-learning-library-example-with-csharp/
6. Smith, C. (2006). The history of artificial intelligence (University of Washington). Retrieved from http://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf
7. Studywolf. (2012, November 25). Reinforcement learning part 1: Q-learning and exploration. Retrieved from https://studywolf.wordpress.com/2012/11/25/reinforcement-learning-q-learning-and-exploration/
8. Turing, A. M. (1950). Computing machinery and intelligence. Mind, 59, 433-460.
9. Harmon, M. E., & Harmon, S. S. (1996). Reinforcement learning: A tutorial. (Wright State University). Retrieved from http://www.cs.toronto.edu/~zemel/documents/411/rltutorial.pdf

Appendix

Here follows the part of our code that relates to Q-learning. We have left out the main class, the class handling serial communication and all of the Arduino code.

The code in full is available on https://github.com/CSivertsen/Learning-Lamp

```
/*
In the following code we are using the work state and situation different from the common convention sur-
rounding Reinforcement Learning.
Here a situation refers to the combination of environmental factors measured on the Arduino, which is more
commonly referred to as the state.
Furthermore the state refers to the combination of the lamp brightness and lamp head position, which could
be likened to the more commonly used action.
*/

class QLearning {

  // This arraylist contains all possible situations
  ArrayList<Situation> situations;

  // This variables contain the last received values from the Arduino
  int currentTime;
  int currentLight;
  int currentTemp;
  boolean currentPres;

  // ReinforcementVal is the value of which a state in being reinforced
  // either positively or negatively.
  int reinforcementVal = 100;
  // equalPoints are the number of points given for being equal for with the current situation
  // Used when calculating reinforcement for similar situations.
  int equalPoints = 2;

  // Declaring the Serial Interface for communicating with the Arduino
  SerialInterface mySI;

  // Holds information about the current state of the lamp
  State currentState;

  // Holds information about the current situation
  Situation currentSituation;
  // Temporarily holds a newly received situation update
  Situation incomingSituation;
```

```
// QLearning constructor
QLearning() {

  // Initializing Serial interface
  mySI = new SerialInterface(this);
  // Initializing default state and situations before anything is received from
  // the Arduino
  currentState = new State(0, 0);
  incomingSituation = new Situation(0, 0, 0, false);

  // A situation object is created for all possible combinations
  // of the 4 situation parameters and stored in an ArrayList
  situations = new ArrayList();

  //time
  for (int time = 0; time < 4; time++) {
    //light
    for (int light = 0; light < 3; light++) {
      //temp
      for (int temp = 0; temp < 3; temp++) {
        //presence
        for (int pres = 0; pres < 2; pres++) {

          boolean boolPres;
          if (pres == 0) {
            boolPres = true;
          } else {
            boolPres = false;
          }

          situations.add(new Situation(time, light, temp, boolPres));
        }
      }
    }
  }
}

// This function is called whenever a situation is received from the Arduino
void updateSituation(Situation incomingSituation) {

  // The incoming situation is compared to all situations in the situations ArrayList
  for (Situation situation : situations) {
    if (incomingSituation.time == situation.time &&
      incomingSituation.light == situation.light &&
      incomingSituation.temp == situation.temp &&
      incomingSituation.presence == situation.presence ) {

      // When the situation have been found that matches the incoming situation we check
      // that this situation isn't already the active situation. If it isn't we update
      // the current situation and check for the highest rated state for that situation.
      if (!situation.equals(currentSituation)) {
```

```
          currentSituation = situation;
          // Printing to console for debugging
          println("Updating situation");
          currentState = currentSituation.findBest(currentState);
        }
      }
    }
  }

  // Called when reinforcement happens
  // The parameter multiply is usually 1 or -1 to switch between positive and negative
  // reinforcement. Can also be used to weight the two types of reinforcements differently.
  void reinforce(float multiply) {

    // Printing to console for debugging
    println("Reinforcement: " + multiply );

    // This for-loop runs through all the possible situations and compares them to the current
    // situation. If a property is equal the value equalPoints is given, otherwise the difference in sub-
tracted.
    for (Situation situation : situations) {
      float timePoints = equalPoints - (abs(situation.time - currentSituation.time));
      float lightPoints = equalPoints - (abs(situation.light - currentSituation.light));
