UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: Smart Remote Control of Joysticks with Arduino

Trabajo de integración curricular presentado como requisito para la obtención
del título de Ingeniero en Tecnologías de la Información

Autor:
Hernández Rodríguez Marcos Antonio

Tutor:
Francisco Javier Hidrobo Torres, Dr.

Urcuquí, febrero de 2023

AUTORÍA

Yo, **Hernández Rodríguez Marcos Antonio**, con cédula de identidad 1804727194, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad del autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, febrero de 2023.

Marcos Antonio Hernández Rodríguez
CI: 1804727194

AUTORIZACIÓN DE PUBLICACIÓN

Yo, **Hernández Rodríguez Marcos Antonio**, con cédula de identidad 1804727194, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior

Urcuquí, febrero de 2023.

_____

Marcos Antonio Hernández Rodríguez
CI: 1804727194

Dedication

Este proyecto está dedicado a mis padres quienes han sido un pilar fundamental en mi carrera y me han apoyado enormemente.

Marcos Antonio Hernández Rodriguez

Acknowledgments

Quiero expresar mi más profundo agradecimiento a mis padres que me han apoyado en el proceso universitario y que siempre estuvieron para mi. A mi hermano Luis que me ayudó mucho con este proyecto.
A mi tutor Oscar Chang quien me ha brindado su confianza y por haber compartido sus conocimientos durante la carrera.
A mis amigos los cuales me llena de alegría haber compartido tiempo con ellos, varios recuerdos y experiencias que vivimos dentro del campus los llevo en el corazón.

Marcos Antonio Hernández Rodriguez

Resumen

Hoy la tecnología se ha convertido en un factor esencial en diferentes campos de la ciencia. El proceso de automatización requiere un alto nivel de precisión para que sus resultados sean sobresalientes en comparación con el control manual. Dispositivos como Arduino han despertado el interés de muchos investigadores, ya que facilitan tareas que van desde las más complejas a las específicas, como detectores de sonido, sensores de movimiento, hasta transformar una casa inteligente. Por otro lado, la inteligencia artificial se ha convertido en un potencial importante en la investigación ya que imita la capacidad del ser humano. Este proyecto se centra en combinar la inteligencia artificial y el microcontrolador Arduino construyendo un sistema de control remoto para un brazo robótico que opere en procesos críticos como el manejo seguro de maquinaria a gran escala. Para la comunicación de estos dispositivos se utilizó módulos como Wi-Fi 2.5GHz debido a su bajo costo. El aprendizaje por refuerzo (RL) establece al brazo robótico como un agente para explorar sus posibles movimientos y replicarlos para que su movimiento sea suave mientras se mantiene el equilibrio. El propósito de este proyecto es mejorar la seguridad operativa ya que se ven afectados por interferencias y ruidos. Finalmente, se compararon los resultados al ejecutar dichas tareas con trabajo manual y así determinar su eficiencia.

Palabras Clave: Inteligencia artificial, Aprendizaje por Refuerzo, Arduino, Conexión Inalámbrica.

Abstract

Today technology has become an essential factor in different fields of science. The automation process requires a high level of precision, so its results are outstanding compared to manual control. Devices like Arduino have piqued the interest of many researchers, as they facilitate tasks ranging from complex to specific ones, such as sound detectors, and motion sensors, to transforming an intelligent house. On the other hand, artificial intelligence has become an essential potential in research since it imitates the capacity of human beings. This project focuses on combining artificial intelligence and Arduino microcontroller to build a remote control system for a robotic arm that operates in potential critical pro- cesses such as large-scale machinery that has to be handled safely. 2.5GHz Wi-Fi modules were used to communicate these devices due to their low cost. Reinforcement Learning (RL) establishes the robotic arm as an agent for exploring their possible movements and replicating them to smooth their movement while maintaining a complex trajectory. This project aims to improve the operational safety of devices with several degrees of freedom that are affected by interference and noise. Finally, the results will be compared when executing such tasks with manual work and thus determine their efficiency.

Key Words: Artificial intelligence, Reinforcement Learning, Arduino, Wireless connection.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

There exists a great interest in the industry to improve handling of remote objects. There are fields such as medicine, electronics, robotics, etc, that needs a precise handling. For example, in medicine the using of a scalpel requires a crucial handling for any medical operation. In robotics, simulating an arm or a hand in a robot drastically depends on correct handling for best results. Therefore using Artificial Intelligence (AI) plays a big role in modern robotics, especially the biological oriented models, based upon Artificial Neural Networks (ANN) and self-taught agents [3] robots can perform tasks autonomously, but not only in terms of the action itself, but also to be able to make decisions about what to do in a given work environment under pre-established parameters. AI can accelerate processes to achieve perfection in production, since they are much more than just machines that perform the most arduous, boring or dangerous tasks.

This project aims to develop a device able to handle distant objects using radio communication integrated with AI making its decisions for accurate results. The combination of these fields creates a tremendous interest for scientists. Since the learning of a robotic and the interaction with the real world depends on past events, it helps to predict future movements. Nowadays, many games use AI because an AI player represents the human response, which has accumulated knowledge based on past events. Games such as Chess, Go, or Tic-Tac-Toe has been highly studied for creating learning algorithms [4]. Since these games have many possibilities, a perfect method must be crucial to optimize all the possible movements to get a victory.

Tic-Tac-Toe is a game played on three-in-a-row boards that can be traced back to ancient Egypt [5]. It consists of two players, X and O, who mark the spaces in a 3×3 grid. The winner is who places his/her marks in a horizontal, vertical, or diagonal row. Game playing agent could be used for a general learning strategy; this agent is an explorer to search for a possible sequence of moves that lead toward capturing rewards. These rewards depend if the agent winning the game. When a process depends on a sequence of moves distributed in discrete-time n, wherein each move a partial reward r(n) is captured, its overall

optimization is defined by the Bellman equation, which establishes the mathematical foundation of Reinforcement learning and Q-learning [6]. Reinforcement learning is based on an agent repeatedly interacting with the environment and learning rewards resulting from a good and poor decisions. It takes on a problem defined by the Markov decision process [7].

In the real world, humans discover by themselves apertures or movements to obtain a better result which implies better reward. For example, in chess, players can predict movements to obtain a win. They develop the capacity to look to the future. Therefore, the Bellman equation establishes that the optimization of control to obtain a maximal reward in a sequential chain of state-decision is solved by a summation of immediate events and events in the future. This process involves a self-taught, reinforcement learning neural agent to develop a player that can explore all the possible movements for winning and getting the process to get the maximal reward in the sequence states. As mentioned, creating a device that can manipulate or handle objects implies a perfect reaction. Implement a joystick module that can control itself is the principal objective of this project.

The joystick modules are mentioned in the project's topic but need to be fully covered. Since the project progressed, it focused on reinforcement learning and constructing the reward and Q matrix. First, it is essential to clarify that the joystick module work. The joysticks control the robot wirelessly, which is explained later in the methodology. The idea was to drive the robot and create a noisy environment so that the connectivity between the radio frequency modules loses communication data and not be able to control the joystick modules. That is when the robot can continue the complex sequence until reaching its destination objective. But for time reasons, the results achieved were obtaining the Q matrix and auto-filling the R matrix.

1.2 Problem statement

Lately, the development of automated systems capable of performing rigorous operations has benefited many work areas. Many of these jobs required personnel maintaining constant control of the machinery to avoid possible failures. An automated system can reduce human intervention to specific jobs; hardware and software work together to replicate a person's work in the most similar or even better way [8].

Kinesthetic teaching manually moves a robot as a person intervenes to record future movements. This type of learning is generally effective in small robots due to various factors such as weight, balance, etcetera. However, with large-scale robots, the result is inadequate [7]. Although many autonomous systems can multi-tasking, they may still need to replace human tasks. Because there are certain circumstances, such as the environment in which said machinery is exposed. These can disable proper function and give unexpected results. That is why reinforcement learning is a method that is based on trying to replicate a particular task through failure and error. Since it is managed by agents who can learn the sequence of steps through negative and positive rewards, once trained, it will be able to find the right path with efficient results [6].

This project features a robotic arm driven by a reinforcement learning agent. The arm comprises five servomotors, four of which are related to the robot's degrees of freedom. The goal is to remotely move the arm in a complex way, from one point to another, by using a simple joystick. During training, the agent explores possible positions or states and uses its sensor to capture rewards and punishment. The agent must learn by exploration and rewards capturing how to move along a complex trajectory that represents a straight line along a rigid wire. Though accomplish this trajectory, the 6DOF has to be changed simultaneously and coordinately, which is the task that the agent must learn. By using its physical sensors and exploration, the robot can fill the reward matrix R and then use this information and the Bellman equation to find a matrix Q that represents an optimal policy in terms of state-decision.

The idea of a robot learning to perform a specific task without explicitly telling it what to do is a hot topic in contemporary robotics [9]. Mapping the steps or replicating sequences from sensors to actuators can be highly computationally costly since the programmers must create the robot's operation step by step. Debugging these programs can be a tedious task for programmers. Then creating an agent learns intuitively to obtain suitable results [10].

To solve robotic´s problems is essential to talk about sensors. These can receive operational and functional data to transmit all collected information to a control unit such as a microcontroller. Combining these two tools makes it possible to create a complex device capable of performing rigorous tasks based on the environment. In some cases, AI can perform better than human behavior.

1.3 Objectives

1.3.1 General Objective

- Create a robot arm capable of learning automatically through reinforcement learning.

1.3.2 Specific Objectives

- Design a robotic arm with servomotors.
- Implement an algorithm capable of moving the arm with joysticks and a wireless connection in Arduino Uno.
- Study the Q-learning technique and its adaptability to real world environments.
- Demonstrate that the arm can learn by itself to move in a complex trajectory and then move in this trajectory following an external command.
- Create a system of rewards and punishment based on external Arduino sensors so that the robot learns to move by itself using a RL agent.