      float tempPoints = equalPoints - (abs(situation.temp - currentSituation.temp));

      // For presence equalPoints are given for being equal. Otherwise 0 points are given.
      float presPoints = 0;
      if (situation.presence == currentSituation.presence) {
        presPoints = equalPoints;
      }

      // The total amount of points are being calculated and the final score becomes a percentage of the re-
inforcementValue.
      // currentSituation where all properties are equal will thus be reinforcementValue * 1 and a situation
that is different
      // on all parameters will be reinforcementValue * 0.
      float totalPoints = (reinforcementVal/(equalPoints * 4)) * (timePoints + lightPoints + tempPoints +
presPoints) * multiply;

      // The current state for every situation is being reinforced with the calculated value
      situation.updateStateVal(totalPoints, currentState);

    }

    // If the reinforcement was negative, the best state for the current situation is being activated.
    if (multiply < 0){
      currentState = currentSituation.findBest(currentState);
    }
  }

  // Diplays a crude visualization of the lamp state to enable easier debugging and testing.
```

```
  void display() {

    fill(currentState.brightness);
    if (currentState.position == 0) {
      rect(0, height/2, width, height/2);
    } else {
      rect(0, 0, width, height/2);
    }
  }

  // Prints the weight for all states in the current situation to the console.
  void showPolicy() {
    println("Policy:");

    for ( int i = 0; i < currentSituation.states.length; i++) {
      println(currentSituation.states[i].weight);
    }
  }
}


class Situation {

  // All situation object contains an array of the possible states.
  final State stateA = new State(0, 0);
  final State stateB = new State(75, 0);
  final State stateC = new State(160, 0);
  final State stateD = new State(255, 0);
  final State stateE = new State(0, 1);
  final State stateF = new State(75, 1);
  final State stateG = new State(160, 1);
  final State stateH = new State(255, 1);

  State[] states;

  int time;
  int light;
  int temp;
  boolean presence;

  // The situation constructor requires values for the ambient parameters
  Situation (int _time, int _light, int _temp, boolean _presence) {
    time = _time;
    light = _light;
    temp = _temp;
    presence = _presence;

    states = new State[]{stateA, stateB, stateC, stateD, stateE, stateF, stateG, stateH};
  }

  // This function is called from the reinforce() function in the QLearning class.
  // The received val is the calculated points for a particular state in this situation.
  void updateStateVal(float val, State currentState) {
```

```
    // Looks up the right state in the state array
    for (State state : states) {
      if (currentState.brightness == state.brightness &&
        currentState.position == state.position) {

        // When the right state is found the calculated points are being added to the existing weight
        state.addVal(val);
      }
    }
  }

  // This function returns the currently highest weighted state for this situation
  // except if the highest rated state is the one that was passed to it. Then it
  // will return the second best. This is to avoid that the lamp does not seem to
  // respond when a highly weighted state in being negatively reinforced.
  State findBest(State currentState) {
    ArrayList<State> highValueStates = new ArrayList();
    float highestValue = -1000000;

    // The loop runs through the state array and finds the highest value
    // skipping the state that was passed to the function.
    for (State state : states) {

      if (!state.equals(currentState)) {

        if (state.getWeight() > highestValue) {
          highestValue = state.getWeight();
        }
      }
    }

    // The array is now looped through a second time to find all states
    // that has the same value as the highestValue. The most highest rated
    // state are put into an ArrayList.
    for (State state : states) {
      if (!state.equals(currentState)) {
        if (state.getWeight() == highestValue) {
          highValueStates.add(state);
        }
      }
    }

    // One of the highest rated value are chosen at random.
    // This is done to handle the case when several cases are
    // rated equally high.
    int pick = int(random(highValueStates.size()));

    // The function returns the highest rated state in the Situation
    return highValueStates.get(pick);
  }
}

// A State object holds a certain combination of lamp position and brightness
// as well as a how highly it is weighted.
class State {
```

```
  int brightness;
  int position;
  float weight;

// Constructor requires a brightness and position value. The default weight is 0.
State(int b, int p) {
  brightness = b;
  position = p;
  weight = 0;
}

// Used to change the weight when reinforcing
void addVal(float val){
  weight += val;
}

// Returns the current weight of the state
float getWeight(){
  return weight;
}

}
```