1.4 Contributions

This research aims to develop an AI in an microcontroller that senses its surroundings through sensors and responds to natural changes. Some problems will be tackled in the development of the project, and the essential contributions will be solved, as defined in the following paragraph: the implementation of AI through the application and adaptation of learning techniques to produce an optimal solution to the problem. Performance is often achieved with the aid of standard libraries and the use of existing functions, reducing the complexity of the code, helping to concentrate on the primary purpose, and reducing the complexity for interested developers and researchers to understand and evaluate the steps of the software.

The contributions of this project are to develop a relatively complex environment for a robot. Furthermore, agent scanning facilitates filling the reward matrix since this matrix is filled dynamically in many cases. As a result, the agent can find rewards corresponding to his actions. Another contribution is an efficient algorithm for reinforcement learning in state action. This project opens the door to self-learning in robots.

Chapter 2

Technical Framework

2.1 Microcontroller

A microcontroller is a small computer made up of a single chip whose primary function is to control integrated computers. Many of those have integrated circuits that are necessary to perform their task [11]. A microcontroller is equipped with a CPU, a memory system, an input/output system, a clock or timing system, and a bus system to interconnect systems. They are commonly used on Arduino boards and have three types of memory. Static Random Access Memory (SRAM) is usually used for local variables. The Electrically Erasable Programmable Read-Only Memory (EEPROM) is the memory space to save values after the board is turned off. The program or sketch is stored in the Flash memory, which is a non-volatile memory [1].

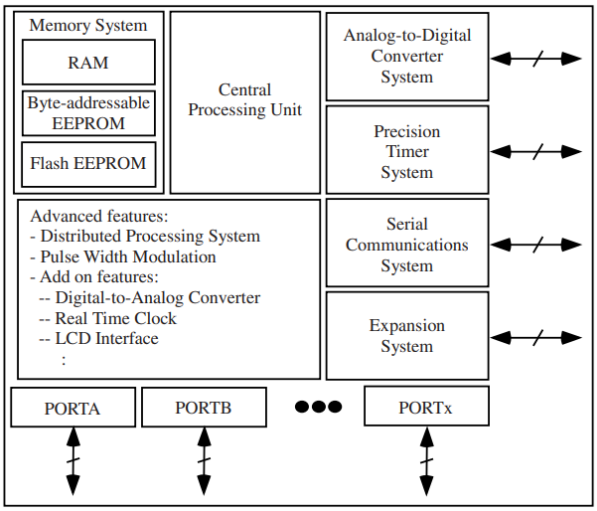

Figure 2.1: Microcontroller block diagram [1]

2.2 Arduino

The Arduino boards are electronic boards based on microcontrollers. It is an open-source code that can be easily programmed, erased, and reprogrammed at any time. It is similar to a mini-computer. The board allows the development of a wide range of projects due to electronics devices[12]. Based on the Atmega328 microcontroller (datasheet), Arduino Uno provides specific characteristics such as fourteen input and output pins. Six of these pins are used for performance, and the other six for simple information. It has a 16 MHz clay resonator, a power connector, and a reset button. Its storage is 32kb in flash memory, 2kb in RAM, and 1kb in EEPROM. Its programming language is high-level, like C++ or C [13].

2.3 HC-SR04 Ultrasonic Distance Sensor

The HC-SR04 sensor is a low-cost distance sensor that uses ultrasound to determine the distance of an object in a range of 2 to 450 cm. The HC-SR04 sensor has two transducers: a piezoelectric transmitter and a receiver, as well as the electronics necessary for its operation. The operation of the sensor is as follows: the piezoelectric emitter emits 8 ultrasound pulses (40KHz) after receiving the command on the TRIG pin, the sound waves travel in the air and bounce when encountering an object, the piezoelectric receiver detects the bouncing sound, then the ECHO pin changes to High (5V) for a time equal to the time it took for the wave since it was emitted until it was detected. The microcontroller measures the time of the ECO pulse, and the distance can be calculated to the object [14].

Figure 2.2: Ultrasonic Distance Sensor

2.4 Servomotor

A servomotor is an electronic motor designed for the movement of gears. It comprises three pins: power supply, ground (GND), and signal. It has metal gears for better performance and more strength than other classic servomotors such as Sg90. It produces a rotation shift from 0 degrees to 180 degrees. The MG90S servomotor has a torque of 2.20 Kg and a speed of 0.1 seconds at 60 degrees at 4.8V of power supply [15]. These servos are very important for handling small-scale robotics due to their incredible ease of operation. Among some advantages, they can work at high speeds if the appropriate weight is placed and constantly work at the same pace.

2.5 nrF24L01 Transceiver

This device allows wirelessly establishing connections that send sensor data, controlling robot data, home automation, and various projects in real time. Its radio frequency works in the 2.4 GHz worldwide ISM frequency band and uses GFSK modulation for data transmission. The data transfer rate can be one of 250kbps, 1Mbps, and 2Mbps.

The module operates at 3.3 V as a power supply. The range of addresses that can work is 125. An advantage is that it can connect to up to six other modules, making it possible to have multiple wireless units communicating with each other in a particular area. Some modules work with duck antennas, and RFX2401C chip, which improve the communication range up to 1000m [16].

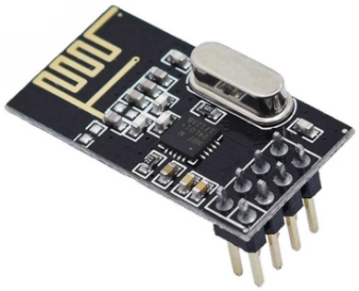

Figure 2.3: nrF24L01 Transceiver

2.6 Joystick Module

A joystick module is an electronic device that allows you to control 2 x axis y axis directions because it has 2 independent potentiometers. This module has 5 pins which are voltage pin (Vcc), ground pin (GND), x axis pin (VRX), y axis pin (VRY) and a push button (SW). Some characteristics of this module is its self-central positioning, its light weight and it is compatible with many microcontrollers and various Arduino models[17]

Technical Specifications:

- Operating Voltage: 5V
- Internal Potentiometer value: 10k
- 2.54mm pin interface leads
- Dimensions: 1.57 in x 1.02 in x 1.26 in (4.0 cm x 2.6 cm x 3.2 cm)
- Operating temperature: 0 to 70 °C

2.7 Electrical Power Supply

Arduino boards need electric power to work like batteries, USB cable, AC adapter, or power supplies. USB port provides 5V DC, it can be a source from PC. AC sockets are used for Arduino which need 7 to 12V [18]. Battery sockets for batteries that supply 3.7 V. VIN is another way to supply, it has to be connected to VIN pin correctly. In this project, 2 types or power supplies are used, one is USB port for Arduino functionality, joystick, modules radio frequency and the second is a switching power supply for all servo motors.

Chapter 3

Theoretical Framework

3.1 Reinforcement Learning (RL)

Reinforcement learning is a machine learning technique that uses an agent capable of interacting with the environment. It performs its task through actions and observations to obtain a better result. It uses a feedback of its errors[19].

This algorithm does not need labeled data or monitoring as it is characterized by trial and error machine learning. Rewards give their primary function in response to their actions [20]. Reinforcement learning has been successfully applied to solving the reaching task with robotic arms. Reinforcement learning has shown great promise in robotics thanks to its ability to develop efficient robotic control procedures through self-training.

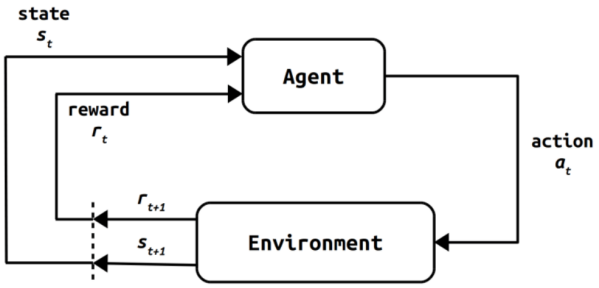

Figure 3.1: Generalized Reinforcement Learning scheme

3.1.1 Agent

An agent is an autonomous program capable of interacting with the environment. It determines the current state and analyzes the next move based on an action. The agent's main objective is to make the best decision through repetitive learning from its mistakes. The observed data is based on sensors excluding external problems that cannot be captured by signals [21]. The agent will increase its rewards based on several episodes of evading states not desired.

3.1.2 Reward

The agent's goal is to move so that it chooses the best path without making a mistake. Every decision it makes will be rewarded. These rewards (Rt) could be negative or penalty if the new state is not the desired one, positive if its movement approaches the determined goal. Agent's mission is to maximize a real-valued reward signal [22]. Accordingly, the reward is the main factor for updating the policy, defining an event's good or bad.

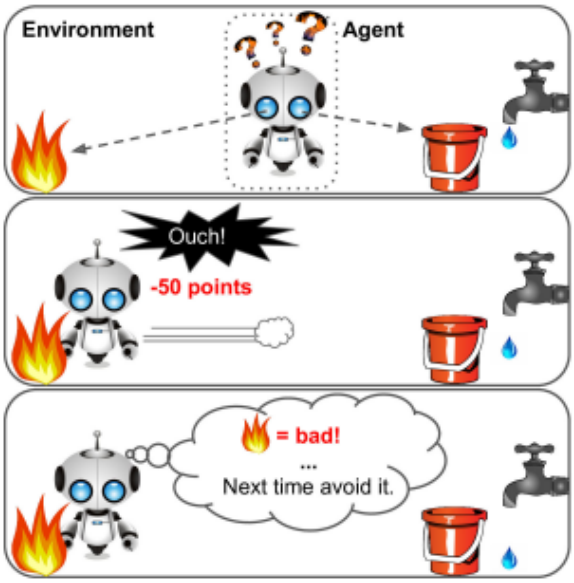

Figure 3.2: Reward based on its action [2]

3.1.3 Environment

The environment is the world where the agent is involved. An agent can explore its environment through movements determined by actions [23]. When the agent acts, it will receive a new state of the environment. One example to represent this idea is Mario, a Nintendo game, where Mario is the agent and the environment is the world. Every action he takes will involve rendering new frames as the new state.

3.1.4 State

The state (S) describes the current situation. For a robot, it could be the exact position of its two legs. For a vehicle to learn the best way, its state would be the exact position where it is located. Each state is within the state space, which is responsible for storing all the possible situations the agent can do. That is why the state must contain relevant information for the agent to choose the best move [24].

$$S_t = f(H_t)$$

Where S is obtained by a function of H. H is defined by the whole process.

3.1.5 Policy

A policy defines the action the agent should choose when in a given situation. A policy is what an agent can do in each state. RL tries to determine the optimal control policy that can be used as a strategy to optimize a criterion in specific processes [22].
The policies can be deterministic

$$a = \pi(s)$$

or stochastic

$$\pi(a|s) = P[A = a|S = s]$$

In real or virtual world scenarios, to avoid predictable and linear results, the policy provides based on the probability of possible actions since the decisions are the purpose of working with probabilities. Encouraging the agent to start a random exploration will improve the behavior and give rewards.

3.1.6 Value Function

$$v_\pi(s) = E_t[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ...|S_t = s]$$

3.1.7 Model

Although a model is not the same as the environment, it can estimate or forecast what the environment will do next. The model aids in the development of a strategy for achieving a mission. P predicts the next state in the model, while R predicts the next reward [23].

$$P^a_{ss'} = P[S_{t+1} = s'|S_t = s, A_t = a$$

$$R^a_s = E[R_{t+1}|S_t = s, A_t = a$$

3.2 Model-based

A Model-Based(MB) is a convenient option for maximizing functionality in various problems, such as complex control systems, to solve a complex task through abstraction, as Kramer said [25].

3.3 Model-free

In contrast to MB, a Model-Free(MF) approach avoids constructing a model and instead learns directly from experiences with the environment [26], without prior guidance or knowledge about the problem.

3.4 Exploration Exploitation

The learning process is more complex than it seems in the real world. RL process is based on trial and error; exploring the environment could be carried out in detail. However, it will not learn or explore new possibilities, interfering with discovering an optimal policy. A random scan can carry out the problem. To select an action, the agent chooses a random option. This random exploration method allows us to discover its environment to obtain the best reward. The goal is for the agent to be confident in his movements due to his repetitions. The exploration involves finding more information about the environment; the exploitation exploits the known information maximizing the reward.

3.5 Markov Process

Most of the problems related to the exploration of the environments using a robot can be solved by Markov Decision Process (MDP). Markov establishes that a current state State (S) contains all useful information from history. This equation implicates that the future is independent of the past given the present actions [27]. The probability of the following state is conditioned by the current state, which contains all relevant information from history.

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, ..., S_t]$$

3.5.1 State Transition

The state transition probability is defined by where s is the current state and s' is the successor

$$P_{ss'} = P[S_{t+1} = s'|S_t = s]$$

Meanwhile Matrix P is the transition probabilities from every state to every successor states.

$$P = from \begin{bmatrix} P_{11} & ... & P_{1n} \\ ... & ... & ... \\ P_{n1} & ... & P_{nn} \end{bmatrix}$$

3.5.2 Bellman Equation

The Bellman equation is an equation used for many Reinforcement Learning algorithms. The value function is decomposed into two pieces by the Bellman equation: immediate reward and discounted future values [28].

$$Q(s, a) = r + \gamma max_{a'} Q(s', a')$$

3.6 R-Matrix

The R-matrix is a table that contains the rewards for all possible states and all possible actions. This information is the combination of possible movements in the environment.

Every decision an agent makes will have a reward; it could be a punishment or a positive reward. The following problem describes the R-matrix in a simple way [29].

Suppose there are five rooms in a building; each room is connected by a door to another room, as we can observe in the graph. There is an agent in any room, and its goal is to move to room five (outside room). Some rooms only have one door to enter another room. Rewards will be assigned based on your decisions, and these values will be put on the matrix. If there is no connection between the two rooms, the reward will be -1. If the rooms are connected, the reward will be 0; finally, the rooms connected to room five will be assigned 100 as a reward.

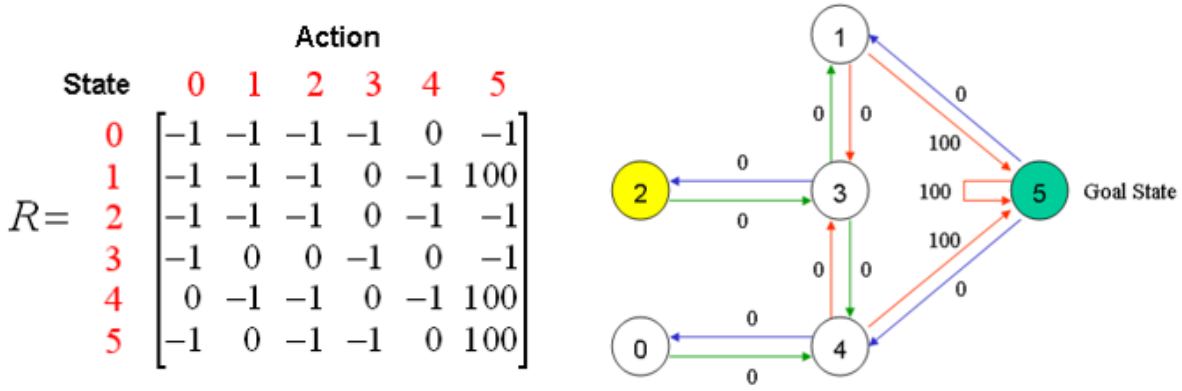

Figure 3.3: Reward Matrix and Graph Problem

3.7 Q-learning

It is off-policy learning of action values Q(s, a). It selects the next action, which is chosen using behavior policy. However, it is considered an alternative successor. Then Q(St, At) is updated towards the value of alternative action. Where γ and α are hyperparameters.

$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma maxQ(S',a') - Q(S_tA))$$

The Q-Learning algorithm is described in Algorithm 1

3.7.1 Q-Matrix

The Q-matrix is a look-up table that helps to store and calculate the maximum expected future reward an agent will get given the pair Action and State. Columns are the possible actions, and rows are the possible states for Q-table [30]. The table will be updated. Its values will increase or decrease iterative during the Q-learning process. Figure 3.4 summarizes the information flow in the Q-table during the Q-learning. The inputs are States and Actions to produce a Q-value to continue updating the table.

Algorithm 1: Fill R matrix

```
Set the gamma parameter, and environment rewards in matrix R;
Initialize matrix Q to zero;
for each episode: do
    Select a random initial state;
    while the goal state hasn't been reached do
        Select one among all possible actions for the current state.
        Using this possible action, consider going to the next state
        Get maximum Q value for this next state based on all possible actions
        Compute: Q(state, action) = R(state, action) + Gamma * Max[Q(next
         state, all actions)]
        Set the next state as the current state
End
```

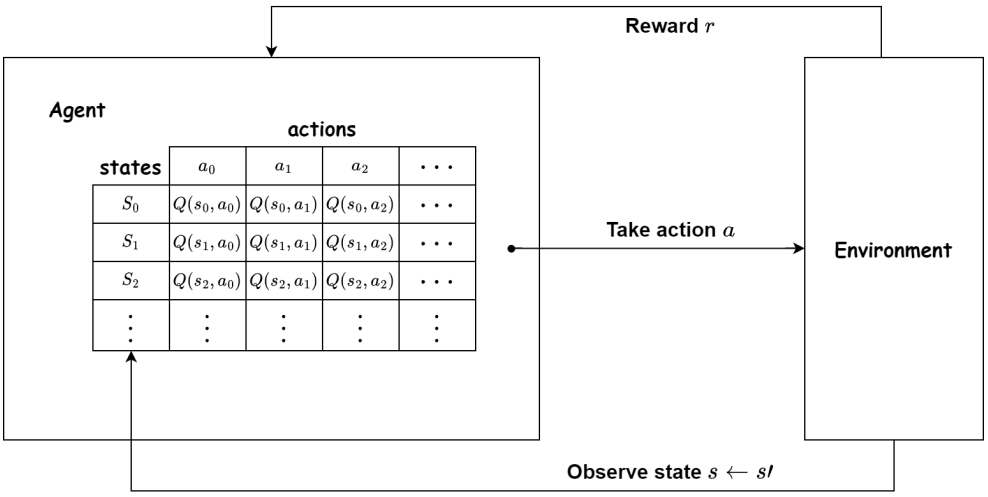

Figure 3.4: Q-table representation

Chapter 4

State of the Art

This section will evaluate and review the related work to this project. There is an overview of a robotic arm that is digitally simulated for the training process with Unity and Tensorflow.

The work in *"A robot arm digital twin utilising reinforcement learning"* [31] provides a simulation of a robot arm that will learn in a virtual environment. The authors use the previous training, and they will replicate it in a physical environment. The simulation is made in Unity. The primary purpose of this paper is to create a method that teaches a digital twin to learn in a virtual environment with the necessary architecture and learning protocols.

The authors use the game engine Unity to simulate their robotic arm since Unity allows them to use a machine learning toolkit to create a training agent that interacts with the virtual environment called agents [32]. Unity does not compute or run the neural network training. All the data is sent to Tensorflow, which controls the training process. The authors use a Python API which handles the robot's action decisions and collects all observations from its environment.

For the design of the robot, its components are printed on a 3D machine. In this case, the authors use three stepper motors; these are NEMA23, NEMA17, and NEMA14 which are supplied to 24V. Arduino boards with an external driver handle the movements of each one. It has a camera as a sensor. Creating an agent involves reinforcement learning; the agent's interaction with the virtual environment is based on movements. The set of states and actions are required to create the Q-table [33]. Markov reward procedures and a discount factor obtain the results. For each state at a given step, the value to be updated will be the sum of its future rewards given to policy. Once the agent collects the rewards, his last process will be to collect the maximum rewards in the fewest steps.