Personal Reflection, Christian Sivertsen - S167916

I have been following the application of machine learning and artificial intelligence in consumer products closely in the last few years. However I did not have any practical experience with the field before starting this course, and thus neural networks, reinforcement learning and other concepts could seem a bit "magical". I wanted to gain more knowledge about the inner working of intelligent systems to be able to critically evaluate its implementation in products and thus I chose to follow this course. This also aligns with my goals for developing my skills in the expertise areas technology and realization and math, data and computing.

Early on in the course, inspired by the two first lectures, I became interested in how intelligence could be used directly in the interaction with the product. Therefore I suggested to my group that we looked into this area, and after some back and forth we chose to base the interaction with our product on a real-life interaction with (somewhat) intelligent beings, namely dogs. We became interested in how humans interacting with dogs, could be used in modelling an interaction with an intelligent product. Based on this model we looked further into operant condition and reinforcement learning as a way of implementing this. Our subject for this interaction became a desk lamp.

In the group I spent most of my time implementing our reinforcement algorithm in Processing code. Early on we spent time on trying to understand and adapt examples on Q-learning that were available on the internet to fit our purpose. However after we had gained a throughout understanding of the concepts of Q-learning, we chose to write our code from scratch. This was mainly done in collaboration with Simone from the group. I implemented our new adapted algorithm, tested and tweaked it on several parameters so it would exhibit the behavior we wanted. Examples of such changes was reinforcement of similar situations, and experimenting with how the lamp would react immediately after being reinforced. Finally I made the serial communication interfaces that allowed the Processing program and our Arduino to exchange information, allowing us to test and demonstrate the product behavior live in class.

After following this course I have now gained an overview of several different approaches to intelligence and machine learning that allows me to consider these approaches for use in projects later on. More specifically I have gained practical knowledge with reinforcement learning, and its qualities and limitations when used in a household product such as our desk lamp.

Most importantly I have through the product offered a suggestion for how intelligent behavior in products can be made transparent for the user, rather being a black boxed. In a situation where we as consumers are constantly being measured and gauged to allow for adaptive behavior in products and services, I believe that it is important to consider how this process can be transparent and allow the user to have the ultimate say, even when products are behaving autonomously. This reflection is in line with my designerly vision of keeping users in control of the technological products that surround them. With the knowledge gained in this course I have better means for evaluating how different approaches to intelligence in interaction stimulate or limit the autonomy of the user.

Personal reflection, Simone Rietmeijer - s153538

It was with full commitment that I subscribed to the course Designing Intelligence in Interaction. The overview that the course was said to give strongly aligned with my need to learn more about this field of design. Thus far my learning path in Industrial Design has been far from regular and I feel like there are some gaps in my knowledge. A course offering an overview of intelligence in interaction was exactly what I thought I needed. Looking back now, I am very happy to realize that I was right. This course gave me the overview I was looking for and I now feel confident to continue my study with the knowledge of the possibilities of intelligence in interaction.