For the experimental setup, the authors pose the virtual robot with six platforms around it. Unity ml-agent helps to detect objects for the training phase. The agent's task is to move a piece of a specific color to the open position. The new position is limited by the number of angles each servo can move. For simulating the camera, a virtual ray is used to perceive

the objects within their environment. All of these features serve as input parameters for training. The results obtained during training suggest that each step is treated in an extolled way. It helps in the intuition process. The combination of virtual simulation with the real environment was necessary for the robot arm to learn a specific task. For future implementation, the authors suggest creating a system that allows carrying out complex tasks, increasing the number of objects in the environment. In addition, handling hyperparameters more rigorously would bring better results.

Chapter 5

Methodology

This project proposes a system based on a robotic arm, which learns by Q-learning the sequence of movement from an initial point to another desired point, following a complex trajectory that resembles a straight line. To achieve this goal, the robot has to learn to move its four degrees of freedom in a coordinated way, just as a human learns to coordinate their hand muscle to write words.

This prototype is based on three essential points: a detailed description of the transmitting device that involves the joystick, a description of the receiving device that involves the robotic arm with its environment, and a description of the training process.

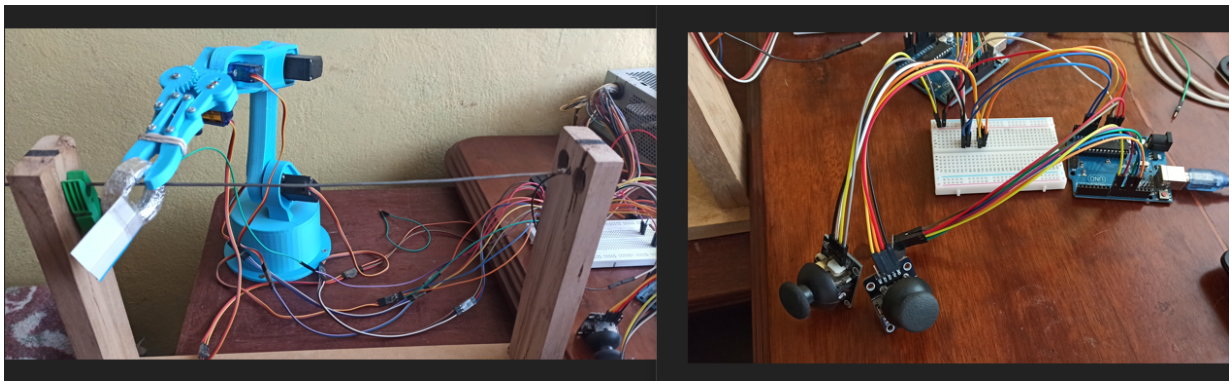

Figure 5.1: A full view of prototype

5.1 Phases of Problem Solving

5.1.1 Description of the Problem

The structure of the arm is designed by the servomotors, which control its movements or device A. Two joysticks control the servomotors' movements, which are constructed in device B. The management of the arm could be abrupt since the joysticks modules send a position based on their analog reading. Therefore, moving from the initial position to

another could be challenging to operate. Using a Q-learning method could solve the route for a complex situation.

The arm route has to be previously trained for a method involving rewards and punishments. All this information is obtained by interacting with the arm and its environment. This environment consists of the movement of a metal ring that surrounds a metal wire. An ultrasonic sensor measures the distance to the right corner. The figure below shows the environment for the training process. First, the agent has to obtain a reward or punishment based on its movements. The arm holds the metal ring and moves it from one point to another. If the metallic ring touches the metal wire, a closed circuit is produced, and the agent will be punished for its action. Meanwhile, if the distance to the sonic sensor is reduced and sensed, the agent obtains a positive reward for its good action. The process repeats until Q-matrix is filled with a near-optimal policy. One of the big problems is filling the reward matrix. In many cases, the reward matrix is filled manually. However, this process becomes problematic when the number of states and actions increases immeasurably. This project proposes an algorithm that automatically explores the environment and obtains the rewards values.

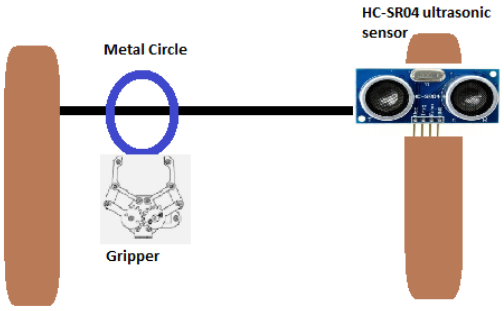

Figure 5.2: Simplified environment design

5.1.2 Analysis of the Problem

One of the big problems in the prototype is the unexpected movements generated by connecting the radio frequency modules. Because the code establishes a connection between them, it continuously reads the data sent by the other module. Another problem in the prototype is the unexpected cases that can cause the real environment surrounding them. Since the training process requires repetitions, there may be failures in the movement of the arm, such as cable disconnect, interference between modules, or an unexpected move of a servomotor.

5.1.3 Implementation

The development of this prototype was divided into several stages: the design of the robotic arm and assembling, the circuit for the connection between two modules and controlling

the robot by joystick modules using Arduino, and finally implementation of the learning code by reinforcement.

Design of Robot Arm

The arm's design was based on a general model of six degrees of freedom representing the servomotors' movements and one extra for the gripper. A 3D printer created each part of the arm. The material is made of filament PLA, polylactic acid, which is a thermoplastic. Once all the pieces are printed, the following step is assembling the arm. Four servomotors were used for the movements. Three of them were MG95 servomotors because this servomotor model has more strength and torque. And two of the rest were sg90 servomotor. Each servomotor is powered by a power supply that provides 5v, able to move everything without a problem. Once the servomotors are assembled with the printed parts, the arm is ready to be connected to Arduino.

Figure 5.3: Robot Arm

Connection and Controlling

Two joystick modules are used to control the arm. Each one handles two servomotors because the joystick handles the X and Y axes. For the gripper handling, one button opens the gripper, and the other button closes it. This control is managed by wireless communication between the two Arduino boards, for which two NRF24L01 transfer modules were used shown in Figure 5.4. The values read by the joystick were captured and sent to the other module for their respective movement.

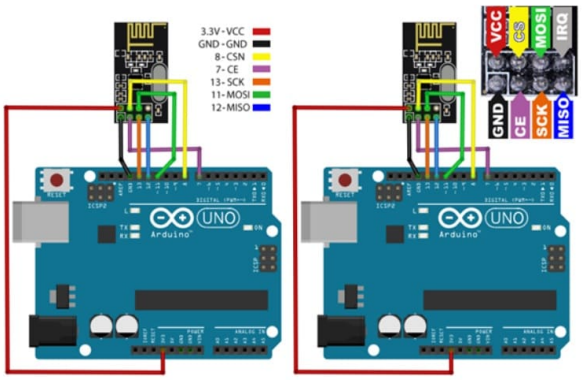

Figure 5.4: Connection for 2 nrf24l01 modules

Learning process in Arduino

This section describes the development of the algorithm that controls agent learning. The arm is considered an agent. The agent is initialized in a starting position or position 0. Filling the R-matrix, the possible number of decisions is analyzed, and each servomotor can take three actions: to move left, right, or not. The states are the combination of four servos, which results in the length of rows in the R matrix. This calculation is explained below. Servo.h library controls the movements of all servos. For the reward matrix, the states are explored. A negative reward is obtained if it touches the wire, and a positive reward if it approaches the goal distance. Figure 5.5 shows the learning process flowchart in a brief way. The creation of the reward matrix depends on its rows representing the states, and the columns are the actions. The values are updated based on the sensors which assign the rewards. The states are calculated by combining the four servomotors representing S1 for the base, S2 for the shoulder, S3 for the elbow, and S4 for the wrist.

- S1 has positions from 0 to 180, where each value corresponds to a degree. The minimum step is 1; S1 can assume 180 different states.
- S2 has positions between 40 to 100; the minimum step is 1; therefore, it can assume 60 different states.
- S3 has positions between 40 to 100, with 60 usable states.
- Finally, the S4 has limited positions between 50 to 80, with 30 possible states.

The combination of 180x60x60x30 results in a total of 19440000 possible states that are the rows in the R matrix. This number is huge to be processed. The proposed solution is to reduce the states to demonstrate the self-learning agent's exploration principle. The new states are represented as follows:

To construct decisions, the agent must take several actions to move to the next state. There are many ways to implement this,each servomotor can explore its environment by changing its axis values. In a current state, it can make three decisions: -1 if its movement is to the left or down, 0 if there is no movement, and +1 if it is to the right or up. For the

movement of the four servomotors, the total of possible decisions is 12. Therefore our R-matrix dimensions are 1250 rows and 12 columns. A principal contribution of this project is to develop an algorithm so that the robot is capable of filling the R matrix by itself using exploration and the feedback obtained from its sensors (feedback oriented learning).

The objective of the robotic arm is to learn to move in a straight line that presents a complex trajectory. Once we have R-matrix proceed to fill Q-matrix using the Bellman equation, this process must use a Q-learning algorithm and sensors to control its steps in the environment. The agent must find an optimal trajectory in real situations.
The Q matrix has the same dimension as the reward matrix. This training is evaluated in a K number of episodes until the Q-matrix obtains confident results. The γ parameter has a range of 0 to 1. When γ is closer to zero, the agent considers immediate rewards. If γ is closer to one, the agent will consider future rewards with greater weight. The Q-matrix will explore the actions without negative rewards (-1), avoiding the short circuit.

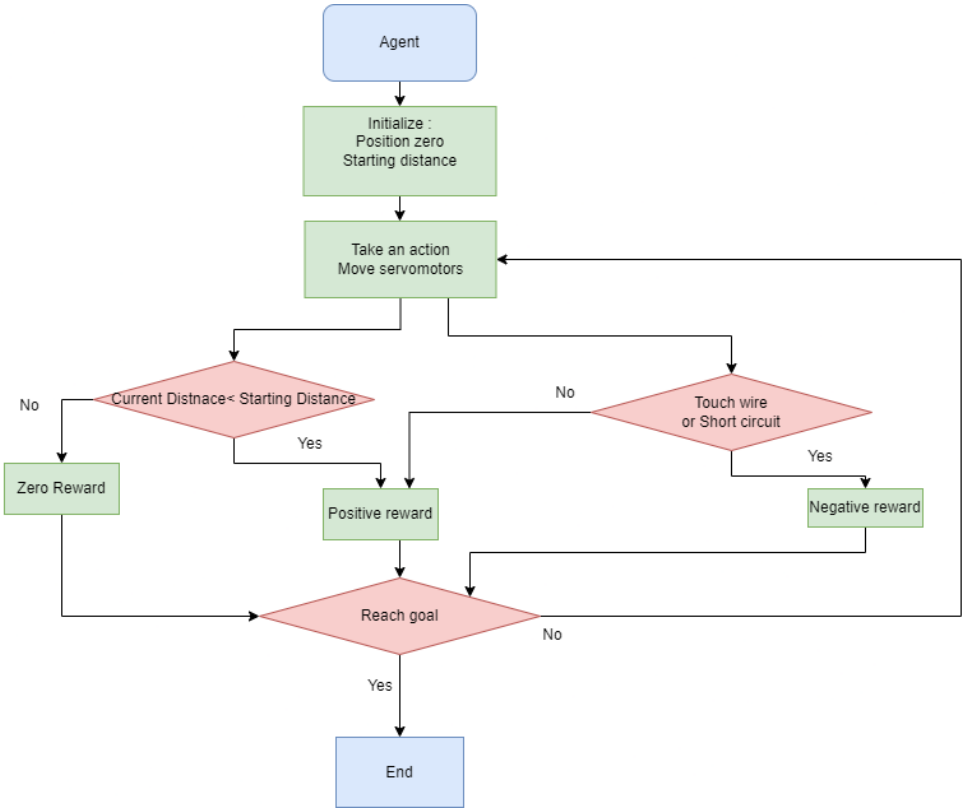

Figure 5.5: Learning process flowchart

5.2 System description

Figure 5.6 shows the whole system designed in a block diagram. It is represented by Devices A, and Device B. Device A is controlled by an Arduino board. It is connected to two joystick modules and one nRF24L01 module. Another Arduino controls device B.

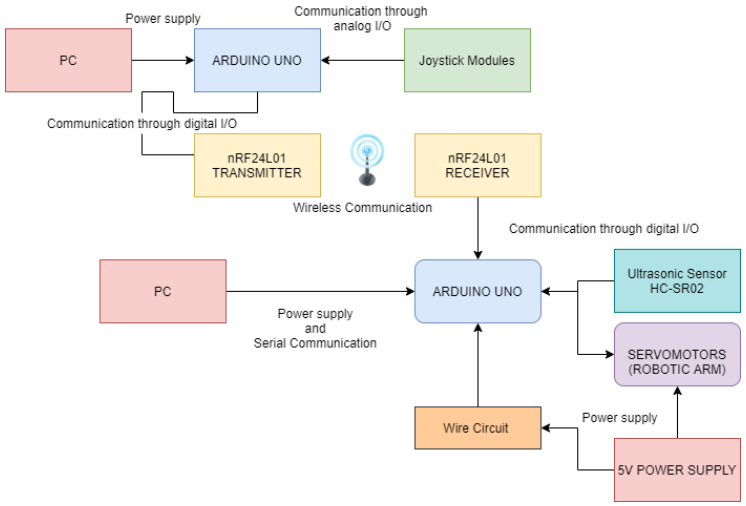

Figure 5.6: Block diagram prototype

It is connected to one nRF24L01 module, five servomotors representing a robotic arm, an HC-SR04 ultrasonic sensor, a metal wire representing a close circuit, and finally, a power supply of 5V for servomotors.

5.2.1 Task of Device A

Device A,shown in the appendix section, is responsible for collecting the information obtained by reading the analog values of the joystick modules. For this, analog pins 0-3 are used. The values obtained are between 0-1023. These values will serve to move each servomotor in device B. To send the values, a wireless connection must be established through pipes with the RF24.h library. This process is in the setup function. The loop function constantly analyzes if the connection exists. When the connection is established, it will write the joystick data packed in a structure, and it will be sent to the other module.

5.2.2 Task of Device B

Device B,shown in the appendix section, unpacks the data previously sent from the transmitter module. The process is divided into the setup function and the loop function. SETUP function initializes the corresponding variables and opens a pipe for the wireless connection.

In the LOOP function, there is a menu that is sent by the console to perform specific tasks. One drives the robot through the joystick modules, so it constantly reads the data transmitted from the other Arduino board. There is the option of training to explore the environment and its possible states. Each action performed obtains a reward that is saved in the reward matrix. This process involves the ultrasonic module, which captures the distance between the module and the metal circle. It is also constantly reading the closed circuit used for punishment in case the metal circle touches the wire. Finally, the matrix Q is updated using the Bellman equation. These values will be saved in the EPROM memory

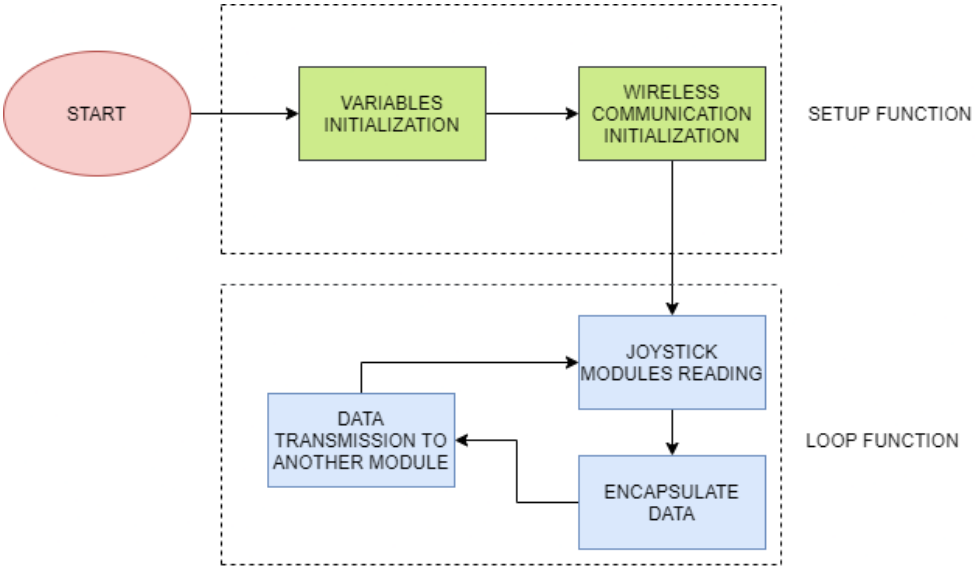

Figure 5.7: Flow process of the Device A in Arduino UNO

of the Arduino.

5.3 Code Structure

The learning process is presented in the following pseudocode. The agent will be exploring all possible states based on its action. The reward R matrix is initialized with -1 values and the Q-matrix with zero values. Generally, a -1 value means an invalid place where the robot should not move. These values are going to be updated by exploration in both matrices in n episodes.

R matrix is crucial for this project because it is quite time and energy-consuming to fill all the values of this matrix by hand by a human. The first contribution of this project is to develop an environment and an agent by which the robot explores the environment, uses its sensors, and fills the R matrix by itself. This opens the door for self-learning in robots.

Since the dimension of the R matrix is relatively short, it has to explore the movement and its action to set the rewards values and change the original -1 values, improving its knowledge about moving into its allowed space. Here the ultrasonic sensor and the metal wire sensor are the keys to accomplishing this learning stage. Initialize the variables, these are ***initialDistance, currentDistance,goalDistance***. The agent starts in a position in the middle of the trajectory. Then the agent must make a random decision for his next move. The sensors are constantly reading their values. If the ***currentDistance*** is greater than the ***initialDistance***, it will not obtain a reward, therefore 0 will be assigned to the entry in the R matrix. If the ***currentDistance*** is less than the ***initialDistance***, it will obtain a reward of 1 and its ***initialDistance*** will be updated. If the agent touches the wire it will produce a short circuit and receive a punishment or negative reward of -1.