One of the features of this course that appealed to me, was the diversity of the lectures. With my background in Communication science, I could relate to the lecture of prof. dr. Rauterberg and this formed a pleasant bridge to the other lectures, which focused more on the technological aspects. The lecture that inspired me most was that on adaptivity in games. This probably has to do with my interest in serious games. Thus far, I did not have the chance to learn more about serious games in a project or elective, which made me appreciate the attention for it in this course even more. The lecture by dr. Van der Spek feels as a good starting point from which to overthink applications and implementations of and possibilities in games in future projects.

Another lecture that aligned with my interests was the lecture of dr. ir. Barakova. Last year I attended her course Intelligent Products and the use of learning algorithms intrigued me. Although I understood the basic principles, I could not completely grasp the translation to 'computer language' by the end of that bachelor course. During the first meeting with our team for DIII, I was happy to learn that my fellow team members preferred to work with the implementation of a learning algorithm. I decided to use this opportunity to expand my knowledge and learn how to make this translation to computer language.

From the start, our team worked together smoothly. We divided the project in two teams: one focusing on the construction of the lamp, one focusing on the implementation of the learning algorithm. Together with Christian I worked on the latter. We first tried to alter existing programs to fit our purposes, but at one point we realized that it might be better to write our own code from the start. For the most part, we did this together behind one computer. While Christian brought his expertise in programming, I contributed with my knowledge of the learning algorithm. Since I have some basic programming skills as well, it was easy to communicate and discuss the code while working on it. This way we could work in a highly efficient manner and I very much enjoyed it. I not only trained myself in programming learning algorithms, but, by watching Christian code, I also trained my skills in programming itself.

In future projects, I hope to implement learning algorithms again. Before taking this course I was not a big fan of intelligent products and systems. It felt scary and out of my reach to realize that all those devices gather personal data without me knowing what happens with it. This changed during the course. I now see the benefits of, for example, a learning system, especially when this is combined with user-friendly interfaces.

The way in which learning capabilities can discard the need to preprogram a device now appeals to me, especially when considering less technological literate target groups. A good example is our lamp. One could program a lamp to behave

according to ones preferences, for example with the use of a timer. Experience learns however, that for many people this is not as easy as it seems. And when one wants more parameters than just the time of day to influence the lamp and the lamp can have multiple states, this will become increasingly complicated. A learning algorithm like the one we used not only simplifies it for many users, I believe it can enable people to use an adaptable device who otherwise could not do so.

That being said, I strongly believe that it is important to make the intelligence transparent. People have the right to know when their behavior is being recorded. This does not have to be overly complicated and I believe it does not even need to be explicitly stated everywhere. Instead of a clear-cut message or warning sign, the project in this course made me realize that it could also be implemented in the interaction. When one teaches a device by praise and punishment, he or she implicitly knows this reinforcement is being recorded.

To conclude, I enjoyed taking this course. The overview that was given filled a gap in my knowledge and I feel I now have a strong starting point from which I can explore various ways of using intelligence in my designs. The course helped to shape my vision on intelligence in interaction and I look forward to use everything I learned in future projects!

Personal reflection, Mick Haegens - S165003

This was my first semester at the Tue. Before I started this master I completed the bachelor Entrepreneurship, Innovation and Technique which focuses mostly on the Business and Entrepreneurship competence area as known at the Industrial Design department of the TUe. One important reason for me to subscribe for the Designing Intelligence in Interaction course is because I decided to focus on technology during my master and because I wanted to learn about intelligence in design.

This course provided me with a clear overview of different theories related to intelligence in design. Among other things, I learned about intelligence optimization, information processing, intelligence in games, pattern recognition and neural networks. Besides all these theories that were addressed during the course, there were also practical lectures that showed me how to bring these theories to practice.

One of the lectures during this course that got my interest was that of dr. ir. Barakova about learning algorithms. The different learning algorithms (e.g. pattern recognition, classification) gave me insight in how these learning algorithms work. I thought it would be much harder for me to understand how these would work, but this lecture made me realize that I could be able to create a learning algorithm myself! To be able to do this I still have to work on my programming skills during the rest of my master, but it got me motivated to come up with a plan to improve on my programming skills.