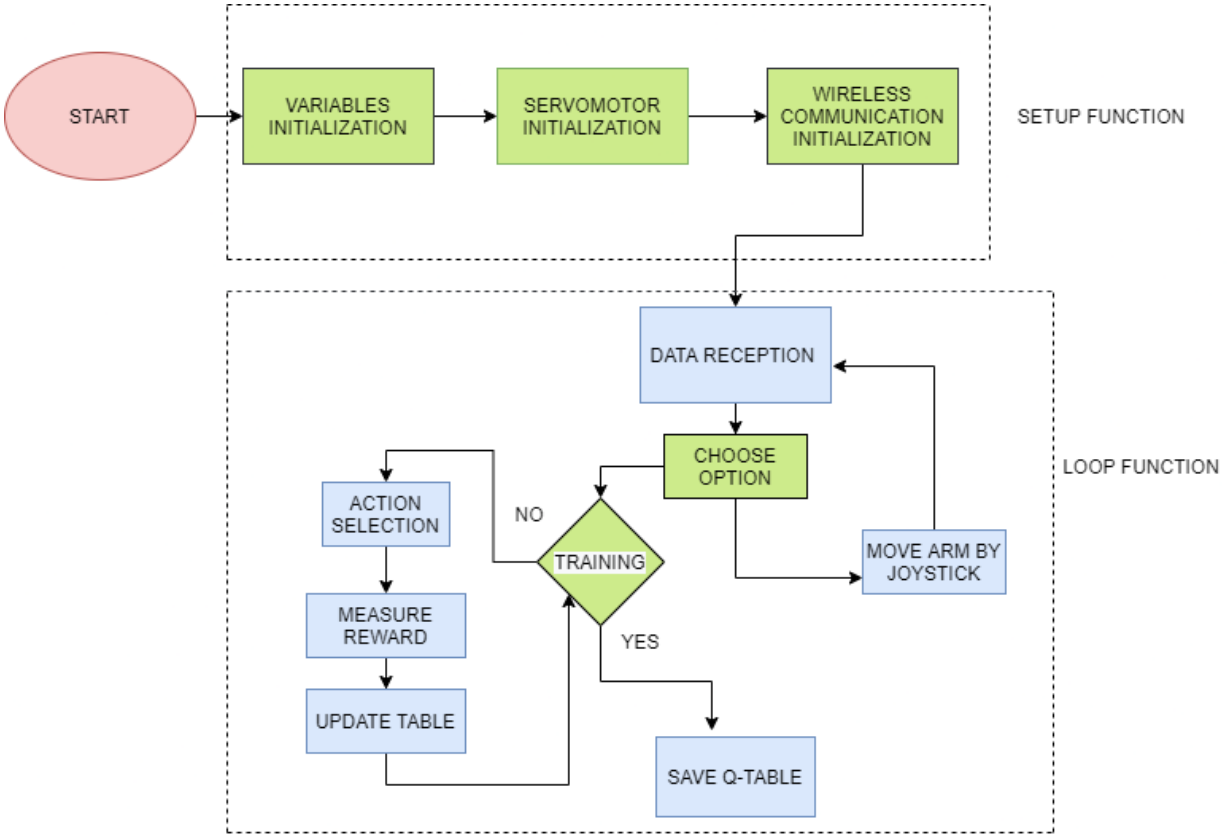

Figure 5.8: Flow process of the Device B in Arduino UNO

Once R matrix is filled the Q-learning begins the ***currentDistance***, agent is set to work by using the Bellman equation where the maximal reward of 100 is located in R position where the arm moves close to the ultrasonic sensor, below a prefixed threshold ***distanceGoal*** the agent obtains a reward of 100. Doing for all rows with no continuous short circuit.

Then the training randomly selects x and y positions, representing states and actions in Q-matrix [x][y]. In the following loop process, the number of iterations is limited by the possible steps that the agent can move. StepSelection selects the neighborhood value in the Rm and action as an input. After this sequence, the agent has to be updated depending on the previous move. The state variable saves the current environment, then Qm is updated using the reward function where action and state are input parameters.

Finally, the function ReachGoalDistance is called to evaluate if the metal circle arrives at the established distance goal. This process iterates until K episodes are done. The output is a Q-table matrix that will be safe in EPROM memory.

Algorithm 2: Fill R matrix

```
Initialize Rm to zero values;
Initialize state_0,row_0 position;
Initialize initialDistance,currentDistance,goalDistance;
for i = 0 to ROWS do
    TakeRandomDecision();
    ReadSensors();
    if currentDistance > initialDistance then
        Rm[i][action] = 0;
    if currentDistance < initialDistance then
        Rm[i][action] = 1;
        initialDistance = currentDistance;
    if currentDistance <= goalDistance then
        Rm[i][action] = 100;
    if shortCircuit() == TRUE then
        Rm[i][action] = -1;
Ouput: R-table matrix
```

Algorithm 3: Training Agent

```
1  State = Environment variables;
2  Initialize Q to zeros values;
3  RewardMatrixFill();
4  K,moves number of episodes and movements respectively;
5  for i = 0 to K do
6      x,y = Random();
7      Agent = Rm[x][y];
8      for j = 0 to moves do
9          action = takeAction();
10         step_x,step_y = StepSelection(action);
11         Agent = Rm[x+step_x][y+step_y];
12         State' = CurrentEnviroment();
13         Q[x][y] = Reward(action,state);
14         if ReachGoalDistance() == TRUE then
15             Break;
16         else
17             Continue
18 Output: Q-table matrix
```

Chapter 6

Results and Discussion

In order to obtain the best experience and results, it is important to identify the number of iterations so that the agent can move intelligently. As an expected result, the agent can fill the reward matrix (R-matrix) that will fill the Q-matrix. The filling R matrix takes an average of 1250 trials. It should be noted that the number of possible states is reduced by the wire inside the ring system, making it possible for Arduino to process the environment and the resulting R matrix.

To fill the Q-matrix, the estimated number of episodes is around 6000. Although their displacements are relatively short, the servos support the movements to carry out their task. Figure 6.1 shows a small portion of the R-matrix. As we can see, there is a large number of negative rewards (-1). The three upper servos can explain this. It depends on a vertical movement, generating more possibility of a short circuit.

Due to the low resolution generated by sensors and local noise generated by the positions of servos, the resultant Q-matrix is noisy. However, this matrix allows the robotic arm to follow a good straight trajectory under simple joystick control.

$$\begin{bmatrix} 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 100 \\ 0 & -1 & -1 & 0 & 1 & -1 & 0 & -1 & 0 & 0 & 100 & 100 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & -1 & -1 & 0 & 1 & 0 & -1 & -1 & 1 & 0 & 100 & 100 \end{bmatrix}$$

Figure 6.1: R matrix reduced, rows represent states and columns are decisions

It is possible to create an environment with Arduino components such as joystick modules, radio frequency modules, modules to detect distance, and the defined short circuit. All these components together were of great help to the 6DOF arm agent to explore a complex environment and learn to move the robot tip following a complex trajectory that resembles a straight line.

The robot can fill the R matrix by itself through exploration and receiving punishment and rewards from the environment using the appropriate algorithm.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 90 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 70 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 51 & 80 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

Figure 6.2: Reduced view of Q-matrix result

After using the Bellman equation for Q-learning, the robot learns a policy that allows it to follow a quasi-optimal path under the command of a remote joystick. Figure 6.3 shows the behavior of the self-taught robot arm interacting with the environment. The robot initializes its positions and moves by following the commands of one joystick, moving it in a proper straight-line sequence.

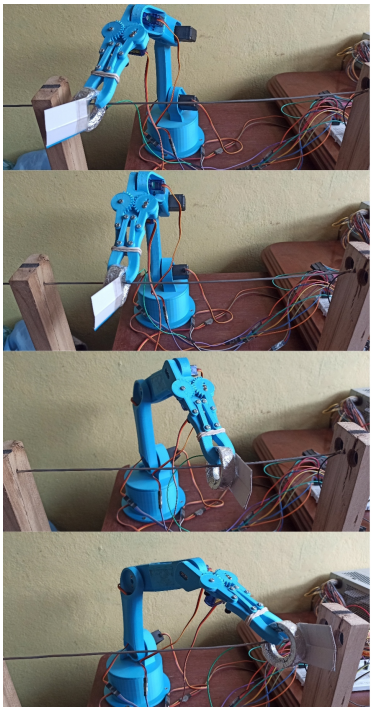

Figure 6.3: Real-world behavior of the robot moving in a straight line

This result demonstrates that the agent can learn by itself to drive the robot so that it follows a complex trajectory controlled by a single joystick. It is the same principle used in self-driven cars. As future work refining the sensors and making the Q matrix bigger, the agent can learn more complex tasks, like handling a surgical scalpel. Create the phase training on another device, and the data captured by Arduino will be sent from the serial port. This possibility provides excellent scope for the current project. For example, integrate a camera so the robot can identify objects.

In Figure 6.4, the cumulative reward x-axis represents the number of episodes trained, while the y-values represent the total absolute value of each variable corresponding to the Q table. The graph demonstrates that the agent is learning, gaining experience after each iteration from the first to episode one hundred. Figure 6.4 plots a summary of all the learning work. This curve shows decent overall growth, and for the last seventy iterations, the agent has had no noticeable increase compared to previous episodes. This behavior comes from the random aspect of the learning process.

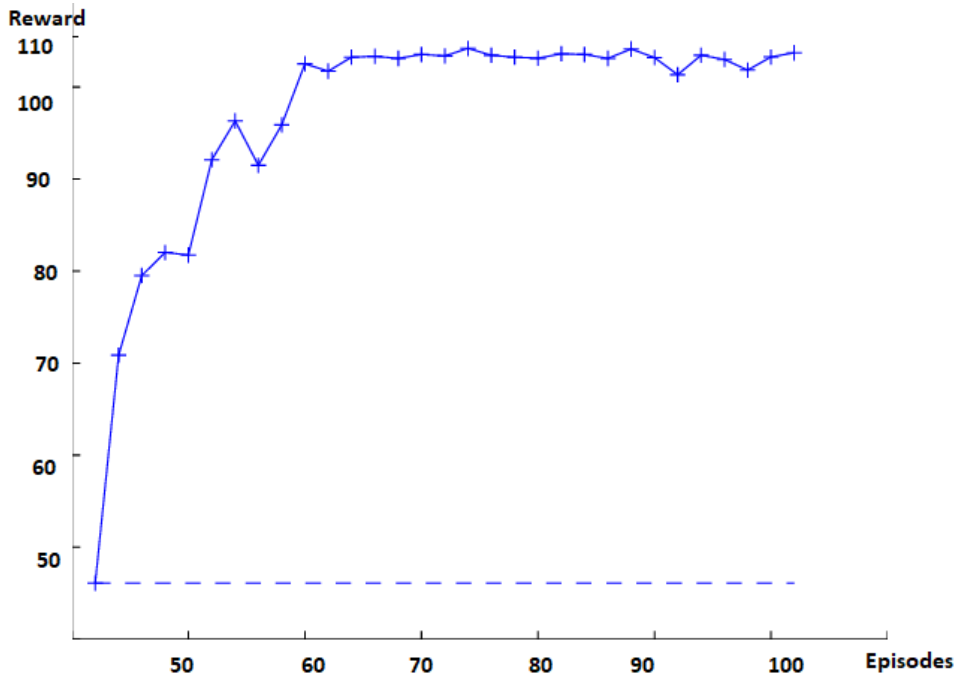

Figure 6.4: Cumulative reward as a function of the number of episodes

Chapter 7

Conclusions

The proposed method shows that it is possible to create an intelligent device to automate robot tasks using artificial intelligence, especially RL. Since the reward system helps hugely in learning.

Within this project's scope, the results are promising even when the computation was carried out in microcontrollers (Arduino Uno). However, in a real environment, there are a large number of variables that can alter the expected result. Therefore, more sensors capable of receiving information would be needed so that the agent would act based on these variables and thus can obtain a better result.

During the progress of this project, there are several points that can be included:

- The Arduino Uno's microcontroller can operate learning methods. However, if the number of variables was to increase significantly, it would open up complications. A possible solution would be to send the data from the sensors through the serial port so that a PC can process the processing.
- The wireless connection between the Arduino allows us to send and receive data almost in real time, demonstrating that communication in open spaces is possible.
- Reinforcement learning is an effective method for robots because agents must interact with the environment and learn from their decisions.
- The development of an algorithm so that the robot can fill the matrix R by itself using the exploration and the feedback obtained from its sensors. For this reason, this method provides an optimal solution if the number of states and actions increases.
- Arduino has become a handy tool for reading signals emitted by components integrated into the Arduino. When connecting with the internet, some components give a great field of research with IoT.

Bibliography

[1] Y. Güven, E. Coşgun, S. Kocaoğlu, H. Gezici, and E. Yılmazlar, "Understanding the concept of microcontroller based systems to choose the best hardware for applications." 2017.

[2] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2019.

[3] M. Mishra and M. Srivastava, "A view of artificial neural network," in *2014 International Conference on Advances in Engineering Technology Research (ICAETR - 2014)*, 2014, pp. 1–3.

[4] S. Karamchandani, P. Gandhi, O. Pawar, and S. Pawaskar, "A simple algorithm for designing an artificial intelligence based tic tac toe game," in *2015 International Conference on Pervasive Computing (ICPC)*, 2015, pp. 1–4.

[5] X. Song, "Experience generation in tic-tac-toe for general game learning," in *2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, 2012, pp. 1–5.

[6] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *International conference on machine learning.* PMLR, 2017, pp. 1146–1155.

[7] P. Kormushev, S. Calinon, and D. G. Caldwell, "Reinforcement learning in robotics: Applications and real-world challenges," *Robotics*, vol. 2, no. 3, pp. 122–148, 2013.

[8] D. A. Norman, "The 'problem'with automation: inappropriate feedback and interaction, not 'over-automation'," *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, vol. 327, no. 1241, pp. 585–593, 1990.

[9] W. Smart and L. Pack Kaelbling, "Effective reinforcement learning for mobile robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 4, 2002, pp. 3404–3410 vol.4.

[10] T. Tamei, T. Matsubara, A. Rai, and T. Shibata, "Reinforcement learning of clothing assistance with a dual-arm robot," in *2011 11th IEEE-RAS International Conference on Humanoid Robots.* IEEE, 2011, pp. 733–738.

[11] D. Ibrahim, *Microcontroller based applied digital control.* John Wiley & Sons, 2006.

[12] L. Louis, "Working principle of arduino and using it as a tool for study," *International Journal of Control, Automation, Control, Communications and Systems*, vol. 1.

[13] N. David, A. Chima, A. Ugochukwu, and E. Obinna, "Design of a home automation system using arduino," *International Journal of Scientific & Engineering Research*, vol. 6, no. 6, pp. 795–801, 2015.

[14] E. J. Morgan, "Hc-sr04 ultrasonic sensor," 2014.

[15] M. Török, T. Erdei, S. Tóth, and G. Husi, "Designing and building a remote-controlled 3d printed prototype robot arm implant," in *IOP Conference Series: Materials Science and Engineering*, vol. 1169, no. 1. IOP Publishing, 2021, p. 012038.

[16] M. Mahbub, "Design and implementation of multipurpose radio controller unit using nrf24l01 wireless transceiver module and arduino as mcu," *International Journal of Digital Information and Wireless Communications*, vol. 9, no. 2, pp. 61–73, 2019.

[17] "Joystick Module." [Online]. Available: https://components101.com/modules/joystick-module

[18] "Arduino Official Store | Boards Shields Kits Accessories." [Online]. Available: https://store.arduino.cc/

[19] F. Xhafa, S. Patnaik, and M. Tavana, *Advances in Intelligent Systems and Interactive Applications: Proceedings of the 4th International Conference on Intelligent, Interactive Systems and Applications (IISA2019)*. Springer Nature, 2019, vol. 1084.

[20] R. S. Sutton, "Introduction: The challenge of reinforcement learning," in *Reinforcement Learning*. Springer, 1992, pp. 1–3.

[21] O. Chang, L. Zhinin-Vera, and F. Quinga-Socasi, "Self-taught neural agents in clever game playing," in *Proceedings of the Future Technologies Conference*. Springer, 2020, pp. 512–524.

[22] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[23] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in multi-agent systems and applications-1*, pp. 183–221, 2010.

[24] J. Hu, "Reinforcement learning explained," Dec. 2016. [Online]. Available: https://www.oreilly.com/radar/reinforcement-learning-explained/

[25] A. Kramer and B. Legeard, *Model-based testing essentials-guide to the ISTQB certified model-based tester: foundation level*. John Wiley & Sons, 2016.

[26] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[27] A. McCallum, D. Freitag, and F. C. Pereira, "Maximum entropy markov models for information extraction and segmentation." in *Icml*, vol. 17, no. 2000, 2000, pp. 591–598.

[28] J. TORRES.AI, "The Bellman Equation," Sep. 2021. [Online]. Available: https://towardsdatascience.com/the-bellman-equation-59258a0d3fa7

[29] Y. Hirashima, Y. Iiguni, A. Inoue, and S. Masuda, "Q-learning algorithm using an adaptive-sized q-table," in *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, vol. 2. IEEE, 1999, pp. 1599–1604.

[30] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation," *Robotics and Autonomous Systems*, vol. 112, pp. 72–83, 2019.

[31] M. Matulis and C. Harvey, "A robot arm digital twin utilising reinforcement learning," *Computers & Graphics*, vol. 95, pp. 106–114, 2021.

[32] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar *et al.*, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, 2018.

[33] M. Bortolini, M. Faccio, F. G. Galizia, M. Gamberi, and F. Pilati, "Adaptive automation assembly systems in the industry 4.0 era: a reference framework and full–scale prototype," *Applied Sciences*, vol. 11, no. 3, p. 1256, 2021.