Another lecture that I found interesting was that of Dr. Funk about the design of intelligent systems. Creating a system while working with the OOSCI library showed me the possibilities creating a smart system. I really enjoyed the exercises that we did during the lecture with the OOSCI library. The reasons mentioned above made me decided to ask Dr. Funk to become my new mentor, so that I could relate my upcoming research project to intelligent systems.

Because my non-industrial background and my lack of skills to make my ideas physical, I decided to mainly focus on building the physical prototype (together with Davide) during the project of this course.
David and I took responsibility for designing and creating the prototyping, where my main focus was on the electronics and writing the Arduino code. I learned how to work with different sensors and actuators and also how write a proper Arduino code. Also did I learn how to make a prototype by using a laser cutter.

Despite the fact that Simone and Christian were mainly responsible for the learning algorithm of our project, they made sure that Davide and me were closely involved with their process. They frequently updated us with their current progress, explained clearly each function of a specific part of the code and they constantly let us take part in the decision making process. This way I learned how I could program a learning algorithm.

To conclude, this course gave me a good overview of the possibilities of creating intelligence in design and it made me realize in what types of intelligence I am interested. Furthermore I learned how to make ideas physical and how I could program a learning algorithm!

Personal reflection, Davide Amorim - S162143

Few years ago, during the projects of my bachelor's programme I was designing objects through a process that was more concerning about aesthetics, materials, ergonomics or even technology available to produce those same objects. At the time, I wondered about how technologies are built into objects and how that influences the way people experiences products. About three years over, I enrolled in a project that enables me to understand how can I create, interact or implement intelligence in everyday things.

I believe that the learning path of a designer must be continuous, and in this course I was interested to explore and understand a new paradigm in product design, the needed to design for intelligent products. Although, before the start of this course my knowledge of intelligent objects was very limited, I must say that throughout the lectures I was able to enrich my knowledge in this field. As, I am a person more connected to a practical approach and the use of tools to explore my creativity, I would like to point out two lectures that I found it really interesting. Firstly, supporting the design of intelligent systems with OOCSI by dr. M. Funk and Pattern recognition and secondly, neural network by dr. J. Hu. However, in this group project we did not made use of these tools, I will definitely, be interested to explore them in future projects.

Concerning to the group project, we decided to built and intelligent lamp that could be trained as a dog. Therefore, we splitted in two groups, the first focused in the implementation of learning algorithm in the lamp and the second on the building of the model and electronics parts of the lamp. Even though, I have the interest to develop my programming and electronic skills, in this project my interest relied on the electronics part. Together with Mick, I was able to explore and learn about different sensors and actuators to use them as inputs to create a communication between user and the lamp. Although, me and Mick were not responsible for the implementation of the algorithm in the lamp, as the group was well structured and the communication was very clear, the two members of the group responsible for that part helped us to understand and let us know how the theory and practice behind the algorithm they were writing works. Therefore, I was able to learn the structure of an algorithm code.

With regard to the outcome learning of the whole project, it is my belief that if we create a more transparent communication between people and intelligent objects, there will be a better tendency for people accept them. I believe this project is a good starting point to explore new approaches towards user interaction with intelligent devices.

Overall, the course gave me access to a wide range of topics. From pattern recognition to adaptivity in games or even acquire knowledge of tools such as OOCSI library for systems synchronization or Neuroph studio to build neural networks. Although, I did not had the time to explore all the content this course could offer me I must say that I feel that all this information gathered along the theoretical and practical lectures will be very welcome to explore in future projects. Moreover, intelligent objects it is something that I definitely want explore in my future projects and the theories and tools this course gave me will be reflected in those future projects.
To finalize, if a few years ago my mind was wondering about how complex could it be for me to use advanced technology such as learning algorithms or pattern recognition in my design projects, this course came to substitute previous questions to answers.