Appendices

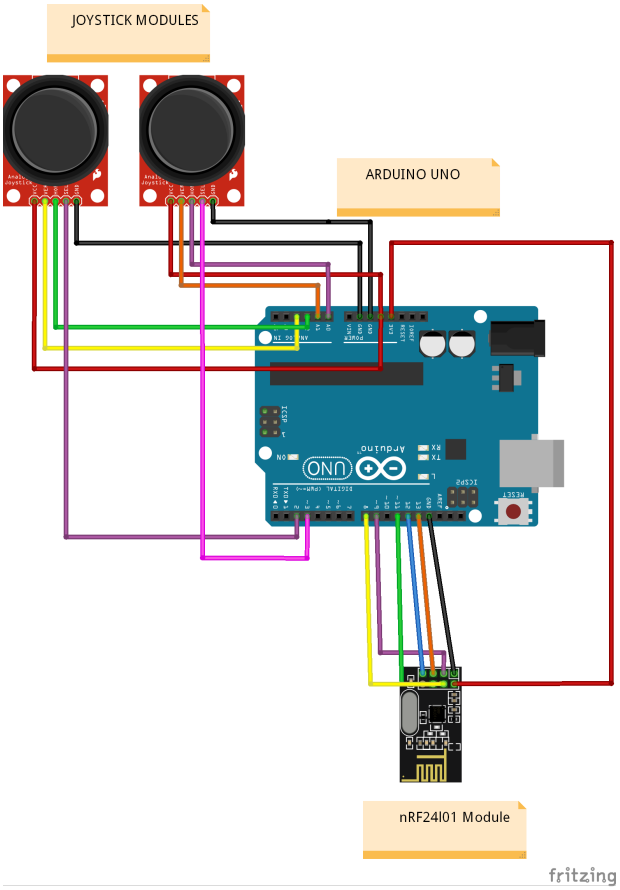

Figure 1: Device A wiring diagram

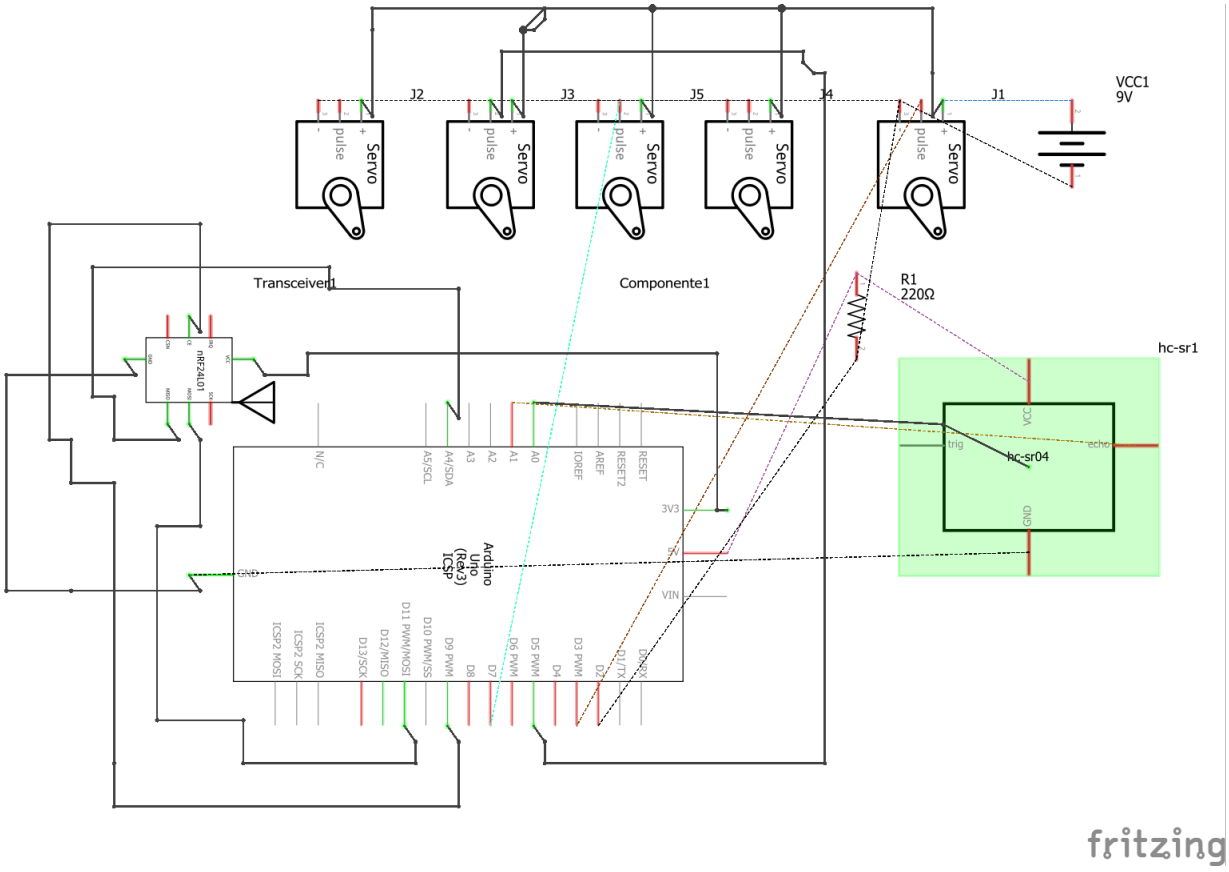

Figure 2: Device A circuit diagram

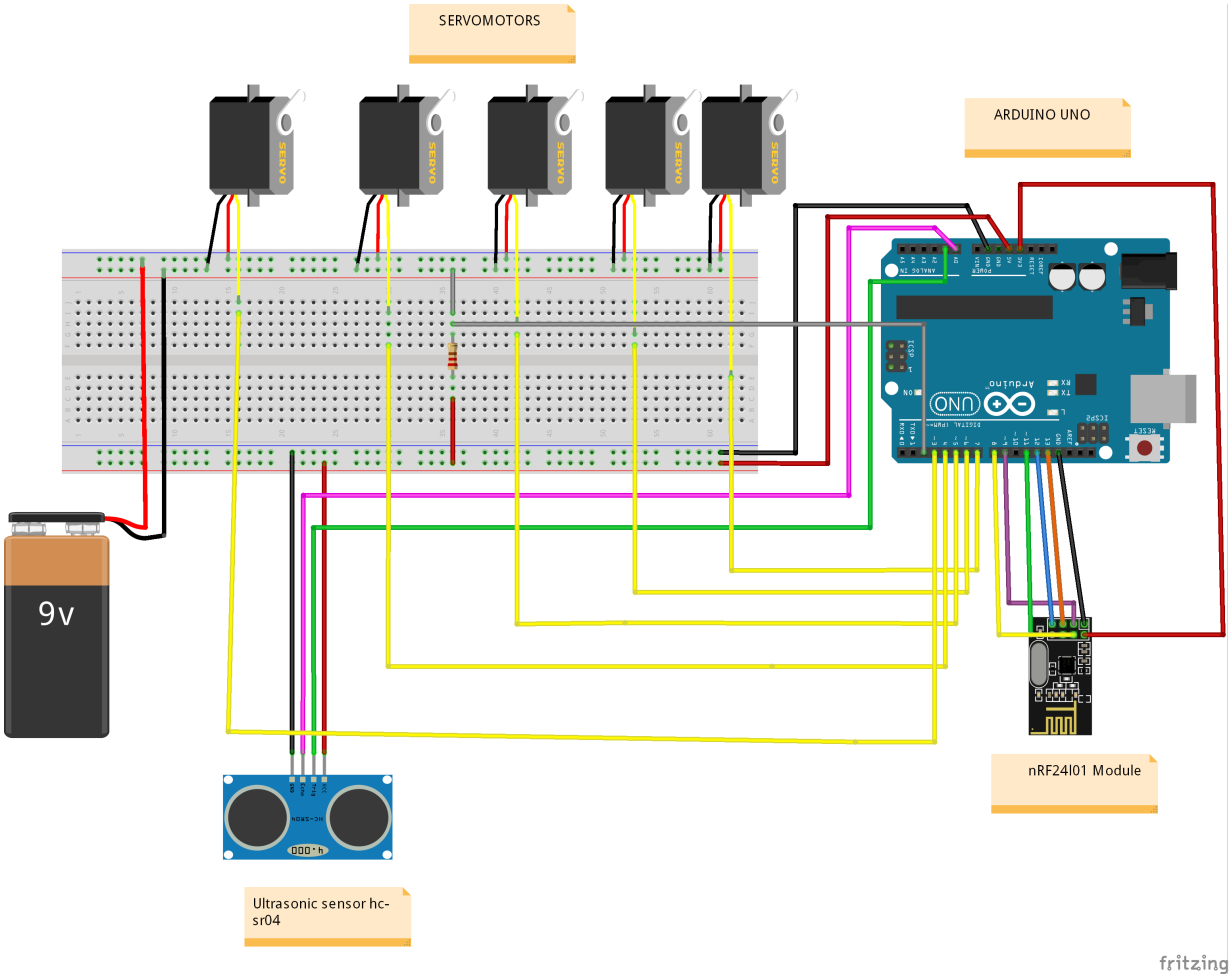

Figure 3: Device B wiring diagram

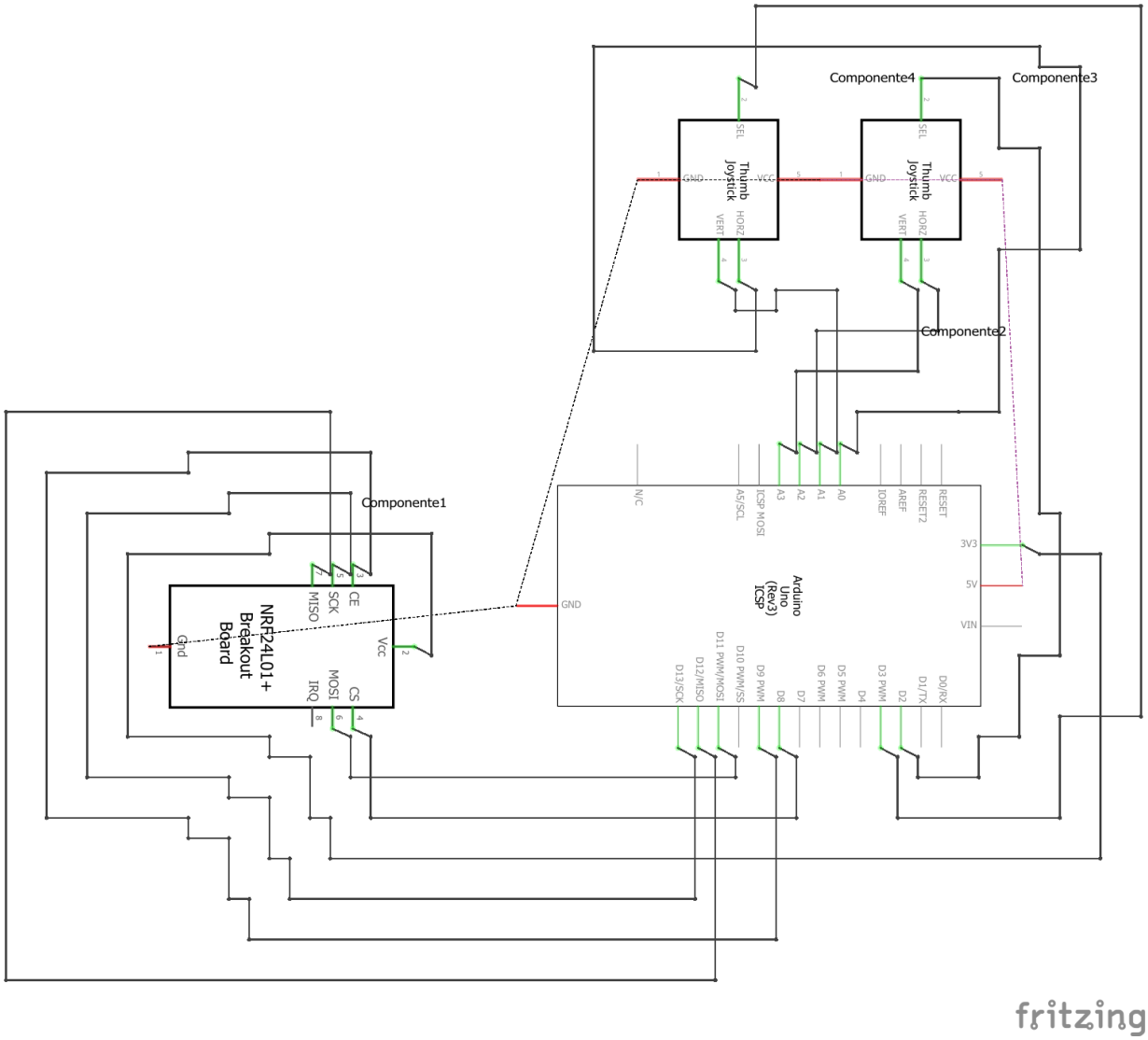

Figure 4: Device B circuit diagram